PROGRESS: Progressive Reinforcement-Learning-Based Surrogate Selection

Stefan Hess[1], Tobias Wagner[1], and Bernd Bischl[2]

[1] Institute of Machining Technology (ISF), TU Dortmund University, Germany
{hess,wagner}@isf.de
[2] Faculty of Statistics, TU Dortmund University, Germany
bischl@statistik.tu-dortmund.de

Abstract. In most engineering problems, experiments for evaluating the performance of different setups are time consuming, expensive, or even both. Therefore, sequential experimental designs have become an indispensable technique for optimizing the objective functions of these problems. In this context, most of the problems can be considered as a black-box. Specifically, no function properties are known a priori to select best suited surrogate model class. Therefore, we propose a new ensemble-based approach which is capable of identifying the best surrogate model during the optimization process by using reinforcement learning techniques. The procedure is general and can be applied to arbitrary ensembles of surrogate models. Results are provided on 24 well-known black-box functions to show that the progressive procedure is capable of selecting suitable models from the ensemble and that it can compete with state-of-the-art methods for sequential optimization.

Keywords: model-based optimization, sequential designs, black-box optimization, surrogate models, kriging, efficient global optimization, reinforcement learning

1 Introduction

The optimization of real-world systems based on expensive experiments or time-consuming simulations poses an important research area. Against the background of increasing flexibility and complexity of modern product portfolios, such kinds of problems have to be constantly solved. The use of surrogate (meta)-models $\hat{f}$ for approximating the expensive or time-consuming objective function $f : \boldsymbol{x} \to y$ represents an established approach to this task. After determining the values of f for the points $\boldsymbol{x}$ of an initial design of experiments, the surrogate model $\hat{f}$ is computed and then used for the further analysis and optimization. Here, we consider deterministic, i.e., noise-free minimization problems. In such a scenario, the former approach has a conceptual drawback. The location of the optimum can only roughly be determined based on the initial design. A high accuracy of the optimization on the model does not necessarily provide an improved quality with respect to the original objective function. As a consequence, the resources expended for the usually uniform coverage of the experimental region for the approximation of the global response surface may be spent more efficiently in order to increase the accuracy of the surrogate in the regions of the actual optimum.

A solution to this problem is provided by sequential techniques, called efficient global optimization (EGO) [16], sequential parameter optimization [1] and sequential designs [24] within the different disciplines. Sequential techniques do not focus on an approximation of a global response surface, but on an efficient way to obtain the global minimum of the objective function f. After evaluating a sparse initial design in the parameter space, much smaller than the actual experimental budget, the surrogate model is fitted and proposes a new point which is evaluated on the original function f then. The point is added to the design and the procedure is repeated until the desired objective value has been obtained or the experimental budget has been spent.

For the design of a sequential technique, the choice of the surrogate model is a crucial decision. Whereas resampling techniques [3] can be used to estimate the global prediction quality in the classical approach, the optimization capabilities of a model have to be assessed in the sequential approach. Therefore, this capability is not necessarily static. Some models may be suited to efficiently identify the most promising basin in the beginning, whereas others are good for refining the approximation in the final stage of the optimization.

In this paper, we tackle the model selection problem for sequential optimization techniques. Hence, the proposed optimization algorithm utilizes a heterogeneous ensemble of surrogate models. An approach to solve the progressive model selection problem is proposed as central scientific contribution. It is designed to identify which models are most promising at a certain stage of the optimization. Preference values are used to stochastically select a surrogate model, which in turn proposes a new design point in each iteration of the algorithm. Based on the quality of this design point, the rating of the model is adjusted by means of reinforcement learning techniques. The procedure is general and can be performed with arbitrary ensembles of surrogate models.

In the following, an overview of related research is provided by means of a brief review of the literature in Sect. 2. Details of the applied methods are presented in Sect. 3 and the actual PROGRESS algorithm is described in Sect. 4 based on these foundations. In Sect. 6, its results on the benchmark specified in Sect. 5 are presented and discussed. The main conclusions are summarized in Sect. 7 and an outlook for further research in this area is introduced.

2 Review

In the following review of the literature we mainly restrict the focus to sequential optimization techniques using ensembles of surrogate models in which the selection or combination of the models for the internal optimization is dynamically adjusted. A more general survey of sequential optimization algorithms has been recently provided by Shan et al. [31]. For the special class of kriging-based optimization algorithms exploiting the uncertainty estimation for a trade-off between local and global search, we refer to the taxonomy of Jones [15].

Ensemble-based sequential optimization procedures can be classified in three basic concepts:

1. All surrogate models are individually optimized and (subsets of) the design points are evaluated on the original objective f.
2. The predictions of all surrogate models are aggregated and their combined value is used for selecting the next design point.
3. A single surrogate model is used in each iteration. The selection of the model is based on dynamically adjusted scores or probabilities.

The first concept is particularly designed for applications in which speed-ups from parallelization can be expected. For instance, Viana [35] applied four different surrogate models in parallel to optimize engineering design problems. It was shown that the resulting procedure is often superior to a sequential optimization only relying on kriging. An application to surrogate-assisted evolutionary algorithms was reported by Lim et al. [20]. In their approach, the best solutions of each surrogate model are evaluated on the original function.

