Sequential Parameter Optimization Applied to Self-Adaptation for Binary-Coded Evolutionary Algorithms

Mike Preuss and Thomas Bartz-Beielstein

Dortmund University, D-44221 Dortmund, Germany.
mike.preuss@cs.uni-dortmund.de, thomas.bartz-beielstein@udo.edu

Summary. Adjusting algorithm parameters to a given problem is of crucial importance for performance comparisons as well as for reliable (first) results on previously unknown problems, or with new algorithms. This also holds for parameters controlling adaptability features, as long as the optimization algorithm is not able to completely self-adapt itself to the posed problem and thereby get rid of all parameters. We present the recently developed *sequential parameter optimization* (SPO) technique that reliably finds good parameter sets for stochastically disturbed algorithm output. SPO combines classical regression techniques and modern statistical approaches for deterministic algorithms as Design and Analysis of Computer Experiments (DACE). Moreover, it is embedded in a twelve-step procedure that targets at doing optimization experiments in a statistically sound manner, focusing on answering scientific questions.

We apply SPO to a question that did not receive much attention yet: Is self-adaptation as known from real-coded evolution strategies useful when applied to binary-coded problems? Here, SPO enables obtaining parameters resulting in good performance of self-adaptive mutation operators. It thereby allows for reliable comparison of modified and traditional evolutionary algorithms, finally allowing for well founded conclusions concerning the usefulness of either technique.

1 Introduction

The evolutionary computation (EC) field currently seems to experience a state of flux, at least as far as experimental research methods are concerned. A purely theoretical approach is not reasonable for many optimization problems. In fact, there is a huge gap between theory and experiment in evolutionary computation. However, empiricism in EC cannot compensate this shortcoming due to a lack of standards. A broad spectrum of presentation techniques makes new results almost incomparable. At present, it is intensely discussed which experimental research methodologies should be used to improve the acceptance and quality of *evolutionary algorithms* (EA).

Several authors from related fields (Hooker (1996), Moret & Shapiro (2001), and Johnson (2002)) and from within EC (Whitley et al. (1996), Eiben & Jela-

sity (2002)) have criticized usual experimental practice. Besides other topics, they ask for increased thoughfulness when selecting benchmark problems, a better structured process of experimentation, including presentation of results, and proper use of statistical techniques.

In the light of the *no free lunch* theorem (NFL, Wolpert & Macready (1997)), research aims are gradually changing from demonstration of superiority of new algorithms towards experimental analysis, addressing questions as: What makes an algorithm work well on a certain problem class? In spite of this development, the so-called *horse race* papers[1] still seem to prevail. Eiben & Jelasity (2002) report about the situation in experimental EC and name the reasons for their discontentment with current practice. Put into positive formulation, their main demands are:

- assembly of concrete research questions and concrete answers to these, no claims that are not backed up by tests
- selection of test problems and instances motivated by these questions
- utilization of adequate performance measures for answering the questions, and
- reproducibility, which requires all necessary details and/or source code to repeat experiments

This line is carried further in Eiben & Smith (2003). Partly in response to these demands, Bartz-Beielstein (2006) proposes SPO as a structured experimentation procedure based on modern statistic techniques (§4). In its heart, SPO contains a parameter tuning method that enables *adapting* parameters to the treated test problems.

Undoubtedly, comparing badly parametrized algorithms is rather useless. However, a good parameter set allows for near-optimal performance of an algorithm. As this statement suggests, parameter tuning is of course an optimization problem on its own. Following from that, some interesting questions arise when algorithms with parameter control operators are concerned, because their own adaptibility is in turn guarded by control parameters, which can be tuned. The broadest of these questions may be where to put effort into when striving for best performance—tune the simple algorithm, or tune the adaptability of more complex algorithms—eventually resulting in recommendations when to apply adaptive operators.

1.1 Parameter Control or Parameter Tuning?

Parameter Control refers to parameter adaptation during the run of an optimization algorithm, whereas parameter tuning improves their setting before the run is started. These two different mechanisms are the roots of the two subtrees of parameter setting methods in the global taxonomy given by Eiben et al. (1999). One may get the impression that they are contradictory and researchers should aim at using parameter control as much as possible. In our view, the picture is somewhat more complex, and the two are rather complementary.

In a production environment, an EA and any alternative stochastic optimization algorithm would be run several times, not once, possibly solving the next or trying

[1] Johnson (2002) uses this term for papers that aim at showing predominance of one algorithm over (all) others by reporting performance comparisons on (standard) test problems.

the same problem again. This obliterates the separating time factor of above. What is more, the EA as well as the problem representation will most likely undergo structural changes during these iterations (new operators, criteria, etc.). These will entail changes in the "optimal" parameters, too.

Furthermore, parameter control methods do not necessarily decrease the number of parameters. For example, the 1/5th adaptation rule by Rechenberg (1973) provides at least three quantities where there has been one—mutation step size—before, namely the time window length used to measure success, the required success rate (1/5), and the rate of change applied. Even if their number remains constant, is it justified to expect that parameter tuning has become simpler now? We certainly hope so, but need more evidence to decide.

In principle, every EA parameter may be (self-)adapted. However, concurrent adaptation of many does not seem particularly successful and is probably limited to two or three at a time. We thus cannot simply displace parameter tuning with parameter control. On the other hand, manual tuning is surely not the answer, as well, and grid based tuning seems intractable for higher order design spaces. At this point, we have three possibilities, either to

- fall back on default values,
- abandon tuning alltogether and take random parameters, or
- to apply a tuning method that is more robust than doing it manually, but also more efficient than grid search.

We argue that SPO is such a method and thus of particular importance for experimental studies on non-trivial EAs, as such utilizing self-adaptation. As the experimental analysis in EC currently is a hot topic, other parameter tuning approaches are developed, too, e.g. Ramos et al. (2005). But whatever method is used, there is hardly a way around parameter tuning, at least for scientific investigations. SPO will be discussed further in §4 .

1.2 Self-adaptation for Binary Representations

When comparing recent literature, it is astounding that for real-coded EAs as e.g. *evolution strategies* (ES), self-adaptation is an almost ubiquitously applied feature, at least as far as mutation operators are concerned. In binary-coded EA however, it did not become a standard method. Optimization practicioners working with binary representations seem largely unconvinced that it may be of any use. What may be the reason for this huge difference? One could first think of traditional reasons. Of the three standard evolutionary methods for numerical optimization, genetic algorithms (GA), evolutionary programming (EP) and evolution strategies, the last one first adopted self-adaptation in various forms to improve mutation operators (Rechenberg (1978); Schwefel (1974, 1981)), followed by EP in Fogel (1992). These two mainly work with real-valued representations. Nevertheless, since then so much time has passed that it is hard to imagine that others would not employ a technique that clearly provides an advantage. Considering this, tradition appears not as a substantial reason for the divergent developments.

In fact, meanwhile, several authors have tried to transfer self-adaptation to EAs for binary encodings: Bäck (1992); Bäck & Schütz (1995); Schütz (1996); Smith & Fogarty (1996); Smith (2001); Stone & Smith (2002); Greenwood (2003). If we

accept the view of an evolutionary epistemology underlying the development of our scientific domain, so that mostly successful changes are inherited to the next stages, these studies have apparently not been very convincing. Either they were ignored by the rest of the field, or they failed to provide reason good enough for a change. Let us assume the latter case. In consequence, most EAs for binary representations published nowadays still do without self-adaptation.

Evolutionary algorithms employing different forms of self-adaptation are usually applied to continuous, or, more rarely, ordinal discrete or mixed search spaces. So the distinction may result from differences in representation. When dealing with real-valued numbers, it is quite easy to come by a good intuition why adapting mutation strengths (also called mutation step sizes) during optimization can speed up search. Features of a fitness landscape changing slowly compared to the fitness values themselves can be learned, like gradients or covariances (Hansen et al. (1995); Hansen & Ostermeier (2001)). In a binary space, this intuition is lacking. Definition of gradients is meaningless if a variable can only take two values. Consequently, reseach is trying different ways here, e.g. linkage learning or distribution estimation as in Pelikan et al. (2000). As much as the experiences from continuous search spaces cannot be simply transferred, neither can the methods. Self-adaptation of mutation rates, as straightforward continuation of existing techniques for real-valued variables, if working at all for binary representations, can be expected to work differently.

2 Aims and Methods

Our aims are twofold: To demonstrate usefulness of the SPO approach for experimental analysis, and to perform an experimental analysis of self-apaptation mechanisms.

Methodologically, we want to convey that SPO is a suitable tool for finding good parameter sets (designs) within a fixed, low budget of algorithm runs. It seems safe to assume that a parameter set exists in the parameter design space that is better than the best of a small ($\approx$ 100) sample. We therefore require that the best configurations detected by SPO are significantly better than the best of a first sample. Furthermore, we want to suggest a flexible yet structured methodology for performing and documenting experimentation by employing a parameter tuning technique.

Concerning the experimental analysis of self-adaptation mechanisms on binary represented problems, our aim is to collect evidence for or against the following conjectures:

- Well parametrized self-adaptation significantly speeds up optimization when compared to well tuned constant mutation rates for many problems.
- Detecting good parameter sets for self-adaptive EAs is usually not significantly harder than doing so for non self-adaptive EAs.
- Problem knowledge, e.g., shared properties or structural similarities of problem classes, gives useful hints for selecting a self-adaptation mechanism.

Before these conjectures are detailed, we give an overview of existing parameter tuning methods.

3 Parameter Optimization Approaches

Modern search heuristics have proved to be very useful for solving complex real–world optimization problems that cannot be tackled through classical optimization techniques (Schwefel et al., 2003). Many of these search heuristics involve a set of *exogenous parameters*, i.e., values are specified before the run is performed, that affect their convergence properties. The population size in EA is a typical example for an exogenous strategy parameter. The determination of an adequate population size is crucial for many optimization problems. Increasing the population size from 10 to 50 while keeping the number of function evaluations constant might improve the algorithm's performance—whereas a further increase might result in a performance decrease, if the number of function evaluations remains constant.

