Tuning Search Algorithms for Real-World Applications: A Regression Tree Based Approach

Thomas Bartz–Beielstein
Dept. of Computer Science
University of Dortmund
Dortmund, Germany
Email: thomas.bartz-beielstein@udo.edu

Sandor Markon
FUJITEC Co.Ltd. World Headquarters
28-10, Shoh 1-chome,
Osaka, Japan
Email: markon@rd.fujitec.co.jp

Abstract— **The optimization of complex real-world problems might benefit from well tuned algorithm's parameters. We propose a methodology that performs this tuning in an effective and efficient algorithmical manner. This approach combines methods from statistical design of experiments, regression analysis, design and analysis of computer experiments methods, and tree-based regression. It can also be applied to analyze the influence of different operators or to compare the performance of different algorithms. An evolution strategy and a simulated annealing algorithm that optimize an elevator supervisory group controller system are used to demonstrate the applicability of our approach to real-world optimization problems.**

I. Introduction

This article is dedicated to the question "how to find a set of working parameters for direct search algorithms when the number of allowed experiments is very low". This problem is relevant for the optimization of many real-world applications (RWA) [1]. Experiences gained during the last years at the collaborative research center "Design and Management of Complex Technical Processes and Systems by Means of Computational Intelligence Methods" in Dortmund show that engineers and other optimization practitioners adopt "standard parameterizations" found in the literature [2]. As experts for their specific optimization problem they want to apply, but not to analyze optimization algorithms. Hence, practitioners might be interested in a comprehensive method that supports the finding of suitable algorithm parameters.

Each algorithm-problem combination requires a specific parameterization. There is no standard parameter setting that works equally well on every problem. The no free lunch theorem for search states that there does not exist any algorithm that is better than another over all possible instances of optimization problems [3]. Many methods have been proposed in the past to tackle this problem. For instance, Jelasity presents a framework for extracting knowledge "that characterizes a whole domain" automatically [4]. Our goal is similar, but we will concentrate on single problem instances only. Therefore, it is related to the statistical methods developed by Kleijnen. His seminal book "statistical tools for simulation practitioners" proposes statistical tools such as regression analysis, analysis of variance etc. [5]. Kleijnen and Pala describe a situation where only a limited number of preliminary investigations are possible to find good parameter settings [6]. Unfortunately, many of these methods require special assumptions (normally distributed data with constant variance), and their results are sometimes hard to interpret.

Santner *et al.* describe modern statistical methods for designing and analyzing deterministic computer experiments. These modern techniques will be referred to as design and analysis of computer experiments (DACE) methods in the remainder of this paper [7]. They have been successfully applied to reduce the computational cost of optimization problems involving time–consuming function evaluations [8].

Tree based methods (classification and regression trees, CART) are intuitively to interpret and do not require assumptions regarding the underlying distribution [9]. We propose a combination of classical statistical tools, DACE and CART.

The main focus of this article lies on complex real-world optimization problems. Therefore we present a comprehensive example that describes problems of practical relevance: elevator supervisory group control, as one instance of "optimization via simulation" problems. We will demonstrate the applicability of our approach to two popular direct search algorithms: simulated annealing and evolution strategies.

The remainder of this article is structured as follows: in Section II, we introduce the elevator supervisory group control problem. Section III describes evolutionary algorithms and simulated annealing. A statistical methodology to set up computer experiments in an efficient manner is discussed in Section IV. Experimental results are presented and discussed in Section V. Finally, Section VI presents a summary and an outlook.

II. Elevator Group Control

The growing number of high rise buildings in the last decades is a challenge to the elevator industry. For instance, almost all new buildings in modern Hong Kong have more than 30 floors. The elevator group controller is the central part of an elevator system. It assigns elevators to customer service calls. These assignments are based on a certain heuristic or policy. This policy should be optimal with respect to overall service quality, customer waiting and transport times, traffic throughput, energy consumption, transportation capacity of the overall system, and many more goals. Especially in high-rise buildings elevator controllers have to meet many of these (sometimes conflicting) goals, making this problem a multi-criteria optimization problem.

In this article we will concentrate on one objective only: the minimization of the time passengers have to wait after having requested service until they can enter the elevator car. This time-span is called the waiting time.

We consider control strategies that have been implemented by *Fujitec*, one of the world's leading elevator manufacturers. *Fujitec* developed a neural network (NN) based controller that is trained by a set of fuzzy controllers. Each fuzzy controller represents control strategies for different traffic situations such as "up-peak"-traffic in office buildings in the morning, or "balanced" traffic" with lower intensity during the day. The resulting control strategies are robust, but not very efficient.

The control strategy of the NN depends on the network structure and the neural weights. Some of these weights are variable, whereas the network structure remains unchanged. Optimization of these weights might lead to an improved controller behavior [10]. A discrete-event based elevator group simulator has been provided by *Fujitec*. Hence, this problem can be characterized as an "optimization via simulation" problem.