The second concept represents a general procedure combining predictions using an ensemble of surrogate models. Here, individual predictions are often aggregated via a weighted average. One major distinction of the techniques in this concept class is whether the technique can only be applied to surrogate models of the some basic type or whether a completely heterogeneous ensemble set is possible. The latter case is obviously preferable because of its increased generality. An early example for the former approach was presented by Hansen et al. [12] and relied upon combinations of different types of neural networks. In a similar manner, Ginsbourger et al. [9] proposed ensembles of kriging models based on different correlation functions.

An early approach for calculating weights for aggregating individual model predictions form a heterogeneous set was based on so-called Bayes factors [17], which from a Bayesian viewpoint denote the conditional probability of a surrogate being the true model given the current training data. More heuristic approaches for weight calculation based on cross-validated model accuracy were proposed and analyzed by Goel et al. [10]. The same means were also applied to select a subset of the ensemble within a tournament selection [37]. For each fold of a cross-validation, a representative surrogate was determined. The mean over the predictions of the selected models was used to guide the sequential optimization. An approach focusing on the combination and tuning of different surrogates was presented by Gorissen et al. [11]. They aggregated the active models of the ensemble by using the mean over the predictions. Since the key aspect of their work was the design of an evolutionary approach for tuning and selecting the models, the evaluation mainly focuses on the prediction quality of the resulting approach after the tuning. A comparison with other ensemble-based sequential techniques was not performed.

Algorithm 1: Model-based optimization

```
1 Let $f$ be the black-box function that should be optimized;
2 Generate initial design set $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$;
3 Evaluate $f$ on design points: $y_i = f(\boldsymbol{x}_i)$;
4 Let D=$\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$;
5 while stopping criterion not met do
6     Build surrogate model $\hat{f}$ based on $D$;
7     Get new design point $\boldsymbol{x}^*$ by optimizing the infill criterion on $\hat{f}$;
8     Evaluate new point $y^* = f(\boldsymbol{x}^*)$;
9     Extend design $D \leftarrow D \cup \{(\boldsymbol{x}^*, y^*)\}$;
```

Currently, only few related approaches exist for the third concept of ensemble techniques. Its main advantage is that it constitutes a very general approach which also allows many heterogeneous models to be integrated into the ensemble since only one model has to be fitted and optimized per iteration. Thus, the selected model can be subject to a more time-consuming tuning to specifically adapt it to the objective function. Friese et al. [8] applied and compared different strategies to assess their suitability for sequential parameter optimization, among them also ensemble-based methods using reinforcement learning. However, these methods were used in a rather out-of-the-box manner, without specifically adapting the generic reinforcement learning techniques to the problem at hand to exploit their full potential. Some of the potential problems, as well as enhancements to overcome them, will be discussed in this paper. Another variant of the approach was applied in the context of operator selection in evolutionary algorithms by Da Costa et al. [5].

3 Methods

In this section, the methodological foundations of our algorithm are introduced. First, the general concept of a model-based optimization (MBO) algorithm is described. Then the multi-armed bandit problem from reinforcement learning which is later transferred to the progressive model selection problem in MBO is presented.

3.1 Model-based Optimization

Response surface models are a common approach in cases where the budget of evaluations available for the original function is severely limited. As this surface can be explored with a much higher amount of evaluations, the optimum of the so-called infill criterion can be accurately approximated using standard optimization methods. This generic MBO approach is summarized in Algorithm 1. The stages of proposing new points and updating the model are alternated in a sequential fashion.

In the following, the steps of the generic MBO algorithm are discussed and some details are provided.

1. For the initial design, many experimental design types are possible, but for nonlinear regression models usually space-filling designs like Latin hypercube sampling (LHS) are used, see [4] for an overview. Another important choice is the size of the initial design. Rules of thumb are usually somewhere between $4d$ and $10d$, the latter being recommended by [16].
2. As surrogate model, kriging was proposed in the seminal EGO paper [16] because it is especially suited for nonlinear, multimodal functions and allows local refinements to be performed, but basically any surrogate model is possible. As presented in Sect. 2, also more sophisticated approaches using ensembles methods have been applied within MBO algorithms.
3. The infill criterion is optimized in order to find the next design point for evaluation. It measures how promising the evaluation of a point $\boldsymbol{x}$ is according to the surrogate model. One obvious choice is the direct use of $\hat{f}(x)$. For kriging models, the expected improvement (EI) [23] is commonly used. It factors in the local model uncertainty in order to guarantee a reasonable trade-off between the exploration of the decision space and the exploitation of the already obtained information. These and other infill criteria have been proposed and assessed in several studies [15, 30, 36, 25].
4. As stopping criterion, a fixed budget of function evaluations, the attainment of a specified y-level, or a combination of both is often used in practice.

3.2 Reinforcement Learning

The model selection problem in MBO can be considered as a "multi-armed-bandit" reinforcement learning problem. Here, in each iteration t an action a_t has to be chosen from a given set of finite choices $\mathcal{A} = \{v_1 \ldots v_m\}$ and we could envision those choices to be arms of a casino slot machine. In MBO, this choice corresponds to selecting a regression model which in turn is used to propose the next design point. Depending on the action a_t, we will receive a reward r_t according to an unknown probability distribution, and our aim is to maximize summed rewards over time.

