Chapter 1
Sequential Parameter Optimization

T. Bartz-Beielstein

Abstract We provide a comprehensive, effective and very efficient methodology for the design and experimental analysis of algorithms. We rely on modern statistical techniques for tuning and understanding algorithms from an experimental perspective. Therefore, we make use of the sequential parameter optimization (SPO) method that has been successfully applied as a tuning procedure to numerous heuristics for practical and theoretical optimization problems. Two case studies, which illustrate the applicability of SPO to algorithm tuning and model selection, are presented.

1.1 Introduction

Contrary to intuition, applying and assessing randomized optimization algorithms is usually not trivial. Due to their flexibility, they fail only gradually as long as the main principle of moving towards better solutions remains intact. Statements of practitioners like "I tried that once, it did not work" thus appear to be somewhat naïve to scientists but simply reflect how the difficulties of a proper experimental evaluation are usually underestimated.

The approach presented in this paper has its origin in *design of experiments* (DOE). Ideas from Fisher (1935) have been refined and extended over the last decades. Ideas, which were successfully applied in agricultural and industrial simulation and optimizations, are applied to problems from computer science.

The concept of active experimentation is fundamental in our approach. First, a very simple model is chosen—a process related to pulling oneself up by one's own bootstraps. Since the model induces a design, design points can be chosen to perform experiments. In the next step, model refinement is applied (or, if the model does not fit at all, a new model is chosen). This rather simple sequential approach is a very powerful tool for the analysis of heuristics.

Dagstuhl Seminar Proceedings 09181
Sampling-based Optimization in the Presence of Uncertainty
http://drops.dagstuhl.de/opus/volltexte/2009/2115

Whereas algorithm engineering (Cattaneo and Italiano, 1999) builds on the existence of theory, this is not a necessary precondition for our approach. However, obtaining general principles from experimentation may later on lead to theory. Nevertheless, this is not the primary goal. Instead, one is striving for "best case" performance in the sense that for a given problem and algorithm, the best possible algorithm configuration is sought.

One instance of this sequential approach to experimentation is presented in this paper, namely, the *sequential parameter optimization* (SPO). An implementation of the SPO has been developed over the last years. The corresponding software package will be referred to as SPOT, an acronym which stands for *sequential parameter optimization toolbox*.

The SPO *toolbox* was developed over the last years by Thomas Bartz-Beielstein, Christian Lasarczyk, and Mike Preuss (Bartz-Beielstein et al., 2005b). The main purpose of SPOT is to determine improved parameter settings for optimization algorithms to analyze and understand their performance.

The development of SPO was inspired by Kleijnen (1987), who did similar investigations in the field of simulation. We applied *design of experiments* (DoE) to

a) stochastic search, e.g., evolution strategies and also to
b) deterministic search, e.g., Nelder-Mead simplex algorithm.

The first studies used classical DoE with (fractional) factorial designs (Beielstein, 2003). In addition to classical regression models, tree based regression models were used in order to handle categorical variables. Our experiments demonstrated that nearly every algorithm can be tuned. Figure 1.1 depicts results from DoE applied to evolution strategies and simulated annealing. In a second step of the SPO development, Kriging models were used to perform the optimization (Bartz-Beielstein et al., 2004b,c). Kriging requires different designs, so-called space filling designs, e.g., Latin hypercube designs. To improve the efficiency of SPO, we implemented a sequential procedure, which starts with a relatively small initial sample. Additional points were added during the optimization procedure. In addition, a procedure to tackle noise was integrated.

SPOT was successfully applied to numerous optimization algorithms, especially in the field of evolutionary computation, i.e., evolution strategies, particle swarm optimization, algorithmic chemistries etc. in the domains of machine engineering (Mehnen et al., 2005; Weinert et al., 2004; Mehnen et al., 2004), aerospace industry (Bartz-Beielstein and Naujoks, 2004), simulation and optimization (Bartz-Beielstein et al., 2005c; Markon et al., 2006), technical thermodynamics (Bartz-Beielstein et al., 2005b), bioinformatics (Volkert, 2006) and agri-environmental policy-switchings (de Vegt, 2005). Other fields of application are in fundamental research, e.g., graph drawing (Tosic, 2006), selection under uncertainty (optimal computational budget allocation) (Bartz-Beielstein et al., 2005a), evolution strategies (Bartz-Beielstein,

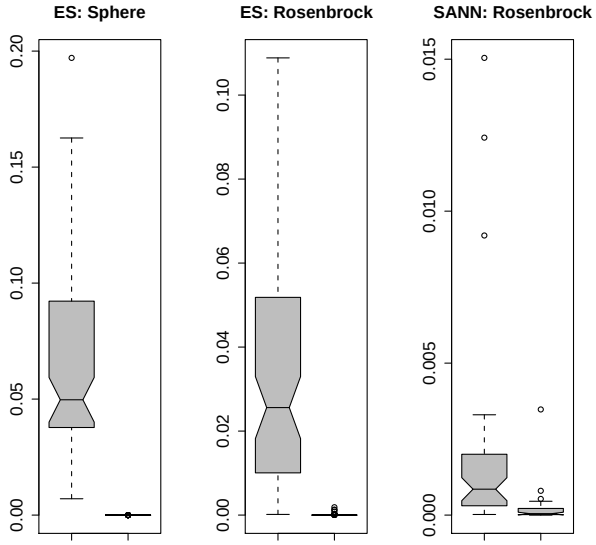

Fig. 1.1 Comparison of the results from 50 runs of an evolution strategy on the sphere function and Rosenbrock's function and of a simulated annealing. The left boxplot in each panel shows the result obtained with the default (standard) parameters recommended in the literature, whereas the right box plots illustrate results from the regression analysis. Problems in this studies are minimization problems, i.e., smaller values are better.

2005), genetic chromodynamics (Stoean et al., 2005), algorithmic chemistry (Bartz-Beielstein et al., 2005b), particle swarm optimization (Bartz-Beielstein et al., 2004a), and numerics (Bartz-Beielstein et al., 2004c). Further projects, e.g., the application of methods from computational intelligence to problems from storm water prediction are subject of current research.

Section 1.2 describes the role of experiments in computer science. The basic ideas from SPO are introduce in Sect. 1.3, and Sect. 1.4 gives a short description of the objectives of the SPO framework. SPOT's key elements of presented in Sect. 1.5. Two case studies are presented in Sect. 1.6 and 1.7. The paper concludes with a discussion of the results in Sect. 1.8.

1.2 Why Perform Experiments in Computer Science?

Theoreticians sometimes claim that experimentation is a "nice to have" feature, but not "necessary" in computer science. Besides reasons claimed by the new experimentalists, there are also practical problems which make experi-

mentation necessary. There are several good reasons to further develop the methodology for experimental work:

- For many practical problems, including but not limited to black box real-valued problems, theory is far behind to non-existent, so that there is no other means to investigate efficiency other than experiment.
- Metaheuristic optimization algorithms consist of basic working principles, any interfaced problem can be solved "in principle." But for attaining considerable efficiency on real-world problems, it is generally needed to adapt the metaheuristic towards the problem, using any available specific knowledge about the application domain. This process of adaptation is crucial but often ignored in favor of the final quality. It shall not be undertaken without a guiding methodology.

Summing up these considerations from theory and practice, the following topics can be considered as relevant for scientific research in experimentation:

- *Investigation* refers to the specification of the right optimization problem, the analysis of algorithms, the determination of important parameters, and the question, what "should be optimized." Important is also the development of reasonable research questions. Furthermore, one should specify, what is going to be explained. And, does it help in practice? Does this investigation enable theoretical advances? To describe the observed behaviour, *conjectures* should be formulated.
- *Comparing* the performance of algorithms is a challenging task. Any reasonable approach here has to regard fairness. It is good to demonstrate performance. However, explaining and understanding performance is much better (Cohen, 1995) . This means, that we should look at the behavior of the algorithms, not only at the results.
- *Quality* in the context of experimental analysis of optimization algorithms is related to robustness, which includes insensitivity to exogenous factors and the minimization of the variability (Montgomery, 2001). In order to find out, for what (problem, parameter, measure) spaces our results hold, invariance properties should be detected.

These elements are well-established in many scientific disciplines. We claim that in addition, tools to analyze the scientific meaning of statistical results are necessary.

