Ensemble Based Optimization and Tuning Algorithms

Martina Friese, Martin Zaefferer,
Thomas Bartz-Beielstein, Oliver Flasch,
Patrick Koch, Wolfgang Konen, Boris Naujoks

Forschungsstelle CIOP
Fachhochschule Köln
Steinmüllerallee 1, 51643 Gummersbach
Tel.: (02261) 8196-0
E-Mail: {Martina.Friese@fh-koeln.de | Martin.Zaefferer@smail.fh-koeln.de}

1 Introduction

Sequential parameter optimization (SPO)[1] is a state-of-the-art tuning methodology for optimization algorithms. It has the following properties:

i) Use the available budget (e.g., simulator runs, number of function evaluations) sequentially, i.e., use information from search-space exploration to guide the search by building one or several meta models, e.g., random forest, linear regression, or Kriging. Choose new design points based on predictions from the meta model(s). Refine the meta model(s) stepwise to improve knowledge about the search space.
ii) Try to cope with noise by improving confidence. Guarantee comparable confidence for search points.
iii) Collect and report tuning process information for exploratory data analysis.
iv) Provide mechanisms both for interactive and automated tuning.

The SPO Toolbox (SPOT) provides standardized interfaces, which enable the integration of several meta models in a convenient manner [2]. Naturally, the question arises, which meta model should be used during the tuning process. Instead of recommending one meta model only, we will analyze an alternative approach: Provide an effective and efficient policy to dynamically combine several meta models to one better, i.e., more reliable meta model. Two research questions are dealt with in this study:

(Q.1) How to dynamically select the *best* meta model amongst an ensemble of meta models?
(Q.2) Based on evaluations of several models and its related predictions and errors (uncertainties), how can the best *combination* be determined?

These two questions are classical exploration—exploitation problems, which has been discussed in the literature for several decades and in different settings (scheduling, design of clinical trials, search).

To tackle these two questions, this experimental study analyses the behavior of two approaches in the SPO framework, namely (i) *single evaluation approaches*, i.e., one model is build in each step, and (ii) *multiple evaluation approaches*, i.e., several models are build in parallel. The assumption, that model building is computationally cheap compared to real function evaluations, justifies the evaluation of several models in parallel. Combining elements from sets of test functions, meta models, and decision rules increases the combinatorial complexity. To discover first trends, we use a very limited set of five test

Original Publication:
Friese, Martina, Martin Zaefferer, Thomas Bartz-Beielstein, Oliver Flasch, Patrick Koch, Wolfgang Konen, and Boris Naujoks. "Ensemble based optimization and tuning algorithms." In: Hoffmann and Hüllermeier, Proc. 21. Workshop Computational Intelligence, pp. 119-134. 2011.

Table 1: Fourteen different ensemble approaches used in this study

Name	Type	Abbreviation
Round Robin	single	sRR
Randomized Choosing	single	sRC
Greedy	single	sG
ϵ-Greedy $\epsilon = 0.25$	single	sEG
SoftMax	single	sSM
Bayesian Learning Automaton	single	sBL
Greedy	multi	mG
ϵ-Greedy	multi	mEG
Ranking	multi	mR
Weighted Ranking	multi	mWR
Quality Driven Grading	multi	mQG
Weighted Averaging	multi	mWA
Weighted Averaging Squared	multi	mWAs
Alternating Recommendation	multi	mAR

functions (Branin, Mexican Hat, Rastrigin, Rosenbrock, and Six Hump), which will be extended in further studies. Meta models are based on three variants of Gaussian process models (Kriging), random forest, and support vector machines. Fourteen different ensemble based approaches are considered in this study. Design of experiments and analysis of variance tools were used to perform a statistical analysis.

In the following section the ensemble approaches, that we used in this study, are described. Section 3 explains the experimental setup, including a short presentation of the meta models that are used as a foundation for the ensembles. Also a detailed description of the objective functions used for this study is given. Section 4 presents results derived from the experiments, which are then statistically analyzed in section 5 and discussed in section 6. Finally section 7 provides an outlook on future work. Note, results from our research are not restricted to tuning problems, they are also of great relevance in different domains (e.g., active learning and bagging).

2 Ensemble Approaches (Policies)

A comprehensive introduction to ensemble-based approaches in decision making is given in [3]. In this section we distinguish between two groups of ensemble approaches: The first group's approaches only choose and build one single meta model in a single SPO step. The second group's approaches build all models, and use the derived information to decide which output to use. The policies from this study are summarized in table 1

2.1 Single Evaluation Approaches

The approaches described as Single Evaluation are based on methods used to solve multi-armed bandit problems [4]. Multi-armed bandit problems are a well known area of research. They are decision problems where a player has to decide, which bandit to play to

maximize his reward. This can be mapped to the application of SPO, where a bandit represents a meta model and the SPO algorithm has to decide which meta model to choose in each sequential step.

