Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation

Ingo Paenke, Jürgen Branke, Yaochu Jin

2006

Preprint:

This is an accepted article published in IEEE Transactions on Evolutionary Computation. The final authenticated version is available online at: https://doi.org/[DOI not available]

Efficient Search for Robust Solutions by Means of Evolutionary Algorithms and Fitness Approximation

Ingo Paenke, Jürgen Branke, *IEEE, Member* and Yaochu Jin, *IEEE, Senior Member*

***Abstract*— For many real-world optimization problems, the robustness of a solution is of great importance in addition to the solution's quality. By robustness, we mean that small deviations from the original design, e.g., due to manufacturing tolerances, should be tolerated without a severe loss of quality. One way to achieve that goal is to evaluate each solution under a number of different scenarios, and use the average solution quality as fitness. However, this approach is often impractical, because the cost for evaluating each individual several times is unacceptable. In this paper, we present a new and efficient approach to estimating a solution's expected quality and variance. Basically, we propose to construct local approximate models of the fitness function, and then use these approximate models to estimate expected fitness and variance. Based on a variety of test function, we demonstrate empirically that our approach significantly outperforms the implicit averaging approach as well as the explicit averaging approaches using existing estimation techniques reported in the literature.**

***Index Terms*— Evolutionary Optimization, Uncertainty, Robustness, Fitness Approximation.**

I. Introduction

In many real world optimization scenarios, it is not sufficient for a solution to be of high quality, but the solution should also be robust. Some examples include

- In manufacturing, it is usually impossible to produce an item exactly according to the design specifications. Instead, the design has to allow for manufacturing tolerances, see e.g., [2], [14], [39].
- In scheduling, a schedule should be able to tolerate small deviations from the estimated processing times or be able to accommodate machine breakdowns [17], [23], [32].
- In circuit design, the circuits should work over a wide range of environmental conditions like different temperatures [36].
- In turbine blade design, the turbine should perform well over a range of conditions, e.g., it should work efficiently at different speeds. Similar requirements exist for airfoil design [31], [40].

There are a number of different possible definitions for robustness (see e.g., [6], p. 127). Generally speaking, robustness means some degree of insensitivity to small disturbances of the environment or the design variables. One definition for robust solutions is to consider the best worst-case performance. Another definition of robust solutions is to consider a solution's *expected* performance over all possible disturbances, which corresponds to a risk-neutral decision maker's choice. In these two definitions for robustness, only one objective is considered, and we denote such approaches *single objective (SO) robustness optimization*. However, robustness of solutions might be better defined by considering both the quality and the risk separately, i.e., by converting the problem into a multi-objective problem [22]. We denote such approaches as *multi–objective (MO) robustness optimization*. This paper suggests model-based fitness approximation methods that can be employed to improve the computational efficiency of both SO and MO approaches to robustness optimization.

Disturbances may appear in both *environmental variables* and *design variables*. In the following, we focus on robustness against disturbances of design variables, which is important, e.g., in the case of manufacturing tolerances. Formally, if $\mathbf{x}$ denotes a design vector (solution) of dimension d, and $f(\mathbf{x})$ is the fitness (in the context of robustness optimization $f(\mathbf{x})$ is also often called *raw* fitness, f_{raw}) of that particular solution, then the *expected* fitness of solution $\mathbf{x}$ is defined as

$$f_{\text{exp}}(\mathbf{x}) \quad = \quad \int_{-\infty}^{\infty} f(\mathbf{x}+\delta) \cdot p(\delta) d\delta, \qquad (1)$$

where δ is a disturbance that is distributed according to the probability density function $p(\delta)$. Similarly, the fitness variance of a solution can be defined as

$$f_{\text{var}}(\mathbf{x}) \quad = \quad \int_{-\infty}^{\infty} \left(f(\mathbf{x}+\delta) - f_{\text{exp}}(\mathbf{x})\right)^2 \cdot p(\delta) d\delta. \quad (2)$$

Unfortunately, for reasonably complex problems, Equations (1) and (2) cannot be computed analytically, usually because f is not known in a closed form. Alternatively, f_{exp} and f_{var} can be estimated by Monte Carlo integration, i.e., by sampling over a number of realizations of δ. However, each sample corresponds to one fitness evaluation, and if fitness evaluations are expensive, this approach is clearly not viable.

Therefore, new approaches are needed which allow to estimate a solution's expected fitness and variance more efficiently. In this paper, we propose to use an approximation model to estimate a solution's robustness. Instead of using the costly raw fitness function in the above mentioned Monte Carlo integration, we rely on the approximation model for that purpose. In principle, this idea could be used in combination with any suitable approximation model like artificial neural networks, kriging models, or Gaussian processes. In this paper, we use local approximation models, which have the advantage of being relatively easy to construct and also seem appropriate to approximate the performance of a solution over a distribution of local disturbances. Within that framework,

Ingo Paenke and Jürgen Branke are with the Institute AIFB, University of Karlsruhe, Germany. Yaochu Jin is with the Honda Research Institute Europe, 63073 Offenbach, Germany. Authors contributed equally.

we compare and discuss a number of alternatives w.r.t. the approximation model used (interpolation or regression), the complexity of the model (linear or quadratic), the number and location of approximation models constructed, the sampling method, and how approximation methods should be exploited for estimation. Empirical results confirm the superiority of our approach to some previous approaches for either SO or MO robustness optimization.

Note that we subsequently use the terms *raw fitness* and *real fitness*. Here raw fitness is used in contrast to robustness (f_{exp}, f_{var}), whereas real fitness is used in contrast to approximated fitness.

The paper is structured as follows. Section II provides a brief overview of related work. We then introduce the evolutionary algorithm (EA) for robustness optimization in Section III. A short discussion of the approximation techniques used can be found in Section IV. Then, in Section V, we present our new approaches to estimating a solution's expected fitness and variance. These approaches are evaluated empirically in Section VI based on a variety of benchmark problems. The paper concludes with a summary of this paper and some ideas for future work.

II. Related Work

There is a wealth of publications regarding the use of approximation models to speed up EAs. Feedforward neural networks [16], [21], radial basis function networks [33], and polynomials [7], [24] have been employed to improve the efficiency of EAs. Besides, estimation of distribution algorithms (EDAs) can also be considered as a class of algorithms that approximate the fitness landscape implicitly [41]. In the following, we will focus on related literature regarding the search for robust solutions. For a general overview on the use of approximation models in combination with EAs, the reader is referred to [18], [19].

As has been mentioned in the introduction, evolutionary approaches to robustness optimization can be categorized into single-objective (SO) and multi-objective (MO) optimization approaches. By far, the majority of research activities in this field follows the SO approach.

A. SO robustness optimization

An analytical expected fitness function is often not available, therefore, it is necessary to estimate a solution's expected fitness. The probably most common approach is to sample a number of points randomly in the neighborhood of the solution $\mathbf{x}$ to be evaluated, and then take the mean of the sampled points as the estimated expected fitness value of $\mathbf{x}$ (see, e.g., [4], [14], [39]). This straightforward approach is also known as *explicit averaging*. The explicit averaging approach needs a large number of additional fitness evaluations, which might be impractical for many real-world applications. To reduce the number of additional fitness evaluations, a number of methods have been proposed in the literature:

1) **Variance reduction techniques:** Using derandomized sampling techniques instead of random sampling reduces the variance of the estimator, thus allowing a more accurate estimate with fewer samples. In [5], [26], *Latin Hypercube Sampling* is employed (cf. Appendix B), together with the idea to use the same disturbances for all individuals in a generation.
2) **Evaluating important individuals more often:** In [4], it is suggested to evaluate good individuals more often than bad ones, because good individuals are more likely to survive and therefore a more accurate estimate is beneficial. In [6], it was proposed that individuals with high fitness variance should be evaluated more often.
3) **Using other individuals in the neighborhood:** Since promising regions in the search space are sampled several times, it is possible to use information about other individuals in the neighborhood to estimate an individual's expected fitness. In particular, in [4] it is proposed to record the *history* of an evolution, i.e. to accumulate all individuals of an evolutionary run with corresponding fitness values in a database, and to use the weighted average fitness of neighboring history individuals. Weights are assigned according to the probability distribution function of the disturbance. We will use this method later for comparison and refer to it as *Weighted history*.

While all of the above methods *explicitly* average over a number of fitness evaluations, Tsutsui and Ghosh present in [37], [38] an idea to simply disturb the phenotypic features before evaluating an individual's fitness. As the EA is revisiting promising regions of the search space, it implicitly averages over a set of disturbed solutions, which can be seen as an *implicit averaging* approach. Using the schema theorem, Tsutsui and Ghosh show that given an infinitely large population size and the proportional selection method, a genetic algorithm with single disturbed evaluations is actually performing as if it would work on f_{exp}. This implicit averaging has proven successful for low-dimensional problems. Subsequently we refer to this approach as *Single disturbed*.

B. MO robustness optimization

In design optimization, several papers have treated the search for robust optimal solutions as a MO problem, see e.g. [8], [9]. However, relatively little attention has been paid to evolutionary MO search for robust solutions. Ray [30], [31] considers robust optimization as a three-objective problem, where the raw fitness, expected fitness and the standard deviation are optimized simultaneously. In that work, a large number of additional neighboring points are sampled to estimate the expected fitness and standard deviation. In [22], search for robust optimal solutions is considered as a trade-off between optimality (the raw fitness) and robustness, which is defined as the ratio between the standard deviation of fitness and the average of the standard deviation of the design variables. To estimate the robustness measure, the mean fitness and the standard deviation are estimated using neighboring solutions in the current generation, without conducting any additional fitness evaluations, but only using the individuals in the current generation to estimate the local fitness variance. This becomes feasible because the local diversity of the population

is maintained by using the dynamic weighted aggregation method [20] for multi-objective optimization.

C. Main Contributions of the Paper

One of the main research efforts in evolutionary search for robust solutions is to reduce the number of computationally expensive fitness evaluations. So far, the main idea has been to calculate the mean fitness in the SO the approach [4], [6] or the fitness variance in the MO approach [22] directly based on the neighboring solutions in the current population or in the entire history of evolution.

