Experimental Analysis of Optimization Algorithms: Tuning and Beyond

Thomas Bartz-Beielstein and Mike Preuß

Abstract This chapter comprises the essence of several years of tutorials the authors gave on experimental research in evolutionary computation. We highlight the renaissance of experimental techniques also in other fields to especially focus on the specific conditions of experimental research in computer science, or more concrete, metaheuristic optimization. The experimental setup is discussed together with the pitfalls awaiting the unexperienced (and sometimes even the experienced). We present a severity criterion as a meta-statistical concept for evaluating statistical inferences, which can be used to avoid fallacies, i.e., misconceptions resulting from incorrect reasoning in argumentation caused by floor or ceiling effects. The sequential parameter optimization is discussed as a meta-statistical framework which integrates concepts such as severity. Parameter tuning is considered as a relatively new tool in method design and analysis, and it leads to the question of adaptability of optimization algorithms. Another branch of experimentation aims for attaining more concrete problem knowledge, we may term it 'exploratory landscape analysis', containing sample and visualization techniques that are often applied but not seen as being a methodological contribution. However, this chapter is not only a renarration of well known facts. We also try a look into the future to estimate what the hot topics of methodological research will be in the next years and what changes we may expect for the whole community.

Thomas Bartz-Beielstein
Institute of Computer Science, Faculty of Computer Science and Engineering Science, Cologne University of Applied Sciences, Germany, e-mail: thomas.bartz-beielstein@fh-koeln.de

Mike Preuss
Algorithm Engineering, Department of Computer Science, TU Dortmund, Germany, e-mail: mike.preuss@tu-dortmund.de

1 Introduction

As in many natural sciences, research on metaheuristics and especially *evolutionary computation* (EC) mainly rests on two pillars: theory and practice. Undoubtedly, theory in EC has made a good step forward during the last decade. However, the larger part of published work in this area is still dealing almost exclusively with the application of EC and related methods to real-world and/or benchmark problems. Qualms regarding the meaningfulness of theoretical approaches are rarely expressed, but doubts concerning the reliability of experimental results are often raised, especially by practitioners. This may lead to the question: "Can we get rid of experimentation in EC as a whole and and resolve to theory only?"

Our experience is that this will not happen, because there are simply too many different real-world applications of EC techniques. Moreover, theory and practice have different rhythms, one may design and implement a useful algorithm modification in minutes or hours, but adapting the existing theory to it may take days or weeks. It may be worth noting that in other related sciences and in philosophy of science, experimentation currently experiences a renaissance [54].

If we presume that experimentation is necessary, we need to ponder how to strengthen the experimental methodology in order to make experimental results more reliable and thus also more useful for theoretical approaches. It may help to make clear what experimental works in EC are actually about and if they can be split into categories. It seems that during the last decades, two motivations for experimental works have been predominant:

- Solving a real-world problem or at least showing that it could be solved by some EC based method
- Demonstrating the ability of a (preferably new and self-defined) algorithm

These two form the extremes, and mixtures with various weights are frequently encountered. They resemble a problem-centric and an algorithm-centric view, respectively. The former is strongly leaning towards engineering and often focusing on representations, simulating, modeling, long runtimes, and technical issues, whereas the latter is much nearer to algorithmics and mathematics. One deals with constructed problems that can be computed fast and of which the most important properties are well known.

Setting up an experiment can be far from trivial as there are numerous mistakes that may render an experiment useless. Rardin and Uszoy [71] state the following: "No one who has ever tried it thinks conducting empirical research on heuristics is easy", and we would like to add that this stems from the complexity of the systems dealt with. Many influences which are simply ignored (or removed 'by construction') in theoretical investigations cannot be removed but must rather be controlled in experimental studies, thereby at least trying to avoid the most obvious mistakes.

Performing experiments in computer science can address the following (related) tasks:

T-1 Find the best parameters or algorithms given k sets of random numbers representing the outcome of some experiments
T-2 Find the best assignment for a set of real variables representing the parameters of the algorithm (within a given range) for a problem class
T-3 Given m possible ways to modify algorithm a (e.g., by using extra operators) find the best combination for a problem class

Regarding task T-1, we will restrict our analysis to problems with $k = 2$. We are using SPOT (introduced in Sec. 3.2) to find the best assignment. SPOT can also be used to determine the best combination of algorithm operators for a given problem class.

Although conceptually different, task T-3 can be tackled in the framework of task T-2. In a complex experiment the experimenter might have to consider, hierarchically, all three tasks (for example he might want to retune the parameters for every different combination of operators in T-3). The tuning procedure can be performed hierarchically, e.g., for every different combination of operators. However, we recommend an alternative approach which includes the operators into the algorithm design. Settings for the operators can be included as factors in the algorithm design and treated in a similar way as numerical parameters, see Chapter 14 in [6] for an example.

We report on these methodological foundations in Sect. 2, also reflecting on approaches in other fields, especially in algorithmics. This section also describes how reports from experimental studies can be structured.

Section 3 introduces the framework of *active experimentation.* It describes the sequential parameter optimization toolbox. Since the comparison of results plays a central role in experimentation, we discuss key elements from a meta-statistical perspective.

The algorithm-centric research has made good progress during the last years, two notable developments are different tuning techniques (e.g., *F-race* [22], *sequential parameter optimization* (SPO) [8, 1] and the *relevance and calibration* method (REVAC) [61]) and new benchmark competitions/libraries as BBOB'09 [35] and the CEC'05 competition [75]. We highlight SPO as one interesting tuning approach and its use for the experimental analysis of a *simulated annealing* (SANN) heuristic which is presented in Sec. 4.

Hypothesis testing is discussed in Sec. 5. We present an introduction and discuss problems related to hypothesis testing as well.

Some researchers claim that scientific progress is based on accepting high-level theories, whereas others view progress “based on the growth of more localized experimental knowledge”[54]. Actually, there is an interesting debate about the importance of experimentation in the philosophy of science, e.g., [24] can be recommended as a good starting point. We will not detail this discussion in this chapter, but will transfer some important results from this

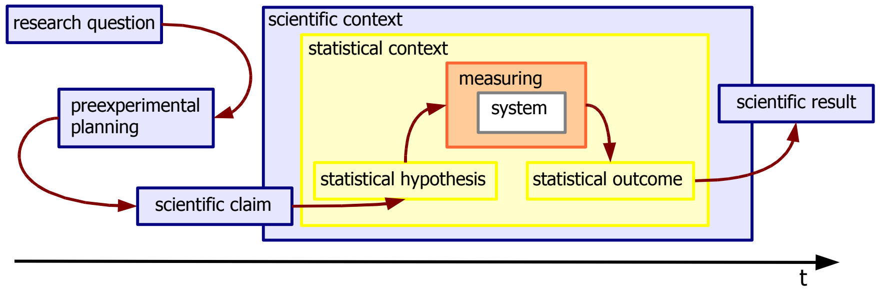


Fig. 1 Steps and contexts of performing an experiment from research question to scientific result

on-going debate in the following. The focus of our work lies on severity as a meta-statistical concept for evaluating statistical inferences, which can be used to avoid fallacies, i.e., misconceptions resulting from incorrect reasoning in argumentation caused by floor or ceiling effects. Severity, as an extension of the power used in hypothesis testing, is introduced in Sec. 6.

Based on the considerations from the previous sections, meta-statistical principles can be applied. Meta-statistical rules, as discussed in Sec. 7, are necessary, because statistical results can be misleading and need some interpretation. Here comes the concept of severity into play, which is one element of the sequential parameter optimization.

For the problem-centric approach, it is often most important to collect problem knowledge during design and test of an optimization method. This resembles some kind of exploratory analysis and is dealt with in Sect. 8.

Finally, the chapter concludes with a summary and a view onto envisaged future developments in Sect. 9.

2 Towards an Experimental Methodology

The following sections provide basic elements and fundamental considerations, which are necessary for active experimentation in computer science. *Active experimentation*, which implements a framework of the approach presented in this section, is visualized in Fig. 1.

Section 2.1 discusses the role of experiments in computer science. Generally, experiments should be based on well-founded research questions. We present research questions related to demonstration, robustness, comparison, understanding, and novelty detection in Sect. 2.2. The key problem which occurs in nearly every experimental study is the selection of an appropriate measure. The most prominent measures are introduced in Sec. 2.3. After an adequate measure is selected, the pre-experimental planning phase can begin. During this phase, which is described in Sec. 2.4, the experimental setup is calibrated. Problems, which are related to parameter settings, are briefly discussed in Sec. 2.5. Then, experiments can be performed, see Sec. 2.6. These experiments can generate lots of data, which are use for the experimental analysis. We consider key features of comparisons in Sec. 2.7. Findings from these analyses should be presented, e.g., as articles. We propose a structured report scheme in Sec. 2.8. We can conclude from our experience that meaningful results require several experiments as discussed in Sec. 2.9. Finally, we consider the determination of scientifically meaningful results in Sec. 2.10.

2.1 Performing Experiments in Computer Science

In theoretical computer science, one is usually interested in pessimistic generalizations, in knowing what an algorithm does in the worst possible case. Experimental results are considered with a certain amount of skepticism. This may have two reasons:

- Many experimental works of the past are not very well crafted. Different from other sciences with very expensive (in time and financial effort) experiments, computer science is in the luxurious position to allow for nearly unlimited sampling. Thus, one shall not stop at the first results but use the obtained knowledge to set up a refined experiment. Every possible setup is like a simple point in a high-dimensional space, defined by the many different factors that could influence the outcome. Sampling just at one point will not allow for any conclusions about, e.g., the overall quality of certain algorithms.
- Experimental investigations rarely care for worst cases. If the treated problem has a real-world background, the worst case view is usually not applicable: one simply does not know how it could look like, and as no analytic formulation is available, there is no alternative way of approaching the worst case. One therefore often follows an average case approach, where the average consists of some reasonably likely cases or representatives thereof. Thus, experiments can only lead to probabilistic reasoning because it is not possible to give a conclusion that holds for every member of a problem class. This approach generalizes neutrally over the samples made and even if the experiment is bug-free and cleverly set up, exact knowledge is only attained for the actually measured points. Performance guarantees cannot be given for any deviating setting.

This said, we do not recommend to cease experimentation altogether. In many situations, it is the only way to advance scientific progress. In others, it can be a valuable addition to theory, as emphasized by the *Algorithm Engineering* approach [26]. However, it is necessary to be aware of the problems one may run into and to follow a suitable experimental methodology instead of doing ad-hoc tests.

Several authors from different fields have cautioned against experimental mistakes and provided guidelines for scientifically meaningful experimentation on algorithms, we name only a small subset here. Moret gives a methodology overview from the viewpoint of algorithmics, also reporting about testing heuristics [59]. Johnson [45] provides a comprehensive list of pitfalls and solutions for experiments in algorithmics, mostly dealing with deterministic algorithms. However, a large part of the list also applies to metaheuristics/evolutionary algorithms. Nevertheless, there are problems stemming from the non-deterministic nature of these algorithms. These are especially treated, e.g., by Hooker [40] and Eiben [29]. In the following, we describe how to start an experimental investigation on metaheuristic optimization methods and how to avoid the most commonly made mistakes.

2.2 Research Questions

Around two decades ago, Cohen [27] hinted at the fact that in artificial intelligence, experimental studies were often not well tasked. In evolutionary computation or metaheuristics fields, the situation at that time was certainly not better. Surprisingly, nowadays many experimental works still come without a clear statement about the overall *research question* that is going to be answered. Instead, the implicitly assumed task often is to show that any new algorithm A is *better* than a standard method A^* or many of them. Sampling comparison data at a small number of points (defined by algorithms, optimization problems, parameters, termination criteria etc.) does not necessarily allow for general conclusions about the compared algorithms. Besides, the tackled research question should not be that general. As an example, it may make more sense to ask under which conditions (problems, runtimes, etc.) A is better than A^* and why. It goes without saying that the research question must be stated in the paper, so that the reader gets a chance to value the experimental findings.

In our view, the following questions or aspects of questions are fundamental for experiments in computer science and may serve as a guideline for setting up new experiments. The experimenter should clearly state if experiments are performed to

1. demonstrate the performance of one algorithm,
2. verify the robustness of one algorithm on several problem instances,
3. compare two (or several) known algorithms,
4. explain and understand an observed behavior, or
5. detect something new.

Each of these five research goals, which can also be characterized as demonstration, robustness, comparison, understanding, and novelty detection, requires a different experimental setup.