After we have obtained some information regarding the pay-offs from the different slot machines, we face the fundamental exploration-exploitation dilemma in reinforcement learning: Should we try to gather more information regarding the expected pay-off of the actions or should we greedily select the action which currently seems most promising? The problem becomes even more difficult if we assume nonstationary rewards, i.e., reward distributions that change over time. In this scenario we always have to allocate a significant proportion of selections for exploring the current situation.

Sutton [32] suggests several ways for balancing exploration and exploitation. One is a probability matching strategy called reinforcement comparison, where the actions are selected stochastically according to a vector $\boldsymbol{q}_t \in \mathbb{R}^m$ of preference values. The main idea is that a high reward should strongly increase the preference value / selection probability of an action, whereas a low reward should decrease it.

Let us assume, we are in iteration t and already have a preference vector $\boldsymbol{q}_t$. These preferences are then transformed to selection probabilities via a softmax function

$$\pi_{t,j} = \exp(q_{t,j}) \left(\sum_{k=1}^{m} \exp(q_{t,k}) \right)^{-1}, j \in \{1, \ldots, m\} \ . \tag{1}$$

Based on these probabilities, we select action a_t and receive its stochastic reward r_t. Assuming we are in a general nonstationary scenario, we now have to decide whether r_t is favourable or not. For this, we compare it with a reference reward $\bar{r}_t$, which encodes the expected, average pay-off across all actions at iteration t. Assuming we already have such an $\bar{r}_t$, the element of the preference vector $\boldsymbol{q}_t$ for the chosen action $a_t = v_j \in \mathcal{A}$ is now updated, while the preferences for all other actions stay the same:

$$q_{t+1,k} = \begin{cases} q_{t,k} + \beta[r_t - \bar{r}_t], & \text{if } k = j \\ q_{t,k} & \text{else.} \end{cases} \tag{2}$$

Here, the strategy parameter β encodes the strength of the adjustment, i.e., the desired trade-off between exploitation (high β) and exploration (low β).

Finally, we update the the reference reward $\bar{r}_t$ via the following exponential smoothing formula

$$\bar{r}_{t+1} = \bar{r}_t + \alpha(r_t - \bar{r}_t) \ . \tag{3}$$

The strategy parameter $\alpha \in (0, 1]$ determines how much influence the current reward has on the reference reward, i.e, how much we shift the reference reward towards r_t. It thus reflects the assumed degree of nonstationarity in the reward distributions.

4 Algorithm

In this section, we address how the action selection by means of the reinforcement comparison can be exploited for model selection in MBO. Regarding models as selectable actions seems straightforward, but apart from that many technical details of the basic method in Sect. 3.2 have to be clarified or adapted. The reward will be based on the improvement in objective value obtained by the proposed x^*. The sum of rewards over time then measures the progress made during optimization. Thereby, the main idea is that models which generated larger improvements in the past should likely be preferred in the future.

Instead of using an expected improvement criterion we directly optimize the response surface of the selected model, i.e., no local uncertainty estimation is used. Although this carries the risk of getting stuck in a local optimum for one model, it offers two important advantages:

1. It is possible to optimize this criterion for arbitrary regression models.

Algorithm 2: PROGRESS

```
1  Let $f$ be the black-box function that should be optimized;
2  Let $E = \{h_1, \dots, h_m\}$ be the regression models in the ensemble;
3  Generate initial design $\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$;
4  Evaluate $f$ on design $\forall i \in \{1, \dots, n\} : y_i = f(\boldsymbol{x}_i)$;
5  Let D=$\{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$;
6  for $j \in \{1, \dots, m\}$ do
7      Build surrogate model $\hat{h}_j$ based on $D$;
8      Select next promising point $\boldsymbol{x}^*(\hat{h}_j)$ by optimizing $\hat{h}_j$;
9      Evaluate new point $y^*(\hat{h}_j) = f(\boldsymbol{x}^*(\hat{h}_j))$;
10     Extend design set $D \leftarrow D \cup \{(\boldsymbol{x}^*(\hat{h}_j), y^*(\hat{h}_j))\}$;
11 Calculate vector of initial rewards $\boldsymbol{r}_0 \in \mathbb{R}^m$ using equation (5);
12 Initialize preference vector $\boldsymbol{q}_1 = \boldsymbol{r}_0$;
13 Initialize reference reward $\bar{r}_1 = median(\boldsymbol{r}_0)$;
14 Let $t = 1$;
15 while stopping rule not met do
16     Calculate model selection probabilities $\boldsymbol{\pi}_t$ from $\boldsymbol{q}_t$ using equation (1);
17     Sample model $h_j$ according to $\boldsymbol{\pi}_t$;
18     Build surrogate model $\hat{h}_j$ based on $D$;
19     Select next promising point $\boldsymbol{x}^*(\hat{h}_j)$ by optimizing $\hat{h}_j$;
20     Evaluate new point $y^*(\hat{h}_j) = f(\boldsymbol{x}^*(\hat{h}_j))$;
21     Extend design set $D \leftarrow D \cup \{(\boldsymbol{x}^*(\hat{h}_j), y^*(\hat{h}_j))\}$;
22     Calculate reward $r_t$ using equation (4);
23     Update preferences using equation (2) to obtain $\boldsymbol{q}_{t+1}$ ;
24     Update reference reward using equation (3) to obtain $\bar{r}_{t+1}$;
25     t ← t+1
```

2. By using a heterogeneous ensemble the models are likely to focus on different basins. The exploration thus emerges from a successful adaptation of the model selection.

