Optimizing Real-World Problems with Differential Evolution and Particle Swarm Optimization

Dem Fachbereich Physik, Elektrotechnik und Informationstechnik
der Universität Bremen

zur Erlangung des akademischen Grades einer
DOKTOR-INGENIEURIN (Dr.-Ing.)
vorgelegte Dissertation

von
Dipl.-Ing. Karin Zielinski
aus Bremen

Referent:	Professor Dr.-Ing. R. Laur
Korreferent:	Professor Dr. phil. nat. D. Silber
Eingereicht am:	11.12.2008
Tag des Promotionskolloquiums:	12.02.2009

Acknowledgements

This work was produced while I was working as a research assistant at the Institute for Electromagnetic Theory and Microelectronics (*Institut für Theoretische Elektrotechnik und Mikroelektronik*, ITEM) at the University of Bremen. Many people helped and supported me during this time, and here I want to take the opportunity to express my gratitude.

First, I thank my thesis committee. Special thanks to Prof. Dr.-Ing. Rainer Laur for giving me a lot of freedom for my research. Although he sometimes joked about me going to so many conferences and spending his money, he always encouraged me to continue. Also many thanks to Prof. Dr. phil. nat. Dieter Silber for being interested in my work and for having so much fun discussing it. Furthermore, I thank Prof. Dr.-Ing. Karl-Dirk Kammeyer and Prof. Dr.-Ing. Rolf Drechsler for being my examiners.

I thank my colleagues for good discussions but also for having fun together. Special thanks to Dr.-Ing. Detmar Westphal and Dipl.-Ing. Ole Bischoff for reading my thesis and providing valuable comments. Also many thanks to the people who supplied me with real-world optimization problems, especially to Dipl.-Ing. Petra Weitkemper who has endured all the seemingly endless revisions of our mutual journal paper. Two students also contributed to this thesis, so thanks to Dipl.-Ing. Xinwei Wang and to Benjamin Begandt for working with me.

Last but not least I want to thank my family, my boyfriend Jakob and my friends for supporting and encouraging me.

Bremen, February 2009

Karin Zielinski

Contents

List of Figures

List of Tables

List of Acronyms and Symbols

ϵ_g	Accuracy for the distance to the global optimum in objective space
ϵ_e	Accuracy for the violation of equality constraints
ρ	Ratio of feasible space to search space
c_1	Control parameter of Particle Swarm Optimization, influences the cognitive component of the velocity update equation
c_2	Control parameter of Particle Swarm Optimization, influences the social component of the velocity update equation
CR	Control parameter of Differential Evolution, influences recombination
D	Dimension, i.e. number of variables of an optimization problem
DE	Differential Evolution
EA	Evolutionary algorithm
EC	Evolutionary computation
EP	Evolutionary Programming
ES	Evolution Strategy
$\vec{f}(\vec{x})$	Vector of objective functions
F	Control parameter of Differential Evolution, influences mutation
FE	Number of function evaluations
FE_{max}	Maximum number of function evaluations
FE_{conv}	Number of function evaluations for reaching convergence
$\vec{g}(\vec{x})$	Vector of inequality constraints
GA	Genetic Algorithm
GP	Genetic Programming
$\vec{h}(\vec{x})$	Vector of equality constraints
J	Number of inequality constraints
K	Number of equality constraints
M	Number of objective functions
MOEA	Multi-objective evolutionary algorithm
NP	Number of individuals, control parameter of e.g. Differential Evolution and Particle Swarm Optimization
NSGA-II	Non-dominated Sorting Genetic Algorithm II
$\vec{p}_i$	Personal best position, used in Particle Swarm Optimization
$\vec{p}_g$	Neighborhood best position, used in Particle Swarm Optimization
PSO	Particle Swarm Optimization
SI	Swarm intelligence
$\vec{u}_i$	Trial vector, used in Differential Evolution
$\vec{v}_i$	Mutated vector in Differential Evolution, velocity in Particle Swarm Optimization

$v(\vec{x})$	Sum of constraint violation
$\vec{v}_{max}$	Maximum velocity for Particle Swarm Optimization
w	Inertia weight, control parameter of Particle Swarm Optimization
$\vec{x}_i$	Parameter vector, also called target vector in Differential Evolution
$\vec{x}_{max}$	Upper limit of the search space
$\vec{x}_{min}$	Lower limit of the search space
X	Parameter space, search space
Y	Objective space

Chapter 1

Introduction

Considering today's competitive markets, the time for the development of technical systems should be as short as possible. Simultaneously, systems become very complex due to manufacturing at the limits of technical feasibility. Furthermore, a high functionality is desired with low power consumption, small size and high reliability. However, the development of a system, e.g. in circuit design, is normally an iterative trial-and-error process. After the basic structure of the design has been established, parameters are repeatedly adjusted by the designer and the system is simulated to verify its performance. Due to the complex interactions of parameters, this is a time-consuming process. The success depends on the experience and knowledge of the designer. From this follows that the goal of a short time-to-market becomes hard to achieve. Therefore, the need arises for fast, easy-to-use optimization algorithms which can be integrated in design processes. Thus, the time-consuming iterative trial-and-error process can be substituted by a structured automated approach for finding optimal parameter settings.

However, a vast variety of optimization algorithms exists, making it difficult to choose a suitable technique. If no (or only little) information is available about the properties of an optimization problem, the application of optimization algorithms from the class of evolutionary algorithms (EAs) and swarm intelligence (SI) has many advantages. They are generally applicable methods which have almost no requirements concerning the optimization problem, e.g. no derivatives of the objective functions are needed. Thus, they can also be applied if the objective functions are not given in analytical form, even if the objective functions have to be simulated for every set of variables separately. The algorithms are inspired by natural phenomena like evolution and swarm behavior which makes them robust search methods for many problems because of the adaptation processes that also characterize nature. In contrast to many other optimization algorithms, EAs as well as algorithms from SI have the capability of finding the global optimum even of highly multimodal functions because of their population-based structure and the involved randomness. The general applicability of these algorithms also means that they can be employed for problems from virtually any field of engineering or science. In this work the focus concerning applications is on problems from electrical engineering.

Developers of optimization algorithms are mostly mathematicians or computer scientists whereas users of optimization algorithms are often engineers or natural scientists because many optimization problems arise in these disciplines. From this follows that the users are often experts from their corresponding fields but not experienced concerning optimization (see also [Lob00] where it is argued that the users actually should not be required to be

optimization experts because toasters can also be used without knowing Ohm's Law). As a result, an important property of optimization algorithms should be ease of use. Because one of the most significant sources of difficulties is the adjustment of control parameters of the optimization algorithms, methods with a low number of control parameters should be preferred. Furthermore, several algorithms require a coding of parameters (usually binary, see Section 3.1). The need for coding also creates difficulties for users because a suitable coding scheme has to be found. This necessity contradicts the demand of easy applicability and should therefore be avoided.
Differential Evolution (DE) and Particle Swarm Optimization (PSO) are two relatively new optimization algorithms which are well suited for many technical problems. One advantage is their ease of use due to the presence of only few control parameters. Another benefit is the real-valued representation that omits the need for coding of parameters. Differential Evolution can be seen as a typical evolutionary algorithm because it closely follows the "survival of the fittest" principle. In contrast to other EAs, DE relies on a simple principle that nevertheless causes automatic adaptation of the step sizes during an optimization run. This property leads to a high convergence speed and DE's successful use in many applications [Pri05].
Because Particle Swarm Optimization is derived from swarm intelligence, it has a different background. Nevertheless, PSO has several similarities with evolutionary algorithms like DE, thus it is also often regarded as an EA. In PSO the behavior of social groups is simulated, hence optimization is achieved by cooperation of individuals and sharing of information. Due to the incorporation of past experiences into the search, the step size is also adapted during an optimization run. Many practical optimization problems have already been successfully solved using PSO [Eng06].

1.1 Motivation and Contributions

Real-world problems are optimization problems which arise in an application, e.g. in engineering or science, instead of being artificially created (see also Section 2.3). Real-world problems are often difficult to optimize because they have characteristics that complicate the application of optimization algorithms. In this work it is discussed which features are associated with real-world problems and how optimization algorithms can be adjusted to handle them.
It is not possible to regard every existing optimization algorithm due to the large number of algorithms. Therefore, the examination must be limited to a small number of promising algorithms which can be robustly used in many applications. Due to the advantages discussed above, Differential Evolution and Particle Swarm Optimization are used in this work. For both algorithms adaptations will be shown which enable them to cope with the difficulties associated with real-world problems. Meanwhile explicit focus on easy applicability is maintained due to the already mentioned discrepancy in knowledge between developers and users of optimization algorithms. Although there are certain similarities, there are also fundamental differences between these algorithms which arise from the different background. Thus, they are compared here regarding their ability to solve real-world problems.
For the application of DE and PSO to real-world problems, a software tool has been developed that was used for conducting the experiments described in this work. The

software tool has been programmed in C++ because the structure of both Differential Evolution and Particle Swarm Optimization suggests using object-oriented programming. For some special applications the software tool has been coupled with Spectre (circuit design, see Section 5.5) and Matlab (optimization of a PI cascade control [Zie08b, Zie08a], not regarded further here). The software tool has been extended based on the demands of the respective problems, with emphasis on easily applicable methods.

One of the most commonly regarded problematic features of real-world optimization problems is the presence of constraints which occur e.g. due to physical limitations. Thus, an efficient method for handling constraints must be found for DE and PSO. Several methods have already been presented in the literature but often they introduce new parameters. Preferably these techniques should also be easy to use, e.g. they should include as few control parameters as possible. This applies to the death penalty and the modified replacement method which do not need any control parameters to be set, so they are especially easy to apply. Furthermore, no comprehensive comparisons are available for DE and PSO with focus on easily applicable methods. For other optimization algorithms some examinations can be found in the literature but generally results are not transferable. Therefore, the death penalty and the modified replacement method are compared for DE and PSO in this work. As basis for the examination a power allocation problem is used that occurs in a communication system. Because of the high demands concerning the accuracy of fulfilling an equality constraint that arise in yield analysis, it is demonstrated how a further adjustment for the handling of equality constraints improves the results of DE and PSO.

Another aspect associated with real-world problems is that often multiple objectives should be optimized simultaneously. The extension of evolutionary algorithms for multi-objective optimization is not a trivial task because the goals in multi-objective optimization differ significantly from the ones in single-objective optimization. Decisions about many implementation details have to be made and their influence is often still not clear. Especially in the context of DE and PSO, multi-objective optimization still is a rather young field that requires more research for finding effective and efficient algorithms. In this work an overview about the issues connected with extending DE and PSO for multi-objective optimization is given. As an exemplary test case, the optimization of several characteristics of an operational amplifier is shown. Multi-objective variants of both DE and PSO are used, and it is shown that they can be successfully applied for this task.

Many real-world problems contain computationally expensive constraint and objective functions, e.g. because the performance of a system must be evaluated via simulations. This property leads to the need of reliable stopping criteria which are able to detect when convergence has been reached. This is especially important if the algorithms should be applied in automatic design processes, meaning an interactive combination of tools for developing systems and optimizing them, e.g. for circuit design. Stopping criteria are seldom examined in the literature although for practical applications the choice of stopping criteria can significantly influence the duration of an optimization run. In the literature the discussion is mostly limited to Genetic Algorithms and it is not clear if the results are also valid for DE or PSO. Furthermore, a systematic classification considering a large variety of stopping criteria is missing.

Therefore, in this work a detailed presentation and classification of stopping criteria is shown. Stopping criteria from literature are regarded but also new approaches for terminating optimization runs are introduced. It is shown that the same classification can

be used for both single-objective optimization and multi-objective optimization although the underlying mechanisms are different. Furthermore, the performance of the stopping criteria is analyzed for DE and PSO. For this examination a real-world problem has been used but also several test functions. The reason is that an extensive evaluation of many stopping criteria was desired. Due to the computational cost of real-world problems, usually they cannot be used for such a study. Instead, a large number of test functions with varying characteristics has been employed. These test functions contain many features which can also be found in real-world problems, thus they are considered to be an adequate substitute. This consideration holds also for some other parts of this work where extensive studies have been done, e.g. the examination of adaptive approaches described in the last paragraph of this section.
The choice of control parameter settings considerably influences the results of optimization runs, regarding both the convergence probability and the convergence speed. Although the number of control parameters is rather low in DE and PSO and some recommendations regarding standard settings can be found in the literature, still some experimentation is usually necessary to find suitable settings. The pursuit of easy applicability leads to the examination of methods which set parameters adaptively, thus the user is relieved from searching for appropriate settings. Several approaches have already been presented in the literature but they usually neglect interactions of parameters. In this work a new method is shown that does not only regard the performance of different control parameters separately but that also considers interaction effects.
A problem in the optimization literature is that new algorithms are developed constantly but there are not enough comparisons in the literature to assess which algorithms are actually superior. Particularly, a disadvantage in comparisons from the literature is that several components are changed at a time. Thus, it cannot be distinguished between features that are useful and other components that only complicate the algorithm without contributing to its performance. This especially holds for adaptive approaches because they usually contain multiple individual components. Therefore, a detailed examination of adaptive algorithms and their individual components is shown for DE that leads to a deeper understanding of their effects.

1.2 Organization

In Chapter 2 the basic principles of optimization are presented. First, the general optimization problem is defined mathematically. A classification of optimization algorithms is given, and the motivation for using evolutionary algorithms is explained. Furthermore, for readers who are inexperienced concerning optimization, a list of terminology and important concepts is given that should simplify the reading and understanding of the following chapters.
The most commonly used representatives of evolutionary algorithms are presented in Chapter 3: Genetic Algorithms, Evolution Strategies, Evolutionary Programming and Genetic Programming. As the focus of this work is on Differential Evolution and Particle Swarm Optimization, the introduction of DE and PSO is the largest part of this chapter. Both algorithms are introduced in their basic form for unconstrained single-objective optimization whereas extended versions for constrained and multi-objective optimization will be covered later.

After the basics for this work have been explained in Chapters 2 and 3, the own contributions are given in the following chapters. In Section 1.1 several aspects have been identified which must be considered when optimizing real-world problems: Constraints, multiple objectives, stopping criteria and adaptive parameter setting. These topics all have their own unique problems and can be regarded independently from each other. Thus, they are discussed in separate chapters in this work. The related literature is also given separately for each part. Each of the following chapters contains a literature review that is supplemented by own contributions where necessary for the optimization of real-world problems.

Constrained optimization is discussed in Chapter 4. It is distinguished between boundary constraints and constraint functions, and the constraint functions are further divided into inequality and equality constraints. Several approaches from literature for dealing with constraints using DE and PSO are given where the focus is on easily applicable methods. Two of these methods are tested for an application example that consists of optimizing a power allocation problem for a CDMA system with interference cancellation. Specific methods for handling equality constraints from literature are also given, and the optimization of a problem from yield analysis that requires an equality constraint to be fulfilled very precisely is shown using DE and PSO. Because evolutionary algorithms are stochastic methods for which it is difficult to obtain a convergence proof, the superiority of DE and PSO in contrast to two simple random methods is shown on the basis of an extensive set of constrained single-objective test problems.

Multi-objective optimization is introduced in Chapter 5. Evaluating solutions is considerably more difficult in multi-objective optimization in contrast to single-objective optimization, thus performance measures for the different goals in multi-objective optimization are given. These will also be used for the definition of stopping criteria for multi-objective optimization in Chapter 6. Additionally, the NSGA-II (Non-dominated Sorting Genetic Algorithm II) is described because the adaptation of concepts from NSGA-II for multi-objective DE is discussed afterwards. Literature surveys are given for both multi-objective DE and multi-objective PSO. Furthermore, an application example is presented. The model of an operational amplifier is introduced that can be simulated to verify its performance. It is shown how DE and PSO can be used for the optimization of its characteristics.

Stopping criteria for single-objective as well as multi-objective optimization are presented in Chapter 6. Stopping criteria are sorted into classes based on the feature that they operate on, and an evaluation of their performance is given.

The adaptive setting of control parameters is discussed in Chapter 7. Besides a literature survey, a new approach considering interactions of control parameters is presented. Several individual components of adaptive algorithms are examined and compared separately, and it is indicated which methods result in superior performance for DE.

In Chapter 8 a summary of the most important results of this work is given. Additionally, it is stated which further extensions may be desirable for future work.

In Appendix A some test problems for multi-objective optimization used in Section 5.3.3 as well as Section 6.2 are specified. Results of these examinations are shown in Appendix B and C, respectively.

Chapter 2

Basic Principles of Optimization

In this chapter the general optimization problem is defined. A short overview about different kinds of optimization algorithms is given, and the choice of evolutionary algorithms for this work is explained. To simplify the reading of the following chapters, some general terms and concepts are given for optimization in general as well as for evolutionary algorithms in particular.

2.1 The General Optimization Problem

In this work an optimization problem is defined as the minimization[1] of one (single-objective) or several (multi-objective) objective functions:

$$\text{Minimize } \vec{f}(\vec{x}) \tag{2.1}$$

where

$$\vec{x} = (x_1, x_2, \ldots, x_D) \in X \subset \mathbb{R}^D \tag{2.2}$$

is the **parameter vector**, X is called **parameter space** (or **search space**) and D is the **dimension** of the problem.

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_M(\vec{x})) : X \to Y \subset \mathbb{R}^M \tag{2.3}$$

is the **objective function vector**, Y is called **objective space** and M is the number of objective functions. Other names used for objective functions in the literature are cost functions (which are usually used for minimization problems) or fitness functions (which are generally used for maximization problems). However, these terms are not clearly defined, e.g. in [Run00] the fitness function is defined as the sum of the objective function and penalty terms depending on the constraints. Several sets of parameters will be introduced in this work, thus in the following it must be clearly distinguished between the parameters $\vec{x}$ which are the variables of the objective function, the control parameters of the algorithms and also the parameters of the stopping criteria which will be introduced in Chapter 6.
A general optimization problem also contains constraints. In this chapter constraints are only mentioned shortly where needed to define the general optimization problem. Further explanations will be provided in Chapter 4. Constraints are usually divided into

[1] Maximization is included in this formulation as $\max(f(\vec{x})) = -\min(-f(\vec{x}))$.

boundary constraints as well as constraint functions. **Boundary constraints** (also called parametric constraints [Run00]) are of the form

$$x_{min,d} \leq x_d \leq x_{max,d} \tag{2.4}$$

where $x_{min,d}$ and $x_{max,d}$ are the lower and upper boundary for parameter d. Thus, the boundary constraints define the search space X. Boundary constraints like given in Equation 2.4 are also sometimes called box constraints [Eng06] because this formulation leads to a search space in the form of a hypercube. Theoretically, boundaries might also be given in other formulations than a hypercube. Boundaries may be somehow curved, e.g. resulting in a spherical search space. However, in the literature generally only search spaces in the form of hypercubes can be found.
If an optimization problem contains only boundary constraints without additional constraint functions, it is referred to as an unconstrained problem [Eng06]. This expression may be confusing due to the term "boundary *constraints*". The reason is that boundary constraints are present in nearly every optimization problem, and they are relatively easy to fulfill in contrast to constraint functions. Nevertheless, in this work boundary constraints are discussed in Chapter 4 because of the formal similarity with constraint functions. Theoretically, boundary constraints might even be handled like any other constraints although this might not make sense, see Section 4.1.
In contrast to the boundary constraints which define the search space, the **constraint functions** define the feasible space (see Figure 2.1 where the gray area indicates the search space and the shaded area indicates the feasible space). Constraint functions are usually divided into inequality constraints

$$g_j(\vec{x}) \leq 0 \qquad j = 1, 2, \ldots, J \tag{2.5}$$

and equality constraints

$$h_k(\vec{x}) = 0 \qquad k = 1, 2, \ldots, K \tag{2.6}$$

where J is the number of inequality constraints and K is the number of equality constraints. Some authors write Equation 2.5 as $g_j(\vec{x}) \geq 0$ (e.g. [Deb01a, Gol89]) while others prefer it like given here (e.g. [Pri05, Eng06]; actually this seems to be the most common form in the literature) but these formulations are equivalent. Only when actually implementing constraints and constraint-handling techniques in a computer program, attention has to be paid that the same formulation is used consistently.

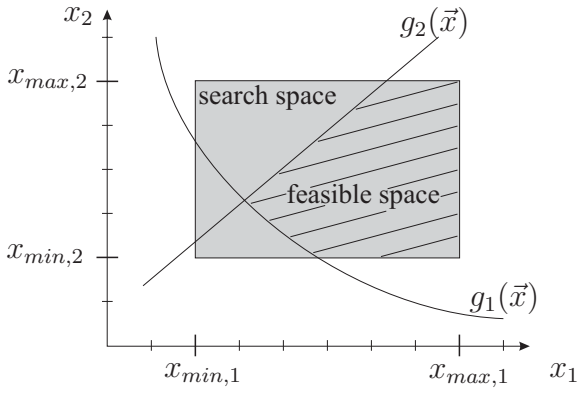


Figure 2.1: Search space and feasible space

Because of numerical problems each equality constraint is usually transformed into an inequality constraint of the form

$$|h_k(\vec{x})| - \epsilon_e \leq 0 \qquad k = 1, 2, \ldots, K \tag{2.7}$$

where ϵ_e defines the accuracy for fulfilling the equality constraint. If a solution $\vec{x} \in X$ satisfies all constraints it is called feasible. Thus, **feasible space** is defined as

$$\mathcal{F} = \{\vec{x} \in X | g_j(\vec{x}) \leq 0 \ \forall j \text{ and } |h_k(\vec{x})| - \epsilon_e \leq 0 \ \forall k\}. \tag{2.8}$$

If the equation $g_j(\vec{x}) = 0$ holds for an inequality constraint, it is said that constraint g_j is active at $\vec{x}$. From this follows that equality constraints are always active at all points of the feasible space $\mathcal{F}$.
In single-objective optimization the **global optimal solution** $\vec{x}^*$ is defined by

$$\forall \vec{x} \in X : f(\vec{x}) \geq f(\vec{x}^*), \tag{2.9}$$

meaning that there does not exist another solution with a smaller objective function value in the search space. For a **local optimal solution** $\vec{x}'$ this condition holds only in a certain vicinity of the solution (see also Figure 2.3):

$$\exists \epsilon > 0 : \forall \vec{x} \in X : \rho(\vec{x}, \vec{x}') < \epsilon \Rightarrow f(\vec{x}) \geq f(\vec{x}') \tag{2.10}$$

where ρ is a distance measure in parameter space X [Bäc97]. This common definition of a local optimum (see also [Bro00]) does not prohibit that ϵ may be chosen in a way that the vicinity of $\vec{x}'$ comprises the whole search space. Thus, the local optimum would become a global optimum. However, in this work this case is excluded. Whenever mentioning a local optimum, it is assumed that still a better solution exists in the search space.
If not one but several objective functions should be minimized, the definition of an optimal solution is less obvious. Generally, the **dominance** relation is used to be able to compare two solutions in multi-objective optimization. A solution $\vec{a} \in X$ dominates another solution $\vec{b} \in X$ ($\vec{a} \prec \vec{b}$) if no objective function value of $\vec{a}$ is worse than the corresponding objective function value of $\vec{b}$, and $\vec{a}$ is better than $\vec{b}$ in at least one objective:

$$\forall m \in \{1, \ldots, M\} : f_m(\vec{a}) \leq f_m(\vec{b}) \tag{2.11}$$

$$\exists m \in \{1, \ldots, M\} : f_m(\vec{a}) < f_m(\vec{b}). \tag{2.12}$$

A solution $\vec{a}' \in X$ is called **non-dominated** regarding a set $X' \subseteq X$ if no other solution exists that dominates $\vec{a}'$:

$$\nexists \vec{a} \in X' : \vec{a} \prec \vec{a}'. \tag{2.13}$$

If $\vec{a}'$ is non-dominated regarding the whole search space X, it is called **Pareto-optimal**. Because in multi-objective optimization usually conflicting goals are optimized (see Chapter 5), generally several or even an infinite number of mutually non-dominated solutions exist. These Pareto-optimal solutions correspond to the global optimum in single-objective optimization. They are trade-off solutions from which the so-called decision maker (usually a human being) selects the final solution to be used. The set of Pareto-optimal solutions in objective space is referred to as the **Pareto-optimal front** whereas the corresponding set of vectors in parameter space is called the **Pareto-optimal set**.

Because multi-objective optimization algorithms do not necessarily find the (complete) Pareto-optimal front, the results of an optimization algorithm will be referred to as the **approximation set** (as in [Kno06], the term approximation set is also used synonymously for the generated front here) or simply as non-dominated solutions.
In Figure 2.2 the dominance relation is illustrated: $\vec{b}$ dominates $\vec{c}$ and $\vec{d}$ because $\vec{b}$ is better than $\vec{c}$ regarding both objectives, and $f_2(\vec{b}) = f_2(\vec{d})$ but $f_1(\vec{b}) < f_1(\vec{d})$. $\vec{a}$ and $\vec{b}$ are mutually non-dominated because $\vec{a}$ is better concerning f_1 but $\vec{b}$ is better concerning f_2. If there are no other solutions in the search space which dominate them, they correspond to the Pareto-optimal solutions. Examples of Pareto-optimal fronts with an infinite number of solutions are shown in Appendix A.

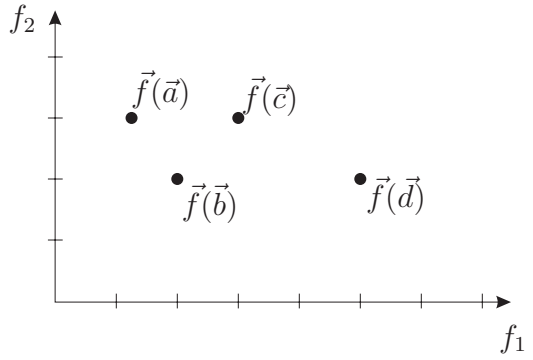


Figure 2.2: Illustration of the dominance relation

2.2 Optimization Algorithms

A large variety of optimization algorithms exists which have been developed for different kinds of problems. Thus, there are many ways for classifying optimization algorithms based on the problems that they were designed for but also based on characteristics of the algorithms themselves. In the following the main characteristics are given to define what kinds of problems and algorithms will be examined in this work (other possibilities for classifying optimization algorithms can be found in [Men04a, Pet01]). The first two properties are related to the optimization problems whereas the latter two attributes refer to the optimization methods.

- Static $\leftrightarrow$ dynamic: Dynamic optimization means that an optimum should be found while environmental conditions are varying, i.e. the optimum might be moving [Sch95]. An example is constructing an optimal traffic lights schedule [Rak08] but also the optimization of vehicle routing and scheduling problems [Sch07]. In contrast, the here regarded problems are static, meaning that the optimum is non-changing. Therefore, dynamic optimization is not regarded here further.

- Continuous $\leftrightarrow$ discrete: Depending on the nature of the parameters, discrete and continuous problems are distinguished. An interesting discussion about the differences between discrete and continuous optimization can be found in [Cle06] but here only the major differences are given to define the scope of this work. Discrete problems may be combinatorial problems like the travelling salesman problem or

vehicle routing and scheduling problems [Sch07] where a large but finite set of parameter combinations exists. Optimizing the performance of a circuit that is built from discrete electronic components may also be seen as a discrete problem if only certain sizes are available and it should be avoided to connect a large number of components in series or in parallel to reach the desired value. Additionally, problems with integer or binary representation of the parameters are subsets of discrete problems [Lam04]. In contrast, all optimization problems examined in this work have continuous variables. Of course the use of computers for conducting the optimization process actually leads to a limited precision but in practice the restrictions are negligible.

- Deterministic ↔ stochastic (local ↔ global): Local optimization algorithms use deterministic methods to find the optimum of a function. Because of their deterministic nature they are generally not able to escape from local minima because they try to move in the direction of descending values of the objective function. Hence, they are dependent on the starting position and have problems with multimodal functions [Jak04] (see Figure 2.3). In contrast, global optimization algorithms use stochastic operators. Thus, they are generally able to find the global optimum of a function regardless of the number of local minima. Due to this advantage, global optimization algorithms are considered in this work.

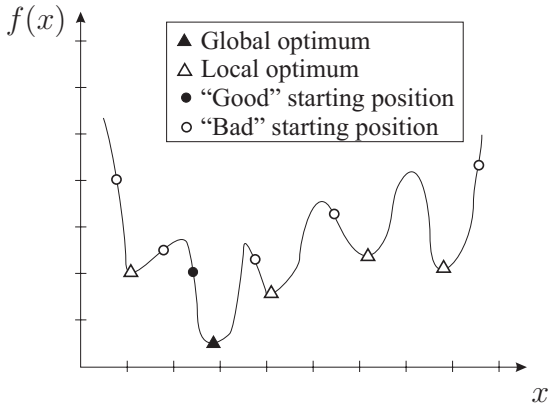


Figure 2.3: Dependence of deterministic algorithms on starting positions

- Population-based ↔ one individual per generation: In this work methods are used that regard more than one individual at the same time (generation). An advantage of population-based methods over techniques with one individual per generation is that information can be shared among the individuals, leading to a better coverage of the search space.

To sum up, in this work population-based stochastic global optimization algorithms are regarded which operate on continuous variables, and the objective function does not change with time.
Evolutionary algorithms are the most widely used global optimization algorithms. They are general methods which have almost no requirements concerning the optimization problem, e.g. they do not need derivatives of the objective function. They can be applied to all problems which can be expressed as given in Section 2.1. Thus, it is sufficient that the

values of constraint and objective functions can be computed for individual parameter sets but the functions do not have to be given in analytical form. The only assumption is that there is a certain coherence in the objective function, meaning that objective function values of parameters which are close to each other are similar. If an objective function is flat and only contains one peak somewhere, evolutionary algorithms do not have an advantage over a pure random search [Jak04]. However, real-world optimization problems generally exhibit some correlation between nearby parameter values, so evolutionary algorithms are preferable in contrast to random search. In Section 4.4 this property is empirically demonstrated based on an extensive set of test functions which are assumed to include several features that also occur in real-world problems. The comparison of blind random search and brute force search with Differential Evolution and Particle Swarm Optimization indeed shows the superiority of evolutionary algorithms over pure random search methods.
In recent years optimization algorithms have been developed which have similar functionality as evolutionary algorithms but do not necessarily apply the principles of evolution (mutation, recombination, selection). Instead, other biological mechanisms are used as inspiration. Some of these new fields are swarm intelligence (with its main representatives Particle Swarm Optimization and Ant Colony Optimization) and artificial immune systems. For these algorithms generally the same advantages hold as discussed above for evolutionary algorithms.
A general term which includes all these methods would be bio-inspired optimization algorithms. However, due to the advantages already mentioned in Chapter 1, the emphasis lies on Differential Evolution and Particle Swarm Optimization in this work. Both methods may be classified as EAs, thus the term "evolutionary algorithms" is kept here.

2.3 Terminology and Concepts

As evolutionary algorithms are inspired from nature, the terminology is also mostly derived from biology, especially genetics. For readers who are unfamiliar with EAs (and optimization in general), an overview about important terms and concepts is given in the following. First, some basics are explained which are absolutely necessary for understanding an evolutionary algorithm. Afterwards, more specific terms are given which will also be used in this work.

- Individual: Individuals may be seen as points in the search space. Besides the position in the search space, they are also associated with information about the violated constraints and the objective function value. Each of them represents a solution of the optimization problem but not necessarily a good one. The term *solution* is also used as a synonym for individual in the following.
- Population: A group of *individuals* which usually interact with each other.
- Parent: An *individual* that participates in the creation of a new individual which is called its *offspring*.
- Child/offspring: Newly generated *individual* that is compared to its *parent* or to some members of the parent *population* (depending on the algorithm) to determine if it receives a place in the successive *generation*.

- Generation: Discrete time unit. In each generation the *evolutionary operators* are applied either to all *individuals* of the current *population* or to a selected subset, depending on the algorithm. The term "iteration" is often used synonymously, especially in Particle Swarm Optimization where the individuals are not replaced by their *offspring* but rather move in the search space.

- Evolutionary operators: The operators which are used in evolutionary algorithms to produce new individuals. The operators are named like their counterparts in natural evolution: *Crossover/recombination*, *mutation* and *selection*.

- Crossover/recombination: *Evolutionary operator*. Several *individuals* (*parents*) interact to produce an *offspring*. This is in contrast to the evolutionary operator *mutation* that only operates on one individual. Crossover is a specific form of recombination in biology but in the literature of evolutionary algorithms these terms are usually used synonymously.

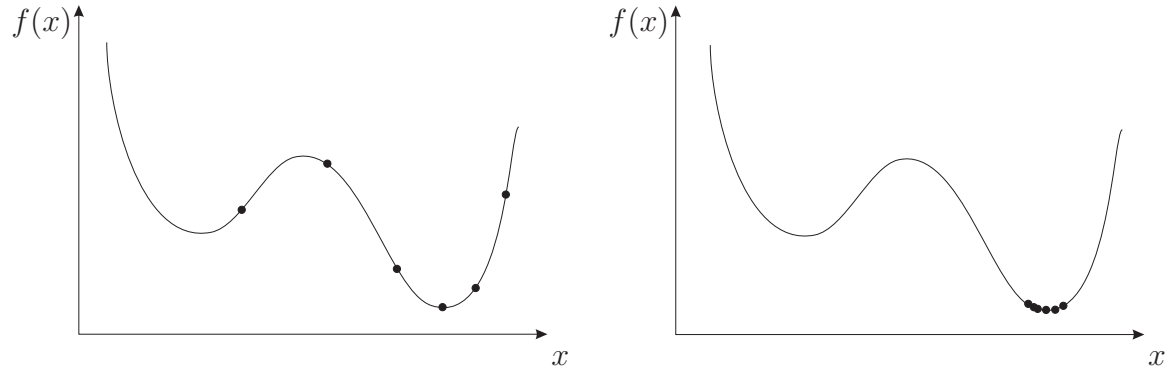
- Mutation: *Evolutionary operator*. Mutation is generally applied after *recombination* to further perturb the *offspring*. An exception is the DE algorithm where first mutation and afterwards recombination is applied. In contrast to recombination that exchanges components from already existing individuals, mutation is generally used to bring new characteristics into the population. In the case of binary-coded individuals, this may be the change of a bit from 0 to 1 or vice versa. For real-coded individuals mutation may be implemented as a random change in a vector component.

- Selection: *Evolutionary operator*. In EAs usually it has to be distinguished between parent selection (selecting the solutions that participate in creating a new individual) and survivor selection (selecting the individuals which are part of the next generation). In this work usually survivor selection is meant when mentioning the term selection. It is indicated where the term is used with different meaning.

In the following several terms and concepts are clarified which are less basic. Most of them do not occur throughout this work but only in limited areas. To simplify the search for specific terms, they are given in alphabetical order.

- Adaptation: See (Self-)Adaptation.

- Convergence: Approaching of a stationary state is referred to as convergence. Usually, this means that the diversity of the population decreases, so that the population is located in a small area of the search space. Convergence to the global minimum is desired but also convergence to local minima may happen. The latter case is often termed premature convergence although premature convergence might also mean a loss of diversity without reaching any kind of optimum. In the literature it is often not checked if a population has actually converged but rather if the optimum was found with a specified accuracy ϵ_g. Using the given definition, convergence of the population has not necessarily been reached (see Figure 2.4). For the comparison of algorithms this approach is sufficient but it will be shown in Chapter 6 that convergence must be regarded in its original meaning for the definition of stopping criteria.



(a) The optimum has been found but the population is not converged.

(b) The population is nearly converged.

Figure 2.4: Convergence

- Convex optimization problem: An optimization problem is convex if all objective functions as well as the feasible region are convex [Deb01a]. A function is convex if the following holds for any two solutions $x_1 \neq x_2$ and $t \in [0, 1]$:

$$f(t \cdot x_1 + (1 - t) \cdot x_2) \leq t \cdot f(x_1) + (1 - t) \cdot f(x_2). \tag{2.14}$$

  This means that the value of the function f between the points x_1 and x_2 is always below the connecting line of $f(x_1)$ and $f(x_2)$ (see Figure 2.5). It follows that a local optimum is also a global optimum.

- Deceptive: An optimization problem is called deceptive if features of the problem tend to lead optimization algorithms away from its global optimum, e.g. by a gradient in the direction of a local optimum. A deceptive problem is usually difficult to optimize.

- Diversity: Expresses how different individuals are, i.e. gives the degree of dispersion. Diversity usually refers to differences in parameter space. For example, in Figure 2.4(a) a high diversity is present in the population whereas in Figure 2.4(b) the diversity is low. Some ideas how to measure diversity are given in [Olo08, Shi08].

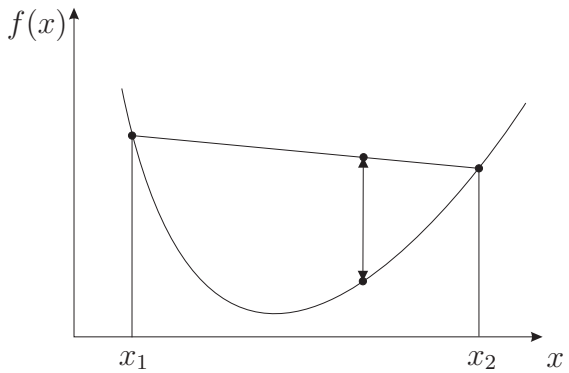


Figure 2.5: Convex function

- Elitist/elitism: Elitist methods ensure that the previously found best population member(s) survive to the next generation without being changed. Elitism can be implemented in different ways, see e.g. Sections 3.5 and 5.3.1. Elitist strategies generally result in faster convergence but the risk of converging to a local optimum increases.

- Exploration vs. exploitation: Exploration means visiting previously unexplored regions of the search space. In contrast, exploitation means using already gathered information about good points in the search space in order to find other good or preferably even better solutions. In an optimization run, early stages are typically characterized by exploration whereas in later stages the task changes to exploitation to fine-tune the solution.

- Feasible: A solution is called feasible if it satisfies all constraints.

- Flat region: A part of the search space where the objective function value does not change. Flat regions make an optimization problem more difficult because algorithms may have problems crossing it (see e.g. Section 3.5.1).

- Heuristic: In contrast to an algorithm which gives exact solutions to a problem and which can be proven to succeed, a heuristic is able to yield approximate solutions but usually no convergence proof exists. In other words, there is a risk of obtaining a suboptimal solution. As for most evolutionary algorithms there is no convergence proof, they can be classified as heuristics.

- Landscape: This term refers to properties, i.e. the shape, of the objective function because of the resemblance with hills and valleys.

- No free lunch theorem: This theorem states that the performance of all optimization algorithms is equivalent when averaged over all possible functions [Wol97]. At first glance it may seem that the statement of this theorem is that all effort in designing optimization algorithms is futile because a pure random search will have the same performance on average. In fact it has to be distinguished between different classes of problems. For random functions the performance of evolutionary algorithms will not be better than the performance of random search (probably even worse because the EA might get stuck in a local optimum). However, if functions contain regularities that might guide a search algorithm towards regions with better function values, evolutionary algorithms do have an advantage (see also Section 2.2, Section 4.4 or [Men04a]). For multi-objective optimization a discussion about the validity of the no free lunch theorem can be found in [Cor03].

- Real-world problem: In contrast to test functions which are usually artificially created and do not have any meaning in the "real world", optimization problems encountered in various disciplines of e.g. engineering or science are called real-world problems. They are often characterized by a large number of parameters and additionally they commonly exhibit several constraint and objective functions. The constraint and objective functions are often computationally expensive to evaluate, e.g. because complex systems have to be simulated. Even the definition of objective functions for real-world problems is not necessarily trivial [Sch95].

- Rotational invariance: A desired property of optimization algorithms (or its operators, respectively). In [Pri05] a rotationally invariant search algorithm is defined as an algorithm whose performance is not dependent on the orientation of the coordinate system.
- (Self-)Adaptation: If a control parameter is not fixed but it is varied according to the state of the optimization run, it is called adaptive. If control parameters are subject to the evolutionary process, they are called self-adaptive [Eib99].
- Step size: Distance between parent and offspring, usually measured in parameter space.
- Stopping criterion (also called stopping rule, stopping condition or termination criterion): Determines when the execution of an optimization algorithm is terminated.

Chapter 3

Evolutionary Algorithms

Evolutionary computation is a subfield of computational intelligence in which evolutionary algorithms are used for solving optimization problems. Computational intelligence itself is a part of artificial intelligence. Besides evolutionary computation, computational intelligence comprises two other biologically motivated fields which are neural networks and fuzzy systems.

The reason for using principles derived from evolution for optimization purposes is the success of natural evolution. In nature, complex organisms have been created that are adapted to all sorts of environments. Similarly, evolutionary algorithms use evolutionary operators to handle different types of complex optimization problems.

The oldest representatives of this class are Genetic Algorithms (from the 1960s), Evolution Strategies (from the 1970s) and Evolutionary Programming (from the 1960s). A kind of subfield of Genetic Algorithms is the Genetic Programming. It was developed later than the before-mentioned algorithms (from the 1980s) but it also belongs to the most famous evolutionary algorithms. In recent years Differential Evolution which was first published in 1995 also received growing attention, among others due to its favorable convergence characteristics.

A commonality of these evolutionary algorithms is the use of the evolutionary operators recombination, mutation and selection to evolve a population of individuals towards better values of the objective function. Differences exist in the representation, the implementation of the operators and the importance of the different operators. Most of the methods were developed independently from each other but in the meantime there is much exchange between the different fields. The borders between the algorithms become fluid, e.g. the classical Genetic Algorithm uses binary coding but variants with real-valued parameters have also been developed in the meantime. Furthermore, concepts like constraint-handling are exchanged.

A rather new class of optimization algorithms can be summarized under the name "swarm intelligence". SI is an artificial intelligence technique that uses populations of agents which interact with one another. The two most popular methods of this class are Ant Colony Optimization (from the early 1990s) and Particle Swarm Optimization (first published in 1995). While Ant Colony Optimization is only applicable to combinatorial problems (and is therefore not discussed further here; the interested reader is referred to [Dor02]), PSO operates on real numbers and is a global optimization technique. Some researchers regard PSO as an evolutionary algorithm but this classification is controversial and will be discussed in more detail in Section 3.6.

In the following first a short introduction to Genetic Algorithms, Evolution Strategies, Evolutionary Programming and Genetic Programming is given. These algorithms can be seen as kind of predecessors of Differential Evolution and Particle Swarm Optimization in whose context DE and PSO have been developed. Because the focus of this work is on DE and PSO, the subsequent introductions of these algorithms contain more details. In this chapter only unconstrained single-objective optimization is described. Extensions for constraints and multiple objectives are covered in Chapters 4 and 5.

3.1 Genetic Algorithm

Genetic algorithms (GAs) are probably the most famous evolutionary algorithms. They were developed by John H. Holland, firstly introduced in the 1960s, and especially a book of Holland from 1975 received much attention [Hol75]. Other commonly cited textbooks that can be used for obtaining further knowledge beyond the basic information that is given here are [Gol89, Deb01a]. There have been attempts to formulate a convergence proof for GAs [Deb01a] but generally they can be seen as heuristics like the other evolutionary algorithms presented here.
In natural individuals, the genetic material consists of chromosomes which themselves are composed of several genes. These genes can have different states which are called the alleles. In classical GAs the same principle is used. The chromosomes are modelled by binary strings where each bit corresponds to a gene. The different alleles of a gene are represented by the two states of a bit (0 or 1).
Optimization problems mostly have real-valued parameters. Thus, the parameters must be converted into bit strings in order to apply a classical GA. In analogy to natural systems, the chromosome in GAs is also called the genotype whereas the corresponding phenotype consists of the objective function parameters. Each parameter is transformed into a binary number, and the whole chromosome is built by linking these binary numbers together. For example, if there is a parameter with the value $x_1 = 9$ and another parameter with the value $x_2 = 5$, the first one might be represented as 01001 and the second one as 00101, making the whole chromosome to 01001 00101. It follows that the precision as well as the search space is determined by the coding. As it will be shown in the following, recombination and mutation operate on the genotype. For the evolutionary operator selection the performance of a solution must be evaluated. For this purpose the phenotype is used to compute the value of the so-called fitness function (which is often equal to the objective function but may also include a dependence on the constraint functions). In the example, the fitness function may be $f(x_1, x_2) = x_1^2 + x_2^2$, thus the performance would be represented by $f(9, 5) = 106$.
In GAs a selection mechanism is employed both for choosing individuals for reproduction as well as for selecting survivors for the next generation. For these purposes usually a fitness-based selection operator is used, e.g. tournament selection, proportionate selection or ranking selection [Deb01a]. Using tournament selection, two solutions are compared and the better performing one wins, e.g. the one with the smaller fitness for minimization tasks. With proportionate selection, the individuals are selected proportional to their fitness. This is also termed "roulette wheel selection". It is depicted in Figure 3.1 where the probability p_i to select individual i is proportional to its fitness f_i: $p_i = \frac{f_i}{\sum_{j=1}^{NP} f_j}$ where NP is the number of individuals. In ranking selection a similar operator is used but first

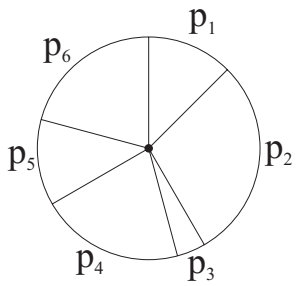


Figure 3.1: Roulette wheel selection

the individuals are sorted based on their fitness value and then they are assigned a rank that equals their new fitness.
Recombination is the main operator in GAs whereas mutation is a secondary operator that is applied with less probability. There are several ways to conduct crossover. The most commonly used variants employ two parents and are given in the following:

- n-point crossover: n points define which part of the chromosome of parent 1 or parent 2 is given to the offspring. In Figure 3.2 two-point crossover is illustrated. Offspring 1 is based on parent 1 but it also receives several bits from parent 2 (light gray). Two points characterize the beginning and the end of the bit sequence that originates from parent 2. Similarly, offspring 2 is based on parent 2 but the bit sequence shown in dark gray comes from parent 1.

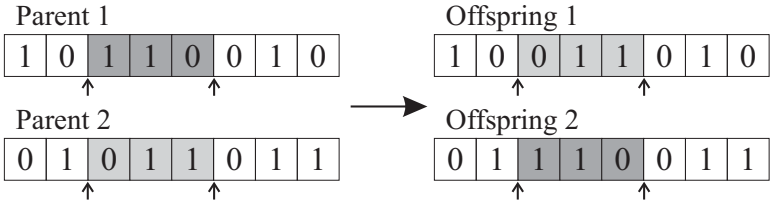


Figure 3.2: Two-point crossover

- Uniform crossover: Bits are taken from each parent with a certain probability.

Recombination is not applied to all members of the mating pool. Instead, it is used with a given probability p_c.
In contrast to recombination, the bit-wise mutation operator does not operate on two parents. Instead, it is applied to a single individual. It is used for bringing diversity into the population by changing a bit with a certain probability p_m.
A drawback of binary-coded GAs is that an unsuitable coding may produce bad results [Deb01c, Ama97]. Problems are for example the existence of Hamming cliffs (the transition to a neighboring solution requires the alteration of more than one bit, e.g. from 01111 to 10000 [Deb01a]) and the need to predefine the precision before conducting the optimization run. Therefore, GAs using real-valued parameters have been developed. These algorithms are able to use the same selection operators as binary-coded GAs because only the fitness of solutions is considered during selection. However, the crossover and mutation operators had to be newly designed because they regard the genotype [Deb01a]. In the literature indications can be found that real-coded GAs perform well [Ama97, Ali04, Wu05], supporting the decision to concentrate on algorithms using real-valued representation here.

3.2 Evolution Strategy

The evolution strategy (ES) was developed in 1963 by two German students: Ingo Rechenberg and Hans-Paul Schwefel [Sch95]. In contrast to GAs that traditionally employ binary strings as parameter representation, ESs use real-valued parameters. In the original form of ESs, only mutation and selection were applied as evolutionary operators, and only one parent and one offspring was used. Later work employed crossover as well, and multi-membered ESs were developed.
In ESs the notation $(\mu/\rho \overset{+}{,} \lambda)$-ES is used to indicate the form of the algorithm. Here μ denotes the number of parents, ρ gives the number of parents that participate in the recombination process and λ is the number of offspring. Two different selection types are used: The $(\mu/\rho + \lambda)$-ES combines all parents and offspring and selects the μ best individuals while the $(\mu/\rho, \lambda)$-ES regards only the offspring for selection. Therefore, it is required for the $(\mu/\rho, \lambda)$-ES that $\lambda \geq \mu$. It should be noted that the $(\mu/\rho + \lambda)$-ES is an elitist algorithm whereas the $(\mu/\rho, \lambda)$-ES is not elitist.
For mutation a Gaussian operator is used. A mutated vector $\vec{y}_i$ is created from a vector $\vec{x}_i$ by

$$\vec{y}_i = \vec{x}_i + \vec{N}(0, \sigma) \tag{3.1}$$

where σ is the mutation length and the components of vector $\vec{N}(0, \sigma)$ are generated using a zero-mean normal distribution with standard deviation σ. As it was discovered that it is beneficial to vary the mutation strength during the optimization run, adaptive strategies were developed so that not only the parameters are evolved but the mutation strength as well. Implementations exist that use the same mutation strength for all population members but also variants have been developed that assign different values of the mutation strength to each individual [Deb01a].
Two different kinds of recombination operators exist: Intermediate or discrete. Using the intermediate recombination operator, an average of the ρ selected solutions is computed. Using the discrete recombination operator, each component of the parameter vector is chosen randomly from one of the ρ parents (similar to the uniform crossover operator in GAs).
The general procedure of ESs is typical for an EA: An initial population is randomly initialized, recombination is performed, the resulting solutions are mutated, survivors for the next generation are selected, and the iterative procedure is terminated if some stopping criterion is fulfilled.
Some convergence proofs for ESs can be found in the literature but as discussed in [Jak04] the practical usefulness is rather low due to the many assumptions made in the process. This is a general problem for evolutionary algorithms because (especially for more sophisticated versions than the basic methods) too many aspects have to be neglected for a convergence proof due to the complexity of the algorithms.

3.3 Evolutionary Programming

Evolutionary programming (EP) was developed by Lawrence J. Fogel in the early 1960s [Deb01a]. Like the ESs, it operates on real-valued parameters. It employs the evolutionary operators mutation and selection but recombination is not used. There are many similarities between ESs and EP but they were developed independently.

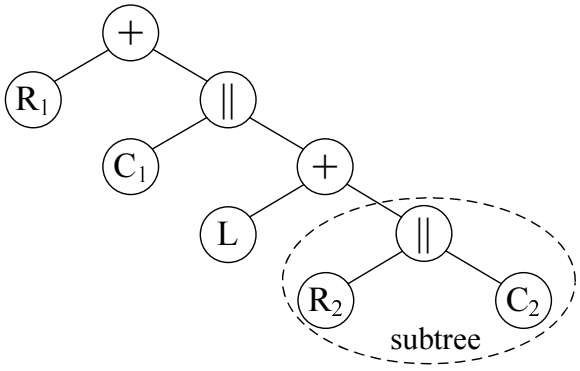


Figure 3.3: Tree data structure used in Genetic Programming

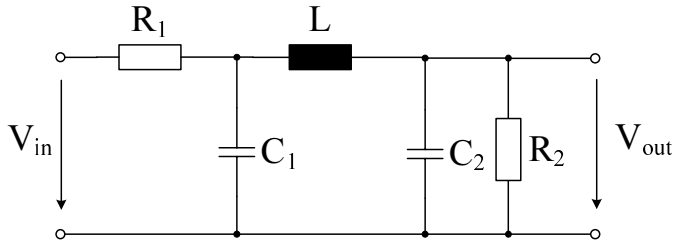


Figure 3.4: Schematic for the tree in Figure 3.3

Like in ESs, offspring are generated by conducting mutation using a normal distribution, and the mutation strength is evolved over time. A difference lies in the selection procedure: First, all parents and offspring are combined. Every solution $\vec{x}_i$ of this combined population is compared with a subset of randomly chosen solutions from the combined population. It is computed how many solutions of the selected set are inferior to $\vec{x}_i$ by evaluating the objective function values. A score is assigned to $\vec{x}_i$ that is proportional to the number of inferior solutions. After this procedure has been done for every solution $\vec{x}_i$ of the combined population, the solutions are sorted due to their score. The solutions with the best scores are selected for the next generation, making EP an elitist algorithm.

3.4 Genetic Programming

Genetic programming (GP) [Koz92] is newer than GAs, ESs and EP as it was developed in the 1980s. It is a variant of GAs where a major difference exists in the data structures: In GP, data is commonly represented as trees instead of the binary-coded or real-coded parameters in GAs. An example is given in Figure 3.3 which is a representation of the circuit shown in Figure 3.4. Otherwise, the process of GP is similar to GAs: In the beginning of an optimization run the individuals are randomly generated, and a fitness is assigned to all of them by comparing the outcome of the solutions with the desired outcome. The crossover operator exchanges subtrees (see Figure 3.3), and the mutation operator replaces an object with another one from a given set.
Among others, GP is successfully applied in circuit design [Koz04]. Using GP, not only the parameter values of existing designs can be optimized but as shown in Figure 3.3 the whole topology itself, i.e. the connection of components, can be created by GP, parallel to optimizing parameter values [Hou05].

3.5 Differential Evolution

In comparison to other evolutionary algorithms, Differential Evolution is a relatively new technique. It was first described by Storn and Price in 1995 [Sto95]. The objective of the authors was to design a method that is easy to use, robust and fast. The first goal is achieved by the small number of user-defined parameters as DE only includes three control parameters. Two of these parameters have rather small ranges of suitable values which contributes to the robustness of DE. Fastness is obtained by the characteristic use of vector differences from the current population for the generation of step sizes. Therefore, an adaptive scaling of step sizes is produced that leads to a high convergence speed (see Section 3.5.1). Moreover, the algorithm is elitist which is a property that is commonly associated with fastness. Some authors also stress that DE is inherently parallel and can be easily executed on parallel computers [Zah03b, Tas04, Pri05]. A convergence proof for DE is missing so far. However, it can be seen in the literature that DE has been successfully applied to many different optimization problems [Pri05].
In the following first the procedure of DE is described for the most commonly used variant DE/rand/1/bin (the notation of the variants is explained in Section 3.5.2). Afterwards, different variants of DE are introduced and recommendations for setting control parameters are given. At the end of this section differences and commonalities of DE with other EAs are summarized. This chapter concentrates on unconstrained single-objective optimization. Extensions for the handling of constrained problems will be discussed in Section 4.2.2, and modifications for multi-objective problems will be addressed in Section 5.3.

3.5.1 General Procedure

The process of Differential Evolution is similar to other evolutionary algorithms: After a random initialization of the population, the individuals are evolved using the evolutionary operators mutation, recombination and selection until a stopping criterion is satisfied (see Figure 3.5).
In contrast to classical Genetic Algorithms that use binary coding for the individuals, the population members in DE consist of real-valued vectors. Their dimension D is equivalent to the number of parameters of the objective function. The number of individuals in each generation G is denoted by the user-defined control parameter NP. The population size NP is usually kept constant during an optimization run (see Section 7.1.1 for exceptions). Thus, each generation contains the individuals $\vec{x}_{i,G}$ with $i \in \{1, \ldots, NP\}$. The components of an individual are referred to as $x_{i,j,G}$ with $j \in \{1, \ldots, D\}$. Sometimes the index G for the generation number is omitted where irrelevant for the issue that is currently discussed. The use of real-valued vectors as individuals contributes to the easiness of DE because no binary coding is necessary for continuous optimization problems (see also [Pri99] and [Fog00] for discussions about advantages of encoding as floating-point numbers).
In the beginning of an optimization run, the DE individuals are usually randomly initialized within the search space:

$$x_{i,j,G=0} = x_{min,j} + rand_j \cdot (x_{max,j} - x_{min,j}) \tag{3.2}$$

where $i \in \{1, \ldots, NP\}$, $j \in \{1, \ldots, D\}$, $x_{min,j}$ and $x_{max,j}$ are the lower and upper limit for parameter j, and $rand_j \in [0, 1]$ is a random variable from a uniform distribution. If there is

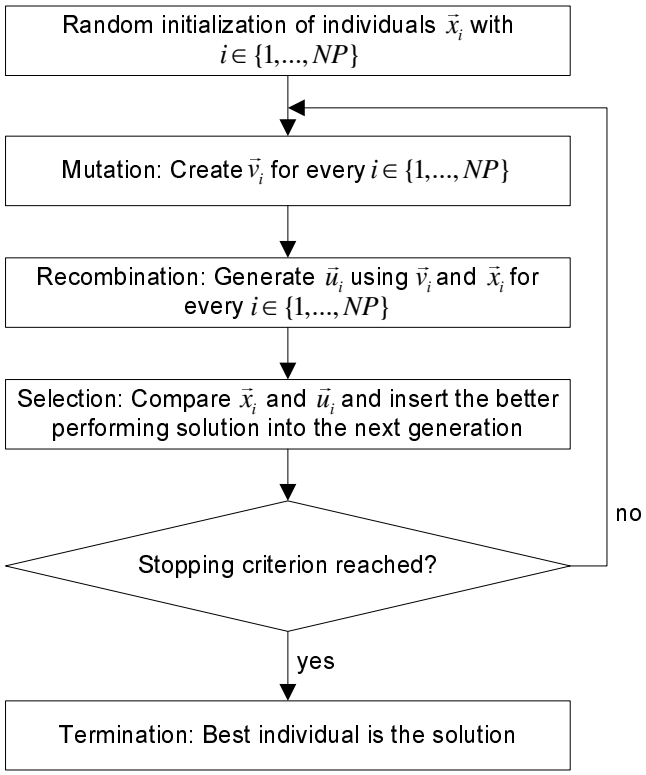


Figure 3.5: Flowchart for Differential Evolution

some a priori knowledge about the location of the global optimum, it is also possible to use another distribution for initialization to increase the convergence speed, e.g. a Gaussian distribution [Pri05]. Because normally no knowledge about the location of the optimum is available, generally the uniform distribution is used. If parameter bounds cannot be estimated reliably, it is also possible to use the estimated bounds only for initialization whereas the individuals' location is not restricted after initialization [Lam04].
Mutation is performed for each population member $\vec{x}_i$. In that context, the solution $\vec{x}_i$ is also commonly called the target vector in the DE literature because it denotes the solution that is currently the target of the evolutionary operators. To familiarize the reader with the general procedure of DE, here only the mutation process of the variant DE/rand/1/bin is described because it is one of the easiest and most commonly used variants. There are also other forms of mutation which will be presented in Section 3.5.2. In DE/rand/1/bin mutation is executed by adding the weighted difference of two randomly chosen population vectors to another individual. It has to be noted that the target vector $\vec{x}_i$ that is later used for comparison with the newly created individual is not involved in the mutation process using this DE variant:

$$\vec{v}_{i,G} = \vec{x}_{r_1,G} + F \cdot (\vec{x}_{r_2,G} - \vec{x}_{r_3,G}) \tag{3.3}$$

with $i \in \{1, \ldots, NP\}$. F is one of the control parameters of DE. It is a real number that is sometimes called the amplification constant or scale factor as it scales the vector difference of $\vec{x}_{r_2,G}$ and $\vec{x}_{r_3,G}$. F is mostly chosen from the interval $[0, 1]$ but some authors also use larger numbers up to 2 [Hua06]. The indices r_1, r_2, r_3 denote three mutually different, randomly chosen members of the current generation which are also different

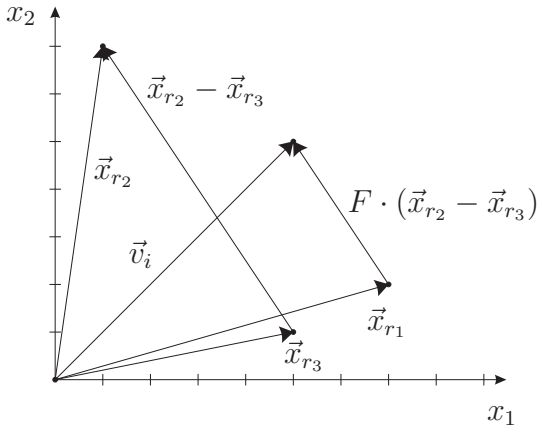


Figure 3.6: Mutation for $D = 2$

from the target vector $\vec{x}_i$. In Figure 3.6 the procedure of generating a mutated vector is shown for $D = 2$.
This mutation operation might look like crossover at first glance because several population members are involved. However, crossover refers to the exchange of vector components which does not happen here. Instead, $\vec{x}_{r_2,G}$ and $\vec{x}_{r_3,G}$ are only used to compute a vector difference, i.e. to generate a step size. Thus, indeed only $\vec{x}_{r_1,G}$ is perturbed by adding a random variation to it. This operation is similar to the mutation in ESs (see Equation 3.1), just the origin of the random variation is different.
The use of population members to generate the random variation during the mutation process is the reason for the automatic adaptation of the step length: An individual is mutated by adding a vector difference, and the magnitude of vector differences changes during the optimization run. In the initial stages of the search, the individuals are scattered throughout the search space. As a result, vector differences are large, and the individuals make large steps to explore the search space. Towards the end of the optimization run, the individuals gather in the vicinity of the optimum. Vector differences are small, and the individuals move in small steps, so the solution will be fine-tuned. Hence, the DE individuals automatically vary their search behavior from emphasizing exploration in the beginning to stressing exploitation towards the end of an optimization run.
In the recombination process a trial vector $\vec{u}_i$ is built for each population member $\vec{x}_i$. In contrast to mutation, the target vector $\vec{x}_i$ is part of the recombination process because the trial vector $\vec{u}_i$ receives components from $\vec{x}_i$ and the mutated vector $\vec{v}_i$:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } rand_j \leq CR \text{ or } j = k \\ x_{i,j,G} & \text{otherwise} \end{cases} \tag{3.4}$$

where $i \in \{1, \ldots, NP\}$ and $j \in \{1, \ldots, D\}$. CR is the crossover constant. It is a control parameter of the DE algorithm that has to be user-defined in the interval $CR \in [0, 1]$. To ensure that the trial vector is different from the target vector in at least one component $j = k$, k is chosen randomly from the interval $k \in \{1, \ldots, D\}$, and k is newly chosen for each population member in every generation. This recombination operation is slightly different from the usual definition of an exchange between several population members because one of the participating solutions is not directly a population member but a mutated solution derived from a population member. In Figure 3.7 recombination is

shown for $D = 2$. In this case only three trial vectors are possible for a given mutated vector ($\vec{u}_{i_1}$, $\vec{u}_{i_2}$, $\vec{u}_{i_3}$). The number of possible trial vectors increases with the dimension and also with the number of possible mutated vectors which may be achieved by a larger population size [Lam00, Pri05].

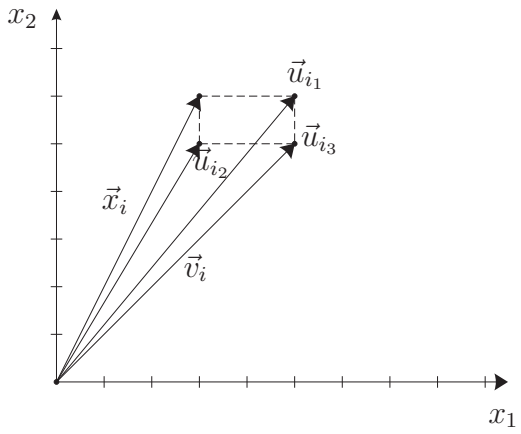


Figure 3.7: Recombination for $D = 2$

The reason for the enforced difference between target vector and trial vector is that they are compared when applying the operator selection. For minimization problems like in this work (see Section 2.1), the vector with the smaller objective function value is inserted into the subsequent generation $G + 1$:

$$\vec{x}_{i,G+1} = \begin{cases} \vec{u}_{i,G} & \text{if } f(\vec{u}_{i,G}) < f(\vec{x}_{i,G}) \\ \vec{x}_{i,G} & \text{otherwise} \end{cases} \tag{3.5}$$

with $i \in \{1, \ldots, NP\}$. This selection scheme contains another difference to other EAs because the trial vector is compared to only one pre-specified member of the population, meaning that the selection process is deterministic.
Especially for functions with flat regions (see Section 2.3), it might be advantageous to prefer the trial vector over the target vector if $f(\vec{u}_i) \leq f(\vec{x}_i)$ instead of $f(\vec{u}_i) < f(\vec{x}_i)$. That way, crossing of flat regions is easier for the individuals [Lam02, Hua06].
The selection operator results in two properties of DE: Because the selection scheme does not allow to replace a population member with an individual that performs worse, so the objective function value cannot deteriorate, it is called greedy [Sto97]. Moreover, the best objective function value cannot get lost when moving from one generation to the next. Thus, DE is an elitist algorithm which is a property that is usually associated with fast convergence behavior [Pri05].
There are two ways of substituting individuals: One possibility is to generate the whole child population first and then apply the selection operator to build the next generation $G+1$, so there is no influence of the child population in generation G. Another possibility is to instantly substitute a parent by the respective offspring if the offspring performs better, meaning that the offspring immediately participates in the evolutionary process in generation G. In [Rob05b] the latter variant is recommended for a multi-objective DE algorithm as it emphasizes elitism. In contrast, in [Kuk04a] it is reported that the same results are obtained with both formulations. Thus, in this work the instant substitution is omitted.

3.5.2 Variants of Differential Evolution

Several variants of DE have been developed that differ in the way mutation and recombination are executed [Onw04b, Mez06a, Pri05]. They are specified using the notation

$$\mathrm{DE}/x/y/z \tag{3.6}$$

where x denotes the vector to be mutated (also called base vector), y is the number of difference vectors and z is the crossover scheme [Pri99]. The vector to be mutated might be a randomly chosen vector (notation: 'rand'), the best vector that was found so far (notation: 'best') or a vector that is located on the connecting line between two solutions, e.g. between the target vector and a random vector (notation: 'current-to-rand') or between the target vector and the best vector (notation: 'current-to-best'). The number of difference vectors is normally set to one or two. Concerning the crossover scheme, a binomial or exponential process can be used (notation: 'bin' or 'exp', respectively). In the binomial process a random variable is compared to CR for every vector component to decide if the respective component should be copied from $\vec{v}_i$ or $\vec{x}_i$ [Pri99]. This is equivalent to the uniform crossover in GAs (see Section 3.1). In contrast, in the exponential process the vector components are taken from $\vec{x}_i$ until the random variable is smaller or equal to CR for the first time. After this, the remaining vector components are copied from $\vec{v}_i$ [Pri99]. This is equivalent to one-point crossover in GAs. The exponential process may also be implemented slightly differently which results in a similarity to two-point crossover [Pri05]. In [Pri99] the use of binomial crossover is recommended but in [Pri08] it is stated that there are no significant differences between the crossover methods.
With the notation given in Equation 3.6, the variant specified in Section 3.5.1 can be described as DE/rand/1/bin. It is one of the earliest DE schemes [Sto95, Sto97] and it is used frequently in the literature [Bab02, Bec05, Lam02, Lam04, Lam00, Onw04a, Rob05b]. In [Bab03b] it is stated that DE/rand/1/bin is the most successful and most widely used DE scheme. [Pri05] refers to this strategy as "classic DE".
There are also variants of DE which are given without the crossover scheme z from Equation 3.6 because recombination is not used, i.e. DE/current-to-rand/1. In contrast to DE/rand/1/bin, this is a rotationally invariant approach [Pri99] (see also Section 2.3).
In Table 3.1 some commonly used DE strategies are given (showing only the mutation process). K is an additional control parameter and $\vec{x}^*$ denotes the best solution found so far. The abbreviation given in the second column is only used in this section.

Table 3.1: Strategies of Differential Evolution

Notation	Abb.	Equation for mutated vector
DE/rand/1	r1	$\vec{v}_i = \vec{x}_{r_1} + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})$
DE/rand/2	r2	$\vec{v}_i = \vec{x}_{r_1} + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3}) + F \cdot (\vec{x}_{r_4} - \vec{x}_{r_5})$
DE/current-to-rand/1	cr1	$\vec{v}_i = \vec{x}_i + K \cdot (\vec{x}_{r_1} - \vec{x}_i) + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3})$
DE/current-to-rand/2	cr2	$\vec{v}_i = \vec{x}_i + K \cdot (\vec{x}_{r_1} - \vec{x}_i) + F \cdot (\vec{x}_{r_2} - \vec{x}_{r_3}) + F \cdot (\vec{x}_{r_4} - \vec{x}_{r_5})$
DE/best/1	b1	$\vec{v}_i = \vec{x}^* + F \cdot (\vec{x}_{r_1} - \vec{x}_{r_2})$
DE/best/2	b2	$\vec{v}_i = \vec{x}^* + F \cdot (\vec{x}_{r_1} - \vec{x}_{r_2}) + F \cdot (\vec{x}_{r_3} - \vec{x}_{r_4})$
DE/current-to-best/1	cb1	$\vec{v}_i = \vec{x}_i + K \cdot (\vec{x}^* - \vec{x}_i) + F \cdot (\vec{x}_{r_1} - \vec{x}_{r_2})$
DE/current-to-best/2	cb2	$\vec{v}_i = \vec{x}_i + K \cdot (\vec{x}^* - \vec{x}_i) + F \cdot (\vec{x}_{r_1} - \vec{x}_{r_2}) + F \cdot (\vec{x}_{r_3} - \vec{x}_{r_4})$

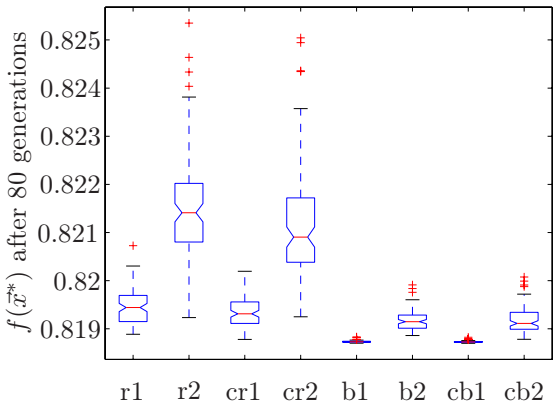


Figure 3.8: Final objective function value for the optimization of a PI cascade control

The strategies from Table 3.1 (using binomial recombination) have been compared within the scope of this thesis for the optimization of a PI cascade control for a multi-mass system [Zie08a]. The results are given in Figures 3.8 and 3.9 in the form of boxplots to permit a clear overview about the data beyond the information that a simple average and standard deviation is able to provide. The lines of the box represent the lower quartile, median and upper quartile of the data. The lines extending from the box show the extent of the rest of the data. By definition, the maximum length of these so-called whiskers is 1.5 times the interquartile range. All data beyond this measure is considered as outliers which are shown as crosses in the plot.
Concerning the final objective function value at the end of the optimization runs, strategies employing one vector difference always performed better than strategies using two vector differences (see Figure 3.8). Besides, methods using the best solution found so far as base vector showed better performance than methods with a randomly chosen base vector. The best final objective function value was obtained by DE/best/1/bin and DE/current-to-best/1/bin. Both strategies also showed a very robust performance, meaning a low range of solutions. Regarding the number of generations for obtaining a robust controller (characterized by $f(x) < 1$), again the strategies employing the best vector found so far

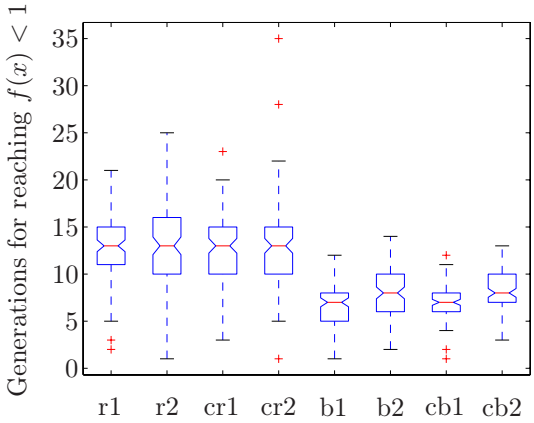


Figure 3.9: Number of generations for reaching $f(x) < 1$ for the optimization of a PI cascade control

as base vector performed better than their counterparts with a random base vector (see Figure 3.9).
In [Zie08a] only one (real-world) optimization problem has been used for the examination. Another study of DE strategies can be found in [Mez06a] where the performance of eight DE strategies has been checked for 13 test problems with varying features. In contrast to the study in [Zie08a], also exponential crossover was tested in [Mez06a]. The following strategies were used in this paper:

- DE/rand/1/bin
- DE/rand/1/exp
- DE/best/1/bin
- DE/best/1/exp
- DE/current-to-rand/1
- DE/current-to-best/1
- DE/current-to-rand/1/bin
- DE/rand/2/dir

where DE/rand/2/dir is a rather new variant that incorporates information about the objective function values by calculating $\vec{v}_i = \vec{x}_{r_1} + \frac{F}{2}(\vec{x}_{r_1} - \vec{x}_{r_2} + \vec{x}_{r_3} - \vec{x}_{r_4})$ with $f(\vec{x}_{r_1}) < f(\vec{x}_{r_2})$ and $f(\vec{x}_{r_3}) < f(\vec{x}_{r_4})$ [Feo04]. A similar mechanism is employed in [Bre08] where one vector difference is used, and its direction is adjusted with a probability of 0.75 so that the gradient is directed towards the better performing solution. In [Mez06a] DE/best/1/bin performed best. An additional interesting result of this examination is that for DE/rand/1 and DE/best/1 the variants with binomial recombination consistently performed better than the strategies with exponential recombination. However, studies like this are always complicated by the dependence on control parameters. In [Mez06a] a random $F \in [0.3, 0.9]$ was generated anew for every generation of each strategy. Because it is assumed that the performance of the algorithm is more sensitive to the choice of CR, CR was tuned for every strategy and each optimization problem separately. That way, the computational effort was high but the best possible performance of each strategy was achieved, enabling a fair comparison.
Apart from the choice of the DE strategy, several other modifications to the basic DE are discussed in the literature. For example, there are attempts to hybridize DE with other optimization techniques to further improve the convergence speed like in [Rog00] where a hybrid of DE and a local search technique is built. To improve the speed but also the search behavior, parallel versions of DE are presented in [Tas04] and also in [Zah03b]. Several subpopulations are developed on different machines and the subpopulations share information by exchanging individuals. This way, optimization problems with expensive objective and constraint functions can be solved faster by employing several machines.

3.5.3 Control Parameters

The basic DE algorithm contains three control parameters: F influences the mutation process, CR is used for recombination, and NP is the population size. Depending on the strategy, also parameter K might have to be set (see Table 3.1). In the literature often $K = F$ is used, e.g. in [Sto99a]. The choice of the DE strategy can also be interpreted as the selection of an additional control parameter which has only some discrete settings. However, in this section the focus is on the setting of F, CR and NP. For these parameters recommendations for single-objective optimization from the literature will be given in the following.

For the population size, settings like $NP \in [5 \cdot D, 10 \cdot D]$ [Sto97] or $NP \in [3 \cdot D, 8 \cdot D]$ [Gäm02] are recommended. For DE/rand/1/bin there must be at least $NP = 4$ individuals while for versions with two vector differences at least five population members are needed. In a parameter study conducted within the scope of this work (published in [Zie06f]) it was shown for a power allocation problem with dimension $D = 16$ (which will be described in detail in Section 4.2.4) that a population size of $NP = 30 \approx 2 \cdot D$ is sufficient for reliable convergence behavior.

The optimal setting of CR is dependent on the decomposability of the objective function. If there are no dependencies between the parameters, a low value of $CR \in [0, 0.2]$ should be selected. High settings of $CR \in [0.9, 1]$ should be chosen if the parameters cannot be optimized independently [Pri05].

It is stated in [Pri05] that values of $F < 0.4$ and $F > 1$ are usually not used. In [Lam04] it is suggested that experiments with unknown optimization problems should be started with $F = 0.9$ (and $CR = 0.9$). In [Zie06f] $F = 0.7$ showed better results, and in [Gäm02] an initial choice of $F = 0.6$ is recommended.

As described in [Lam00] there are two similar processes that can prevent the DE algorithm from converging to the global optimum: Premature convergence and stagnation. In contrast to premature convergence (see Section 2.3), stagnation is marked by no convergence to any point, diversity in the population and occasionally new individuals in the population. According to [Lam00], the reason for stagnation is that only a limited number of new vectors can be built. To overcome this problem, it is suggested that F and CR should not be exactly equal to 1 and NP should not be smaller than 20.

To sum up, for DE some recommendations exist for the setting of control parameters but they are not always in agreement with each other. Thus, some experimentation is usually necessary to find suitable settings for an unknown optimization problem. This holds especially for multi-objective optimization. It has been shown in the literature that good settings for multi-objective optimization may differ considerably from recommended settings for single-objective optimization [Kuk04a]. For example, F and CR may have to be chosen considerably lower to obtain good results. As a consequence, several variants of DE have been developed which use adaptive parameter setting. These will be discussed in Section 7.1.1.

3.5.4 Comparison with other Evolutionary Algorithms

Like other evolutionary algorithms, Differential Evolution follows the "survival of the fittest" principle. Nevertheless, there are some differences in contrast to other EAs. These differences consist of

- the selection of individuals for procreation,
- the order of applying the evolutionary operators,
- the representation of parameters,
- the implementation of the operators and
- the selection scheme.

In DE, every population member is allowed to procreate in each generation whereas in other EAs the parents are often selected using fitness-based operators. One child is generated for every parent in DE, using first mutation and afterwards recombination whereas in GAs the order of applying these operators is reverse. In some variants of e.g. ESs the recombination is even omitted. The real-valued representation is also used in ESs and EP whereas (at least classical) GAs use binary-coded individuals. The difference in the implementation of the operators becomes obvious when comparing Section 3.5.1 and Section 3.5.2 with Sections 3.1, 3.2, 3.3 and 3.4. The DE selection operator compares a parent (target vector) with its child (trial vector) and deterministically determines which one of them is inserted into the next generation. In other EAs the individuals to be compared are not necessarily pre-defined. Furthermore, the decision which individuals to include in the next generation is often done stochastically, even though the fitness of the individuals is also regarded. In [Xu08] this difference is summarized in the statement that DE adopts a greedier and less stochastic approach than other EAs.
DE is similar to ESs [Mad02, Xue05] but DE is less complicated because the self-adjusting capability is derived from differences between population members that are used to generate new vectors (see also Section 3.5.1). Therefore, it is unnecessary to specify external mechanisms which may also include new parameters [Deb01a].
In GAs mutation is applied with a low probability, so more emphasis is placed on recombination. In contrast, in DE (as well as in ESs) mutation is also an important operator.
In general, DE can be seen as farther away from natural evolution than other EAs (especially GAs). This is reflected in the terminology (target vectors and trial vectors in contrast to chromosomes and genes) but the operators are also more abstract. For example, the selection operator that deterministically chooses the better performing solution is unrealistic in nature: As argued in [Ken01], an individual with better characteristics has a higher probability to survive but chance also has an influence. For example, a black animal in a snowy environment might just be lucky that no predator comes along.
A further in-depth discussion of DE properties and DE dynamics can be found in [Pri99].

3.6 Particle Swarm Optimization

Same as Differential Evolution, the Particle Swarm Optimization algorithm was firstly introduced in 1995 [Ken95], so it is also a rather new optimization algorithm. It simulates social behavior of groups like bird flocks or fish swarms where individuals adjust their behavior based on own successes but also on successes of other group members.
Like other evolutionary algorithms, PSO was initially developed for unconstrained single-objective optimization problems. In this section first the general procedure of the PSO algorithm is discussed. Extensions of PSO for constrained single-objective as well as

unconstrained and constrained multi-objective optimization are discussed in Sections 4.2.3 and 5.4. Additionally, some of the most commonly used neighborhood topologies are presented and the control parameters of PSO are described in this section. Because PSO is derived from swarm intelligence but it is often regarded as an evolutionary algorithm, a discussion of similarities and differences with typical evolutionary algorithms is also given.