Optimization via Simulation

Computer simulations are suitable means to optimize many actual real-world problems. Imagine e.g. a sequence of traffic signals along a certain route or elevators movements in high-rise buildings. "Optimization via simulation" subsumes all problems in which the performance of the system is determined by running a computer simulation. As the result of a simulation run is a random variable, we cannot optimize the actual value of the simulation output, or the performance of the system y. The goal of optimization via simulation is to optimize the expected performance $E[y(x_1, x_2, \dots, x_n)]$, where the x_i denote the controllable input variables [11].

In general, we consider global optimization problems. Function values may be provided by some sort of "black box", e.g. by running a simulation. For a given collection of function arguments corresponding function values can be generated, but not much more is supposed to be known. Random search algorithms and related stochastic optimization approaches are popular methods to optimize these problems [2].

III. Direct Search Algorithms

Evolutionary algorithms (EA) are randomized search heuristics that are well-known for their problem solving capabilities, especially for complex real-world applications. They can be characterized as direct search methods, since they do not construct a model of the fitness function [12]. Additionally to evolution strategies (ES), that build a special class of EA, a simulated annealing algorithm will be considered in this article.

TABLE I

Exogenous parameters of a "standard" evolution strategy. From [13]. The * symbol denotes qualitative factors.

Symbol	Factor	Parameter	Range	Typical Values
μ	P	Number of parent individuals	$\mathbb{N}$	15
$\nu = \lambda/\mu$	S	Offspring-parent ratio	$\mathbb{R}_+$	7
$\sigma_i^{(0)}$	InitS	Initial standard deviations	$\mathbb{R}_+$	3
κ	K	Maximum age	$\mathbb{R}_+$	1
c_τ	TauMult	Multiplier for individual and global mutation parameters	$\mathbb{R}_+$	1
n_σ	NSigma*	Number of standard deviations. D denotes the problem dimension	$\{1, D\}$	D
ρ	Rho*	Mixing number	$\{1, \mu\}$	μ
r_x	XReco*	Recombination operator for object variables	$\{i, d\}$	d (discrete)
r_σ	SReco*	Recombination operator for strategy variables	$\{i, d\}$	i (intermediate)

A. *Evolution Strategies*

An ES-algorithm run can be described briefly as follows: the parental population is initialized at time (generation) $t = 0$. Then λ offspring individuals are generated in the following manner: a parent family of size ρ is selected randomly from the parent population. Recombination is applied to the object variables and the strategy parameters. The mutation operator is applied to the resulting offspring vector. Selection is performed to generate the next parent population. If a termination criterion is not fulfilled, the generation counter (t) is incremented and the process continues with the generation of the next offspring.

Thus, an ES requires the specification of the following *exogenous* parameters before an optimization run is started:

1) Number of parent individuals: μ.
2) Number of offspring individuals: λ.
   Based on μ and λ, the selection pressure ν is defined as the offspring–parent ratio λ/μ. For given μ and ν values, λ is calculated as $\mu \cdot \nu$ and rounded to the nearest whole number.
3) Initial mean step sizes (standard deviations of the mutations of the decision variables): $\sigma_i^{(0)}$, $i = 1, \dots, n_\sigma$.
   The algorithms' performance may increase if problem specific $\sigma_i^{(0)}$ values for each dimension are chosen. To prevent an exponential blow up in the number of ES parameterizations, we assume scaled object variables. Therefore, only one initial step size is necessary for all dimensions: $\sigma^{(0)} = \sigma_i^{(0)} \quad \forall\ i \in \{1, \dots, D\}$. The

relevance of this parameter decreases with an increasing number of permitted iteration steps. As many real-world optimization problems permit only a small number of iterations, the selection of an adequate $\sigma^{(0)}$ value might improve the performance.

4) Number of standard deviations: n_σ with $1 \leq n_\sigma \leq D$. D denotes the problem dimension.
5) Multiplier for the individual mutation (learning) parameter and the global mutation parameter: c_τ. The value c_τ is used as a scaling parameter for τ_0 and τ: $\tau_0 = c_\tau/\sqrt{2D}$ and $\tau = c_\tau/\sqrt{2\sqrt{D}}$. Mutation is based on the extended log-normal rule, that enables learning perpendicular mutation ellipsoids [14].[1]
6) Mixing number: ρ.
The mixing number denotes the size of the parent family (mating population) that is chosen from the parent pool of size μ to create one offspring. The mixing number is treated as a qualitative factor with two levels: b denotes the bisexual scheme with $\rho = 2$, whereas m is the multisexual recombination scheme with $\rho = \mu$.
7) Recombination operator for object variables: r_x.
We consider discrete (d) and intermediate (i) recombination methods that depend on the mixing number ρ.
8) Recombination operator for strategy variables: r_σ.
9) Selection mechanism, maximum life span: κ.
Plus-strategies $(\mu + \lambda)$, and comma-strategies (μ, λ) can be generalized by introducing the parameter κ that defines the maximum age (in generations) of an individual. If κ is set to 1, we obtain the comma-strategy, if κ equals $+\infty$, we model the plus-strategy.