SPO is based on statistical *design of experiments* (DOE) which has its origins in agriculture and industry. However, DOE has to be adapted to the special requirements of computer programs. For example, computer program are per se deterministic, thus a different concepts of randomness has to be considered. Law & Kelton (2000) and Kleijnen (1987, 1997) demonstrated how to apply DOE in simulation. Simulation is related to optimization (simulation models equipped with an objective function define a related optimization problem), therefore we can benefit from simulation studies.

DOE related parameter studies were performed to analyze EA: Schaffer et al. (1989) proposed a complete factorial design experiment, Feldt & Nordin (2000) use statistical techniques for designing and analyzing experiments to evaluate the individual and combined effects of genetic programming parameters. Myers & Hancock (2001) presented an empirical modeling of genetic algorithms. François & Lavergne (2001) demonstrate the applicability of *generalized linear models* to design evolutionary algorithms. These approaches require up to 100,000 program runs, whereas SPO is applicable even if a small amount of function evaluations are available only.

Because the search for useful parameter settings of algorithms itself is an optimization problem, meta-algorithms have been proposed. Bäck (1996) and Kursawe (1999) presented meta-algorithms for evolutionary algorithms. But these approaches do not solve the original problem completely, because they require the determination of additional parameter setting of the meta-algorithm. Furthermore, we argue that the experimenter's skill plays an important role in this analysis. It cannot be replaced by automatic "meta" rules.

SPO can also be run on auto-pilot without any user intervention. This convenience cannot be obtained for free: The user gains limited insight into the working mechanisms of the tuned algorithm and, which is even more severe, the validity and thus the predictive power of the regression model might be quite poor.

The reader should note that our approach is related to the discipline of *experimental algorithmics*, which offers methodologies for the design, implementation, and performance analysis of computer programs for solving algorithmic problems (Demetrescu & Italiano, 2000; Moret, 2002). Further valueable approaches have been proposed by McGeoch (1986), Barr & Hickman (1993), and Hooker (1996).

Design and analysis of computer experiments (DACE) as introduced in Sacks et al. (1989) models the deterministic output of a computer experiment as the realization of a stochastic process. The DACE approach focuses entirely on the correlation structure of the errors and makes simplistic assumptions about the regressors. It describes "how the function behaves," whereas regression as used in classical DOE

describes "what the function is" (Jones et al., 1998, p. 14). DACE requires other experimental designs than classical DOE, e.g., *Latin hypercube designs* (LHD) (McKay et al., 1979).

We claim that it is beneficial to combine some of these well-established ideas from DOE, DACE, and further statistical techniques to improve the acceptance and quality of evolutionary algorithms.

4 Sequential Parameter Optimization Methodology

How can optimization practitioners determine if concepts developed in theory work in practice? Hence, experiments are necessary. Experiment has a long tradition in science. To analyze experimental data, statistical methods can be applied. It is not a trivial task to answer the final question "Is algorithm A better than algorithm B?" Results that are statistically significant are not automatically scientifically meaningful.

Example 4.1 (Floor and ceiling effects). The statistical meaningful result "all algorithms perform equally" can be scientifically meaningless, because the problem instances are too hard for any algorithm. A similar effect occurs if the problem instances are too easy. The resulting effects are known as floor or ceiling effects, respectively. □

SPO is more than a simple combination of existing statistical approacches. It is based on the new experimentalism, a development in the modern philosophy of science, which considers that an experiment can have a life of its own. SPO provides a statistical methodology to learn from experiments, where the experimenter should distinguish between statistical significance and scientific meaning.

An optimization run is considered as an experiment. An optimal parameter setting, or statistically speaking, an optimal *algorithm design*, depends on the problem at hand as well as on the restrictions posed by the environment (i.e., time and hardware constraints). Algorithm designs are usually either determined empirically or set equal to widely used default values. SPO is a methodology for the experimental analysis of optimization algorithms to determine improved algorithm designs and to learn, how the algorithm works. The proposed technique employs computational statistic methods to investigate the interactions among optimization problems, algorithms, and environments.

An optimization practitioner is interested in robust solutions, i.e., solutions independent from the random seeds that are used to generate the random numbers during the optimization run. The proposed statistical methodology provides guidelines to design robust algorithms under restrictions, such as a limited number of function evaluations and processing units. These restrictions can be modeled by considering the performance of the algorithm in terms of the (expected) best function value for a limited number of function evaluations.

To justify the usefulness of our approach, we analyze the properties of several algorithms from the viewpoint of a researcher who wants to develop and understand self-adaptation mechanisms for evolutionary algorithms. SPO provides numerical and graphical tools to test if the statistical results are really relevant or have been caused by the experimental setup only. It is based on a framework that permits a

delinearization of the complex steps from raw data to scientific hypotheses. Substantive scientific questions are broken down into several local hypotheses, that can be tested experimentally. The optimization process can be regarded as a process that enables learning. SPO consists of the twelve steps that are reported in Table 1. These steps and the necessary statistical techniques will be presented in the following. SPO has been applied on search heuristics in the following domains:

1. machine engineering: design of mold temperature control (Mehnen et al., 2005; Weinert et al., 2004; Mehnen et al., 2004)
2. aerospace industry: airfoil design optimization (Bartz-Beielstein & Naujoks, 2004)
3. simulation and optimization: elevator group control (Bartz-Beielstein et al., 2005c; Markon et al., 2006)
4. technical thermodynamics: nonsharp separation (Bartz-Beielstein et al., 2005b)
5. economy: agri-environmental policy-switchings (de Vegt, 2005)

Other fields of application are in fundamental research:

1. algorithm engineering: graph drawing (Tosic, 2006)
2. statistics: selection under uncertainty (optimal computational budget allocation) for PSO (Bartz-Beielstein et al., 2005a)
3. evolution strategies: threshold selection and step-size adaptation (Bartz-Beielstein, 2005)

Table 1: *Sequential parameter optimization (*SPO*). This approach combines methods from computational statistics and exploratory data analysis to improve (tune) the performance of direct search algorithms.*

Step	Action
(S-1)	Preexperimental planning
(S-2)	Scientific claim
(S-3)	Statistical hypothesis
(S-4)	Specification of the
	(a) optimization problem
	(b) constraints
	(c) initialization method
	(d) termination method
	(e) algorithm (important factors)
	(f) initial experimental design
	(g) performance measure
(S-5)	Experimentation
(S-6)	Statistical modeling of data and prediction
(S-7)	Evaluation and visualization
(S-8)	Optimization
(S-9)	Termination: If the obtained solution is good enough, or the maximum number of iterations has been reached, go to step (S-11)
(S-10)	Design update and go to step (S-5)
(S-11)	Rejection/acceptance of the statistical hypothesis
(S-12)	Objective interpretation of the results from step (S-11)

4. other evolutionary algorithms: genetic chromodynamics (Stoean et al., 2005)
5. computational intelligence: algorithmic chemistry (Bartz-Beielstein et al., 2005b)
6. particle swarm optimization: analysis und application (Bartz-Beielstein et al., 2004a)
7. numerics: comparison and analysis of classical and modern optimization algorithms (Bartz-Beielstein et al., 2004b)

Further projects, e.g., vehicle routing and door-assignment problems and the application of methods from computational intelligence to problems from bioinformatics are subject of current research. An SPO-toolbox is freely available under the following link: `http://www.springer.com/3-540-32026-1`.

4.1 Tuning

In order to find an optimal algorithm design, or to tune the algorithm, it is necessary to define a performance measure. Effectivity (robustness) and efficiency can guide the choice of an adequate performance measure. Note that optimization practitioners do not always choose the absolute best algorithm. Sometimes a robust algorithm or an algorithm that provides insight into the structure of the optimization problem is preferred. From the viewpoint of an experimenter, design variables (factors) are the parameters that can be changed during an experiment. Generally, there are two different types of factors that influence the behavior of an optimization algorithm:

- problem specific factors, e.g., the objective function
- algorithm specific factors, i.e., the population size or other exogenous parameters

We will consider experimental designs that comprise problem specific factors and exogenous algorithm specific factors. Algorithm specific factors will be considered first. *Endogenous* can be distinguished from *exogenous parameters* (Beyer & Schwefel, 2002). The former are kept constant during the optimization run, whereas the latter, e.g., standard deviations, are modified by the algorithms during the run. Consider $\mathcal{D}_A$, the set of all parameterizations for one algorithm. An *algorithm design* X_A is a set of vectors, each representing one specific setting of the design variables of an algorithm. A design can be specified by defining ranges of values for the design variables. A *design point* $x_a \in \mathcal{D}_A$ presents exactly one parameterization of an algorithm. Note that a design can contain none, one, several or even infinitely many design points. The *optimal algorithm design* is denoted as X_A^*. The term "optimal design" can refer to the best design point x_a^* as well as the most informative design points (Pukelsheim, 1993; Santner et al., 2003).

Let $\mathcal{D}_P$ denote the set of all problem instances for one optimization problem. *Problem designs* X_P provide information related to the optimization problem, such as the available resources (number of function evaluations) or the problem's dimension.

An *experimental design* $X_E \in \mathcal{D}$ consists of a problem design X_P and an algorithm design X_A. The run of a stochastic search algorithm can be treated as an experiment with a stochastic output $Y(x_a, x_p)$, with $x_a \in \mathcal{D}_A$ and $x_p \in \mathcal{D}_P$. If the random seed is specified, the output would be deterministic. This case will not be considered further, because it is not a common practice to specify the seed that is used in an optimization run. Our goals of the experimental approach can be stated as follows:

(G-1) *Efficiency.* To find a design point $x_a^* \in \mathcal{D}_A$ that improves the performance of an optimization algorithm for one specific problem design point $x_p \in \mathcal{D}_P$.
(G-2) *Robustness.* To find a design point $x_a^* \in \mathcal{D}_A$ that improves the performance of an optimization algorithm for several problem design points $x_p \in \mathcal{D}_P$.

Performance can be measured in many ways, e.g., as the best or the average function value for n runs. Statistical techniques to attain these goals will be presented next.