3.6.1 General Procedure

Due to the origin of PSO, some concepts and terms exist which differ from typical EA terminology. The PSO individuals are also called particles, and they contain some additional information apart from the current position $\vec{x}_i$. Because the individuals are regarded as particles flying through the search space, they also include a current velocity $\vec{v}_i$. To model the behavior of social groups, each particle receives the information about the neighborhood best position $\vec{p}_g$. This is the position that yields the best objective function value found so far in a certain neighborhood (neighborhood topologies will be discussed in the subsequent subsection). Because in social groups the members tend to rely not only on information from the group but also on their own experience, additionally every particle has the knowledge of the personal best position $\vec{p}_i$. This is the position with the best objective function value found so far by the respective particle. All positions and velocities are real-valued vectors with D components where D is the dimension that equals the number of objective function parameters. The population size is denoted by NP, and as for other evolutionary algorithms it is a user-defined parameter.
In the first generation the positions and velocities have to be initialized. Mostly, the current position is set to a random value within the given boundaries $\vec{x}_{max}$ and $\vec{x}_{min}$ as described in Equation 3.2 for DE. The personal best position of a particle is initialized with the respective current position. There are also other possibilities for initialization [Val08] but this is the most common form. Furthermore, the neighborhood best position is initialized by determining the particle with the lowest objective function value in the respective neighborhood.
Concerning the initialization of velocities, different strategies are possible. Either they can be set to zero in the beginning or they can be assigned some random value, e.g. in dependence on the size of the search space. Up to now, there is not much research on this matter. In this work the setting of the velocity is chosen randomly from the interval $[-v_{max,j}, v_{max,j}]$ where $j \in \{1, \ldots, D\}$ and the maximum velocity is one half of the search space in each dimension:

$$v_{max,j} = \left(x_{max,j} - x_{min,j}\right)/2. \tag{3.7}$$

In contrast to the usual definition of a velocity as distance divided by time, in PSO the velocity is defined as difference of positions (see also Equation 3.9). A different interpretation is that each position difference is implicitly divided by one generation which corresponds to a discrete time unit here. It is ensured that velocities do not exceed $v_{max,j}$ during the optimization run by resetting larger velocities to $v_{max,j}$. The reason is that there are cases reported in the literature where oscillations with increasing magnitude occurred [Ken01].
After initialization in the first generation, update equations for velocity and position are applied to each particle in every generation until a stopping criterion is fulfilled (see Figure 3.10). The update equations compute a new velocity $\vec{v}_i(t+1)$ based on the current

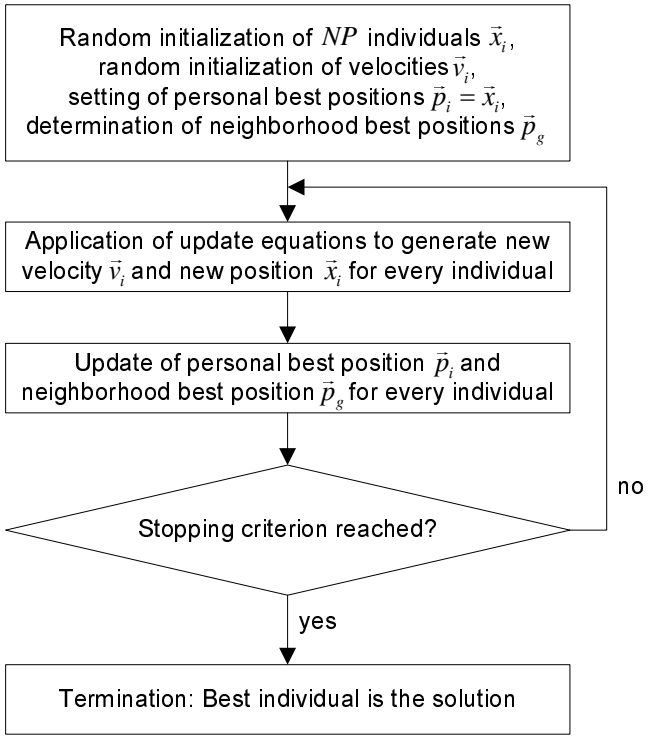


Figure 3.10: Flowchart for Particle Swarm Optimization

velocity as well as a cognitive and a social component, and the position is updated by adding the new velocity to the current position:

$$\vec{v}_i(t+1) = w \cdot \vec{v}_i(t) + c_1 r_1[\vec{p}_i(t) - \vec{x}_i(t)] + c_2 r_2[\vec{p}_g(t) - \vec{x}_i(t)] \quad (3.8)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (3.9)$$

with $i \in \{1, \ldots, NP\}$. In Equation 3.8 the first term on the right-hand side represents the inertia of the particle, i.e. the tendency to fly in the same direction as in the previous time step. Therefore, the parameter w is called inertia weight. The second term is the cognitive component that causes movement towards the personal best position. The third term is the social component that draws the particle towards the neighborhood best position. The cognitive and social components are weighted with the control parameters c_1 and c_2. A stochastic element is provided by the random numbers r_1 and r_2 which are randomly chosen from $[0, 1]$. More detailed information about the control parameters w, c_1 and c_2 will be provided in Section 3.6.3.
In the original formulation of the update equations given in the first paper written about PSO [Ken95], it does not become exactly clear how often the random numbers should be refreshed. Because in that paper the random numbers are simply written as "rand()", it might mean that they should be recalculated once for every particle or once for each component of the velocity. In order to emphasize the second option, the velocity update equation should be reformulated. This can be done by giving the equation for the j-th component of the i-th particle ($j \in \{1, \ldots, D\}$, $i \in \{1, \ldots, NP\}$):

$$v_{i,j}(t+1) = w \cdot v_{i,j}(t) + c_1 r_{1,j}[p_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}[p_{g,j}(t) - x_{i,j}(t)]. \quad (3.10)$$

The difference can also be seen in Figures 3.11 and 3.12 where the movement of a particle

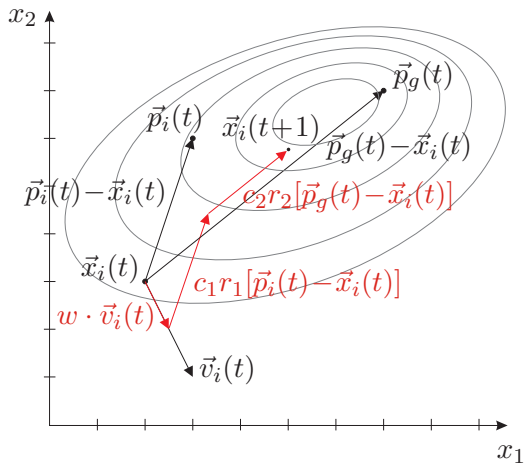


Figure 3.11: Update according to Equation 3.8 for a particle in two dimensions

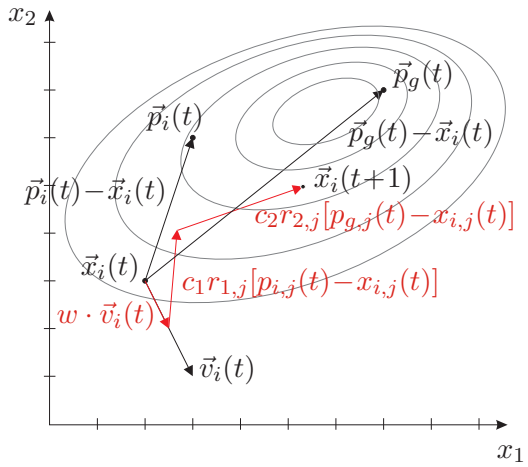


Figure 3.12: Update according to Equation 3.10 for a particle in two dimensions

from $\vec{x}_i(t)$ to $\vec{x}_i(t+1)$ according to the update equations given in Equation 3.8 and Equation 3.10 is shown, respectively. In Figure 3.11 the vector $c_1r_1[\vec{p}_i(t) - \vec{x}_i(t)]$ is parallel to $[\vec{p}_i(t) - \vec{x}_i(t)]$, and $c_2r_2[\vec{p}_g(t) - \vec{x}_i(t)]$ is parallel to $[\vec{p}_g(t) - \vec{x}_i(t)]$. In contrast, these vectors are not parallel in Figure 3.12 because each component is scaled independently.

For several years nothing has been published about this ambiguity but in recent years emphasis of this difference begins to appear [Ken07, Wil05, Cle06, Zie09]. In [Wil05] it is concluded that refreshing the random numbers only once for every particle might result in limitations of the particle trajectories. Consequently, a loss of diversity would occur if the personal and neighborhood best positions stay the same for a long time. However, it is stated that using a different random number for each dimension of every particle also has a disadvantage because this PSO variant is rotationally variant (see Section 2.3). A third formulation that includes the use of rotation matrices is proposed in [Wil05] but this approach is computationally expensive, and furthermore an additional parameter is introduced. As a result, there exist three formulations of PSO which all have specific disadvantages. The drawback of the first variant (according to Equation 3.8) might be compensated most easily because the main problem is introducing enough diversity into

the search. If the personal and neighborhood best positions change frequently, enough diversity might already be present. As stated in [Wil05], there are also several possibilities to generate additional diversity: Using random velocities at initialization (the effect of this may fade over time), employing local neighborhoods (see Section 3.6.2), inserting a craziness operator (an additional operator resembling mutation that introduces a random element to the search) or increasing the number of particles which contribute to the update equations (social awareness).

In a parameter study conducted within the scope of this thesis (published in [Zie09]), both variants for refreshing the random numbers have been tested for a large range of parameter settings and also using different neighborhood topologies on the basis of the power allocation problem described in Section 4.2.4. It was shown that the form according to Equation 3.8 often has a better performance than using Equation 3.10, especially for local neighborhood topologies. In contrast, the results deteriorate significantly for the *gbest* topology which is a global neighborhood topology (see Section 3.6.2). This difference can be explained by the loss in diversity that is discussed in [Wil05]. It is shown that indeed the use of local neighborhoods helps to obtain additional diversity.

After the application of the update equations, the new position is compared to the personal best position. For unconstrained single-objective minimization problems, the personal best position is overwritten by the new position if the new position yields a lower objective function value:

$$\vec{p}_{i,G+1} = \begin{cases} \vec{x}_{i,G+1} & \text{if } f(\vec{x}_{i,G+1}) < f(\vec{p}_{i,G}) \\ \vec{p}_{i,G} & \text{otherwise.} \end{cases} \tag{3.11}$$

with $i \in \{1, \ldots, NP\}$. As for DE, it is also possible to substitute the personal best position if the current position and the personal best position have equal performance in order to facilitate crossing of flat regions. For PSO this might not be as important as for DE. The reason is that the particles are still able to move in a flat region in contrast to DE individuals which may be handicapped due to the greedy selection scheme. Nevertheless, the mobility may be increased.

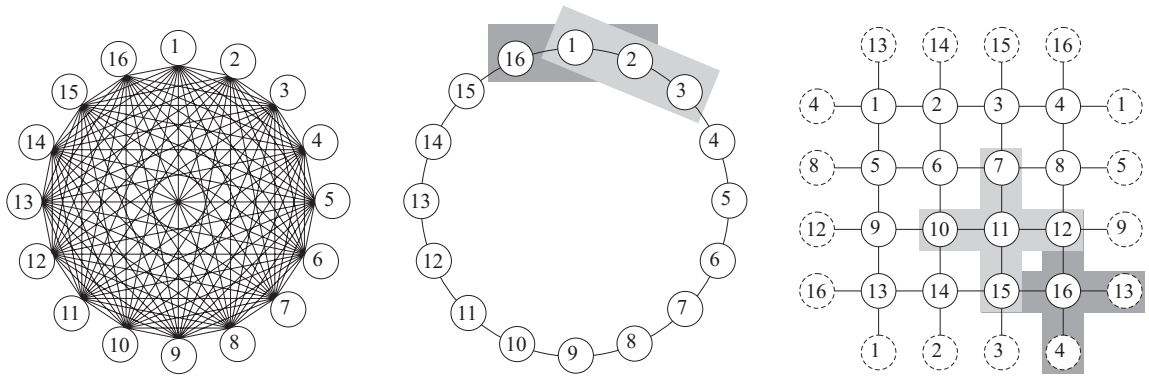
If the personal best position is changed, it is also compared to the neighborhood best position. If the new personal best position has a lower objective function value than the neighborhood best position, the neighborhood best position is also updated. Again, the new position might also be used when it is equal to the previous neighborhood best position.

Similar as described for DE in Section 3.5.1, the update of the best positions may be synchronous (the best positions are updated after all particles have moved in the current generation) or asynchronous (the best positions are updated after the move of each particle) [Car01, Eng06]. In [Car01] it is shown that convergence may be accelerated using asynchronous updates. In the software tool developed for the thesis at hand, the asynchronous update is also used.

3.6.2 Neighborhood Topologies

Several neighborhood topologies have been developed that differ in the number of neighbors per neighborhood and also in the structure. The neighborhood that was developed first is the *gbest* variant [Ken95]. Here, the neighborhood consists of the whole swarm,

so the particles have the information of the globally best solution found so far (see Figure 3.13(a)).
In other topologies the neighborhoods consist of a subset of the population. In the same year as PSO was firstly presented (1995), the *lbest* variant was introduced [Ebe95]. Using this topology, each particle generates a neighborhood consisting of itself and two neighbors. It is also possible to include more than two neighbors but using two neighbors is the most common form of the *lbest* neighborhood topology. Although the term "neighbor" might imply that the particles are close to each other, either regarding parameter space or objective space, in fact the neighborhood is only defined using the indices of the particles. Because every particle generates a neighborhood, neighborhoods are overlapping (see also Figure 3.13(b) where two example neighborhoods are indicated which are generated by particles 1 and 2). It is commonly assumed that the *gbest* neighborhood converges faster but the *lbest* variant experiences less premature convergence, thus it can be used for more complex problems [Ken06b].
The *gbest* and *lbest* variants have become standard choices for neighborhood topologies. They are probably still the most commonly used topologies although several other concepts have been developed in the meantime [Men04b]. A promising candidate is the *von-Neumann* neighborhood topology. It showed good results concerning the convergence probability as well as the convergence speed in [Ken02] and also in a parameter study conducted within the scope of this thesis [Zie09]. Using the *von-Neumann* neighborhood, each particle possesses four neighbors. The topology can be visualized as a two-dimensional lattice that is wrapped on all four sides (torus). Thus, the *von-Neumann* neighborhood of a particle consists of the particles above and below as well as the particles to the left and right (see Figure 3.13(c) where the neighborhoods of particles 11 and 16 are shown). If no neighbor exists (e.g. there is no particle to the right as well as below particle 16 in Figure 3.13(c)), it is generated by wrapping the indices. This procedure makes particles 4 and 13 neighbors of particle 16 in the example. As for *lbest*, neighborhoods are overlapping as can be seen in Figure 3.13(c).



(a) *gbest* (b) *lbest* (c) *von-Neumann*

Figure 3.13: Neighborhood topologies (with $NP = 16$)

An interesting, relatively newly developed neighborhood topology is the so-called "Fully Informed Particle Swarm" (FIPS) [Men04b]. According to [Ken06b], it provides a more realistic model of social relationships by affecting each particle by "a statistical summary of the state of their immediate social network rather than the unique performance of one

individual". The velocity update equation is changed as follows:

$$\vec{v}_i(t+1) = \chi \left(\vec{v}_i(t) + \sum_{j=1}^{N_i} \frac{U(0,\phi)(\vec{p}_j - \vec{x}_i)}{N_i} \right) \tag{3.12}$$

where χ is the constriction coefficient that is discussed in more detail in Section 3.6.3, $U(0,\phi)$ is a random number between 0 and ϕ, and N_i is the number of neighbors for particle i. The position update equation remains the same as given in Equation 3.9. Unfortunately, it is stated in [Ken06b] that it is still not clear which version of PSO performs best.

3.6.3 Control Parameters

In order to apply the update equations described for the basic PSO algorithm in Section 3.6.1, the control parameters w, c_1 and c_2 need to be set. Besides, the neighborhood topology has to be chosen which can (analogous to the strategy used in DE) also be regarded as a control parameter that only allows some discrete settings. A maximum velocity does not necessarily need to be used but often it is set to some fraction of the search space (see Section 3.6.1).
In the following several examinations from the literature concerning PSO control parameters are summarized. Analytical as well as empirical studies are discussed.
Some authors decrease w over time (e.g. [Par00, Løv01, Kri02, Iwa06a]) to emphasize exploration in the beginning of an optimization run but to fine-tune solutions towards the end of an optimization run (exploitation). This approach has the disadvantage that several additional parameters have to be chosen, similar to the annealing scheme that has to be selected for the Simulated Annealing algorithm [Ber01]. Moreover, it is not clear how this approach can be used if the number of generations is not preassigned.
In [Cle02] a theoretical analysis of the influence of control parameters on the performance of PSO is done using a deterministic version of PSO. A PSO variant is derived that does not use the inertia weight but instead a constriction coefficient χ:

$$\vec{v}_i(t+1) = \chi \cdot (\vec{v}_i(t) + \varphi_1 r_1[\vec{p}_i(t) - \vec{x}_i(t)] + \varphi_2 r_2[\vec{p}_g(t) - \vec{x}_i(t)]) \tag{3.13}$$

with

$$\chi = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \tag{3.14}$$

and $\varphi = \varphi_1 + \varphi_2$, $\varphi > 4$. Based on the results from nine test functions it is concluded in [Cle02] that it might be unnecessary to specify problem-dependent parameter settings. Often $\varphi = 4.1$ is used [Men04b] to fulfill the condition $\varphi > 4$, leading to $\chi \approx 0.7298$. Mostly, $\varphi_1 = \varphi_2 = 2.05$ is chosen which means an identical influence of the cognitive and the social component on the movement of particles.
It should be noted that the control parameters of the cognitive and social components are denoted as φ_1 and φ_2 in Equation 3.13 to distinguish them from the parameters c_1 and c_2 used in Equation 3.8. Mostly, this differentiation is not done in the literature. Thus, comparisons of parameter settings are complicated, especially if an inertia weight version is compared with a constriction factor version. For certain parameter settings Equations 3.8

and 3.13 are equivalent [Ebe00]: $w = \chi$, $c_1 = \varphi_1 \cdot \chi$ and $c_2 = \varphi_2 \cdot \chi$. With the above-mentioned parameter settings, the inertia weight version is equal to the constriction factor version for $w \simeq 0.73$ and $c_1 = c_2 \simeq 1.5$. There is no principal difference between these formulations but in the literature it has been established to connect the constriction version with the parameter settings just mentioned. Different parameter settings are more commonly used with the inertia weight version. Furthermore, a linearly decreasing inertia weight as already described in this section is only possible in the inertia weight version. Varying χ in the constriction version would have the undesired effect that φ_1 and φ_2 are also scaled.
The PSO variant with constriction according to Equations 3.13 and 3.14 is often considered as a canonical or standard form of PSO [Bra07] although these parameter settings are not optimal for every optimization problem as empirically shown in [Car01].
In [Tre03] also a theoretical examination is done. The author concludes on the basis of a deterministic version of PSO that convergence will occur if $w < 1$, $c_1 + c_2 > 0$ and $2w - (c_1 + c_2) + 2 > 0$.
Moreover, three different swarm sizes are studied in [Tre03] ($NP = \{15, 30, 60\}$). It is concluded that 30 is a good setting even though four of the test functions have a rather high dimension of $D = 30$. More particles increase the success rate but also increase the computational cost.
In [Kad06] it is stated that stability conditions that have been derived from deterministic PSO versions are not generally valid for the stochastic case. Stability conditions are derived by regarding the PSO equations as a time-invariant linear second-order dynamic model but no general recommendations concerning parameter settings are given.
On the basis of [Kad06], in [Sam07] another examination of optimal parameter settings based on a closed loop stability analysis is done. The authors also conduct experiments with 21 test functions using the *gbest* neighborhood topology. It is concluded that the parameter settings $w = 0.6$, $c_1 = 0.103$ and $c_2 = 2.897$ provide the best results.
Considering the randomness inherent in PSO, a convergence analysis based on stochastic process theory is presented in [Jia07]. The authors show for a *gbest* neighborhood topology that each particle will converge to the global best position found by the swarm for a certain range of parameter settings. However, there is no proof that the global best position will actually correspond to the optimum of the optimization problem. If $c_1 = c_2 = c$ (this is often the case in the literature to simplify the search for suitable parameter settings), the range of parameters is characterized by $0 \leq w < 1$, $0 < c < 2(1 + w)$ and

$$\frac{5c - \sqrt{25c^2 - 336c + 576}}{24} < w < \frac{5c + \sqrt{25c^2 - 336c + 576}}{24}. \tag{3.15}$$

The authors state that many parameter settings recommended in the literature fulfill these conditions.
One of the earliest parameter examinations for PSO can be found in [Ebe95]. A PSO variant still without inertia weight or constriction factor is used (corresponding to a setting of $w = 1$). The maximum velocity and the acceleration constant are varied for the examination ($c_1 = c_2 = \{0.5, 1.0, 2.0\}$). Results are given for two neighborhood sizes (2, 6). The results concerning c_1 and c_2 in terms of median iterations for meeting an error criterion are not conclusive: For a neighborhood size of 6 the setting $c_1 = c_2 = 0.5$ yields the best results but for a neighborhood size of 2 the results of $c_1 = c_2 = 2.0$ are mostly better than for $c_1 = c_2 = 0.5$.

In [Par02b] a Differential Evolution algorithm is used for the calculation of control parameter settings in each iteration of the PSO algorithm. The *gbest* neighborhood topology is used, and the settings of control parameters are allowed in the ranges $0.4 \leq w \leq 1.2$ and $0.1 \leq c_1, c_2 \leq 4$. A problem-dependent population size is used: $NP = 20$ for 2- and 3-dimensional problems, $NP = 40$ for 6-dimensional problems and $NP = 60$ for a 10-dimensional problem. No examinations regarding the population size are reported. Results from twelve test functions indicate that the inertia weight should be chosen from $[0.6, 0.8]$. The sum of c_1 and c_2 is around 3.5, and the setting of c_1 is always larger than c_2. For c_1 mean values in the range $[1.9, 2.2]$ are reported, and for c_2 mean values are in the range $[0.9, 1.5]$.
In [Car01] the dependence of PSO on NP, the neighborhood size, and the ratio of φ_1 and φ_2 is studied among others. One parameter is changed at a time while the others stay fixed. Five test functions are used for the examination, with dimension $D = 30$ for four functions and $D = 2$ for the remaining function. Despite the high dimension a population size of 30 is found to be sufficient ($5 \leq NP \leq 200$ with step size of 5 is tried). Variation of the neighborhood size yields the result that the *gbest* variant is better than *lbest*. For an examination of the cognitive/social ratio, φ_1 is varied between 0 and 4.1, and φ_2 is set to $\varphi_2 = 4.1 - \varphi_1$. The setting of $\varphi_1 = \varphi_2 = 2.05$ that is associated with the constriction variant does not yield the best performance in this study. Instead, $\varphi_1 = 2.8$ and $\varphi_2 = 1.3$ are recommended. The authors assume that increasing the influence of the social component leads to getting trapped in local minima.
It can be seen that recommended parameter settings from the literature often do not match. Besides, in the already mentioned parameter study published in [Zie09] it is shown that there may be complex interactions between w, c_1 and c_2. Therefore, it is difficult to derive general conclusions about suitable parameter settings. For simple problems it might be sufficient to use parameter settings according to the constriction variant (see e.g. [Zie08b] where a comparably simple optimization problem is described). Otherwise, adaptive approaches as presented in Chapter 7 might be beneficial.

3.6.4 Comparison with other Evolutionary Algorithms

The structure of Particle Swarm Optimization is very similar to other EAs since it uses a randomly initialized population that is evolved iteratively until a stopping criterion is fulfilled. It is not a typical evolutionary algorithm because it does not use the evolutionary operator selection. PSO is a cooperative approach where the individuals learn from one another instead of a competitive "survival of the fittest" approach like in EAs.
Nevertheless, PSO is usually classified as an EA because the update equations used in PSO resemble the evolutionary operators recombination and mutation which are employed in other EAs. In [Ken01] it is stated that in Equation 3.8 the term dependent on the personal best position can be seen as mutation and the term with the neighborhood best position may be regarded as recombination. Furthermore, it is argued that self-organization is also a property of evolution [Ken01]. There are authors which oppose this classification, e.g. in [Eng06] it is stated that PSO has similarities with EAs but that the differences outweigh the similarities. Consequently, PSO should be seen as part of the swarm intelligence field but not the EC field although there are also roots in EC.
If the PSO terminology is dropped that each particle moves from one iteration to the next, particles can also be regarded as being replaced by an offspring in the subsequent

generation. This leads to a closer similarity with EAs.
Concepts which distinguish PSO from EAs are e.g. the use of a velocity as well as a memory. Besides, an important topic in the PSO literature is the choice of a suitable neighborhood topology. This is a characteristic that is not part of other EAs. However, the neighborhood topology has a similar meaning for PSO as the choice of the strategy for DE: It determines which vector differences are involved in creating a new solution, thus influencing the step size and direction. A difference is that in PSO the personal best position and the velocity also participate in generating a new solution, hence the neighborhood topology only influences one of three parts of the update equation.
Although both DE and PSO incorporate vector differences in the creation of new solutions, the processes are different because in DE the target vector is often not included in vector differences. In contrast, the current position is generally part of the vector differences in PSO (there are some exceptions, see [Ken06a]). Consequently, a different search behavior is obtained.
For the update of the personal and neighborhood best positions similar rules are applied as for the comparison of trial and target vectors in DE. However, a greedy selection scheme is employed for DE. As a result, the DE individuals cannot be replaced by offspring which perform worse. In contrast, there is no such restriction for the PSO particles because only the best positions are subject to these rules. The result is again a different search behavior which has important consequences in constrained optimization. These will be discussed in Section 4.2.3.

Chapter 4

Constrained Optimization

Most evolutionary algorithms are initially developed for unconstrained optimization. Because real-world problems mostly include constraints, the algorithms are augmented later with techniques for constraint-handling which are the topic of this chapter.
As already mentioned in Section 2.1, there are different types of constraints. In the following first the motivation for using boundary constraints is discussed. Afterwards, methods for keeping individuals inside boundaries during the optimization process are described. In Section 4.2 a classification of techniques for handling constraint functions is shown, together with a description of some of the most popular constraint-handling methods. A short overview about constraint-handling methods used for DE and PSO in the literature is given. Furthermore, the application of DE and PSO to a constrained single-objective real-world problem is shown that consists of optimizing the power allocation scheme for a problem from communications. Based on this problem, two constraint-handling techniques are compared which are characterized by their ease of use: The death penalty and the modified replacement method. Optimization problems including equality constraints are often even more complicated to optimize than problems containing only inequality constraints. Therefore, some specialized methods for handling equality constraints are described in Section 4.3, and their usefulness is demonstrated for a problem from yield analysis. To show the superiority of evolutionary algorithms over random search methods, a comparison between DE, PSO, blind random search and brute force search is shown in Section 4.4 based on a commonly used set of constrained benchmark functions. This chapter ends with a short summary and directions for future work in Section 4.5.

4.1 Boundary Constraints

As mentioned in Section 2.1 during the definition of the general optimization problem, usually boundary constraints $x_{min,j}$ and $x_{max,j}$ according to Equation 2.4 are present in optimization problems to define the search space (see also Figure 4.1). In the following, reasons for using boundary constraints are given. Afterwards, it is described how individuals can be forced to stay inside the given limits.
The use of boundary constraints usually has two reasons. The first one is that it may be necessary to keep the parameter values in a particular range, e.g. if models are only valid for certain input variables or if physical limitations exist. For example, there are no negative values for electronic components like resistors or capacitors. In this case it is especially important that individuals do not cross boundaries (or that the respective

objective or constraint function is not evaluated for individuals outside the search space). Hence, it is reasonable to handle boundary constraints differently than other constraint functions because there are constraint-handling methods which permit (or even explicitly encourage) the existence of infeasible individuals as it will be shown in the following section. Techniques should be used that prevent the algorithm from exploring the regions outside the boundaries, especially if it is ensured that the optimum cannot be situated outside the boundaries (see also [Pri99]).
The second reason for using boundary constraints is that the search space should be restricted. Thereby, the optimization problem is simplified and the computational cost is reduced. In that case it should again be ensured that the global optimum is situated inside the boundaries. If that is not possible because no information about the global optimum is available, it is possible to use the boundaries only for initialization and allow the boundaries to be crossed later, or no boundaries may be set at all. When using computers for the optimization process (which is generally the case), in fact there are always some boundaries which are defined by the largest and smallest numbers that are possible for the used data structure. However, in general this restriction has no consequences.
When comparing the performance of optimization algorithms, it is important to ensure that the same boundaries are used. Otherwise, the computational effort is not comparable because the computational cost for exploring the search space increases with its size.

4.1.1 Methods for the Handling of Boundary Constraints

Different methods can be used to keep the individuals inside the given search space. The techniques which will be presented in the following are generally applicable methods which can be used for any EA. They are applied when a new solution is generated, meaning that for DE they are used for the trial vectors before they are compared to the target vectors. For PSO they may modify a newly calculated current position before it is compared to the personal and neighborhood best position.
If an individual is generated that violates boundary constraints, it can be forced back to the search space as follows:

- The violating parameter may be set to the boundary value (this is done e.g. in [Bre06a]). It is a simple method but it tends to decrease the diversity of the population [Pri99].

- The parameter may be re-initialized. No diversity problems are generated by this method [Pri99] but if convergence has been nearly reached, it may be delayed because the individual will most likely "jump" to another part of the search space due to the random initialization.

- The parameter value may be newly calculated using mutation and recombination (or the update equations, respectively) until a position is found that does not violate any boundary constraint. It is regarded as a less efficient method than re-initializing for DE [Lam04].

- The parameter may be reflected back from the boundary by the amount of violation [Rön05]. For an individual $\vec{x}_i$ that violates parameter j in generation G, this can

be done as follows:

$$x_{i,j,G} = \begin{cases} 2 \cdot x_{min,j} - x_{i,j,G} & \text{if } x_{i,j,G} < x_{min,j} \\ 2 \cdot x_{max,j} - x_{i,j,G} & \text{if } x_{i,j,G} > x_{max,j} \\ x_{i,j,G} & \text{otherwise.} \end{cases} \tag{4.1}$$

This approach is also used in [Bre06c] where an equal probability is given for using this method as well as setting the parameter to the boundary value.

- The limit-exceeding parameter j of an individual $\vec{x}_i$ in generation G may be reset to a point in the middle between the old position in generation $G-1$ and the violated limit:

$$x_{i,j,G} = \begin{cases} \frac{1}{2}\left(x_{i,j,G-1} + x_{min,j}\right) & \text{if } x_{i,j,G} < x_{min,j} \\ \frac{1}{2}\left(x_{i,j,G-1} + x_{max,j}\right) & \text{if } x_{i,j,G} > x_{max,j} \\ x_{i,j,G} & \text{otherwise} \end{cases} \tag{4.2}$$

Thereby, the boundary is approached asymptotically [Pri99]. This approach allows to continue the search in the same region as before without diversity problems. Furthermore, there is no risk of possibly applying the evolutionary operators or update equations many times until an individual inside the boundaries is found. Due to these advantages, this approach is used in this work.
An example is shown in Figure 4.1 where for generation G initially an individual $\vec{x}'_{i,G}$ is generated that violates the boundary $x_{min,2}$. Consequently, the second parameter of this individual is modified according to Equation 4.2, resulting in the position $\vec{x}_{i,G}$.

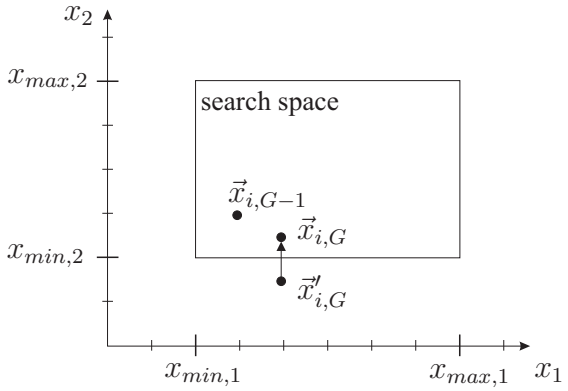


Figure 4.1: Handling of boundary constraints

If this method is applied for PSO, it may be beneficial to also adjust the velocity because otherwise the individual may repeatedly cross the boundaries. Several possibilities exist for the actual implementation but in this work they have not been examined, so this is a possible topic for future work.

It might be assumed that this method leads to slow convergence if the optimum is located near a boundary. However, it must be kept in mind that this method is not the only means of moving the individuals. Instead, it will generally only be applied to a small number of individuals. Due to the self-adjusting capability of DE

and PSO, the step sizes will already be quite small when approaching convergence. Thus, crossing of boundaries will only happen occasionally, so the convergence speed is not influenced noticeably.

Additional methods for the handling of boundary constraints are given in [Alv05] for a multi-objective PSO algorithm. Because methods for the handling of boundary constraints do not depend on the objective function(s) but only on the parameters, the techniques described in [Alv05] as well as the methods presented in this section can be used for both single-objective and multi-objective optimization.
It should be noted that preventing individuals to cross boundaries is debated in the PSO community as several researchers feel it is better not to disturb the motion of the particles [Ken07]. It is argued that if the objective function cannot be evaluated outside the boundaries due to physical limitations, the particles could nevertheless be allowed to go outside of boundaries but without evaluating the objective function. Because positions outside bounds would never become personal or neighborhood best position, the particles would sooner or later be drawn back to the search space inside the boundaries, and they would have no influence on other particles. The question remains what is the benefit of letting individuals go outside the boundaries. If it is known that searching outside the boundaries will provide no useful knowledge, there is no reason to do it (unless to simplify a study of the motion of particles). In contrast, if it is not sure that the lower and upper bounds are set correctly for the respective optimization problem, i.e. if there is doubt whether or not the optimum is situated inside the boundaries, then methods for preventing individuals to cross boundaries should not be applied. This case might arise for problems with computationally expensive objective and constraint functions because limiting the search space drastically reduces the computational effort for optimization. However, if intermediate results give reason to assume that improvement can be found outside the boundaries, the population should be allowed to cross them.

4.2 Constraint Functions

In this section, methods for handling constraints in optimization problems will be presented and compared. These methods are basically capable of handling inequality as well as equality constraints. However, they may not always be effective for equality constraints due to the small feasible space that complicates thorough sampling of the search space. Specialized methods which have been explicitly developed for equality constraints will be discussed in Section 4.3. In the following, first some general comments concerning constraint-handling are given, a classification of methods for handling constraints is shown and some techniques are introduced where the focus is on easy applicability. Afterwards, it is described how exactly these methods may be used in DE and PSO, and a short literature survey of constraint-handling in DE and PSO is given. Furthermore, the application of constraint-handling methods for DE and PSO is illustrated by an exemplary real-world problem from communications. For more information regarding constraint-handling techniques the reader should refer to [Coe02a, Koz99, Mic96, Mez04].
The term "constraint functions" refers to functions which describe dependencies of the objective function parameters that exist additionally to the boundary constraints [Lam04]. In contrast to the boundary constraints which are usually given in a very simple form that

is easy to observe (see Equation 2.4), constraint functions may have a very complicated form.
As mentioned in Section 2.1, constraint functions are commonly divided into inequality constraints (Equation 2.5) and equality constraints (Equation 2.6). An example for an inequality constraint is the demand of stability when optimizing e.g. operational amplifiers, filters or oscillators, thus the phase margin is required to be in a certain range. In engineering design problems mostly inequality constraints are encountered but in Section 4.3.2 also an example for an engineering problem that arises in yield analysis will be shown that includes an equality constraint.
Because of numerical difficulties, equality constraints are usually transformed into inequality constraints using Equation 2.7. Values like $\epsilon_e = 10^{-4}$ or $\epsilon_e = 10^{-3}$ are frequently used for the transformation in the literature [Mez04, Lia06a]. Especially for real-world problems often a considerably better accuracy is needed (see Section 4.3). Because a definite standard setting for ϵ_e does not exist, comparisons with literature may be complicated because generally results of examinations using different tolerances cannot be compared.
A feasible solution must fulfill Equations 2.5 and 2.7. For the comparison of infeasible solutions, in several constraint-handling methods a measure must be used that gives information about the amount of infeasibility. In that case the sum of constraint violation can be calculated:

$$v(\vec{x}) = \sum_{j=1}^{J} G_j(\vec{x}) + \sum_{k=1}^{K} H_k(\vec{x}) \tag{4.3}$$

where

$$G_j(\vec{x}) = \begin{cases} g_j(\vec{x}) & \text{if } g_j(\vec{x}) > 0 \\ 0 & \text{if } g_j(\vec{x}) \leq 0 \end{cases} \tag{4.4}$$

and

$$H_k(\vec{x}) = \begin{cases} |h_k(\vec{x})| & \text{if } |h_k(\vec{x})| - \epsilon_e > 0 \\ 0 & \text{if } |h_k(\vec{x})| - \epsilon_e \leq 0. \end{cases} \tag{4.5}$$

This means that terms are only added to $G_j(\vec{x})$ and $H_k(\vec{x})$ if the respective constraint is not fulfilled. If all constraints are fulfilled, the solution is feasible and its sum of constraint violation is zero.

4.2.1 Constraint-handling methods

Many ways exist for considering constraint functions in optimization problems. According to [Coe02a] they can be classified in the following categories:

- Penalty functions,
- special representation and operators,
- repair algorithms,
- separation of constraints and objectives,
- hybrid methods.

Many constraint-handling methods are general techniques that can be inserted into any EA. Furthermore, they can not only be used for single-objective optimization but they can be easily adapted for multi-objective optimization. There also exist some specialized procedures that have been developed for specific algorithms. In this work the focus is on easily applicable methods, hence the categories special representation and operators as well as hybrid methods are not discussed further. Repair algorithms refer mostly to combinatorial optimization problems (see Section 2.2) and are also omitted.
A common approach for constraint-handling is the use of **penalty functions** which is also a rather old approach that was introduced in the 1940s [Coe02a]. A constrained optimization problem is transformed into an unconstrained problem by adding terms in dependence on the constraint violation to the objective function:

$$\tilde{f}(\vec{x}) = f(\vec{x}) + \sum_{j=1}^{J} r_{ineq,j} \cdot G_j(\vec{x}) + \sum_{k=1}^{K} r_{eq,k} \cdot H_k(\vec{x}) \tag{4.6}$$

where $r_{ineq,j}$ and $r_{eq,k}$ are penalty factors which usually have to be set by the user. Different methods have been suggested, e.g. static, dynamic or adaptive penalties. Mostly these methods insert additional parameters that have to be adjusted. This is a drawback of penalty methods, especially because appropriate settings are highly dependent on the optimization problem.
An exception is the so-called **death penalty**. Using this method, all infeasible points are rejected. It is the easiest way to handle constraints but it is unfavorable in certain situations, e.g. if the search space consists of several feasible regions that are separated from each other because the gaps between the feasible regions may not be traversed by the individuals. Another unfavorable situation may arise in the presence of highly constrained search spaces, meaning that the feasible region is small in comparison with the whole search space. The algorithm has to randomly generate solutions until feasible positions are found for the entire initial population. This may take considerable computational time.
A different approach for constraint-handling is described in [Deb00] for single-objective optimization and in [Deb02] for multi-objective optimization. In the following this approach will be referred to as **modified replacement method**. According to the classification from [Coe02a] that is given above, it belongs to the category **separation of constraints and objectives**. The method is based on preferring feasible over infeasible solutions but in contrast to the death penalty the existence of infeasible solutions is permitted. More exactly, if comparing two vectors $\vec{x}$ and $\vec{y}$, $\vec{x}$ is considered better than $\vec{y}$ if:

- Both solutions are feasible but $\vec{x}$ has a better performance concerning objective function value(s), meaning $f(\vec{x}) < f(\vec{y})$ or $\vec{x} \prec \vec{y}$ for single-objective or multi-objective optimization, respectively.
- $\vec{x}$ is feasible and $\vec{y}$ is not.
- Both solutions are infeasible but $\vec{x}$ has a lower sum of constraint violation than $\vec{y}$ ($v(\vec{x}) < v(\vec{y})$).

Using this approach, the original selection method is used if two feasible individuals are compared. A feasible solution is always preferred over an infeasible solution. In

the comparison of two infeasible solutions, the one that is closer to the feasible space should win to direct the search towards the feasible region. Consequently, the sum of constraint violation is used as basis for the comparison. The objective function is not considered for infeasible solutions. This is an advantage especially for real-world problems because often the objective function cannot be evaluated for infeasible individuals, and furthermore computational time is saved. If several constraint functions are present in an optimization problem, and especially if there are large differences in their magnitudes, the constraints should be scaled before summing up to ensure equal treatment of all constraints. This recommendation does not apply if there is some information available about the importance of the individual constraints but that is usually not the case.
In the modified replacement method it is assumed that feasible solutions should always be preferred over infeasible solutions, and furthermore that an infeasible solution with lower constraint violation is always better than an infeasible solution with higher constraint violation. Other constraint-handling techniques may make different assumptions, e.g. that the infeasible solution with the best objective function value should be included in the next generation or that not the amount of constraint violation according to Equation 4.3 should be used for a comparison of infeasible solutions but the number of violated constraints [Eng06]. For example, the popular stochastic ranking method [Run00] introduces a probability that only the objective function is used for the comparison of two infeasible solutions. Moreover, there are approaches in the literature which explicitly try to establish a balance between feasible and infeasible solutions instead of decreasing the number of infeasible solutions as fast as possible [Yuc04, Wan08a].
Another rather popular constraint-handling method from the category "separation of constraints and objectives" reformulates a constrained single-objective optimization problem with $J+K$ constraints as a multi-objective optimization problem with $J+K+1$ objectives [Coe02a]. As a result, the search does not necessarily aim for a solution that fulfills all constraints but trade-off solutions with minimal constraint violations are found. Because of the many problems associated with multi-objective optimization that will be discussed in the following chapter and that arise especially if the number of objectives is high, this approach is not examined here further.
As can be seen, constraint-handling methods can be distinguished based on the number of additional parameters and also based on the fact if they allow the existence of infeasible solutions (or even encourage it as in stochastic ranking). Both the death penalty and the modified replacement procedure do not introduce new parameters, making them easy-to-use methods. Furthermore, both methods can be used for single-objective as well as multi-objective optimization. A difference is that the death penalty explicitly prohibits infeasible individuals while infeasible solutions are permitted using the modified replacement procedure. It will be examined in this section if this leads to performance differences for DE and PSO.

4.2.2 Differential Evolution for Constrained Optimization

Like most optimization algorithms, Differential Evolution was initially developed for unconstrained single-objective optimization. Because of the significance of constraints for real-world problems, it was later enhanced for the optimization of constrained single-objective problems. In the literature different approaches can be found which normally vary the comparison of the trial vectors $\vec{u}_i$ and the respective target vectors $\vec{x}_i$ dur-

ing selection. In the following it will be described in detail how the death penalty and the modified replacement procedure can be applied for DE, and implications of these constraint-handling methods for DE are discussed. Furthermore, some references to related literature are given.

If the **death penalty** is applied for DE, all individuals must be initialized with feasible solutions in the beginning. If only a limited number of function evaluations is allowed for comparability reasons, many of them might be needed for finding feasible initial solutions, based on the size of the feasible space. If an initial population has been found, the operators mutation and recombination can be applied as described in Section 3.5. During selection trial vectors are only accepted if they are feasible and if they additionally have a smaller objective function value than the corresponding target vectors. Consequently, the implementation of the death penalty for DE is easy because it mainly consists of repeating the initialization procedure until enough feasible solutions have been found and adapting the selection scheme slightly to check for feasibility. In case of an unconstrained optimization problem, the algorithm is the same as the original DE described in Section 3.5.

The implementation of the **modified replacement method** is similarly easy for DE: The selection procedure must be modified to consider not only the objective function value but also feasibility and constraint violation. As for the death penalty, the algorithm equals the original DE for unconstrained optimization problems. When applying the modified replacement procedure, infeasible individuals are driven to feasible space. It also means that individuals which become feasible may never be replaced by an offspring that is located in infeasible space again. This might make crossing of infeasible space or moving in a highly constrained search space difficult because many trial vectors will not be accepted. To cross infeasible regions, the DE individuals must generate suitable vector differences for finding a feasible position that furthermore yields a better objective function value than the old position.

The modified replacement method is often used in the DE literature but slightly differing formulations can be found. In [Bre06c] the constraint-handling method is almost the same as the modified replacement method described in the previous section but the mean constraint violation is regarded instead of the sum of constraint violation in the comparison of two infeasible individuals. However, as this is only a scaling with a constant $(J + K)$, the effect will be the same.

A computationally more expensive variant of the modified replacement procedure can be found in [Hua06] where each single constraint-violation $G_j(\vec{x})$ and $H_k(\vec{x})$ is scaled with the largest violation $G_{max,j}$ and $H_{max,k}$ that was found so far for the respective constraint:

$$v(\vec{x}) = \frac{\sum_{j=1}^{J} w_{1,j} \cdot G_j(\vec{x}) + \sum_{k=1}^{K} w_{2,k} \cdot H_k(\vec{x})}{\sum_{j=1}^{J} w_{1,j} + \sum_{k=1}^{K} w_{2,k}} \tag{4.7}$$

where $w_{1,j} = \frac{1}{G_{max,j}}$ and $w_{2,k} = \frac{1}{H_{max,k}}$. The scaling of constraints should eliminate unwanted effects due to differing magnitude of the constraints, so each constraint is treated equally.

In [Lam04] the modified replacement method is used with the following modification: The comparison of two infeasible solutions is not based on the sum of constraint violation but on Pareto-optimality. Thus, a solution is considered better if it does not violate any constraint more than another solution, and it violates at least one of the constraints less, meaning that dominance in constraint violation space is checked.

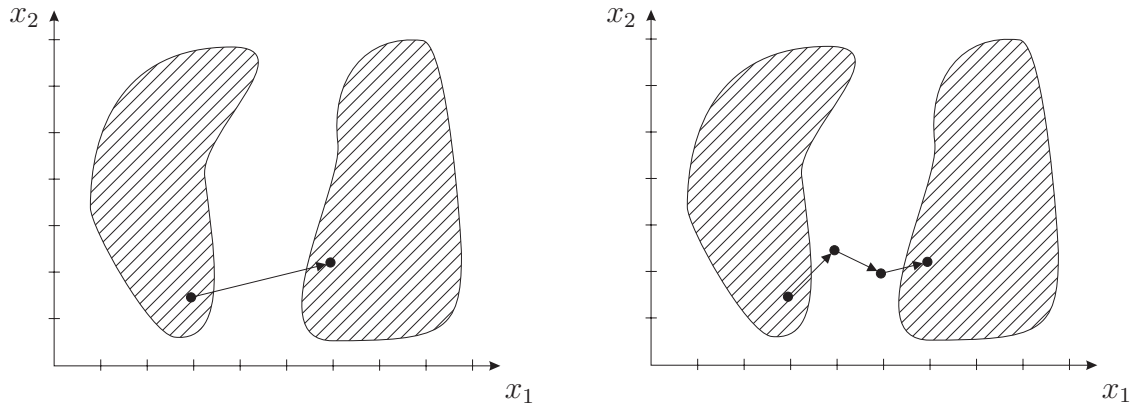
4.2.3 Particle Swarm Optimization for Constrained Optimization

The basic PSO algorithm was also developed for unconstrained single-objective optimization problems but several constraint-handling methods have been used for PSO in the meantime. As there is no selection operator in PSO, modifications have to be done in another place. This is usually either the movement of a particle to a new position or the update of the personal and neighborhood best positions. In the following the application of the death penalty and the modified replacement method is described for PSO, and some references to the literature of PSO for constrained optimization are given.

When the **death penalty** is applied to PSO, the same consideration for the generation of the initial population holds as for DE: Enough feasible solutions must be found to initialize the population. Depending on the size of the feasible space, this may require many function evaluations. When the initial population has been generated, the update equations are used as described in Section 3.6 to generate new positions for the individuals. If an individual moves to an infeasible position, it is reset to its previous position. Thus, the implementation of the death penalty is also easy for PSO. Same as for DE, the death penalty does not create a modification in the basic algorithm but it is only an extension, meaning that for unconstrained optimization the behavior of PSO will be unchanged. Similar as discussed for boundary constraints in Section 4.1, problems might arise if the velocity is kept unchanged because the individuals might try to move into the same direction repeatedly. However, a change of direction might also be caused when the neighborhood best position changes because a particle in the neighborhood has found a better position.

The **modified replacement method** is applied for PSO when a newly generated position $\vec{x}_i$ is compared to the personal best position $\vec{p}_i$ to determine if $\vec{p}_i$ should be replaced by $\vec{x}_i$ because the new position yields a better solution than the present personal best position. The rule is also used when a solution is compared to the neighborhood best position $\vec{p}_g$ to determine if the neighborhood best position can be updated with a better performing solution. Again, the implementation is simple and the original PSO algorithm remains unaltered in case of an unconstrained problem.

Although the modified replacement procedure can be applied to both DE and PSO, different behavior of the algorithms will be the result. The individuals of both algorithms will be driven to feasible space by this constraint-handling method but as already mentioned in Section 4.2.2, the DE individuals cannot become infeasible if they have been feasible once. In contrast, the PSO particles are able to move to infeasible space at any time, so they are less restricted in their movement. The PSO particles are less directly influenced by the constraint-handling method as only the personal and neighborhood best positions are subject to the feasibility rules. As a result, especially in heavily constrained search spaces PSO has an advantage in contrast to DE because the particles are able to move more freely and may therefore explore the search space more thoroughly than the DE individuals. This behavior is illustrated in Figure 4.2 where the feasible space is disconnected, so it consists of two separate areas: In Figure 4.2(a) a feasible DE individual can cross the infeasible space only if a step size can be generated that is large enough and if the objective function value is lower at the new position. In contrast, in Figure 4.2(b) it is shown that the PSO particles are allowed to move into the infeasible space, thus they are also able to reach the disconnected feasible region with several smaller steps. As a result, the probability is larger that a disconnected feasible region will be found.



(a) Differential Evolution (b) Particle Swarm Optimization

Figure 4.2: Differences between DE and PSO when applying the modified replacement procedure (shaded area: feasible space)

In the following some references to literature are given where PSO is used for constrained single-objective optimization, using the approaches discussed here and also others.
Among others, the death penalty is used for PSO in [Hu02]. It is reported that the method is successful for inequality constraints but exhibits problems with equality constraints because feasible individuals have to be found for the entire initial population.
The first use of the modified replacement procedure in PSO is reported in [Pul04a], with the difference that the constraint violations are scaled in [Pul04a]. It is shown that the approach has a competitive performance when compared to three other methods.
In [Muñ06] the modified replacement procedure is also used but additionally a strategy is incorporated that is specifically designed for handling equality constraints. For this purpose an external file is maintained that preserves individuals that are feasible at different tolerance levels ϵ_e which are changed during an optimization run. In every generation the personal best position that performs best of the entire population is inserted into the archive, and the best solution from the archive replaces this personal best position.
A static penalty approach is used in [Li04], and it shows good results for three real-world problems. In [Par02a] a dynamic penalty function approach is used for the handling of constraints. Several test functions are successfully optimized but the use of penalty functions has the disadvantage that penalty values have to be chosen, thus the effort for parameter tuning is increased.
In [Coa03] the constraint-handling techniques of [Hu02] (death penalty) and [Par02a] (dynamic penalty) are compared. Because generally the dynamic penalty approach leads to faster convergence but the death penalty mostly has a better average accuracy ϵ_g, it is concluded that the choice of constraint-handling methods is problem-dependent. Again, it is shown that penalty factors require some fine-tuning effort.
An approach that is specifically developed for PSO is proposed in [Lia06b] where several subswarms are built, and the constraints (and the objective function) are dynamically assigned to subswarms. The method is evaluated in [Lia06b] based on the demands given for the Special Session on Constrained Real Parameter Optimization at the Congress on Evolutionary Computation 2006 in [Lia06a] (these demands will be discussed in more detail in Section 4.4). In comparison to the other approaches that were tested in this special session, the method from [Lia06b] reached very good results (rank 2 out of 12

papers), making the use of several subswarms to a rather complicated but promising means for constraint-handling.
Several subpopulations are also used in [Chu04] to model the symbiosis phenomenon from nature that is used for constraint-handling of a distributed multi-objective PSO algorithm. However, more comparisons are necessary to draw conclusions about the performance of the algorithm.

4.2.4 Application: Power Allocation Problem

In this section the application of DE and PSO to two similar problems from communications is described. The problems consist of optimizing a power allocation scheme for iterative parallel interference cancellation (PIC) as well as iterative successive interference cancellation (SIC). In the literature problems like the PIC have also been optimized, e.g. in [Cai04]. However, Linear Programming techniques have been employed which have the disadvantage that only local optima can be found, so the algorithm becomes dependent on starting positions (see Section 2.2). Furthermore, the formulation of the constraints is restricted. The latter disadvantage prevents the optimization of SIC with these techniques whereas DE and PSO do not have this restriction.
In the following first some information about the background of the optimization problem is given. Afterwards, solutions for PIC und SIC are shown which have been generated using DE as well as PSO. The PIC optimization problem is then taken as basis for a comparison of two constraint-handling methods which are the death penalty and the modified replacement method.

4.2.4.1 Background

In communications the problem exists how to allow multiple users to transmit data in a system simultaneously. Classical methods include TDMA (Time Division Multiple Access) and FDMA (Frequency Division Multiple Access). Using these techniques, time slots or certain frequency bands are assigned to the individual users, thus ensuring orthogonality, i.e. no mutual disturbance between the users. An alternative is random spread CDMA (Code Division Multiple Access) that uses codes to differentiate between users. CDMA has some promising properties which are exploited e.g. in UMTS (Universal Mobile Telecommunications System). For example, there is no hard limit of supportable users in a cell as in TDMA or FDMA at the cost that orthogonality is not given anymore, so interference occurs. Each additional user attending the system slightly degrades the bit error rate. Therefore, this effect is called soft degradation. Another advantage is the robustness against narrowband interferers.
Several techniques can be applied to improve the performance of a CDMA system (see e.g. [Lin04]). Here the focus is on the uplink of the system. The base station receives messages from several users simultaneously. Thus, there will be multi-user interference that may significantly degrade the detection performance.
The effect of multi-user interference can be avoided by using interference cancellation methods which iteratively estimate the interference. If the multi-user interference can be estimated reliably, it can be subtracted from the received signal before detection. The initial estimation may not be reliable but if it is subtracted from the received signal, a new estimation can be made that is less disturbed by interference. Therefore, the interference

cancellation is applied iteratively, assuming that each iteration leads to improvement up to perfect estimation of interference. In other words, convergence of the applied interference cancellation technique is reached. In this case only the channel noise disturbs the detection. The so-called single-user bound (SUB) is obtained that describes the case in which only one user is present, meaning that no multi-user interference occurs. To ensure that the best possible performance is always reached, the convergence of the interference cancellation technique will always be demanded in the following.
Parallel interference cancellation as well as successive interference cancellation can be used to estimate the multi-user interference. Using PIC, the users are processed in parallel. In contrast, using SIC an initial estimation and subtraction of the interference is done for the first user before considering the second user and so on. After all users have been processed successively, the second iteration starts, again beginning with the first user and continuing with the following users. An advantage of SIC is that due to the improved estimation of interference introduced by the first user the estimation of the second user will already be better than in the case of PIC. Generally, SIC should be preferred because of its faster convergence behavior but due to the easier analysis of PIC, it is considered here also.
The effective influence of the remaining interference for each iteration can be described using the multi-user efficiency η [Wei05]. For PIC it is given by

$$\eta = \frac{SINR}{SNR} = \frac{2\sigma_s^2/(\sigma_n^2 + \sigma_{\mathrm{MUI}}^2)}{2\sigma_s^2/\sigma_n^2} = \frac{1}{1 + \beta\mu P/\sigma_n^2} \tag{4.8}$$

where SINR is the signal-to-interference-plus-noise-ratio and SNR is the signal-to-noise-ratio. The variables σ_s^2, σ_n^2 and σ_{MUI}^2 are the variance of the desired signal, the variance of the noise and the variance of the remaining multi-user interference after cancellation, respectively. While σ_s^2 and σ_n^2 are constant, σ_{MUI}^2 changes in each iteration. The system load is denoted by β where $\beta = D/N$, D is the number of users and N is the so-called spreading length that gives the length of the codes [Ver98]. μ is the remaining mean squared error of the estimated symbols after channel decoding (channel coding is a standard technique that is used for improving the bit error rate and that is used in nearly every communication system). P is the power that the base station receives from each user. For more detailed information about CDMA see [Ver98].
The multi-user efficiency $\eta^{(m)}$ in the m-th iteration is dependent on the remaining interference which itself is dependent on the quality of the interference estimation in the last iteration. These dependencies can be integrated into one formula:

$$\eta^{(m)} = \psi(\eta^{(m-1)}). \tag{4.9}$$

The function ψ specifies the behavior of the interference cancellation and channel decoding for all iterations. The multi-user efficiency in a particular iteration can be described only by the efficiency in the previous iteration and some system-dependent constants (β, P, σ_n^2). Therefore, ψ can be determined for all iterations if these constants are known. Because the effect of the considered channel coding cannot be described analytically, the function ψ has to be calculated once in advance (for this work it was saved in a file that is read by the optimization tool, and intermediate values are interpolated).
Perfect interference cancellation is obtained for $\eta = 1$, corresponding to the single-user bound. To reach this point, the condition $\eta^{(m)} > \eta^{(m-1)}$ must hold for all possible values

of η. This demand can also be written as

$$\psi(\eta) > \eta \quad \forall\, \eta \in [0, 1). \tag{4.10}$$

If Equation 4.10 is not fulfilled, the detection gets stuck and convergence cannot be reached. This may result in a very high bit error rate, possibly making communication impossible for all users.
Convergence of PIC can be illustrated graphically if $\eta^{(m)} = \psi(\eta^{(m-1)})$ is plotted, and it is checked whether this function crosses the bisecting line. An example is shown in Figure 4.3. The transfer function was determined for an exemplary system. It can be seen that the interference cancellation will be successful because the transfer function is always above the bisecting line. Additionally, in Figure 4.3 the trajectory is given that shows the behavior for the individual iterations. Each step corresponds to one iteration: When going from one iteration (m) to the next $(m+1)$, the value of the y-axis $(\eta^{(m)})$ is projected to the x-axis $(\eta^{(m-1)})$ which is shown by the horizontal line. The new value for the y-axis is determined by moving vertically up until the transfer function is reached.

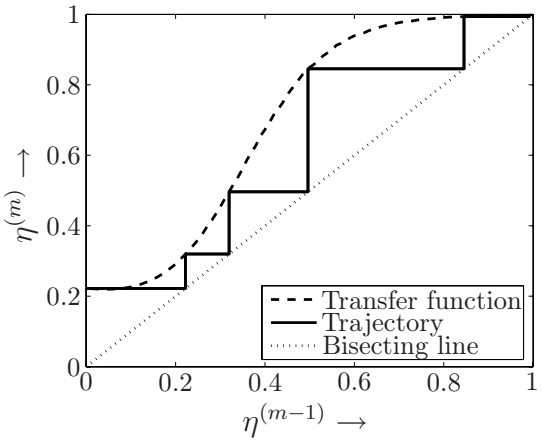


Figure 4.3: Transfer function and trajectory

The analysis of the SIC is more complicated. The error variance μ is not the same for all users as it was the case for PIC. Instead, all D users have different variances $\mu_d^{(m)}$ at each iteration m. Because the remaining errors of the users are independent, the variances can be added for the interference calculation. Therefore, the multi-user efficiency for user d in iteration m is given by

$$\eta_d^{(m)} = \frac{1}{1 + \frac{1}{N}\left(\sum_{i=1}^{d-1} \mu_i^{(m)} + \sum_{i=d+1}^{D} \mu_i^{(m-1)}\right)\frac{P}{\sigma_n^2}}. \tag{4.11}$$

As can be seen in Equation 4.11, the function for the multi-user efficiency is no longer dependent on only one parameter μ as in Equation 4.8. Instead, there is an individual influence from each user.
Due to this multidimensional dependencies, a simple two-dimensional plot of the transfer function as for the PIC is no longer possible because the transfer function would change after every iteration and for every user. Therefore, the trajectory has to be considered for an analysis of the convergence behavior of SIC. The trajectory describes the behavior of

the system at discrete points of interest which correspond to the iterations. For PIC the trajectory can be abstracted to the transfer function because of the parallel processing (see Figure 4.3). Because for SIC the trajectory must be considered for checking convergence of the interference cancellation method instead of the transfer function, the corresponding constraint of the optimization problem must be formulated differently. A more general condition for convergence of an iterative detection scheme is used: The single-user performance, corresponding to a multi-user efficiency of $\eta = 1$, must be reached by the trajectory within a limited number of iterations. In this work the condition above is relaxed to avoid numerical instabilities by allowing a small deviation ε from $\eta = 1$ because the lost concerning the bit error rate is negligible in practical applications.

Optimization of power allocation Equal power levels for all users have been assumed so far. To improve the performance, different power levels $P(d)$ can be assigned to the users d ($1 \leq d \leq D$). As a consequence, μ_d also becomes different for the users. In that case, Equation 4.8 for PIC and Equation 4.11 for SIC can be rewritten as follows:

$$\eta = \frac{1}{1 + \frac{1}{N} \sum_{d=1}^{d=D} \mu_d \cdot \frac{P(d)}{\sigma_n^2}} \tag{4.12}$$

$$\eta_d^{(m)} = \frac{1}{1 + \frac{1}{N} \left(\sum_{i=1}^{d-1} \mu_i^{(m)} P(i) + \sum_{i=d+1}^{D} \mu_i^{(m-1)} P(i) \right) \frac{1}{\sigma_n^2}}. \tag{4.13}$$

It can be seen that η depends on the distribution of the power levels. Consequently, the convergence of the interference cancellation can be influenced by changing the power distribution. If unequal powers are used, the stronger users improve more than the weaker users degrade because of their higher or lower power, respectively. This behavior is caused by the nonlinear characteristics of the system. The stronger users can be estimated more reliably due to their increased power and furthermore due to the decreased interference of the other users. As a consequence, the interference of the stronger users can be cancelled more reliably, thus the weaker users will improve also.

As in all wireless communication systems, it is desirable to use as little transmit power as possible to maximize battery life and minimize radiation exposure. The result is an optimization problem where the objective function is the sum of powers of all users, and the parameters are the powers of the individual users. Because convergence to the SUB is demanded, it has to be incorporated into the optimization as a constraint. Details of the systems are described in [Wei05].

The interference cancellation occurs in the base station. The base station must also know the results of the optimization of the power allocation scheme, but the actual optimization process may theoretically happen somewhere else and it can also be done offline. The reason is that not the transmit power but the received power is optimized. This may be done independent from the actual distribution of the users but only dependent on system parameters like the number of users and the variance of the noise. The optimization may be executed for different scenarios and the results are saved. Based on the actual situation, the base station can afterwards calculate the necessary transmit powers and communicate this information to the individual users.

Apart from the constraints that result from the interference cancellation methods, boundary constraints have to be regarded during the optimization process. Because no negative power exists, the design variables have to be non-negative. In preliminary tests sometimes a trivial solution has been found using PIC for which all powers are zero except for one user. Multi-user interference is not present in this case but the solution has no practical meaning. It is prevented in the following by setting a lower limit $x_{min,d} > 0$ for the parameters (with $x_{min,d} \leq P(d) \leq x_{max,d}$ as well as $x_{min,j} = x_{min,k} = x_{min}$ and $x_{max,j} = x_{max,k} = x_{max}$ for all $j, k \in \{1, \ldots, D\}$). There is no rule for choosing the exact value but it must be ensured that the lower bound is not present in the final solution because this would mean that a better solution may exist. In this case the lower limit was set to $x_{min,PIC} = 0.5$ for PIC[1]. For SIC also problems appeared in preliminary tests using $x_{min} = 0$ as some solutions have been found in which one user got almost no power. This is avoided in the same way as for PIC in the following by setting a lower limit $x_{min,SIC} = 0.5$.

To limit the search space, i.e. to decrease the computational cost, it is also reasonable to set an upper boundary. In principle the upper limit can be chosen arbitrarily but with the same restriction as for the lower bound: The upper bound should not appear in the final solution. If it does appear in the final solution, then the upper bound should be increased. In the following $x_{max} = 4$ is used for both PIC and SIC. In Section 4.2.4.3 it will be discussed that the ratio of feasible space to the whole search space can be altered for PIC by varying the upper bound. Thus, for the evaluation of two constraint-handling techniques the upper bound will be varied in $2 \leq x_{max} \leq 4$.

In preliminary tests it could be seen that permutations of solutions appeared for PIC. Because of the parallel processing the permutations have no influence on the results. Hence, computational cost can be saved by rearranging the users in ascending order. For SIC this method cannot be applied because the users are processed in a predetermined order, so the convergence behavior of the SIC is dependent on the order of the users.

In summary, the optimization problem can be formulated as follows: The sum of powers should be minimized using the powers of the individual users as parameters where the powers $P(d)$ are arranged in a vector $\vec{x} = (P(1), ..., P(D))$. This problem is solved here for $D = 16$ users. A feasible solution must fulfill the condition for convergence of the interference cancellation methods, and furthermore no negative powers are allowed. Mathematically, the optimization problem using PIC is the following:

$$\begin{aligned} \min_{\vec{x}} f(\vec{x}) \quad &= \quad \min_{P(1),..,P(D)} \quad \sum_{d=1}^{d=D} P(d) \\ &\text{s. t.} \quad \begin{cases} \psi(\eta) > \eta \quad \forall\, \eta \in [0,1) \\ P(d) \geq x_{min,d} > 0 \quad \forall\, d. \end{cases} \end{aligned} \tag{4.14}$$

For SIC the objective function is the same as for PIC but the constraints differ. The

[1]Due to normalization, the powers are given without unit here. Not the absolute value is important but only the ratio of power over σ_n^2 which is fixed here to $\sigma_n^2 = 1$ for simplicity.

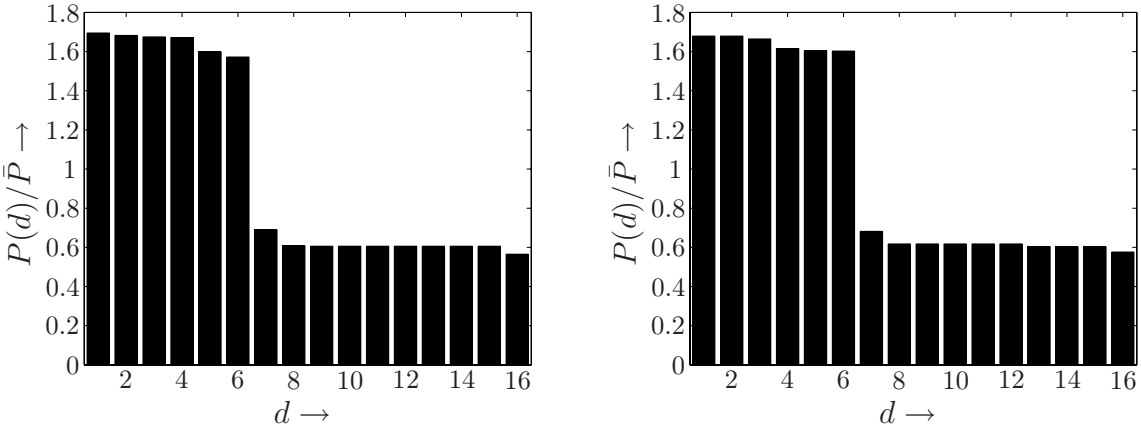
optimization problem can be described as follows:

$$\begin{aligned} \min_{\vec{x}} f(\vec{x}) \quad &= \quad \min_{P(1),..,P(D)} \quad \sum_{d=1}^{d=D} P(d) \\ &\text{s. t.} \quad \begin{cases} \eta_d^{(m_{max})} = 1 - \varepsilon, m_{max} < \infty \quad \forall\, d \\ P(d) \geq x_{min,d} > 0 \quad \forall\, d \end{cases} \end{aligned} \tag{4.15}$$

where the allowed deviation from $\eta = 1$ is set to $\varepsilon = 0.001$. m_{max} is the maximum number of iterations that must be below infinity to ensure convergence of the interference cancellation method in finite time. In this work $m_{max} = 20$ is used because in preliminary examinations it has been shown that more iterations lead to slightly reduced power but considerably higher computational effort (see also [Wei06]).

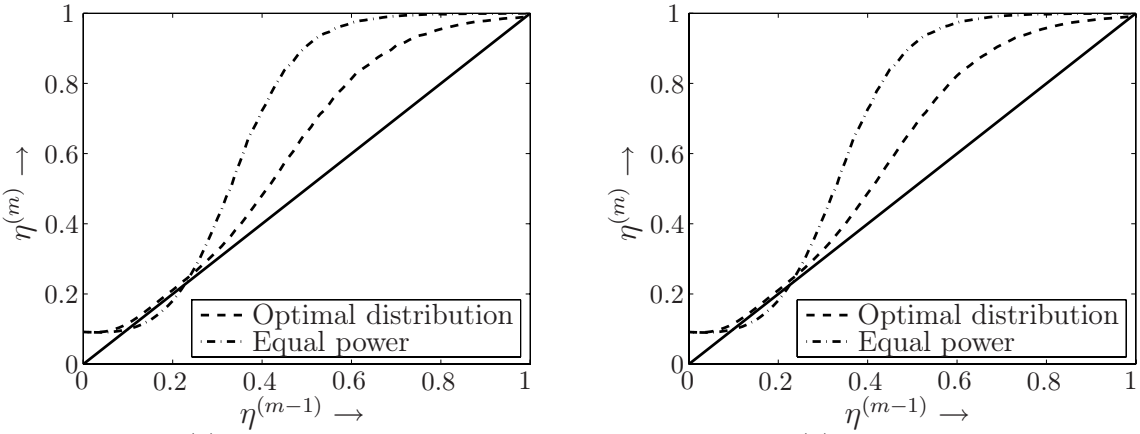
4.2.4.2 Solutions for Parallel Interference Cancellation and Successive Interference Cancellation

Results for the power allocation problem using parallel interference cancellation and successive interference cancellation will be shown in the following. In this section the results from the communications point of view are shown and also some results from the optimization point of view will be discussed. A detailed examination of two constraint-handling techniques will be shown in Section 4.2.4.3.
To obtain the best possible results (and also to examine the influence of control parameters on DE and PSO), parameter studies have been conducted for the PIC optimization problem within the scope of this thesis that have been published in [Zie06f] for DE and in [Zie09] for PSO. It was found out that the parameter settings $F = 0.7$, $CR = 0.9$ and $NP = 30$ worked best for DE whereas $w = 0.6$, $c_1 = 0.2$, $c_2 = 1.6$, the *von-Neumann* neighborhood topology and $40 \leq NP \leq 100$ gave the best results for PSO (for refreshing the random numbers according to Equation 3.8 as well as according to Equation 3.10).
The best objective function value that was found during the extensive parameter study for PSO was $f(\vec{x}) = 18.39$. The resulting power profile is shown in Figure 4.4(a) where the power is normalized due to comparability reasons. The absolute value can be calculated by multiplication with $\bar{P}$ which is the mean power over all users. The corresponding transfer function is given in Figure 4.5(a). It can be seen that the transfer function of the optimized distribution is always above the bisecting line, thus the interference cancellation technique converged for the optimized results. If an equal power distribution with the same overall power and the same number of users is used, it can be seen in Figure 4.5 that the bisecting line is crossed, meaning that the detection gets stuck.
Because in [Zie06f] a lower number of function evaluations was allowed for DE than in [Zie09] for PSO ($40,000$ instead of $80,000$ function evaluations), not the results from [Zie06f] are shown here for DE but new simulations have been done. These simulations were conducted with the mentioned best settings and the same number of function evaluations as for PSO in [Zie09] to ensure easy comparability. The best result out of 100 runs was $f(\vec{x}) = 18.43$. The corresponding power profile and the transfer function are shown in Figure 4.4(b) and 4.5(b), respectively. Apart from small deviations, the resulting power profile is very similar to the results of PSO. The same holds for the transfer function. Again, the detection would get stuck for an equal power distribution.



(a) Results for PSO (b) Results for DE

Figure 4.4: Optimized power profile for PIC with $D = 16$ ($\beta = 4$)

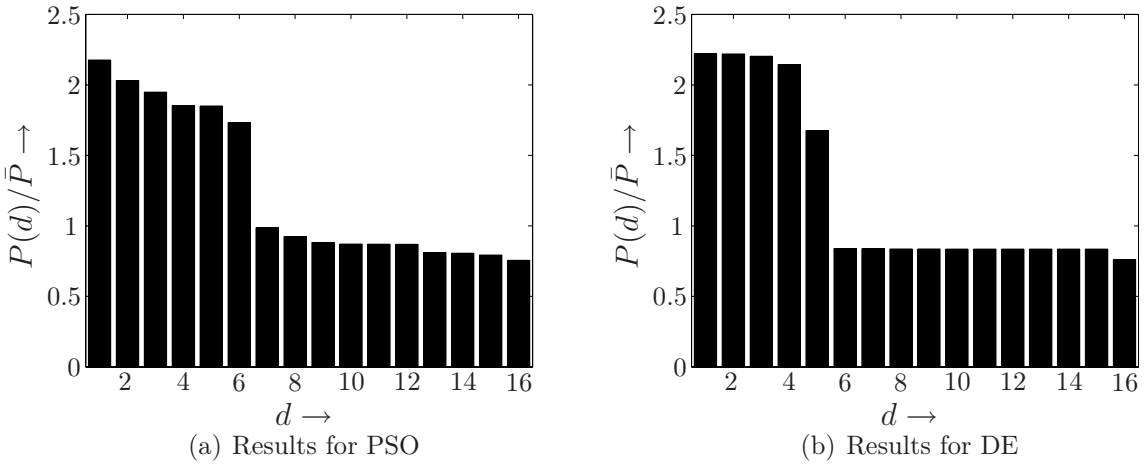


(a) Results for PSO (b) Results for DE

Figure 4.5: Transfer characteristic of PIC for equal and optimized power profile, $D = 16$

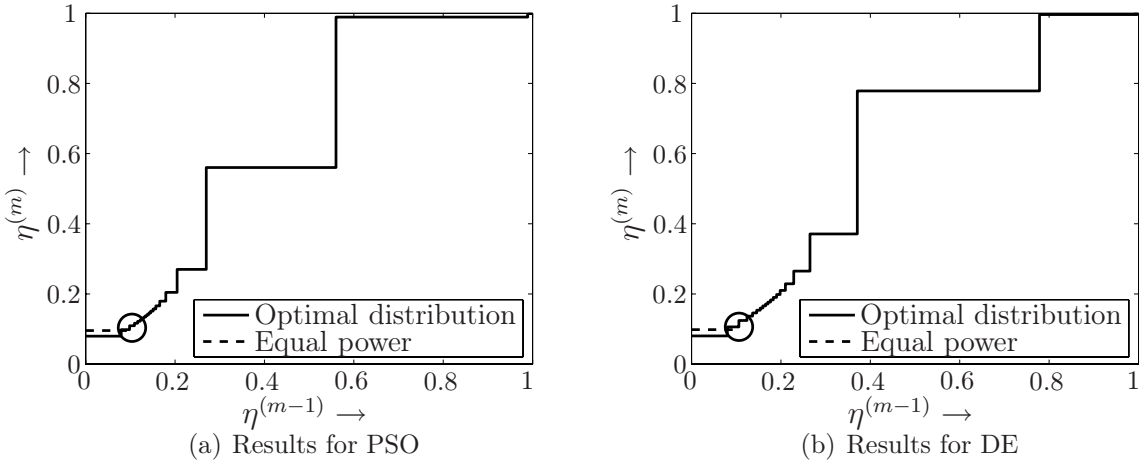
For the PIC optimization problem, the objective function value reached by PSO is slightly better than the result of DE ($f(\vec{x}) = 18.39$ vs. $f(\vec{x}) = 18.43$) although from the communications point of view the difference in negligible. A reason might be the different behavior in constrained search spaces that has already been described in Section 4.2.3: The PSO particles are able to become infeasible at any time whereas the DE individuals must stay feasible if they have been feasible once, making the crossing of infeasible regions more difficult.

Results for SIC using $x_{min} = 0$ have already been shown for DE in [Zie06f]. The applicability of PSO for SIC has been tested later, and it was discovered that using PSO some solutions contain a user that is not given any power, so a lower limit $x_{min} > 0$ had to be introduced (see also Section 4.2.4.1). Because here results for DE and PSO should be shown that were generated using the same conditions, the optimization runs using DE have been repeated with $x_{min} = 0.5$. The best objective function value was $f(\vec{x}) = 19.58$ for which the power profile is shown in Figure 4.6(b) and the trajectory is given in Figure 4.7(b). For PSO a corresponding examination resulted in the best value



(a) Results for PSO (b) Results for DE

Figure 4.6: Optimized power profile for SIC with $D = 16$ ($\beta = 4$)



(a) Results for PSO (b) Results for DE

Figure 4.7: Trajectory of SIC for equal and optimized power profile, $D = 16$

of $f(\vec{x}) = 20.16$ for which the power profile and the trajectory are shown in Figure 4.6(a) and Figure 4.7(a), respectively. The resulting power profiles differ in some details, e.g. in the power profile optimized with DE five users have powers which are above average whereas this holds for six users using the power profile optimized with PSO. This finding suggests that the SIC optimization problem has a complex landscape that makes optimization difficult. As it could be expected from the different power profiles, the behavior of the trajectories for the optimized results from DE and PSO is also different. However, both trajectories reach $\eta \approx 1$ in a limited number of iterations, thus the interference cancellation is successful in both cases. Similarly to PIC, the detection gets stuck if an equal power distribution with the same overall power and the same number of users is used (marked by the circles in Figure 4.7).

For both DE and PSO the best parameter settings from [Zie06f] and [Zie09] have been used for this examination, the population size has been set to $NP = 60$ for PSO, and 100 runs with a maximum number of function evaluations $FE_{max} = 10^6$ have been conducted, respectively. This procedure is implicitly based on the assumption that the structures of

the PIC and the SIC optimization problem are similar, so that good results can be reached using the same parameter settings. This assumption is not necessarily valid. Although the background of the problems is similar and the objective function is calculated in the same way, the constraints are different and may lead to different landscapes. Thus, results from parameter studies are not necessarily transferable. Actually, the results for PSO are worse than for DE whereas for PIC the opposite relation was found. It is assumed that this result is caused by different problem structures, hence even better results for SIC may be obtained with different parameter settings. Because conducting an exhaustive parameter study means high computational cost and the focus of this work was mainly on showing the applicability of DE and PSO for these optimization problems (and on the examination of constraint-handling techniques as discussed in Section 4.2.4.3), this is not examined further here.
For both optimization algorithms it is shown that the power profiles optimized by DE and PSO have a considerably better performance than an equal power distribution. Due to the application of interference cancellation methods the load for current UMTS systems could be increased from $\beta \approx 1$ to $\beta \approx 3$ while still reaching the SUB, corresponding to an increase from 4 to 12 users for the considered example. If the optimized power profile is employed, it was shown that convergence of the interference cancellation techniques can be achieved for a load of $\beta = 4$ which would not be possible for an equal power distribution. Therefore, in this case the number of users was increased from 12 to 16 due to the optimized power profile.

4.2.4.3 Comparison of the Death Penalty and the Modified Replacement Method

There are several characteristics that may make a constrained optimization problem difficult to optimize [Mic96, Mez05], e.g.

- a high dimensionality,
- a high number of inequality constraints,
- a high number of equality constraints,
- a high number of active constraints at the optimum,
- the type of objective function (linear, quadratic, cubic, polynomial, nonlinear),
- a disconnected feasible region and
- the size of the feasible space with respect to the whole search space.

In [Mez04] it is stated that the death penalty is only recommended for problems with a convex search space and furthermore feasible regions with considerable size with respect to the whole search space. Moreover, in [Mic96] it is noted that the performance of the death penalty is not as robust as other constraint-handling techniques, i.e. the standard deviation of solutions is high. This result has been achieved using different implementations of EAs in [Mic96] but neither DE nor PSO has been among them. Consequently, it is not clear if this result is transferable to these algorithms. DE is similar to ESs [Mad02, Xue05] and the death penalty is a constraint-handling technique that is easy to

implement and that is very popular in the ES community [Coe02a]. Therefore, the latter two aspects (applicability for problems with different ratio of the feasible region to the whole search space as well as the standard deviation of solutions) will be examined and compared with the modified replacement method here. Results will also be compared with PSO.
The PIC optimization problem can be used as basis for this examination because in [Zie09] it is shown that the ratio of feasible space to the whole search space varies considerably for different settings of the upper bound of the search space x_{max}. For this purpose the ratio was estimated by using the metric ρ which is defined in [Mic96] and [Koz99] as

$$\rho = \frac{|\mathcal{F}|}{|\mathcal{S}|} \tag{4.16}$$

where $|\mathcal{F}|$ is the size of the feasible space and $|\mathcal{S}|$ is the size of the search space. To estimate $|\mathcal{F}|$ and $|\mathcal{S}|$, 1,000,000 points are randomly generated in the search space, and it is determined whether they are feasible or not. In Table 4.1 the results for the PIC optimization problem are shown. It can be seen that the ratio of feasible space to search space is below 0.1% for $x_{max} = 2$ and increases to above 90% for $x_{max} = 4$. Hence, the constraint-handling techniques can be tested for various ratios of feasible space to the whole search space if x_{max} is varied. Naturally, the size of the search space also changes when x_{max} is varied but the effect is assumed to be secondary. This is also confirmed by the results which will be presented in this section: The largest deterioration can be seen for small x_{max} whereas the effect of the increasing search space is visible sometimes but less pronounced. In the following results are examined for $x_{max} \in \{2, 2.5, 3, 3.5, 4\}$ while also varying the population size from 20 to 100 in steps of 10. The same control parameter settings are used as given in Section 4.2.4.2.

Table 4.1: Ratio of feasible space to search space ρ

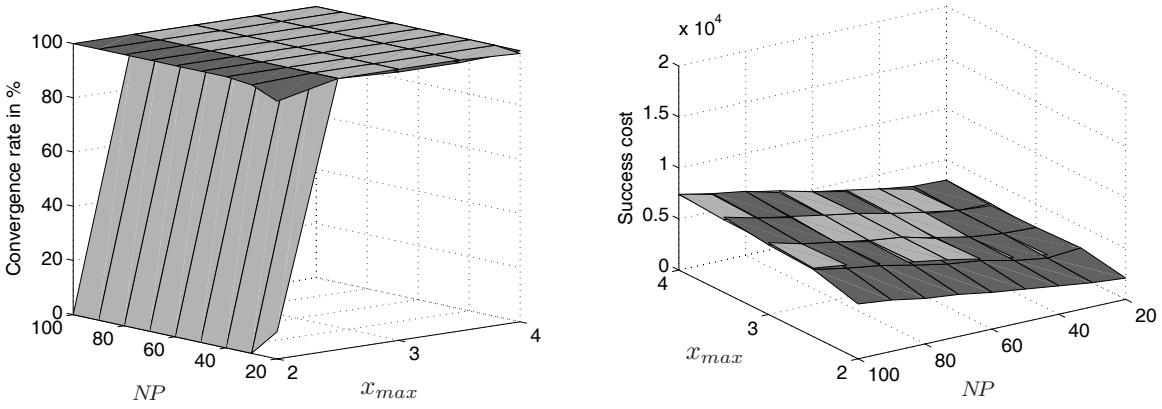
x_{max}	ρ
2	0.000175
2.5	0.18155
3	0.588716
3.5	0.816328
4	0.91698

For the comparison of the constraint-handling methods the following approach is followed: In preliminary tests the objective function value after a certain number of generations was recorded. Based on these results, convergence for the PIC optimization problem was defined as reaching an objective function value of $f(\vec{x}) \leq 18.5$ in [Zie06f] and [Zie09]. Additionally, in preliminary tests a suitable setting for the maximum number of function evaluations FE_{max} has been established. In this case the term "number of function evaluations" refers to evaluations of the constraint function because it is calculated once for every individual in each generation while the objective function is only evaluated for feasible individuals. In the following $FE_{max} = 80,000$ is used, and it is documented in how many runs the global optimum is found and how many function evaluations are necessary

to reach the optimum for the first time. Therefore, statements about the convergence probability and convergence speed can be derived.
The average number of function evaluations for convergence $FE_{conv,av}$ may be misleading as performance measure for the convergence speed because it does not consider the number of runs in which the algorithms do not successfully converge to the optimum. Instead, the average number of function evaluations for convergence $FE_{conv,av}$ is weighted with the total number of runs t divided by the number of successful (converged) runs s:

$$\text{success cost} = \frac{FE_{conv,av} \cdot t}{s} \tag{4.17}$$

where the total number of runs is $t = 100$ for every parameter combination here, and a run is successful if one individual reaches an objective function value of $f(\vec{x}) \leq 18.5$. A smaller value of the success cost indicates better performance. The success cost is also named success performance, e.g. in [Lia06a, Pri06]. Because the term "performance" might indicate that a higher value is better, the performance measure was renamed to "success cost" here. In the following all figures showing the success cost will be scaled to $20,000$ for comparability reasons.
Furthermore, it is examined if the statement from [Mic96] that the standard deviation of solutions is high for the death penalty also holds for DE and PSO. For this purpose, the average objective function value as well as the standard deviation of objective function values after $FE_{max} = 80,000$ function evaluations will also be shown in the following.
In Figure 4.8(a) the convergence rate of the death penalty (light gray) and the modified replacement method (dark gray) is given in dependence on the population size NP and the upper bound x_{max} for DE. It can be seen that the performance of the death penalty and the modified replacement method is mostly similar. Only for the smallest ratio of feasible space to the whole search space ($x_{max} = 2$) the performance of the death penalty deteriorates significantly. In contrast, the performance of the modified replacement method stays constant, except for a slightly worse convergence rate for a small number of individuals. For small population sizes the optimum is still found in some optimization runs using the death penalty but for larger population sizes not enough feasible positions are found to initialize the population.