Using the existing neighboring solutions to calculate the mean and variance of the fitness is only a very rough approximation of the Monte Carlo Integration. To address this problem, this paper suggests to construct computationally efficient models using available solutions to replace the expensive fitness function in calculating the mean and variance of fitness values. If the model is sufficiently good, we can estimate the mean and variance much more reliably using the Monte Carlo method. Both interpolation and regression methods in combination with a variety of model distribution techniques, such as single model, nearest model, ensemble, and multiple models are investigated. The effectiveness of using models to estimate the mean and variance of fitness values are verified on 6 test problems for the SO approach and 3 test problems for the MO approach to robust optimization.

III. Evolutionary Algorithm for Robustness Optimization

The evolutionary search for robust solutions proposed in this paper uses approximation models to estimate a solution's expected fitness as well as the variance without additional fitness evaluations. While in principle, this idea is independent of the approximation model, we use local approximations of the fitness surface [24], [33]. Training data for the approximation model are solely collected online, i.e., the EA starts with an empty *history* and collects data during the run. In each generation, the real fitness function is evaluated at the location of the current individuals, these data are then stored in a database which we denote *history*. See Algorithm 1 for the pseudo-code of the algorithm. With this data collection strategy, the total number of real fitness evaluations equals the number of generations times population size, which is the same as required for a standard EA. Thus, additional fitness evaluations needed for robustness evaluation is avoided.

In robustness optimization, the question on how to deal with constraints is a challenging research topic. Compared to optimization based on the raw fitness, a solution is no longer strictly feasible or infeasible in search for robust solutions. Instead, it might be feasible with a certain probability. Promising approaches to handle constraints are presented in [26], [30]. However, a further discussion of this topic is beyond the scope of this paper. In the EA used in this paper, we simply bounce off the boundary if a an individual would lie outside the parameter range. Samples drawn from an infeasible region are set to a bad constant value.

Algorithm 1 Robustness EA

$t \leftarrow 0$
initialize population $P(0)$
evaluate $P(0)$ with real f
add $P(0)$ to the *History*
estimate f_{exp} (f_{var}) of $P(0)$ individuals
REPEAT
  selection: mating pool $M(t) \leftarrow select(P(t))$
  recombination: $M'(t) \leftarrow recombine(M(t))$
  mutation: $M''(t) \leftarrow mutate(M'(t))$
  evaluate $M''(t)$ with real f
  add $M''(t)$ to the *History*
  estimate f_{exp} (f_{var}) of $M''(t)$ individuals
  update-population: $P(t+1) \leftarrow u(P(t) \cup M''(t))$
  $t \leftarrow t+1$
UNTIL termination criterion met

IV. Fitness Approximation

A. Interpolation and Regression Techniques

We attempt to estimate the expected fitness and the variance of each candidate solution based on an approximate model that is constructed using history data collected during the optimization. In this way, no additional fitness evaluations are needed. For fitness approximation, we use interpolation and local regression. In the following, we provide a short description of interpolation and regression techniques. This kind of models has been used in [24] to smooth out local minimums in evolutionary optimization. Readers are referred to [25] for further details on interpolation and regression techniques.

A quadratic polynomial interpolation or regression model can be described as follows:

$$f(\mathbf{x}) = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \sum_{i=1}^{d} \sum_{j=1}^{d} \beta_{d-1+i+j} x_i x_j, \qquad (3)$$

where d is the dimensionality of $\mathbf{x}$. Equation (3) can be rewritten as:

$$f(\mathbf{x}) = \sum_{j=1}^{n_c} X_j \beta_j, \qquad (4)$$

where $X = [1\ x_1\ ...\ x_d\ x_1x_2\ ...\ x_d\ x_1^2\ ...\ x_d^2]$ and β is the vector of model coefficients of length $n_c = (d+1)(d+2)/2$. Typically, we have a set of $n_{in} \geq n_c$ training data $(X^{(1)}, y^{(1)})$, $(X^{(2)}, y^{(2)})$,..., $(X^{(n_{in})}, y^{(n_{in})})$, and the goal is to find model parameters β that minimize the error on the training data.

The most popular estimation method is the *least square method*, which minimizes the residual sum of squared errors:

$$\min J = \sum_{i=1}^{n_{in}} (y^{(i)} - f(X^i))^2, \qquad (5)$$

Additionally, weights can be assigned to the residual distances, i.e.,

$$\min J = \sum_{i=1}^{n_{in}} w_i (y^{(i)} - f(X^i))^2, \qquad (6)$$

where w_i is a weight for the i-th training sample. This is also known as *local regression*. As training data we choose

all history data which lie within the range of the disturbance distribution of the estimation point, i.e. n_{in} can be different for different approximation points. If, however, n_{in} is smaller than the desired minimum number of training data (as specified beforehand) data points from outside the disturbance range are used, too (cf. Table II in the simulation studies section). Weights are usually assigned w.r.t. the distance between the location of the training sample point and the so-called **fitting point** around which the model is constructed (denoted as $\mathbf{x}_{fp}$ hereafter). As weight function w_i, the *tricube function* is used (Equation 7), where b denotes the bandwidth that is chosen such that it covers all model input data:

$$w_{\text{tricube}}(x^{(i)}, x^{(0)}) = \left(1 - \left(\frac{\parallel x^{(i)} - x^{(0)} \parallel_2}{b}\right)^3\right)^3. \quad (7)$$

Solving Equation (6), we get:

$$\beta^* = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \quad (8)$$

where $\mathbf{W} > 0$ is a diagonal matrix with w_i as diagonal elements, $\mathbf{X} = [X^{(1)}\ X^{(2)}\ ...\ X^{(n_{in})}]^T$, and $\mathbf{y} = [y^{(1)}\ y^{(2)}\ ...\ y^{(n_{in})}]$.

Note that to fully determine the parameters in the model, it is required that the number of training data n_{in} be equal to or larger than the number of coefficients n_c. The special case of $n_{in} = n_c$ represents interpolation. In this case, a β^* exists such that residual error in Equation (6) can be reduced to zero, i.e., the approximate function intersects all training data points. We need to find a β such that Equation (9) is true:

$$\mathbf{X}\beta = \mathbf{y}. \quad (9)$$

By inverting $\mathbf{X}$ we get the solution

$$\beta^* = \mathbf{X}^{-1}\mathbf{y}. \quad (10)$$

Our motivation to distinguish between interpolation and regression is that a surface which is generated by an interpolation model intersects the nearest available real fitness points whereas regression aims at smoothening a given data set. With a deterministic fitness function, there is no need of smoothening because there exists no noise that can be removed by regression. However, regression has the advantage of covering a larger neighborhood space.

From the large number of available interpolation methods [1] we decided to use one of the simplest methods, which chooses the nearest available history points as training data to construct the model. As a result, some of the points on the approximated surface are interpolated, and some are actually extrapolated ([29], Chapter 3). Nevertheless, we denote this method *interpolation* throughout this paper. This type of interpolation may return a discontinuous surface. A method that generates a continuous landscapes is *Natural neighbor interpolation* [35], which chooses the training data such that the model fitting point lies within the convex hull defined by the training data. The drawback of natural neighbor interpolation is its high computational complexity, particularly when the dimension of the design space is high. Thus, we use the standard interpolation method which uses the nearest neighbors, in this paper. For local regression, we choose the nearest available history data as training data for the model, too. Again, the distance is measured with respect to the model fitting point.

Successful interpolation and regression requires $\mathbf{X}$ to have a rank of n_c, i.e., n_c of the training samples have to be linearly independent. This might not always be the case, in particular when the EA converges. If, in regression, we find $\mathbf{X}$ to be singular, we simply add the nearest available data points (which are not yet used) to the model, and check again for linear dependencies. This loop is repeated until X has a full rank. In this case, Equation (8) is solved by Cholesky decomposition [29]. In interpolation, however, linearly dependent training data points in $\mathbf{X}$ need to be detected and replaced by other points. Therefore, we first need to check for linear dependencies before solving Equation (10). In our methods, this is done using QR decomposition with column pivoting [13]. If $\mathbf{X}$ has a full rank, Equation (10) can be solved by LU decomposition [13], otherwise this loop continues until $\mathbf{X}$ has a full rank.

It should be pointed out that when interpolation is used, it is possible to produce severely incorrect estimations in situations when the history data are ill-distributed, for example, when some of the nearest neighbors are located very close to each other compared to their distance to the model fitting point on a steep part of the fitness function. In order to reduce the bias introduced by such wrong estimations, we need to detect severe outliers. In particular, history data are sorted with regard to their fitness value. An approximation $\widehat{f}(\mathbf{x}_s)$ at a sample point $\mathbf{x}_s$ is defined as an outlier, if

$$\widehat{f}(\mathbf{x}_s) \notin [H_{0.01} - 5\,(H_{0.99} - H_{0.01})\,;\, H_{0.99} + 5\,(H_{0.99} - H_{0.01})], \quad (11)$$

where $H_{0.01}$, $H_{0.99}$ represent the 0.01 and 0.99 quantiles of the sorted history fitness values. This detection method is common in the realm of box plots. Outliers are replaced by the average real fitness of the current population (which is available at that time, cf. Algorithm 1). Theoretically, this method can cut off extremely good (correctly estimated) individuals, however, with the setting as in Equation 11 this is very unlikely. In our experiments, it is found that cutting off extreme estimated fitness values leads to more reliable results.

B. Computational Complexity

The motivation to use approximate models instead of additional samples is that in many applications the computational cost of a fitness evaluation is larger than the cost of building up an approximate model. In the following we briefly present a theoretical analysis of the computational overhead for the approximate models used in this paper, namely interpolation and regression.

The three steps for building up an approximation model are:

1) **Calculate the Euclidean distance** between the model fitting point and all available history training data (n_h). Since the Euclidean distance can be computed in linear time (w.r.t. the dimensionality d), the overall cost are in the order $\mathrm{O}(n_h d)$.