Thus single evaluation approaches only build and evaluate one model in each step. The choice in the subsequent step is then based on a success/failure quality criterion (Bernoulli distributed feedback), unless the choice is completely random or predefined like in the Round Robin or Random Choice approaches. Success means that SPOT found a new and better function value with the help of the chosen model, failure indicates that no progress was made.

Some of the algorithms mentioned below are for stationary problems, some also work for non stationary problems. It is likely that our application is not perfectly stationary, since the number of data points available for model building rises throughout a SPOT run. Some models might excel in the beginning, others at the end of the run. However since the number of steps (or decisions) done in each run is rather low, this non-stationarity might be hard to grasp. Therefor we consider approaches of both types in here.

2.1.1 Round Robin

The meta models used in this ensemble approach are chosen in a circular order independent of their previously achieved gain, starting with the first model in the list of models.

2.1.2 Randomized Choosing

As another baseline comparison Random Choice is used, where the meta model to be used in each step is selected randomly from the list of available models. Again the previous success of the model is not a factor.

2.1.3 ϵ-Greedy

A greedy decision, is a decision that chooses exploitation over exploration. So in each step the greedy decsion would be to choose a meta model that provided the most reward so far. The ϵ-Greedy algorithm chooses the best model with $P = \epsilon$ and with a probability of $P = 1 - \epsilon$ one of the remaining models is chosen. In this way the ϵ parameter can be used to balance between exploitation and exploration. The exact choice of this parameter however is hard to recommend, and also the less greedy choices are treated all the same, even though they might have varying success. Still the ϵ-Greedy can prove to be surprisingly good when compared to other algorithms [5].

In our study we use two settings for epsilon, $\epsilon = 1$ and $\epsilon = 0.75$. Thus we compare a rather greedy version with a completely greedy version. We refer to the completely greedy version simply as Greedy, and the other as ϵ-Greedy.

2.1.4 SoftMax

The SoftMax strategy uses a probability vector where each element represents the probability for a corresponding meta model to be chosen. The probability vector is updated by using a boltzman distribution, depending on the reward received for the chosen models.

$p(i) = \frac{e^{Q(i)/\tau}}{\sum_{k=1}^{K} e^{Q(k)/\tau}}$

Parameter τ defines how much influence a difference in rewards has on the probability vector. K is the number of meta models, $p(i)$ is the probability for model m_i to be chosen and $Q(i)$ is weighted average reward for each model. The weighted average reward is computed by using a parameter $\alpha \in [0, 1]$ that defines how fast the algorithm forgets, thereby managing to deal with non-stationary problems:

$Q_{t+1}(i) = Q_t(i) + \alpha[r_{t+1}(i) - Q_t(i)]$

Everytime a model is chosen and its success is evaluated, $Q(i)$ is updated with the above formula. The reward r_{t+1} is zero if no progress was made with the model, or one if it was successful.

The temperature parameter τ and the recency weighting parameter α have to be chosen initially.

The softmax algorithm and the weighted averaging for nonstationary problems are described by Sutton and Barto in [6].

2.1.5 Bayesian Learning Automaton

As proposed by Granmo [7], *Bayesian learning automatons* can be used to solve bandit problems which have Bernoulli distributed feedback. Granmo described an algorithm for a two-armed bandit, which can easily be expanded to a multi-armed bandit.

For each bandit (e.g. meta model in our case) the two integer values α and β (both initialized with 1) are tracked. In case of a reward, the models m_i associated α_i value is incremented, else β_i is incremented. When deciding which model is to be used in the current step, a random value is sampled from the beta distribution for each model, using α_i and β_i as shaping parameters of the distribution. The model with the highest sampled value is chosen.

In contrast to epsilon greedy and SoftMax algorithm, this approach has the advantage of not needing any parameters to be set.

2.2 Multiple Evaluation Approaches

The algorithms described in this section evaluate all meta models in each sequential step. The response of the ensemble meta model is than built dependent on the answer of the meta models, considering i.e. the error each model has made or its uncertainty.

2.2.1 Greedy

In this case the Greedy approach is slightly differently defined. For the single evaluation approaches, the greedy choice was to build and evaluate the most successful model. Here however all models are built, and the greedy decision is simply to choose the response of the model with the smallest error measure.
That means, in each sequential step the response of that meta model is chosen, which has the smallest mean squared prediction error on its prediction for the previous sequential step. This error is calculated as the mean of the squared differences between the predicted and actual function values.
Since there is no previous prediction done in the first sequential step, the calculated error is null for each meta model. In this initial step the meta model is chosen randomly.

2.2.2 ϵ-Greedy

Again the ϵ-Greedy approach simply differs from the Greedy approach in one point: that the greedy decision is not always made. With a probability $P = 1 - \epsilon$, a meta models response is chosen randomly, independent of its previous prediction error. With a probability of $P = \epsilon$ the meta model will be chosen greedy.

