Tuning Evolutionary Algorithms Overview and Comprehensive Introduction – Extended Abstract –

Thomas Beielstein

Department of Computer Science, University of Dortmund, Germany
thomas.beielstein@udo.edu,
WWW home page:http://ls11-www.cs.uni-dortmund.de

1 Introduction

At present, it is intensely discussed which type of experimental research methodologies should be used to improve the acceptance and quality of evolutionary algorithms (EAs). A broad spectrum of presentation techniques makes new results in evolutionary computation (EC) almost incomparable. Discussions and sessions related to this subject took part during the congress on evolutionary computation (CEC) and on the genetic and evolutionary computation conference (GECCO). In [1], Eiben and Jelasity list explicitely four problems, that result from this situation:

- the lack of a standardized test-functions, or benchmark problems,
- the usage of different performance measures,
- the impreciseness of results, and therefore no clearly specified conclusions, and
- the lack of reproducibility of experiments.

EC shares these problems with other scientific disciplines. Solutions from these other disciplines, that have been successfully applied for many years, might be transferable to EC. Here we can mention: statistical design of experiments [2], design of computational experiments to test heuristics [3, 4], experimental designs for simulation [5], or deterministic computer experiments [6].

We suggest to use techniques that are well known in statistics under the name design of experiments (DOE) for many decades. In our approach, an experiment consists of a problem and its related fitness function, an algorithm, and a quality criterion: we will use design of experiments, regression analysis, and generalized linear models, to improve algorithm performance. The main focus in this paper lies on natural problem classes: its elements are problems that are based on real-world optimization problems in contrast to artificial problem classes [1].

Thus, the approach presented here might be interesting for optimization practitioners that are confronted with a complex real-world optimization problem in a situation where only few preliminary investigations are possible to find good parameter settings [7, 8]. Kleijnen and Pala describe a competition, organized by the Business Section of the Netherlands Society for Statistics and Operation

Research: Only 32 runs are permitted to obtain the maximum output for a given simulation model by selecting the best combination of six inputs [9]. Furthermore, DOE is applicable a priory to tune different parameter settings of two algorithms to provide a fair comparison.

The approach presented here can be used to solve the following tasks:

Investigation: Analyzing and tuning models and optimization criteria: i.e. what are important parameters, what should be optimized?

Comparison: Comparing the performance of competing stochastic search algorithms such as evolutionary algorithms, simulated annealing etc.

Conjecture: Understanding, further research, based on statistics and visualization techniques

Quality: Improving the robustness of simulation runs.

Additionally, these methods can be used in other contexts to improve the optimization runs, i.e. to generate systematically feasible starting points that are better than randomly generated initial points. Our approach has to be distinguished from the following approaches:

1. Classical design of experiments as used in industrial optimization and computer experiments. Although experimental design is a sub-discipline within mathematical statistics, its techniques must be adapted if applied to simulation models: Stochastic simulation as introduced in Eq. 3 uses pseudo-random numbers. Since common or antithetic seeds can be used, the simulation practitioner has much more control over the noise in the experiments [8]. *Randomness is replaced by pseudo-randomness.* Therefore, we can control the source of variability. The different simulation runs for one specific factor combination can be performed under exactly the same conditions. Blocking and randomization, important techniques to reduce the systematic influence of different experimental conditions, are unnecessary in simulation. The random number seed is the only random element in a random simulation model.
2. The Meta-EA approach, i.e. [10, 11], might locate good parameter sets without providing much insight as how sensitive performance is to parameter changes.
3. Schaffer's study of control parameters of GAs [12]. Schaffer proposes a complete factorial design experiment that requires $8,400$ run configurations, each configuration was run to $10,000$ fitness function evaluations. In contrast to this, we propose an approach that requires a small amount of fitness function evaluations only. Furthermore, our approach takes the underlying problem instance into account and does not draw any conclusions that are problem independent.
4. In contrast to Eiben et al. [13], our approach is based on parameter tuning, and not on parameter control. Parameter control deals with parameter values that are changed during the optimization run. The assumption that specific problems require specific EA parameter settings is common to both approaches [14].