We discourage mixing too many different aspects of an experimental investigation into one experiment. Rather, one shall consider if it makes sense to split the investigation into multiple experiments, each one reported separately. This simplifies understanding the outcome and also enhances *reproducibility*, which should be a primary concern when presenting exciting new facts obtained by experiment. If we want to employ statistical techniques as hypothesis tests to bolster up our confidence in the findings, it is necessary to switch the context from a domain-specific scientific one to a statistical one and back. We can first formulate one or several scientific claims. As an example, we consider the claim: "Algorithm A is faster than algorithm A^* under the defined conditions (test problems, performance measure, parameter settings, etc.)." These have then to be formulated as statistical hypotheses, which can be tested by experimentation.

2.3 What to Measure?

Once the direction of the experimental investigation is set, one has to decide how to measure. This may not be a trivial issue. McGeoch [55] demonstrates that even for deterministic algorithms, the influence of the measure should not be underestimated and that setting it up properly can be a decisive factor for obtaining interesting results. When investigating nondeterministic optimization algorithms on not too simple problems, there are two principal possibilities. We can employ a quality task and measure how long the algorithm needs to get there, or we can set up a budget (usually regarded as equivalent to runtime in black-box optimization) task and measure the performance obtained under scarce resources. As discussed in the BBOB'09 setup [35], fixing a quality task (there also called *horizontal* measuring) often leads to a better understanding of algorithm performance than fixing a budget task (*vertical* measuring). The competition was thus run under the *expected running time* (ERT) measure:

$$ERT(f_{target}) = \frac{\#FEs(f_{best}(FE) \geq f_{target})}{\#succ} \qquad (1)$$

The number of 'unsuccessful evaluations' (where the observed objective function value is worse than a given target, $f_{best}(FE) \geq f_{target}$, restarts are assumed) per repeat is summed up and divided by the number of successes ($\#succ$) of attaining the target function value f_{target}. The ERT is certainly good at capturing the performance of algorithms under relaxed resource conditions. However, real-world problems often do not allow such generous conditions so that only a few hundred or thousand function evaluations can be invested. It is a philosophical question if one can justify applying the term 'optimization' in this context, but apart from that it is obvious that an algorithm with a good final best function value does not necessarily provide a good approximation of this value fast. Thus it also makes sense to set a budget task in the predicted range of allowed function evaluations under application conditions and ask which algorithm provides the best solution and how reliable it is.

Next to the ERT, some often used measures are the *mean best fitness* (MBF) and the *success rate* (SR). However, these come with some difficulties. Averaging is very sensitive regarding outliers, so it may be more suitable to work with the median instead of the mean. Success rates were frequently used in the past, but this measure removes the whole runtime information. A fast method always reaching 100% becomes indistinguishable from a slow one also reaching this success rate. For setting up proper conditions for measuring, one may rely on runtime distributions as proposed by Hoos and Stützle [41]. Chapter 7 in [1] presents 18 different performance measures and mentions relevant publications.

Sometimes however, the available base measures do not match the intention of the experiment well. For example, if one has a certain, possibly unusual tradeoff between quality and runtime in mind. In these cases, it may be necessary to define a new measure as e.g. suggested by [71] (see [67] for an example). However, this path should be walked with caution: it does not make sense to stick to a measure that does not express what one actually wants to investigate. But too many measures render results incomparable.

2.4 Pre-experimental Planning

If research question and measure are chosen, and the implementation issues have been resolved, it is time for the first tests. We name this phase *pre-experimental planning* and its main purpose is to check if the envisioned experimental setup is meaningful. This may apply to the selection of f_{target} values for measuring, or the maximum allowed number of function evaluations, or the set of benchmark problems one is going to investigate.

During the pre-experimental planning phase, several practical problems have to be considered, e.g., how many comparisons should be performed? How many repeat runs of each algorithm should be done? Should a one-stage or multi-stage procedure be used? Classical textbooks on statistics provide useful answers to these questions, which are related to *pre-data* information, i.e., before the experimental data is available. We highly recommend Bechhofer et al.'s comprehensive work "Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons".[20]

Additionally, one shall try to make sure that other possible problems that could influence the outcome of an experiment are found and remedied. As Knight [47] argues, one should apply common sense concerning setup and results. Are the first results plausible? Or do they hint to a possible code problem? What makes a meaningful difference in my test cases? We would explicitly suggest to use as much visualization as possible at this stage. A problem revealed only after the experiment is finished is much more annoying than one found early, especially if time constraints do not allow for redoing the whole experiment.

At the end of the pre-experimental phase, one should be able to set up an experiment in a way that it leads to results finally tackling the given research question. We may assume that not all possible problems with the setup can be identified during this first phase, but it serves as a filter preventing the most obvious mistakes, some of which we highlight in the following.

2.5 Fair Parameter Settings

There are different opinions on how much effort shall be put into obtaining good parameters for the algorithms which are to be compared. It certainly makes sense to compare algorithms under default parameters. This resembles an application setting where parameter adaptation to a problem is not possible, e.g., due to time restrictions. On the other hand, it may also be important to know how the algorithm would perform under good parameter settings. In this case, tuning algorithms can be applied before running the comparison. In either case, the comparison shall be fair, meaning that the same amount of tuning should go into setting up each algorithm. It is clear that a tuned algorithm will perform better than an algorithm running under default parameters in most cases, this is hardly worth an experimental investigation. For newly invented algorithms, a robust set of default parameters may not be available, thus the ad-hoc parameter values chosen to make it run can serve as such. However, it is even more necessary to explore parameter effects and interactions in this case, e.g. by applying tuning. Regardless of the statements above, it may be a viable research question to ask if *any* parameter setting for a new algorithm leads to a performance advantage over a standard algorithm, but then the next question should be: how robust are these parameters, or is the advantage only achieved for very specific problems, and if so, for which ones?

2.6 Performing the Experiments

Now that the research question and the performance measure for a given problem and algorithm are fixed, experiments can be performed. It is important to set up the scientific claims and their matching statistical hypotheses *before* looking at the obtained results [59] to achieve as much objectivity as possible. Otherwise, one could set up hypotheses in a way so that they are always supported, which renders the scientific contribution insignificant. On the other hand, this requires a change in how experimental results are received. More than the raw numbers or outcomes of statistical tests, the knowledge gain is important. Do the results contain previously unknown facts? This may also happen if two well-known algorithms, neither of which is new, are compared on a new set of benchmark functions; it is not necessary to always invent new algorithms. Or do they support or even refute known facts?

Additionally, we would like to give a practical hint here: "Never watch a running experiment." Once one can be sure that the experiment is indead going to produce useful data, one shall wait with the analysis until the experiment is finished. The reason is simply that one is easily mislead by the first results coming in and may get a wrong impression that is later on hard to get rid of, even in the face of the full data set, so that the experimenter is in danger of losing the necessary neutrality.

2.7 Key Features of Comparisons

Many experimental studies are based on comparisons. Consider, e.g., tuning procedures which can be used to improve algorithm's performance. Obviously, each tuning procedure itself requires comparisons, because the performance of the algorithm before, say

$$x = \mathrm{perf}(A), \tag{2}$$

and after the tuning, say $x^* = \mathrm{perf}(A^*)$ has to be compared. Many tuning procedures are based on stochastic data, i.e., noisy data. This noise can be caused by

1. the algorithm, e.g., evolutionary algorithms,
2. the problem, e.g., simulation model,
3. or both.

Therefore, the comparison of two real values x and x^* is not sufficient and multiple runs of the algorithm have to be performed. We are considering (at least) two data vectors: $\mathbf{x}$ and $\mathbf{x}^*$, where $\mathbf{x}$ denotes the vector of n performance values of the untuned algorithm A and $\mathbf{x}^*$ the vector of m runs of the tuned algorithm A^*. Note, a similar situation might occur even if algorithm and problem are purely deterministic when multiple problem instances are tested.

In many cases we are facing the following fundamental problem after all the experiments were performed, i.e., post-data: given two data sets, $\mathbf{x}$ and $\mathbf{x}^*$, representing data from associated algorithms A and A^*, respectively. Decide, whether A is better than A^*.

In order to answer this question, performance has to be measured. Although simple statistics such as the mean or median of the run time are adequate to gain a first impression of the algorithm's performance, a sound statistical analysis requires more sophisticated measures. At this stage, statistical tests can be are carried out.

R-1: Research question. State in short the general objective of the experiment.

R-2: Pre-experimental planning. Report the results of first tests which are important for setting up the main experiment, e.g. for choosing problems, parameter settings, termination criteria.

R-3: Task. Formulate one or several scientific claims (only applicable if more concrete than the research question) and give matching statistical hypotheses, together with significance conditions.

R-4: Setup. Here goes everything that is needed to replicate the experiment, if not previously described. This consists of the applied algorithms, test problems, parameter settings, important outer conditions (e.g., if relevant, details of the employed hardware). Now, experiments can be performed.

R-5: Results/Visualization. This holds numerical results or links to the tables or figures made of them and also reports on the outcome of the hypothesis tests.

R-6: Observations. Unexpected or otherwise notable behavior that has been detected by reviewing the results, *without* interpretation.

R-7: Discussion of the statistical relevance. Statistical hypotheses from step R-3 are reconsidered (accepted/rejected).

R-8: Discussion of the scientific meaning. Attempts to give explanations for the results/observations obtained and puts the results of the experiment in a context with other scientific findings. This paragraph is meant to contain subjective statements which might lead to new hypotheses or research questions based on the results from current experiments.

Fig. 2 Structured report

2.8 Reporting Results

After the results are in, they should be visualized to enable a basic consistency check. Figures are much easier to interpret than tables, so this effort is not wasted and greatly helps when looking for interesting effects. For conducting the experiment as well as to obtain a structured report of it, it may be helpful to work alongside the eight-step procedure presented in Fig. 2 and to write down decisions, setups and results as they are obtained during the experiment. This structure is largely similar to the one often applied in natural sciences for many decades.

Note that we separate observations from discussion. This may seem artificial and the distinction is not in all cases obvious. However, the intention is to keep objective differences apart from their interpretation. If e.g. an algorithm is surprisingly good on specific problem instances, this is surely an observation. Giving a presumed reason why this is the case belongs to the discussion, as another author may come up with another explanation even if the observation can be replicated.

2.9 Iterating the Experimental Process

As already stated by [55] and others, experimentation with algorithms should not limit itself to a one-shot event but rather be regarded as an iterated process where the results obtained from the first experiment lead to new hypotheses and a refined experimental setup for the next. E.g., it may happen that the first experiment revealed some unexpected facts and one has an intuition concerning the causing factors which can be tested in a second experiment.

In order to support the incremental experimentation process, we recommend to keep an *experimental journal* of all experiments undertaken in a specific context. The journal should contain at least a list of running numbers, time stamps, names/research questions, and a short description of the outcome and is helpful for obtaining an overview of the progress of an investigation and keeping the data well organized.

2.10 Scientifically Meaningful Results?

Finally, after performing all these tests as described in Sec. 2.7, one fundamental qualm remains: "How can we guarantee that results are scientifically meaningful?" This question is related to *post-data* information—it includes data which is available after the experiments were performed. We will focus on this question in the following by introducing the concept of severity. A technical treatment of the concept of severity is given in Sec. 6.

In the severe testing philosophy, the quantitative assessment offered by error statistics provides tools to test how well probed hypotheses are. Mayo [53] introduces the concept of severity as follows: "Stated simply, *a passing result is a severe test of hypothesis H just to the extent that it is very improbable for such a passing result to occur, were H false.*"

Although this approach is based on classical hypothesis testing, i.e., the Neyman-Pearson statistical paradigm, it is relevant to different statistical frameworks, e.g., non parametric approaches. Classical hypotheses testing dominates today's scientific publications, therefore this first approach is justified by everyday's practice.

3 Active Experimentation

Section 3.1 presents *active experimentation* as a framework, which implements features of the experimental methodology introduced in Sec. 2. This framework can be used for demonstration, robustness, comparison, understanding, and novelty detection. In Sec. 3.2 the sequential parameter optimization toolbox is introduced. It comprehends the computational steps of the active experimentation framework, i.e., design of experiments, response surface methods, or statistical analysis and visualization. Automated experimentation has gained much attention in the last years. Several automated approaches were proposed, especially in the context of demonstration and robustness, e.g., automated tuning of algorithms. Therefore, we will compare automated and interactive approaches in Sec. 3.3.