4.2 Stochastic Process Models as Extensions of Classical Regression Models

The classical DOE approach consists of three steps: Screening, modeling, and optimization. Each step requires different experimental designs. Linear regression models are central elements of the classical design of experiments approach (Draper & Smith, 1998; Montgomery, 2001). We propose an approach that extends the classical regression techniques, because the assumption of a linear model for the analysis of computer programs and the implicit model assumption that observation errors are independent of one another are highly speculative (Bartz-Beielstein, 2006). To keep the number of experiments low, a sequential procedure has been developed. Our approach relies on a stochastic process model, that will be presented next.

We consider each algorithm design with associated output as a realization of a stochastic process. *Kriging* is an interpolation method to predict unknown values of a stochastic process and can be applied to interpolate observations from computationally expensive simulations. Our presentation follows concepts introduced in Sacks et al. (1989), Jones et al. (1998), and Lophaven et al. (2002b).

Consider a set of m design points $x = (x^{(1)}, \ldots, x^{(m)})^T$ with $x^{(i)} \in \mathbb{R}^d$. In the *design and analysis of computer experiments* (DACE) *stochastic process model*, a deterministic function is evaluated at the m design points x. The vector of the m responses is denoted as $y = (y^{(1)}, \ldots, y^{(m)})^T$ with $y^{(i)} \in \mathbb{R}$. The process model proposed in Sacks et al. (1989) expresses the deterministic response $y(x^{(i)})$ for a d-dimensional input $x^{(i)}$ as a realization of a regression model $\mathcal{F}$ and a stochastic process Z,

$$Y(x) = \mathcal{F}(\beta, x) + Z(x). \tag{1}$$

DACE Regression Models

We use q functions $f_j : \mathbb{R}^d \to \mathbb{R}$ to define the regression model

$$\mathcal{F}(\beta, x) = \sum_{j=1}^{q} \beta_j f_j(x) = f(x)^T \beta. \tag{2}$$

Regression models with polynomials of orders 0, 1, and 2 have been used in our experiments. Regression models with a constant term only, i.e., $f_1 = 1$, have been applied successfully to model the data and to predict new data points in the sequential approach.

DACE Correlation Models

The random process $Z(\cdot)$ (Equation 1) is assumed to have mean zero and covariance $V(w,x) = \sigma^2 \mathcal{R}(\theta, w, x)$ with process variance σ^2 and correlation model $\mathcal{R}(\theta, w, x)$. Consider an algorithm with d factors (parameters). Correlations of the form

$$\mathcal{R}(\theta, w, x) = \prod_{j=1}^{d} \mathcal{R}_j(\theta, w_j - x_j)$$

will be used in our experiments. The correlation function should be chosen with respect to the underlying process (Isaaks & Srivastava, 1989). Lophaven et al. (2002a) discuss seven different models. The Gaussian correlation function is a well-known example. It is defined as

$$\text{GAUSS} \ : \qquad \mathcal{R}_j(\theta, h_j) = \exp(-\theta_j h_j^2), \tag{3}$$

with $h_j = w_j - x_j$, and for $\theta_j > 0$. The regression matrix R is the matrix with elements

$$R_{ij} = R(x_i, x_j) \tag{4}$$

that represent the correlations between $Z(x_i)$ and $Z(x_j)$. The vector with correlations between $Z(x_i)$ and a new design point $Z(x)$ is

$$r(x) = \left(R(x_1, x), \ldots, R(x_m, x)\right). \tag{5}$$

Large θ_j's indicate that variable j is active: function values at points in the vicinity of a point are correlated with Y at that point, whereas small θ_j's indicate that also distant data points influence the prediction at that point. The *empirical best unbiased linear predictor* (EBLUP) can be shown to be

$$\hat{y}(x) = f^T(x)\hat{\beta} + r^T(x)R^{-1}(y - F\hat{\beta}), \tag{6}$$

where

$$\hat{\beta} = \left(F^T R^{-1} F\right)^{-1} F^T R^{-1} y \tag{7}$$

is the generalized least-squares estimate of β in Equation 1, $f(x)$ are the q regression functions in Equation 2, and F represents the values of the regression functions in the m design points.

Maximum likelihood estimation methods to estimate the parameters θ_j of the correlation functions from Equation 3 are discussed in Lophaven et al. (2002a). DACE methods provide an estimation of the prediction error on an untried point x, the *mean squared error* (MSE) of the predictor

$$\text{MSE}(x) = E\left(\hat{y}(x) - y(x)\right). \tag{8}$$

The stochastic process model, which was introduced as an extension of the classical regression model will be used in our experiments. Next, we have to decide how to generate design points, i.e., which parameter settings should be used to test the algorithm's performance.

Space Filling Designs and Expected Improvement

Often, designs that use sequential sampling are more efficient than designs with fixed sample sizes. Therefore, we specify an initial design $X_A^{(0)} \in \mathcal{D}_A^{(0)}$ first. *Latin hypercube sampling* (LHS) was used to generate the initial algorithm designs. Consider n number of levels being examined and d design variables. A Latin hypercube is a matrix of n rows and d columns. The d columns contain the levels $1, 2, \ldots, n$, randomly permuted, and the d columns are matched at random to form the Latin hypercube. The resulting *Latin hypercube designs* are space-filling designs. McKay et al. (1979) introduced LHDs for computer experiments, Santner et al. (2003) give a comprehensive overview.

Information obtained in the first runs can be used for the determination of the second design $X_A^{(1)}$ in order to choose new design points sequentially and thus more efficiently.

Sequential sampling approaches have been proposed for DACE. For example, in Sacks et al. (1989) sequential sampling approaches were classified to the existing meta-model. We will present a sequential approach that is based on the expected improvement. In Santner et al. (2003, p. 178) a heuristic algorithm for unconstrained global minimization problems is presented. Consider one problem design point x_p. Let $y_{\min}^{(t)}$ denote the smallest known minimum value after t runs of the algorithm, $y(x)$ be the algorithm's response, i.e., the realization of $Y(x)$ in Equation (1), and let x_a represent a specific design point from the algorithm design X_A. Then the improvement is defined as

$$I(x_a) = \begin{cases} y_{\min}^{(t)} - y(x_a), & y_{\min}^{(t)} - y(x_a) > 0 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

for $x_a \in \mathcal{D}_A$. As $Y(\cdot)$ is a random variable, its exact value is unknown. The goal is to optimize its expectation, the so-called *expected improvement*. New design points, which are added sequentially to the existing design, are attractive "if either there is a high probability that their predicted output is below [minimization] the current observed minimum and/or there is a large uncertainty in the predicted output." This leads to the *expected improvement heuristic* (Bartz-Beielstein, 2006). Based on theorems from Schonlau (1997, p. 22) we implemented a program to estimate and plot the main factor effects. Furthermore, three dimensional visualizations produced with the DACE toolbox (Lophaven et al., 2002b) can be used to illustrate the interaction between two design variables and the associated mean squared error of the predictor.

Algorithm 1 describes the SPO in a formal manner. The selection of a suitable problem instance is done in the pre-experimental planning phase to avoid floor and ceiling effects (l.2). Latin hypercube sampling can be used to determine an initial set of design points (l.3). After the algorithm has been run with these k initial parameter settings (l.5), the DACE process model is used to discover promising design points (l.10). Note that other sample statistics than the mean, e.g., the median, can be used in l.6. The m points with the highest expected improvement are added to the set of design points, where m should be small compared to s. The update rule for the number of reevalutions $r^{(t)}$ (l.13-15) guarantees that the new best design point $x_b^{(t+1)}$ has been evaluated at least as many times as the previous best design point $x_b^{(t)}$. Obviously, this is a very simple update rule and more elaborate rules are possible. Other termination criteria exist besides the budget based termination (l.17).

Algorithm 1 Sequential parameter optimization

procedure SPO($\mathcal{D}_A$, $\mathcal{D}_P$) /* *Algorithm und problem design* */
  Select $p \in \mathcal{D}_P$ and set $t = 0$ /* *Select problem instance* */
  $X_A^{(t)} = \{x_1, x_2, \ldots, x_k\}$ /* *Sample k initial points, e.g., LHS* */
  repeat
    $y_{ij} = Y_j(x_i, p) \forall x_i \in X_A^{(t)}$ and $j = 1, \ldots, r^{(t)}$ /* *Fitness evaluation* */
    $\overline{Y}_i^{(t)} = \sum_{j=1}^{r^{(t)}} y_{ij}^{(t)} / r^{(t)}$ /* *Sample statistic for the ith design point* */
    x_b with $b = \arg\min_i(\overline{y}_i)$ /* *Determine best point* */
    $Y(x) = \mathcal{F}(\beta, x) + Z(x)$ /* *DACE model from Eq. 1* */
    $X_S = \{x_{k+1}, \ldots, x_{k+s}\}$ /* *Generate s sample points, $s \gg k$* */
    $y(x_i)$, $i = 1, \ldots, k + s$ /* *Predict fitness from the DACE model* */
    $I(x_i)$ for $i = 1, \ldots, s + k$ /* *Expected improvement (Eq. 9)* */
    $X_A^{(t+1)} = X_A^{(t)} \cup \{x_{k+i}\}_{i=1}^{m} \notin X_A^{(t)}$ /* *Add m promising points* */
    if $x_b^{(t)} = x_b^{(t+1)}$ **then**
      $r^{(t+1)} = 2r^{(t)}$ /* *Increase number of repeats* */
    end if
    t = t+1 /* *Increment iteration counter* */
  until Budget exhausted
end procedure

4.3 Experimental Reports

Surprisingly, despite around 40 years of empirical tradition, in EC a standardized scheme for reporting experimental results never developed. The natural sciences, e.g. physics, possess such schemes as de-facto standards. We argue that for both groups, readers and writers, an improved report structure is beneficial: As with the common overall paper structure (introduction, conclusions, etc.), a standard provides guidelines for readers, what to expect, and where. Writers are steadily reminded to describe the important details needed to understand and possibly replicate their experiments. For the structured documentation of experiments, we propose organizing their presentation into 7 parts, as follows.

ER-1: **Focus/Title**
Briefly names the matter dealt with, the (possibly very general) objective, preferably in one sentence.

ER-2: **Pre-experimental planning**
Reports the first—possibly explorative—program runs, leading to task and setup (steps ER-3 and ER-4) . Decisions on used benchmark problems or performance measures may often be influenced by the outcome of preliminary runs. This may also include negative results, e.g. modifications to an algorithm that did not work, or a test problem that turned out to be too hard, if they provide new insight.