5. Considering the exemplary study of simulated annealing by David Johnson's group [15, 16], our approach is related to the discipline experimental algorithmics [17]. Thus it can be added to the category 'assessment of heuristics' as classified in [18]. But in contrast to approaches from experimental algorithmics, that provide methodologies for the design, implementation, and performance analysis of computer programs for solving algorithmic problems, our goal is to provide methods for very complex real-world problems.
6. Empirical modeling of genetic algorithms as presented by Myers and Hancock [19]. Their methodology has a different goal than our approach: we are trying to tune an evolutionary algorithm with the fewest amount of experiments, whereas Myers and Hancock's approach requires 129,600 program runs.
7. François and Lavergne demonstrate the applicability of generalized linear models to design evolutionary algorithms [20]. This approach is similar to [19]. Again, data sets of size 1,000 or even more are necessary, although only a simplified evolutionary algorithm with two parameters (Moses) is designed.
8. Sacks et al. model the deterministic output of a computer experiment as the realization of a stochastic process [6]. This approach differs significantly from our approach, the same applies to design and analysis of computer experiment (DACE) methods [21].

Despite of the differences mentioned above, it might be beneficial to adapt some of these well-established ideas from other fields of research to improve the acceptance and quality of evolutionary algorithms.

This paper is structured as follows: section 2 describes how evolutionary algorithms can be treated as experiments. Evolution strategies (ES), as a special class of evolutionary algorithms, and their parameterizations are introduced in this section. How stochastic optimizers for complex real-world problems can be compared is discussed in Sec. 3. DOE methods that are used to perform a comparison of different simulation run configurations are presented in Section 5. Finally, section 6 introduces the DOE-software package JEA and gives some constructive examples.

The main contribution of this paper is to propose an answer to the complex question: how to determine strategy-parameters for search algorithms that are suitable to optimize efficiently and effectively real–world optimization problems? Since the solution to this question requires the determination of a adequate performance measure for algorithms, we are able to provide a partial answer to the problems mentioned by Eiben and Jelasity.

2 Evolutionary Algorithms

2.1 EA as Experiments

The popularity of evolutionary algorithms has constantly increased in the last decades, but there are many degrees of freedom when starting an optimization run. The optimization of real-world problems requires good initial strategy-parameters, since many real-world problems need high–dimensional inputs and

Table 1. Fractional factorial 2^{5-2}_{III} design. This design was used in [22] to tune a multi criteria evolutionary algorithm for airfoil design optimization. The first column shows the run configuration. Capital letters (A,B,C) and their combinations denote population size, offspring-parent ratio, selection mechanism (age), and recombination operators respectively. Columns 7-11 translates the DOE factor representation into the corresponding ES parameter values, cp. Tab. 2. The remaining columns present results y_i from 5 simulation runs with different random seeds r_0.

$\boldsymbol{p}_i$	A	B	C	$D = AB$	$E = AC$	μ	r	κ	R_1	R_2	y_1	y_2	y_3	y_4	y_5
1	−	−	−	+	+	2	2	1	GI	GI	.0	.07	.0	.07	.06
2	+	−	−	−	−	5	2	1	GD	GD	.06	.0	.07	.0	.09
3	−	+	−	−	+	2	5	1	GD	GI	.0	.0	.0	.16	.0
4	+	+	−	+	−	5	5	1	GI	GD	.18	.35	.67	.14	.03
5	−	−	+	+	−	2	2	250	GI	GD	.09	.0	.19	.08	.01
6	+	−	+	−	+	5	2	250	GD	GI	.11	.17	.02	.1	.15
7	−	+	+	−	−	2	5	250	GD	GD	.18	.0	.04	.21	.45
8	+	+	+	−	+	5	5	250	GD	GI	.38	.41	.01	.24	.21

are computationally expensive. Therefore only few optimization runs are possible, that should be performed with 'good' strategy parameters. Optimization practitioners are interested in the development of models to perform the optimization with a minimum amount of simulation runs. DOE techniques can be applied to optimization algorithms such as evolutionary algorithms, considering the run of an algorithm as an experiment, gaining insightful conclusions into the behavior of the algorithm and the interaction and significance of its parameters. The first two steps in this analysis are called screening and modeling. In a third step, we can use RSM to improve significant parameter settings (optimization). Therefore, DOE methods, combined with response surface methods, might lead to an improved EA design. From the viewpoint of an experimenter, factors can be defined as parameters, variables, and behavioral relationships that can be changed during an experiment. How factors can be interpreted from a statistical point of view will be discussed in Sec. 5.