Beyer and Schwefel provide a comprehensive introduction to this special class of EA [14]. An in-depth discussion of evolutionary algorithms and other direct search methods can be found in [15].[2]

B. Simulated Annealing

Simulated annealing (SANN) is an (imperfect) analogy with the following phenomenon from thermodynamics: molecules of a liquid move freely at high temperatures. This mobility is lost, if the liquid is cooled down. Slow cooling enables the system to reach its minimum energy state: the molecules form a pure crystal that is completely ordered. If this cooling is too quick, a suboptimal state is reached.

Metropolis et al. [16] incorporated this method into numerical calculations. The method SANN in this article is taken from the freely available software package R [17]. It uses the Metropolis function for the acceptance probability. 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 [18]. The exogenous parameters for the standard

TABLE II
EXOGENOUS PARAMETERS FOR THE STANDARD SIMULATED ANNEALING (SANN). DESCRIPTION TAKEN FROM THE R DOCUMENTATION [17].

Symbol	Factor	Parameter	Default Values
t	temp	Controls the SANN method. Starting temperature for the cooling schedule	10
m	tmax	Number of function evaluations at each temperature for the SANN method	10

simulated annealing are shown in Tab. II. In addition to these parameters the SANN requires the specification of a vector of scaling values for the parameters.

IV. Experimental Analysis of Search Heuristics

The execution of a computer optimization program will be treated as an experiment [5]. The input parameters are called factors, whereas the output performances are called responses. We will use statistical experimental design to perform these experiments in an efficient manner. Our goal is to determine which of the exogenous algorithm's parameters have the greatest effect on the performance measure (screening), or which parameter setting might lead to an improved performance (modeling and optimization).

Experimental designs are well-known means in many other scientific disciplines (biology, psychology) to decide which particular configuration is tested in advance. As randomness is replaced by pseudo-randomness, computer experiments enable the experimenter to proceed from passive observation to active experimentation [5].

The software library `rpart` developed by Therneau and Atkinson was used to construct the regression trees [19]. It is part of the statistical software package R, that was used to analyze the experimental results [17]. The MATLAB toolbox DACE (design and analysis of computer experiments) was used to construct the Kriging approximation models [20].

A. Experimental design (DOE)

The "one-factor-at-a-time" method has been considered for a long time as the only correct way to perform experiments. It varies experimental factors one at a time, whereas the remaining factors are held constant. However, this method turned out to be inefficient in terms of the experimental runs needed: i.e., it is necessary to assume that there are no interactions, that means that the effect would be the same at the other settings of the other variables [21]. 2^k factorial designs are much more economical strategies than "one-factor-at-a-time" sequences of runs. The experimenter chooses two levels for each factor and performs optimization runs at each of the 2^k factor level combinations. The capital letters in the first row of Tab. III represent factors, and a minus and a plus sign denote the two levels of the factor. Choosing the levels "+" and "-" as 7 and 20 for the factor P respectively, leads to the configuration shown in the second column of Tab. IV.

[1]Although $c_\tau = 1$ leads to "standard" values that can be found in the literature: $\tau_0 = 1/\sqrt{2D}$ and $\tau = 1/\sqrt{2\sqrt{D}}$, we cannot recommend this parameterization. I.e., high dimensional fitness functions might require completely different τ resp. τ_0 values.

[2][12] is an updated and translated version of [15].

TABLE III

FRACTIONAL FACTORIAL 2^{9-5}_{III} DESIGN. THIS DESIGN IS USED FOR SCREENING THE ES PARAMETERS.

	A	B	C	D	E= ABC	F= BCD	G= ACD	H= ABD	J= ABCD
1	−	−	−	−	−	−	−	−	+
2	+	−	−	−	+	−	+	+	−
3	−	+	−	−	+	+	−	+	−
4	+	+	−	−	−	+	+	−	+
5	−	−	+	−	+	+	+	−	−
6	+	−	+	−	−	+	−	+	+
7	−	+	+	−	−	−	+	+	+
8	+	+	+	−	+	−	−	−	−
9	−	−	−	+	−	+	+	+	−
10	+	−	−	+	+	+	−	−	+
11	−	+	−	+	+	−	+	−	+
12	+	+	−	+	−	−	−	+	−
13	−	−	+	+	+	−	−	+	+
14	+	−	+	+	−	−	+	−	−
15	−	+	+	+	−	+	−	−	−
16	+	+	+	+	+	+	+	+	+

2^{k-p} fractional factorial designs are suitable to estimate factor effects when restricted computational resources prohibit full factorial designs or when many factors are under consideration. They can estimate the main effects and some interactions. Box *et al.* present rules for constructing fractional factorial designs [21].