1.3 Sequential parameter optimization

SPO comprehends the following steps:

1. Pose a scientific question.
2. Break the (complex) question into (simple) statistical hypotheses for testing. Mayo (1996, p. 190) writes:

> Our approach to experimental learning recommends proceeding in the way one ordinarily proceeds with a complex problem: break it into smaller pieces, some of which, at least, can be tackled. One is led to break things down if one wants to learn [...] Setting out all possible answers to this one question becomes manageable, and that is all that has to be "caught" by our not-H.

3. For each hypothesis:
   a) Select a model (e.g., regression trees) to describe a functional relationship
   b) Select an experimental design
   c) Generate data
   d) Refine the model until the hypothesis can be accepted/rejected
4. Draw conclusions from the hypotheses to assess the scientific meaning of the results from this experiment. Here, Mayo (1996)'s concept of severity comes into play. Mayo (1996, p. 178) writes: " 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.*"

1.4 Objectives

During the first stage of the experimentation, SPOT treats the algorithm A as a black box. A set of input variables, say $\mathbf{x}$, is passed to A. Each run of the algorithm produces some output, $\mathbf{y}$. SPOT tries to determine a functional relationship F between $\mathbf{x}$ and $\mathbf{y}$. Since experiments are run on computers, pseudo-random numbers are taken into consideration if

i) the underlying objective function f is stochastically disturbed, e.g., measurement errors or noise occur, and/or
ii) the algorithm A uses some stochastic elements, e.g., mutation in evolution strategies.

This situation can be described as follows:

$$\mathbf{u} \xrightarrow{f} \mathbf{v} \tag{1.1}$$

$$\mathbf{x} \xrightarrow{F} \mathbf{y} \tag{1.2}$$

In the following, we will classify elements of $\mathbf{x}$ in the following manner.

1. Variables, which are necessary for the algorithm, belong to the algorithm design, whereas
2. variables, which are needed to specify the optimization problem f, belong to the problem design.

The set $\mathbf{x}$ is divided into the algorithm design $\mathbf{x}_a$ and the problem design $\mathbf{x}_p$.

1.5 Elements of the SPOT framework

1.5.1 SPOT tasks

SPOT is not *per se* a meta-algorithm. We are also interested in the resulting algorithm designs, not in the solutions to the primordial problem only. SPOT provides tools to perform the following tasks

1. *Initialize.* An initial design which is written to the designfile is generated. This is usually the first step during experimentation. Information from the ROI and the CONF file are used. A clean start can be performed. This step is necessary if the user wants to perform a new experiment or delete results from previous run.
2. *Run.* This is usually the second step. The optimization algorithm is started with configurations from the design file. The algorithm writes results to the result file. Information from the design and algorithm problem design files are used in this step.
3. *Sequential step.* A new design, based on information from the resultfile, is generated. This new design is written to the design file. 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 a few algorithms runs are possible, user-specified prediction models can easily be integrated into SPOT.
4. *Report.* An analysis, based on information from the resultfile, is generated. Since the report uses information from the resultfile, 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 the initialization for a certain number of times.

1.5.2 Running SPOT

SPOT can be started from the command shell in the bin directory of your SPOT installation.[1] The formal command reads:

[1] SPOT can be downloaded from `http://www.gm.fh-koeln.de/campus/personen/lehrende/thomas.bartz-beielstein/00489/`. Previous versions of the SPOT relied on functions provided by the MATLAB Kriging toolbox DACE developed by Lophaven et al. (2002). Starting with version 0.5, an R (Ihaka and Gentleman, 1996) version will be available, too. The following description refers to this R implementation. Its installation is very simple: Extract the zip-file from the SPOT WWW-page in a folder (directory) of your choice. Surely, an R-system must be available on your computer. A JAVA based GUI is available, too.

```
spot <task> <configurationfile>
```

where `task` can be one of the tasks described in Sect. 1.5.1, i.e., init, seq, run, rep, or auto, and `configurationfile` is the name to the SPOT configuration file. SPOT uses simple text files as interfaces to the algorithm to the statistical tools.

1. Files provided by the user:
   a. *Region of interest* (ROI) files specify the region, over which the algorithm parameters are tuned. categorical variables such as the recombination operator in ES, are encoded as numerical values, e.g., "1" represents "no recombination" and "2" stands for "discrete recombination."
   b. *Algorithm design files* (APD) are used to specify parameters used by the algorithm, e.g., problem dimension, objective function, starting point, or initial seed.
   c. *Configuration* files (CONF) specify SPOT specific parameters, such as the prediction model or the initial design size.
2. Files generated by SPOT
   a. *Design* files (DES) specify algorithm designs. They are generated automatically by SPOT and will be read by the optimization algorithms.
   b. After the algorithm has been started with a parameterization from the algorithm design, the algorithm writes its results to the *result file* (RES). Result files provide the basis for many statistical evaluations/visualizations. They are read by SPOT to generate prediction models. Additional prediction models can easily be integrated into SPOT.

Figure 1.2 illustrates SPOT interfaces and the data flow. Note, that the problem design can be modified, too. This can be done to analyze the robustness (effectivity) of algorithms.

SPOT can be run in an automated and an interactive mode. Similarities and differences of the automated and the interactive process are shown in Fig. 1.3.

1.6 Case Study I: Algorithm Tuning

To introduce SPOT's functionality, we present a case study, which analyses a simple evolution strategy. Starting from scratch, we do not now anything about the functional relationship F from Equation 1.1. This situation can be characterized as a chicken-and-egg problem: The experimenter has to decide which comes first: a design for the data $\mathbf{x}$ or the specification of a model F?

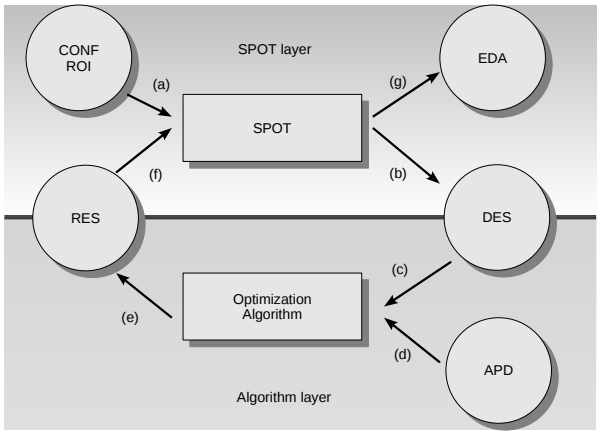

Fig. 1.2 SPO interfaces. The SPOT loop can be described as follows: Configuration and region-of-interest files are read by SPOT (a). SPOT generates a design file (b). The algorithm reads the design file and (c) extra information, e.g., about the problem dimension from the algorithm-problem design file (d). Output from the optimization algorithm are written to the result file (e). The result file is used by SPOT to build the prediction model (f). Data can be used by EDA tools to generate reports, statistics, visualizations, etc. (g)

For example, assuming a linear model F, we should use factorial designs to specify $\mathbf{x}$. But, in order to validate linearity, we need some data $\mathbf{x}$.

We prefer the following approach: Select a simple model F_0 and a related design $\mathbf{x}_0$, generate some data $\mathbf{y}_0$, fit the (simple) model F_0, predict a few new design points $\mathbf{x}_1$ based on F_0, generate further data $\mathbf{y}_1$, refine the model (F_1), and continue. The chicken-and-egg problem defines a sequential approach in a natural manner, which improves F and y at the same time. This approach motivated the term *sequential parameter optimization*. In many situations, it can be shown, that sequential approaches are much more efficient than one-at-a-time approaches Armitage (1975).

We will discuss the basic output from the R analysis during the sequential approach. The analysis is not complicated and requires only basic knowledge from statistics. The reader will get interesting insights into the working mechanism of his algorithms by executing a few lines of R code. SPOT comes with some elementary R scripts which can be extended easily.