3.1 Definition

Definition 1 (Active Experimentation). *Active experimentation* (AEX) is a framework for tuning and understanding of algorithms. AEX employs methods from error statistics to obtain reliable results. It comprises the following elements:

AEX-1: Scientific questions
AEX-2: Statistical hypotheses
AEX-3: Experiments
AEX-4: Scientific meaning

□

These elements can be explained as follows. Starting point of the investigation is a scientific question (AEX-1). This question often deals with assumptions about algorithms, e.g., influence of parameter values or new operators. This (complex) question is broken down into (simple) statistical hypotheses (AEX-2) for testing. Next, experiments can be performed for each hypothesis:

a) Select a model, e.g., a linear regression model to describe a functional relationship.
b) Select an experimental design.
c) Generate data, i.e., perform experiments.
d) Refine the model until the hypothesis can be accepted/rejected.

Finally, to assess the scientific meaning of the results from an experiment, conclusions are drawn from the hypotheses. This is step (AEX-4) in the active experimentation framework. Here, the concept of severity as introduced in Sect. 6 comes into play.

3.2 Sequential Parameter Optimization Toolbox

We introduce the *sequential parameter optimization toolbox* (SPOT) as one possible implementation of the experimental approach in the AEX framework. The SPO *toolbox* was developed over recent years by Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss [8]. Main goals of SPOT are (i) the determination of improved parameter settings for optimization algorithms and (ii) to provide statistical tools for analyzing and understanding their performance.

Definition 2 (Sequential Parameter Optimization Toolbox). The sequential parameter optimization toolbox implements the following features, which are related to step AEX-3.

SPOT-1: Use the available budget (e.g., simulator runs, number of function evaluations) sequentially, i.e., use information from the exploration of the search space to guide the search by building one or several meta models. Choose new design points based on predictions from the meta model(s). Refine the meta model(s) stepwise to improve knowledge about the search space.

SPOT-2: If necessary, try to cope with noise (see Sect. 2.7) by improving confidence. Guarantee comparable confidence for search points.

SPOT-3: Collect information to learn from this tuning process, e.g., apply exploratory data analysis.

SPOT-4: Provide mechanisms both for interactive and automated tuning.

□

The article entitled "sequential parameter optimization" [8] was the first attempt to summarize results from tutorials and make this approach known to and available for a broader audience. Since 2004, a series of tutorials was presented during the leading conferences in the field of computational intelligence, e.g.,[19, 18, 16, 15, 14, 13, 11, 10].

SPOT was successfully applied in the fields of bioinformatics [79, 33, 32], environmental engineering [48, 30], shipbuilding [72], fuzzy logic [82], multimodal optimization [68], statistical analysis of algorithms [50, 78], multicriteria optimization [80], genetic programming [51], particle swarm optimization [9, 49], automated and manual parameter tuning [31, 74, 42, 43], graph drawing [77, 65], aerospace and shipbuilding industry [63], mechanical engineering [56], and chemical engineering [39]. Bartz-Beielstein [3] collects publications related to the sequential parameter optimization.

SPOT employs a sequentially improved model to estimate the relationship between algorithm input variables and its output. This serves two primary goals. One is to enable determining good parameter settings, thus SPOT may be used as a tuner. Secondly, variable interactions can be revealed that help to understand how the tested algorithm works when confronted with a specific problem or how changes in the problem influence the algorithm's performance. Concerning the model, SPOT allows the insertion of virtually every available meta model. However, regression and Kriging models or a combination thereof are most frequently used as prediction models (as defined as $\boldsymbol{F}$ in Algorithm 1). Bartz-Beielstein [4, 5] describes integration and usage of theses prediction models in SPOT.

Algorithm 1 presents a formal description of the SPOT scheme. This scheme consists of two phases, namely the first construction of the model

Algorithm 1: SPOT

```
   // phase 1, building the model:
1  let $A$ be the tuned algorithm;
2  generate an initial population $\mathbf{P} = \{\mathbf{p}^1, \ldots, \mathbf{p}^m\}$ of $m$ parameter vectors;
3  let $k = k_0$ be the initial number of tests for determining estimated utilities;
4  foreach $\mathbf{p} \in \mathbf{P}$ do
5      run $A$ with $\mathbf{p}$ $k$ times to determine the estimated utility $x$ of $\mathbf{p}$;
   // phase 2, using and improving the model:
6  while termination criterion not true do
7      let $\mathbf{a}$ denote the parameter vector from $\mathbf{P}$ with best estimated utility;
8      let $k'$ the number of repeats already computed for $\mathbf{a}$;
9      build prediction model $F$ based on $\mathbf{P}$ and $\{x^1, \ldots, x^{|\mathbf{P}|}\}$;
       // (alternatively: use several prediction models in parallel)
10     generate a set $\mathbf{P}'$ of $l$ new parameter vectors by random sampling;
11     foreach $\mathbf{p} \in \mathbf{P}'$ do
12         calculate $f(\mathbf{p})$ to determine the predicted utility $F(\mathbf{p})$;
13     select set $\mathbf{P}''$ of $d$ parameter vectors from $\mathbf{P}'$ with best predicted utility
       ($d \ll l$);
14     run $A$ with $\mathbf{a}$ $k - k' + 1$ times and recalculate its estimated utility using all
           $k + 1$ test results; // (improve confidence)
15     let $k = k + 1$;
       // (alternatively: use more enhanced update schemes like OCBA)
16     run $A$ $k$ times with each $\mathbf{p} \in \mathbf{P}''$ to determine the estimated utility $F(\mathbf{p})$;
17     extend the population by $\mathbf{P} = \mathbf{P} \cup \mathbf{P}''$;
```

(lines 1–5) and its sequential improvement (lines 6–17). Phase 1 determines a population of initial designs in algorithm parameter space and runs the algorithm k times for each design. Phase 2 consists of a loop with the following components:

1. Update the model by means of the obtained data.
2. Generate a (large) set of design points and compute their utility by sampling the model.
3. Select the seemingly best design points and run the algorithm for these.
4. The new design points are added to the population and the loop starts over if the termination criterion is not reached.

A counter k is increased in each cycle and used to determine the number of repeats that are performed for each setting to be statistically sound in the obtained results. Consequently, this means that the best design points so far are also run again to obtain a comparable number of repeats. SPOT provides tools to perform the following tasks:

1. *Initialize.* An initial design is generated. This is usually the first step during experimentation. The employed parameter region and the constant algorithm parameters have to be provided by the user.

2. *Run.* This is usually the second step. The optimization algorithm is started with configurations of the generated design. Additionally information about the algorithms problem design are used in this step. The algorithm provides the results to SPOT.
3. *Sequential step.* A new design is generated. A prediction model is used in this step. Several generic prediction models are available in SPOT already. To perform an efficient analysis, especially in situations when only few algorithms runs are possible, user-specified prediction models can easily be integrated into SPOT. Prediction models can also be used in parallel [7], which results in the so-called ensemble-based modeling approach. To improve confidence, the number of repeats can be increased. *Optimal computational budget allocation* (OCBA) [25, 12] is implemented as the default method for assigning new evaluations to algorithm configurations.
4. *Report.* An analysis, based on information from the results, is generated. Since all data flow is stored in files, new report facilities can be added very easily. SPOT contains some scripts to perform a basic regression analysis and plots such as histograms, scatter plots, plots of the residuals, etc.
5. *Automatic* mode. In the automatic mode, the steps *run* and *sequential* are performed after an initialization for a predetermined number of times.

3.3 Comparison of Automated and Interactive Tuning

SPOT can be run in an automated and in an interactive mode. The automated mode might be of interest for the user, who is primarily interested in the result and who can afford a tuning procedure which is not restricted to a very small number of algorithm runs. Similar to microscopes in biology, SPOT can be used as a "datascope" to gain insight into algorithm behavior, by revealing factor effects and their importance to the experimenter. Such insights can not only be used to guide the interactive parameter optimization process, but also be of intrinsic value to the developer or end user of a target algorithm.

The classical response surface methodology (as discussed in Chap. 15 of [23]) underlying our interactive approach was developed not only for finding parameter settings that achieve improved performance, but also to provide insights into how the performance of a target algorithm is affected by parameter changes. This latter question is related to the analysis of the response surface in the region of interest. Contour plots, which can easily be obtained in the SPOT framework, are useful tools to answer it.

We recommend using classical regression models at the first stage of an interactive approach, because these models can be interpreted quite easily; features of the response surface can be seen directly from the regression equation $Y = X\beta$. This is not the case for more sophisticated prediction models, such as neural networks or Gaussian process models. Furthermore, as demonstrated in [42], it is possible to obtain competitive results using such simple models. Nevertheless, in principle, more complex regression models could be used in the context of the interactive sequential parameter optimization approach. Furthermore, we note that observations and hypotheses regarding the dependence of a given target algorithm's performance on its parameter settings could also be obtained by analyzing more complex models, including the Gaussian process models constructed by the automatic sequential parameter optimization procedures.

Clearly, the interactive approach makes it easy to use results from early stages of the sequential parameter optimization process to effectively guide decisions made at later stages. For example, looking back at the initial stages of the process, the experimenter can detect that the set of variables studied at this stage was chosen poorly, or that inappropriate ranges were chosen for certain variables. We note that the models used in early stages of the automated procedures also provide guidance to later stages of the process. However, the interactive process leaves room for expert human judgment, which can often be more effective in terms of the improvement achieved based on a small number of target algorithm runs.

The human expertise required to use the interactive approach successfully can be seen as a drawback compared to fully automated approaches. However, by providing dedicated support for the various operations that need to be carried out in this context, SPOT eases the burden on the experimenter and lowers the barrier to using the interactive approach effectively.

4 Case Study: Tuning Simulated Annealing

This section presents a working example to demonstrate essential principles of AEX introduced in Sec. 3. The following study will be referred at later on in Sec. 7, which discusses meta-statistical principles. This case study was set up to illustrate key elements of the AEX framework. It does not present a complete analysis of the simulated annealing, but can serve as a starting point for an experimental analysis.

Our goal is to determine an improved parameter setting for a simulated annealing search heuristic. This goal can be formulated as the following scientific question(this refers to AEX-1):

> Can we modify the algorithm design in such a manner that SANN's performance and robustness is improved?

Furthermore, we are seeking for tools which provide support in deciding whether this improvement is scientifically (or in practice) meaningful.

4.1 Simulated Annealing

SANN belongs to the class of stochastic global optimization methods. The R implementation, which was investigated in our study, uses the Metropolis function for the acceptance probability. It is a variant of the simulated annealing algorithm given in [21]. SANN uses only function values but is relatively slow. It will also work for non-differentiable functions. By default the next candidate point is generated from a Gaussian Markov kernel with scale proportional to the actual temperature. Temperatures are decreased according to the logarithmic cooling schedule as given in [21]; specifically, the temperature is set to $\texttt{temp}/\log(((t-1)/\texttt{tmax})\times\texttt{tmax}+\texttt{exp}(1))$, where t is the current iteration step and `temp` and `tmax` are specifiable via control. SANN is not a general-purpose method but can be very useful in getting to a good value on a very rough surface.

SANN uses two design variables, which were tuned during our study:

`temp` is the starting temperature for the cooling schedule. Defaults to 10.
`tmax` is the number of function evaluations at each temperature. Defaults to 10.

The interval from 1 to 50 was chosen as the *region of interest* (ROI) for both design variables in our experiments. The total number of function evaluations (of the Branin function, see Sect. 4.2) was set to $\texttt{maxit} = 250$ for all experiments. The starting point, i.e., the initial value for the parameters to be optimized over, was $\mathbf{x}_0 = (10, 10)$.

4.2 Description of the Objective Function

The Branin function

$$f(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10 \times \left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10,$$

with

$$x_1 \in [-5, 10] \text{ and } x_2 \in [0, 15]. \tag{3}$$

was chosen as a test function, because it is well-known in the global optimization community, so results are comparable. It has three global minima, $\mathbf{x}_1^* = [3.1416, 2.2750]$, $\mathbf{x}_2^* = [9.4248, 2.4750]$ and $\mathbf{x}_3^* = [-3.1416, 12.2750]$ with $y^* = f(\mathbf{x}_i^*) = 0.3979$, $(i = 1, 2, 3)$.

Results from the corresponding tuning experiments will be used in Sec. 7.1 to discuss particular aspects of the active experimentation framework.

5 Hypothesis Testing

We will describe the classical Neyman-Pearson statistical framework, which includes pre-data statistics such as significance levels, errors of the first and second kind, and power.

5.1 Neyman-Pearson Tests

To illustrate the concept of hypothesis testing, we introduce some basics from statistics. Hypothesis testing is the key element of step AEX-2 in the active experimentation framework.