B. Regression Analysis

Consider the linear regression model: $y = X\beta + \epsilon$. Given the results y_i from n optimization runs with corresponding experimental settings x_i $(i = 1, \dots, n)$, we can apply least squares methods to determine the values of q regression coefficients β_j $(j = 1, \dots, q \leq n)$, that minimize the sum of squares deviations between the observed and the fitted values in the regression model.

The statistical software package `R` provides the function `stepAIC` to perform an automated stepwise regression model selection based on Akaike's information criterion (AIC). AIC determines a log-likelihood value, according to the formula 2*(-maximized log-likelihood + npar), where npar represents the number of parameters in the fitted model, to determine the relevant parameters in the regression model. This value is an analogue to the cost-complexity parameter from Eq. 2.

How regression analysis can be used to perform a line search to improve the algorithm's performance is described in detail in [5], [1]. Kleijnen provides an in-depth discussion of regression methods in simulation, Bartz–Beielstein gives an introduction and overview over regression analysis and related methods for direct search algorithms.

C. Tree Based Regression

The screening phase during the DOE analysis is concerned with the search for interesting parts of the data. An exact model is only of minor importance at this stage of exploration. Tree-based regression methods might be advantageous, because they are simple and easy to interpret. Furthermore, they do not require specific assumptions regarding the distribution of the (fitness) values y_i.

Another attraction of tree based methods is their ability to handle missing values. Surrogate splits can be used if missing values are found in the tree growing phase.

Regression trees can be used for a combination of qualitative and quantitative factors such as population size and selection operators. Growing and pruning regression trees can be seen as a type of variable selection [22].

A regression tree is constructed in the following manner: the variable, that splits the tree best, is selected first. After splitting the data into two groups, this procedure is applied separately to each subgroup until a minimum size of the subgroup is reached (5 is a common value) or until no improvement can be made [19]. Cross-validation is used in a second stage of the tree construction to prune the tree.

1) Splitting criteria: In the terminology of classical analysis of variance, the splitting criterion can be characterized as a rule to maximize the between-groups sum-of-squares. Let SS_T denote the sums of squares for the node, and SS_L, SS_R the sums of squares for the left and right son. Then

$$SS_T - (SS_L + SS_R) \tag{1}$$

is used as a splitting criterion.

2) Cutting trees: To discuss the cutting methods in more detail, we introduce some symbols: $\overline{y}$ denotes the arithmetic mean of N values, $\overline{y} = \sum_{i=1}^{N} y_i/N$. The deviance at leaf i is defined as $D_i = \sum_{\text{cases j}} (y_j - \overline{y}_i)^2$, and $R(T)$ is the sum of D_i over the leaves that belong to the subtree T. $|T|$ is the number of terminal nodes in the tree.

A tree may be too complex to describe the data concisely. Pruning is an analogue to variable selection in regression (principle of parsimony). Cost-complexity pruning considers rooted subtrees of the tree T and "snips" off the least important splits. Consider the cost-complexity parameter k. As the set of rooted subtrees of T that minimize

$$R_k(T) = R(T) + k \cdot |T|, \tag{2}$$

is nested, we can find the optimal tree by sequential snipping on the current tree [9], [22].

TABLE IV

FRACTIONAL FACTORIAL DESIGN FOR EVOLUTION STRATEGIES.

	μ	ν	$\sigma^{(0)}$	n_σ	c_τ	ρ	r_x	r_σ	κ
1	7	4	1	1	1	2	i	i	1
2	20	4	1	1	2	2	d	d	$+\infty$
3	7	7	1	1	2	10	i	d	$+\infty$
4	20	7	1	1	1	20	d	i	1
5	7	4	5	1	2	10	d	i	$+\infty$
6	20	4	5	1	1	20	i	d	1
7	7	7	5	1	1	2	d	d	1
8	20	7	5	1	2	2	i	i	$+\infty$
9	7	4	1	12	1	10	d	d	$+\infty$
10	20	4	1	12	2	20	i	i	1
11	7	7	1	12	2	2	d	i	1
12	20	7	1	12	1	2	i	d	$+\infty$
13	7	4	5	12	2	2	d	d	1
14	20	4	5	12	1	2	i	i	$+\infty$
15	7	7	5	12	1	10	d	i	$+\infty$
16	20	7	5	12	2	20	i	d	1

rpart combines tree construction and pruning in one method. The threshold complexity parameter cp plays a central role in the rpart implementation. Let T_0 denote the tree with no splits. The following scaled version of Eq. 2 is implemented in rpart:

$$R_{cp}(T) = R(T) + R(T) \cdot cp \cdot |T| \cdot R(T_0). \tag{3}$$

This criterion can be interpreted as follows: let R^2 denote the proportion of the total variation about the mean $\overline{y}$ explained by the regression:

$$R^2 = \sum (\hat{y}_i - \overline{y})^2 / \sum (y_i - \overline{y})^2. \tag{4}$$

If any split does not increase the overall R^2 of the model by at least cp, than this split is not considered any further by the program [19]. The rpart program runs by default a 10-fold-cross-validation.