Generally, there are two different factors that influence the behavior of an optimization algorithm: problem specific factors, i.e. the fitness function, and algorithm specific factors, i.e. the population size. Latter can be divided into endogenous and exogenous factors. Exogenous factors, or 'exogenous strategy parameters' as they are called in [23], are kept constant during the optimization run, whereas endogenous strategy parameters, i.e. step-widths, are modified by the algorithms during the run.

2.2 Evolution Strategies: Algorithm Specific Factors

The methodology presented in this paper leads to results that are tailored for one specific algorithm-optimization combination. It is transferable to any kind

of EA design or even any parameterizable search stochastic algorithm such as simulated annealing, tabu search or genetic algorithms. To give an example, we will analyze evolution strategies. Evolution strategies build a special class of EAs. Beyer and Schwefel provide a comprehensive introduction to evolution strategies [23].

An ES-algorithm run can be described briefly as follows (see Fig. 1): The parental population is initialized at time (generation) $t = 0$. λ offsprings 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. A termination criterion is tested. If this criterion is not fulfilled, the generation counter (t) is incremented and the process continues with the generation of the next offspring generation.

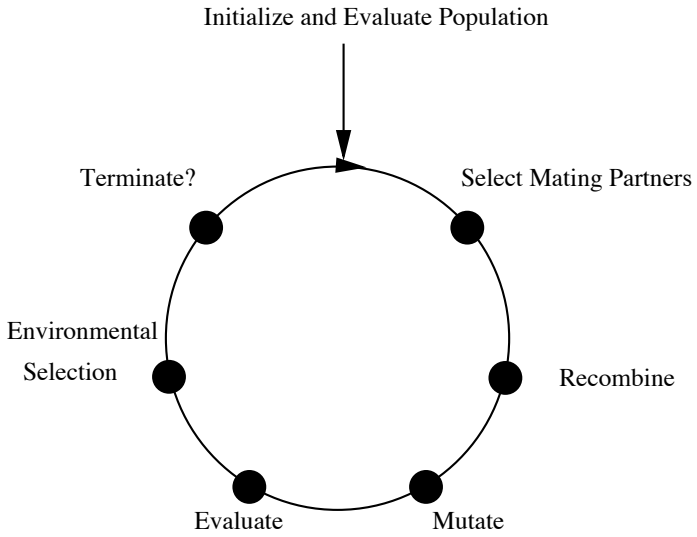

Fig. 1. Evolutionary algorithms.

In the following, we extend the classical $(\mu/\rho \stackrel{+}{,} \lambda)$ ES notation to an experimental design vector representation. We consider the following eleven parameters or control variables:

1. Number of parent individuals: μ
2. Number of offspring individuals: λ. Based on μ and λ, we can define the offspring–parent ratio: $\nu := \lambda/\mu$.
3. Selection mechanism: 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. This representation will be used throughout the rest of this article.
4. Number of standard deviations: n_σ
5. Global mutation parameter: τ_0
6. Individual mutation parameter: τ_i
7. Mixing number: ρ, the size of the parent family that is chosen from the parent pool of size μ to create λ offsprings.
8. Recombination operator for object variables: R_1. We consider global intermediate GI, global discrete GD, local intermediate LI, and local discrete LD recombination. Discrete (dominant) recombination is recommended for the object variables.
9. Recombination operator for strategy variables: R_2 . Beyer and Schwefel recommend intermediate recombination to reduce large fluctuations in the evolutionary dynamics [23].
10. Random seed: r_0.
11. Total number of iterations: N_{tot}.

The following vector notation provides a very compact description of an ES parameter design [23, 24].:

$$\boldsymbol{p}_{\mathrm{ES}} = (\mu, \nu, \kappa, n_\sigma, \tau_0, \tau_i, \rho, R_1, R_2, r_0, N_{\mathrm{tot}})' \,. \tag{1}$$

This representation will be used throughout the rest of this article and is summarized in Tab. 2, which also shows typical parameter settings. The exogenous strategy parameters are kept constant during one optimization run. They are summarized in Table 2. A vector notation can be used for the optimization and problem specific parameters such as the fitness function f, the problem dimension D, the number of repeats for each scenario N_{exp} etc.

$$\boldsymbol{d} = (f, D, n_{\max}, N_{\mathrm{exp}}, N_{\mathrm{reeval}}, \ldots)'. \tag{2}$$

A simulation run configuration consists of the problem specific design configuration, e.g. represented by $\boldsymbol{d}$ as shown in Eq. 2 and the parameter design configuration $\boldsymbol{p}$, e.g. as shown in Eq. 1.