The complete procedure of the PROGressive REinforcement-learning-based Surrogate Selection (PROGRESS) is shown in Algorithm 2. It is an enhanced instance of the generic Algorithm 1, whereby the methodical contributions are: the initialization phase (lines 6 to 13), the stochastic model selection (lines 16 to 17) and the preference vector and reference reward updates (lines 22 to 24). Details are provided in the following.

4.1 Rewards

Let y_{min} be the minimum response value of the design before the integration of $\boldsymbol{x}^*(\hat{h}_j)$. The reward of a chosen surrogate model h_j is then given by

$$r_t(h_j) = \phi(y_{min} - f(\boldsymbol{x}^*(\hat{h}_j))) \ , \tag{4}$$

where $\boldsymbol{x}^*(\hat{h}_j)$ is the next point proposed by model $\hat{h}_j$ and ϕ is a simple linear rescaling function which will be detailed in the next section. Thereby, it is possible that a model produces negative rewards. We intentionally use all available information, also of iterations in which no improvements in the optimization of f are achieved, as this happened a considerable amount of times in our experiments. Surrogate models that poorly approximate interesting regions of the the objective function are directly downgraded. Clipping rewards at zero would discard this information.

4.2 Initialization Phase and Reward Rescaling

Until now we have not defined how preference values and reference reward are initialized before the first iteration $t = 1$. When using the generic reinforcement learning approach for MBO, we have to keep two peculiarities in mind: First, there is a high potential for large rewards in the first sequential step, as no optimization of the objective has been performed so far – this fact could cause a strong overrating of the first selected model. Second, it is hard to decide on a reasonable initial value for the reference reward – particularly, if independence with respect to the scaling of the objective function is desired.

In the initialization phase all models of the ensemble $E = \{h_1, \ldots, h_m\}$ are fitted to the initial design, each one proposes a point, which in turn is evaluated by $\rho_j = f(\boldsymbol{x}^*(\hat{h}_j))$, $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_m)^T$. To obtain our initial reward vector $\boldsymbol{r}_0$ we scale these values to $[0, 1]$ by applying the linear transformation $r_{0,j} = \phi(\rho_j)$, with ϕ defined to be:

$$\phi(x) = \frac{x - \min(\boldsymbol{\rho})}{\max(\boldsymbol{\rho}) - \min(\boldsymbol{\rho})} \ . \tag{5}$$

As a consequence, the first reward is estimated for all models using a comparable setup, i. e., the y_{min} of the initial design. This tranformation ϕ is also applied to all upcoming rewards.

The initial reference reward $\bar{r}_1$ is defined to be the median of the transformed rewards as a robust representative. The initial vector of probabilities $\boldsymbol{\pi}_1$ is now simply the softmax transformation of $\boldsymbol{q}_1$.

4.3 Sequential update

In the sequential part of PROGRESS, a surrogate model is stochastically chosen according to the current probability vector $\boldsymbol{\pi}_t$, fitted to the current design and proposes a new point $\boldsymbol{x}^*(\hat{h}_j)$ by optimization of its response surface. After its reward has been determined based on equation 4, the original formulas of Sect. 3 can be used for updating the preferences and the reference reward. The algorithm stops when the stopping criterion (in most cases the budget of function evaluations) is met.

5 Experimental Setup

PROGRESS and all experiments in this article have been implemented in the statistical programming language R [26]. We analyzed the performance of our algorithm on the 24 test functions of the BBOB noise-free test suite [13], which is a common benchmarking set for black-box optimization. It covers a variety of functions that differ w.r.t. problem features like separability, multi-modality, ill-conditioning and existence of global structure. A summary of these functions and their respective properties is provided in Table 1. Their function definitions, box constraints and global minima were taken from the `soobench` R package [21]. The dimension of the scalable test functions was set to $d = 5$.

Table 1. Overview of the considered test functions and the problem features covered.

separable	low or moderate cond.	high cond. and unimodal	adequate glob. struct.	weak glob. struct.
1 Sphere	6 Attractive Sector	10 Ellipsoidal Function	15 Rastrigin	20 Schwefel
2 Ellipsoidal	7 Step Ellipsoidal	11 Discus Function	16 Weierstrass	21 Gallagher's Gaussian (101-me Peaks)
3 Rastrigin	8 Rosenbrock (original)	12 Bent Cigar	17 Schaffers F7	22 Gallagher's Gaussian (21-hi Peaks)
4 Bueche-Rastrigin	9 Rosenbrock	13 Sharp Ridge	18 Schaffers F7 (ill)	23 Katsuura
5 Linear Slope		14 Different Powers (rotated)	19 Composite Griewank Rosenbrock	24 Lunacek bi-Rastrigin