(a) Convergence rate

(b) Success cost

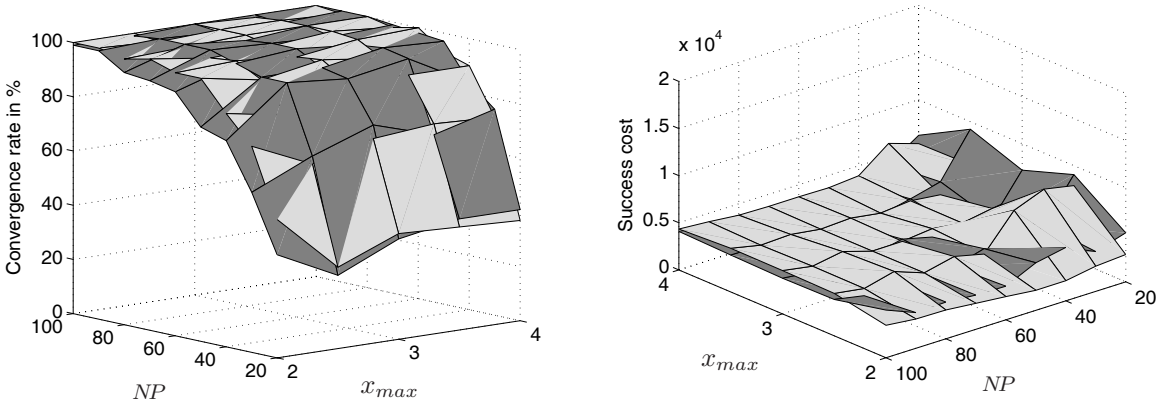
Figure 4.8: Results for DE (light gray: death penalty; dark gray: modified replacement)

Except for $x_{max} = 2$, the success cost is similar for the death penalty and the modified replacement procedure (see Figure 4.8(b)). The success cost increases with growing population size NP which is a common result for EAs (usually, convergence becomes faster with decreasing population size but also premature convergence happens more often due to decreased diversity). The success cost also increases slightly with growing x_{max} which might be caused by the increasing search space.
In [Zie09] the same examination has been done for Particle Swarm Optimization. The results were very different concerning the constraint-handling techniques: If the random numbers were refreshed only once for each particle, not a single optimization run converged to the optimum using the death penalty. If the random numbers were recalculated once for every component of the velocity, not more than 2 converged runs could be found for any parameter combination (see Table 4.2). In contrast, the modified replacement method yielded the results given in Figure 4.9(a). No large variations in performance can be noticed with varying x_{max}. Only for $x_{max} = 2$ (smallest ratio of feasible space to search space) the performance is slightly worse. For large population sizes the convergence rate reaches approximately 100% but the performance decreases for $NP = 50$ until for $NP = 20$ the convergence rate is about $30 - 50\%$.

Table 4.2: Convergence rate (in %) for PSO using the death penalty with random numbers refreshed for every component of the velocity

	x_{max}				
NP	2	2.5	3	3.5	4
20	0	1	1	1	0
30	0	0	0	0	0
40	0	1	0	0	0
50	0	0	0	0	0
60	0	0	1	0	0
70	0	1	0	0	0
80	0	2	0	0	0
90	0	1	1	0	1
100	0	2	0	0	1

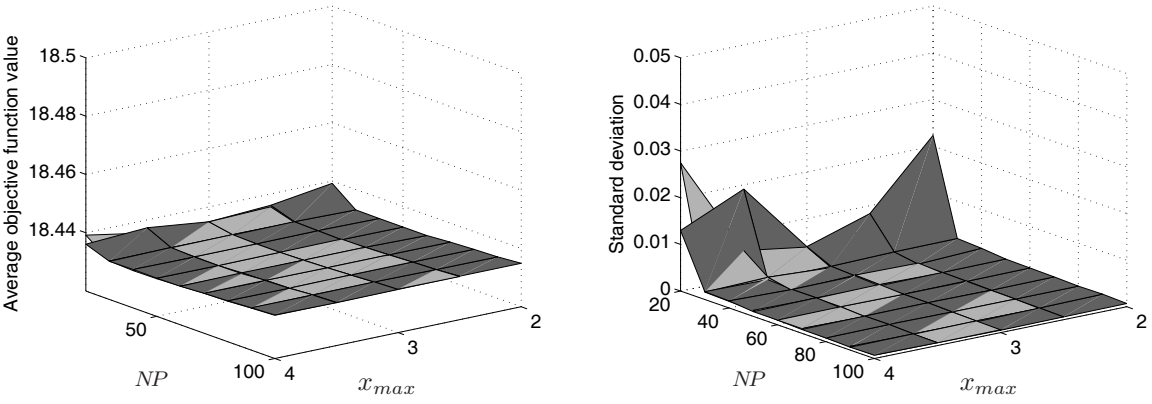
The success cost is only given for the modified replacement method in Figure 4.9(b) because the death penalty always resulted in a success cost above $20,000$ due to its small convergence rate. Again, the performance is very similar for both ways of refreshing the random numbers, and furthermore it is independent from x_{max}. For $NP \geq 50$ the success cost is approximately constant while for smaller NP the success cost rises due to the decreasing convergence rate.
In the following the robustness of the methods is examined by evaluating the results after a fixed number of function evaluations $FE_{max} = 80,000$. Apart from the standard deviation of objective function values, also the average objective function value after FE_{max} is given. Figure 4.10 shows that there are no large differences concerning the average and standard deviation of returned solutions between the death penalty and the modified replacement method for DE (of course with the difference that the death penalty does not work satisfactorily for $x_{max} = 2$ as described previously).



(a) Convergence rate (b) Success cost

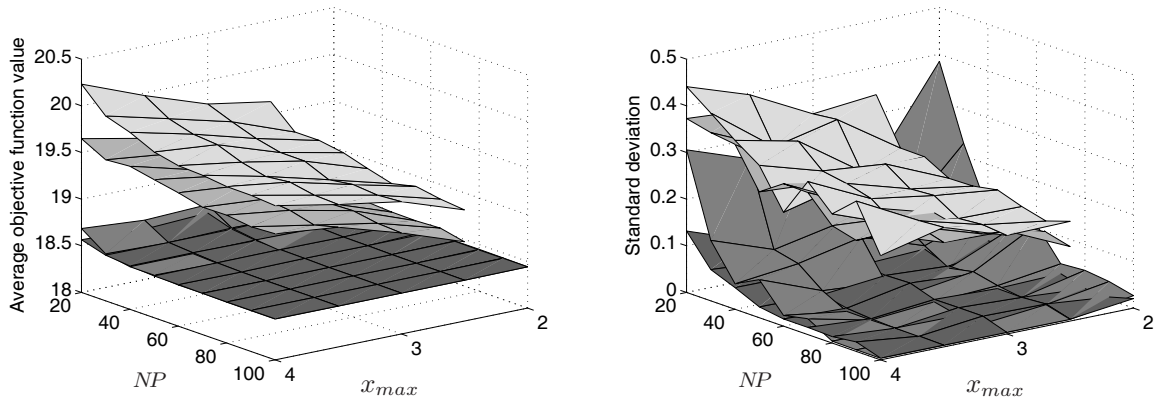
Figure 4.9: Results for PSO (light gray: random numbers refreshed for every component of the velocity; dark gray: random numbers recalculated only once for every particle)

The situation for PSO is very different as can be seen in Figure 4.11 where the following is shown (from dark gray to light gray): 1. random numbers refreshed for every component of the velocity with modified replacement method; 2. random numbers recalculated only once for every particle with modified replacement method; 3. random numbers refreshed for every component of the velocity with death penalty; 4. random numbers recalculated only once for every particle with death penalty. The average objective function value for the death penalty is considerably higher than for the modified replacement method for both ways of refreshing the random numbers. The standard deviation of objective function values is also increased for the death penalty, again for both ways of refreshing the random numbers. It should be noted that the figures for DE and PSO are scaled differently.



(a) Average objective function value (b) Standard deviation of objective function values

Figure 4.10: Solutions at the end of optimization runs for DE (light gray: death penalty; dark gray: modified replacement)



(a) Average objective function value

(b) Standard deviation of objective function values

Figure 4.11: Solutions at the end of optimization runs for PSO (the different colors are explained in the text)

A detailed examination of the data shows that the initialization of the whole population for the death penalty only fails for $x_{max} = 2$, especially for large NP (of course for both DE and PSO). Apart from that, the average as well as the standard deviation of objective function values are constant for different values of x_{max} for DE. The results only become worse for small population sizes ($NP \leq 20$), especially for increasing x_{max}. For PSO this behavior is much more pronounced: The performance with the death penalty generally deteriorates for increasing x_{max} and also for decreasing NP. The reasons are that a smaller population size makes it harder to sample the search space. The increased size of the search space also has the effect that scanning of the search space becomes more difficult. This also explains the random distribution of optimization runs which converged to the optimum for PSO using the death penalty in Table 4.2: PSO has trouble sampling the search space but purely by chance an individual might reach the vicinity of the global optimum, so the optimum is found. Apparently the increased diversity of the approach with random numbers refreshed for every component of the velocity helps reaching the optimum in that case because at least some optimization runs converged. On contrast, the approach with recalculating the random numbers only once for each particle did not converge in a single run.
The standard deviation of both DE and PSO with the modified replacement method increases clearly visible when the population size becomes small. This indicates that it becomes harder to sample the search space, so the performance of the algorithms becomes more dependent on initial positions. However, the results are independent from x_{max}.

Summary The results for the PIC optimization problem using DE and PSO confirm the statement from [Mez04] that the death penalty does not work well if the ratio ρ of feasible space to the whole search space becomes very small. However, for moderate and larger values of ρ the death penalty yields good results when applied to DE. For PSO the performance using the death penalty is always bad. These results show that constraint-handling methods may result in different behavior when applied to different optimization algorithms. The behavior when using the modified replacement method is also different for DE and PSO: The convergence rate is approximately constant for population sizes

of $NP \in \{20, 30, \ldots 100\}$ for DE whereas for PSO a considerable dependence on the population size can be seen.

The statement from [Mic96] that the performance of the death penalty is not robust could not be confirmed for DE. The standard deviation of solutions only increased for small population sizes, and this also happened using the modified replacement method. In contrast, for PSO the death penalty yielded considerably worse results concerning the average as well as the standard deviation of objective function values.

Especially for PSO the modified replacement method clearly yielded superior results when compared to the death penalty. As a consequence, the modified replacement method should be preferred. The implementation of the modified replacement method takes more effort than the death penalty because the constraint violation has to be calculated for infeasible individuals but it is still rather easy to implement. Furthermore, it contains no additional parameters and the information about the amount of constraint violation can be successfully used to guide individuals towards feasible regions.

4.3 Equality Constraints

Despite its simplicity, the modified replacement method based on the feasibility rules from [Deb00] described in the previous section is successful for a large range of optimization problems using DE or PSO, especially if only inequality constraints are present [Zie06a, Zie06b]. Optimization problems containing equality constraints are harder to optimize because the feasible space becomes very small, particularly when the remaining constraint violation must be very small [Xie04a]. Often the population concentrates too quickly on one part of the search space that leads to decreased constraint violation and cannot generate enough diversity to search for a better objective function value once the feasible region has been reached [Zie07e, Sto99b]. In the literature several methods are described to handle equality constraints by varying the allowed constraint violation during the optimization run [Tak06, Ham02]. Thus, an optimization run is started with a relatively large ϵ_e that is gradually refined until the desired constraint violation is reached. That way, there will be feasible individuals from the beginning of the optimization run, so the objective function value can be considered earlier. Of course this approach is only possible if the objective function can be evaluated for infeasible individuals. This may not always be the case, especially in real-world problems.

Different approaches exist in the literature that differ in the way ϵ_e is adjusted. Some of them will be summarized in the following. However, in the literature usually no comparisons are done which show how much improvement can be obtained by using these methods. Therefore, it will be shown here for an optimization problem from yield analysis that varying ϵ_e during the optimization run can help to improve results significantly.

4.3.1 Related Literature

In [Tak06] the individuals are sorted according to their constraint violation after initialization of the population, and $\epsilon_e(0)$ is set to the largest constraint violation $\epsilon_{e,0}$ that occurs in the best 20% of the individuals. In G_c generations the allowed constraint violation is

decreased to the desired value $\epsilon_{e,final}$:

$$\epsilon_e(G) = \begin{cases} \epsilon_{e,final} + (\epsilon_{e,0} - \epsilon_{e,final}) \cdot \left(1 - \frac{G}{G_c}\right)^{cp} & 0 < G < G_c \\ \epsilon_{e,final} & G \geq G_c. \end{cases} \tag{4.18}$$

Parameter G_c determines the time for reaching the final value of ϵ_e. Parameter cp influences the development between $G = 0$ and $G = G_c$ (see Figure 4.12 where the allowed constraint violation is decreased from $\epsilon_{e,0} = 10^{-2}$ to $\epsilon_{e,final} = 10^{-3}$ in $G_c = 1000$ generations): ϵ_e decreases linearly for $cp = 1$ whereas for larger cp the allowed constraint violation ϵ_e changes more rapidly in early generations while in later generations ϵ_e decreases less pronounced. In [Tak06] $\epsilon_{e,final}$ is given as 0 but because of numerical problems it might be better to allow a small deviation $\epsilon_{e,final} > 0$. The other parameters are set to $cp = 5$ and $G_c = 0.2 \cdot G_{max} = 2500$ in [Tak06] (where $G_{max} = 12500$ is the maximum number of generations, corresponding to a maximum number of $FE_{max} = 500,000$ function evaluations with $NP = 40$), and a DE algorithm is employed with $F = 0.7$ and $CR = 0.9$.

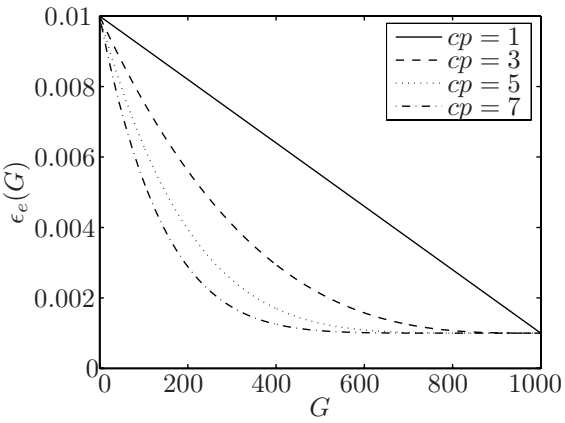


Figure 4.12: Development of $\epsilon_e(G)$ for different cp

In [Ham02] a method called ASCHEA (Adaptive Segregational Constraint Handling Evolutionary Algorithm) is described. ASCHEA contains several interesting features but as the focus is on equality constraints here, only the handling of equality constraints is regarded in the following. Two methods are proposed and compared in [Ham02]: Dynamic adjustment and adaptive adjustment of ϵ_e. In contrast to [Tak06], different values of ϵ_e are used for each equality constraint. In the dynamic adjustment, the ϵ_e value is reduced by dividing it by a user-defined parameter f_{ϵ_e} every time the feasible percentage of the population is larger than another user-defined parameter τ_r. Therefore, two new parameters are introduced. In contrast to [Tak06], the desired accuracy is not pre-defined but the ϵ_e value changes according to the state of the population regarding feasibility. Because the degree of feasibility after modifying ϵ_e is not considered, there might be no single feasible individual after modification of ϵ_e. In contrast, in the adaptive adjustment it is ensured that a certain percentage of the population τ_e still fulfills constraint k after adjusting $\epsilon_{e,k}$. Again, ϵ_e is adjusted if the percentage of feasible individuals τ_r excesses a certain value that is user-defined. Besides, Equation 2.7 is modified, so that there are two individual parameters for the violation of the equality constraint in positive and negative direction where $\epsilon^-_{e,k}(G) \leq h_k(G) \leq \epsilon^+_{e,k}(G)$. In [Ham02] parameters are set as follows: For

the dynamic adjustment $f_{\epsilon_e} = 1.01$ and $\tau_r = 0.6$; for the adaptive adjustment $\tau_r = 0.7$ and $\tau_e = 0.3$. It could not be determined which approach is superior. A problem arises in the presence of several equality constraints (and also if inequality constraints exist additionally). In that case the desired goal of having feasible individuals in the population might not be reached because solutions fulfilling a constraint might not fulfill another constraint.
In [Mez04] a dynamic adjustment similar to the one used in ASCHEA is employed in an ES. In one set of experiments ϵ_e is divided by 1.000001 (which is an empirically determined parameter setting) for each new generation and the initial value is $\epsilon_e(0) = 0.001$ whereas in another set of experiments $\epsilon_e(G+1) = \epsilon_e(G)/1.00195$ and the initial value is again $\epsilon_e(0) = 0.001$. For one optimization problem $\epsilon_e(0) = 3.0$ was used due to problems with generating feasible solutions during early stages of the optimization runs. The factor for decreasing ϵ_e was also changed to 1.0145, so that in the end a constraint violation of 0.00003 was reached. This already shows that it is important to employ suitable parameter settings if this approach is used. As in [Ham02], the constraint-handling mechanism presented in [Mez04] comprises much more than just the handling of equality constraints but other details should not be discussed further here.

4.3.2 Application: Worst Case Methods for Yield Analysis

In the following the motivation for applying worst-case methods for yield analysis in microsystems is given. The principle work flow of worst-case analysis is shortly described, with emphasis on the optimization problem that arises during the application of worst-case methods. This optimization problem is solved for an exemplary test case that considers an electrothermal actuator. Furthermore, the difference in performance between the modified replacement method based on the feasibility rules from [Deb00] and the (slightly adapted) method from [Tak06] that was specifically designed for handling equality constraints is shown.

4.3.2.1 Background

The parameter values of fabricated microsystems are often not equal to the ideal values which are defined in the design (also called nominal design parameters $\vec{x}_{nom}$). The reasons are process variations like inaccurate etching rate, lithographic aberrations and imaging imperfections [Vud07]. Because of the trend of decreasing feature sizes, this effect tends to become even more pronounced in the future due to the process complexities involved in the production. As a result, it is not necessarily sufficient to search for parameters that lead to optimal performance in the design phase. Additionally, robust performance is needed, so that the effect of variations of the design parameters occurring in production will be as small as possible. This can be achieved by applying yield analysis in the design phase by considering process statistics. The idea is to maximize the distance of the nominal design to the specification boundary to obtain the largest possible robustness.
The traditional approach for yield analysis is the use of Monte-Carlo methods. Many simulations of the considered design are carried out using not the nominal design parameters but choosing random values. These random values are based on an assumed probability distribution depending on the parameter variations which would occur due to process variations. The performance of the simulated systems is compared with the

specification value P_{spec}. The yield can be estimated by calculating the ratio of systems which would meet the desired performance to all simulated systems. In order to obtain a reliable estimation, a large number of designs have to be simulated.

Worst-case methods have considerable advantages over Monte Carlo methods because of the reduced number of required simulations and the availability of a yield metric [Vud07]. The accuracy of worst-case methods is slightly inferior when compared to Monte Carlo methods but the decreased simulation effort balances this disadvantage. This holds especially for designs with a large number of variables. For these designs the computational cost increases drastically for Monte Carlo methods whereas the increase in simulation effort is considerably less for worst-case methods.

For the application of worst-case analysis a model of the system performance in dependence on its parameters is needed. For yield analysis in microelectronics usually linear approximations are sufficient for building the model. The generally more complex nature of microsystems has to be reflected by also including higher-order terms. Methods from Design of Experiments offer a computationally economic way for building these models. Thereby, not only the main effects (linear approximations) may be considered but also e.g. interaction effects and quadratic terms [Vud07]. The specification boundary is defined by setting the function f_{perf} that is obtained via the Design of Experiments equal to the specification value. The worst-case parameter set has to be found which is the point on the specification boundary that is closest to the nominal point. The distance between the worst-case parameter set and the nominal point is the worst-case distance β that is used as a measure for representing yield. Assuming Gaussian parameter distributions, the yield is given in [Vud07] as:

$$\text{yield} = \frac{1}{2}\left(1 + \text{erf}\left(\frac{\beta}{\sqrt{2}}\right)\right) \cdot 100. \tag{4.19}$$

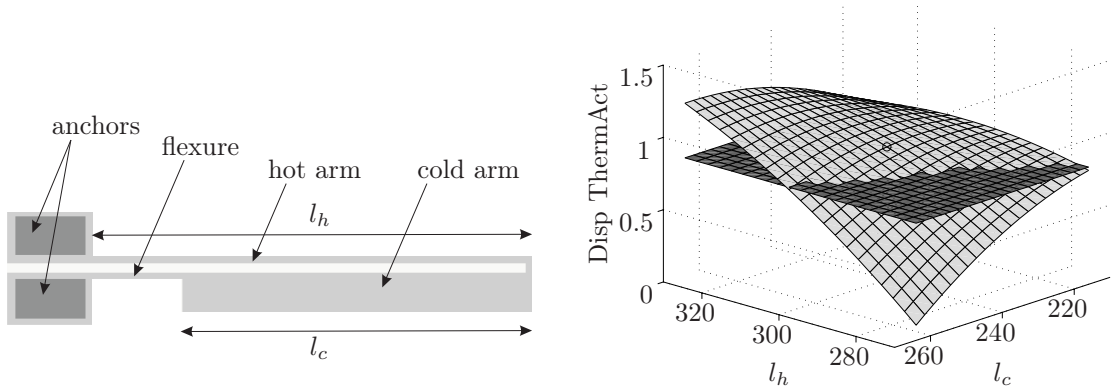
Finding the worst-case parameter set is an optimization problem. It can be formulated as minimizing the Euclidean norm between the nominal point $\vec{x}_{nom}$ and any point $\vec{x}$ in the design space, with the non-linear equality constraint that the performance value of $\vec{x}$ has to equal the specification value, i.e. the parameter set has to lie on the specification boundary. Furthermore, boundary constraints $\vec{x}_{min}$ and $\vec{x}_{max}$ are defined in dependence on the assumed maximum parametric variations. Therefore, the optimization problem can be stated as follows:

$$\begin{aligned} \min_{\vec{x}} f(\vec{x}) \quad &= \quad \min_{x_1,..,x_D} \quad \sqrt{\sum_{d=1}^{D}(x_d - x_{nom,d})^2} \\ &\text{s. t.} \quad \begin{cases} f_{perf}(\vec{x}) - P_{spec} = 0 \\ x_{min,d} \leq x_d \leq x_{max,d} \qquad d \in \{1,\ldots,D\}. \end{cases} \end{aligned} \tag{4.20}$$

This optimization problem becomes complicated because the equality constraint must be satisfied very precisely for successful application of the worst-case methods. In other words, the remaining constraint violation must be considerably lower than the accuracies of 10^{-3} or 10^{-4} which are commonly demanded in the literature [Ham02, Mez04],

4.3.2.2 Test Case

In the following an exemplary case from yield analysis is shown. A U-shaped electrothermal actuator is considered according to the schematic in Figure 4.13(a). In [Vud07] it has already been shown that optimization algorithms like GAs and pattern search strategies can be successfully employed for solving this problem. This test case is used here to demonstrate the performance of different constraint-handling methods using DE and PSO for the case that an equality constraint has to be fulfilled very accurately. The results of this study have also been published in [Zie07e].



(a) The basic structure of the micro-electro thermal actuator

(b) Performance function f_{perf} (light gray), performance specification P_{spec} (dark gray) and nominal point $\vec{x}_{nom}$ (circle)

Figure 4.13: Exemplary optimization problem from yield analysis

The actuator works as follows: Because the cross-sections of the arms are not equal, the upper arm (hot arm) will become warmer than the lower arm (cold arm) due to the higher current density in the hot arm. The hot arm will experience more thermal expansion than the cold arm, leading to a downward motion of the actuator.
To enable an easy analysis, only two parameters are considered: The length of the hot arm l_h and the length of the cold arm l_c. The boundaries of the design space for the optimization problem are determined by the statistical parametric room defined by the process. In this case, the maximum parametric variations are assumed to be 10% of their nominal values $l_{h,nom} = 300\mu m$ and $l_{c,nom} = 240\mu m$. The considered performance of the system is the displacement of the actuator while applying a specific current at the anchor pads of the actuator. The performance specification has been considered to be a moderate displacement of $0.81\mu m$ at a particular load current of $1.5mA$. Figure 4.13(b) shows a graphical representation of the optimization problem.
An easy approach for solving the described optimization problem is to apply the modified replacement method which has been explained in Section 4.2. However, it can be shown that the accuracies which can be reached with this method using either DE or PSO do not satisfy the demands introduced by the worst-case methods. If constraint violations of $\epsilon_{e,final} \geq 10^{-6}$ are allowed, the optimum is reliably found by both DE and PSO. For smaller constraint violations the performance deteriorates significantly until for $\epsilon_{e,final} \leq 10^{-11}$ the optimum is hardly ever found (see Figure 4.14). Three different population sizes ($NP = \{20, 50, 100\}$) have been examined to assure that the reason for the

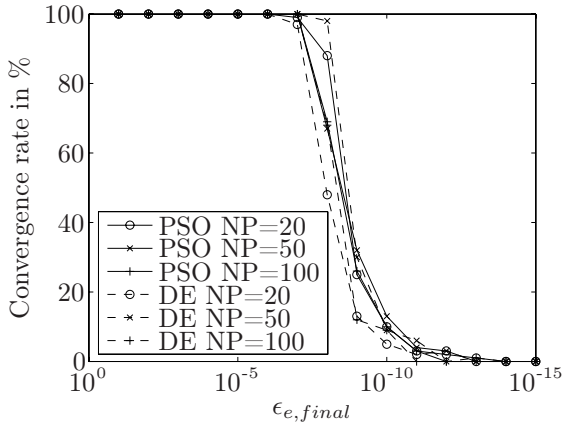
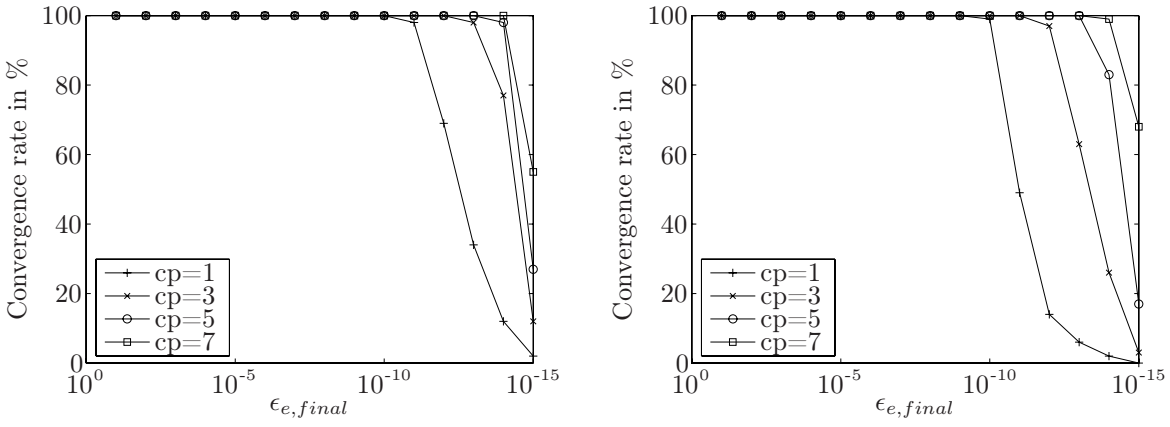


Figure 4.14: Convergence rate with fixed $\epsilon_{e,final}$

bad performance is not insufficient diversity. Because the optimization problem contains only two parameters, it would normally be assumed that a population size of $NP = 20$ is sufficient. Larger population sizes would be avoided because although the convergence rate tends to increase with the population size, the computational cost also increases. Interestingly, the convergence rate is very similar for all population sizes which were examined. For the other control parameters standard settings have been used: $F = 0.7$ and $CR = 0.9$ for DE (which are the same settings that are used in [Tak06] and which also belong to common DE standard settings [Zie06f]), and $w = 0.73$, $c_1 = c_2 = 1.5$ for PSO (corresponding to the well-known constriction variant that is commonly regarded as a standard for PSO [Bra07]), together with the *von-Neumann* neighborhood topology. Of course a possibility exists that the results might vary for different settings but an exhaustive analysis of control parameter settings would take considerable computational time. Because this possibility is assumed to be rather small, this examination is limited to standard settings.

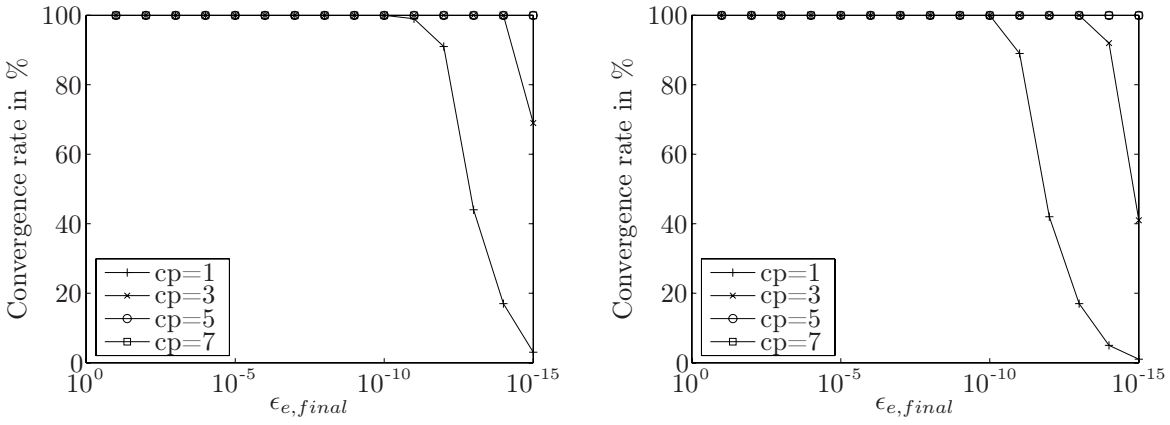
If ϵ_e is varied during the optimization run according to the schedule from [Tak06] given in Equation 4.18 (with the difference that a small value $\epsilon_{e,final} > 0$ is allowed here in contrast to [Tak06]), both DE and PSO perform significantly better (see Figure 4.15). Results are only shown for $NP = 50$ but similar results have been obtained for $NP = 20$ and $NP = 100$. Surprisingly, DE and PSO again show very similar performance, as also done for fixed ϵ_e during an optimization run. For settings of $cp = \{5, 7\}$ and $T_c = \{0.1T_{max}, 0.2T_{max}\}$ (where $T_{max} = 12500$ is the maximum number of generations, corresponding to $FE_{max} = 500,000$ with $NP = 50$), a convergence rate of 100% is reached for $\epsilon_{e,final} \in \{10^{-1}, 10^{-2}, \ldots, 10^{-15}\}$. Thus, the demands of worst-case methods are fulfilled. These parameter settings are in accordance with the settings from [Tak06] which are given as $cp = 5$ and $T_c = 0.2T_{max}$. For $\epsilon_{e,final} \geq 10^{-10}$ the performance is robust for all settings of cp and T_c which were tested. For smaller $\epsilon_{e,final}$ the convergence rate decreases for small values of cp and T_c (the convergence speed also deteriorates, see [Zie07e]). This result is not completely in accordance with [Tak06] because in [Tak06] it is stated that larger T_c but smaller cp lead to a more robust (and also slower) search process.

To sum up, it was shown that a considerably better performance may be obtained using a specialized method for handling equality constraints instead of the modified replacement method described in Section 4.2. It might be argued that the method with varying ϵ_e has a



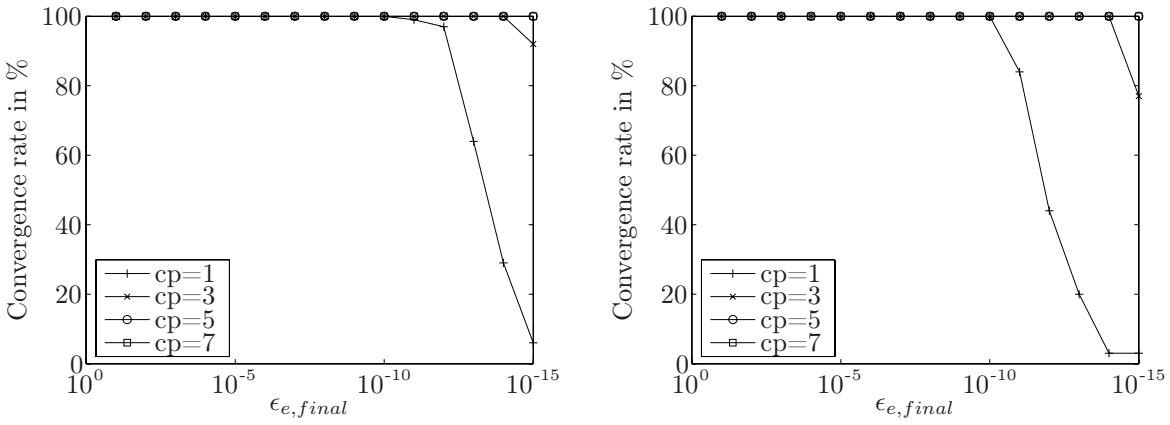
(a) PSO with $T_c = 0.02T_{max}$

(b) DE with $T_c = 0.02T_{max}$



(c) PSO with $T_c = 0.1T_{max}$

(d) DE with $T_c = 0.1T_{max}$



(e) PSO with $T_c = 0.2T_{max}$

(f) DE with $T_c = 0.2T_{max}$

Figure 4.15: Convergence rate with varying allowed constraint violation ϵ_e and $NP = 50$

disadvantage because two new parameters are introduced. However, the same parameters that were used in [Tak06] also showed good and robust performance in [Zie07e], indicating that not much effort must be spent to find suitable parameter settings.
Here a relatively simple case has been regarded as the optimization problem contains only one equality constraint. Additional complexity might arise by considering several equality constraints which may also have different magnitudes. In that case it would be beneficial to allow different settings of ϵ_e for the constraints during the optimization run, as described for ASCHEA in [Ham02]. When using an adaptive adjustment of ϵ_e as in [Ham02], the presence of inequality constraints would also mean additional difficulty of the problem. The reason is that the feasibility of population members cannot be guaranteed anymore by setting the allowed constraint violation accordingly. The development of methods capable of handling these types of problems is left for future work.

4.4 Comparison of Differential Evolution and Particle Swarm Optimization with Blind Random Search and Brute Force Search

Because evolutionary algorithms are complex methods including random elements, theoretical examinations of their behavior are generally more difficult than for deterministic optimization algorithms. Convergence proofs are usually either not available at all or they are derived based on simplified models (see e.g. [Cle02, Tre03, Kad06, Sam07, Jia07] for theoretical examinations of PSO; for DE less theoretical studies are available, see e.g. [Lam04, Pri05]). Nevertheless, evolutionary algorithms are often applied because of their flexibility and global search capabilities. In this section it is shown that they are indeed superior to simple random methods like blind random search and brute force search. In the following a short definition for blind random search and brute force search is given, and a comparison of these methods with DE and PSO is shown based on a set of constrained single-objective test problems.
Blind random search (BRS) means that a given number of solutions from the search space is selected randomly (using a uniform distribution) and independently from each other, and the solution with the best objective function value represents the result [Gen04]. In [Ama97] a blind random search is compared to a GA. It is shown that for complex problems an appropriate representation and suitable evolutionary operators have to be selected, and only in that case the GA is better than random search.
Another easy search method is to examine all points of a grid that divides each axis of the search space into evenly spaced sections. This method is called brute force search (BFS) or enumeration in [Pri05] or "Rastermethode" in [Jak04]. A problem of brute force search is that the computational effort strongly increases with the dimension. If an equal number of N points should be examined on each axis, then the number of function evaluations is $FE = N^D$. In the following examination a maximum number of function evaluations of $FE_{max} = 500,000$ will be used. Therefore, the number of points per axis will be $N = \lfloor \exp^{\frac{1}{D} \cdot \ln FE_{max}} \rfloor = \lfloor \exp^{\frac{1}{D} \cdot \ln 500,000} \rfloor = \lfloor \exp^{\frac{1}{D} \cdot 13.12} \rfloor$. In Table 4.3 the number of points per axis in dependence on the dimension is shown. While for $D = 2$ a comparably good coverage can be obtained with $N = 707$, for $D = 3$ this number has already decreased to $N = 79$. For $D > 5$ less than 10 points will be used per axis, and for $D = 20$ there is only

Table 4.3: Number of points per axis with $FE_{max} = 500,000$

D	2	3	4	5	6	7	8	9	10	13	15	20
N	707	79	26	13	8	6	5	4	3	2	2	1

one point per axis.
In this work blind random search as well as brute force search will be compared to DE and PSO. A problem in the optimization literature is that comparisons are often difficult to make because

- authors use different test functions for the evaluation of algorithms,
- if the same functions are used, different accuracies are demanded (for equality constraints as well as for the distance to the known best solution),
- different numbers of function evaluations are allowed,
- the bounds are set differently.

Therefore, in this work a standardized test set is used that was defined for the Special Session on Constrained Real Parameter Optimization at the Congress on Evolutionary Computation 2006 (CEC06) in [Lia06a]. The test set comprises 24 constrained single-objective test functions, and performance measures are also specified. Similar special sessions have been organized for unconstrained single-objective optimization at CEC05, multi-objective optimization at CEC07, and large scale, i.e. high-dimensional optimization at CEC08.
Table 4.4 shows a summary of the features of the test functions. $\rho = \frac{|\mathcal{F}|}{|\mathcal{S}|}$ specifies the estimated ratio of feasible space to the whole search space (as already mentioned in Section 4.2.4.3). The following four columns of Table 4.4 give information about the number and type of constraints: LI is the number of linear inequality constraints, NI is the number of nonlinear inequality constraints, LE is the number of linear equality constraints and NE is the number of nonlinear equality constraints. Moreover, the number of active constraints at the optimum (see also Section 2.1) is given by a. The last column shows the best solutions that were known after the first submission to the mentioned conference to ensure comparability with [Zie06a, Zie06b] (in these papers DE and PSO are tested adhering to the mentioned instructions). Some of the results were improved later, therefore some solutions shown in the following are actually better than the ones given in Table 4.4. As for most of the functions it is not clear if the given solution corresponds to the global optimum, better solutions might still be found. It should be noted that for g20 no feasible solution has been found so far [Lia06a]. For g22 some participants of CEC06 did obtain feasible solutions [Tak06, Lia06b, Hua06] but the given optimum has not been found. The rest of the participants did not even find feasible solutions for g22 once [Tas06, Mez06b, Kuk06c, Muñ06, Sin06, Bre06c, Run06, Zie06a, Zie06b].
This test set allows extensive testing because a large variety of problem features is represented. Therefore, it is used here for the comparison. Results for blind random search and brute force search will be shown using the given conditions and they will be compared with solutions using DE and PSO from [Zie06a, Zie06b]. As demanded in [Lia06a], 25

Table 4.4: Details of the test functions from [Lia06a]

Problem	D	Type of function	ρ	LI	NI	LE	NE	a	$f(\vec{x}^*)$
g01	13	quadratic	0.0111%	9	0	0	0	6	-15
g02	20	nonlinear	99.9971%	0	2	0	0	1	-0.80361910412559
g03	10	polynomial	0.0000%	0	0	0	1	1	-1.00050010001000
g04	5	quadratic	52.1230%	0	6	0	0	2	-30665.53867178332
g05	4	cubic	0.0000%	2	0	0	3	3	5126.4967140071
g06	2	cubic	0.0066%	0	2	0	0	2	-6961.81387558015
g07	10	quadratic	0.0003%	3	5	0	0	6	24.30620906818
g08	2	nonlinear	0.8560%	0	2	0	0	0	-0.0958250414180359
g09	7	polynomial	0.5121%	0	4	0	0	2	680.630057374402
g10	8	linear	0.0010%	3	3	0	0	6	7049.24802052867
g11	2	quadratic	0.0000%	0	0	0	1	1	0.7499
g12	3	quadratic	4.7713%	0	1	0	0	0	-1
g13	5	nonlinear	0.0000%	0	0	0	3	3	0.053941514041898
g14	10	nonlinear	0.0000%	0	0	3	0	3	-47.7648884594915
g15	3	quadratic	0.0000%	0	0	1	1	2	961.715022289961
g16	5	nonlinear	0.0204%	4	34	0	0	4	-1.90515525853479
g17	6	nonlinear	0.0000%	0	0	0	4	4	8853.53967480648
g18	9	quadratic	0.0000%	0	13	0	0	6	-0.866025403784439
g19	15	nonlinear	33.4761%	0	5	0	0	0	32.6555929502463
g20	24	linear	0.0000%	0	6	2	12	16	?
g21	7	linear	0.0000%	0	1	0	5	6	193.724510070035
g22	22	linear	0.0000%	0	1	8	11	19	236.430975504001
g23	9	linear	0.0000%	0	2	3	1	6	-400.055099999999584
g24	2	linear	79.6556%	0	2	0	0	2	-5.50801327159536

optimization runs are executed for each test function, every optimization run includes $500,000$ evaluations of the objective function, a run is considered to be successful if the given global optimum is found with an accuracy of $\epsilon_g = 0.0001$, and equality constraints are transformed into inequality constraints with the accuracy $\epsilon_e = 0.0001$. Not all performance measures are shown here that are suggested in [Lia06a] because their presentation needs a considerable amount of space. Only the most informative performance measures are summarized here to allow a clear overview:

- The feasible rate shows the proportion out of 25 runs in which at least one feasible individual was found.
- The success rate represents the proportion of runs in which the global optimum was reached (with an accuracy of $\epsilon_g = 0.0001$).
- The success cost is the mean number of function evaluations that is needed for convergence divided by the success rate, as already introduced in Section 4.2.4.3.
- To allow an evaluation of the best possible performance of the respective algorithm, the distance of the best objective function value that was found in 25 runs after $500,000$ function evaluations to the best known solution is shown.

Table 4.5: Feasible rate and success rate for blind random search, brute force search, Differential Evolution and Particle Swarm Optimization

Problem	Feasible Rate in %				Success Rate in %			
	BRS	BFS	DE	PSO	BRS	BFS	DE	PSO
g01	64	100	100	100	0	0	100	52
g02	100	0	100	100	0	0	84	0
g03	0	100	100	100	0	0	0	0
g04	100	100	100	100	0	0	100	100
g05	0	0	100	100	0	0	100	16
g06	100	100	100	100	0	0	100	100
g07	28	0	100	100	0	0	100	8
g08	100	100	100	100	44	0	100	100
g09	100	100	100	100	0	0	100	100
g10	96	0	100	100	0	0	100	32
g11	0	0	100	100	0	0	100	100
g12	100	100	100	100	84	100	100	100
g13	0	0	100	100	0	0	32	0
g14	0	0	100	100	0	0	100	0
g15	0	0	100	100	0	0	100	80
g16	100	100	100	100	0	0	100	100
g17	0	0	100	100	0	0	20	0
g18	0	0	100	100	0	0	100	80
g19	100	100	100	100	0	0	100	8
g20	0	0	0	0	0	0	0	0
g21	0	0	100	8	0	0	60	0
g22	0	0	0	0	0	0	0	0
g23	0	100	100	100	0	0	0	0
g24	100	100	100	100	0	0	100	100

In Table 4.5 the feasible rate and the success rate are given, and in Table 4.6 the success cost and the distance to the best solution are shown. For brute force search the success cost does not have any meaning because it depends on the way the grid is processed. Therefore, it is not given here. Besides, all optimization runs have the same performance for BFS because a deterministic search method is used. Thus, the feasible rate and the success rate are either 0% or 100%.

It is apparent that the results of blind random search and brute force search are worse than the results of DE and PSO. There are many functions for which not even feasible solutions were found with BRS or BFS. The global optimum was only found for two functions (g08, g12) using BRS whereas BFS was successful for only one function (g12). When observing the results for these functions more closely, it can be seen that the condition $(f(\vec{x}) - f(\vec{x}^*)) \leq 0.0001$ is fulfilled for a comparably large range of parameter values, making it easier for BRS and BFS to find the solution. Besides, both functions have a low dimension of $D = 2$. This also confirms the statement from [Jak04, Gen04, Pri05] that random search quickly becomes worse with increasing dimension.

It is concluded that blind random search and brute force search are not able to compete with sophisticated algorithms like Differential Evolution or Particle Swarm Optimiza-

Table 4.6: Success cost and distance to $f(\vec{x}^*)$ for blind random search, brute force search, Differential Evolution and Particle Swarm Optimization

Problem	Success Cost				Distance of best solution to $f(\vec{x}^*)$			
	BRS	BFS	DE	PSO	BRS	BFS	DE	PSO
g01	-	n.a.	33414	146530	7.61	9.00	0	0
g02	-	n.a.	134879	-	4.79E-01	-	6.54E-09	1.03E-01
g03	-	n.a.	-	-	-	1.00E+01	2.20E-01	5.91E-01
g04	-	n.a.	15986	20546	1.23E+02	1.86E+02	7.64E-11	7.64E-11
g05	-	n.a.	107076	2276363	-	-	-1.82E-12	-9.09E-13
g06	-	n.a.	7143	20043	2.80E+02	3.22E+02	4.55E-11	4.55E-11
g07	-	n.a.	93793	4091031	4.06E+02	-	7.98E-11	2.76E-05
g08	516842	n.a.	1086	2360	1.06E-05	1.35E-04	8.20E-11	8.20E-11
g09	-	n.a.	25805	58129	6.58E+01	8.46E+02	-9.81E-11	-9.81E-11
g10	-	n.a.	119217	1332999	3.82E+03	-	6.28E-11	9.86E-09
g11	-	n.a.	13380	16386	-	-	0	0
g12	207221	n.a.	5104	4893	6.34E-06	0	0	0
g13	-	n.a.	830322	-	-	-	4.19E-11	4.52E-02
g14	-	n.a.	68226	-	-	-	8.51E-12	2.09E-01
g15	-	n.a.	57968	221033	-	-	6.08E-11	6.08E-11
g16	-	n.a.	11592	33335	6.99E-02	1.01	6.52E-11	6.52E-11
g17	-	n.a.	1328459	-	-	-	8.19E-11	3.38E+01
g18	-	n.a.	79557	239026	-	-	1.56E-11	1.56E-11
g19	-	n.a.	177229	4566044	3.07E+02	5.98E+01	4.63E-11	4.74E-11
g20	-	n.a.	-	-	-	-	-	-
g21	-	n.a.	162691	-	-	-	-3.32E-10	2.98E+02
g22	-	n.a.	-	-	-	-	-	-
g23	-	n.a.	-	-	-	4.00E+02	3.00E+02	3.00E+02
g24	-	n.a.	3024	7262	8.26E-04	1.23E-02	4.67E-12	4.67E-12

tion for the problems examined here. Because the test set is large and the optimization problems exhibit many different features (see Table 4.4), it can be expected that this conclusion will be valid for a large range of optimization problems. The advantage of DE and PSO lies in the assumption that in the vicinity of good solutions even better solutions might exist, together with a randomness that helps to escape from local optima. This assumption is usually valid for real-world problems (and most test problems). In contrast, evolutionary algorithms do not have an advantage over pure random methods for functions without any coherence. This confirms the no free lunch theorem (see also Sections 2.2 and 2.3).

Comparing DE and PSO, DE always has an equal or better performance than PSO, with the exception that the success cost of PSO is slightly better for g12. Naturally, the performance is heavily dependent on the parameter settings. They have been selected based on recommendations from literature as well as preliminary tests, thus $F = 0.7$, $CR = 0.9$ and $NP = 50$ have been employed for DE whereas $w = 0.8$, $c_1 = 0.5$, $c_2 = 2.0$, the *lbest* neighborhood topology and $NP = 50$ have been used for PSO. It can be expected that the results of PSO can be improved using more suitable parameter settings or an adaptive approach. Besides, for both algorithms the modified replacement method was

used for constraint-handling, meaning that especially results for optimization problems containing equality constraints (which were identified in [Zie06a, Zie06b] as the most challenging problems) might be improved using a better method for handling equality constraints as described in Section 4.3. Besides, it might be advantageous to scale the constraints before the calculation of the sum of constraint violation to avoid undesired effects from different magnitudes of the constraints.

4.5 Summary and Future Work

Constraint-handling is an important topic because most real-world problems contain constraints. In the optimization literature dissertations can be found which are exclusively dedicated to this topic [Mez04] whereas here mainly a short overview is given, with particular focus on easily applicable methods.

It is usually reasonable to use a method for keeping individuals inside boundaries to avoid problems with positions which cannot be evaluated due to physical limitations or simply to lower the computational cost for optimization. Several methods for handling boundary constraints are described in the literature. Generally, methods should be preferred which do not decrease the diversity of the population and which do not delay convergence.

For constraint functions even more constraint-handling techniques exist in the literature. For easy applicability they should contain as few parameters as possible. Especially for inequality constraints there are techniques without any parameters which nevertheless work for a large range of optimization problems. An examination based on a power allocation problem showed that the death penalty might fail if the size of the feasible space to the whole search space becomes small. In contrast, the modified replacement method based on the feasibility rules from [Deb00] is generally also able to solve problems with a small feasible space. This indicates the advantage of using knowledge about the constraint violation for directing the individuals to feasible regions.

Furthermore, it was shown in this section that the same constraint-handling method might result in different behavior when applied to different optimization algorithms: The death penalty worked considerably worse for PSO in contrast to DE. As a result, recommendations have to be carefully examined when applying a constraint-handling technique to a different algorithm.

For the handling of equality constraints often specialized methods are needed, particularly if it is important that the equality constraints are fulfilled very accurately. It was shown for an exemplary problem from yield analysis that the modified replacement method yields good results for both DE and PSO if the allowed constraint violation is larger or equal to 10^{-6}. If lower constraint violations are desired the convergence rate deteriorates significantly. In contrast, a method that varies the allowed constraint violation during the optimization run was able to achieve convergence rates of 100% even for allowed constraint violations of 10^{-15} using appropriate parameter settings. The focus for future work should be on developing methods for problems with a mixture of several inequality constraints and equality constraints. Furthermore, the methods should be able to handle equality constraints with differing magnitude to be able to efficiently solve problems like stated in [Dee07].

Because of the randomness included in evolutionary algorithms and the lack of convergence proofs, it might be doubted if they really have an advantage over simple random

techniques. It was shown here based on a test set of 24 constrained single-objective optimization problems containing different features that DE and PSO are indeed superior to blind random search and brute force search.

Chapter 5

Multi-Objective Optimization

The basic form of most optimization algorithms was designed to minimize one objective function without constraints. However, in real-world problems usually not only constraints as discussed in the previous chapter but also several objective functions $\vec{f}(\vec{x})$ exist. These objectives functions are often conflicting like minimizing the cost but maximizing the performance of a product. Therefore, in a comparison of two solutions $\vec{a}$ and $\vec{b}$ it is not obvious anymore which one performs better if e.g. $f_1(\vec{a}) < f_1(\vec{b})$ but $f_2(\vec{a}) > f_2(\vec{b})$ (assuming minimization of all objective functions) and either the relative importance of the goals is identical or it cannot be defined clearly. Although the dominance relation according to Equations 2.11 and 2.12 is introduced to be able to compare solutions with each other, there are still solutions which are incomparable. As a result, usually not one global best solution but several trade-off solutions exist which are the Pareto-optimal set (see Section 2.1).
To solve a multi-objective optimization problem, either the optimization problem has to be transformed into a single-objective problem or the optimization algorithms have to be adapted. Classical approaches usually use transformations to single-objective optimization problems. A common method is optimizing the weighted sum of the objective functions [Deb01a]:

$$\min \tilde{f}(\vec{x}) = \min \sum_{m=1}^{M} w_m \cdot f_m(\vec{x}). \tag{5.1}$$

A difficulty of this approach is the specification of the weights w_m because different solutions are obtained depending on the weights [Bol01]. Because the sensitivity to the weights is often very high [Kno08], adjusting the weights might be a difficult optimization problem itself. Additionally, non-convex problems pose problems for weighted sum approaches [Zit00, Fle05]. Another problem with using single-objective optimization techniques for multi-objective optimization problems is that each optimization run results in only one solution. The implications of this fact will be discussed further in the following because the goals for multi-objective optimization have to be defined first.
Newer approaches are mostly based on Pareto-optimality. With these methods not a single solution is generated in one optimization run but a whole set of non-dominated solutions which ideally correspond to the Pareto-optimal set (see Section 2.1). From this set a particular solution can be chosen for application, e.g. production. That means that the decision making process is shifted in comparison to weighted sum approaches. Instead of having to specify the preferences in form of weights before the optimization

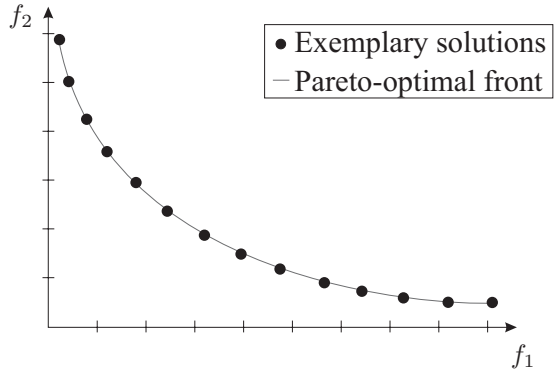
run, a solution is selected after the non-dominated solutions have been generated. As a result, more information about practicability of solutions is available for the (human) decision maker.

On the basis of the possibilities that population-based methods using Pareto-optimality offer, the following goals are defined for multi-objective optimization: First, the obtained non-dominated solutions should be as close as possible to the Pareto-optimal front of the optimization problem. This goal is analog to the demand of convergence to the global optimum in single-objective optimization. Often an infinite number of Pareto-optimal solutions exist. Naturally, only a finite number of solutions can be generated during an optimization process. Furthermore, the number of generated solutions must be limited because otherwise the computational cost would become too large. Nevertheless, the largest possible freedom of choice should be offered to the decision maker. Therefore, a well-distributed approximation set is demanded which is a goal that consists itself of two requirements: An extent that is as large as possible and furthermore a distribution that is as evenly spaced as possible. Pareto-optimal fronts may be disconnected, so in that case an exactly uniform distribution of solutions is not possible. Nevertheless, the non-dominated solutions should cover all regions of the Pareto-optimal front and reproduce the curvature of the underlying Pareto-optimal front as correctly as possible. These demands do not have a counterpart in single-objective optimization because in that case only one solution is generated.

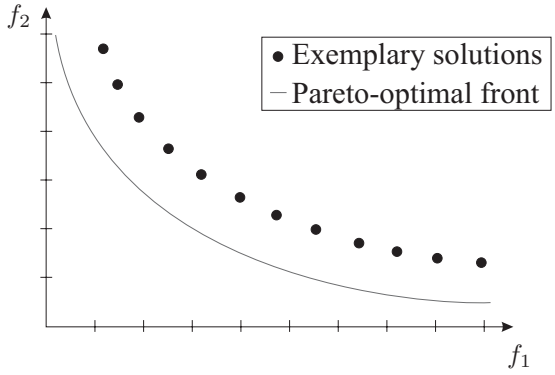
In Figure 5.1 the goals of multi-objective optimization are illustrated. Figure 5.1(a) shows a set of well-distributed non-dominated solutions which have a large extent and which have converged to the Pareto-optimal front. In Figure 5.1(b) the non-dominated solutions have not reached the Pareto-optimal front whereas the demands concerning the distribution are fulfilled. In Figure 5.1(c) the non-dominated solutions are on the Pareto-optimal front and an evenly spaced distribution has been obtained but the extent of the solutions is worse than in Figure 5.1(a). Figure 5.1(d) shows a set of solutions with good convergence and large extent but with less uniform distribution than in Figure 5.1(a).

Usually, not all Pareto-optimal solutions are found even using algorithms with concepts from Pareto-optimality because especially for floating-point representation the number of Pareto-optimal solutions can approach infinity. Nevertheless, normally a good approximation of the Pareto-optimal front can be achieved because methods are included in the algorithms that target the desired characteristics of the distribution (large extent and uniform distribution). Of course with single-objective optimization methods also several Pareto-optimal solutions can be generated by restarting the algorithm several times (with varying weights) but it is more difficult to control the distribution of solutions. In algorithms using concepts from Pareto-optimality, special operators are employed for this goal which is not possible in single-objective optimization. It might be argued that if the relative importance of the objectives is known before optimization, it might be sufficient to use a weighted sum method (see also [Deb01a]). However, an exact quantification in the form of weights is seldom possible or many computational resources have to be spent for fine-tuning of weights [Sch02]. Bearing in mind that the results of optimization runs are usually highly sensible to the values of the weights, an approach based on Pareto-optimality should be preferred.

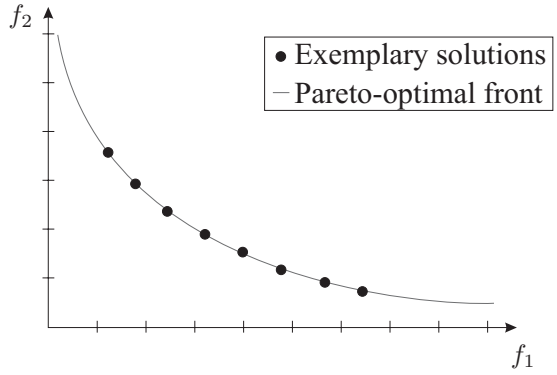
Because the result of a multi-objective optimization algorithm using concepts from Pareto-optimality consists of a whole set of solutions, it becomes difficult to compare results. Therefore, performance measures for multi-objective optimization are shown in the fol-



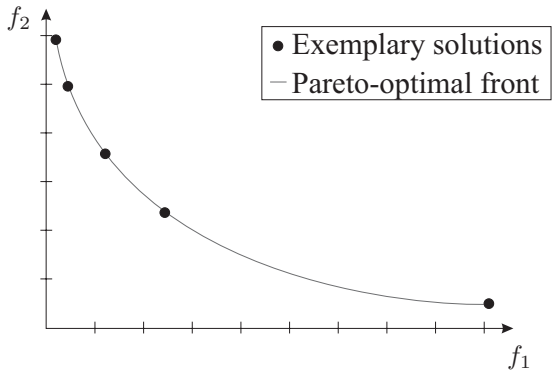
(a) Non-dominated solutions with good distribution



(b) Convergence deteriorated in comparison with Figure 5.1(a)



(c) Extent of solutions worse than in Figure 5.1(a)



(d) Solutions less uniformly distributed than in Figure 5.1(a)

Figure 5.1: Goals in multi-objective optimization

lowing which will also be used in Chapter 6 for the definition of stopping criteria. One of the most successful multi-objective evolutionary algorithms (MOEAs) is the NSGA-II [Deb02]. It is based on a Genetic Algorithm but it contains several useful features for multi-objective optimization that can be transferred to other algorithms. This is often done in the literature. Furthermore, when new algorithms or a new extension for an existing algorithm is presented, its performance is mostly evaluated in contrast to NSGA-II. Because of this large impact on multi-objective optimization, the NSGA-II is introduced shortly in the following. Later, it will be shown how concepts from NSGA-II can be used in multi-objective DE and PSO algorithms. A short literature review also shows multi-objective DE and PSO algorithms based on different mechanisms. Because the focus of this work is on DE and PSO, multi-objective algorithms based on other EAs like SPEA2 (Strength Pareto Evolutionary Algorithm 2 [Zit02]) or PAES (Pareto Archive Evolution Strategy [Kno00]) will not be discussed further. For information about these algorithms the interested reader should refer to the literature, e.g. to [Coe03, Deb01a]. Furthermore, for DE several variants based on NSGA-II are introduced that result in different amounts of elitism and diversity, and a detailed comparison is done. At the end of this chapter the application of multi-objective DE and PSO algorithms to an application example is shown that consists of optimizing the gain, bandwidth and current consumption of an operational amplifier.

5.1 Performance Measures

In single-objective optimization it is easy to compare results of different optimization runs: The objective function values are evaluated that have been reached with a given number of objective function evaluations, and the solution with smaller objective function value is regarded as better (for minimization problems). For multi-objective optimization the situation is more complicated because several objective functions exist. Thus, an easy comparison of objective function values is not possible as already discussed in the beginning of this chapter. Besides, a whole set of solutions has to be regarded. Consequently, other performance measures have to be found to compare results.
In contrast to single-objective optimization where the goal is convergence to the global optimum, in multi-objective optimization usually two goals are formulated: Convergence to the Pareto-optimal front and a well-distributed approximation set, meaning a large extent and a uniform distribution of solutions. Different performance measures have been developed which measure either one of the goals or both together. Up to now no single performance measure has been found that is able to satisfactorily evaluate all aspects. Generally, too much information gets lost while transforming the multi-dimensional data to a scalar value. Thus, in comparisons often several performance measures are given.
Many performance measures only make sense when used in comparisons while others can also be used to illustrate the performance of one particular algorithm. In the literature often also the obtained non-dominated solutions are shown for $M = 2$ or $M = 3$ objective functions. The reason is that they are generally easier to interpret than the often abstract performance measures because they allow an overview about the performance, e.g. single outliers can be identified which may distort the result of performance measures.
In the following, performance measures are given following the classification in [Deb01a]: It is distinguished between performance measures evaluating the closeness to the Pareto-optimal front, metrics that regard the diversity of non-dominated solutions, and measures that consider both the previously mentioned goals. The variety of performance measures in the literature is large, e.g. in [Eng06, Zit03, Kno06] also performance measures for multi-objective optimization are given.

5.1.1 Metrics Evaluating Closeness to the Pareto-Optimal Front

For the performance measures in this section it is usually required to know the Pareto-optimal front. Thus, they are mostly only applicable for test functions. The Pareto-optimal front can be given in two different forms: Either it can be calculated analytically or a list of solutions is available.
The **error ratio** ER counts the number n of solutions of a non-dominated set which are not members of the Pareto-optimal front and divides it by the number m of all solutions of the respective set: $ER = \frac{n}{m}$. Thus, the error ratio can only take values $ER \in [0, 1]$, and a smaller ER means better convergence to the Pareto-optimal front. Disadvantages of this metric are that the Pareto-optimal front must be known with a high resolution and the distance to the Pareto-optimal front is not evaluated. Furthermore, for real-valued representation problems may arise because a solution will not be considered as part of the Pareto-optimal front if it differs even insignificantly from it. To avoid the latter disadvantage a threshold can be defined, so that solutions in a certain distance ϵ_{ER} of the Pareto-optimal front will be considered as converged (see also Section 6.2).

The **set coverage metric** $\mathcal{C}(A, B)$ can be used to compare the results of two sets of solutions with each other. The two sets may be the results of two different algorithms or the outcome of an algorithm can be compared with the Pareto-optimal front. This metric determines the percentage of members of a set B that are dominated by members of set A:

$$\mathcal{C}(A, B) = \frac{|\vec{b} \in B | \exists \vec{a} \in A : \vec{a} \preceq \vec{b}|}{|B|}. \tag{5.2}$$

The set coverage metric results in values $\mathcal{C}(A, B) \in [0, 1]$. $\mathcal{C}(A, B) = 1$ means that all solutions of B are dominated by solutions of A or they are equal to each other. $\mathcal{C}(A, B) = 0$ means that no solutions of B are dominated by or equal to solutions of A. It must be noted that generally $\mathcal{C}(A, B) \neq 1 - \mathcal{C}(B, A)$.

The **generational distance** *GD* calculates an average distance of the solutions of a non-dominated set Q from the Pareto-optimal set P^*:

$$GD = \frac{\left(\sum_{i=1}^{|Q|} d_i^p\right)^{\frac{1}{p}}}{|Q|} \tag{5.3}$$

where $p = 1$ is used in this work as it is also often done in the literature [Deb01a], and d_i is the shortest distance from member i of Q to a member of P^*. For $GD = 0$ the solutions in Q and the Pareto-optimal front are identical. For increasing *GD* the average distance to the Pareto-optimal front becomes larger. To use this performance measure it is necessary to know the Pareto-optimal front.

The **maximum Pareto-optimal front error** *MFE* calculates the worst distance between the members of a non-dominated set and the Pareto-optimal front. A small value indicates better performance. A disadvantage is that it does not take the distribution of all solutions into consideration, so it can give misleading results concerning the state of convergence. Again, the Pareto-optimal front has to be known to use this metric.

5.1.2 Metrics Evaluating Diversity among Non-Dominated Solutions

The **spacing** metric S calculates the deviation of the distances between the solutions of the non-dominated set Q:

$$S = \sqrt{\frac{1}{|Q|} \sum_{i=1}^{|Q|} (d_i - \bar{d})^2} \tag{5.4}$$

where

$$d_i = \min_{\vec{x}_k \in Q \wedge k \neq i} \sum_{m=1}^{M} |f_m(\vec{x}_i) - f_m(\vec{x}_k)| \tag{5.5}$$

is a distance measure to the nearest neighbor (not Euclidean distance is measured but the sum of differences in objective function values) and $\bar{d} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} d_i$ is the average distance. A small value for S means that the distribution of the solutions is nearly uniform. A disadvantage is that the extent of the solutions is not considered in this metric.

In contrast to spacing, the **spread** metric also takes the extent of solutions into account:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} \left|d_i - \bar{d}\right|}{\sum_{m=1}^{M} d_m^e + |Q|\,\bar{d}}. \tag{5.6}$$

To calculate d_i it is possible to use any distance measure between neighboring solutions like the Euclidean distance, the sum of the absolute differences in objective values or the crowding distance (the calculation of crowding distance will be explained in Section 5.2). $\bar{d}$ is the mean value of these distance measures. d_m^e is the distance between the extreme solutions of the non-dominated set and the Pareto-optimal front for the m-th objective function. For this reason the knowledge of the extreme solutions of the Pareto-optimal front is required. For an ideal distribution which covers the whole range of the Pareto-optimal front uniformly, the spread metric will be zero. Unfortunately, for $\Delta > 0$ it is not possible to distinguish between effects from a non-uniform distribution or from a Pareto-optimal front that is only partly discovered.
The **maximum spread** is defined as the length of the diagonal of a hyperbox formed by the extreme function values from the non-dominated set (given as normalized version here):

$$\bar{D} = \sqrt{\frac{1}{M}\sum_{m=1}^{M}\left(\frac{\max_{i=1}^{|Q|} f_m(\vec{x}_i) - \min_{i=1}^{|Q|} f_m(\vec{x}_i)}{f_{m,max} - f_{m,min}}\right)^2} \tag{5.7}$$

where $f_{m,max}$ and $f_{m,min}$ are the maximum and minimum values of the Pareto-optimal solutions for the m-th objective function (if the Pareto-optimal front is not known, the normalization can be omitted). For two-objective problems this metric refers to the Euclidean distance between the two extreme solutions in objective space.
The **chi-square-like deviation measure** has originally been suggested for multimodal optimization but it can also be employed for multi-objective optimization. It uses a neighborhood parameter, so that for every Pareto-optimal solution the number of solutions of the non-dominated set in its neighborhood is counted. It is assumed that every neighborhood should ideally contain the same number of solutions. This metric also requires a known Pareto-optimal front.

5.1.3 Metrics Evaluating Closeness and Diversity

Hypervolume *HV* is a commonly used performance measure that evaluates both the closeness to the Pareto-optimal front as well as diversity. The volume of a hypercube limited by the obtained non-dominated solutions and a reference point $\vec{r}$ is calculated in objective space (see Figure 5.2). For the reference point a vector is used whose components are equal or worse than the worst objective function values that were found. A better performance is indicated by a larger hypervolume. Because objective functions can have a different range of values, it is ensured that no objective is favored by normalizing all values using the minimum and maximum objective function values that were found. If the Pareto-optimal front is known, it is also possible to evaluate the hypervolume ratio or the hypervolume difference of a non-dominated set with respect to the Pareto-optimal front.
Because EAs are stochastic methods, the approximation sets generated in different optimization runs are not necessarily equal. Another possibility for evaluating the performance of multiple optimization runs besides the calculation of e.g. average performance measures is to use **attainment surfaces** [Fon96, Kno06]. Attainment surfaces show graphically which part of the search space is dominated by solutions generated in a certain percentage of optimization runs. Examples will be shown in Appendix B (Figures B.11-B.14).

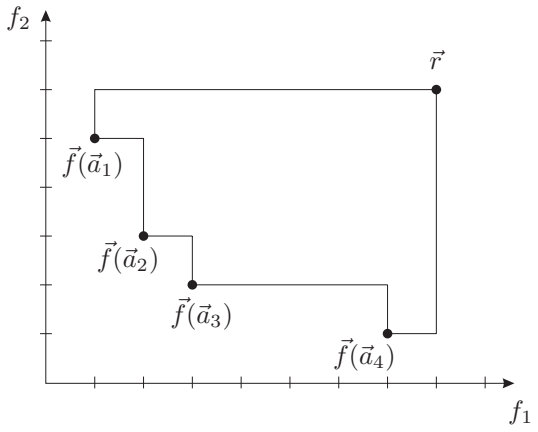


Figure 5.2: Calculation of hypervolume

Another method for measuring closeness to the Pareto-optimal front and diversity together is to combine metrics for the individual goals. In the **weighted metric** several metrics are added together, scaled with different weights. Similar as for the spread metric, a disadvantage is that the effects of the individual goals is obliterated. A different approach is the **non-dominated evaluation metric** that interprets two metrics as a two-objective optimization problem, therefore the performance of algorithms can be evaluated by regarding dominance of the metrics.

5.2 Non-dominated Sorting Genetic Algorithm II

NSGA-II is one of the most famous evolutionary algorithms for multi-objective optimization. The most prominent features of NSGA-II are its low computational complexity, elitist approach and a method for diversity that does not need additional parameters. The underlying Genetic Algorithm will not be discussed here. Instead, the focus is on the concepts used for handling multi-objective optimization problems which can also be transferred to other algorithms.
In the beginning of an optimization run an initial population of size NP is randomly generated. In principle, the following procedure is the same as for single-objective optimization: Parents are selected, an offspring population of size NP is generated by employing recombination and mutation operators, and then the survivors are selected which are part of the next generation. Only the parent selection as well as survivor selection are done differently.
After an offspring population has been built, the parent and the offspring populations are joined, forming a set of $2 \cdot NP$ members. This set is sorted into non-dominated fronts, so that members of one front are non-dominated, and the best solutions are stored in front $\mathcal{F}_1$ (see Figure 5.3). In [Deb02] it is discussed that a naive implementation of a method for sorting the population based on domination would require $\mathcal{O}(M \cdot NP^3)$ calculations whereas NSGA-II needs only $\mathcal{O}(M \cdot NP^2)$ computations. The storage requirement increases from $\mathcal{O}(NP)$ to $\mathcal{O}(NP^2)$ but for multi-objective optimization problems usually computation time is much more important than storage space. The lower computation time is realized by a sophisticated procedure for comparing solutions: First, for each solution $\vec{x}_i$ the domination count n_i (number of dominating solutions) and the set S_i of

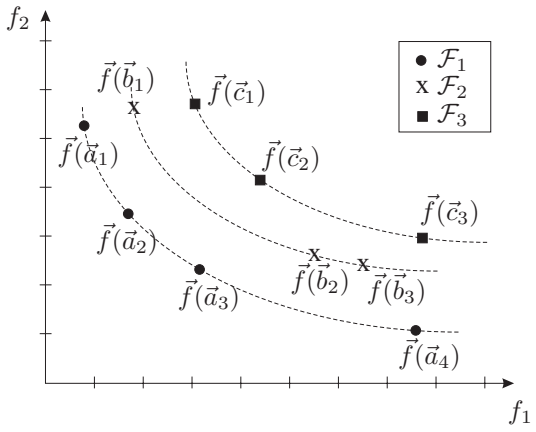


Figure 5.3: Non-dominated fronts

solutions that are dominated by $\vec{x}_i$ are calculated. Solutions with $n_i = 0$ build the first non-dominated front $\mathcal{F}_1$. For every member of $\mathcal{F}_1$, the domination count of all solutions of their corresponding sets S_i is decreased by 1. The solutions which reach a domination count of 0 build the second non-dominated front $\mathcal{F}_2$. This procedure is repeated until all solutions are sorted into non-dominated fronts $\mathcal{F}_1 \dots \mathcal{F}_l$ where l is the index of the last front. The index of the fronts is also called the **Pareto rank** of the solutions of the respective front.
The next generation is built by subsequently adding fronts beginning at $\mathcal{F}_1$. Generally, there will be a front $\mathcal{F}_a$ with $a \in \{1, \dots, l\}$ that cannot be added completely to the next generation without exceeding the desired population size NP. Therefore, individuals from $\mathcal{F}_a$ which contribute most to diversity are chosen for the next generation. For this purpose, the **crowding distance** is used which is a measure for the closeness of individuals surrounding a solution.
The crowding distance can be evaluated either in objective space or in parameter space, meaning that the distribution of solutions is influenced either in parameter or in objective space. Generally, it is used in objective space. To compute the crowding distance for a set of population members, the solutions are sorted according to their objective function value. This is done once for every objective function, and the corresponding distances are added. The individuals with the smallest or largest objective function value are assigned an infinite crowding distance (or an arbitrary large number for practical purposes) because the extent of solutions should be as large as possible. For all other solutions the crowding distance is calculated according to

$$d_{\vec{x}_i} = \sum_{m=1}^{M} \frac{f_m(\vec{x}_{i+1}) - f_m(\vec{x}_{i-1})}{f_{m,max} - f_{m,min}} \tag{5.8}$$

where f_m corresponds to the m-th objective function. A large crowding distance is preferable because it indicates that there are no other population members in the near vicinity of the solution. In Figure 5.4 the calculation of crowding distance is illustrated. Solutions $\vec{a}_1$ and $\vec{a}_4$ would receive a crowding distance of infinity because they are the extreme solutions. For solutions $\vec{a}_2$ and $\vec{a}_3$, $\tilde{d}_{\vec{a}_2,1}$ and $\tilde{d}_{\vec{a}_2,2}$ (or $\tilde{d}_{\vec{a}_3,1}$ and $\tilde{d}_{\vec{a}_3,2}$, respectively) would be scaled with the corresponding maximum and minimum objective function values, and the resulting terms would be added.

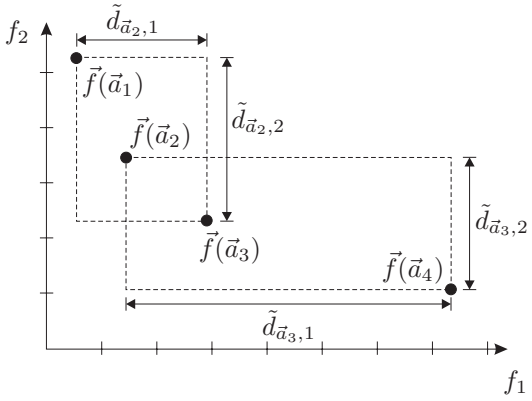


Figure 5.4: Calculation of crowding distance

The parent selection procedure is also based on non-domination rank and diversity preservation. If two individuals with different rank are compared, the one with the lower non-domination rank wins. In the comparison of two individuals which have the same rank, the one with the larger crowding distance is preferred.
NSGA-II was also extended for constrained multi-objective optimization problems in [Deb02] by adopting feasibility rules similar to the ones described in Section 4.2. Again, a solution $\vec{x}$ is considered better than another solution $\vec{y}$ if $\vec{x}$ is feasible and $\vec{y}$ is infeasible or if both are infeasible and $\vec{x}$ has the lower amount of constraint violation. Only the comparison of two feasible solutions has to be changed, so that a solutions wins that dominates its competitor.

5.3 Multi-Objective Differential Evolution

Like other EAs, Differential Evolution was developed for unconstrained single-objective optimization in the beginning. When adapting the algorithm for multi-objective optimization, the operators mutation and recombination may remain unchanged because the objective function is not regarded in these operators. In contrast, selection in single-objective DE is based on objective function values, so this operator must be adapted for multi-objective DE. Because the goals in multi-objective optimization are different from single-objective optimization (see the beginning of this chapter), the other operators might be changed as well to reflect the resulting demands of a high diversity. However, this is seldom done in the literature (see Section 5.3.2 where in [Xue03] an exception is described that adapts the mutation operator for multi-objective optimization).
In the following, it is explained how concepts from NSGA-II can be used for building a multi-objective DE algorithm. Because these concepts can be transferred in different ways, four algorithm variants are generated by combining two different implementations of the selection scheme with two variants of calculating the crowding distance. The former variation has already been discussed in the literature whereas the latter has been developed within the scope of this thesis. Afterwards, a short overview about literature for multi-objective DE is given. It is indicated where algorithms similar to the already mentioned four variants are shown but also different approaches are described. At the end of this section a comparison of the four multi-objective DE variants is shown.

5.3.1 Adaptation of Concepts from NSGA-II for Multi-Objective Differential Evolution

When adapting Differential Evolution for multi-objective optimization, the comparison of objective function values during selection in single-objective optimization can be substituted by the dominance relation. As a result, the question arises what to do if the compared solutions are non-dominated regarding each other, so it cannot be decided which one is better. The problem which one to choose for the next generation can be solved by using techniques from NSGA-II: Adding the respective trial vector to the population and deciding later (after generating trial vectors for each vector of the current generation) which solutions to keep based on dominance and a diversity measure. However, in NSGA-II not only non-dominated child solutions are added to the population but all solutions which have been generated. Consequently, the question appears how to proceed for multi-objective DE. One possibility is to keep the direct comparison between target vector $\vec{x}_i$ and trial vector $\vec{u}_i$. Thus, $\vec{u}_i$ replaces $\vec{x}_i$ in the next generation if the trial vector dominates the target vector, $\vec{u}_i$ is discarded if it is dominated by $\vec{x}_i$, and the trial vector is only added to the population if it is non-dominated regarding the corresponding target vector. This variant will be called variant 1 in the following. The other possibility is closer to NSGA-II than to the original DE selection scheme by skipping the direct comparison and instead adding all trial vectors to the population for later evaluation. This variant will be called variant 2 in the following.

DE is an elitist algorithm because in a comparison of two solutions always the one with the better properties wins. Elitism is even more pronounced in the second variant compared to the first variant: The direct comparison between target vectors and trial vectors allows discarding of individuals which are relatively good in comparison to the whole population but worse than the competitor. In contrast, only the best NP individuals are kept using variant 2. Therefore, it is assumed that the second variant will converge faster but the first variant will be more successful for complex problems due to its higher diversity. It has to be noted that the differences between variants 1 and 2 will only persist as long as trial vectors are found which dominate the corresponding target vectors. Especially in later stages of an optimization run, there may be only a small percentage of the population members for which this assumption holds. In that case differences between variants 1 and 2 will be small.

Using variant 2, the population size equals $2 \cdot NP$ after applying the evolutionary operators. Depending on the number of non-dominated target and trial vectors, the population size will be in $\{NP, \ldots, 2 \cdot NP\}$ for variant 1. For both variants the population is ordered into non-dominated fronts as described in Section 5.2 for NSGA-II.

There are different possibilities how to proceed for the calculation of crowding distances. In NSGA-II the crowding distance is calculated for each front separately. Thus, when the next generation is built and there is a front $\mathcal{F}_a$ which can not completely be added without exceeding the desired population size, only members of front $\mathcal{F}_a$ are regarded for the calculation of crowding distances. This approach may be unfavorable if an individual is far away from members of its own front but very close to an individual of another front $\mathcal{F}_i$ with $1 \leq i < a$. As a result this individual is considered to contribute much to diversity when in fact it does not. Furthermore, the outermost individuals of front $\mathcal{F}_a$ have a crowding distance of infinity to facilitate a large extent of the distribution. Thus, they are always kept regardless of the closeness of other solutions. However, there may

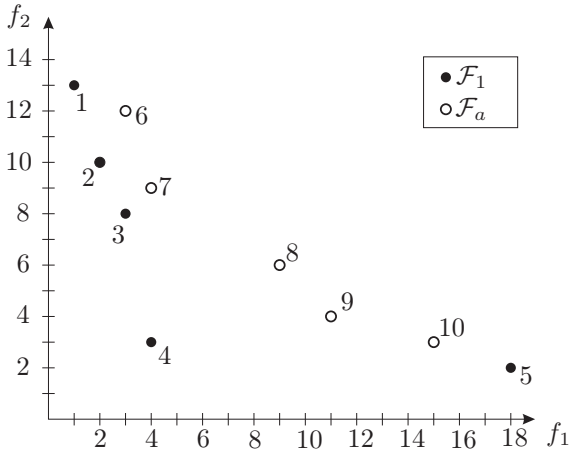


Figure 5.5: Fronts $\mathcal{F}_1$ and $\mathcal{F}_a$

be solutions in other fronts that already provide an extent of equal or even larger size. Thus, it may not be necessary to keep the outermost solutions of front $\mathcal{F}_a$.

Therefore, two algorithm variants will be examined: In variant A the calculation of crowding distances is conducted in the same way as in NSGA-II whereas in variant B individuals from fronts $\mathcal{F}_i$ with $1 \leq i \leq a$ are included during the calculation of crowding distances for individuals from front $\mathcal{F}_a$. Hence, in variant B solutions receive a crowding distance of infinity only if there are no other solutions in fronts with lower index that provide an equally large or even larger extent.