ER-3: **Task**
Concretizes the question in focus and states scientific and derived statistical hypotheses to test. Note that one scientific hypothesis may require several, sometimes hundreds of statistical hypotheses. In case of a purely explorative study,

as with the first test of a new algorithm, statistical tests may be not applicable. Still, the task should be formulated as precise as possible.

ER-4: **Setup**
Specifies problem design and algorithm design, containing fixed and variable parameters and criteria of tackled problem, investigated algorithm and chosen performance measuring. The information in this part should be sufficient to replicate an experiment.

ER-5: **Experimentation/Visualization**
Gives raw or produced (filtered) data on the experimental outcome, additionally provides basic visualizations where meaningful.

ER-6: **Observations**
Describes exceptions from the expected, or unusual patterns noticed, without subjective assessment or explanation. As an example, it may be worthwile to look at parameter interactions. Additional visualizations may help to clarify what happens.

ER-7: **Discussion**
Decides about the hypotheses specified in step 4.3, and provides necessarily subjective interpretations for the recorded observations.

This scheme is tightly linked to the 12 steps of experimentation suggested in Bartz-Beielstein (2006) and depicted in Table 1, but on a slightly more abstract level. The scientific and statistical hypothesis steps are treated together in part ER-3, and the SPO core (parameter tuning) procedure, much of which may be automated, is included in part ER-5. In our view, it is especially important to divide parts ER-6 and ER-7, to facilitate different conclusions drawn by others.

5 Self-Adaptation Mechanisms and Test Problems

In order to prepare a meaningful experiment, we want to take up the previously cited warning words from Johnson (2002) and Eiben & Jelasity (2002) concerning the proper selection of ingredients for a good setup and carefully select mechanisms and problems to test.

5.1 Mechanisms: Self-Adaptive, Constant, Asymmetrical

Meyer-Nieberg & Beyer (2006), to be found in this volume, give an extensive account of the history and current state of adaptation mechanisms in evolutionary algorithms. However, in this work, we consider only self-adaptive mechanisms for binary represented (often combinatorial) problems. Self-adaptiveness basically means to introduce unbiased deviations into control parameters and let the algorithm chose the value that apparently works best. Another possibility would be to use a rule set as in the previously stated 1/5th rule by Rechenberg (1973), which grants adaptation, but not self-adaptation. The mechanism suggested by Greenwood (2003) is of this type. Others aim for establishing a feedback loop between the probability of using one operator and the fitness advancement this operator is responsible for, e.g. Julstrom (1997); Igel & Kreutz (2003); Thierens (2005).

Thus, whereas several adaptation techniques have been proposed for binary representations, few are purely self-adaptive. The mechanisms proposed in Bäck & Schütz (1995) and Smith (2001) have this property, though they origin from very different sources. The former is a variant of a self-adaptation scheme designed for real-valued search spaces; the latter has been shaped especially for binary search spaces and to overcome premature convergence. Mutation of the mutation rate in Bäck & Schütz (1995) is accomplished according to the formula:

$$p'_k = \frac{1}{1 + \frac{1-p_k}{p_k} \exp(-\gamma N_k(0,1))}, \tag{10}$$

where $N_k(0,1))$ is a standard normally distributed random variable. p'_k stands for the new, and p_k the current mutation rate, which must be prevented from becoming 0 as this is a fixpoint of the iteration. The approach of Smith (2001) is completely different; it employs a discrete, small number q of mutation rates, one of which is selected according to the *innovation rate* z, so that the probability for alteration in one generation respects $p_a = z \cdot (q-1)/q$. The q values are given as $p_m \in \{0.0005, 0.001, 0.0025, 0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0.1\}$ and thus unbalanced in the interval $[0, 1]$. We employ a different set to facilitate comparison with the first mechanism that produces balanced mutation rates. Thereby, the differences are reduced to continuity or discreteness with and without causal relationship between new and old values (the second mechanism is stateless). In the following experiment, we also set q to ten, with $p_m \in \{0.001, 0.01, 0.05, 0.1, 0.3, 0.7, 0.9, 0.95, 0.99, 0.999\}$.

Depending on the treated problem, it sometimes pays to use asymmetric mutation, that is, different mutation rates for ones and zeroes, as has been shown e.g. for certain instances of the Subset Sum problem by Jelasity (1997). In Jelasity et al. (2002) a meta-algorithm—itself an EA—was used to learn good mutation rate pairs. In this work, we also want to enable comparison between constant and self-adaptive asymmetric mutation rates by simply extending the two named symmetric self-adaptive mechanisms to two independently adapted mutation rates in the fashion of the asymmetric constant rate operator. It shall not be concealed that recent theoretical investigations also deal with asymmetric mutation operators, e.g. Jansen & Sudholt (2005), even though their notion of asymmetry is slightly different.

5.2 Problems and Expectations

We chose a set of six test problems that maybe divided into three pairs, according to similar properties. Our expectation is that shared problem attributes lead to comparable performance and parametrization of algorithms. However, we are currently not able to express problem similarity quantitatively. Thus, our experimental study may be seen as a first explorative attempt in this direction.

wP-PEAKS and SUFSAMP

wP-PEAKS stands for *weighted P-PEAKS* generator, a modification of the original problem by Jong et al. (1997), which employed random N-bit strings to represent the location of P peaks in search space. A small/large number of peaks results in weakly/strongly epistatic problems. Originally, each peak represented a global

optimum. We employ a modified version (Giacobini et al. (2006)) by adding weights $w_i \in \mathbb{R}_+$ with only $w_1 = 1.0$ and $w_{[2...P]} < 1.0$, thereby requiring the optimization algorithm to find the one peak bearing the global optimum instead of just any peak. In our experiments, P and N were 100, and $w_i \in [0.9, 0.99]$ for local optima.

The SUFSAMP problem has been introduced by Jansen et al. (2005) as test for the fraction of neighborhoods actually visited by an EA when building the offspring generation. The key point is that only one of the direct neighbours, the one that provides the highest fitness gain, leads towards the global optimum. Thus, it is essential to perform sufficient sampling (high selection pressure) to maintain the chance of moving in the right direction. We used an instance with bitlength 30 for our tests, this is just beyond the point where the problem is solved easily.

Both problems are by far not similar, but still somewhat related because they are extreme in requiring global and local exploration. In case of the wP-PEAKS, the whole search space is to cover to find the best peak, in SUFSAMP the local neighborhood must be covered well to get hold of the path to the global optimum.

MMDP and COUNTSAT

The Massively Multimodal Deceptive Problem (MMDP) as suggested by Goldberg et al. (1992) is a multiple of a 6 bit deceptive subproblem which has its maximum for 6 or 0 bits set, its minimum for 1 or 5 bits set, and a deceptive local peak at 3 set bits. The fitness of a solution is simply the sum over all subproblems. From the mode of construction, it becomes clear that mutation is deliberately mislead here whether recombination is essential. We use an instance with 40 blocks of 6 bit, each.

The COUNTSAT problem is an instance of the MAXSAT problem suggested by Droste et al. (2000). Its fitness only depends on the number of set bits, all set to 1 in the global optimum. Our test instance is of length 40.

These two problems share the property that their fitness values are invariant to permutations of the whole (COUNTSAT) or separate parts (MMDP) of the genome. Consequently, Giacobini et al. (2006) empirically demonstrate the general aptitude of self-adaptive mutation mechansisms on these problems.

Number Partitioning and Subset Sum

The (Min) Number Partitioning Problem (MNP) requires arranging a set of long numbers, here 35 with 10 digits length each, into two groups adding up to the same sum. Fitness of a solution candidate is measured as the remaining difference in sums, meaning this is a minimization problem. It was treated e.g. in Berretta & Moscato (1999); Berretta et al. (2004) by means of a *memetic algorithm*. We used a randomly determined instance for each optimization run.

The Subset Sum problem is related to the MNP in that from a given collection of positive integers, a subset must be chosen to achieve a predetermined sum. This time, the target sum does not directly depend on the overall sum of the given numbers but can be any attainable number. Additionally, all solution candidates approaching it from above are counted as infeasible. We use the same setup as in Jelasity et al. (2002), made up of 100 numbers from within $[0, 2^{100}]$ and a density of 0.1, thus constructing the target sum from 10 of the original numbers. For each optimization run, an instance of this form is randomly created.

As stated, both are set selection problems. However, the expected fraction of ones in a good solution is different: Around 0.5 for the MNP, and 0.1 for the Subset

Sum. Therefore, me may expect asymmetric mutation rates (self-adpative or not) to perform well on the latter, and symmetric on the former.

6 Assessment via Parameter Optimization

Within this section, we perform and discuss an SPO-based experiment to collect evidence for or against the claims made in §2. However, allowing for multiple starts during an optimization run on one and simultaneously employing algorithms with enormous differences in speed and quality on the other hand complicates assessing performance by means of standard measures like the MBF (mean best fitness), AES (average evaluations to solution), or SR (success rate). This is also confirmed by other views, e.g. Rardin & Uzsoy (2001), who express that dealing with aggregated measures, especially for multistart algorithms, requires some creativity. Additionally, we want to investigate the "tunability" of an algorithm-problem pair, for which no common measures exist. We therefore resort to introducing the needed, prior to describing the experiment itself.

6.1 Adequate Measures

LHS Average and Best

For assessing tunability of an algorithm towards a problem, the best performance found by a parameter optimization method shall be related to a base performance that is easily obtained without, or by manual tuning. Additionally, comparison to a simple hillclimber makes sense, as the more complex EAs should be able to attain better solutions than these to claim any importance.

The performance achieved by manual tuning surely depends on the expertise of the experimenter. As a substitute for this hardly quantifiable measure we propose to employ the best performance contained in an LHS of size $10 \times$ #parameters, which in our case resembles the result of an initial SPO step. In contrast to the best, the average performance of all LHS design points is an approximation of the expected quality of a random configuration. Moreover, the variance of this quantity hints to low large the differences are. Large variances may indicate two things at once: a) There are very bad configurations the user must avoid, and b) there exist very good configurations that cover only a small amount of the parameter space and are thus hard to find. Low variances, however, mean that the algorithm is neither tunable nor mis-configurable. In consequence, we suggest to observe the measures f_{hc}, the MBF of a standard hillclimber, f_{LHSa}, the average fitness of an LHS design, σ_{LHSa}, its standard deviation, f_{LHSb}, fitness of the best configuration of an LHS, and f_{SPO}, the best performance achieved when tuning is finished.