2) **Sort training data** w.r.t. to the Euclidean distance. The complexity of the *Quicksort* algorithm [15] is in the best case $\mathrm{O}(n_h \log_2 n_h)$ and the worst case $\mathrm{O}(n_h^2)$.
3) **Computing the interpolation / regression polynomial**. In interpolation, the most expensive element is QR decomposition which requires $\mathrm{O}(n_c^3)$ *flops* [1]. In regression the most expensive element is Cholesky decomposition which requires $\mathrm{O}(n_c^3)$ flops, where n_c is the number of model coefficients. This means both approximation methods have a complexity of $\mathrm{O}({n_c}^3)$.

The overall complexity for building up one approximation model sums up to

$$\mathrm{O}(n_h d + n_h^2 + {n_c}^3). \tag{12}$$

In case of singularities, matrix $\mathbf{X}$ is modified and step 3 is repeated, thus, the computational cost increases. On a state-of-the-art personal computer, the computation time is in the milisecond order, which can be regarded as negligible compared to expensive fitness evaluations of many real-world problems. For example, a computational fluid dynamics simulation for blade design optimization takes often from tens of minutes to several hours. Of course, the computation time can no longer be negligible if a more complex model is constructed with a large number of samples.

V. Robustness Estimation

Since we cannot expect the overall fitness function to be of linear or quadratic nature, any linear or quadratic approximation model is usually only a local approximation. Thus, for estimating the fitness at different points in the search space, different approximation models have to be constructed. In this section, we discuss the questions of *where* approximation models should be constructed, *how many* should be constructed, and *how they should be used* to estimate f_{exp} and f_{var} of all individuals of the population.

A. Integral approximation

The integrals of Equation (1) are estimated for a given point $\mathbf{x}^0$ by evaluating (w.r.t. the *approximated* fitness function $\widehat{f}$) a set of n samples $\mathbf{x}_i = \mathbf{x}^0 + \delta_i$ in the neighborhood of $\mathbf{x}^0$. We get the estimations

$$\begin{aligned} \widehat{f}_{\text{exp}}(\mathbf{x}^0) &= \sum_{i=1}^{n} \frac{1}{n} \widehat{f}(\mathbf{x}_i), \\ \widehat{f}_{\text{var}}(\mathbf{x}^0) &= \sum_{i=1}^{n} \frac{1}{n} [\widehat{f}(\mathbf{x}_i) - \widehat{f}_{\text{exp}}(\mathbf{x}^0)]^2. \end{aligned} \tag{13}$$

To generate the samples x_i, we use *Latin hypercube sampling* (refer to Appendix B) which has proven to be the most accurate sampling method given a limited number of sample points [5]. In *Latin hypercube sampling*, the number of samples solely depends on the number of quantiles and is independent of the dimensionality, thus the size of the sample set can be arbitrarily scaled.

[1] fbp - fbating point operation, i.e., one addition, subtraction, multiplication, or division of two fbating-point numbers

B. Model distribution

As has been explained above, we estimate a solution's robustness based on the estimated fitness of a number of sampled points in the neighborhood. Since local approximation models are only reliable in a small neighborhood of their fitting point, several models are needed to evaluate a population. One important question therefore is where to construct the approximation models. In principle, one might attempt to place them at strategically favorable positions, so that a maximum accuracy can be obtained with a minimum number of models. In this paper, we used two simple strategies: to construct one model around each individual in the population, and to construct one model around each sample point. While the latter promises to be more accurate, it requires to build many more models and therefore demands for much higher computational resources.

The next question is how the models are used for estimating the fitness of a particular sample. We have tested three possible approaches. First, one might use the model constructed around an individual to estimate the fitness of all samples used to evaluate this individual. Second, one can use the *nearest* model (i.e., where the Euclidean distance to the model fitting point is minimal) for each sample, because that model probably has the highest accuracy. And finally, one might combine the estimates of several models in the hope that the estimation errors of the different models cancel out.

Overall, we have tested the following four different settings:

- **Single model:** In this straight-forward approach, we build one approximation model per individual, i.e. the models' *fitting points* are the same as the individuals' locations in the search space, and we use this model for all sample points generated to estimate that individual's fitness. This approach, of course, assumes that the simple linear or quadratic models are sufficient to approximate the raw fitness function within the range of expected disturbances δ.
- **Nearest model:** Again, one model is constructed around each individual, but we always use the nearest model to estimate the fitness of a sample. Note that the nearest model can be that of a neighboring individual and is not always the one of the associated individual (which can have a greater distance).
- **Ensemble:** This approach is also based on one model constructed around each individual. However, we estimate the function value at a particular sample point $\mathbf{x}_s$ by a weighted combination (ensemble) of models that correspond to the m nearest fitting points. The approximated functional value is calculated as follows:

$$\begin{aligned} \widehat{f}_{\text{ENS}}(\mathbf{x}_s) &= \frac{1}{\sum_{1 \le i \le m} v_i} \sum_{1 \le i \le m} v_i \, \widehat{f}_i(\mathbf{x}_s) \\ v_i &= \frac{1}{\|\mathbf{x}_{fp} - \mathbf{x}_s\|_2} \quad , \end{aligned} \tag{14}$$

  where v_i is a weight function and m the ensemble size.
- **Multiple models:** In this approach, a separate model is constructed around each sample, and exactly this model is used to estimate the sample's fitness.

The first three methods share the same advantage that the number of approximation models needs to be constructed is small. In fact, they even use the same model locations, namely, exactly one model at the location of each individual in the population.

Fig.1 illustrates the difference between the *Single model* and *Multiple models* approaches for 1-dimensional case using a linear interpolation model. It can be seen that in this example, a single interpolation model cannot fully describe the local fitness landscape that is given by the history data.

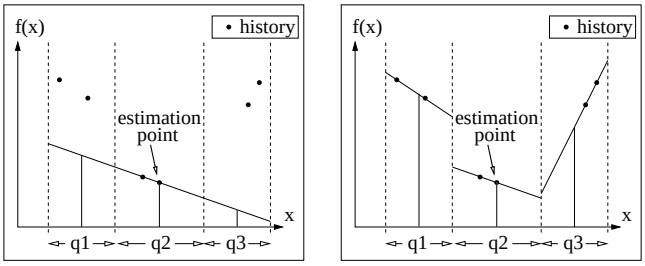

Fig. 1. Illustration of *Single model* (**left**) and *Multiple models* (**right**) for the 1-dimensional case. The quantiles of the δ distribution are denoted $q1$, $q2$ and $q3$. The dotted lines represent boundaries of the quantiles. Samples are drawn from the center of each quantile and evaluated with the approximation model.

C. Estimator properties

In general, a desired property of an estimator is that the model is as accurate as possible, i.e., the estimation error $e = |\widehat{f}(\mathbf{x}^0) - f(\mathbf{x}^0)|$ is minimal. In most applications, it is also desired that the estimator is *unbiased*, i.e., the *expected* estimation error is zero, $E(\widehat{f}(\mathbf{x}^0)) = f(\mathbf{x}^0)$. In the context of an EA, however, a low standard deviation of the estimation error σ_e seems more important, provided that the biases on different points are consistent. The following example illustrates the point: Assume that for a given estimator, e has a probability distribution, with mean μ_e and standard deviation σ_e. Consider the extreme case $\sigma_e = 0$: With rank-based selection, an EA performs exactly as if the real fitness is used independent of μ_e, but even with fitness proportional selection the influence of μ_e on the composition of the next generation is low. We conclude that the standard deviation of the estimation error is the important estimator property in the context of EAs. See [16] for more discussions on error measures for models in fitness approximation.

D. Computational Complexity

In Section IV-B we calculated the computational cost of building a single approximation model. Using a number of approximate models to estimate robustness incurs additional computational cost. The computational cost using the four proposed model distribution methods varies. Due to space limitations, we do not present a detailed analysis for each method, but briefly list the cost components of which each robustness estimation is composed.

- **Building up approximation models.** The cost for building one approximation model as given in Equations 12 are to be multiplied by the number of models that are needed per estimation. In case of Single model, Nearest model and Ensemble, only 1 model is built per estimation whereas in Multiple models n models are built (cf. Equation 13).
- **Constructing the Latin hypercube set** is possible in linear time, i.e. $\mathrm{O}(n)$ where n is the number of samples.
- Cost incurred by **evaluating polynomials**: For a single polynomial the complexity is $\mathrm{O}(n_c)$ where n_c is the number of model coefficients. *How many* polynomials are to be evaluated depends on the number of sample points n and the model distribution method, i.e. in case of Single model, Nearest model and Multiple models, 1 polynomial is evaluated per sample point, but in case of Ensemble a set of m polynomials is evaluated, where m is the ensemble size.
- **Calculating $\widehat{f}_{\mathbf{exp}}(x)$, or** $(\widehat{f}_{\mathbf{exp}}(x), \widehat{f}_{\mathbf{var}}(x))$, i.e. averaging (the square) is done in linear time w.r.t. the sample size, i.e. $\mathrm{O}(n)$ (cf. Equation 13).
- **Additional cost:** In case of Nearest model, additional calculations are to be done in order to determine the nearest model. In Ensemble additional cost is incurred because the m nearest models need to be found and the averaging over the polynomial evaluation results needs to be done.

