# Learning Feature-Parameter Mappings for Parameter Tuning via the Profile Expected Improvement

Jakob Bossek
University of Münster, Germany
bossek@wi.uni-muenster.de

Bernd Bischl
TU Dortmund, Germany
bischl@statistik.tu-dortmund.de

Tobias Wagner
TU Dortmund, Germany
wagner@isf.de

Günter Rudolph
TU Dortmund, Germany
guenter.rudolph@tu-dortmund.de

## ABSTRACT

The majority of algorithms can be controlled or adjusted by parameters. Their values can substantially affect the algorithms' performance. Since the manual exploration of the parameter space is tedious – even for few parameters – several automatic procedures for *parameter tuning* have been proposed. Recent approaches also take into account some characteristic properties of the problem instances, frequently termed *instance features*.

Our contribution is the proposal of a novel concept for feature-based algorithm parameter tuning, which applies an approximating surrogate model for learning the continuous feature-parameter mapping. To accomplish this, we learn a joint model of the algorithm performance based on both the algorithm parameters and the instance features. The required data is gathered using a recently proposed acquisition function for model refinement in surrogate-based optimization: the *profile expected improvement*. This function provides an avenue for maximizing the information required for the *feature-parameter mapping*, i.e., the mapping from instance features to the corresponding optimal algorithm parameters. The approach is validated by applying the tuner to exemplary evolutionary algorithms and problems, for which theoretically grounded or heuristically determined feature-parameter mappings are available.

## Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*

## Keywords

Model-based Optimization; Parameter Tuning; Evolutionary Algorithms



*GECCO '15, July 11 - 15, 2015, Madrid, Spain*



ISBN 978-1-4503-3472-3/15/07...$15.00

DOI: http://dx.doi.org/10.1145/2739480.2754673

## 1. INTRODUCTION

In recent years, the systematic use of optimization approaches for tuning the parameters of evolutionary computation methods – and other computational algorithms in general – has attained increasing attention [7]. Many researchers have documented the superiority of automatic tuning approaches over manual trial-and-error procedures, e.g., see [2, 13, 15]. In particular, employing surrogate models in the tuning process is known to usually speed-up the discovery of good parameter settings while also providing information w.r.t. the parameters' effects and interactions [2, 5, 13].

As evolutionary computation methods are often considered to be general-purpose approaches, they have to cope with a wide variety of application problems. In many cases, however, a single parameter set cannot achieve suitable results for all problems in a domain of interest, and hence, setting algorithm parameters w.r.t. instance features that describe the characteristics of the problem at hand might be beneficial. For continuous black-box optimization, e.g., exploratory landscape analysis provides a large set of features for problem instances [17, 18] that have already been used for the related problem of algorithm selection [4].

The feature-dependent configuration problem is tackled in, e.g., ISAC [15] and HYDRA [20]. While the former employs a clustering in feature space and configures the algorithm on each cluster, the latter uses the instance features for algorithm selection based on a pre-constructed set of complementary algorithm configurations. Hence, both algorithm do not provide a continuous mapping from features to parameters, as either fixed instance clusters or algorithm configurations are pre-constructed. Surrogate models could efficiently learn the joint relation between instance features and algorithm parameters to algorithm performance. But to the best of our knowledge, no sequential model-based optimization technique for parameter tuning actually defines a point acquisition function that takes instance features into account during the sequential planning of experiments. The popular sequential model-based SMAC algorithm [13] does use instance features, but only to stabilize its search process for a single configuration which works well on all instances.

Motivated by applications in reliability engineering, Ginsbourger et al. [9] recently proposed a criterion for sequential experiments with regard to the estimation of profile optima. The authors are considering scenarios where the variable

space is partitioned into a group of controllable decision variables and a second group of uncontrollable nuisance parameters. They now would like to estimate a worst-case scenario. i.e., for each control paramater the worst response w.r.t. to all nuisance settings must be estimated. They applied kriging models, also popular in model-based parameter tuning [1,16]. While their perspective and motivation is quite different to ours, the mapping from instance features to optimal algorithm parameters can also be considered as a profile optimum and we can rather directly transfer their acquisition function to the scenario of parameter tuning.

Our contribution is the adoption of the approach by [9] to the field of model-based optimization for automatic parameter tuning. Note that the method neither directly optimizes parameters for a single instance nor (on average) for a set of instances, but instead defines a new sequential design-of-experiments procedure to learn a mapping from instance features to algorithm parameters. The resulting model provides the basis for a tuner that explicitly uses features of problem instances to predict the corresponding parameters.

We evaluate the approach on synthetic test problems and exemplary parameter tuning tasks. In this first concept study, we restrict ourselves to the simple case of a single feature and algorithm parameter, as it allows the visualization of the feature-parameter mappings and their development in the sequential planning process.

# 2. METHODOLOGY

In this section, the methodological foundations of our feature-based parameter tuning approach are presented. After a brief review of the general procedure of Sequential Model-Based Optimization (SMBO), we describe its extension utilizing the Profile Expected Improvement for learning the joint surrogate model.

## 2.1 Sequential Model-Based Optimization

Let $f : \Theta \to \mathbb{R}$ be a black-box function we would like to minimize over our search space $\Theta$. In case of SMBO, we usually assume that $f$ is expensive to evaluate, so we can only spend a severely restricted budget of function evaluations.

**Algorithm 1:** Generic SMBO approach.

**Data**: Expensive function $f$, Parameter space $\Theta$.
**Result**: Estimated minimum $\boldsymbol{\theta}^*$.
**begin**
  $\mathcal{X} \leftarrow$ generateDesign($\Theta$)
  Evaluate $\boldsymbol{y} \leftarrow f(\mathcal{X})$
  **while** *Termination condition not satisfied* **do**
    $\hat{m} \leftarrow$ fitModel($\mathcal{X}, \boldsymbol{y}$)
    $y_{\min} \leftarrow \min(y)$
    $\boldsymbol{\theta}_{new} \leftarrow$ optimizeInfillCrit($\Theta, \hat{m}, y_{\min}$)
    Evaluate $y_{new} \leftarrow f(\boldsymbol{\theta}_{new})$
    append($\mathcal{X}, \boldsymbol{\theta}_{new}$), append($\boldsymbol{y}, y_{new}$)
  **return** *Estim. minimum* $\boldsymbol{\theta}^*$ *from design* $\mathcal{X}$.