3) Complexity: The standard error (SE) and cross validation provide a rule (1-SE rule) to find the best number of splits: it takes the smallest cross validation error, adds the corresponding standard error to determine the least cross validation error that is smaller than this number. rpart's plotcp method enables a visual selection of a suitable cp value based on the 1-SE rule.

V. Experimental Results

The applicability of our approach to real-world optimization problems is demonstrated in this section. The tree-based regression approach is applied to three important tasks of experimental research: the analysis of the influence of existing operators, the integration of new or modified operators, and the comparison of different algorithms.

Eiben distinguishes three types of optimization problems: design problems (create one excellent solution at least once), repetitive problems (find good solutions for different problem instances), and on-line control problems (repetitive problems that have to be solved in real-time) [23]. As the elevator supervisory group control problem belongs to the second type of problems, $5,000$ fitness function evaluations have been chosen as a termination criterion (more than 90% of the execution time is spent on fitness evaluations).

As mentioned in Sec. II, the starting point of the optimization has been determined by a set of fuzzy controllers. This point is already close to the optimum (ceiling effect). Therefore, also minor improvements of the fitness values are relevant. Fitness function values smaller than 32.0 (our goal is to minimize the waiting time) have not been found so far. In the following, N_{exp} denotes the number of performed experiments for each configuration (the repeats with different random seeds).

A. Analyzing existing operators

Based on simulation data provided by *Fujitec Ltd. (Japan)* the influence of exogenous parameter settings on the performance of evolution strategies was investigated at *NuTech Solutions GmbH (Germany)* and the *Collaborative Research Center SFB 531, University of Dortmund (Germany)*. It was an interesting question whether or not the kappa selection scheme improves the performance. Starting point of our investigations was the data that have been collected during optimization runs in the last years. These data gave no clear evidence whether or not it might be beneficial to use kappa strategies. As the elevator group controller was modified (updated) during these investigations several times, results are not directly comparable. New experiments with the actual controller have been performed for this article. Future updates of the controller may change its behavior considerably. Therefore, methods used to obtain the results are of primary interest, whereas the results per se are of temporary interest only.

TABLE V

Evolution strategy: symbols and levels. Values chosen with respect to results found during optimization runs with different controller. The * symbol denotes qualitative factors.

Symbol	Factor	−1 Level	+1 Level
μ	P	7	20
ν	S	4	7
$\sigma^{(0)}$	InitS	1	5
c_τ	TauMult	1	2
κ	K	$+\infty$	1
n_σ	NSigma*	1	12
ρ	Rho*	b	m
r_x	XReco*	i	d
r_σ	SReco*	i	d

TABLE VI

First experiments to analyze the influence of κ selection on the performance of an evolution strategy optimizing the elevator group control problem. 32 experiments.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	34.27	0.2785	−1.2E+02	7.937E−40
P	−0.03506	0.01815	1.9	0.06356
InitS	−0.096	0.05964	1.6	0.1187
Rho1	0.1728	0.1193	−1.4	0.1586

Previous results, that were obtained with a different controller version, recommended the experimental design shown in Tab. V. Additionally to these 16 experimental runs, a second set of 16 runs was performed. The latter were based on run configurations shown in Tab. I. These configurations are discussed in detail in [1].

The following experiments were performed to investigate the question if the kappa selection scheme ($\kappa = 4$) improves the performance of an evolution strategy when optimizing the elevator group control problem.

Experiments, based on the experimental design from Tab. IV, were performed to gather the data. Each experimental setting was repeated only once due to the costly function evaluations. The all time best fitness function value, that has been found during the optimization run, has been chosen as response value y.

The 1-SE rule selects a tree with 2 splits. Figure 1 shows the unpruned tree. The first split partitions the $n = 32$ observations into groups of 8 and 24 events. The first group includes