2.2.3 Ranking

This approach tries to combine the response of meta models to one response, that all meta models contributed to, rather than to choose one response. In order to do this all meta models are evaluated and their responses are ranked. The overall response is then built as the sum of the exponentiated ranks. The following formula describes how the different meta models are combined to one response:

$$\hat{Y} = \sum \exp\left(rank\left(\hat{y}_i\right)\right).$$

Here $\hat{y}_i$ denotes the prediction of the meta models, whereas $\hat{Y}$ denotes the combined response. The rankings of the single responses $\hat{y}_i$ are exponentiated to provide a combined ranking which prefers configurations that are evenly ranked by all meta models, and punishes configurations that are rated unbalanced.

2.2.4 Weighted Ranking

This approach differs from the previously named in that point, that here the ranks that are used to build the sum for the overall response are weighted. To determine an appropriate weight, the models mean squared prediction error of the last sequential step is scaled and used as a weight.

$$\hat{Y} = \sum \exp\left(rank\left(\hat{y}_i\right)/weight\right)$$

2.2.5 Quality Driven Grading

In this approach all meta models are fitted $n + 1$ times. Once on the complete data set and n times on reduced subsamples of the original data set. This procedure provides a clue towards the meta model's response underlying certainty. This knowledge about the Meta models uncertainty and the error it has made in previous predictions is then used to weight the models prediction on the actual data set. Thereupon the meta model with the best weighted response is chosen.

2.2.6 Weighted Averaging

The Weighted Averaging approach does not choose a specific models result but rather combines it by averaging. Since bad models should not deteriorate the overall result, we simply introduced a weighting scheme. Every models result for a single design point is weighted by its overall error, the sum over all models yields the final value assigned to the design point. Additionally we tested a setting were the error weight was squared to punish bad models harsher. Of course the result of the weighted averaging is not to be considered in the same scaling as the original data. But since SPO is here only used as an optimizer, this does not pose a problem.

The error measure used for this (and the next approach) is defined as the mean absolute error of the model built in the last step, over all points that have been evaluated on the objective function.

2.2.7 Alternating Recommendation

The concept of SPO allows in every sequential step not only one new suggestion of a design point to be evaluated on the target function, but multiple ones. So instead of chosing a model and using all results from that single model, the Alternating Recommendation ensemble considers all models build. In contrast to the averaging approach, each model stays untouched.

Ordered by the models error (computed as in the above approach), each model in turn can propose a new design point. The most accurate model recommends the first point (the one it prefers the most), the second best the second one, and so on. If each model recommended a point, the next round starts, where the "best" model can suggest its the second best point (if that point was not yet recommended by another model).

3 Experimental Setup

3.1 Meta Models Used During SPOT Runs

SPOT processes data sequentially, i.e., starting from a small initial design, further design points are generated using a meta model. Many meta models are available in R. Similar as for the design generators the user has the option of choosing between state-of-the-art meta models for tuning his algorithm or writing his own meta model and use it as a plugin for

Table 2: Six SPOT meta models used in this study

Type (R function)	Name of the SPOT plugin	Abbreviation
Gaussian processes (mlegp)	`spotPredictMlegp`	ML
Kriging surface estimate (Krig)	`spotPredictKrig`	KR
Gaussian processes (gausspr)	`spotPredictGausspr`	GP
Random forest (randomForest)	`spotPredictRandomForest`	RF
Support Vector Machine (ksvm)	`spotPredictKsvm`	KS

SPOT. The default SPOT installation contains several meta models. Table 2 summarizes meta models used for experiments described in this document. For the experiment those models were both tested individually, as well as in ensemble combinations.

3.1.1 Random Forest-based Parameter Tuning

The Random Forest method from the R package `randomForest` implements Breiman's algorithm, which is based on Breiman and Cutler's original Fortran code, for classification and regression [8]. It is implemented as a SPOT plugin, which can be selected via setting the command `seq.predictionModel.func` according to Table 2 in SPOT's configuration file.

3.1.2 Maximum Likelihood Estimates of Gaussian Processes

SPOT provides a plugin for the *Maximum Likelihood Estimation of Gaussian process* (`mlegp`) package which is available in R. The package `mlegp` finds maximum likelihood estimates of Gaussian processes for univariate and multi-dimensional responses, for Gaussian processes with product exponential correlation structures; constant or linear regression mean functions; no nugget term, constant nugget terms, or a nugget matrix that can be specified up to a multiplicative constant [9].

`mlegp` is implemented as a SPOT plugin, which can again be selected via setting `seq.predictionModel.func` according to Table 2 in SPOT's configuration file.

3.1.3 Support Vector Machines

For this study we added a plugin to SPOT that provides us with a Support Vector Machine implementation for regression in R. It provides an interface to the `ksvm` function as provided with the package `kernlab` [10], which is based on an algorithm for solving the SVM QP problem by John Platt [11]. For our experiments, the function `ksvm()` was used with default parameters.