In the next section, we will discuss how different algorithms can be compared. Based on these considerations we are able to apply DOE methods to improve the behavior of the ES for a given optimization configuration. This task can be interpreted as the determination of optimal values of $\boldsymbol{p}^*$ for a given problem design $\boldsymbol{d}$ or as an analysis of a regression meta-model [25].

3 Comparing Stochastic Optimizers

3.1 Performance Measures

Before we can compare two algorithms, we have to specify a concrete measure for their comparison. Unfortunately, there are nearly as many different measures

Table 2. DOE parameter for ES.

Symbol	Parameter	Typical Values
μ	number of parent individuals	$10 \ldots 100$
$\nu = \lambda/\mu$	offspring-parent ratio	$5 \ldots 7$
n_σ	number of standard deviations	$1 \ldots D$
τ_0	global mutation parameter	$1/\sqrt{2\sqrt{D}}$
τ_i	individual mutation parameter	$1/\sqrt{2D}$
κ	age	$1 \ldots \infty$
ρ	mixing number	2
R_1	recombination operator for object variables	{discrete}
R_2	recombination operator for strategy variables	{intermediate}
N_{tot}	total number of fitness function evaluations	$5 \cdot 10^3$

for the goodness of an algorithm as optimization algorithms, i.e. the quality of the best solution, the percentage of runs terminated successfully, the number of iterations or time steps required to obtain the results, the robustness of the algorithm, or the distance between the best and easily found solutions. The performance measure under consideration should make the comparison well-defined, algorithmic, reproducible and fair. A good overview over the experimental analysis of algorithms is given in [26]. To demonstrate that there is no canonical measure, we will list some typical measures:

- The mean best fitness can be defined as the average value of the best fitness values found at termination for one specific run configuration. Considering the quality of the best solution, it is a common practice to show a graph of the solution quality versus time. A model based on this measure will be discussed in Sec. 5.6.
- If the optimal solution is known, the percentage of run configurations terminated successfully (success rate) can be used to measure the performance of an algorithm [27]. Eiben et al. discuss the relationship between success rate and mean best fitness [1]. We will present a model based on the success rate in Sec. 5.6.
- Schaffer proposes the following technique to determine the number of fitness function evaluations at which to compare different optimization algorithms: choose a point at which some, but not all run configurations, are doing well. 'Doing well' is defined as at least 10% of the run configurations located the optimum at least 50% of the time [12].
- Time-dependent measures such as the performance profile can be used to measure the computational effort [28]. A measure to determine the quality-effort relationship can be defined as the ratio of time to produce a solution within 5 percent of the best-found solution value $t_{0.05}$ to the time to produce that best value t_{best} [29]: $r_{0.05} = t_{0.05}/t_{best}$. Since running-times depend on

the computer system, measures of computational effort might be advantageous: Counting operations, especially for major subtasks such as fitness function calls can be mentioned in this context explicitly. The performance ratio and the performance profile can be based on the number of function evaluations.
- Robustness can be defined as a good performance over a wide range of instances of one test problem or even over a wide range of different test problems.
- To measure the algorithm speed, the average number of evaluations to a solution can be used. For runs finding no solutions, the maximum number of evaluations can be used.
- Barr and Hickman discuss performance measures for parallel algorithms [3].

Selection of Measures We will consider the quality of the best solution found during an optimization run as a performance measure and propose two different approaches:

The first approach is based on the performance values found during the run, whereas the second approach interprets the outcome of an experiment as a proportion. In the second approach, the success rate is used to analyze a binomial model. This will lead to logistic regression models.

Since many optimization runs produce experimental results that do not follow a Gaussian distribution, our conclusions will be based on generalized linear models (GLMs). On the other hand, if the performance results are Gaussian, or can be easily transformed to Gaussian distributed values, classical analysis of variance (ANOVA) or regression model can be recommended.

