technische universität
dortmund

Towards Optimal Parameterizations of the *S*-Metric Selection Evolutionary Multi-Objective Algorithms

Simon Wessing

Algorithm Engineering Report
TR09-2-006
Sep. 2009
ISSN 1864-4503

Faculty of Computer Science
Algorithm Engineering (Ls11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

technische universität
dortmund

Diplomarbeit

Towards Optimal Parameterizations of the $\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm

Simon Wessing

24 July 2009

Betreuer:
Prof. Dr. Günter Rudolph
Dipl.-Inf. Nicola Beume

Fakultät für Informatik
Algorithm Engineering (Ls11)
Technische Universität Dortmund
http://ls11-www.cs.uni-dortmund.de

Contents

Chapter 1

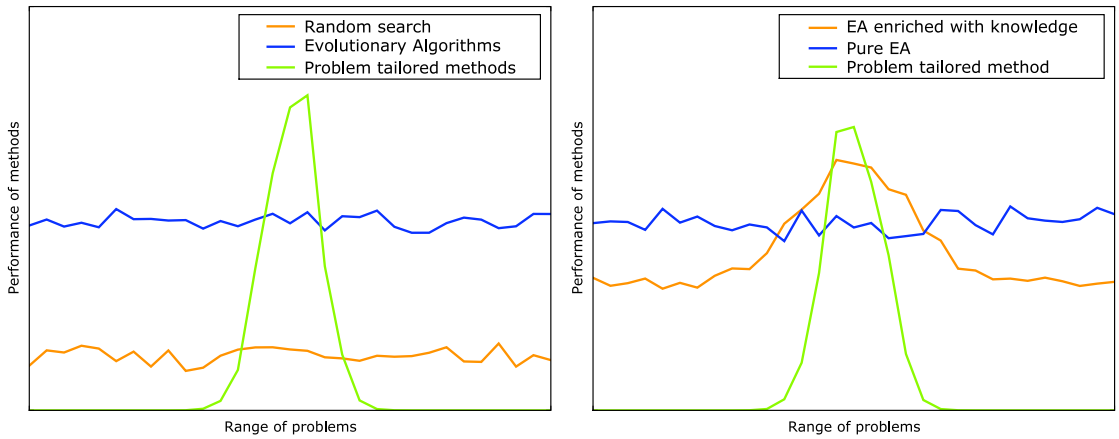
Introduction

1.1 Motivation and Background

Heuristics are often used to approach problems, even if algorithms exist that could find a globally optimal solution. For difficult problems, e. g. NP-hard ones, an exact solution often is not computationally feasible, other problems are just not understood. This is where heuristics come into play: They can provide a tradeoff between runtime or "time to understand" and solution quality. This compromise is achieved by disregarding some of the problem's attributes. In the extreme case the whole problem is treated as a black box, looking only at the problem's results and not how they are generated. But this attitude is not entirely possible. Every algorithm implicitly makes assumptions about problems' properties[1]. These assumptions are inherent in the code structure and can be adjusted through the algorithm's parameters, if it has any. So, we can try to optimize the parameter setting. Note that problem-specific algorithms use not only assumptions, but certain knowledge and are parameterless. In the 1980s, Goldberg [22] attempted to relate different kinds of algorithms to each other (see Figure 1.1(a)). At that time, it was a popular belief that some algorithms would be generally better than others. Wolpert and Macready [59] showed, that under certain circumstances, the average performance over all problems is the same for all algorithms. That work became known as the "No free lunch theorem". So, nowadays we do not make any general statements about algorithms' performance, but always regarding to specific problems. We also see it legitimate to tune black box algorithms to perform well only on a single problem instance. Figure 1.1(b) illustrates this shift in algorithms' understanding.

A category of heuristics that has been applied successfully to many different problems in the last decades are Evolutionary Algorithms (EA). In the case of EAs, an assumption of *strong causality* is generally made [44]. Strong causality means the property of a system that small variations in the cause only provoke small variations in the effect. In this thesis,

[1] Perhaps except for random and exhaustive search.



(a) Goldberg's view in the 1980s

(b) Michalewicz' view in the 1990s [35]

Figure 1.1: The figure shows how the notion of optimization algorithms changed in the course of time.

EAs are applied to multi-objective optimization problems (MOOP). MOOPs contain at least two objective functions that are to be optimized. The objective functions usually are in conflict with each other and are *incommensurable*. This means that there is no intuitive way to tell how much improvement in one objective outweighs a decline in another. Nonetheless, this has often been done in the past by aggregating the objective functions (e. g. as weighted sum) to obtain a single-objective problem [36]. This approach is known as *a priori* method because the tradeoff between the conflicting objectives is defined before the optimization is started. It is problematic, because for a reasonable tradeoff, information on the problem is needed which is usually not available before the optimization. The alternative approach, known as *a posteriori* method [36], solves this problem by delaying the tradeoff decision until optimization has finished. But then, the optimization phase has to produce a set of solution candidates to provide some options to the decision maker. Because EAs are naturally able to maintain a set of solutions throughout the optimization, they are predestinated for this task. We will now see an example from everyday life to get an understanding of multi-objective problems, before we begin with formal definitions.

1.1.1 Example. Many people are confronted with the problem of buying a new car sometime during their lives. As this is a costly investment, most people carefully weigh the pros and cons before making a decision. But every person has individual requirements. While someone needs a lot of transport capacity, another one might prefer comfort and a third person does not want to spend much money. So, there are different optimal cars for different people. The different needs are the decision maker's intuitive objective functions. In this example we will focus on engine power and energy consumption. Both objective functions are interconnected with each other. Power should be maximized and energy con-

sumption should be minimized, but each function cannot be optimized without decreasing quality in the other to some degree. Figure 1.2 shows a plot of energy consumption versus engine power for more than 800 cars. The data was collected by the german automobile association ADAC between 2003 and 2008 [1]. It shows that there is indeed a correlation between power and consumption. The main observation from this example is, regardless of the different engine types, that not all cars can be compared with each other without incorporating additional preference information from the decision maker.

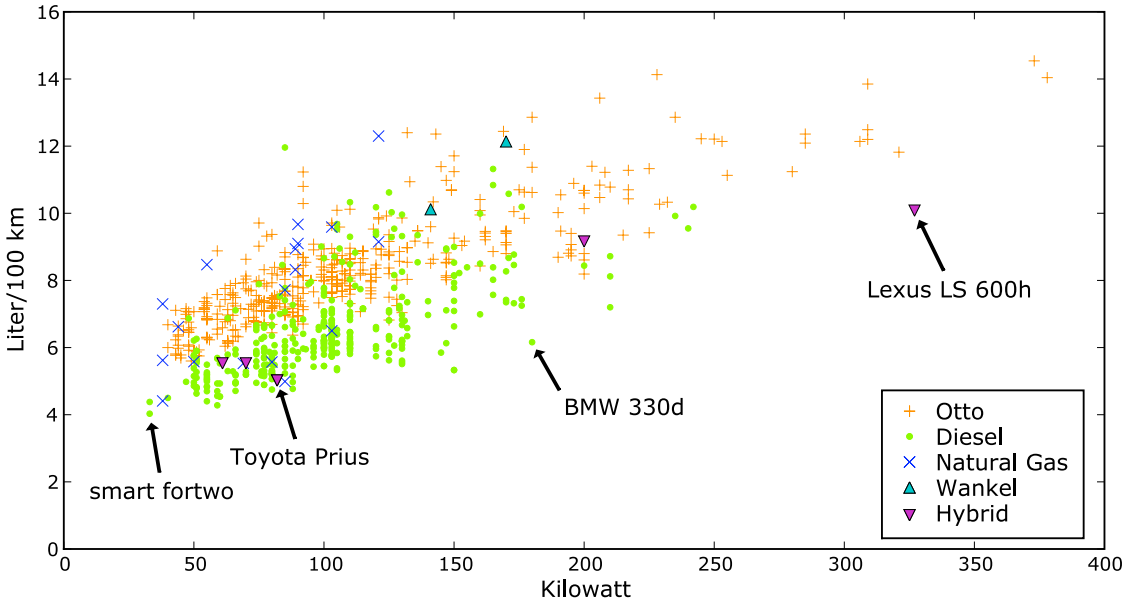


Figure 1.2: Fuel consumption versus engine power. Engine types are distinguished with different markers. Note that engine types using different fuels cannot be compared directly. The cars get better towards the bottom right corner.

1.2 Multi-Objective Optimization

After the general situation was explained in the last section, we are now providing a theoretical background to Multi-Objective Optimization (MOO). Sometimes, the term Multi-criteria Optimization is used synonymously. Introductions to MOO with a focus on Evolutionary Algorithms are given by Deb [9], Coello Coello et al. [7] or Branke et al. [5]. In the following, we will restrict ourselves to subsets of $\mathbb{R}^n$ as search space. So, a vector $\mathbf{x} = (x_1, \ldots, x_n)^T$ of decision variables $x_i \in \mathbb{R}$ is called candidate solution. Let us now define a MOOP formally.

1.2.1 Definition (MOOP). A Multi-Objective Optimization Problem according to [9, page 13] is defined as:

$$\begin{array}{rll} \text{Minimize/Maximize} & f_m(\mathbf{x}), & m = 1, 2, \ldots, M; \\ \text{subject to} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array}$$

In the following we will only consider minimization problems. Note that maximization can be achieved by negating the objective functions because $\min\{f(\mathbf{x})\} = -\max\{-f(\mathbf{x})\}$. In this general form, the problem contains J inequality and K equality constraints, which will also not be considered in the rest of the thesis. Only interval constraints for each of the n variables will be regarded. Every solution $\mathbf{x}$ can be assigned a vector of objective values $(f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}))^T$. The objective values determine in which relation solutions stand to each other.

1.2.2 Definition (Dominance relations). Coello Coello et al. define the following relations between solutions [7, page 244]. A solution $\mathbf{x}$ weakly dominates another solution $\mathbf{y}$ ($\mathbf{x} \preceq \mathbf{y}$) if

$$\forall i \in \{1, \ldots, M\} : f_i(\mathbf{x}) \leq f_i(\mathbf{y}).$$

A solution $\mathbf{x}$ dominates another solution $\mathbf{y}$ ($\mathbf{x} \prec \mathbf{y}$) if

$$\forall i \in \{1, \ldots, M\} : f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \wedge \exists j \in \{1, \ldots, M\} : f_j(\mathbf{x}) < f_j(\mathbf{y}).$$

The dominance relation $\prec$ is a strict partial order [9, page 29]. A solution $\mathbf{x}$ strongly dominates another solution $\mathbf{y}$ ($\mathbf{x} \prec\prec \mathbf{y}$) if

$$\forall i \in \{1, \ldots, M\} : f_i(\mathbf{x}) < f_i(\mathbf{y}).$$

We write $\mathbf{x} \parallel \mathbf{y}$ if $\mathbf{x}$ and $\mathbf{y}$ are incomparable ($\neg(\mathbf{x} \preceq \mathbf{y}) \wedge \neg(\mathbf{y} \preceq \mathbf{x})$).

1.2.3 Definition (Non-domination and Pareto-optimality). A solution $\mathbf{x} \in A$ is said to be non-dominated if there is no $\mathbf{x}' \in A$ with $\mathbf{x}' \prec \mathbf{x}$. If A is the entire search space, $\mathbf{x}$ is Pareto-optimal [9, page 31].

1.2.4 Definition (Pareto-set and Pareto-front). The set of Pareto-optimal solutions

$$\mathcal{P} := \{\mathbf{x} \in \Omega \mid \nexists \mathbf{x}' \in \Omega : \mathbf{x}' \prec \mathbf{x}\}$$

is called the Pareto-optimal set or Pareto-set for short [7, page 11]. The Pareto-front $\mathcal{PF}$ of a Pareto-optimal set $\mathcal{P}$ is defined as

$$\mathcal{PF} := \{u = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T \mid \mathbf{x} \in \mathcal{P}\}.$$

In continuous search spaces, $\mathcal{P}$ is usually innumerable. In practice, the goal of multi-objective EAs is to find a finite set of solutions that provides a good approximation of the unknown $\mathcal{P}$. So, the corresponding points in objective space should be as near as possible to the Pareto-front and evenly distributed over the whole Pareto-front [9, page 24]. To illustrate all these definitions, we view another example.

1.2.5 Example. Consider a problem with one decision variable $x \in \mathbb{R}$ and two objective functions $f_1(x) = x^2$, $f_2(x) = (x-2)^2$. The Pareto-optimal set of this problem is $\mathcal{P} = \{x | x \in [0, 2]\}$. Figure 1.3 shows how the decision space maps onto the objective space.

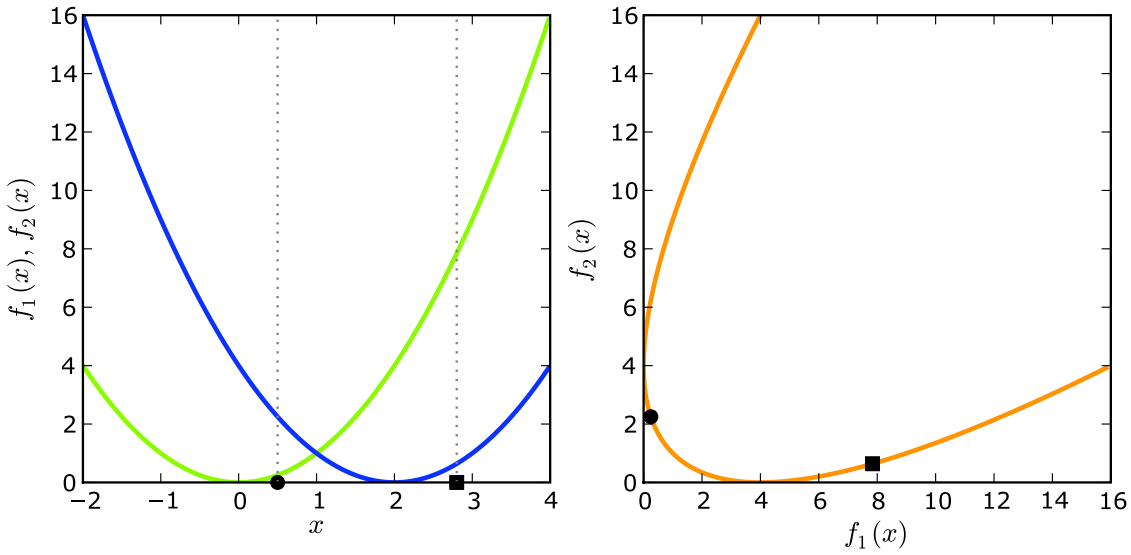


Figure 1.3: The one dimensional decision space (left) is mapped to a two dimensional objective space (right). The circle is Pareto-optimal (because it is $\in [0, 2]$) while the square is not. Nonetheless, the two points are incomparable. Note that on the left, plots of the objective functions are added for clarity. The decision space only corresponds to the x-axis. The Pareto-front is the section of the orange curve that runs in $[0, 4]$ in both dimensions.

1.3 Thesis Structure

Chapter 2 gives an introduction to Evolutionary Algorithms. The main focus there is on the $\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) by Emmerich et al. [17], a state-of-the-art algorithm for multi-objective optimization, which is the subject of this work. The algorithm is described in detail in Section 2.3. The population size is the only parameter the user needs to specify for it. Depending on which variation operators are chosen for the optimization task, additional parameters must be adjusted. Altogether, the thesis deals with the tuning of these parameters. The first goal is to investigate their impact on the optimization performance. In a second step, constants that were previously

not considered as parameters are examined as well. The variation concepts (that can be used in any EA) are also covered in Chapter 2. The parameter optimization is carried out with a method called Sequential Parameter Optimization (SPO), which was proposed by Bartz-Beielstein [2]. Its main idea is to treat optimizer runs as experiments, using methods from Design of Experiments (DoE) [37] and Design and Analysis of Computer Experiments (DACE) [47, 48]. Chapter 3 explains SPO and how it can be used on multi-objective problems. The experiments were carried out on a set of test problems. Chapter 4 contains all the experimental results obtained for this work. Chapter 5 then summarizes the results and gives a short outlook to possible future research. The work also contains two appendices containing additional information. Appendix A contains definitions of all the used test problems. The task for this thesis also involves the reimplementation of the used algorithms and problems in Python [57]. Appendix B documents this work.

Chapter 2

Evolutionary Algorithms

The purpose of this chapter is to give an introduction to the field of EAs, but especially to multi-objective EAs, because this work exclusively deals with MOOPs. The field of MOO was already dealt with in Section 1.2. So, after the origins and principles of evolutionary computation are explained briefly in the next section, the focus is shifted to EMOAs in Section 2.2. The SMS-EMOA, as subject of all experiments in Chapter 4, is covered extensively. Finally, two algorithm-independent variation concepts are introduced.

2.1 Principles of EAs

Evolutionary Algorithms are biologically inspired, heuristic optimization algorithms. EAs are a subarea of Computational Intelligence [18]. They maintain a population of individuals that are subject to the principle of natural selection. Candidate solutions are regarded as the individuals' genomes. The objective values are used to compute a fitness ranking, which forms the basis of selection. EAs have several independent origins: On the one hand there are evolution strategies by Schwefel [50, 51] and Rechenberg [44], on the other hand genetic algorithms by Holland [24]. Other early contributors were Fogel [20] with evolutionary programming and Koza [31] with genetic programming. Induced by different areas of application, each group developed slightly different conceptions. Nowadays, the distinction has become blurred and everything is handled under the generic term evolutionary algorithms. A general introduction to the field is given by Eiben and Smith [16] or with a focus on evolution strategies by Beyer and Schwefel [4].

Figure 2.1 shows an outline of an EA. It begins with the initialization, which contains especially the specification of the initial population. Often, the individuals are generated uniformly random distributed. The individuals are immediately evaluated. The algorithm then enters its optimization loop, beginning with parent selection. The selected individuals produce offspring by means of recombination and mutation. Recombination creates children by combining genes from at least two parents, while mutation varies just a single individual.

This means that like in natural evolution, there is an undirected and random variation involved in the process. The population is evaluated again afterwards. The obtained fitness values are then used to carry out a survivor selection. Depending on the termination criterion, the process continues with parent selection or terminates. One iteration in this loop is considered a generation.

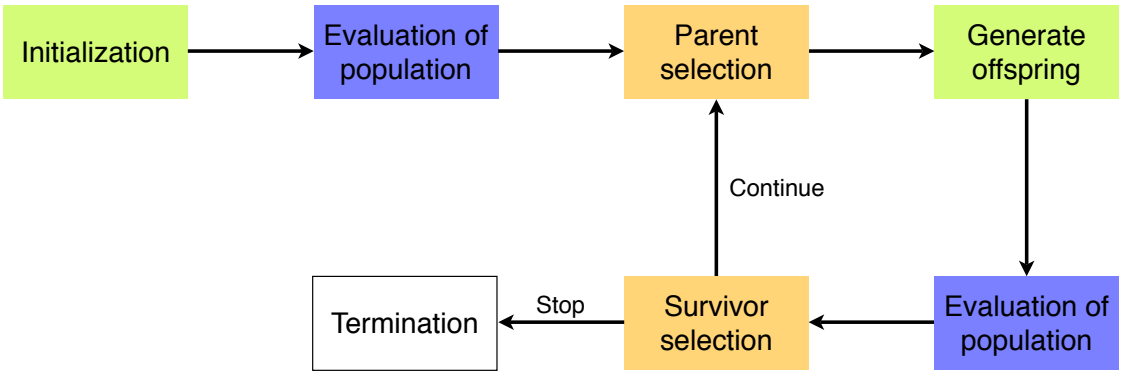


Figure 2.1: Abstract flow chart of an EA. Similar stages carry the same colors.

EAs are often descibed by their selection approach. In the following, μ depicts the population size and λ the number of offspring in each generation. When $(\mu + \lambda)$ selection is used, the μ best individuals in each generation are chosen from the parents and the offspring, while in (μ, λ) the survivors are only chosen from the offspring. In this case, λ must be greater than μ. The survivors form the population in the next generation. Assigning each individual a maximal lifetime $0 \leq \kappa \leq \infty$ gives a generalization of both approaches, leading to a (μ, κ, λ)-algorithm. Next, special features of EAs in multi-objective optimization are outlined.

2.2 Evolutionary Multi-criteria Optimization

The field of Evolutionary Multi-criteria Optimization (EMO) is still quite young. Schaffer's Vector Evaluated Genetic Algorithm (VEGA) [49] from 1984 is considered the first EMOA, but it took approximately 10 years more until research again made some significant progress. An important discovery in this context was non-dominated sorting (NDS), proposed by Goldberg [22] in 1989, an algorithm to partition the population into Pareto-set approximations. A set of solutions A is called Pareto-set approximation if $\forall \mathbf{x}, \mathbf{y} \in A : \mathbf{x} \parallel \mathbf{y}$. NDS works as follows: The set of minimal elements is computed from a population. This set is called the first non-dominated front. All individuals in it are assigned rank 1. The process is then repeated with the remaining individuals until each individual is assigned a front and thus a rank. This algorithm has a worst case runtime of $O(n^3)$. Theoretically faster algorithms are available, e. g. in [11].

The NSGA by Srinivas and Deb [55] (1994) was the first algorithm that used non-dominated sorting to evaluate individuals' fitness. Its successor, the Elitist Non-Dominated Sorting Genetic Algorithm (NSGA-2), is still in use today. Its main improvement over NSGA is having an *elitist* selection, which means that it guarantees to preserve the best μ individuals (according to the dominance relation in the multi-objective case) in every generation [4]. Some studies show that elitism is generally favorable in multi-objective optimization [45, 66] and at least, all state-of-the-art EMOAs nowadays are elitist.

Table 2.1 shows the main differences between NSGA-2 and SMS-EMOA. Both algorithms have drawbacks. The NSGA-2 is known to perform relatively bad on high dimensional objective spaces [58, 40], compared to SMS-EMOA and other algorithms. The SMS-EMOA, on the other hand, has a high runtime. To alleviate this burden, its selection pressure is low.

Table 2.1: Comparison of NSGA-2 and SMS-EMOA

	NSGA-2	SMS-EMOA
Survivor Selection	$(\mu + \mu)$	$(\mu + 1)$
Fitness criteria	1. Non-dominated sorting rank 2. Crowding distance	1. Non-dominated sorting rank 2. Hypervolume contribution

After non-dominated sorting, the NSGA-2 sorts each front with a niching criterion called crowding distance that helps maintaining diversity in the objective space. "The crowding distance of the i-th solution in its front [...] is the average side-length of the cuboid" spanned by its nearest neighbors [11]. The criterion evaluates individuals the weaker the more crowded their neighborhood is. Boundary points are assigned maximal fitness. Figure 2.2 illustrates the crowding distance in an example front. The runtime of crowding distance computation is $O(Mn \log n)$ for M objective functions and n individuals [11]. Instead of the crowding distance, the SMS-EMOA uses a hypervolume-based fitness criterion, which is described in the next section.

The SMS-EMOA also has things in common with other EAs than the NSGA-2. As the name indicates, the Indicator-based Evolutionary Algorithm (IBEA) by Zitzler and Künzli [65] also contains a selection that is based on quality indicators. In their case, the indicator is used to make pairwise comparisons of all individuals with each other. Zitzler and Künzli even modularized their algorithm, so that the user can employ his own indicator, expressing his preferences. The IBEA uses a $(\mu + \mu)$ survivor selection. NSGA-2 and IBEA both use binary tournament selection as parent selection, in contrast to the SMS-EMOA, which chooses parents uniformly random distributed. In binary tournament selection, two individuals are randomly selected and the better one is carried over into the mating pool.

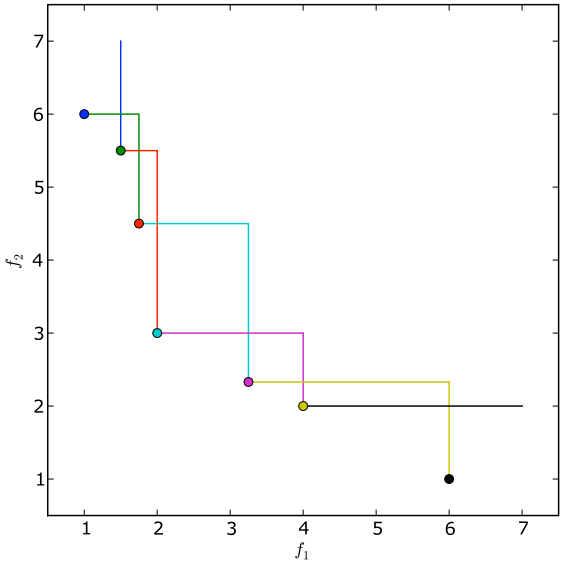


Figure 2.2: The crowding distance used in the NSGA-2 gives an estimate of the distance of each solutions neighbors. The extreme solutions are assigned a distance of ∞ and thus highest priority.

The mating pool just stores the individuals scheduled for reproduction. This procedure is repeated until the mating pool reaches the desired size and reproduction can begin.

2.3 $\mathcal{S}$-Metric Selection Evolutionary Multi-Objective Algorithm

The SMS-EMOA [17] uses a hypervolume-based selection criterion, called $\mathcal{S}$-metric selection, where the NSGA-2 uses the crowding distance measure. To understand how this selection works, we first have to introduce the $\mathcal{S}$-metric performance measure, also called hypervolume indicator.

2.3.1 Definition ($\mathcal{S}$-metric). The $\mathcal{S}$-metric of a set M can be defined as Lebesgue measure Λ of the union of hypercubes a_i defined by a non-dominated point m_i and a reference point x_{ref} [7, 17]:

$$\mathcal{S}(M) := \Lambda\left(\left\{\bigcup_i a_i \mid m_i \in M\right\}\right) = \Lambda\left(\bigcup_{m \in M} \{x \mid m \preceq x \preceq x_{\text{ref}}\}\right).$$

It was first introduced by Zitzler and Thiele [66] as "the size of the objective value space which is covered by a set of nondominated solutions". So, the quality of a population is determined by the hypervolume that the individuals in the population together dominate with regard to x_{ref} (see Figure 2.3). The SMS-EMOA calculates for each individual how much hypervolume it exclusively dominates. This information is then used in the selection

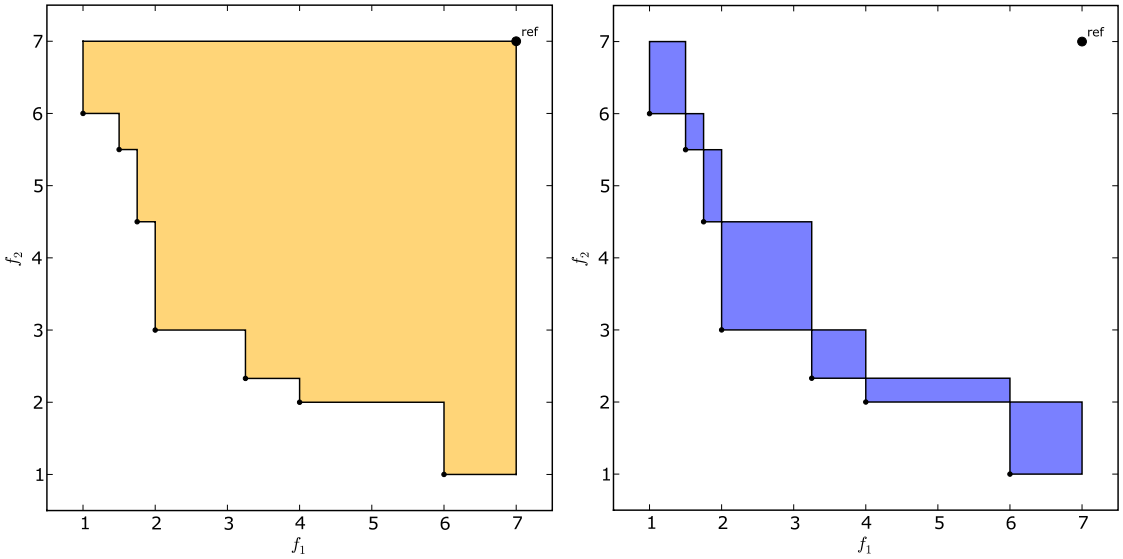


Figure 2.3: In this example we assume a two-dimensional problem where both objective functions are to be minimized. The reference point is set to $(7, 7)^T$. On the left, you see the dominated hypervolume of a possible non-dominated front. Obviously, dominated individuals would contribute nothing to the measure. The same front is plotted on the right, but now only the hypervolume that is dominated exclusively by each point is marked.

process, when the individual with the minimal contribution is removed. This approach guarantees that the hypervolume of the population is monotonically increasing [17]. Thanks to this feature, it is unneccessary to maintain an archive parallel to the population, like some other algorithms, e. g. SPEA-2 by Zitzler et al. [62], do. Unfortunately, the calculation of the $\mathcal{S}$-metric is computationally expensive. The fastest algorithm known to date has a runtime of $O(n \log n + n^{d/2})$, as shown by Beume and Rudolph [3]. Bringmann and Friedrich [6] show that even deciding whether a solution has the least hypervolume contribution is NP-hard. Nonetheless, the SMS-EMOA employs this indicator to benefit from its favorable properties, namely the ability to reward convergence to the Pareto-front as well as distribution along the front at the same time [29]. Additionally, the $\mathcal{S}$-metric favors individuals that achieve a balanced compromise between the objective functions [66].

Algorithm 2.1 shows the outline of the SMS-EMOA. In the selection's first step, the population is divided into non-dominated fronts $\{R_1, \ldots, R_I\}$. The function fast-nondominated-sort(Q) is described in [11]. Then a reference point $\mathbf{p}$ is constructed from the population Q as shown in Algorithm 2.2. It only affects the hypervolume contribution of the extremal points. The reference point's coordinates are chosen as $w_j + 1$, where w_j is the worst objective value in the population in the jth dimension. Due to the fact that the SMS-EMOA uses a $(\mu + 1)$ selection scheme, it only needs to sort the last front by hypervolume contribution

Algorithm 2.1 SMS-EMOA

1: $P_0 \leftarrow \text{init}()$ // initialize random start population of μ individuals
2: $t \leftarrow 0$
3: **repeat**
4: $q_{t+1} \leftarrow \text{generate}(P_t)$ // generate one offspring by variation operators
5: $Q \leftarrow P_t \cup \{q_{t+1}\}$
6: $\{R_1, \ldots, R_I\} \leftarrow$ fast-nondominated-sort(Q) // all I non-dominated fronts of Q
7: $\mathbf{p} \leftarrow \text{constructReferencePoint}(Q)$
8: $r \leftarrow \text{argmin}_{s \in R_I}\{\Delta_{\mathcal{S}}(s, R_I, \mathbf{p})\}$ // detect element of R_I with lowest $\Delta_{\mathcal{S}}(s, R_I, \mathbf{p})$
9: $P_{t+1} \leftarrow Q \setminus \{r\}$ // eliminate detected element
10: $t \leftarrow t + 1$
11: **until** stop criterion reached

to find the least valuable individual. To do this, the selection calculates the exclusively dominated hypervolume $\Delta_{\mathcal{S}}(s, R_I, \mathbf{p})$ for each individual s in the last front R_I.