First, we will focus on fitting a linear regression model. The model

$$y = \beta_0 + \sum_{j=1}^{k} \beta_j x_j + \varepsilon \tag{1.3}$$

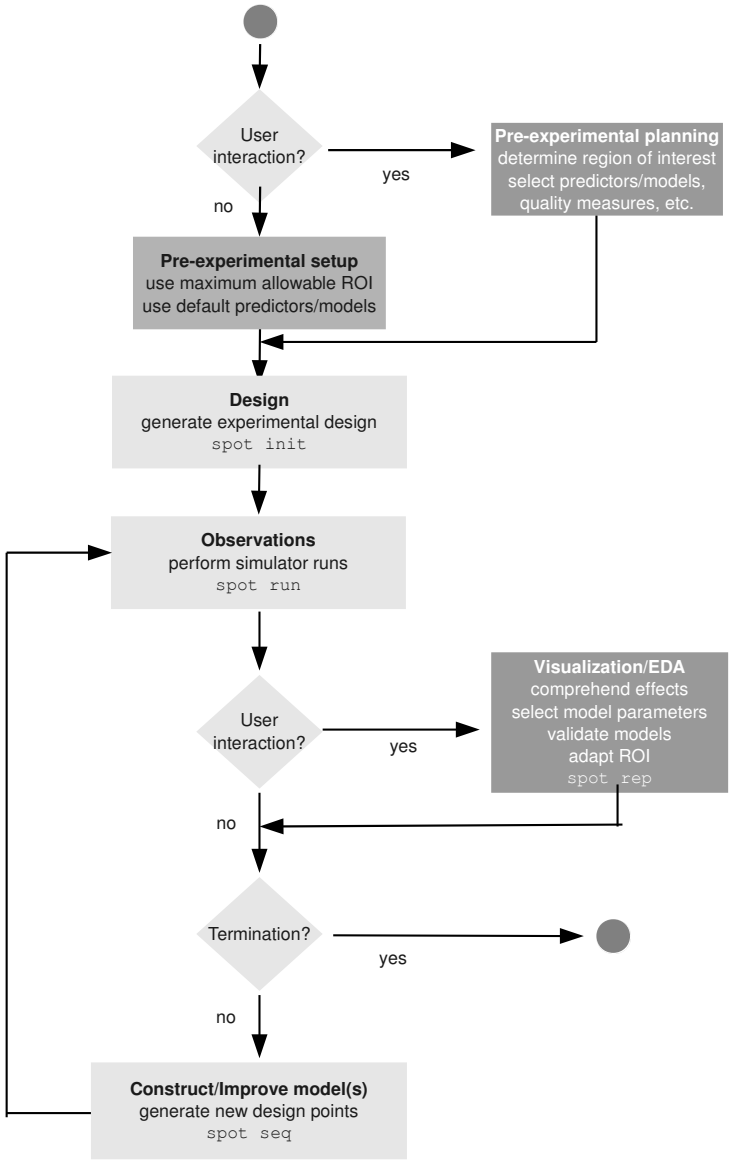

Fig. 1.3 The sequential parameter optimization process. *White font color* was chosen to mark elements that are used in the interactive process only. A *typewriter font* was chosen for the corresponding SPOT commands. To start the automated mode, simply use the task `auto`.

Procedure 1: (1+1)-ES()

```
1  t := 0;
2  initialize(x, σ);
3  repeat
4  |  y_p := f(x_p);
5  |  x_o := x_p + σ(N(0,1), N(0,1), ..., N(0,1))^T;
6  |  y_o := f(x_o);
7  |  if y_o ≤ y_p then
8  |  |  x_p := x_o;
9  |  modify σ according to 1/5th rule;
10 |  t := t + 1;
11 until TerminationCriterion() ;
12 return (x_p, y_p)
```

is called a multiple linear regression model with k *regression variables* and *response* y. The regression variables represent algorithm parameters, e.g., initial step size s_0, multiplier for step sizes a, and size of the memory$g$.

We consider a simple *evolution strategy* (ES), the so-called (1+1)-ES, see Procedure 1. The 1/5th rule states that σ should be modified according to the rule

$$\sigma(t+1) := \begin{cases} \sigma(t)a, & \text{if } P_s > 1/5 \\ \sigma(t)/a, & \text{if } P_s < 1/5 \\ \sigma(t), & \text{if } P_s = 1/5 \end{cases} \tag{1.4}$$

where the factor a is usually between 1.1 and 1.5 and P_s denotes the success rate (Beyer, 2001). The factor a depends particularly on the measurement period g, which is used to estimate the success rate P_s. During the measurement period, g remains constant. For $g = n$, where n denotes the problem dimension, Schwefel (1995) calculated $1/a \approx 0.817$. Beyer (2001) states that the "choice of a is relatively uncritical" and that the 1/5th rule has a "remarkable validity domain." He also mentions limits of this rule.

Based on these theoretical results, we can derive certain scientific hypotheses. One might be formulated as follows: *Given a spherical fitness landscape, the (1+1)-ES performs optimal, if the step-sizes σ is modified according to the 1/5th rule as stated in Eq. (1.4).* This statement is related to the primary model.

In the experimental model, we relate primary questions or statements to questions about a particular type of experiment. At this level, we define an objective function, a starting point, a quality measure, and parameters used by the algorithm. These parameters are summarized in Table 1.1.

Data from these experiments are related to an experimental question, which can be stated as follows: *Determine a value for the factor a, such that the (1+1)-ES performs best with respect to the quality measure specified in Table 1.1. And, is this result independent from the other parameters pre-*

Table 1.1 (1+1)-ES parameters. The first three parameters belong to the algorithm design, whereas the remaining parameters are from the problem design (see Sect. 1.4).

Name	Symbol	Factor name in the algorithm design
Initial stepsize	$\sigma(0)$	S0
Stepsize multiplier	a	A
History	$g = n$	G
Starting point	$\mathbf{x_p}$	
Problem dimension	n	
Objective function	$f(\mathbf{x}) = \sum x_i^2$	
Quality measure	Expected performance, e.g., $E(y)$	
Initial seed	s	
Budget	$t_{\max}$	

sented in Table 1.1? Note, that Table 1.1 contains all factors that are used in this experiment.

Finally, we consider the data model. A common practice in statistics is to seek data models that use similar features and quantities of the primary hypothesis. For example, if the primary model includes questions related to the mean value μ of some random variable X, then the data model might use the sample mean $\bar{x}$. Here, we are interested in the expected performance of the (1+1)-ES. Thus, we can use the average from, say ten, repeats, $\overline{y_i}$, to estimate the expected performance. In addition, standard errors or confidence intervals can be reported. So, in the data model, raw data are put into a canonical form to apply analytical methods and to perform hypothesis tests.

1.6.1 Performing the Experiments

Classical *response surface methods* (RSM) use three steps: screening, modeling, and optimization (Kleijnen, 1987; Montgomery, 2001). Design complexity is increased during the RSM process, so complex models can be fitted in the last phase of the RSM process. SPOT is closely related to RSM. Because SPOT includes rules to analyze the scientific relevance (severity) of results from the statistical analysis and specifies rules for learning from error, it can be seen as an extension of the classical RSM framework.

1.6.1.1 Pre-experimental planning

This section describes, how basic ideas from regression analysis can be applied to analyse evolution strategies. SPOT allows the specification of

a) upper and lower limits of the region of interest (experimental region)
b) the number of repeats for the initial design, and
c) the type of design to be generated, e.g., LHD

Table 1.2 Regions of interest used in the pre-experimental planning phase. S0 and A denote real valued variables, whereas G is an integer value.

Factor	low	high	type
S0	0.1	5	FLOAT
A	1	2	FLOAT
G	1	100	INT

We have chosen a Latin hypercube design for the first experiments, see Tab. 1.3. This design comprehends three factors with different types. Although factorial designs are recommended for DOE, a space filling design was chosen for the pre-experimental planning phase. This design was chosen, because we expect interesting model features in the center of the experimental region. And, we are interested in using other models, e.g., tree-based regression models, or Kriging, in parallel. These models require space filling designs. So, choosing an LHD can be seen as a good compromise, especially, if we do not know whether linear regression models are adequate.

A scatter plot was used to analyse results from the first design. It is based on ten design points, where each design point represents one algorithm configuration. Each run was performed once, see Fig. 1.4. The algorithm performs quite well with parameters chaosen from these ROI.

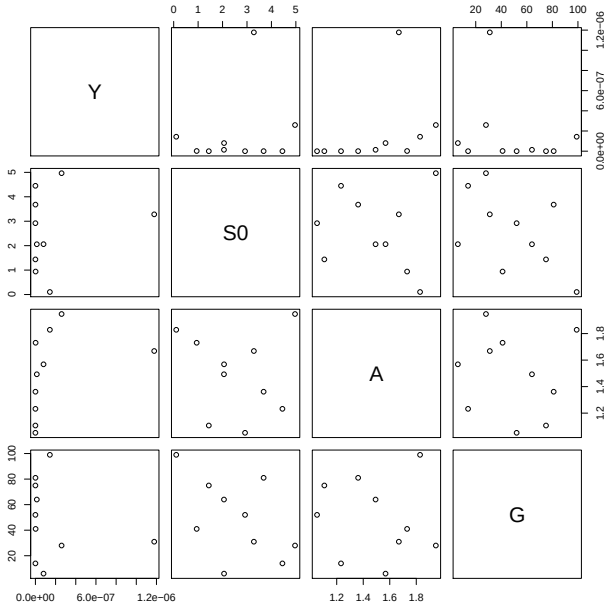

Fig. 1.4 Scatter plot from the pre-experimental planning phase to determine the region of interest.