The general outline of SMBO is given in Algorithm 1. In each iteration, we approximate $f$ based on all available evaluations via a surrogate model, optimize an infill criterion (or acquisition function) which balances exploration of the search space and exploitation of promising areas, evaluate the proposed point and update our model.

Kriging models [14] are frequently used in the context of SMBO due to their flexibility with regard to possible response surfaces and the possibility to estimate the model uncertainty. The latter is necessary for the Expected Improvement (EI) acquisition function which provides the foundation of the Efficient Global Optimization (EGO) algorithm by Jones et al. [14]. In the following, we discuss how to transfer SMBO approaches like EGO to parameter tuning.

## 2.2 SMBO for Parameter Tuning

Let us assume a configurable algorithm $\mathcal{A}$ with parameter space $\Theta$ which has to solve problem instances $I \in \mathcal{I}$ drawn i.i.d. from distribution $P_{\mathcal{I}}$. Hence, $P_{\mathcal{I}}$ reflects the probability with which instances $I \in \mathcal{I}$ will be presented to $\mathcal{A}$. A cost function $c(I, \boldsymbol{\theta})$ assigns a cost or performance value to $\mathcal{A}$ with configuration $\boldsymbol{\theta}$ on instance $I \in \mathcal{I}$.

We can configure $\mathcal{A}$ either on a single instance $I$ by solving[1]

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} f(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}\in\Theta} c(I, \boldsymbol{\theta})$$

or with regard to the expectation over all instances

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} f(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}\in\Theta} \int_{I\in\mathcal{I}} c(I, \boldsymbol{\theta}) dP_{\mathcal{I}},$$

usually by approximating the integral via summing over a given, finite set of representative instances $\mathcal{J} \subset \mathcal{I}$

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Theta} f(\boldsymbol{\theta}) = \arg\min \sum_{I\in\mathcal{J}} c(I, \boldsymbol{\theta}).$$

One obvious problem of the latter formulation is that we only try to find a configuration which behaves well on average for all instances in $\mathcal{J}$ and do not exploit any specific knowledge we might have about the instances and their properties. This drawback is addressed in the following.

## 2.3 Feature Parameter Mapping

Now, we will assume a feature mapping $\delta : \mathcal{I} \to \Delta$, assigning instance $I$ the describing feature vector $\delta(I)$. In general, these features are continuous, such that$\Delta \subseteq \mathbb{R}^p$. The feature mapping should reflect the properties of $I$ and should not be too expensive to calculate. In this case, it should help us to determine a reasonable parameter setting $\boldsymbol{\theta}$ for $\mathcal{A}$ on $I$. Formally, we would like to obtain a feature parameter mapping (FPM)

$$m : \boldsymbol{\delta} \in \Delta \to \boldsymbol{\theta}^*_{\boldsymbol{\delta}} \in \Theta \tag{1}$$

where $\boldsymbol{\theta}^*_{\boldsymbol{\delta}}$ minimizes our cost when we integrate out over all instances $I$ which map to $\boldsymbol{\delta}$:

$$\boldsymbol{\theta}^*_{\boldsymbol{\delta}} = \arg\min_{\boldsymbol{\theta}\in\Theta} \int_{\delta(I)=\boldsymbol{\delta}} c(I, \boldsymbol{\theta}) dP_{\mathcal{I}}$$

With a slight abuse of notation we will now extend the function $c$ from the space $\mathcal{I} \times \Theta$ to the space$\Delta \times \Theta$ by setting $c(\boldsymbol{\delta}, \boldsymbol{\theta}) = \int_{\delta(I)=\boldsymbol{\delta}} c(I, \boldsymbol{\theta}) dP_{\mathcal{I}}$

Our task now is to learn this profile, i.e. the whole set of all $\boldsymbol{\theta}^*_{\boldsymbol{\delta}}$ elements, as good as possible during an offline training phase. Given some prior experiments $(\boldsymbol{\delta}_i, \boldsymbol{\theta}_i, c_i)_{i=1\ldots,n}$, where $\mathcal{A}$ was run on instance $I_i$ with feature vector $\boldsymbol{\delta}_i = \delta(I_i)$ and produced numerical costs $c_i = c(I_i, \boldsymbol{\theta}_i)$, we would

[1]To simplify notation, we have assumed here that the algorithm is deterministic, otherwise replace $c(I, \boldsymbol{\theta})$ by $E_{\boldsymbol{\theta}}(I, \boldsymbol{\theta})$.

like to learn a joint model $\hat{c}$ on$\Delta \times \Theta$ that predicts the costs of a configuration $\boldsymbol{\theta}$ when run on an instance with features $\boldsymbol{\delta}_i$. We have to stress in this context that the use of the features allows the instances to be reasonably located in the respective space $\Delta$, but that, depending on the mapping $\delta$, we may observe different instances $I \in \mathcal{I}$ which are mapped to the same feature vector $\boldsymbol{\delta}$. As a consequence, we may only obtain a single sample of a distribution of results based on a single experiment.

This leads to the following approximation of the FPM. Given a new instance $I$, we calculate its feature vector $\delta(I)$ and optimize the joint model in the subspace while fixing its first argument to $\delta(I)$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Theta} \hat{c}(\delta(I), \boldsymbol{\theta})$$

to obtain the configuration $\boldsymbol{\theta}^*$ we would use to run $\mathcal{A}$ on $I$. Hence, assuming we could learn a perfect model $\hat{c}$ for $c$, we could predict accurately for a given instance $I$, how well $\mathcal{A}$ would perform on average on all instances with the same features as $I$.

To learn this joint model, we again employ a regression approach as in SMBO, now defined on$\Delta \times \Theta$ and with the aim of estimating the profile with minimal costs. The main problem is to find an efficient sampling procedure to iteratively plan our configuration experiments. For this, we cannot use the EI, as the aim is no longer to only estimate the global optimum, but to approximate the whole profile of $\boldsymbol{\theta}^*_{\boldsymbol{\delta}}$ for all $I \in \mathcal{I}$.