Algorithm 2.2 constructReferencePoint(Q)

1: $\mathbf{o} \leftarrow (1, 1, \ldots, 1)^T$ // set the offset to 1 in every dimension
2: $\mathbf{w} \leftarrow \text{first}(Q)$ // set the vector of worst objective values to the first point of Q
3: **for all** $q_i \in Q$ **do**
4: **for** $j = 1$ to $|q_i|$ **do** // for every dimension
5: $w_j \leftarrow \max\{w_j, q_{ij}\}$ // set w_j to the worse value
6: **end for**
7: **end for**
8: **return** $\mathbf{w} + \mathbf{o}$ // return the reference point

2.4 Variation

Generally, variation operators are problem-dependent and thus belong to the individual. So, they can be used with any algorithm. This thesis deals with two different variation concepts for $\mathbb{R}^n$ as search space. Both are incorporating information from the current population into the variation process to obtain an adaptive behavior.

2.4.1 Simulated Binary Crossover and Polynomial Mutation

Simulated Binary Crossover (SBX) was devised by Deb et al. [10] to carry over the behavior of single-point crossover in binary search spaces to real valued search spaces. In single-point crossover, the two parents' bitstrings are cut at the same (random) position. One of the resulting two pairs is then swapped between the individuals. SBX always creates two

children from two parents. Polynomial Mutation utilizes the same probability distribution to vary a single individual. These two variation operators together will simply be called *SBX variation* in the remainder of the thesis. Figure 2.4 shows the effects of the respective operators in an example situation.

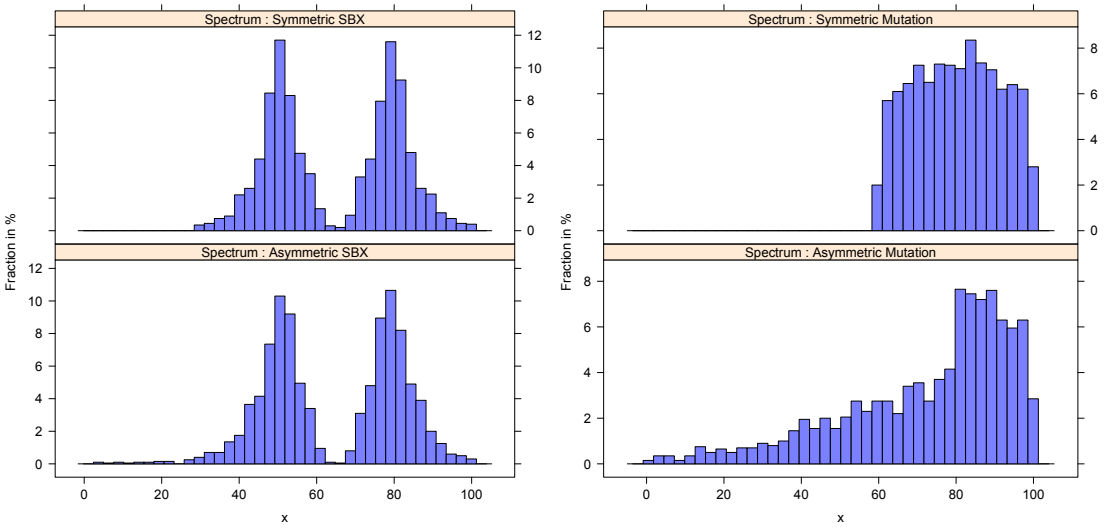


Figure 2.4: Each subfigure shows a sample of 2000 variations. The figure on the left shows the empirical probability distribution for recombination of parent values $x_1 = 50$ and $x_2 = 80$. On the right, mutations of parent value $x_2 = 80$ are shown. The boundaries of $x_{\min} = 0$ and $x_{\max} = 100$ differently affect the symmetric and asymmetric variants.

SBX variation contains several parameters that may be adjusted by the user. The variance of the distributions is controlled by the parameters $\eta_c, \eta_m \in \mathbb{R}_+$. p_c and p_m describe the probability to vary a single decision variable. This means that the impact of the η values is directly dependent on the probabilities. Finally, p_s is the probability to choose the symmetric variant of variation. It controls how the variation deals with a problem's interval constraints $x_{\min}$ and $x_{\max}$. The probability p_s is included in the experiments because the variation's behavior was undocumentedly changed from symmetric to asymmetric in the authors' original implementation.

In the following we show how to create two childrens' decision variables $x_i^{(1,t+1)}$ and $x_i^{(2,t+1)}$ from two parent variables $x_i^{(1,t)}, x_i^{(2,t)} \in \mathbb{R}$ with SBX in the symmetric case. First, we define a spread factor β_i as the difference of the childrens' values divided by the difference of the parents' values:

$$\beta_i := \left| \frac{x_i^{(2,t+1)} - x_i^{(1,t+1)}}{x_i^{(2,t)} - x_i^{(1,t)}} \right|.$$

Our goal is to distribute the spread factor β_i according to the following polynomial distribution:

$$P(\beta_i) := \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c}, & \text{if } \beta_i < 1 \\ 0.5(\eta_c + 1)\frac{1}{\beta_i^{\eta_c+2}}, & \text{else.} \end{cases}$$

Using a uniformly random distributed number $u \in [0, 1[$, we can obtain the ordinate β_{q_i} from the probability distribution, so that the integral between zero and β_{q_i} is equal to u:

$$\beta_{q_i} = \begin{cases} (2u)^{\frac{1}{\eta_c+1}}, & \text{if } u \leq 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta_c+1}}, & \text{else.} \end{cases}$$

The childrens' decision variables at position i are then calculated as:

$$\begin{aligned} x_i^{(1,t+1)} &= 0.5\left[(1 + \beta_{q_i})x_i^{(1,t)} + (1 - \beta_{q_i})x_i^{(2,t)}\right] \\ x_i^{(2,t+1)} &= 0.5\left[(1 + \beta_{q_i})x_i^{(1,t)} + (1 + \beta_{q_i})x_i^{(2,t)}\right] \end{aligned}$$

Deb and Beyer [12] showed that SBX' behavior is *self-adaptive*, i. e. it is able to adapt its variation strength to an actual problem situation, without employing a fixed adaption rule. Instead, the adaptive behavior emerges as a byproduct of the optimization. This is a desireable feature, because it reduces an EA's dependence on fine-tuned parameters. Self-adaptation was originally introduced as a feature of gaussian mutation in evolution strategies [4], which is not dealt with in this work.

2.4.2 Differential Evolution

Another variation method that copes well with $(\mu + 1)$ selection is Differential Evolution (DE), developed by Storn and Price [56]. In DE, variation is carried out by adding the difference vector of two or more randomly chosen individuals to another one. The authors call this process mutation. The outcome is then further perturbed by a discrete recombination with another individual.

The classic DE algorithm also contains a special selection scheme, which lets the offspring only compete with its parent (the one recombination was done with), achieving a crowding effect. However, only the DE variation is picked here to be used with the SMS-EMOA. Algorithm 2.3 shows the pseudocode for DE variation. It takes a parent $\mathbf{x}$ and a vector $\mathbf{p}$ of other individuals as input. The variation contains three user-adjustable parameters that remain fixed during the optimization. *DIFFS* defines the number of difference vectors that are added to an individual. F is a scaling factor to vary the length of the difference vector. Finally, *CR* controls the crossover rate, similar to p_c in SBX.

The DE variant used here is the plain version that was originally proposed by the authors. Although a variety of more sophisticated versions, being self-adaptive [27, 60] or

Algorithm 2.3 deVariation(**x**, **p**)

```
1: j ← randomChoice({1, ..., n}) // uniformly distributed
2: for i = 1 to n do
3:   if i == j or random() < CR then // random(): uniformly distributed, ∈ [0, 1[
4:     y_i ← p_{1,i}
5:     for k = 1 to DIFFS do
6:       ℓ ← 2k // index of next parent
7:       y_i ← y_i + F(p_{ℓ,i} − p_{ℓ+1,i})
8:     end for
9:   else
10:    y_i ← x_i // discrete recombination part
11:  end if
12: end for
13: return y
```

at least adaptive [61], exists today, they were all disregarded in the experiments for two reasons. First, unknown effects should be ruled out from the experiments, because only few experience with SPO for multi-objective algorithms is available. So, the setup should be held as simple as possible, decreasing the risk of failure. Second, SBX is used in its original version, too, although a variant with additional parameter control exists from Deb et al. [13]. However, this variant seems to be seldom used.

Chapter 3

Sequential Parameter Optimization

After we have now learned about EAs, we are going to provide a basis for their optimization. So, this chapter first introduces some vocabulary that is needed for describing computer experiments. The approach of SPO is then generally explained and the special requirements of multi-objective optimization are addressed. This encompasses the definition of quality indicators and their usage on test problems. Finally, an example is given how to carry out an experiment with SPO.

3.1 Basics of Computer Experiments

Experiments in this work either follow standard procedures from Design of Experiments (DoE) or Design and Analysis of Computer Experiments (DACE). DoE is the classic approach, that has been originally developed in agricultural applications by Fisher [19] in the 1930s. DACE is of course considerably younger and tries to accomodate the peculiarities of computer experiments, e. g. determinism of computer programs.

We will now describe the most important terms. The experiment's input variables are called factors. A factor's value is called level. The region of interest (ROI) specifies the lower and upper bounds of each factor. An experimental design is a procedure to choose sample points from the region of interest. In this work we will be using two kinds of experimental designs: The full factorial design and the latin hypercube design. The former has a fixed number of 2^k sample points for k factors with two levels and places the points on the boundaries of the ROI. Latin hypercube designs are space-filling designs. They divide the ROI into a grid and place the points randomly distributed in the interior of the ROI, so that every row and every column contains at least one point. Both designs have an important advantage compared to a one-factor-at-a-time design: They can detect interaction effects between factors. Figure 3.1 illustrates the difference between them. The factorial design is usually used in DoE and a linear model is assumed for the data. DoE also works well with qualitative factors. DACE uses space-filling designs and gaussian random function models.

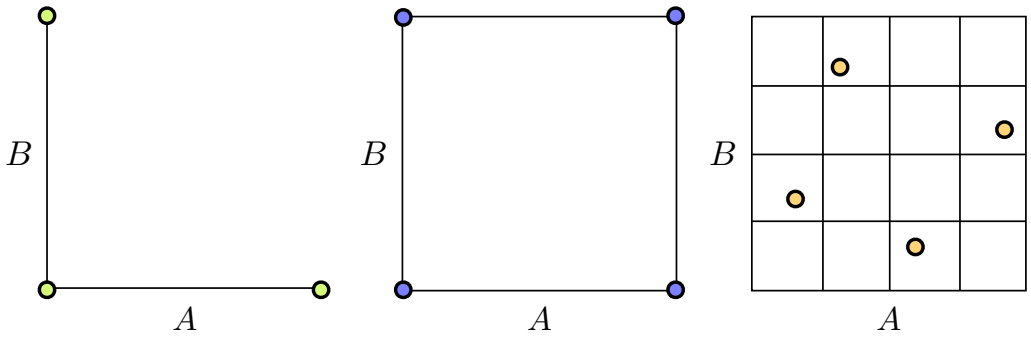


Figure 3.1: Different experimental designs for $k = 2$ factors. On the left is a one-factor-at-a-time design with $k + 1$ samples and in the middle a full factorial design with 2^k samples. The number of sample points is predefined by both designs, while the latin hypercube design on the right can have an arbitrary number of samples.

Santner et al. [48] cover this topic in detail. The model's purpose is to predict unkown values of the stochastic process that is investigated in the experiment. In the analysis of experiments' results, one distinguishes main and interaction effects. Bartz-Beielstein [2, page 48] describes the interpretation of effects as follows:

> The intuitive definition of a main effect of a factor A is the change in the response produced by the change in the level of A averaged over the levels of the other factors. The average difference between the effect of A at the high level of B and the effect of A at the low level of B is called the interaction effect AB of factor A and factor B.

3.2 General Approach of SPO

SPO was developed by Bartz-Beielstein [2] for parameter tuning of stochastic optimization algorithms. It was the first tuning procedure that was specifically targeted at this task. By now, another procedure, called Relevance Estimation and Value Calibration (REVAC), has been developed by Nannen and Eiben [39, 38]. Of course, parameter tuning is again an optimization problem that can also be tackled by arbitrary optimization algorithms. The aforementioned special approaches additionally try to provide insight into the parameters' effects, which helps on deciding if any improvement is scientifically meaningful. Recently, Smit and Eiben began comparing the different tuning approaches empirically [53].

SPO treats optimizer runs as experiments. The algorithm's parameters that need to be set before the optimization's start are considered as the experiment's design variables, also called factors. The population size μ of EAs is an example for such a parameter. Parameters that are adapted during the algorithm run are called endogenous parameters. Those that remain constant are called exogenous. The experimental design consists of the algorithm

design and the problem design. An experiment's factors that are problem specific belong to the problem design while algorithm specific factors belong to the algorithm design. SPO deals with improving an algorithm design's parameter set for one specific problem design.

The workflow can be divided into three phases: Experiment construction, parameter optimization and reporting. The workflow is sketched in Figure 3.2. In the first phase the preexperimental planning (S-1) has to be done. This could be for example a preliminary study to determine which parameters should be considered and what the experiment's goal is. A scientific claim (S-2) and a statistical hypothesis (S-3) must be specified next. Such a hypothesis must be falsifiable by a statistical test. An introduction to statistical hypothesis testing can be found in [33]. Then the experimental design (S-4) has to be chosen. This comprises the optimization problem, resource constraints, an initialization method, a termination method, an algorithm with parameters, an initial experimental design and a performance measure. In the optimization phase the experiment is carried out (S-5). Its results are used to learn a prediction model (S-6). Usually, DACE Kriging by Lophaven et al. [34] is used for this purpose, but other models, e. g. regression trees, are also possible. The Kriging model is also able to predict the mean squared error (MSE) on an untried point in the region of interest. We have to mention that DACE Kriging was originally intended to model deterministic functions. SPO accounts for this requirement by sampling each design point multiple times and using the mean value. The model is evaluated and visualized in (S-7). Optimization is carried out in (S-8). In (S-9) the termination criterion is tested. If it is not fulfilled, a new design point is generated in (S-10), whereupon the process continues at (S-5). Else, the reporting begins in (S-11). Bartz-Beielstein states that "an experiment is called sequential if the experimental conduct at any stage depends on the results obtained so far" [2, page 79]. At this point it becomes clear why the name Sequential Parameter Optimization was chosen. The third phase contains the rejection or acceptance of the statistical hypothesis (S-11) and the final discussion and interpretation of the results (S-12).

3.3 Quality Assessment

To evaluate an optimizer run, a performance measure has to be selected. In single-criterion optimization, this is usually the mean best fitness value. In the multi-objective case it gets more complicated, because we need a mapping from the population of solutions to a scalar value. One possibility of doing this is with the hypervolume indicator, which was already described in Section 2.3. So, we set $I_H(A) := -\mathcal{S}(A)$ for a population A, to get a minimizing indicator. Besides I_H, a few other quality indicators exist [64] that provide the mentioned mapping. First, we define a few prerequisites. A unary quality indicator is a function $I : \Psi \to \mathbb{R}$ that assigns each Pareto-set approximation a real number. A binary indicator $I : \Psi \times \Psi \to \mathbb{R}$ does the same for pairs of approximation sets. Here, Ψ denotes the

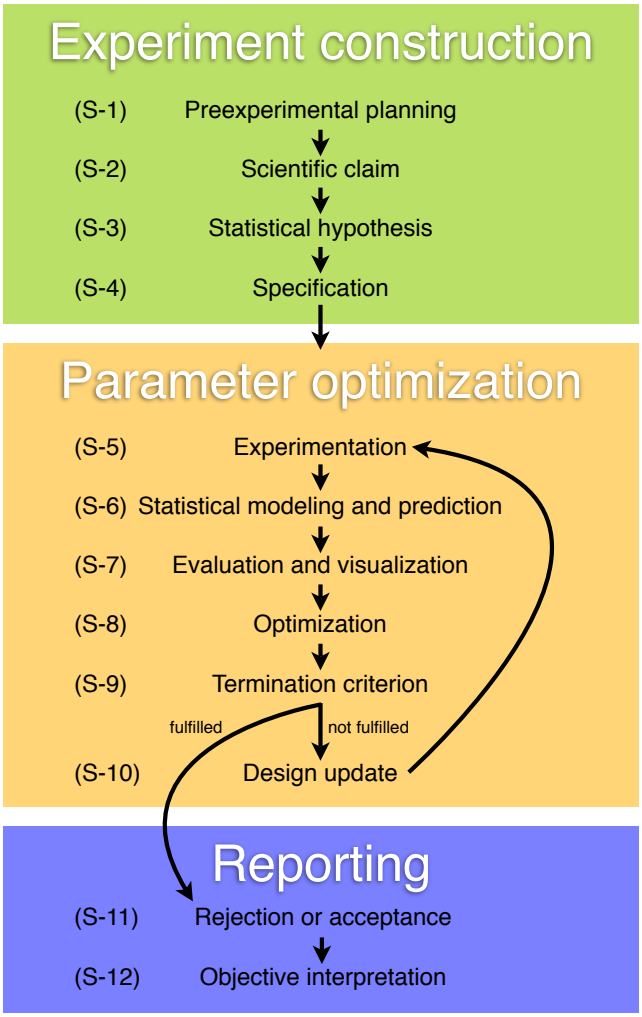


Figure 3.2: Workflow of SPO. You can see the three main phases experiment construction, parameter optimization and reporting. The "Sequential" in the name SPO comes from the loop in the optimization phase.

set of all Pareto-set approximations in the search space. In this thesis, we will concentrate on monotonic indicators, i. e. indicators I that satisfy

$$\forall A, B \in \Psi : A \preceq B \Rightarrow I(A) \leq I(B). \tag{3.1}$$

In Equation 3.1, the weak dominance relation $\preceq$ has been generalized to operate on sets as follows:

$$A \preceq B \Leftrightarrow \forall \mathbf{y} \in B : \exists \mathbf{x} \in A : \mathbf{x} \preceq \mathbf{y}.$$

To date, I_H is the only indicator that is known to be even strictly monotonic. Zitzler et al. [64] deal with the topic in greater detail. We choose two more indicators, that Knowles et al. focus on in [30], for a comparison.

3.3.1 Definition (ϵ_+ indicator). The ϵ_+ indicator is a special case of the ϵ indicator proposed by Zitzler et al. [67, 30]. The binary ϵ_+ indicator $I^B_{\epsilon+}$ is defined as

$$I^B_{\epsilon+}(A, B) = \inf_{\epsilon \in \mathbb{R}} \{\forall \mathbf{x}_2 \in B : \exists \mathbf{x}_1 \in A : \mathbf{x}_1 \preceq_{\epsilon+} \mathbf{x}_2\}$$

using the additive ϵ-dominance relation $\preceq_{\epsilon+}$:

$$\mathbf{x}_1 \preceq_{\epsilon+} \mathbf{x}_2 \iff \forall i \in \{1, \ldots, n\} : f_i(\mathbf{x}_1) \leq \epsilon + f_i(\mathbf{x}_2).$$

The indicator measures the minimal distance by which A must be shifted so that it weakly dominates B. The unary ϵ_+ indicator $I_{\epsilon+}(A)$ can be obtained by simply replacing B with a fixed reference set R.

3.3.2 Definition ($R2$ indicator). The $R2$ indicator is a special case of the R indicator proposed by Hansen and Jaszkiewicz [23, 30]. It uses utility functions $u : \mathbb{R}^k \to \mathbb{R}$ to map k-dimensional objective vectors onto a scalar value. The binary $R2$ indicator I^B_{R2} is defined as

$$I^B_{R2}(A, B) = \frac{\sum_{\boldsymbol{\lambda} \in \Lambda} u^*(\boldsymbol{\lambda}, A) - u^*(\boldsymbol{\lambda}, B)}{|\Lambda|},$$

where u^* is the maximum value reached by the utility function $u_{\boldsymbol{\lambda}}$ with weight vector $\boldsymbol{\lambda}$ on a Pareto-set approximation A [64]. Λ is a set of parameterizations for the utility function. The actual function used here is the augmented Tschebycheff function:

$$u_{\boldsymbol{\lambda}(z)} = \left(\max_{j \in 1..n} \lambda_j |z^*_j - z_j| + \rho \sum_{j \in 1..n} |z^*_j - z_j| \right).$$

The function aggregates the non-linear Tschebycheff function and a weighted linear sum [36], trying to reward solutions in concave as well as convex regions of the Pareto-front approximation. A value of $\rho = 0.01$ is chosen here. Again, the unary indicator I_{R2} is obtained by using a reference set instead of B. All mentioned indicators are to be minimized.

3.4 Preparing SPO for Multi-Objective Optimization

SPO was applied to multi-objective optimization for the first time by Naujoks et al. [41]. They optimized the SMS-EMOA's population size and varation, using the hypervolume indicator as utility measure. This work differs from theirs in some points. First, the former also investigates quality assessment, comparing three quality indicators and several evaluation approaches. Second, every single variation parameter can be optimized, while Naujoks et al. only differentiate between SBX variation and an evolution strategy variation [41]. Third, this work deals with benchmark problems, while Naujoks et al. investigated an airfoil design problem. The former problems are artificially created, while the latter is a real-life problem.

While real-life problems are probably more relevant to people, artificial problems offer some advantages. They are usually designed to have certain properties that make it easier for the experimenter to measure an algorithm's performance. For example, the true Pareto-front should be known for a test problem. Another plus factor is the problem's formal definition, supporting the reproducibility of results. For these reasons, we are using problems from a competition at the Congress on Evolutionary Computation (CEC) 2007 [26] in the experiments. In this competition, the quality indicators are set up according to Algorithm 3.1.

Algorithm 3.1 CEC 2007 Evaluation

1. Set the upper reference point to the worst possible objective values of the problem. If these values are unknown, estimate them by evaluating a sample of randomly generated solutions.

2. Set the lower reference point to the ideal point of the Pareto-front. This point should be easily obtainable because the Pareto-front of the problems is known.

3. Normalize the objective space between these two points, so that the lower point is mapped to $(1, \ldots, 1)^T$ and the upper point is mapped to $(2, \ldots, 2)^T$.

4. $I_{\epsilon+}$ and I_{R2} are initialized with the two reference points and a sample of the Pareto-front. I_H is initialized with $(2.1, \ldots, 2.1)^T$ as reference point.

To be able to compare results between experiments, this approach is chosen for most experiments in Chapter 4. The normalization has a few drawbacks, though. First, because the worst objective values are usually estimated, it still can happen that not all solutions of the algorithm's result dominate $(2, \ldots, 2)^T$. These points have to be filtered out or repaired. Second, the Pareto-front can be very small compared to the whole feasible objective space. This causes that differences between indicator values become very small, which might lead to numerical problems. Third, the Pareto-front has to be known. A more practical approach would be Algorithm 3.2. This approach works especially well with SPO, because you almost always have an initial algorithm design that you want to compare with. So, the old algorithm has to be run anyway for this comparison. Doing this before SPO, one can use these runs' results to obtain a reference point.

Algorithm 3.2 Evaluation for practical problems

1. Run your current optimization algorithm a number of times and save the resulting populations.

2. Set the upper reference point to the worst obtained objective values and use it with I_H.

The procedure could be easily extended to obtain the necessary reference sets for $I_{\epsilon+}$ and I_{R2}. However, it does not fix the first point of criticism. Experiment 4.2.3 compares the two approaches on a problem where they lead to quite different reference points. To complete this chapter, an example is given that elaborately describes the application of SPO.

3.4.1 Example (Performing SPO). This example shows how SPO is applied to an actual problem. The core of the optimization, (S-5) to (S-10), is executed by the SPO Toolbox, a Matlab implementation of SPO [2].

(S-1) *Preexperimental planning.* From our work with EMOAs, we know that on different problems, parameters may need completely different settings to achieve good performance. We also have the impression that there are often parameters that are overlooked.

(S-2) *Scientific claim.* We claim that the SMS-EMOA could achieve a reasonably better performance if its parameters were specifically adjusted for the current problem.

(S-3) *Statistical hypthesis.* Our null hypothesis (H_0) is that the parameter configuration found by SPO does not lead to better results than the default configuration in Table 3.1. The alternative hypothesis is that it does lead to better results. We require a significance level of 5% to reject H_0. The significance level is the probability that the decision of rejecting H_0 is actually wrong. By our choice, we would tolerate being wrong in one of 20 cases.

Table 3.1: The new configuration must succeed in a performance comparison to these default parameters. The second row shows the region of interest (ROI) for each parameter. This is the range over which we conduct our search.

Parameter	μ	η_c	η_m	p_c	p_m	p_s
Default Value	100	20.0	15.0	1.0	0.1	0.0
ROI	$\{3, \ldots, 200\}$	$[0, 40]$	$[0, 40]$	$[0, 1]$	$[0, 1]$	$[0, 1]$

(S-4) *Specification.* We decide to optimize the SMS-EMOA for an application on a problem called S_ZDT1 (see Definition A.2.1). As S_ZDT1 is a real valued problem, we need an appropriate individual. We choose one with SBX variation (see Section 2.4.1), which leads to η_m, η_c, p_m, p_c and p_s as tunable parameters. We also add μ to our investigation. The performance is measured with the hypervolume indicator. The SMS-EMOA gets a budget of 10000 problem evaluations, SPO has to get along with 500 algorithm runs.

(S-5) *Experimentation.* The experimental runs are performed.

(S-6) *Statistical modeling and prediction.* Kriging is used as surrogate model for the results of the SMS-EMOA runs.

(S-7) *Evaluation and visualization.* The SPO Toolbox plots the interaction of the parameters during the optimization process. Figure 3.3 shows such a plot of the model together with the mean squared error (MSE). Additionally, an overview of the effects is generated (see Figure 3.4), that is also obtained from the Kriging model.

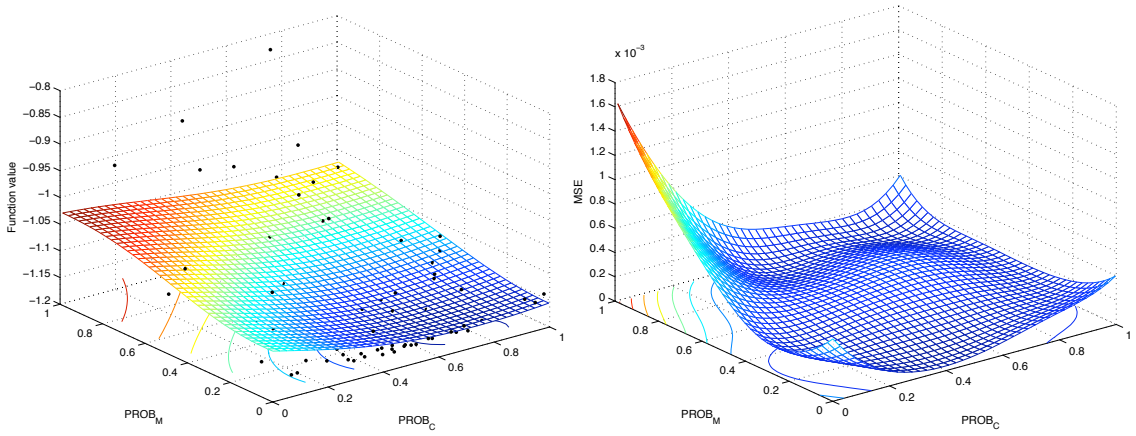


Figure 3.3: The Kriging model (left) and mean squared error (right) for SBX variation probabilities on S_ZDT1. The model shows the interaction effect of p_m and p_c. One can see by the MSE, that the area of worse function values (in red) has been less explored.

(S-8) *Optimization.* This step is rather a super-category for (S-9) and (S-10). So, it contains the check of the termination condition and the design update. Table 3.2 shows the final configuration that was found by SPO.

Table 3.2: The final configuration that was found by SPO.

Parameter	μ	η_c	η_m	p_c	p_m	p_s
Optimized Value	39	20.21	39.03	0.52	0.02	0.82

(S-9) *Termination.* If the termination criterion is reached, the process is continued at (S-11).

(S-10) *Design update.* In every iteration, two new design points are generated and added to the algorithm design. The best one is reevaluated. Then we go back to (S-5).

(S-11) *Rejection or acceptance.* After SPO has finished, the default and the final configuration are run again 50 times. The final configuration achieves a mean I_H value of -1.168, while the default configuration only reaches -1.151. The standard deviation is in both cases smaller than 0.005. A U-test [25] calculates a probability smaller than 2.2e-16 for the case that we obtain our actual results under the condition that the null hypothesis is true. This

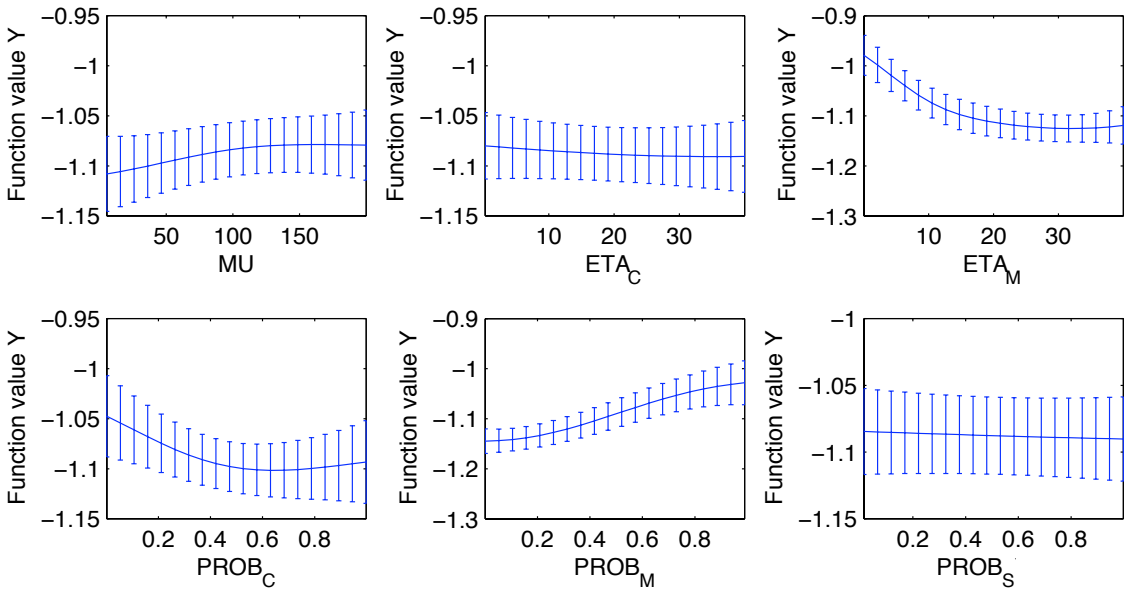


Figure 3.4: An overview of the main effects on S_ZDT1 with I_H and 10000 problem evaluations. The errorbars depict 90% confidence intervals.

probability is called *p-value*. Because it is smaller than the significance level we specified, we can reject H_0.