4 Test functions

Although the no free lunch theorem (NFL) for search states that there does not exist any algorithm that is better than another over all possible instances of optimization problems, this result does not imply that we should not compare one algorithm to another. Keeping in mind that we are considering problems of practical interest, the reader may refer to the discussion in [30, 31].

5 Analysis: Design of Experiments

5.1 Experiments

As we have classified important parameters of evolution strategies in Sec. 2, and defined a measure for their comparison in Sec. 3, we can conduct experiments to assess the significance of single parameters such as population size or selective pressure. We should keep in mind that simulation runs are treated as experiments: we begin by formulating a hypothesis, then set up experiments to gather data that do verify or falsify this hypothesis. We will use guidelines from experimental algorithmics in our experimental studies [18]:

1. State a clear set of objectives: formulate the question as a hypothesis.
2. After an experimental design is selected, simply gather data. Do not modify the hypothesis until all data have been collected.
3. Analyze the data to test the hypothesis stated above.
4. In the next cycle of experimentation a new hypothesis can be tested.

5.2 How to choose exogenuous parameters

Evolutionary algorithms and many other related algorithms such as simulated annealing [32] or tabu search [33], require the determination of exogenous parameter settings before the optimization run is performed. Thus, the choice of an adequate exogenous parameter setting is based on expert knowledge. Johnson suggests to explain the corresponding adjustment process in detail, and therefore to include the time for the adjustment in all reported running-times to avoid a serious underestimate [26]. An important step to make this procedure more transparent and more objective is to use DOE techniques: they provide an algorithmical procedure to tune the exogenous parameter settings for the algorithms under consideration before the complex real-world optimization task is optimized or two algorithms are compared.

Besides the improved performance of an algorithm, fine-tuning of exogenous parameters might reveal information about its robustness. This may lead to new insights in the rôle of the offspring-parent ratio ν, or the relationship between recombination and mutation operator . Experimental design provides an excellent way of deciding which simulation runs should be performed so that the desired information can be obtained with the least amount of experiments [34, 35, 7, 25, 36].

5.3 Basic DOE Terminology

The input parameters and structural assumptions, that define a simulation model are called *factors*, the output value(s) are called *response(s)*. The different values of parameters are called *levels*. Levels can be qualitative, i.e. selection scheme, or quantitative, i.e. population size. An experimental design is a set of factor level combinations. Kleijnen defines DOE as 'the selection of combinations of factor levels that will be simulated in an experiment with the simulation model'[37]. One parameter design setting, cp. Eq. 1, is run for different pseudo-random number settings, resulting in replicated outputs. We will discuss linear regression models and their extensions, the so-called generalized linear models [38, 39].

5.4 Linear Regression Models

Generally, a simulation model can be represented as follows:

$$y = f_1(z_1, \ldots, z_k, r_0), \tag{3}$$

where f_1 is a mathematical function, e.g. $f_1 : \mathbb{R}^{k+1} \to \mathbb{R}$: Given the values of the argument z_i and the random number seed r_0, the simulation program yields

exactly one value. The Taylor series expansion yields the first order approximation $y = f_2 = \sum_{i=0}^{k} \beta_i z_k$. The last equation is the basis for regression models based on simulation data. Our goal is to use least square methods to estimate the linear model $y = X\beta + \epsilon$, where the y denotes a column vector with the n responses, ϵ is the vector of n error terms, and β denotes is the vector with q parameters β_j ($n \geq q$). The error term ϵ has expectation $E(\epsilon) = 0$ and variance $V(\epsilon) = \sigma^2$. X is the $(n \times q)$ matrix of independent regression variables. x_0 is the dummy variable equal to $\mathbf{1}$, the remaining $q-1$ variables correspond to the simulation parameters z. Therefore, we obtain the $(n \times q)$ regression matrix

$$X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1,q-1} \\ \vdots & & & & \vdots \\ 1 & x_{i1} & x_{i2} & \cdots & x_{i,q-1} \\ \vdots & & & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{n,q-1} \end{pmatrix}. \tag{4}$$

Experimental settings (designs), where the regression matrix X satisfies $XX^T = nI$, are called orthogonal. In the following, we will consider orthogonal designs with two factors for each level (2^k factorial designs and 2^{k-p} fractional factorial designs), and orthogonal design where center points are added to the 2^k design (central composite designs). A variable x is called standardized, if x ranges between -1 and $+1$. The original variables with range $[l, h]$ can be standardized using the linear transformation $x = a + bz$ with $a = (l + h)/2$ and $b = (l - h)/2$. Thus, the entry -1 in the regression matrix denotes a factor at its low level, and $+1$ a factor at its high level. The intuitive definition of a *main effect* of a factor is the change in the response produced by the change in the level of that factor averaged over the levels of the other factors, whereas the *interaction* effect AB of factor A and factor B is the average difference between the effect of A at the high level of B and the effect of A at the low level of B.