If only one front exists, variants A and B are equal. In other cases, variant B is expected to result in higher diversity. As shown in [Zie05a] (which was generated within the scope of this thesis), variant B is able to avoid gaps in the distribution of individuals that would have been caused by using variant A: Figure 5.5 shows ten individuals that are sorted into two fronts, and a population size of $NP = 8$ is desired. First, all solutions from front $\mathcal{F}_1$ are added to the next generation. Furthermore, three individuals from front $\mathcal{F}_a$ have to be selected for the next generation by evaluating their crowding distances. The crowding distances calculated for variant A and B, respectively, are displayed in Table 5.1. Using variant A, individuals 6, 7 and 10 would be chosen for the next generation because these solutions feature the largest crowding distances. In contrast, variant B produces the largest crowding distances for individuals 8, 9 and 10. In Figure 5.5 it can be seen that variant B results in larger diversity because the gap between individuals 4 and 5 is

	Crowding distance	
Solution number	Variant A	Variant B
6	∞	0.33
7	1.17	0.48
8	1.14	0.78
9	0.83	0.63
10	∞	0.68

Table 5.1: Crowding distance using variant A and variant B

filled whereas individuals 6 and 7 which reside in an already covered part of the space are omitted. The disadvantage is avoided that using variant A the outermost solutions from front $\mathcal{F}_a$ are always selected for the next generation regardless of how many individuals from the previous fronts are in their vicinity.
The effect of using variant B instead of variant A is assumed to be especially pronounced in the beginning of an optimization run because there will be many fronts. In later stages often only one front exists because all population members are non-dominated regarding each other. In [Deb01c] it is stated that it may be disadvantageous if algorithms first try to reach the Pareto-optimal front and only afterwards work on finding a good distribution of solutions. Using algorithm variant B, this unfavorable behavior is expected to be prevented or at least reduced.
Variants 1 and 2 as well as variants A and B can be combined to form a total of four different variants:

- 1A: Trial vectors and corresponding target vectors are compared directly (original DE selection scheme), calculation of crowding distance like in NSGA-II.
- 1B: Trial vectors and corresponding target vectors are compared directly (original DE selection scheme), adapted calculation of crowding distance.
- 2A: All trial vectors are added to the population, calculation of crowding distance like in NSGA-II.
- 2B: All trial vectors are added to the population, adapted calculation of crowding distance.

It should be noted that the desired number of non-dominated solutions might be different from the population size. For example, in the Special Session on Performance Assessment of Multi-Objective Optimization Algorithms at CEC07 (see also Section 4.4) 800 solutions were demanded in some cases (see [Hua07a] where initially 3000 solutions were required but due to problems with the computational cost this number was reduced later to 800). Because population sizes are not usually chosen that large, in that case it is possible to use an archive in which non-dominated solutions are stored. The archive is updated after every new generation based on domination and crowding distance (see also [Zie07b]).

5.3.2 Related Literature

One of the first multi-objective DE algorithms is described in [Abb01] and extended with self-adapted control parameters in [Abb02]. Using that approach, individuals are accepted only when they dominate their parents, and a distance metric is only used when the number of non-dominated solutions exceeds a given threshold.
A similar method that was inspired by NSGA-II is reported in [Mad02]. Different implementations are checked where one is similar to variant 1A (the direct comparison of parent and offspring is kept but only after assigning non-dominated ranks and calculating crowding distances for the whole population) and the other is equivalent to variant 2A by keeping only the best NP from $2 \cdot NP$ solutions. The author reports that the second implementation yields better results because of its faster convergence while the first implementation has good diversity but slow convergence.

[Ior04] follows a different approach by incorporating the DE mutation and recombination scheme into NSGA-II for solving rotated problems. Therefore, a similar method as variant 2A is used that shows good results in comparison to NSGA-II. A newer paper by the same authors [Ior06] investigates the incorporation of directional information by using two difference vectors: One should point in the direction of the Pareto-optimal front (by calculating the difference between the current individual and a solution that dominates it) and the other should help to maintain diversity (by computing the difference of two vectors with the same rank as the currently regarded individual).
In [Xue03] not only selection but also mutation is modified for multi-objective optimization. This is done because a DE scheme is used that includes the current best individual for mutation (see Section 3.5.2). If the target vector is dominated by other solutions, one of them is randomly chosen as best individual. The target vector is selected itself as best solution if it is non-dominated. The authors state that the original selection procedure of NSGA-II is too elitist to produce good results because individuals of better non-dominated rank will always be kept regardless of the density of solutions around it. Instead, they use a rank assignment technique [Gol89] but penalize the fitness based on the density of solutions. The original approach is adapted in [Xue05] where a fuzzy logic controller is used to tune control parameters.
A method based on the Vector Evaluated Genetic Algorithm (VEGA [Sch84]) is described in [Par04]. It is a multi-population variant where originally each population optimizes one of the objective functions, and the populations share information by migration of the best individuals. It is modified by using the dominance operator for selection but no diversity measure is used. An interesting characteristic in comparison to other approaches is that it can be easily parallelized by evolving populations on different machines.
In [Rob05b] three variants of multi-objective DE are tested which basically correspond to variant 1A and differ in the vectors which are compared for selection. Apart from a variant where trial vectors are compared to the corresponding target vector, also variants are checked where the trial vectors are compared to the most similar individual in the population. The similarity is either determined in parameter space or in objective space. As the latter two variants do not show better performance than the first but are computationally more expensive, the first variant is recommended for use. This algorithm is also tested against other multi-objective evolutionary algorithms in [Rob05a]. It is emphasized that individuals which enter the population should immediately be able to participate in the procreation process of other vectors to accelerate convergence. Besides the discussed variant, in [Tuš07] also multi-objective DE variants based on other algorithms than NSGA-II are shown. An interesting observation in [Tuš07] is that although quality indicators often show statistically significant differences between algorithm variants, the differences are not necessarily visible in plots of the approximation sets or attainment surfaces.
Another multi-objective DE algorithm is GDE3 (Generalized Differential Evolution 3) [Kuk05]. It also modifies the selection rule of the basic DE algorithm in a similar way as described here in variant 1. A difference is that the comparison of two infeasible solutions is based on dominance in constraint violation space. GDE3 also incorporates non-dominated sorting and a method for maintaining diversity that was improved in [Kuk06a]. In that paper a method for pruning (reduction of population size if NP is exceeded) is shown that is not based on crowding distance. Fast calculation as well as applicability for more than two objectives is emphasized, and the results for two and three objectives are very good. Methods for maintaining diversity apart from crowding distance are also developed for

other multi-objectives EAs besides DE, e.g. a MaxiMin technique is described in [Sol05] or a process called δ-similar elimination is proposed in [Sat06] for NSGA-II.

5.3.3 Analysis of Convergence Behavior and Diversity for Different Variants of Multi-Objective Differential Evolution

In Section 5.3.1, four variants of multi-objective Differential Evolution based on concepts from NSGA-II have been introduced. It has been discussed that these variants will most likely result in different convergence speed and different amounts of diversity. The mentioned variants were firstly compared in [Zie05a] (which is a work produced within the scope of this thesis) based on three constrained multi-objective optimization problems (CONSTR, SRN and TNK from [Deb02]). Convergence to the Pareto-optimal front was only visually evaluated whereas the diversity was measured using the spacing metric described in Section 5.1.2. Similar results have been found for the four variants. It was assumed that the test functions were too easy to demonstrate the strengths of the different implementations.
Therefore, in [Zie07d, Zie08d] an extensive test of the variants is shown ([Zie07d] is also a work produced within the scope of this work, and [Zie08d] is an extended version published in a journal): The performance is not only checked at some discrete numbers of generations as done in [Zie05a] (where results have been checked after 200 and 500 generations) but the performance is continuously monitored and displayed, so different behavior over time can be detected. Furthermore, more informative performance measures are applied. In the following, the results of this examination will be summarized.
As basis for the examination, some of the most widely used multi-objective test problems have been used which are referred to as ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 [Zit00] (sometimes also denoted as T_1, T_2, T_3, T_4 and T_6). The names of these test problems are based on the names of three researchers which are Zitzler, Deb and Thiele [Deb01a]. The problem formulation as well as a graphical representation of the Pareto-optimal fronts is shown in Appendix A. A problem ZDT5 also exists but because its variables are represented as binary strings, it has been omitted. All optimization problems are bi-objective, thus an extension of this examination for more objectives is left for future work.
For every test problem and each variant, 100 independent optimization runs have been conducted where the initial population has been equal for every variant. The control parameters have been set to $F = 0.7$, $CR = 0.9$ and $NP = 100$, and the maximum number of generations is 1000.
Several performance measures have been used. Because using spacing (see Section 5.1.2), maximum spread (see Section 5.1.2) and hypervolume (see Section 5.1.3) almost no differences could be seen between the variants, they are omitted here.
For problems ZDT1, ZDT2, ZDT3 and ZDT6 the Pareto-optimal fronts as shown in [Zit00] have been successfully generated. In Figures B.1-B.4 the combined non-dominated solutions of all independent runs are shown for four different generations. There are noticeable differences between the algorithm variants regarding the non-dominated solutions after 50 and 100 generations, and especially for ZDT6 still after 200 generations. After 1000 generations all algorithm variants have converged to the Pareto-optimal front, and no differences can be seen anymore.
For function ZDT4 none of the algorithm variants has been able to generate enough

diversity (the populations converged to a single point), so the results for ZDT4 are not shown. In [Mad02] also convergence problems are reported for the optimization of ZDT4 using a multi-objective DE algorithm. It is assumed that this behavior is caused by unsuitable control parameter settings [Kuk04a]. It should be noted that all results shown here may vary for different control parameter settings. This is an issue that always complicates comparisons of optimization algorithms.

In Figures B.5-B.7 the average set coverage metric (see Section 5.1.1) is shown for all combinations of algorithm variants. Often the differences are rather small, so it is difficult to draw conclusions from these results. For ZDT1 1A is better than 1B in a certain range of generations, furthermore 2B is better than 2A, 1A is better than 2A, and 1B is better than 2A (for the latter two combinations the reverse relations hold in earlier generations) but for most of the other problems the algorithm variants show different behavior. The most pronounced differences can be seen for ZDT6 where 2A and 2B are inferior to 1A and 1B, indicating that the original DE selection scheme is favorable. Again the first generations show different relations than later stages. This can be explained by the increased elitism of the NSGA-II selection scheme that leads to faster convergence in early generations.

An alternative for evaluating the performance of multiple runs apart from calculating averages of performance measures is to join the obtained solutions of all runs at a specific generation and compute performance measures for the combined approximation set [Sat06]. As a result, instead of the average performance the focus is on the best possible performance that can be reached. In this case diversity measures like spacing do not make sense but the set coverage metric can be evaluated. The development of the set coverage metric for the combined solutions of all independent runs (Figures B.8-B.10) is less smooth than the average set coverage metric, and more differences between the algorithm variants can be seen. The reason is that regarding the combined non-dominated solutions means that only the best solutions of 100 independent runs are shown. Runs with bad performance are not included here but they affect the average set coverage metric. Based on the set coverage metric of combined solutions, variant 2B is mostly better than 2A, especially in early stages (Figure B.8). This was already expected in Section 5.3.1. For the comparison of 1A and 1B the results differ for different optimization problems as well as the considered generation (Figure B.8), thus it cannot be decided which is better. Furthermore, variant 1B is generally preferable to 2A, and variant 1A is mostly better than 2A (Figure B.9), suggesting again the superiority of the original DE selection scheme. For most problems variant 1A is better than 2B, only for ZDT3 opposed behavior is shown (Figure B.10). The comparison of variants 1B and 2B is mostly inconclusive with the exception of ZDT6 for which 1B is better than 2B (Figure B.10).

The illustration of results using attainment surfaces (see Section 5.1.3) needs much space if it should be avoided to have confusingly many lines in one figure. Therefore, only some examples are given here: In Figures B.11-B.14 the solutions are shown that were reached in 0%, 50% and 100% of the optimization runs in 100 generations for ZDT1, ZDT2, ZDT3 and ZDT6, respectively.

In summary, the results of variants 1A and 1B are generally better than the results of 2A and 2B, indicating that it is preferable to use the original DE selection scheme instead of adding all trial vectors to the population. As a result, elitism is less pronounced and the diversity is increased. Considering the calculation of crowding distance, 2B is generally better than 2A but no conclusive statements can be made regarding variants 1B and 1A. This can be explained by the fact that the diversity of variant 2 is lower than in variant

1, thus the effect of the increased diversity due to the adapted calculation of crowding distances is more visible in the comparison of 2A and 2B. In Sections 5.5 and 6.2.4 where a multi-objective DE algorithm is used, the original DE selection scheme combined with the adapted crowding distance calculation is employed. In future work it should be tested how the diversity measure can be further improved, see also Section 5.3.2.

5.4 Multi-Objective Particle Swarm Optimization

In [Eng06] three classes of multi-objective PSO algorithms are distinguished:

- Aggregation-based methods (weighted sum approaches),
- criterion-based methods (not all objectives are regarded simultaneously),
- dominance-based methods (the concept of Pareto-optimality is employed).

Several disadvantages are mentioned for aggregation-based and criterion-based methods: Using an aggregation-based method, the weights have to be adjusted which might be difficult as already discussed in the beginning of this chapter. Besides, even if several optimization runs are conducted to generate multiple solutions, it is hard to obtain a good diversity of the solutions. Criterion-based approaches are often only applicable for problems with two objectives. Consequently, this section concentrates only on dominance-based methods. These methods also comprise the majority of approaches discussed in [Eng06].
The update equations of PSO can be applied for both single-objective as well as multi-objective optimization (although it is possible to change them or to include additional operators like done in [Coe04] to introduce more diversity). In contrast, the management of personal and neighborhood best solutions has to be varied for multi-objective optimization. Above all, the comparison cannot be based on objective function values as for single-objective optimization. Instead, the dominance relation is used. Furthermore, the goal is not to generate a single solution but an approximation of the Pareto-optimal front, meaning that multiple solutions have to be produced.
For this purpose multi-objective PSO algorithms usually maintain an archive of non-dominated solutions. Mostly, the *gbest* neighborhood is used for multi-objective PSO algorithms, thus the archive is a substitute for the global best position in single-objective optimization: During the application of the update equations, a solution from the archive is selected that serves as the global best solution (also called global guide). There are several issues connected with the use of an archive [Bar03, Pul04b, Alv05, Coe04, Li03]:

- Restricted or unlimited archive,
- size of the archive in case of a restricted archive,
- insertion of solutions into the archive,
- deletion of solutions from the archive in case of a restricted archive,
- selection of guides from the archive.

An unlimited archive may store all non-dominated solutions that have been found during the search, having the advantage of a good diversity. A disadvantage is that the computational complexity associated with checking non-dominance and selecting guides increases considerably with growing archive size. Also the required storage space becomes larger. Hence, most multi-objective PSO algorithms use restricted archives.
In case of a restricted archive, the decision which solutions to add to the archive and which ones to delete has to be carefully considered to ensure a good diversity of solutions. Generally, solutions are accepted to the archive if they dominate solutions currently present in the archive or if they are non-dominated with respect to the solutions in the archive. If the archive is already full, new non-dominated solutions will be inserted if they contribute to a better distribution of solutions. Different mechanisms can be used to check the influence of solutions on diversity, e.g. the crowding distance from NSGA-II (see Section 5.2). Similarly, if solutions have to be deleted from the archive because the allowed number of solutions is exceeded, different measures can be applied to decide which solutions to delete from the archive, e.g. an adaptive grid, a distance metric or a randomized distance metric [Bar03].
The easiest way to select a guide from the archive is to use a uniform probability for all solutions. This approach might not be best concerning diversity because the probability is high to select a solution which is located in a region where already many non-dominated solutions have been found. In [Bar03] two other methods are given: The first variant selects guides using an anti-clustering selection that prevents particles from densely populated areas to be selected. The second approach increases the probability of a solution to be selected based on the number of new non-dominated solutions that have been generated using this solution as a guide.
There are algorithms in the literature in which the personal best position is also chosen from the archive [Bar03] but mostly the personal best position is handled differently [Coe02b]. Often only one personal best position is stored that is determined by using the dominance relation: If the current position $\vec{x}_i$ dominates the personal best position $\vec{p}_i$, the personal best position is replaced. If $\vec{x}_i$ is dominated by $\vec{p}_i$, the personal best position does not change. If both are non-dominated, it is randomly decided which to keep [Zie07a, Coe02b]. It is also possible to store several non-dominated personal best positions for each particle but this is seldom done in the literature. One example can be found in [Moo99] (see Section 5.4.1).
After having specified the most important concepts used in multi-objective optimization with PSO, in the following a short overview about multi-objective PSO algorithms used in the literature is given.

5.4.1 Related Literature

Although multi-objective optimization with PSO is still a rather young field, there are already many different implementations. Some examples of them will be given in the following but for a more complete overview the interested reader should refer e.g. to [Pul04b, Coe04, Eng06].
Presumably the first multi-objective PSO algorithm reported in the literature is the one described in [Moo99]. For each particle all non-dominated solutions found by itself are stored as its personal best positions, and guides are chosen randomly from the non-dominated solutions. The *lbest* neighborhood topology is used where the neighborhood

best position is randomly chosen from all non-dominated solutions existing in the neighborhood.
Another one of the first approaches for multi-objective optimization using PSO is described in [Coe02b]. Each solution has one personal best solution that is updated either if the current position dominates the old personal best position or it is randomly decided which to keep if the old personal best position and the current position are non-dominated. The global archive is inspired by the adaptive grid used in PAES: The objective space is divided into hypercubes, and the probabilities of solutions to be deleted from the archive as well as the probabilities of solutions to be chosen as guides depend on the number of solutions in the same cell of the grid, thus ensuring diversity. An improved version of the algorithm has been published in [Coe04] where a method for constraint-handling analog to NSGA-II and a mutation operator is added. The mutation operator was inserted to improve the exploration of the search space. Therefore, it is used more extensively in the beginning of an optimization run whereas its influence is decreased towards the end of an optimization run. In [Coe04] it is stated that the main advantage of the adaptive grid is the lower computational complexity when compared to other methods.
In [Bar03] several ways of deleting members from the archive and selecting guides are examined using methods from Design of Experiments. Restricted archives are recommended for difficult optimization problems. Based on three bi-objective test functions, the authors conclude among others that distance-based metrics are better suited for the deletion of solutions from the archive than an adaptive grid. Furthermore, random selection of guides is recommended.
Random selection of a guide is also done in [Pul04b] (where the guide is called leader). Several subswarms are used because it is argued that convergence may be delayed if guides are selected which are far away from the respective particle. A clustering algorithm is used for grouping the particles into the different subswarms.
The algorithm shown in [Li03] uses concepts of NSGA-II: Instead of only comparing the current position of a particle with the respective personal best position, all personal best positions and current positions are combined, and a non-dominated sorting as in NSGA-II is done. A new swarm is built by subsequently adding particles, beginning with the lowest, i.e. the best, Pareto rank (see Section 5.2). The selection of guides is based on diversity. For this purpose niching methods based on niche count and crowding distance are compared in [Li03] (the term "niching" summarizes techniques that are used for finding multiple solutions in a single run of an optimization algorithm, e.g. they are also employed for multimodal problems to locate different optima [Pre06, Eng06]). As motivation for this algorithm, it is stated that for most multi-objective PSO algorithms the basic form of PSO is used but that the sharing of information in this basic form does not provide enough selection pressure to fulfill the demands of multi-objective optimization.
In [Alv05] the archive is not restricted because it is argued that a restricted archive may lead to oscillation or shrinking of the approximation set. Each particle has one personal best position that is updated if the current position dominates it or if both solutions are non-dominated. The authors state that previous work has shown that distance metrics in objective space may be susceptible to different scalings of the objective functions. Therefore, three different methods for selecting global guides are examined which are only based on Pareto dominance. Moreover, a so-called turbulence factor is added that consists of a small stochastic perturbation that should improve the exploratory capabilities of the particles.

In [Zie07a] (which was generated within the scope of this thesis) a multi-objective PSO algorithm using concepts from NSGA-II for multi-objective optimization is shown. An archive is used that is filled as well as truncated using methods from NSGA-II. Furthermore, this algorithm uses adaptive setting of the PSO parameters w, c_1 and c_2 based on methods from Design of Experiments. This aspect of the algorithm will be discussed in Section 7.2.
It is apparent that the extension of basic single-objective PSO for multi-objective optimization is not a trivial task. Several decisions must be done e.g. regarding strategies for filling and truncating the archive as well as choosing solutions from the archive as guides with respect to the goal of a good diversity. Many PSO variants for multi-objective optimization also incorporate additional mechanisms for improving exploration like a mutation operator in [Coe04] or the turbulence factor in [Alv05]. It can be concluded that a simple, straight-forward implementation of multi-objective PSO may have difficulties when optimizing the complex problems used for testing algorithms [Zie07a] and that additional diversity may be needed [Li03].
In Sections 5.5 and 6.2.4 where a multi-objective PSO algorithm is used, a rather basic form without additional operators for generating diversity is used. A *gbest* version is used with a restricted archive where solutions enter or leave the archive based on methods from NSGA-II, and particles choose their global guides randomly from this archive. Only one personal best position is allowed for each particle as described before.

5.5 Application: Operational Amplifier

In modern semiconductor fabrication processes the need to design and manufacture integrated circuits with a small budget and hard time limits but still with a high performance increases more and more. Usually, there are many specifications that have to be fulfilled simultaneously. Furthermore, the influence of single parameters as well as the relation between parameters becomes more complicated, especially when approaching the boundaries of technical possibilities. Thus, circuit design is a complex and time-consuming task.
During a typical design process the designer first creates a basic design by assigning tasks to various stages. Afterwards, the values of the different components have to be established in a way that the performance matches the specifications for the circuit to be developed. Despite the knowledge and experience of the designer, this is an iterative time-consuming process because of the complexities involved. If optimization algorithms can be applied for assigning values to the parameters, time can be saved and even better results can be found compared to the intuitive approach. In the literature there are works using specialized algorithms based on Genetic Programming where the topology of the circuit is evolved also [Koz04] but here it is assumed that the topology is known. Thus, optimization of parameter values is sufficient, and optimization of the topology is not regarded further.
Because usually several specifications exist, the application of single-objective optimization algorithms to circuit design is complicated. A weighted sum approach can be used but then the disadvantages discussed in the beginning of this chapter apply: It is difficult to choose appropriate weights for the objectives. In [Bol03] the weights are adjusted during the optimization run but more freedom of choice for the decision maker can be

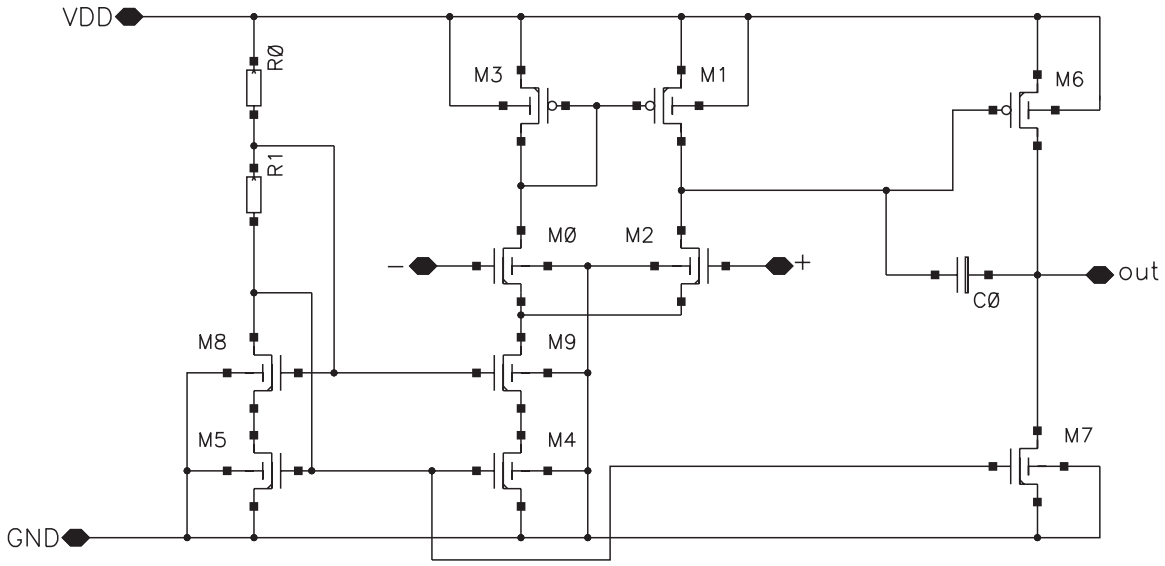


Figure 5.6: Operational amplifier

reached by applying multi-objective optimization algorithms. The generated approximation set offers multiple alternatives from which the decision maker may select a solution for manufacturing.

In the following the optimization of an operational amplifier according to Figure 5.6 will be shown that was also published in [Pal08]. In this example a fixed topology of the operational amplifier is assumed that has been generated by an analog designer. The parameters of the optimization problem are the widths and the lengths of the transistors, the resistance values of the resistors and the capacitance value of the capacitor. A priori knowledge about current mirrors is utilized by combining the widths and lengths of the correspondent transistors into one parameter, respectively. This applies to M0 and M2, furthermore M1 and M3, and also M4, M5, M8 and M9. Thereby, the number of parameters is reduced as it is also done in other circuit design applications [Koz04]. As a result, the optimization problem has a dimension of $D = 13$.

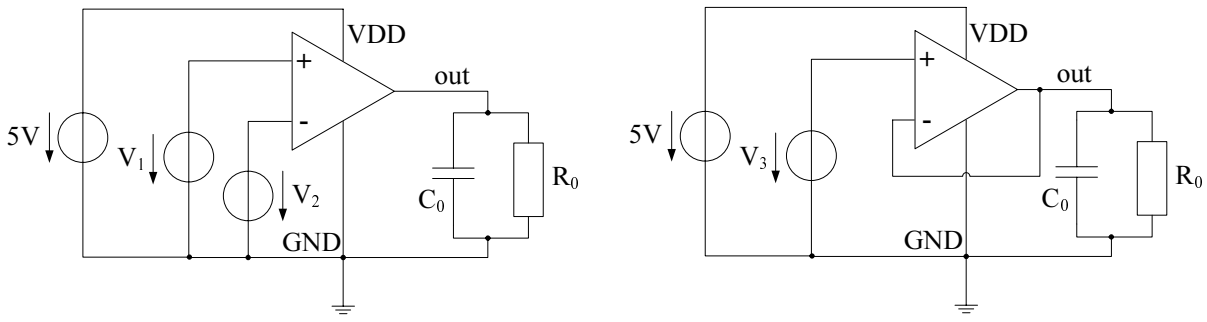
The model of the operational amplifier can be simulated using the tool Spectre which is a part of the Design Framework II distributed by Cadence. In this work the interface is realized via script files. The optimization tool based on DE and PSO writes parameters to a script file which is afterwards called to start a simulation. The output of the simulations (i.e. the values of the objective and constraint functions) is written to a file that is read by the optimization tool. The information is processed and new parameter values are generated depending on the optimization algorithm that is used.

Many objectives can be selected for being optimized [Pal08]. For the present exemplary test case only some of the most important characteristics of the operational amplifier have been chosen which are the overall amplifier gain, bandwidth and current consumption. The gain and the bandwidth should be maximized and the current consumption should be minimized. The gain (in dB) is evaluated by measuring the output voltage at the frequency 1Hz (the operational amplifier has low-pass characteristics). For an assessment of the bandwidth the 3dB cut-off frequency is considered. The current consumption is computed as the difference of the current at the output node and at the supply node while a load is connected to the output node (the load will be specified in the following).

An essential demand for the operational amplifier is stability. It can be checked by observing the phase margin that must be larger or equal to 20°. Thus, the phase margin is included in the optimization problem as a constraint.

The output signal of the operational amplifier should have the same form (e.g. sinusoidal) as the input signal in unity-gain configuration (see also next paragraph and Figure 5.7(b)). In preliminary tests solutions generated during an optimization run often exhibited distorted signals. As a result, the form of the signal must also be taken into account in the optimization problem. No single parameter was found that is able to correctly reflect the form of the signal, therefore a combination of several parameters is used. Fourier transform of the output signal is performed by the simulator, and the magnitude of the result at the frequencies 0 and 10kHz should be maximized. This equals the constant component as well as the frequency of the desired sinusoidal signal. All other frequencies should be suppressed, i.e. their magnitude should be minimized. In addition, the integrated difference between the output and the input signal is minimized (this is possible due to the unity-gain configuration that causes the gain to be equal to 1). To keep the example simple, these variables are not included as objective functions but as constraints into the optimization problem.

To check all these characteristics, two simulations have to be executed. An AC simulation is needed to determine the gain, the bandwidth and the stability. For this purpose the configuration shown in Figure 5.7(a) is used. V_2 only has a DC component of 2.5V to simulate the operational amplifier in the center of its operating range. V_1 has a DC component of 2.5V and an AC component with an amplitude of 1V. The frequency is changed during the simulation from 1Hz to 30MHz. After the AC simulation has been done, the phase margin is evaluated. To save computational cost, a transient simulation is only done afterwards if the phase margin indicates stability of the circuit. The transient simulation is used to determine the current consumption as well as the characteristics that are responsible for the form of the signal. For the transient simulation the unity-gain configuration shown in Figure 5.7(b) is used. In that case a feedback loop is created, so that the gain corresponds to 1. V_3 has an offset of 2.5V and a sinusoidal signal with an amplitude of 2.5V, so that the signal covers the whole range of the supply voltage. The frequency of 10kHz has been chosen because in that case the phase difference between input and output is small, enabling an easier analysis. For both the AC and the transient simulation the load consists of a resistor $R_0 = 700\text{k}\Omega$ in parallel to a capacitor $C_0 = 1\text{pF}$.



(a) For AC simulation

(b) For transient simulation

Figure 5.7: Circuit configuration

As the indicator of optimization quality, an original design specification is regarded as reference for which the parameters have been tuned by an experienced designer based on his knowledge. The main task is to show how the optimization algorithms can be used for the same task and to check whether they are able to provide similar or even better results. Consequently, the constraints which should ensure the sinusoidal form of the output signal are formulated in a way that the results must be at least as good as the ones from the original design. For the objectives the reference values are as follows: A gain of 52.66dB, a bandwidth of 3.031kHz and a current consumption of 14.9μA (in the original design, the current consumption was only 4.9μA but later it was decided to allow larger currents).

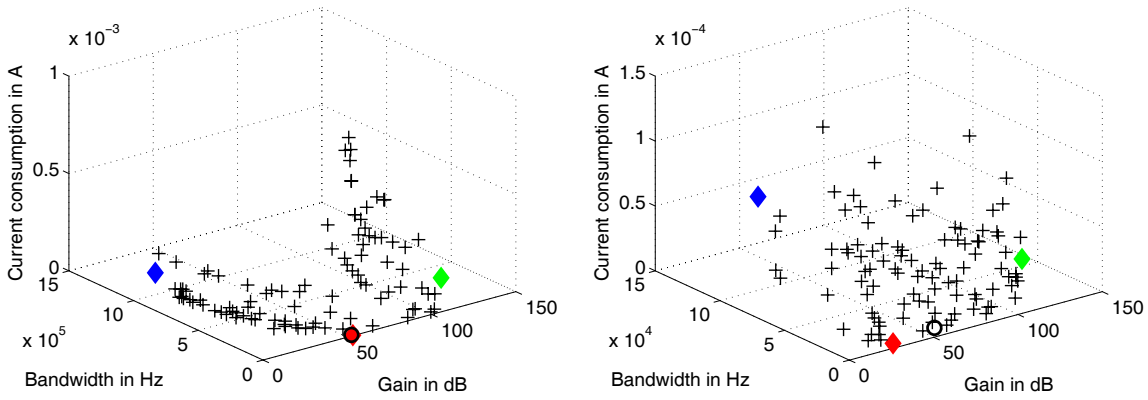
Because the objective functions are computationally very expensive, only few optimization runs were carried out. It should be noted that these runs are examples, and the performance might vary for different runs because of the randomness involved in the evolutionary process. For the same reason, none of the performance measures given in Section 5.1 are applied, and no statistical analysis of the results has been carried out. Instead, figures of non-dominated solutions are displayed that show the principle suitability of DE and PSO for the given optimization problem.

Because the application of single-objective optimization methods is easier, it is worth a thought if problems with several objectives can be converted into single-objective optimization problems. The possibility of building a weighted sum of the objective functions has already been discarded because the selection of the weights is difficult and minimal variations may lead to very different results [Deb01a]. For circuit design frequently boundary values for certain properties are given, e.g. the bandwidth of the circuit must not be below a specific value. Therefore, instead of optimizing all properties of a circuit, it is also possible to take these boundary values as constraints into the optimization problem and select only one property that is of particular importance for being optimized, thus generating a single-objective optimization problem. In the literature also approaches can be found where the optimization run is started with many objectives and the goals are changed during the search depending on intermediate results and user preferences [Fle05], e.g. goals can be excluded from the search or transformed into constraints. This approach requires considerable interaction with the user, so it is not regarded further here.

Because the objective and constraint functions of this optimization problems are computationally expensive, it was unacceptable to find suitable control parameter settings by testing several parameter combinations. Instead, adaptive DE and PSO variants have been used for both single-objective and multi-objective optimization. These variants are described in Section 7.2 and in [Pal08]. For DE the control parameters F and CR have been adaptively controlled. The population size has been set to $NP = 20$ for single-objective optimization whereas $NP = 100$ has been used for multi-objective optimization. For PSO the control parameters w, c_1 and c_2 have been adaptively controlled. For single-objective optimization a population size of $NP = 40$ and the *von-Neumann* neighborhood topology was used whereas for multi-objective optimization $NP = 80$ and the *gbest* neighborhood topology was employed. For both multi-objective DE and PSO the archive size was set to 100.

When optimizing the operational amplifier design with single-objective DE, a gain of 71.17dB was achieved. This is almost 20dB more than in the original design. With single-objective PSO an even higher value of 73.68dB was reached. Interestingly, even after 200 generations when the neighborhood best positions have already converged to one

part of the search space, there are many infeasible particles in the population, indicating that the optimum is located at or near the border between feasible and infeasible space. Unfortunately, nothing is known about the shape of this border. There might even be several disjoint feasible regions. It might be the ability of the particles to get into feasible space that resulted in the better gain value found by PSO in contrast to DE (see also Sections 4.2.2 and 4.2.3): Both DE and PSO are basically able to cross infeasible space but DE individuals can do it only if vector differences of the current population allow reaching a feasible point because a feasible individual is not allowed to become infeasible. In contrast, PSO individuals can become infeasible at any time, so they are able to cross an infeasible region in a different way than DE individuals.
The non-dominated solutions reached by the multi-objective DE and PSO algorithms are shown in Figure 5.8. The circle indicates the original design. The solutions shown in color are the best solutions regarding one objective, respectively: Green means the largest gain, blue means the largest bandwidth, and red means the lowest current consumption. It can be seen that DE as well as PSO were able to find solutions of similar quality as the original design. Additionally, the non-dominated solutions include a large variety of other trade-off solutions from which a decision maker can choose an appropriate design, thus giving more freedom of choice to the user.



(a) Results for DE

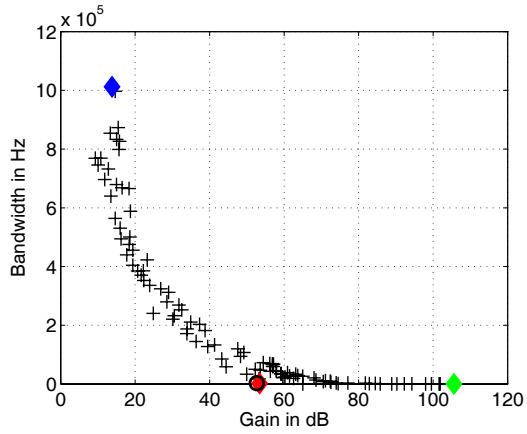
(b) Results for PSO

Figure 5.8: Non-dominated solutions for the operational amplifier in Figure 5.6

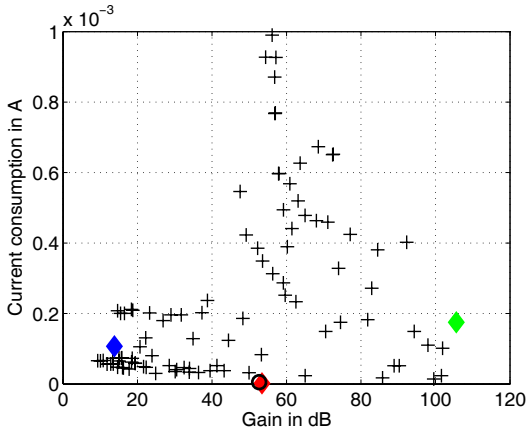
DE yielded even better results than PSO (note also the different scaling in Figures 5.8(a) and 5.8(b)). The best solutions regarding each objective are given in Table 5.2. From Figure 5.8 and Table 5.2 it becomes apparent that e.g. a high gain usually means that the bandwidth will be relatively low and vice versa. This relation can be seen even better

Table 5.2: Extreme solutions for Differential Evolution

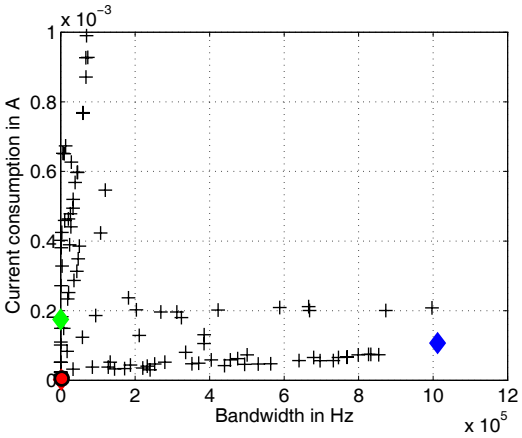
Color in Figure 5.8(a)	Gain	Bandwidth	Current consumption
Green	**105.6dB**	27.8Hz	175μA
Blue	13.8dB	**1.012Mhz**	107μA
Red	53.4dB	1.82kHz	**1.7μA**



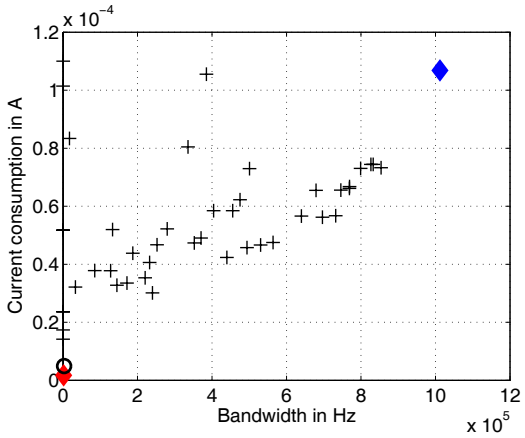
(a) Gain and bandwidth



(b) Gain and current consumption



(c) Bandwidth and current consumption



(d) Bandwidth and current consumption (zoomed)

Figure 5.9: Two-dimensional plots of the solutions using Differential Evolution

in Figure 5.9(a) where only gain and bandwidth are shown for this design. It is clearly visible that there is a strong dependence between gain and bandwidth. The reason is that a high gain can be achieved if the transistors of the output stage are large. This also means that there will be a large capacitance, leading to a small bandwidth.

The connection between gain and current consumption does not seem to be that distinct (see Figure 5.9(b)). For example, an average gain of 60dB can be realized with a relatively high and also with a relatively low current consumption. The reason is that the gain can be influenced by both the input and the output stage of the operational amplifier. Depending on the stage that is mainly responsible for the gain with a certain parameter set, the current consumption will be different. If only the lower right corner of Figure 5.9(b) is regarded, it can be seen that there is still a distinct difference between the best gain of 105.6dB that has a current consumption of 175μA and another solution that still has a gain of 99.5dB but that has a current consumption of only 14μA.

If all solutions are regarded, a definite connection between the bandwidth and the current consumption cannot be seen (see Figure 5.9(c)). However, if only solutions with a current consumption of lower than 120μA are displayed (see Figure 5.9(d)), a trade-off between bandwidth and current consumption is clearly visible. A similar constraint may also

already be inserted into the optimization process because usually only solutions in a certain range are acceptable for a specific application. Thus, characteristics like the gain are still treated as objectives but with an additional constraint.
It can be seen that the optimization of the operational amplifier is a complicated task because there are complex interactions, and the properties of the operational amplifier can be influenced by different stages. Therefore, it is important to support the designer with optimization algorithms which are able to provide trade-off solutions from which the designer can choose the final solution to be manufactured. It has been shown that Differential Evolution and Particle Swarm Optimization are well suited for the application in circuit design. Here only an exemplary test case has been shown for which a rather small number of objectives has been regarded. In future work more objectives should be included in the optimization process. For that case also better visualization techniques are needed as presented in [Adr07] to allow a concise overview about optimization results. As can be seen in Figure 5.8, even for three objectives the display of results may be enhanced. Furthermore, the coupling between the optimization tool and the simulation tool may be further improved to speed up the optimization process.

5.6 Summary and Future Work

It was shown in this chapter that many questions arise when extending single-objective optimization algorithms for multi-objective optimization. Due to the presence of several objectives, a simple comparison of objective function values is no longer possible to evaluate the performance of solutions. Instead, more complicated performance measures are needed which transform the multi-dimensional solutions into an easier assessable form.
From all multi-objective optimization algorithms which have been developed so far, NSGA-II is probably the one which has the largest impact. Many algorithms incorporate concepts from NSGA-II, and algorithm comparisons generally include results from NSGA-II. In this chapter four variants of multi-objective Differential Evolution based on NSGA-II have been compared that differ in the selection scheme and in the assignment of crowding distance. It was indicated that it is preferable to use the original DE selection scheme instead of adding all trial vectors to the population, so elitism is less pronounced and diversity is increased. Because of the already increased diversity, no conclusive statements about the effects from the adapted crowding distance calculation could be made for the DE selection scheme. It is assumed that effects will only be visible for more complicated optimization problems. In contrast, improved results could be seen when using the adapted crowding distance calculation for the NSGA-II selection scheme, showing that this method is indeed able to increase the diversity.
For Particle Swarm Optimization even more issues appear when extending it for multi-objective optimization because it has to be defined what the personal best position and neighborhood best position mean in a multi-objective context. Usually archives are used to store non-dominated solutions but still it has to be decided which solutions to include in the archive, which ones to delete if a restricted archive is full, and which ones to use as global best positions in the update equations. Besides, it seems to be beneficial to include additional operators to improve diversity for optimizing complex problems.
Nevertheless, it has been shown that even without additional operators a multi-objective PSO and also a multi-objective DE algorithm can be successfully used for the optimization

of an operational amplifier. The quality of a reference design that was tuned by an experienced analog designer has been reached by both algorithms. Additionally, a large range of other non-dominated solutions has been generated, meaning that these optimization algorithms are able to offer considerable freedom of choice to a decision maker.
Multi-objective optimization is a very active research topic. Similar to constrained optimization, there are also dissertations in the literature which exclusively treat multi-objective optimization [Zit99, Vel99, Kno02, Fie03], thus only some issues could be shown here. Multi-objective optimization techniques still need to be improved to be able to cope with the complex problems that arise e.g. in engineering. Besides diversity issues (which may be regarded in objective space but depending on the application also in parameter space [Kuk06b, Deb01b]), the focus should be on methods for dealing with computationally expensive objective functions [Hug06]. Furthermore, optimization problems with more than two objectives (also called many-objective optimization) provide even more challenges than the bi-objective problems which are often regarded in the literature due to easier visualization [Adr07]. However, there are visualization techniques like parallel coordinates that allow graphical illustrations of results even for many more than two objectives [Adr07]. In the context of many-objective problems, it is also important to examine how objectives can be excluded or transformed into constraints depending on intermediate results of the optimization process to decrease the problem complexity [Fle05]. Also the further examination of appropriate performance measures is important. Much work has been done already [Zit03] but often the performance measures are still difficult to evaluate, e.g. different (and also contradicting) results can be obtained [Kuk05, Tuš07].

Chapter 6

Stopping Criteria

After initialization, evolutionary algorithms work iteratively by successively generating new generations of solutions (see Figures 3.5 and 3.10). Usually, they do not have a built-in mechanism for terminating an optimization run (in contrast to some other optimization algorithms [Pet01]). Thus, "external" stopping conditions are needed which cause the execution of an algorithm to end.

Even the performance of a very good optimization algorithm may be bad for practical purposes when it is not stopped at a proper time. Usually the primary goal for the application of optimization algorithms is convergence to the global optimum and the secondary goal is to use the least computational effort. However, the detection of convergence to the optimum is not necessarily a trivial task, especially if real-world problems are optimized for which no knowledge about the global optimum is available. As a consequence, it is not easy to decide when the execution of an optimization algorithm should be terminated. If the execution of the algorithm is stopped too early, convergence may not yet been reached. In contrast, if the algorithm is terminated too late, computational resources are wasted. Because real-world problems usually contain computationally expensive objective and constraint functions that may result in optimization runs which take several days, it is imperative that unnecessary function evaluations are avoided. In that case, it is reasonable to terminate an optimization run if it is expected that hardly any additional improvement may be achieved. In other words, the ratio of solution quality to calculation time should be maximized [Rud04].

This is especially important if many scenarios which vary in their parameter settings should be optimized. An example is the power allocation problem presented in Section 4.2.4. It is interesting to have several results in dependence on the target bit error rate to meet the different quality requirements that result from diverse applications like voice or multimedia services. Furthermore, if different distributions of user locations should be regarded in a Monte Carlo simulation, many optimization runs are needed, so wasting of computational resources has to be prevented.

If an optimization run is monitored by an experienced user of optimization algorithms and the available information is detailed enough, an appropriate time for stopping the run may be derived by observing the development of several characteristics which will be discussed in the following. However, for the integration of optimization algorithms into automatic design processes a proper time for terminating a run must be detected automatically.

For theoretical work about convergence properties or for a comparison of different algo-

rithm implementations the definition of stopping criteria is generally not difficult because for this purpose usually test functions with known optima are employed. In that case, the execution of the algorithm can be terminated if the optimum is found with a given accuracy, and the involved computational effort can be used to analyze the performance of different algorithms. An alternative is to terminate after a defined number of function evaluations and to evaluate the distance of the best individual to the optimum. This approach works well for theoretical work when algorithm variants are tested against each other but for real-world problems the situation is different because the optimum is usually unknown.

A stopping rule for problems with unknown optimum that is widely used in the literature is to terminate the execution of an algorithm after a given maximum number of function evaluations FE_{max} (called *LimFuncEval* in the following). This approach is associated with two problems: Suitable settings for FE_{max} vary considerably for different optimization problems. Generally no correlation can be seen between features of an optimization problem and the required number of function evaluations, so usually FE_{max} has to be figured out by trial-and-error methods. Besides, features of the optimization problem are not necessarily known, particularly for real-world problems. A further problem is that the number of objective function evaluations FE_{conv} that is needed for convergence for one specific optimization problem may also be subject to large variations due to the stochastic nature of EAs. This statement holds for many different implementations of DE and PSO as can be seen in [Hua06, Tak06, Tas06, Bre06c, Kuk06c, Mez06b, Zie06a] for DE and in [Lia06b, Muñ06, Zie06b] for PSO. All these papers have been presented during the CEC06 Special Session on Constrained Real Parameter Optimization that has already been discussed in Section 4.4. The range of FE_{conv} can be assessed by regarding the standard deviation of function evaluations for reaching a predefined error measure. This measure was often in the range of $10^3 - 10^4$ but also up to 10^5. As a consequence, a considerable safety margin of function evaluations must be included to ensure that the optimum is found before termination of the algorithm if criterion *LimFuncEval* should be used.

Due to these reasons, it is important to examine other alternatives for stopping the execution of evolutionary algorithms besides termination after a fixed number of function evaluations. In order to deal with the problem that is caused by fluctuations of FE_{conv}, the stopping criteria have to be able to detect when convergence is reached. Thus, they have to react adaptively to the current state of an optimization run. The stopping criteria have to ensure that the algorithm is executed long enough to obtain convergence to the global optimum but without wasting of computational resources. An example is shown in Figure 6.1 where a typical development of the best objective function value $f(\vec{x}^*)$ is given over time (using a DE algorithm for function g02, see Section 4.4). Although the best function value stagnates for some time around $30,000$ function evaluations, the optimization run should be continued because better results can still be achieved. An ideal stopping time would be around $60,000 - 70,000$ function evaluations when it is assured that the best function value will not improve further. Stopping after $70,000$ function evaluations could be considered as wasting computational resources in this case. These problems with temporary stagnation are not limited to DE but may also arise using other algorithms, e.g. in [Jak04] similar problems are reported for an EA.

Unfortunately, it seems to be impossible to define a stopping criterion without introducing one or more parameters. The parameter settings generally depend on the given optimiza-

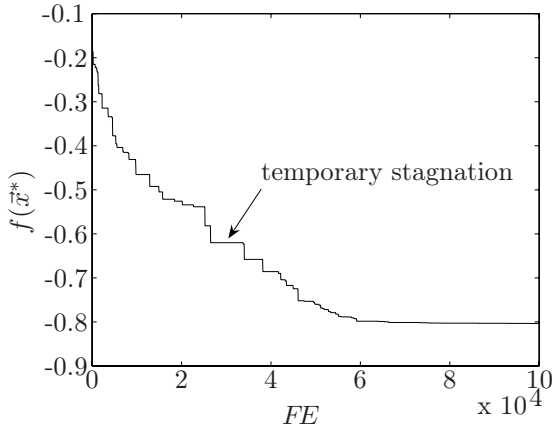


Figure 6.1: Exemplary optimization run

tion problem. In the following it is investigated if there are stopping criteria for which the parameter settings are robust for different optimization problems or if parameters can be set depending on certain aspects of the problem. It is assumed that the general behavior of different optimization problems to stopping criteria is similar. It should be kept in mind that limiting the number of function evaluations as a stopping criterion also incorporates the choice of a problem-dependent parameter FE_{max}. Even if a problem with known optimum is regarded and an optimization run should be terminated when the optimum is found with a certain accuracy, the accuracy has to be set by the user. Therefore, it is favorable to examine other possibilities for stopping that contain the advantage of reacting adaptively to the state of the optimization run. In that case, a large range of the number of function evaluations for convergence FE_{conv} does not lead to a high computational overhead because of a cautiously set safety margin.

Stopping criteria for single-objective optimization have to be different from the ones for multi-objective optimization because the goals are also different. In single-objective optimization usually one global optimum should be found (there are some exceptions where all optima of multimodal functions are wanted [Tho04, Zah04] but that is not regarded further here) whereas in multi-objective optimization usually several Pareto-optimal solutions should be generated. Therefore, the populations develop differently, so stopping criteria for multi-objective optimization have to rely on different measures than stopping criteria for single-objective optimization.

Despite the importance of stopping criteria for real-world problems, not much information about this topic is available in the literature. It is not a new topic as e.g. already in 1977 this topic was addressed for single-objective optimization in [Sch77]. Nevertheless, few has been done in the meantime, and even less for multi-objective optimization. In particular, an overview about stopping criteria is missing that evaluates the available literature and classifies the stopping conditions based on the property that they observe.

It is stated in [Sch95] that the effectiveness of a stopping criterion is closely related to the procedure of a certain optimization strategy and not automatically transferable to other algorithms. Nevertheless, there is few effort in the literature to transfer former findings from older optimization algorithms to new ones like DE and PSO and to compare the outcomes.

Therefore, in the following a large variety of different implementations of stopping criteria

is discussed for single-objective as well as multi-objective optimization. A classification is shown that is based on the property that is considered for deriving conclusions about the optimization run. This classification has been developed within the scope of this thesis. Considering the focus of this work, stopping criteria will be regarded which can be used for DE and PSO but the applicability for other EAs is also discussed. Because there is not much literature concerning stopping criteria for DE and PSO, some references are also given for other EAs, especially GAs. Furthermore, the performance of a large number of stopping criteria is evaluated for both single-objective and multi-objective optimization.

6.1 Single-Objective Optimization

Different characteristics can be used for deriving conclusions about the current state of an optimization run. In principle any phenomenon can be used that exhibits a definite trend from the beginning to the end of an optimization run. For instance both the improvement as well as the movement of individuals are typically large in the beginning of an optimization run and both become small when approaching convergence. Another property that may be regarded is the distribution of population members: The individuals are scattered throughout the search space initially but usually converge to one point towards the end of an optimization run. Consequently, each of these features is basically usable for detecting convergence.
Any of the before-mentioned population characteristics like improvement, movement and distribution can be used in various implementations for the creation of stopping conditions but the performance of different implementations is not necessarily similar. Therefore, different alternatives for measuring population characteristics are introduced in the following. The criteria are grouped into several classes based on the property which they regard for detecting convergence. After the presentation of the criteria, an assessment of their performance for DE and PSO is given. The performance assessment is a summary of several examinations which were done within the scope of this thesis and which are presented in [Zie05b, Zie05c, Zie06e, Zie06d, Zie07c, Zie08c].

6.1.1 Classification

The stopping criteria which are discussed in this work have been grouped into six classes:

- Reference criteria,
- criteria based on limited resources,
- improvement-based criteria,
- movement-based criteria,
- distribution-based criteria,
- combined criteria.

In the following these classes are described in detail. Where possible, the stopping criteria are visualized for a simple case where the objective function is only dependent on one parameter.

6.1.1.1 Reference Criteria

Reference criteria refer to stopping conditions which make use of the information about the global optimum. Thus, they are only applicable if the global optimum is known. This is usually only the case for test functions but not for real-world problems. There are exceptions where information about the optimum is available, e.g. due to physical limits [Pri05]. Apart from these exceptions, reference criteria are mostly used for the assessment of convergence properties of an optimization algorithm or for comparisons of several algorithms.
If the optimum is known, an optimization algorithm may be terminated when the optimum is reached with a specified accuracy ϵ_g. Mostly, it is regarded as sufficient if one individual fulfills this condition, i.e. the difference of its objective function value to the global optimum is $\epsilon < \epsilon_g$ (see Figure 6.2). For constrained problems it must also be assured that the respective individual is feasible.

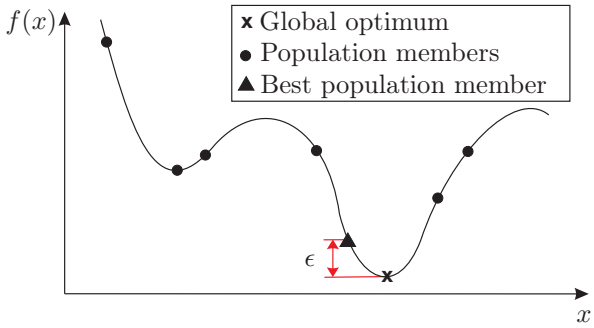


Figure 6.2: Reference criteria

As it is possible that the optimum is found because an individual was initialized close to it by chance, it might be argued that this condition makes no reliable statement about convergence properties of the algorithm. In that case it might be demanded that a certain percentage of the population is located near the optimum before terminating a run. Of course this only makes sense when assuming that all or at least most of the population members will converge to the optimum eventually but that is usually true for DE and PSO. However, especially for increasing search space the probability of initializing a position within the given accuracy of the optimum is very low.

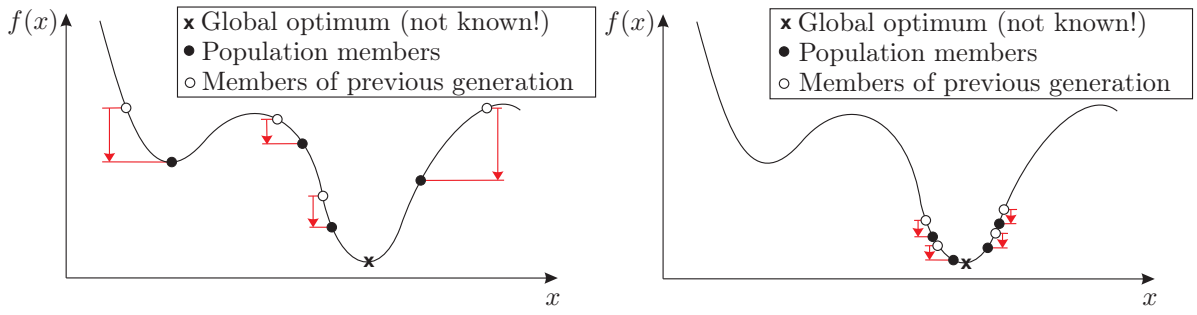
6.1.1.2 Criteria based on Limited Resources

In the literature a commonly used termination criterion is to stop after reaching a specified number of generations G_{max} or a maximum number of objective function evaluations FE_{max} (*LimFuncEval*). It is also possible to stop if a certain CPU time has been spent. These criteria are often used for comparisons: Algorithms are executed several times using a defined budget of e.g. function evaluations, and afterwards statistics like the average, minimum and maximum objective function values are checked. Additionally, they are often used for real-world problems. The disadvantages associated with stopping after a fixed number of function evaluations have already been discussed in the beginning of this chapter to motivate the use of adaptive stopping criteria, so they are only summarized shortly here: The number of function evaluations needed for convergence FE_{conv} is strongly dependent on the objective function, thus a suitable setting for FE_{max} has to be found

using trial-and-error methods, and it also has to be considered that FE_{conv} may have a large range.
There are also cases for which the use of this criterion makes sense (apart from algorithm comparisons as described above), e.g. there may be applications where it is necessary to terminate after a certain time has been spend. This may be the case for on-line (real-time) applications or if computer time is limited [Pri05]. Furthermore, it might be reasonable to use a maximum number of generations G_{max} or a maximum number of function evaluations FE_{max} in combination with the adaptive stopping criteria introduced in the following to prevent the algorithm from running forever if a criterion is not able to stop the run. In that case, a large setting should be used for G_{max} or FE_{max}, respectively, because it is only used as a precaution.
In the literature also stopping criteria based on a limited number of generations can be found where an appropriate setting of G_{max} has been theoretically determined. For example, in [Ayt00] an upper bound on the number of iterations required to reach the global optimum with a certain level of confidence is derived for a GA. Thus, depending on the desired confidence, computational resources will be assigned to the algorithm. However, in [Saf04] it is stated that this stopping criterion is of little practical interest because for a purely random algorithm a bound can be calculated that corresponds to the result for GAs. Even without this limitation this criterion would still exhibit the problem that computational resources may be wasted due to fluctuations in the number of function evaluations necessary for convergence because no attempt is made to adaptively detect the time when convergence is obtained.

6.1.1.3 Improvement-based Criteria

In the beginning of an optimization run usually large improvement in terms of objective function values is achieved by the individuals (see Figure 6.3(a)). In contrast, the improvement becomes small during later stages of an optimization run when individuals converge towards the global optimum (see Figure 6.3(b)). As a consequence, it can be assumed that convergence has been obtained if the improvement of the objective function value decreases to a small value. Because improvement and movement do not necessarily occur continuously and there may be stages without any improvement although the optimum has not yet been found like shown in Figure 6.1, these measures should be monitored for several consecutive generations. Because improvement can be measured in different ways,



(a) Beginning of an optimization run: Large improvement

(b) End of an optimization run: Small improvement

Figure 6.3: Improvement-based criteria

three conditions are examined here:

- *ImpBest*: The improvement of the best objective function value Δf^*_G is monitored for each generation G:

$$\Delta f^*_G = \frac{f(\vec{x}^*_{G-1}) - f(\vec{x}^*_G)}{|f(\vec{x}^*_{G-1})|} \tag{6.1}$$

  where $f(\vec{x}^*_G) = \min_i f(\vec{x}_{i,G})$ is the best solution of the current generation G and $f(\vec{x}^*_{G-1})$ is the best solution of the previous generation $G-1$. The improvement is defined for minimization problems (see also Section 2.1), thus improvement will be positive if $f(\vec{x}^*_G) < f(\vec{x}^*_{G-1})$. It has to be noted that the improvement cannot be calculated if $f(\vec{x}^*_{G-1}) = 0$, so this case has to be handled differently. If also $f(\vec{x}_G) = 0$, the improvement can be set to 0. In other cases the absolute difference $f(\vec{x}^*_{G-1}) - f(\vec{x}^*_G)$ may be checked.

  If the improvement falls below a user-defined threshold t for ΔG consecutive generations, the optimization run will be terminated:

$$\Delta f^*_g < t \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.2}$$

  In case of a constrained optimization problem, *ImpBest* must be modified because the best solution might be infeasible. The modification can be done as follows: If the best solution is feasible in the current generation but it was infeasible in the previous generation, the improvement is assigned an arbitrary high number because there was definitely an improvement, so the algorithm should not stop. If an individual is infeasible in both the current and the previous generation, the improvement is calculated based on the constraint violation according to Equation 4.3.

  A similar approach is also discussed e.g. in [Ber01] for Particle Swarm Optimization and furthermore in [Syr95] to determine a suitable switch-over point from a Genetic Algorithm to a local optimization technique.

- *ImpAv*: Because the best objective function value might not correctly reflect the state of the whole population, the average improvement $\overline{\Delta f}_G$ computed from the whole population is examined for this criterion:

$$\overline{\Delta f}_G = \frac{1}{NP} \sum_{i=1}^{NP} \frac{f(\vec{x}_{i,G-1}) - f(\vec{x}_{i,G})}{|f(\vec{x}_{i,G-1})|}. \tag{6.3}$$

  Similar to *ImpBest*, the improvement is positive if the objective function values decrease from generation $G-1$ to G and furthermore the case $f(\vec{x}_{i,G-1}) = 0$ must be intercepted. An optimization run is terminated if the average improvement is below a given threshold t for ΔG generations:

$$\overline{\Delta f}_g < t \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.4}$$

  For PSO either the current positions $\vec{x}_i$ or the personal best positions $\vec{p}_i$ can be taken as basis for *ImpAv*. In early examinations that have been done within the scope of this thesis, the current positions have been used [Zie05c]. In [Zie07c] the personal best positions have been employed instead because the current positions

have many fluctuations whereas the development of the personal best positions is more smooth, so decisions about termination of an optimization run can be made more reliably. This way the application of this criterion to PSO is also more similar to its application to DE. Thus, it might be assumed that suitable parameter settings are more alike.

For constrained optimization the same modification can be done as described for *ImpBest*: The transition from infeasibility to feasibility is rewarded with an arbitrary high number and the improvement for infeasible individuals is calculated on the basis of constraint violation.

This criterion is also used in [Esp03] to stop a local search procedure that is embedded in a Genetic Algorithm. For the same purpose similar criteria as *ImpBest* and *ImpAv* are also employed in [Vas97]. Generally, criteria like *ImpBest* and *ImpAv* can be used for many optimization algorithms. They are frequently found in the literature although sometimes it is not expressed clearly if the best or the average fitness is regarded.

- *NoAcc*: This criterion is slightly different for DE and PSO because it is based on the internal working of the algorithms. Therefore, it has to be checked whether it can be adapted if it should be applied to different optimization algorithms.

  Because DE incorporates a greedy selection scheme, the acceptance of trial vectors means that there is improvement in the population. Based on this fact, it is monitored if at least one trial vector has been accepted in a specified number of generations ΔG. The optimization run is terminated if this condition is violated:

  $$|\{\vec{x}_{i,g} | i \in \{1, \ldots, NP\} \text{ and } \vec{x}_{i,g} \neq \vec{x}_{i,g-1}\}| = 0 \quad \forall g \in \{G - \Delta G, \ldots, G\}. \tag{6.5}$$

  For PSO *NoAcc* is adapted by observing if at least one new personal best position has been found in a predefined number of generations [Zie07c]. Note that in early work that was done within the scope of this thesis, it has been checked for PSO if there is still improvement in any neighborhood [Zie05c]. The formulation based on personal best positions has the advantage that it is more similar to the DE formulation.

  NoAcc uses the information whether an individual has been accepted for DE or if a personal best position has been updated for PSO. Because the constraint-handling method is included in the selection scheme and in the update of best positions, for constrained optimization no explicit adjustment has to be done for *NoAcc*.

  An advantage of *NoAcc* is that only one parameter has to be set whereas most other stopping conditions that are presented here require the setting of two parameters.

  NoAcc is also described for DE in [Pri05]. It is recommended to set ΔG not too low because long periods without improvement may occur during optimization runs. It is assumed that this behavior might be more common in DE than in other EAs.

  In [Jak04] two similar criteria are used for determining a suitable time for switching from an EA to a local search technique. A neighborhood structure is defined, and the number of consecutive generations in which no improvement was found in any neighborhood is checked. For another criterion the number of consecutive generations in which no individual was accepted in any neighborhood is monitored.

There are some stopping criteria in the literature which are different from the ones presented here but may also be regarded as improvement-based. For example, in [Rim07] the similarity between the decision when to terminate an optimization run of a GA and the decision when to exercise a call option in economics and finance is utilized to derive a so-called cost-benefit stopping criterion. A stopping boundary is built that considers the termination payoff (the fitness reached) and the cost for continuing the optimization run, thus it can be seen as improvement-based criterion. A disadvantage is that a model of the algorithm behavior must be built for every optimization problem, meaning that the algorithm must be run many times for collecting statistics about the fitness.
Another variant of an improvement-based criterion is presented in [Fal01]. An algorithm is used that represents a type of GA but with a problem-dependent data structure consisting of a vector of integer numbers. A linear function is approximated using the solutions of the last ΔG iterations, and the execution of the algorithm is terminated if the slope of the function approaches zero. $\Delta G = 100$ is used in [Fal01], and the slope must be less or equal to 0.01.

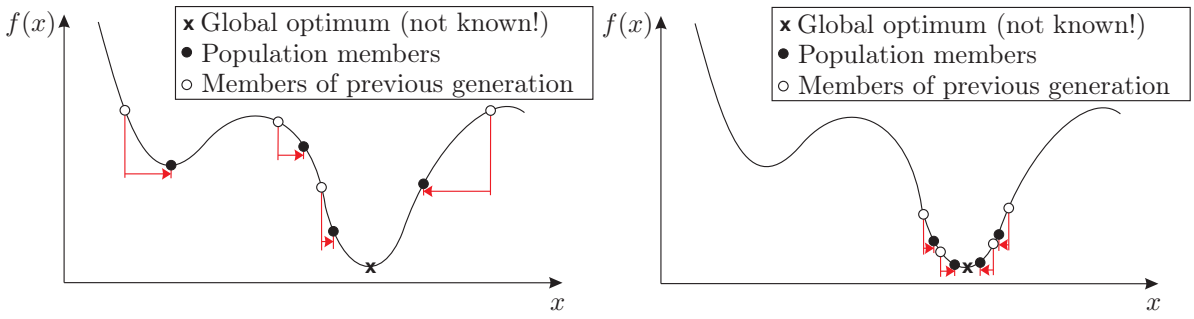
6.1.1.4 Movement-based Criteria

In the beginning of an optimization run the individuals are randomly scattered in the search space. As a result, large step sizes are generated in mutation and recombination for DE because they are dependent on vector differences (see Figure 6.4(a)). Because the neighborhood best solutions might be far away from the current positions, the PSO particles will also have a large velocity. Towards the end of an optimization run the population generally converges to one point in the search space. Thus, step sizes become small because the individuals are close to each other (see Figure 6.4(b)). Consequently, the movement of individuals in parameter space can also be used to derive a stopping criterion:

- *MovPar*: If the average movement of the population members

$$\overline{\Delta x}_G = \frac{1}{NP \cdot D} \sum_{i=1}^{NP} \sum_{j=1}^{D} |x_{i,j,G-1} - x_{i,j,G}| \tag{6.6}$$

  is below a threshold t for a given number of generations ΔG, the optimization run



(a) Beginning of an optimization run: Large movements

(b) End of an optimization run: Small movements

Figure 6.4: Movement-based criteria

is terminated:

$$\overline{\Delta x}_g < t \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.7}$$

In contrast to *ImpBest* and *ImpAv*, it does not make sense to define the movement in relation to the previous value (see Equations 6.1 and 6.3) because parameter space is regarded here.

Similar as for *ImpAv*, this criterion was calculated using current positions for PSO in [Zie05c] but personal best position have been regarded in [Zie07c] because it is assumed that their development is more smooth and therefore better suited for defining stopping conditions.

Criterion *MovPar* does not need any adjustment for constrained optimization because only the changes in positions of individuals are observed, regardless of their feasibility. It is possible to expand this criterion by demanding a certain percentage of the individuals (or at least the best one) to be feasible.

MovPar is also usable for other evolutionary algorithms with real-coded variables. It might be possible to adapt it also for binary-coded individuals if a suitable distance measure can be found. Moreover, stopping criteria like this are used in classical optimization algorithms like hill climbing techniques [Sch77, Sch95].

Movement of individuals can be measured both in parameter and in objective space (see also [Saf04] where for GAs it is distinguished between genotypical and phenotypical termination criteria). Because of the greedy selection scheme of DE, the objective function value can only improve but not deteriorate. As a result, a stopping criterion based on movement in objective space would be equal to an improvement-based criterion. In contrast, a criterion *MovObj* could be used for other evolutionary algorithms, e.g. PSO, which permit deterioration of objective function values. This holds only if current positions are regarded as done in [Zie05c]. If personal best positions are used as in [Zie07c], *MovObj* is also the same as *ImpAv* for PSO.

- *MovObj*: If the average movement of the population members in objective space

$$|\overline{\Delta f}_G| = \frac{1}{NP} \sum_{i=1}^{NP} \left| \frac{f(\vec{x}_{i,G-1}) - f(\vec{x}_{i,G})}{f(\vec{x}_{i,G-1})} \right| \tag{6.8}$$

is below a threshold t for ΔG generations, the optimization run is stopped:

$$|\overline{\Delta f}_g| < t \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.9}$$

For criterion *MovObj* the same adjustment for constrained optimization problems can be done as for *ImpBest* and *ImpAv*.

6.1.1.5 Distribution-based Criteria

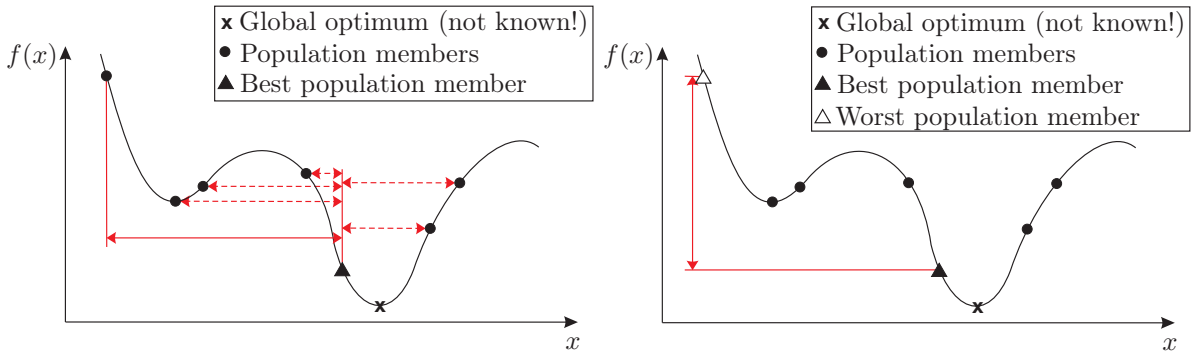
In single-objective optimization the DE and PSO population members usually converge to one point in the search space towards the end of an optimization run whereas they are scattered throughout the search space in the beginning of a run due to the random initialization. As a result, the distribution of individuals can be used to derive conclusions about the state of an optimization run. In contrast to criteria based on improvement

or movement, the distribution of individuals does not need to be checked for several consecutive generations because oscillations are considerably less likely to appear.
Several possibilities exist to measure the distribution of individuals. One of the easiest alternatives is the following:

- *MaxDist*: The maximum distance of any population member to the individual with the best objective function value is monitored in parameter space:

$$\Delta x_{max-best,G} = \max_i \sqrt{\sum_{j=1}^{D} \left(x_{i,j,G} - x^*_{j,G}\right)^2}. \tag{6.10}$$

  Criterion *MaxDist* is visualized in Figure 6.5(a) where the dashed arrows indicate that the distances of all population members to the best individual have to be calculated but only the largest distance (solid arrow) is used for *MaxDist*. It should be noted that the largest distance does not need to be between the best and the worst individual (as it is the case in the figure incidentally) because the shape of the objective function might be different.



(a) Parameter space (b) Objective space

Figure 6.5: Distribution-based criteria

  If the maximum distance falls below a threshold m, the optimization run is terminated:

$$\Delta x_{max-best,G} < m. \tag{6.11}$$

  For PSO this criterion has been checked using current positions in [Zie05c] whereas personal best positions have been used in [Zie07c] to avoid fluctuations.

  For *MaxDist* no adjustment for constraint-handling is needed because only positions are examined, same as for criterion *MovPar*.

  Two similar criteria based on the maximum swarm radius and based on a cluster analysis are also discussed in [Ber01] for PSO. In [Pri05] a slightly different stopping condition is suggested for DE. It ends the execution of the algorithm if the longest vector difference is several orders below the desired accuracy of the optimum. In contrast to *MaxDist* the best solution is not used for this criterion. Instead only the largest distance between two population members is regarded. As a result, *MaxDist* needs less computational effort because the best member is identified anyway, and only differences to the best member are calculated.

If more information should be included than only observing the maximum distance to the best individual, the positions of all population members might be regarded as follows:

- *StdDev*: The standard deviation of positions of all population members is examined:

$$\sigma_{x,G} = \sqrt{\frac{1}{NP-1}\sum_{i=1}^{NP}\left(r_{i,G} - \bar{r}_G\right)^2} \tag{6.12}$$

with the distance $r_{i,G}$ of population member $\vec{x}_i$ to the origin

$$r_{i,G} = \sqrt{\sum_{j=1}^{D}\left(x_{i,j,G}\right)^2} \tag{6.13}$$

and the average position

$$\bar{r}_G = \frac{1}{NP}\sum_{i=1}^{NP} r_{i,G}. \tag{6.14}$$

The optimization run is stopped if the standard deviation of positions drops below a given threshold m:

$$\sigma_{x,G} < m. \tag{6.15}$$

This criterion has also been checked using current positions for PSO in [Zie05c] whereas personal best position were regarded in [Zie07c].

Same as for *MaxDist* and *MovPar* no adjustment for constrained optimization is necessary because only positions are considered.

In [Zah03b] a similar criterion is used for DE. In [Pri05] it is also suggested to monitor the standard deviation of population vectors. For termination of the algorithm it is recommended that the standard deviation should be several orders of magnitude smaller than the desired accuracy of the optimum.

In [Olo08] several measures of particle swarm diversity are given. An evaluation showed that the most suitable one is the average distance around the swarm center. This is very similar to the standard deviation defined here, with the exception that the normalization is slightly different.

Especially the observation of PSO individuals during optimization runs has inspired the following generalization of *MaxDist* in [Zie05c]:

- *MaxDistQuick*: During an optimization run a state may occur where most populations members have already converged to the vicinity of the optimum but some individuals are still searching in regions which are far away. Therefore, instead of examining the maximum distance of all population members to the current best individual, only a subset of the current population is used. For this purpose, the population members are sorted due to their objective function value using a Quicksort algorithm. Only for the best $p \cdot NP$ (with $0 < p \leq 1$) of the population members it is checked if their distance to the best member is below a threshold m.

  In contrast to *MaxDist* that does not need to be adjusted for constrained optimization because only positions are considered, *MaxDistQuick* needs to be adapted. The

reason is the sorting procedure that regards objective function values which cannot necessarily be calculated for infeasible individuals. There are basically two possibilities how to adjust *MaxDistQuick*: Either it is also checked if the best $p \cdot NP$ of the individuals are feasible (this method is used in this work) or the sorting is adjusted, so that feasible individuals are always regarded better than infeasible individuals, and infeasible individuals are sorted according to their constraint violation.

MaxDist can be derived from *MaxDistQuick* by setting p to 100%.

A generalization of *StdDev* can be defined as follows:

- *StdDevQuick*: Similar to *MaxDistQuick*, the population is first sorted due to their objective function value using a Quicksort algorithm. The standard deviation of positions is then calculated for the best $p \cdot NP$ of the population and compared to the user-defined threshold m.

  For *StdDevQuick* the same adjustment for constrained optimization can be done as for *MaxDistQuick*.

  Similar to the relationship between *MaxDist* and *MaxDistQuick*, *StdDev* is a special case of *StdDevQuick* with $p = 100\%$.

All distribution-based criteria that have been mentioned so far are calculated in parameter space. Another possibility to evaluate the distribution of the population members is to regard objective space:

- *Diff*: The difference between best and worst objective function value

$$\Delta f_{max-min,G} = \max_i f(\vec{x}_{i,G}) - \min_i f(\vec{x}_{i,G}) \tag{6.16}$$

  is tested in each generation if it is below a given threshold d (see Figure 6.5(b)):

$$\Delta f_{max-min,G} < d. \tag{6.17}$$

  For constrained optimization it may also be reasonable to demand that at least $p \cdot NP$ of the individuals are feasible. Otherwise *Diff* could lead to early termination of an optimization run if e.g. only two individuals are feasible and they are close to each other by chance but the population has not converged yet. Fortunately, it will be discussed later (see Section 6.1.2) that parameter p may be omitted as there was no dependence on p to be seen in any of the conducted examinations. In the actual implementation it only has to be assured that there are at least two feasible individuals in the population to avoid access to possibly non-initialized memory.

  A similar implementation of this criterion without parameter p is described in [Pri05] and [Bab03a] for DE, and in [Sch77, Sch95] for ESs. Same as for the criteria based on the longest vector difference and the standard deviation, it is recommended in [Pri05] to set d to a value that is several orders of magnitude lower than the desired accuracy of the optimum.

Apart from the already mentioned distribution-based criteria from literature which are similar to the criteria examined during the scope of this work, another stopping criterion that can be regarded as a distribution-based criterion but follows a different approach can

be found in [Her05] for a so-called generic EA. The distribution is not measured in one population but several optimization runs are conducted in parallel using the same random numbers but different initial populations. It is stopped when the populations are equal for all parallel optimization runs (where the search space is restricted to a limited number of discrete points). It is an interesting stopping criterion because in practical applications it is also common to perform multiple runs of an optimization algorithm and to choose the best solution of all runs as the result. However, again the fact is not taken into account that optimization runs might require different numbers of function evaluations for convergence, thus computational resources are wasted.
No specific information about DE or PSO is used for the distribution-based criteria, so in principle they can be used for other algorithms also. Only if another representation than real-coded vectors is used for the positions of the individuals, the distribution-based criteria in parameter space have to be adapted.

6.1.1.6 Combined Criteria

Because functions have different features, it might be concluded that a combination of different stopping criteria may result in good performance. Besides, this is exactly what a human observer would do: Check several characteristics to obtain an overview about the state of the population that is as comprehensive as possible. Therefore, for example an criterion like *Diff* that is easy to check can be tested first. Because the first criterion might fail for certain characteristics of the objective function (e.g. it was shown in former work [Zie05c] that *Diff* fails for functions with a flat surface; this will also be discussed in Section 6.1.2), a second criterion that is based on another mechanism might be evaluated after the stopping condition of the first criterion has been fulfilled. In former work the following combined criteria were tested:

- *ComCrit*: First, the improvement-based criterion *ImpAv* is evaluated. If *ImpAv* indicates that the optimization run should be stopped, the distribution-based criterion *MaxDist* is regarded additionally.

- *Diff_ MaxDistQuick*: In this case, distribution-based criteria in objective and parameter space were joined (*Diff* and *MaxDistQuick*). *MaxDistQuick* is only checked if the stopping condition of *Diff* has been fulfilled.

  This criterion was suggested in [Zie05c] because *Diff* resulted in reliable stopping behavior for all functions except for one with a flat surface. The combination with a distribution-based criterion in parameter space should avoid this behavior.

The combined criteria are composed of several already presented stopping criteria. The same adjustments for constrained optimization can be done for them as for the individual criteria, and the same comments regarding the applicability for other optimization algorithms hold.

6.1.2 Assessment of Performance

Several examinations of stopping criteria for single-objective optimization have been done within the scope of this thesis [Zie05b, Zie05c, Zie06e, Zie06d, Zie07c, Zie08c]. In this section the results of these studies are summarized and evaluated. When reviewing the

results, it must be kept in mind that the conditions were not the same in all examinations. Thus, in the following first some details about the examinations are given before summarizing the findings.

6.1.2.1 Experimental Settings

The first examination about stopping criteria for single-objective optimization that was done within the scope of this thesis has regarded eight unconstrained test problems. The results were published in [Zie05b] for DE and PSO. The examination was extended by a run time analysis of the stopping criteria [Zie05c]. A constrained real-world problem consisting of the power allocation problem with parallel interference cancellation described in Section 4.2.4 has been regarded in [Zie06d] and [Zie07c] for PSO ([Zie07c] is an extended version of [Zie06d] that has been published in a journal) and in [Zie06e] for DE. Furthermore, 16 constrained test problems have been examined in [Zie08c] for DE.

All examinations have been conducted using the variant DE/rand/1/bin but different parameter settings have been applied for DE (see Table 6.1). For PSO not only different parameter settings but also different neighborhood topologies were used (see Table 6.2). Besides the parameter settings, in Tables 6.1 and 6.2 also details of the test problems are specified (from left to right): Number of test problems used in the respective examination, presence of constraints, dimension of test functions, characteristics. Furthermore, the desired accuracy of the global optimum ϵ_g (see also Section 2.3) and some remarks are given. In some of the cited papers not all details of the results could be given due to space limitations. Some of them are presented here for the first time where useful for the thorough evaluation of stopping criteria. This especially applies to [Zie05c].

Not every stopping criterion has been used in each examination, either because it was shown earlier that it does not provide good results or because it was developed later (or because criteria are equal to each other, e.g. *ImpAv* and *MovObj* for DE). Table 6.3 provides an overview which criterion has been used in which examination.