1.6.1.2 Performing the first regression analysis

To regress fitness, y, on the these quantitative predictors, we obtain the output in R as shown in Fig. 1.5.

```
Call:
lm(formula = log(Y) ~ log(S0) + log(A) + log(G), data = df0002)
Residuals:
    Min      1Q  Median      3Q     Max
-3.6093 -1.7175  0.7059  1.4426  3.0768
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -28.0456     5.6820  -4.936  0.00261 **
log(S0)       0.4115     0.9836   0.418  0.69022
log(A)       27.2698     4.8056   5.675  0.00129 **
log(G)       -0.9124     1.2219  -0.747  0.48346
---
Residual standard error: 2.791 on 6 degrees of freedom
Multiple R-squared: 0.8671,Adjusted R-squared: 0.8006
F-statistic: 13.05 on 3 and 6 DF,  p-value: 0.004872
```

Fig. 1.5 Output from the first regression model in R.

The R output gives a numerical summary of the residuals and a table of the regression parameters. Here, we see that the intercept of the fitted line is $b_0 = -28.0456$ with $\text{se}(b_0) = 5.6820$, and the estimated regression coefficient b_1 is 0.4115 with $\text{se}(b_1) = 0.9836$. The multiplier A has the largest effect, because its value reads 27.2698. The memory G has only minor impact on the performance of the ES. The table reports also t-statistics and p-values for individual tests of the hypotheses that the true intercept is zero, and that the true slope is zero. The `Residual standard error` is an expression of the variation of the observations around the fitted line. It can be used to estimate σ. `Multiple R-Squared` and `Adjusted R-Squared` are reported as well. The values related to the `F-statistic` describe an F-test for the hypothesis that the regression coefficients are is zero.

In addition, an ANOVA table can be generated. Based on an ANOVA table we can compare two models:

a) Model M_1, which includes S0, A, and G and
b) Model M_2, which includes the parameter with the highest significance, namely A

As expected, it indicates that there is no significant improvement of the model once S0 and G are included. Removing the parameters S0 and G from the model is recommended. R has a build-in function which adds parameters one at a time to the current model. The `add1` function adds parameters one after one from a list and shows the resulting statistics. The default output table reports the *Akaike Information Criterion* (AIC), defined as minus twice log

likelihood plus $2p$ where p is the rank of the model (the number of effective parameters). For performing model searches by AIC, R has the `stepAIC` function. Since selection and adjustment of information criteria is a difficult task (and beyond the scope of this introduction), we simply show the output from `stepAIC` applied to our example.

```
Call:
lm(formula = Y ~ A, data = df0002normlogy)
Residuals:
    Min      1Q  Median      3Q     Max
-3.7071 -2.6383  0.2352  2.4411  3.8783
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -20.6070     0.9672 -21.305 2.48e-08 ***
A             8.1253     1.4940   5.439 0.000617 ***
---
Residual standard error: 3.059 on 8 degrees of freedom
Multiple R-squared: 0.7871,Adjusted R-squared: 0.7605
F-statistic: 29.58 on 1 and 8 DF,  p-value: 0.000617
```

Fig. 1.6 Output from the model based on R's stepAIC function.

The final model, suggested by `stepAIC`, includes the parameter A only.

Residual plots can be used to check model assumptions. Common checks comprehend

a) a plot of residuals versus fitted values and
b) normal probability plots of residuals.

In addition, visual inspections, e.g., on the basis of added-variable plots (Fig. 1.7) can be performed. Added-variable plots focus on one variable at a time and take into account the influence of the other predictors (Draper and Smith, 1998; Fox, 2002). To determine the influence of predictor x_j on the response y, added-variable plots are generated as follows. Let x_{-i} denote the set of regressors

$$\{x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}.$$

1. Regress y on x_{-j} , obtaining residuals $e(j)$.
2. Regress x_j on predictors x_{-j} , obtaining residuals $x(j)$
3. Plot $e(j)$ versus $x(j)$.

All ES-runs were performed using the same number of function evaluations. The results clearly indicate that A should be reduced. Smaller values improve the algorithms performance. The linear model was able to detect this trend. This was the first step of the analysis. The experimental setup shown in Tab. 1.3 includes a systematic variation of three parameters.

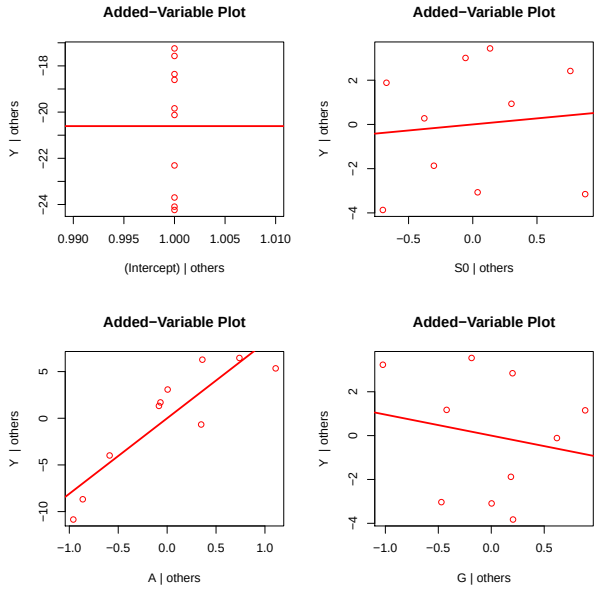

Fig. 1.7 Added-variable plot based on 10 runs from the initial design of ES on the 10 dimensional sphere function. Smaller y values represent better solutions.

Summarizing results from the regression model analysis

The regression models used in this study illustrate that A should be decreased. Step-wise regression results in the simplified model

$$\hat{y} = -20.61 + 8.12a,$$

which indicates that the multiplier for the step sizes should be decreased. A similar result was obtained from a tree-based regression. The result occurs independently from the chosen model.

1.6.1.3 Steepest descent

Results from the regression analysis provide information about the steepest descent in a natural manner. We proceed with the steepest descent based on the model `Y ~ S0 + A + G`. The procedure of steepest descent is performed as follows: Starting from the design center, we move sequentially in the direction of the maximum decrease in the response. This direction is parallel to the normal of the fitted response surface. Usually in RSM, the path of the steepest descent is taken as the line through the center of the ROI and normal to the fitted surface, i.e., the steps are proportional to the β_i's (regression coeffi-

cients). In general, the following procedure for determining the coordinates of the points on the path of the steepest descent can be applied (Montgomery, 2001):

1. Select the variable we know most about or the variable that has the largest absolute regression coefficient $|b_i|$.
2. Determine the stepsize in the other variables as

$$\Delta x_i = \frac{b_i}{b_j}\Delta x_j \qquad i = 1,2,\ldots,k; \qquad i \neq j$$

3. Convert the Δx_i stepwidths from the coded to the δx_i in the natural variables.

As the largest stepwidth we recommend the value that leads to the border of the ROI. The resulting values for the steepest descnt are shown in Table **??**. Note, that the steepest descent experiments do not have to be performed in sequence. For example, run 10 can be performed before run 2 or a simple interval search can be performed. We recommend generating a graph of these results to determine the a new region of interest following the direction of the steepest descent, cf. Fig. 1.8. The region of interest is moved down the response surface.

Table 1.3 Steepest descent.

	S0	A	G
1	2.54	1.50	52.50
2	2.52	1.45	53.05
3	2.50	1.41	53.60
4	2.49	1.37	54.15
5	2.47	1.32	54.70
6	2.45	1.28	55.25
7	2.44	1.23	55.80
8	2.42	1.19	56.35
9	2.40	1.14	56.90
10	2.39	1.10	57.45
11	2.37	1.05	58.01

These settings are used for additional runs of ES. Figure 1.8 plots the yield at each step along the path of the steepest descent.