(S-12) *Objective interpretation.* Together with Figure 3.4 we can now try to interpret Table 3.2. The parameters η_c and p_s do not seem to have much influence. From the rest, η_m and p_m are most important. Both parameters indicate that mutation should be carried out seldom and with little perturbation. Also, a smaller population size is slightly better for this run length and recombination should be applied only with a mediocre frequency, i.e. with a probability approximately between 0.5 and 0.8.

To judge if we have found a scientifically meaningful result, let us recall the origins of the variation and the problem. SBX variation is certainly expected to perform well on ZDT problems, because the same author [10, 63] was involved in both works. Additionally, the default configuration in Table 3.1 was tested on ZDT problems. Our result is not too far away from the default, which indicates that S_ZDT1 is still quite similar to ZDT1. So, we conclude that we have found a slightly improved configuration for S_ZDT1.

Chapter 4

Experimental Results

The chapter consists of four sections. At first, the correlation and performance of different quality indicators is investigated, aggregating results from all problems. The aim of that section is to justify the indicators' use. The next section goes into detail about SBX variation, accompanied by some spot checks of indicators' behavior in different situations. Then, DE variation is covered and compared to SBX. At last, some experiments regarding the SMS-EMOA's selection are carried out.

All experiments are made with test problems from the CEC 2007 suite [26], which is described in Appendix A. However, the following results are not directly comparable to any other that were obtained before in the environment of the CEC 2007 contest, because a number of bugs in the implementation were fixed by the author (see Appendix B.2). Additionally, new reference sets are generated for the evaluation. The experiments are executed on a batch system running Linux as operating system. The used SPO version is the official Matlab implementation provided by Bartz-Beielstein [2]. All EAs and test problems are implemented by the author in Python 2.5 [57]. They are described in Appendix B. The computers' hardware configuration varies, but execution speed is not measured anyway. The reporting on experiments is done according to Preuss [43]. This standardized scheme especially separates statements of varying objectivity, making results more comprehensible.

4.1 Correlation Between Quality Indicators

4.1.1 Raw Correlation

Research Question: Do the ϵ_+ indicator ($I_{\epsilon+}$), the $R2$ indicator (I_{R2}) and the hypervolume indicator (I_H) correlate in their evaluation of randomly generated Pareto set approximations?

Preexperimental planning: Intentionally, each indicator favors different properties of Pareto set approximations. Nonetheless, they are all monotonic, which means there must

be some correlation in their judgement. The experiment should give a basic feeling for the behavior of the quality indicators.

Task: The results are obtained with Pearson's correlation coefficient [54]. The statistical null hypothesis (H_0) is that no correlation between the indicators exists.

Setup: Random populations are created and their non-dominated fronts evaluated with the three indicators. Each population consists of random individuals that are uniformly distributed in the feasible solution space. The population size is randomly drawn from $[5, s]$, where s is the default reference set size for the problem (see Table A.1). For each problem, 50 populations are created, leading to a total sample size of 950. Note that we negate the hypervolume so that all indicators are to be minimized.

Results/Visualization: I_H and $I_{\epsilon+}$ show a correlation of 0.41, $I_{\epsilon+}$ and I_{R2} 0.54 and I_H and I_{R2} even 0.76. All correlations have p-values smaller than 2.2e-16 and are thus statistically significant. So, H_0 is rejected. Figure 4.1 shows plots of the data.

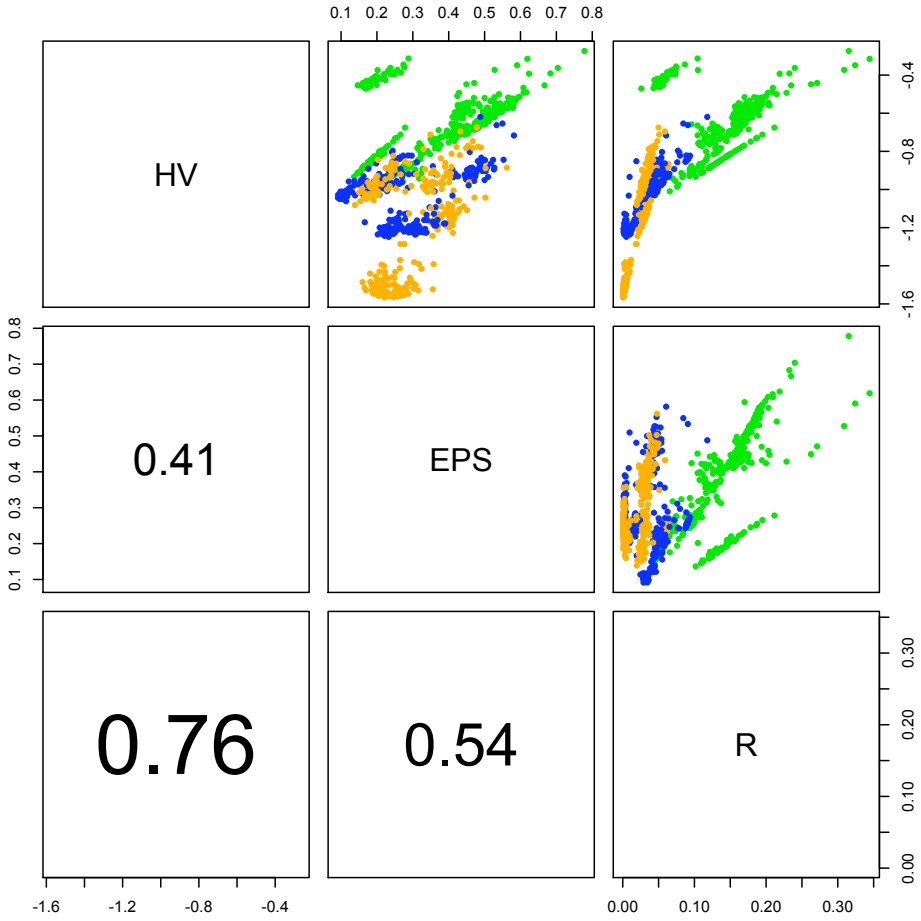


Figure 4.1: Correlation of indicators is shown both visually and in numbers. The main diagonal specifies which indicators are plotted against each other in each panel. The plots also distinguish different numbers of objectives. Green color means two, blue three, and orange five objectives. I_H and I_{R2} show the highest correlation.

Observations: The plots show that there are several clusters in the data. Some are rather cloudy while others are almost perfect straight lines. The slope of the correlation between I_H and I_{R2} seems to be dependent on the number of objectives.

Discussion: The clusters stem from the different problems thrown into the mix. This suggests that correlations are much higher on some problems if they are treated individually. However, the case shall not be investigated further here, because the question if SPO results obtained with different indicators also show similarities is more interesting.

4.1.2 Correlation in SPO

Research Question: Does SPO deliver similar results even when different quality indicators are used as objective? Which indicator is most suitable for SPO?

Preexperimental planning: Experiment 4.1.1 can be seen as preliminary study to this experiment. Additionally, some SPO runs are carried out to determine the parameters' region of interest (see Table 3.1), avoiding the production of outliers. Especially $\mu = 1$ seems to irritate the search process. Any performance comparison between configurations will be based on the mean value of a sample of runs, because this is also what SPO is set up to optimize.

Task: The different quality indicators should give similar and reasonable results when used with SPO. It would be especially nice if it could be shown that optimization with I_H also leads to better results for $I_{\epsilon+}$ and I_{R2}. Spearman's rank correlation coefficient [25] is used to compare how the indicators *evaluate* the results. The indicator's performance in *optimizing* over all problems is tested with a U-Test [25]. The significance level is always 5%.

Setup: SPO is applied to all two- and three-dimensional test problems in the CEC 2007 suite. Five-dimensional problems are excluded from the experiments because of the SMS-EMOA's high runtime on them. For every problem, SPO is carried out with each indicator. Also, two different run lengths of the SMS-EMOA are examined. The short run length represents a budget that is normally not sufficient to reach the Pareto-front, while the long run length hopefully is sufficient. So, there are six SPO runs for every problem. Each of the six SPO results is verified by doing 50 SMS-EMOA runs with the respective configuration. Again, each SMS-EMOA run is evaluated with all three indicators and the mean value is computed for each indicator. These mean values are then converted to ranks on every problem, resulting in a 3×3 matrix for each run length. Thus, we gain information about indicator performance and correlation in the same experiment.

The individuals are using simulated binary crossover and polynomial mutation as variation operators, which were described in Section 2.4.1. Table 4.1 shows an overview of the specification. The region of interest for the parameters was already mentioned in Table 3.1.

Table 4.1: The setups for experiments with SBX variation. The different configurations add up to 78 separate SPO runs.

Problems	Two- and three-dimensional CEC 2007 problems
SPO budget	500 algorithm runs
Algorithm initialization	Uniform random
Stopping criterion	$500 \cdot M$ and $5000 \cdot M$ problem evaluations
Algorithm	SMS-EMOA
Parameters	$\mu, \eta_c, \eta_m, p_c, p_m, p_s$
Initial experimental design	Latin Hypercube (50 points, 3 repeats per point)
Performance measures	$I_H, I_{\epsilon+}, I_{R2}$

Results/Visualization: Detailed results and the found configurations for each problem are presented in the following experiments. Table 4.2 shows a compilation of the performance results. The rank correlation between I_H and $I_{\epsilon+}$ is 0.54 for $500 \cdot M$ evaluations and 0.41 for $5000 \cdot M$ evaluations. The correlation between I_H and I_{R2} is 0.58 for $500 \cdot M$ evaluations and 0.65 for $5000 \cdot M$. $I_{\epsilon+}$ and I_{R2} have correlations of 0.35 and 0.06. Except for the last value, all correlations are significant.

Table 4.2: The summed-up ranks achieved by the indicators. A lower rank-sum is better. The main diagonal almost always contains the row- and column-wise minimum.

$500 \cdot M$		Evaluated			Total
		I_H	$I_{\epsilon+}$	I_{R2}	
Optimized	I_H	20	24	24	**68**
	$I_{\epsilon+}$	26	22	28	76
	I_{R2}	32	32	26	90

$5000 \cdot M$		Evaluated			Total
		I_H	$I_{\epsilon+}$	I_{R2}	
Optimized	I_H	22	27	25	**74**
	$I_{\epsilon+}$	28	18	29	75
	I_{R2}	26	31	22	79

Observations: I_H is the most successful indicator, attaining the lowest rank-sum over all indicators and problems. As in Experiment 4.1.1, I_H and I_{R2} have the highest correlation. Nonetheless, I_H achieves a significantly better performance than I_{R2} (column "Total" in Table 4.2) with $500 \cdot M$ evaluations. All other differences in rank-sums are not statistically significant. By the way, the found parameters themselves do also correlate between the indicators (not shown here). The highest correlation can be seen in μ, p_c and p_m.

Discussion: As expected, it is most efficient to optimize with the indicator that is also used for evaluation. In other words, the indicators are different enough to separate each

one from the others. Our expectation for the performance of I_H was fulfilled, because optimization with I_H achieves the best overall score.

4.2 SBX Variation

The results that were presented in a summarized form in Experiment 4.1.2 are now examined in detail for each individual test problem. Every parameter configuration found is compared to the configuration in Table 3.1. The default value for p_s is set to zero because the reference implementation of SBX variation uses the asymmetric variant. Generally, it is not expected that p_s has much influence, so it will be simply left out from the reporting unless the contrary is found to be true.

4.2.1 SBX on OKA2

Research Question: Which parameter configurations will be found for the problem OKA2? Will they improve performance on the respective indicator compared to the default configuration?

Preexperimental planning: Compared to the other problems in the test collection, OKA2 is outstanding for its low number of decision variables (see Table A.1). Nonetheless, it is a relatively new problem and probably not among the easiest ones.

Task: After SPO has finished, the found configuration is run 50 times. Each run is evaluated with every indicator, so that the configurations can be compared with each other and the default configuration. Comparisons with the default configuration are done by a statistical test: The null hypothesis is that there is no difference in means. The alternative hypothesis is that the optimized configuration produces a lower mean best value than the default configuration. The results are considered statistically significant if a U-Test [25] rejects the null hypothesis at a significance level of 5%. The optimized configurations are not compared with a statistical test, but only ranked by their mean performance. This data was aggregated in Experiment 4.1.2.

Setup: The setup was already described in Experiment 4.1.2.

Results/Visualization: The parameter configurations found by SPO are shown in Table 4.3. Figure 4.2 shows the effects of μ on short runs, as seen by the Kriging model.

Observations: SPO's effect plots do not indicate much influence of the parameters for $I_{\epsilon+}$ and I_{R2} at 1000 evaluations. All six configurations are statistically significant improvements compared to the default configuration, though. For 10000 evaluations, a high population size is chosen by the majority.

Discussion: Parameter values seem to differ widely between indicators and run lengths. This would be consistent with the perceived weak influence of the parameters. On the other hand, the Kriging model probably failed on $I_{\epsilon+}$ and I_{R2} (for 1000 evaluations). Anyway,

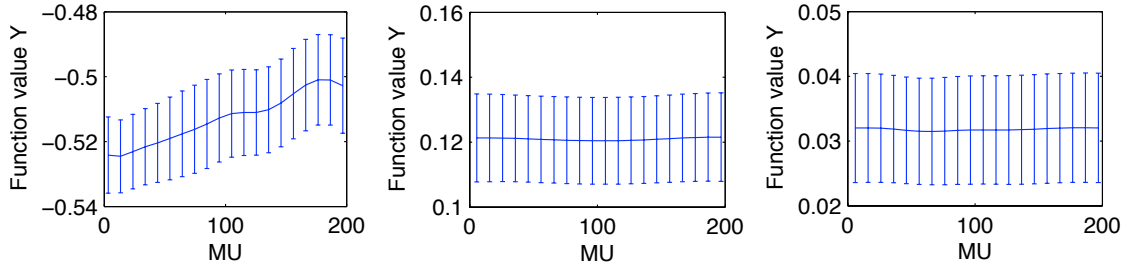


Figure 4.2: Main effects for μ on OKA2 with 1000 evaluations. It seems that on the short runs, μ has of all parameters the strongest influence, at least for I_H. Neither the overview for $I_{\epsilon+}$, nor for I_{R2} shows any noteworthy effect for any parameter. The errorbars depict 90% confidence intervals.

Table 4.3: SPO results for SBX variation on OKA2. Each row specifies a configuration and the setting it was found in, i.e. the indicator that was used for optimization. The last three columns indicate which configuration performs best in the evaluation with each indicator, regarding the mean best value.

Evaluations	Indicator	Configuration						Rank		
		μ	η_c	η_m	p_c	p_m	p_s	I_H	$I_{\epsilon+}$	I_{R2}
1000	I_H	13	18.51	11.01	0.61	0.49	0.88	**1**	3	3
	$I_{\epsilon+}$	23	27.07	22.08	0.42	0.43	0.24	2	**1**	2
	I_{R2}	56	30.86	8.55	0.29	0.66	0.67	3	2	**1**
10000	I_H	107	21.52	27.43	0.45	0.69	0.76	2	2	2
	$I_{\epsilon+}$	180	22.57	33.14	0.03	0.45	0.23	**1**	**1**	3
	I_{R2}	57	21.02	3.43	0.49	0.57	0.21	3	3	**1**

compared to other problems, it is remarkable that OKA2 seems to be one of the few where the quality indicators' correlation is low. For this reason, OKA2 is analyzed in greater detail in Experiment 4.2.5.

4.2.2 SBX on SYM-PART

Research Question: Which parameter configurations will be found for the problem SYM-PART? Will they improve performance on the respective indicator compared to the default configuration?

Preexperimental planning: SYM-PART was designed as a testbed for algorithms' ability to maintain diversity in the decision space. As this aspect is not considered here, SYM-PART should not be very difficult for the SMS-EMOA. The feasible objective space is very large compared to the Pareto-front on this problem (see Figure A.2), which might be a problem for the performance measurement.

Task: The task is the same as in Experiment 4.2.1.

Setup: The setup was already described in Experiment 4.1.2.

Results/Visualization: The parameter configurations found by SPO are shown in Table 4.4. H_0 is rejected for all configurations, which means the improvements are statistically significant.

Table 4.4: SPO results for SBX variation on SYM-PART.

Evaluations	Indicator	Configuration						Rank		
		μ	η_c	η_m	p_c	p_m	p_s	I_H	$I_{\epsilon+}$	I_{R2}
1000	I_H	4	20.70	18.52	0.59	0.07	0.27	**1**	**1**	**1**
	$I_{\epsilon+}$	5	38.59	39.15	0.92	0.19	0.66	2	2	2
	I_{R2}	5	13.71	13.51	0.56	0.10	0.38	3	3	3
10000	I_H	9	5.05	39.48	0.01	0.02	0.03	2	2	2
	$I_{\epsilon+}$	13	37.16	38.73	0.63	0.15	0.89	3	3	3
	I_{R2}	7	4.86	34.25	0.01	0.09	0.86	**1**	**1**	**1**

Observations: The effect plots (not shown here) indicate that η_m has a high influence and should approximately be greater than 20. While p_m is chosen constantly low, p_c is only low on the long runs. Especially I_H and I_{R2} suggest very low variation probabilities for 10000 evaluations. The two show a high correlation in all their parameter results. All the indicators also perfectly correlate in their evaluation of the configurations. The population size is continuously low, the most successful configurations actually feature the smallest population size.

Discussion: The surprisingly low recombination probabilities may stem from the symmetry in the decision space. SYM-PART allows solutions that are very different in decision space to be equally fit. The experiment indicates that doing SBX between these solutions will probably lead to worse offspring. The proposed population sizes are surprisingly small, which certainly means there is not much diversity maintained here. However, this was not the task. The question, if normalizing the whole objective space on this problem is a good idea, still persists and will be tackled again in Experiment 4.2.3.

4.2.3 Quality Assessment on SYM-PART

Research Question: Is the normalization of the objective space a handicap for SPO or is it harmless? Can the random noise be reduced while keeping normalization?

Preexperimental planning: To investigate the research question, a problem is needed where normalization has a great impact. The Figures in Appendix A show that this is the case for SYM-PART. Another idea is to subtract the initial population's hypervolume

I_H^0 from the final indicator value. The assumption is that some noise from the random initialization could be neutralized. This could be important, because this experiment will use very short algorithm runs of only 1000 problem evaluations.

Task: We have Algorithms 3.1 and 3.2 as alternatives for our evaluation approach. We call this factor A. The indicator choice is called factor B. So we have a full factorial design with two factors $A \in \{3.1, 3.2\}$ and $B \in \{I_H, I_H - I_H^0\}$ that determine the setup. Each combination is run 20 times. The outcomes of the SPO runs are analyzed regarding their performance and also their variance. A U-test [25] is used to compare the distributions' means. Again, we require a significance level of 5%.

Setup: The evaluation is only done with the hypervolume indicator. The part of the setup that is not fixed by the factors A and B is taken from previous experiments, namely the ones described in Tables 3.1 and 4.1. Because many repetitions are carried out, only 1000 problem evaluations are given to the SMS-EMOA. The problem's true nadir point is $(2, 2)^T$. The reference point obtained with Algorithm 3.2 is $(28.71, 30.40)^T$ while the reference point used in Algorithm 3.1 would correspond to $(550, 550)^T$ (cf. Figure A.2). These numbers show that SYM-PART is indeed easy to optimize: A budget of 1000 problem evaluations is enough to dominate at least 80% of the objective space.

Results/Visualization: Table 4.5 shows the standard deviations obtained with the two setups. Figure 4.3 shows the main effects for the two factors.

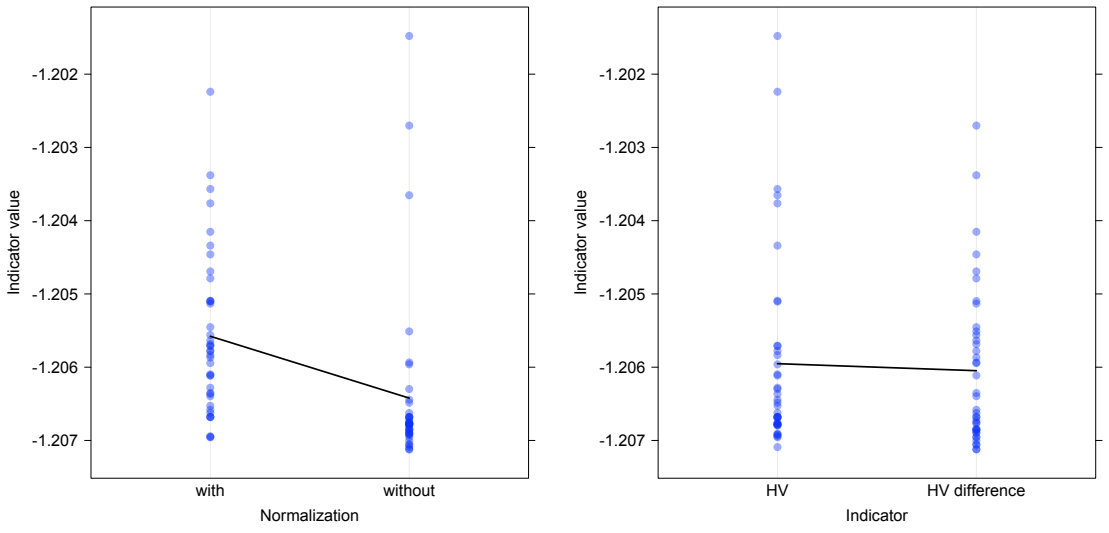


Figure 4.3: Main effects for the two factors A (left) and B (right). The black lines connect the samples' mean values.

Observations: Factor A yields a significant improvement, but factor B does not. The standard deviation of almost all parameters is lower when no normalization is used.

Table 4.5: The parameters' standard deviation on the 20 runs.

No.	Factors		Mean I_H	Standard Deviation						
	A	B		I_H	μ	η_c	η_m	p_c	p_m	p_s
1	3.1	I_H	-1.2056	0.00125	1.39	12.85	9.00	0.20	0.13	0.31
2	3.1	$I_H - I_H^0$	-1.2055	**0.00093**	1.12	**11.31**	10.39	0.25	0.13	0.28
3	3.2	I_H	-1.2063	0.00135	1.04	12.05	**4.33**	**0.17**	**0.07**	0.28
4	3.2	$I_H - I_H^0$	**-1.2065**	0.00098	**0.60**	13.23	4.98	0.23	0.14	**0.24**

Discussion: Admittedly, the use of Algorithm 3.2 does probably not make such a big difference on most other CEC 2007 problems. Often the reference points obtained by the two evaluation algorithms are quite similar. To evaluate if there is really no impact of factor B, further studies would have to be carried out.

4.2.4 SBX on ZDT-based Problems

Research Question: Which parameter configurations will be found for ZDT-based problems? Will they improve performance on the respective indicator compared to the default configuration?

Preexperimental planning: The ZDT problems by Zitzler, Deb and Thiele [63] have been manipulated for the CEC 2007 contest to overcome some of their drawbacks (see Appendices A and B). This experiment is again designed for a first exploration of these problems. R_ZDT4 is based on the same problem as S_ZDT4. Especially they feature the same Pareto-front. ZDT4, the problem they are based on, is multimodal and considered the most difficult of the ZDT problem collection [63]. Also, R_ZDT4 could be considered more difficult than S_ZDT4 because of the rotation it includes. On the other hand, R_ZDT4 only contains ten decision variables, while S_ZDT4 has 30. It will be interesting to see if any differences can be found between them.

Task: The task is the same as in Experiment 4.2.1.

Setup: The setup was already described in Experiment 4.1.2.

Results/Visualization: The parameter configurations found by SPO are shown in Table 4.6. All configurations are significant improvements, except for R_ZDT4, where the U-test indicates that those for I_H and I_{R2} with 10000 evaluations are not. One interaction effect and the main effects overview for 10000 evaluations and I_H on S_ZDT1 were already shown in Figures 3.3 and 3.4. Figure 4.4 shows an overview of the main effects on S_ZDT2 with I_H and 1000 evaluations.

Observations: p_m is quite low in all configurations. The effects on S_ZDT1 with 10000 evaluations and I_H were already covered in Example 3.4.1. Additionally, we now see that the population size for 10000 evaluations on S_ZDT1 and R_ZDT4 is on all indicators

Table 4.6: SPO results for SBX variation on ZDT-based problems.

Problem	Evaluations	Indicator	Configuration						Rank		
			μ	η_c	η_m	p_c	p_m	p_s	I_H	$I_{\epsilon+}$	I_{R2}
S_ZDT1	1000	I_H	10	13.72	9.55	0.79	0.10	0.03	2	3	3
		$I_{\epsilon+}$	14	0.89	8.33	0.60	0.04	0.20	**1**	**1**	**1**
		I_{R2}	4	28.95	8.03	0.50	0.05	0.59	3	2	2
	10000	I_H	39	20.21	39.03	0.52	0.02	0.82	3	3	3
		$I_{\epsilon+}$	86	4.06	8.26	0.64	0.01	0.98	2	**1**	2
		I_{R2}	23	7.51	33.33	0.50	0.04	0.49	**1**	2	**1**
S_ZDT2	1000	I_H	4	33.87	9.49	0.80	0.02	0.66	2	2	2
		$I_{\epsilon+}$	33	22.41	38.01	0.74	0.14	0.05	3	3	3
		I_{R2}	7	35.12	4.51	0.68	0.03	0.17	**1**	**1**	**1**
	10000	I_H	18	9.21	17.18	0.35	0.01	0.12	2	2	3
		$I_{\epsilon+}$	69	1.10	0.09	0.65	0.001	0.52	**1**	**1**	**1**
		I_{R2}	5	13.71	13.51	0.56	0.10	0.38	3	3	2
S_ZDT4	1000	I_H	4	13.19	17.68	0.75	0.08	0.06	2	2	2
		$I_{\epsilon+}$	4	33.87	9.49	0.80	0.02	0.66	**1**	**1**	**1**
		I_{R2}	23	26.45	28.58	0.56	0.05	0.29	3	3	3
	10000	I_H	7	35.12	4.51	0.68	0.03	0.17	2	**1**	2
		$I_{\epsilon+}$	9	24.52	15.20	0.44	0.06	0.36	3	3	3
		I_{R2}	8	35.32	12.21	0.82	0.02	0.82	**1**	2	**1**
R_ZDT4	1000	I_H	5	17.96	12.12	0.52	0.17	0.19	**1**	**1**	2
		$I_{\epsilon+}$	24	15.33	16.28	0.78	0.13	0.24	2	3	**1**
		I_{R2}	19	3.33	21.20	0.61	0.02	0.49	3	2	3
	10000	I_H	196	34.96	33.94	0.99	0.04	0.28	2	2	2
		$I_{\epsilon+}$	151	32.75	8.54	0.86	0.04	0.82	**1**	**1**	**1**
		I_{R2}	140	4.82	20.07	0.92	0.04	0.39	3	3	3
S_ZDT6	1000	I_H	11	28.53	9.06	0.68	0.13	0.38	2	2	2
		$I_{\epsilon+}$	7	29.03	8.84	1.00	0.11	0.46	**1**	**1**	**1**
		I_{R2}	47	12.81	16.73	0.85	0.06	0.36	3	3	3
	10000	I_H	16	30.14	31.48	0.97	0.05	0.28	3	3	3
		$I_{\epsilon+}$	6	19.53	31.60	0.73	0.03	0.003	**1**	**1**	**1**
		I_{R2}	7	17.81	16.71	0.97	0.04	0.78	2	2	2

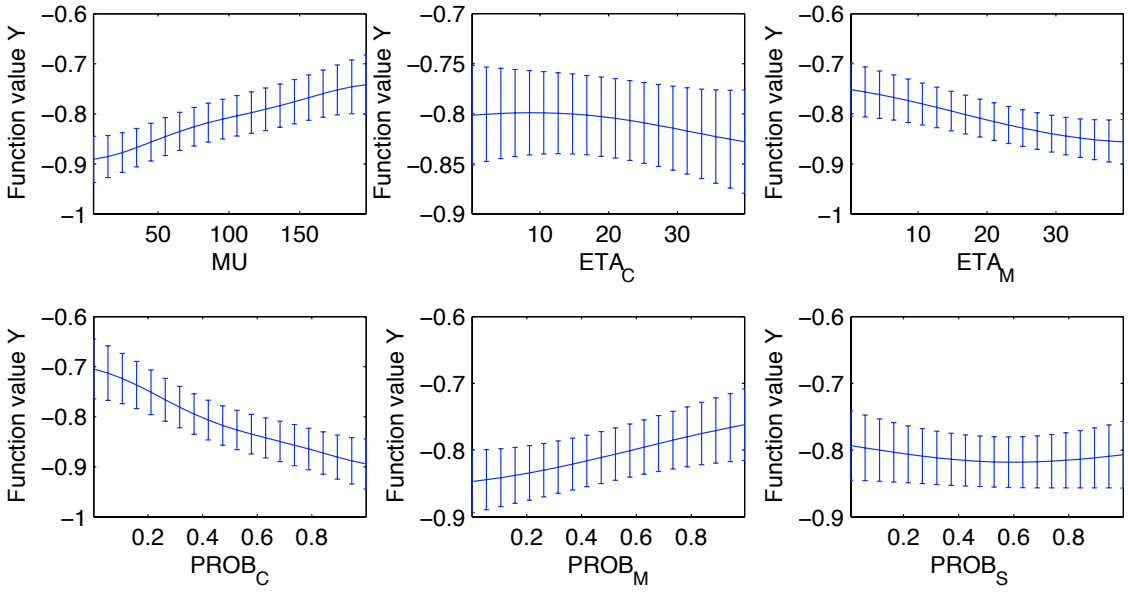


Figure 4.4: Main effects on S_ZDT2 with I_H and 1000 evaluations. η_c and p_s show hardly any influence, μ and p_c cause the strongest effects. The errorbars depict 90% confidence intervals. The overviews for $I_{\epsilon+}$ and I_{R2} look quite similar.

at least four times higher than for 1000 evaluations. The difference is not as big on the other problems. The population size on R_ZDT4 is chosen much higher than on S_ZDT4 for 10000 evaluations. The indicators correlate strongly in their evaluation of the found configurations. It also stands out that in total five SPO runs perform worst in the evaluation with the indicator they were optimized for.