## 2.4 Profile Expected Improvement

As potential solution, Ginsbourger et al. have introduced the Profile-EI (PEI) [9]. With regard to their work, the *Profile-Improvement* PI($\boldsymbol{\delta}$, $\boldsymbol{\theta}$) of a potential $(i+1)$-th experiment $(\boldsymbol{\delta} = \delta(I), \boldsymbol{\theta})$ on an instance $I$ is defined by

$$PI(\boldsymbol{\delta}, \boldsymbol{\theta}) = \max\left\{0, C_{\boldsymbol{\delta},\boldsymbol{\theta}} - \max\{c_{\min}, \hat{c}(\boldsymbol{\delta}, \hat{\boldsymbol{\theta}}^*_{\boldsymbol{\delta}})\}\right\},$$

where $\hat{c}(\boldsymbol{\delta}, \hat{\boldsymbol{\theta}}^*_{\boldsymbol{\delta}})$ are the estimated costs for the estimated best configuration on the given feature vector $\boldsymbol{\delta}$ depending on the current approximation of the FPM, $c_{\min} = \min_{i=1,\ldots,n} c_i$ is the minimum observed cost value over all experiments, and $C_{\boldsymbol{\delta},\boldsymbol{\theta}}$ is the corresponding random variable from the posterior distribution defined by the current surrogate model $\hat{c}$.

In case of a Gaussian process model, $C_{\boldsymbol{\delta},\boldsymbol{\theta}}$ is normally distributed as $N(\hat{c}(\boldsymbol{\delta}, \boldsymbol{\theta}), \hat{s}(\boldsymbol{\delta}, \boldsymbol{\theta})^2)$, with the mean prediction $\hat{c}(\boldsymbol{\delta}, \boldsymbol{\theta})$ and the local uncertainty $\hat{s}(\boldsymbol{\delta}, \boldsymbol{\theta})$. This corresponds to the definition of the EI in EGO. In contrast to EGO, where we would only use $c_{\min}$ for the approximation of the current optimum, the potentially infinite number of instances $I \in \mathcal{I}$ requires a more specific definition.

Ginsbourger et al. recommend to use the predicted costs $\hat{c}(\boldsymbol{\delta}, \hat{\boldsymbol{\theta}}^*_{\boldsymbol{\delta}})$ of the approximated model optimum

$$\hat{\boldsymbol{\theta}}^*_{\boldsymbol{\delta}} = \arg\min_{\boldsymbol{\theta} \in \Theta} \hat{c}(\boldsymbol{\delta}, \boldsymbol{\theta})$$

for the fixed feature vector $\boldsymbol{\delta}$. The capping by means of the overall observed optimum cost value $c_{\min}$ is recommended to ensure global exploration, even in cases $\hat{c}(\boldsymbol{\delta}, \boldsymbol{\theta})$ locally overshoots the actual cost function [9].

With these definitions, the *Profile-Expected-Improvement* is then computed as $PEI(\boldsymbol{\delta}, \boldsymbol{\theta}) = E\left(PI(\boldsymbol{\delta}, \boldsymbol{\theta})\right)$ in a similar fashion as the EI. After plugging in the respective proxies, we obtain the following closed-form definition:

$$PEI(\boldsymbol{\delta}, \boldsymbol{\theta}) = \begin{cases} \hat{s}(\boldsymbol{\delta}, \boldsymbol{\theta})\left[g(\boldsymbol{\delta}, \boldsymbol{\theta})\Phi(g(\boldsymbol{\delta}, \boldsymbol{\theta})) \right. \\ \quad \left. + \phi(g(\boldsymbol{\delta}, \boldsymbol{\theta}))\right], & \text{if } \hat{s}(\boldsymbol{\delta}, \boldsymbol{\theta}) \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

where $\phi$ and $\Phi$ are the probability density and cumulative density function of the standard normal distribution, respectively, and $g(\boldsymbol{\delta}, \boldsymbol{\theta}) = (c^{\boldsymbol{\delta}}_{\min} - \hat{c}(\boldsymbol{\delta}, \boldsymbol{\theta}))/\hat{s}(\boldsymbol{\delta}, \boldsymbol{\theta})$.

**Algorithm 2:** FBSMBO-Training

**Data**: Training instances $\mathcal{J}$, parameter space $\Theta$, feature mapping $\delta$, algorithm $\mathcal{A}$, cost function $c(I, \boldsymbol{\theta})$.
**Result**: Mapping $\hat{m} : \Delta \to \Theta$.
**begin**
  $\mathcal{X} \leftarrow$ generateDesign($\mathcal{J}, \delta, \Theta$)
  $\mathcal{X}$ now consists of tuples $(I, \boldsymbol{\delta}, \boldsymbol{\theta})$
  $\boldsymbol{y} \leftarrow$ Evaluate each element of $\mathcal{X}$ with $c(I, \boldsymbol{\theta})$
  **while** *Termination condition not satisfied* **do**
    $\hat{c} \leftarrow$ fitModel($\mathcal{X}, \boldsymbol{y}$)
    $c_{\min} \leftarrow \min(\boldsymbol{y})$
    **for** $I_j \in \mathcal{J}$ **do**
      $\boldsymbol{\delta} \leftarrow \delta(I_j)$
      $c^{\boldsymbol{\delta}}_{\min} \leftarrow$ optMeanOnInstance($\hat{c}, I_j, c_{\min}$)
      $PEI[j], \boldsymbol{\theta}[j] \leftarrow$ optPEIOnInstance($\hat{c}, I_j, c^{\boldsymbol{\delta}}_{\min}$)
    Set $I_{new}, \boldsymbol{\theta}_{new}$ by taking the max of PEI[$\cdot$]
    Run $\mathcal{A}(I_{new}, \boldsymbol{\theta}_{new})$, i.e. $y_{new} \leftarrow c(I_{new}, \boldsymbol{\theta}_{new})$;
    append $(\mathcal{X}, [I_{new}, \delta(I_{new}), \boldsymbol{\theta}_{new}])$
    append $(\boldsymbol{y}, y_{new})$
  **return** $\hat{m} : \Delta \to \Theta$

## 2.5 Feature-Based SMBO

Our Feature-Based SMBO (FBSMBO) configurator works in a similar fashion as the standard SMBO algorithm. We start by generating an initial design $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with $\mathbf{x}_i = [\boldsymbol{\delta}_i, \boldsymbol{\theta}_i] \in \Delta \times \Theta$, which is evaluated by running the associated experiments. Hence, in addition to SMBO an instance $I_i$ is connected to each design point, since algorithm $\mathcal{A}$ needs the instance for the evaluation of the design point and the respective feature vector $\delta(I_i)$ is used as additional input for the model fitting. This data is then used to fit the machine learning model $\hat{c}$ of the cost landscape.