Based on visual inspection of the yields in Fig. 1.8, the new central point was determined to be the tenth point of the steepest descent. This new central point leaves some space for variation of the A values, say in the interval $[1.01, 1.2]$. Based on the best value obtained with the steepest descent, we build a new model with center point:

$$\mathbf{x}_c = [S0, A, G] = [2.5, 1.125, 40].$$

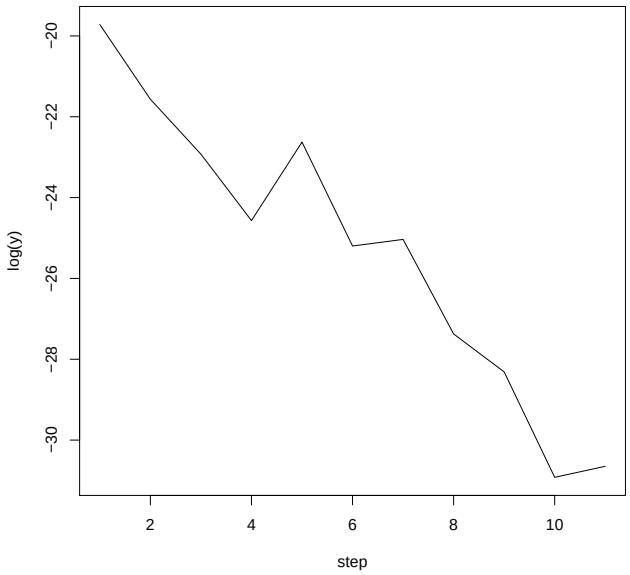

Fig. 1.8 *Left:* Function values $f(x)$ (Sphere) versus steps along the path of the steepest descent. Indices denote the ten steps from the steepest descent. Note, that these values are based on one repeat only, so variation in the data, e.g., the peak (index 5) are not surprising.

The specification of the new region of interest requires user knowledge. The new center point was determined by interpreting graphical results which were based on the steepest descent. Next, we have to determine a new region around $\mathbf{x}_c$. Sometimes, especially when a classical factorial design is used during the first step, it can useful to increase the region of interest at this stage. However, we have chosen a space-filling design for the first step. Therefore, we have to decrease the region of interest. As a rule of thumb, which is to be reconsidered in any situation, we use at least $\pm$ 1/5th of the values at the new central point. For example, if the value of the new initial step size S0 is 5, we define a new region of interest for this values as the interval $[4,6]$. Here, the new region of interest reads as follows:

$$S0 \in [2,3], A \in [1.05,2], G \in [20,80].$$

Latin hypercube sampling with 100 points and three repeats each is used for a graphical exploration of the ROI. Figure 1.9 displays a fit of the response surface which is based on the complete data set (320 runs of the target algorithm). A local regression model based on R's `loess()` function is fitted to the data.

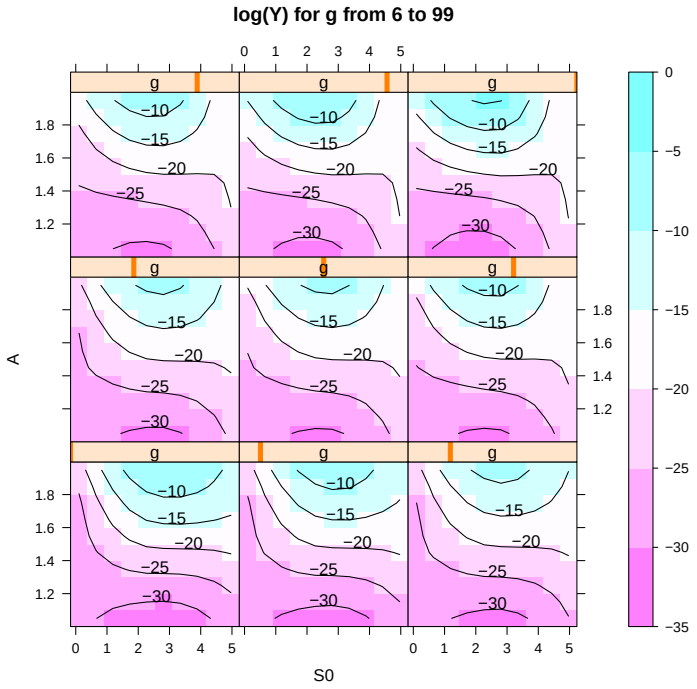

Fig. 1.9 Contour plot based on the complete data set (ES-Sphere with 320 function evaluations). Smaller values are better. Better configurations are placed in the lower left corner of the panels. A is plotted versus S0, while values of the factor with the last but one effect, namely G, are varied with the slider on top of each panel.

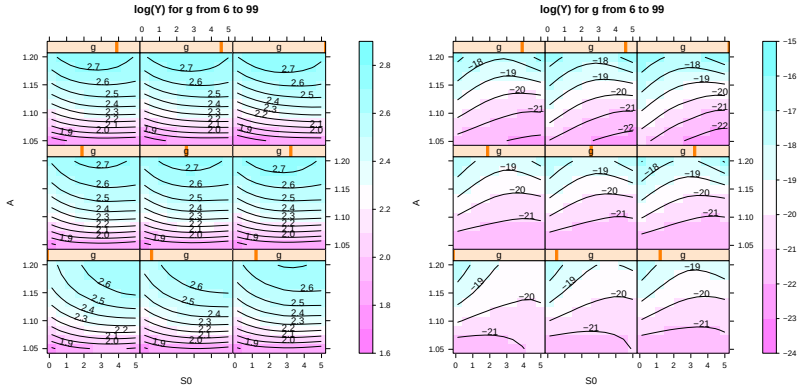

Fig. 1.10 Contour plot based on the complete data set (ES-Sphere with 320 function evaluations). Smaller values are better. Better configurations are placed in the lower left corner of the panels. A is plotted versus S0, while values of the factor with the last but one effect, namely G, are varied with the slider on top of each panel. Left: problem dimension = 100, right: starting point (100,100,..)

Summarizing results from the RSM

Results from this case study support results presented in Beyer (2001). The factor a should be chosen in the range from 1.1 to 1.5. Further experiments show that this result is independent from the starting point and problem dimension as illustrated in Fig. 1.9 and Fig. 1.10.

1.6.2 Additional model considerations

The analysis of optimization algorithms requires the investigation of categorical variables, e.g., in evolution strategies, several variants of the recombination operator can be used (Beyer and Schwefel, 2002). SPOT allows the coding of non-numerical variables as factors. *Dummy regressors* or *contrasts* can be used to represent levels of a factor. SPOT allows can handle the following variables:

1. real values,
2. integers and
3. categorical variables.

SPOT's ROI file is used to specify this typing. Maindonald and Braun (2003) illustrate the treatment of dummy variables for classical regression in R.

Classical regression is only one technique that can be used in the SPOT framework to predict interesting design points. Alternatively, tree-based models can be used to cope with categorical variables, see also Fig. 1.11. Tree-based methods can be used for regression as well as for classification (Breiman et al., 1984). Maindonald and Braun (2003) recommend to use tree-based methods in tandem with parametric approaches, because tree-based regression may suggest interaction terms that ought to appear in a parametric model. However, tree-based methods require more data than classical regression techniques. That is, if only a few runs of the algorithms are possible, it may be necessary to use parametric models. On the other hand, tree-based methods may be helpful to explore new data sets very quickly. The experimenter gets on overview which variables have the greatest effects on the algorithm's performance.

We have mentioned only two prediction models, parametrized regression and regression trees. Further models, e.g., local regression or Kriging are available.