Table 6.1: Details of examinations concerning stopping criteria for single-objective DE

	DE parameters			Test problems					
Citation	*NP*	*F*	*CR*	#	Constr.	*D*	Characteristics	ϵ_g	Remarks
[Zie05c], [Zie05b]	20	0.9	0.5	8	No	2	3 unimodal (1 with flat surface), 5 multimodal	10^{-3}	Results for PSO were also shown in [Zie05c] and [Zie05b] (see Table 6.2)
[Zie06e]	30	0.7	0.9	1	Yes	16	Power allocation problem from Section 4.2.4	0.07	Parameters set according to parameter study in [Zie06f]; results for PSO in [Zie06d] and [Zie07c]
[Zie08c]	50	0.7	0.9	16	Yes	2-15	Different features, functions chosen based on [Zie06a]	10^{-4}, 10^{-2}	Similar examination for PSO not yet done; allowed remaining constraint violation for equality constraints is $\epsilon_e = 10^{-4}$

Table 6.2: Details of examinations concerning stopping criteria for single-objective PSO

Citation	PSO parameters					Test problems				ϵ_g	Remarks
	Neigh.	NP	w	c_1	c_2	#	Constr.	D	Characteristics		
[Zie05c], [Zie05b]	*lbest*	20	0.8	1.8	1.7	8	No	2	3 unimodal (1 with flat surface), 5 multimodal	10^{-3}	Results for DE were also shown in [Zie05c] and [Zie05b] (see Table 6.1); $\vec{x}_i$ used for stopping conditions
[Zie06d], [Zie07c]	*von-Neumann*	64	0.6	0.4	1.4	1	Yes	16	Power allocation problem from Section 4.2.4	0.07	Parameters set according to parameter study in [Zie09]; results for DE in [Zie06e]; [Zie07c] is an extended version (journal) of [Zie06d]; $\vec{p}_i$ used for stopping conditions

Table 6.3: Stopping criteria used in different examinations

Stopping criterion	[Zie05b] (DE)	[Zie05b] (PSO)	[Zie05c] (DE)	[Zie05c] (PSO)	[Zie06e] (DE)	[Zie06d, Zie07c] (PSO)	[Zie08c] (DE)
ImpBest			x	x	x	x	x
ImpAv			x	x	x	x	x
NoAcc			x	x	x	x	x
MovPar		x	x	x	x	x	x
MovObj				x			
MaxDist	x	x	x	x	x	x	x
MaxDistQuick	x	x	x	x	x	x	x
StdDev	x	x	x	x	x	x	x
StdDevQuick							x
Diff			x	x	x	x	x
ComCrit	x	x	x	x	x	x	
Diff_MaxDistQuick					x	x	

6.1.2.2 Results

ImpBest The results of *ImpBest* are clearly the worst in almost all examinations (an exception is [Zie07c] where a high convergence rate could be found for some parameter combinations using PSO). The reason can already be seen in Figure 6.1: The decrease of the best objective function value may stagnate for some time during an optimization run although the optimum has not been found yet. If only the improvement of the best solution is regarded, wrong conclusions may be derived that lead to early termination of the optimization run. It might be tried to balance the effect of the stagnation of the best function value by using a large setting for the number of consecutive generations ΔG which are regarded for *ImpBest*. This might also mean that many function evaluations are wasted when convergence has been obtained.

ImpAv The performance of *ImpAv* is always better than the performance of *ImpBest*. This result confirms the assumption that a more reliable statement about the state of the population can be given when regarding the improvement of all population members instead of only monitoring the best one. For PSO this criterion still had a very bad performance in [Zie05c] whereas in [Zie07c] it was considerably better. This result supports the assumption that the personal best positions and not the current positions should be regarded for stopping criteria. Using DE convergence rates of 100% could be found for several parameter combinations in the examinations of [Zie05c] (not shown in the paper due to space limitations), [Zie06e] and [Zie08c]. Unfortunately, it is difficult to choose parameter settings. The influence of parameter t is larger than the influence of g, and suitable parameter settings vary considerably for different functions. Besides, it is not clear how parameter settings should be modified for varying demands concerning the accuracy of the result. A connection between parameter settings and the desired accuracy cannot be seen, thus trial-and-error methods have to be used for figuring out suitable parameter settings.

NoAcc Naturally, all improvement-based stopping criteria exhibit problems for functions with a flat surface [Zie05c] because improvement is low for a large part of the search, so the stopping criteria terminate the optimization runs too early. Apart from that, *NoAcc* showed relatively good results for DE in [Zie05c, Zie06e]. However, the missing scalability to values smaller than 1 led to late detection of convergence and thereby to a high additional computational effort in [Zie08c]. Furthermore, appropriate parameter settings vary for different functions. For PSO *NoAcc* had a poor performance in [Zie05c] and even worse results in [Zie07c]. Consequently, this criterion cannot be recommended for PSO. It is usable for DE if appropriate parameter settings can be found. An advantage is that it is easy to check because only the number of accepted individuals has to be counted. On the other hand, *NoAcc* cannot be recommended without hesitation due to the already mentioned disadvantages (scalability, late detection of convergence), together with the missing connection between parameter settings and the desired accuracy of the result.

MovPar Criterion *MovPar* had a good performance for DE and PSO in [Zie05c]. The only exception was a function with a flat surface for which DE was constantly terminated too early. This result can be explained by DE's greedy selection scheme. For PSO different behavior was noticed because the current positions and not the personal best positions were monitored for calculating stopping conditions. Choosing of suitable parameter settings is again difficult as can be seen in [Zie06e, Zie08c] for DE and in [Zie07c] for PSO. With appropriate parameter settings good results may be achieved. Same as for *ImpAv*, the dependence on t is larger as the dependence on g although the influence of g cannot be neglected. It is unclear how parameter settings must be modified for different demands concerning accuracy.

MovObj An examination of *MovObj* was only done in [Zie05c] for PSO using current positions for the calculation of stopping criteria because this criterion equals *ImpAv* for DE as well as for PSO using personal best positions for stopping criteria. The results were bad for a function with a flat surface but for six out of seven other functions convergence rates of 100% have been reached reliably, and for the remaining function the convergence

rate was also above 90% (not all of these results have been published in [Zie05c] due to space limitations). Therefore, this criterion may be used for PSO but the same limitations hold as already discussed for *ImpAv* and *MovPar*: Suitable parameter settings can only be found with trial-and-error methods, and a connection between parameter settings and the demanded accuracy cannot be seen.

MaxDistQuick In contrast to the before-mentioned criteria, for the distribution-based criteria some relation could be seen between the desired accuracy of the result ϵ_g (see also Section 6.1.2.1) and the parameter settings. In the following an overview about parameter settings which showed a good performance is given for DE using *MaxDistQuick*:

- [Zie05c]: $m = \{10^{-3}, 10^{-4}\}$ (with a demanded accuracy of $\epsilon_g = 10^{-3}$). For smaller m, the dependence on p decreased.
- [Zie06e]: $m = \{10^{-1}, 10^{-2}\}$ (with a demanded accuracy of $\epsilon_g = 0.07$). Again, only a negligible dependence on p could be seen.
- [Zie08c]: In this examination the results are less clear because of the broad range of functions. For several functions convergence rates of 100% have already been found with $m = 10^{-2}$ but for the majority of functions $m = 10^{-4}$ provided better results for an accuracy of $\epsilon_g = 10^{-4}$. For an accuracy of $\epsilon_g = 10^{-2}$ most functions were terminated at a suitable time with $m = 10^{-2}$. For the functions that could not even be terminated reliably using $m = 10^{-5}$ for an accuracy of $\epsilon_g = 10^{-4}$, smaller settings had to be used. Only for function g18 no suitable parameter settings could be found, neither for $\epsilon_g = 10^{-4}$ nor for $\epsilon_g = 10^{-2}$. The dependence on p was again minor. Because the number of function evaluations increased only slightly with growing p, large values of p may be used. Keeping in mind the additional computational effort associated with the sorting in *MaxDistQuick*, it is sufficient to use *MaxDist*.

For PSO the following results were obtained:

- [Zie05c]: A larger dependence on p could be seen than for DE. This behavior was already noticeable when examining a reference criterion that led to termination when a certain percentage p of the population has converged. While for DE the number of function evaluations for convergence increased linearly with growing p, for PSO linear behavior was noticed for small p whereas the behavior changed to a quadratic function for $p > 0.6$. The reason is that often a situation occurs where the majority of particles has converged to the optimum while some particles still wander through the search space. This behavior generally does not occur that distinctly for DE due to the greedy selection scheme. A similar behavior was also noticed in [Olo08] where the authors state that outliers are a characteristic of PSO. Therefore, in [Zie05c] it was recommended to use $0.3 \leq p \leq 0.6$ and $m = \{10^{-3}, 10^{-4}\}$ for an accuracy of $\epsilon_g = 10^{-3}$ for *MaxDistQuick* (p should not be chosen too small because otherwise premature convergence may occur). The computational effort increases due to the incorporated Quicksort algorithm [Zie05b] but it is assumed that this computational effort is negligible when optimizing computationally expensive real-world problems.
- [Zie07c]: Similar conclusions as in [Zie05c] were reached: $0.3 \leq p \leq 0.5$ and $m = 10^{-2}$ showed the best performance for an accuracy of $\epsilon_g = 0.07$.

In summary, *MaxDistQuick* is an interesting criterion that leads to reliable termination when proper parameter settings have been found. Mostly, parameter m should be set to values of the same order of magnitude or one order lower than the desired accuracy of the result. In that case reliable termination is obtained after the algorithm has converged but without wasting many computational resources. As there are functions that need different settings of m, still some test runs are necessary when applying *MaxDistQuick* but this is also the case for *LimFuncEval*.

StdDevQuick The criterion *StdDevQuick* has been introduced in [Zie08c] for DE. In previous examinations only the special case *StdDev* has been examined. As *StdDev* and *MaxDist* rely on similar mechanisms, similar results have been obtained. *StdDev* sometimes yielded slightly better results than *MaxDist* [Zie06e]. Furthermore, a difference for DE is that suitable parameter settings were shifted. Generally, smaller settings of m were needed to achieve the same convergence rates ($m = 10^{-4}$ for an accuracy of $\epsilon_g = 10^{-3}$ in [Zie05c] and $m = \{10^{-2}, 10^{-3}\}$ for an accuracy of $\epsilon_g = 0.07$ in [Zie06e]). The dependence on p was negligible but for *MaxDistQuick* the influence was also more noticeable when using PSO.
For PSO the shift of suitable parameter settings in comparison to *MaxDist* could not be seen in [Zie05c] ($m = 10^{-1}$ had to be used for both *MaxDist* and *StdDev*). In [Zie07c] the same effect appeared as for DE in [Zie06e]: Suitable parameter settings were shifted and *StdDev* yielded better results than *MaxDist*. The best results were reached for $m = 10^{-2}$ in [Zie07c] for an accuracy of $\epsilon_g = 0.07$ (for *MaxDist* the best setting was $m = 10^{-1}$ although for *MaxDistQuick* with smaller p the setting $m = 10^{-2}$ was better).
Recapitulating, similar conclusions can be derived as for *MaxDistQuick*: Reliable detection of convergence is obtained if suitable parameter settings are used. Often a connection can be seen between appropriate parameter settings and the desired accuracy but this does not hold for every function. Parameter settings which yield good results are often one order of magnitude lower than for *MaxDistQuick*. This can be explained by the fact that not actual vector differences are regarded here but only the standard deviation of positions. Furthermore, the results of *StdDev* (or *StdDevQuick*, respectively) are often better in terms of convergence rate than the results of *MaxDist* or *MaxDistQuick*, respectively. Convergence is also detected more quickly. Because the results for *MaxDistQuick* varied considerably more with p for PSO than for DE, in future work the generalized criterion *StdDevQuick* should also be tested for PSO.

Diff For criterion *Diff* difficulties were noticed for a function with a flat surface in [Zie05c] for both DE and PSO. This can be explained by the fact that when all individuals yield the same objective function value, the algorithm terminates using criterion *Diff* regardless of the distribution in parameter space. This criterion only has a chance for termination at a reasonable time if at least one objective function value is initialized differently from the other ones. For the other test functions in [Zie05c], criterion *Diff* resulted in reliable convergence behavior for $d \leq 10^{-3}$ (DE) or $d \leq 10^{-1}$ (PSO), respectively. In [Zie06e] the new parameter p was introduced into *Diff* for constrained problems using DE. The performance was relatively constant concerning p, and reliable convergence behavior was found for $10^{-2} \geq d \geq 10^{-4}$ (where the number of function evaluations increased for decreasing d, thus $d = 10^{-2}$ gave the best result). For PSO also no trend could be seen regarding p in [Zie07c] for a constrained problem, and the best result was shown

for $d = 10^{-2}$. In [Zie08c] the clearest indication was found that there is a connection between parameter settings and the desired accuracy of the result: Convergence rates of 100% have been achieved for all 16 functions for $d \leq 10^{-5}$ when demanding an accuracy of $\epsilon_g = 10^{-4}$. For $\epsilon_g = 10^{-2}$ a similar result has been obtained as convergence rates of 100% have been achieved for all functions with $d \leq 10^{-3}$ which is again one order of magnitude smaller than the desired accuracy. It can be concluded that choosing suitable settings of parameter d is easier than for other stopping criteria because a connection to the demanded accuracy can be made (at least for DE; for PSO still more extensive examinations are necessary). As no dependence on p could be seen in any examination, this parameter can be omitted.

Combined Criteria The combined criteria *ComCrit* and *Diff_MaxDistQuick* always needed more function evaluations for detecting convergence than the individual criteria in all examinations. Besides, selecting appropriate parameter settings was complicated because three parameters have to be set for both stopping criteria in contrast to one or two parameters for the individual criteria, respectively. Moreover, the connection between parameter settings and problem features like the desired accuracy which could be seen for some of the individual criteria was obliterated. Although the idea of combined criteria seemed promising, these disadvantages discourage from using them.

Summary In summary, criterion *Diff* is the most advantageous from all stopping criteria which were tested within the scope of this thesis. In most cases convergence was detected reliably and quickly. Furthermore, setting of parameters is easy. Parameter p could be omitted as no dependence on it could be seen. Thus, only one parameter has to be set whereas most of the other criteria include two parameters. In addition, parameter d seems to be closely linked to the desired accuracy of the results (this conclusion holds mainly for DE whereas for PSO more examinations have to be done). A limitation of *Diff* is that it yields bad results when an optimization problem contains an objective function with a flat surface. Fortunately, it can be argued that this is a special case that rarely occurs. Moreover, it can be discovered easily when the optimization run is monitored, so in that case another stopping criterion could be employed.
Additionally, *Diff* is advantageous in contrast to the distribution-based criteria in parameter space if several parameter combinations yield the same objective function value, especially if the positions are in different regions of the search space. In this case the distribution-based criteria in parameter space would waste computational resources while the algorithm tries to converge to one point in the search space, with no or only little improvement of the objective function value. In contrast, the optimization run would be terminated earlier using *Diff*.
Although it was shown in [Zie05b] that the time complexity is similar for most stopping criteria (except for the additional effort for the Quicksort algorithm), the absolute time for calculating stopping conditions may still vary. For real-world problems with computationally expensive constraint and objective functions it can generally be assumed that the computational effort for calculating stopping conditions is negligible but there may also be applications where this effort is noticeable. Apart from the advantages discussed above, *Diff* has the additional benefit that it is easy to calculate: The best objective function value is usually monitored anyway, meaning that only the worst objective function value has to be found, and the difference between best and worst value has to be computed.

Thus, a stopping criterion like *Diff* can be applied to unknown optimization problems, and it is expected to need very little computational effort for determining a suitable setting for its parameter. Afterwards, it can adaptively detect a suitable time for termination even for problems with large fluctuations in the number of function evaluations for convergence.

6.2 Multi-Objective Optimization

It was shown in the previous section that distribution-based stopping criteria provide the best results for single-objective optimization. For multi-objective problems this concept cannot be transferred in the same way because usually multiple Pareto-optimal solutions exist, meaning that the individuals will not converge to one point in the search space. Monitoring the movement of individuals as done for single-objective optimization may also lead to false conclusions because the individuals may still move along the Pareto-optimal front after the population has converged to it. If the improvement of individuals should be taken as basis for stopping criteria, the problem arises how to define improvement in the presence of several objectives.

Therefore, stopping criteria based on other mechanisms have to be found for multi-objective optimization. One possibility is to use performance measures like the ones presented in Section 5.1. The performance measures usually reduce the complexity of assessing the performance of multiple non-dominated solutions to evaluating single numbers which may be compared easily, thus enabling the definition of stopping criteria. Not all performance measures are suitable for this task. Therefore, in Section 6.2.1 the performance measures from Section 5.1 will be discussed regarding their applicability for stopping criteria.

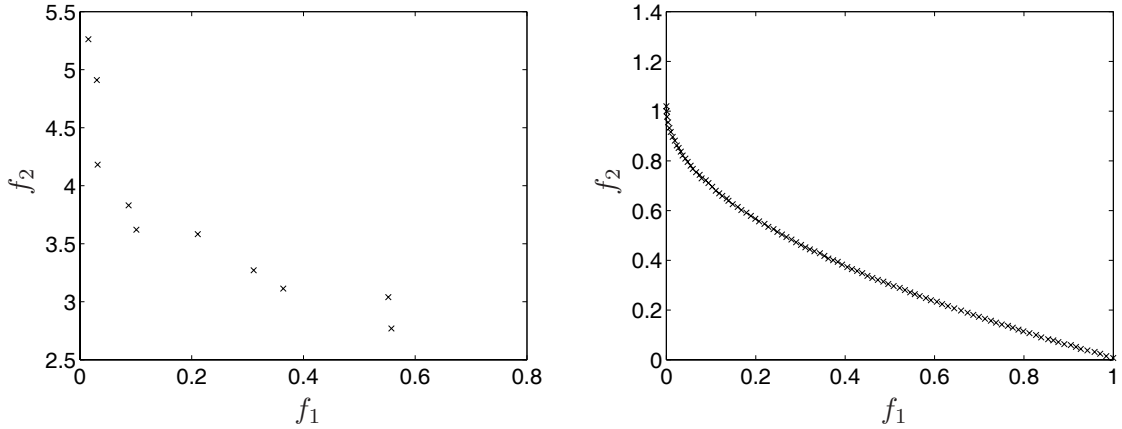
Multi-objective optimization algorithms often contain internal mechanisms which may also be exploited for the definition of stopping conditions. In Section 6.2.2 it is discussed which mechanisms may be used for terminating optimization runs of DE and PSO. Additionally, stopping criteria that have been presented in the literature for NSGA-II but that may also be applied for DE and PSO are given.

Although the mechanisms used for determining stopping conditions are different from the ones employed in single-objective optimization, it is possible to make a similar classification like shown for single-objective optimization in Section 6.1.1. A comprehensive classification has not yet been published in the literature, so it is presented here for the first time. The classification of the stopping criteria based on performance measures as well as internal mechanisms is shown in Section 6.2.3, and in Section 6.2.4 an evaluation of their performance is shown.

6.2.1 Suitability of Performance Measures for Stopping Criteria

In this section the performance measures presented in Section 5.1 are discussed regarding their suitability for defining stopping criteria. The same order is used as in Section 5.1: First, the metrics evaluating closeness to the Pareto-optimal front are reviewed, followed by the metrics considering diversity among non-dominated solutions, and last the metrics regarding both closeness to the Pareto-optimal front and diversity among the non-dominated solutions are discussed.

The error ratio counts the number of solutions which are not members of the Pareto-optimal front. Thus, knowledge of the Pareto-optimal front is needed. Therefore, this performance measure can only be used as reference criterion for functions with known Pareto-optimal front. For problems with real-valued parameters it should be adapted to avoid accuracy problems by allowing a small deviation from the Pareto-optimal front because the probability is low that it will be met exactly.
The set coverage metric can be used in different ways: If the Pareto-optimal front is known, it can be used as reference criterion by comparing the non-dominated solutions of the current generation with the Pareto-optimal front and terminating if a certain threshold has been reached. On the other hand, it is also possible to compare non-dominated solutions generated in consecutive generations and stop an optimization run when reaching a threshold. Thereby, the set coverage metric can also be used for optimization problems with unknown Pareto-optimal front.
The generational distance is defined as the average distance of a set of non-dominated solutions to the Pareto-optimal front. Thus, again a reference criterion can be developed for which the Pareto-optimal front must be known. Similar as for the set coverage metric, it is also possible to derive conclusions about the state of the optimization run by calculating the generational distance between non-dominated solutions of consecutive generations. However, the calculations necessary for checking this criterion add considerable computational effort because for each solution of the current generation the nearest solution of the Pareto-optimal front (or the non-dominated solutions of the previous generation, respectively) has to be found.
Using the maximum Pareto-optimal front error as basis for a stopping criterion is unfavorable. The computational effort for checking it is similar to the generational distance because the distance of all population members to the reference set must be calculated to determine the largest distance but it gives less information. As a consequence, it will not be regarded further in the following.
The before-mentioned performance measures evaluate the closeness to the Pareto-optimal front. It is also possible to extract information about the state of the optimization run by monitoring diversity measures like spacing. In the beginning of an optimization run, usually there will be only few non-dominated solutions that are unequally spaced (see Figure 6.6(a) which was generated using a DE algorithm for function ZDT1, see Section 5.3.3 and Appendix A). Because most multi-objective optimization algorithms contain internal mechanisms for ensuring a good diversity of non-dominated solutions, the non-dominated solutions will become more evenly distributed towards convergence to the Pareto-optimal front (see Figure 6.6(b)). As a consequence, the variation of the spacing can be monitored. If the variation decreases to a small value, it can be concluded that convergence is reached. For the example in Figure 6.6, the spacing is $S = 8.69 \cdot 10^{-2}$ for Figure 6.6(a) whereas it is $S = 2.02 \cdot 10^{-3}$ for Figure 6.6(b).
Because the performance measure spread considers (among others) the distance to the outermost solutions of the Pareto-optimal front, it is only applicable if the Pareto-optimal front is known. Therefore, it is less favorable for the definition of stopping criteria than spacing.
The maximum spread gives less information about the state of the optimization run in contrast to spacing because only the distance between the extreme solutions is regarded. Thus, it is not considered here further, although theoretically it might also be used as basis for a stopping criterion.



(a) Beginning of an optimization run (b) End of an optimization run

Figure 6.6: Non-dominated solutions

The chi-square-like deviation measure may be used as basis for a stopping criterion but it can only be used as reference criterion because the Pareto-optimal front is needed. Besides, it is similar to the error ratio if a small deviation from the Pareto-optimal front is allowed for the error ratio as described above. Thus, it is not considered further.
Hypervolume considers both the closeness to the Pareto-optimal front and the diversity of the non-dominated solutions. It can be used for the definition of stopping criteria in different ways: One possibility is to compare the hypervolume of the current generation with the hypervolume of the Pareto-optimal front, creating a reference criterion. Another option is to compare the hypervolume of the current and the previous generation and to terminate an optimization run if the variation of the hypervolume becomes small. Thus, the Pareto-optimal front is not needed. For all calculations of hypervolume it has to be kept in mind that the same reference point must be used for comparability reasons.
Attainment surfaces are a means for summarizing several runs of an algorithm to statistically evaluate its performance. They are basically a visual aid that is very useful for comparing algorithms but they cannot be used for defining stopping criteria.
In principle it is possible to use combined performance measures for stopping criteria. However, the influence of the individual measures cannot be assessed anymore using the weighted metric, making it difficult to reach decisions about terminating optimization runs. The non-dominated evaluation metric is rather difficult to quantify into one scalar number that may be checked for a stopping condition. Consequently, these combined performance measures are not considered further.

6.2.2 Suitability of Internal Mechanisms for Stopping Criteria

Stopping criteria may also be defined based on algorithm-specific mechanisms. In contrast to stopping conditions derived from performance measures, the internal mechanisms often vary for different algorithms. Hence, their applicability will be less general.
Criterion *NoAcc* can be used for multi-objective optimization in basically the same way as presented for single-objective optimization in Section 6.1.1.3: The number of generations is counted in which at least one new individual was generated that is better than a previously found solution. Only the measure for determining which solution is better has

to be changed. In single-objective optimization it is checked if trial vectors are accepted for DE or if personal best positions are updated for PSO. The decision about the acceptance and the update is based on objective function values, feasibility and constraint violation. For multi-objective optimization, the consideration of the objective function values is substituted by regarding dominance. It is also possible to include crowding distances in the case of non-dominated solutions. Furthermore, it has to be decided if the population of current solutions or the archive (see Sections 5.3.1 and 5.4) should be regarded for the stopping criterion.
For multi-objective optimization even less literature about stopping criteria is available than for single-objective optimization. For DE and PSO not a single reference could be found. The only work that evaluates stopping criteria for multi-objective optimization which has been found is [Rud04] where three stopping criteria are presented for NSGA-II. Because the multi-objective DE and PSO algorithms used here rely on concepts from NSGA-II, these stopping criteria can be applied for DE and PSO as well. They are described in the following:

- Identical Pareto rank: In the beginning of an optimization run usually many non-dominated fronts of different Pareto rank are generated (see Section 5.2). The number of fronts decreases when approaching convergence until all individuals have the same Pareto rank of 1. Thus, an optimization run may be terminated when all individuals have a Pareto rank of 1. In [Rud04] it is argued that checking this condition is not a suitable stopping criterion for NSGA-II because still some progress towards the Pareto-optimal front is noticed after all individuals have become non-dominated. Because different algorithms may react differently to the same mechanisms, it was nevertheless decided to test this criterion for DE and PSO in this work.

- Creation of dominating individuals: The number of newly generated individuals that dominate at least one previously non-dominated individual may be counted to derive a stopping criterion. In [Rud04] it is stated that no tendency regarding this number can be seen for NSGA-II as it continuously varies between 0% and 20% of the population. Again, it can be argued that this relation might be different for DE and PSO.

  This criterion is similar to *NoAcc* as it is also based on the number of newly generated individuals which are better than previously non-dominated solutions. Instead of checking if at least one improved solution was generated in a specified number of generations, it counts the number of improved solutions.

- Stabilization of crowding distance: Similar to regarding spacing as a performance measure (see Section 6.2.1), the crowding distance may be monitored to derive information about the state of the optimization run because it gives information about the distribution of individuals. Different implementations are possible as it is stated in [Rud04] that the minimum, the average as well as the maximum crowding distance stabilize towards reaching convergence. According to [Rud04] the maximum crowding distance is most suitable for defining a stopping condition because it stabilizes last, and a better distribution of solutions is achieved than at the time when the minimum or the average crowding distance stabilizes.

6.2.3 Classification

Although different mechanisms are used for the stopping criteria, the same classification as presented for single-objective optimization in Section 6.1.1 can be done for multi-objective optimization. In the following, criteria based on performance measures as discussed in Section 6.2.1 and criteria based on internal mechanisms as given in Section 6.2.2 are grouped according to the classification in Section 6.1.1. Except for the stopping criteria from [Rud04] which have been discussed in Section 6.2.2, all of these criteria are presented here for the first time. The classification of stopping criteria for multi-objective optimization is also newly developed.
When a multi-objective algorithm is regarded that uses an archive to store non-dominated solutions (as it is the case here, see Sections 5.3.1 and 5.4.1), it must be decided if the stopping criteria should be applied for the current population or for the archive because mostly both cases are possible. Generally, it makes sense to use the archive because it is the archive that will be given to the decision maker as the final solution of the optimization problem. For DE there may not be a large difference between the archive and the population due to the greedy selection scheme (mainly the number of solutions might vary). In contrast, for PSO the population members are able to deteriorate and for the personal best positions no method for ensuring diversity is applied. Hence, for PSO it is especially important to use the archive wherever possible. Therefore, in this work always the archive is used for the definition of stopping criteria, with the exception of the criterion based on identical Pareto rank because dominated solutions will never be able to enter the archive.

6.2.3.1 Reference Criteria

As presented in Section 6.1.1.1, reference criteria for single-objective optimization can be defined by measuring the difference in objective function values. In contrast, more complicated performance measures have to be applied for multi-objective optimization which transform a set of non-dominated solutions into a measure that is easier to evaluate. As discussed in Section 6.2.1, several performance measures can be used for creating reference criteria in dependence on the knowledge of the Pareto-optimal front (see also Section 5.1):

- *RefCritER*: If the error ratio ER decreases to or below a certain value ER_{stop}, the optimization run is terminated:

$$ER \leq ER_{stop}. \tag{6.18}$$

  As discussed in Section 6.2.1, the application of this criterion for problems with real-valued representation requires an additional parameter ϵ_{ER} for the threshold up to which distance a solution is regarded to be on the front.

- *RefCritC*: If the set coverage metric $\mathcal{C}(P^*, Q)$ calculated using the non-dominated front Q of the current generation and the Pareto-optimal front P^* decreases to a value C_{stop}, meaning that only few members of Q are dominated by solutions of the Pareto-optimal front, the optimization run is terminated:

$$\mathcal{C}(P^*, Q) \leq C_{stop}. \tag{6.19}$$

- *RefCritGD*: An optimization run is stopped if the generational distance GD falls below a pre-defined value GD_{stop}:

$$GD \leq GD_{stop}. \tag{6.20}$$

- *RefCritHV*: The algorithm might be terminated if the ratio of the hypervolume HV of the current generation to the hypervolume HV_P of the Pareto-optimal front reaches a certain value p_{HV} (with $0 < p_{HV} \leq 1$):

$$\frac{HV}{HV_P} \geq p_{HV}. \tag{6.21}$$

  When implementing this criterion, it has to be assured that always the same reference point is used for both fronts. Thus, it might be necessary to recalculate the hypervolume of the Pareto-optimal front if non-dominated solutions have been generated in the current generation which exceed the reference point in at least one dimension.

Each of the discussed reference criteria can be used for other multi-objective optimization algorithms as well because no reference to specific characteristics is done but only generally applicable performance measures are used as basis.

6.2.3.2 Criteria based on Limited Resources

There is no difference between single-objective and multi-objective optimization concerning the criteria based on limited resources (which might also have contributed to their popularity, besides their simplicity). Therefore, stopping after a certain number of function evaluations, generations or CPU time can be done for multi-objective optimization as described in Section 6.1.1.2 for single-objective optimization.

6.2.3.3 Improvement-based Criteria

Same as for single-objective optimization, an optimization run should be terminated if the improvement becomes so low that no noticeable variation in the results can be expected anymore if the algorithm is allowed to continue. Improvement-based criteria for multi-objective optimization can be created in different ways. In the following first the criteria derived from the discussion about performance measures in Section 6.2.1 are given:

- *ImpC*: The set coverage metric is used for evaluating the improvement achieved when moving from one generation to the next:

$$\frac{\mathcal{C}\left(Q_{g-1}, Q_{g-2}\right) - \mathcal{C}\left(Q_g, Q_{g-1}\right)}{\mathcal{C}\left(Q_{g-1}, Q_{g-2}\right)} \geq t_C \quad \forall g \in \{G - \Delta G, \ldots, G\}. \tag{6.22}$$

  A large value of $\mathcal{C}\left(Q_g, Q_{g-1}\right)$ indicates good performance because many solutions from the previous generation are dominated by solutions of the current generation. If $\mathcal{C}\left(Q_g, Q_{g-1}\right)$ becomes small in contrast to $\mathcal{C}\left(Q_{g-1}, Q_{g-2}\right)$, the improvement is low. If this condition holds for ΔG consecutive generations, the optimization run will be terminated.

- *ImpGD*: Instead of calculating the generational distance in dependence on the Pareto-optimal front, it is calculated between the fronts generated in two consecutive generations, respectively, so that GD_G is the generational distance calculated between the solutions from generation G and the solutions from generation $G-1$. If the improvement of the generational distance is equal to or below a threshold t_{GD} for ΔG generations, the execution of the optimization algorithm is stopped:

$$\frac{GD_{g-1} - GD_g}{GD_{g-1}} \leq t_{GD} \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.23}$$

  In Equation 6.23 it is considered that smaller values indicate better performance for *GD*, thus the improvement will be positive if *GD* decreases from one generation to the following.

- *ImpHV*: The optimization run is terminated if the improvement of the hypervolume is equal to or below a threshold t_{HV} for a specified number of generations ΔG:

$$\frac{HV_g - HV_{g-1}}{HV_{g-1}} \leq t_{HV} \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.24}$$

  With this definition, the improvement will be positive if *HV* increases from one generation to the next because *HV* indicates better performance with increasing values.

  Because in generation G a solution might be found that exceeds the reference point in at least one dimension, the archive of the previous generation $G-1$ must be saved. If necessary, it may be scaled with the new value and its hypervolume may be recalculated.

Same as for the reference criteria, the improvement-based criteria discussed so far may also be applied for other multi-objective optimization algorithms due to the use of generally applicable performance measures.
As already discussed in Section 6.2.2, a stopping criterion may be defined that is based on internal mechanisms of the algorithms and that is very close to an improvement-based criterion for single-objective optimization:

- *NoAcc_MO*: An optimization run is terminated if in ΔG consecutive generations not a single solution was generated that dominates a solution from the archive. This criterion is equal for DE and PSO in contrast to single-objective optimization.

  Other multi-objective evolutionary optimization algorithms may work differently but some kind of selection is usually present, meaning that this criterion will generally be applicable although slight adjustments may be necessary.

In Section 6.2.2, stopping criteria from [Rud04] which are based on internal mechanisms have been given. Two of them can be classified as improvement-based:

- Identical Pareto rank (*IdParetoRank*): The optimization run is terminated when all individuals have the same Pareto rank of 1. In this case it is reasonable to regard the population and not the archive for the stopping criterion because only non-dominated solutions are allowed to enter the archive.

Naturally, this criterion makes most sense for algorithms which calculate the Pareto rank anyway as it is the case for Differential Evolution here. For other algorithms, including the Particle Swarm Optimization algorithm employed here, the calculation of the Pareto rank is possible but it adds additional computational effort.

- Creation of dominating individuals (*DomInd*): The number of individuals is counted which dominate at least one archive member. An optimization run is stopped if the number is below or equal to a threshold N.

  For multi-objective optimization algorithms which do not use an archive this criterion must be adapted. Same as for *NoAcc_MO* a similar criterion should be possible to define for most algorithms.

6.2.3.4 Movement-based Criteria

In single-objective optimization usually all population members eventually converge to one point in the search space. Towards convergence, the DE individuals and the PSO particles will have decreasing step sizes. The reason is that the DE individuals create movement via vector differences which will become small. The PSO particles rely on the difference between current position and the personal and neighborhood best positions which will also decrease (the influence of the velocity of the previous time step also decreases because usually the inertia weight is below 1, see Section 3.6.3). Consequently, it can be concluded that convergence has been reached when the movement of individuals becomes small.
In contrast, in multi-objective optimization usually convergence to the Pareto-optimal front is desired. There will be multiple solutions, and the population members are still able to move along the Pareto-optimal front after it has been found. Therefore, movement of the individuals like defined for single-objective optimization in Section 6.1.1.4 is not suitable for the definition of stopping criteria in multi-objective optimization.
Theoretically, it may be possible to define equivalents to movement-based criteria in objective space for multi-objective optimization by adjusting the improvement-based criteria based on performance measures so that the absolute value of change of the performance measures is monitored. Because this modification is not expected to result in a large difference in performance, movement-based criteria are not considered further for multi-objective optimization.

6.2.3.5 Distribution-based Criteria

It must be clearly distinguished between distribution-based stopping criteria for single-objective optimization and for multi-objective optimization because different principles are used. Distribution-based criteria in single-objective optimization rely on the assumption that the population members will be close to each other eventually when approaching convergence. In contrast, in multi-objective optimization goals are generally in conflict to each other, thus the individuals should not converge to one point in the search space. Instead, a distribution along the Pareto-optimal front is required that is as equally spaced as possible. It can be assumed that in the beginning of an optimization run the distribution changes frequently because often new solutions are found which are maybe even located in different parts of the search space than the solutions which have been found before. In contrast, the changes in the distribution will be smaller towards the end of the optimization run. Although the individuals will not necessarily stop moving along

the Pareto-optimal front, meaning that there will always be small variations in the distribution, it is assumed that the distribution will stabilize in contrast to the beginning of the optimization run. Consequently, it is possible to monitor performance measures that evaluate the distribution of individuals for the definition of stopping criteria:

- *DistSpacing*: The change of spacing from one generation to the next is regarded. If it is equal or below a threshold t_s for ΔG generations, the optimization run is terminated:
$$\frac{S_{g-1} - S_g}{S_{g-1}} \leq t_s \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.25}$$

- *DistSpread*: Similar to *DistSpacing*, the change of spread from one generation to the next is checked. An optimization run is stopped, if the change of spread is equal or below a threshold t_{sp} for ΔG generations:
$$\frac{\Delta_{g-1} - \Delta_g}{\Delta_{g-1}} \leq t_{sp} \quad \forall g \in \{G-\Delta G, \ldots, G\}. \tag{6.26}$$

  Because spread includes the distance to the outermost solutions of the Pareto-optimal front, it can only be applied if these solutions are known. Nevertheless, it is not classified as reference criterion here because its main task is to evaluate the distribution of a non-dominated set.

In Section 6.2.2 a stopping criterion introduced in [Rud04] was discussed that relies on an internal mechanism of NSGA-II that reflects the distribution of individuals:

- Stabilization of maximum crowding distance (*MaxCD*): Because oscillations may still occur even after stabilization of the maximum crowding distance d_l, in [Rud04] it is suggested to use the standard deviation σ_L of the maximum crowding distance over L generations for the definition of a stopping criterion. If it falls below a threshold δ_{lim}, the optimization run is terminated:
$$\sigma_L = \sqrt{\frac{1}{L}\sum_{l=1}^{L}(d_l - \bar{d_L})^2} < \delta_{lim} \tag{6.27}$$

  where $\bar{d_L}$ is the average of d_l over L generations. It is furthermore noted in [Rud04] that all crowding distances are normalized. In [Rud04] the parameters are set to $L = 40$ and $\delta_{lim} = 0.02$ with a population size of 100, and it is recommended to change δ_{lim} when varying the population size. It is also noted that the parameters were chosen in a way that stopping would rather occur a little too late than too early because priority is given to convergence to the Pareto-optimal front in contrast to computational cost.

Similar as for the improvement-based criteria, the stopping criteria based on performance measures are generally applicable for any multi-objective optimization algorithm. The criterion based on maximum crowding distance may theoretically also be applied for algorithms which do not internally make use of the crowding distance but in that case the additional computational effort has to be considered.

6.2.3.6 Combined Criteria

Same as for single-objective optimization, it is possible to combine several criteria to account for functions with varying features. To save computational effort, it would again be advisable not to check several criteria in every generation but to examine one criterion first, and only if its termination condition is fulfilled, a second criterion might be evaluated. Stopping criteria which rely on different mechanisms should be chosen for generating combined criteria. Many combinations are possible, e.g. it might be beneficial to combine *ImpC* with *DistSpacing*, so one criterion relies on improvement based on dominance and the other on the distribution of solutions.
However, for single-objective optimization it was shown that combined criteria are more difficult to handle than the individual criteria. They usually contain a larger number of parameters, furthermore a dependence of parameter settings on the desired accuracy of the results could not be established anymore, and generally no advantage for combined criteria could be seen (see [Zie06e, Zie07c] and also Section 6.1.2). Therefore, combined criteria are not tested for multi-objective optimization here.

6.2.4 Assessment of Performance

The stopping criteria defined in Section 6.2.3 have been examined in [Beg08] (which is a student research project developed within the scope of this thesis) based on the test functions given in Appendix A. In the following the experimental settings, the main results and a short summary are given.

6.2.4.1 Experimental Settings

The control parameters of DE have been set to $F = 0.7$, $CR = 0.9$ and $NP = 100$, and the archive size has also been set to 100. An exception had to be made for ZDT4 because convergence has not been reached with these settings (see also Section 5.3.3). Therefore, $F = 0.1$ and $CR = 0.05$ have been used for ZDT4. To avoid wasting of many computational resources in the case of an unsuitable stopping criterion, the number of generations has been limited to $G_{max} = 2000$.
For PSO the control parameters $w = 0.73$, $c_1 = 1.5$, $c_2 = 1.5$, $NP = 100$ and a *gbest* neighborhood topology have been used. The size of the *gbest* archive was restricted to 100 solutions. For ZDT2 and ZDT4 the generated diversity seemed to be too low because PSO repeatedly had only one member in the *gbest* archive. Variation of the parameters did not lead to better results, thus all optimization runs have been done with the same parameter settings (additional operators for generating diversity like described in 5.4.1 might have helped but here a basic multi-objective PSO algorithm should be examined). Therefore, the results for ZDT2 and ZDT4 are only discussed for stopping criteria that terminated at least one run before reaching the maximum number of generations. Actually, also important conclusions can be derived for these test problems because the ability of stopping criteria to terminate optimization runs with bad performance is also interesting. Because PSO converged faster than DE in preliminary test runs for ZDT1, ZDT3 and ZDT6 and computational resources should be saved, the maximum number of generations has been set to $G_{max} = 1000$ for PSO.
To further limit the computational effort for this study, three representative settings have been regarded for the main parameter of every stopping criterion whereas one or two

settings have been considered for thresholds (see Table C.1 in Appendix C). Suitable settings have been found in preliminary test runs. 25 optimization runs have been done for every parameter combination.
The results are evaluated based on boxplots (see also Section 3.5.2) of the generational distance which show the distance to the Pareto-optimal front for the final result. Additionally, the numbers of generations at which termination of the runs was induced are displayed as boxplots. Furthermore, the numbers of runs which were terminated before reaching the maximum number of generations are given. All of these results can be found in Appendix C.
Based on these performance measures, it will be discussed if all runs could be stopped before reaching the maximum number of generations, the robustness concerning parameter settings, scalability of the stopping criteria parameters, the computational effort for computing the stopping criteria (even a stopping criterion that reliably terminates all runs at a suitable number of generations might become unreasonable if its computational effort is too high), differences between the behavior of stopping criteria for Differential Evolution and Particle Swarm Optimization and if runs experiencing premature convergence can be stopped.
For several stopping criteria, especially the improvement-based and distribution-based criteria, the beginning of an optimization run was problematic because erratic changes occurred that might sometimes have induced termination. To avoid this undesirable behavior, additionally it was checked for these criteria if 95% of the desired size of the archive has been reached. This condition might be disadvantageous if the algorithm finds only few solutions, so that e.g. in the case of convergence to one point in the search space the algorithm would not be terminated. Thus, this condition should be substituted in future work. One possibility is to check the stopping criteria only after the initial phase is over but this would again burden the user with the choice of a suitable number of generations. In the course of this work another possibility was discovered that will be discussed in the following subsection.

6.2.4.2 Results

For single-objective optimization the reference criteria have been omitted in the evaluation of stopping criteria performance because the definition of reference criteria can be easily done in dependence on the difference between the objective function value of the global optimum and the best objective function value found so far in an optimization run. In contrast, reference criteria for multi-objective optimization have to rely on more complicated measures. Thus, a discussion of results for reference criteria is also included here although their use is generally limited to test problems.
RefCritER yields good results for DE by consistently terminating all optimization runs (with the exception of ZDT4 but this is due to problems of the algorithm that did not converge close enough), see Figures C.1 and C.2. The threshold parameter ϵ_{ER} influences the number of generations at which the optimization run was terminated more than the parameter ER_{stop}, and the quality of the results in terms of generational distance improves for a smaller ϵ_{ER}. The same dependence can be seen for PSO (see Figures C.18 and C.19). For PSO this criterion also shows reliable termination of optimization runs (with the exception of ZDT2 and ZDT4 due to algorithm problems).
The worst results of the reference criteria have been obtained with *RefCritC*. The reason is

that at least some solutions from the generated approximation set must be non-dominated with the Pareto-optimal front. This is difficult to achieve because often a small deviation is still existing. Although there is a parameter C_{stop} that may be changed, scalability is bad because at least one solution must be non-dominated with the Pareto-optimal front. Therefore, for DE this criterion was not able to terminate any optimization run (see Figure C.3). For PSO several runs could be stopped but the termination occurred quite late (see Figure C.20). As a consequence, *RefCritC* cannot be recommended for either DE or PSO. It is assumed that the same problems will be experienced with other optimization algorithms.

For *RefCritGD* even small changes in the parameter GD_{stop} lead to large variations in the number of generations at which termination occurred as well as in the generational distance for DE (see Figure C.4). This behavior is not noticeable for PSO for which the results are very similar for all settings of GD_{stop} (see Figure C.21). Thus, this stopping criterion is applicable for both DE and PSO but while the performance is quite robust for PSO, for DE some experimentation might be necessary to find a suitable setting for GD_{stop}.

For *RefCritHV* the same behavior concerning parameter p_{HV} is noticed as for GD_{stop} when using *RefCritGD*: For DE small changes in the parameter lead to large variation in the results (see Figure C.5) whereas for PSO the difference of results is not that distinct (see Figure C.22).

For *ImpC* there is a considerable variation in the number of generations at which the optimization runs are stopped for different settings of ΔG using DE (see Figure C.6). This means that the performance of the stopping criterion is not robust regarding ΔG (with the exception of ZDT4 for which the number of generations as well as the generational distance is more similar for different settings of ΔG). For PSO a similar behavior can be found (see Figure C.23).

For *ImpGD* an even larger difference in the number of generations can be seen for varying settings of ΔG for DE and for PSO (see Figures C.7 and C.24). The generational distance also varies strongly, especially for DE. Thus, for the application of this stopping criterion some effort for tuning the parameter t_{GD} will be necessary.

The criterion *ImpHV* has been examined for the case $t_{HV} = 0$ (see Figures C.8 and C.25), meaning no improvement has been found concerning this performance measure, as well as a small threshold of $t_{HV} = 0.0001$ (see Figures C.9 and C.26). For both DE and PSO there is a large difference concerning the number of generations at which termination occurred for these different settings of t_{HV} whereas the difference in the results regarding generational distance are noticeable but not that distinct. There is also a noticeable difference in the results for different settings of ΔG (with exception of PSO with $t_{HV} = 0.0001$ where the differences are smaller) which means that some effort may be required to find suitable settings for a specific problem. In contrast to most other stopping criteria, *ImpHV* was able to induce termination for ZDT2 and ZDT4 several times using PSO although the global optimum has not been found. This is a desirable effect that runs are also terminated when it is apparent that the global optimum will not be found. Unfortunately, termination occurred for only some optimization runs whereas it would be better to have reliable termination also in this case.

Applying *NoAcc_MO*, for DE all runs are reliably terminated (with exception of ZDT3 where some runs are not stopped). The difference between the different settings of ΔG is noticeable but not very distinct (see Figure C.10). The generational distance is similar

for all settings of ΔG for DE except for ZDT1 and ZDT2 where the first setting ($\Delta G = 5$) showed a considerably higher generational distance. For PSO there are noticeable differences in the number of generations at which termination occurred but the generational distance is mostly quite similar (see Figure C.27). An advantage of *NoAcc_MO* in comparison to other criteria is that the computational effort for computing it is very low, so its application is favorable for both DE and PSO.
Examinations with *IdParetoRank* confirm the results of [Rud04] for DE: The optimization runs are constantly stopped too early (see Figure C.11). The missing scalability of the criterion might generally be seen as advantage because no parameter has to be set but as the criterion stops the optimization runs before convergence has been reached, the missing scalability is a disadvantage here. For PSO a different result is obtained due to the different selection scheme (see Figure C.28): DE's greedy selection scheme reduces the number of ranks quickly whereas for PSO also dominated individuals may be generated, thus the stopping criterion is often fulfilled very late. As a consequence, it is not suitable for either DE or PSO. However, for DE this criterion does not add computational effort because the Pareto rank is calculated anyway for this algorithm in its present form. Thus, it could be used in combination with a more computationally expensive criterion. A possibility would be to check the Pareto rank first and only if *IdParetoRank* is fulfilled, another criterion is used for the rest of the optimization run. This way, computational resources can be saved.
Using *DomInd* as stopping criterion, the optimization runs are usually terminated when there is still a large difference to the Pareto-optimal front, i.e. the runs are terminated too early (see Figures C.12 and C.29). There is also not much possibility for adjusting the parameter N (the number of individuals that dominate at least one archive member) because $N = 2$ which is the lowest number that was used here is already very small, and setting the number of individuals to $N = 0$ would be the same as using *NoAcc_MO* with $\Delta G = 1$. Thus, it might be better to use *NoAcc_MO* which is a similar criterion but that regards the number of generations without generating dominating individuals instead of the number of dominating individuals. As a result, *DomInd* cannot be recommended for both DE and PSO which is the same result that was reached in [Rud04] for NSGA-II.
With only few exceptions for ZDT3 and ZDT4, all optimization runs of DE have been successfully terminated using *DistSpacing* (see Figures C.13 and C.14). In almost all cases there is a noticeable difference between the results for different parameter settings, meaning that there will be some effort required for finding a suitable setting. A similar result is obtained for PSO (see Figures C.30 and C.31). It might be speculated that a criterion like *DistSpacing* should also be suitable to terminate runs with insufficient convergence towards the Pareto-optimal front. The reason why this behavior was not noticed for ZDT2 and ZDT4 using PSO here might be the additional condition that 95% of the archive must be occupied that was inserted due to misleading results in the initial phase of the optimization run. Therefore, in future work it should be checked if another condition for the initial phase leads to better results.
As expected because *DistSpread* is very similar to *DistSpacing*, the results are also nearly identical for both DE and PSO (see Figures C.15, C.16, C.32 and C.33). It must be noted that computing the spread requires the knowledge of the Pareto-optimal front (or at least the outermost solutions of the Pareto-optimal front). As a consequence, *DistSpacing* should be preferred, especially for real-world optimization problems.
Regarding *MaxCD* as stopping criterion, all DE optimization runs are successfully ter-

minated (see Figure C.17). There are significant differences in the results for different settings of the parameter δ_{lim}, thus there will be some effort for finding suitable settings. Using PSO, the optimization runs for ZDT1, ZDT3 and ZDT6 are almost always successfully terminated. Even for ZDT2 and ZDT4 termination before reaching the maximum number of generations sometimes happens (see Figure C.34). In summary, it can be concluded that this criterion is suitable for DE and PSO as it was also stated for NSGA-II in [Rud04]. A difference is that in [Rud04] $\delta_{lim} = 0.02$ has been recommended whereas for DE as well as PSO considerably smaller values had to be used (see Table C.1).

Summary Reference criteria can usually only be used for test problems because the Pareto-optimal front has to be known. Besides, for these criteria it is assumed that the algorithm will converge to the Pareto-optimal front with a given precision, i.e. optimization runs with premature convergence will not be terminated (or only if the distance to the Pareto-optimal front is below the demanded accuracy of the results). Apart from these disadvantages, *RefCritER*, *RefCritGD* and *RefCritHV* yielded good results for both DE and PSO although some tuning of the parameters may be necessary when applying them, especially for *RefCritGD* and *RefCritHV* using DE. All of these criteria are scalable, so runs which reach the vicinity of the Pareto-optimal front may be terminated. In contrast, *RefCritC* requires solutions to be non-dominated with the Pareto-optimal front which is a criterion that is much harder to fulfill. Thus, it is concluded that it is less suitable than the other criteria. It has to be noted that some computational effort is required to check the criteria, especially for a higher number of objectives.
The improvement-based criteria terminate an optimization run if only little improvement is noticed. For the criteria *ImpC*, *ImpGD* and *ImpHV* reasonable behavior was found for both DE and PSO although again some tuning of the parameters may be necessary. It must be kept in mind that computational effort is added that may be considerably large, especially if the number of objectives increases. For *NoAcc_MO* good performance has been found for both DE and PSO. Regarding the computational effort, *NoAcc_MO* is also favorable because only a variable has to be added that counts the number of accepted solutions. As also observed in [Rud04] for NSGA-II, *IdParetoRank* as well as *DomInd* are not suitable as stopping criteria for neither DE nor PSO. However, checking for an identical Pareto rank may be conducted before switching to a computationally more expensive stopping criterion for DE: The algorithm in its present form calculates this information anyway and *IdParetoRank* is usually fulfilled before the algorithm reaches the vicinity of the Pareto-optimal front.
Interestingly, the distribution-based criteria showed good results. Although the use of them for stopping criteria may not seem very intuitive because there is no obvious connection to the convergence towards the Pareto-optimal front, it should not be forgotten that multi-objective optimization has several goals in contrast to single-objective optimization (see Chapter 5). The distribution of individuals will only stabilize, i.e. the change in the distribution will become small, if the population approaches convergence. For real-world optimization problems, *DistSpread* is generally not suitable in contrast to *DistSpacing* or *MaxCD* because information about the Pareto-optimal front is needed. Regarding the computational effort, these criteria are similar to each other. Because the crowding distance is used internally by the DE and PSO algorithms employed in this work, it might seem computationally more efficient to use *MaxCD* instead of *DistSpacing* as stopping criterion. It must be kept in mind that the crowding distance is used

to choose which solutions to keep in the case that the maximum population or archive size is exceeded, meaning that the crowding distance will change after its calculation. Efficient tracking of the deleted solutions and their neighbors might help by adding only small additional computational effort, thus indeed making *MaxCD* preferable in contrast to *DistSpacing*.

6.3 Summary and Future Work

When humans observe the development of an optimization run, they usually have a good idea about the correct time for ending the optimization run because they consider different features like the improvement of objective function values or the distribution of the population. However, to be able to use optimization algorithms in automatic design processes, the goal is to automatically terminate an optimization run without human intervention. In this chapter stopping criteria have been presented which also consider information about the optimization run to decide a proper time for terminating a run. Different measures have been applied and evaluated for this purpose, and it also had to be distinguished between single-objective and multi-objective optimization.

Several implementations of improvement-based and movement-based stopping criteria provided good results for single-objective optimization in some cases but it is difficult to adjust the parameters accordingly. Furthermore, no connection could be seen between the desired accuracy of the result and the parameter settings. In general, it is better to monitor the whole population instead of only the best solution.

For distribution-based stopping criteria operating in parameter space it is easier to guess suitable parameter settings but still different settings may be needed for different functions. If suitable parameter settings have been found, termination of the optimization run is carried out reliably. Finding appropriate settings for a distribution-based stopping criterion in objective space (*Diff*) seems even easier. For a large range of functions it is sufficient to use settings of one order of magnitude smaller than the desired accuracy of the result. Further advantages of this criterion are that it contains only one parameter (after omitting one parameter because no dependence on it could be seen), it is easy to calculate, and it also terminates a run if the same objective function value is yielded by different parameters. For DE it has shown good performance in an extensive examination using different accuracies. For PSO a similarly thorough investigation is left for future work.

For multi-objective optimization the definition of appropriate stopping criteria is even more important because real-world problems usually contain multiple objectives. It is also even more challenging because instead of a single solution a whole set of non-dominated solutions is desired. Consequently, it becomes more difficult to find suitable measures for defining stopping criteria. In this work performance measures as well as internal mechanisms of the algorithms have been used for deriving stopping conditions. Despite the different underlying mechanisms, these stopping criteria can be categorized in the same way as for single-objective optimization. A classification of stopping criteria for multi-objective optimization has been shown for the first time in this work. Because especially the calculation of performance measures may require considerable computational effort that will increase even to a greater extent if more objectives are regarded, attention has to be paid that the advantage gained by using suitable stopping criteria is not outweighed

by the computational effort for calculating the stopping criteria. The estimation of the required effort in relation to the gained advantage is strongly dependent on the application, i.e. on the ratio between the computational effort for calculating the stopping criteria to the computational effort for determining the values of the constraint and objective functions. It is beneficial to exploit internal variables that are calculated for the algorithm anyway, if possible. One promising criterion is to terminate if in a certain number of consecutive generations no solution has been generated that is better than an archive member ($NoAcc\_MO$), thus the improvement of the algorithm is evaluated. Another option is to regard the distribution by checking the stabilization of the maximum crowding distance ($MaxCD$).
In future work emphasis should be placed on finding stopping criteria that are able to end optimization runs with insufficient convergence towards the Pareto-optimal front. This also includes the case of convergence to one point in the search space. In this work a thorough investigation regarding this was complicated by the condition that 95% of the desired archive size must be reached. This condition should ensure that the algorithm is not terminated in the initial phase of optimization runs because of misleading information. For DE this condition could be substituted by checking $IdParetoRank$ that was shown to terminate optimization runs before sufficient convergence has been reached. For PSO $IdParetoRank$ is not applicable, thus other measures have to be found, e.g. $DomInd$ could be used that also ended optimization runs rather early. It can be expected that new performance measures will be developed in the future which may also be examined regarding their applicability for stopping criteria.
It is not clear if control parameter settings have an influence on stopping criteria. Thus, based on the results of this work, the impact of parameters like F and CR for DE as well as w, c_1 and c_2 for PSO should be examined in future work. Furthermore, the influence of the population size should be analyzed. Features of the optimization problems might also change results, e.g. the dimensionality, the number of local optima or the presence of constraints (particularly because several criteria have to be modified for constrained optimization).
Except for the power allocation problem, the stopping criteria introduced in this work have mostly been used for test problems. The reason is that an extensive evaluation of stopping criteria was needed. That is only possible for problems which are not too computationally expensive. For future work it would be interesting to test at least some criteria which showed good performance for test functions for more real-world problems. One example where stopping criteria will definitely be useful is in circuit design as described in Section 5.5 due to the computationally expensive objective and constraint functions which have to be evaluated via simulation.
Many of the stopping criteria that are examined in this work may also be employed for other evolutionary algorithms but the performance will not necessarily be equal. Therefore, in future work these stopping criteria should also be tested for other algorithms, and the performance should be compared with the results for DE and PSO.

Chapter 7

Adaptive Strategies for Setting Control Parameters

Optimization algorithms usually contain several control parameters which have to be set by the user and which influence the convergence behavior. The convergence probability is affected, e.g. certain settings might lead to premature convergence because not enough diversity is generated. Furthermore, especially for real-world problems with computationally expensive objective functions, optimization algorithms are required to spend as few function evaluations as possible for reaching convergence, and the convergence speed is also dependent on parameter settings.

In the literature attempts can be found to identify parameter settings which yield good results, e.g. in [Lam04, Gäm02, Kuk06d] for DE and in [Tre03, Cle02, Car01] for PSO. The results of these examinations are often contradicting, especially for PSO, showing that it is generally difficult to find settings which work well for a large range of optimization problems. Finding suitable parameter settings is not only difficult for users who are inexperienced concerning optimization but also for optimization experts this is not necessarily a trivial problem. Sometimes it is tried to establish a connection between settings of parameters and features of the problem but this attempt is seldom successful. There are some exceptions, e.g. in [Lam04] a low setting for CR is recommended for problems with a low degree of interaction between the function parameters. In contrast, CR should be large for rotated non-decomposable functions and also for multimodal functions for which the local optima are not aligned with the coordinate axes [Pri05]. However, connections like these can rarely be seen. Additionally, especially for real-world problems generally not much information is available about features of the problem landscape. Thus, even if a connection between features and parameter settings can be made, this knowledge can not necessarily be applied.

Another problem is that results for single-objective optimization can not necessarily be transferred to multi-objective optimization. This is not surprising because there are different goals in single-objective and multi-objective optimization. For example, it is stated in [Kuk04b] that although a high crossover rate is generally recommended for single-objective optimization using DE, in multi-objective optimization a high crossover rate might cause one objective to be optimized faster than others, leading to convergence to a single point of the Pareto-optimal front.

If optimal parameter settings for a specific optimization problem are not known, generally trial-and-error methods are applied to identify suitable parameter settings. For most

parameters in EAs a range of applicable settings is given. If that range is small, it might be assumed that finding good parameter settings is not difficult. However, there are often interaction effects between different parameters, increasing the effort for parameter tuning. Furthermore, due to the randomness involved in EAs, it might not be sufficient to conduct only one trial run for every parameter combination because the results might vary considerably for different runs (this is also documented in the beginning of Chapter 6). Especially for real-world problems with computationally expensive objective functions, the resulting computational effort might prohibit this time-consuming process.
Because optimal parameter settings are problem-dependent, a better approach is to adaptively set parameters. In that case, suitable settings are found during optimization runs. It is usually assumed that different parameter settings are optimal in different stages of the optimization run [Eib99] because the tasks are also different: In early phases a good coverage of the search space must be ensured (exploration) whereas towards the end of an optimization run the search must concentrate on the vicinity of the best solution for fine-tuning the results (exploitation). From this follows an additional advantage of adaptively setting parameters because the parameters are able to vary over time.
Methods for adaptive parameter setting can be categorized due to different criteria. The approaches can work on different levels of an EA: Parameters can be adapted for the whole population (population level) or for each individual separately (individual level) or also independently for every component of individuals (component level) [Ang95]. Besides, parameter control can be classified as deterministic, adaptive or self-adaptive [Eib99]. Deterministic parameter control modifies parameter settings according to a pre-defined rule without considering feedback from the search. An example is linearly decreasing the inertia weight in Particle Swarm Optimization as described in Section 3.6.3. In contrast, adaptive parameter control uses feedback from the search to determine in which way parameter settings should be modified. In self-adaptive parameter control usually each individual has a separate set of control parameters which are subjected to the evolutionary process. This way the parameters evolve towards better settings while the individuals search for better objective function values.
In [Eib99] it is stated that most studies about parameter control concentrate on only one parameter at a time. Because parameters often interact, this approach may not be able to identify optimal parameter settings. Unfortunately, tuning several parameters simultaneously generally leads to a strong increase in computational cost compared to examining one parameter at a time. In this chapter it will be shown how these disadvantages can be avoided by applying methods from Design of Experiments for parameter control. Thus, interactions of parameters are regarded but without generating large computational cost. The use of Design of Experiments in optimization is not new as it is already employed as an optimization method in [Deb05] and it is used for off-line parameter tuning in [Bar05] but up to now its advantages have not been exploited for on-line parameter tuning as it is suggested in this work.
A general problem in the optimization literature is how to conduct comparisons. One aspect is that comparing algorithms is complicated due to the influence of control parameter settings which make a comprehensive comparison very time-consuming (see e.g. [Zie06f] and [Zie09] which have been generated within the scope of this work, but also [Mez06a]). Another aspect is that more sophisticated algorithms, e.g. algorithms using parameter control, usually consist of several components. Generally, it is not clear which of these components are effective means for improving the performance and which ones

only complicate the algorithms without contributing substantially to a good performance, i.e. they could also be omitted. In the literature this aspect is usually not considered. Therefore, an extensive comparison of several adaptive algorithms will be shown in the following where individual components are regarded separately to allow conclusions about their performance.
In this chapter first a discussion of literature about adaptive parameter control for Differential Evolution and Particle Swarm Optimization is given. Afterwards, it is described how methods from Design of Experiments may be applied for adaptive parameter control. Furthermore, for DE a comparison of several adaptive variants is shown which regards the individual components of these methods. The chapter ends with a short summary and indications for future work.

7.1 Related Literature

Because in this work the focus is on DE and PSO, the literature review will concentrate on adaptive strategies used for these algorithms. For other EAs also methods for adaptive parameter tuning have been developed (in fact, self-adaptive parameter control is a characteristic of ESs), and an overview can be found in [Lob07].

7.1.1 Adaptive Differential Evolution in the Literature

In [Hua06] the so-called "Self-adaptive Differential Evolution" algorithm (SaDE) is described. Not only the parameter settings but also the DE strategy (see Section 3.5.2) is changed during an optimization run. Four different strategies are used which are DE/rand/1, DE/current-to-best/2, DE/rand/2 and DE/current-to-rand/1. In a previous paper only the two firstly mentioned strategies were used [Qin05]. All strategies have the same probability to be selected in the beginning of an optimization run. Later, the probabilities are modified by calculating the ratio r_s of the number of trial vectors which successfully entered the next generation divided by the number of generated trial vectors for every strategy. This number is sampled for 20 generations (which is called learning period). It should be noted that the learning period is a newly introduced parameter for which no further information is given regarding its sensitivity to changes. In a previous paper the learning period was set to 50 generations [Qin05]. No attempt was made to adapt NP. Settings for F are randomly taken from a normal distribution with mean 0.5 and standard deviation 0.3 where F is limited to $F \in (0, 2]$. The mean and the standard deviation are kept constant over the run. As a result, F is not constant but there is no feedback from the search. This means that one parameter is substituted by two new parameters (mean and standard deviation). However, it can be expected that the settings of the new parameters will work for a larger range of functions (but maybe at the expense of longer optimization runs). Because the authors assume that the setting of CR is more sensitive to the optimization problem (in a previous paper they also stated that F is more related to the convergence speed [Qin05]), CR is adapted based on feedback from the search. Like F, it is also selected from a normal distribution with a fixed standard deviation (0.1) but the mean CRm does not stay constant. In the beginning $CRm = 0.5$ is used. Each individual is randomly assigned a CR value that is kept constant for five generations and then a new value is calculated using the same distribution. After 20

generations, CRm is recalculated based on r_s, and the described procedure is repeated. Although CR is adaptively chosen, some new parameters are introduced: The standard deviation of the normal distribution, the number of generations for which CR stays constant and the period after which CRm is changed (in a previous paper, it has been 25 generations [Qin05]). Values are given for these parameters but the effects of changing them is not discussed. To speed up convergence, in [Hua06] a local search method (Sequential Quadratic Programming) is used every 500 generations on several individuals including the best one found so far.
In [Liu05] the DE parameters F and CR are dynamically adjusted based on fuzzy logic controllers. The resulting algorithm is called "Fuzzy Adaptive Differential Evolution" (FADE). Knowledge from empirical examinations regarding the connection between control parameter settings and the performance of the algorithm is used to generate the fuzzy rules. Only one DE variant is selected (DE/rand/1/bin) without discussing the influence that a change of the variant may have. In a comparison with a DE algorithm with static parameter settings of $F = CR = 0.9$ the authors conclude that for problems with a low dimensionality of $D = 3$ no advantage of the adaptive method is noticeable but for a higher dimensionality of $D = 50$ it outperforms the method with static parameter settings. Of course a considerable amount of knowledge of the algorithm designer has to be included to derive the fuzzy rules. No attempt was made to change the population size, instead it is set to $10 \cdot D$.
Another approach to adapt control parameters of DE using fuzzy logic is described in [Xue05] for multi-objective optimization. The algorithm is called "Fuzzy Logic Controlled Multi-Objective Differential Evolution" (FLC-MODE). In the mutation operator two vector differences are used which are both scaled by parameter F. Furthermore, the best individual is added weighted with parameter γ. These two parameters are adaptively controlled by considering the population diversity (which is calculated as the number of non-dominated solutions divided by the population size in [Xue05]) and the generation percentage (which equals the ratio of the number of already performed generations to the maximum generation). Initial values are set to $\gamma = 0.7$ and $F = 0.5$, ranges are $\gamma \in [0.05, 1]$ and $F \in [0.1, 3]$, and the fuzzy logic controller is evaluated every five generations. Because of the fuzzy logic controller new parameters are introduced which can be manually tuned or optimized by a GA. Nothing is mentioned in [Xue05] about the effort to tune these parameters, so it might be assumed that the algorithm does not become easier but the tuning of parameters is only shifted.
A self-adaptive DE algorithm called jDE is described in [Bre06a], and its extension jDE-2 is presented in [Bre06c, Bre06b]. Here each individual $\vec{x}_{i,G}$ (with $i \in \{1, \ldots, NP\}$) has its own settings of control parameters, denoted $F_{i,G}$ and $CR_{i,G}$, so the self-adaptive mechanism is conducted at the individual level. The control parameters are updated using the following equations:

$$F_{i,G+1} = \begin{cases} F_l + rand_1 \cdot F_u & \text{if } rand_2 < \tau_1 \\ F_{i,G} & \text{otherwise} \end{cases} \tag{7.1}$$

$$CR_{i,G+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ CR_{i,G} & \text{otherwise} \end{cases} \tag{7.2}$$

where $\tau_1 = \tau_2 = 0.1$, $F_l = 0.1$, $F_u = 0.9$, and the random numbers $rand_j \in [0, 1]$ (with $j \in \{1, 2, 3, 4\}$) are from a uniform distribution. Thus, $F \in [0.1, 1.0]$ and $CR \in [0, 1]$

are adapted based on the probabilities τ_1 and τ_2. NP is a fixed parameter for jDE that is set to 100 in [Bre06a] whereas $NP = 200$ is used in [Bre06c, Bre06b]. In [Bre06c] the jDE-2 algorithm sets boundary-violating individuals with equal probability either to the boundary that they are violating or reflects them from the bound (in [Bre06a] the respective component is set to the boundary). Furthermore, in contrast to the jDE algorithm that uses only strategy DE/rand/1/bin [Bre06a], in jDE-2 three strategies are used (DE/rand/1/bin, DE/current-to-best/1/bin, DE/rand/2/bin) and each strategy has its own control parameter settings. Every strategy is used by an equal part of the population in each generation, and every individual changes its strategy in each generation. To improve the diversity, a further modification of the jDE-2 algorithm is that the k worst individuals are replaced every l generations with randomly chosen individuals where $l = 1000$ and $k = 70$ in [Bre06c]. Unfortunately, no comparison is available that evaluates the effect of the different procedures to improve the jDE algorithm. For constraint-handling, the modified replacement method as described in Section 4.2 is used in [Bre06c, Bre06b] whereas in [Bre06a] unconstrained optimization problems are regarded. In [Bre06b] the values of the control parameters are shown for some functions of the CEC06 Special Session on Constrained Real Parameter Optimization (see Section 4.4). It can be seen that the values chosen by the algorithm vary considerably for different functions.
In [Ali04] a constant setting of $CR = 0.5$ is used and the population size is set dependent on the dimension ($NP = 7 \cdot D$ as well as $NP = 12 \cdot D$ have been tried). F is adaptively controlled based on the maximum and minimum objective function values f_{max} and f_{min} of the current generation:

$$F = \begin{cases} \max\left(l_{min}, 1 - \left|\dfrac{f_{max}}{f_{min}}\right|\right) & \text{if } \left|\dfrac{f_{max}}{f_{min}}\right| < 1 \\ \max\left(l_{min}, 1 - \left|\dfrac{f_{min}}{f_{max}}\right|\right) & \text{otherwise} \end{cases} \tag{7.3}$$

where $l_{min} = 0.4$ is used in [Ali04]. As a result, in early stages of the optimization run when the maximum and minimum objective function values usually differ considerably from each other, a relatively large value of F is used. In contrast, in later stages of the run when objective function values are similar to each other, a small value of F will result. The authors do not give any indication how this method can be applied for constrained optimization problems. Apart from adapting F, the authors also include several other new techniques in their DE algorithm, e.g. the use of an auxiliary population, the use of pre-calculated differentials and also local search, but here the focus is on adaptive parameter setting, thus the other mechanisms are not further discussed.
In [Abb02] a self-adaptive variant of the multi-objective DE algorithm presented in [Abb01] (see Section 5.3.2) is given, called "Self-adaptive Pareto Differential Evolution" (SPDE). F is taken from a normal distribution $N(0,1)$ with mean 0 and standard deviation 1. In contrast, CR is self-adapted on the individual level, i.e. each individual has its own value CR_i. The corresponding value for a newly generated individual is calculated as follows:

$$CR_i = CR_{r_1} + N(0,1) \cdot (CR_{r_2} - CR_{r_3}) \tag{7.4}$$

where r_1, r_2 and r_3 are randomly chosen indices (see also Section 3.5.1). Furthermore, an additional mutation operation is inserted that is conducted in dependence on a mutation rate MR_i that is determined in the same way as CR_i. The crossover operation is done

differently than in DE/rand/1/bin (see Equation 3.4) as no target vector is included:

$$u_{i,j,G} = \begin{cases} v_{i,j,G} & \text{if } rand_j \leq CR_i \text{ or } j = k \\ x_{r_1,j,G} & \text{otherwise} \end{cases} \tag{7.5}$$

where $i \in \{1, \ldots, NP\}$ and $j \in \{1, \ldots, D\}$.
In [Zah02] theoretical results about the expected variance after the application of the variation operators as well as empirical knowledge is used to adapt control parameters on the component level based on population diversity. The goal is not to get the fastest possible convergence but to avoid premature convergence. In further work [Zah03a] also subpopulations are used which increase the diversity and also allow an easier execution of the algorithm on parallel machines. A new parameter still has to be chosen but parameters F and CR are adaptively set.
Most adaptive versions of DE try to tune F and/or CR but in [Teo06] the focus is on adapting the population size. Thus, the developed algorithm is called "Differential Evolution with Self-Adapting Populations" (DESAP). The author tests two different versions where the corresponding parameter is encoded in each individual and the population size is determined by building an average over the population. It is not clear which advantage is achieved by giving each individual an additional parameter because the population size is a parameter on the population level. Different trends of the population size are found for the different versions, and often the population size hardly changes from its initial value $NP = 10 \cdot D$. For five test functions a similar performance is found as for a DE with fixed population size.

7.1.2 Adaptive Particle Swarm Optimization in the Literature

For PSO it is harder to find adaptive approaches. Descriptions of adaptive approaches are often restricted to one single paper without further development like it was done e.g. for jDE or SaDE. The only method that gained some popularity up to now is a parameterless PSO algorithm called TRIBES developed by Maurice Clerc [Cle06]. This algorithm starts with only one particle and adds (and removes) particles based on the performance of particles. The particles are organized in so-called tribes which are subsets of the population that share their personal best positions with each other while the information flow between tribes is more restricted. The author concludes based on several test functions that TRIBES has a competitive performance with several state-of-the-art algorithms [Cle06]. The author also points out that the gain in ease of use might sometimes mean a loss in effectiveness.
In contrast to Clerc's algorithm that does not need any parameters to be set, other approaches mostly concentrate on eliminating some of the parameters but often at the expense that new parameters are introduced. These algorithms might still be easier to use than a standard PSO algorithm if the number of parameters is reduced and/or the new parameters are more robust, meaning that the performance of the algorithm varies less with changing parameter settings. Unfortunately, there do not exist any comparisons concerning the effectiveness of the algorithms (it has to be noted that in this case the effort for parameter tuning would have to be considered [Cle06]). In [Zha07] parameters c_1 and c_2 are adaptively assigned based on a clustering analysis. No attempt is made to also set the inertia weight w adaptively although e.g. in a parameter study conducted within the

scope of this work it is shown that the inertia weight might have a considerable influence on the performance of PSO [Zie09]. The value of w is adaptively controlled in [Iwa06b] based on the average velocity of all particles but no attempt is made to adaptively tune c_1 and c_2 as well. In [Hsi08] the population size is adaptively changed based on the fact if the global best position has been updated in previous generations.
Approaches for adaptive parameter setting still need to be developed further for PSO. Because many ideas used in PSO are derived from other EAs, inspiration might again also be taken from works like [Ang95, Eib99, Bey01, Liu06, Zam07, Hua07b, Lob07].

7.2 Design of Experiments for Adaptive Parameter Control

For both DE and PSO it has been shown that there are interactions between the control parameters [Kuk06d, Zie09]. Unfortunately, most of the adaptive approaches discussed in the previous section do not consider this problem. Similarly, when conducting parameter studies often only one parameter at a time is varied, neglecting the possibility of interaction effects [Eib99]. In contrast, methods from Design of Experiments (DoE) [Mon01, Mye02] can be used to adaptively tune parameters while also regarding interaction effects. In the following a short introduction to Design of Experiments is given. Afterwards the application of DoE methods for adaptive parameter control for DE as well as PSO is presented.
Design of Experiments comprises statistical techniques for model building and optimization. It is applicable for identifying the most influential factors in an experiment with many factors (screening) and it is also useful for reliability analysis and robust design optimization [Sim04]. Using DoE it can be determined statistically if changing a certain parameter yields a significant effect or if performance variations are caused by randomness. Not only the influence of single parameters can be analyzed but also the interaction of parameters. Due to sophisticated designs (settings of input variables) less test runs are necessary as if only one of the parameters is regarded at a given time. DoE can be used for optimization purposes [Deb05] but in this work DE and PSO are used for the actual optimization while the ability of DoE methods to draw conclusions from only little information is employed for the adaptive setting of control parameters.
DoE was developed in the 1930s for agricultural experiments [Kle04]. Early applications of DoE were based on real (non-simulated) systems while later problems also included simulations. A difference between these two variants is that in real systems random components exist that are accounted for by taking the same data multiple times (replicates) while simulations are often deterministic. This is also termed design and analysis of computer experiments (DACE) [Sim04] and it is used e.g. for building surrogate models in optimization [Ong04, Jin05]. Although the problem examined here is based on computer experiments, the analysis is done with DoE methods instead of considering DACE. This is due to the fact that the experiments involve randomness because the process of generating new solutions is observed which contains a stochastic component for evolutionary algorithms. In each generation a performance measure is calculated that is regarded as one replicate.
The problem examined here can be characterized more specifically as evolutionary operation (EVOP) [Mye02]. It is a method for continuous monitoring and improvement of a

process that equals the optimization run here. It is especially suitable for processes with varying performance. This is usually the case for an optimization run as the task changes from exploration in the early stages to exploitation in the later stages.
The first step in applying DoE is the choice of an appropriate design. The most simple designs are called two-level factorial designs, meaning that for every factor (parameter) two settings are examined at a given time. Figure 7.1 shows a graphical representation of two-level factorial designs where on the left two factors A and B are regarded whereas on the right a third factor C is also included. All of these factors have a low and a high setting (indicated by "-" and "+", respectively). It can be seen that four design points have to be checked for two parameters whereas eight design points are involved for three parameters. Half of the design points use the upper and the lower level of each parameter, respectively. If more information is needed, e.g. if it is necessary to correctly reflect the curvature of a function, response surface designs are used [Mye02]. Usually two-level factorial designs are employed for EVOP. An alternative is the use of a simplex design [Mye02] because less runs are required but the determination of interaction effects is complicated. Because control parameters of DE and PSO are expected to have interactions, a two-level factorial design is used here.

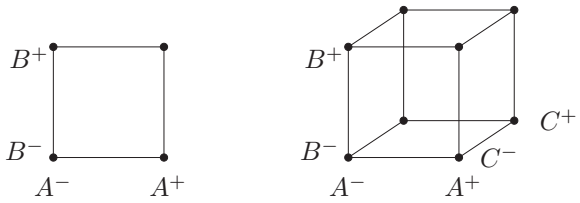


Figure 7.1: Two-level factorial designs with two and three factors

Results are evaluated using a statistical method called ANOVA [analysis of variance]. In the following some basics of ANOVA are given. A detailed introduction can be found in [Mye02].
If a two-level factorial design with factors A and B is assumed, the main effect $E(A)$ of factor A is calculated by computing the average process outcome $\bar{Y}_{A^+}$ with high setting of A and subtracting the average outcome with low setting $\bar{Y}_{A^-}$:

$$\begin{aligned} E(A) &= \bar{Y}_{A^+} - \bar{Y}_{A^-} \\ &= \frac{1}{2n}\left(Y_{A^+,B^+} + Y_{A^+,B^-} - Y_{A^-,B^+} - Y_{A^-,B^-}\right) \end{aligned} \tag{7.6}$$

where n is the number of replicates. Y_{A^+,B^+} is the sum of the process outcome y of all replicates with high level of factor A and high level of factor B:

$$Y_{A^+,B^+} = \sum_{i=1}^{n} y_{A^+,B^+,i}. \tag{7.7}$$

Y_{A^+,B^-}, Y_{A^-,B^+} and Y_{A^-,B^-} are calculated in the same way.
Similar to the main effect of factor A, the interaction effect of factors A and B is calculated:

$$E(AB) = \frac{1}{2n}\left(Y_{A^+,B^+} + Y_{A^-,B^-} - Y_{A^+,B^-} - Y_{A^-,B^+}\right). \tag{7.8}$$

Using the calculated effects, the sum of squares is calculated for main effects and interactions. This is shown for factor A in the following:

$$SS_A = n \cdot (E(A))^2. \tag{7.9}$$

Additionally, the sum of squares for the model error is needed. First, the total sum of squares is calculated according to

$$SS_T = \left(\sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{n} y_{ijk}^2 \right) - \frac{1}{4n} \left(\sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{n} y_{ijk} \right)^2 \tag{7.10}$$

where a and b are the numbers of levels for factor A and B, respectively, and $a = b = 2$ is used in this work.
The result can be used to determine the sum of squares of the model error:

$$SS_E = SS_T - SS_A - SS_B - SS_{AB}. \tag{7.11}$$

With the sum of squares and the degrees of freedom df for each component, the mean square MS can be calculated:

$$MS = \frac{SS}{df}. \tag{7.12}$$

The degrees of freedom are given in Table 7.1 [Mon01]. The mean square for the error can only be calculated if $n \geq 2$ holds for the number of replicates.

Table 7.1: Degrees of freedom

Effect	Degrees of freedom
A	$a - 1$
B	$b - 1$
AB	$(a - 1) \cdot (b - 1)$
Error	$a \cdot b \cdot (n - 1)$
Total	$a \cdot b \cdot n - 1$

A value F_0 can be calculated for every main effect and interaction using the corresponding mean square as well as the mean square of the error MS_E. The following equation shows the calculation of F_0 for factor A:

$$F_0 = \frac{MS_A}{MS_E}. \tag{7.13}$$

F_0 is used for checking the hypothesis that the corresponding effect is significant by comparing F_0 to a reference value. Reference tables are given in dependence on the confidence coefficient α [Mon01]. If the calculated F_0 is larger than the reference value, the associated effect is significant, e.g. with a confidence of 95% if $\alpha = 0.05$ is used.

7.2.1 Adaptive Differential Evolution based on Design of Experiments

Differential Evolution basically contains three control parameters: F, CR and NP. In this work the application of DoE methods for the adaptive adjustment of the control parameters F and CR is shown. First, single-objective optimization is discussed as also presented in [Zie06c]. Later a variant applicable for multi-objective problems is shown as introduced in [Zie07b] (both of these papers have been generated within the scope of this thesis).
For a two-level factorial design two settings of each parameter are regarded, respectively. One possibility for initializing would be to use random values but usually it is beneficial to incorporate available information. Therefore, the initial values are set according to guidelines from literature: $F_- = 0.5$, $F_+ = 0.9$, $CR_- = 0.2$ and $CR_+ = 0.9$ (see Section 3.5.3). Every combination of F and CR is applied to one fourth of the population. Instead of using fixed subpopulations, it is chosen randomly in each generation which parameter settings are used for which individuals. Otherwise, a different development of the subpopulations might influence the results.
For the calculation of effects the percentage of successful trial vectors (successful means that the solution entered the next generation) to generated trial vectors is computed for every parameter combination. If this percentage is used for the calculation of effects, successful trial vectors are rewarded regardless of the amount of change. As a result, parameter values might be favored that induce small improvements of many individuals. Other parameter settings might lead to larger improvements but for fewer individuals. Because these settings might be essential to escape from local optima, the ratio of successful trial vectors to all trial vectors is weighted with the average improvement for the respective parameter combination. The improvement is based on objective function values for feasible individuals and based on constraint violations for infeasible individuals. If an individual was infeasible in the previous generation but it is feasible in the current generation, no improvement can be calculated. Thus, the individual is assigned the average improvement of the other individuals with the same parameter combination. Another possibility would be to assign an arbitrary high number.
The flowchart in Figure 7.2 gives an overview about the adaptive adjustment of CR and F. After a new generation has been generated, it is checked if there is a significant effect of the parameters. If not, the next generation is built that is regarded as a further replicate in the DoE analysis. Because the magnitude of effects may change over time which may complicate the detection of significant effects, the DoE measures are reset after some time if no significant effect has been found. Several ways of determining a suitable time for resetting the DoE analysis have been tested in [Wan07] (which is a student project developed within the scope of this thesis):

- If the average improvement of the current generation is only a small fraction (1%, 10%) of the improvement averaged over all generations that participate in the current DoE analysis.

- If the average improvement of the current generation is only a small fraction (1%, 10%) of the average improvement of the first generation of the current DoE analysis.