The regression models used in our PROGRESS ensemble and their respective R packages are listed in Table 2: A second order (polynomial) response surface model (RSM) [14], a kriging model with power exponential covariance kernel [29], multivariate adaptive regression splines (MARS) [6], a feedforward neural network [14] with one hidden layer, a random forest [14], a gradient boosting machine (GBM) [7] and a regression tree (CART) [14]. Table 2 also lists (constant) parameter settings of the regression models which deviate from default values and box constraints for parameters which were tuned prior to model fitting in every iteration of PROGRESS based on all currently observed design points (lines 7 and 17 in Alg. 2). To accomplish this, we used a 10-fold repeated cross-validation (5 repetitions) to measure the median absolute prediction error and minimized this criterion in hyperparameter space by CMAES with a low number of iterations. Integer parameters were treated by a rounding strategy and CMAES was always started at the point in hyperparameter space which was discovered as optimal during the previous tuning run of the same model (or a random point if no such point is available).

The response surfaces of the regression models were also optimized by CMAES with 100 iterations and $\lambda = 10d$ offsprings. This setting deviates from the literature recommendation for technical reasons as more parallel model evaluations (predictions) reduce the computational overhead and lead to a better global search quality with the same number of iterations. We performed 10 random CMAES restarts and one additional restart at the currently best point of the observed design points for further exploration of the search space and to reduce the risk of getting stuck in a local optimum.

Table 2. Surrogate models, R packages, parameters and tuning ranges.

model	R package	parameter	value / range	model	R package	parameter	value / range
RSM	`rsm` [18]	-	-	RF	`randomForest` [19]	`ntree`	$\{50, \ldots, 500\}$
NNET	`nnet` [34]	`size`	5			`mtry`	$\{1, \ldots, 5\}$
		`decay`	$[0, 0.5]$	MARS	`earth` [22]	`degree`	$\{1, 2, 3\}$
GBM	`gbm` [27]	`interaction.depth`	$\{4, \ldots, 8\}$			`penalty`	$\{2, 3, 4\}$
		`shrinkage`	$[0.025, 0.05]$			`nprune`	$\{1, \ldots, 10\}$
		`bag.fraction`	$[0.5, 0.8]$	CART	`rpart` [33]	-	-
		`n.minobsinnode`	1	KM	`DiceKriging` [28]	`nugget.estim`	`TRUE`

The learning parameters α and β of PROGRESS were manually tuned prior to the experiments. In order to not bias the results by an overtuning to the problems, these parameters were fixed to $\alpha = 0.1$ and $\beta = 0.25$ for all of the test instances. Thereby, our aim was to find a global parametrization leading to robust results and a successful adaptation of the selection probabilities.

We compared PROGRESS with an established global optimizer for expensive black-box functions, namely EGO based on a kriging with a power exponential kernel (implementation taken from the R package `DiceOptim` [28]; default parameters except for kernel). We also ran a random LHS to provide a baseline result. For PROGRESS we considered two variants, one with the above mentioned hyperparameter tuning and one without, where default settings for all regression models were set. All algorithms were allowed a budget of 200 function evaluations, from which 50 were allocated to an initial maximin LHS design. This is in accordance with the "$10 \cdot d$ rule" proposed in [16]. This initial design is shared by all MBO algorithms in our experiments within one replication to reduce variance in subsequent comparisons. All algorithm runs were replicated 20 times. In order to organize and parallelize our experiments the `BatchExperiments` R package [2] was used.

6 Results

In the following results we report the distance of the best visited design point to the global optimum in objective space as measure of algorithm performance. As we consider the optimization of the black-box function as the task to be solved, we do not report global prediction quality indicators of the internal models. Neither do we calculate expected run times to obtain a certain target level as we assume an a priori fixed budget of function evaluations. The performance distributions of the 20 runs of each algorithm on the 24 test functions of the benchmark are shown in Fig. 1 on a logarithmic scale. As the scales and hardness of the test functions varies considerably, no unique scale is used for the box plots either.

The most obvious result is the superiority of the sequential designs over the static random LHS. In all cases, except for functions 12 and 23, at least one of the three sequential algorithms outperforms the static approach. Test functions 12 and 23 can apparently not be approximated well enough by any of the considered regression models to provide helpful guidance for the optimization process. This might be due to the high condition number (function 12) or the