3.1.4 Kriging surface estimate

Another plugin was added to SPOT that integrates the `Krig` method of the `fields` Package [12]. This method fits a surface to irregularly spaced data. The Kriging model

assumes that the unknown function is a realization of a Gaussian random spatial processes. For our experiments, the `Krig()` function was used with default parameters.

3.1.5 gausspr (kernlab)

The third meta model from the gaussian processes family in this study uses the `gausspr` function. This function is also part of the `kernlab` [10] package. The `gausspr()` function is called with default parameters as well.

3.2 Test Functions

Our main goal when choosing the test functions was to obtain a preferably small number of these, which cover a variety of different difficulty criteria. The following criteria were chosen beforehand:

1. The function's optimum does not lie at the origin.
2. The function is not symmetric.
3. The function is multi-modal.
4. The function has many local minima.

In addition, the functions should be well known in the optimization community to improve reproducibility and comparability of results.

The chosen test functions cover the different difficulty criteria. To gain some additional difficulty and stay consistent with SPOT's original area of application, we added fitness-proportional noise to all test functions. This is the most common case for real-world settings: values and variability both change together. We restricted ourselves to the two-dimensional instances of the test functions as the higher dimensional instances require a much higher budget of target function evaluations and thus a modified setup. The number of function evaluations was chosen as the termination criterion.

The list of chosen test functions is:

1. Branin function was chosen as a test function, because it is multimodal and not symmetric.
2. Six Hump function was chosen as a test function, because it is multimodal with many local minima. It is also not rotationally, but point symmetric around the origin.
3. Rosenbrock Function was chosen as a test function, because it is unimodal; The global minimum lies inside a long, narrow, parabolic shaped, slowly descending valley, what makes it even harder to find. The function is not rotationally but axially symmetric.
4. Rastrigin function was chosen as a test function, because it has a large number of local minima and only one global minimum. The function is not rotationally but axially symmetric.
5. Mexican Hat function was chosen as a test function, because it is multimodal with many local minima.

For more detailed information about the test functions used here (e.g. region of interest, formulas, optimal values) we refer to [13].

Table 3: SPOT Setup

SPOT Setup Parameter	Value
auto.loop.nevals	100
init.design.size	10
init.design.repeats	2
init.design.func	"spotCreateDesignLhd"
init.design.retries	100
spot.ocba	TRUE
seq.ocba.budget	3
seq.design.size	200
seq.design.oldBest.size	3
seq.design.new.size	4
seq.design.func	"spotCreateDesignLhd"

3.3 SPOT in a Nutshell

SPOT uses the available budget (e.g., simulator runs, number of function evaluations) sequentially, i.e., it uses information from the exploration of the search space to guide the search by building one or several meta models. Predictions from meta models are used to select new design points. Meta models are refined to improve knowledge about the search space. SPOT provides tools to cope with noise, which typically occurs when real world applications, e.g., stochastic simulations, are run. It guarantees comparable confidence for search points. Users can collect information to learn from this optimization process, e.g., by applying *exploratory data analysis* (EDA) [14, 15]. Last, but not least, SPOT provides mechanisms both for interactive and automated tuning [16, 17]. An R version of this toolbox for interactive and automatic optimization of algorithms can be downloaded from CRAN.[1] Programs and files from this study can be requested from the author.

Table 3 shows the settings used for SPOT. As the function is noisy, OCBA (Optimal Computational Budget Allocation) is used since it has shown superior results in previous studies [13]. For the test functions are only two dimensional the `auto.loop.nevals` parameter, which limits the number of function evaluations, is set to 100. Other settings are chosen due to experience of the authors.

3.4 Best Case Comparison

To get an impression of how good our models and ensemble approaches could possibly get, we introduce a Best Case to our experiments. There we do not use any kind of meta models, but instead use the noise target function itself as a meta model. In a way one could say we assume that this would be the "perfect" meta model because it completely represents the real target function (allthough a quadratic or linear meta model that directly points to the global optimum might rather be perfect in a optimization context).

Doing so we see how good our meta models (or ensembles) could possibly get if they actually were able to reproduce the target function, under the circumstances that our ex-

[1] `http://cran.r-project.org/web/packages/SPOT/index.html`

perimental setting presents. This is simply a comparison to better evaluate the results of the approaches we test. In plots and tables we will refer to this testing comparison as `TST`.

4 Experimental Results

We are comparing 18 algorithms (14 ensembles variants and 5 meta models) on five noisy test functions. A typical question at this point is "Which comparison method should be used?"

The first step of our analysis relies on EDA. EDA comprehends methods such as plotting the raw data, e.g., histograms, plotting simple statistics such as mean plots, standard deviation plots, box plots, and main effects plots of the raw data. Observations based on EDA help us to select adequate techniques for the statistical analysis that will be performed in the second step. Step two is described in Section 5.