AEB: Average Evaluations to Best

The AES needs a definition of success and is thus a problem-centric measure. If the specified success rarely occurs, its meaning is questionable. Lowering the quality of what counts as success may lead from floor to ceiling effect if performance differences of the algorithms in scope are large and the meaning of success is not carefully

balanced. We turn this measure into an algorithm-centric one by taking the average of the number of evaluations needed to obtain the peak performance in each single optimization run. Stated differently, this is the average time an algorithm continues to make progress. This quantity still implies dependency of the time horizon of the experiment (the total number of evaluations allowed), but in contrast to the AES, it is always defined. The AEB alone is of limited value, as fast converging algorithms are prefered, regardless of the quality of their final best solution. However, it is useful for defining the following measure.

RFA: Ressource Favoring Aggregate

Assessing robustness and efficiency of an optimization algorithm at the same time is difficult, especially in situations where the optimum is seldomly reached, or even worse, is unknown. Common fitness measures as the MBF, AES, or SR, apply only to one of the two aspects. Combined measures as the success performance suggested by Auger & Hansen (2005) require definition of success, as well. Two possibilities remain to resolve this dilemma: Multicriterial assessment, or aggregation. We choose the latter and use a robustness favoring aggregate (RFA), defined as:

$$RFA = \Delta f \cdot \left| \frac{\Delta f}{AEB} \right| , \quad \Delta f = \begin{cases} MBF - f_{hc} & : \quad minimization \\ f_{hc} - MBF & : \quad maximization \end{cases} \tag{11}$$

The RFA consists of fitness progress, relative to the hillclimber, multiplied by the linear relative progress rate, which depends on the average number of evaluations to reach the best fitness value (AEB). It weights robustness higher than efficiency, but takes both into account. Using solely the linear relative progress rate is no alternative; it rates fast algorithms achieving poor quality equal to slow algorithms able to attain good quality. If an algorithm performs better than the hillclimber (in quality), its RFA is negative, positive otherwise. Note that this is only one of a virtually unlimited number of ways to define an aggregate.

6.2 Experiment

Apply SPO to EAs including different self-adaptive mechanisms and fixed rate mutation, compare tunability and achievable quality.

Pre-experimental planning:
The originally intended performance measure (MBF) has been exchanged with the RFA fitness measure, due to first experiences with certain EA configurations on the MMDP problem, namely with asymmetric mutation operators. RFA provides a gradient even between the best found configurations that always solved the problem to optimality before, even for increased problem sizes (100 or more 6-bit groups).

Task:
Based on the aims named in §2, we investigate two scientific claims:

1. Tuned self-adaptative mutation allows for better performance than tuned constant mutation for many problems.
2. The tuning process is not significantly harder for self-adaptive mutation.

Table 2: *Algorithm design for the six EA variants. Only asymmetric mutation operators need a second mutation rate and its initialization range (*). The learning rate (**) is only applicable to adaptive mutation operators, hence not used for variants with constant mutation rates. Algorithm designs thus have 7, 8, 9, or 10 parameters.*

Parameter name	$\mathbb{N}/\mathbb{R}$	Min	Max	Parameter name	$\mathbb{N}/\mathbb{R}$	Min	Max
Population size μ	$\mathbb{N}$	1	500	Maximum age κ	$\mathbb{N}$	1	50
Selection pressure λ/μ	$\mathbb{R}_+$	1	10	Recombination prob. p_r	$\mathbb{R}$	0	1
Stagnation restart r_s	$\mathbb{N}$	5	30	Learning rate** τ	$\mathbb{R}$	0.0	1.0
Mutation rate p_m	$\mathbb{R}_+$	10^{-3}	1.0	Mut. rate range $r(p_m)$	$\mathbb{R}$	0	0.5
Mutation rate 2* p_{m2}	$\mathbb{R}_+$	10^{-3}	1.0	Mut. rate 2 range* $r(p_{m2})$	$\mathbb{R}$	0	0.5

Additionally, the third aim shall be achieved by searching for patterns in the virtual relation between problem class properties and performance results. We perform statistical hypothesis testing by employing bootstrap permutation tests with the commonly used significance level of 5%. For accepting the first hypothesis, we demand a significant difference between the fastest self-adaptive and the fastest non-adaptive EA variant for at least half of the tested problems. The second claim is more difficult to test; in absence of a well-defined measure, we have to resort to just expressing the impression obtained from evaluating data and visualizations.

Setup:
The optimization algorithm used is a panmictic (μ, κ, λ)-ES with different mutation operators and 2-point crossover for recombination. As κ is one of the factors the parameter optimization is allowed to change, the degree of elitism (complete for $\kappa \geq \max(\#\text{generations})$, none for $kappa = 1$), that is the time any individual may survive within the population, maybe varied. As common for binary representations, mutation works with bit flip probabilities (rates). We employ 6 different modes of performing mutation, namely constant rates, the self-adaptative method of Schütz (1996), and the self-adaptive method of Smith (2001), each in turn symmetrical and asymmetrical (2 mutation rates, one for 0s and 1s, respectively). Table 2 shows the algorithm design for all variants. Note that not every parameter applies to all mutation operators. Parameter ranges are chosen relatively freehanded, bounded only by ressource limits (μ, λ) or to enable a reasonable gradient (r_s , κ) in configuration qualities. Two features of the used algorithm variants may appear unusual, namely the recombination probability, allowing for gradually switching recombination on and off, and the stagnation restart, enabling restarts after a given number of generations without improvement.

SPO parameters are kept at default values where possible. However, the total budget of allowed runs always is a design choice. We set this to 1000 as compromise between effectivity and operability. The initial LHS size is set to $10 \cdot \#\text{parameters}$ —following practical experience, also suggested by Schonlau et al. (1998)— with 4 repeats per configuration, the maximum number of repeats to 64. Testing is performed at siginificance level 5 %.

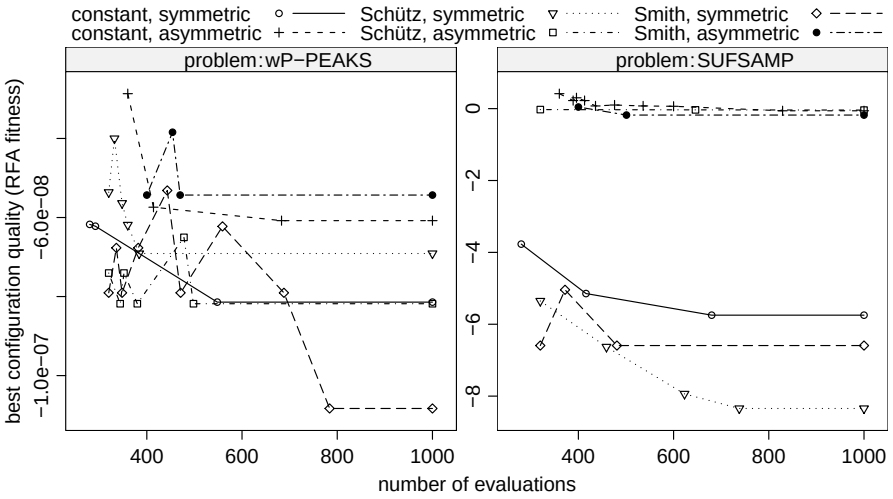

Fig. 1: *SPO performance on weighted P-PEAKS (left) and SUFSAMP (right) problems. Points denote the best configurations found up to the corresponding number of evaluations (runs), using all repeats available after* 1000 *runs to reduce noise. Except the last one, all points stand for changed configurations regarded as better by SPO. The ones tried inbetween are estimated as worse than the current best. The graphs start on the left with the best configuration of the initial LHS (f_{LHSb}).*

The tackled problems are the 6 named in §5. For each problem and mutation operator variant ($6 \times 6 = 36$), we perform a separate SPO run (1000 algorithm runs each).

Experimentation/Visualization:
The development of the SPO runs on either of the 6 problems is presented in figures 1, 2, and 4. Note that for the depicted configurations, we use information gained after a SPO is finished, which is more than that available at runtime, due to possible re-evaluations (additional runs) of good configurations. Restriction to runtime information leads to heavily increased noise that would render the figures almost useless. Numerical results for the weighted P-PEAKS, Subset Sum, and MMDP problems are given in table 4, the last column giving error probabilities for rejecting the hypothesis that f_{LHSb} and f_{SPO} are equal (p-values), based on all measures of this configuration obtained during the SPO run.

Observations:
As is evident from figures 1 to 4, there are two different behaviors of the six algorithm variants on on the considered test problems: Either a clear distinction into two groups of three can be recognized (SUFSAMP, COUNTSAT, and MMDP), or the curves are much more intermingled. A closer look at the constituents of these groups reveals that the separating feature is the symmetry type. For the COUNTSAT and MMDP problems, the better group is formed by all three asymmetric mutation variants, whereas all three symmetric variants are performing better on the SUFSAMP problem.