Discussion: The used problems have $n = 30$ decision variables (except for R_ZDT4). It is interesting to see that the p_m values lie around $\frac{1}{n} \approx 0.03$, a value that is often suggested for mutation probabilty by a rule of thumb [4, page 16]. For a number of parameters, it is difficult to make out trends between 1000 and 10000 evaluations, because the results are varying a lot between indicators. It seems strange that very different configurations were found although the indicators have a strong correlation. Thus, S_ZDT2 is considered again in Experiment 4.2.5. It is exceptional that in some settings SPO fails to deliver improved configurations on R_ZDT4. We consider this a *floor effect*: The problem is so hard that any configuration fails. As far as can be seen, the higher population size for 10000 evaluations is the only real difference between the parameters on S_ZDT4 and R_ZDT4.

4.2.5 Variance of Results

Research Question: How robust is SPO with the respective quality indicators?

Preexperimental planning: The previous experiments sometimes produced widely scattered parameter configurations. It is important to know where this variance comes from,

because it could stem from an ill-designed experimental setup. First, the quality indicator itself could fail to deliver the desired meaningful feedback. Second, the normalization of the problems' objective space (shown in Appendix A) might degenerate the Pareto-front to such an extent that evaluation becomes impossible. So, the goal is to repeat SPO runs with all indicators to observe the impact of random noise.

Task: The experiment is designed to give an idea of how robust SPO is at finding parameter configurations. The results are analyzed graphically with histograms and by computing the standard deviation for each parameter. Two-sided Kolmogorov-Smirnov (KS) tests [8] are carried out for each parameter to find out if different indicators come to identical distributions. H_0 is that two compared samples come from the same distribution. We require a significance level of 5% to reject H_0.

Setup: The setup is very limited due to SPO's high runtime. From the previously examined problems, OKA2 and S_ZDT2 with 1000 problem evaluations are selected. These two are chosen because of the different impressions they gave regarding the correlation of the indicators and the variance of the parameters (see Experiments 4.2.1 and 4.2.4). Each experiment is repeated 20 times for every quality indicator. The initial latin hypercube design is always the same, but the SMS-EMOA is run each time with a different random seed. The remaining variables of the experiments are identical to those of Experiment 4.1.2.

Results/Visualization: Figure 4.5 shows histograms generated from the 20 runs with I_H on S_ZDT2. Accordingly, Figure 4.6 contains some selected parameters for OKA2, now regarding all indicators. Table 4.7 shows the standard deviations measured for all indicators in this experiment. By the way, the overall best configurations are presented in Table 4.8.

Table 4.7: Standard deviations of 20 SPO runs on OKA2 and S_ZDT2.

Problem	Indicator	Standard Deviation					
		μ	η_c	η_m	p_c	p_m	p_s
OKA2	I_H	**10.160**	10.630	8.585	0.253	**0.172**	0.213
	$I_{\epsilon+}$	11.189	**8.801**	**8.534**	0.136	0.206	**0.201**
	I_{R2}	24.090	10.591	12.680	**0.104**	0.198	0.248
S_ZDT2	I_H	**11.430**	12.484	**10.464**	0.172	0.046	**0.289**
	$I_{\epsilon+}$	13.960	12.889	13.780	0.136	**0.035**	0.291
	I_{R2}	16.620	**12.418**	11.971	**0.108**	0.048	0.294

Observations: For S_ZDT2, η_m values of the best configurations are extremely small, near their minimal possible value of zero. Population size and p_m exhibit small variances, p_c at medium level, η_c, η_m and p_s have a high variance. This can be observed for all three indicators. Similarly, no significant differences can be found between the sampling

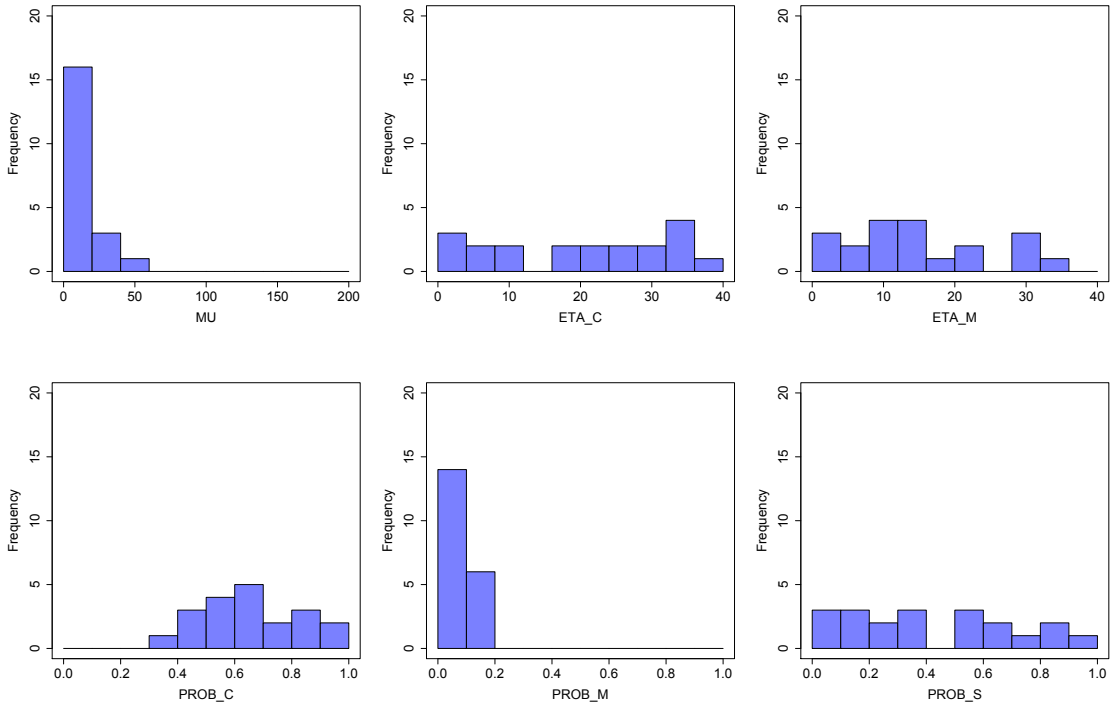


Figure 4.5: Histograms of 20 SPO runs with I_H on S_ZDT2 (cf. Figure 4.4), showing the densities of the parameter values. The regions of interest are divided into ten bins. Each bar depicts the absolute number of values in that bin. The histograms for $I_{\epsilon+}$ and I_{R2} look quite similar and are not shown here.

distributions. OKA2 behaves differently. All three distributions of μ are found to be mutually different and, except for η_c, all other parameters contain different distributions in at least one combination, too. The ranking of the indicators for both problems is surprisingly similar to the one found before.

Discussion: The results for S_ZDT2 are largely conform with Experiment 4.2.4 (cf. Figure 4.4). The similar η_m values of the best configurations are a counterexample to the idea that a high variance means the parameter is meaningless. But it shows that some parameters are more easily optimized than others. This is as expected, and caused by the parameters' interaction with each other. For example, η_m directly interacts with p_m. Interestingly, OKA2 is, in contrast to S_ZDT2, indeed a problem where the indicators do not correlate much. So, the correlation of the indicators when used in SPO is indeed dependent on the problem, but there is no hint for any general problems induced by using SPO on MOOPs in the proposed fashion. Also, no evidence was found that any of the three indicators is generally more robust than the others, in terms of variance of the found parameters. The experiment shows, though, that a multi-start approach is still beneficial

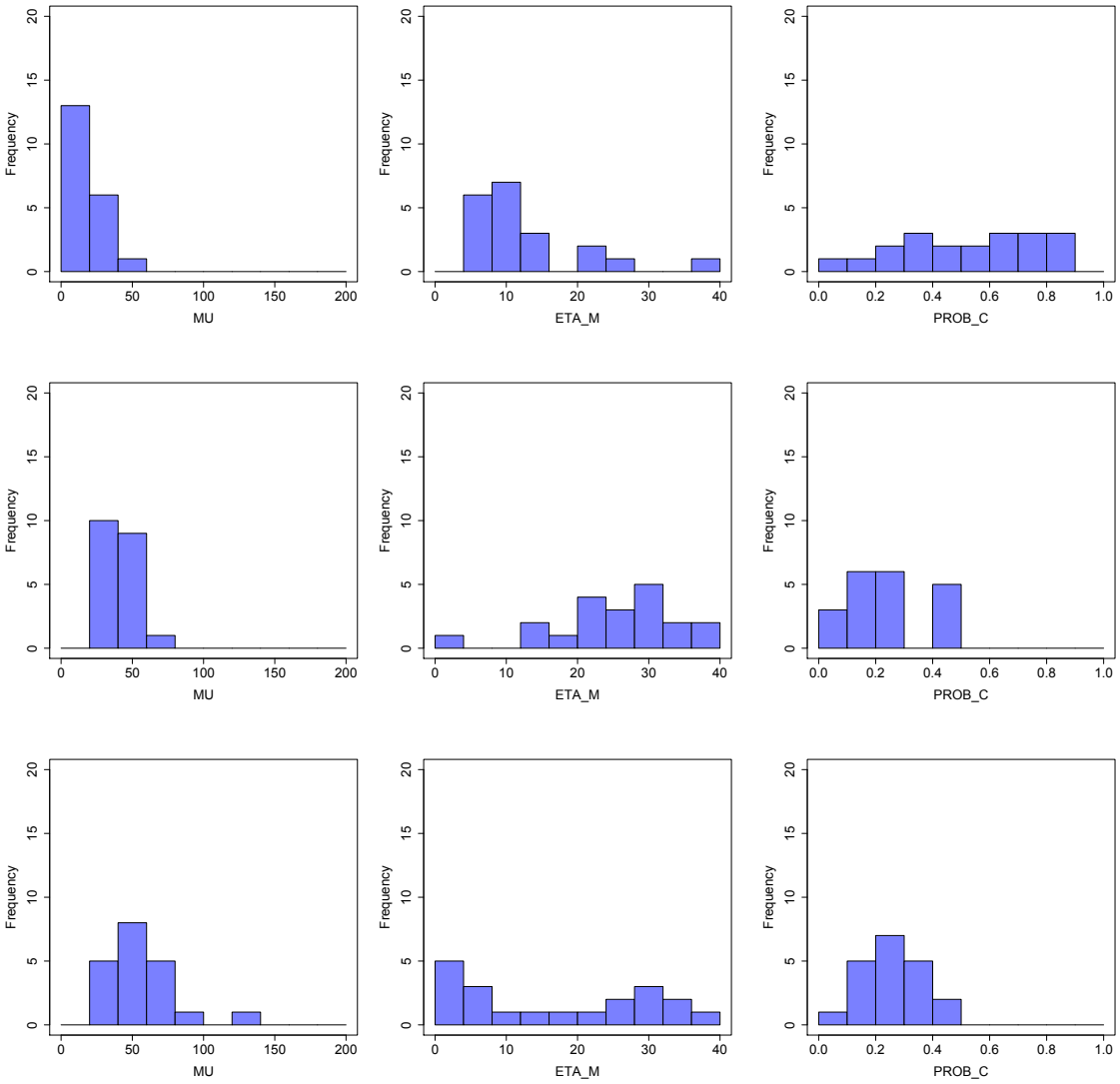


Figure 4.6: Histograms of μ, η_m and p_c on OKA2. Slight differences can be seen between I_H (first row), $I_{\epsilon+}$ and I_{R2} (last row). KS tests affirm this impression by testing each sample against the others in its column.

Table 4.8: *Best-of-20* configurations for SBX variation on OKA2 and S_ZDT2 (1000 problem evaluations). Like in previous experiments, the configurations are ranked by drawing a sample of 50 runs and comparing the mean best value.

Problem	Indicator	Configuration						Rank		
		μ	η_c	η_m	p_c	p_m	p_s	I_H	$I_{\epsilon+}$	I_{R2}
OKA2	I_H	34	4.31	13.41	0.24	0.84	0.96	**1**	2	2
	$I_{\epsilon+}$	41	7.64	14.21	0.12	0.54	0.76	2	**1**	3
	I_{R2}	56	30.86	8.55	0.29	0.66	0.67	3	3	**1**
S_ZDT2	I_H	3	24.54	0.62	0.68	0.04	0.59	2	2	2
	$I_{\epsilon+}$	7	6.79	0.15	0.87	0.04	0.63	3	3	3
	I_{R2}	4	28.90	0.88	0.46	0.05	0.37	**1**	**1**	**1**

for SPO in the chosen setup. Another question would be if alternatively increasing the budget of a single run leads to a similar improvement.

4.2.6 SBX on DTLZ-based Problems

Research Question: Which parameter configurations will be found for the DTLZ-based problems? Will they improve performance on the respective indicator compared to the default configuration?

Preexperimental planning: Analogously to the ZDT problems, some DTLZ problems have been extended for the CEC 2007 competition [26]. The problem instances used here have three objectives. Although it might not suffice to make up for the increased difficulty of the problem, the number of problem evaluations is increased by 50%, compared to two-dimensional problems.

Task: The task is the same as in Experiment 4.2.1.

Setup: The setup was already described in Experiment 4.1.2.

Results/Visualization: The parameter configurations found by SPO are shown in Table 4.9. All configurations are significant improvements compared to the default configuration. Figure 4.7 shows two results that were obtained when validating the I_H configurations for 15000 evaluations.

Observations: The mutation probability p_m is quite low and again around $\frac{1}{n}$ on all problems. Other parameters of the configurations show a lot of variance, e. g. for μ. For R_DTLZ2, all configurations with rank one contain a low η_c value, and p_c is rather low, too. This means that recombination is done relatively seldom, but with a high perturbation.

Discussion: The indicators often show oppositional evaluations and some configurations seem to be very different from each other. This makes an interpretation difficult because

Table 4.9: SPO results for SBX variation on DTLZ-based problems.

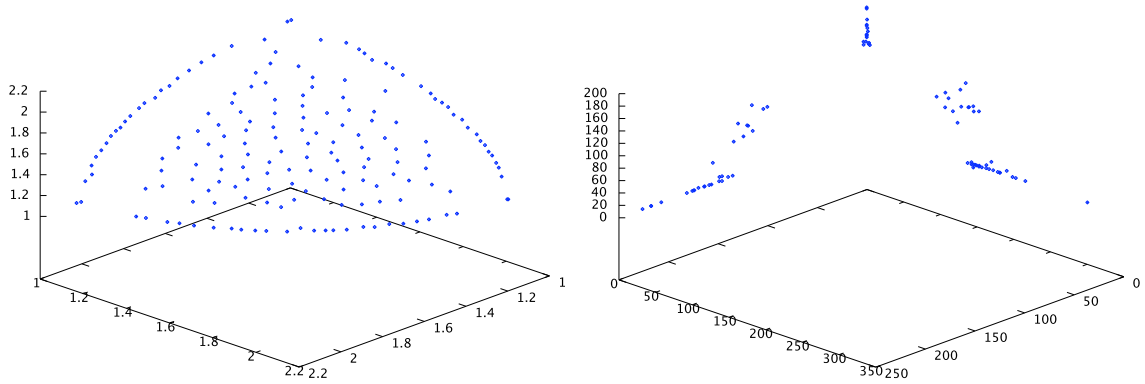
Problem	Evaluations	Indicator	Configuration						Rank		
			μ	η_c	η_m	p_c	p_m	p_s	I_H	$I_{\epsilon+}$	I_{R2}
S_DTLZ2	1500	I_H	9	28.39	10.47	0.42	0.10	0.79	3	2	3
		$I_{\epsilon+}$	12	29.33	16.38	0.58	0.09	0.36	**1**	**1**	2
		I_{R2}	30	32.05	31.73	0.65	0.10	0.56	2	3	**1**
	15000	I_H	172	36.10	21.86	0.53	0.02	0.02	**1**	2	**1**
		$I_{\epsilon+}$	196	7.17	15.27	0.46	0.01	0.61	2	**1**	3
		I_{R2}	12	36.36	25.86	0.11	0.02	0.81	3	3	2
R_DTLZ2	1500	I_H	12	16.58	32.50	0.25	0.31	0.86	2	2	2
		$I_{\epsilon+}$	38	1.01	33.71	0.83	0.05	0.32	**1**	**1**	3
		I_{R2}	9	3.96	33.11	0.05	0.63	0.55	3	3	**1**
	15000	I_H	179	3.72	39.54	0.21	0.17	0.05	**1**	**1**	3
		$I_{\epsilon+}$	71	1.49	29.01	0.14	0.0007	0.34	3	2	**1**
		I_{R2}	15	19.83	15.33	0.06	0.13	0.31	2	3	2
S_DTLZ3	1500	I_H	27	29.91	22.10	0.27	0.01	0.37	**1**	2	**1**
		$I_{\epsilon+}$	11	34.19	35.83	0.90	0.02	0.22	3	3	3
		I_{R2}	27	27.46	12.28	0.53	0.0002	0.21	2	**1**	2
	15000	I_H	83	13.93	24.37	0.50	0.03	0.15	**1**	3	**1**
		$I_{\epsilon+}$	5	25.20	4.97	0.19	0.04	0.78	3	**1**	3
		I_{R2}	90	13.29	35.93	0.62	0.001	0.35	2	2	2

the results are very inconsistent. Figure 4.7(b) gives a hint why this might be the case, at least on S_DTLZ3: Similar to SYM-PART, the objective space is large compared to the Pareto-front (cf. Table A.1). So, the interesting details are not covered in the evaluation. Additionally, S_DTLZ3 seems to be more difficult than S_DTLZ2. In comparison, 4.7(a) shows a nice coverage of and convergence to the Pareto-front. It also shows the typical distribution of points generated by the SMS-EMOA on concave problems [40].

4.2.7 SBX on WFG Problems

Research Question: Which parameter configurations will be found for the WFG problems? Will they improve performance on the respective indicator compared to the default configuration?

Preexperimental planning: The WFG problems are from the WFG toolkit [28]. They have three objectives, like the DTLZ-based problems. Because of the exponential complexity of the hypervolume computation, the SMS-EMOAs runtime is significantly increased



(a) S_DTLZ2 (b) S_DTLZ3

Figure 4.7: Populations on shifted DTLZ problems. Both have identical Pareto-fronts. (a) shows a typical result of a validation run for the I_H configuration on S_DTLZ2 (15000 evaluations). (b) shows another I_H result on S_DTLZ3, that is still far away from the Pareto-front. This population achieves a normalized I_H value of -1.330990, which is the 22nd best result of 50 runs.

compared to two objectives. For some reason, it is even higher on WFG than on DTLZ-based problems. For this reason, the Kriging-based optimization phase of SPO is sacrified on the long runs, to use the computing time for more interesting experiments. So, for experiments with 15000 evaluations, only the initial design was computed, comprising 150 of the 500 originally planned algorithm runs.

Task: The task is the same as in Experiment 4.2.1.

Setup: The setup was already described in Experiment 4.1.2.

Results/Visualization: The parameter configurations found by SPO are shown in Table 4.10. All configurations are significant improvements compared to the default configuration.

Observations: The observed variation probabilities are often opposing the ones seen on ZDT and DTLZ-based problems. Especially with 1500 evaluations, p_m is often higher than p_c. I_H and I_{R2} choose the same configuration on WFG1 and WFG8 with 15000 evaluations. This is a consequence of omitting the optimization phase. $I_{\epsilon+}$ favors the highest population sizes, which is generally not the case on other problems.

Discussion: The fact that I_H and I_{R2} choose the same configurations reinforces the result of Experiment 4.1.1 that the two have a high correlation. Their evaluations are often directly opposing those of $I_{\epsilon+}$.

4.2.8 Summary

For the first time, $I_{\epsilon+}$ and I_{R2} were used as performance measures in SPO and compared to each other and I_H. For $5000 \cdot M$ problem evaluations, SPO carried out 19.2 optimization

Table 4.10: SPO results for SBX variation on the WFG problems.

Problem	Evaluations	Indicator	Configuration						Rank		
			μ	η_c	η_m	p_c	p_m	p_s	I_H	$I_{\epsilon+}$	I_{R2}
WFG1	1500	I_H	16	33.43	3.26	0.07	0.99	0.95	**1**	**1**	**1**
		$I_{\epsilon+}$	21	8.18	7.54	0.18	0.95	0.86	3	2	3
		I_{R2}	9	6.78	35.53	0.01	1.00	0.93	2	3	2
	15000	I_H	30	32.05	31.73	0.65	0.10	0.56	**1**	2	**1**
		$I_{\epsilon+}$	142	31.50	29.30	0.15	0.92	0.79	3	**1**	3
		I_{R2}	30	32.05	31.73	0.65	0.10	0.56	**1**	2	**1**
WFG8	1500	I_H	7	24.45	34.96	0.13	0.87	0.60	**1**	2	**1**
		$I_{\epsilon+}$	38	35.30	24.31	0.42	0.25	0.02	3	**1**	3
		I_{R2}	6	31.44	36.57	0.30	0.55	0.96	2	3	2
	15000	I_H	15	19.83	15.33	0.06	0.13	0.31	**1**	2	**1**
		$I_{\epsilon+}$	115	27.76	20.31	0.07	0.12	0.65	3	**1**	3
		I_{R2}	15	19.83	15.33	0.06	0.13	0.31	**1**	2	**1**
WFG9	1500	I_H	31	24.61	31.54	0.39	0.71	0.95	**1**	**1**	**1**
		$I_{\epsilon+}$	69	24.83	31.42	0.90	0.41	0.69	3	2	3
		I_{R2}	17	37.29	17.03	0.33	0.05	0.57	2	3	2
	15000	I_H	152	11.98	33.25	0.70	0.14	0.67	**1**	2	**1**
		$I_{\epsilon+}$	193	9.14	39.47	0.71	0.01	0.85	2	**1**	2
		I_{R2}	83	13.93	24.37	0.50	0.03	0.15	3	3	3

steps on average. The final configuration was also the best in 40% of these steps. The numbers for $500 \cdot M$ evaluations are similar (19.4 steps, 36%). This high percentage means that improved configurations are seldom found and seems to indicate that optimization via the Kriging model is not very successful.

The population size has the strongest influence throughout the problems. Figure 4.8 shows how the different setups on average influence it. It appears plausible that the population size for $500 \cdot M$ evaluations is the lowest for I_H, because the Pareto-front is not reached anyway. So, it is probably sufficient to optimize a single, balanced solution to maximize the hypervolume. The influence of p_s is indeed very weak. Not a single problem was found where it played a role. The remaining parameters show mixed effects.

The obtained parameter configurations are often very different from each other, even on the same problem. This seems normal, because Smit and Eiben made the same observation in their experiments [53]. Considering the stochasticity of the optimization and the different preferences of the indicators, the configurations are of course not neccessarily all globally

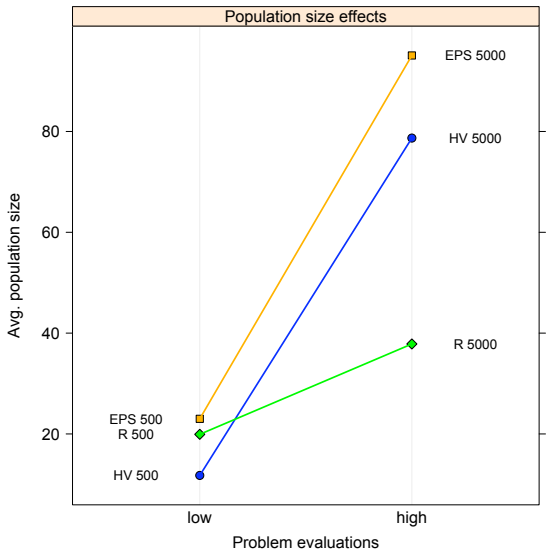


Figure 4.8: The figure shows the mean population sizes obtained with each indicator on the different run lengths. The values are averaged over all 13 problems.

optimal (if such optima exist). But, as the statistical tests show, there is some strong evidence that they are better than the default configuration, and better than at least two other optimized configurations, on the respective problem. The practitioner may use the given parameters as a direction for his search for an optimal configuration, regarding his personal quality preference.

4.3 DE Variation

The focus is now finally shifted from the quality indicators and general effects while using SPO to the tuned parameters themselves. Due to time constraints, the remaining experiments are only executed with one quality indicator. Because of the slight theoretical (strict monoticity) and practical (no reference set) advantages mentioned in Chapter 3 and the positive result of Experiment 4.1.2, hypervolume is chosen now. DE variation is considered here, because it is a popular choice for variation in MOO. Four of eight contestants in the CEC 2007 competition used differential evolution [27, 32, 60, 61]. It also has the advantages of being easier to implement than SBX and having a lower runtime in the Python implementation at hand. The default parameters for DE variation shown in Table 4.11 are chosen according to [32].

Table 4.11: Every new configuration must succeed in a performance comparison to the default parameters. The second row shows the region of interest for each parameter.

Parameter	μ	*DIFFS*	*CR*	*F*
Default Value	100	1	0.1	0.5
ROI	$\{6, \ldots, 120\}$	$\{1, 2, 3\}$	[0, 1]	[0, 2]

4.3.1 DE on OKA2

Research Question: Which parameter configurations will be found for DE variation on OKA2? How does DE compare to SBX variation?

Preexperimental planning: The DE experiments should not just repeat the SBX experiments. So, optimization with $I_{\epsilon+}$ and I_{R2} is skipped and a comparison with the SBX results is added.

Task: Again, after SPO has finished, the new DE configuration is run 50 times and evaluated with I_H. This sample is compared to the outcome of the default configuration and the optimized SBX configuration. For the latter comparison, a two-sided U-Test [25] is employed. The null hypothesis is that there is no difference in means, while the alternative hypothesis is that there is a difference. We require a significance level of 5% to reject the null hypothesis.

Setup: Tables 4.11 and 4.12 show the regions of interest and setup for the experiments with DE variation. In comparison to the previous experiments, the upper bound of μ has been reduced to save some runtime. The high population sizes were only seldom optimal anyway. The lower bound was increased to ensure the population contains enough individuals to carry out DE variation.

Table 4.12: The setups for experiments with DE variation.

Problems	Two- and three-dimensional CEC 2007 problems
SPO budget	500 algorithm runs
Algorithm initialization	Uniform random
Stopping criterion	$500 \cdot M$ and $5000 \cdot M$ problem evaluations
Algorithm	SMS-EMOA
Parameters	μ, *CR*, *F*, *DIFFS*
Initial experimental design	Latin Hypercube (50 points, 3 repeats per point)
Performance measure	I_H

Results/Visualization: Table 4.13 shows the performance results of DE and SBX variation. Table 4.14 contains the newly found DE configurations. Figure 4.9 shows some surrogate models learned from the sampled points.

Table 4.13: Performance results for the different configurations on OKA2.

Evaluations	Configuration	Min	Max	Mean	Stddev
1000	Default SBX	-0.5318	-0.4738	-0.5053	0.01267
	Opt. SBX	**-0.5694**	**-0.5160**	**-0.5438**	**0.01054**
	Default DE	-0.5313	-0.4762	-0.4976	0.01288
	Opt. DE	-0.5517	-0.4550	-0.5238	0.02093
10000	Default SBX	-0.5765	-0.5462	-0.5610	0.01018
	Opt. SBX	**-0.5839**	-0.5442	**-0.5725**	0.01103
	Default DE	-0.5671	-0.5357	-0.5480	**0.00777**
	Opt. DE	-0.5802	**-0.5470**	-0.5676	0.00971

Table 4.14: SPO results for DE variation on OKA2.

Evaluations	Configuration			
	μ	*DIFFS*	*CR*	*F*
1000	18	1	0.59	0.34
10000	32	1	0.81	0.22

Observations: Tuned SBX variation reaches significantly better mean values than tuned DE variation. At least the performance of tuned DE is significantly better than of its default configuration.

Discussion: Figure 4.9 indicates that the Kriging model for 1000 evaluations can be regarded as failed. Similar difficulties with OKA2 already appeared in Experiment 4.2.1, so it is unlikely that the problem is variation-specific. Because normalization does not overly change the problem (cf. Figure A.1), we also rule it out as cause. The remaining hypothesis is that the problem's characteristics themself are violating Kriging's prerequisites to the modeling of the data. So, Experiment 4.3.2 is scheduled to investigate if a different encoding of the parameters can aid the surrogate model. Please note that even if Kriging fails to model the data, SPO still correctly evaluates any configurations, because the model is only responsible for suggesting new configurations, not evaluating them. Also the whole initial design, which comprises 150 of the 500 algorithm runs, is unaffected.

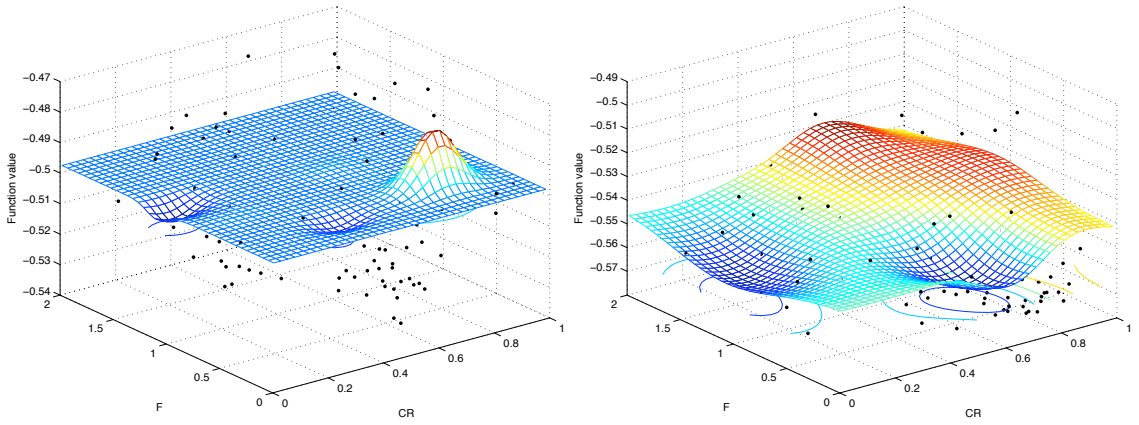


Figure 4.9: The plot on the left shows the Kriging model of F and CR obtained with 1000 evaluations, the right one with 10000. Note that F and CR are only two of the four dimensions of the parameter search space.