To visualize our results we show two boxplots in Fig. 1. Both show the resulting data, scaled to an interval of values from zero to one, since the real range of values for each test function can be quite different. The first plot shows results for all experiments over all test functions. This demonstrates that while one could make out some small differences, the overall data set hardly allows to make any good conclusion about which algorithm performs best. It can however be said that ML seems to be the best of the single models. The second plot, illustrating results from Rosenbrock's function only, shows how the resulting data vary more strongly if viewed only on that single test functions. In this case ML also seems to perform best on this single test function, while this can not be verified with every other test function. Especially the results from Rastrigin show very little differences between the algorithms. Besides that it can be seen in both plots, that while ML seems to be reasonably good, there is still room for improvement, as the "best case comparison", i.e., `TST`, is the single experiment that always provides the best results.

5 Analysis

Our analysis comprehends statistical tools such as *analysis of variance* (ANOVA). First, we have to decide whether we want to compare results with a reference algorithm. This procedure is adequate if one well established algorithm is the gold standard. Given n algorithms, this techniques requires $n-1$ comparisons only. Otherwise, pairwise comparisons can be used. Note, the combinatorial complexity of pairwise comparisons is large. A standard approach from statistics reads as follows [13].

(S.1) Use classical analysis of variance to determine whether there are differences between the treatment means. Under normality assumptions, use ANOVA for performing one-way location analysis. Otherwise, Kruskal-Wallis Rank Sum Test or its equivalent for two groups, the Wilcoxon rank sum test can be used. [18].

(S.2) Next, if the answer from the first step is positive, analyze *which* means differ using *multiple comparison methods*. Under normality assumptions, *Tukey's Honest Significant Differences* (TukeyHSD) tests can be used. Otherwise, the *Dunnett-Tukey-Kramer Pairwise Multiple Comparison tests* is recommended. [19]