5.5 Tuning

Generally, we suggest the following three-stage approach: Screening – Modeling – Optimization.

1. Screening: Screening considers only the main effects and no interactions. Therefore, we recommend fractional-factorial 2^{k-p} designs. These are orthogonal designs and require a moderate number of experiments. If we cannot differentiate between two effects, these effects are called confounded. An 2^{k-p} design is of resolution R if no p-factor effect is confounded with another effect that has less than $R - p$ factors [34]. Roman numerals denote the corresponding design resolution. Our first experiments are based on resolution III designs (see Tab. 1). These designs ensure that no main effect is confounded with any other main effect, but main effects can be confounded with two-factor interactions.

2. Modeling: Interactions are taken into account. At this stage resolution IV or resolution V designs are recommended.
3. Optimization: Central composite designs, that require a relatively high number of runs, are often used at this experimental stage. They can be combined with response surface methods.

We apply the standard techniques from regression analysis for meta-model validation, cf. [40]. A linear approximation may be valid in a sub-domain of the full experimental area. In RSM, we determine the direction of improvement using the 'path of the steepest descent' (minimization problem) based on the estimated first-order model [25]. The path of the steepest descent is perpendicular to the fitted first-order model. If no further improvement along the path of the steepest descent is possible, we can explore the area by fitting a local first-order model and obtain a new direction for the steepest descent. This step is repeated several times until we find the expected optimum area. As a criterion we may use that the linear model is inadequate close to the optimum. Optionally, a second-order model that requires additional runs and special RSM designs, can be fitted in the expected optimum area. The optimal values are estimated by taking the derivations of the second-order regression model. We combine in our approach DOE and RSM techniques, that are adapted to the special needs and restrictions of the optimization task.

A numerical example of this procedure is omitted in this extended abstract but will be included in the full article. Alternatively, the reader is referred to [22], where the parameterization of an ES on the airfoil-design optimization problem is discussed.

5.6 Generalized Linear Models

Linear models, regression analysis, and analysis of variance as discussed so far are applicable to problems that have errors that are Gaussian. In many situations the optimization practitioner has to face response values that follow some skewed distribution or have non-constant variance. To deal with non-normal responses, data transformations are often an effective way. The choice of an adequate transformation can be difficult. Draper and Smith discuss the need for transformation and present different transformation methods [40]. Since the transformation may result in incorrect values for the response value, i.e. $\log Y$, if $Y < 0$, generalized linear models provide an alternative [39].

A Model Based on the Quality of the Best Solution Histograms can give first hints, whether or not it might be adequate to use a specific distribution (i.e. a Gamma distribution might be better suited than the Gaussian distribution). Next we can consider QQ-plots and or a KS-test to support the assumption. Stepwise model selection, by adding or removing terms, can be used to find a more parsimonious regression model [38]. The statistical software package R provides functions such as `stepAIC` that automate this selection process [41].

An example is omitted here, the reader is referred to [20] for an example that uses a simplified EA and artifical test problems.

A Logistic Regression Model Based on the Success Rate Whether or not the optimization run has located a pre-specified optimum can be used as a performance measure for algorithms. In this case, where the outcome variable can take only two values, a linear regression model is not appropriate, but a logistic regression model might be adequate. Thus, each (optimization run, problem)-combination generates a binary response value:

$$\mathbf{D} \times \mathbf{P} \rightarrow \mathbb{B}, \tag{5}$$

with $\boldsymbol{d} \in \boldsymbol{D}$ and $\boldsymbol{p} \in \boldsymbol{P}$ as defined in Eq. 1 and Eq. 2 respectively. As N_{exp} optimization runs for each factor-level setting will be performed, the number of successful runs can be seen as a random variable having a binomial distribution.