The evaluation of the parameter/feature-vector pair will result in a noisy response for two reasons. First, most evolutionary computation methods are stochastic in nature which may lead to different outcomes of the same parameterization on the same instance. Second, the feature mapping, such as the input dimension of a respective instance, may result in the same feature-vector $\boldsymbol{\delta}$ for different problem instances. If experiments with the same algorithm parameters $\boldsymbol{\theta}$ are performed on all these instances, the performance values $\boldsymbol{y}$ will likely differ. As the surrogate model $\hat{c}$ maps from $\boldsymbol{\theta} \times \boldsymbol{\delta}$ to $\boldsymbol{y}$, a high variation within the response of a single input vector can occur. Hence, the *fitModel* is implemented by utilizing the reinterpolation procedure proposed by Forrester et al. in [8] for noisy computer experiments. In this procedure, first a regressing kriging model is fitted. Then, the predictions of this model are used as proxies for the expected response values of the design points, to which another inter-

polating model is fitted. The resulting model is then used for determining further design points in an SMBO manner. By these means, the variations in the responses are smoothed while allowing the variance of the model still to be used for ensuring a global search [8].

Within the FBSMBO loop (see Algorithm 2), FBSMBO iterates over all instances $I \in \mathcal{I}$ to compute (a) the respective minimum cost value proxies $c^{\boldsymbol{\delta}}_{\min}$ for the PEI formula (optMeanOnInstance) and (b) the optimal configuration and PEI value on the subspace of algorithm parameters given the fixed instance features $\boldsymbol{\delta} = \delta(I)$: $\max_{\boldsymbol{\theta} \in \Theta} PEI(\boldsymbol{\delta}, \boldsymbol{\theta})$. Here the feature vector is fixed and optimization takes place on the subset of algorithm parameters.

The instance $I$ with maximal $PEI(\boldsymbol{\delta}, \boldsymbol{\theta}^*_{\boldsymbol{\delta}})$ and corresponding estimated optimal parameter vector is selected to be considered for the next experiment. This process is repeated until a stopping condition is met. The offline phase returns an FPM $\hat{m} : \Delta \to \Theta$, which is constructed from the joint model $\hat{c}(\boldsymbol{\delta}, \boldsymbol{\theta})$ by optimizing over the subspace of algorithm parameters given a fixed feature vector $\boldsymbol{\delta}$. The FPM can be applied to determine parameters for previously unseen instances $I$ as part of the online (application) phase, as shown in section 2.3.

# 3. EVALUATION

In this section, we empirically evaluate the proposed FBSMBO approach for parameter tuning. We start by analyzing the behavior of the configurator on synthetic examples, before we apply it to some practical parameter tuning scenarios. All of our experiments were performed in R by using the BatchJobs and BatchExperiments packages [3].

## 3.1 Synthetic functions

For the first set of experiments, we consider an artificial two-dimensional scenario. The first dimension will be regarded as the "instance feature" $\boldsymbol{\delta}$, while the second dimension represents the "algorithm parameter" $\boldsymbol{\theta}$. We hence model a two-dimensional cost function $c : \Delta \times \Theta \to \mathbb{R}$, which we assume to follow a deterministic test function. This allows quick test runs and a visual check of the results to be performed in a known scenario. For the preliminary experiments, we selected the well-known black-box optimization functions Branin, Himmelblau, and Goldstein-Price [11, 14]. Moreover, we defined a simple Double-Sum to obtain profile curves $\boldsymbol{\theta}^*_{\boldsymbol{\delta}}$ of different complexity and shape.

For the active learning of the FPM, we ran both the PEI infill strategy and – for comparison – a random sampling (RANDOM) on each test function. The latter selects the instances and algorithm parameters $(\boldsymbol{\delta}, \boldsymbol{\theta})$ evaluated for fitting the model completely at random. Each strategy was started from the same initial design of 10 points and was allowed a budget of 30 subsequent evaluations. This experiment was replicated 30 times, resulting in 30 models $\hat{m}$ for the FPM of each strategy. To quantify the approximation quality of each FPM $\hat{m}^S$ based on infill strategy $S$, we computed the mean absolute difference between the algorithm performance based on the proposed and the known optimum parameters over an equidistant grid of features $\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_k \in \Delta$. More specific, if $m$ is the true FPM based on the original test function, the performance measure is computed by the formula

$$PERF(S) = \frac{1}{k} \sum_{i=1}^{k} \left| c(\boldsymbol{\delta}_i, \hat{m}(\boldsymbol{\delta}_i)) - c(\boldsymbol{\delta}_i, m(\boldsymbol{\delta}_i)) \right| .$$

We compare the infill strategies by means of this quality measure, whereby lower PERF values are favorable.