4.3.2 Logarithmic Representation

Research Question: Do more sophisticated representations of input and transformations of output data aid the surrogate modeling?

Preexperimental planning: Experiment 4.3.1 again showed the problems Kriging has on OKA2 with short algorithm runs. As the surrogate model is a core component that guides SPO's search, a better model would of course greatly benefit the optimization success. A first approach would be using a logarithmic representation for μ. Obviously, the hypervolume is closely coupled with the population size. The preexperimental planning for Experiment 4.1.2 already reinforced this impression. So, by setting $\mu = e^x$ and optimizing $x = \ln \mu$ with SPO we accomodate the effect that e. g. changing μ from 1 to 2 has a much greater impact than from 101 to 102. This effect is interrelated with the fact that during any optimization, the I_H value can at best converge to the hypervolume of the true Pareto-front. Therefore we can also try to make up for this by using $-\exp(|I_H|)$ as indicator value. Although the research question from Experiment 4.2.3 could be incorporated into this experiment, too, we do not expect any interesting results here because the reference point used with normalization is quite similar to the one found with Algorithm 3.2 on OKA2.

Task: The task is to minimize the variance and optimize the mean performance of SPO. A U-test [25] is employed to test the significance of performance differences.

Setup: We conceive a full factorial experimental design for the factors $A \in \{\mu, \ln(\mu)\}$ and $B \in \{I_H, -\exp(|I_H|)\}$, leading to four different setups. Each setup is run 20 times.

Results/Visualization: Figure 4.10 shows the two main effects for the factors. Table 4.15 shows the standard deviations for all the parameters.

Table 4.15: The parameters' standard deviation on the 20 runs.

No.	Factors		Mean I_H	Standard Deviation				
	A	B		I_H	μ	*DIFFS*	*CR*	F
1	μ	I_H	-0.5326	0.0025	6.1078	**0.3663**	0.2153	**0.1423**
2	$\ln(\mu)$	I_H	**-0.5347**	**0.0024**	**2.9784**	0.4104	0.2084	0.1624
3	μ	$-\exp(\lvert I_H \rvert)$	-0.5325	0.0029	3.9617	**0.3663**	**0.1749**	0.1589
4	$\ln(\mu)$	$-\exp(\lvert I_H \rvert)$	-0.5338	0.0037	3.9904	0.4702	0.2031	0.2324

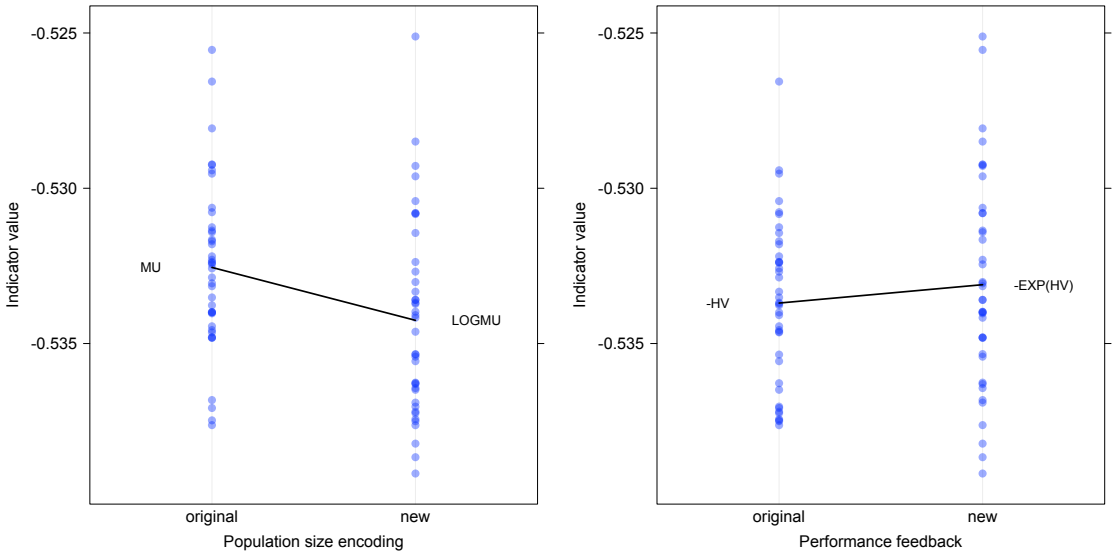


Figure 4.10: Main effects for the two factors A (left) and B (right). The black lines connect the samples' mean values.

Observations: The effect for factor A is significant at the 5% level, while the effect for B is not. Table 4.15 shows that especially configuration 2 decreases the variance of μ. It is also smaller in DE variation than in SBX (cf. Table 4.7). Factor B cannot achieve any significant improvements, whether alone or in interaction with A. To the human eye, the Kriging model is in most runs still incapable of adequately modeling the response surface.
Discussion: The insignificance of factor B might be due to the low number of problem evaluations spent. The described convergence problem does probably not emerge, because the Pareto-front is not reached anyway. With hindsight, it seems more appropriate to carry out this experiment over a range of problems, instead of doing 20 repeats on a single problem. This would rule out effects that are specific to this problem.

4.3.3 DE on SYM-PART

Research Question: Which parameter configurations will be found for DE variation on SYM-PART? How does DE compare to SBX variation?
Preexperimental planning: The series of experiments is continued without any new considerations. Because this experiment was run before Experiment 4.3.2, the logarithmic encoding of the population size is not used.
Task: The task is the same as in Experiment 4.3.1.
Setup: The setup is identical to the one in Experiment 4.3.1.
Results/Visualization: Table 4.16 shows the performance results of DE and SBX variation. Table 4.17 contains the newly found DE configurations.

Table 4.16: Performance results for the different configurations on SYM-PART.

Evaluations	Configuration	Min	Max	Mean	Stddev
1000	Default SBX	-1.1823	-1.1374	-1.1640	0.00883
	Opt. SBX	**-1.2074**	**-1.2040**	**-1.2063**	**0.00076**
	Default DE	-1.0766	-0.9891	-1.0335	0.02262
	Opt. DE	-1.2036	-1.1406	-1.1935	0.01093
10000	Default SBX	-1.2084	-1.2068	-1.2074	0.00030
	Opt. SBX	-1.2097	-1.2090	-1.2095	0.00014
	Default DE	-1.1556	-1.0930	-1.1194	0.01227
	Opt. DE	**-1.2099**	**-1.2097**	**-1.2098**	**0.00004**

Observations: Unlike in Experiment 4.2.2, there is a noticeable difference in population sizes between 1000 and 10000 evaluations. Tuned SBX variation reaches significantly better mean values than tuned DE variation for 1000 evaluations. The opposite is true for 10000

Table 4.17: SPO results for DE variation on SYM-PART.

Evaluations	Configuration			
	μ	*DIFFS*	*CR*	*F*
1000	16	1	0.65	0.56
10000	38	1	0.66	0.51

evaluations. The performance of tuned DE is also significantly better than of its default configuration.

Discussion: Generally, the first DE experiments give the impression that DE variation needs a higher population size than SBX variation to work well.

4.3.4 DE on ZDT-based Problems

Research Question: Which parameter configurations will be found for DE variation on ZDT-based problems? How does DE compare to SBX variation?

Preexperimental planning: Again, all ZDT-based problems are combined in one experiment.

Task: The task is the same as in Experiment 4.3.1.

Setup: The setup is identical to the one in Experiment 4.3.1.

Results/Visualization: Table 4.18 shows the performance results of DE and SBX variation. Table 4.19 contains the newly found DE configurations. They are all significant improvements compared to the default configurations.

Observations: On S_ZDT4, there is no significant difference between DE and SBX variation for 1000 evaluations, but the latter is better for 10000 evaluations. There are also no significant performance differences between SBX and DE variation on R_ZDT4. The found *DIFFS* values on R_ZDT4 and S_ZDT4 are identical. In half of the configurations $DIFFS = 2$ is used. Another observation is that the default SBX configuration performs much better than the default DE configuration.

Discussion: It is uncertain if the observed performance differences between DE and SBX are really meaningful because the results are very inconsistent, not only between problems, but also between run lengths. Although the respective differences are mostly statistically significant, the question arises whether SPO was able to find the optimal configurations.

4.3.5 DE on DTLZ-based Problems

Research Question: Which parameter configurations will be found for DE variation on DTLZ-based problems? How does DE compare to SBX variation?

Preexperimental planning: No new considerations were introduced into the experiment.

Table 4.18: Performance results for the different configurations on ZDT-based problems.

Problem	Evaluations	Configuration	Min	Max	Mean	Stddev
S_ZDT1	1000	Default SBX	-1.0618	-0.9840	-1.0189	0.02027
		Opt. SBX	**-1.1268**	**-1.0583**	**-1.1024**	**0.01454**
		Default DE	-0.9326	-0.8351	-0.8706	0.01930
		Opt. DE	-1.0872	-0.9780	-1.0403	0.02641
	10000	Default SBX	-1.1571	-1.1367	-1.1508	0.00434
		Opt. SBX	**-1.1709**	-1.1504	-1.1684	0.00453
		Default DE	-0.8985	-0.8540	-0.8755	0.01154
		Opt. DE	-1.1700	**-1.1587**	**-1.1686**	**0.00197**
S_ZDT2	1000	Default SBX	-0.9967	**-0.9019**	-0.9488	**0.02282**
		Opt. SBX	**-1.0419**	-0.8711	**-0.9878**	0.04184
		Default DE	-0.8200	-0.7037	-0.7451	0.02750
		Opt. DE	-0.9543	-0.7686	-0.9037	0.03321
	10000	Default SBX	-1.0861	-1.0580	-1.0668	**0.00560**
		Opt. SBX	**-1.1319**	**-1.0931**	**-1.1271**	0.00854
		Default DE	-0.8095	-0.6998	-0.7394	0.02330
		Opt. DE	-1.1219	-1.0693	-1.0928	0.01842
S_ZDT4	1000	Default SBX	-1.0227	-0.8919	-0.9505	0.02662
		Opt. SBX	-1.1122	-0.8941	-1.0407	0.04307
		Default DE	-0.9068	-0.8047	-0.8489	**0.02526**
		Opt. DE	**-1.1219**	**-0.9936**	**-1.0546**	0.02732
	10000	Default SBX	-1.1476	-1.0918	-1.1241	0.01532
		Opt. SBX	**-1.2071**	**-1.1896**	**-1.2029**	**0.00303**
		Default DE	-0.8948	-0.7889	-0.8441	0.01951
		Opt. DE	-1.1860	-1.1376	-1.1654	0.01072
R_ZDT4	1000	Default SBX	-1.1298	-1.0617	-1.0994	**0.01834**
		Opt. SBX	**-1.1876**	-1.0232	-1.1214	0.03894
		Default DE	-1.1210	-1.0179	-1.0605	0.02170
		Opt. DE	-1.1784	**-1.0695**	**-1.1350**	0.02377
	10000	Default SBX	-1.2037	-1.1477	-1.1908	0.01021
		Opt. SBX	-1.2040	-1.1741	**-1.1939**	0.00644
		Default DE	-1.1166	-1.0594	-1.0845	0.01538
		Opt. DE	**-1.2043**	**-1.1824**	-1.1923	**0.00585**
S_ZDT6	1000	Default SBX	-0.7601	-0.7128	-0.7340	0.01063
		Opt. SBX	**-0.7887**	**-0.7264**	**-0.7592**	0.01567
		Default DE	-0.6697	-0.6392	-0.6573	**0.00653**
		Opt. DE	-0.7213	-0.6745	-0.7013	0.01074
	10000	Default SBX	-0.8848	-0.8530	-0.8658	0.00713
		Opt. SBX	-0.9460	-0.7711	-0.9293	0.02350
		Default DE	-0.6673	-0.6503	-0.6584	**0.00409**
		Opt. DE	**-0.9799**	**-0.9058**	**-0.9574**	0.01577

Table 4.19: SPO results for DE variation on ZDT-based problems.

Problem	Evaluations	Configuration			
		μ	*DIFFS*	*CR*	*F*
S_ZDT1	1000	12	1	0.29	0.54
	10000	56	2	0.36	0.25
S_ZDT2	1000	23	1	0.14	1.38
	10000	58	1	0.20	0.78
S_ZDT4	1000	19	2	0.47	0.13
	10000	53	1	0.20	0.78
R_ZDT4	1000	22	2	0.17	0.49
	10000	88	1	0.19	0.31
S_ZDT6	1000	32	2	0.59	0.11
	10000	27	2	0.31	0.48

Task: The task is the same as in Experiment 4.3.1.
Setup: The setup is identical to the one in Experiment 4.3.1.
Results/Visualization: Table 4.20 shows the performance results of DE and SBX variation. Table 4.21 contains the newly found DE configurations. Figure 4.11 gives an example of typical effects with DE variation.
Observations: All differences between optimized DE and SBX variation and between the optimized configurations and the respective default configurations are significant. DE is only better on the rotated problem. Some mean values of configurations with 15000 evaluations on S_DTLZ2 exceed the hypervolume of the reference set. This is possible, because the reference set is designed to provide a good coverage of the Pareto-front and not the maximal hypervolume (seed Table A.1 and Figure A.7). Crossover constant *CR* and scaling factor F are both extraordinarily small on S_DTLZ3.
Discussion: The three problems seem to be surprisingly easy to optimize, as the obtained hypervolume values are very near to the values of the reference set. S_DTLZ2 seems to be an easier problem than R_DTLZ2, because the SMS-EMOA obtains better indicator values with every variation there.

4.3.6 DE on WFG Problems

Research Question: Which parameter configurations will be found for DE variation on WFG problems? How does DE compare to SBX variation?
Preexperimental planning: Thanks to the reduced population size and DE variation being faster than SBX variation, the whole SPO budget of 500 algorithm runs could be

Table 4.20: Performance results for the different configurations on DTLZ-based problems.

Problem	Evaluations	Configuration	Min	Max	Mean	Stddev
S_DTLZ2	1500	Default SBX	-1.3285	-1.3222	-1.3270	0.00127
		Opt. SBX	**-1.3297**	-1.3189	**-1.3291**	0.00151
		Default DE	-1.3237	-1.3091	-1.3189	0.00302
		Opt. DE	-1.3297	**-1.3266**	-1.3290	**0.00051**
	15000	Default SBX	-1.33016	-1.33010	-1.33013	0.011e-3
		Opt. SBX	**-1.33022**	**-1.33020**	**-1.33021**	**0.004e-3**
		Default DE	-1.32391	-1.32104	-1.32235	0.569e-3
		Opt. DE	-1.33021	-1.33013	-1.33020	0.016e-3
R_DTLZ2	1500	Default SBX	-1.3240	-1.2894	-1.3100	0.00859
		Opt. SBX	-1.3271	-1.2891	-1.3204	0.00802
		Default DE	-1.3103	-1.2040	-1.2531	0.02416
		Opt. DE	**-1.3277**	**-1.3154**	**-1.3243**	**0.00257**
	15000	Default SBX	-1.32982	-1.32776	-1.32960	0.315e-3
		Opt. SBX	-1.32995	-1.32981	-1.32989	**0.031e-3**
		Default DE	-1.29346	-1.22876	-1.26384	13.336e-3
		Opt. DE	**-1.33006**	**-1.32986**	**-1.32997**	0.042e-3
S_DTLZ3	1500	Default SBX	-1.3228	-1.3113	-1.3165	0.00271
		Opt. SBX	**-1.3309**	**-1.3273**	**-1.3304**	**0.00068**
		Default DE	-1.3131	-1.2961	-1.3054	0.00461
		Opt. DE	-1.3293	-1.3171	-1.3257	0.00257
	15000	Default SBX	-1.33084	-1.32975	-1.33057	0.226e-3
		Opt. SBX	**-1.33100**	**-1.33094**	**-1.33099**	**0.014e-3**
		Default DE	-1.30638	-1.29560	-1.30284	2.207e-3
		Opt. DE	**-1.33100**	-1.33092	-1.33097	0.021e-3

Table 4.21: SPO results for DE variation on DTLZ-based problems.

Problem	Evaluations	Configuration			
		μ	*DIFFS*	*CR*	*F*
S_DTLZ2	1500	22	1	0.38	0.26
	15000	120	1	0.62	0.09
R_DTLZ2	1500	38	1	0.84	0.26
	15000	86	1	0.69	0.20
S_DTLZ3	1500	28	3	0.04	0.08
	15000	80	1	0.07	0.02

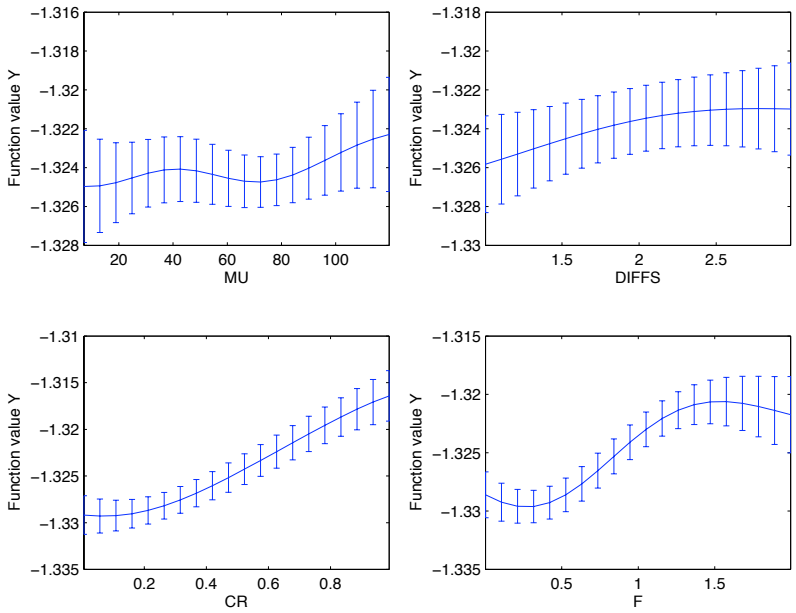


Figure 4.11: The main effects with 15000 evaluations on S_DTLZ2. The errorbars depict 90% confidence intervals.

applied to the WFG problems in this experiment. Please recall that this was not possible in Experiment 4.2.7.

Task: The task is the same as in Experiment 4.3.1.

Setup: The setup is identical to the one in Experiment 4.3.1.

Results/Visualization: Table 4.22 shows the performance results of DE and SBX variation. Table 4.23 contains the newly found DE configurations.

Observations: Except for WFG1 with 15000 evaluations, SBX is always significantly better than DE variation. The only DE configuration that wins against its SBX rival

Table 4.22: Performance results for the different configurations on WFG problems.

Problem	Evaluations	Configuration	Min	Max	Mean	Stddev
WFG1	1500	Default SBX	-0.9170	-0.8971	-0.9051	**0.00463**
		Opt. SBX	**-1.0007**	**-0.9798**	**-0.9936**	0.00463
		Default DE	-0.9458	-0.8575	-0.8677	0.01235
		Opt. DE	-0.9270	-0.8926	-0.9084	0.00762
	15000	Default SBX	-0.9741	-0.9597	-0.9684	**0.00341**
		Opt. SBX	-1.0493	**-1.0126**	-1.0370	0.00752
		Default DE	-0.9078	-0.8826	-0.8929	0.00545
		Opt. DE	**-1.1075**	-0.9859	**-1.0633**	0.02513
WFG8	1500	Default SBX	-1.1383	-1.0694	-1.1101	0.01352
		Opt. SBX	**-1.2377**	**-1.2013**	**-1.2231**	**0.00870**
		Default DE	-1.1357	-1.0704	-1.1062	0.01512
		Opt. DE	-1.2204	-1.1748	-1.1985	0.00911
	15000	Default SBX	-1.2313	-1.2083	-1.2197	0.00493
		Opt. SBX	**-1.2753**	**-1.2623**	**-1.2707**	**0.00258**
		Default DE	-1.2586	-1.1520	-1.2003	0.04597
		Opt. DE	-1.2689	-1.2536	-1.2615	0.00277
WFG9	1500	Default SBX	-1.2126	-1.1204	-1.1651	0.02048
		Opt. SBX	**-1.2358**	**-1.1667**	**-1.2147**	**0.01315**
		Default DE	-1.2075	-1.1062	-1.1474	0.02407
		Opt. DE	-1.2341	-1.1293	-1.2012	0.01921
	15000	Default SBX	-1.2596	-1.2250	-1.2459	0.00679
		Opt. SBX	-1.2663	**-1.2477**	**-1.2614**	**0.00346**
		Default DE	-1.2048	-1.1684	-1.1894	0.00738
		Opt. DE	**-1.2700**	-1.2376	-1.2607	0.00970

Table 4.23: SPO results for DE variation on WFG problems.

Problem	Evaluations	Configuration			
		μ	*DIFFS*	*CR*	*F*
WFG1	1500	26	2	0.38	0.09
	15000	24	1	0.28	0.25
WFG8	1500	19	2	0.25	0.25
	15000	72	1	0.23	0.20
WFG9	1500	47	2	0.46	0.10
	15000	96	2	0.23	0.08

features a rather small population size. The default DE configuration again performs clearly worse than any other. Like on ZDT-based problems, two vector differences are used in the majority of cases.

Discussion: Interestingly, SBX variation is more successful although less resources were spent on optimizing it. This is another hint that SPO's optimization phase is not very successful in MOO.

4.3.7 Summary

The experiments show that the decision which variation is chosen is less important than the decision to tune the chosen variation operator. The differences between the default and optimized configurations are much bigger than between different optimized configurations. For $500 \cdot M$ problem evaluations, SBX is better than DE on ten problems, while the opposite is true on only one problem (there are two ties). For $5000 \cdot M$ evaluations, SBX wins seven times and DE five times (there is one tie). So, SBX is winning more often, but that does not mean a lot. First, the set of problems is probably biased, because there are many separable problems and only few rotated ones on which DE proved to be better. Second, the result is more balanced on the longer runs, so it would be interesting if DE becomes superior if the run length is extended even furter.

The default DE configuration almost always performs worst. This is surprising, because the configuration was used by Kukkonen and Lampinen [32] in the CEC 2007 competition. However, the authors state that they chose a configuration that performs well on average over all problems, while we tuned it for every problem separately. Additionally, the configuration was designed for the GDE 3 algorithm [32], while we used it with the SMS-EMOA. On the other hand, SBX was challenged in the same way. It is also remarkable that DE literature seems to be quite focused on parameter setting [32] and benchmark results are often obtained with a special DE configuration for each problem [56]. In contrast, the SBX

configurations given by Sharma et al. [52] for the CEC 2007 competition are quite similar to the default configuration used in our experiments, which was taken from [15]. So, we suppose that DE variation is less robust than SBX variation. It is probably highly recommended to use a (self-)adaptive variant of DE variation [27, 60, 61], which also entered the CEC 2007 contest.

The average population size for $500 \cdot M$ evaluations with DE is 25, which is slightly higher than the average for SBX variation. For $5000 \cdot M$ evaluations, the average μ is 64, which is in the middle of the SBX averages (cf. Figure 4.8).

4.4 Selection

Variation is the SMS-EMOA's component that most obviously lends itself to optimization. Nonetheless, selection also contains some less well-known parameters. One issue is a selection variant introduced by Naujoks, Beume and Emmerich [40], which only uses $\mathcal{S}$-metric selection when the population contains just one non-dominated front. In the other case, an individual's fitness is determined by the number of individuals that dominate it. This selection is called dominating points (DP) selection in the remainder. It assigns every point a rank which is equal to the number of other points that dominate it. The fitness is the higher the lower the rank is.

Another parameter hides in the constant offset $(1, \ldots, 1)^T$ that is used in the reference point construction. Therefore, we use Algorithm 4.1, which is only a slight variation of Algorithm 2.2, to make this parameter variable. We can even have a different offset in each dimension.

Algorithm 4.1 constructReferencePoint(Q, $\mathbf{o}$)

1: $\mathbf{w} \leftarrow \text{first}(Q)$ // set the vector of worst objective values to the first point of Q
2: **for all** $q_i \in Q$ **do**
3: **for** $j = 1$ to $|q_i|$ **do** // for every dimension
4: $w_j \leftarrow \max\{w_j, q_{ij}\}$ // set w_j to the worse value
5: **end for**
6: **end for**
7: **return** $\mathbf{w} + \mathbf{o}$ // return the reference point

4.4.1 Selection Variants

Research Question: Which influence do selection parameters have on the SMS-EMOA's performance?

Preexperimental planning: In a first attempt, it was tried to optimize the selection parameters with SPO. Doing this, two problems arise: First, the decision for or against

using the number of dominating points for selection is binary and thus not suited for Kriging. Second, no significant influence of the offsets in the reference point construction could be found. Figure 4.12 shows the main effects on S_DTLZ3. So, a full factorial experimental design with two factors is also carried out separately from the SPO runs.

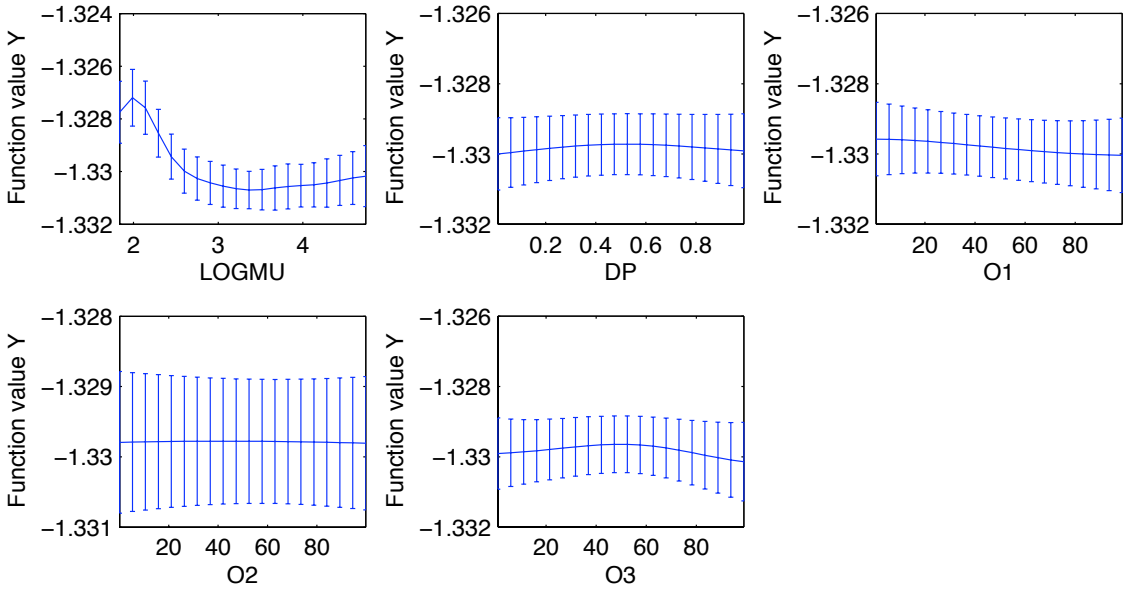


Figure 4.12: The main effects of selection parameters on S_DTLZ3. The errorbars depict 90% confidence intervals.

Task: The SPO results are not verified, because the configurations would be probably evaluated as improvement simply through the optimization of μ. The separate factorial experiment is expected to provide more information. The main effects are tested with a two-sided U-Test [25]. The significance level is 5%.

Setup: Table 4.24 shows the default selection parameters and the region of interest for the configuration in this experiment. Additionally, SBX variation is used here, because it seemed to be more robust than DE in Section 4.3. The SPO runs are carried out with SBX default parameters, hoping that selection is fairly independent from variation.

The factorial experiment uses the respective SBX configuration optimized by I_H for each problem (see Section 4.2). Each of those configurations is extended by the new selection parameters. The boundaries of the ROI are used as low and high levels of the factors. Factor $DP \in \{0, 1\}$ defines if the number of dominating points is used for selection, factor $\vec{o}$ controls the offset vector for reference point construction. It can be either $(1, \ldots, 1)^T$ or $(100, \ldots, 100)^T$. Every configuration is run 50 times with $5000 \cdot M$ problem evaluations.

Results/Visualization: Using the dominating points for selection leads to a significantly worse performance on S_ZDT1 and S_DTLZ3, which is shown in Figure 4.13. There is no effect on the other problems. An offset of $(100, \ldots, 100)^T$ produces significantly better

Table 4.24: The default configuration and the parameters' region of interest. The $\ln(\mu)$ values correspond to $\mu = 100$ as default value and a region of interest of $\{6, \ldots, 120\}$.

Parameter	$\ln(\mu)$	DP	o_1	o_2	o_3
Default Value	4.60517	0	1	1	1
ROI	[1.79, 4.79]	$\{0, 1\}$	[0, 100]	[0, 100]	[0, 100]

results than $(1, \ldots, 1)^T$ on S_ZDT4, S_DTLZ2, WFG1, WFG8 and WFG9, while there is no difference elsewhere. Figure 4.14 shows the effect on S_ZDT4 and WFG1. For the sake of completeness, the SPO results are reported in Table 4.25.

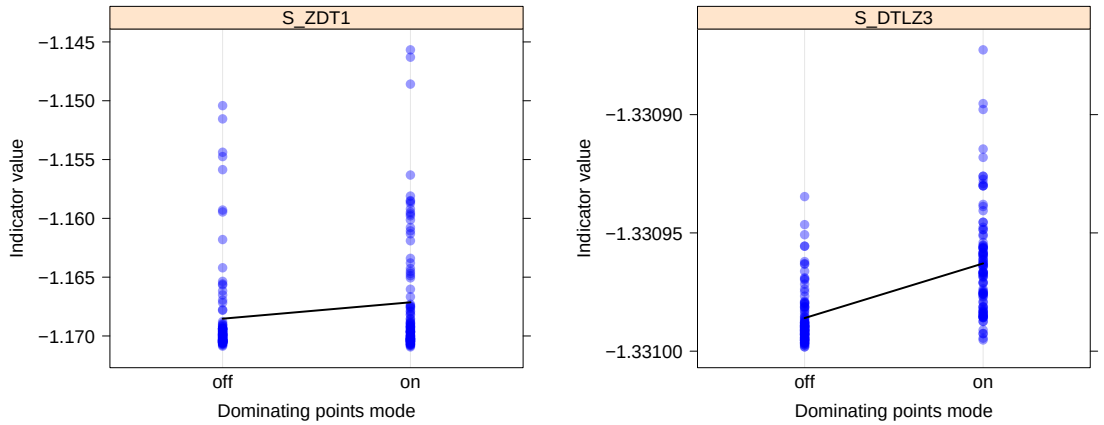


Figure 4.13: The main effects of the two problems where *DP* yields significantly worse results. The black lines connect the samples' mean values.