Example 1. To analyse the choice of the prediction model on the prediction quality and SPOT's ability to improve optimization algorithms (tuning), we performed the following study: Results from two different SPOT runs are compared. The first run uses a tree-based regression model, whereas in the second run the dummy-variable regression model was used. Both models used exactly the same setting, i.e., five SPOT iterations, initial design

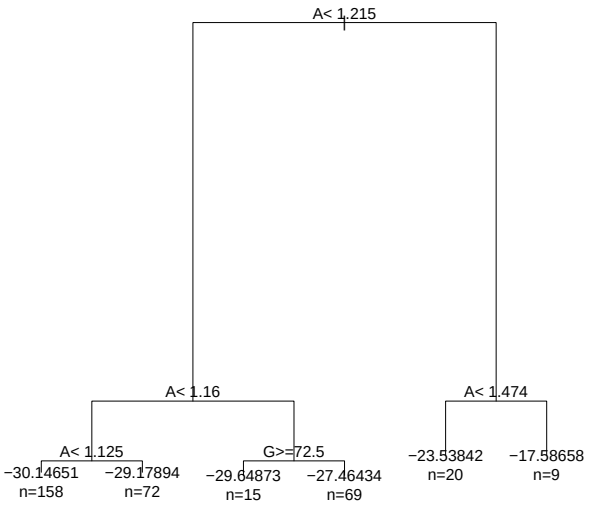

Fig. 1.11 Tree-based regression. The same data as in Fig. 1.10 were used. The factor A has the largest effect. $\log(y)$ values were used to grow the tree.

size of 50 points and 2 repeats. An ES, which optimizes the 10-dimensional sphere function with 1000 function evaluations was analyzed. Distributions of the results from these two SPOT runs are compared to the distribution of the randomly generated initial design, see Fig. 1.12. Both models were able to find better results than the randomized design. These results are in line with observations from other studies: Tree-based models can be applied very easily to unknown explanatory variables. They can cope with categorical and numerical data. Parametrized models perform better than tree-based models, however, the costs for modeling are higher. □

1.6.3 The Sequential Approach

We have discussed the initial setup for the SPO loop (initial design) and the analysis from one step. SPOT can be proceeded as follows: Based on the prediction model, e.g., linear regression or tree-based regression, interesting algorithm design points are generated. These design points are evaluated, i.e., the algorithm is run with the corresponding parameters. Then, an analysis as described in Sect. 1.6.1.2 can be performed. This analysis provides an

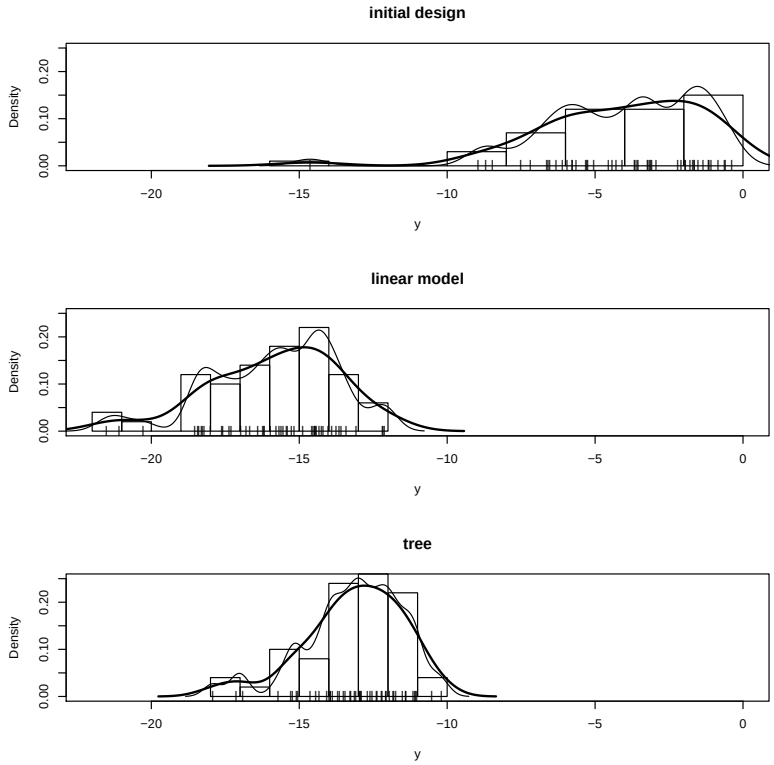

Fig. 1.12 Comparison of two SPOT prediction models with randomly generated configurations (from *top* to *bottom*). First, the results from randomly generated algorithm designs are shown. We have chosen the initial design (LHD), which is generated randomly. Next, results from 50 repeats of the best algorithm design determined with the linear and the tree-based regression model are shown.

improved predictor, which can be used to propose new design points, and so forth. As depicted in Fig. 1.3, this procedure can be performed in an automated manner. Result from the automated approach reads:

$$s_0 = 4.99, a = 1.10, g = 71.$$

Obviously, the result from the automated approach supports the findings from the manual approach, i.e., similar values for the parametrization of the (1+1)-ES are determined.

1.7 Case Study II: Prediction of Fill Levels in Stormwater Tanks

1.7.1 Problem

In this section we consider a case study from real-world optimization. Many real-world problems are related to prediction. Since the problem considered in this section is a new problem, no reference solutions were available. Therefore, a data-driven modeling approach was chosen.

We try to find answers for the following two questions:

- How to chose an adequate method or prediction model?
- And, how to tune the chosen prediction model?

Before we try to find answers to these problems, we will take a look at the problem first.

The problem is devoted to the task of predicting fill levels in stormwater tanks, see Fig. 1.13 The predictions are based on rain measurements and soil

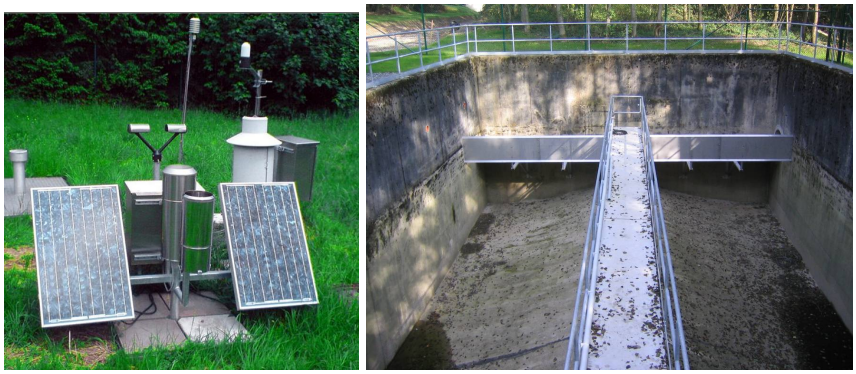

Fig. 1.13 *Left:* Rain gauge (pluviometer). *Right:* Stormwater tank.

conditions. We used 150,000 data sets that were noisy and included many infeasible entries. The goal of the modeling was to minimize the prediction error for a time period of 108 days. The root mean squared error (RMSE) was chosen as objective function, because it enables comparisons to results from industry. Typical results are shown in Fig. 1.15. Klein (2002) discuss important aspects related to the selection of the objective function.

A typical problem which occurs during these predictions can be characterized as follows: Standard and CI-based modeling methods show larger prediction errors when trained on rain data with strong intermittent and bursting behaviour as shown in Fig. 1.15.

Fig. 1.14 Rain gauging station which collected the 150,000 data points.

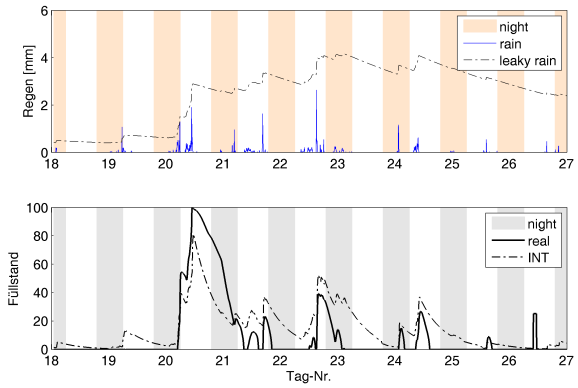

Fig. 1.15 Top: Rain measurements. Based on these data, the filling levels were predicted. A comparison of the predicted values and the real values in shown in the second figure.

1.7.2 Models under comparison

We compared six different modeling approaches (many more approaches are available). *Neural Networks* (NN) are well suited to model complex and unknown relationships between in- and output values Bartz-Beielstein et al. (2007). The NN learns a functional relationship between in- and output values. *Echo State Networks* (ESN) are recurrent NNs, which use a reservoir of recurrent neurons in order to generate dynamic signals. NN and ESN were investigated, but not considered in our final comparisons, because they were outperformed by the other methods. Their poor behaviour is caused by the following error, shown in Fig. 1.16. *Finite Impulse Response filter* (FIR-Filter), are standard tools from signal processing. In time series modeling, a nonlinear autoregressive exogenous model (NARX) Siegelmann et al. (1997)

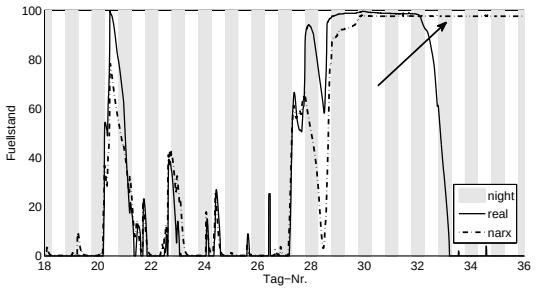

Fig. 1.16 NARX modeling error. The arrow shows the region, where the model stops oscillating. It predicts constantly filling levels larger than 90% This error appears systematically in each NARX simulation.

is a nonlinear autoregressive model which has exogenous inputs. This means that the model relates the present value of the time series to both

- past values of the same series; and
- present and past values of the driving (exogenous) series.