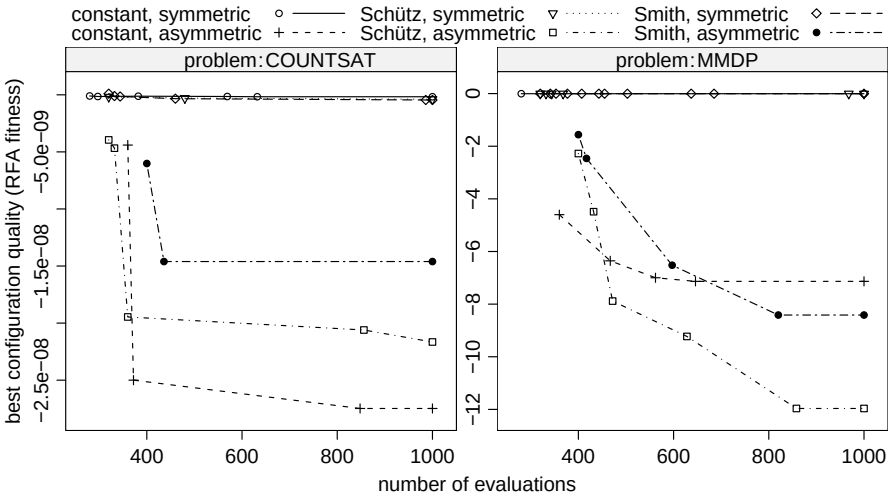

Fig. 2: *SPO performance on COUNTSAT (left) and MMDP (right) problems. Remarks of fig. 1 apply here as well.*

For the wP-PEAKS problem, we obtain two suprising obvservations:

1. At both ends of the performance spectrum, we find the self-adaptation variant of Smith, the symmetric being the best, the asymmetric the worst.
2. Ordering of the symmetric/asymmetric variants of one type is not the same for all, for the variants using the self-adaptation of Schütz, it is reversed.

As the SUFSAMP problem was constructed to favor enumerating the local neighborhood, it can be expected that SPO adjusts the selection pressure to high values. A look at the parameter distributions of the best variant, figure 3, reveals that SPO surprisingly circumvented doing so by more frequently using higher maximum age values; at the same time, the mean stagnation restart time was also increased.

The number partitioning problem results suprisingly indicate constant asymmetric mutation as the best and self-adaptation after Schütz as the worst variant, the latter being obviously very hard to improve. The tuning process on this as well as on the Subset Sum problem (figure 4), looks much more disturbed than e.g. on the MMDP, containing many huge jumps.

Discussion:
A look at table 4 reveals that for the chosen, discrete test problems, averaged performances and standard deviations are most often very similar. In contrast to the situation usually found for real-valued test problems treated with SPO, the noise level here makes further improvement very difficult. It obviously cannot be easily lowered by increasing the number of repeats as the best configurations typically already reached the imposed maximum of 64 runs. This hardship is probably due to the discrete value set of the objective functions which hinder achieving narrow result distributions. It does not render SPO useless, but its performance seems to be limited tighter than for the real-valued case.

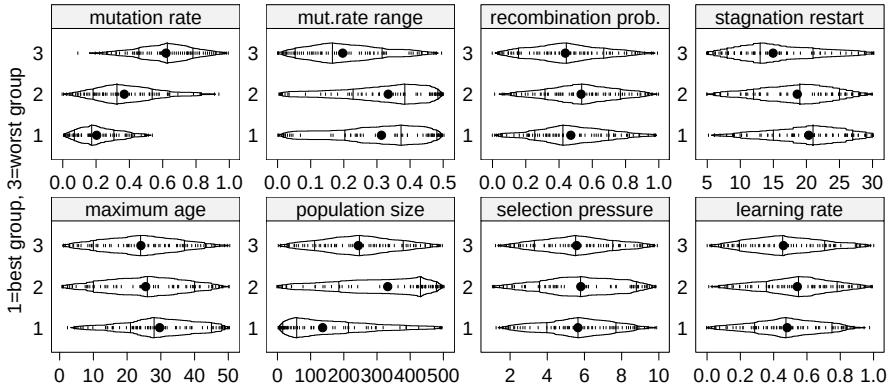

Fig. 3: *Parameter distributions of configurations chosen by SPO (including the initial LHS) on the SUFSAMP problem with symmetric self-adaptation after Schütz, separated into three equally sized groups according to measured fitness.*

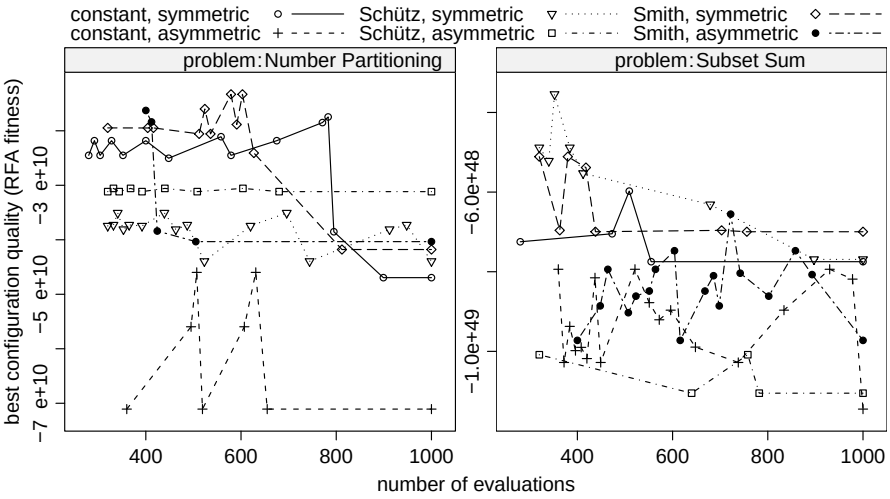

Fig. 4: *SPO performance on Number Partitioning (left) and Subset Sum (right).*

Another unexpected conclusion is that asymmetric mutation operators often perform very well on functions usually treated with symmetric constant mutation rates. COUNTSAT and MMDP are best approximated with asymmetric mutation operators, the former with the constant, the latter with the self-adaptive variant by Schütz. Table 3 lists the 2 best variants for each problem and the p-Values obtained from the comparison. Concerning aim 1 of §2, we cannot verify the original claim. There are problems for which self-adaptation is very useful, and there are some where it is not. Only in 2 of the six test cases, they have significant advantage.

Table 3: *Bootstrap Permutation hypothesis tests between performances of the best constant and the best self-adaptive variant for each problem. The tests use all best configuration runs available after SPO is finished, this is in most cases 64.*

Problem	Best Variant	Second Variant	p-Value	Significant?
wP-PEAKS	Smith, symmetric	constant, symmetric	0.125	No
SUFSAMP	Schütz, symmetric	constant, symmetric	2e-05	Yes
COUNTSAT	constant, asymmetric	Schütz, asymmetric	0.261	No
MMDP	Schütz, asymmetric	constant, asymmetric	0.008	Yes
Number Part.	constant, asymmetric	Schütz, symmetric	0.489	No
Subset Sum	constant, asymmetric	Schütz, asymmetric	0.439	No

Concerning the effort needed for tuning constant and self-adaptive mutation operators, we cannot account for a clear difference, thereby adhering to the original claim that tuning efforts are similar. However, there is neither statistical evidence in favor nor against this statement.

Nevertheless, when comparing the final best configurations to the average LHS results, it becomes clear that without any tuning, algorithms may easily be misconfigured, leading to performance values much worse than that of a simple EA-based hillclimber. Moreover, the best point of an initial LHS appears as a reasonable estimate for the quality achievable by tuning. Within this experiment, the LHS best fitness f_{LHSb} has always been superior to that of the (single-start) hillclimber.

Is it possible to derive any connection between the properties of the tackled problems and the success of different self-adaptation variants? We can detect similarities in the progression of the SPO runs, e.g. between COUNTSAT and MMDP, and Number Partitioning and Subset Sum, like the huge jumps found in case of the latter; these probably indicate result distributions much more complex than for the former. However, it appears difficult to foresee if self-adaptation enhanced EAs will perform better or worse. In case of the MMDP, recombination is likely to preserve existing good blocks that may be optimized individually so that mutation rate schedules, once learned, can be successfully reaplied. But without further tests, this is rather speculation.

7 Conclusions

We motivated and explained the SPO procedure and, as a test case, applied it to self-adaptive EA variants for binary coded problems. This study revealed why self-adaptation is rarely applied for these problem types: It is hard to predict whether it will increase performance, and which variant to choose. However, if properly parametrized, it can be significantly faster than well tuned standard mutation operators. Modeling only a rough problem-mechanism interaction would require a much more thouroughly conducted study than presented here.

Surprisingly, specialized and seldomly used operators like the (constant and self-adaptive) asymmetric mutation performed very well when parametrized accordingly. This probably leads the way to superior operators still to develop—with or without self-adaptation. Mutation rate learning seems to provide rather limited potential.

Table 4: *Best found configuration performances, mean value of LHS, best of LHS, and best after SPO is finished, on the weighted P-PEAKS, Subset Sum, and MMDP problems. Besides the RFA fitness measure used within SPO (to minimize, relative to hillclimber), we also give the MBF and AEB performance measures. AEB values are meant as multiples of* 10^3*. Note that wP-PEAKS and MMDP(40) are to be maximized with optimal values at* 1.0 *and* 40.0*, respectively, and Subset Sum is to be minimized to* 0.0 *.*