VI. Simulation Studies

The first goal of the simulation studies is to investigate empirically whether fitness approximation is able to effectively guide the evolutionary search for robust solutions. Additionally, we compare performance of interpolation and regression for fitness approximation. Another interesting issue in fitness approximation is the influence of model distribution on the performance of fitness approximation and thus on the search effectiveness. Finally, we compare our methods to the previously proposed methods *Single disturbed* [37], [38] for SO robustness optimization and *Weighted history* [4], [22] for both SO and MO robustness optimization (cf. Section II). Note that in the MO case *Weighted history* estimates both f_{exp} and f_{var} empirically by evaluating the nearest neighbors of the history. This is an extension of the original MO method [22] because here the estimate is calculated based on the current population only, and no weighting is used. Preliminary experiments showed that adding these two features to the method improves the performance. Additionally, we run the EA using the *raw fitness* as optimization criterion, i.e. the EA is run without robustness scheme. Since f_{exp} is different from the raw fitness optimum, this setting is expected to have poor performance, and only serves for reference purposes. Finally, we run the EA, estimating robustness by evaluating samples with the real fitness instead of approximations. In other words, the EA has a very good estimator for the *real* f_{exp} as defined in Equation 1. We denote this *real* f_{exp} although this is strictly speaking only a very good estimation. This method of course requires a large number of real fitness function evaluations. For example in our 5-dimensional test problems, it requires 250000 fitness evaluations, which is 50 times (the number of

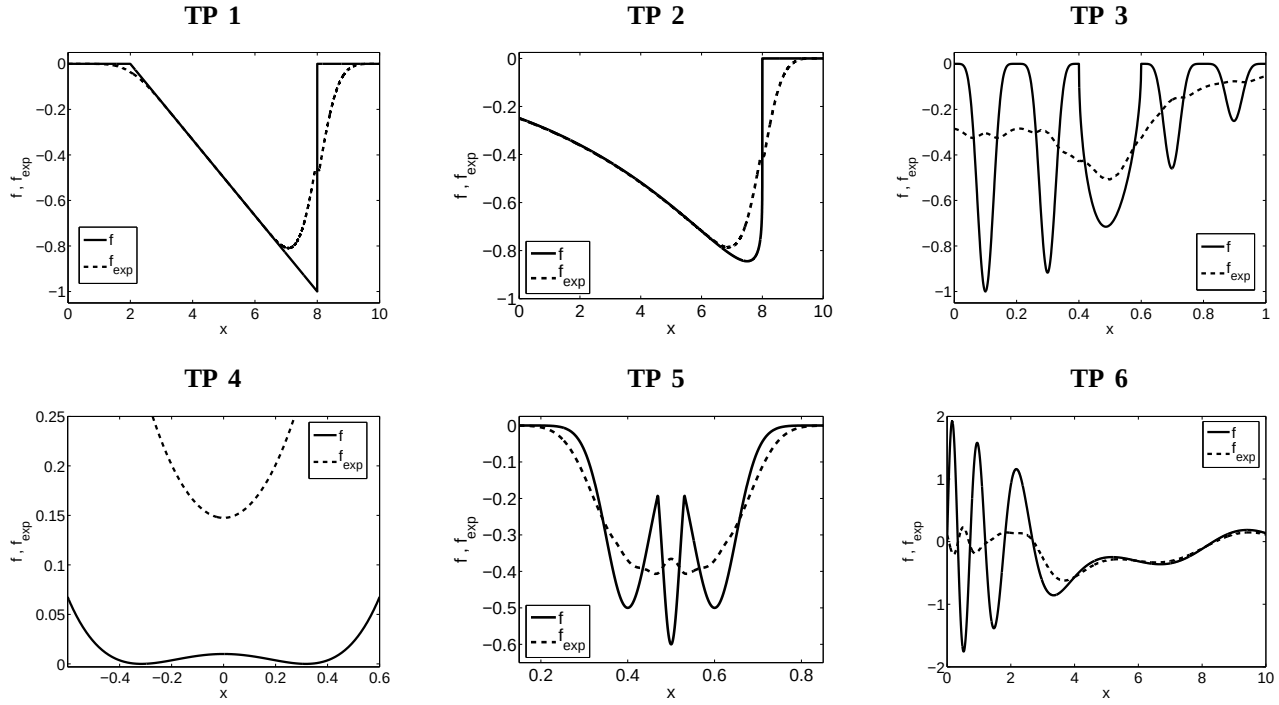

Fig. 2. 1-dimensional test problems for SO robustness optimization (TP 1-6). Figures show f and f_{exp} (in TP 4 we zoom into the interesting area).

samples) higher than when approximation models are used. This becomes infeasible in many real-world applications. For our test problems, however, it provides a good reference of the performance that our approximation methods could reach. For clarity, Table I summarizes all compared methods.

TABLE I
SUMMARY OF COMPARED METHODS

New techniques (linear and quadratic , interpolation and regression)	
SM	Single model
ENS-5	Ensemble method with ensemble size 5
NEAR	Nearest model
MM	Multiple models
Benchmark techniques used for comparison	
$f$raw	Raw fitness optimization (without robustness scheme)
Single disturbed	Individual is disturbed before evaluation
Weighted history	Estimation of f_{exp} based on the weighted mean of (previously evaluated) neighbors
real f_{exp}	Estimation of f_{exp} based on real fitness function evaluations (latin hypercube sampling)

A. Experimental Settings

1) Test Problems: To compare different algorithms for solving robust optimization, a number of test problems (TPs) are suggested in this paper. We identify four categories of TPs for SO robustness according to the fitness landscape change from the raw fitness to the effective fitness. Additionally, three TPs are designed for MOO approach to robust solutions, of which TP 7 has a continuous Pareto front and the TP8 has a discrete Pareto front. All TPs considered in this work are minimization problems and a detailed description of the TPs can be found in Appendix A.

In the following, we attempt to divide the TPs into four categories according to the differences between the raw and expected fitness landscapes.

- **Identical Optimum (Category 0):** Raw fitness and robust optimum are identical. Since these problems could be solved by simply optimizing the raw fitness, they are not really challenging. In the simulation studies we do not test problems of this category.
- **Neighborhood Optimum (Category 1):** Raw fitness and robust optimum are located on the same hill (w.r.t. raw fitness).
- **Local-Global-Flip (Category 2):** A raw fitness local optimum becomes the robust optimum.
- **Max-Min-Flip (Category 3):** The robust optimum (min.) is located at a raw fitness maximum.

The above categorization is not tailored for our approximation approach but illustrates challenges to robustness optimization in general. With regard to approximate models, another meaningful categorization might be to distinguish between **continuous** and **discontinuous** fitness landscapes, since the latter are expected to be more difficult to be approximated.

Now we present six test problems (TPs 1-6, see Fig.2) part of which are taken from the literature. All test problems are scalable to arbitrary dimensions. In this work, experiments are conducted on the TPs of dimensions 2, 5 and 10.

TP 1, which is taken from [6], is a discontinuous Category 1 test problem. Although it is uni-modal, the problem might be difficult to be approximated because of the discontinuity.

TP 2 is a continuous version of TP 1 and thus is a Category 1 problem, too. We expect the approximation methods to perform better on TP 2 than on TP1.

TP 3 is taken from [37] and is a variant of the function used

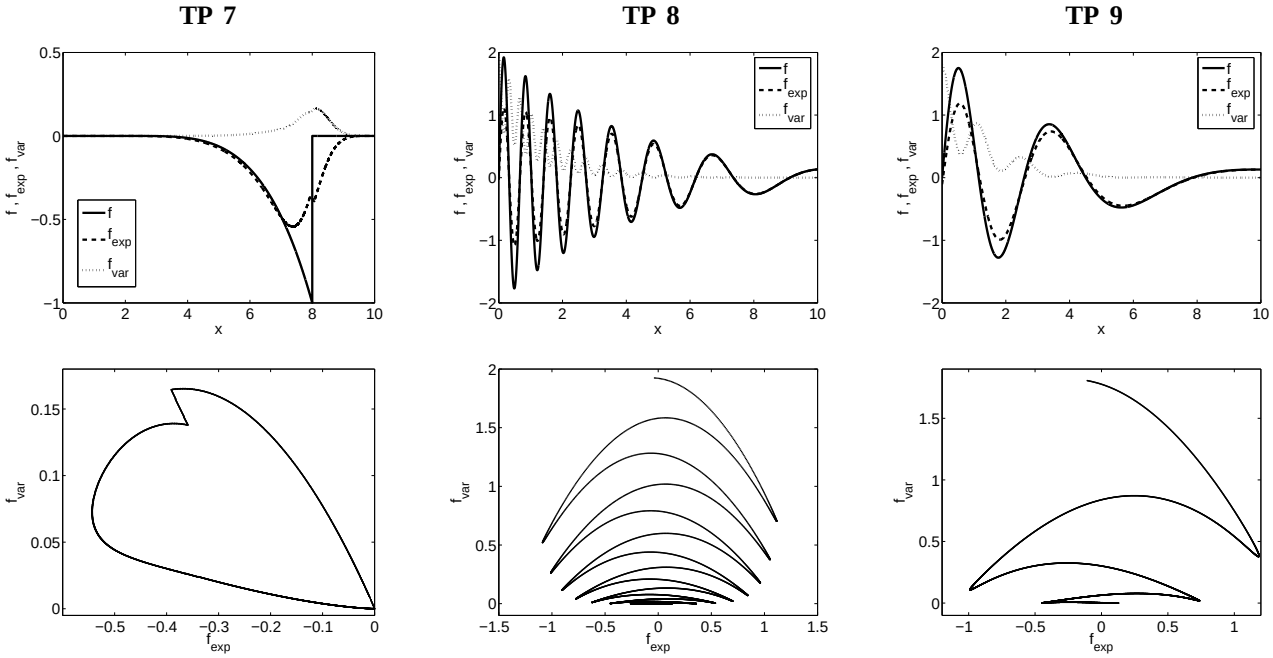

Fig. 3. 1-dimensional test problems for MO robustness optimization (TP 7-9). **upper row:** f, f_{exp}, f_{var}, **lower row:** trade-off between f_{exp} and f_{var}

in [11]. There are four sharp peaks and one broader peak. The global optimum for f_{exp} is located on the third (broad) peak, which is a local optimum in the raw fitness landscape. Thus, TP 3 is a Category 2 test problem. In [37], this test problem was tested for dimensions 1 and 2, in our simulation studies we will use this test function in up to 10 dimensions. In particular in higher dimensions, this test function becomes extremely difficult since the number of local optima equals 5^d.

TP 4 is multi-modal w.r.t. f, whereas the f_{exp} landscape is uni-modal [34]. In the 1-dimensional illustration (Fig.2), we see that the raw fitness optima (located on a d-dimensional sphere) are "merged" to a single robust optimum ($\mathbf{x}_i = 0$). Interestingly, the robust optimum (minimum) is a maximum in the raw fitness landscape. Therefore TP 4 is Category 3 test problem.

TP 5 is similar to TP 4, but here a single robust optimum is divided into multiple optima. Since the new robust optima are located where the raw fitness maxima are, TP 5 falls into Category 3, too.