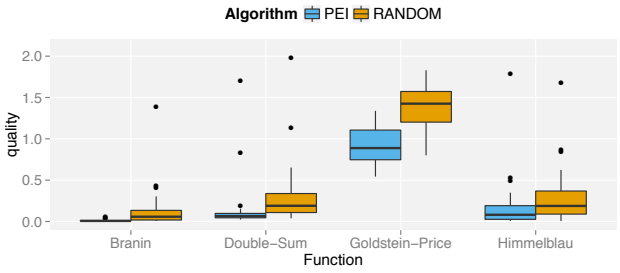



**Figure 1: Boxplots of the distributions of the performance values reached in the experiments separated by test function and infill strategy used.**

Figure 1 shows the distribution of performance values for each infill strategy used. Obviously, the quality of the model learned with the PEI is notably better. For Branin, Double-Sum and Goldstein-Price the upper quartile of PEI coincides with the lower quartile of random sampling. In case of the Himmelblau function, the upper quartile at least coincides with the respective median.

| Function | Min. | Median | Mean | Max. | p-value |
|---|---|---|---|---|---|
| Branin | -1.38 | **-0.05** | **-0.13** | 0.01 | **0.01** |
| Double-Sum | -3.95 | **-0.11** | **-0.30** | 1.43 | **0.04** |
| Goldstein-Price | -1.03 | **-0.54** | **-0.46** | 0.37 | **0.00** |
| Himmelblau | -1.55 | **-0.07** | **-0.13** | 1.72 | 0.09 |

**Table 1: Statistics based on the paired differences of the performance values of both strategies per run and $p$-values of the respective $t$-test for the synthetic functions.**

In addition, Table 1 summarizes measures of central tendency and the $p$-values of the paired $t$-test checking the one-sided null hypothesis of a zero mean of the paired differences per experiment. For significance level $\alpha = 0.05$, all functions but Himmelblau result in significant improvements. This result is strong, since the test functions are known to be predicted well with only about half of the design points used in our experiments [14]. Nevertheless, we see notable statistical evidence for the superiority of the PEI approach as compared with the naive sampling of random infill points for profile curve reconstruction.

## 3.2 Mutation rate of the (1+1)-GA

In this section, we analyze whether FBSMBO is capable of learning the optimum parameter configurations for the (1+1) Genetic Algorithm (GA) on the `OneMax` test function. The (1+1)-GA operates on binary representations of dimension $n$, maintains a population of only a single individual and makes solely use of a simple bit-flip mutation operator with mutation rate $p_m \in (0, 1)$. For linear functions of type $y(\mathbf{x}) = \tau + \sum_{i=1}^{n} w_i x_i$, $\tau \in \mathbb{R}$ and non-negative weights $w_i, i = 1, \ldots, n$, the expected runtime of the (1+1)-GA is in $\Theta(n \log n)$ [6]. This result, however, is only valid if $p_m = c/n$ for some constant $c$. Setting $c = 1$, this requirement resulted in the common recommendation for the mutation rate widely used in the field of evolutionary computation. It defines a *reference FPM*, as it defines the mu-

tation rate $p_m$ as a function of the problem feature decision space dimension $n$.

We applied FBSMBO to approximate a similar FPM based on experiments with the (1+1)-GA on the `OneMax` function

$$\texttt{OneMax} : \{0,1\}^n \to \mathbb{N}, \texttt{OneMax}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}_i.$$

The main research question is whether FBSMBO is capable of finding parameter configurations, which are competitive with the theoretical reference parameters with regard to the actual average runtime.

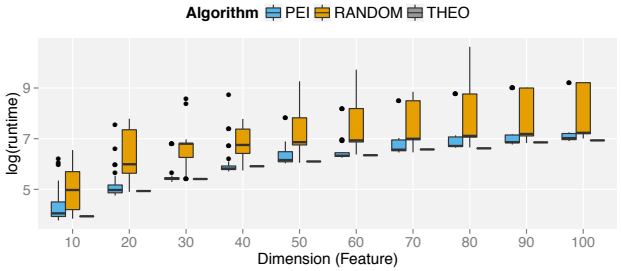



**Figure 2: Boxplots of the ERT of the (1+1)-GA with mutation rates $p_m$ set according to the FPMs obtained by FBSMBO with Profile-EI (PEI), random sampling (RANDOM), or the theoretical recommendation (THEO).**

To answer this question, we ran 30 experiments with PEI and random sampling. In each experiment, we started with an initial design of size 20 and performed 100 sequential optimization steps to learn the optimal feature parameter mapping. The training set consists of all $n$-dimensional `OneMax` problems with $n \in \{10, \ldots, 100\}$, as preliminary experiments indicated, that for $n < 10$ the effect of the initialization resulted in a too high relative variance. During the training phase, the number of evaluations used by the GA until finding the optimum solution was used as cost function $c$. For the a-posteriori evaluation of the resulting FPMs, the (1+1)-GA with the respective mutation probability performed 10 runs with at most $n^2$ function evaluations at each design point to estimate the *expected running time* (ERT) [11]. This resulted in 30 runtime estimates for each infill strategy. As a reference performance, we used the ERT of the (1+1)-GA with the theoretically recommended value $p_m = \frac{1}{n}$.

Figure 2 shows boxplots of the distributions of the log-transformed ERTs partitioned by problem dimension with only a part of the considered dimensions being shown for reasons of visual clarity. The superiority of the FPMs learned using the PEI infill strategy opposed to ones of random sampling can clearly be observed. The recommended parameters outperform the latter for all values of the feature parameter. The Wilcoxon test[2] for paired samples rejects the null hypothesis of equal runtimes in all cases.

Moreover, Figure 2 reveals that there is a low variance in the PEI recommendations. This indicates that the corresponding configurator determined similar FPMs in all experiments. In comparison, the FPMs learned with RANDOM show a high variance. Hence, the PEI-based training phase provides more robust FPMs.

This observation is supported by the visualization of the FPMs in Figure 3. Except for the experiments 2, 3, 10, 11, 14, 15, and 24, the PEI profile curves mimic the theoretical profile curves. In contrast, the profile curves obtained by a RANDOM sampling of the training data have a very high variance with no recurring pattern. In some cases, the theoretical trend of decreasing $p_m$ with increasing $n$ is even inverted. Whereas the focused sampling around the estimated FPM stabilizes model prodiction, RANDOM sampling may lead to penalized, nonconverging algorithm runs. These runs lead to a higher variation in the model landscape and thus to a deterioration of the prediction quality.

The restriction to one problem feature and one algorithm parameter makes it possible to visually analyze this issue over the sequential training process. Figure 4 shows a typical FPM based on the kriging model after the initial design, and its evolution after 50 respectively 100 sequential infill experiments. It is hard for the model to predict the FPM using only the initial design (Figures 4a and 4b).