Following the definition in Mayo [53], *Neyman and Pearson* (N-P) tests can be described as follows. The set of all possible values of the sample $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ with realizations $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is $\mathscr{X}$ and Θ is the set of all possible values of the unknown parameters θ. A *statistical model* is represented as a pair $(\mathscr{X}, \Theta)$.

A null hypothesis, H_0, and an alternative hypothesis, H_1 are stated. These hypotheses partition the parameter space of the statistical model. The generic form of the null and alternative hypotheses is

$$H_0 : \theta \in \Theta_0 \text{ versus } H_1 : \theta \in \Theta_1, \text{ where } (\Theta_0, \Theta_1) \text{ is a partition of } \Theta.$$

We will use $P(\mathbf{x}; H)$ for the probability of $\mathbf{x}$ under H to avoid confusion with conditional probabilities, $P(\mathbf{x}|H)$, where H denotes a random variable (Bayes' rule).

A *test statistic* $d(\mathbf{X})$ reflects the distance from H_0 in the direction of H_1. To simplify the following discussion, we consider a sample $\mathbf{X} = (X_1, X_2, \ldots X_n)$ where the X_i's are assumed to be normal, independent, and identically distributed with known standard deviation σ, i.e., $X_i \sim \mathscr{N}(\mu, \sigma^2)$. Here, the unknown parameter θ is the mean μ of the normal distribution. The test statistic is

$$d(\mathbf{X}) = \frac{\overline{X} - \mu_0}{\sigma/\sqrt{n}} = \frac{\overline{X} - \mu_0}{\sigma_x}, \tag{4}$$

where $\overline{X}$ is the sample mean, μ_0 is the hypothesized population mean under the null hypothesis, and σ_x denotes the standard error. We will consider one-sided tests in the following. Based on the pre-specified α value, the *critical value* $c_{1-\alpha}$, which partitions Θ into the *region of acceptance*, $C_0(\alpha)$ and the *region of rejection*, $C_1(\alpha)$ of the null hypothesis, can be determined as the quantile $z_{1-\alpha}$ of the standard normal distribution.

$$C_0(\alpha) = \{\mathbf{x} \in \mathscr{X} : d(\mathbf{x}) \le c_{1-\alpha}\}$$
$$C_1(\alpha) = \{\mathbf{x} \in \mathscr{X} : d(\mathbf{x}) > c_{1-\alpha}\}.$$

The *type I error probability* (or error of the first kind, α error) is

$$P(d(\mathbf{X}) > c_{1-\alpha}; H_0) \leq \alpha,$$

and represents the probability that the null hypothesis is rejected erroneously. The *type II error probability* (or error of the second kind, β error) is

$$P(d(\mathbf{X}) \leq c_{1-\alpha}; H_1) = \beta(\mu_1),$$

where μ_1 is the hypothesized population mean under the alternative hypothesis, *Error-statistical methods* describe methods using error probabilities based on the relative frequencies of errors in repeated sampling. Probability is used to quantify how frequently methods are able of discriminating between alternative hypotheses and their reliability of error detection [54]. Following Mayo [53], we will use the term *error statistics* for hypothesis tests, statistical significance tests, and related error probability methods.

Example 1. We consider one particular test $T(\mathbf{x};\alpha;\theta) = T(\alpha)$ about the mean with significance level $\alpha = 0.025$. The null hypothesis $H_0 : \mu \leq \mu_0$ is tested versus the alternative hypothesis $H_1 : \mu > \mu_0$. Here, $c_{1-\alpha}$ can be determined as the quantile $z_{1-\alpha}$ of the standard normal distribution, e.g., $c_{1-0.025} = z_{1-0.025} = 1.96$, therefore

$$\begin{aligned} C_0(\alpha) &= \{\mathbf{x} \in \mathscr{X} : d(\mathbf{x}) \leq 1.96\}, \\ C_1(\alpha) &= \{\mathbf{x} \in \mathscr{X} : d(\mathbf{x}) > 1.96\}. \end{aligned}$$

Furthermore, let $\mu_0 = 12$, $\sigma = 2$, and $n = 100$. The null hypothesis $H_0 : \mu \leq 12$ is tested versus the alternative hypothesis $H_1 : \mu > 12$. The test rule derived from this test reads: Reject H_0 if $d(\mathbf{x}_0) > 1.96$, or if $\bar{x} = \mu_0 + d(\mathbf{x}_0) \times \sigma_x > 12.39$, see Fig. 3. If the observed value of the test statistic falls into the rejection region, we will reject the null hypothesis at a 2.5% significance level. □

Acceptance and rejection are associated with certain actions, e.g., publishing a result about effects of modifying the recombination operator of an evolutionary algorithm. But, how can we be sure that this action is justified, e.g., scientifically correct or meaningful? The *behavioristic rationale* answers this as follows:

> We are justified in 'accepting/rejecting' hypotheses in accordance with tests having low error probabilities because we will rarely err in repeated applications.[64]

This rationale, which was formulated by J. Neyman and E.S. Pearson in 1933, is based on the idea that error probabilities are means to determine "the evidence a set of data x_0 supplies for making warranted inferences about the process giving rise to data x_0." [53, p.5]

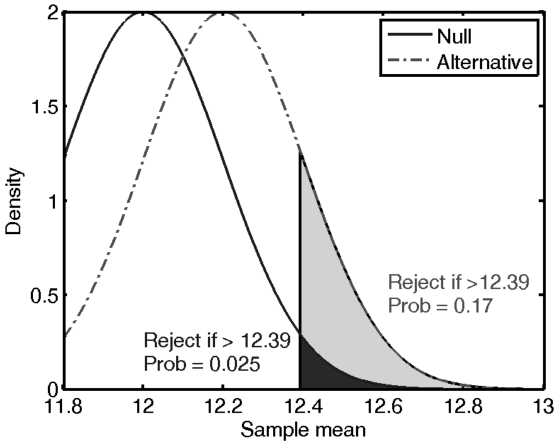


Fig. 3 One-sided hypotheses test. Based on the null and alternative hypotheses and α, the significance level, a test can be performed. We assume a known standard deviation, say $\sigma = 2$ and a sample size of $n = 100$. If the mean value $\bar{x}$ is larger than 12.39, the null hypothesis is rejected, otherwise it is accepted. The dark gray shaded region represents the Type I error probability. The alternative hypothesis reads $\mu_1 = 12.2$. The light gray shaded region (which includes also the dark gray shaded region) represents the power of the test, i.e., $1-\beta$

5.2 Power of a Test

The severity concept, which will be introduced in Sect. 6, is related to the power of a test. The power of a test is a standard concept in hypotheses testing. It is the test's probability of correctly rejecting the null hypothesis, i.e., the complement of the false negative rate, β. The *power of a test* is defined as

$$\mathrm{POW}(T(\alpha);\mu_1) = P(d(\mathbf{X}) > c_{1-\alpha};\mu_1), \text{ for } \mu_1 > \mu_0. \tag{5}$$

Power curves illustrate the effect on power of varying the alternate hypothesis. Severity, which uses post-data information, was introduced by Mayo [53] as "the attained power". Severity can be seen as an extension of the power of a test, cf. Sec.6.2.

Example 2. The power of the test specified in Example 1, where the null hypothesis $H_0 : \mu \leq 12$ is tested versus the alternative hypothesis $H_1 : \mu > 12$ can be determined with (5) as follows:

$$\mathrm{POW}(T(\alpha);\mu_1) = P(d(\mathbf{X}) > c_{1-\alpha};\mu = \mu_1) = P\left(\frac{\overline{X}-\mu_0}{\sigma_x} > c_{1-\alpha};\mu = \mu_1\right) \tag{6}$$

Since $\mu = \mu_1$,

$$\frac{\overline{X} - \mu_0}{\sigma_x} \qquad \text{follows a} \qquad \mathscr{N}\left(\frac{\mu - \mu_0}{\sigma_x}, 1\right)$$

distribution. Therefore

$$\mathrm{POW}(T(\alpha); \mu_1) = 1 - \Phi\left(c_{1-\alpha} + \frac{\mu_0 - \mu_1}{\sigma_x}\right), \tag{7}$$

where Φ denotes the cumulative distribution of the probability density function of the standard normal distribution, i.e.,

$$\Phi(x) = P(t < x) = \frac{1}{\sqrt{2\pi}} \int_{t=-\infty}^{t=x} \exp\left(-\frac{t^2}{2}\right) dt.$$

We are using the values from Example 1, i.e., $\alpha = 0.025$, $c_{1-0.025} = z_{1-0.025} = 1.96$, $\mu_0 = 12$, $\sigma = 2$, and $n = 100$. Power depends on the value of the alternative, i.e., μ_1. For $\mu_1 = 12.2$, we determine:

$$\begin{aligned} &\mathrm{POW}(T(\alpha = 0.025); \mu_1 = 12.2) \\ &= P\left(Z > 1.96 + \frac{12 - 12.2}{0.2}\right) = 1 - \Phi(0.96) = 0.1685. \end{aligned}$$

To determine the power for various μ_1 values, we obtain

$$\mathrm{POW}(T(\alpha = 0.025); \mu_1) = P\left(Z > 1.96 + \frac{12 - \mu_1}{0.2}\right) \text{ with } Z \sim \mathscr{N}(0,1), \tag{8}$$

see Fig. 4. □

Very significant results can be obtained with high power, even if the size of the effect is of no practical relevance: The effect is there, but its magnitude is of little value. This is similar to the situation with p values, see [17]. On the other extreme, a study with low power will have indecisive results, even if the effect is real and relevant.

N-P theory has been under attack, basically for the following three problems.

P-1: N-P tests are too coarse, because they tell us to reject or accept a certain hypothesis H, but do not indicate the level of rejection or acceptance.

P-2: Since statistical significance and not scientific importance is considered, N-P test give rise to fallacies of rejection and of acceptance.

P-3: N-P tests focus on pre-data, i.e., information from new data is not considered.

The power of a test does not depend on the experimental result $\mathbf{x}_0$, it remains the same for different outcomes. Even if the experimental result gives better evidence for accepting or rejecting the null hypothesis, the power will be identical. Power is no solution to problems P-1 to P-3, because the power of a test retains its coarseness, it does not consider its scientific importance, and

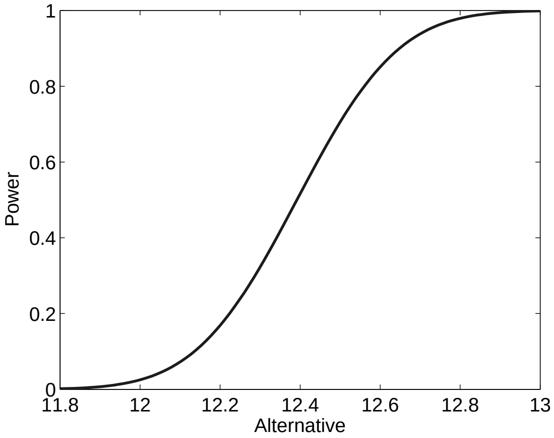


Fig. 4 Power of a test. We are using the values from Example 1, i.e., $\alpha = 0.025$, $c_{1-0.025} = z_{1-0.025} = 1.96$, $\mu_0 = 12$, $\sigma = 2$, and $n = 100$. Power depends on the alternative, i.e., μ_1. For $\mu_1 = 12.2$, we determine: $\mathrm{POW}(T(\alpha = 0.025); \mu_1 = 12.2) = 0.1685$

it relies on pre-experimental data. This applies to confidence intervals as well, because they do not consider the experimental outcome. Mayo introduces severity as a basic concept for post-data inference. Example 4 on page 26 illustrates the difference between power and severity.

6 Severity

6.1 Motivation

Severity provides a meta-statistical principle for evaluating proposed statistical inferences. It tells us how “well probed” (not “how probable”) hypotheses are and is an attribute of the test procedure as a whole. That is, severity should be calculated after the test procedure is finished. Once the data $\mathbf{x}_0$ of the test $T(\alpha)$ are available, they enable us to evaluate the severity of the test and the resulting inference.

Example 3. In order to exemplify the concept of severity, we consider the following situation (see also [52, p.183]): a randomized search algorithm, say (A), has scored high success rate on a test problem. It is able to detect the optimum in 96.3 percent of the runs. Consider the following situations:

1. First, suppose that it would be extraordinary for an algorithm, say A^*, who has no domain knowledge at all to have a score as high, or higher, than A. Is this score good evidence that A is well suited for solving this

problem? Based on A's and A^*'s test results, the severity rationale would be in this case that this inference is warranted.