TP 6 is a variant of the function used in [22]. When the feasible range of $x_i, i = 1, ..., d$, is restricted to $[0; 10]$, the optimum w.r.t. f_{raw} is at $\mathbf{x}_i = 0.5$, whereas the f_{exp} optimum is at $\mathbf{x}_i = 3.5$. Similar to TP 3, this test problem becomes very difficult for a large d. For TP 3, no clear assignment to one of the categories is possible, however, it combines aspects of TP 2 and TP 3 and can thus be seen as a mixed Category 1-Category 2 problem.

For MO robustness optimization we define problems that provide a trade-off between the first objective f_{exp} and the second objective f_{var}, i.e. problems with a Pareto front in a $f_{\text{exp}} - f_{\text{var}}$ space. Since MOEAs aim at finding a set of Pareto-optimal solutions, the test problems may categorized according to the continuity of the Pareto front. For MO approaches to robustness optimization, we carried out empirical studies on a set of 3 test problems (TPs 7-9), see Fig.3.

TP 7 is extremely simple w.r.t. the optimization of a single objective f or f_{exp}. For MO optimization with f_{exp} and f_{var}, it provides a *continuous* Pareto-front. The challenge to the MOEA here is to converge to a population which has a broad coverage of the Pareto front. Of course the difficulty increases with an increase of the dimensionality. In the simulation studies we set $d = 5$.

TP 8, which is taken from [22], provides a *discontinuous* Pareto-front. Since the number of separated Pareto-optimal solutions increases rapidly with the increase of the dimension, we used this test problem with $d = 2$.

TP 9 is a variant of TP 8 and has similar properties. The main difference is that the number of Pareto-optimal solutions is relatively lower. We used this test problem with $d = 5$.

2) Performance Measure: In the SO case, we choose the best individual of the final generation w.r.t. the approximated $\widehat{f}_{\text{exp}}$ as the final solution. Then, we re-evaluate f_{exp} of the final solution using the real fitness function and *Stratified Sampling* (see Appendix B) with a large number of samples to get a rather accurate estimate of the real expected fitness value. In the figures, we simply refer to this criterion as *fitness*. To reduce the influence of randomness, all reported results in the SO simulation studies are averaged over 20 runs with different random seeds. Statements about significance are based on 1-sided t-tests and 1-sided Fisher tests with a significance level of 97.5%.

A similar method in MO would have been to compute the set of non-dominated solutions based on the approximated fitness values of the final generation. However, the non-dominated solutions based on the approximation model may no longer be non-dominated when they are re-evaluated with

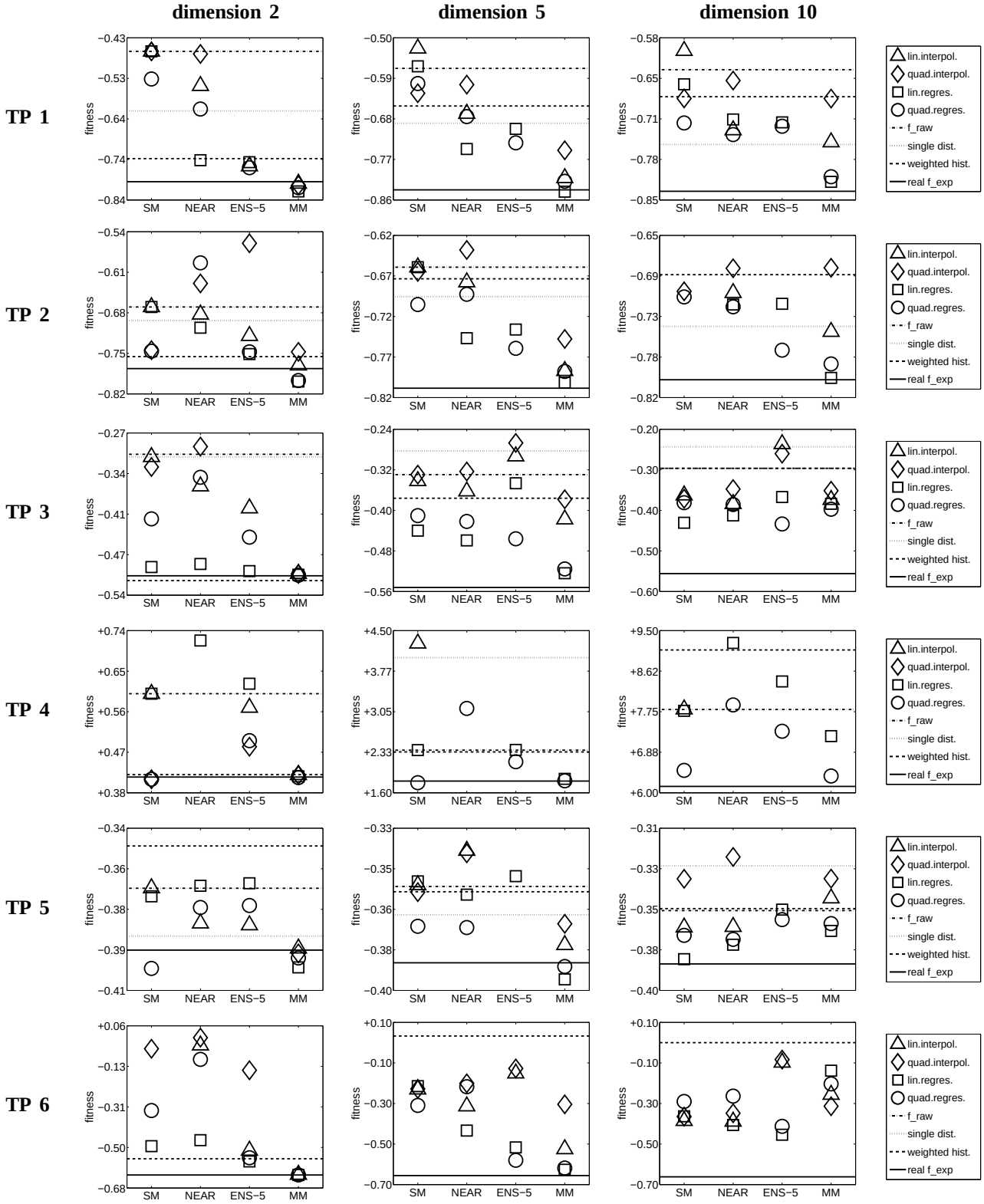

Fig. 4. Results of SO simulation studies (cf. Table I): All test problems (TP 1 - TP 6) in dimensions $2, 5, 10$, averaged over 20 runs. If symbols are missing in the figures, this indicates that the corresponding method's performance is worse than the largest fitness value on the respective axis.

the real fitness function. Therefore, we decided to use a simpler technique: We evaluate f_{exp} and f_{var} of the entire final generation using the real fitness and compute the non-dominated front based on these evaluations. As performance criterion, we plot the 50% *attainment surface* [12] based on the attainment surfaces of 11 runs. The 50% attainment surface can be interpreted as the *typical* result. We refer to it as *median attainment surface*. Since the *median attainment surface* only

allows a qualitative comparison, we additionally used a quantitative performance index. From the large number of proposed performance indices [28], we used the *hypervolume* [42].

3) Robustness Estimation Methods and Modeling Techniques: The compared modeling techniques are *linear interpolation*, *quadratic interpolation*, *linear regression* and *quadratic regression*. For robustness estimation the following four methods have been tested: *Single model* (SM), *Multiple models* (MM), *Nearest model* (NEAR) and the *Ensemble* method (ENS). For the ensemble method, a number of ensemble sizes have been tested. Considering all test problems, an ensemble size of 5 turned out to be most effective and stable. We therefore present in all figures *Ensemble* with an ensemble size of 5 (ENS-5). In the simulation, the sample size (n in Equation (13)) of Latin hypercube sampling was set to 10, 50 and 100 when the dimension equals 2, 5 and 10, respectively. Other related parameters are provided in Table II.

4) Evolutionary Algorithm: A conventional evolution strategy has been employed for SO search of robust optimal solutions, whose parameters are outlined in Table II. NSGA-II [10], which is one of the most efficient MOEAs, has been employed for the MO search of robust solutions and the parameters used in the simulations are listed in Table II. Note that instead of the simulated binary crossover (SBX) used in the original algorithm [10], the conventional 1-point crossover and mutation have been adopted. With these settings, the total number of calls to the real fitness function amounts to 5000 in our simulations.

TABLE II
EA PARAMETERS.

Parameters of the standard ES	
(μ, λ) - *reproduction scheme*	$(15, 100)$
standard evolution strategy	$\sigma_{init} \in [0.01; 1.0]$
recombination (obj. variables)	*discrete*
recombination (strat. variabl.)	*generalized intermediate*
no. generations	50
NSGA-II parameters	
representation	*gray coding*
number of bits (representation)	30
crossover probability	0.9
number of crossover points	1
flip probability (mutation)	0.01
population size	100
no. generations	50
Approximation parameters	
history size (max)	5000
regression bandwidth	*disturbance range*
regression min. training data	$2\times$ *no. model coeffi cients*
regression weight function	*tricube*

B. SO Results

All results of the SO simulation studies are presented in Fig.4. As can be seen, many of our new methods (geometric symbols) yield excellent performance on the 2- and 5-dimensional test problems (compared to the reference solution *real* f_{exp} denoted by the solid line). In dimension 10, however, our best methods fail to achieve a solution quality comparable to the case when using real f_{exp} on two multi-modal test problems (TP 3 and TP 6). This is to be expected, taking into account that the same number of fitness evaluations (5000) has been allowed independent of the number of dimensions.

1) Estimation methods: When comparing the different estimation methods, the results provide clear evidence that the *Multiple models* method works best. In the 2-dimensional problems the best regression method combined with *Single model* in most cases achieves a similar solution quality. However, only in one (TP 4) of the six 5-dimensional test problems, *Multiple models* does not outperform the other methods. In the 10-dimensional test problems the performance difference between *Multiple models* and the other techniques are reduced. On the one hand, this is because building a larger number of models yields more accurate fitness surface approximations only if the space is sufficiently covered with history data. Meanwhile, some of the 10-dimensional Test problems (TP 3,TP 4) seem to be too difficult to find a robust solution with the limited number of 5000 fitness evaluations. Here, none of the methods finds the global optimum. On most test problems, *Nearest model* is slightly better than the simple *Single model* method. Using an *Ensemble* of the 5 nearest models yields an additional benefit when using the (more stable) regression models.