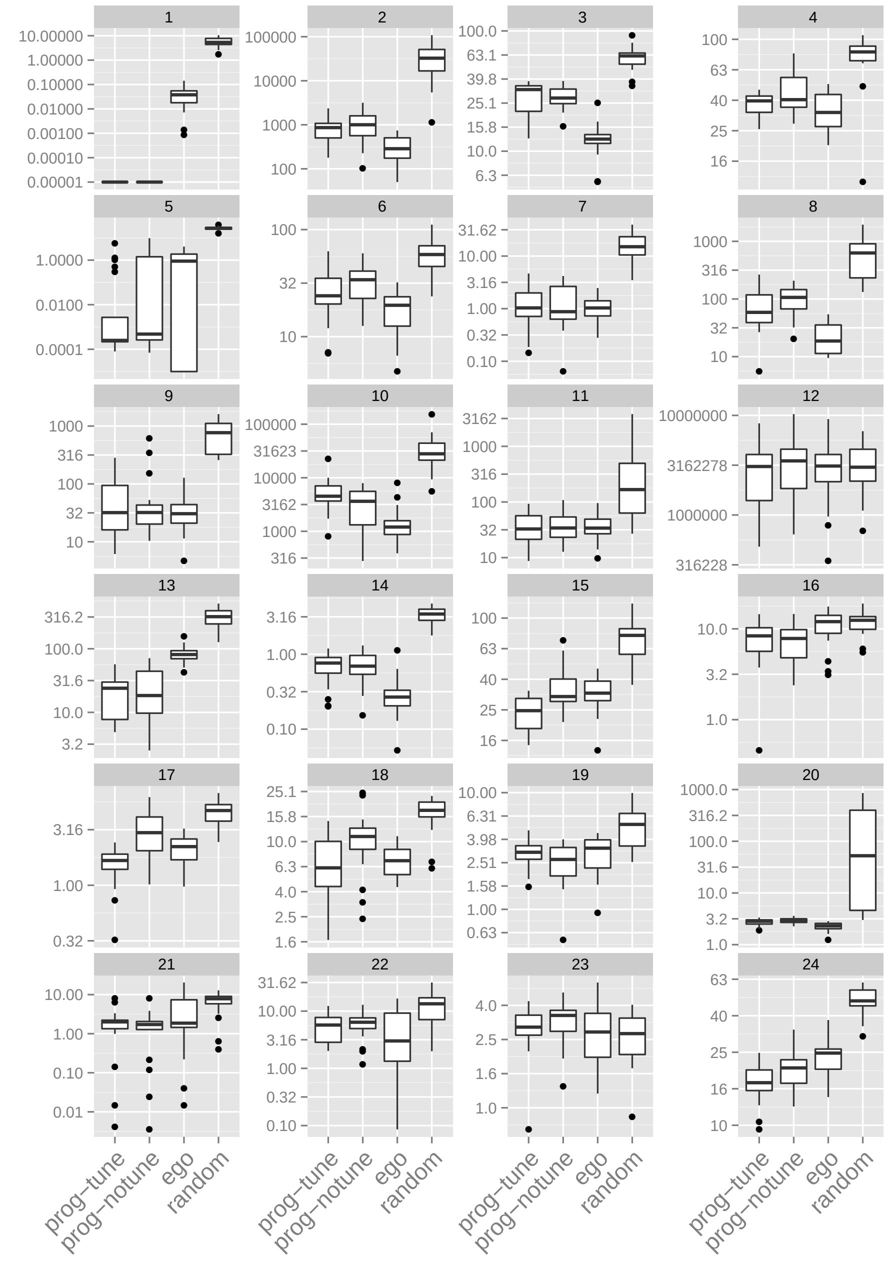

Fig. 1. Performance box plots per algorithm and test function. Displayed is the difference in objective space between best design point during optimization and global minimum on a log10 scale. PROGRESS is run in two variants, one with hyperparameter tuning of the selected surrogate model in each iteration and one without.

absence of global structure and an extreme number ($> 10^5$) of local optima (function 23). The comparison of the two versions of PROGRESS with and without hyperparameter tuning shows that both variants obtain similar results in most instances. There are cases (e.g., function 15, 17, 19), however, where hyperparameter tuning leads to a significantly better outcome.

The EGO algorithm and the use of kriging models represent the state-of-the-art in sequential designs. If PROGRESS can in general compete with EGO, it can be considered as successful, as it manages to detect suitable surrogate models within the strictly limited budget. If we assign equal importance to all test functions in the benchmark, none of the two algorithms clearly dominates the other. Whereas PROGRESS outperforms EGO on the functions 1, 5, 13, 15, 16, 17, 21, and 24, the opposite holds on the functions 2, 3, 6, 8, 10, 14 and 20. On the remaining test cases, both algorithms show no significant difference in performance. Hence, PROGRESS is competitive with EGO and is the preferable choice on about one third of the benchmark set. For this reason, the proposed procedure can be considered as successful.

To obtain a more detailed understanding of the principles behind the surrogate selection, we analyzed the progression of the selection probabilities in three exemplary runs. These are shown in Fig. 2. We also display the total number of selections per model during the whole run.

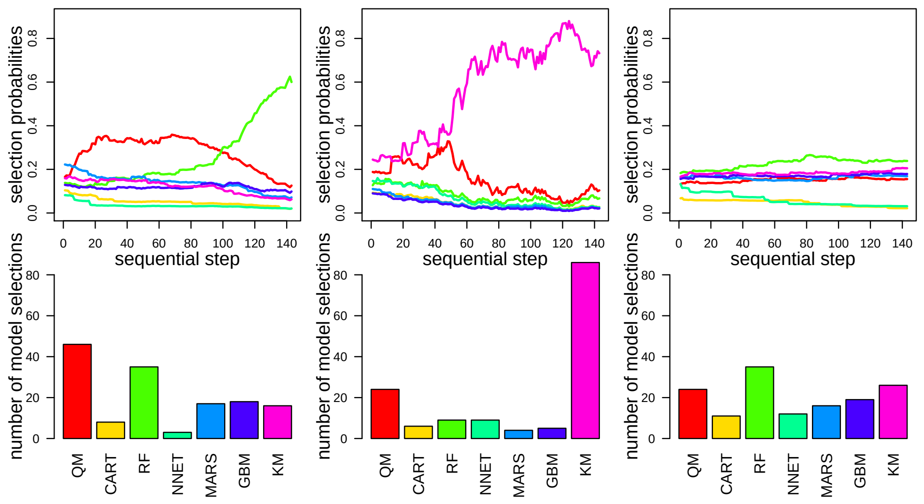

Fig. 2. Progression of selection probabilities and number of selections in PROGRESS (with tuning). From left to right: Lunacek bi-Rastrigin (24), Gallagher's Gaussian 21-hi Peaks (22) and Rastrigin (3).