2. Next, suppose that it would be no problem for an algorithm A^* who has no domain knowledge, e.g., random search, to have a score as high as 96 percent. Again, we may ask the same question: is this score of 96.3 percent good evidence that A is well suited for this test problem? Based on information about A's and A^*'s high score results, the severity rationale would be in this case that this inference is not warranted. The severity concept should provide tools for detecting ceiling effects.

□

6.2 Severe Tests

These considerations lead to the definition of severity as a concept for post-data inference. Here, we are facing the situation that a test has been performed and a decision "accept" or "reject" hypothesis H) has been made. The following definition of a severe test is presented in [53, p.7]:

Definition 3 (Severe Test). A statistical hypothesis H passes a *severe test* T with data $\mathbf{x}_0$ if,

S-1 $\mathbf{x}_0$ agrees with H, and

S-2 with very high probability, test T would have produced a result that accords less well with H than $\mathbf{x}_0$ does, if H were false.

□

Instead of calculating the power, which does not include information from the test result,

$$\mathrm{POW}(T(\alpha);\mu_1) = P(d(\mathbf{X}) > c_{1-\alpha};\mu = \mu_1),$$

for $\mu_1 > \mu_0$, see (5), Mayo [53] introduces the attained power or *severity*

$$\mathrm{SEV}(T(\alpha);d(\mathbf{x}_0);\mu \leq \mu_1) = P(d(\mathbf{X}) > d(\mathbf{x}_0);\mu > \mu_1) \qquad (9)$$

in case of acceptance of the null and

$$\mathrm{SEV}(T(\alpha);d(\mathbf{x}_0);\mu > \mu_1) = P(d(\mathbf{X}) \leq d(\mathbf{x}_0);\mu \leq \mu_1) \qquad (10)$$

in case of rejection of the null. In order to simplify notation, we suppress the arguments $T(\alpha)$ and $d(\mathbf{x}_0)$ in the following and use the abbreviations $\mathrm{SEV}(\mu \leq \mu_1)$ and $\mathrm{SEV}(\mu > \mu_1)$, respectively.

Equation 9 states that $\mu \leq \mu_1$ passes the test with high severity if there is a very high probability that $d(\mathbf{x}_0)$ would have been larger than it is, were $\mu > \mu_1$. And Equation 10 states that $\mu > \mu_1$ passes the test with high severity if there is a very high probability that $d(\mathbf{x}_0)$ would have been smaller than it is, were $\mu \leq \mu_1$. Note, severity depends on the test and the test result, i.e., it includes post-data information from $d(\mathbf{x}_0)$ instead of $c_{1-\alpha}$. Similar to the calculation of the power, the severity can be determined. Note, that based on severity criterion S-1, we have to determine whether data from the test result lead to an acceptance or an rejection of the hypothesis H.

6.2.1 Severity in the Case of Acceptance of the Null

First, the determination of severity of acceptance for test $T(\alpha)$ is considered. For example, this situation arises if no difference in means can be found. Based on the outcome $d(\mathbf{x}_0) \leq c_{1-\alpha}$, H_0 has survived the test. In this case, a statistically insignificant result ("accept H_0" or "$\mu \leq \mu_1$") is considered. Severity can be calculated as follows

$$\begin{aligned}
\mathrm{SEV}(\mu \leq \mu_1) =& P(d(\mathbf{X}) > d(\mathbf{x}_0); \mu \leq \mu_1 \text{ is false })\\
=& P(d(\mathbf{X}) > d(\mathbf{x}_0); \mu > \mu_1)\\
=& P\left(\frac{\overline{X}-\mu_0}{\sigma_x} > \frac{\overline{x}_0-\mu_0}{\sigma_x}; \mu > \mu_1\right)\\
=& P\left(Z > \frac{\overline{x}_0-\mu}{\sigma_x}\right) \text{ with } Z \sim \mathscr{N}(0,1)\\
=& 1-\Phi\left(\frac{\overline{x}_0-\mu}{\sigma_x}\right). \qquad (11)
\end{aligned}$$

□

Note, a t distribution is used if σ is unknown. In this case, the test statistic (4) reads

$$d(\mathbf{x}_0) = \frac{\overline{X}-\mu_0}{S_n/\sqrt{n}},$$

where S_n is defined as

$$S_n = \sqrt{\frac{1}{n-1}\sum(X_i-\overline{X})^2}.$$

6.2.2 Severity in the Case of Rejection of the Null Hypothesis

Severity can be calculated as

$$\mathrm{SEV}(\mu > \mu_1) = 1 - \mathrm{SEV}(\mu \leq \mu_1),$$

because

$$\begin{aligned}\mathrm{SEV}(\mu > \mu_1) =& P\left(d(\mathbf{X}) \leq d(\mathbf{x}_0); \mu > \mu_1 \text{ is false }\right)\\ =& \Phi\left(\frac{\bar{x}_0 - \mu}{\sigma_x}\right).\end{aligned}$$

□

A comparison of (7) and (11) shows that severity does not directly use the information from the critical value $c_{1-\alpha}$ and from the significance level α. This information is used indirectly, because the inference (accept/reject) is used to calculate severity.

Example 4. Similar to the calculation of the power of a test in Example 2 we will determine the severity. We are using the values from Example 1, i.e., $\alpha = 0.025$, $c_{1-0.025} = z_{1-0.025} = 1.96$, $\mu_0 = 12$, $\sigma = 2$, and $n = 100$. Again, the null hypothesis $H_0 : \mu \leq 12$ is tested versus the alternative hypothesis $H_1 : \mu > 12$. Similar to power, severity is evaluated at a point $\mu_1 = \mu_0 + \gamma$, where γ denotes the difference from μ_0 which is considered meaningful. Here, we have chosen $\gamma = 0.2$ which results in $\mu_1 = 12 + 0.2 = 12.2$. As can be seen from (11) severity depends on the experimental outcome, i.e., $\bar{x}_0$. For $\bar{x}_0 = 11.8$, we obtain:

$$\begin{aligned}\mathrm{SEV}(\mu \leq \mu_1) =& P\left(Z > \frac{11.8 - 12.2}{0.2}\right)\\ =& P(Z > -2) \text{ with } Z \sim \mathcal{N}(0,1)\\ =& 1 - \Phi(-2) = 0.977.\end{aligned}$$

□

In case of a rejection of the alternative, the power of a test provides a lower bound for the severity. This can be seen from (6) and (9). Power and severity are the same, if $d(\mathbf{x}_0)$ equals $c_{1-\alpha}$.

6.2.3 Usage of the Severity Concept

The framework presented in this section can be used as a meta-statistical check to evaluate which inferences are warranted. Figure 5 illustrates this check. The severity for three different outcomes $\bar{x}_0$ is shown. Severity increases for smaller values of $\bar{x}_0$. Power curve and severity curve coincide for $\bar{x}_0 = 12.39$, i.e., $d(\mathbf{x}_0) = c_{1-\alpha}$ or $\bar{x}_0 = \mu_0 + c_{1-\alpha} \times \sigma_x$.

The curves from Fig. 5 can be used to compare the severity of the assertion $\mu \leq 12.4$ for different experimental outcomes: First, $\bar{x}_0 = 12.1$ is considered. Here, we obtain

$$\begin{aligned}&\mathrm{SEV}(T(\alpha = 0.025); d(\mathbf{x}_0) = 12.1; \mu \leq 12.4)\\ &= P\left(Z > \frac{12.1 - 12.4}{0.2}\right) = 1 - \Phi(-1.5) = 0.9332.\end{aligned}$$

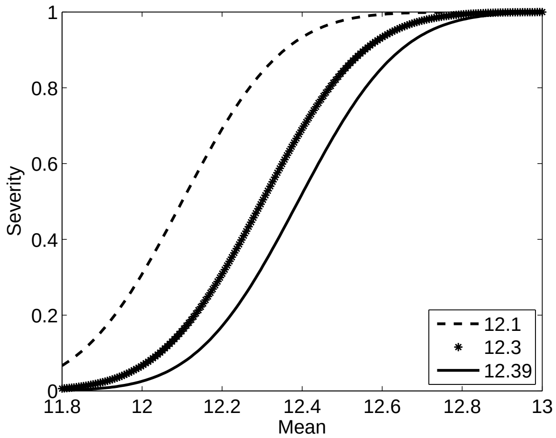


Fig. 5 Severity for three different results $\bar{x}_0$: 12.1, 12.3, and 12.39. These curves can be interpreted as follows: consider, e.g., $\bar{x}_0 = 12.3$, which gives $d(\mathbf{x}_0) = 1.5$: the assertion that $\mu \leq 13$ severely passes because $\mathrm{SEV}(\mu \leq 13) = 0.9998$

So, the assertion $\mu \leq 12.4$ passes with high severity. Next, the experimental outcome $\bar{x}_0 = 12.39$ is considered. Here, we obtain

$$\begin{aligned}&\mathrm{SEV}(T(\alpha = 0.025); d(\mathbf{x}_0) = 12.39; \mu \leq 12.4)\\&= P\left(Z > \frac{12.39 - 12.4}{0.2}\right) = 1 - \Phi(-0.05) = 0.5199,\end{aligned}$$

i.e., this experimental result decreases severity.

The severity curves from Fig. 5 can also be used in the following way: Let $\bar{x}_0 = 12.1$. The practitioner selects a relatively high severity value, say 0.9. The related μ value is calculated, say 12.4. That is, the assertion that "$\mu \leq 12.4$" severely passes with the result $\bar{x}_0 = 12.1$.

In addition, we present one useful application of the severity criterion for the experimental analysis of algorithms.

Example 5. Comparing a newly developed algorithm A^* to the best known algorithm A might lead to a situation where the result is interpreted too readily as positive evidence of no difference in their performances. Here, we are using the following values: $\alpha = 0.025$, $c_{1-0.025} = z_{1-0.025} = 1.96$, $\mu_0 = 0.0$, $\sigma = 2$, and $n = 100$. The null hypothesis $H_0 : \mu \leq 0$ is tested versus the alternative hypothesis $H_1 : \mu > 0$. Let the test $T(\alpha)$ yield a statistically insignificant result $\bar{x}_0 = 0.3$, i.e., the alternative is rejected. The experimenter states that "any discrepancy from $\mu_0 = 0$ is absent or no greater than 0.1." How severely does $\mu \leq 0.1$ pass with $\bar{x}_0 = 0.3$? We obtain

$$\begin{aligned}&\mathrm{SEV}(T(\alpha = 0.025); d(\mathbf{x}_0) = 0.3; \mu \leq 0.1)\\ &= P\left(Z > \frac{0.3-0.1}{0.2}\right) = 0.1587.\end{aligned}$$

So, even if a difference of 0.1 exists, such a result would occur 84 % of the time. Clearly, severity does not support the experimenter's statement in this case. □

Severity was developed as an error statistical tool in the framework of the *new experimentalism.* The new experimentalists claim that theories present only heuristic rules, leading us to experimental knowledge [24]. They view progress in terms of the accumulation of experimental discoveries. These findings are independent of high-level theory. How to produce scientifically meaningful results is the central theme in the research of the new experimentalists. Bartz-Beielstein [1, 2] demonstrates how these concepts can be transferred from the philosophy of science to computer science.

7 Meta-statistical Principles

This section refers to the third and fourth step (AEX-3 and AEX-4, respectively) of the active experimentation framework. As introduced in Sec. 4, we will present a working example to illustrate particular aspects of the active experimentation framework. Next, experiments will be performed. The corresponding results are shown in Sec. 7.1. These results will be used to discuss differences between statistical significant and scientifically meaningful results in Sec. 7.2. Finally, ceiling effects will be revisited in Sec. 7.3.

7.1 Results from Default, Random, and Tuned Settings

Experiments are performed at this stage, i.e., SPOT is used to execute algorithm runs in an objective and reproducible manner. This is step AEX-3 from the active experimentation framework. As a baseline for our experiments, we run SANN one hundred times—first, with default parameters (`tmax` = `temp` = 10), and second, with randomly chosen parameter values from the interval $[1,50]$.[1] These experiments were performed to quantify the benefit of tuning for our experiments. Results from these two experiments are shown in the first and second result row from Tab. 1. SANN was not

[1] SPOT can generate one hundred randomly chosen design points of the SANN by using the following setting in the CONF file: `init.design.size = 100` and `init.design.repeats = 1`.

able to determine the optimal function value with these settings. Now that the baseline results are available, we can examine SANN's tunability.