Observations: The bigger offsets have only positive effects, especially on the three-dimensional problems. Dominating points selection cannot yield any improvements, but it also produces worse results on only two of the 13 problems.

The population sizes found by SPO tend to be smaller than the previously obtained ones in the SBX experiments, but there is a noticeable correlation of 0.90 between the values (p-value = 2.4e-5). The SPO results for *DP* seem arbitrary and for S_DTLZ3 also inconsistent with the result from the factorial experiment.

Discussion: Dominating points selection proves to be a reasonable alternative, reducing runtime without losing much quality. With hindsight, the SPO runs should have been carried out with a fixed μ, so that the effects of the remaining parameters stood out more clearly. On the other hand, it is interesting to see that the resulting population sizes are quite similar to previously found ones, although the variation was not optimized, the region of interest was different and the logarithmic encoding was used here. So, we have a nice

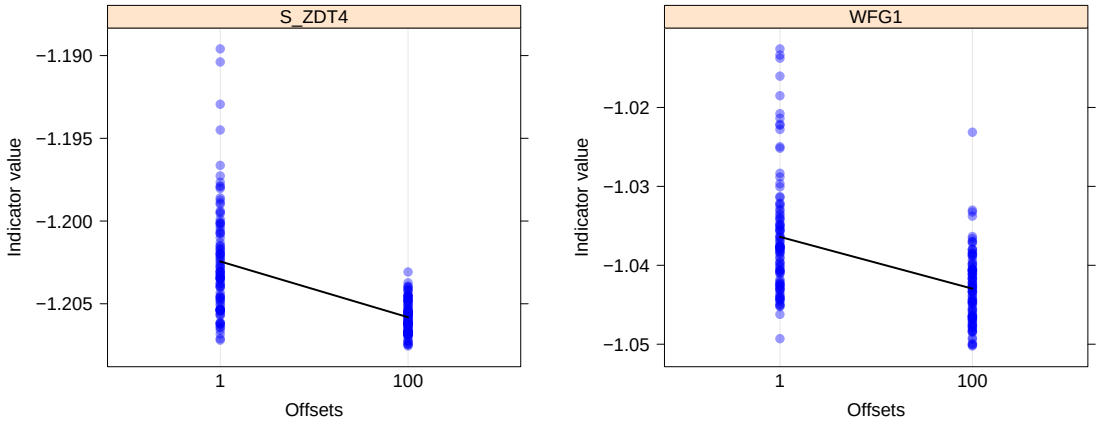


Figure 4.14: The main effects of the two clearest effects of $\vec{o}$. The remaining three problems with significant effects are not shown here.

validation of our results of Experiment 4.3.2, showing that the logarithmic encoding works with SBX and over a range of all problems as well.

The upper bound for the offsets probably should have been higher, perhaps even in the thousands, because some problems (e. g. S_DTLZ3, see Table A.1) have feasible objective spaces with such dimensions. Optimizing *DP* with SPO did not work. Treating each offset-dimension o_i individually did not provide any additional insight yet, either. Further experiments are required to draw any conclusions here. Using higher offsets seems to be very promising, though. On some problems, it has a remarkable positive effect. Please note that the SPO runs are not directly comparable to the factorial experiment, because different variation parameters were used. Analogously, it was not expected to obtain population sizes identical to experiments in Section 4.2 (see Table 4.25).

Table 4.25: The selection parameters obtained with SPO. In the last column, the μ values from Section 4.2 are repeated for a comparison.

Problem	Configuration					μ in Section 4.2
	μ	DP	o_1	o_2	o_3	
OKA2	50	1	73.3	14.8	–	107
SYM-PART	6	1	86.6	67.0	–	9
S_ZDT1	28	0	76.9	69.2	–	39
S_ZDT2	7	0	38.3	5.6	–	18
S_ZDT4	6	0	66.2	0.03	–	7
R_ZDT4	69	1	15.4	46.2	–	196
S_ZDT6	6	1	9.8	12.5	–	16
S_DTLZ2	118	0	78.6	2.3	84.3	172
R_DTLZ2	68	1	78.1	48.6	78.0	179
S_DTLZ3	28	1	32.3	90.8	28.0	83
WFG1	16	1	10.4	88.4	13.8	30
WFG8	28	0	82.5	46.1	31.9	15
WFG9	77	1	11.1	0.1	74.5	152

Chapter 5

Summary and Outlook

5.1 Summary

This work motivated and demonstrated parameter optimization for evolutionary multi-objective algorithms, an area which is still in its beginnings. To this end, an introduction to multi-objective optimization and to evolutionary algorithms was given. Central point of the work was the SMS-EMOA, an algorithm that has already proven its capabilities in several applications [17, 40].

This work also contains a very comprehensive part of experimental results, which gives valuable information for further development of the SMS-EMOA and Sequential Parameter Optimization (SPO) itself. The parameter optimization was executed on a range of 13 different problems with two variation operators and two algorithm run lengths. Experiments used statistical hypothesis testing and were executed and structured according to up-to-date guidelines for experiments in evolutionary computation [43, 2]. The use of SPO was justified and explained. In total, well over 400 single SPO runs, with durations between one hour and several weeks, were executed on a batch system. The use of the interpreted Python programming language emerged as a problem here, because of its inferior execution speed compared to machine languages. The results obtained in the experiments with the CEC 2007 benchmark problems can be summarized as follows:

- The population size usually has a great influence on the observed performance and a great diversity of values could be observed. However, the results are dependent on the used quality indicator and the algorithm run length. Additionally, the decision maker may have his own needs, too. So, the results' value is probably limited.

- The hypervolume indicator I_H is more successful than $I_{\epsilon+}$ and I_{R2} for optimizing the SMS-EMOA with Simulated Binary Crossover (SBX) as variation (compare Experiment 4.1.2).

- Over all problems, SBX variation is more successful than Differential Evolution (DE), but DE variation performs better on rotated problems. It is more important to tune the chosen variation operator than to choose between SBX and DE. DE seems to be more sensitive to its parameter setting (compare Section 4.3).

- Experimental setup is not a trivial task. Any reference data for evaluation has to be chosen carefully, so that differences are measurable. The population size should be encoded in a logarithmic fashion, to overcome problems in the modeling of results. The CEC 2007 evaluation approach does not work on SYM-PART and S_DTLZ3, because the reference point is too far away from the Pareto-front. In this case, the indicator's ability to reward a good distribution of the approximation set vanishes.

- The influence of offsets for reference point construction on the performance was investigated for the first time. Higher offsets are often better, especially on three-dimensional problems. The effects seem to be difficult to find with SPO, though (compare Section 4.4).

- Dominating points selection performs slightly worse than $\mathcal{S}$-Metric selection, but often there is no measurable difference. This is consistent with a previous result [40]. Because of its greatly reduced runtime compared to $\mathcal{S}$-Metric selection, it is a favorable option, especially for higher numbers of objectives (compare Section 4.4).

5.2 Outlook

Still, many of the experiments could be extended or improved to obtain more general results. This encompasses running the experiments on a wider range of problems or with more repeats. It would also be possible to combine some experimental designs that were handled separately in this thesis, because the research questions emerged at different times. There also seems to be some room left for performance improvements in SPO, because of the problems with Kriging. In the long term, the question comes up if there is a possibility to develop a tuning procedure that can get along without unary quality indicators. Currently, the already existing quality indicators are used rather as a workaround to map the resulting multi-objective populations onto a one-dimensional utility measure. This has the advantage of modularity, but it seems to be quite prone to misconfigurations. It would be interesting if there are simpler, integrated approaches possible.

Regarding the variation concepts, the study could be continued with the (self-)adaptive variants of SBX and DE, investigating if they can be tuned as well and if they yield any improvements over their basic versions. On the algorithmic side, the experimental setup could be easily modified to allow multiple starts of the investigated optimization algorithm. To do this, an integer parameter for the number of runs would be added to the

algorithm design and the fixed budget of problem evaluations would be split between the runs. The resulting populations would then be merged and collectively evaluated by the quality indicator, representing one repeat of the configuration from SPO's point of view. So, no modification of SPO is neccessary for this approach.

Appendix A

Test Problems

In the following the test problems used in the experiments are defined. Table A.1 gives an overview of the parameters used on the respective problem instances. The objective space for each objective function is normalized to $[1, 2]$, according to the CEC 2007 contest rules [26]. For each problem, a reference set of Pareto-optimal points is provided to enable evaluation of populations with the ϵ_+ indicator and the $R2$ indicator.

A.1 Individual Problems

A.1.1 Definition (OKA2). The test problem OKA2 [42] is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= x_1, \\
f_2(\mathbf{x}) &= 1 - \frac{1}{4\pi^2}(x_1 + \pi)^2 + |x_2 - 5\cos(x_1)|^{\frac{1}{3}} + |x_3 - 5\sin(x_1)|^{\frac{1}{3}},
\end{aligned}
$$

with $x_1 \in [-\pi, \pi]$ and $x_2, x_3 \in [-5, 5]$. Solutions $(x_1, x_2, x_3) = (\xi, 5\cos(\xi), 5\sin(\xi))$ are Pareto-optimal for $\xi \in [-\pi, \pi]$. Figure A.1 shows the reference set for the problem. The Pareto-front is concave. OKA2 is the only problem in this set that has a fixed number of decision variables.

A.1.2 Definition (SYM-PART). This test problem is a modification of the original SYM-PART problem [46] to raise the number of decision variables from two to 30. The definition below differs from the one in the CEC 2007 paper, but is consistent with the competition's implementation.

$$
\begin{aligned}
f_1(\mathbf{x}) &= \Big((x'_1 + a - t_1 c_2)^2 + (x'_2 - t_2 b)^2 + \cdots + (x'_{D-1} + a - t_1 c_2)^2 + (x'_D - t_2 b)^2\Big)/|\mathbf{x}|, \\
f_2(\mathbf{x}) &= \Big((x'_1 - a - t_1 c_2)^2 + (x'_2 - t_2 b)^2 + \cdots + (x'_{D-1} - a - t_1 c_2)^2 + (x'_D - t_2 b)^2\Big)/|\mathbf{x}|,
\end{aligned}
$$

Table A.1: Test problems in the CEC 2007 suite.

Test problem	No. of objectives	No. of variables	Reference set size	Objective space bounds		I_H of reference set (normalized)	Pareto-front shape
				lower	upper		
OKA2	2	3	300	$(-4,-4)^T$	$(5,5)^T$	-0.6067	concave
SYM-PART	2	30	300	$(0,0)^T$	$(500,500)^T$	-1.2100	convex
S_ZDT1	2	30	300	$(1,1)^T$	$(2,10)^T$	-1.1728	convex
S_ZDT2	2	30	300	$(1,1)^T$	$(2,10)^T$	-1.1358	concave
S_ZDT4	2	30	300	$(1,1)^T$	$(3,1000)^T$	-1.2098	convex
R_ZDT4	2	10	300	$(1,1)^T$	$(4,500)^T$	-1.2098	convex
S_ZDT6	2	30	300	$(1,1)^T$	$(3,20)^T$	-1.0417	concave
S_DTLZ2	3	30	500	$(1,1,1)^T$	$(10,10,10)^T$	-1.3301	concave
S_DTLZ2	5	30	1000	$(1,1,1,1,1)^T$	$(10,10,10,10,10)^T$	-1.6105	concave
R_DTLZ2	3	30	500	$(1,1,1)^T$	$(10,10,10)^T$	-1.3301	concave
R_DTLZ2	5	30	1000	$(1,1,1,1,1)^T$	$(10,10,10,10,10)^T$	-1.6105	concave
S_DTLZ3	3	30	500	$(1,1,1)^T$	$(6000,6000,6000)^T$	-1.3310	concave
S_DTLZ3	5	30	1000	$(1,1,1,1,1)^T$	$(6000,\ldots,6000)^T$	-1.6105	concave
WFG1	3	24	500	$(0,0,0)^T$	$(10,10,10)^T$	-1.3031	convex
WFG1	5	28	1000	$(0,0,0,0,0)^T$	$(10,10,10,10,12)^T$	-1.5777	convex
WFG8	3	24	500	$(0,0,0)^T$	$(10,10,10)^T$	-1.3030	concave
WFG8	5	28	1000	$(0,0,0,0,0)^T$	$(10,10,10,10,12)^T$	-1.5997	concave
WFG9	3	24	500	$(0,0,0)^T$	$(10,10,10)^T$	-1.2029	concave
WFG9	5	28	1000	$(0,0,0,0,0)^T$	$(10,10,10,10,12)^T$	-1.3659	concave

where

$$\mathbf{x}' = \begin{pmatrix} \cos\omega & -\sin\omega & 0 & 0 & \\ \sin\omega & \cos\omega & 0 & 0 & \\ 0 & 0 & \cos\omega & -\sin\omega & \dots \\ 0 & 0 & \sin\omega & \cos\omega & \\ & & \vdots & & \ddots \end{pmatrix} \mathbf{x},$$

$$t_1 = \operatorname{sgn}\left(\operatorname{sgn}(x_1') \cdot \left\lceil \frac{|x_1'| - c_2/2}{c_2} \right\rceil\right), \quad t_2 = \operatorname{sgn}\left(\operatorname{sgn}(x_2') \cdot \left\lceil \frac{|x_2'| - b/2}{b} \right\rceil\right),$$

$$a = 1, \quad b = 10, \quad c = 8, \quad c_2 = c + 2a = 10.$$

Figure A.2 shows the reference set for the problem. The Pareto-front is convex. Thanks to the signum functions in t_1 and t_2, the problem has nine areas of attraction in the decision space that yield the same fitness.

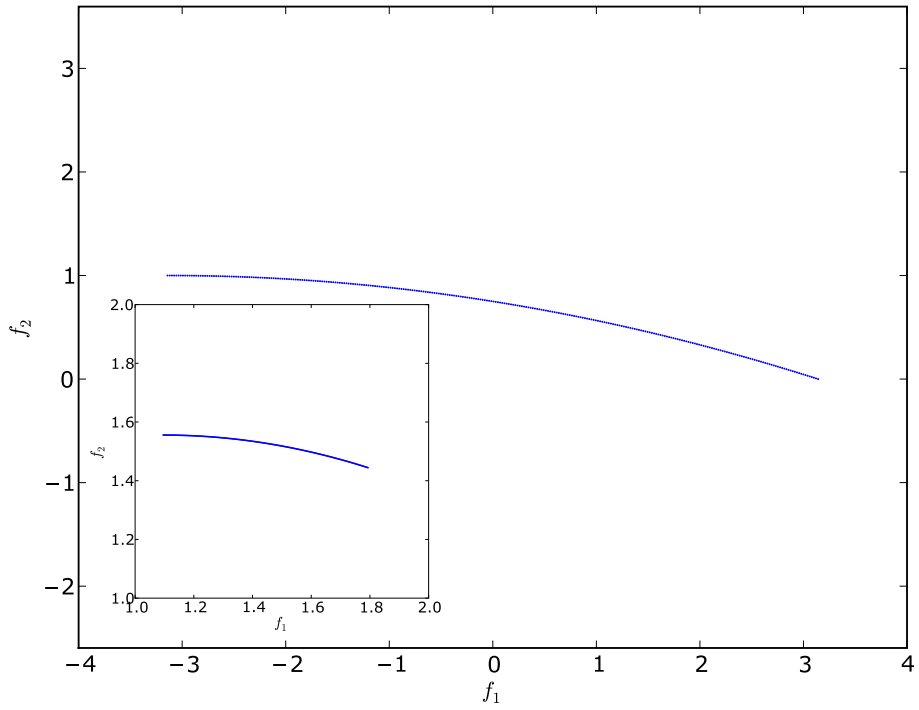


Figure A.1: The reference set for OKA2. The small image in the bottom left corner shows the reference set after normalization to the interval $[1, 2]$.

A.2 Extended ZDT and DTLZ Problems

Table A.2 shows an overview of vectors and matrices involved in the problems mentioned in this section. These problems were derived from ZDT [63] and DTLZ [15] problems by shifting ($\mathbf{z} = \mathbf{x} - \mathbf{o}$) or rotating ($\mathbf{z} = \mathbf{M}\mathbf{x}$) the search space. $\mathbf{M}$ and $\mathbf{o}$ contain fairly noisy

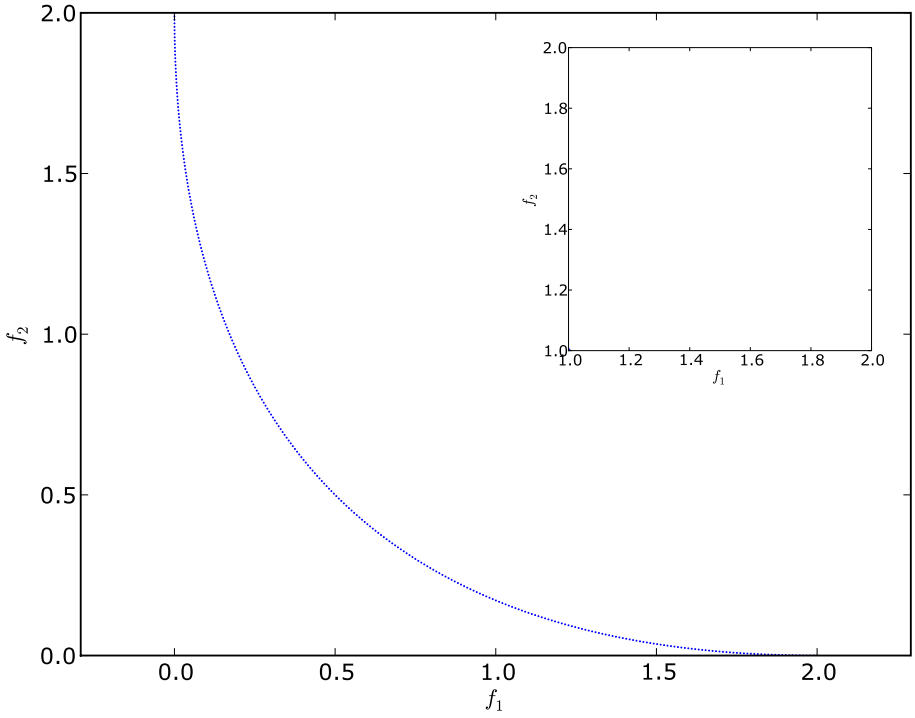


Figure A.2: The reference set for SYM-PART. In the upper right corner you can see that after normalization, all points lie very close to $(1, 1)$ and are therefore not visible.

data to obfuscate the interconnections in the problem. The data (and a C implementation of all 19 problems) can be obtained from the contest's website [26]. Moreover, all extended problems have in common a stretching function $S(p) = 2/(1 + e^{-p})$, where $p = \sqrt{\sum_{i \in I} p_i^2}$ and $I \subseteq \{1, \ldots, D\}$ is the set of indices of all decision variables used in the particular objective function.

A.2.1 Definition (S_ZDT1). The test problem S_ZDT1 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(p_1)(z_1' + 1), \\
f_2(\mathbf{x}) &= S\left(\sqrt{\sum_{i=1}^{D} p_i^2}\right)\left(g(\mathbf{x})\left(1 - \sqrt{z_1'/g(\mathbf{x})}\right) + 1\right), \\
g(\mathbf{x}) &= 1 + 9 \cdot \frac{\sum_{i=2}^{D} z_i'}{D - 1},
\end{aligned}
$$

where $z_i' = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}$, $p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}$, $i = 1, \ldots, D$.

The Pareto-optimal solutions have values of $x_1 \in [o_1, 1 + o_1]$ and $x_i = o_i$ for $i = 2, \ldots, D$. Figure A.3 shows the reference set for the problem. The convex shape of the Pareto-front is determined by the term $\sqrt{z_1'/g(\mathbf{x})}$ in f_2. ZDT-based problems always have two objectives.

Table A.2: Variable names used in all extended problems.

Variable	Purpose
$\mathbf{x}$	Genome on which the EA operates
$\mathbf{o}$	Offsets for shifting the search space
$\mathbf{M}$	Rotation matrix for rotating the search space
$\mathbf{z}, \mathbf{z}'$	Calculated from the genome
$\mathbf{d}$	Extended lengths of the lower bounds
λ	Scale factors
$\mathbf{p}$	Penalty values
$\mathbf{x}_{\min}$	Lower bounds (search space)
$\mathbf{x}_{\max}$	Upper bounds (search space)
D	Number of decision variables
M	Number of objectives

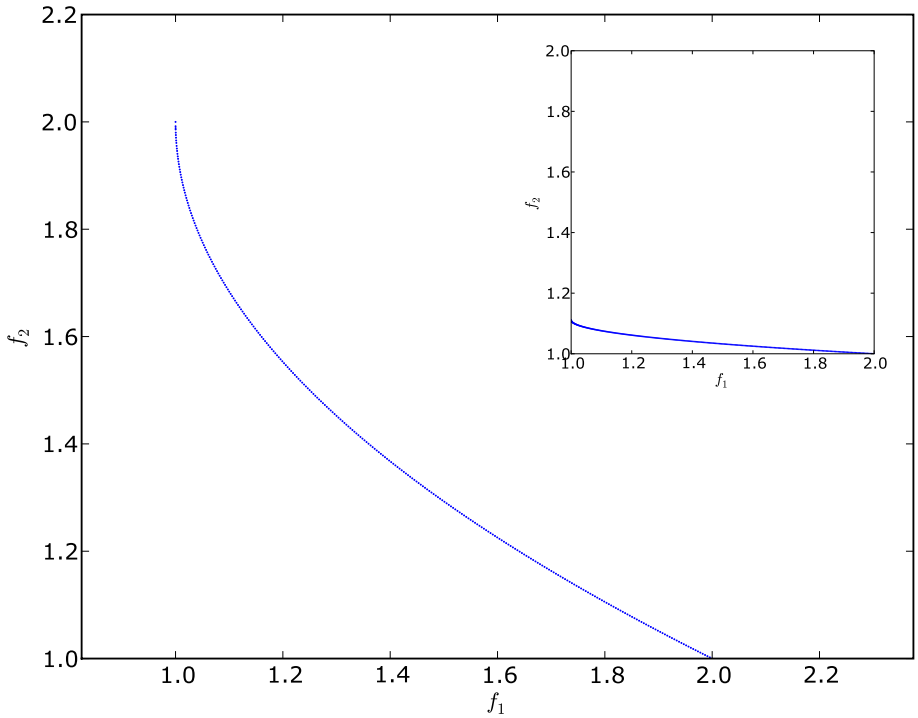


Figure A.3: The reference set for S_ZDT1.

A.2.2 Definition (S_ZDT2). The test problem S_ZDT2 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(p_1)(z_1' + 1), \\
f_2(\mathbf{x}) &= S\left(\sqrt{\sum_{i=1}^{D} p_i^2}\right)\left(g(\mathbf{x})\left(1 - (z_1'/g(\mathbf{x}))^2\right) + 1\right), \\
g(\mathbf{x}) &= 1 + 9 \cdot \frac{\sum_{i=2}^{D} z_i'}{D-1},
\end{aligned}
$$

where $z_i' = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}$, $p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}$, $i = 1, \ldots, D$.

The Pareto-optimal solutions have values of $x_1 \in [o_1, 1 + o_1]$ and $x_i = o_i$ for $i = 2, \ldots, D$. Figure A.4 shows the reference set for the problem. The Pareto-front is concave.

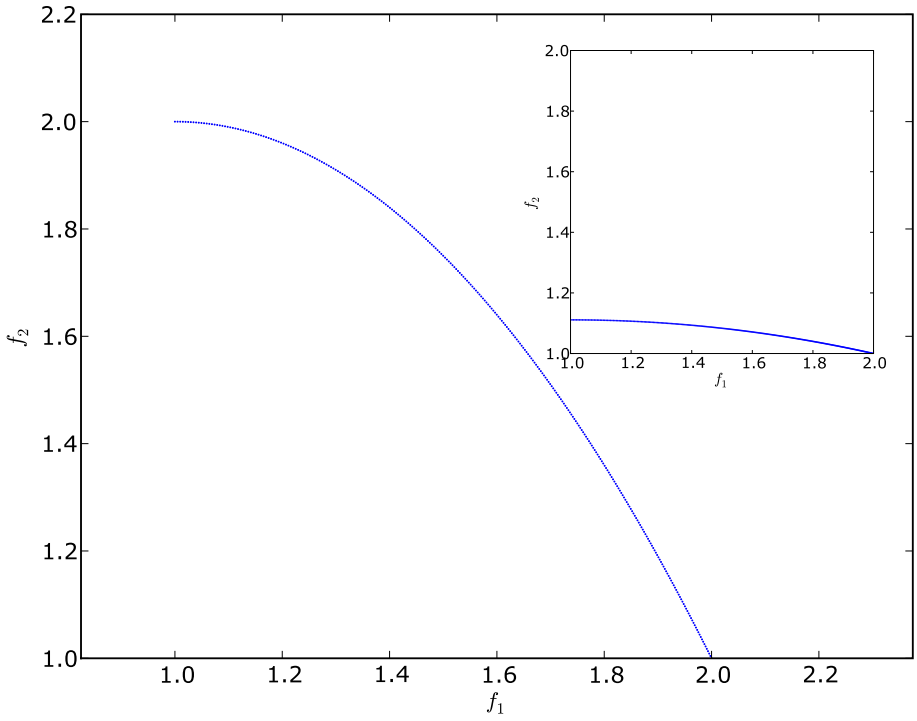


Figure A.4: The reference set for S_ZDT2.

A.2.3 Definition (S_ZDT4). The test problem S_ZDT4 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(p_1)(z_1' + 1), \\
f_2(\mathbf{x}) &= S\left(\sqrt{\sum_{i=1}^{D} p_i^2}\right)\left(g(\mathbf{x})\left(1 - \sqrt{z_1'/g(\mathbf{x})}\right) + 1\right), \\
g(\mathbf{x}) &= 1 + 10(D-1) + \sum_{i=2}^{D}\left(z_i'^2 - 10\cos\left(4\pi z_i'\right)\right),
\end{aligned}
$$

where $z_1' = \begin{cases} z_1, & z_1 \geq 0 \\ -\lambda_1 z_1, & z_1 < 0 \end{cases}$, $p_1 = \begin{cases} 0, & z_1 \geq 0 \\ |z_1|/d_1, & z_1 < 0 \end{cases}$,

$z_i' = \begin{cases} z_i, & z_i \geq -5 \\ -5 - \lambda_i(z_i + 5), & z_i < -5 \end{cases}$, $p_i = \begin{cases} 0, & z_i \geq -5 \\ |z_i + 5|/d_i, & z_i < -5 \end{cases}$, $i = 2, \ldots, D$.

The Pareto-optimal solutions have values of $x_1 \in [o_1, 1 + o_1]$ and $x_i = o_i$ for $i = 2, \ldots, D$. Figure A.5 shows the reference set for the problem. The problem differs from S_ZDT1 in its function $g(\mathbf{x})$, which is multi-modal here, making the problem more difficult.

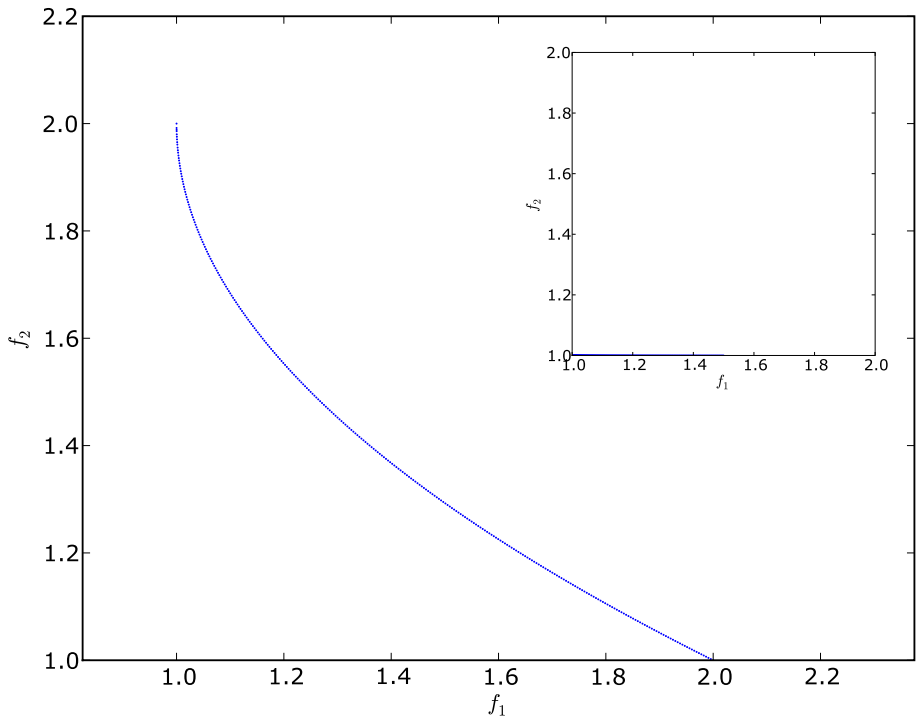


Figure A.5: The reference set for S_ZDT4 and R_ZDT4. The Pareto-front is identical to the one of S_ZDT1, but the objective space bounds are different. So, normalization has a more problematic effect, leading to an almost horizontal Pareto-front (invisible on the f_1 axis). A new reference set was generated because the similarity was not recognized at that time.