As already discussed in Section V-C, a low standard deviation of the estimation error σ_e is expected to improve the performance of a fitness estimation based EA. This could be a reason why the *Ensemble* method performs better than the *Nearest model* method: Fitness approximations usually suffer from some approximation error. For estimating f_{exp}, models are evaluated at sample points and the f_{exp} estimation is a result of averaging. Technically speaking, the convolution of n_s approximation error distributions reduces σ_e. However, this assumes the n_s approximation error distributions to be statistically independent. This is not realistic because many approximation errors result from the same approximation model. In the *Single model* method, for instance, all n_s samples are evaluated with just a single model. But even in the *Multiple models* case, the models built at n_s sample points are likely to be equal or similar if the history data density is low. If statistical dependencies are present, σ_e is increased by the covariances of the approximation error distributions. The *Ensemble* method features an additional convolution that potentially reduces the standard deviation of the approximation error for a single sample. However, all ensembles are constructed based on the same set of available approximation models. Thus, the σ_e-reducing convolution effect is diminished by statistical dependence. Based on the above discussions, *Ensemble* would be expected to perform *at least* as good as *Nearest model*. However, *Ensemble* suffers from taking models into account with a fitting point that has a larger distance from the sample, which naturally increases σ_e. In our simulation studies, the σ_e reducing convolution effect of *Ensemble* seems to be dominating, since *Ensemble* (with 5 models) yields better results than the *Nearest model* method. This must be due to the convolution effect, because *Ensemble* and *Nearest model* make use of the same set of approximation models.

To summarize, the *Multiple models* method is the *most effective* and reliable modeling method on the tested problems, although at a relatively high cost, compared to *Ensemble* which

can be considered as extremely *efficient* in this case.

2) Approximation methods: From Fig.4, we find that the regression methods outperform the interpolation methods in most cases (the circle and square are below the diamond and triangle in the same column). Considering all 4 estimation methods on the 6 test problems in all dimensions (totals to $4 \cdot 6 \cdot 3 = 72$ scenarios), the two exceptions occur on 2-dimensional TP 4 when *Ensemble* method is used and on the 10-dimensional TP 6, which has shown to be too difficult for all methods. The superiority of regression can clearly be observed on the 5-dimensional test problems.

The main reason is that the interpolation methods are likely to produce severely wrong estimations: By counting the number of outliers, we found that interpolation is much more vulnerable to outliers. As a result, the standard deviation of the estimation error (σ_e) is larger in interpolation than in regression. This observation has been verified empirically in an additional experiment, where 1000 and 10000 data points were randomly generated in the 5-dimensional space. Based on these data sets, f_{exp} was estimated with different approximation methods. By running the experiment multiple times and comparing the estimations to the real f_{exp} we get an empirical distribution of the estimation error e. The resulting empirical standard deviation σ_e is presented in Table III for different approximation methods (TP 6, *Multiple models*, $d = 5$). We find a significant difference between the σ_e produced by interpolation and regression. These results show that regression

TABLE III

σ_e (TP 6, *Multiple models*, $d = 5$)

Hist.data	*lin.interp.*	*quad.interp.*	*lin.regr.*	*quad.regr.*
1000	0.38	0.47	0.22	0.30
10000	0.25	0.28	0.18	0.13

is clearly the preferred method in our simulations.

3) Approximation polynomials: Concerning the two polynomials used in the regression methods, no clear conclusion can be drawn on whether a linear or quadratic model performs better. In general, it would be expected that a quadratic regression model performs at least as good as a linear model because a linear polynomial is a subset of a quadratic. However, a quadratic model requires significantly more training data than a linear model. By adding data points of a larger distance to the model fitting point, the *local fitting* might become worse although polynomials of a higher order are used. With *Multiple models*, the model is only evaluated at its fitting point.

Whether to choose a linear or a quadratic model of course will depend strongly on the problem properties. However, in higher dimensions building up a quadratic model is no longer feasible, because a large number of training data (at least the number of model coefficients $n_c = (d + 1)(d + 2)/2$) are required. Since the linear model has demonstrated good performance on our test problems when it is combined with Multiple models, we propose to use the linear model in this case.

4) Weighted history: Comparing our approach to *Weighted history* (dashed line), we find that particularly in dimensions higher than 2, the *Multiple models* method combined with regression models performs significantly better on all test problems. In the 2-dimensional case, our approach is superior in all test problems except TP 3 and TP 4. This may be explained by the asymmetry that is given around the global f_{exp} optimum of most of the problems where our approach is superior (TP $\{1,2,6\}$): Since *Weighted history* only takes into account the Euclidean distance of available history data, the sample average might be strongly biased. In higher dimensions, *Weighted history* fails to find an acceptable solution. This is due to the sparsity of history data in higher dimensions. Sufficient estimation accuracy with the *Weighted history* approach requires that a minimum number of history data lie in the stochastic neighborhood of the individual. In contrast, when using approximation models, additional history data points from outside the disturbance range can be used to construct the model.

To summarize, estimating robustness based on approximation models seems to be more effective than using a *Weighted history* method.

5) Single disturbed: In the SO case, we finally compare the proposed methods (explicit sampling with help of approximate models) with the *Single disturbed* approach with implicit averaging [37]. From Fig.4, we see that on all test problems, this approach fails to find an acceptable solution (dotted line). This is at the first sight surprising, since TP 3 is taken from reference [37] where the single disturbed approach has proven successful in the 2-dimensional case. However, in that paper a binary-coded genetic algorithm with proportional selection is employed, whereas the real-coded evolution strategy with strategy parameter self-adaptation is employed in this work.

For a fair comparison of the methods, 5 settings for the EA with single disturbed evaluations have been tested, in particular EA's without self-adaptation combined with different selection pressures were tested, and compared against our best approximation method (achieved with Setting 1). The different settings are listed in Table IV, the results of the simulation are shown in Fig.5.

TABLE IV

PARAMETER SETTINGS FOR COMPARISON

Setting 1	"Standard setting", i.e. $(15, 100)$, standard evolution strategy with $\sigma_{init} \in [0.01; 1.0]$, recombination (objective parameters): discrete; recombination (strategy parameters): generalized intermediate
Setting 2	as Setting 1, but *no* strategy parameter self-adaptation, object variables are mutated normally distributed with $\sigma = 0.1$
Setting 3	as Setting 2, but $(50, 100)$
Setting 4	as Setting 1, but *no* strategy parameter self-adaptation, object variables are mutated normally distributed with $\sigma = 0.5$
Setting 5	as Setting 4 but $(50, 100)$

As can be seen, *Single disturbed* performs worse than the explicit averaging method using an approximate model independent of the parameter settings. Also, *Single disturbed* seems to be particularly ineffective in combination with self-adaptation. This can be seen by comparing Settings 1 and 2. Since the step-size cannot be adjusted without self-adaptation, we tested different (fixed) step-sizes. Reducing the selection pressure in form of a larger parent population size (Settings 3 and 5) improves the performance of *Single disturbed*.

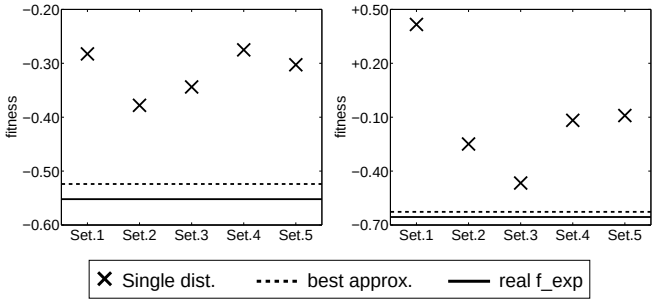

Fig. 5. Comparison of the best approximation method with Tsutsui and Ghosh's implicit averaging (Single disturbed) on different EA parameter settings. **left:** TP 3, $d = 5$; **right:** TP 6, $d = 5$ (averaged over 20 runs).

For a better understanding of how self-adaptation of the step-sizes influences the performance of the algorithms, we recorded the adaptation of the step-sizes for the explicit averaging using *Multiple models* combined with a quadratic regression model, and *Single disturbed* using the standard setting for TP 6 of dimension 2 (cf. Fig.6). Clearly, in the case of *Multiple models with quadratic regression*, the self-adaptation works properly and the step-sizes converge after a certain number of generations, whereas in the *Single disturbed* approach, the step-sizes for both variables diverge seriously. This phenomenon indicates that the implicit averaging method does not work for standard evolution strategies with strategy parameter self-adaptation, probably because the estimated expected fitness is too noisy and thus the self-adaptation fails. A further analysis of this finding is beyond the scope of this paper. We refer to [3] for an extensive analysis on the effect of noise in evolution strategies. To summarize, *explicit averaging*

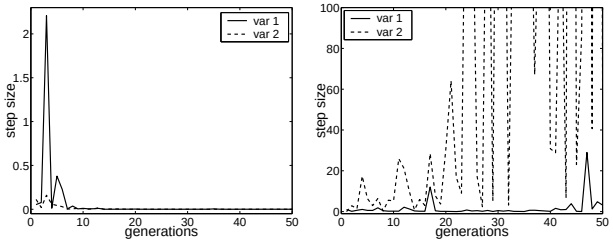

Fig. 6. Self-adaptation of the step-sizes (typical run). **left:** *Multiple models* with quadratic regression; **right:** *Single disturbed.*

based on approximation models seems to be more effective than *implicit averaging* as in *Single disturbed*.

6) Convergence process: Finally, Fig.7 shows some typical convergence plots.

In the initial generations, none of the methods produces a sufficiently accurate estimation. However, after some generations, the space is filled with history data and the estimations become increasingly accurate. With an increasing estimation accuracy, the algorithm approaches the global optimum, in the case when the regression models are used in combination with *Multiple Models* or *Ensemble*. The interpolation methods do not manage to reduce the estimation error significantly over time, and thus fails to converge to the global optimum.