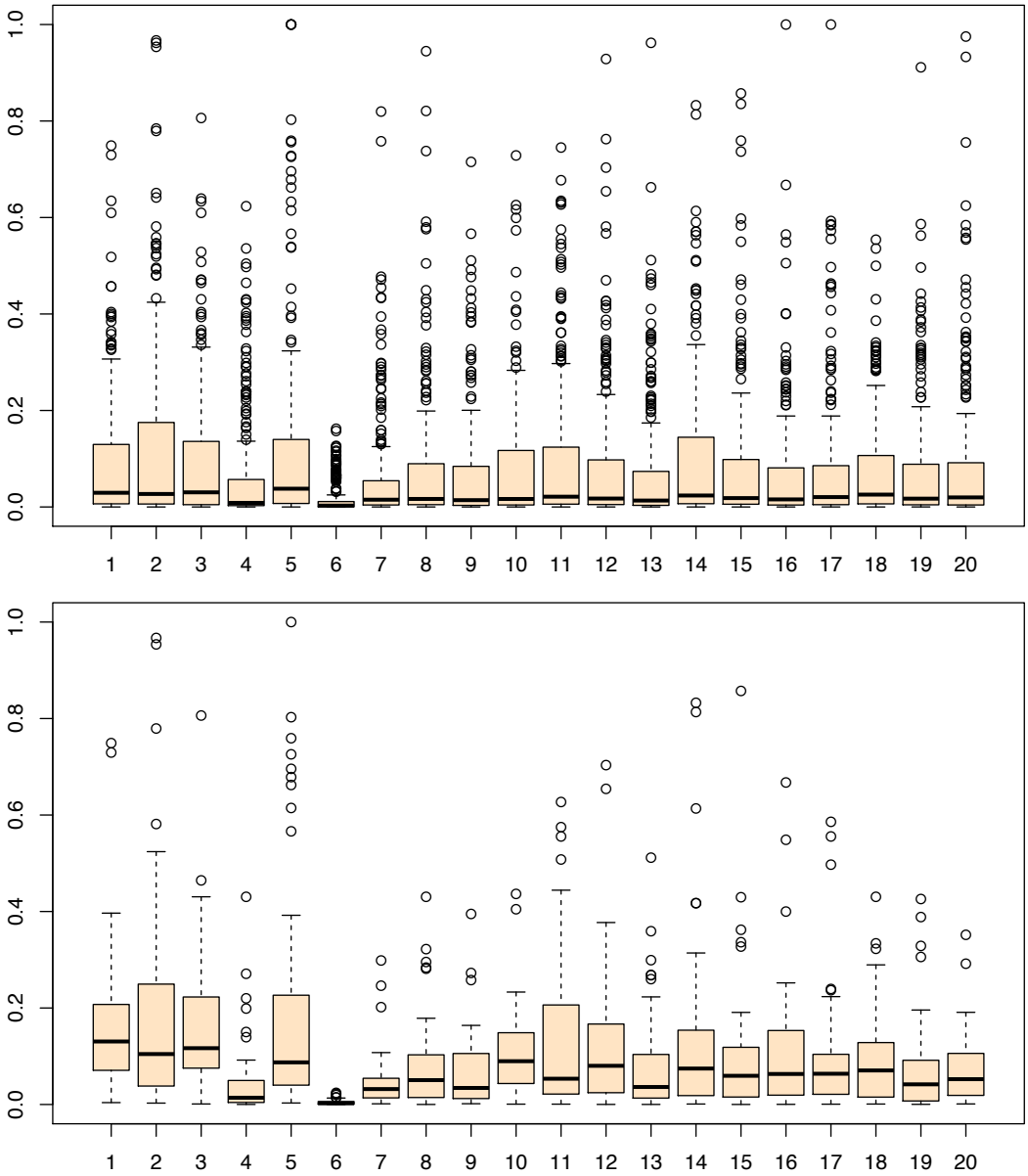

Figure 1: First boxplot shows summary of all normalized data over all functions, second plot shows data from Rosenbrock only. 1=GP, 2=KR, 3=KS, 4=ML, 5=RF, 6=TST, 7=mAR, 8=mEG, 9=mG, 10=mQG, 11=mR, 12=mWA, 13=mWAs, 14=mWR, 15=sBL, 16=sEG, 17=sG, 18=sRC, 19=sRR, 20=sSM

To handle the experimental complexity, our analysis is based on a multi-stage approach ("divide-and-conquer"). We partition the set of policies into three groups: single-evaluation, multiple-evaluation, and non-ensemble approaches. During the first stage, policies within each group are compared. During the second stage, the three best policies from each group are compared.

5.1 Single-Evaluation Approaches

A Kruskal-Wallis rank sum test of the null that the location parameters of the distribution of Y are the same in each group (sample) is performed first for each test function. Kruskal-Wallis rank sum test reveals that there is a significant difference for Branin (p-value = 0.06555) and MexicanHat (p-value = 0.0822), whereas differences for the Rastrigin, Rosenbrock, and Six Hump are not statistically significant:

```
              diff          lwr        upr     p adj
sEG-sBL  -0.04311923 -0.356219595 0.2699811 0.9987499
sG-sBL   -0.01101152 -0.324111883 0.3020888 0.9999985
sRC-sBL   0.08920151 -0.223898853 0.4023019 0.9642148
sRR-sBL   0.03498914 -0.278111229 0.3480895 0.9995469
sSM-sBL   0.31006374 -0.003036621 0.6231641 0.0539600*
sG-sEG    0.03210771 -0.280992652 0.3452081 0.9997025
sRC-sEG   0.13232074 -0.180779622 0.4454211 0.8306643
sRR-sEG   0.07810837 -0.234991999 0.3912087 0.9799607
sSM-sEG   0.35318297  0.040082610 0.6662833 0.0168528*
sRC-sG    0.10021303 -0.212887334 0.4133134 0.9416678
sRR-sG    0.04600065 -0.267099711 0.3591010 0.9982932
sSM-sG    0.32107526  0.007974897 0.6341756 0.0407568*
sRR-sRC  -0.05421238 -0.367312742 0.2588880 0.9962660
sSM-sRC   0.22086223 -0.092238133 0.5339626 0.3314643
sSM-sRR   0.27507461 -0.038025756 0.5881750 0.1215595
```

These results can be interpreted as follows: sSM-sBL, sSM-sEG, and sSM-sG differ in performance on Branin's function. To give an example, consider the ninth line from the output, which compares sSM and sEG:

```
sSM-sEG   0.35318297  0.040082610 0.6662833 0.0168528*
```

Since we are considering minimization problems, the positive value 0.35318297 indicates that sSm reached a higher function value than sEG. 95% confidence intervals and p-values are also calculated. Summarizing, we can state that sEG is best, sSM worst on Branin's function. Taking p-values and confidence intervals into consideration, we can easily detect significant differences. For example, the difference -0.04311923 from the first line (sEG-sBL) is not statistically significant.

A similar analysis, performed on results from the Mexican Hat function, reveals that sRC is worst, sRR best on Mexican Hat, and sEG performs reasonable good. Therefore, the ϵ-greedy policy sEG is chosen as the best single evaluation approach.

5.2 Multiple-Evaluation Approaches

Kruskal-Wallis rank sum tests for the multiple-evaluation approaches indicate that these policies differ in their performance on Branin (p-value = 0.0006006) and Rosenbrock (p-value = 0.003778). They show similar performances on Rastrigin, Mexican Hat, and Six Hump.

The statistical analysis reveals that mWR is outperformed by mAR, mEG, mWA, and mWAs. We can conclude that mWAs and mAR are the best policies, whereas mWR performs worst on Branin's function.