Table 1 SANN results. Results from $n = 100$ repeats. Smaller values are better. The optimal function value is $y^* = 0.3979$

Model	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Default	0.3982	0.4037	0.4130	0.8281	0.5032	6.1120
Random	0.3988	0.5326	1.2160	2.0720	2.9820	8.8800
Tuned	**0.3979**	**0.3987**	0.4000	**0.4010**	**0.4022**	**0.4184**

The final best configuration found by SPOT reads `temp` = 1.115982 and `tmax` = 38.

Now that these results are available, we would like to determine their statistical significance and scientific meaning.

7.2 Spurious Effects

As mentioned in Sect. 2.7, our focus lies on meta-statistical principles that can be applied after the experiments are performed. These principles are necessary to avoid fallacies, i.e., misconceptions resulting from incorrect reasoning in argumentation caused by spurious effects. Following Cohen [28], we define *spurious effects* as effects that suggest that a treatment is

- effective when it is not or
- not effective when it is.

One prominent example for spurious effects is the *ceiling effect*. If one wants to investigate performance differences between different methods, it is important to select the test problems/settings so that these differences indeed can occur. It is of little interest to see result tables with nearly all the methods always obtaining success rates of 100%. This would be a *ceiling effect*: the test problems are too easy, all algorithms "crash into the ceiling". On the other hand, test problems can also be too hard, then we have a *floor effect* because most measured algorithms never obtain a measurable progress: all remain "on the floor". Especially quality tasks that may not be reached (and thus not counted for success rates) have to be set up with care. When floor/ceiling effects occur, there is almost no variability in the data and thus all compared algorithms appear to obtain similar performance. Ceiling effects occur when test problems are not sufficiently challenging. In the hypothesis testing framework from statistics, the situation can be formulated as the following claim.

Claim 1 *Let A and A^* denote two algorithms and consider the hypothesis*

$$H : perf(A) \geq perf(A^*).$$

If A and A^* achieve a performance which is close to the maximum level of performance, H should not be confirmed due to a ceiling effect. □

Claim 1 describes a situation in which there is a high probability that algorithms A and A^* reach a similar high performance, i.e., the difference in their performances $\text{perf}(A^*) - \text{perf}(A)$ is small. This corresponds to S-1 and S-2 from Definition 3: With very low probability, the comparison of perf(A) with perf(A^*) would have produced a result that accords with the hypothesis "H: there is no difference in their performances" as well or better than the test result does, if H were false and a given difference were present. Consequently, severity can be used to detect ceiling effects.

7.3 Ceiling Effects Revisited

Now the necessary tools for performing a post-data analysis are available and can be applied to the results from the SANN case study (see Sect. 4). This refers to the fourth step of the active experimentation framework, i.e., AEX-4. The function `spotSeverity()` and a related plotting functions are implemented in the R version of the SPO toolbox[2] which is available via CRAN [4].

Summary statistics from these two run configurations were shown in Tab. 1. These results indicate that SANN with tuned parameters outperforms SANN with default parameters. We will apply error statistical tools to analyze the scientific meaning of this result. Based on results from this case study, a power and a severity plot is generated, see Fig.6. A histogram illustrates the importance of EDA tools:

In a second experiment, we increased the number of SANN function evaluations from `maxit` = 250 to 1,000,000. This problem design is used to illustrate a ceiling effect (the problem is too easy, because of the large number of function evaluations): again, each configuration is run one hundred times. First, we will compare simple summary statistics from these two run configurations, see Tab. 2.

Both algorithms show the same behavior (up to four digits after the decimal) if the number of function evaluations is set to $10e6$. However, a t-test claims that there is a statistical significant difference in means. In addition, we generate a t-test:

[2] R is a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques. CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R, see http://cran.r-project.org.

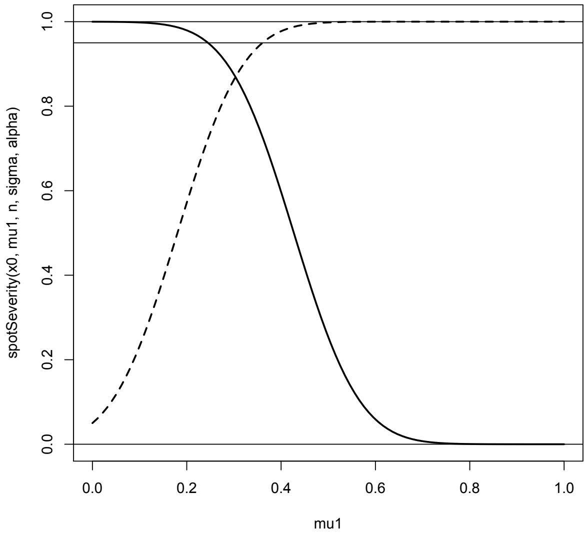


Fig. 6 Comparison of the tuned and the default SANN configuration on the Branin function with 250 function evaluations. Each run configuration was run one hundred times. The null hypothesis “there is no difference in means” is rejected. The dotted line illustrates the power of the test, whereas the *solid* line represents the severity. This plot was generated with the function `spotPlotSeverity()` from R’s SPOT package

Table 2 SANN results. Results from $n = 100$ repeats. Smaller values are better. The optimal function value is $y^* = 0.3979$

Model	`maxit`	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Tuned	250	**0.3979**	**0.3987**	0.4000	**0.4010**	**0.4022**	**0.4184**
Default	250	0.3982	0.4037	0.4130	0.8281	0.5032	6.1120
Tuned	1e6	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
Default	1e6	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979

```
        Paired t-test
t = -8.8975, df = 99, p-value = 1.384e-14
alternative hypothesis: true difference in means is less than 0
```

In this situation, error statistical tools and tools from exploratory data analysis might be helpful. Figure 8 shows the corresponding plots of power and severity. The severity interpretation of rejection shows that only very small differences in mean ($\mu < 2e-6$) pass with high severity.

A histogram (see Fig. 9) illustrates the importance of EDA tools. Results from the tuned SANN have a smaller standard deviation. However, it is up to the practitioner to decide whether a difference as small as $1e-6$ is of

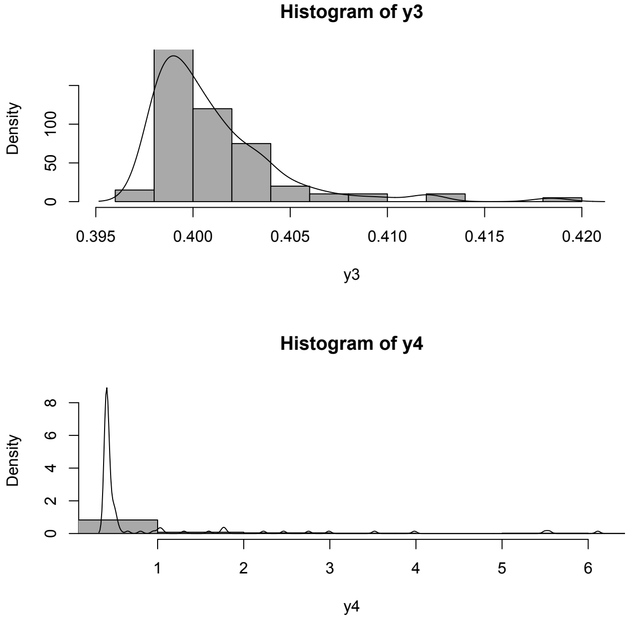


Fig. 7 Comparison of the tuned and the default SANN configuration on the Branin function with 250 function evaluations. Each run configuration was run one hundred times. The null hypothesis "there is no difference in means" is rejected

importance for this kind of problem instance. Error statistical tools provide support for this decision.

Now, we can answer the question from Sec. 4:

> SANN's performance and robustness could be improved. Severity and EDA provide useful decision support tools.

8 Exploratory Landscape Analysis

If the treated problem is not a constructed benchmark and thus most of its properties are unknown, it makes sense to use the test runs done with the optimization algorithm of choice for acquiring some additional problem knowledge. If one evaluation of the optimization problem takes on the order of minutes or more to compute, one cannot simply apply standard tuning techniques (as documented in Sect. 3). Instead, one could generate a surrogate model from the evaluated points and tune on this surrogate problem [69], or integrate the tuning process within the algorithm itself, which is e.g. easily

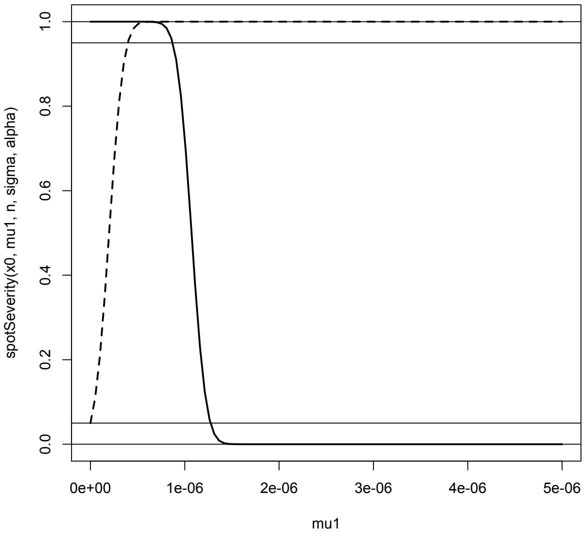


Fig. 8 Comparison of the tuned and the default SANN configuration on the Branin function with $1e6$ function evaluations. Each run configuration was run one hundred times. The null hypothesis "there is no difference in means" is rejected. The dotted line illustrates the power of the test, whereas the *solid* line represents the severity

possible when restarts are performed [81]. *Exploratory Landscape Analysis* (ELA) [58, 57] follows another approach, namely to detect problem properties first in order to to make a reasonably informed decision for some optimization algorithm.

8.1 Important Problem Properties

Problem properties which need to be determined for setting up an optimization algorithm matching the problem well are (more than these may be suitable, depending on the problem and the optimization algorithm):

Multimodality Most classic optimization algorithms inherently expect a unimodal (convex) problem. However, experience shows that most simulator-based problems are multimodal. But how multimodal? Does it have few local optima (as Schwefel's problem 2.13), or many (as Rastrigin's problem)? In the first case, a niching or time parallel method may be useful, in the latter case, one has to rely on multistarts (see [66] for a discussion) or on using large populations to inherently average out the many peaks in

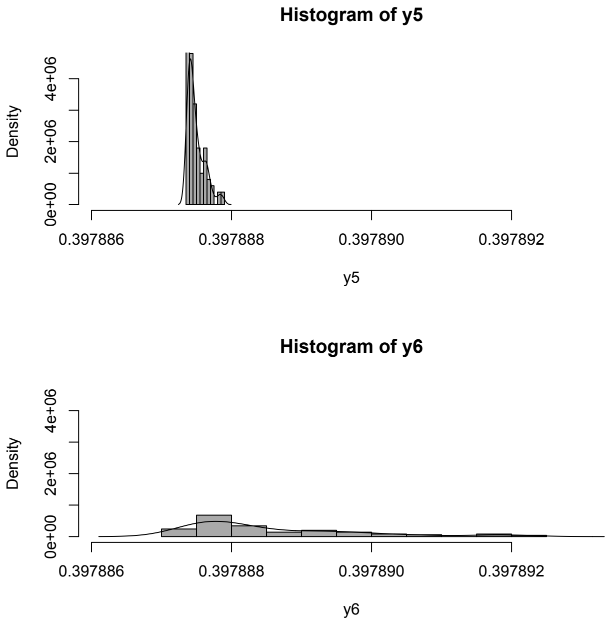


Fig. 9 Comparison of the tuned and the default SANN configuration on the Branin function with $1e6$ function evaluations. Each run configuration was run one hundred times. The null hypothesis "there is no difference in means" is rejected

order to detect the global basin structure. If indeed a convex problem is found, classic optimization methods as BFGS (see e.g. [62]) are most likely more effective than evolutionary ones.

Global basin structure Rastrigin's problem is not as difficult as it may seem at first. It has a huge amount of local optima, but also a global basin structure due to the quadratic term: seen from a large distance, it appears as parabola. Problems without global structure are more difficult because one virtually needs to "look in every corner." As an example, one may refer to the Gaussian mixture problem generator by Gallagher [34]. See [36] for more examples.

Separability If a problem is fully or partly separable, it may be partitioned into subproblems which are then of lower dimensionality and should be considerably easier to solve. For benchmark problems, it is known that separable problems get inseparable by simple geometric transformations as rotation [73].

Variable scaling Even if the considered search space bounds are the same for all variables, the problem may behave very different in the single dimensions. It can be essential to perform small steps in some dimensions, and large ones in others. Some algorithms as e.g. the CMA-ES [37] handle such problems well, but most standard EA variants do not.