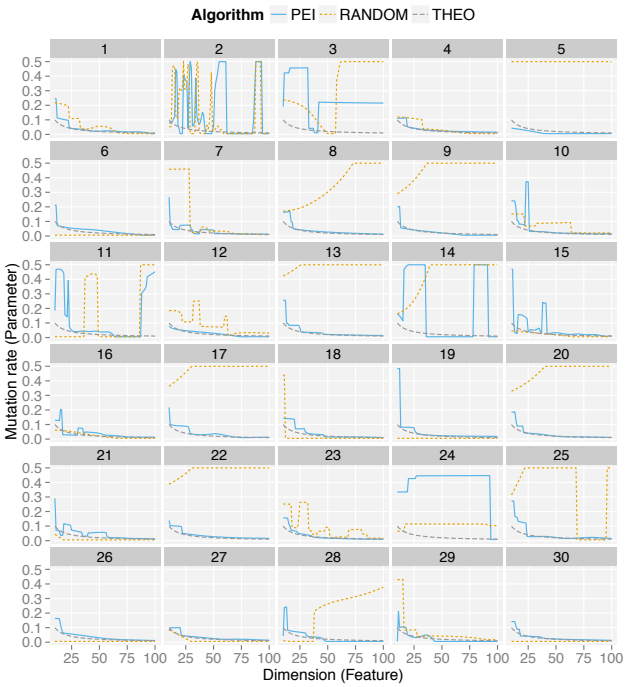
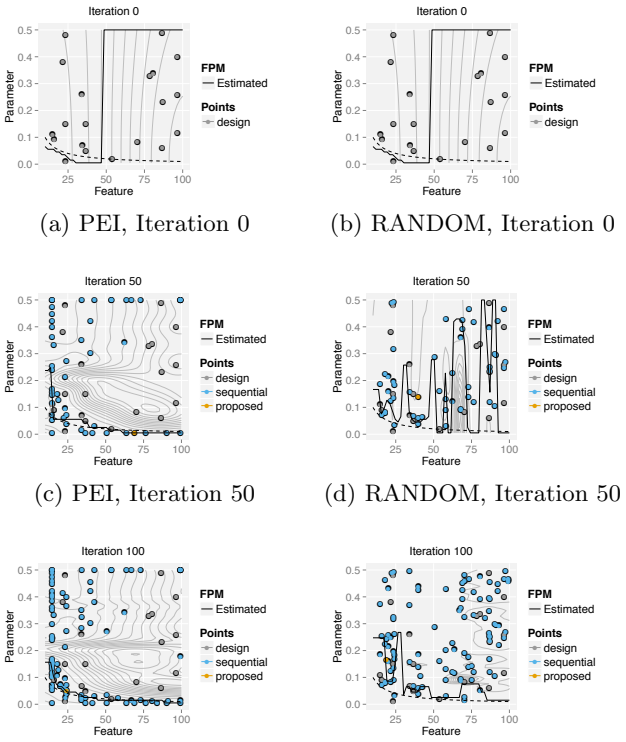



**Figure 3: FPMs obtained from theory (THEO) or the model based on the training data obtained by the respective sampling strategy (PEI or RANDOM) in run 1 to 30.**

However, after 50 iterations, the PEI-based FPM can predict the general trend of the reference profile curve $p_m = \frac{1}{n}$ (see Figure 4c). The increased sample density of the sequential points (blue) in the vicinity of the FPM is clearly visible. In contrast, the RANDOM-based FPM shows an oscillating behavior (see Figure 4d) indicating an instable model. Even if the oscillations can be reduced with 100 sequential samples, the approximation of PEI-based FPM remains better.

[2]We used the Wilcoxon test in this case, because the normality assumptions of the $t$-test were violated.

FBSMBO with a PEI-based infill strategy produces FPMs, which are 1) almost similar regarding runtime and 2) mimic the theoretically known FPMs. We can thus confirm its suitability based on this study.




(a) PEI, Iteration 0 (b) RANDOM, Iteration 0

(c) PEI, Iteration 50 (d) RANDOM, Iteration 50

(e) PEI, Iteration 100 (f) RANDOM, Iteration 100

**Figure 4: Typical sampling during the training phase of FBSMBO for the configuration of the (1+1)-GA. Model landscape, proposed FPM (solid black) and reference FPM (dashed black) after 0, 50 and 100 sequential experiments according to the PEI (left column) or random sampling (right column) infill strategy.**

## 3.3 Population size for the CMA-ES

In our second practical example, we consider the tuning of the well-known *Covariance-Matrix-Adaption Evolutionary Strategy* (CMA-ES) [10], which is one of the state-of-the-art optimizers for numerical black-box optimization. Specifically, we consider the offspring size $\lambda$, as the official CMA-ES implementation by Nikolaus Hansen proposes a default setting $\lambda = 4 + \lfloor 3 \log(n) \rfloor$ which can be considered as *reference FPM*. We term this setting the $\lambda$-rule in the following. Again $n$ denotes the number of decision space dimensions and is considered as the only problem feature. In contrast to the former GA experiment, the $\lambda$-rule comes with no theoretical guarantees and is of heuristic nature. For instance, Hansen et al. showed by empirical means that for highly multimodal functions $\lambda \gg n$ can be beneficial [12]. This result indicates that the FPM should be adapted for different kinds of test problems.

Our experiments aim at learning an FPM from the problem dimension $n$ to the offspring size $\lambda$ by applying FBSMBO to the CMA-ES on different subsets of the well-known noiseless testbed of the BBOB2009 benchmarking functions. More specific, we investigated the first three BBOB2009 groups of functions: the separable ones, the ones with low or moderate condition, and the high or ill-conditioned ones [11]. For each, models $\hat{m}$ for estimating the FPM are learned. The experimental setup is similar to the one described in the previous section, but we needed to reduce the number of experiments due to an increased computational effort of a single experiment. Concretely, after the initial design of size 10 we reduced the budget to 40 sequential experiments based on PEI and RANDOM sampling strategy. Again 30 independent experiments with different initial designs were performed on each BBOB group.