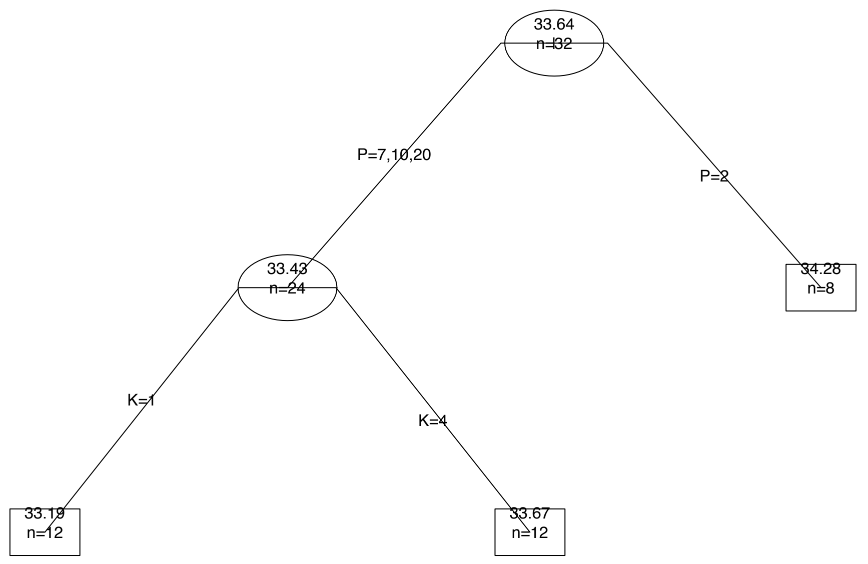

Fig. 1. Regression tree. The label of each node shows the mean fitness of an evolution strategy optimizing the elevator group control problem. The first split partitions the 32 events in the root node into two groups of 8 and 24 events. The average fitness value in the first (left) subnode reads 33.43, it is 34.28 in the second subnode. The population size ($P = \mu$) was chosen to perform the first split. The second split is based on the kappa value. The corresponding cost-complexity plot and the regression analysis show that only the first split is significant.

experimental runs with a population size of $\mu = 2$ and an average value of 34.28, the second group contains population sizes of $\mu = 7, 10$ and 20 with a fitness value of 33.43.

The corresponding classical regression analysis indicates that the population size μ has a significant effect. The reduced regression model in Tab. VI includes the initial step size $\sigma^{(0)}$ and the size of the mating pool ρ, but these factors are not highly significant. Finally, the algorithm was run with the tuned parameter setting. The tuned algorithm was able to find a solution with fitness $y = 32.252$.

These first results from 32 experiments indicate that an increased population size improves the performance. Classical regression analysis and tree-based regression support this assumption. DACE methods have not been used at this stage of experimentation, since the model contains 4 qualitative factors. Although a further increase of the population size is theoretically possible, we did not analyze this option any further due to the huge computational effort of these simulation experiments. We will concentrate our analysis on enhanced selection techniques such as threshold selection (cf. the newly developed Gupta selection scheme introduced below) instead. The kappa selection scheme did not produce the expected benefit.

Summarizing we can state that there is no evidence that kappa selection improves the ES performance in this case. A comma strategy performs equally good or even better than the kappa strategy. The population size seems to be more important than introducing an aging mechanism.

It is important to annotate that this result applies to this specific instance of an optimization problem only. Other elevator controllers might exhibit a different behavior. Results from the final run supported the assumption that our conclusions are correct.

B. Development of a new operator and comparision to existing standards

A common problem for direct search methods is the selection of a subset of k "good" candidate out of a set of n $(1 \leq k < n)$ under uncertainty. The goal of subset selection is to identify a subset containing the best candidate. Gupta proposed a single stage procedure (Gupta selection), that is applicable when the fitness of the candidates are balanced and are normal with common variance [24].

From $N \geq 2$ candidate solutions $X_1, \ldots, X_N$ n fitness values $Y_{i1}, \ldots, Y_{in}$ $(1 \leq i \leq N)$ are determined. Candidate X_i has a fitness value with unknown mean μ_i and common unknown variance σ_i^2. The sample mean of the ith candidate is $\overline{y}_i$, and $\overline{y}_{[1]} \leq \ldots \leq \overline{y}_{[N]}$ is the related order statistic. The selection of a (random-size) subset that contains the "best" candidate is called a correct selection (CS). Based on the pooled estimate $\hat{s}^2$ of the common variance σ^2 (with $m = N \cdot (n-1)$ degrees of freedom) the ith candidate is included in the selection if $\overline{y}_i \geq \overline{y}_{[N]} - h\hat{s}^2\sqrt{2/n}$. This selection scheme guarantees that for a pre-specified constant P^* a correct

TABLE VII

COMPARISON OF PLUS TO GUPTA SELECTION (MINIMIZATION).

Configuration	median	mean fitness	s.d.	best	N_{exp}
Plus selection	33.4980	33.6592	0.5784	32.8920	10
Gupta selection	33.1220	33.0736	0.4420	32.2040	10

selection is made with probability $P\{CS|(\mu,\sigma^2\} \geq P^*$. The equicoordinate critical point of the equicorrelated multivariate central t-distribution $h = T^{(1-P^*)}_{N-1,m,1/2}$ plays an important role in this selection scheme.

Applying the Gupta selection to evolutionary algorithms leads to dynamic population sizes: if the fitness function values are disturbed by noise, the population size is increased, whereas the population size is reduced in certain environments. To reduce the required memory (and to keep the implementation simple), for the ith candidate its averge fitness value $\overline{y}_i$ and the sum of its squared fitness values $\sum_{t=1}^{k} y_i^2(t)$ only have to be stored. Only the last 5 fitness values are used to determine these values (sliding window technique).