- After a predetermined number of generations (5, 10, 20, 30).

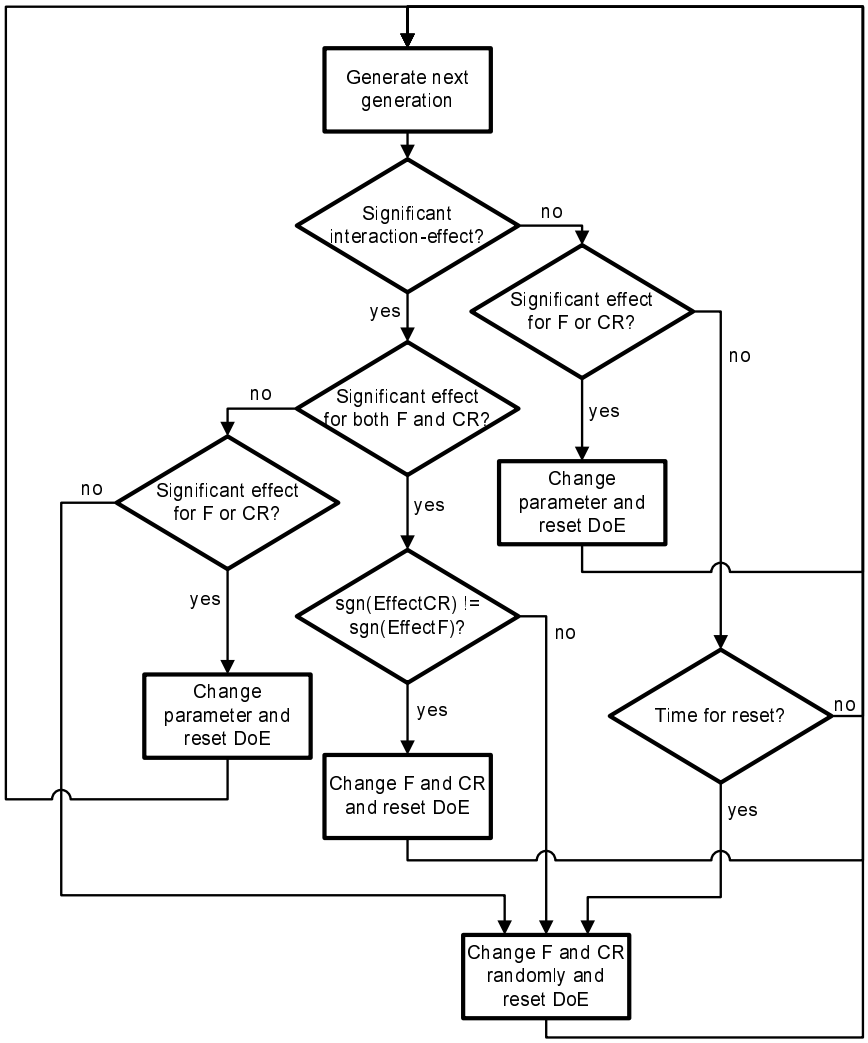


Figure 7.2: Flowchart for adaptive adjustment of F and CR

In [Mye02] it is suggested to restart the DoE calculations if after five to eight cycles (generations) no significant effects are detected. At first this restriction seemed to be too strict for the regarded problem whereas the first two possibilities seemed to be more promising due to their adaptive nature. However, it was found out in [Wan07] based on the test functions from [Lia06a] that resetting after a predetermined number of generations yields the best results. More specifically, resetting after 10 generations showed the best performance.
If a significant main effect is detected, the corresponding parameter is adjusted. If the interaction effect is also significant, it is checked additionally if both main effects are significant and have the same sign. In this case a definite decision in which direction settings should be changed is difficult, so a random change is made. A random modification is also done if an interaction effect is significant but the corresponding main effects are not. Instead of making a random change, in both cases it is also possible to disperse the parameter settings by decreasing the lower setting and increasing the higher setting, if possible without crossing the limits of the respective parameter.
In the present implementation of adaptive control it is concluded from the statistical analysis which parameter should be changed but no statements are derived about the

preferable amount of change. Instead, fixed step sizes of 0.05 and 0.1 are used. Based on the limits of parameters that were taken from literature ($F \in [0, 2]$ and $CR \in [0, 1]$, see Section 3.5.3), if the calculation of effects suggests to increase parameter settings, the lower value is increased by 0.1 to adapt it towards better settings. The higher value is increased by 0.05 to check if larger settings give even better results without moving too far from an already discovered good setting. Similarly, the lower value is decreased by 0.05 and the higher value is decreased by 0.1 if lowering of parameter settings is desired. In this process it is avoided that high and low settings have the same value because this would prevent further calculations of effects. Additionally, it is ensured that neither F nor CR will be equal to or below zero and that the parameters do not increase over the specified limits.
If this adaptive parameter control should be applied for multi-objective optimization, the basic structure can remain identical. Only the performance measure that is the basis of the DoE analysis must be varied because the same formulation as for single-objective optimization is not possible. Again, it should be based on the results of the selection operation. Therefore, a performance value is introduced for each parameter combination: If a trial vector dominates the corresponding target vector, the performance value of the respective parameter combination is increased by 2. If both vectors are non-dominated, the performance value is raised by 1. If the trial vector is dominated by the target vector, the respective parameter combination did not lead to an improved solution, so it does not get an increase in the performance value.
Apart from the performance measure, it might also be reasonable to change the initial settings of F and CR for multi-objective optimization. Thus, the result from [Kuk04a] would be reflected that suitable parameter settings for multi-objective optimization often differ from good settings for single-objective optimization.
An evaluation of the proposed adaptive approach for single-objective optimization has been shown in [Zie06c] based on the power allocation problem described in Section 4.2.4. The adaptive approach is compared with a DE algorithm using fixed parameter settings which have been tuned as shown in [Zie06f]. It is concluded that the adaptive approach needs a higher population size of $NP \geq 52$ to reach convergence rates of 100% in contrast to the algorithm with tuned fixed settings that achieves the same convergence rate with $NP \geq 30$. The average number of function evaluations is higher for the adaptive approach than for tuned fixed settings but the computational cost for parameter tuning has to be considered also, so the newly introduced adaptive approach is considered promising.
The multi-objective variant of this method has been tested in [Zie07b] based on the test set for the Special Session on Performance Assessment of Multi-Objective Optimization Algorithms at the Congress on Evolutionary Computation 2007 (see also Section 4.4). Despite the difficulty of the test set, the Pareto-optimal fronts have been found in most cases.

7.2.2 Adaptive Particle Swarm Optimization based on Design of Experiments

Similarly as described for Differential Evolution, an adaptive parameter control based on DoE can be implemented for Particle Swarm Optimization. For DE the parameters F and CR which influence the mutation and recombination processes have been chosen to be adaptively controlled. Here their equivalents w, c_1 and c_2 are selected for the adaptive control because they determine the creation of new solutions (see Section 3.6.1). The

adaptive parameter control has been tested for multi-objective optimization in [Zie07a] (which is a paper that was generated within the scope of this thesis), and a variant for single-objective optimization is newly described here.
Due to the increased number of parameters, it is possible to calculate interaction effects of higher order. Because they are usually of secondary importance [Mon01], these terms are neglected here. As a result, the general procedure is very similar to the one described for Differential Evolution in the previous section.
The main difference is the choice of the performance measure for the DoE analysis. For DE the performance measure is based on the selection procedure. Because in PSO selection is not applied, the performance measure may be calculated in dependence on the update of the best positions. Similar to the case of multi-objective DE, in [Zie07a] a performance value is introduced for every parameter combination that is increased by 2 if a newly generated solution dominates a member of the *gbest* archive, and it is increased by 1 if it only dominates its previous personal best position.
For single-objective optimization, the performance measure can be constructed analogous to the one for single-objective DE. The number of newly accepted personal best positions divided by the number of all generated new positions for each parameter combination is regarded, multiplied with the achieved improvement in objective function values or constraint violations, respectively.
Another difference to DE is that because three parameters are subject to the adaptive control here, $2^3 = 8$ parameter combinations have to be regarded which are applied to one eighth of the population, respectively (randomly chosen in every generation). Consequently, it might be reasonable to choose a rather high setting for the population size to have a larger basis for determining the significant effects.
Ranges and initial settings of the parameters should be selected based on recommendations from literature. Thus, e.g. $w \in [0, 1]$, $c_1 \in [0, 4]$ and $c_2 \in [0, 4]$ might be chosen (see Section 3.6.3) but settings of 0 should be excluded. As initial settings the values $\vec{w} = \{0.4, 0.8\}$, $\vec{c_1} = \{1.2, 1.9\}$ and $\vec{c_2} = \{0.5, 1.2\}$ have been chosen in [Zie07a]. The reasons are that most authors use values of $0.5 \leq c_1, c_2 \leq 2$ [Par02a, Xie04b], exploration should be emphasized in the beginning (meaning that c_1 should be larger than c_2), and for the inertia weight often values around $w = 0.7$ produce good results [Tre03, Cle02] (see also Section 3.6.3).
Same as the adaptive approach for multi-objective DE, the multi-objective PSO algorithm with adaptive parameter setting has been tested based on the demands for the Special Session on Performance Assessment of Multi-Objective Optimization Algorithms at the Congress on Evolutionary Computation 2007 [Zie07a]. The results of PSO are worse than the performance of DE but it must be noted that a rather basic multi-objective PSO algorithm has been taken as basis. As discussed in Section 5.4.1, it seems to be necessary to include additional operators in multi-objective PSO to enhance its exploratory capability. Thus, the results are expected to improve if suitable operators are added.

7.3 Evaluation of Adaptive Variants for Differential Evolution

The evaluation of optimization algorithms and especially the analysis of adaptive variants is often complicated because several features are modified concurrently. For Differential

Evolution these features may be adaptation of parameters, adjustment of the strategy and addition of local search or other special operators. Thus, it is difficult to analyze which of these procedures is actually responsible for changes in the performance and which ones only complicate the algorithm without contributing anything to its performance. Therefore, in the following a detailed analysis of several adaptive approaches from literature is shown by regarding individual components of these algorithms separately to examine their effectiveness. This study has also been published in [Zie08e].
As specified in Section 7.1.1, several adaptive DE algorithms have been presented in the literature. For this study some of the most promising algorithms have been selected which were also presented in the CEC06 Special Session on Constrained Real Parameter Optimization (see Section 4.4): jDE and SaDE. To limit the complexity of the study, only jDE and not jDE-2 is considered. Both jDE and SaDE assign an individual setting for F and CR to each individual and adapt these settings during the optimization run. A difference is that SaDE considers feedback from the search whereas jDE uses self-adaptive adjustment of parameters. Additionally, in SaDE several DE strategies are used and the probabilities for selecting a specific strategy are adapted. Furthermore, the adaptive DE based on DoE introduced in Section 7.2.1 is tested. To simplify the analysis, the same constraint-handling according to the modified replacement method described in Section 4.2 and the same handling for boundary constraints according to Equation 4.2 is used for all algorithm variants here, and local search is omitted.
More precisely, in order to examine which components of the adaptive approaches are especially effective, the following algorithm variants are considered:

- V1: DE_DoE
- V2: jDE'
- V3: SaDE'
- V4: jDE' with strategy control as in SaDE'
- V5: SaDE' with adaptive F
- V6: SaDE' without strategy control
- V7: SaDE' with eight strategies
- V8: DE_DoE with strategy control as in SaDE'

Here, V1 is the DE_DoE algorithm presented in [Zie06c] and Section 7.2.1. V2 and V3 are close to the original jDE and SaDE algorithms given in [Bre06a] and [Hua06], respectively (as described above, the same handling for constraint functions as well as boundary constraints is used for all algorithms and local search is omitted). In V4 the jDE' algorithm is used for parameter control while a strategy control as in SaDE' is also employed. In V5 SaDE' is used as basis and F is adaptively controlled in the same way as CR (with an initial mean of $Fm = 0.5$ and a standard deviation of 0.3, like suggested in [Hua06]). To examine the influence of strategy control on SaDE', V6 only uses the parameter control of SaDE' without strategy control. Because several more strategies exist besides the ones employed in SaDE' (see Section 3.5.2), in V7 all strategies given in Table 3.1 are used. In V8 the strategy control of SaDE' is employed in DE_DoE.

For all experiments a population size of $NP = 52$ has been used (for DE_DoE the population size must be divisible by four). The maximum number of function evaluations is 500,000, and the same test set is used as described in Section 4.4 (where results for g20, g22 and g23 are not shown because none of these algorithms was able to find the global optimum).
The success rate of all algorithm variants is given in Table 7.2 and the average number of function evaluations for convergence is shown in Table 7.3.

Table 7.2: Success rate in %

Problem	V1	V2	V3	V4	V5	V6	V7	V8
g01	100	100	100	100	100	100	100	100
g02	76	72	44	88	20	28	16	84
g03	0	0	16	0	92	100	92	0
g04	100	100	100	100	100	100	100	100
g05	44	32	100	88	100	92	64	84
g06	100	100	100	100	100	100	100	100
g07	60	96	100	100	100	100	4	72
g08	100	100	100	100	100	100	100	100
g09	96	100	100	100	100	100	100	100
g10	28	40	100	96	100	100	0	76
g11	92	100	100	100	100	100	100	100
g12	100	100	100	100	100	100	100	100
g13	0	0	16	0	44	4	24	0
g14	52	0	100	100	44	4	0	72
g15	92	60	100	100	100	100	48	92
g16	100	100	100	100	100	100	100	100
g17	4	0	36	4	32	4	8	8
g18	68	84	84	80	72	72	60	80
g19	28	24	48	100	60	4	0	72
g21	20	60	68	80	92	68	36	40
g24	100	100	100	100	100	100	100	100

To evaluate the different methods of parameter control, the strategy control of SaDE' has been disabled for V6, and results for V6 are compared with jDE' (V2) and DE_DoE (V1). The success rate of V6 is better than for V1 and V2 for many functions although there are also exceptions (g02, g14, g19). The average number of function evaluations for convergence is also often comparable to or better than the results for V1 and V2. It is concluded that the parameter control of SaDE' is generally superior to the methods employed in jDE' and DE_DoE. This may be explained by the use of different control parameters for each individual and the adjustment of CRm based on feedback from the search. For future work it could be tried to improve the performance of DE_DoE by also using different settings for each individual.
Furthermore, it is tested if advantages can be gained by controlling F adaptively in the same way as done for CR (V5), therefore variants V3 (SaDE') and V5 are compared. Regarding the success rate, it depends on the function if a better performance is reached. For g03, g13, g19 and g21 the performance is better with adaptive F whereas for g02, g14 and g18 the performance becomes worse. The average number of function evaluations

Table 7.3: Average number of function evaluations for convergence

Problem	V1	V2	V3	V4	V5	V6	V7	V8
g01	28180	20605	24277	22902	24005	20221	18039	25165
g02	194869	68767	447694	62372	37901	206936	325195	182247
g03	-	-	327083	-	280131	151734	151789	-
g04	37488	27761	15868	29775	14792	16782	11345	17789
g05	255793	228783	102745	239549	171724	250927	176413	247343
g06	16858	13521	9416	16348	9238	8530	6827	12061
g07	199319	210108	51793	128084	97990	159242	64946	172512
g08	1356	1277	1491	1415	1445	1286	816	1338
g09	127950	49097	16077	43141	14200	19637	11662	46958
g10	233591	325477	60265	224051	164689	244380	-	171405
g11	41082	29114	22542	45232	21391	24357	10154	63000
g12	2313	2041	2002	2333	2044	1822	1448	2182
g13	-	-	247711	-	316994	373682	281284	-
g14	284388	-	47177	163763	321290	439743	-	126777
g15	173911	270119	60450	162129	98147	164098	280830	216409
g16	18812	15368	24880	16064	10777	25068	18222	15286
g17	406848	-	255101	431151	331256	443380	154002	275375
g18	111613	87950	40206	115294	31143	43680	17171	100533
g19	318225	351489	83244	246529	218128	396186	-	241520
g21	266365	151667	108141	174406	139919	171408	131463	185859
g24	4484	4823	4119	5321	3814	3767	3260	4124

for convergence is better for g02, g03, g09, g16 and g18 with adaptive F, and it is worse for g05, g07, g10, g13, g14, g15, g17, g19 and g21 (functions for which the results are similar are not specified here). Although the success rate averaged over all functions can be slightly improved by using an adaptive F (from 78% to 80%), the overall performance tends to become worse when considering the average number of function evaluations for convergence. For future work it would be interesting to observe the development of CRm and Fm over time to achieve more detailed insight in the way the adaptive parameter control interacts with the optimization problem.

To analyze the influence of strategy control, a comparison of each algorithm with and without strategy control is done. For the jDE' algorithm (V2 without strategy control and V4 with strategy control) the strategy control clearly results in improved performance concerning the success rate as it is always equal to or higher than the results for V4 (with exception of g18 where the success rate is slightly better without strategy control). The average number of function evaluations for convergence is often in a similar range. For some functions V2 is better (g06, g11, g18, g21, g24) and for other functions V4 is better (g02, g07, g09, g10, g15, g19), so no definite conclusion can be given regarding this performance measure.

Comparing SaDE' with and without strategy control (V3 and V6, respectively), the success rate with strategy control is better than or equal to the variant without strategy control except for function g03. Concerning the average number of function evaluations for convergence, V3 is also mostly better than V6 (for g05, g07, g09, g10, g13, g14, g15, g17, g19 and g21 whereas V3 was worse for g02 and g03).

For DE_DoE the variant V1 is compared with V8 that uses the strategy control of SaDE'. The success rate of V8 is always better than or equal to the results of V1. Concerning the average number of function evaluations for convergence, also an improvement can be seen for nearly every function when using strategy control (exceptions are only functions g11 and g15).
It is concluded that strategy control is able to improve the results for all regarded algorithms, especially concerning the success rate. To analyze the effect of using more than the four strategies given in [Hua06], variant V3 (SaDE') is compared to variant V7 that uses the parameter and strategy control described in SaDE' but with eight instead of four strategies. However, the results using V7 deteriorate for many functions concerning the success rate (exceptions are g03 and g13). Interestingly, the average number of function evaluations for convergence improves for several functions using eight instead of four strategies (exceptions are only g05, g07, g13, g15 and g21). The reason for this behavior is not yet clear. For future work the probabilities of the strategies should be monitored during optimization runs to develop a theory for this behavior.
In future work also the local search procedure employed in SaDE and the random substitution used in jDE-2 should be analyzed. Besides, in Section 7.1.1 many other adaptive approaches are mentioned which may also be examined.

7.4 Summary and Future Work

Several approaches for adaptive parameter control of Differential Evolution and Particle Swarm Optimization have already been presented in the literature but usually the influence of interactions between parameters is neglected. Thus, an approach based on Design of Experiments has been shown here that regards interactions without adding much computational complexity.
In general, more comparisons are needed that evaluate performance differences of adaptive approaches. In particular, it is often not clear which components of an adaptive algorithm are mainly responsible for an improved performance and which operators only add complexity but without contributing to the performance. Therefore, an extensive analysis of three adaptive approaches of DE was conducted. It showed that adapting the strategies yields good results. Furthermore, adaptive parameter control on the individual level showed favorable behavior. For future work these results can be used to further improve the adaptive approach using Design of Experiments presented in this chapter by employing strategy control as well as developing a method to introduce different parameter settings for each individual. Besides, there are still many operators from literature which may be tested for future work.
Similar mechanisms for adjusting the control parameters might also be used for PSO. Because the choice of the strategy for DE is similar to the selection of the neighborhood topology for PSO, it might also be experimented with different neighborhood structures. Some work about this is already described in [Men04b] where several topologies have been examined and the promising "Fully Informed Particle Swarm" has been presented (see also Section 3.6.2). It might also be possible to adaptively assign guides based on performance (e.g. it might not only be considered if the solution has the best performance in a certain neighborhood but also if it improved its performance recently) but this is left for future work.

Chapter 8

Summary and Future Work

In this chapter the most important findings of this work are summarized. Based on these results, indications for future work are given.

8.1 Summary

In science and engineering often tasks arise which can be stated as optimization problems: One or several measures should be maximized or minimized in dependence on some variables which themselves may be subject to certain constraints. Mostly, the users of optimization algorithms are not experts concerning optimization, leading to the need of easy-to-use optimization algorithms.
The application of optimization algorithms is often more complex than a user would assume. The problems begin with the definition of the objective function(s) which may be straightforward in some optimization problems but it is not necessarily simple to specify. A further problem may be the determination of the constraints because often users implicitly make some assumptions and do not insert them into the optimization tool. Additionally, reasonable boundaries should be set which encompass the global optimum and which limit the search space to reduce the computational effort.
The next question is which optimization algorithm to use. A vast amount of optimization algorithms exist, thus it is difficult to choose. Because in this work the focus is on easily applicable algorithms which should have global search capability, Differential Evolution and Particle Swarm Optimization have been selected. Both algorithms have a relatively low number of parameters, real-valued representation, fast convergence behavior and almost no requirement concerning properties of the objective function(s).
Besides these problems which are related to the definition of the optimization problem and the choice of optimization algorithms, especially real-world problems often include additional difficulties which is the main topic of this thesis.
Because real-world problems usually contain constraints, a suitable constraint-handling technique is necessary. As the focus of this work is on easily applicable methods, it should contain as few parameters as possible. This holds for the modified replacement method that does not need any parameter to be specified. Despite its simplicity, it was shown in this work that the modified replacement method achieves good results for both DE and PSO for the optimization of a power allocation problem from communications.
Furthermore, it was shown that the constraint-handling technique should be adjusted for optimization problems with equality constraints. Because the feasible space becomes very

small which interferes with the movement of the individuals (especially using DE), either techniques should be used which do not necessarily evaluate feasible solutions as better than infeasible solutions or which vary the allowed constraint violation during the run. In this work a method adhering to the latter principle has shown very good performance for a problem from yield analysis that requires an equality constraint to be fulfilled very precisely.

Multi-objective optimization is a very active research area that has a high relevance for real-world problems because mostly several objectives should be regarded simultaneously. In this work multi-objective extensions for DE and PSO have been presented. The optimization of an exemplary operational amplifier has been shown considering the gain, bandwidth and current consumption as objectives whereas the stability and the desired form of the signal have been formulated as constraints. Similar results have been reached as in a reference design that has been tuned by hand by an experienced designer. Moreover, the optimization algorithms offer the advantage of suggesting several trade-off solutions from which a decision maker may select the design to be manufactured.

Stopping criteria considerably influence the computational cost associated with an optimization problem. Especially for real-world problems with computationally expensive objective and constraint functions it is crucial to stop the execution of an optimization algorithm at a reasonable time. Despite their importance, examinations of stopping criteria for evolutionary algorithms are surprisingly rare in the literature. Sometimes it is mentioned that there are also other stopping criteria besides terminating after a predefined number of function evaluations but neither a detailed discussion and classification nor an evaluation of different implementations is available. In this work a large variety of stopping criteria has been presented and classified based on the property that they regard. It was shown that a similar classification is possible for single-objective and multi-objective optimization although the underlying measures are different. A thorough examination revealed that a distribution-based criterion that regards objective space by terminating a run if the difference of objective function values between the best and worst individual of a generation falls below a threshold yields the best results for single-objective optimization. Only one additional parameter has to be set for this stopping criterion. Especially for Differential Evolution it was sufficient for a large range of functions to set this parameter to one order of magnitude lower than the desired accuracy of the result, thus the effort for parameter tuning is low. Furthermore, only few computations are necessary for checking the criterion.

For multi-objective optimization performance measures may be used for defining stopping conditions but the evaluation often causes considerable computational cost. Therefore, it is better to rely on internal mechanisms of the algorithms for deriving conclusions about the state of the optimization run. Good results have been reached for both DE and PSO if an optimization run was terminated when in several consecutive generations no solutions have been generated which dominate an archive member. Furthermore, distribution-based criteria showed a good performance. In contrast to single-objective optimization, it is not checked whether the individuals are close to each other but rather the uniformity of the distribution of the solutions. For algorithms which internally calculate crowding distances, this measure can be taken as basis for a stopping criterion without much additional computational effort.

To simplify the application of optimization algorithms, especially for users who are inexperienced concerning optimization, adaptive approaches for the setting of control param-

eters have been examined. Because in the literature interactions between parameters are usually neglected, an approach for adaptive parameter setting based on Design of Experiments was presented that allows to regard both main effects and interaction effects of the parameters. Often it is difficult to compare different approaches with each other because several mechanisms differ, thus for Differential Evolution an extensive examination regarding individual components of adaptive algorithms was conducted. It was indicated that it is beneficial to use local control parameter settings depending on information about the state of the optimization run. A further result was that especially strategy control is very effective for improving the performance.

8.2 Future Work

Although many aspects of DE and PSO and especially their application to real-world problems have been examined here, there are still details which might be examined further. In Sections 4.5, 5.6, 6.3 and 7.4 already some indications for future work have been given, thus here only some of the most important ones are summarized.

Both Differential Evolution and Particle Swarm Optimization are heuristics which have been shown to work for many problems but there is no formal convergence proof. In the PSO community there are some efforts to derive a convergence proof (see Section 3.6.3) but mostly simplified versions are used for which it is questionable if results can be transferred to more sophisticated variants. For DE even less theoretical work is available. For future work it would be desirable to gain more theoretical insight.

For several problems Differential Evolution is superior to Particle Swarm Optimization which might be caused by its greedy selection scheme (see Section 4.4 or [Zie06a, Zie06b]). It was also shown that PSO might result in better performance if the control parameters are carefully tuned (see Section 4.2.4.2 or [Zie06f, Zie09]). Furthermore, PSO has an advantage in heavily constrained search spaces as the PSO particles are able to search them more efficiently than the DE individuals. Therefore, it cannot be said generally which of these algorithms performs better. This is also in accordance with the no free lunch theorem (see Section 2.3). For future work it should be examined which features exactly make an optimization problem more suitable for DE or PSO, as already started in [Lan07].

Concerning that topic, comparisons should also be conducted more carefully in the optimization literature. Because different settings of the control parameters may result in highly different performances, it is generally unfair just to compare implementations with one fixed set of parameters.

Another possibility is to build hybrid algorithms that combine favorable features from different algorithms. In [Cle06] it is stated that one of the most promising hybrids of PSO is a combination with DE. Hence, it might be interesting to further examine hybrid algorithms in future work. Some works can already be found in the literature, e.g. the neighborhood concept from PSO is used for a DE algorithm in [Wan08b] and a mixture of DE and PSO is presented in [Xu08]. For other EAs also several hybrids can be found in the literature where a local search procedure is used for fine-tuning of solutions. Thus, the possibly missing fine-tuning capability of one algorithm can be compensated by another algorithm that lacks the ability for global search. The discussion of a suitable time for switching from the global to the local technique is closely related to the topic of stopping

criteria, and similar mechanisms can be used for this task [Syr95, Jak04]. Because of the adapting step sizes of both DE and PSO, such hybrids might not be as necessary as for other evolutionary algorithms but advantages may still be gained as e.g. in [Hua06] a local search procedure is used to speed up convergence of DE.
Many aspects have to be considered for the successful optimization of real-world problems. In this work the presence of constraints of various types was discussed, and it was shown that especially for equality constraints there is still some work necessary. Among others, methods do not currently exist for optimizing problems with a mixture of several inequality constraints and equality constraints where the equality constraints might also have differing magnitude.
Multi-objective optimization is a topic that still poses problems for optimization tools, especially if the number of objectives increases above two or three which is also termed many-objective optimization in the literature. Not only the visualization techniques have to be improved to allow a better overview about results but also the algorithms themselves might be further enhanced, especially the method for efficiently estimating the crowdedness of solutions.
The stopping criteria shown in this work have mostly been evaluated on the basis of test problems. It would be interesting to examine their behavior for more real-world problems in future work.
The creation of adaptive methods for setting control parameters is difficult because an optimization problem within the optimization problem is generated. In fact optimization algorithms have been used in the literature for the tuning of parameters of another optimization algorithm that solves the actual optimization problem, e.g. in [Par02b] DE is used to tune the parameters of PSO. Many adaptive approaches eliminate the need for setting certain parameters but introduce some new parameters. If these new parameters are more robust than the substituted ones or the number of new parameters is smaller than the original number, this is already an improvement. Nevertheless, in future work these methods should still be enhanced.
A topic that has not been discussed in this work but that may be a source of difficulty for an optimization algorithm is a large number of variables of the objective function. This is also termed large scale optimization (there has also been a special session on this topic at the Congress on Evolutionary Computation 2008, indicating the interest of the community). The complexity of a problem usually rises with a large number of variables, and the size of the search space quickly increases which is also referred to as "curse of dimensionality". As a consequence, optimization algorithms are needed which sample the search space very efficiently in order to deal with these problems which may also arise in real-world problems. Both DE [Bre08, Zam08] and PSO [Hsi08] have already been tested for this kind of problems but it is apparent that still mechanisms need to be developed which explore large search spaces more efficiently.

Appendix A

Test Problems for Multi-Objective Optimization

In the following five test problems from [Zit00] are shown that are commonly used in multi-objective optimization. All of these test problems are bi-objective and require the minimization of functions f_1 and f_2 where $f_2 = g \cdot h$ is the multiplication of two functions g and h. The Pareto-optimal front can be calculated analytically for all of the following optimization problems by setting $g = 1$.
ZDT1 has a dimension of $D = 30$ and the variables are in the range $x_i \in [0, 1]$. The Pareto-optimal front is convex (see Figure A.1).

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \ldots, x_D) &= 1 + 9 \cdot \sum_{i=2}^{D} \frac{x_i}{D-1} \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \tag{A.1}$$

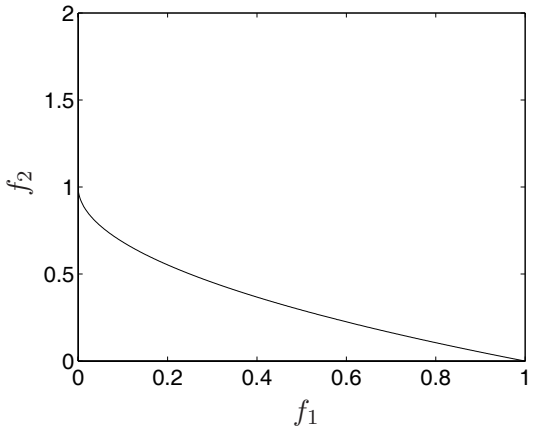


Figure A.1: Pareto-optimal front of ZDT1

ZDT2 has a non-convex Pareto-optimal front (see Figure A.2), a dimension of $D = 30$ and variables from $x_i \in [0, 1]$.

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \ldots, x_D) &= 1 + 9 \cdot \sum_{i=2}^{D} \frac{x_i}{D-1} \\ h(f_1, g) &= 1 - (f_1/g)^2 \end{aligned} \tag{A.2}$$

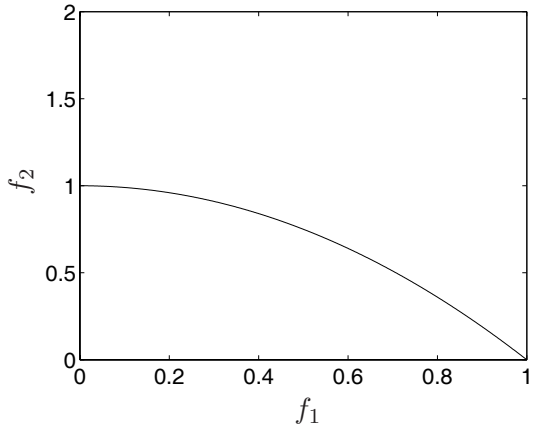


Figure A.2: Pareto-optimal front of ZDT2

ZDT3 also has a dimension of $D = 30$ and variables in the range $x_i \in [0, 1]$. The Pareto-optimal front is discontinuous (see Figure A.3).

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \ldots, x_D) &= 1 + 9 \cdot \sum_{i=2}^{D} \frac{x_i}{D-1} \\ h(f_1, g) &= 1 - \sqrt{f_1/g} - (f_1/g) \cdot \sin(10\pi f_1) \end{aligned} \tag{A.3}$$

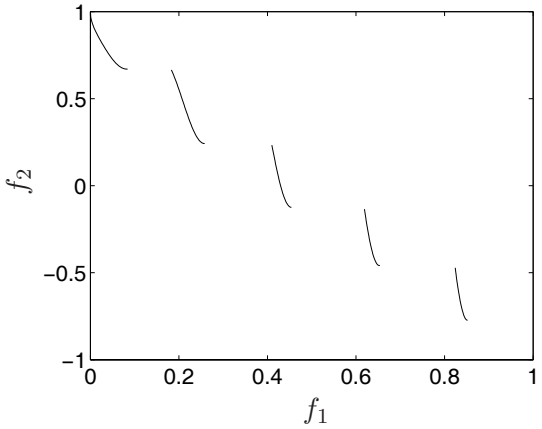


Figure A.3: Pareto-optimal front of ZDT3

The dimension of ZDT4 is $D = 10$. The variables are in the range $x_1 \in [0, 1]$ and $x_2, \ldots, x_D \in [-5, 5]$. ZDT4 is hard to solve because of 21^9 local Pareto-optimal fronts. The Pareto-optimal front is given in Figure A.4.

$$\begin{aligned} f_1(x_1) &= x_1 \\ g(x_2, \ldots, x_D) &= 1 + 10 \cdot (D-1) + \sum_{i=2}^{D} \left(x_i^2 - 10\cos(4\pi x_i) \right) \\ h(f_1, g) &= 1 - \sqrt{f_1/g} \end{aligned} \tag{A.4}$$

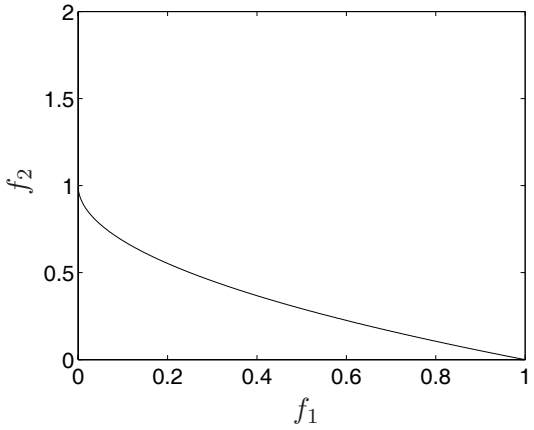


Figure A.4: Pareto-optimal front of ZDT4

ZDT6 is a 10-dimensional problem with variables $x_i \in [0, 1]$. It has a non-convex Pareto-optimal front (see Figure A.5) that creates difficulty by a non-uniform distribution of Pareto-optimal solutions. Furthermore, the density of solutions decreases towards the Pareto-optimal front.

$$\begin{aligned} f_1(x_1) &= 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ g(x_2, \ldots, x_D) &= 1 + 9 \cdot \left(\left(\sum_{i=2}^{D} x_i \right) \frac{1}{D-1} \right)^{0.25} \\ h(f_1, g) &= 1 - (f_1/g)^2 \end{aligned} \tag{A.5}$$

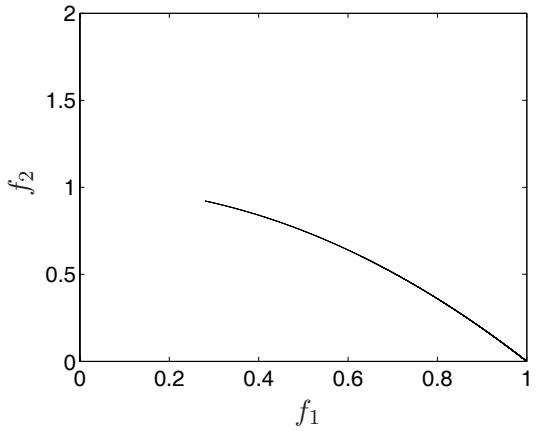
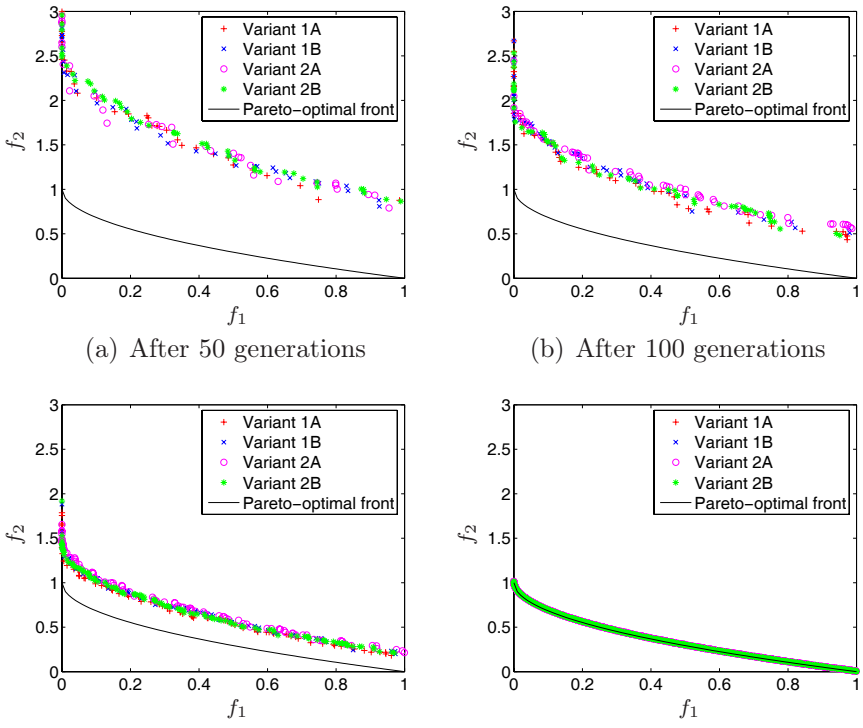


Figure A.5: Pareto-optimal front of ZDT6

Appendix B

Results for Different Variants of Multi-Objective Differential Evolution

In this appendix the results of the study described in Section 5.3.3 are shown.



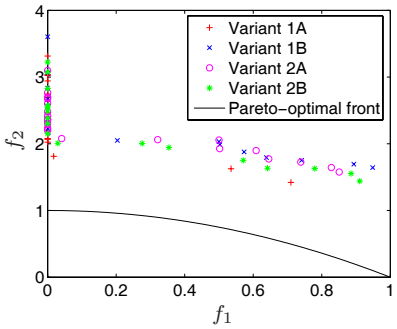
(a) After 50 generations

(b) After 100 generations

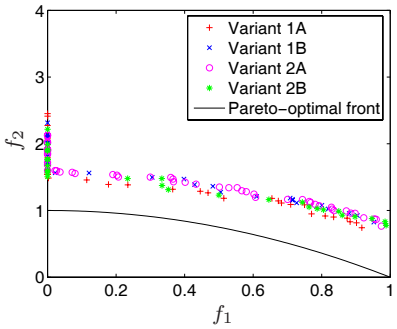
(c) After 200 generations

(d) After 1000 generations

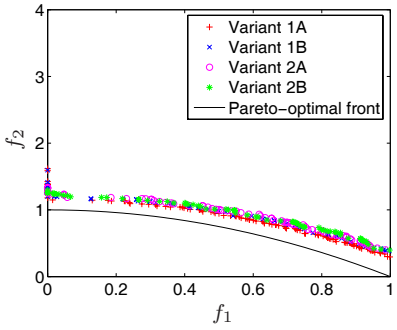
Figure B.1: Non-dominated solutions for ZDT1



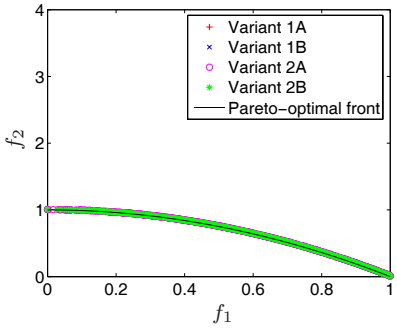
(a) After 50 generations



(b) After 100 generations

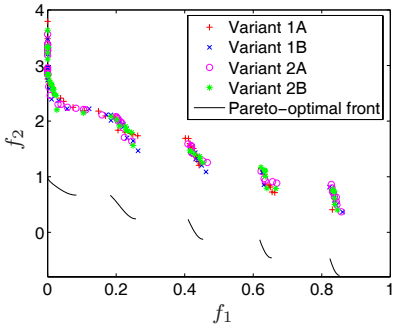


(c) After 200 generations

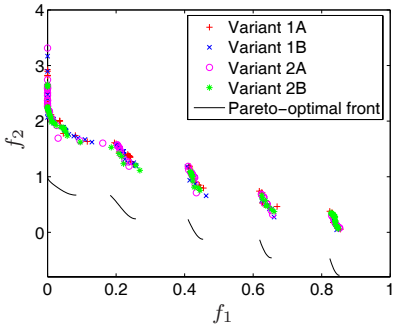


(d) After 1000 generations

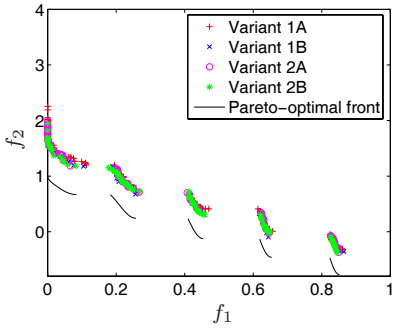
Figure B.2: Non-dominated solutions for ZDT2



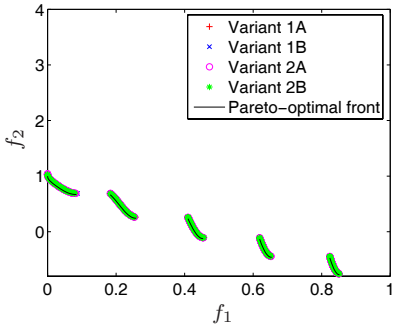
(a) After 50 generations



(b) After 100 generations

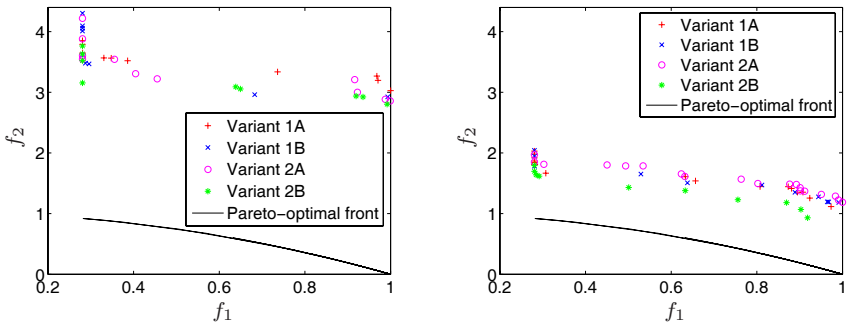


(c) After 200 generations



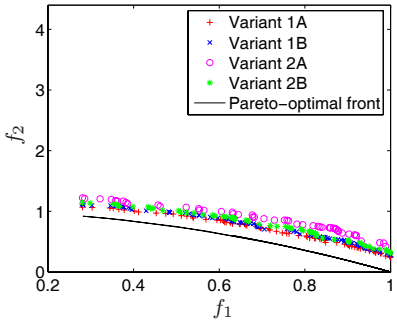
(d) After 1000 generations

Figure B.3: Non-dominated solutions for ZDT3

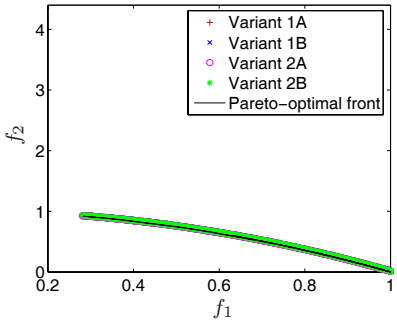


(a) After 50 generations

(b) After 100 generations

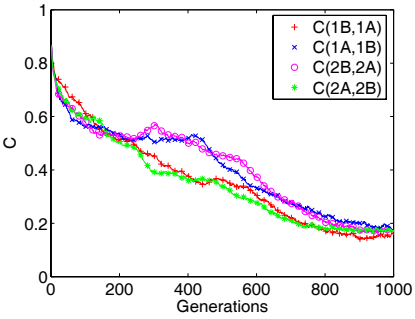


(c) After 200 generations

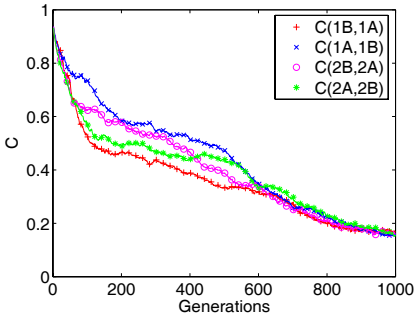


(d) After 1000 generations

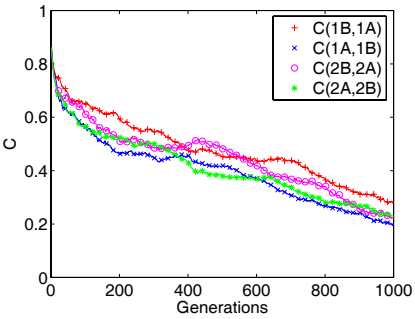
Figure B.4: Non-dominated solutions for ZDT6



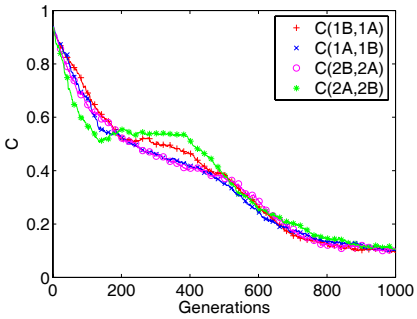
(a) Results for ZDT1



(b) Results for ZDT2

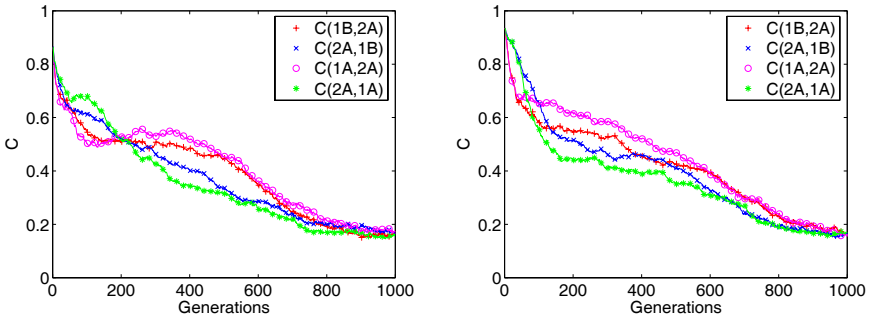


(c) Results for ZDT3



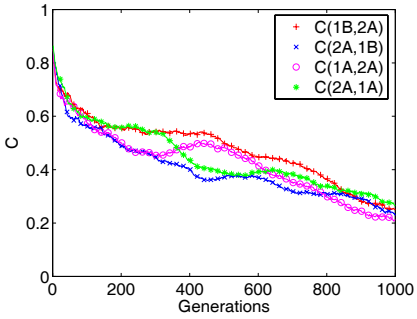
(d) Results for ZDT6

Figure B.5: Comparison of variants 1A and 1B as well as 2A and 2B based on the average set coverage metric

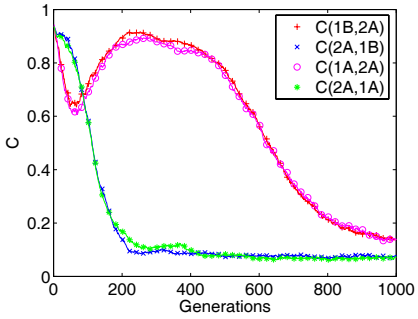


(a) Results for ZDT1

(b) Results for ZDT2

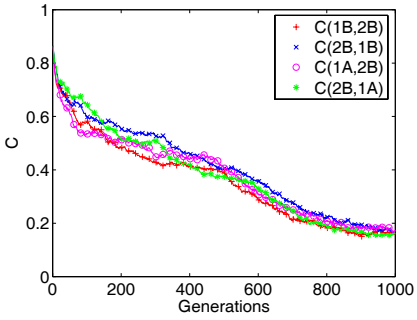


(c) Results for ZDT3

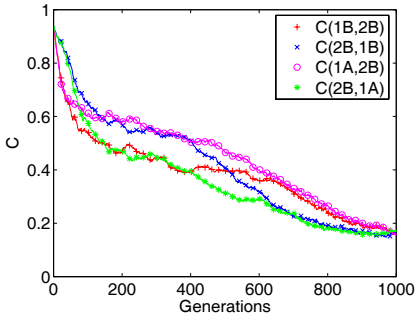


(d) Results for ZDT6

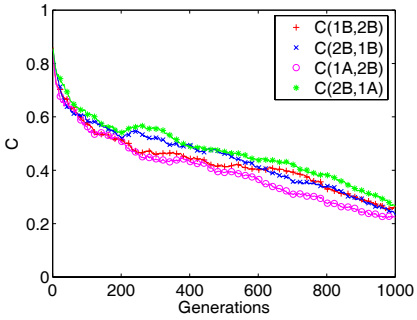
Figure B.6: Comparison of variants 1B and 2A as well as 1A and 2A based on the average set coverage metric



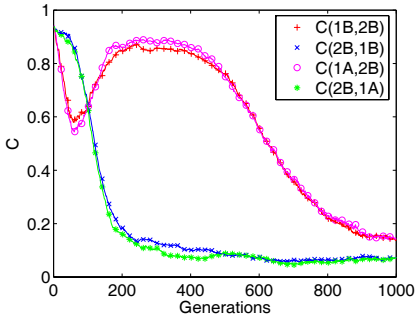
(a) Results for ZDT1



(b) Results for ZDT2

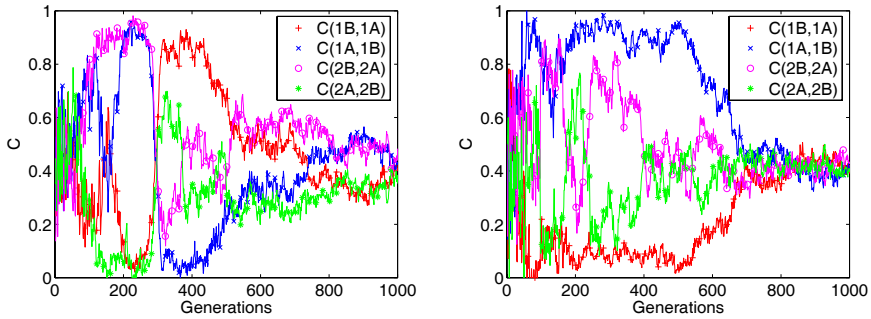


(c) Results for ZDT3



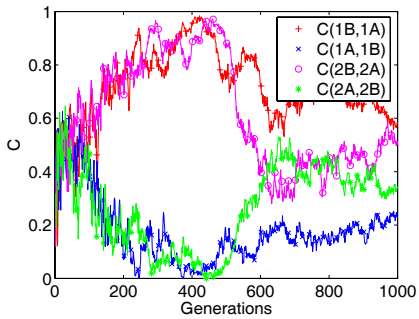
(d) Results for ZDT6

Figure B.7: Comparison of variants 1B and 2B as well as 1A and 2B based on the average set coverage metric

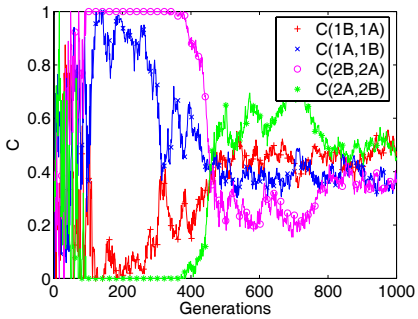


(a) Results for ZDT1

(b) Results for ZDT2

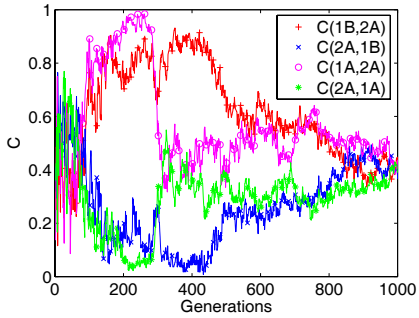


(c) Results for ZDT3

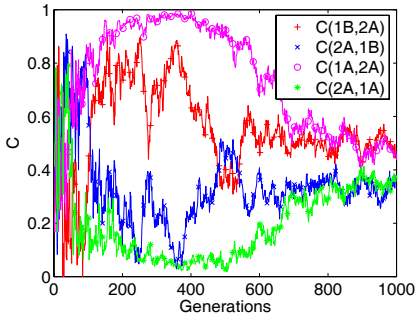


(d) Results for ZDT6

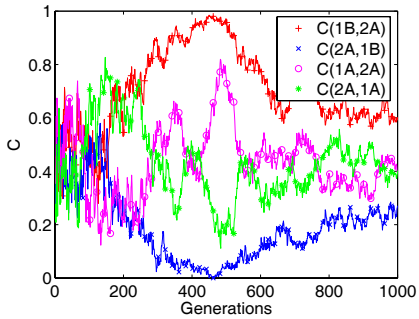
Figure B.8: Comparison of variants 1A and 1B as well as 2A and 2B based on the set coverage metric of combined solutions



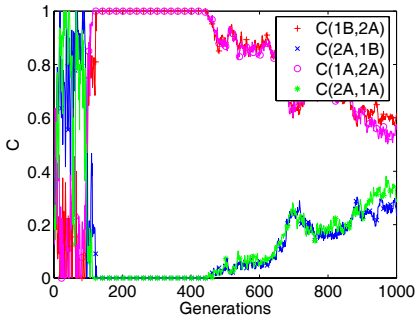
(a) Results for ZDT1



(b) Results for ZDT2

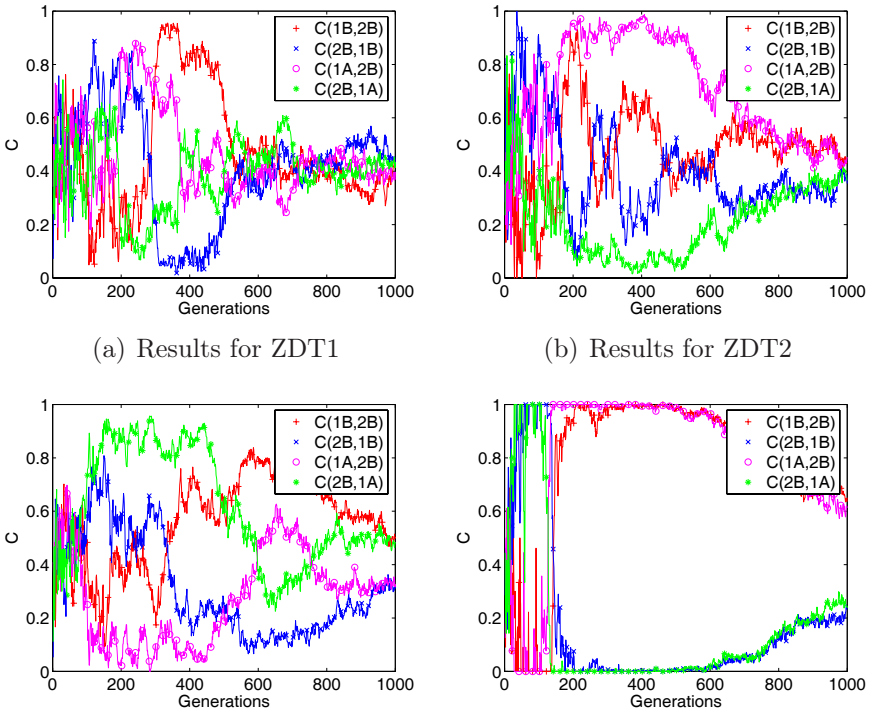


(c) Results for ZDT3



(d) Results for ZDT6

Figure B.9: Comparison of variants 1B and 2A as well as 1A and 2A based on the set coverage metric of combined solutions



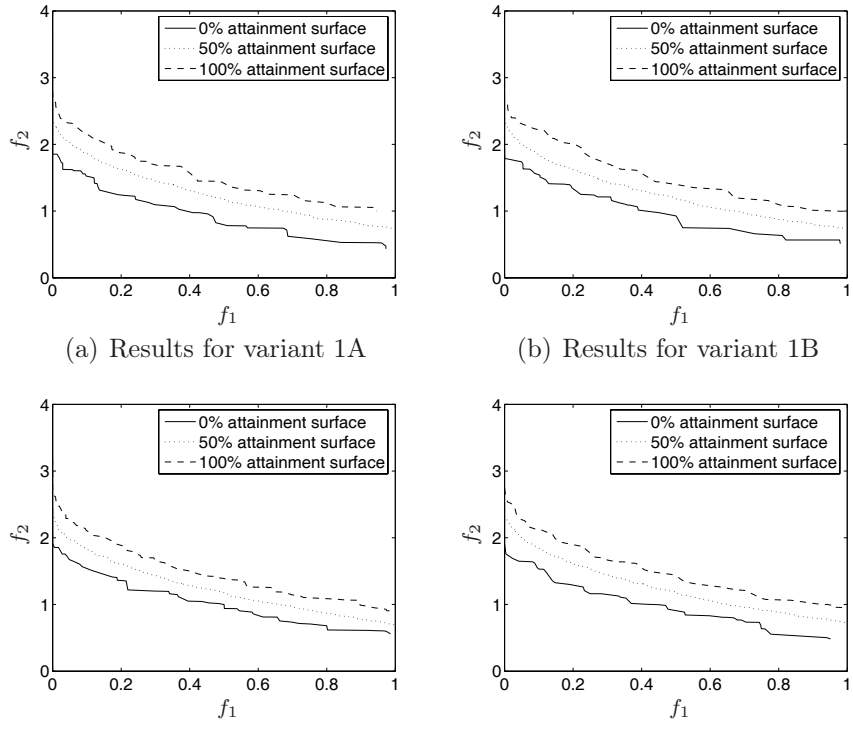
(a) Results for ZDT1

(b) Results for ZDT2

(c) Results for ZDT3

(d) Results for ZDT6

Figure B.10: Comparison of variants 1B and 2B as well as 1A and 2B based on the set coverage metric of combined solutions



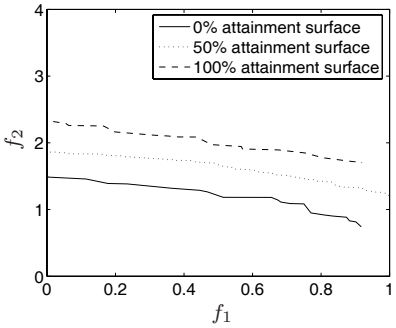
(a) Results for variant 1A

(b) Results for variant 1B

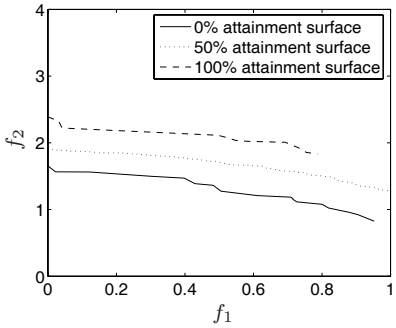
(c) Results for variant 2A

(d) Results for variant 2B

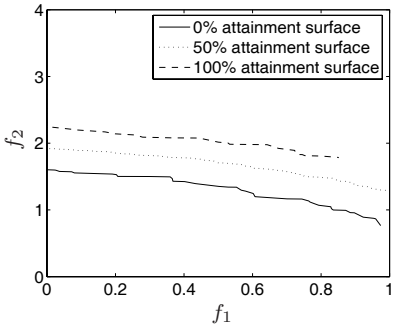
Figure B.11: 0%, 50% and 100% attainment surfaces for ZDT1 after 100 generations



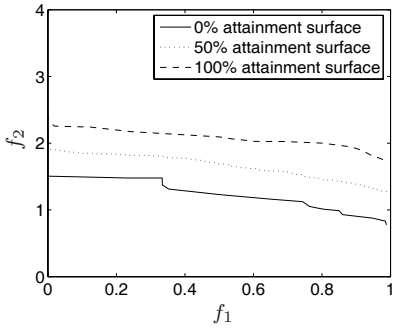
(a) Results for variant 1A



(b) Results for variant 1B

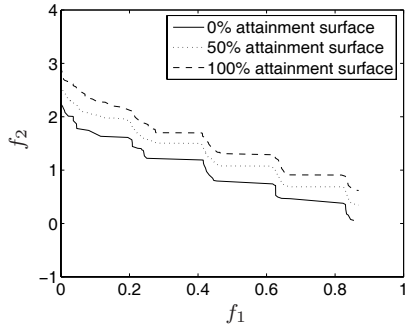


(c) Results for variant 2A

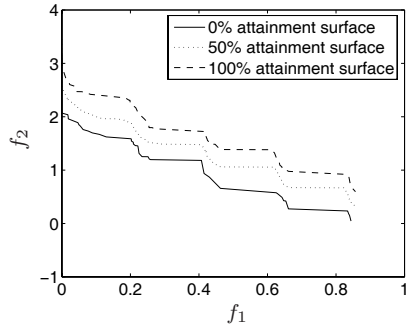


(d) Results for variant 2B

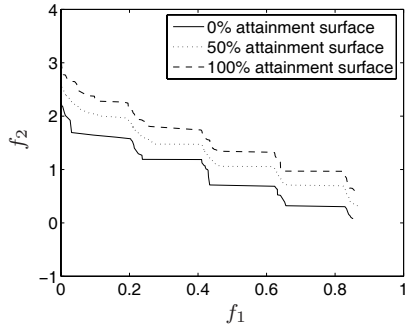
Figure B.12: 0%, 50% and 100% attainment surfaces for ZDT2 after 100 generations



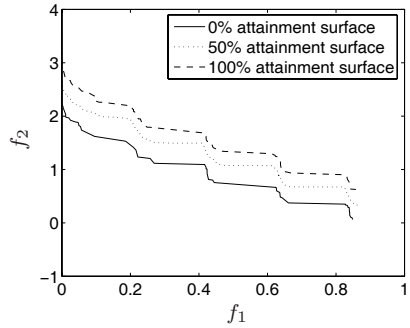
(a) Results for variant 1A



(b) Results for variant 1B

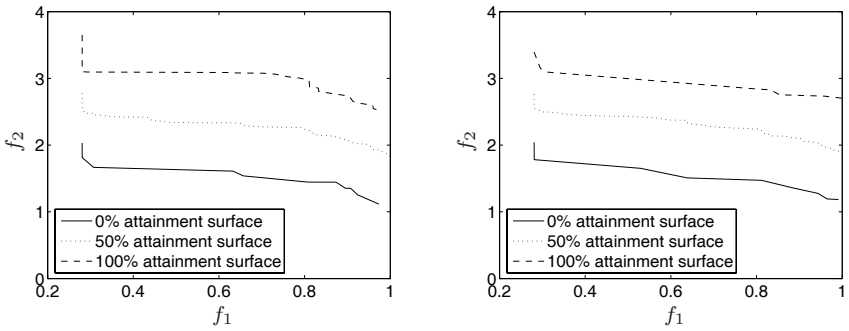


(c) Results for variant 2A



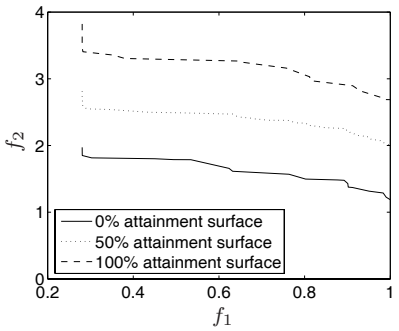
(d) Results for variant 2B

Figure B.13: 0%, 50% and 100% attainment surfaces for ZDT3 after 100 generations

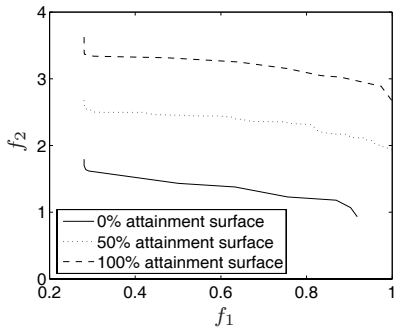


(a) Results for variant 1A

(b) Results for variant 1B



(c) Results for variant 2A



(d) Results for variant 2B

Figure B.14: 0%, 50% and 100% attainment surfaces for ZDT6 after 100 generations

Appendix C

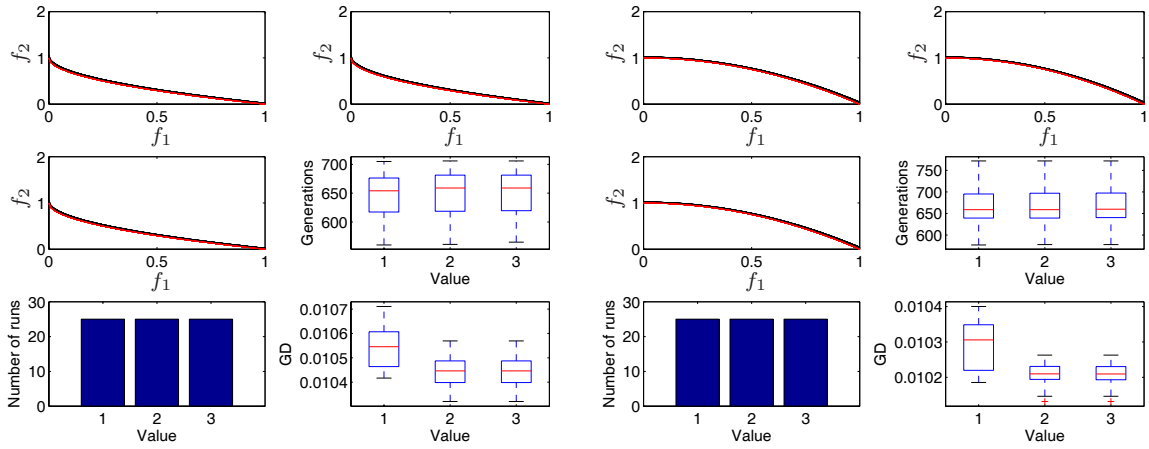
Results of Stopping Criteria for Multi-Objective Optimization

In this appendix the results of the stopping criteria defined in Section 6.2 for multi-objective optimization are shown. The stopping criteria and the corresponding parameters are given in Table C.1. For each stopping criterion (and for each setting of the threshold) a figure has been generated, consisting of five subplots showing the results for the different functions that are given in Appendix A. Each of the subplots is itself divided into six separate figures (an exception is *IdParetoRank* because only one setting of its parameter exists): The first three show the non-dominated fronts from all runs for the three values of the stopping parameter given in Table C.1, together with the Pareto-optimal front (in red). The fourth figure shows the number of generations at which the optimization runs have been terminated, the fifth shows the number of runs which have been terminated before reaching the maximum number of generations, and the sixth gives the generational distance of the generated non-dominated fronts to the Pareto-optimal front.

Table C.1: Parameters for the stopping criteria

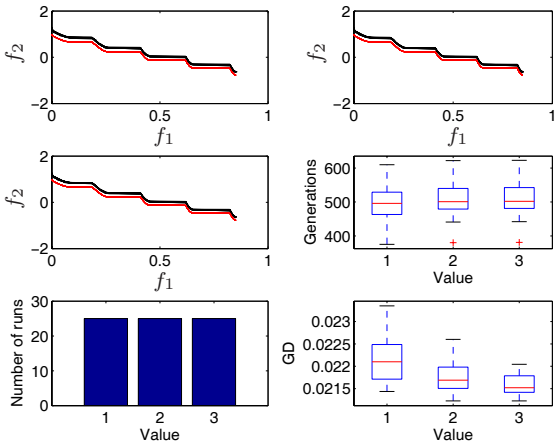
Stopping	Threshold			Parameter			
criterion	Name	Value 1	Value 2	Name	Value 1	Value 2	Value 3
RefCritER	ϵ_{ER}	0.015	0.01	ER_{stop}	0.2	0.1	0.05
RefCritC	-	-	-	C_{stop}	0.9	0.7	0.3
RefCritGD	-	-	-	GD_{stop}	0.02	0.015	0.01
RefCritHV	-	-	-	p_{HV}	0.9	0.95	0.98
ImpC	t_C	0.01	-	ΔG	6	8	10
ImpGD	t_{GD}	0.01	-	ΔG	4	6	8
ImpHV	t_{HV}	0	0.0001	ΔG	3	5	7
NoAcc_MO	-	-	-	ΔG	5	6	7
IdParetoRank	-	-	-	-	-	-	-
DomInd	-	-	-	N	5	3	2
DistSpacing	t_s	0	0.0001	ΔG	6	7	8
DistSpread	t_{sp}	0	0.0001	ΔG	7	8	9
MaxCD	-	-	-	δ_{lim}	0.002	0.0009	0.0002

C.1 Results for Differential Evolution

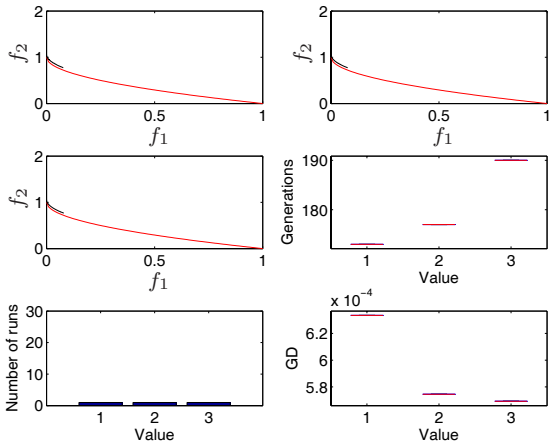


(a) ZDT1

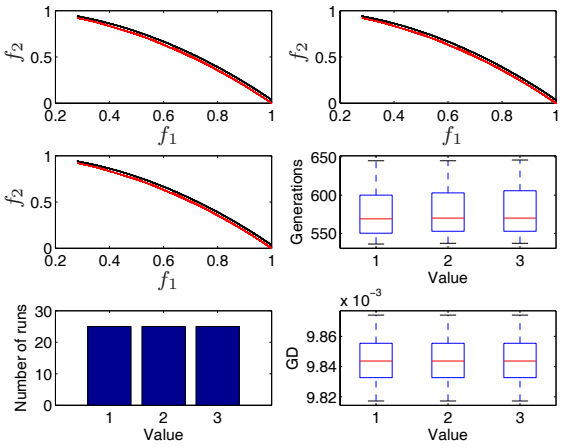
(b) ZDT2



(c) ZDT3

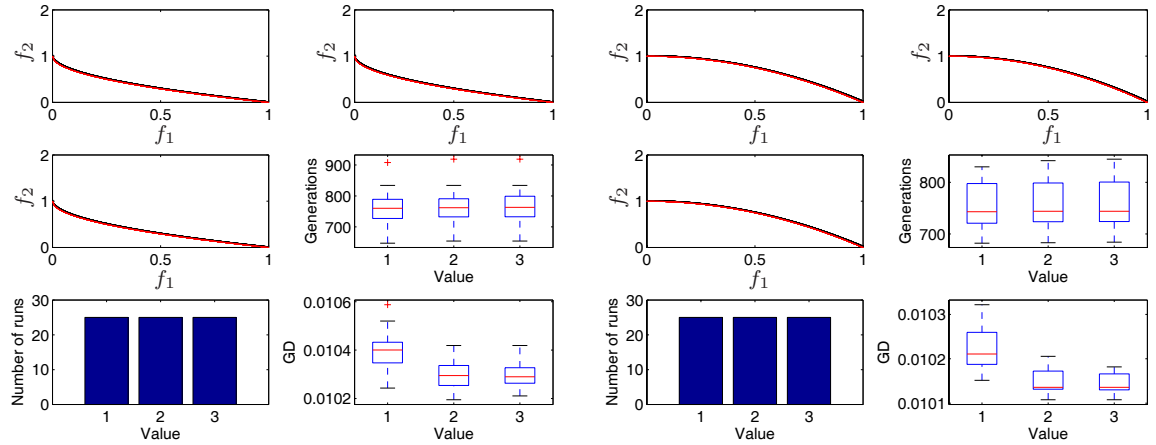


(d) ZDT4



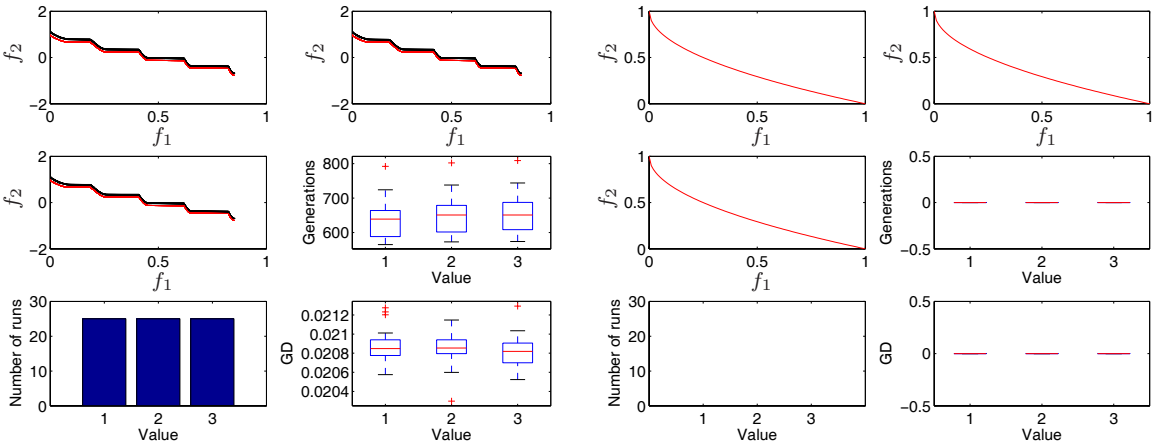
(e) ZDT6

Figure C.1: *RefCritER* using DE with $\epsilon_{ER} = 0.015$



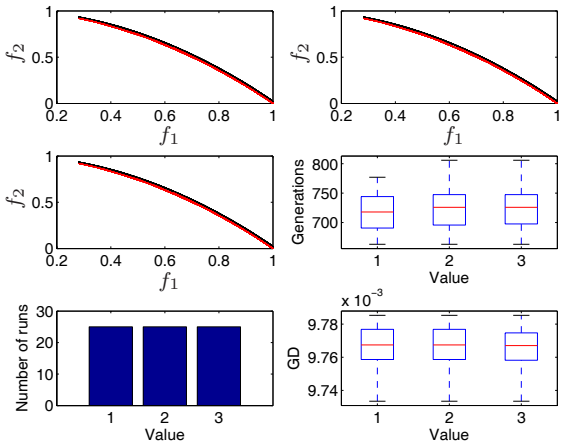
(a) ZDT1

(b) ZDT2



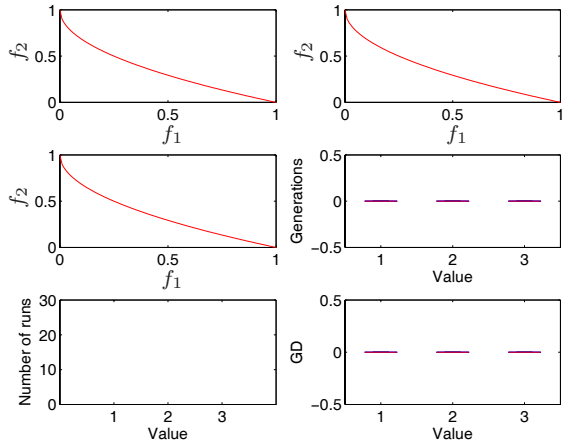
(c) ZDT3

(d) ZDT4

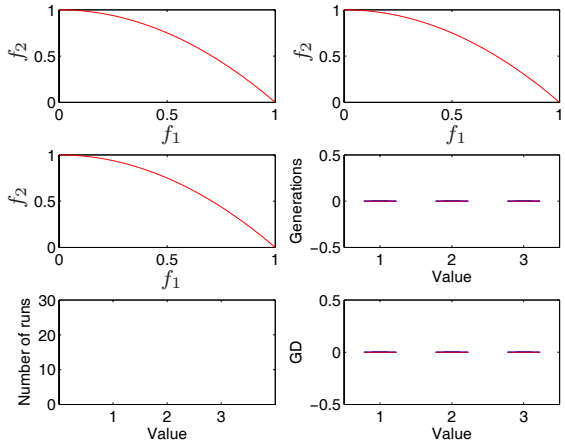


(e) ZDT6

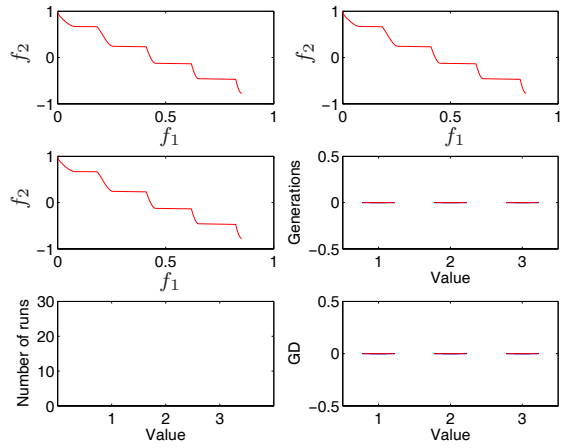
Figure C.2: *RefCritER* using DE with $\epsilon_{ER} = 0.01$