A.2.4 Definition (R_ZDT4). The test problem R_ZDT4 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(p_1)(z_1' + 1), \\
f_2(\mathbf{x}) &= S\left(\sqrt{\sum_{i=1}^{D} p_i^2}\right)\left(g(\mathbf{x})\left(1 - \sqrt{z_1'/g(\mathbf{x})}\right) + 1\right), \\
g(\mathbf{x}) &= 1 + 10(D-1) + \sum_{i=2}^{D}\left(z_i'^2 - 10\cos\left(4\pi z_i'\right)\right),
\end{aligned}
$$

where $z_1' = \begin{cases} -\lambda_1 z_1, & z_1 < 0 \\ z_1, & 0 \le z_1 \le 1 \\ \lambda_1 z_1, & z_1 > 1 \end{cases}$, $p_1 = \begin{cases} -z_1, & z_1 < 0 \\ 0, & 0 \le z_1 \le 1 \\ z_1 - 1, & z_1 > 1 \end{cases}$,

$z_i' = \begin{cases} -5 - \lambda_i(z_i + 5), & z_i < -5 \\ z_i, & -5 \le z_i \le 5 \\ 5 - \lambda_i(z_i - 5), & z_i > 5 \end{cases}$, $p_i = \begin{cases} -5 - z_i, & z_i < -5 \\ 0, & -5 \le z_i \le 5 \\ z_i - 5, & z_i > 5 \end{cases}$, $i = 2, \ldots, D.$

The Pareto-front is identical to the one of S_ZDT4. Figure A.5 shows the reference set for the problem.

A.2.5 Definition (S_ZDT6). The test problem S_ZDT6 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(p_1)(1 - \exp(-4z_1')\sin^6(6\pi z_1') + 1), \\
f_2(\mathbf{x}) &= S\left(\sqrt{\sum_{i=1}^{D} p_i^2}\right)\left(g(\mathbf{x})\left(1 - (z_1'/g(\mathbf{x}))^2\right) + 1\right), \\
g(\mathbf{x}) &= 1 + 9 \cdot \left(\frac{\sum_{i=2}^{D} z_i'}{D-1}\right)^2,
\end{aligned}
$$

where $z_i' = \begin{cases} z_i, & z_i \ge 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}$, $p_i = \begin{cases} 0, & z_i \ge 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}$, $i = 1, \ldots, D.$

The Pareto-optimal solutions have values of $x_1 \in [o_1, 1 + o_1]$ and $x_i = o_i$ for $i = 2, \ldots, D$. Figure A.6 shows the reference set for the problem. The function f_1 causes a non-uniform distribution of solutions along the concave Pareto-front.

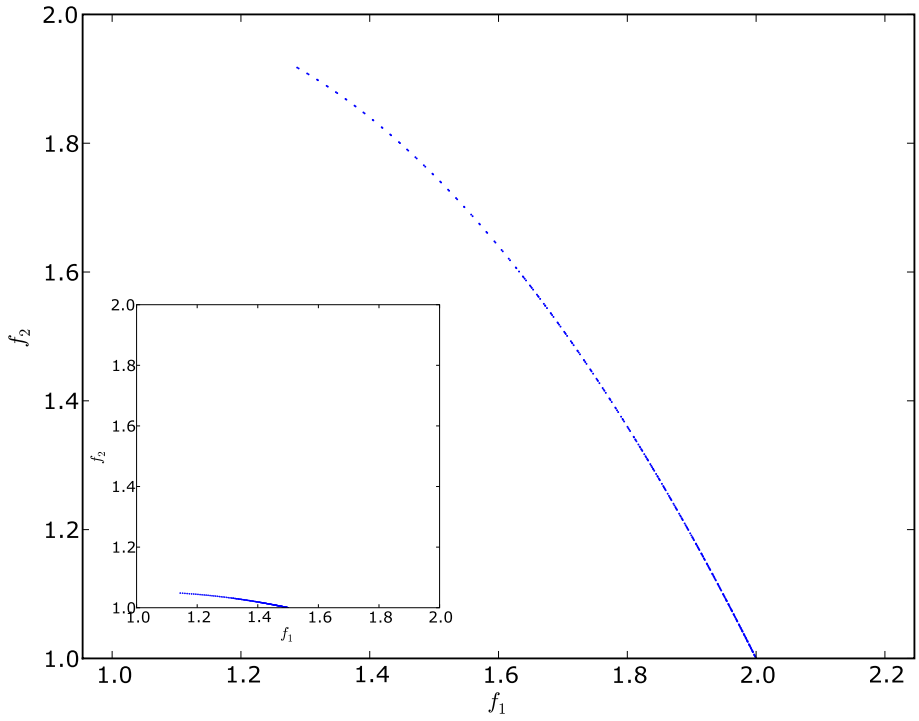


Figure A.6: The reference set for S_ZDT6.

A.2.6 Definition (S_DTLZ2). The test problem S_DTLZ2 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(psum_1)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\cos\left(z_{M-2}'\frac{\pi}{2}\right)\cos\left(z_{M-1}'\frac{\pi}{2}\right)+1\right),\\
f_2(\mathbf{x}) &= S(psum_2)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\cos\left(z_{M-2}'\frac{\pi}{2}\right)\sin\left(z_{M-1}'\frac{\pi}{2}\right)+1\right),\\
f_3(\mathbf{x}) &= S(psum_3)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\sin\left(z_{M-2}'\frac{\pi}{2}\right)+1\right),\\
&\vdots\\
f_{M-1}(\mathbf{x}) &= S(psum_{M-1})\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\sin\left(z_2'\frac{\pi}{2}\right)+1\right),\\
f_M(\mathbf{x}) &= S(psum_M)\left((1+g(\mathbf{x}_M))\sin\left(z_1'\frac{\pi}{2}\right)+1\right),\\
g(\mathbf{x}_M) &= \sum_{x_i\in\mathbf{x}_M}(z_i'-0.5)^2,
\end{aligned}
$$

where $z_i' = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}$, $p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}$, $i = 1,\ldots,D$.

Pareto-optimal solutions satisfy $\sum_{m=1}^{M}(f_m-1)^2 = 1$. This can be achieved by setting $x_i = 0.5 + o_i$ for $x_i \in \mathbf{x}_M$ and choosing $x_i \in [o_i, 1+o_i]$ else. Figure A.7 shows the reference set for the problem. The Pareto-front, a part of a sphere, is identical to the one

of R_DTLZ2 and S_DTLZ3. DTLZ-based problems can principally have an arbitrary number of objectives.

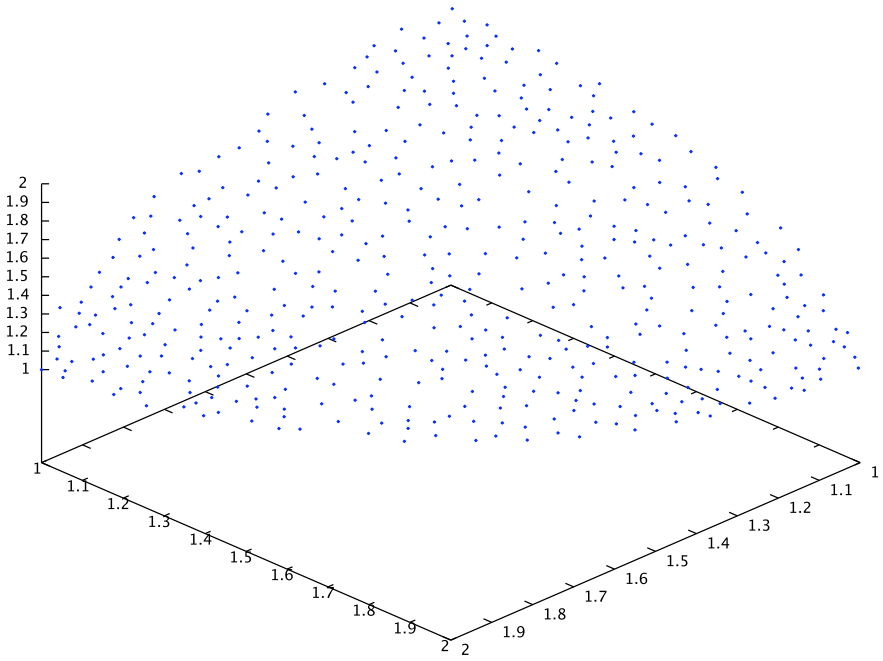


Figure A.7: The reference set for S_DTLZ2, R_DTLZ2 and S_DTLZ3.

A.2.7 Definition (R_DTLZ2). The test problem R_DTLZ2 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(psum_1)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\cos\left(z_{M-2}'\frac{\pi}{2}\right)\cos\left(z_{M-1}'\frac{\pi}{2}\right)+1\right),\\
f_2(\mathbf{x}) &= S(psum_2)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\cos\left(z_{M-2}'\frac{\pi}{2}\right)\sin\left(z_{M-1}'\frac{\pi}{2}\right)+1\right),\\
f_3(\mathbf{x}) &= S(psum_3)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\sin\left(z_{M-2}'\frac{\pi}{2}\right)+1\right),\\
&\vdots\\
f_{M-1}(\mathbf{x}) &= S(psum_{M-1})\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\sin\left(z_2'\frac{\pi}{2}\right)+1\right),\\
f_M(\mathbf{x}) &= S(psum_M)\left((1+g(\mathbf{x}_M))\sin\left(z_1'\frac{\pi}{2}\right)+1\right),\\
g(\mathbf{x}_M) &= \sum_{x_i\in\mathbf{x}_M}(z_i'-0.5)^2,
\end{aligned}
$$

where $z_i' = \begin{cases} -\lambda_i z_i, & z_i < 0 \\ z_i, & 0 \le z_i \le 1 \\ \lambda_i z_i, & z_i > 1 \end{cases}$, $p_i = \begin{cases} -z_i & z_i < 0 \\ 0, & 0 \le z_i \le 1 \\ z_i - 1, & z_i > 1 \end{cases}$, $i = 1,\ldots,D.$

Pareto-optimal solutions satisfy $\sum_{m=1}^{M}(f_m-1)^2 = 1$. Figure A.7 shows the reference set for the problem.

A.2.8 Definition (S_DTLZ3). The test problem S_DTLZ3 is defined as

$$
\begin{aligned}
f_1(\mathbf{x}) &= S(psum_1)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\cos\left(z_{M-2}'\frac{\pi}{2}\right)\cos\left(z_{M-1}'\frac{\pi}{2}\right)+1\right),\\
f_2(\mathbf{x}) &= S(psum_2)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\cos\left(z_{M-2}'\frac{\pi}{2}\right)\sin\left(z_{M-1}'\frac{\pi}{2}\right)+1\right),\\
f_3(\mathbf{x}) &= S(psum_3)\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\cos\left(z_2'\frac{\pi}{2}\right)\ldots\sin\left(z_{M-2}'\frac{\pi}{2}\right)+1\right),\\
&\vdots\\
f_{M-1}(\mathbf{x}) &= S(psum_{M-1})\left((1+g(\mathbf{x}_M))\cos\left(z_1'\frac{\pi}{2}\right)\sin\left(z_2'\frac{\pi}{2}\right)+1\right),\\
f_M(\mathbf{x}) &= S(psum_M)\left((1+g(\mathbf{x}_M))\sin\left(z_1'\frac{\pi}{2}\right)+1\right),\\
g(\mathbf{x_M}) &= 100\Big(|\mathbf{x_M}|+\sum_{x_i\in\mathbf{x_M}}\left((z_i'-0.5)^2-\cos(20\pi(z_i-0.5))\right)\Big),
\end{aligned}
$$

where $z_i' = \begin{cases} z_i, & z_i \geq 0 \\ -\lambda_i z_i, & z_i < 0 \end{cases}$, $p_i = \begin{cases} 0, & z_i \geq 0 \\ |z_i|/d_i, & z_i < 0 \end{cases}$, $i = 1,\ldots,D$.
Pareto-optimal solutions have $x_i = 0.5 + o_i$ for $x_i \in \mathbf{x}_M$ and $x_i \in [o_i, 1+o_i]$ else. Figure A.7 shows the reference set for the problem.

A.3 WFG Problems

Table A.3: Variable names used in all WFG problems.

Variable	Purpose
M	Number of objectives
n	Number of decision variables
$\mathbf{x}$	Vector of M underlying parameters
x_M	Distance parameter
$x_{1:M-1}$	Position parameters
$\mathbf{z}$	Vector of $k+l=n\geq M$ working parameters
$D>0$	Distance scaling constant
$A_{1:M-1}\in\{0,1\}$	Degeneracy constants
$h_{1:M}$	Shape functions
$S_{1:M}>0$	Scaling constants
$t^{1:p}$	Transition vectors

Table A.3 shows the meaning of some variables used in WFG problems. The problems are described in greater detail in [28]. All WFG problems are of the following form.

$$
\begin{array}{ll}
\text{Given} & \mathbf{z} = (z_1, \ldots, z_k, z_{k+1}, \ldots, z_n) \\
\text{Minimize} & f_{m=1:M}(\mathbf{x}) = Dx_M + S_m h_m(x_1, \ldots, x_{M-1}) \\
\text{Where} & \mathbf{x} = (x_1, \ldots, x_M) \\
& \quad = (\max\{t_M^p, A_1\}(t_1^p - 0.5) + 0.5, \ldots, \max\{t_M^p, A_{M-1}\}(t_{M-1}^p - 0.5) + 0.5, t_M^p) \\
& \mathbf{t}^p = (t_1^p, \ldots, t_M^p) \leftarrow\!\shortmid \mathbf{t}^{p-1} \leftarrow\!\shortmid \ldots \leftarrow\!\shortmid \mathbf{t}^1 \leftarrow\!\shortmid \mathbf{z}_{[0,1]} \\
& \mathbf{z}_{[0,1]} = (z_{1,[0,1]}, \ldots, z_{n,[0,1]}) \\
& \quad = (\frac{z_1}{z_{1,\max}}, \ldots, \frac{z_n}{z_{n,\max}})
\end{array}
$$

The notation $\mathbf{t}^{i+1} \leftarrow\!\shortmid \mathbf{t}^i$ means that the output of transformation $\mathbf{t}^i$ is used as input for $\mathbf{t}^{i+1}$. In the following, only the shape function h_m and transformation functions $\mathbf{t}^i$ for the actual problem are shown.

A.3.1 Definition (WFG1). The test problem WFG1 is defined as

$$
\begin{array}{lrcl}
\text{Shape} & h_{m=1:M-1} & = & \text{convex}_m \\
 & h_M & = & \text{mixed}_M \text{ (with } \alpha = 1 \text{ and } A = 5) \\
\mathbf{t^1} & t^1_{i=1:k} & = & y_i \\
 & t^1_{i=k+1:n} & = & \text{s\_linear}(y_i, 0.35) \\
\mathbf{t^2} & t^2_{i=1:k} & = & y_i \\
 & t^2_{i=k+1:n} & = & \text{b\_flat}(y_i, 0.8, 0.75, 0.85) \\
\mathbf{t^3} & t^3_{i=1:n} & = & \text{b\_poly}(y_i, 0.02) \\
\mathbf{t^4} & t^4_{i=1:M-1} & = & \text{r\_sum}(\{y_{(i-1)k/(M-1)+1}, \ldots, y_{ik/(M-1)}\}, \\
 & & & \qquad \{2(\frac{(i-1)k}{M-1} + 1), \ldots, \frac{2ik}{M-1}\}) \\
 & t^4_M & = & \text{r\_sum}(\{y_{k+1}, \ldots, y_n\}, \{2(k+1), \ldots, 2n\}).
\end{array}
$$

Please refer to [28] for a detailed explanation of transformation functions like s_linear, b_flat, b_poly, r_sum and shape functions like convex_m and mixed_m. Pareto-optimal solutions have values of $z_{i=1:k} \in [0, 2i]$ and $z_{i=k+1:n} = 2i \cdot 0.35$. Figure A.8 shows the reference set for the problem. The Pareto-front has a convex shape with ripples on it.

A.3.2 Definition (WFG8). The test problem WFG8 is defined as

$$
\begin{array}{lrcl}
\text{Shape} & h_{m=1:M-1} & = & \text{concave}_m \\
\mathbf{t^1} & t^1_{i=1:k} & = & y_i \\
 & t^1_{i=k+1:n} & = & \text{b\_param}(y_i, \text{r\_sum}(\{y_1, \ldots, y_{i-1}\}, \{1, \ldots, 1\}), \frac{0.98}{49.98}, 0.02, 50) \\
\mathbf{t^2} & t^2_{i=1:k} & = & y_i \\
 & t^2_{i=k+1:n} & = & \text{s\_linear}(y_i, 0.35) \\
\mathbf{t^3} & t^3_{i=1:M-1} & = & \text{r\_sum}(\{y_{(i-1)k/(M-1)+1}, \ldots, y_{ik/(M-1)}\}, \{1, \ldots, 1\}) \\
 & t^3_M & = & \text{r\_sum}(\{y_{k+1}, \ldots, y_n\}, \{1, \ldots, 1\}).
\end{array}
$$

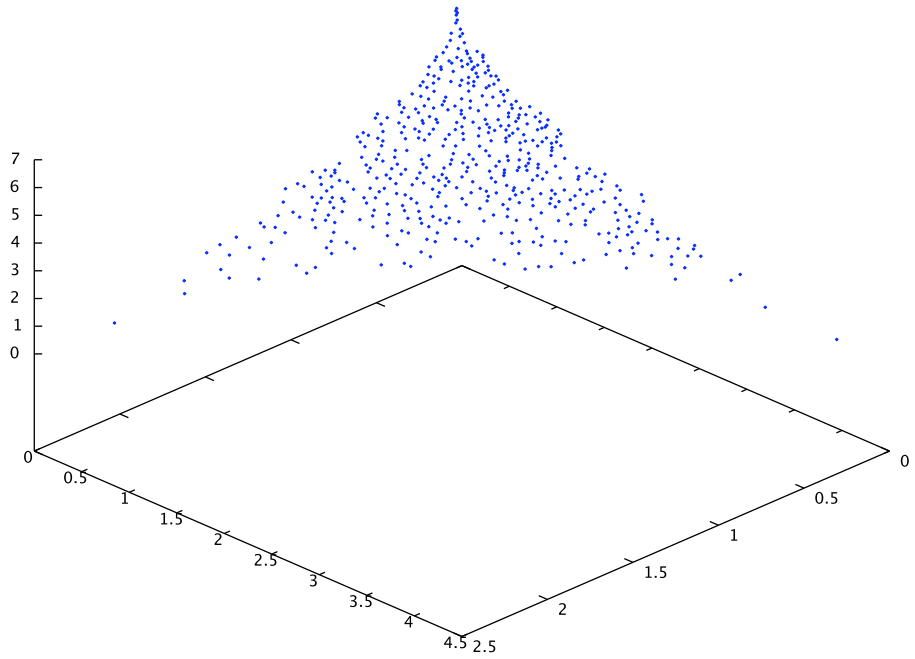


Figure A.8: The reference set for WFG1.

Pareto-optimal solutions have values of $z_{i=1:k} \in [0, 2i]$ and $z_{i=k+1:n} = 2i \cdot 0.35^{(0.02+49.98(\frac{0.98}{49.98}-(1-2u)\lvert\lfloor 0.5-u\rfloor+\frac{0.98}{49.98}\rvert))^{-1}}$ for $u = \text{r\_sum}(\{z_1, \ldots, z_{i-1}\}, \{1, \ldots, 1\})$ respectively. Figure A.9 shows the reference set for the problem. The Pareto-front is concave.

A.3.3 Definition (WFG9). The test problem WFG9 is defined as

$$
\begin{array}{llcl}
\text{Shape} & h_{m=1:M-1} & = & \text{concave}_m \\
\mathbf{t^1} & t^1_{i=1:n-1} & = & \text{b\_param}(y_i, \text{r\_sum}(\{y_{i+1}, \ldots, y_n\}, \{1, \ldots, 1\}), \frac{0.98}{49.98}, 0.02, 50) \\
 & t^1_n & = & y_n \\
\mathbf{t^2} & t^2_{i=1:k} & = & \text{s\_decept}(y_i, 0.35, 0.001, 0.05) \\
 & t^2_{i=k+1:n} & = & \text{s\_multi}(y_i, 30, 95, 0.35) \\
\mathbf{t^3} & t^3_{i=1:M-1} & = & \text{r\_nonsep}(\{y_{(i-1)k/(M-1)+1}, \ldots, y_{ik/(M-1)}\}, \frac{k}{M-1}) \\
 & t^3_M & = & \text{r\_nonsep}(\{y_{k+1}, \ldots, y_n\}, l).
\end{array}
$$

Pareto-optimal solutions have $z_{i=1:k} \in [0, 2i]$ and $z_{i=k+1:n} = 2i \cdot \begin{cases} 0.35^{(0.02+1.96u)^{-1}}, & i \neq n \\ 0.35, & i = n \end{cases}$ for $u = \text{r\_sum}(\{z_{i+1}, \ldots, z_n\}, \{1, \ldots, 1\})$ respectively. Figure A.10 shows the reference set for the problem. The Pareto-front is concave.

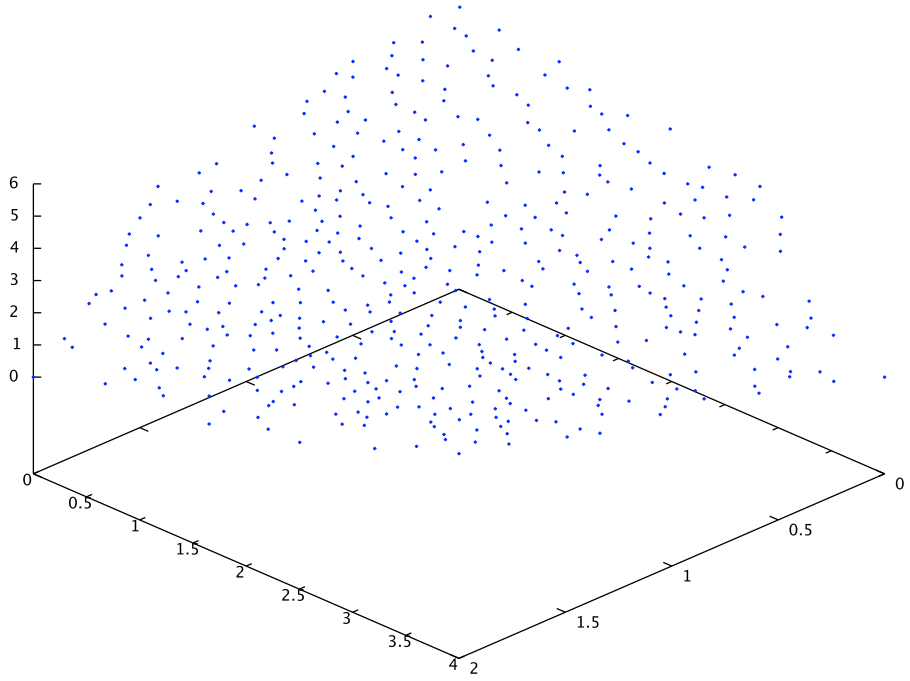


Figure A.9: The reference set for WFG8.

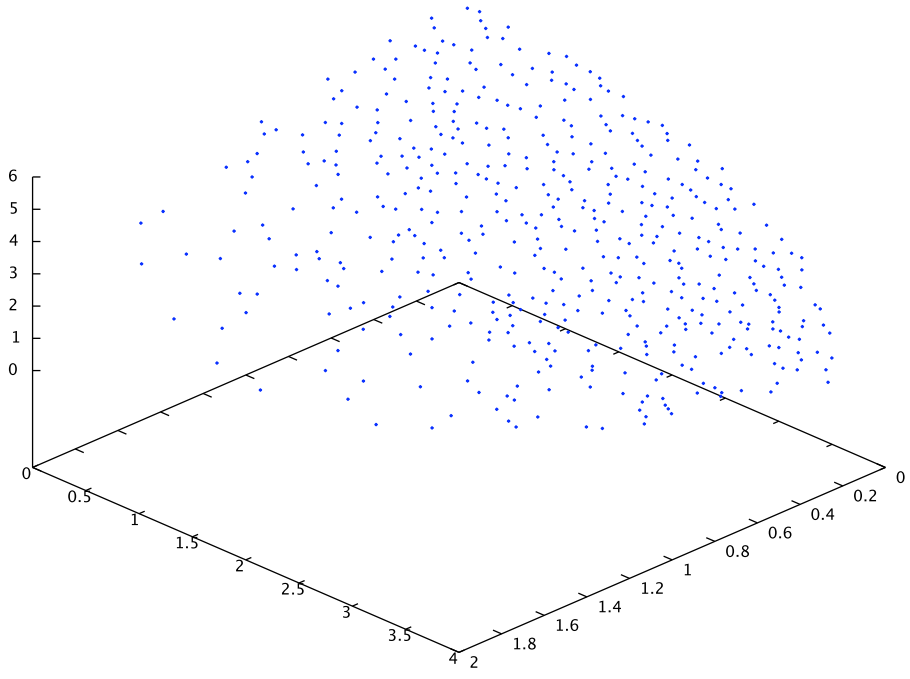


Figure A.10: The reference set for WFG9. The Pareto-front is identical to WFG8, however this was not recognized when the reference sets were generated.

Appendix B

Framework documentation

All experiments for this thesis were carried out on a newly implemented framework for evolutionary multi-objective optimization. Besides the various optimization algorithms, it contains test problems to evaluate the algorithms' performance. The whole framework is written in Python [57], an interpreted, high-level programming language. The design was guided by the following postulations:

No redundant objective function evaluations should be made. In real world problems, function evaluations are often highly computationally expensive (e. g. simulator runs).

The class design should reflect the biological model. Biological evolution is already the paradigm for the working principle of EAs. So, unneccessary deviations should be avoided to not confuse the user.

Code reuse should be of high significance. Also known as the design philosophy "Don't repeat yourself", it leaves less space for potential errors. Additionally, existing errors surface more often and are less easily overlooked. Finally, discovered errors can be fixed quickly.

To achieve the requirements above, an object-oriented, highly modularized approach was chosen. As many of an EA's components as possible were put into separate classes. Although Python does not support access modifiers to restrict attribute access, information hiding is introduced as far as possible. For example, the individual's genome and the according variation operators are encapsulated by the individual. This makes the EAs independent of the genome's encoding.

The following documentation lists the classes structured in packages and modules as they appear in the source code. UML diagrams illustrate the interesting parts of the framework in each section. The function signatures are not shown in the diagrams to improve readability. As Python does not support function overloading, the signatures can become quite complicated, containing optional arguments.

B.1 Package `emo`

B.1.1 Individuals

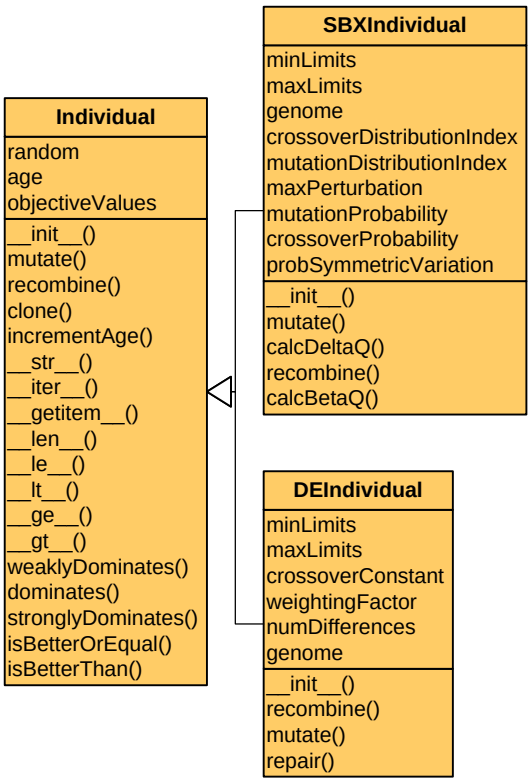


Figure B.1: The module `individual` contains the classes `Individual`, `RGAIndividual` and `DE-Individual`.

Individual

The `Individual` class should be used as base class for all individuals that are going to be used in the framework. It overrides the comparison operators to compare individuals by their objective values. All of them conform to the dominance relation as described in Chapter 1. An attribute `age` allows for tracking of its lifespan. The class designedly has no attributes that are problem-dependent, especially the genome. So, the variation operators `mutate` and `recombine` are abstract methods and have to be overwritten in the subclasses. Mutation must meet the following conditions:

- Mutation happens in-place and nothing is returned.
- As mutation renders the current objective values invalid, they should be deleted.

Recombination must meet the following conditions:

- Recombination takes a sequence of n parents as input. The object whose method is called is the $(n+1)$th parent. n can be an arbitrary natural number.
- It returns an arbitrary number of children as an iterable object.
- The returned children should be newly-created objects.

The encapsulation of the actual variation in the individual makes it possible not only to have different strategy parameters in each individual, but also completely different strategies, as long as the genomes are compatible. As indicated by the method signature, the parent whose `recombine` method is called is in an exceptional position. It is not only contributing its genome to the offspring's creation, but also decides on the inheritance of the strategy parameters. This way, self-adaptation is very simple, even for recombination.

SBXIndividual

The class `SBXIndividual` provides mutation and recombination operators that were originally invented by Deb and Agrawal [10] for real-coded genetic algorithms. Both operators use polynomial probability distributions to simulate the variation of binary encoded genomes. Not surprisingly, this individual's `genome` is encoded as a vector of real numbers, which is the default encoding for most available benchmark problems in the literature [63, 15, 26]. Usually, every variable in the vector is bounded by an upper and a lower limit. These are stored in the vectors `minLimits` and `maxLimits`.

DEIndividual

The class `DEIndividual` uses Differential Evolution [56] as variation. It has the three parameters `crossoverConstant` $\in [0,1]$, `weightingFactor` $\in [0,2]$ and `numDifferences` $\in \mathbb{N}$. The whole variation happens in the `recombine` method. The `mutate` method is empty. The `genome` is a vector of real numbers, just like in `SBXIndividual`. The `repair` function is used after variation to ensure that all interval constraints are satisfied.

B.1.2 Objective Functions

ObjectiveFunction

The class `ObjectiveFunction` is the building block for most problems. It has an attribute `isMaximizing` that indicates if the objective function should be maximized or minimized. Another attribute counts the `evaluations` of the function. It is incremented whenever the `__call__` method is run. While this has the drawback that all subclasses have to call the superclass method or count the evaluations themselves, it provides a way to count evaluations independently from the algorithm that carries out the optimization. The objective value is returned, the individual should not be modified.

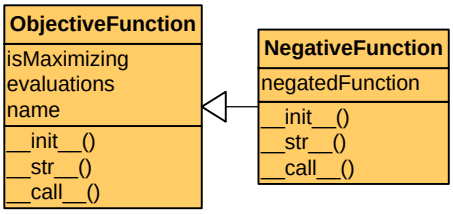


Figure B.2: The module `objective` contains the classes `ObjectiveFunction` and `NegativeFunction`.