Search space homogeneity Most benchmark sets are created with a homogeneous problem structure in mind, which is expressed by a single, relatively simple formula. However, real-world problems do not necessarily behave like this. The CEC'05 benchmark set [75] contains hybrid problems that consist of different ones blended into another, so that the resulting problem behaves differently in different search space areas.

Basin size homogeneity As e.g. emphasized by Törn [76], the basin size of the global optimum certainly influences the hardness of a problem. However, in the presence of many optima, the size relations of all encountered basins can lead to additional difficulties. Many algorithms for multimodal problems (e.g. most niching EA methods) assume similar basin sizes and use appropriately adjusted distances for differentiating between basins. If size differences are huge, these methods are doomed to fail.

Global to local optima contrast This property refers to the height (quality) differences between global and local peaks in comparison to the average fitness level of a problem. It thus determines if very good peaks are easily recognized as such. Together with basin size homogeneity, the influence of this property on niching methods has been reviewed in [70].

Size of plateaus Plateaus make optimization problems harder as they do not provide any information about good directions to turn to. Large plateaus effectively cut the search space into parts that prevent path oriented optimization algorithms from moving from one embedded peak area (possibly also a multimodal landscape itself) to another one.

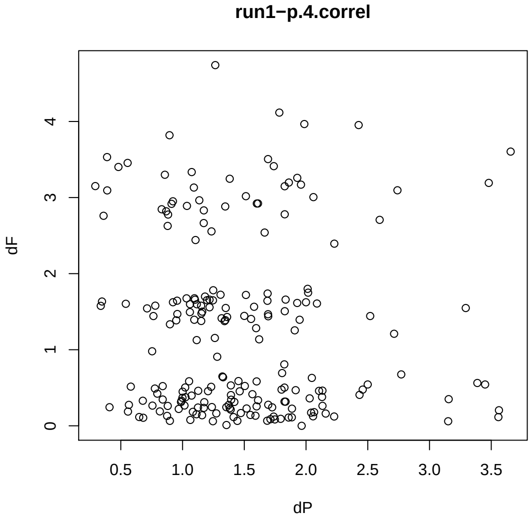
This set of properties is a rather phenomenologically (in the sense that these features have been either explicitly modeled into benchmark problems or observed in visualizations of existing ones) motivated collection which stresses the importance of the global structure in real-valued search spaces. Some time ago, many measures based on mathematical properties of mainly bit-coded problems have been suggested and employed for expressing hardness of problems for evolutionary algorithms. The most prominent of these may be the *fitness-distance correlation* (FDC) [46]. However, further theoretical investigations of Jansen [44] and He et al. [38] have largely found the existing measures unsuitable for predictive purposes, so watching out for new properties surely makes sense. Furthermore, in exploratory landscape analysis, one is especially interested in what can be achieved with only few evaluations of the problem, as the ultimate goal usually is to set up a good optimization algorithm for expensive problems with unknown properties.

8.2 Exploratory Testing

Attempts of experimentally acquiring property knowledge on expensive functions are usually performed manually, without a guiding algorithm, rather following the intuition of the experimenter which during the process adapts to the already known facts.

Sampling, dimension reduction techniques and especially visualization are important techniques to obtain problem knowledge. Interestingly, search points visited during stagnation phases may reveal interesting problem properties. We provide a real-world example as proof of concept. Figure 10 shows a fitness distance correlation plot from the last best point of one optimization run, using all successively sampled points (around 300). The treated problem [72] is the 15-variable engineering task to construct a ship propulsion system with high efficiency and low cavitation (low pressure bulbs at high velocity spots that lead to noise and deterioration), formulated as constraint penalized single objective problem. Simulation times are on the order of minutes. The question we tackled here was to find out why optimization always got stuck early. The plot shows several layers of fitness values, most likely stemming from the penalization when hitting a constraint.

Fig. 10 Quality to search space distance correlation around a good solution, Δquality (dF) over Δsearch space distance (dP). Obtained from the last 300 samples performed by an (1+1)-ES on a mediocre expensive engineering problem. All sampled points are equal or worse than the center point. The plot reveals a layered structure which is most likely due to constraint penalties applied to infeasible solutions.



Exploratory testing (sampling) can focus either on global or local features of a problem. Global sampling may employ any space-filling design as e.g. a *Latin hypercube design* (LHD) which is useful for obtaining a rough idea of the problem nature. Putting the sampled points into a model enables visualizing the landscape, e.g. Kriging models as employed in Sect. 3 may be used. However, note that a model comes with certain assumptions (such as smoothness of the landscape), so that some features as e.g. high-frequency ruggedness may completely go unnoticed.

Testing locally makes sense if one needs to find out which properties of the problem lead to stagnation in the optimization process, and then shall be conducted in the neighborhood of the best yet obtained points. However, for high-dimensional problems, running extensive grid tests in all possible variable combinations around a best solution is infeasible. Nevertheless, one may try grid tests in some combinations, especially if the variables can be grouped according to domain knowledge.

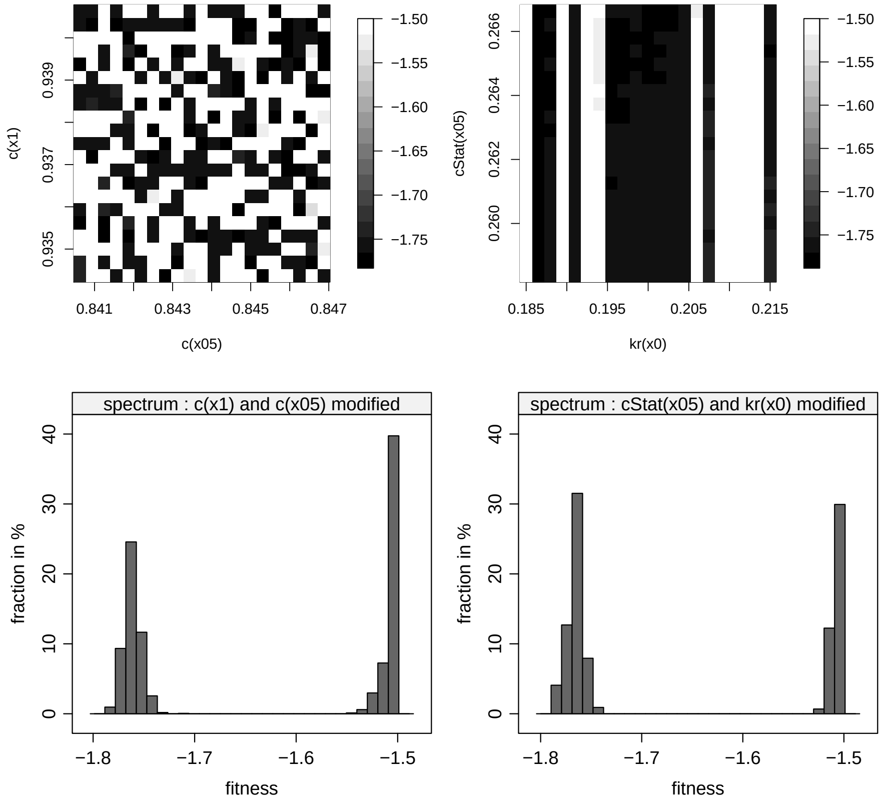


Fig. 11 Grid sample around a good search point identified in a prior optimization run, 20% of the available search space in each of the 2 tested dimensions were covered with 21x21 samples. Upper row: two different 2-variable combinations visualized as contour plot. Lower row: histograms of same data. The variables on the left are expected to interact strongly, the ones on the right should be nearly uncorrelated

We investigate this for the ship propulsion problem described above, choosing two variable pairs with different properties. The first two variables, $c(x1)$ and $c(x05)$ both refer to the shape of the rotor and should interact strongly, the other two ($cStat(x05)$ and $kr(x0)$) are expected to interact only weakly. Figure 11 shows the results of 21x21 point grid scans around the same best found point as employed in figure 10, keeping all other variables constant. The histograms help in assessing the frequency of encountering infeasible solutions (fitness values around -1.5) which is surprisingly similar in both situations, while the landscapes are completely different. For the two strongly interacting variables, we obtain a highly rugged landscape, but for the weakly interacting ones it is relatively flat with linear cliffs. We can safely assume that this contrast makes the problem harder.

An alternative approach to grid scans would be a local model (e.g. generalized linear or kriging), generated from a space-filling sample around the interesting point. However, even identifying the properties of optimization problems near high fitness spots may keep other important properties secret. Unless a rather complete scan of the problem in all dimensions is possible (which would remove the necessity for an optimization algorithm), we can not be sure to have revealed the information that could be helpful for setting up an optimization process fully. Our landscape analysis remains exploratory.

9 Summary and Future Developments

In this chapter, we have run through the current state of experimental research in evolutionary computation as we see it.

We presented severity as a meta-statistical rule for evaluating statistical tests. Limitations of simply accepting or rejecting hypotheses are avoided, which refers to Problem P-1 from Sect. 5. If an inference could only be said to pass a test with low severity, then there fails to be evidence for this inference. Summarizing, severity provides a method that quantifies the extend of the difference from the null hypothesis that is (or is not) warranted by data $\mathbf{x}_0$. This provides an answer to Problem P-2. One important feature of the severity concept is the extension of significance level and power, which are pre-data error probabilities. In contrast to the power of a test, severity uses values from the test statistic, and enables a post-data interpretation of statistical tests. This provides an answer to problem P-3.

Active experimentation is a flexible and general framework which can be applied in many situations. Note, that we do not claim that AEX is the only suitable way for tuning algorithms. Far from it! We state that AEX presents only one possible way—which might not be the best for your specific problem. We highly recommend other approaches in this field, namely F-race [22] and REVAC [60].

We have also elaborated on the specific problems when setting up an experimental investigation in evolutionary computation, and provided hints on how to avoid the most common mistakes, next to suggestions for how to write up and iterate experiments in order to concretize and validate the findings.

For cases where the problem properties are largely unknown, we suggest employing an *exploratory landscape analysis* approach which is in a relatively unstructured way often applied by practitioners already, without considering it as a working scheme of its own. Better understanding of (optimization) algorithm performance however needs to achieve some kind of parameter to property matching and on this path, visualization may play the key role.

As a general conclusion, we feel that still, some more emphasis on experimental methodology is needed and much work is still left undone in this area. Especially, the cooperation between theory and practice shall be improved, and moving towards each other may be an important task for the near future: theory should consider current experimental results as starting point for investigations, and established theory should be validated (e.g. concerning assumptions made) by means of structured experimental analysis.

Acknowledgements

This work has been supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grants FIWA (AIF FKZ 17N2309), MCIOP (AIF FKZ 17N0311), and by the Cologne University of Applied Sciences under the research focus grant COSA.

References

1. T. Bartz-Beielstein. *Experimental Research in Evolutionary Computation—The New Experimentalism.* Natural Computing Series. Springer, Berlin, Heidelberg, New York, 2006.
2. T. Bartz-Beielstein. How experimental algorithmics can benefit from Mayo's extensions to Neyman-Pearson theory of testing. *Synthese*, 163(3):385–396, 2008. DOI10.1007/s11229-007-9297-z.
3. T. Bartz-Beielstein. Sequential parameter optimization—an annotated bibliography. CIOP Technical Report 04/10, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science, April 2010.
4. T. Bartz-Beielstein. SPOT: An R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. CIOP Technical Report 05/10, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science, Jun 2010. Comments: Related software can be downloaded from http://cran.r-project.org/web/packages/SPOT/index.html.