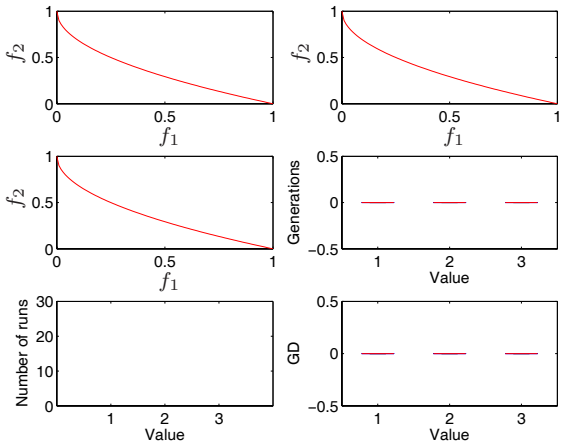
(a) ZDT1



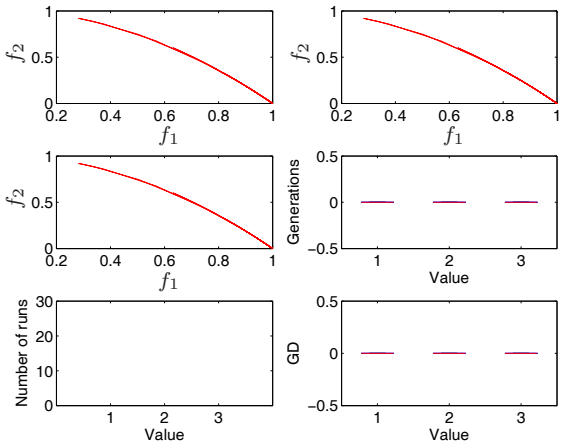
(b) ZDT2



(c) ZDT3

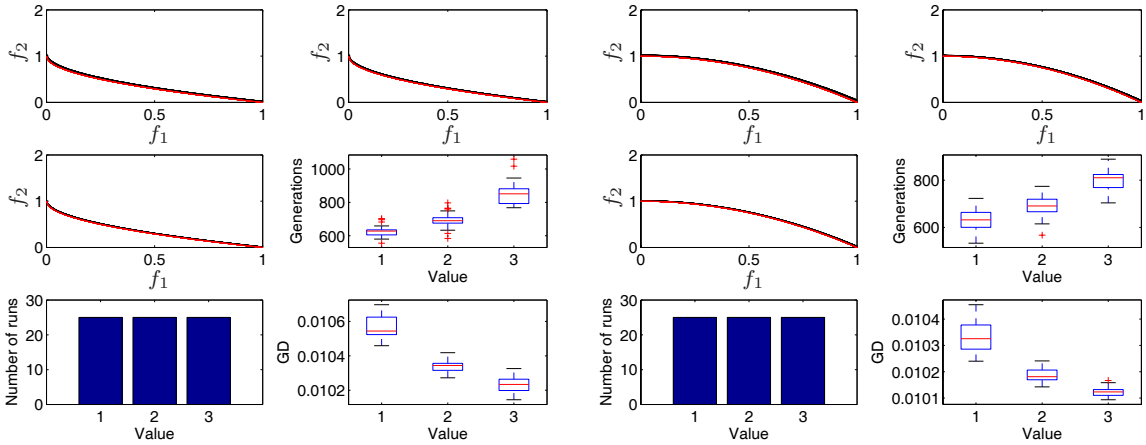


(d) ZDT4



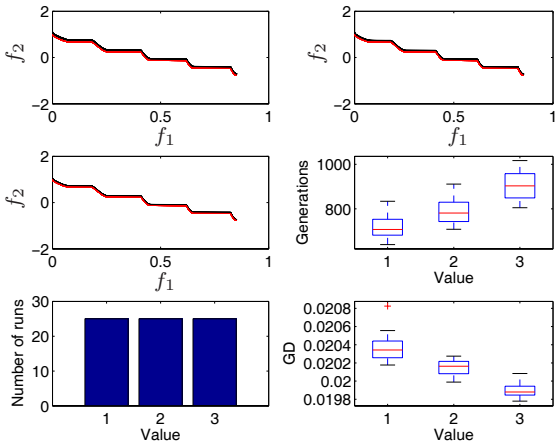
(e) ZDT6

Figure C.3: *RefCritC* using DE

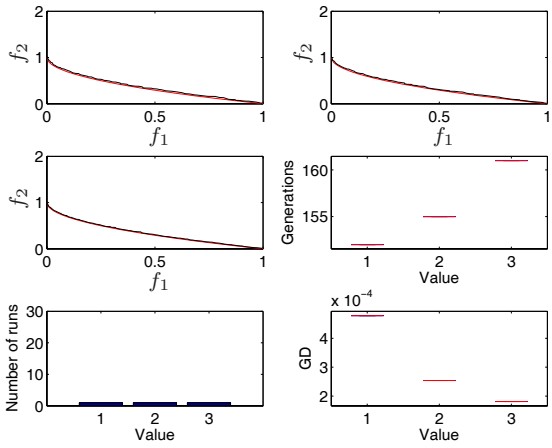


(a) ZDT1

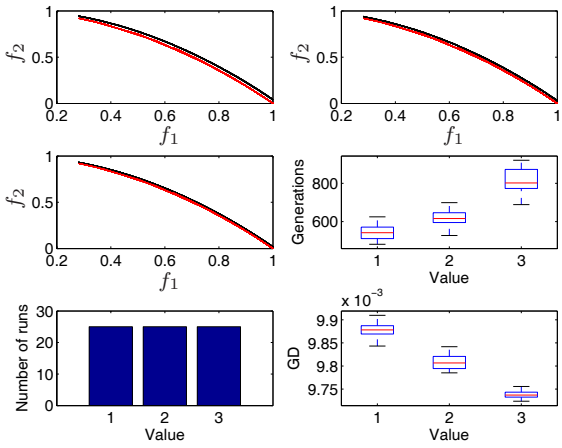
(b) ZDT2



(c) ZDT3

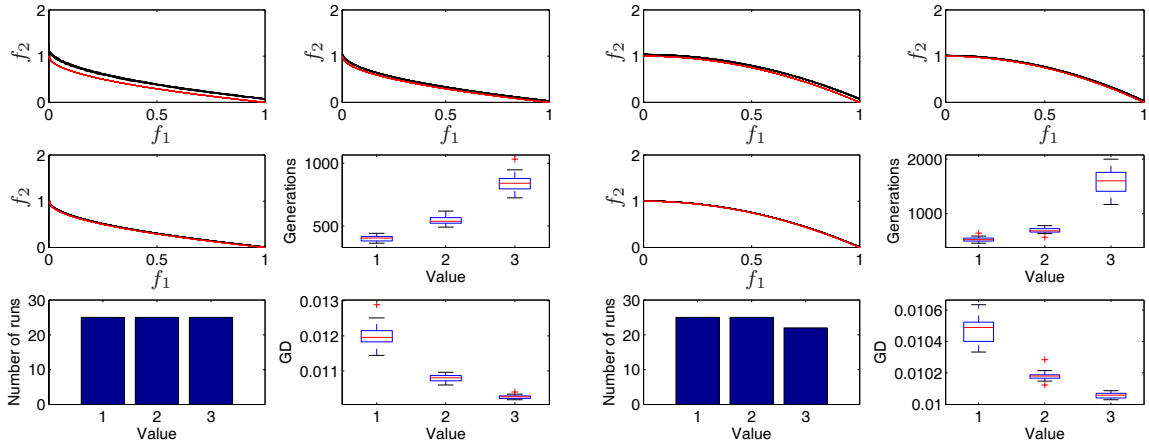


(d) ZDT4



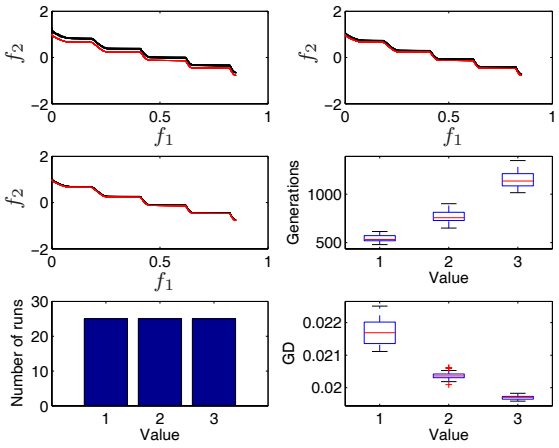
(e) ZDT6

Figure C.4: *RefCritGD* using DE

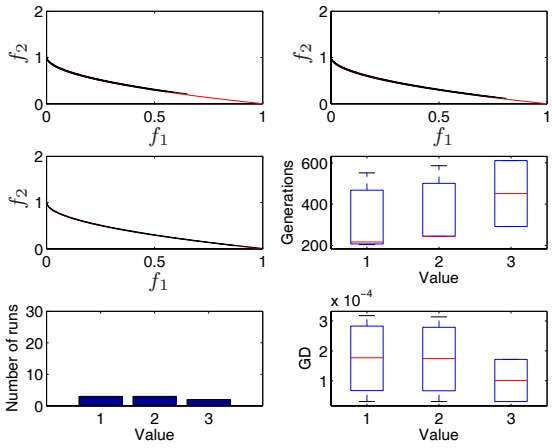


(a) ZDT1

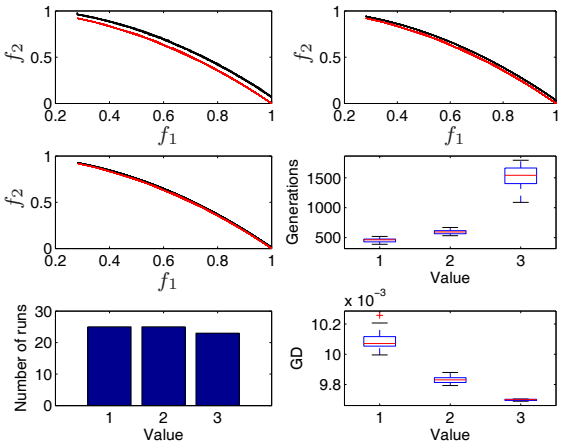
(b) ZDT2



(c) ZDT3

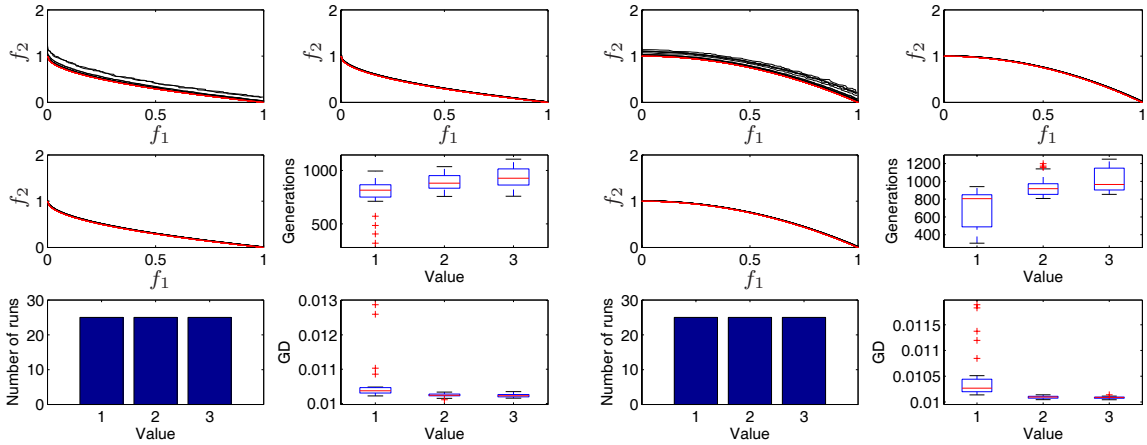


(d) ZDT4



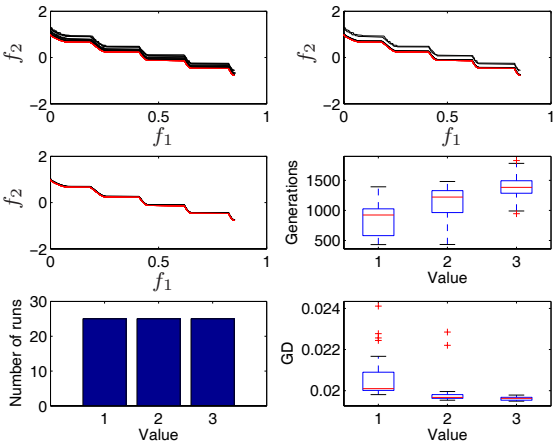
(e) ZDT6

Figure C.5: *RefCritHV* using DE

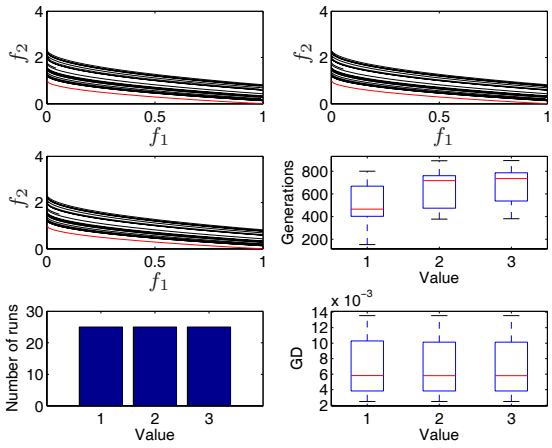


(a) ZDT1

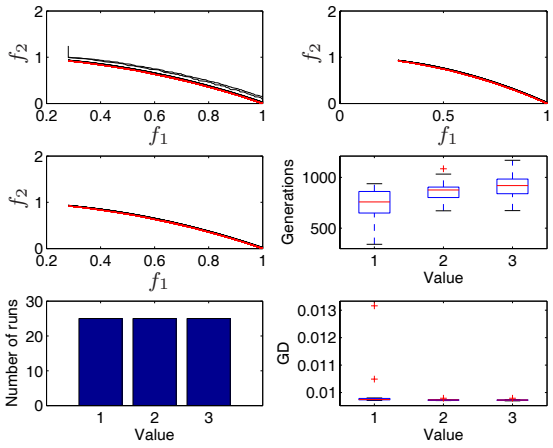
(b) ZDT2



(c) ZDT3

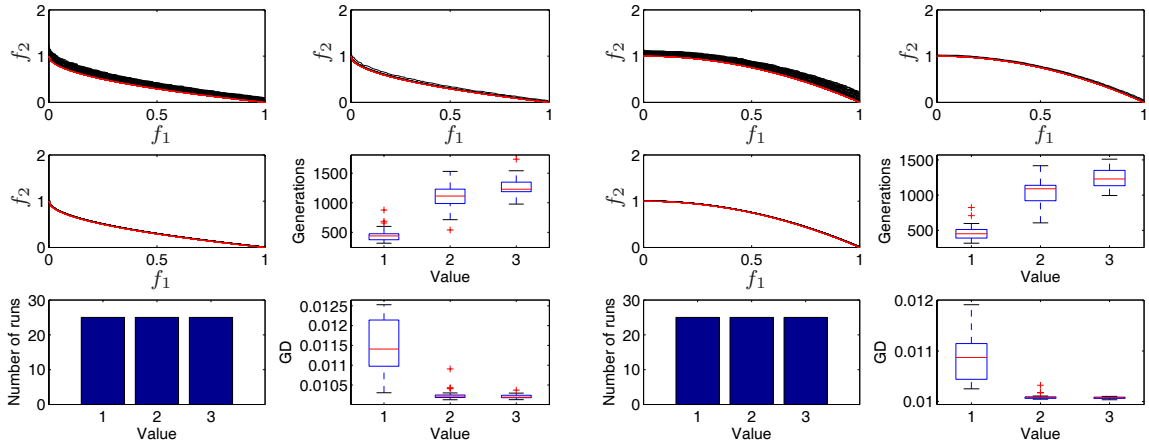


(d) ZDT4



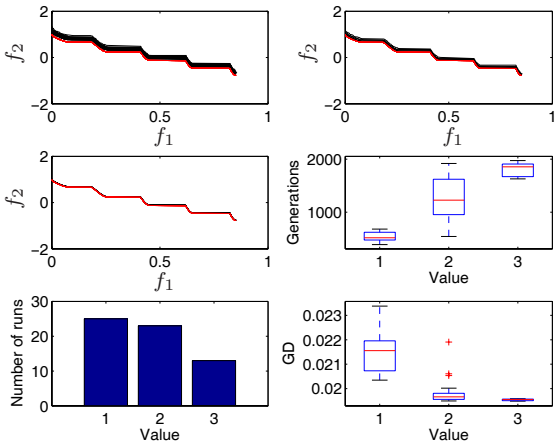
(e) ZDT6

Figure C.6: *ImpC* using DE

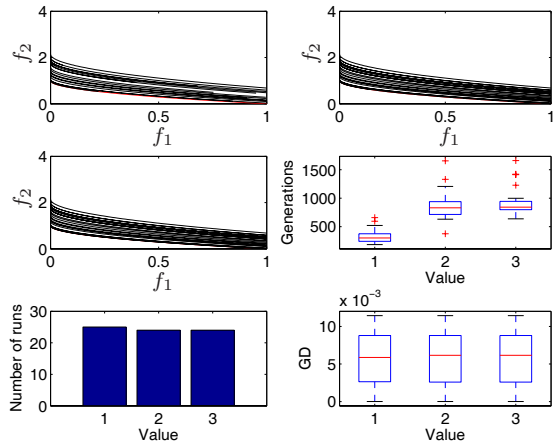


(a) ZDT1

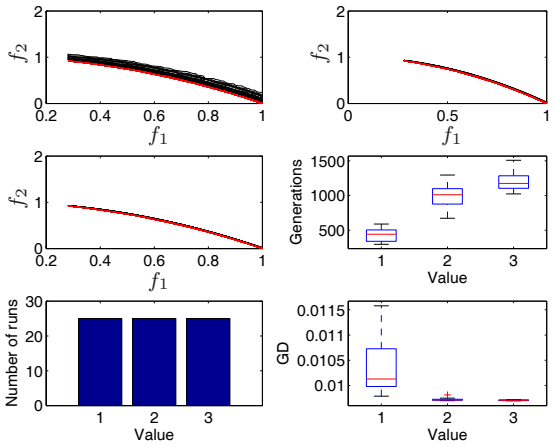
(b) ZDT2



(c) ZDT3

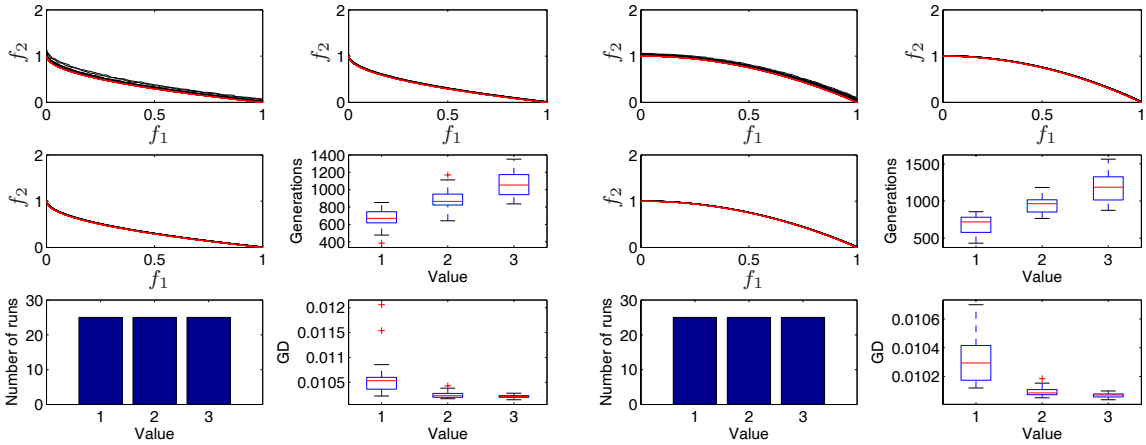


(d) ZDT4



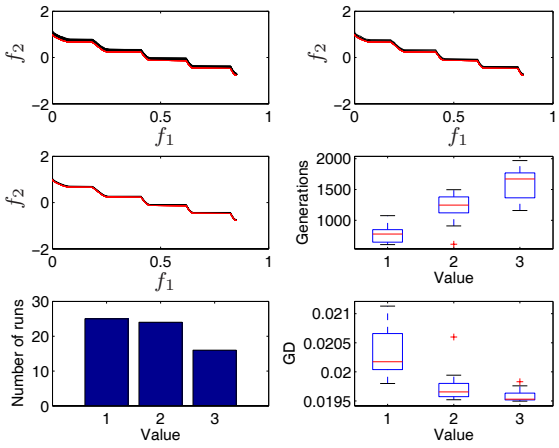
(e) ZDT6

Figure C.7: *ImpGD* using DE

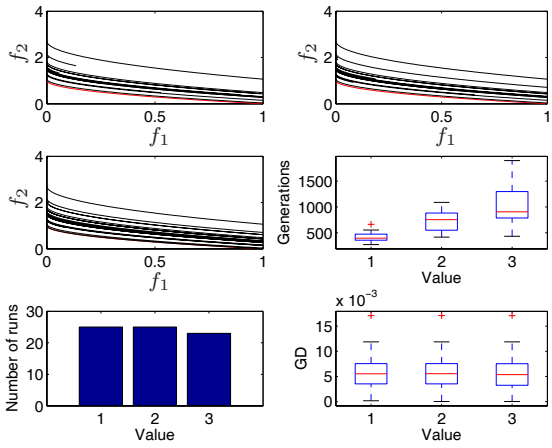


(a) ZDT1

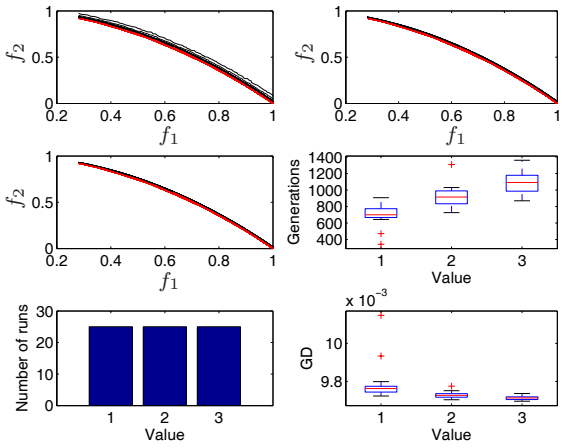
(b) ZDT2



(c) ZDT3

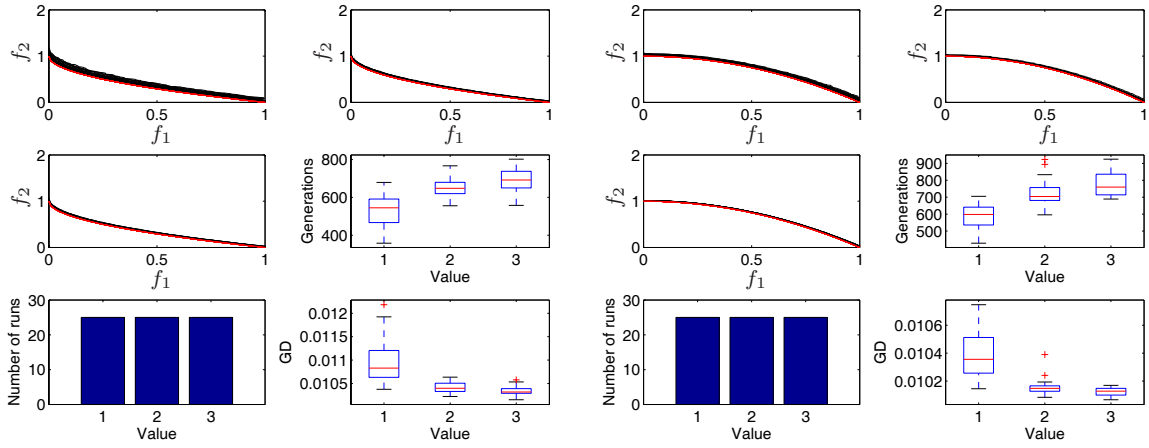


(d) ZDT4



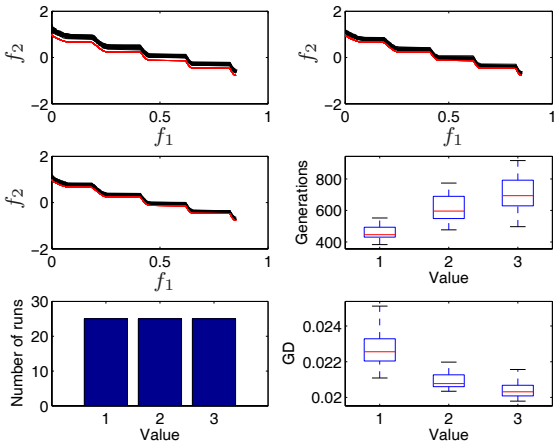
(e) ZDT6

Figure C.8: *ImpHV* using DE with $t_{HV} = 0$

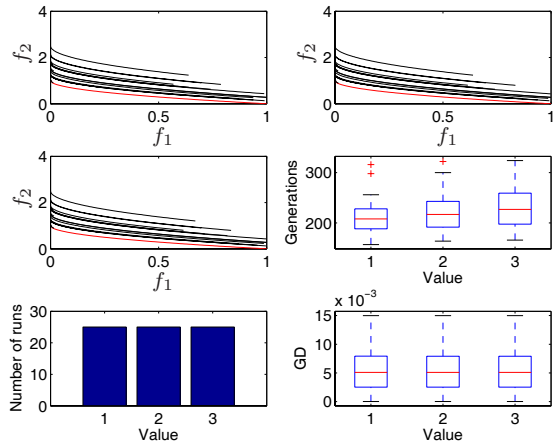


(a) ZDT1

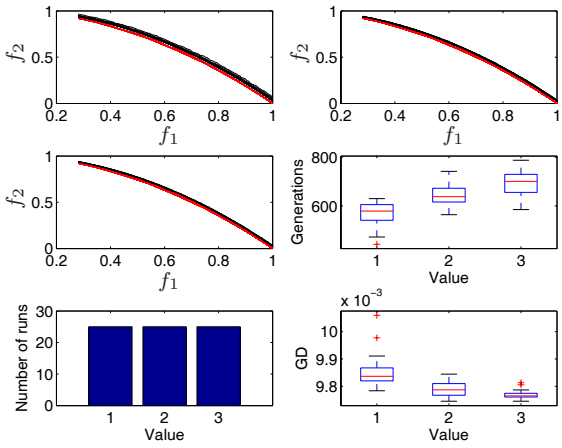
(b) ZDT2



(c) ZDT3

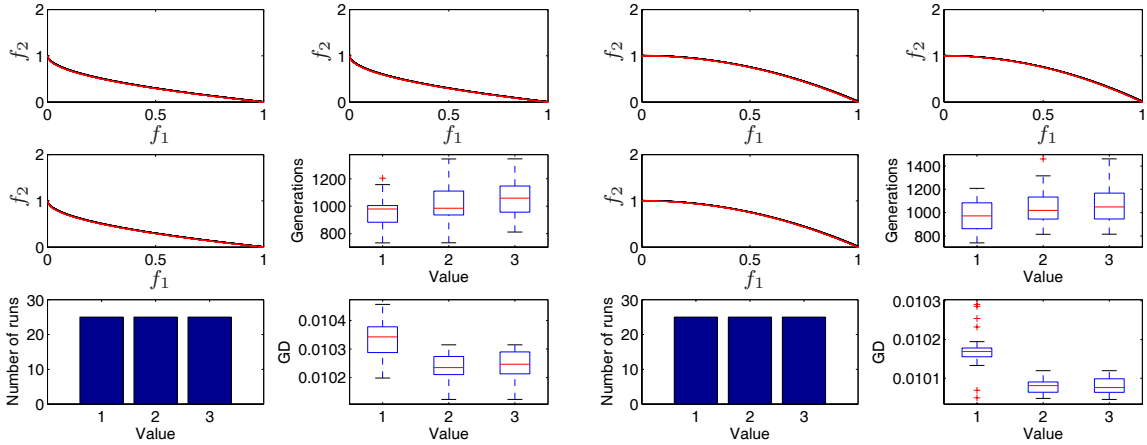


(d) ZDT4



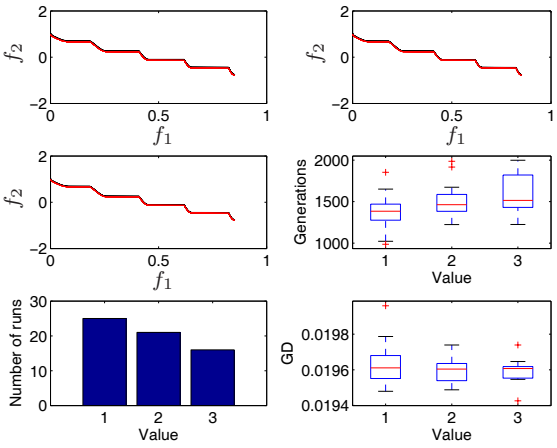
(e) ZDT6

Figure C.9: *ImpHV* using DE with $t_{HV} = 0.0001$

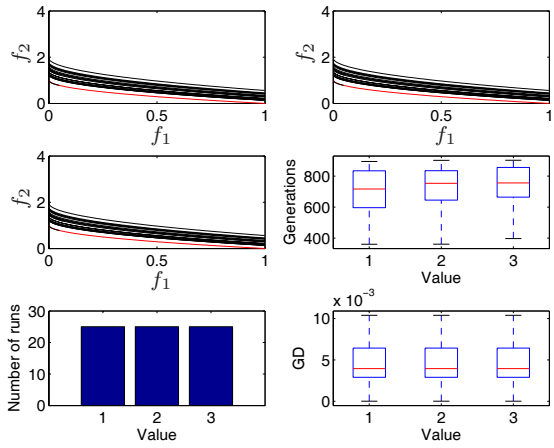


(a) ZDT1

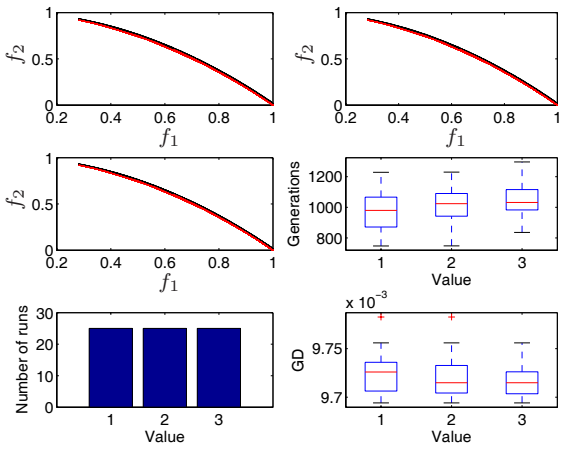
(b) ZDT2



(c) ZDT3

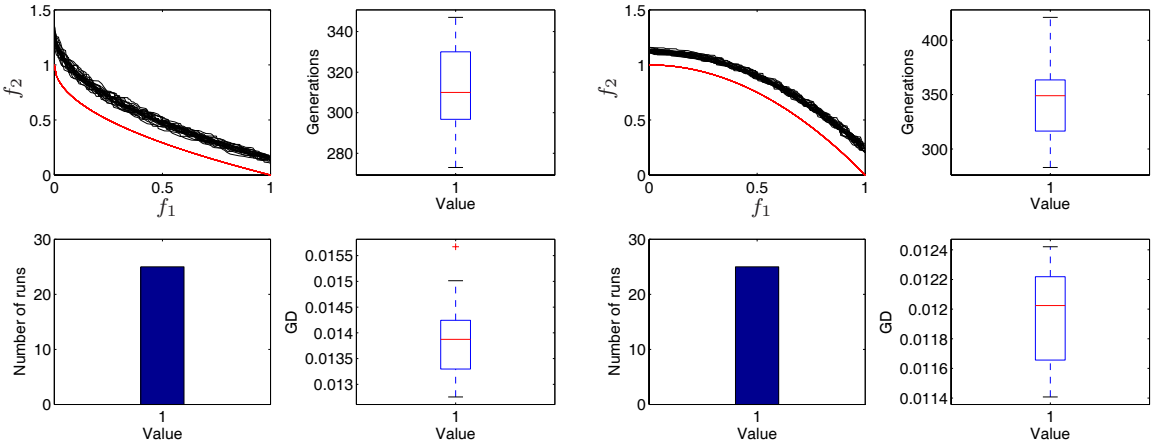


(d) ZDT4



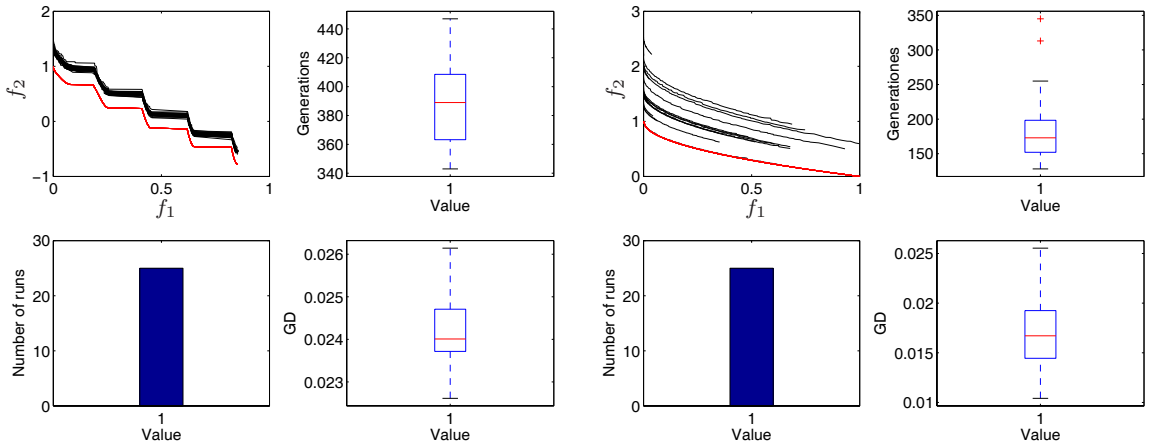
(e) ZDT6

Figure C.10: *NoAcc_MO* using DE



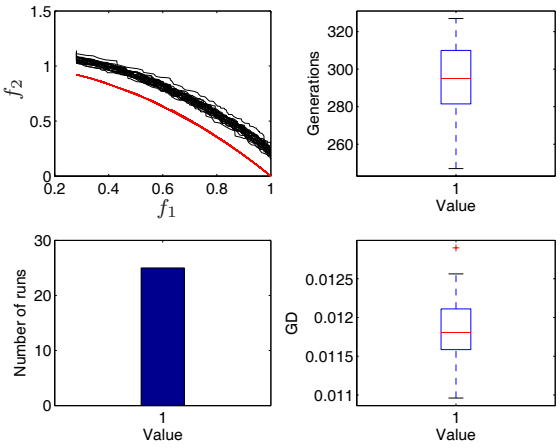
(a) ZDT1

(b) ZDT2



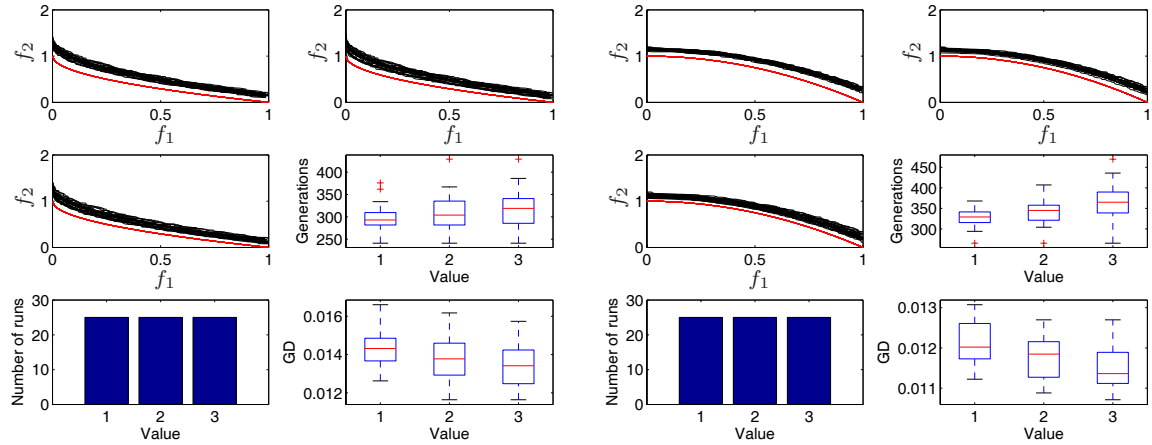
(c) ZDT3

(d) ZDT4



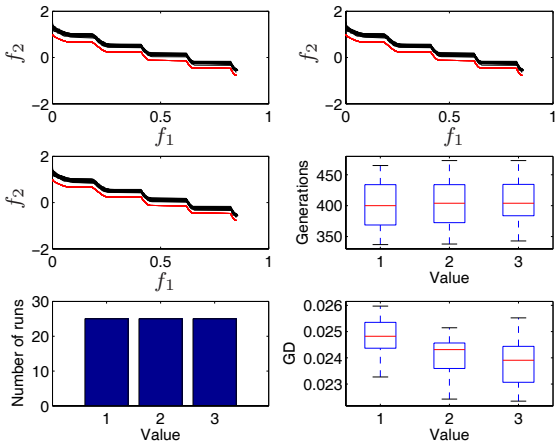
(e) ZDT6

Figure C.11: *IdParetoRank* using DE

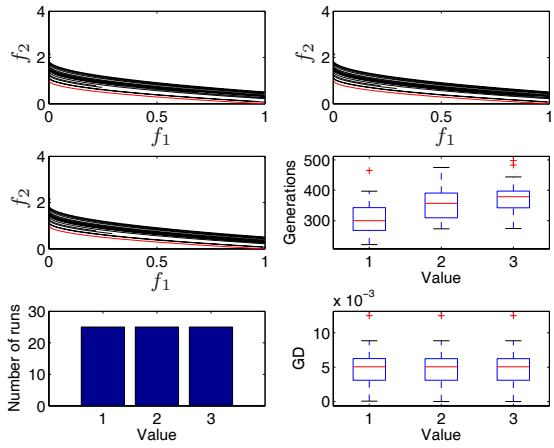


(a) ZDT1

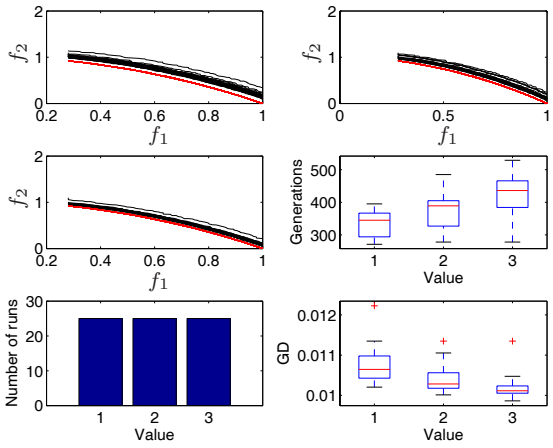
(b) ZDT2



(c) ZDT3

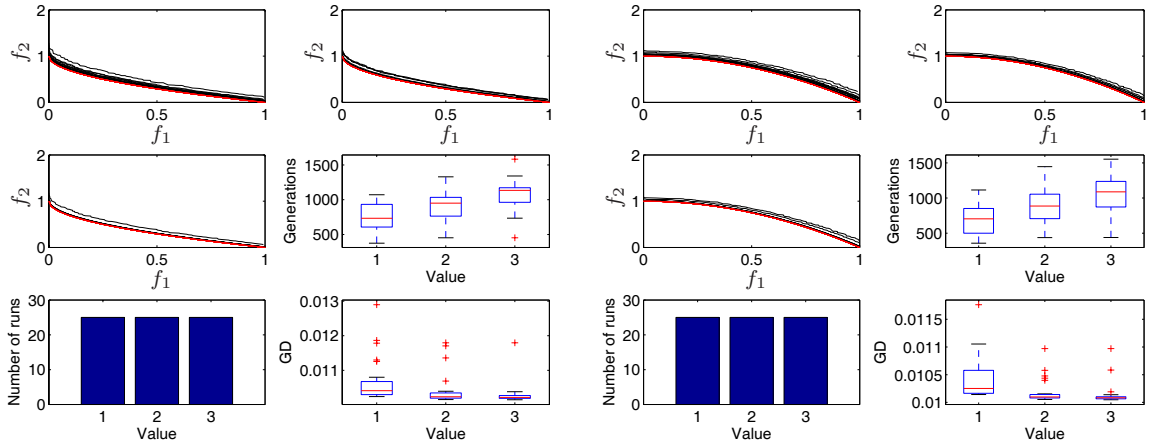


(d) ZDT4



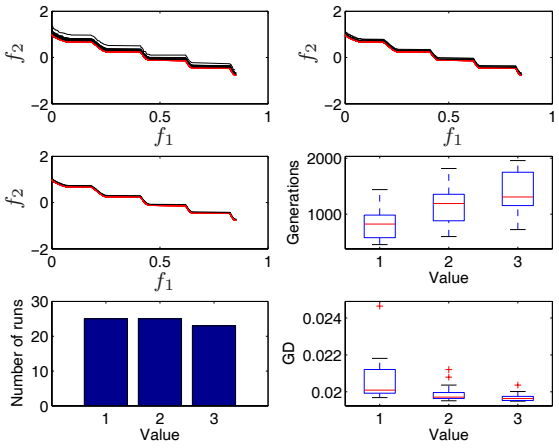
(e) ZDT6

Figure C.12: *DomInd* using DE

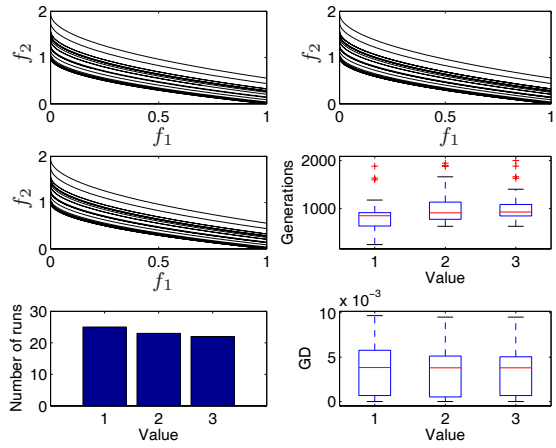


(a) ZDT1

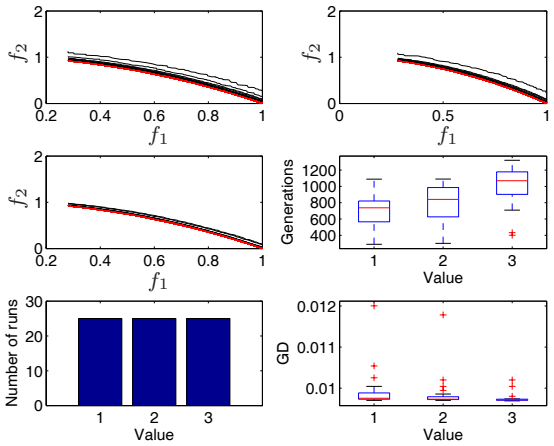
(b) ZDT2



(c) ZDT3

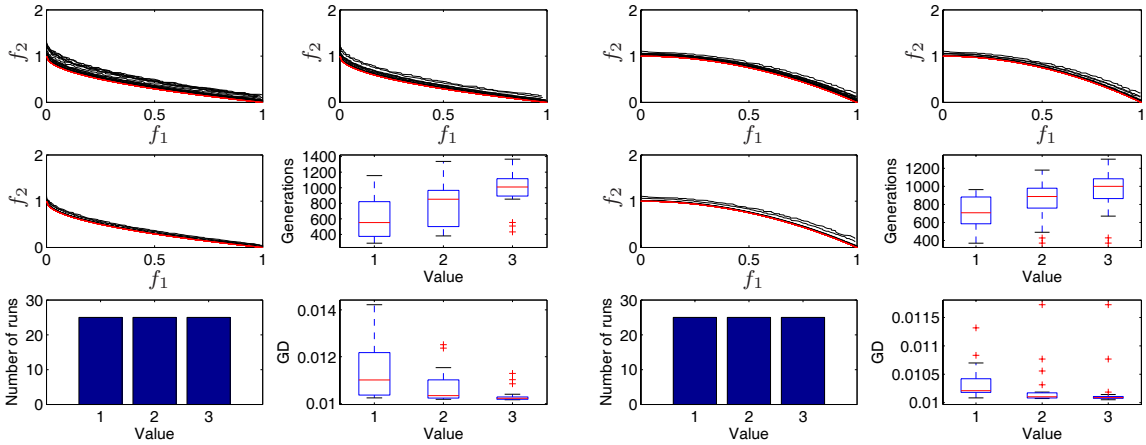


(d) ZDT4



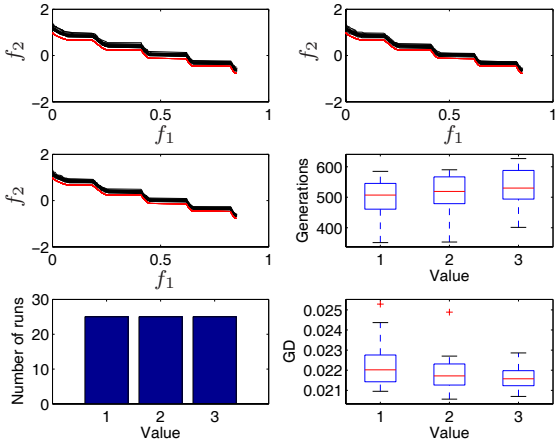
(e) ZDT6

Figure C.13: *DistSpacing* using DE with $t_s = 0$

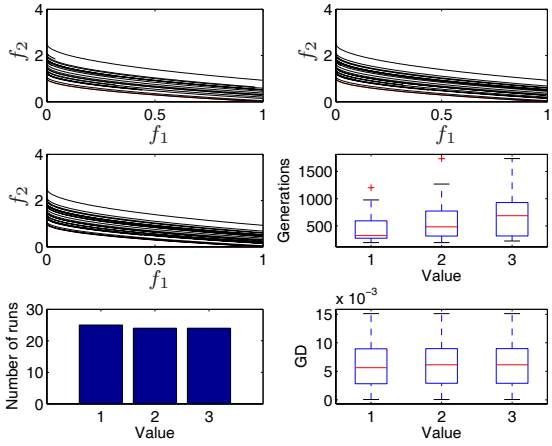


(a) ZDT1

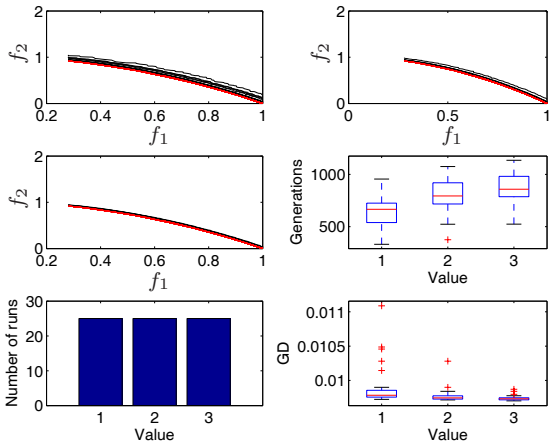
(b) ZDT2



(c) ZDT3

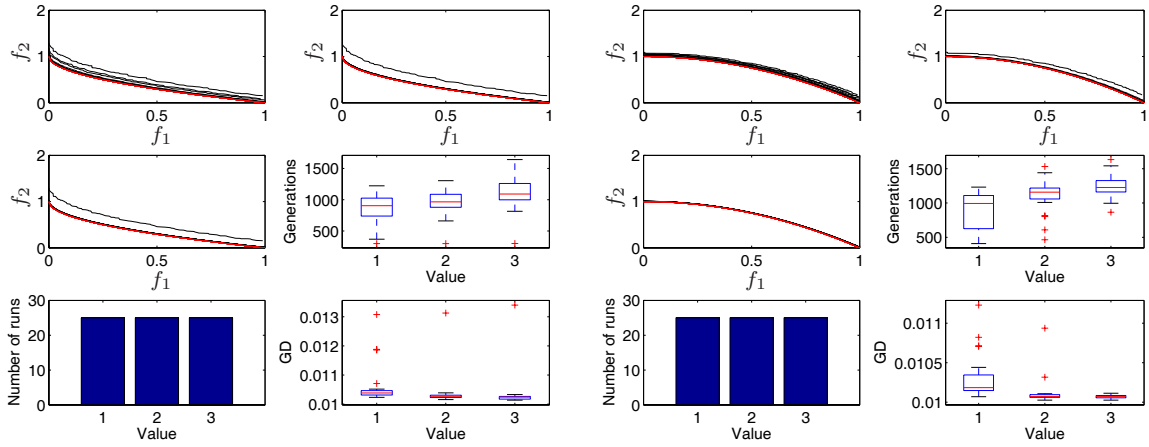


(d) ZDT4



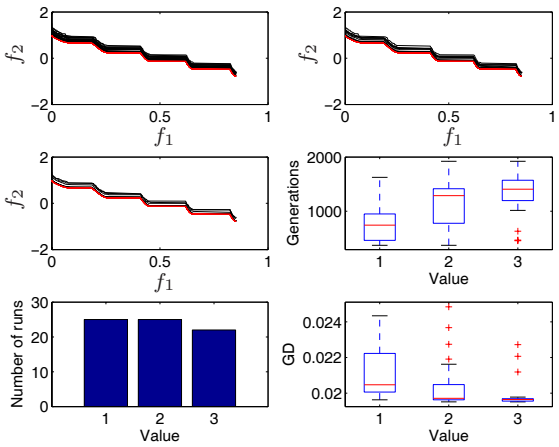
(e) ZDT6

Figure C.14: *DistSpacing* using DE with $t_s = 0.001$

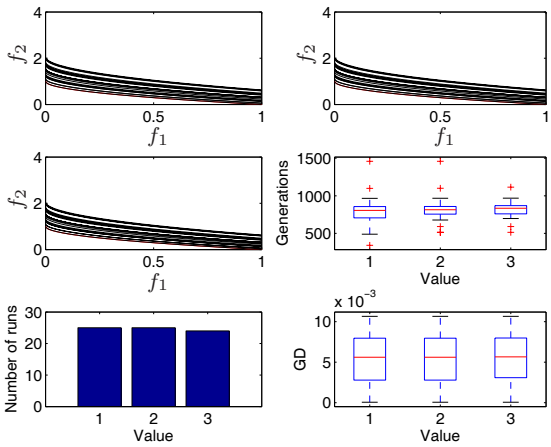


(a) ZDT1

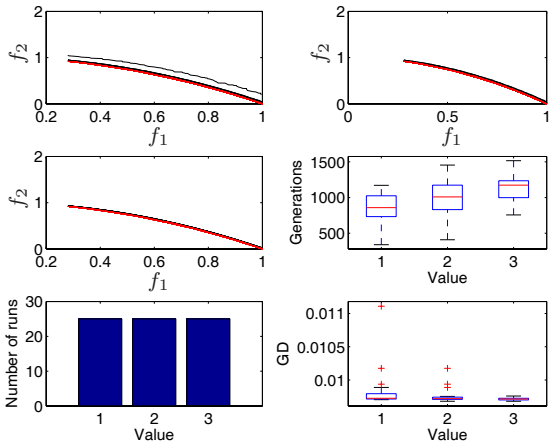
(b) ZDT2



(c) ZDT3

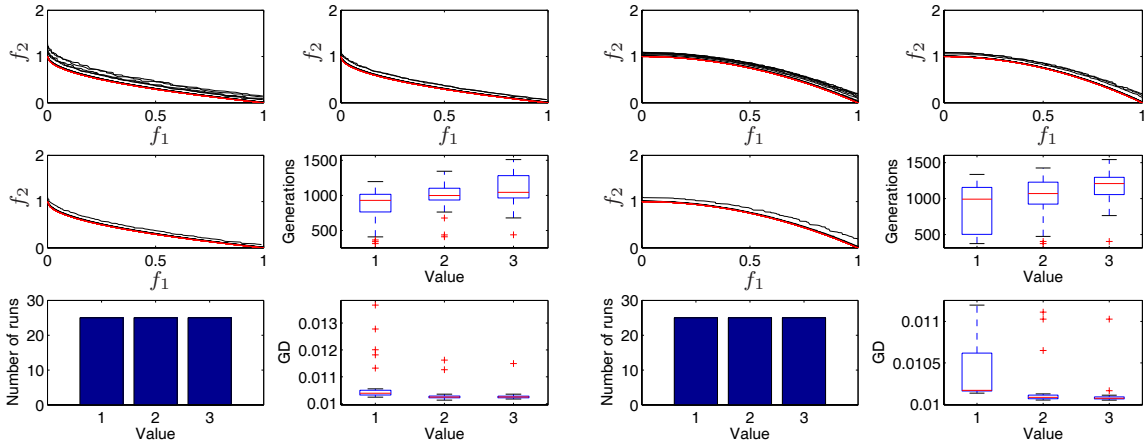


(d) ZDT4



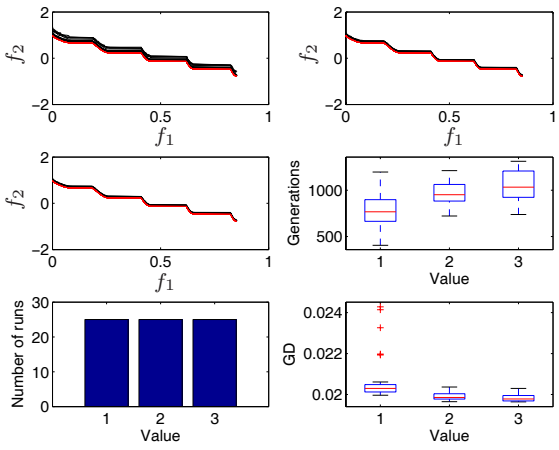
(e) ZDT6

Figure C.15: *DistSpread* using DE with $t_{sp} = 0$

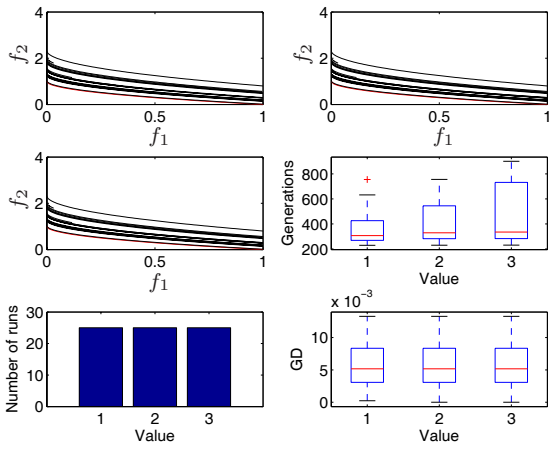


(a) ZDT1

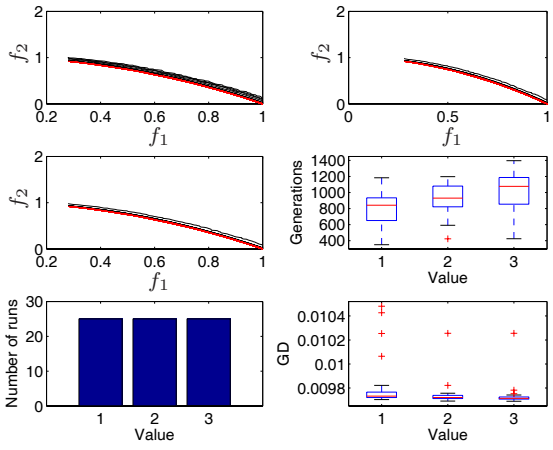
(b) ZDT2



(c) ZDT3

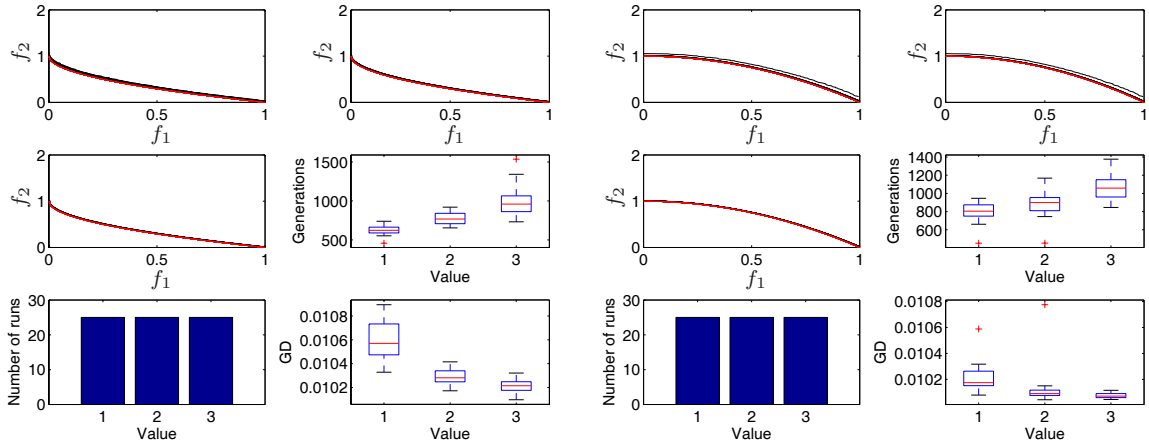


(d) ZDT4



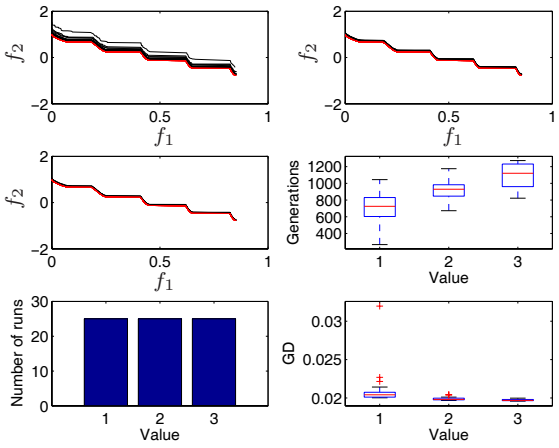
(e) ZDT6

Figure C.16: *DistSpread* using DE with $t_{sp} = 0.0001$

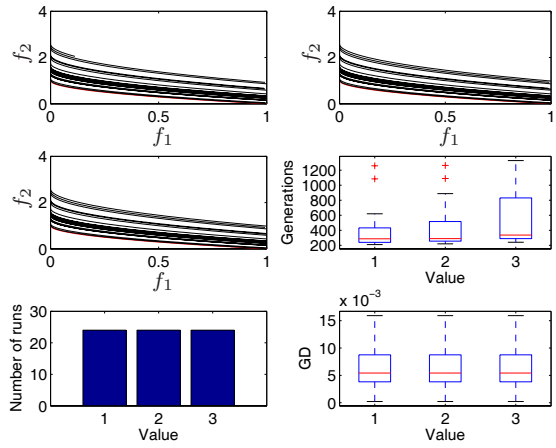


(a) ZDT1

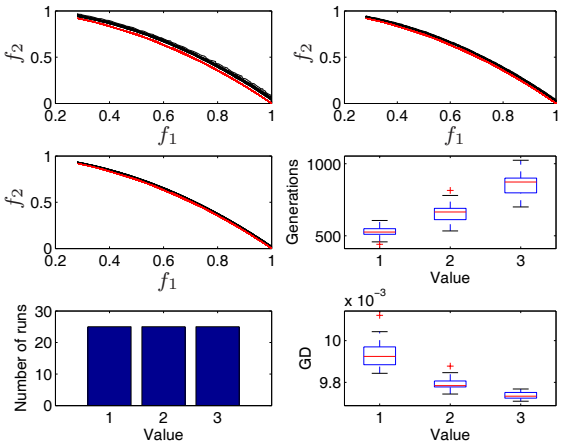
(b) ZDT2



(c) ZDT3



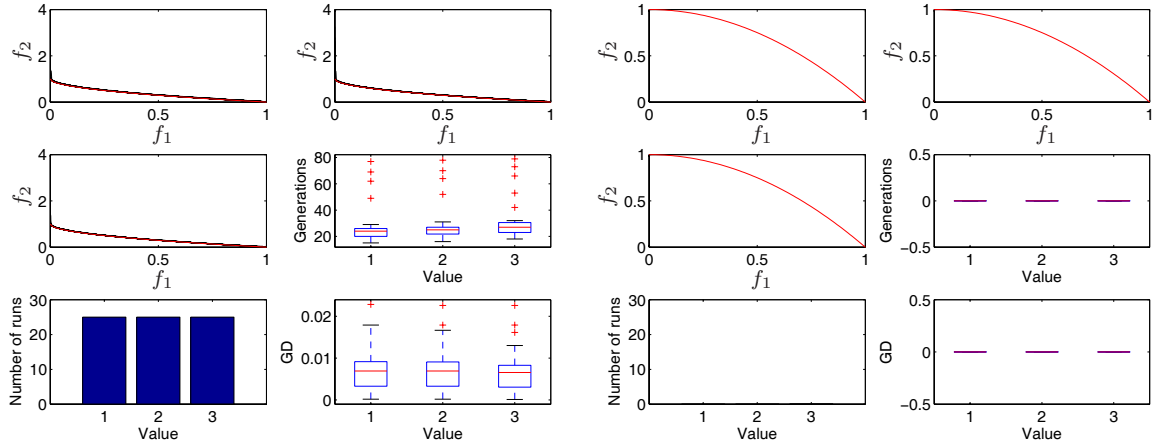
(d) ZDT4



(e) ZDT6

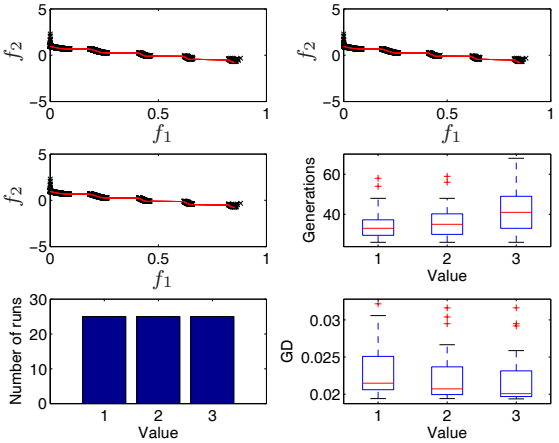
Figure C.17: *MaxCD* using DE

C.2 Results for Particle Swarm Optimization

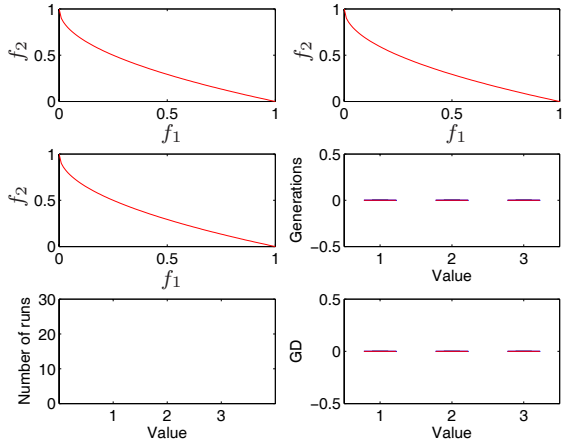


(a) ZDT1

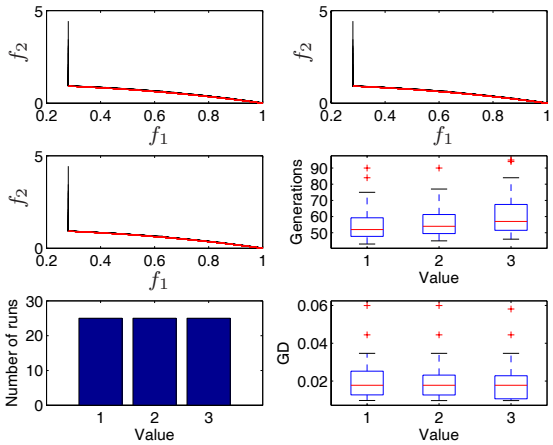
(b) ZDT2



(c) ZDT3

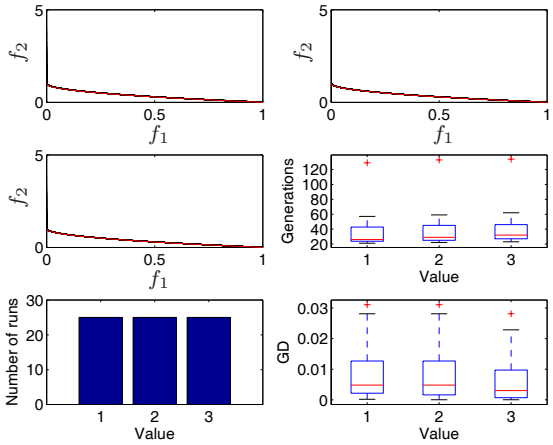


(d) ZDT4

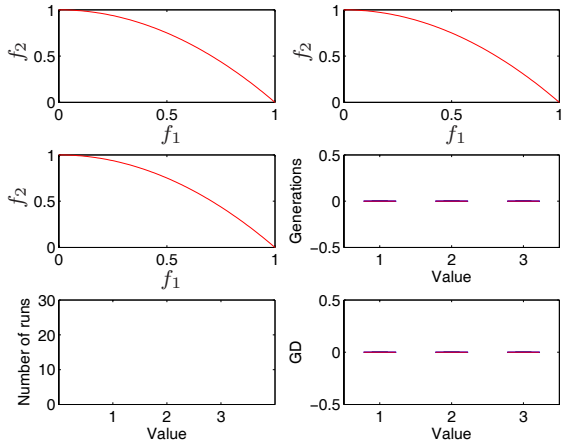


(e) ZDT6

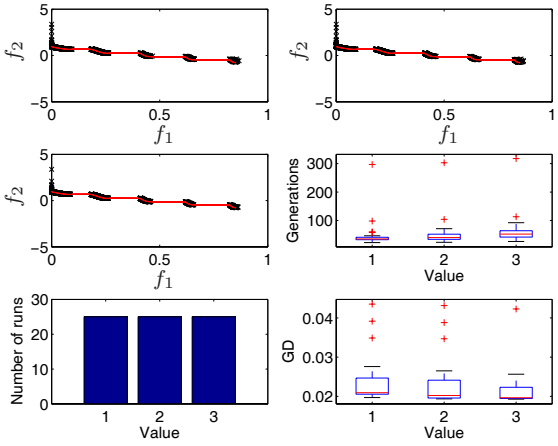
Figure C.18: *RefCritER* using PSO with $\epsilon_{ER} = 0.015$



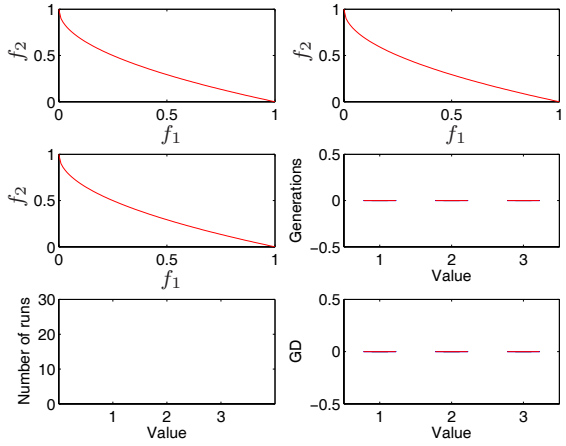
(a) ZDT1



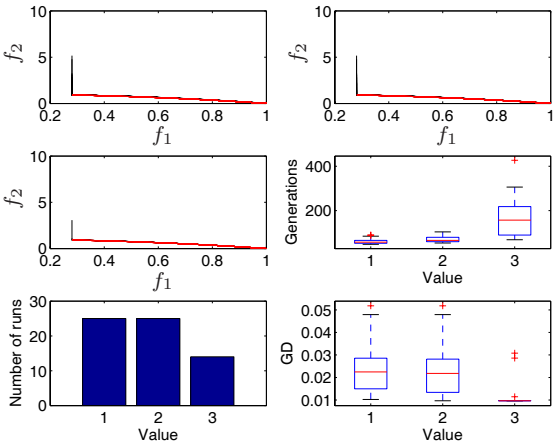
(b) ZDT2



(c) ZDT3

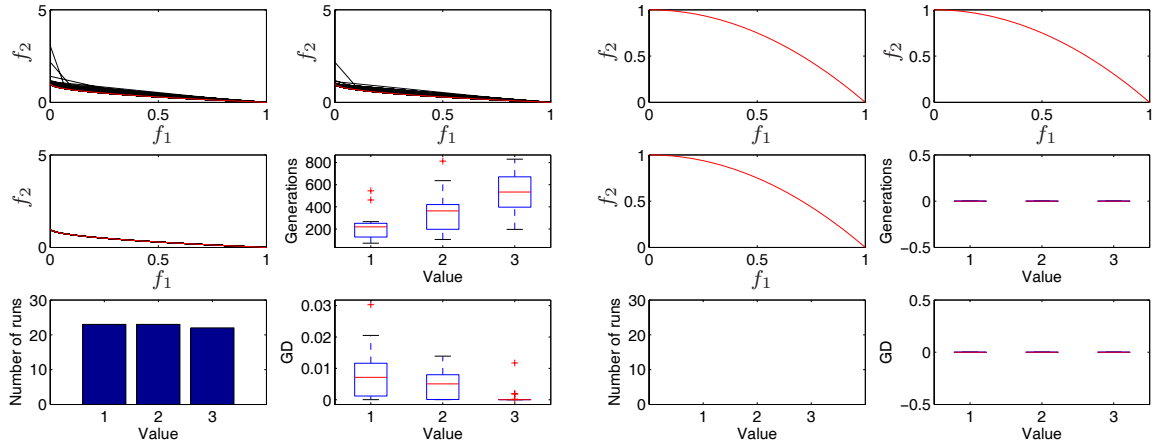


(d) ZDT4



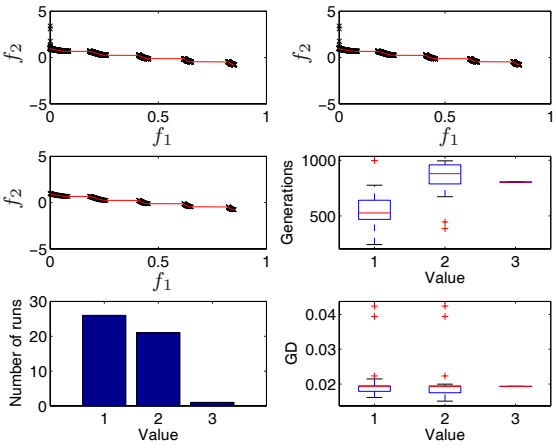
(e) ZDT6

Figure C.19: *RefCritER* using PSO with $\epsilon_{ER} = 0.01$

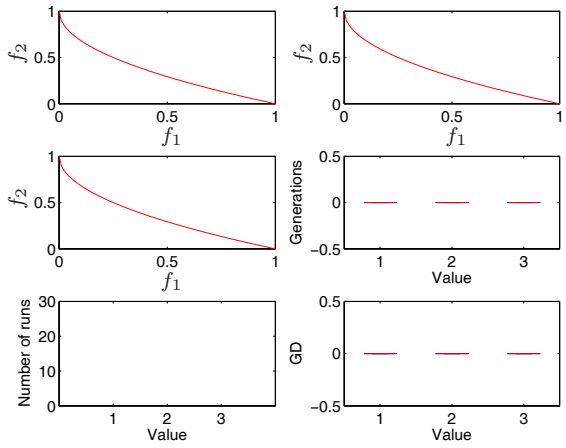


(a) ZDT1

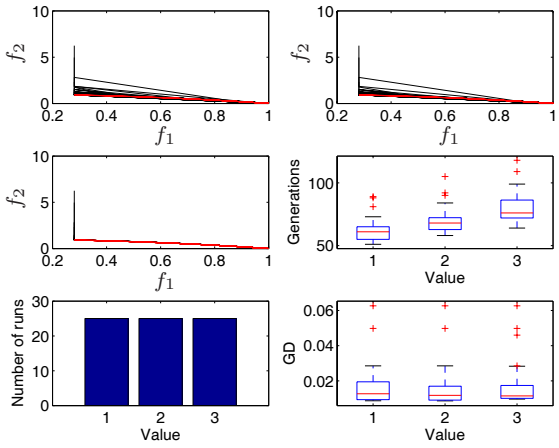
(b) ZDT2



(c) ZDT3

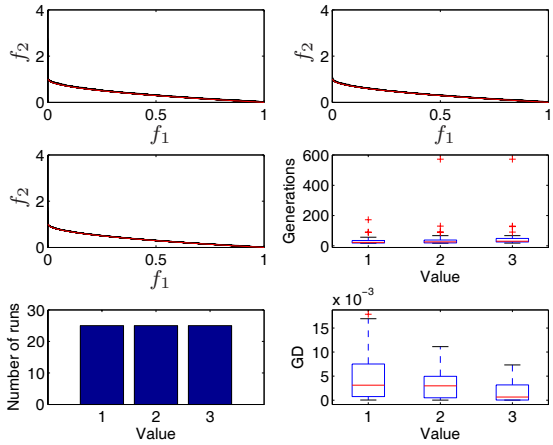


(d) ZDT4

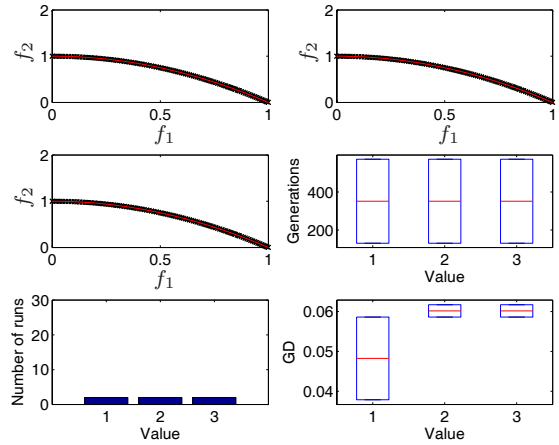


(e) ZDT6

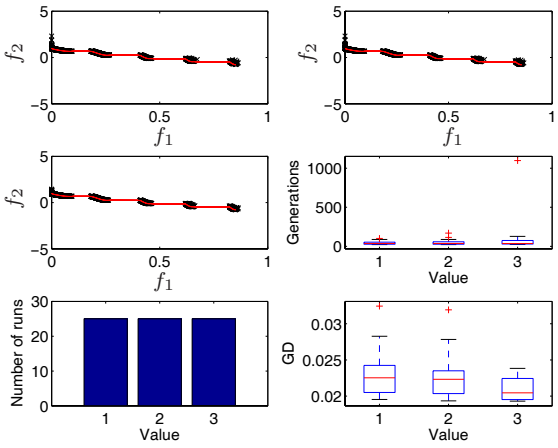
Figure C.20: *RefCritC* using PSO



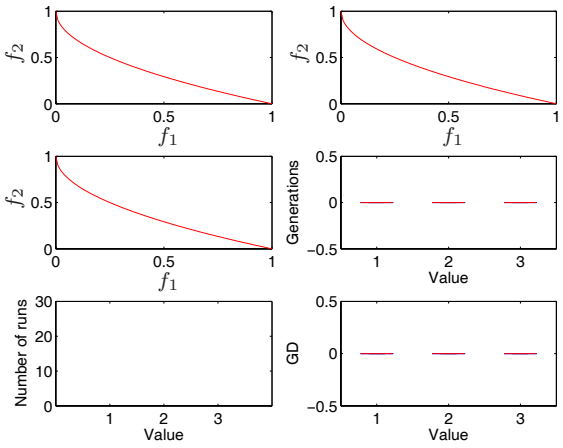
(a) ZDT1



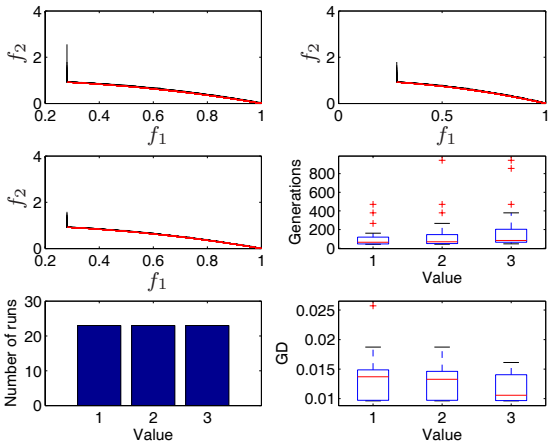
(b) ZDT2



(c) ZDT3

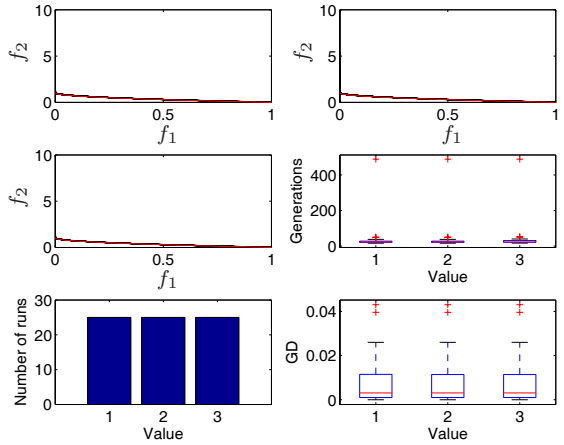


(d) ZDT4

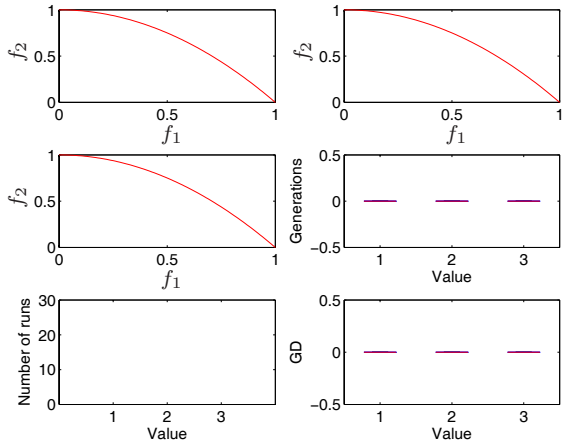


(e) ZDT6

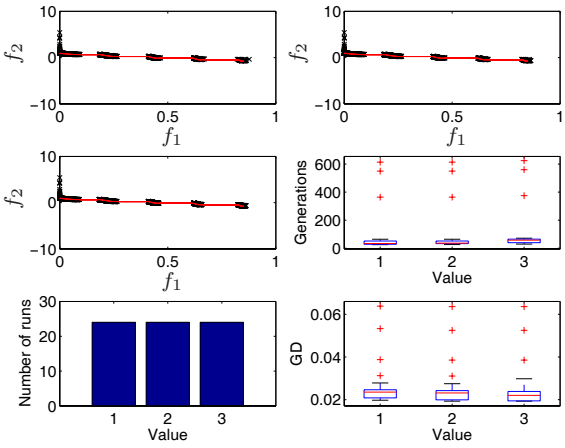
Figure C.21: *RefCritGD* using PSO



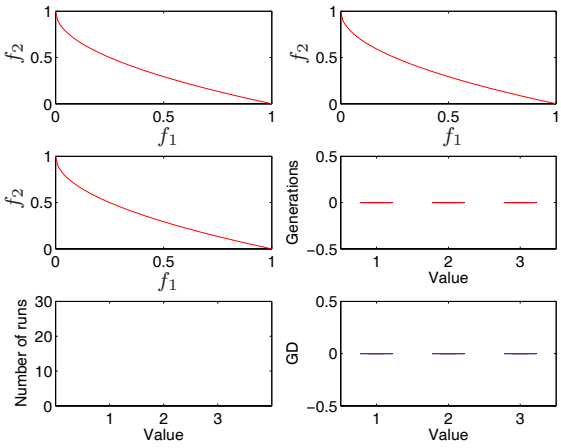
(a) ZDT1



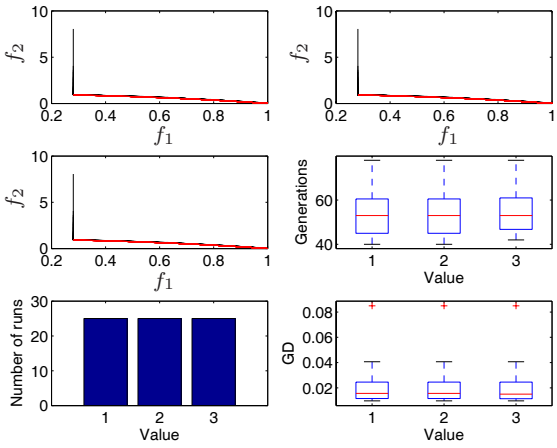
(b) ZDT2



(c) ZDT3

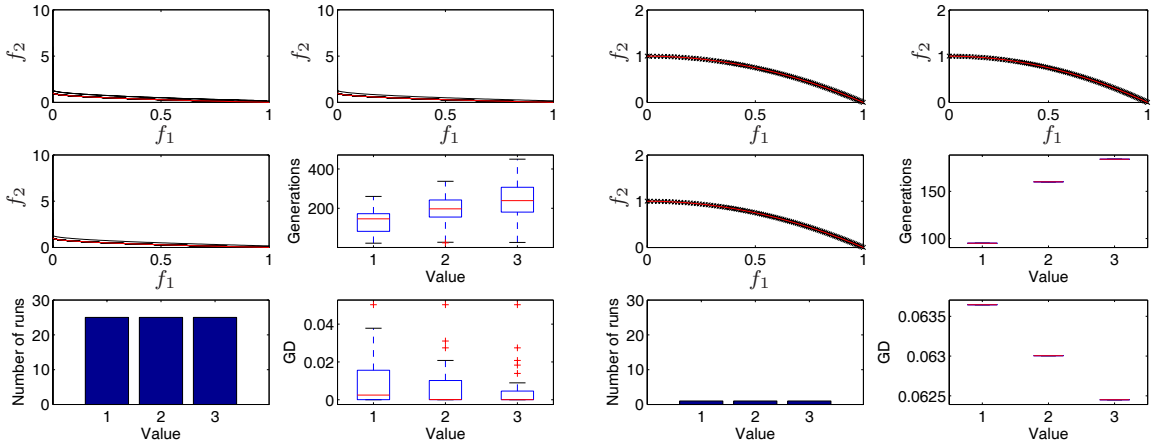


(d) ZDT4



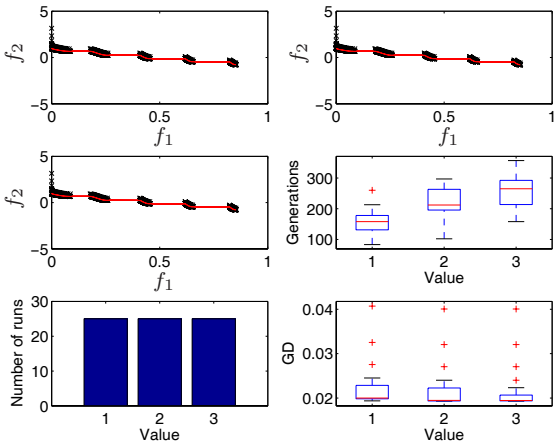
(e) ZDT6

Figure C.22: *RefCritHV* using PSO

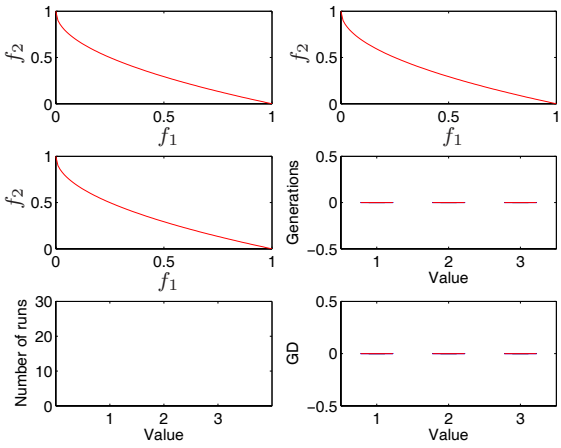


(a) ZDT1

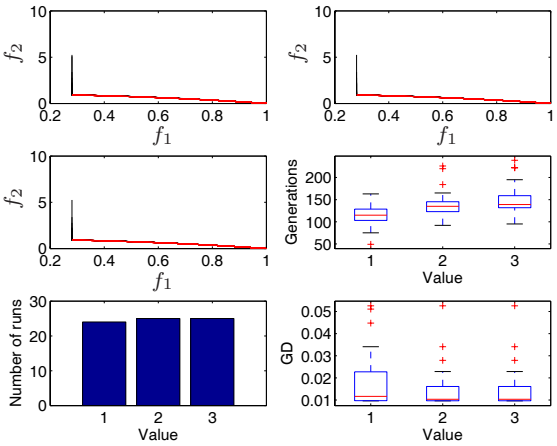
(b) ZDT2



(c) ZDT3

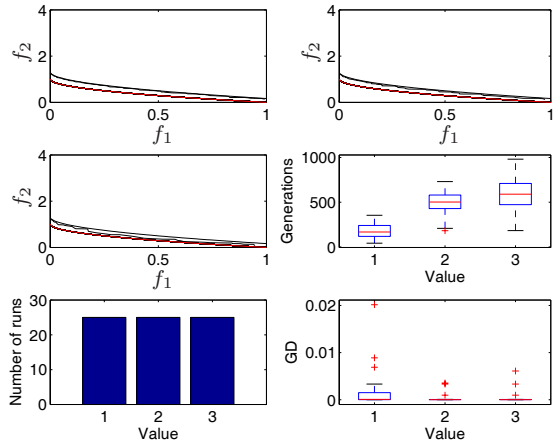


(d) ZDT4

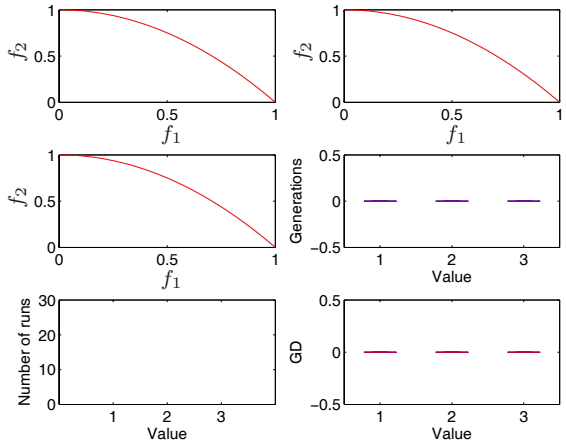


(e) ZDT6

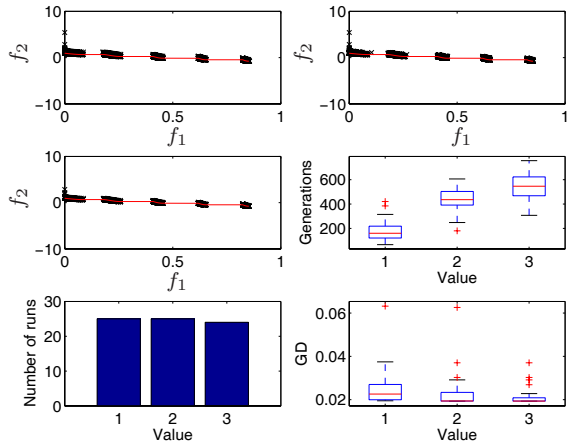
Figure C.23: *ImpC* using PSO



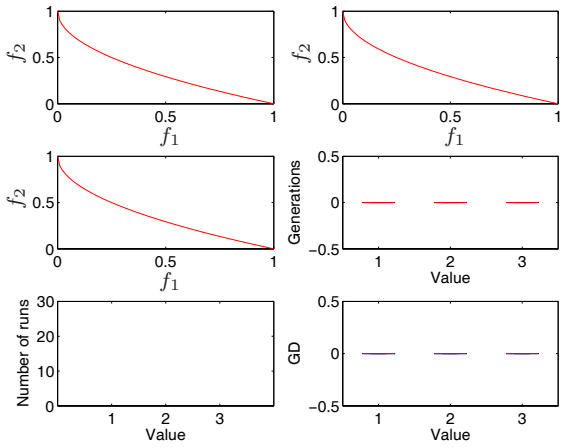
(a) ZDT1



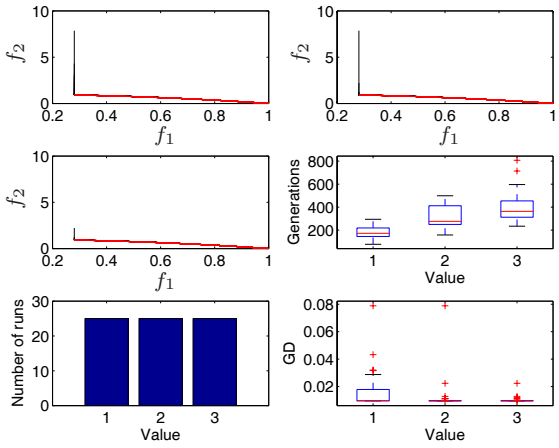
(b) ZDT2



(c) ZDT3

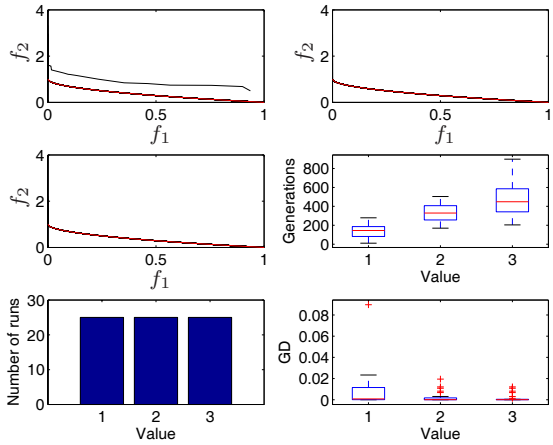


(d) ZDT4

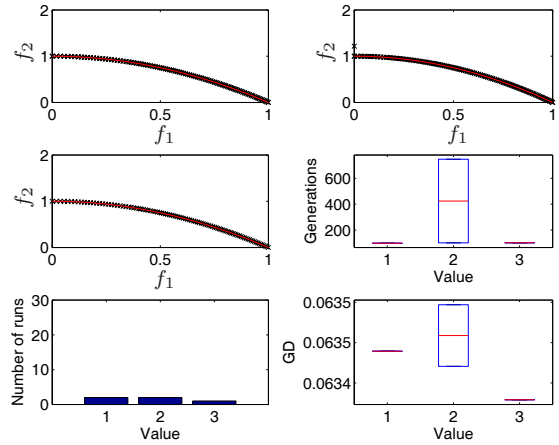


(e) ZDT6

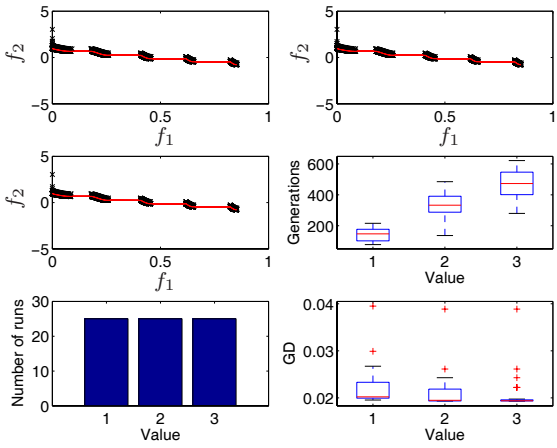
Figure C.24: *ImpGD* using PSO



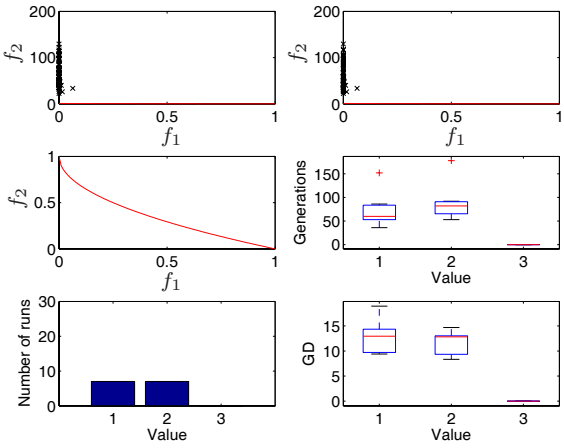
(a) ZDT1



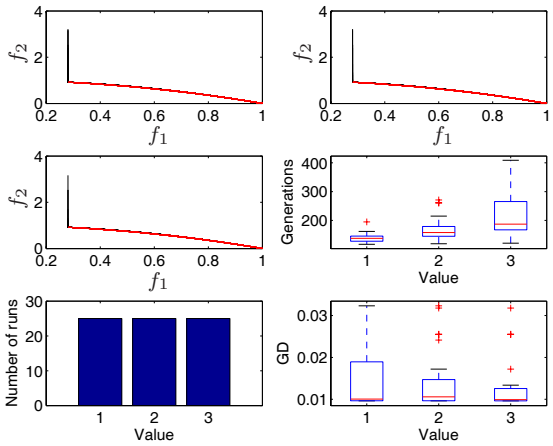
(b) ZDT2



(c) ZDT3

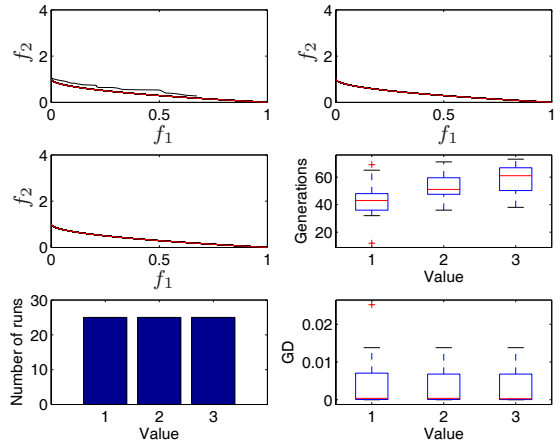


(d) ZDT4

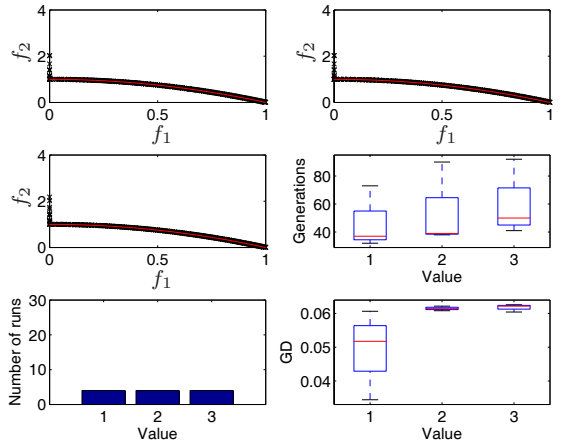


(e) ZDT6

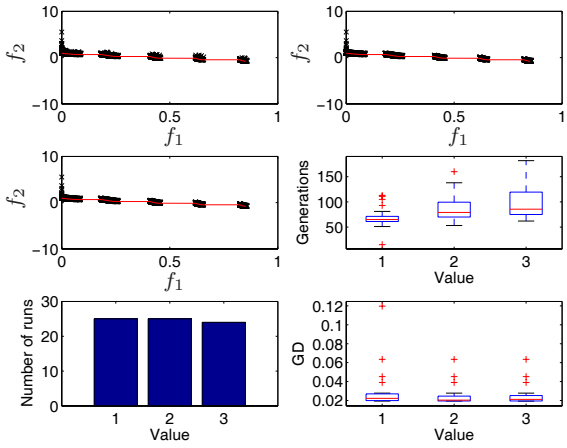
Figure C.25: *ImpHV* using PSO with $t_{HV} = 0$