	Hillclimber, f_{hc}		LHS mean, f_{LHSa}				LHS best, f_{LHSb}				SPO best, f_{SPO}				
Adaptation	MBF	AEB	RFA	stddev	MBF	AEB	RFA	stddev	MBF	AEB	RFA	stddev	MBF	AEB	p-val
Problem: wP-PEAKS															
none, sym	0.945	<5	4.7e-07	9.5e-07	0.852	61.9	-6.2e-08	6.7e-08	0.984	52.1	-8.1e-08	8.2e-08	0.993	50.4	0.29
none, asym	0.945	<5	4.2e-07	8.8e-07	0.851	63.4	-2.9e-08	8.3e-08	0.964	48.8	-6.1e-08	2.5e-08	0.983	30.4	0.00
Schütz, sym	0.945	<5	5.0e-07	9.1e-07	0.844	60.5	-5.7e-08	3.1e-08	0.995	57.3	-6.9e-08	1.1e-07	0.980	46.2	0.46
Schütz, asym	0.945	<5	4.1e-07	1.1e-06	0.858	62.0	-7.4e-08	8.2e-08	0.984	48.3	-8.2-08	4.4e-08	0.984	24.2	0.35
Smith, sym	0.945	<5	4.9e-07	1.0e-06	0.846	58.5	-7.9e-08	1.2e-07	0.990	54.2	-1.1e-07	1.6e-07	0.987	46.1	0.12
Smith, asym	0.945	<5	4.0e-07	8.6e-07	0.858	63.1	-5.4e-08	3.4e-08	0.984	41.6	-5.4e-08	3.4e-08	0.984	41.6	0.50
Problem: Subset Sum															
none, sym	5.5e+26	36.0	3.4e+59	3.9e+60	4.7e+31	54.7	-7.3e+48	1.2e+49	1.6e+26	49.3	-7.8e+48	6.8e+48	8.1e+25	49.2	0.44
none, asym	5.5e+26	36.0	1.4e+59	6.4e+59	3.2e+31	52.5	-7.9e+48	1.0e+49	7.7e+25	55.6	-1.1e+49	1.6e+49	7.7e+25	38.0	0.18
Schütz, sym	5.5e+26	36.0	1.9e+59	1.1e+60	5.0e+31	53.9	-4.9e+48	6.6e+48	2.0e+26	51.7	-7.7e+48	5.6e+48	4.3e+25	51.2	0.04
Schütz, asym	5.5e+26	36.0	2.5e+59	1.9e+60	2.9e+31	49.4	-1.0e+49	1.2e+49	4.2e+25	51.0	-1.1e+49	1.4e+49	7.1e+25	48.1	0.42
Smith, sym	5.5e+26	36.0	2.6e+59	1.7e+60	5.0e+31	55.7	-5.1e+48	5.2e+48	1.3e+26	58.3	-7.0e+48	9.0e+48	8.8e+25	52.6	0.21
Smith, asym	5.5e+26	36.0	3.2e+59	4.5e+60	2.8e+31	51.1	-9.7e+48	1.3e+49	7.3e+25	52.5	-9.7e+48	1.3e+49	7.3e+25	52.5	0.50
Problem: MMDP															
none, sym	23.80	58.1	-1.1e-03	1.4e-03	29.3	62.0	-4.3e-03	2.3e-03	37.0	54.3	-4.3e-03	4.3e-04	37.8	45.7	0.48
none, asym	23.80	58.1	-0.225	0.783	39.9	7.4	-4.60	4.83	40	<5	-7.13	11.7	40	<5	0.23
Schütz, sym	23.80	58.1	-1.3e-03	1.7e-03	30.0	63.0	-4.3e-03	2.8e-03	36.0	48.3	-4.6e-03	2.3e-03	37.6	54.3	0.37
Schütz, asym	23.80	58.1	-0.155	0.274	39.8	8.1	-2.28	0.894	40	<5	-12.0	10.9	40	<5	0.00
Smith, sym	23.80	58.1	-1.4e-03	1.8e-03	30.0	62.1	-4.1e-03	2.4e-03	37.3	59.6	-4.6e-03	2.1e-03	36.8	46.1	0.18
Smith, asym	23.80	58.1	-0.143	0.234	39.8	7.4	-1.56	0.774	40	<5	-8.41	6.59	40	<5	0.00

The invented performance measures, RFA and AEB, were found capable of leading the meta-search into the right direction. Unfortunately, the absence of well-defined measures for the tuning process itself currently prevents effectively using SPO for answering questions concerning the tunability of an algorithm-problem combination. This problem we want to tackle in future work.

Recapitulating, SPO works on binary problems, and proves to be a valuable tool for experimental analysis. However, there is room for improvement, first and foremost by taking measures to reduce the variances within the results of a single configuration.

Acknowledgment

The research leading to this paper has been supported by the DFG (Deutsche Forschungsgemeinschaft) as project no. 252441, "Mehrkriterielle Struktur- und Parameteroptimierung verfahrenstechnischer Prozesse mit evolutionären Algorithmen am Beispiel gewinnorientierter unscharfer destillativer Trennprozesse". T. Bartz–Beielstein's research was supported by the DFG as part of the collaborative research center "Computational Intelligence" (531).

The autors want to thank Thomas Jansen for providing his code of the SUF-SAMP problem, and Silja Meyer-Nieberg and Nikolaus Hansen for sharing their opinions with us.

References

Auger, A. & Hansen, N. (2005). Performance Evaluation of an Advanced Local Search Evolutionary Algorithm. In B. McKay & others (Eds.), *Proc. 2005 Congress on Evolutionary Computation (CEC'05)* Piscataway NJ: IEEE Press.

Bäck, T. (1992). Self-adaptation in genetic algorithms. In F. Varela & P. Bourgine (Eds.), *Toward a Practice of Autonomous Systems: proceedings of the first European conference on Artificial Life* (pp. 263–271).: MIT Press.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* New York NY: Oxford University Press.

Bäck, T. & Schütz, M. (1995). Evolution strategies for mixed-integer optimization of optical multilayer systems. In J. R. McDonnell, R. G. Reynolds, & D. B. Fogel (Eds.), *Evolutionary Programming IV: Proceedings of the Fourth Annual Conference on Evolutionary Programming* (pp. 33–51).: MIT Press, Cambridge, MA.

Barr, R. & Hickman, B. (1993). Reporting computational experiments with parallel algorithms: Issues, measures, and experts' opinions. *ORSA Journal on Computing*, 5(1), 2–18.

Bartz-Beielstein, T. (2005). Evolution strategies and threshold selection. In M. J. Blesa Aguilera, C. Blum, A. Roli, & M. Sampels (Eds.), *Proceedings Second International Workshop Hybrid Metaheuristics (HM'05)*, volume 3636 of *Lecture Notes in Computer Science* (pp. 104–115). Berlin, Heidelberg, New York: Springer.

Bartz-Beielstein, T. (2006). *Experimental Research in Evolutionary Computation—The New Experimentalism.* Berlin, Heidelberg, New York: Springer.

Bartz-Beielstein, T., Blum, D., & Branke, J. (2005a). Particle swarm optimization and sequential sampling in noisy environments. In R. Hartl & K. Doerner (Eds.), *Proceedings 6th Metaheuristics International Conference (MIC2005)* (pp. 89–94). Vienna, Austria.

Bartz-Beielstein, T., de Vegt, M., Parsopoulos, K. E., & Vrahatis, M. N. (2004a). *Designing Particle Swarm Optimization with Regression Trees.* Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence CI–173/04, Universität Dortmund, Germany.

Bartz-Beielstein, T., Lasarczyk, C., & Preuß, M. (2005b). Sequential parameter optimization. In B. McKay & others (Eds.), *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1 (pp. 773–780). Piscataway NJ: IEEE Press.

Bartz-Beielstein, T. & Naujoks, B. (2004). *Tuning Multicriteria Evolutionary Algorithms for Airfoil Design Optimization.* Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence CI–159/04, Universität Dortmund, Germany.

Bartz-Beielstein, T., Parsopoulos, K. E., & Vrahatis, M. N. (2004b). Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis & Computational Mathematics (ANACM)*, 1(2), 413–433.

Bartz-Beielstein, T., Preuß, M., & Markon, S. (2005c). Validation and optimization of an elevator simulation model with modern search heuristics. In T. Ibaraki, K. Nonobe, & M. Yagiura (Eds.), *Metaheuristics: Progress as Real Problem Solvers*, Operations Research/Computer Science Interfaces (pp. 109–128). Berlin, Heidelberg, New York: Springer.

Berretta, R., Cotta, C., & Moscato, P. (2004). Enhancing the performance of memetic algorithms by using a matching-based recombination algorithm. In *Metaheuristics: computer decision-making* (pp. 65–90). Norwell, MA: Kluwer Academic Publishers.

Berretta, R. & Moscato, P. (1999). The number partitioning problem: an open challenge for evolutionary computation? In *New ideas in optimization* (pp. 261–278). Maidenhead, UK: McGraw-Hill Ltd.

Beyer, H.-G. & Schwefel, H.-P. (2002). Evolution strategies—A comprehensive introduction. *Natural Computing*, 1, 3–52.

de Vegt, M. (2005). *Einfluss verschiedener Parametrisierungen auf die Dynamik des Partikel-Schwarm-Verfahrens: Eine empirische Analyse.* Interner Bericht der Systems Analysis Research Group SYS–3/05, Universität Dortmund, Fachbereich Informatik, Germany.

Demetrescu, C. & Italiano, G. F. (2000). What do we learn from experimental algorithmics? In *MFCS '00: Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science* (pp. 36–51). Berlin, Heidelberg, New York: Springer.

Draper, N. R. & Smith, H. (1998). *Applied Regression Analysis.* New York NY: Wiley, 3rd edition.

Droste, S., Jansen, T., & Wegener, I. (2000). A natural and simple function which is hard for all evolutionary algorithms. In *IEEE International Conference on Industrial Electronics, Control, and Instrumentation (IECON 2000)* (pp. 2704–2709). Piscataway, NJ: IEEE Press.

Eiben, A. & Jelasity, M. (2002). A critical note on experimental research methodology in EC. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002)* (pp. 582–587). Piscataway NJ: IEEE.

Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2), 124–141.

Eiben, A. E. & Smith, J. E. (2003). *Introduction to Evolutionary Computing*. Berlin, Heidelberg, New York: Springer.

Feldt, R. & Nordin, P. (2000). Using factorial experiments to evaluate the effect of genetic programming parameters. In R. Poli & others (Eds.), *Genetic Programming, Proceedings of EuroGP'2000*, volume 1802 of *Lecture Notes in Computer Science* (pp. 271–282). Berlin, Heidelberg, New York: Springer.

Fogel, D. B. (1992). *Evolving Artificial Intelligence*. PhD thesis, University of California, San Diego.

François, O. & Lavergne, C. (2001). Design of evolutionary algorithms—a statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2), 129–148.

Giacobini, M., Preuß, M., & Tomassini, M. (2006). Effects of scale-free and small-world topologies on binary coded self-adaptive CEA. In *6th European Conf. Evolutionary Computation in Combinatorial Optimization, Proc. (EvoCOP'06)*, Lecture Notes in Computer Science (pp. 86–98). Berlin: Springer.

Goldberg, D. E., Deb, K., & Horn, J. (1992). Massively multimodality, deception and genetic algorithms. In R. Männer & B. Manderick (Eds.), *Parallel Prob. Solving from Nature II* (pp. 37–46).: North-Holland.

Greenwood, G. W. (2003). Adapting mutations in genetic algorithms using gene flow principles. In R. Sarker & others (Eds.), *Proc. 2003 Congress on Evolutionary Computation (CEC'03), Canberra* (pp. 1392–1397). Piscataway NJ: IEEE Press.