NegativeFunction

The class `NegativeFunction` enables an algorithm to change between maximization and minimization automatically. It holds a reference to the original function in `negatedFunction` and the `isMaximizing` attribute is the opposite of `negatedFunction`'s goal. Finally, when the negative function is called, it calls the original function and returns that value multiplied by -1.

B.1.3 Problems

The module `problem` contains the classes `Problem`, `TestProblem` and `ProblemSuite`.

Problem

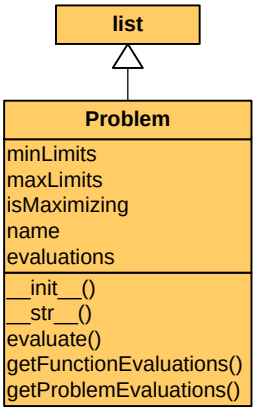


Figure B.3: The class `Problem` is the base class for all problems used in the framework.

Figure B.3 shows the base class for all problems. It basically represents a list of objective functions. All contained objective functions must adhere to the problem's `isMaximizing` attribute. An interface that encapsulates them is added by the methods `evaluate` and `getProblemEvaluations`. The encapsulation enables an optimizing algorithm to handle problems that do not possess distinct objective functions. The `evaluate` method writes

the objective values into the individual's `objectiveValues` list. `getFunctionEvaluations` returns the sum of evaluations of all contained objective functions.

TestProblem

`TestProblem` is a subclass of `Problem` that provides some additional functionality like normalization and sampling of the Pareto-front needed for benchmarks.

ProblemSuite

`ProblemSuite` is the base class for test case collections. It is derived from the Python `list`. Subclasses can overwrite the constructor to fill themselves with problems.

B.1.4 Algorithms

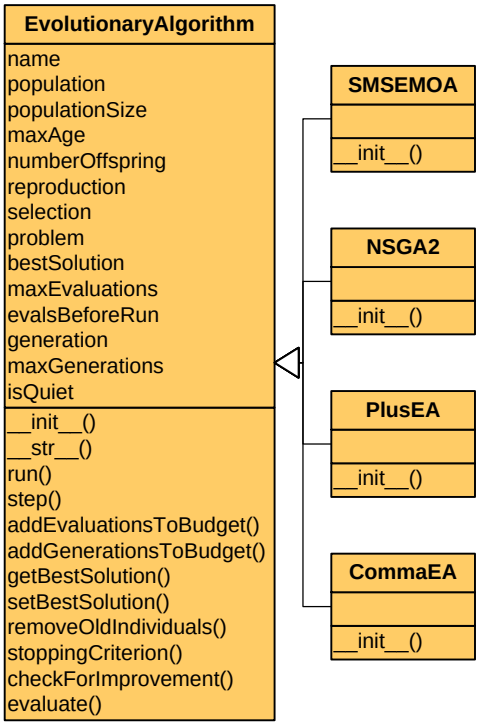


Figure B.4: The module `algo` contains the classes `EvolutionaryAlgorithm`, `CommaEA`, `PlusEA`, `NSGA2` and `SMSEMOA`.

EvolutionaryAlgorithm

`EvolutionaryAlgorithm` is a customizable (μ, κ, λ)-EA. `reproduction`, `selection` and the `problem` have been outsourced into separate classes, leaving not much more than

the algorithm's main loop. Both the number of generations and the number of function evaluations can be used as a stopping criterion.

CommaEA

This class is intended to provide a convenient constructor to create a standard (μ , λ) evolution strategy.

PlusEA

This class is intended to provide a convenient constructor to create a standard ($\mu + \lambda$) evolution strategy.

NSGA-2

The Non-Dominated Sorting Genetic Algorithm 2 (NSGA-2) by Deb et al. [11] is a frequently used algorithm for multi-objective optimization.

SMS-EMOA

The S-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-EMOA) by Emmerich et al. [17] uses a hypervolume-indicator-based selection component as a specialty.

B.1.5 Sorting

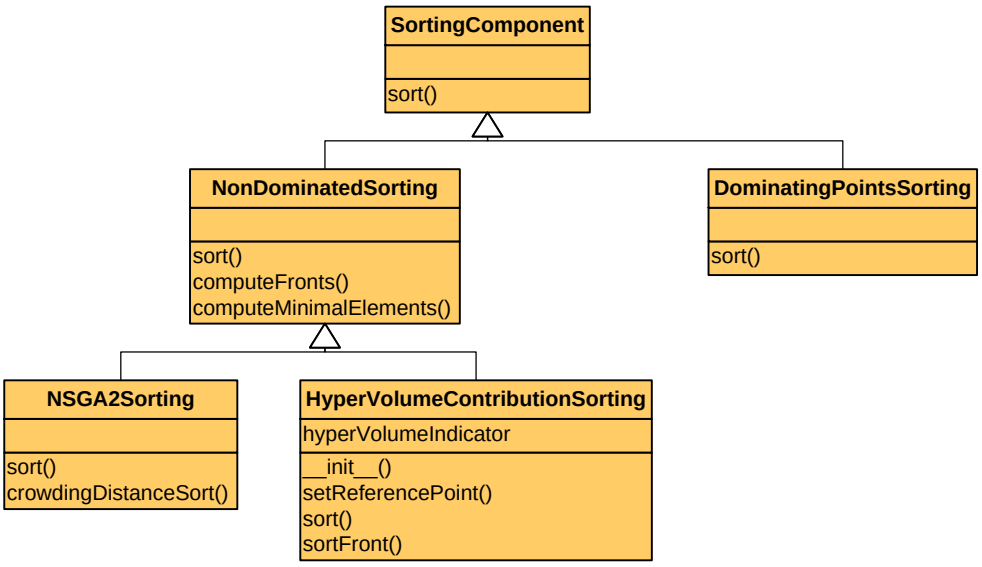


Figure B.5: The module `sorting` contains the classes `SortingComponent`, `NonDominatedSorting`, `NSGA2Sorting` and `HyperVolumeContributionSorting`.

SortingComponent

SortingComponents play an important role for Selection objects. The interface is simple: Every subclass has to overwrite the sort method that takes a list of individuals as input. The sorting should be descending in fitness.

NonDominatedSorting

Non-dominated sorting was proposed by Goldberg [22]. Many algorithms use this sorting as first order selection criterion. The class provides a method computeFronts to compute the non-dominated fronts in the population. sort returns the concatenation of these fronts into a flat list. computeMinimalElements only retrieves the first non-dominated front.

NSGA2Sorting

As the name suggests, this sorting is used in the NSGA-2. First, non-dominated sorting is used to obtain the ranking of fronts. Each front is then sorted internally with crowdingDistanceSort. This niching mechanism helps maintaining diversity in the objective space.

HyperVolumeContributionSorting

This sorting is used by the SMS-EMOA. In a first step, the poluation is divided into non-dominated fronts. Then sortFront computes for every individual how much hypervolume it exclusively contributes to its front.

DominatingPointsSorting

DominatingPointsSorting assigns every point a rank which is equal to the number of other points that dominate it. The sorting order is then ascending in these ranks.

B.1.6 Selection

Selection

The Selection class is responsible for reducing the number of indviduals in the population. The select method is abstract. Overwriting methods should return a list of individuals without modifying the original population. The reduceTo method takes the survivors that are returned by the select method and replaces the population with them.

UniformSelection

The UniformSelection draws a uniformly distributed random sample from the population. So, it applies an undirected selection pressure.

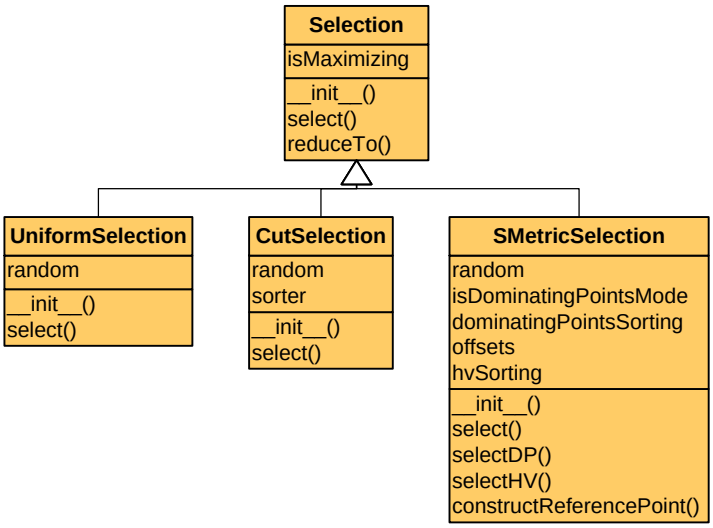


Figure B.6: The module `selection` contains the classes `Selection`, `UniformSelection`, `CutSelection` and `SMetricSelection`.

CutSelection

The `CutSelection` class uses a `SortingComponent` to obtain a fitness-based ranking of the individuals. So, its properties are determined by the actual sorting component. For example, the selection can only be elitist if the sorting supports it.

SMetricSelection

The S-Metric selection is used by the SMS-EMOA. It uses `HyperVolumeContribution-Sorting` to obtain the fittest individuals. Due to the fact that the SMS-EMOA uses a $(\mu + 1)$ selection scheme, it only needs to sort the last front by hypervolume contribution to find the least valuable individual. It contains a boolean flag `isDominatingPointsMode` that decides if the alternative `dominatingPointsSorting` is used for selection when there is more than one non-dominated front. `offsets` contains a positive offset value for each objective. These are used in `constructReferencePoint`.

B.1.7 Reproduction

Reproduction

The `Reproduction` class works similar to `Selection`. The `breed` method is responsible for producing the offspring. The new individuals are returned as list while the original population is unmodified. During reproduction, recombination is used with a probability of `recombinationProbability`, otherwise an individual is cloned. The `extendBy` method appends the output of `breed` to the population. The `numberParents` parameter specifies

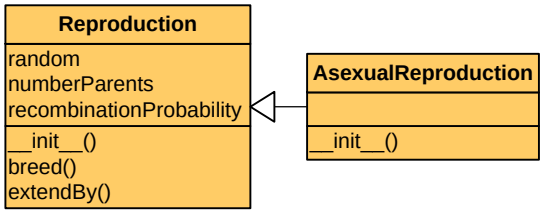


Figure B.7: The module `reproduction` contains the classes `Reproduction` and `AsexualReproduction`.

how many parents are involved in each recombination process. `AsexualReproduction` is a special case of `Reproduction` where `recombinationProbability` is set to zero.

B.1.8 Performance Indicators

QualityIndicator

`QualityIndicator` is the abstract base class for all quality indicators. It needs information whether the analysed problem is maximizing or not. The method `assess` computes the first non-dominated front of the population and then calls `assessNonDominatedFront`, which is abstract.

RIndicator

`RIndicator` implements a number of R indicator variants, as defined by Hansen and Jaszkiewicz [23].

EpsilonIndicator

The `EpsilonIndicator` class implements the ϵ_+ indicator defined by Zitzler et al. [67].

HyperVolumeIndicator

`HyperVolumeIndicator` is an abstract base class for implementations of the performance measure also called $\mathcal{S}$-Metric. It contains the `referencePoint` that is neccessary for hypervolume computation.

FonsecaHyperVolume

`FonsecaHyperVolume` is an algorithm by Fonseca et al. [21] for hypervolume computation. The implementation at hand corrensponds to the algorithm called "Version 3" in the paper. Its runtime complexity is $O(n^{d-1})$ where n is the number of points and d the number of dimensions.

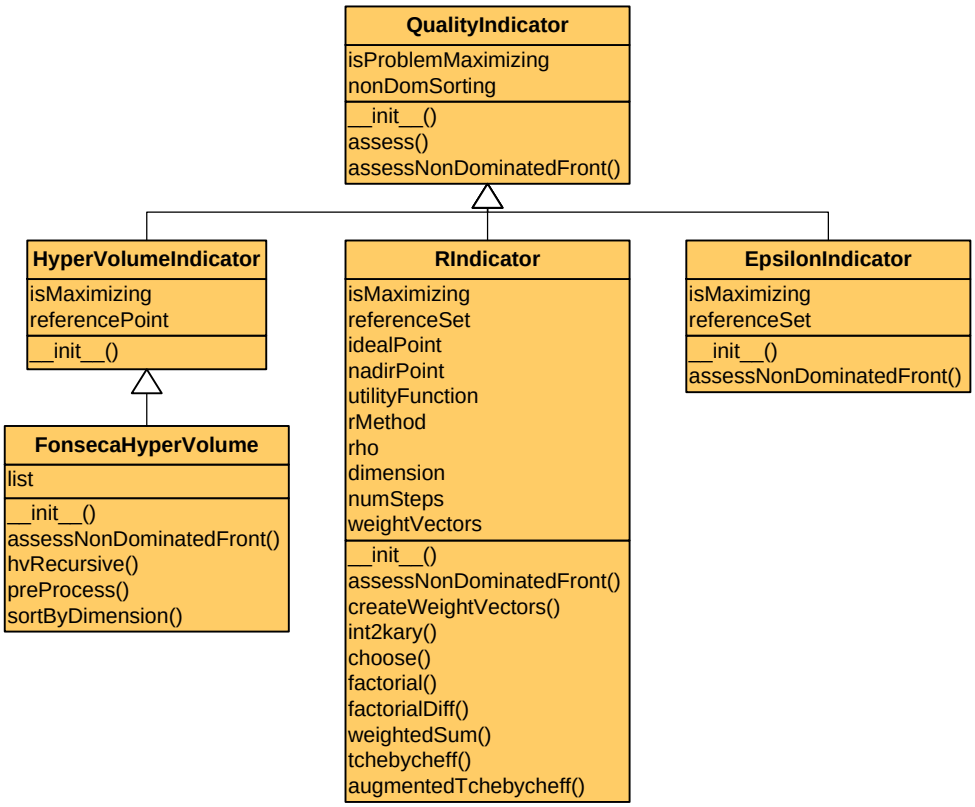


Figure B.8: The module `performance` contains the classes `QualityIndicator`, `RIndicator`, `EpsilonIndicator`, `HyperVolumeIndicator` and `FonsecaHyperVolume`.

B.2 Package emo.benchmark

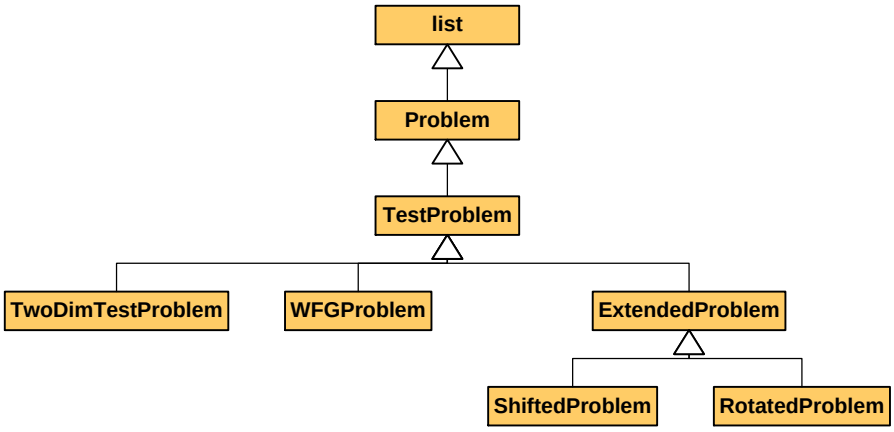


Figure B.9: The inheritance relationships for the base classes in the `benchmark` package.

The `benchmark` package contains several modules with multi-objective test problems. The terms "test problem", "test case" and "benchmark" are all used synonymously here.

B.2.1 ZDT

The test case collection ZDT by Zitzler et al. [63] contains six two-dimensional test problems. ZDT5, the only problem which implies a binary parameter space here, seems to be less frequently used for benchmarks than the others, which are designed for a continuous parameter space.

B.2.2 DTLZ

The test problem suite DTLZ by Deb et al. is defined differently in two papers. [14] contains nine problems, while [15] contains only seven. Additionally, the numbering differs from the fifth problem on. However, the first four problems already satisfy the dependency of the CEC 2007 test case collection (see section B.2.4), and thus are the only ones implemented here. DTLZ can generate problems with an arbitrary number of objective functions.

B.2.3 WFG

WFG by Huband et al. [28] is another recent toolkit for test problem generation. It was designed to enable certain features that previous test problems do not exhibit, e. g. non-separable objective functions. While it allows the user to create his own problem with specified properties, it already contains nine example test problems which are referred to as `WFG1`–`WFG9`. These problems are subclasses of `WFGProblem`.

B.2.4 CEC 2007

The CEC 2007 test suite [26] was compiled from various selected problems for a competition at the correspondent conference. The collection contains the following problems:

- `OKA2` by Okabe et al. [42].
- `SYM-PART` by Rudolph et al. [46].
- `S_ZDT1`, `S_ZDT2`, `S_ZDT4`, `R_ZDT4` and `S_ZDT6` which are ZDT problems modified by shifting or rotating the parameter vector and extending the problem domain.
- `S_DTLZ2`, `R_DTLZ2`, `S_DTLZ3` which are rotated or shifted DTLZ problems.
- `WFG1`, `WFG8` and `WFG9` from the WFG toolkit (see section B.2.3).

A reference implementation of these problems in C can be obtained at the competition's homepage [26]. However, all problems are completely reimplemented in Python for this framework. During testing, several bugs surfaced in the original implementation. All bugs mentioned below are fixed in the Python implementation. The problem definitions in Appendix A also follow the suggestions made in this section.

OKA2

The original implementation contains a bug in $f_2(x)$, where `fabs` should be used instead of `abs`. The usage of `abs` causes a rounding down to the next integer.

Shifted ZDT Problems

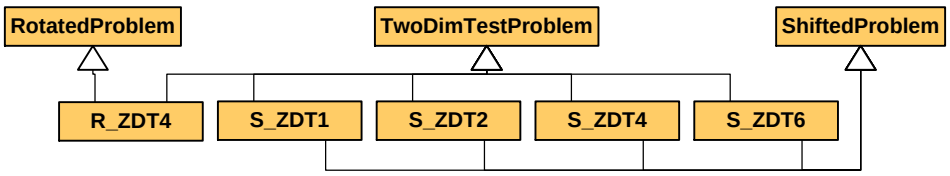


Figure B.10: The inheritance relationships for extended ZDT problems.

The $f_2(x)$ functions of these problems contain a penalty term $S(\sqrt{\sum_{i=1}^{D} p_i^2})$. However, the original implementation leaves out p_1 and thus only calculates $S(\sqrt{\sum_{i=2}^{D} p_i^2})$. Note that in R_ZDT4, which is defined identically in this aspect, all D penalty values are considered.

Rotated problems

Both rotated problems deviate from the definition in the way they calculate the $\mathbf{z}'$ vector. To date, these deviations also occur in the implementation of the CEC 2009 test suite.

Additionally, the definition is ambiguous because the rotated problems distinguish three cases for calculating the penalties p_i. For example, R_ZDT4 uses the definition

$$p_1 = \begin{cases} -z_1, & z_1 < 0 \\ 0, & 0 \leq z_1 \leq 1 \\ z_1 - 1, & z_1 > 1 \end{cases}$$

for a penalty and as an objective function

$$f_1(x) = \begin{cases} z_1' + 1, & z_1 \geq 0 \\ S(p)(z_1' + 1), & z_1 < 0. \end{cases}$$

If the user sticks to the definition of $f_1(x)$, the penalty in the case $z_1 > 1$ is disregarded. However, because $S(0) = 1$ it would be sufficient to write $f_1(x) = S(p)(z_1' + 1)$. This applies to all extended problems in the test suite and is also how the implementation works. For $\mathbf{z}'$, the original R_ZDT4 implementation uses

$$z_1' = \begin{cases} -\lambda_1 z_1, & z_1 < 0 \\ z_1, & 0 \leq z_1 \leq 1 \\ 1 - \lambda_1(z_1 - 1), & z_1 > 1 \end{cases}$$

instead of

$$z_1' = \begin{cases} -\lambda_1 z_1, & z_1 < 0 \\ z_1, & 0 \leq z_1 \leq 1 \\ \lambda_1 z_1, & z_1 > 1 \end{cases}$$

in the definition.

DTLZ-derived problems

The original R_DTLZ2 implementation uses

$$z_i' = \begin{cases} -\lambda_i z_i, & z_i < 0 \\ z_i, & 0 \leq z_i \leq 1 \\ 1 - \lambda_i(z_i - 1), & z_i > 1 \end{cases}$$

instead of

$$z_i' = \begin{cases} -\lambda_i z_i, & z_i < 0 \\ z_i, & 0 \leq z_i \leq 1 \\ \lambda_i z_i, & z_i > 1 \end{cases}$$

in the definition. Additionally, the used g function is not

$$g(\mathbf{x_M}) = \sum_{x_i \in \mathbf{x_M}} (z_i' - 0.5)^2$$

as stated, but

$$g(\mathbf{x_M}) = 100\Big(|\mathbf{x_M}| + \sum_{x_i \in \mathbf{x_M}} \big((z_i' - 0.5)^2 - \cos(20\pi(z_i - 0.5))\big)\Big).$$

So, the problem is rather based on DTLZ3 than DTLZ2. The two only differ in their g functions. For all DTLZ-derived problems, Pareto-optimal solutions do not have to satisfy $\sum_{m=1}^{M}(f_m)^2 = 0.5$ as stated in [26], but $\sum_{m=1}^{M}(f_m - 1)^2 = 1$. The reference sets provided with the original implementation contain correct values.

List of Figures

List of Algorithms

Bibliography

[1] Allgemeiner Deutscher Automobilclub. Ecotest. `http://www.adac.de/ecotest/`. 2 January 2009.

[2] Thomas Bartz-Beielstein. *Experimental Research in Evolutionary Computation - The New Experimentalism*. Natural Computing Series. Springer, Heidelberg, Berlin, 2006.

[3] Nicola Beume and Günter Rudolph. Faster S-metric calculation by considering dominated hypervolume as Klee's measure problem. In *Proceedings of the Second IASTED International Conference on Computational Intelligence*, pages 233–238. ACTA Press, 2006.

[4] Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing: an international journal*, 1(1):3–52, 2002.

[5] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, editors. *Multiobjective Optimization, Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*. Springer, 2008.

[6] Karl Bringmann and Tobias Friedrich. Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. In *Evolutionary Multi-Criterion Optimization*, volume 5467 of *Lecture Notes in Computer Science*, pages 6–20. Springer, 2009.

[7] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, Berlin, 2007.

[8] William J. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, 1998.

[9] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[10] Kalyanmoy Deb and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148, 1995.

[11] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 849–858, London, UK, 2000. Springer.

[12] Kalyanmoy Deb and Hans-Georg Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):197–221, 2001.

[13] Kalyanmoy Deb, Karthik Sindhya, and Tatsuya Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1187–1194, New York, NY, USA, 2007. ACM.

[14] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable test problems for evolutionary multi-objective optimization. Technical Report 112, Computer Engineering and Networks Laboratory, ETH Zurich, July 2001.

[15] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC 2002)*, pages 825–830. IEEE Press, 2002.

[16] Agoston E. Eiben and James E. Smith. *Introduction to Evolutionary Computing.* Springer, 2003.

[17] Michael Emmerich, Nicola Beume, and Boris Naujoks. An EMO algorithm using the hypervolume measure as selection criterion. In *Proc. Evolutionary Multi-Criterion Optimization, 3rd Int'l Conf. (EMO 2005)*, pages 62–76. Springer, 2005.

[18] Andries P. Engelbrecht. *Computational Intelligence: An Introduction.* Wiley, 2003.

[19] Ronald A. Fisher. *The design of experiments.* Oliver and Boyd, Edinburgh, 8th edition, 1966.

[20] Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence through Simulated Evolution.* John Wiley, New York, USA, 1966.

[21] Carlos M. Fonseca, Luís Paquete, and Manuel López-Ibáñez. An improved dimension-sweep algorithm for the hypervolume indicator. In *IEEE Congress on Evolutionary Computation, (CEC 2006)*, pages 1157–1163, July 2006.

[22] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[23] Michael Pilegaard Hansen and Andrzej Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report IMM-REP-1998-7, Institute of

Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark, March 1998.

[24] John H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, 1975.

[25] Myles Hollander and Douglas A. Wolfe. *Nonparametric Statistical Methods.* John Wiley & Sons, New York, 1973.

[26] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss, and S. Huband. Problem definitions for performance assessment of multi-objective optimization algorithms. Technical report, Nanyang Technological University, Singapore, 2007. `http://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-07/CEC07.htm`.

[27] V. L. Huang, A. K. Qin, P. N. Suganthan, and M. F. Tasgetiren. Multi-objective optimization based on self-adaptive differential evolution algorithm. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3601–3608, Sept. 2007.

[28] Simon Huband, Phil Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, Oct. 2006.

[29] Joshua D. Knowles and David Corne. Properties of an adaptive archiving algorithm for storing nondominated vectors. *IEEE Transactions on Evolutionary Computation*, 7(2):100–116, 2003.

[30] Joshua D. Knowles, Lothar Thiele, and Eckart Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical report, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 2005.

[31] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, December 1992.

[32] Saku Kukkonen and Jouni Lampinen. Performance assessment of generalized differential evolution 3 (GDE3) with a given set of problems. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3593–3600, 2007.

[33] E. L. Lehmann and Joseph P. Romano. *Testing Statistical Hypotheses.* Springer, New York, 3rd edition, 2005.

[34] Søren N. Lophaven, Hans B. Nielsen, and Jacob Søndergaard. DACE – A Matlab Kriging Toolbox. Technical Report IMM-REP-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark, August 2002.

[35] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs*. Springer, London, UK, 3rd edition, 1996.

[36] Kaisa Miettinen. Introduction to multiobjective optimization: Noninteractive approaches. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, editors, *Multiobjective Optimization*, volume 5252 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2008.

[37] Douglas C. Montgomery. *Design and Analysis of Experiments*. Wiley, New York, 4th edition, 1997.

[38] Volker Nannen and A. E. Eiben. Relevance estimation and value calibration of evolutionary algorithm parameters. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 975–980, 2007.

[39] Volker Nannen and A.E. Eiben. A method for parameter calibration and relevance estimation in evolutionary algorithms. In *GECCO '06: Proceedings of the 8th annual conference on genetic and evolutionary computation*, pages 183–190, New York, NY, USA, 2006. ACM.

[40] Boris Naujoks, Nicola Beume, and Michael Emmerich. Multi-objective optimisation using S-metric selection: application to three-dimensional solution spaces. In *The 2005 IEEE Congress on Evolutionary Computation (CEC 2005)*, volume 2, pages 1282–1289, 2005.

[41] Boris Naujoks, Domenico Quagliarella, and Thomas Bartz-Beielstein. Sequential parameter optimisation of evolutionary algorithms for airfoil design. In *Proc. Design and Optimization: Methods & Applications, (ERCOFTAC'06)*, pages 231–235. University of Las Palmas de Gran Canaria, 2006.

[42] Tatsuya Okabe, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. On test functions for evolutionary multi-objective optimization. In *Parallel Problem Solving from Nature (PPSN VIII)*, pages 792–802, 2004.

[43] Mike Preuss. Reporting on Experiments in Evolutionary Computation. Technical Report CI-221/07, University of Dortmund, SFB 531, 2007.

[44] Ingo Rechenberg. *Evolutionsstrategie '94*, volume 1 of *Werkstatt Bionik und Evolutionstechnik*. Friedrich Frommann Verlag (Günther Holzboog KG), Stuttgart, 1994.

[45] Günter Rudolph. Evolutionary search under partially ordered fitness sets. In *Proceedings of the International NAISO Congress on Information Science Innovations (ISI 2001)*, pages 818–822. ICSC Academic Press: Millet/Sliedrecht, 2001.

[46] Günter Rudolph, Boris Naujoks, and Mike Preuss. Capabilities of EMOA to detect and preserve equivalent pareto subsets. In *Proc. Evolutionary Multi-Criterion Optimization, Fourth Int'l Conf.(EMO 2007)*, volume 4403 of *Lecture Notes in Computer Science*, pages 36–50, Berlin, 2007. Springer.

[47] Jerome Sacks, William J. Welch, Toby J. Mitchell, and Henry P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, November 1989.

[48] Thomas J. Santner, Brian J. Williams, and William I. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

[49] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[50] Hans-Paul Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik. Diploma thesis, Technische Universität Berlin, Hermann-Föttinger-Institut für Strömungstechnik, 1965.

[51] Hans-Paul Schwefel. Experimentelle Optimierung einer Zweiphasendüse Teil I. Technical Report No. 35 of the Project MHD-Staustrahlrohr 11.034/68, AEG Research Institute, Berlin, 1968.

[52] Deepak Sharma, Abhay Kumar, Kalyanmoy Deb, and Karthik Sindhya. Hybridization of SBX based NSGA-II and sequential quadratic programming for solving multi-objective optimization problems. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3003–3010, 2007.

[53] Selmar K. Smit and Agoston E. Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Proceedings of the 2009 IEEE Congress on Evolutionary Computation*. IEEE Press, 2009.

[54] G. W. Snedecor and W. G. Cochran. *Statistical methods*. Iowa State University Press, 8th edition, 1989.

[55] N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[56] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[57] Guido van Rossum et al. Python. `http://www.python.org`. 13 November 2008.

[58] Tobias Wagner, Nicola Beume, and Boris Naujoks. Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In *Proc. Evolutionary Multi-Criterion Optimization, 4th Int'l Conf. (EMO 2007)*, volume 4403 of *Lecture Notes in Computer Science*, pages 742–756. Springer, 2007.

[59] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, Apr 1997.

[60] Ales Zamuda, Janez Brest, Borko Boskovic, and Viljem Zumer. Differential evolution for multiobjective optimization with self adaptation. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3617–3624, Sept. 2007.

[61] Karin Zielinski and Rainer Laur. Differential evolution with adaptive parameter setting for multi-objective optimization. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3585–3592, Sept. 2007.

[62] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K.C. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

[63] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[64] Eckart Zitzler, Joshua D. Knowles, and Lothar Thiele. Quality assessment of pareto set approximations. In Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Slowinski, editors, *Multiobjective Optimization*, volume 5252 of *Lecture Notes in Computer Science*, pages 373–404. Springer, 2008.

[65] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature – PPSN VIII*, pages 832–842, Birmingham, UK, 2004. Springer.

[66] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms – a comparative case study. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 292–304, London, UK, 1998. Springer.

[67] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 24. Juli 2009

Simon Wessing