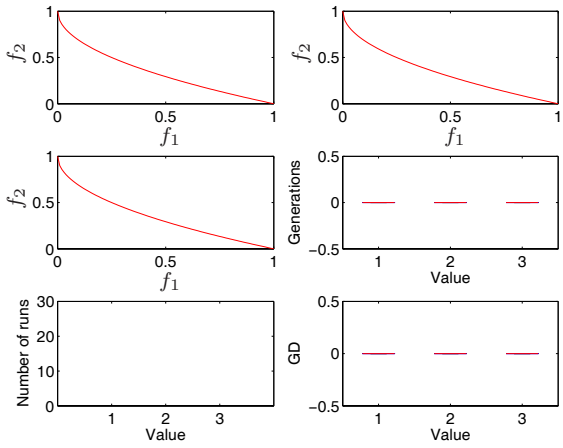
(a) ZDT1



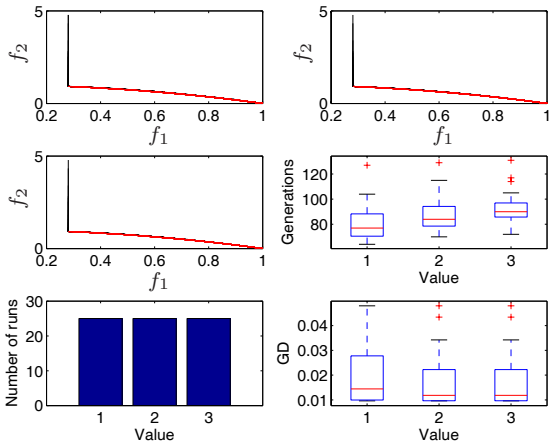
(b) ZDT2



(c) ZDT3

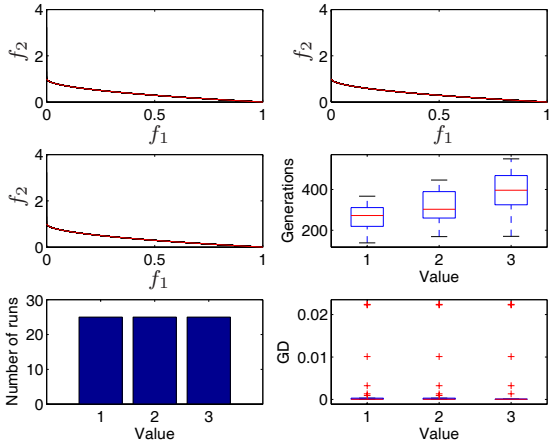


(d) ZDT4

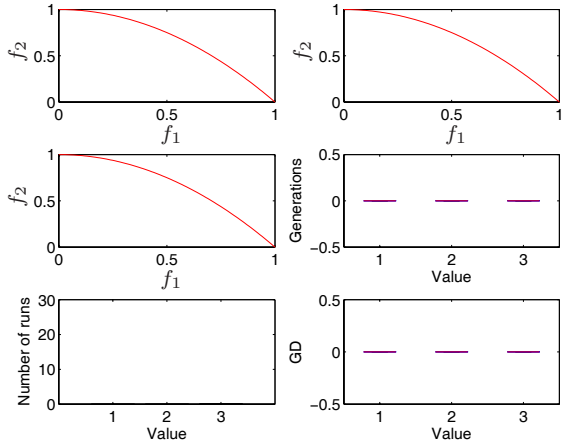


(e) ZDT6

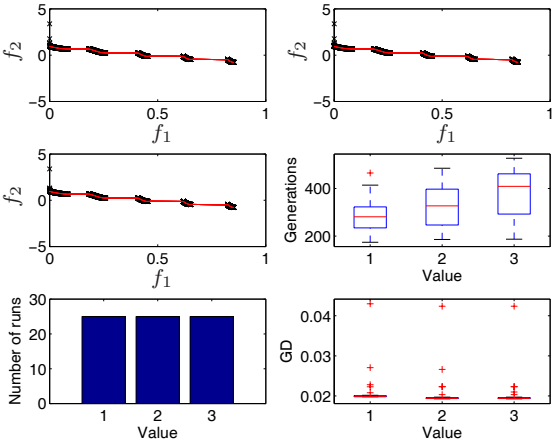
Figure C.26: *ImpHV* using PSO with $t_{HV} = 0.0001$



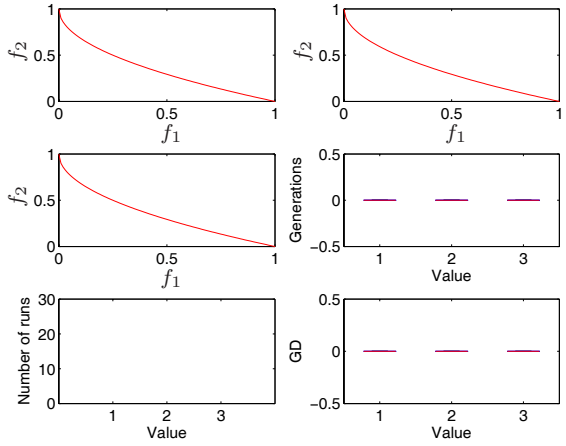
(a) ZDT1



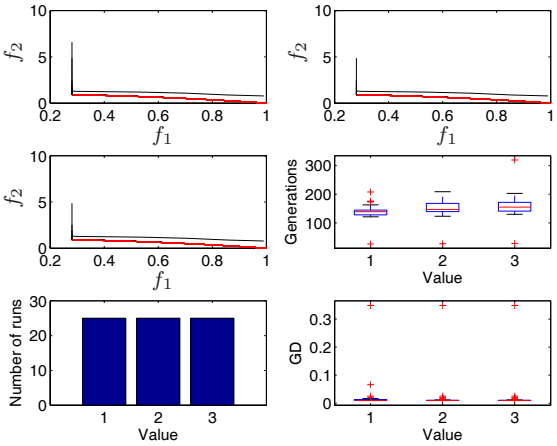
(b) ZDT2



(c) ZDT3

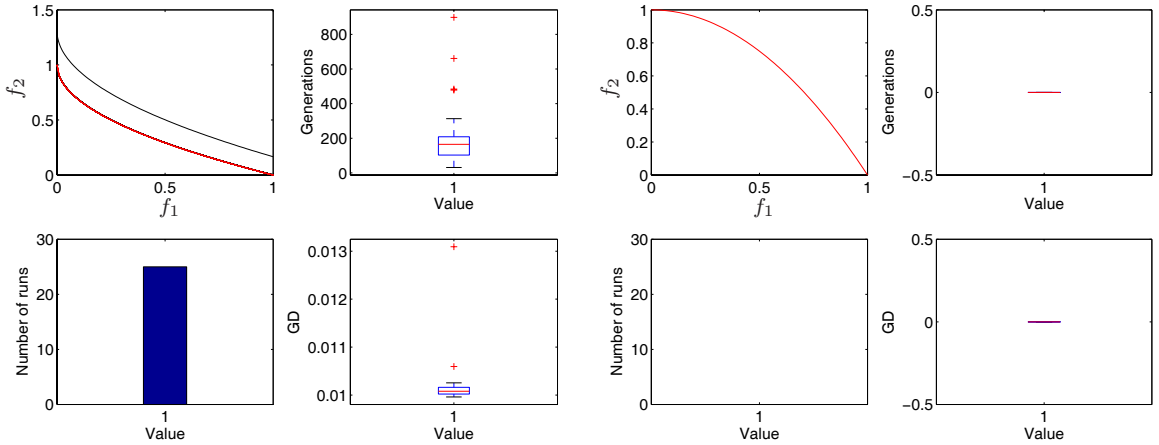


(d) ZDT4



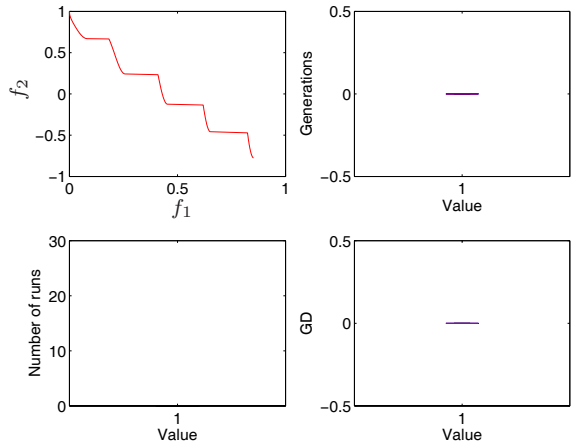
(e) ZDT6

Figure C.27: *NoAcc_MO* using PSO

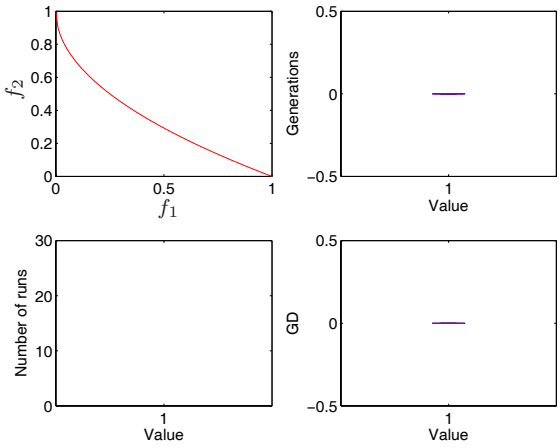


(a) ZDT1

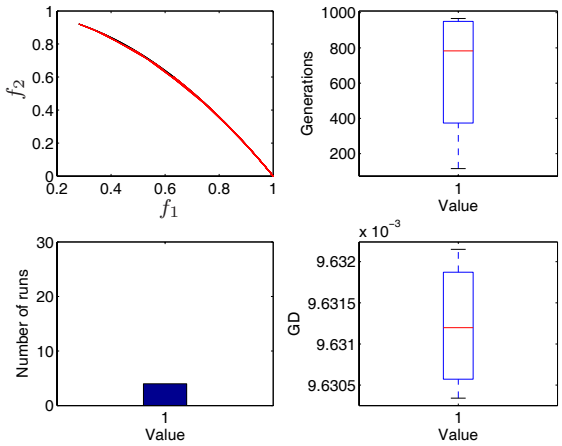
(b) ZDT2



(c) ZDT3

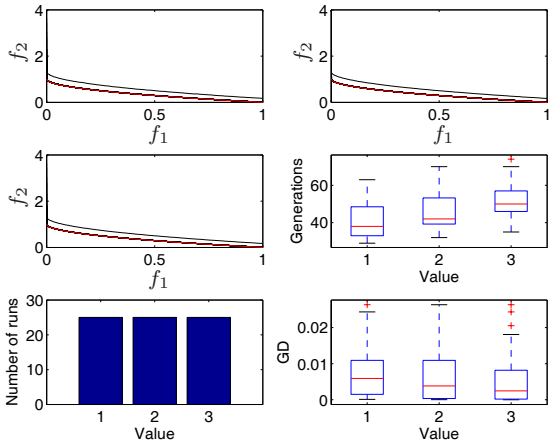


(d) ZDT4

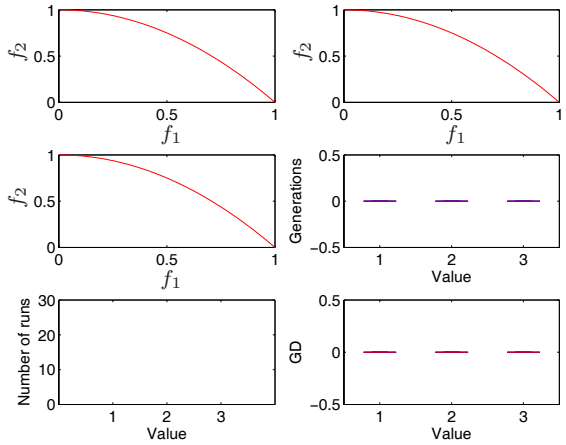


(e) ZDT6

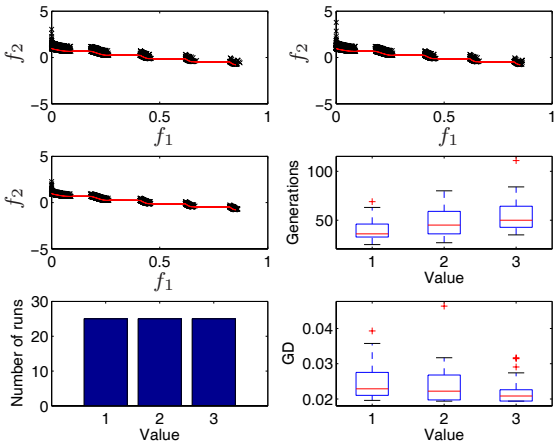
Figure C.28: *IdParetoRank* using PSO



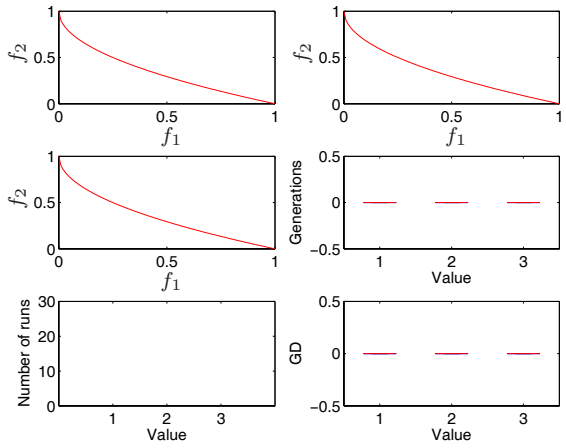
(a) ZDT1



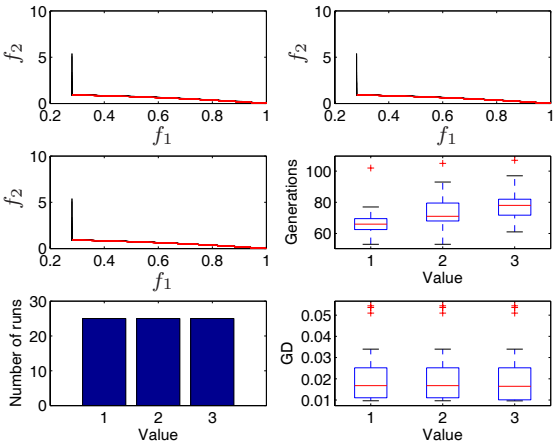
(b) ZDT2



(c) ZDT3

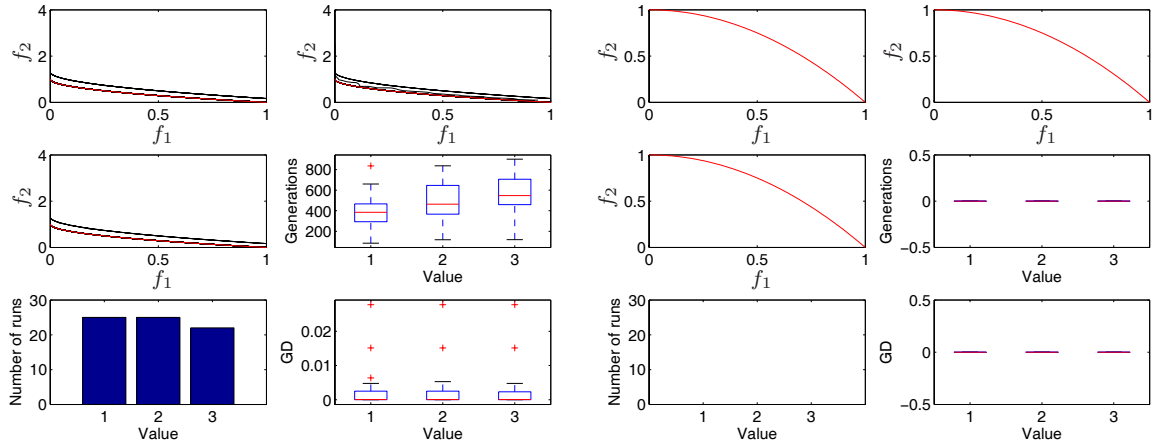


(d) ZDT4



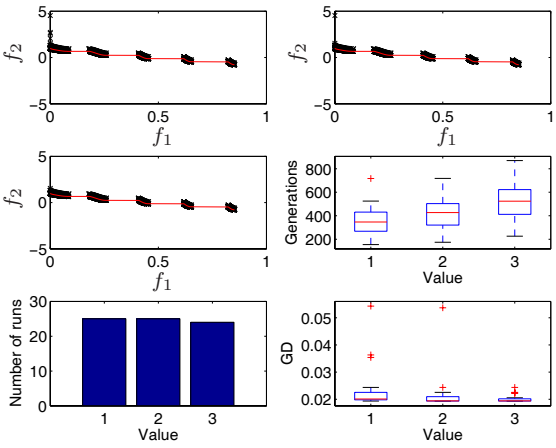
(e) ZDT6

Figure C.29: *DomInd* using PSO

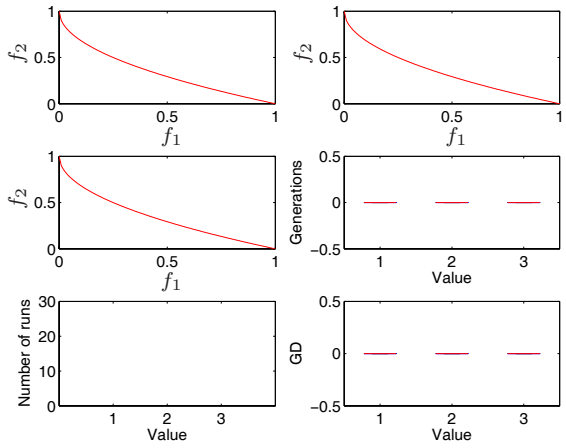


(a) ZDT1

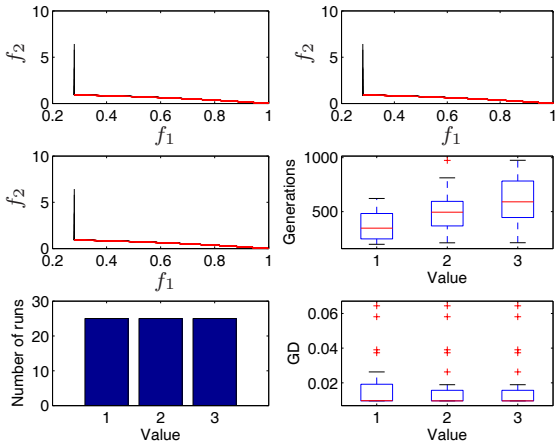
(b) ZDT2



(c) ZDT3

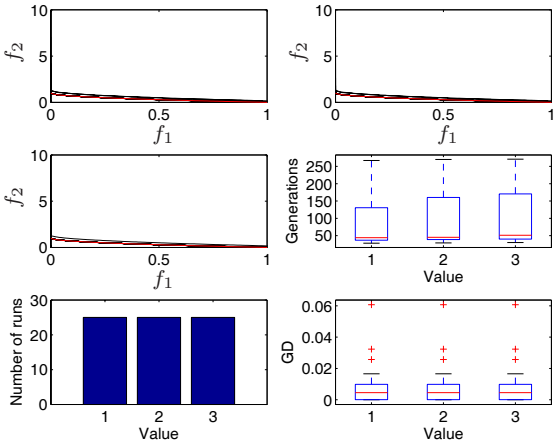


(d) ZDT4

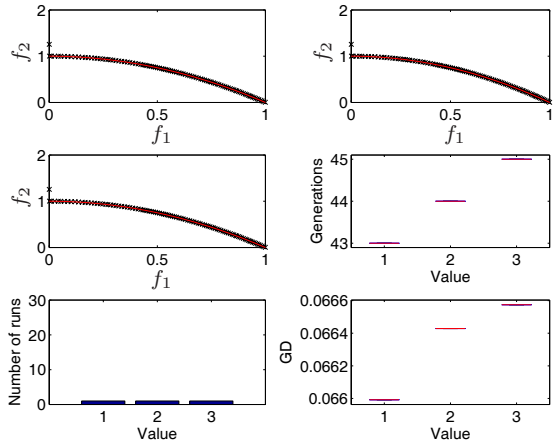


(e) ZDT6

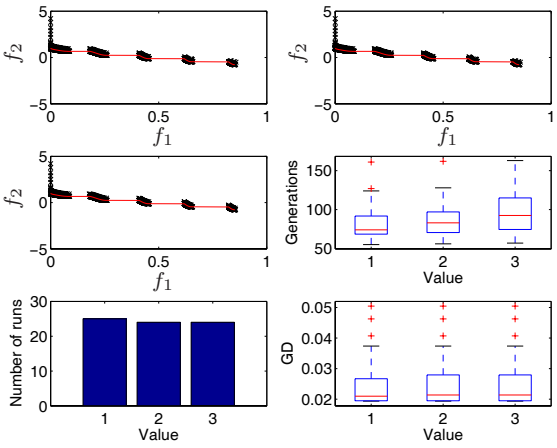
Figure C.30: *DistSpacing* using PSO with $t_s = 0$



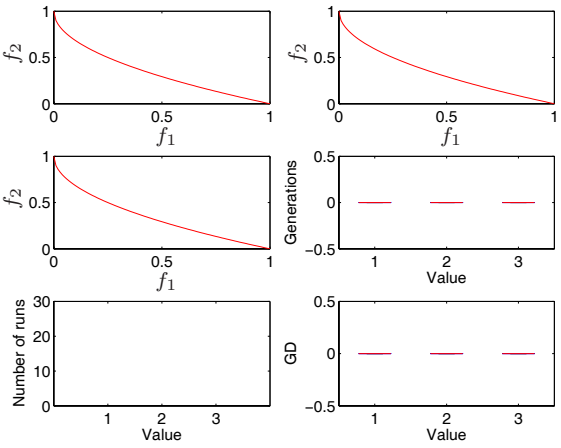
(a) ZDT1



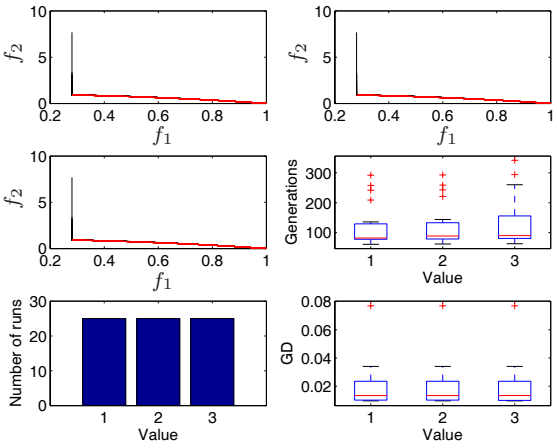
(b) ZDT2



(c) ZDT3

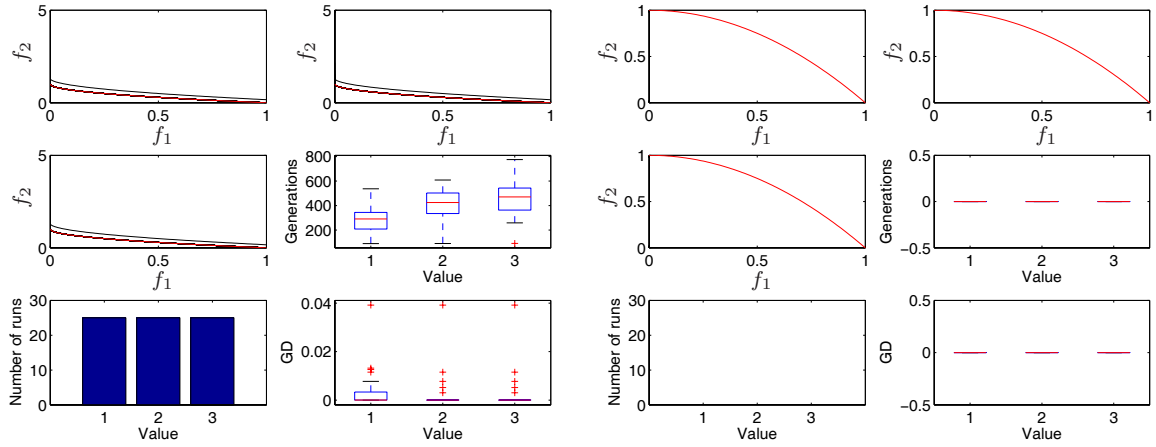


(d) ZDT4



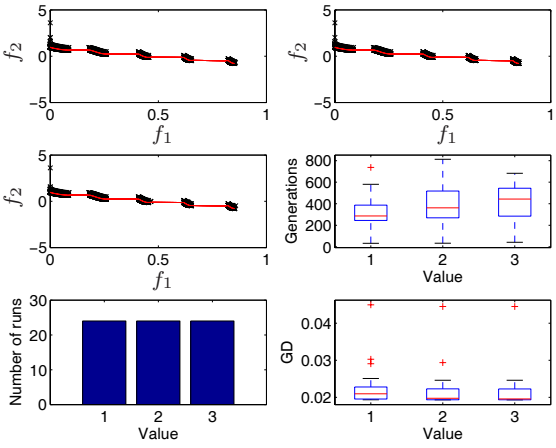
(e) ZDT6

Figure C.31: *DistSpacing* using PSO with $t_s = 0.001$

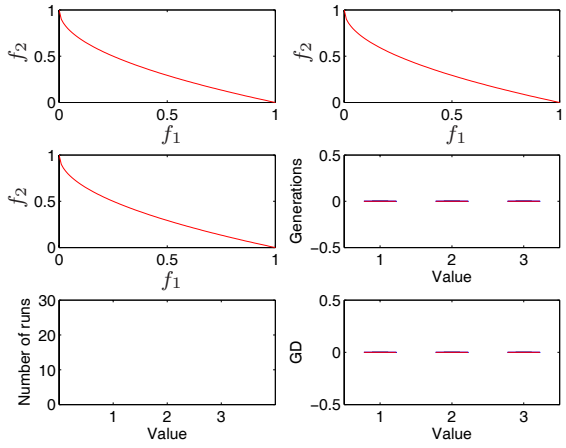


(a) ZDT1

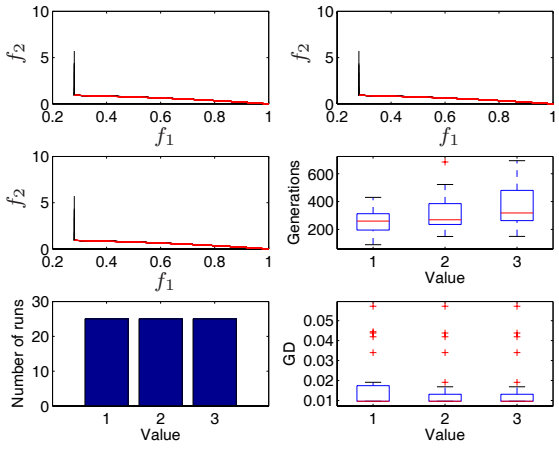
(b) ZDT2



(c) ZDT3

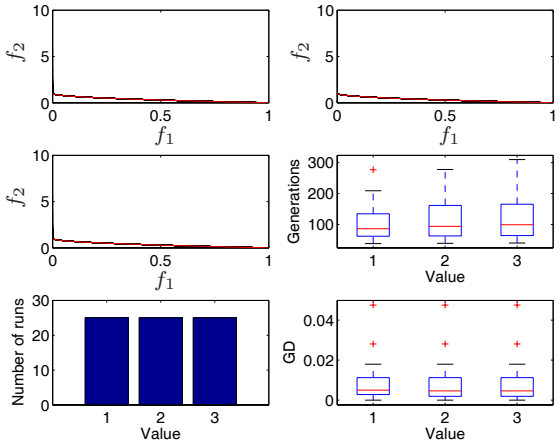


(d) ZDT4

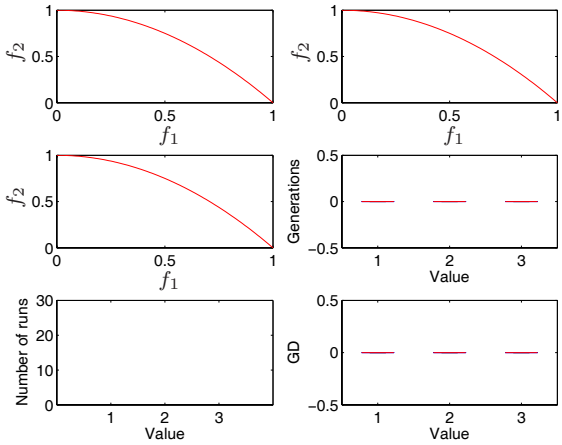


(e) ZDT6

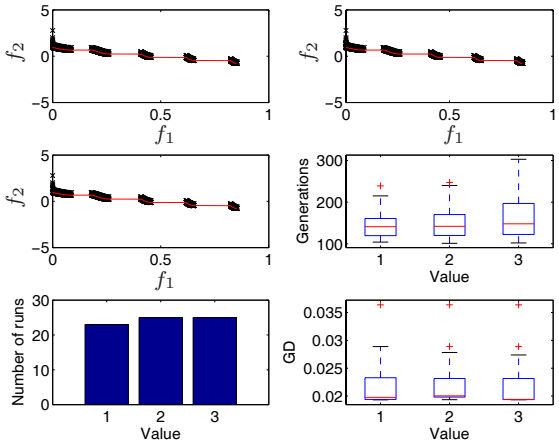
Figure C.32: *DistSpread* using PSO with $t_{sp} = 0$



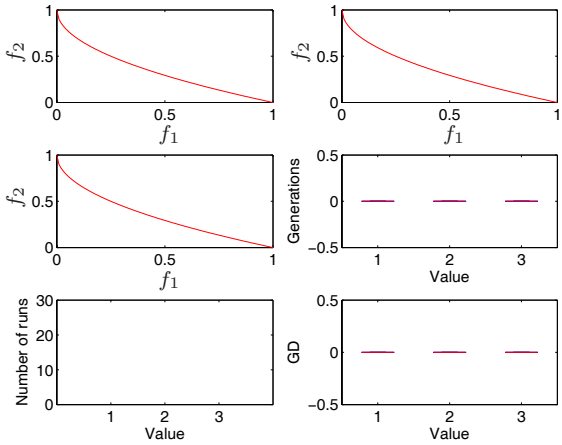
(a) ZDT1



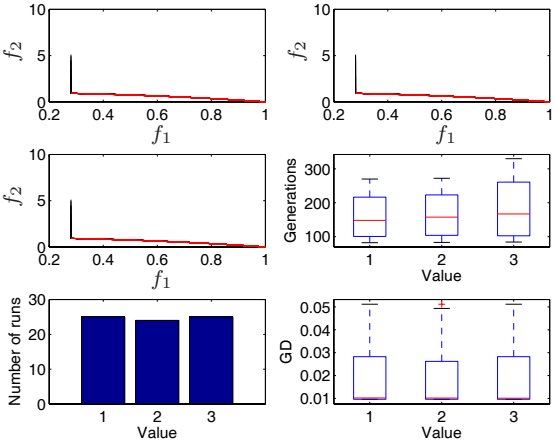
(b) ZDT2



(c) ZDT3

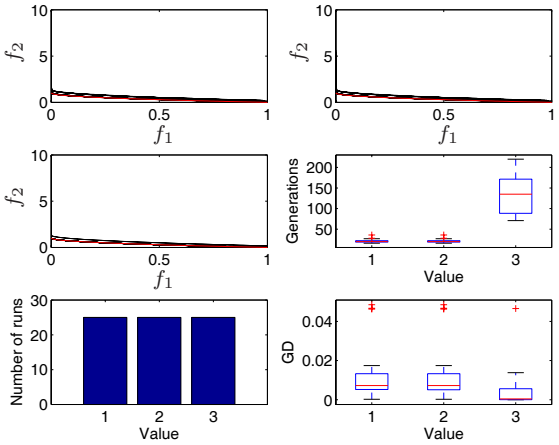


(d) ZDT4

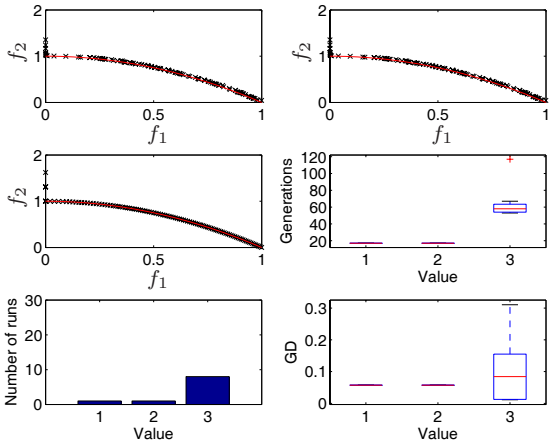


(e) ZDT6

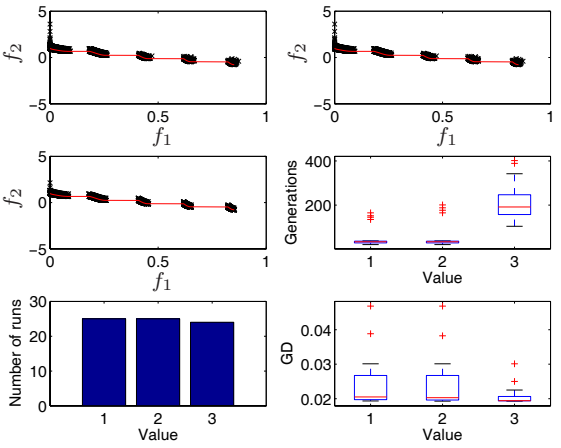
Figure C.33: *DistSpread* using PSO with $t_{sp} = 0.0001$



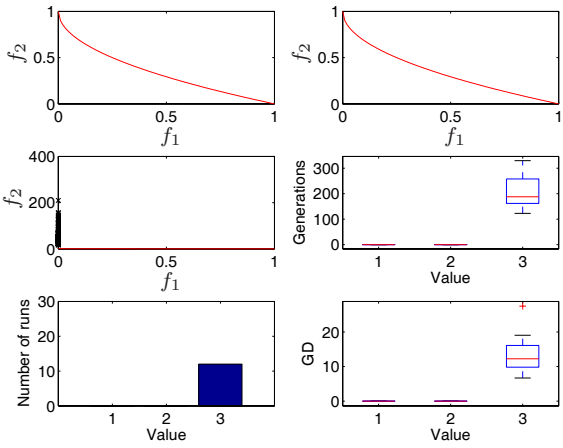
(a) ZDT1



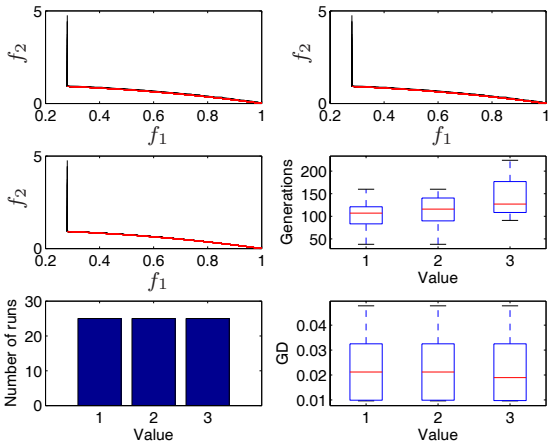
(b) ZDT2



(c) ZDT3



(d) ZDT4



(e) ZDT6

Figure C.34: *MaxCD* using PSO

Bibliography

[Abb01] Hussein A. Abbass, Ruhul Sarker, and Charles Newton. PDE: A Pareto-frontier Differential Evolution Approach for Multi-objective Optimization Problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2001.

[Abb02] Hussein A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 831–836, Honolulu, HI, USA, 2002.

[Adr07] Salem F. Adra, Ian Griffin, and Peter J. Fleming. A Comparative Study of Progressive Preference Articulation Techniques for Multiobjective Optimisation. In *Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization*, 2007.

[Ali04] M. M. Ali and A. Törn. Population Set Based Global Optimization Algorithms: Some Modifications and Numerical Studies. *Computers and Operations Research*, 31(10):1703–1725, 2004.

[Alv05] Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, 2005.

[Ama97] José Nelson Amaral, Adalberto Teixeira Castelo Neto, and Alessandro Valério Dias. Genetic Algorithms in Optimization: Better than Random Search? In *Proceedings of the International Conference on Engineering and Informatics*, 1997.

[Ang95] Peter J. Angeline. Adaptive and Self-Adaptive Evolutionary Computation. In Marimuthu Palaniswami and Yianni Attikiouzel, editors, *Computational Intelligence*, pages 152–163. IEEE Press, 1995.

[Ayt00] Haldun Aytug and Gary J. Koehler. New Stopping Criterion for Genetic Algorithms. *European Journal of Operational Research*, 126(3):662–674, 2000.

[Bab02] B. V. Babu and Rakesh Angira. A Differential Evolution Approach for Global Optimization of MINLP Problems. In *Proceedings of 4th Asia-Pacific Conference on Simulated Evolution And Learning*, 2002.

[Bab03a] B. V. Babu and Rakesh Angira. New Strategies of Differential Evolution for Optimization of Extraction Process. In *Proceedings of International Symposium & 56th Annual Session of IIChE (CHEMCON 2003)*, Bhubaneswar, India, 2003.

[Bab03b] B. V. Babu and M. Mathew Leenus Jehan. Differential Evolution for Multi-Objective Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2003.

[Bäc97] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel. Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[Bar03] Thomas Bartz-Beielstein, Philipp Limbourg, Jörn Mehnen, Karlheinz Schmitt, Konstantinos E. Parsopoulos, and Michael N. Vrahatis. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2003.

[Bar05] Thomas Bartz-Beielstein. *New Experimentalism Applied to Evolutionary Computation*. PhD thesis, University of Dortmund, 2005.

[Bec05] Ricardo Landa Becerra and Carlos A. Coello Coello. Optimization with Constraints using a Cultured Differential Evolution Approach. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, 2005.

[Beg08] Benjamin Begandt. Implementierung und Bewertung von Abbruchkriterien für multikriterielle Optimierung. Studienarbeit (student research project), University of Bremen, 2008.

[Ber01] Frans van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, University of Pretoria, 2001.

[Bey01] Hans-Georg Beyer and Kalyanmoy Deb. On Self-Adaptive Features in Real-Parameter Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3):250–270, 2001.

[Bol01] Hilmar Bolte, Dagmar Peters, Oliver Nüssen, Rainer Laur, Erich Huck, Dietmar Richter, and Gerhard Gärtner. Verhaltensmodelle und multikriterielle Optimierung im Entwurfsprozeß von komplexen Systemen am Beispiel einer Mehrfachionisationskammer. In *Proceedings der ASIM, 15. Symposium Simulationstechnik*, 2001.

[Bol03] Hilmar Bolte. *Modellbildung und Designoptimierung eines neuartigen radiometrischen Sensors*. PhD thesis, University of Bremen, 2003.

[Bra07] Daniel Bratton and James Kennedy. Defining a Standard for Particle Swarm Optimization. In *Proceedings of the IEEE Swarm Intelligence Symposium*, 2007.

[Bre06a] Janez Brest, Sašo Greiner, Borko Bošković, Marjan Mernik, and Viljem Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[Bre06b] Janez Brest, Viljem Žumer, and Mirjam Sepesy Maučec. Control Parameters in Self-Adaptive Differential Evolution. In *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications*, pages 35–44, Ljubljana, Slovenia, 2006.

[Bre06c] Janez Brest, Viljem Žumer, and Mirjam Sepesy Maučec. Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 919–926, Vancouver, BC, Canada, 2006.

[Bre08] Janez Brest, Aleš Zamuda, Borko Bošković, Mirjam Sepesy Maučec, and Viljem Žumer. High-Dimensional Real-Parameter Optimization using Self-Adaptive Differential Evolution Algorithm with Population Size Reduction. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2032–2039, 2008.

[Bro00] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*. Harri Deutsch, 2000.

[Cai04] Giuseppe Caire, Ralf R. Müller, and Toshiyuki Tanaka. Iterative Multiuser Joint Decoding: Optimal Power Allocation and Low-Complexity Implementation. *IEEE Transactions on Information Theory*, 50(9):1950–1973, 2004.

[Car01] Anthony Carlisle and Gerry Dozier. An Off-The-Shelf PSO. In *Proceedings of the Workshop on Particle Swarm Optimization*, 2001.

[Chu04] Ji Chunlin. A Revised Particle Swarm Optimization Approach for Multi-Objective and Multi-Constraint Optimization. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, 2004.

[Cle02] Maurice Clerc and James Kennedy. The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[Cle06] Maurice Clerc. *Particle Swarm Optimization*. ISTE Ltd, 2006.

[Coa03] Genevieve Coath and Saman K. Halgamuge. A Comparison of Constraint-Handling Methods for the Application of Particle Swarm Optimization to Constrained Nonlinear Optimization Problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2419–2425, 2003.

[Coe02a] Carlos A. Coello Coello. Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, 2002.

[Coe02b] Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1051–1056, 2002.

[Coe03] Carlos A. Coello Coello. Evolutionary Multiobjective Optimization: Current and Future Challenges. In J.M. Benitez, O. Cordon, F. Hoffmann, and R. Roy, editors, *Advances in Soft Computing - Engineering, Design and Manufacturing*, pages 243–256. Springer-Verlag, 2003.

[Coe04] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 2004.

[Cor03] David Corne and Joshua Knowles. Some Multiobjective Optimizers are Better than Others. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2003.

[Deb00] Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338, 2000.

[Deb01a] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[Deb01b] Kalyanmoy Deb and Tushar Goel. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, pages 67–81, 2001.

[Deb01c] Kalyanmoy Deb, Amrit Pratap, and T. Meyarivan. Constrained Test Problems for Multi-Objective Evolutionary Optimization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, 2001.

[Deb02] Kalyanmoy Deb, Amrit Pratap, Samir Agrawal, and T. Meyarian. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[Deb05] Kalyanmoy Deb. A Population-Based Algorithm-Generator for Real-Parameter Optimization. *Soft Computing*, 9(4):236–253, 2005.

[Dee07] Neele von Deetzen and Sara Sandberg. Design of Unequal Error Protection LDPC Codes for Higher Order Constellations. In *Proceedings of the IEEE International Conference on Communications*, 2007.

[Dor02] Marco Dorigo and Thomas Stützle. The Ant Colony Optimization Metaheuristic: Algorithms, Applications and Advances. In Fred Glover and Gary Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, chapter 9, pages 251–285. Kluwer Academic Publishers, Boston, MA, 2002.

[Ebe95] R.C. Eberhart and James Kennedy. A New Optimizer Using Particle Swarm Theory. In *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, 1995.

[Ebe00] R.C. Eberhart and Y. Shi. Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2000.

[Eib99] Ágoston Endre Eiben, Robert Hinterding, and Zbigniew Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124–141, 1999.

[Eng06] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, 2006.

[Esp03] Felipe P. Espinoza. *A Self-Adaptive Hybrid Genetic Algorithm for Optimal Groundwater Remediation Design*. PhD thesis, University of Illinois, 2003.

[Fal01] André O. Falcão and José G. Borges. Designing an Evolution Program for Solving Integer Forest Management Scheduling Models: An Application in Portugal. *Forest Science*, 47(2):158–168, 2001.

[Feo04] Vitaliy Feoktistov and Stefan Janaqi. New Strategies in Differential Evolution. In *Proceedings of the 6th International Conference on Adaptive Computing in Design and Manufacture*, 2004.

[Fie03] Jonathan Edward Fieldsend. *Novel Algorithms for Multi-Objective Search and their application in Multi-Objective Evolutionary Neural Network Training*. PhD thesis, University of Exeter, 2003.

[Fle05] Peter J. Fleming, Robin C. Purshouse, and Robert J. Lygoe. Many-Objective Optimization: An Engineering Design Perspective. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, 2005.

[Fog00] David B. Fogel. Real-Valued Vectors. In Thomas Baeck, David B. Fogel, and Zbigniew Michalewicz, editors, *Evolutionary Computation 1: Basic Algorithms and Operators*, pages 136–138. Institute of Physics Publishing, 2000.

[Fon96] Carlos M. Fonseca and Peter J. Fleming. On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In *Proceedings of Parallel Problem Solving from Nature*, pages 584–593, 1996.

[Gäm02] Roger Gämperle, Sibylle D. Müller, and Petros Koumoutsakos. A Parameter Study for Differential Evolution. In A. Grmela and N.E. Mastorakis, editors, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 293–298. WSEAS Press, 2002.

[Gen04] James E. Gentle, Wolfgang Härdle, and Yuichi Mori, editors. *Handbook of Computational Statistics*. Springer, 2004.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[Ham02] Sana B. Hamida and Marc Schoenauer. ASCHEA: New Results Using Adaptive Segregational Constraint Handling. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2002.

[Her05] German Hernandez, Kenneth Wilder, Fernando Nino, and Julian Garcia. Towards a Self-Stopping Evolutionary Algorithm Using Coupling from the Past. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 615–620, 2005.

[Hol75] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1975.

[Hou05] Hao-Sheng Hou, Shoou-Jinn Chang, and Yan-Kuin Su. Practical Passive Filter Synthesis Using Genetic Programming. *IEICE Transactions on Electronics*, E88-C(6):1180–1185, 2005.

[Hsi08] Sheng-Ta Hsieh, Tsung-Ying Sun, Chan-Cheng Liu, and Shang-Jeng Tsai. Solving Large Scale Global Optimization Using Improved Particle Swarm Optimizer. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2008.

[Hu02] Xiaohui Hu and Russell C. Eberhart. Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics*, 2002.

[Hua06] V.L. Huang, A.K. Qin, and P.N. Suganthan. Self-Adaptive Differential Evolution Algorithm for Constrained Real-Parameter Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 324–331, Vancouver, BC, Canada, 2006.

[Hua07a] V.L. Huang, A.K. Qin, K. Deb, P.N. Suganthan, J.J. Liang, M. Preuss, and S. Huband. Problem Definitions for Performance Assessment & Competition on Multi-Objective Optimization Algorithms. Technical report, Nanyang Technological University, Singapore, January 2007.

[Hua07b] V.L. Huang, A.K. Qin, P.N. Suganthan, and M.F. Tasgetiren. Multi-Objective Optimization based on Self-Adaptive Differential Evolution Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3601–3608, Singapore, 2007.

[Hug06] Evan J. Hughes. Assessing Robustness of Optimisation Performance for Problems With Expensive Evaluation Functions. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006.

[Ior04] Antony Iorio and Xiaodong Li. Solving Rotated Multi-objective Optimization Problems Using Differential Evolution. In *Proceeding of the 17th Joint Australian Conference on Artificial Intelligence*, 2004.

[Ior06] Antony Iorio and Xiaodong Li. Incorporating Directional Information within a Differential Evolution Algorithm for Multiobjective Optimization. In *Proceedings of the Conference on Genetic and Evolutionary Computation*, 2006.

[Iwa06a] Masao Iwamatsu. Locating All the Global Minima Using Multi-Species Particle Swarm Optimizer: The Inertia Weight and the Constriction Factor Variants. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006.

[Iwa06b] Nobuhiro Iwasaki, Keiichiro Yasuda, and Genki Ueno. Dynamic Parameter Tuning of Particle Swarm Optimization. *IEEJ Transactions on Electrical and Electronic Engineering*, 1(4):353–363, 2006.

[Jak04] Wilfried Jakob. *Eine neue Methode zur Erhöhung der Leistungsfähigkeit Evolutionärer Algorithmen durch die Integration lokaler Suchverfahren.* PhD thesis, Universität Karlsruhe, 2004.

[Jia07] M. Jiang, Y. P. Luo, and S. Y. Yang. Stochastic Convergence Analysis and Parameter Selection of the Standard Particle Swarm Optimization Algorithm. *Information Processing Letters*, 102(1):8–16, 2007.

[Jin05] Yaochu Jin. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Computing*, 9(1):3–12, 2005.

[Kad06] Visakan Kadirkamanathan, Kirusnapillai Selvarajah, and Peter J. Fleming. Stability Analysis of the Particle Dynamics in Particle Swarm Optimizer. *IEEE Transactions on Evolutionary Computation*, 10(3):245–255, 2006.

[Ken95] James Kennedy and Russell Eberhart. Particle Swarm Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, 1995.

[Ken01] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, 2001.

[Ken02] James Kennedy and Rui Mendes. Population Structure and Particle Swarm Performance. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1671–1676, Honolulu, HI, USA, 2002.

[Ken06a] James Kennedy. In Search of the Essential Particle Swarm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2006.

[Ken06b] James Kennedy and Rui Mendes. Neighborhood Topologies in Fully Informed and Best-of-Neighborhood Particle Swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36(4):515–519, 2006.

[Ken07] James Kennedy. Some Issues and Practices for Particle Swarms. In *Proceedings of the IEEE Swarm Intelligence Symposium*, 2007.

[Kle04] Jack P.C. Kleijnen. Design and Analysis of Monte Carlo Experiments. Technical report, Tilburg University, 2004.

[Kno00] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[Kno02] Joshua D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, University of Reading, 2002.

[Kno06] Joshua D. Knowles, Lothar Thiele, and Eckart Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical report, ETH Zurich, February 2006. Revised version, first version January 2005.

[Kno08] Joshua Knowles, David Corne, and Kalyanmoy Deb. Introduction: Problem Solving, EC and EMO. In Joshua Knowles, David Corne, and Kalyanmoy Deb, editors, *Multiobjective Problem Solving from Nature*, Natural Computing Series, pages 1–28. Springer, 2008.

[Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[Koz99] Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[Koz04] John R. Koza, Lee W. Jones, Martin A. Keane, Matthew J. Streeter, and Sameer H. Al-Sakran. Toward Automated Design of Industrial-Strength Analog Circuits by Means of Genetic Programming. In Una-May O'Reilly, Rick L. Riolo, Gwoing Yu, and William Worzel, editors, *Genetic Programming Theory and Practice II*, pages 121–142. Kluwer Academic Publishers, 2004.

[Kri02] Thiemo Krink, Jakob S. Vesterstrøm, and Jacques Riget. Particle Swarm Optimisation with Spatial Particle Extension. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2002.

[Kuk04a] Saku Kukkonen and Jouni Lampinen. A Differential Evolution Algorithm for Constrained Multi-Objective Optimization: Initial Assessment. In *Proceedings of the IASTED International Conference on Artificial Intelligence and Applications*, pages 96–102, Innsbruck, Austria, 2004.

[Kuk04b] Saku Kukkonen and Jouni Lampinen. An Extension of Generalized Differential Evolution for Multi-Objective Optimization with Constraints. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, 2004.

[Kuk05] Saku Kukkonen and Jouni Lampinen. GDE3: The third Evolution Step of Generalized Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2005.

[Kuk06a] Saku Kukkonen and Kalyanmoy Deb. A Fast and Effective Method for Pruning of Non-Dominated Solutions in Many-Objective Problems. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*, 2006.

[Kuk06b] Saku Kukkonen and Kalyanmoy Deb. Improved Pruning of Non-Dominated Solutions Based on Crowding Distance for Bi-Objective Optimization Problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3995–4002, 2006.

[Kuk06c] Saku Kukkonen and Jouni Lampinen. Constrained Real-Parameter Optimization with Generalized Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 911–918, 2006.

[Kuk06d] Saku Kukkonen and Jouni Lampinen. An Empirical Study of Control Parameters for the Third Version of Generalized Differential Evolution (GDE3). In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 7355–7362, 2006.

[Lam00] Jouni Lampinen and Ivan Zelinka. On Stagnation of the Differential Evolution Algorithm. In *Proceedings of the 6th International Mendel Conference on Soft Computing*, 2000.

[Lam02] Jouni Lampinen. Multi-Constrained Nonlinear Optimization by the Differential Evolution Algorithm. In Rajkumar Roy, Mario Köppen, Seppo Ovaska, Takeshi Furuhasi, and Frank Hoffmann, editors, *Soft Computing and Industry - Recent Applications*, pages 305–318. Springer, London, 2002.

[Lam04] Jouni Lampinen and Rainer Storn. Differential Evolution. In Godfrey C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 123–166. Springer-Verlag, Berlin Heidelberg, 2004.

[Lan07] William B. Langdon and Riccardo Poli. Evolving Problems to Learn about Particle Swarm Optimizers and Other Search Algorithms. *IEEE Transactions on Evolutionary Computation*, 11(5):561–578, 2007.

[Li03] Xiaodong Li. A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2003.

[Li04] Ning Li, Fei Liu, Debao Sub, and Chang Huang. Particle Swarm Optimization for Constrained Layout Optimization. In *Proceedings of the 5th World Congress on Intelligent Control and Automation*, pages 2214–2218, 2004.

[Lia06a] J.J. Liang, Thomas Philip Runarsson, Efrén Mezura-Montes, Maurice Clerc, P.N. Suganthan, Carlos A. Coello Coello, and K. Deb. Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore, March 2006.

[Lia06b] J.J. Liang and P.N. Suganthan. Dynamic Multi-Swarm Particle Swarm Optimizer with a Novel Constraint-Handling Mechanism. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 316–323, Vancouver, BC, Canada, 2006.

[Lin04] Shu Lin and Daniel J. Costello. *Error Control Coding*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[Liu05] J. Liu and J. Lampinen. A Fuzzy Adaptive Differential Evolution Algorithm. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9(6):448–462, 2005.

[Liu06] Shih-Hsi Liu, Marjan Mernik, and Barrett R. Bryant. Entropy-Driven Exploration and Exploitation in Evolutionary Algorithms. In *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications*, 2006.

[Lob00] Fernando Lobo. *The Parameter-Less Genetic Algorithm: Rational and Automated Parameter Selection for Simplified Genetic Algorithm Operation*. PhD thesis, Universidade Nova de Lisboa, 2000.

[Lob07] Fernando G. Lobo, Cláudio F. Lima, and Zbigniew Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*. Springer, Berlin Heidelberg New York, 2007.

[Løv01] Morten Løvbjerg, Thomas Kiel Rasmussen, and Thiemo Krink. Hybrid Particle Swarm Optimiser with Breeding and Subpopulations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2001.

[Mad02] Nateri K. Madavan. Multiobjective Optimization Using a Pareto Differential Evolution Approach. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2002.

[Men04a] Rui Mendes. *Population Topologies and Their Influence in Particle Swarm Performance*. PhD thesis, University of Minho, 2004.

[Men04b] Rui Mendes, James Kennedy, and José Neves. The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210, 2004.

[Mez04] Efrén Mezura-Montes. *Alternative Techniques to Handle Constraints in Evolutionary Optimization*. PhD thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, Mexico City, 2004.

[Mez05] Efrén Mezura-Montes and Carlos A. Coello Coello. Identifying On-Line Behavior and Some Sources of Difficulty in Two Competitive Approaches for Constrained Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1477–1484, 2005.

[Mez06a] Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos A. Coello Coello. A Comparative Study of Differential Evolution Variants for Global Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 485–492, Seattle, Washington, USA, 2006.

[Mez06b] Efrén Mezura-Montes, Jesús Velázquez-Reyes, and Carlos A. Coello Coello. Modified Differential Evolution for Constrained Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 332–339, Vancouver, BC, Canada, 2006.

[Mic96] Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[Mon01] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons, 2001.

[Moo99] Jacqueline Moore and Richard Chapman. Application Of Particle Swarm To Multiobjective Optimization. Department of Computer Science and Software Engineering, Auburn University, 1999.

[Muñ06] Angel E. Muñoz-Zavala, Arturo Hernández-Aguirre, Enrique R. Villa-Diharce, and Salvador Botello-Rionda. PESO+ for Constrained Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 935–942, Vancouver, BC, Canada, 2006.

[Mye02] Raymond H. Myers and Douglas C. Montgomery. *Response Surface Methodology - Process and Product Optimization Using Designed Experiments*. John Wiley and Sons, 2002.

[Olo08] Olusegun Olorunda and Andries P. Engelbrecht. Measuring Exploration/Exploitation in Particle Swarms using Swarm Diversity. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1128–1134, 2008.

[Ong04] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong. Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 307–331. Springer, 2004.

[Onw04a] Godfrey C. Onwubolu. Differential Evolution for the Flow Shop Scheduling Problem. In Godfrey C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 585–611. Springer-Verlag, Berlin Heidelberg, 2004.

[Onw04b] Godfrey C. Onwubolu. Optimizing CNC Drilling Machine Operations: Traveling Salesman Problem-Differential Evolution Approach. In Godfrey C. Onwubolu and B.V. Babu, editors, *New Optimization Techniques in Engineering*, pages 537–566. Springer-Verlag, Berlin Heidelberg, 2004.

[Pal08] A.K. Palit, K.K. Duganapalli, K. Zielinski, D. Westphal, D. Popovic, W. Anheier, and R. Laur. Evolutionary Computations for Design Optimization and Test Automation in VLSI Circuits. In Ang Yang, Yin Shan, and Lam Thu Bui, editors, *Success in Evolutionary Computation*, volume 92/2008 of *Studies in Computational Intelligence*, pages 277–303. Springer-Verlag, Berlin Heidelberg, 2008.

[Par00] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis. Stretching Technique for Obtaining Global Minimizers Through Particle Swarm Optimization. In *Proceedings of the Workshop on Particle Swarm Optimization*, 2000.

[Par02a] K.E. Parsopoulos and M.N. Vrahatis. Particle Swarm Optimization Method for Constrained Optimization Problems. In P. Sincak, J. Vascak, V. Kvasnicka, and J. Pospichal, editors, *Intelligent Technologies - Theory and Applications: New Trends in Intelligent Technologies*, pages 214–220. IOS Press, 2002.

[Par02b] K.E. Parsopoulos and M.N. Vrahatis. Recent Approaches to Global Optimization Problems Through Particle Swarm Optimization. *Natural Computing*, 1(2-3):235–306, 2002.

[Par04] K.E. Parsopoulos, D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, and M.N. Vrahatis. Vector Evaluated Differential Evolution for Multiobjective Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004.

[Pet01] Dagmar Peters. *Unterstützung des Entwurfsprozesses von Mikrosystemen mit direkten Optimierungsverfahren*. PhD thesis, University of Bremen, 2001.

[Pre06] Mike Preuss. Niching Prospects. In *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications*, 2006.

[Pri99] Kenneth V. Price. An Introduction to Differential Evolution. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 79–108. McGraw-Hill, London, 1999.

[Pri05] Kenneth V. Price, Rainer M. Storn, and Jouni A. Lampinen. *Differential Evolution - A Practical Approach to Global Optimization*. Springer-Verlag Berlin Heidelberg, 2005.

[Pri06] Kenneth V. Price and Jani I. Rönkkönen. Comparing the Uni-Modal Scaling Performance of Global and Local Selection in a Mutation-Only Differential Evolution Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 7387–7394, Vancouver, BC, Canada, 2006.

[Pri08] Kenneth Price and Rainer Storn. http://www.icsi.berkeley.edu/~storn/code.html, website as in August 2008.

[Pul04a] Gregorio Toscano Pulido and Carlos A. Coello Coello. A Constraint-Handling Mechanism for Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1396–1403, Portland, OR, USA, 2004.

[Pul04b] Gregorio Toscano Pulido and Carlos A. Coello Coello. Using Clustering Techniques to Improve the Performance of a Multi-Objective Particle Swarm Optimizer. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 225–237, 2004.

[Qin05] A.K. Qin and P.N. Suganthan. Self-Adaptive Differential Evolution Algorithm for Numerical Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1785–1791, Edinburgh, UK, 2005.

[Rak08] Anna Rakitianskaia and Andries P. Engelbrecht. Cooperative Charged Particle Swarm Optimiser. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 933–939, 2008.

[Rim07] Sunisa Rimcharoen, Daricha Sutivong, and Prabhas Chongstitvatana. Optimal Stopping Time of Compact Genetic Algorithm on Deceptive Problem Using Real Options Analysis. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007.

[Rob05a] Tea Robič. Performance of DEMO on New Test Problems: A Comparison Study. In *Proceedings of the Fourteenth International Electrotechnical and Computer Science Conference ERK*, volume B, pages 121–124, 2005.

[Rob05b] Tea Robič and Bogdan Filipič. DEMO: Differential Evolution for Multiobjective Optimization. In *Proceedings of the Conference on Evolutionary Multiobjective Optimization*, 2005.

[Rog00] T. Rogalsky and R.W. Derksen. Hybridization of Differential Evolution for Aerodynamic Design. In *Proceedings of the 8th Annual Conference of the Computational Fluid Dynamics Society of Canada*, 2000.

[Rön05] Jani Rönkkönen, Saku Kukkonen, and Kenneth V. Price. Real-Parameter Optimization with Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2005.

[Rud04] Olga Rudenko and Marc Schoenauer. A Steady Performance Stopping Criterion for Pareto-based Evolutionary Algorithms. In *Proceedings of the 6th International Multi-Objective Programming and Goal Programming Conference*, Hammamet, Tunisia, 2004.

[Run00] Thomas P. Runarsson and Xin Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, 2000.

[Run06] Thomas Philip Runarsson. Approximate Evolution Strategy using Stochastic Ranking. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2760–2767, 2006.

[Saf04] Martín Safe, Jessica Carballido, Ignacio Ponzoni, and Nélida Brignole. On Stopping Criteria for Genetic Algorithms. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, pages 405–413, 2004.

[Sam07] Nayan R. Samal, Amit Konar, Swagatam Das, and Ajith Abraham. A Closed Loop Stability Analysis and Parameter Selection of the Particle Swarm Optimization Dynamics for Faster Convergence. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007.

[Sat06] Masahiko Sato, Hernán E. Aguirre, and Kiyoshi Tanaka. Effects of δ-Similar Elimination and Controlled Elitism in the NSGA-II Multiobjective Evolutionary Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3980–3987, 2006.

[Sch77] Hans-Paul Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser Verlag, Basel, 1977.

[Sch84] James David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.

[Sch95] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. John Wiley and Sons, 1995.

[Sch02] Frank Schmiedle, Nicole Drechsler, Daniel Große, and Rolf Drechsler. Heuristic Learning based on Genetic Programming. *Genetic Programming and Evolvable Machines*, 3(4):363–388, 2002.

[Sch07] Jörn Schönberger and Herbert Kopfer. On Decision Model Adaptation in Online Optimization of a Transport System. In H.-O. Günther, D.C. Mattfeld, and L. Suhl, editors, *Management logistischer Netzwerke. Entscheidungsunterstützung, Informationssysteme und OR-Tools*, pages 361–381, Berlin Heidelberg, 2007. Springer.

[Shi08] Yuhui Shi and Russell C. Eberhart. Population Diversity of Particle Swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1063–1067, 2008.

[Sim04] T.W. Simpson, A.J. Booker, D. Ghosh, A.A. Giunta, P.N. Koch, and R.-J. Yang. Approximation Methods in Multidisciplinary Analysis and Optimization: A Panel Discussion. *Structural and Multidisciplinary Optimization*, 27(5):302–313, 2004.

[Sin06] Ankur Sinha, Aravind Srinivasan, and Kalyanmoy Deb. A Population-Based, Parent Centric Procedure for Constrained Real-Parameter Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 943–949, 2006.

[Sol05] E. J. Solteiro Pires, P. B. de Moura Oliveira, and J. A. Tenreiro Machado. Multi-Objective MaxiMin Sorting Scheme. In *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, pages 165–175, 2005.

[Sto95] Rainer Storn and Kenneth Price. Differential Evolution - A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, CA, 1995.

[Sto97] Rainer Storn and Kenneth Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[Sto99a] Rainer Storn. Designing Digital Filters with Differential Evolution. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 109–125. McGraw-Hill, London, 1999.

[Sto99b] Rainer Storn. System Design by Constraint Adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 3(1):22–34, 1999.

[Syr95] Michael Syrjakow and Helena Szczerbicka. Combination of Direct Global and Local Optimization Methods. In *Proceedings of the IEEE International Conference on Evolutionary Computing*, pages 326–333, Perth, WA, Australia, 1995.

[Tak06] Tetsuyuki Takahama and Setsuko Sakai. Constrained Optimization by the ϵ Constrained Differential Evolution with Gradient-Based Mutation and Feasible Elites. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 308–315, 2006.

[Tas04] D.K. Tasoulis, N.G. Pavlidis, V.P. Plagianakos, and M.N. Vrahatis. Parallel Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004.

[Tas06] M. Fatih Tasgetiren and P.N. Suganthan. A Multi-Populated Differential Evolution Algorithm for Solving Constrained Optimization Problem. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 340–347, 2006.

[Teo06] Jason Teo. Exploring Dynamic Self-adaptive Populations in Differential Evolution. *Soft Computing*, 10(8):673–686, 2006.

[Tho04] René Thomsen. Multimodal Optimization Using Crowding-Based Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004.

[Tre03] Ioan Cristian Trelea. The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. *Information Processing Letters*, 85(6):317–325, 2003.

[Tuš07] Tea Tušar and Bogdan Filipič. Differential Evolution Versus Genetic Algorithms in Multiobjective Optimization. In *Proceedings of the Conference on Evolutionary Multiobjective Optimization*, 2007.

[Val08] Yamille del Valle, Ganesh Kumar Venayagamoorthy, Salman Mohagheghi, Jean-Carlos Hernandez, and Ronald G. Harley. Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems. *IEEE Transactions on Evolutionary Computation*, 12(2):171–195, 2008.

[Vas97] J.A. Vasconcelos, R.R. Saldanha, L. Krähenbühl, and A. Nicolas. Genetic Algorithm Coupled with a Deterministic Method for Optimization in Electromagnetics. *IEEE Transactions on Magnetics*, 33(2):1860–1863, 1997.

[Vel99] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* PhD thesis, Air Force Institute of Technology, 1999.

[Ver98] S. Verdú. *Multiuser Detection.* Cambridge University Press, Cambridge, UK, 1998.

[Vud07] Shyam Praveen Vudathu, Duane Boning, and Rainer Laur. A Critical Enhancement in the Yield Analysis of Microsystems. In *Proceedings of the 45th IEEE International Reliability Physics Symposium*, Phoenix, AZ, USA, 2007.

[Wan07] Xinwei Wang. Untersuchung zur adaptiven Parameterwahl des Differential Evolution Algorithmus basierend auf Design of Experiments. Projektarbeit (student project), University of Bremen, 2007.

[Wan08a] Yong Wang, Zixing Cai, Yuren Zhou, and Wei Zeng. An Adaptive Tradeoff Model for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 12(1):80–92, 2008.

[Wan08b] Yu-Xuan Wang and Qiao-Liang Xiang. Exploring New Learning Strategies in Differential Evolution Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 204–209, 2008.

[Wei05] Petra Weitkemper, Volker Kühn, and Karl-Dirk Kammeyer. Analysis of Iterative Successive Interference Cancellation in SC-CDMA Systems. In *Fifth International Workshop on Multi-Carrier Spread Spectrum*, 2005.

[Wei06] Petra Weitkemper, Karin Zielinski, Karl-Dirk Kammeyer, and Rainer Laur. Optimization of Interference Cancellation in Coded CDMA Systems by Means of Differential Evolution. In *4th International Symposium on Turbo Codes & Related Topics in connection with the 6th International ITG-Conference on Source and Channel Coding*, 2006.

[Wil05] Daniel N. Wilke. Analysis of the Particle Swarm Optimization Algorithm. Master's thesis, University of Pretoria, 2005.

[Wol97] David H. Wolpert and William G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[Wu05] Xiang Wu, Bayan S. Sharif, and Oliver R. Hinton. An Improved Resource Allocation Scheme for Plane Cover Multiple Access Using Genetic Algorithm. *IEEE Transactions on Evolutionary Computation*, 9(1):74–81, 2005.

[Xie04a] Xiao-Feng Xie, Wen-Jun Zhang, and De-Chun Bi. Handling Equality Constraints by Adaptive Relaxing Rule for Swarm Algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004.

[Xie04b] Xiao-Feng Xie, Wen-Jun Zhang, and De-Chun Bi. Optimizing Semiconductor Devices by Self-Organizing Particle Swarm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2004.

[Xu08] Xing Xu, Yuanxiang Li, Shenlin Fang, Yu Wu, and Feng Wang. A Novel Differential Evolution Scheme Combined with Particle Swarm Intelligence. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1057–1062, 2008.

[Xue03] Feng Xue, Arthur C. Sanderson, and Robert J. Graves. Pareto-based Multi-objective Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2003.

[Xue05] Feng Xue, Arthur C. Sanderson, Piero P. Bonissone, and Robert J. Graves. Fuzzy Logic Controlled Multi-objective Differential Evolution. In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pages 720–725, Reno, NV, USA, 2005.

[Yuc04] Ming Yuchi and Jong-Hwan Kim. Grouping-based Evolutionary Algorithm: Seeking Balance Between Feasible and Infeasible Individuals of Constrained Optimization Problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 280–287, 2004.

[Zah02] Daniela Zaharie. Parameter Adaptation in Diffential Evolution by Controlling the Population Diversity. In *Proceedings of the 4th Workshop on Symbolic and Numeric Algorithms for Scientific Computing*, pages 385–397, Timisoara, Romania, 2002.

[Zah03a] Daniela Zaharie. Control of Population Diversity and Adaptation in Differential Evolution Algorithms. In *Proceedings of Mendel 2003, 9th International Conference on Soft Computing*, 2003.

[Zah03b] Daniela Zaharie and Dana Petcu. Parallel Implementation of Multi-Population Differential Evolution. In *Proceedings of the 2nd Workshop on Concurrent Information Processing and Computing*, 2003.

[Zah04] Daniela Zaharie. A Multipopulation Differential Evolution Algorithm for Multimodal Optimization. In *Proceedings of Mendel 2004, 10th International Conference on Soft Computing*, 2004.

[Zam07] Aleš Zamuda, Janez Brest, Borko Bošković, and Viljem Žumer. Differential Evolution for Multiobjective Optimization with Self Adaptation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3617–3624, Singapore, 2007.

[Zam08] Aleš Zamuda, Janez Brest, Borko Bošković, and Viljem Žumer. Large Scale Global Optimization Using Differential Evolution With Self-Adaptation and Cooperative Co-Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3719–3726, 2008.

[Zha07] Zhi-Hui Zhan, Jing Xiao, Jun Zhang, and Wei-Neng Chen. Adaptive Control of Acceleration Coefficients for Particle Swarm Optimization Based on Clustering Analysis. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007.

[Zie05a] Karin Zielinski, Dagmar Peters, and Rainer Laur. Constrained Multi-Objective Optimization Using Differential Evolution. In *Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 2005.

[Zie05b] Karin Zielinski, Dagmar Peters, and Rainer Laur. Run Time Analysis Regarding Stopping Criteria for Differential Evolution and Particle Swarm Optimization. In *Proceedings of the 1st International Conference on Experiments/Process/System Modelling/Simulation/Optimization*, Athens, Greece, 2005.

[Zie05c] Karin Zielinski, Dagmar Peters, and Rainer Laur. Stopping Criteria for Single-Objective Optimization. In *Proceedings of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 2005.

[Zie06a] Karin Zielinski and Rainer Laur. Constrained Single-Objective Optimization Using Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 927–934, Vancouver, BC, Canada, 2006.

[Zie06b] Karin Zielinski and Rainer Laur. Constrained Single-Objective Optimization Using Particle Swarm Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1550–1557, Vancouver, BC, Canada, 2006.

[Zie06c] Karin Zielinski and Rainer Laur. Parameter Adaptation for Differential Evolution with Design of Experiments. In *Proceedings of the IASTED International Conference on Computational Intelligence*, pages 212–217, San Francisco, USA, 2006.

[Zie06d] Karin Zielinski and Rainer Laur. Stopping Criteria for Constrained Optimization with Particle Swarms. In *Proceedings of the Second International Conference on Bioinspired Optimization Methods and their Applications*, 2006.

[Zie06e] Karin Zielinski, Petra Weitkemper, Rainer Laur, and Karl-Dirk Kammeyer. Examination of Stopping Criteria for Differential Evolution based on a Power Allocation Problem. In *Proceedings of the 10th International Conference on Optimization of Electrical and Electronic Equipment*, volume 3, pages 149–156, Braşov, Romania, 2006.

[Zie06f] Karin Zielinski, Petra Weitkemper, Rainer Laur, and Karl-Dirk Kammeyer. Parameter Study for Differential Evolution Using a Power Allocation Problem Including Interference Cancellation. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 6748–6755, Vancouver, BC, Canada, 2006.

[Zie07a] Karin Zielinski and Rainer Laur. Adaptive Parameter Setting for a Multi-Objective Particle Swarm Optimization Algorithm. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3019–3026, Singapore, 2007.

[Zie07b] Karin Zielinski and Rainer Laur. Differential Evolution with Adaptive Parameter Setting for Multi-Objective Optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3585–3592, Singapore, 2007.

[Zie07c] Karin Zielinski and Rainer Laur. Stopping Criteria for a Constrained Single-Objective Particle Swarm Optimization Algorithm. *Informatica*, 31(1):51–59, 2007.

[Zie07d] Karin Zielinski and Rainer Laur. Variants of Differential Evolution for Multi-Objective Optimization. In *Proceedings of the IEEE Symposium on Computational Intelligence in Multicriteria Decision Making*, pages 91–98, Honolulu, HI, USA, 2007.

[Zie07e] Karin Zielinski, Shyam Praveen Vudathu, and Rainer Laur. Influence of Different Deviations Allowed for Equality Constraints on Particle Swarm Optimization and Differential Evolution. In *Proceedings of the International Workshop on Nature Inspired Cooperative Strategies for Optimization*, Acireale, Italy, 2007.

[Zie08a] Karin Zielinski, Matthias Joost, Rainer Laur, and Bernd Orlik. Choosing Suitable Variants of Differential Evolution and Particle Swarm Optimization for the Optimization of a PI Cascade Control. In *Proceedings of the 11th International Conference on Optimization of Electrical and Electronic Equipment*, 2008.

[Zie08b] Karin Zielinski, Matthias Joost, Rainer Laur, and Bernd Orlik. Comparison of Differential Evolution and Particle Swarm Optimization for the Optimization of a PI Cascade Control. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3119–3126, Hong Kong, 2008.

[Zie08c] Karin Zielinski and Rainer Laur. Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimization. In Uday Chakraborty, editor, *Advances in Differential Evolution*, volume 143/2008 of *Studies in Computational Intelligence*, pages 111–138. Springer-Verlag, Berlin Heidelberg, 2008.

[Zie08d] Karin Zielinski and Rainer Laur. Alternatives for Handling Multi-Objective Optimization Problems with Differential Evolution, accepted for the International Journal of Information Technology and Intelligent Computing, publication expected in 2008.

[Zie08e] Karin Zielinski, Xinwei Wang, and Rainer Laur. Comparison of Adaptive Approaches for Differential Evolution. In *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature*, 2008.

[Zie09] Karin Zielinski, Petra Weitkemper, Rainer Laur, and Karl-Dirk Kammeyer. Optimization of Power Allocation for Interference Cancellation with Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 13(1):128–150, 2009.

[Zit99] Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1999.

[Zit00] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[Zit02] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In K.C. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, pages 95–100. International Center for Numerical Methods in Engineering, 2002.

[Zit03] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.