Finite Impulse Response filter (FIR) were tested during the pre-experimental planning phase of this project. Although relatively simple, they produced promising results. *Ordinary Differential Equations* (ODE) can be characterized as classical, state of the art methods. One disadvantage of the ODEs used in our study is that it relies on an exponential decay of the filling levels. This behaviour is not in correspondence with the real behaviour of the system, because different soil types produce variations in the delay of the water flow. Therefore, the ODE model was transformed in a more flexible model which is based on *Integral equations* (INT).

Each method requires the specification of several parameters (here: 2 – 13), before it can be run. NARX use two parameters (neurons and delay states), FIR five parameters (evaporation, delay, scaling, decay, length), ODE six parameters, and INT 13 parameters. Parameters for the ODE and INT models are listed in Tab. 1.4. The reader is referred to Konen et al. (2009) and Bartz-Beielstein et al. (2008) for further details.

1.7.3 Simulation Model Selection Based on SPOT

The SPO procedure used in this study uses the following three steps, which are related to the standard procedures from DoE, especially *response surface methods* (RSM), see, e.g., Montgomery (2001):

I. Pre-experimental planning
II. Screening

Table 1.4 Factors of the INT-Model. The ODE-Model uses a subset of 6 factors (shaded light gray): $\alpha, \beta, \tau_{\text{rain}}, \Delta, \alpha_L, \beta_L$.

Parameter	Symbol	manuell	Best SPOT	SPOT region
Abklingkonstante Füllstand (Filter g)	α	0.0054	0.00845722	[0,0.02]
Abklingkonstante Filter h	α_H	0.0135	0.309797	{0 ... 1}
Abklingkonstante 'leaky rain'	α_L	0.0015	0.000883692	{0 ... 0.0022}
Einkopplung Regen in Füllstand	β	7.0	6.33486	{0 ... 10}
Einkopplung Regen in 'leaky rain'	β_L	0.375	0.638762	{0 ... 2}
Einkopplung K-Term in Füllstand	h_0	0.5	6.87478	{0 ... 10}
Schwelle für 'leaky rain'	Δ	2.2	7.46989	{0 ... 10}
Flankensteilheit aller Filter	κ	1	1.17136	{0 ... 200}
Zeitverzögerung Füllstand zu Regen	τ_{rain}	12	3.82426	{0 ... 20}
Startzeitpunkt Filter h	τ_{in3}	0	0.618184	{0 ... 5}
Endzeitpunkt Filter h	τ_{out3}	80	54.0925	{0 ... 500}
Endzeitpunkt Filter g	τ_{out}	80	323.975	{0 ... 500}
RMSE		12.723	9.48588	

III. Modeling and optimization

We will discuss these steps in the following.

1.7.3.1 Step I: Pre-experimental planning

During the pre-experimental planning phase, no planning based on existing data can be done, only test runs can be performed. No optimality conditions applicable. The aim of our experiments during this phase is to determine region of interest intervals, i.e., the experimental region. As a rule of thumb, intervals should courageously be chosen. And, we have to define some mechanism for the treatment of infeasible factor settings, e.g., a penalty function.

1.7.3.2 Step II: Screening

During the screening phase, experiments with short running times were performed. Extreme values of the experimental region were considered. One goal was the detection of outliers which destroy the SPOT meta-model.

Unbalanced factor effects indicate not correctly specified ROIs, see Fig. 1.17.

The situation before the ROI were adjusted is shown on the left in Fig. 1.17. The right panel in Fig. 1.17 illustrates the situation after the ROI was modified.

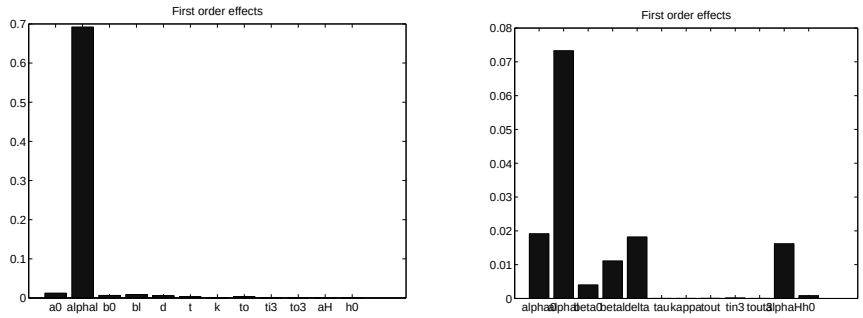

Fig. 1.17 *Left:* Unbalanced factor effects may indicate wrongly specified ROI. Varying factor α_l results in large changes in the output, whereas the other factors have nearly no influence at all. *Right:* Same situation but with modified ROI.

1.7.3.3 Step III: Modeling and Optimization

As a consequence of the screening phase, a reduced design can be used during the third phase (optimization), e.g., the number of parameters of the INT model could be reduced from 13 to 6. Reduced designs enable the application of more complex experimental models, e.g., quadratic instead of linear models. Results from this phase are shown in Fig. 1.18.

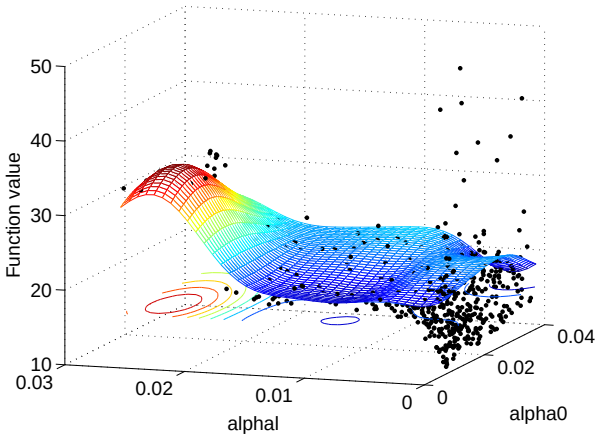

Fig. 1.18 Results from the optimization phase.

Table 1.5 Comparison. RSME

Method	randomized design	manually chosen	SPOT
FIR	25.42	25.57	20.10
NARX	85.22	75.80	38.15
ODE	39.25	13.60	9.99
INT	31.75	12.72	9.49

1.8 Results and Discussion

The topics discussed in this paper can be summarized as follows. We presented tools to improve our understanding how algorithms work. These tools can be used to improve algorithm's performance and to tailor algorithms to specific problem instances.

A systematic comparison of different prediction methods was presented in this case-study. We applied SPOT to determine the best parameters for each method. Our analysis was based on a practical optimization problem with predictions based on rain data with strong intermittent and bursting behavior. These are hard problems for standard and CI-based modeling methods. Our analysis reveals that models developed specific to the problem show a smaller prediction error than general models.

SPOT is applicable to diverse forecasting methods and automates the time-consuming parameter tuning process. The best manual result found by an expert could be improved with SPOT by 30%. SPOT was used to analyze parameter influence and allows simplification and/or refinement of the model design.

Results found by the experts were compared to results from SPOT. These comparisons were discussed with the experts, who were able to learn and get new insights into the model's behaviour. And, these discussion are helpful for the model validation—results found by SPOT could be checked by the experts, e.g., for constraint violations that were not specified in the initial model. No bias, and no systematic error could be detected by the experts, therefore we can claim that SPOT was able to determine improved parameter setting that can be used in practice..

An important question that is subject of our current research is related to the initial design size. First results are shown in Fig. 1.20.

It is an open question which models are preferable. For example, it is unclear when classical linear regression models or stochastic process models should be used. The choice of the model influences the selection of an appropriate (in some sense even optimal) design, e.g., classical factorial vs. space filling designs.