5. T. Bartz-Beielstein. Writing interfaces for the sequential parameter optimization toolbox SPOT. CIOP Technical Report 07/10, Cologne University of Applied Sciences, Cologne University of Applied Science, Faculty of Computer Science and Engineering Science, July 2010.
6. T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors. *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, Berlin, Heidelberg, New York, 2010.
7. T. Bartz-Beielstein, M. Friese, O. Flasch, W. Konen, P. Koch, and B. Naujoks. Ensemble-based modeling. CIOP Technical Report 06/11, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science, July 2011.
8. T. Bartz-Beielstein, C. Lasarczyk, and M. Preuß. Sequential parameter optimization. In B. McKay et al., editors, *Proceedings 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland*, volume 1, pages 773–780, Piscataway NJ, 2005. IEEE Press.
9. T. Bartz-Beielstein, K. E. Parsopoulos, and M. N. Vrahatis. Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis and Computational Mathematics (ANACM)*, 1(2):413–433, 2004.
10. T. Bartz-Beielstein and M. Preuss. CEC tutorial on experimental research in evolutionary computation. In *IEEE Congress on Evolutionary Computation, Tutorial Program*, 2004. Tutorials given at CEC in 2004 and 2005.
11. T. Bartz-Beielstein and M. Preuß. Experimental research in evolutionary computation (tutorial). Genetic and Evolutionary Computation Conf. (GECCO 2005), Washington DC, June, 2005.
12. T. Bartz-Beielstein and M. Preuss. Considerations of budget allocation for sequential parameter optimization (SPO). In L. Paquete et al., editors, *Workshop on Empirical Methods for the Analysis of Algorithms, Proceedings*, pages 35–40, Reykjavik, Iceland, 2006.
13. T. Bartz-Beielstein and M. Preuß. Experimental research in evolutionary computation (tutorial). Genetic and Evolutionary Computation Conf. (GECCO 2006), Seattle WA, July, 2006.
14. T. Bartz-Beielstein and M. Preuß. Experimental research in evolutionary computation–the future of experimental research (tutorial). Genetic and Evolutionary Computation Conf. (GECCO 2007), University College London UK, July, 2007.
15. T. Bartz-Beielstein and M. Preuß. Experimental research in evolutionary computation–the future of experimental research (tutorial). Genetic and Evolutionary Computation Conf. (GECCO 2008), Atlanta, Georgia, US, July, 2008.
16. T. Bartz-Beielstein and M. Preuß. Experimental research in evolutionary computation–the future of experimental research (tutorial). Genetic and Evolutionary Computation Conf. (GECCO 2009), Montreal, CA, July, 2009.
17. T. Bartz-Beielstein and M. Preuss. The future of experimental research. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 17–46. Springer, Berlin, Heidelberg, New York, 2010.
18. T. Bartz-Beielstein and M. Preuß. Tuning and experimental analysis in evolutionary computation: what we still have wrong (tutorial). Genetic and Evolutionary Computation Conf. (GECCO 2010), Portland, OR, July, 2010.
19. T. Bartz-Beielstein and M. Preuss. Automatic and interactive tuning of algorithms. In N. Krasnogor and P. L. Lanzi, editors, *GECCO (Companion)*, pages 1361–1380. ACM, 2011.
20. R. E. Bechhofer, T. J. Santner, and D. M. Goldsman. *Design and Analysis of Experiments for Statistical Selection, Screening, and Multiple Comparisons*. Wiley, New York NY, 1995.

21. C. J. P. Belisle. Convergence theorems for a class of simulated annealing algorithms. *Journal Applied Probability*, 29:885–895, 1992.
22. M. Birattari. *Tuning Metaheuristics*. Springer, Berlin, Heidelberg, New York, 2005.
23. G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for Experimenters*. Wiley, New York NY, 1978.
24. A. F. Chalmers. *What Is This Thing Called Science*. University of Queensland Press, St. Lucia, Australia, 1999.
25. C. H. Chen. An effective approach to smartly allocate computing budget for discrete event simulation. In *Proceedings of the 34th IEEE Conference on Decision and Control*, pages 2598–2605, 1995.
26. M. Chimani and K. Klein. Algorithm engineering: Concepts and practice. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010.
27. P. R. Cohen. A survey of the eighth national conference on artificial intelligence: Pulling together or pulling apart? *AI Magazine*, 12(1):16–41, 1991.
28. P. R. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge MA, 1995.
29. A. E. Eiben and M. Jelasity. A critical note on experimental research methodology in EC. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002)*, pages 582–587. IEEE Press, 2002.
30. O. Flasch, T. Bartz-Beielstein, A. Davtyan, P. Koch, W. Konen, T. D. Oyetoyan, and M. Tamutan. Comparing CI methods for prediction models in environmental engineering. CIOP Technical Report 02/10, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Faculty of Computer Science and Engineering Science, Cologne University of Applied Sciences, Germany, Feb 2010.
31. T. Fober. Experimentelle Analyse Evolutionärer Algorithmen auf dem CEC 2005 Testfunktionensatz. Master's thesis, Universität Dortmund, Germany, 2006.
32. T. Fober, S. Glinca, G. Klebe, and E. Hüllermeier. Superposition and alignment of labeled point clouds. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(6):1653–1666, 2011.
33. T. Fober, M. Mernberger, G. Klebe, and E. Hüllermeier. Evolutionary construction of multiple graph alignments for the structural analysis of biomolecules. *Bioinformatics*, 25(16):2110–2117, 2009.
34. M. Gallagher and B. Yuan. A general-purpose tunable landscape generator. *IEEE Trans. Evolutionary Computation*, 10(5):590–603, 2006.
35. N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.
36. N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.
37. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
38. J. He, C. Reeves, C. Witt, and X. Yao. A note on problem difficulty measures in black-box optimization: Classification, realizations and predictability. *Evolutionary Computation*, 15(4):435–443, 2007.
39. F. Henrich, C. Bouvy, C. Kausch, K. Lucas, M. Preuß, G. Rudolph, and P. Roosen. Economic optimization of non-sharp separation sequences by means of evolutionary algorithms. *Computers and Chemical Engineering*, 32(7):1411–1432, 2008.
40. J. N. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33–42, 1996.

41. H. H. Hoos and T. Stützle. Evaluating Las Vegas algorithms: Pitfalls and remedies. In G. F. Cooper and S. Moral, editors, *UAI '98: Proc. of the 14th Conf. on Uncertainty in Artificial Intelligence*, pages 238–245. Morgan Kaufmann, 1998.
42. F. Hutter, T. Bartz-Beielstein, H. Hoos, K. Leyton-Brown, and K. P. Murphy. Sequential model-based parameter optimisation: an experimental investigation of automated and interactive approaches empirical methods for the analysis of optimization algorithms. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 361–414. Springer, Berlin, Heidelberg, New York, 2010.
43. F. Hutter, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy. Time-bounded sequential parameter optimization. In *Proc. of LION-10*, 2010. To appear.
44. T. Jansen. On classifications of fitness functions. pages 371–385, 2001.
45. D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, pages 215–250. AMS, 2002.
46. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.
47. K. Knight, P. Langley, and P. R. Cohen. What makes a compelling empirical evaluation? *IEEE Intelligent Systems*, 11:10–14, 1996.
48. W. Konen, T. Zimmer, and T. Bartz-Beielstein. Optimized modelling of fill levels in stormwater tanks using CI-based parameter selection schemes (in german). *at-Automatisierungstechnik*, 57(3):155–166, 2009.
49. O. Kramer, B. Gloger, and A. Goebels. An experimental analysis of evolution strategies and particle swarm optimisers using design of experiments. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, GECCO '07, pages 674–681, New York, NY, USA, 2007. ACM.
50. C. W. G. Lasarczyk. *Genetische Programmierung einer algorithmischen Chemie.* PhD thesis, Technische Universität Dortmund, 2007.
51. C. W. G. Lasarczyk and W. Banzhaf. Total synthesis of algorithmic chemistries. In *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1635–1640, New York, NY, USA, 2005. ACM.
52. D. G. Mayo. *Error and the Growth of Experimental Knowledge.* The University of Chicago Press, Chicago IL, 1996.
53. D. G. Mayo and A. Spanos. Severe testing as a basic concept in a neyman–pearson philosophy of induction. *British Journal for the Philosophy of Science*, 57:323–357, 2006.
54. D. G. Mayo and A. Spanos. *Error and Inference.* Cambridge University Press, Cambridge, 2010.
55. C. C. McGeoch. Toward an experimental method for algorithm simulation. *INFORMS Journal on Computing*, 8(1):1–15, 1996.
56. J. Mehnen, T. Michelitsch, C. Lasarczyk, and T. Bartz-Beielstein. Multi-objective evolutionary design of mold temperature control using DACE for parameter optimization. *International Journal of Applied Electromagnetics and Mechanics*, 25(1–4):661–667, 2007.
57. O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 829–836, New York, NY, USA, 2011. ACM.
58. O. Mersmann, M. Preuss, and H. Trautmann. Benchmarking evolutionary algorithms: towards exploratory landscape analysis. In *Proceedings of the 11th international conference on Parallel problem solving from nature: Part I*, PPSN'10, pages 73–82, Berlin, Heidelberg, 2010. Springer-Verlag.
59. B. M. Moret and H. D. Shapiro. Algorithms and experiments: The new (and old) methodology. *Journal of Universal Computer Science*, 7(5):434–446, 2001.

60. V. Nannen. *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*. PhD thesis, Vrije Universiteit Amsterdam, 2009.
61. V. Nannen and A. E. Eiben. A Method for Parameter Calibration and Relevance Estimation in Evolutionary Algorithms. In M. Cattolico, editor, *Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006*, pages 183–190. ACM, 2006.
62. J. C. Nash. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation*. IOP, Bristol, UK, 2nd edition, 1990.
63. B. Naujoks, D. Quagliarella, and T. Bartz-Beielstein. Sequential parameter optimisation of evolutionary algorithms for airfoil design. In G. e. a. Winter, editor, *Proc. Design and Optimization: Methods and Applications, (ERCOFTAC'06)*, pages 231–235. University of Las Palmas de Gran Canaria, 2006.
64. J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society, A*, 231:289–337, 1933.
65. N. H. Pothmann. Kreuzungsminimierung für k-seitige Buchzeichnungen von Graphen mit Ameisenalgorithmen. Master's thesis, Universität Dortmund, Germany, 2007.
66. M. Preuss. Niching Prospects. In B. Filipic and J. Silc, editors, *Bioinspired Optimization Methods and their Applications (BIOMA 2006)*, pages 25–34. Jozef Stefan Institute, Ljubljana, Slovenia, 2006.
67. M. Preuss and T. Bartz-Beielstein. Sequential parameter optimization applied to self-adaptation for binary-coded evolutionary algorithms. In F. Lobo, C. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, Studies in Computational Intelligence, pages 91–120. Springer, 2007.
68. M. Preuss, G. Rudolph, and F. Tumakaka. Solving multimodal problems via multiobjective techniques with Application to phase equilibrium detection. In *Proceedings of the International Congress on Evolutionary Computation (CEC2007). Piscataway (NJ): IEEE Press. Im Druck*, 2007.
69. M. Preuss, G. Rudolph, and S. Wessing. Tuning optimization algorithms for real-world problems by means of surrogate modeling. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 401–408, New York, NY, USA, 2010. ACM.
70. M. Preuss, C. Stoean, and R. Stoean. Niching foundations: basin identification on fixed-property generated landscapes. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 837–844, New York, NY, USA, 2011. ACM.
71. R. L. Rardin and R. Uzsoy. Experimental evaluation of heuristic optimization algorithms: A tutorial. *J. Heuristics*, 7(3):261–304, 2001.
72. G. Rudolph, M. Preuss, and J. Quadflieg. Two-layered surrogate modeling for tuning optimization metaheuristics. Algorithm Engineering Report TR09-2-005, Faculty of Computer Science, Algorithm Engineering (Ls11), Technische Universität Dortmund, Germany, September 2009.
73. R. Salomon. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems*, 39:263–278, 1996.
74. S. K. Smit and A. E. Eiben. Comparing Parameter Tuning Methods for Evolutionary Algorithms. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 399–406, May 2009.
75. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore, May 2005. http://www.ntu.edu.sg/home/EPNSugan.

76. A. Törn, M. Ali, and S. Viitanen. Stochastic Global Optimization: Problem Classes and Solution Techniques. *Journal of Global Optimization*, 14(4):437–447, 1999.
77. M. Tosic. Evolutionäre Kreuzungsminimierung. Diploma thesis, University of Dortmund, Germany, January 2006.
78. H. Trautmann and J. Mehnen. Statistical methods for improving multi-objective evolutionary optimisation. *International Journal of Computational Intelligence Research (IJCIR)*, 5(2):72–78, 2009.
79. L. Volkert. Investigating ea based training of hmm using a sequential parameter optimization approach. In G. G. Yen et al., editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 2742–2749, Vancouver, BC, Canada, 16-21 July 2006. IEEE Press.
80. S. Wessing. Towards optimal parameterizations of the s-metric selection evolutionary multi-objective algorithms. Algorithm Engineering Report TR09-2-006, Universität Dortmund, Germany, September 2009.
81. S. Wessing, M. Preuss, and G. Rudolph. When parameter tuning actually is parameter control. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO '11, pages 821–828, New York, NY, USA, 2011. ACM.
82. Y. Yi. *Fuzzy Operator Trees for Modeling Utility Functions.* PhD thesis, Philipps-Universität Marburg, 2008.