For an introduction into logistic regression the reader is referred to [42], Myers and Hancock present an example that uses a genetic algorithm to solve consistent labeling problems [19].

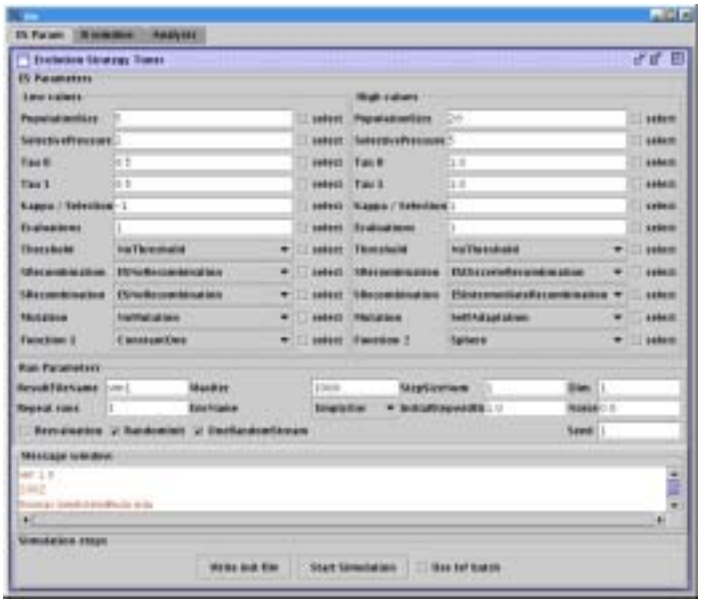

Fig. 2. The DOE software package BEA combines JAVA classes and the statistical software package R. The figure shows a screen-shot of the JEA GUI (XJEA).

6 The JEA-Software Package

The JAVA evolutionary algorithms (JEA) software package provides access to standardized implementations of evolutionary algorithms and related optimization algorithms such as particle swarm optimizers. JEA combines methods from

statistical DOE and simulation analysis. It enables the simulation practitioner to tune special algorithms for a specific optimization problem and to compare different variants of one (or even of several) algorithms. Fig. 2 shows the JEA graphical user interface XJEA, that enables a convenient access to JAVA and R objects. The related software package JEAT is described in [43].

7 Summary and Outlook

This paper introduces a framework that can be used to improve the acceptance and quality of research in the field of evolutionary computation. It summarizes state of the art techniques for EA parameter tuning. Parameter tuning is beneficial in the following situations: Real-world optimization problems allow only a few preliminary experiments to find good EA parameter settings. As the commonly used 'one factor at a time approach' is considered as inefficient and ineffective, we recommend DOE methods. To enable a fair and more objective comparison of different optimization algorithms, these algorithms should be tuned a priory.

Referring to the methodology proposed by François and Lavergne, GLM are used to handle responses that are not Gaussian. If the responses are normally distributed, classical regression analysis can be applied.

References

1. A.E. Eiben and M. Jelasity. A critical note on experimental research methodology in ec. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'2002)*, pages 582–587. IEEE Press, 2002.
2. R. A. Fisher. *The Design of Experiments*. Oliver and Boyd, Edinburgh, 1935.
3. R. Barr and B. Hickman. Reporting computational experiments with parallel algorithms: Issues, measures, and experts' opinions. *ORSA Journal on Computing*, 1992.
4. J. Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1996.
5. W.D. Kelton. Experimental design for simulation. In J.A. Joines, R.R. Barton, K. Kang, and P.A. Fishwick, editors, *Proceedings of the 2000 Winter Simulation Conference*, 2000.
6. J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.
7. J. Kleijnen. *Statistical Tools for Simulation Practitioners*. Marcel Dekker, New York, 1987.
8. J. P. C. Kleijnen. Experimental design for sensitivity analysis, optimization, and validation of simulation models. In J. Banks, editor, *Handbook of simulation*. Wiley, New York, 1997.
9. J. P. C. Kleijnen and O. Pala. Maximizing the simulation output: a competition. *Simulation*, 73:168–173, September 1999.
10. T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