Considering Rosenbrock's function, the situation is a little bit more complicated: mAR performs better than mQG, mR, mWA, and mWR; mEG outperforms mR, mWAs performs better than mWR. Summarizing, we can conclude that mAR is a reasonable choice for Rosenbrock's function.

Finally, we conclude that the alternating recommendation policy mAR is the best policy.

5.3 Non-Ensemble Approaches

Simple EDA reveals that ML is the best meta model in our comparisons on Branin, Mexican Hat, Rosenbrock, and Six Hump. Interestingly, all meta models exhibit a similar performance on Rastringin's function. Random forest performs surprisingly good on Rastrigin. Therefore, ML was chosen as the best candidate for our final comparison.

5.4 Final Comparison

Our final comparison focuses on sEG, mAR, and ML. Kruskal-Wallis rank sum tests show significant differences on Rastrigin (p-value = 0.03759), Rosenbrock (p-value = 0.0006051), and SixHump (p-value = .006867). On Rastrigin's function, the multiple comparison tests produce the following output:

```
              diff       lwr       upr     p adj
mAR-ML  -0.5506559 -1.897131 0.7958195 0.5980376
sEG-ML  -1.0878970 -2.434372 0.2585785 0.1386036*
sEG-mAR -0.5372410 -1.883716 0.8092344 0.6129415
```

sEG outperforms ML on Rastrigin, whereas the remaining two comparisons do not reveal a clear winner. Interestingly, the situation is different on Rosenbrock's function.

```
               diff         lwr       upr     p adj
mAR-ML  -0.006053078 -0.23477911 0.2226730 0.9978378
sEG-ML   0.298016354  0.06929032 0.5267424 0.0068343*
sEG-mAR  0.304069432  0.07534340 0.5327955 0.0056251*
```

mAR and ML perform equally good, whereas sEG is outperformed by ML and mAR. Finally, we review results on Six Hump. Here, for sEG and mAR no difference in their performances could be detected. ML performs better than both ensemble approaches:

```
                diff           lwr         upr     p adj
mAR-ML   0.0043309488 -0.0003971800 0.009059078 0.0800071*
sEG-ML   0.0039823639 -0.0007457649 0.008710493 0.1172052*
sEG-mAR -0.0003485849 -0.0050767138 0.004379544 0.9833436
```

6 Discussion

Our research focuses on situations, in which ensemble-based approaches outperform classical approaches. Reflecting the experimental analysis, we have learned that combining several test problems blurs the significant differences which are visible on subsets. Therefore, we re-arranged our experimental setup and the corresponding analysis. Significant differences could be observed on single test functions. As expected, ML performs best on four of the five test functions. Interestingly, we discovered a different behavior on Rastrigin's function. This will give the starting point for further investigations. So, we did not find simple answers to (Q.1) and (Q.2), but we are able to formulated interesting follow-up questions.

One further observation is related to the Round-Robin policy: It performs reasonable well in several settings and deserves further attention in forthcoming experiments.

7 Summary and Outlook

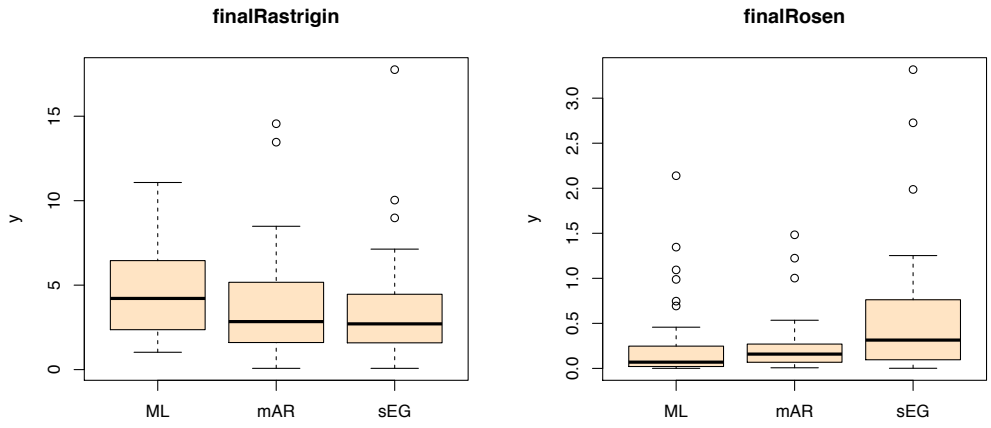

Figure 2: Box plots comparing final best approaches on Rastrigin (*left*) and Rosenbrock (*right*). These box plots illustrate a simple observation: the selection of the best algorithm (policy) depends on the test function. There are settings, in which ensemble based approaches are outperformed by single meta models, and vice versa

There is no general recommendation, that can be derived from results of this study. We learned, that even on simple test functions, ensemble-based approaches can outperform classical meta model approaches. However, combining results from several test functions may blur significant effects. We will consider special functions in our forthcoming experiments, rather than simply increasing the number of test functions in our portfolio.