In order to demonstrate the performance of new algorithms or operators, it is a common practice to compare them to standard implementations. We propose a modification of this practice: for a given (set of) test problems, compare the tuned new algorithm to the tuned standard algorithm and mention the extra costs for the tuning procedures.

As the Gupta selection scheme belongs to the plus selection schemes, a comparison to the "classical" plus selection scheme was performed. The corresponding evolution strategies have been tuned with regression tree methods and the results from runs with the tuned parametrizations have been compared. The parametrizations of the tuned algorithms read: $\mu = 15$, $\nu = 7$, $\sigma^{(0)} = 3$, $n_\sigma = 36$, $c_\tau = 1$, $\rho = m$, $r_x = i$, $r_\sigma = d$, and $\mu = 10$, $\nu = 7$, $\sigma^{(0)} = 5$, $n_\sigma = 36$, $c_\tau = 1$, $\rho = m$, $r_x = i$, $r_\sigma = d$, for the plus and Gupta selection respectively. A comparison of tuned evolution strategies with plus and Gupta selection respectively is shown in Tab. VII. A t-test reveals that the difference is significant.

The Gupta selection scheme consists of several components: the reevaluation, the sliding window technique, the "cooling scheme" for the probability of a correct-selection, the variation of the population size etc. In addition to the analysis presented here, a detailled analysis of the significance of these components has to be performed. This analysis may lead to a deeper understanding of the selection scheme and might be helpful to close the gap between experiment and theory.

TABLE VIII

SIMULATED ANNEALING. FULL FACTORIAL 2^2 DESIGN

	A	B	tmax	temp
1	−	−	5	5
2	+	−	10	5
3	−	+	5	10
4	+	+	10	10

C. Simulated Annealing

The simulated annealing algorithm requires the determination of two exogenous strategy parameters only, see Tab. II: starting temperature for the cooling schedule, and the number of function evaluations at each temperature. Since these are only quantitative factors, the SANN algorithm provides an interesting example to compare classical regression analysis, tree-based regression and DACE methods.

In the following, we will tackle the question: which exogenous strategy parameters influence the performance of SANN optimizing the elevator group control problem instance provided by *Fujitec* significantly?

The problem instance used for these experiments differs from the optimization problem in the previous examples. They are based on a different controllers, their fitness function values are not comparable.

Due to the small number of parameters, a 2^2 full factorial design shown in Tab. VIII was chosen to perform the first experimental runs. The overall mean of the fitness values reads 1382.0. The tree based analysis leads to the hypothesis that 10 is a good value for both Temp and TMax. Additionally, TMax appears to have an greater influence on the algorithm's performance than Temp.

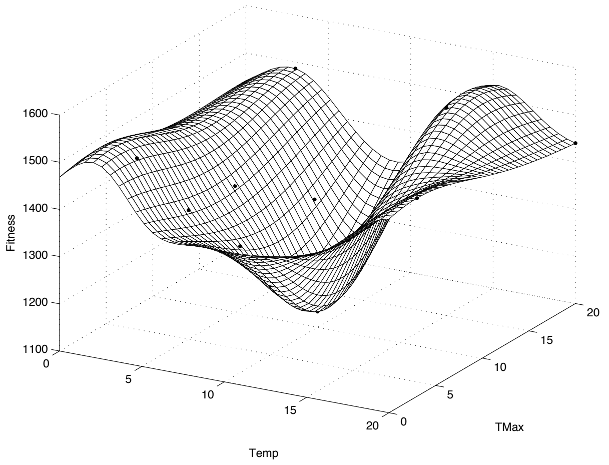

Fig. 2. Kriging approximation of the simulated annealing parameterization. The tree based regression analysis recommends a similar parameter setting. Classical regression techniques failed.

The classical regression analysis approach fails for this problem instance. It was not able to determine signicficant factors.

Figure 2 shows a Kriging surface approximation based on a Gaussian correlation function [20]. The results from this model are in correspondence with the tree-based result. The best function values might be obtained in the vicinity of the point (Temp, TMax) $= (10, 10)$.

Table IX presents a comparison of the different regression techniques: a technique is considered as flexible, if it can cope with different types of variables (quantitative, qualitative) and does not require assumptions on the underlying distribution. The results should be plausible, even complex interactions should be detected. Exactness can be related to the question whether the method is able to model gathered data and to predict new values. Finally, we are judging the availabilty of related literature and software packages as well.

TABLE IX
COMPARISON OF DIFFERENT ANALYSIS METHODS.

	Classical regression analysis [25]	Tree-based regression [9]	DACE [7]
Flexibility	-	++	-
Plausibility	+	++	++
Exactness	-	+-	++
Availability	++	+-	-

This overview reveals that a comparison of different techniques is useful. E.g. tree-based regression can be used at the first stage to screen out the important factors. If only a few quantitative factors remain in the model, DACE techniques can be applied to get an exact approximation of the functional relationsship between parameter settings and algorithms performance. Sequential designs have been applied successfully during our analysis [7].