11. F. Kursawe. *Grundlegende empirische Untersuchungen der Parameter von Evolutionsstrategien — Metastrategien.* Dissertation, Fachbereich Informatik, Universität Dortmund, 1999.
12. J. D. Schaffer, R. A. Caruana, L. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, CA, 1989. Morgan Kaufman.
13. A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, 3(2):124–141, 1999.
14. D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
15. D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. part i, graph partitioning. *Operations Research*, 37(6):865–892, 1989.
16. D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation. part ii, graph coloring and number partitioning. *Operations Research*, 39(3):378–406, 1991.
17. C. Demetrescu and G. F. Italiano. What do we learn from experimental algorithmics? In *Mathematical Foundations of Computer Science*, pages 36–51, 2000.
18. B. Moret. Towards a discipline of experimental algorithmics, 2000.
19. R. Myers and E.R. Hancock. Empirical modelling of genetic algorithms. *Evolutionary Computation*, 9(4):461–493, 2001.
20. O. François and C. Lavergne. Design of evolutionary algorithms – a statistical perspective. *IEEE Transactions on Evolutionary Computation*, 5(2):129–148, April 2001.
21. R. J. Beckman M. D. McKay and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.
22. T. Beielstein and B. Naujoks. Tuning multi criteria evolutionary algorithms for airfoil design optimization. Technical report, Universität Dortmund, 2003. (to appear).
23. H.-G. Beyer and H.-P. Schwefel. Evolution strategies – A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.
24. T. Beielstein and S. Markon. Threshold selection, hypothesis tests, and DOE methods. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 777–782. IEEE Press, 2002.
25. J. Kleijnen and W. Van Groenendaal. *Simulation - A Statistical Perspective.* Wiley, Chichester, 1992.
26. D. Johnson. A theoretician's guide to the experimental analysis of algorithms, 1996.
27. T. Beielstein, J. Dienstuhl, C. Feist, and M. Pompl. Circuit design using evolutionary algorithms. In David B. Fogel, Mohamed A. El-Sharkawi, Xin Yao, Garry Greenwood, Hitoshi Iba, Paul Marrow, and Mark Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1904–1909. IEEE Press, 2002.
28. E. D. Dolan and J. J. More. Benchmarking optimization software with performance profiles. Technical Report ANL/MCS-P861-1200, Argonne National Laboratory, 2001.

29. R. Barr, B. Golden, J. Kelly, M. Rescende, and W. Stewart. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9–32, 1995.
30. D.L. Whitley, K.E. Mathias, S. Rana, and J. Dzubera. Building better test functions. In L. Eshelman, editor, *Proc. of the Sixth Int. Conf. on Genetic Algorithms*, pages 239–246, San Francisco, CA, 1995. Morgan Kaufmann.
31. D. Whitley, J.P. Watson, A. Howe, and L. Barbulescu. Testing, evaluation and performance of optimization and learning systems. Technical report, The GENITOR Research Group in Genetic Algorithms and Evolutionary Computation, Colorado State University, 2002.
32. L. Ingber and B. Rosen. Genetic algorithms and very fast simulated reannealing: A comparison. *Mathematical Computer Modelling*, 16(11):87–100, 1992.
33. Fred Glover and M. Laguna. Tabu search. In C. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, Oxford, England, 1993. Blackwell Scientific Publishing.
34. 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.
35. G. E. P. Box and N. R. Draper. *Experimental Model Building and Response Surfaces*. Wiley, 1987.
36. A.M. Law and W.D. Kelton. *Simulation Modelling and Analysis*. McGraw-Hill Series in Industrial Egineering and Management Science. McGraw-Hill, New York, 3rd edition, 2000.
37. J. P. C. Kleijnen. Experimental designs for sensitivity analysis of simulation models. In A. W.Heemink et al., editor, *Proceedings of EUROSIM 2001*, 2001.
38. J. M. Chambers and T. H. Hastie, editors. *Statistical Models in S*. Wadsworth & Brooks/Cole, Pacific Grove, California, 1992.
39. P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman and Hall, 2nd edition, 1989.
40. N. R. Draper and H. Smith. *Applied regression analysis*. Wiley series in probability and statistics. Wiley, New York, 3rd edition, 1998.
41. R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5(3):299–314, 1996.
42. D. Collett. *Modelling Binary Data*. Chapman and Hall, 1991.
43. T. Beielstein and C. Feist. Jeat – a JAVA based test frame for tuning and comparing algorithms. Technical report, Universität Dortmund, 2003. (to appear).