Lasarczyk (2007) implemented enhanced noise handling techniques such as *Optimal Computational Budget Allocation* Chen et al. (2003) to improve SPOT's efficiency for stochastic search and optimization algorithms. In gen-

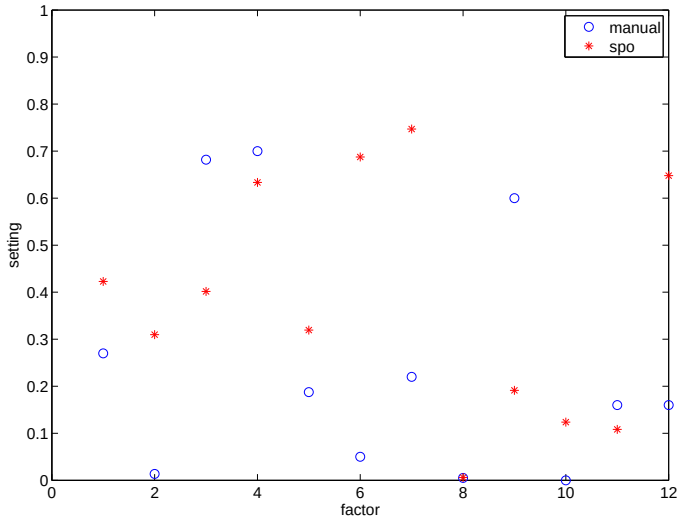

Fig. 1.19 A graph which was used in the validation process. It shows the best values from the experts in comparison to best values from the SPOT runs for 12 parameters.

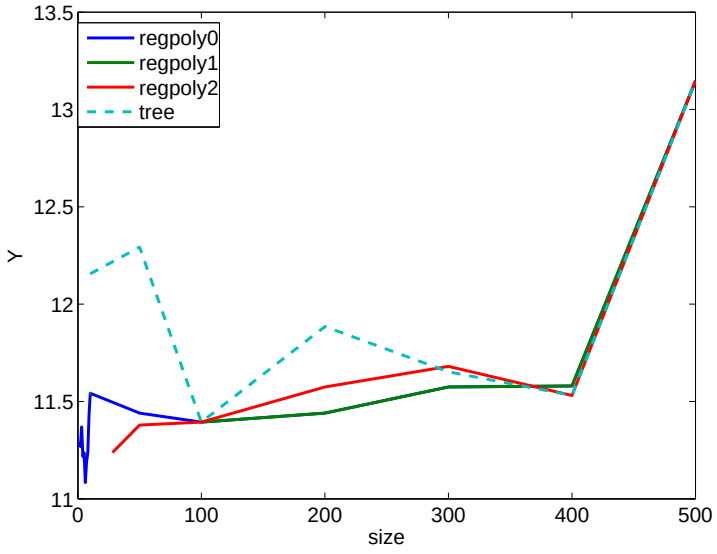

Fig. 1.20 Results plotted versus initial design sizes. Smaller Y values are better. Lines represent different prediction models. The best result was obtained with a small initial design size and the `regpoly0` model, where worst results were obtained when no sequential step was performed.

eral, further statistical tools, especially to bridge the gap between statistical and scientific significance should be developed. Surely, these methods should be standardized to improve comparability.

We are planning to provide SPOT interfaces for important optimization algorithms. The integration of the CMAES (Hansen, 1998), PSO, evolution strategies is a first step into this direction. SPOT's simple and open specification enables a quick integration of additional search algorithms.

SPO is a methodology — more than just an optimization algorithm (Bartz-Beielstein, 2008). Please check `http://www.gm.fh-koeln.de/~bartz` for updates, software, etc.

Acknowledgements

This work was supported by the research center COSA at Cologne University of Applied Sciences. I would like to thank M. Bongards and T. Hilmer for providing data and photos. C. Claes developed the FIR-model during my lectures on the *Computational Intelligence* (Bartz-Beielstein et al., 2007).

References

Armitage, P. (1975). *Sequential medical trials.* Blackwell, Oxford, U.K., 2nd edition.

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

Bartz-Beielstein, T. (2008). How experimental algorithmics can benefit from Mayo's extensions to Neyman-Pearson theory of testing. *Synthese*, 163(3):385–396. DOI 10.1007/s11229-007-9297-z.

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

Bartz-Beielstein, T., Bongards, M., Claes, C., Konen, W., and Westenberger, H. (2007). Datenanalyse und prozessoptimierung für kanalnetze und kläranlagen mit CI-methoden. In Mikut, R. and Reischl, M., editors, *Proc. 17th Workshop Computational Intelligence*, pages 132–138. Universitätsverlag, Karlsruhe.

Bartz-Beielstein, T., de Vegt, M., Parsopoulos, K. E., and Vrahatis, M. N. (2004a). Designing particle swarm optimization with regression trees.

Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence CI–173/04, Universität Dortmund, Germany.

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

Bartz-Beielstein, T. and Naujoks, B. (2004). Tuning multicriteria evolutionary algorithms for airfoil design optimization. Interner Bericht des Sonderforschungsbereichs 531 Computational Intelligence CI–159/04, Universität Dortmund, Germany.

Bartz-Beielstein, T., Parsopoulos, K. E., and Vrahatis, M. N. (2004b). Analysis of particle swarm optimization using computational statistics. In Simos, T.-E. and Tsitouras, C., editors, *Proceedings International Conference Numerical Analysis and Applied Mathematics (ICNAAM)*, pages 34–37, Weinheim, Germany. Wiley-VCH.

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

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

Bartz-Beielstein, T., Zimmer, T., and Konen, W. (2008). Parameterselektion für komplexe modellierungsaufgaben der wasserwirtschaft – moderne CI-verfahren zur zeitreihenanalyse. In Mikut, R. and Reischl, M., editors, *Proc. 18th Workshop Computational Intelligence*, pages 136–150. Universitätsverlag, Karlsruhe.

Beielstein, T. (2003). Tuning evolutionary algorithms—overview and comprehensive introduction. Interner Bericht des Sonderforschungsbereichs 531 *Computational Intelligence* CI–148/03, Universität Dortmund, Germany.

Beyer, H.-G. (2001). *The Theory of Evolution Strategies*. Springer, Berlin, Heidelberg, New York.

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

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth, Monterey CA.

Cattaneo, G. and Italiano, G. (1999). Algorithm engineering. *ACM Comput. Surv.*, 31(3):3.

Chen, J., Chen, C., and Kelton, D. (2003). Optimal computing budget allocation of indifference-zone-selection procedures. Working paper, taken from `http://www.cba.uc.edu/faculty/keltonwd`. Cited 6 January 2005.

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge MA.

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

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

Fisher, R. A. (1935). *The Design of Experiments*. Oliver and Boyd, Edinburgh.

Fox, J. (2002). *An R and S-Plus Companion to Applied Regression*. Sage.

Hansen, N. (1998). *Verallgemeinerte individuelle Schrittweitenregelung in der Evolutionsstrategie*. Mensch & Buch, Berlin, Germany.

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314.

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

Klein, G. (2002). The fiction of optimization. In Gigerenzer, G. and Selten, R., editors, *Bounded Rationality: The Adaptive Toolbox*, pages 103–121. MIT Press, Cambridge MA.

Konen, W., Zimmer, T., and Bartz-Beielstein, T. (2009). Optimierte modellierung von füllständen in regenüberlaufbecken mittels ci-basierter parameterselektion. *at – Automatisierungstechnik*, 57(3):155–166.

Lasarczyk, C. W. G. (2007). *Genetische Programmierung einer algorithmischen Chemie*. PhD thesis, Technische Universitï¿½t Dortmund.

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

Maindonald, J. and Braun, J. (2003). *Data Analysis and Graphics using R—an Example-based Approach*. Cambridge University Press, Cambridge UK.

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

Mayo, D. G. (1996). *Error and the Growth of Experimental Knowledge*. The University of Chicago Press, Chicago IL.

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

Mehnen, J., Michelitsch, T., Lasarczyk, C. W. G., and Bartz-Beielstein, T. (2005). Multiobjective evolutionary design of mold temperature control using DACE for parameter optimization. In Pfützner, H. and Leiss, E., ed-

itors, *Proceedings Twelfth International Symposium Interdisciplinary Electromagnetics, Mechanics, and Biomedical Problems (ISEM 2005)*, volume L11-1, pages 464–465, Vienna, Austria. Vienna Magnetics Group Reports.

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

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. Wiley, New York NY.

Siegelmann, H., Horne, B., and Giles, C. (1997). Computational capabilities of recurrent NARX neural networks.

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

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

Volkert, L. (2006). Investigating EA based training of HMM using a sequential parameter optimization approach. In Yen, G. G., Lucas, S. M., Fogel, G., Kendall, G., Salomon, R., Zhang, B.-T., Coello, C. A. C., and Runarsson, T. P., editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 2742–2749, Vancouver, BC, Canada. IEEE Press.

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