VI. Summary and Outlook

A method was presented that enables the optimization practitioner to determine relevant parameter settings for optimization algorithms. Based on statistical design of experiments, classical regression analysis, tree based regression, and DACE models, the practitioner can select parameters in a pre-experimental phase, that lead to an improved performance of the algorithm. Summarizing we can state:

- Regression trees appear to be useful tools to complement classical and modern regression analysis techniques. The tree-based approach is consistent with the other approaches and sometimes easier to interpret than these methods [26]. Like DACE methods, tree based methods can extract information from the data even if the classical approach fails.
- Standard parameterizations of search algorithms might be improved significantly. This was demonstrated for a variant of a simulated annealing and an evolution strategy.
- Although test-suites might give hints for good starting points, it is worth to tune the parameterizations, especially when RWA are to be optimized.
- The applicability of our approach to an NP-hard sequential decision-making problem, the elevator supervisory group control, was demonstrated.

The approach presented here can also be applied to compare different optimization algorithms, e.g. ES to SANN, or to validate theoretical results. However, this is beyond the scope of this article, and will be subject of a further study.

Acknowledgments

I would particularly like to thank Oliver Ritthoff, Department of Computer Science (Artificial Intelligence), University of Dortmund, and Mike Preuss, Department of Computer Science (Systems Analysis), University of Dortmund, for many discussions. T. Bartz–Beielstein's research was supported by the DFG as a part of the collaborative research center "Computational Intelligence" (531).

References

[1] T. Bartz-Beielstein. Experimental analysis of search heuristics – overview and comprehensive introduction. *(Submitted to the EC journal)*, 2004.

[2] H.-P. Schwefel, I. Wegener, and K. Weinert, editors. *Advances in Computational Intelligence – Theory and Practice*. Natural Computing Series. Springer, Berlin, 2003.

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

[4] M. Jelasity. Towards automatic domain knowledge extraction for evolutionary heuristics. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 755–764, Berlin, 2000. Springer.

[5] J.P.C. Kleijnen. *Statistical Tools for Simulation Practitioners*. Marcel Dekker, New York, 1987.

[6] J. P. C. Kleijnen and O. Pala. Maximizing the simulation output: a competition. *Simulation*, 73:168–173, September 1999.

[7] T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

[8] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In J. J. Merelo Guervós et al., editor, *Parallel Problem Solving from Nature – PPSN VII, Proc. Seventh Int'l Conf., Granada*, pages 361–370, Berlin, 2002. Springer.

[9] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[10] T. Beielstein, C.P. Ewald, and S. Markon. Optimal elevator group control by evolution strategies. In *Proc. 2003 Genetic and Evolutionary Computation Conf. (GECCO'03), Chicago*, Berlin, 2003. Springer.

[11] J. Banks, J. S. Carson II, B. L. Nelson, and D. M. Nicol. *Discrete Event System Simulation*. Prentice Hall, 2001.

[12] H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. Wiley Interscience, New York, 1995.

[13] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[14] H.-G. Beyer and H.-P. Schwefel. Evolution strategies – A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[15] H.-P. Schwefel. *Numerische Optimierung von Computer–Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research*. Birkhäuser, Basel, 1977.

[16] R. Metropolis, A. Rosenbluth, A. Teller, and E. Teller. Simulated annealing. *Journal of Chemical Physics*, 21:1087–1092, 1953.

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

[18] C. J. P. Belisle. Convergence theorems for a class of simulated annealing algorithms. *Journal Applied Probability*, 29:885–895, 1992.

[19] T. M. Therneau and E. J. Atkinson. An introduction to recursive partitioning using the rpart routines. Technical Report 61, Department of Health Science Research, Mayo Clinic, Rochester, 1997.

[20] S.N. Lophaven, H.B. Nielsen, and J. Søndergaard. DACE - A Matlab Kriging Toolbox. Technical Report IMM-REP-2002-12, Informatics and Mathematical Modelling, Technical University of Denmark, 2002.

[21] G. E. P. Box, W. G. Hunter, and J. S. Hunter. *Statistics for experimenters*. Wiley series in probabilty and mathematical statistics: Applied probability and statistics. Wiley, 1978.

[22] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer, 3rd edition, 1999.

[23] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[24] S. S. Gupta. On some multiple decision (selection and ranking) rules. *Technometrics*, 7:225–245, 1965.

[25] D. C. Montgomery. *Design and Analysis of Experiments*. John Wiley & Sons, New York, NY, 5th edition, 2001.

[26] J. M. Chambers and T. H. Hastie, editors. *Statistical Models in S*. Wadsworth & Brooks/Cole, Pacific Grove, California, 1992.