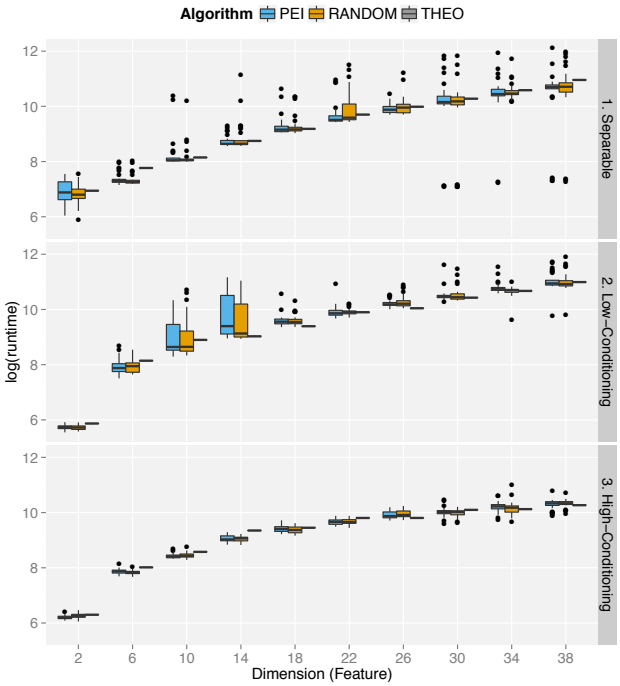



**Figure 5: Boxplots of the ERT of the CMA-ES with offspring size $\lambda$ set according to the FPMs obtained by FBSMBO with Profile-EI (PEI), random sampling (RANDOM), or the heuristic $\lambda$-rule (THEO).**

The instance training set consists of all functions from the respective BBOB group with dimensions $n \in \{2, \ldots, 40\}$ that have a BBOB instance ID $i \in \{1, 2, 3\}$, where $i$ denotes the specific variant of the test function. To estimate the ERT [11] of the CMA-ES, we ran it five times on each design point with a captime of $200,000$ function evaluations. If the global optimum was not approximated up to a tolerance of 0.03 within the given time budget in any of the five runs[3], the execution was penalized by a value of $3 \cdot 200,000$.

In Figure 5, boxplots of the resulting ERT distributions are shown in groups according to exemplary problem dimensions and the distinct BBOB groups. All strategies are capa-

[3]The tolerance was set in order to have a sufficiently high number of successful (unpenalized) experiments within the maximum budget (cf. Table 2). Higher budgets were not feasible due to restrictions with regard to the computational time required for the experiments.

ble of identifying the increase of $\lambda$ with increasing dimension. In general, no significant differences between the $\lambda$-rule and the FPMs obtained by PEI and RANDOM can be observed with respect to the median performance. Nevertheless, the results of the sampling strategies are not satisfying due to the high variation in the ERTs – in particular on the second BBOB group.

| Group | function | success | failure |
|---|---|---|---|
| group 1 | Büche-Rastrigin | 3 (0.9) | 320 (99.1) |
| | Ellipsoidal | 288 (95.7) | 13 (4.3) |
| | Linear Slope | 249 (84.1) | 47 (15.9) |
| | Rastrigin | 27 (9.9) | 246 (90.1) |
| | Sphere | 307 (100.0) | 0 (0.0) |
| group 2 | Attractive Sector | 372 (100.0) | 0 (0.0) |
| | Rosenbrock (original) | 372 (95.6) | 17 (4.4) |
| | Rosenbrock (rotated) | 350 (95.6) | 16 (4.4) |
| | Step Ellipsoidal | 87 (23.3) | 286 (76.7) |
| group 3 | Bent Cigar | 299 (100.0) | 0 (0.0) |
| | Different Powers | 315 (100.0) | 0 (0.0) |
| | Discus | 303 (100.0) | 0 (0.0) |
| | Ellipsoidal | 272 (94.1) | 17 (5.9) |
| | Sharp Ridge | 197 (67.0) | 97 (33.0) |

**Table 2: Absolute and relative (percentage) number of successes (finding the global minimum in at least one of the five runs) and failures (running until the captime in all five runs) for the functions of the considered BBOB groups.**

For further analyzing this variation, Table 2 shows an overview of the successful and failed runs separated by BBOB group and test function. In the table, "failed" means that the global minimum was not reached up to the desired tolerance within the captime in any of the five repetitions. Each group – groups one and two in particular – contains at least one function which is rarely minimized successfully. In addition, the ERTs of the CMA-ES on the distinct functions vary in orders of magnitude. Hence, it is possible that identical design points show extremely distinct ERT values. The feature vector $\boldsymbol{\delta}$ and the BBOB classification do not provide sufficient information about the characteristics of each instance. The intra-group heterogeneity is high. For instance, the first group of separable functions consists of the unimodal Sphere function as well as the highly multimodal Rastrigin and Büche-Rastrigin functions with approximately $11^n$ local optima. The noise level of the cost function to be learned by the model is thus extremely high. In addition, the penalization forces discontinuities within the response surface. The underlying Gaussian process model of FBSMBO, hence, has to cope with a high noise variance and irregular response values. Both aspects deteriorate the prediction quality of the model. As the latter issue could be addressed by using log- or rank-transformation of the performance values before model fitting [19], the former aspect requires the use of more powerful instance features [4]. We will investigate this in near future.

For improving the model quality, we reduced the set of training instances considerably and tried to learn FPMs for the Sphere function only. As a consequence, inhomogeneities within the training instances no longer exist. The observed ERTs for this setup are shown in Figure 6. It can be observed that the ERTs of the CMA-ES can be improved in comparison to the $\lambda$-rule for almost all problem dimensions. This observation holds for the FPMs obtained by both infill strategies. The PEI-based FPMs, however, still improve the ERTs by a statistically significant amount for all problem dimensions.

The reasons for these improvements are shown in Figure 7. The computed FPMs differ considerably from the general $\lambda$-rule, while still showing a monotonically increasing trend with higher problem dimensions. Whereas the FPM recommends offspring sizes around $\lambda = 5$ for $n = 10$ and between $\lambda = 6$ and $\lambda = 9$ for $n = 38$, the $\lambda$-rule would recommend $\lambda = 6$ for $n = 10$ and $\lambda = 15$ for $n = 38$. Again, the FPMs obtained by the PEI-based training data show a more consistent trend than the ones with random sampling. They also maintain monotonicity in almost all cases.