Hansen, N. & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2), 159–195.

Hansen, N., Ostermeier, A., & Gawelczyk, A. (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L. J. Eshelman (Ed.), *Proc. 6th Int'l Conf. on Genetic Algorithms* (pp. 57–64). San Francisco, CA: Morgan Kaufmann Publishers, Inc.

Hooker, J. (1996). Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1), 33–42.

Igel, C. & Kreutz, M. (2003). Operator adaptation in evolutionary computation and its application to structure optimization of neural networks. *Neurocomputing*, 55(1-2), 347–361.

Isaaks, E. H. & Srivastava, R. M. (1989). *An Introduction to Applied Geostatistics*. Oxford, U.K.: Oxford University Press.

Jansen, T., De Jong, K. A., & Wegener, I. (2005). On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13(4), 413–440.

Jansen, T. & Sudholt, D. (2005). Design and analysis of an asymmetric mutation operator. In B. McKay & others (Eds.), *Proc. 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1 (pp. 190–197). Piscataway NJ: IEEE Press.

Jelasity, M. (1997). A wave analysis of the subset sum problem. In T. Bäck (Ed.), *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)* (pp. 89–96). San Francisco, CA: Morgan Kaufmann.

Jelasity, M., Preuß, M., & Eiben, A. E. (2002). Operator learning for a problem class in a distributed peer-to-peer environment. In J. J. M. Guervós, P. Adamidis, H.-G. Beyer, J. L. Fernández-Villacañas, & H.-P. Schwefel (Eds.), *Parallel Problem Solving from Nature – PPSN VII, Proc. Seventh Int'l Conf., Granada* (pp. 172–183). Berlin: Springer.

Johnson, D. S. (2002). A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser, D. S. Johnson, & C. C. McGeoch (Eds.), *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges* (pp. 215–250). Providence: American Mathematical Society.

Jones, D., Schonlau, M., & Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13, 455–492.

Jong, K. A. D., Potter, M. A., & Spears, W. M. (1997). Using problem generators to explore the effects of epistasis. In T. Bäck (Ed.), *Proceedings of the Seventh ICGA* (pp. 338–345).: Morgan Kaufmann.

Julstrom, B. A. (1997). Adaptive operator probabilities in a genetic algorithm that applies three operators. In *SAC '97: Proceedings of the 1997 ACM symposium on Applied computing* (pp. 233–238). New York, NY: ACM Press.

Kleijnen, J. P. C. (1987). *Statistical Tools for Simulation Practitioners.* New York NY: Marcel Dekker.

Kleijnen, J. P. C. (1997). Experimental design for sensitivity analysis, optimization, and validation of simulation models. In J. Banks (Ed.), *Handbook of Simulation.* New York NY: Wiley.

Kursawe, F. (1999). *Grundlegende empirische Untersuchungen der Parameter von Evolutionsstrategien – Metastrategien.* Dissertation, Fachbereich Informatik, Universität Dortmund, Germany.

Law, A. & Kelton, W. (2000). *Simulation Modeling and Analysis.* New York NY: McGraw-Hill, 3rd edition.

Lophaven, S., Nielsen, H., & Søndergaard, J. (2002a). *Aspects of the Matlab Toolbox DACE.* Technical Report IMM-REP-2002-13, Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark.

Lophaven, S., Nielsen, H., & Søndergaard, J. (2002b). *DACE—A Matlab Kriging Toolbox.* Technical Report IMM-REP-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, Copenhagen, Denmark.

Markon, S., Kita, H., Kise, H., & Bartz-Beielstein, T., Eds. (2006). *Modern Supervisory and Optimal Control with Applications in the Control of Passenger Traffic Systems in Buildings.* Berlin, Heidelberg, New York: Springer.

McGeoch, C. C. (1986). *Experimental Analysis of Algorithms.* PhD thesis, Carnegie Mellon University, Pittsburgh PA.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2), 239–245.

Mehnen, J., Michelitsch, T., Bartz-Beielstein, T., & Henkenjohann, N. (2004). Systematic analyses of multi-objective evolutionary algorithms applied to real-world problems using statistical design of experiments. In R. Teti (Ed.), *Proceedings Fourth International Seminar Intelligent Computation in Manufacturing Engineering (CIRP ICME'04)*, volume 4 (pp. 171–178). Naples, Italy.

Mehnen, J., Michelitsch, T., Bartz-Beielstein, T., & Lasarczyk, C. W. G. (2005). Multiobjective evolutionary design of mold temperature control using DACE

for parameter optimization. In H. Pfützner & E. Leiss (Eds.), *Proceedings Twelfth International Symposium Interdisciplinary Electromagnetics, Mechanics, and Biomedical Problems (ISEM 2005)*, volume L11-1 (pp. 464–465). Vienna, Austria: Vienna Magnetics Group Reports.

Meyer-Nieberg, S. & Beyer, H.-G. (2006). Self-adaptation in evolutionary algorithms. In F. Lobo, C. Lima, & Z. Michalewicz (Eds.), *Parameter Setting in Evolutionary Algorithms*, Studies in Computational Intelligence. Berlin Heidelberg New York: Springer.

Montgomery, D. C. (2001). *Design and Analysis of Experiments*. New York NY: Wiley, 5th edition.

Moret, B. M. & Shapiro, H. D. (2001). Algorithms and experiments: The new (and old) methodology. *Journal of Universal Computer Science*, 7(5), 434–446.

Moret, B. M. E. (2002). Towards a discipline of experimental algorithmics. In M. Goldwasser, D. Johnson, & C. McGeoch (Eds.), *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, *DIMACS Monographs 59* (pp. 197–213). Providence RI: American Mathematical Society.

Myers, R. & Hancock, E. (2001). Empirical modelling of genetic algorithms. *Evolutionary Computation*, 9(4), 461–493.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (2000). Linkage problem, distribution estimation, and bayesian networks. *Evolutionary Computation*, 8(3), 311–340.

Pukelsheim, F. (1993). *Optimal Design of Experiments*. New York NY: Wiley.

Ramos, I. C. O., Goldbarg, M. C., Goldbarg, E. G., & Neto, A. D. D. (2005). Logistic Regression for Parameter Tuning on an Evolutionary Algorithm. In B. McKay & others (Eds.), *Proc. 2005 Congress on Evolutionary Computation (CEC'05)*, volume 2 (pp. 1061–1068). Piscataway NJ: IEEE Press.

Rardin, R. L. & Uzsoy, R. (2001). Experimental evaluation of heuristic optimization algorithms: A tutorial. *J. Heuristics*, 7(3), 261–304.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Stuttgart: Frommann-Holzboog Verlag.

Rechenberg, I. (1978). Evolutionsstrategien. In B. Schneider & U. Ranft (Eds.), *Simulationsmethoden in der Medizin und Biologie* (pp. 83–114). Berlin: Springer-Verlag.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical Science*, 4(4), 409–435.

Santner, T. J., Williams, B. J., & Notz, W. I. (2003). *The Design and Analysis of Computer Experiments*. Berlin, Heidelberg, New York: Springer.

Schaffer, J. D., Caruana, R. A., Eshelman, L., & Das, R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 51–60). San Mateo CA: Morgan Kaufman.

Schonlau, M. (1997). *Computer Experiments and Global Optimization*. PhD thesis, University of Waterloo, Ontario, Canada.

Schonlau, M., Welch, W. J., & Jones, R. R. (1998). Global versus local search in constrained optimization of computer models. In *New Development and Applications in Experimental Design*, number 34 in IMS Lecture Notes (pp. 11–25). Institute of Mathematical Statistics, Beachwood, OH.

Schütz, M. (1996). *Eine Evolutionsstrategie für gemischt-ganzzahlige Optimierprobleme mit variabler Dimension.* Interner Bericht der Systems Analysis Research Group SYS–1/96, Universität Dortmund, Fachbereich Informatik.

Schwefel, H.-P. (1974). *Adaptive Mechanismen in der biologischen Evolution und ihr Einfluß auf die Evolutionsgeschwindigkeit.* Technical report, Technical University of Berlin. Abschlußbericht zum DFG-Vorhaben Re 215/2.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models.* Chichester: Wiley.

Schwefel, H.-P., Wegener, I., & Weinert, K., Eds. (2003). *Advances in Computational Intelligence—Theory and Practice.* Berlin, Heidelberg, New York: Springer.

Smith, J. & Fogarty, T. C. (1996). Self-adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC '96)* (pp. 318–323).: IEEE Press, NY.

Smith, J. E. (2001). Modelling gas with self adaptive mutation rates. In L. Spector & et al. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (pp. 599–606). San Francisco, California, USA: Morgan Kaufmann.

Stoean, C., Preuss, M., Gorunescu, R., & Dumitrescu, D. (2005). Elitist Generational Genetic Chromodynamics - a New Radii-Based Evolutionary Algorithm for Multimodal Optimization. In B. McKay & others (Eds.), *Proc. 2005 Congress on Evolutionary Computation (CEC'05)*, volume 2 (pp. 1839 – 1846). Piscataway NJ: IEEE Press.

Stone, C. & Smith, J. E. (2002). Strategy parameter variety in self-adaptation of mutation rates. In W. B. Langdon & et al. (Eds.), *GECCO* (pp. 586–593).: Morgan Kaufmann.

Thierens, D. (2005). An adaptive pursuit strategy for allocating operator probabilities. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation* (pp. 1539–1546). New York, NY: ACM Press.

Tosic, M. (2006). Evolutionäre Kreuzungsminimierung. Diploma thesis, University of Dortmund, Germany.

Weinert, K., Mehnen, J., Michelitsch, T., Schmitt, K., & Bartz-Beielstein, T. (2004). A multiobjective approach to optimize temperature control systems of moulding tools. *Production Engineering Research and Development, Annals of the German Academic Society for Production Engineering*, XI(1), 77–80.

Whitley, D., Rana, S. B., Dzubera, J., & Mathias, K. E. (1996). Evaluating evolutionary algorithms. *Artificial Intelligence*, 85(1-2), 245–276.

Wolpert, D. & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.