In the first case shown in the left plot, a dynamic shift between a model capturing global trends (QM) and a more detailed model for local optimization

(RF) is accomplished during the optimization. While the former efficiently identifies the basins of the Lunacek bi-Rastrigin Function[3], the latter succeeds in guiding the search through the rugged area around the optimum. This synergy allows PROGRESS to significantly outperform EGO on this function. The center plot shows an experiment on Gallagher's Gaussian 21-hi Peaks function. As "kriging" is merely a different term for "Gaussian process", it is unsurprising this model is the best choice to approximate the 21 local optima of this test function. PROGRESS adapts its preference values after only a few iterations and learns to select the kriging model with high probability and therefore scores comparable to EGO. In the remaining plot on the right-hand side, the selection probabilities are shown for a test case where PROGRESS showed inferior results compared to EGO. Here, PROGRESS is not able to learn a superior model, but can only downgrade the apparently inappropriate models NNET and CART. A possible explanation for this problem might be the lack of balance between improvements and deteriorations on this function. For functions on which PROGRESS is inferior to EGO, such a problem can often be observed. We therefore consider this as one of the main starting points for future improvements.

7 Conclusions and Outlook

In this paper we presented PROGRESS, a new optimization algorithm for progressive surrogate selection based on reinforcement learning techniques. We demonstrated that the algorithm can compete with the established efficient global optimization (EGO) algorithm, which is the state-of-the-art for optimizing black-box problems within a strictly limited experimental budget. While EGO was superior in some cases of our considered benchmark cases and kriging therefore probably the best choice for approximating the response surface of these functions, PROGRESS outperformed EGO in roughly one third of cases due to the adaptive determination of the best fitting surrogate model during the different stages of the sequential optimization. Furthermore, the results indicate that hyperparameter tuning for the regression models in the ensemble can potentially improve the outcome of PROGRESS on some problems.

In future research, our algorithmic decisions will be further validated and/or refined. For instance, the scaling of the rewards and the calculation of the reference reward might offer potential for improvements. Moreover, more experiments could be performed to get a better understanding of the effects and interactions of the algorithm parameters. Alternatively, other reinforcement learning techniques can be implemented and benchmarked with the reinforcement comparison with respect to their suitability for an adaptive model selection. Finally, in order to enhance the time-efficiency of PROGRESS, the necessity of a hyperparameter tuning in every sequential loop will be further examined.

[3] The weak global structure of this function is rooted in the existence of two basis of almost the same size

8 Acknowledgements

This paper is based on investigations of the project D5 of the Collaborative Research Center SFB/TR TRR 30 and of the project C2 of the Collaborative Research Center SFB 823, which are kindly supported by the Deutsche Forschungsgemeinschaft (DFG).

References

1. Bartz-Beielstein, T., Lasarczyk, C.G., Preuss, M.: Sequential Parameter Optimization. In: McKay, B., et al. (eds.) Proc. 2005 Congress on Evolutionary Computation (CEC'05), Edinburgh, Scotland, pp. 773–780. IEEE Press, Los Alamitos, CA (2005)
2. Bischl, B., Lang, M., Mersmann, O., Rahnenfuehrer, J., Weihs, C.: BatchJobs and BatchExperiments: Abstraction Mechanisms for Using R in Batch Environments. Submitted to Journal of Statistical Software (2012a)
3. Bischl, B., Mersmann, O., Trautmann, H., Weihs, C.: Resampling Methods for Meta-Model Validation with Recommendations for Evolutionary Computation. Evolutionary Computation 20(2), 249–275 (2012b)
4. Bursztyn, D., Steinberg, D.M.: Comparison of Designs for Computer Experiments. Journal of Statistical Planning and Inference 136(3), 1103–1119 (2006)
5. DaCosta, L., Fialho, A., Schoenauer, M., Sebag, M.: Adaptive Operator Selection with Dynamic Multi-Armed Bandits. In: Proc. 10th Conf. Genetic and Evolutionary Computation (GECCO '08), pp. 913–920. ACM, New York, NY, USA (2008)
6. Friedman, J.: Multivariate Adaptive Regression Splines. The Annals of Statistics 19(1), 1–67 (1991)
7. Friedman, J.: Greedy Function Approximation: a Gradient Boosting Machine. The Annals of Statistics 29(5), 1189–1232 (2001)
8. Friese, M., Zaefferer, M., Bartz-Beielstein, T., Flasch, O., Koch, P., Konen, W., Naujoks, B.: Ensemble Based Optimization and Tuning Algorithms In: Hoffmann, F., Hüllermeier, E. (eds.): Proc. 21. Workshop Computational Intelligence, pp. 119–134 (2011)
9. Ginsbourger, D., Helbert, C., Carraro, L.: Discrete Mixtures of Kernels for Kriging-Based Optimization. Quality and Reliability Engineering International 24(6), 681–691 (2008)
10. Goel, T., Haftka, R.T., Shyy, W., Queipo, N.V.: Ensemble of Surrogates. Struct. Multidisc. Optim. 33(3), 199–216 (2007)
11. Gorissen, D., Dhaene, T., Turck, F.: Evolutionary Model Type Selection for Global Surrogate Modeling. The Journal of Machine Learning Research 10, 2039–2078 (2009)
12. Hansen, L., Salamon, P.: Neural Network Ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(10), 993–1001 (1990)
13. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Tech. Rep. RR-6829, INRIA (2009), `http://hal.inria.fr/inria-00362633/en/`
14. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics. Springer, New York, USA (2009)