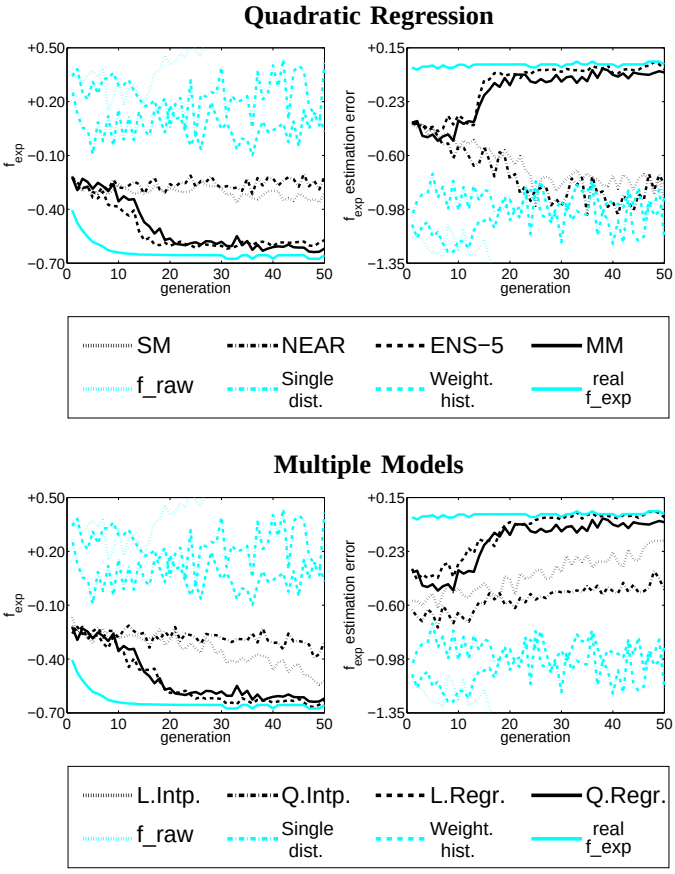

Fig. 7. Convergence and development of the estimation error on the 5-dimensional TP 6. Estimation error is calculated $\widehat{f}_{\text{exp}}(x) - f_{\text{exp}}(x)$ where $\widehat{f}_{\text{exp}}(x)$ is the estimated expected fitness and $f_{\text{exp}}(x)$ is the real expected of the best individual (averaged over 20 runs).
Upper row: Quadratic regression with different model distribution methods (cf. Table I), performance (**top-left**) and corresponding estimation error (**top-right**). **Lower row**: Multiple models with different approximation methods, performance (**bottom-left**) and corresponding estimation error (**bottom-right**). Convergence and improvement of estimation accuracy are reinforcing.

C. MO Results

In the MO approach we use the *Multiple models* method only, since this method has shown to be most effective in the SO simulations. Fig.8 compares the median attainment surface achieved by different approximation models for different problems. For clarity, the methods are additionally compared in Table V based on their rank regarding the hypervolume (area above the median attainment curve in Fig.8).

Let us first consider the 5-dimensional TP 7. The median attainment surface produced when using the real $(f_{\text{exp}}, f_{\text{var}})$ dominates all suggested methods almost everywhere. The *Weighted history* method fails to find solutions with a low variance. A possible explanation for this might be the sparsity of history data: If only a small number of history data points are located within the disturbance range of an individual (perhaps only one), this results in seriously wrong estimations of the fitness variance. At the same time the effect on f_{exp}-estimation might be moderate since there exists no trade-off between the raw fitness and the expected fitness on the MO test problems.

Among the approximation models, the quadratic models

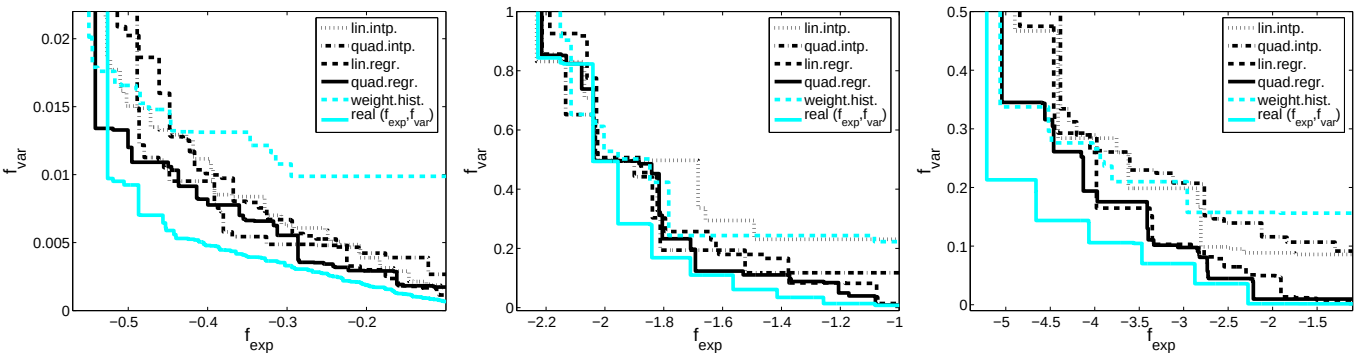

Fig. 8. MO simulation results. Median attainment surfaces (11 runs): **left:** TP 7, $d = 5$, **center:** TP 8, $d = 2$, and **right:** TP 9, $d = 5$.

TABLE V
METHODS RANKED ACCORDING TO HYPERVOLUME OF MEDIAN ATTAINMENT SURFACE (LOW RANKS CORRESPOND TO LARGER HYPERVOLUME AND ARE BETTER)

Approach	TP 7 ($d = 5$)	TP 8 ($d = 2$)	TP 9 ($d = 5$)	Avg
Weighted history	6	5	4	5
Linear interpolation	4	6	5	5
Quadratic interpolation	3	3	6	4
Linear regression	5	4	3	4
Quadratic regression	2	2	2	2
Real f_{exp}	1	1	1	1

seems to work best, and among those, regression works better than interpolation.

Next, we consider the 2-dimensional TP 8 (Fig.8(b)). Here the results are less clear. However, the quadratic regression model again yields the best results of all approximation models. Also, *Weighted history* and linear interpolation are clearly inferior.

Finally, Fig.8(c) shows the results on the 5-dimensional TP 9. Clearly, the quadratic regression model performs best (again), followed by the linear regression model. Again, *Weighted history* fails to find solutions for a low variance. Somewhat surprisingly, as can be seen in Table V, quadratic interpolation performs poorly on this problem.

VII. CONCLUSIONS AND DISCUSSIONS

The goal of this work was to explore new methods for efficient search for robust solutions, i.e., methods that require only a small number of fitness function evaluations. We investigated how information about the fitness surface that is collected throughout the run of an EA can be exploited most effectively for the purpose of robustness estimation. For both SO and MO approaches to robust solutions, we showed that the suggested methods improve the search performance compared to some well-known existing approaches to robustness optimization like implicit averaging (*Single disturbed*) [37] or weighted explicit averaging (*Weighted history*). Comparing different approximation models, we found for both approaches that regression seems to be the preferred approximation method for robustness optimization. This was somewhat surprising as the fitness function is deterministic. However, the reason of the poor performance of interpolation lies in its property of being liable to severe estimation errors. Thus, the standard deviation of the estimation error increases and misguides the search.

We introduced multiple methods to distribute and evaluate approximation models. Besides the rather intuitive *Single model* and *Multiple models*, we also investigated two additional approaches called *Nearest model* and *Ensemble*. Although the ensembles were based on a very simple model distribution, this approach yields significant improvements in some cases. Although at a much higher cost, the *Multiple models* approach guides the search most effectively in our simulations.

No definite conclusion can be drawn concerning whether a linear or a quadratic model guides the search better. However, since it becomes impossible with a limited number of training data to build a quadratic model in higher dimensions, a linear model must be used, which fortunately turned out to be as good as the quadratic in most cases when combined with the best model distribution method, namely Multiple models.

Two promising ideas for future research arise from the above findings. First, it is very desirable to develop a more sophisticated model distribution strategy so that the models are able to better describe the fitness landscape searched by the current population. Second, it would be very interesting to further investigate the influence of ensembles of approximation models on the search performance.

The MO approach to robustness optimization represents a greater challenge to our new methods. The findings regarding the choice of the approximation model are consistent with the findings in SO: Regression is the recommended method, and the quadratic model seems preferable (for variance estimation).

Scalability is an important issue if the proposed methods are going to be employed to solve real-world problems. With an increasing dimensionality d the approximation models require a larger number of input data n_{in}. However, when local linear regression is used, which has shown to be very effective in combination with *Multiple models*, n_{in} increases linearly with d. Thus the computational cost for building high-dimensional linear regression models is still acceptable compared to that of fitness evaluations in many real-world problems. Applying the proposed algorithms to complex real-world problems will be one of our future research topics.

ACKNOWLEDGMENTS

We would like to thank Tatsuya Okabe for his technical assistance. Yaochu Jin and Ingo Paenke would like to thank Bernhard Sendhoff and Edgar Körner for their support. Ingo Paenke thanks Hartmut Schmeck for his support. Authors are grateful to Xin Yao and anonymous reviewers whose comments and suggestions have been very helpful in improving the quality of this paper. The simulation code was implemented based on the SHARK C++ library package, which is public software available at http://shark-project.sourceforge.net/.