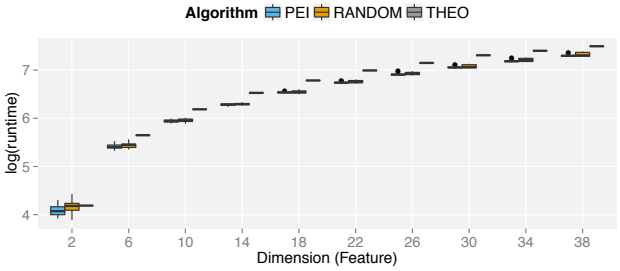



**Figure 6: Boxplots of the ERT of the CMA-ES with offspring size $\lambda$ set according to the FPMs obtained by FBSMBO with Profile-EI (PEI), random sampling (RANDOM), or the heuristic $\lambda$-rule (THEO) on the sphere functions.**

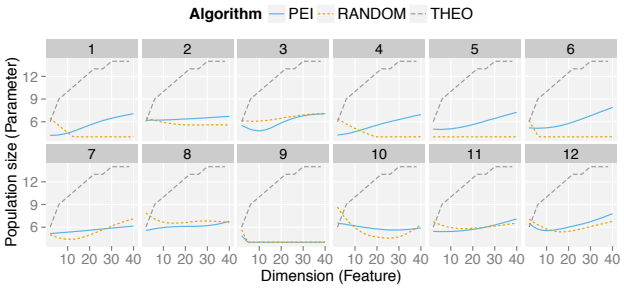



**Figure 7: FPMs obtained from theory (THEO) or the model based on the training data obtained by the respective sampling strategy (PEI or RANDOM) in run 1 to 12 of the CMA-ES on the sphere function.**

## 4. CONCLUSIONS

In this paper, we proposed how to use a surrogate model to learn a mapping from features describing the characteristics of problem instances to the respective optimum control parameters of the optimization algorithm. Our experiments revealed that the concept of profile expected improvement is capable to guide the sequential sampling process for refining the joint surrogate model. The feature parameter mapping obtained from the so-trained model can significantly increase the quality of the resulting configurator. In addition, practical experiments with the CMA-ES highlighted the requirement to find features that are meaningful with regard to the

algorithm's performance. In this context, ELA features [17] might be a good starting point.

In the presented case studies, we only learned mappings from a single feature to a single continuous parameter. In the future, we plan further experiments to confirm the suitability and the potential of our approach for the general case of multiple features and algorithm parameters. Clearly, using more than just a single feature for the mapping should have beneficial effects. Further options for improving the configurator are transformations of the performance values prior to the modeling step.

In contrast to the tuning of continuous algorithm parameters, so-called algorithm configuration is more complicated. For instance, categorical and dependent parameters have to be considered. It should be noted that our ideas, as they mainly deal with the definition of an appropriate acquisition function, can in principle be applied to more complex parameter spaces. Forthcoming work will investigate the FBSMBO approach to configure algorithms including more complex parameters by exchanging the currently used continuous kriging model with a better suited model for mixed spaces (e.g. random forests).

# 5. REFERENCES


[1] T. Bartz-Beielstein, C. Lasarczyk, and M. Preuss. Sequential parameter optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2005, 2-4 September 2005, Edinburgh, UK*, pages 773–780, 2005.

[2] M. Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*. Springer, 1st ed. 2005. 2nd printing edition, 2009.

[3] B. Bischl, M. Lang, O. Mersmann, J. Rahnenführer, and C. Weihs. BatchJobs and BatchExperiments: Abstraction mechanisms for using R in batch environments. *Journal of Statistical Software*, 64(11), 3 2015.

[4] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuss. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proc. of the 14th GECCO conference*, pages 313–320, 2012.

[5] S. P. Coy, B. L. Golden, G. C. Runger, and E. A. Wasil. Using experimental design to find effective parameter settings for heuristics. *Journal of Heuristics*, 7(1):77–97, 2001.

[6] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1–2):51–81, 2002.

[7] A. E. Eiben and S. K. Smit. Evolutionary algorithm parameters and methods to tune them. In Y. Hamadi, E. Monfroy, and F. Saubion, editors, *Autonomous search*, pages 15—36. Springer, Berlin Heidelberg, 2012.

[8] A. I. J. Forrester, A. J. Keane, and N. W. Bressloff. Design and analysis of 'noisy' computer experiments. *AIAA Journal*, 44(10):2331–2339, 2006.

[9] D. Ginsbourger, J. Baccou, C. Chevalier, F. Perales, N. Gerland, and Y. Monerie. Bayesian adaptive reconstruction of profile optima and optimizers. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):490–510, 2014.

[10] N. Hansen. The CMA evolution strategy: a comparing review. In J. A. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer, Berlin, Heidelberg, 2006.

[11] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2009.

[12] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In X. Yao et al., editors, *Proc. of PPSN VIII*, volume 3242 of *LNCS*, pages 282–291, 2004.

[13] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. A. Coello Coello, editor, *Proc. of LION-5, Rome*, pages 507–523. Springer, 2011.

[14] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(13):455–492, 1998.

[15] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC – instance–specific algorithm configuration. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proc. 19th European Conference on Artificial Intelligence*, pages 751–756. IOS Press, 2010.

[16] P. Koch, B. Bischl, O. Flasch, T. Bartz-Beielstein, C. Weihs, and W. Konen. Tuning and evolution of support vector kernels. *Evolutionary Intelligence*, 5(3):153–170, 2012.

[17] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory landscape analysis. In *Proc. of the 13th GECCO conference*, pages 829–836, New York, NY, USA, 2011. ACM.

[18] O. Mersmann, M. Preuss, and H. Trautmann. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Proc. of PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 73–82. Springer, 2010.

[19] T. Wagner and S. Wessing. On the effect of response transformations in sequential parameter optimization. *Evolutionary Computation*, 20(2):229–248, 2012.

[20] L. Xu, H. Hoos, and K. Leyton-Brown. Hydra: Automatically configuring algorithms for portfolio-based selection. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, 2010.