15. Jones, D.R.: A Taxonomy of Global Optimization Methods Based on Response Surfaces. Journal of Global Optimization 21(4), 345–383 (2001)
16. Jones, D., Schonlau, M., Welch, W.: Efficient Global Optimization of Expensive Black-Box Functions. Journal of Global Optimization 13(4), 455–492 (1998)
17. Kass, R.E., Raftery, A.E.: Bayes Factors. Journal of the American Statistical Association 90(430), 773–795 (1995)
18. Lenth, R.V.: Response-Surface Methods in R, Using rsm. Journal of Statistical Software 32(7), 1–17 (2009)
19. Liaw, A., Wiener, M.: Classification and Regression by randomForest. R News 2(3), 18–22 (2002)
20. Lim, D., Ong, Y.S., Jin, Y., Sendhoff, B.: A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation. In: Thierens, D., et al. (eds.) Proc. 9th Annual Genetic and Evolutionary Computation Conference (GECCO 2007), pp. 1288–1295. ACM, New York, NJ (2007)
21. Mersmann, O., Bischl, B.: soobench: Single Objective Optimization Benchmark Functions (2012), `http://CRAN.R-project.org/package=soobench`, R package version 1.0-73
22. Milborrow, S.: earth: Multivariate Adaptive Regression Spline Models (2012), `http://CRAN.R-project.org/package=earth`, R package version 3.2-3
23. Mockus, J.B., Tiesis, V., Zilinskas, A.: The Application of Bayesian Methods for Seeking the Extremum. In: Dixon, L.C.W., Szegö, G.P. (eds.) Towards Global Optimization 2, pp. 117–129. Elsevier North-Holland, New York, NY (1978)
24. Myers, R.H., Montgomery, D.C., Anderson-Cook, C.M.: Response Surface Methodology. Wiley, Hoboken, NJ, 3rd edn. (2009)
25. Picheny, V., Wagner, T., Ginsbourger, D.: A Benchmark of Kriging-Based Infill Criteria for Noisy Optimization. Submitted to Structural and Multidisciplinary Optimization (2012)
26. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2012), `http://www.R-project.org/`, ISBN 3-900051-07-0
27. Ridgeway, G.: gbm: Generalized Boosted Regression Models (2012), `http://CRAN.R-project.org/package=gbm`, R package version 1.6-3.2
28. Roustant, O., Ginsbourger, D., Deville, Y.: DiceKriging, DiceOptim: Two R Packages for the Analysis of Computer Experiments by Kriging-Based Metamodeling and Optimization. Journal of Statistical Software 51(1), 1–55 (2012), `http://www.jstatsoft.org/v51/i01/`
29. Santner, T., Williams, B., Notz, W.: The Sesign and Analysis of Computer Experiments. Springer Verlag, New York (2003)
30. Sasena, M.J., Papalambros, P., Goovaerts, P.: Exploration of Metamodeling Sampling Criteria for Constrained Global Optimization. Engineering Optimization 34, 263–278 (2002)
31. Shan, S., Wang, G.G.: Survey of Modeling and Optimization Strategies to Solve High-Dimensional Design Problems with Computationally-Expensive Black-Box Functions. Structural and Multidisciplinary Optimization 41(2), 219–241 (2010)
32. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. Cambridge Univ Press (1998)
33. Therneau, T.M., port by Brian Ripley., B.A.R.: rpart: Recursive Partitioning (2012), `http://CRAN.R-project.org/package=rpart`, R package version 3.1-54
34. Venables, W.N., Ripley, B.D.: Modern Applied Statistics with S. Springer, New York, 4th edn. (2002)

35. Viana, F.A.C.: Multiple Surrogates for Prediction and Optimization. Ph.D. thesis, University of Florida (2011)
36. Wagner, T., Emmerich, M., Deutz, A., Ponweiser, W.: On Expected-Improvement Criteria for Model-Based Multi-Objective Optimization. In: Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G. (eds.) Proc. 11th Int'l. Conf. Parallel Problem Solving From Nature (PPSN XI). LNCS, vol. 6238, pp. 718–727. Springer, Berlin (2010)
37. Wichard, J.D.: Model Selection in an Ensemble Framework. In: International Joint Conference on Neural Networks, pp. 2187–2192 (2006)