APPENDIX A
TEST PROBLEMS

The disturbance of each design variable is normally distributed with $N(0,\sigma)$, where σ is chosen with respect to the shape of the test function. In order to have a finite probability distribution we cut the normal distribution at its 0.05- and 0.95- quantiles. The test problems (test function, feasible $\mathbf{x}$-domain and σ) are listed below:

$$f_1(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{2_b}(x_i) \ , \ \mathbf{x} \in [0;10]^d \ , \sigma = 0.5 \ ,$$
$$\text{with } f_{1_b}(x_i) = \begin{cases} \frac{2-x_i}{6} & : 0.2 < x_i \leq 0.8 \ , \\ 0 & : \text{otherwise} \ . \end{cases}$$

$$f_2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{2_b}(x_i) \ , \ \mathbf{x} \in [0;10]^d \ , \sigma = 0.5 \ ,$$
$$\text{with } f_{2_b}(x_i) = \begin{cases} -(8-x_i)^{0.1} e^{-0.2(8-x_i)} & : x_i < 8 \ , \\ 0 & : \text{otherwise} \ . \end{cases}$$

$$f_3(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{3_b}(x_i) \ , \ \mathbf{x} \in [0;1]^d \ , \sigma = 0.0625 \ ,$$
$$\text{with } f_{3_b}(x_i) = \begin{cases} e^{-2ln2(\frac{x-0.1}{0.8})^2} |\sin(5\pi x)|^{0.5} & : 0.4 < x_i \leq 0.6 \ , \\ e^{-2ln2(\frac{x-0.1}{0.8})^2} \sin^6(5\pi x) & : \text{otherwise} \ . \end{cases}$$

$$f_4(\mathbf{x}) = \left(\mathbf{x}^2 - \alpha\right)^2 \ , \ \mathbf{x} \in [-5;+5]^d \ , \sigma = 0.5 \ ,$$
$$\text{with } \alpha = \begin{cases} 0.1 & : d = 2 \ , \\ 0.175 & : d = 5 \ , \\ 0.3 & : d = 10 \ . \end{cases}$$

$$f_5(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{5_b}(x_i) \ , \ \mathbf{x} \in [0;1]^d \ , \sigma = 0.05 \ ,$$
$$\text{with } f_{5_b}(x_i) = \begin{cases} -0.5e^{-0.5\frac{(x_i-0.4)^2}{0.05^2}} & : x_i < 0.4696 \ , \\ -0.6e^{-0.5\frac{(x_i-0.5)^2}{0.02^2}} & : 0.4696 \leq x_i \leq 0.5304 \ , \\ -0.5e^{-0.5\frac{(x_i-0.6)^2}{0.05^2}} & : \text{otherwise} \ . \end{cases}$$

$$f_6(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{6_b}(x_i) \ , \ \mathbf{x} \in [0;10]^d \ , \sigma = 0.5 \ ,$$
$$\text{with } f_{6_b}(x_i) = 2\sin(10e^{(-0.2x_i)}x_i)e^{(-0.25x_i)} \ .$$

$$f_7(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{7_b}(x_i) \ , \ \mathbf{x} \in [0;10]^d \ , \sigma = 0.5 \ ,$$
$$\text{with } f_{7_b}(x_i) = \begin{cases} \frac{(x_i-2)^4}{6} & : 2 < x_i \leq 8 \ , \\ 0 & : \text{otherwise} \ . \end{cases}$$

$$f_8(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{8_b}(x_i) \ , \ \mathbf{x} \in [0;10]^d \ , \sigma = 0.1 \ ,$$
$$\text{with } f_{8_b}(x_i) = 2\sin(10e^{(-0.08x_i)}x_i)e^{(-0.25x_i)} \ .$$

$$f_9(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{d} f_{9_b}(x_i) \ , \ \mathbf{x} \in [0;10]^d \ , \sigma = 0.5 \ ,$$
$$\text{with } f_{9_b}(x_i) = 2\sin(3e^{(-0.08x_i)}x_i)e^{(-0.25x_i)} \ .$$

APPENDIX B
SAMPLING TECHNIQUES

The following sampling techniques are mentioned in this paper:

- *Stratified sampling* [29] divides the space of possible disturbances into regions of equal probability according to the probability distribution of the noise and draws one sample from every region.
- *Latin hypercube sampling* [27]: In order to draw n samples, the range of disturbances in each dimension is divided into n parts of equal probability according to the probability distribution, and n random samples are chosen such that each quantile in each dimension is covered by exactly one sample.

For an arbitrary distribution, the division into regions of equal probability is done by calculating the respective quantiles. Fig.9 illustrates the sampling methods for the case of uniform

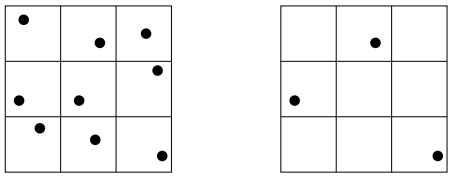

Fig. 9. **left:** Stratified sampling, **right:** Latin hypercube sampling

distribution. Here, possible sample sets for the 2-dimensional case are depicted. In this illustration, the number of quantiles is 3 for both sampling methods.

REFERENCES

[1] P. Alfeld. Scattered data interpolation in three or more variables. In T. Lyche and L. Schumaker, editors, *Mathematical Methods in Computer Aided Geometric Design*, pages 1–33. Academic Press, 1989.

[2] D.K. Anthony and A.J. Keane. Robust-optimal design of a lightweight space structure using a genetic algorithm. *AIAA Journal*, 41(8):1601–1604, 2003.

[3] D. Arnold. *Noisy Optimization with Evolution Strategies*. Kluwer Academic Publishers, 2002.

[4] J. Branke. Creating robust solutions by means of an evolutionary algorithm. In A.E. Eiben, T. Bäck, M.Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN V*, pages 119–128. Springer, 1998.

[5] J. Branke. Reducing the sampling variance when searching for robust solutions. In L. Spector et al., editor, *Genetic and Evolutionary Computation Conference (GECCO '01)*, pages 235–242. Morgan Kaufmann, 2001.

[6] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.

[7] J. Branke and C. Schmidt. Fast convergence by means of fitness estimation. *Soft Computing*, 9(1):13–20, 2005.

[8] W. Chen, J. Allen, K.Tsui, and F. Mistree. A procedure for robust design: Minimizing variations caused by noise factors and control factors. *ASME Journal of Mechanical Design*, 118:478–485, 1996.

[9] I. Das. Robustness optimization for constrained nonlinear programming problems. *Engineering Optimization*, 32(5):585–618, 2000.

[10] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. 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 849–858, 2000.

[11] K. Deb and D.E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, 1989.

[12] C.M. Fonseca and P.J. Fleming. On the performance assessment and comparison of stochastic multiobjective optimizers. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 584–593, 1996.

[13] G.H. Goloub and C.F. van Loan. *Matrix Computations*. The John Hopkins Press, 3rd edition, 1996.

[14] H. Greiner. Robust optical coating design with evolution strategies. *Applied Optics*, 35(28):5477–5483, 1996.

[15] C.A.R. Hoare. Quicksort. *Computer Journal*, 5(1), 1962.

[16] M. Hüsken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for evolutionary design optimization. *Soft Computing*, 9(1):21–28, 2005.

[17] M. T. Jensen. Generating robust and flexible job shop schedules using genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 7(3):275–288, 2003.

[18] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[19] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments – A survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[20] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? In *Genetic and Evolutionary Computation Conference*, pages 1042–1049. Morgan Kaufmann, 2001.

[21] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.

[22] Y. Jin and B. Sendhoff. Trade-off between optimality and robustness: An evolutionary multiobjective approach. In C.M. Fonseca et al., editor, *International Conference on Evolutionary Multi-criterion Optimization*, volume 2632 of *LNCS*, pages 237–251. Springer, 2003.

[23] V.J. Leon, S.D. Wu, and R.H. Storer. Robustness measures and robust scheduling for job shops. *IIE Transactions*, 26:32–43, 1994.

[24] K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscapes. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 4(3):172–183, 2000.

[25] C. Loader. *Local Regression and Likelihood*. Springer, 1st edition, 1999.

[26] D. H. Loughlin and S. R. Ranjithan. Chance-constrained genetic algorithms. In *Genetic and Evolutionary Computation Conference*, pages 369–376, 1999.

[27] M. D. McKay, W. J. Conover, and R. J. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

[28] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimization. In *IEEE Congress on Evolutionary Computation*, pages 878–885, 2003.

[29] W. H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition edition, 1992.

[30] T. Ray. Constrained robust optimal design using a multi-objective evolutionary algorithm. In *IEEE Congress on Evolutionary Computation*, pages 419–424. IEEE, 2002.

[31] T. Ray and H. M. Tsai. A parallel hybrid optimization algorithm for robust airfoil design. In *AIAA Aerospace Sciences Meeting and Exhibit*, pages 11474–11482, 2004.

[32] C. R. Reeves. A genetic algorithm approach to stochastic fbwshop sequencing. In *Proceedings of the IEE Colloquium on Genetic Algorithms for Control and Systems Engineering*, number 1992/106 in IEE Digest, pages 13/1–13/4. IEE, London, 1992.

[33] R.G. Regis and C.A. Shoemaker. Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation*, 8(5):490–505, 2004.

[34] B. Sendhoff, H. Beyer, and M. Olhofer. The influence of stochastic quality functions on evolutionary search. In K. C. Tan, M. H. Lim, X. Yao, and L. Wang, editors, *Recent Advances in Simulated Evolution and Learning*, volume 2 of *Advances in Natural Computation*, pages 152–172. World Scientific, New York, 2004.

[35] R Sibson. A brief description of natural neighbor interpolation. In V Barnet, editor, *Interpreting Multivariate Data*, pages 21–36. John Wiley, 1981.

[36] A. Thompson. Evolutionary techniques for fault tolerance. In *Proc. UKACC International Conference on Control*, pages 693–698, 1996.

[37] S. Tsutsui and A. Ghosh. Genetic algorithms with a robust solution searching scheme. *IEEE Transactions on Evolutionary Computation*, 1(3):201–208, 1997.

[38] S. Tsutsui, A. Ghosh, and Y. Fujimoto. A robust solution searching scheme in genetic search. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 543–552. Springer, 1996.

[39] D. Wiesmann, U. Hammel, and T. Bäck. Robust design of multilayer optical coatings by means of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2(4):162–167, 1998.

[40] Y. Yamaguchi and T. Arima. Aerodynamic optimization for the transonic compressor stator blade. In I. C. Parmee and P. Hajela, editors, *Optimization in Industry*, pages 163–172. Springer, 2002.

[41] Q. Zhang and H. Mühlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Transactions on Evolutionary Computation*, 8(2):127–136, 2004.

[42] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.