The box plots from Fig. 2 nicely illustrate this situation: ML and sEG show a completely different behavior on Rastrigin and Rosenbrock.

Further topics that will be tackled in our experiments are as follows:

1. How to choose ϵ values for the greedy-search approach?
2. We will increase problem dimensions, and analyze objective functions with certain features to gain more insight into the working mechanism of ensemble-based approaches.
3. We will further analyze adaptive region of interest approaches, i.e., we do not use a fixed region of interest during the optimization.

Acknowledgements

This work has been supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grants FIWA (AIF FKZ 17N2309), MCIOP (AIF FKZ 17N0311), and by the Cologne University of Applied Sciences under the research focus grant COSA.

Literatur

[1] Bartz-Beielstein, T.; Parsopoulos, K. E.; Vrahatis, M. N.: Design and analysis of optimization algorithms using computational statistics. *Applied Numerical Analysis and Computational Mathematics (ANACM)* 1 (2004) 2, S. 413–433.

[2] Bartz-Beielstein, T.: SPOT: An R Package For Automatic and Interactive Tuning of Optimization Algorithms by Sequential Parameter Optimization. CIOP Technical Report 05/10, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science. URL `http://arxiv.org/abs/1006.4645`. Comments: Related software can be downloaded from http://cran.r-project.org/web/packages/SPOT/index.html. 2010.

[3] Polikar, R.: Ensemble based systems in decision making. *Circuits and Systems Magazine, IEEE* 6 (2006) 3, S. 21–45. URL `https://maanvs03.gm.fh-koeln.de/webstore/Classified.d/Poli06a.d/Poli06a.pdf`.

[4] Gittins, J.: Bandit Processes and Dynamic Allocation Indices. *J. R. Statist. Soc. B* **41** (1979), S. 148–164.

[5] Vermorel, J.; Mohri, M.: Multi-armed bandit algorithms and empirical evaluation. In: *In European Conference on Machine Learning*, S. 437–448. Springer. 2005.

[6] Sutton, R. S.; Barto, A. G.: *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. MIT Press. URL `http://www.cse.iitm.ac.in/~cs670/book/the-book.html`. 1998.

[7] Granmo, O.-C.: A Bayesian Learning Automaton for Solving Two-Armed Bernoulli Bandit Problems. In: *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*, S. 23–30. Washington, DC, USA: IEEE Computer Society. ISBN 978-0-7695-3495-4. 2008.

[8] Breiman, L.: Random Forests. *Machine Learning* 45 (2001) 1, S. 5 –32.

[9] Dancik, G. M.; Dorman, K. S.: mlegp. *Bioinformatics* 24 (2008) 17, S. 1966–1967.

[10] Karatzoglou, A.; Smola, A.; Hornik, K.; Zeileis, A.: kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software* 11 (2004) 9, S. 1–20. URL `http://www.jstatsoft.org/v11/i09/`.

[11] Platt, J. C.: Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In: *Advances in Large Margin Classifiers*, S. 61–74. MIT Press. 2000.

[12] Furrer, R.; Nychka, D.; Sain, S.: *fields: Tools for spatial data*. URL `http://CRAN.R-project.org/package=fields`. R package version 6.3. 2010.

[13] Bartz-Beielstein, T.; Friese, M.: Sequential Parameter Optimization and Optimal Computational Budget Allocation for Noisy Optimization Problems. CIOP Technical Report 02/11, Research Center CIOP (Computational Intelligence, Optimization and Data Mining), Cologne University of Applied Science, Faculty of Computer Science and Engineering Science. URL `http://maanvs03.gm.fh-koeln.de/webpub/CIOPReports.d/Bart11a.d/Bart11a.pdf`. 2011.

[14] Tukey, J.: The philosophy of multiple comparisons. *Statistical Science* 6 (1991), S. 100–116.

[15] Chambers, J.; Cleveland, W.; Kleiner, B.; Tukey, P.: *Graphical Methods for Data Analysis*. Belmont CA: Wadsworth. 1983.

[16] Bartz-Beielstein, T.; Preuss, M.: The Future of Experimental Research. In: *Experimental Methods for the Analysis of Optimization Algorithms* (Bartz-Beielstein, T.; Chiarandini, M.; Paquete, L.; Preuss, M., Hg.), S. 17–46. Berlin, Heidelberg, New York: Springer. 2010.

[17] Bartz-Beielstein, T.; Lasarczyk, C.; Preuss, M.: The Sequential Parameter Optimization Toolbox. In: *Experimental Methods for the Analysis of Optimization Algorithms* (Bartz-Beielstein, T.; Chiarandini, M.; Paquete, L.; Preuss, M., Hg.), S. 337–360. Berlin, Heidelberg, New York: Springer. 2010.

[18] Hollander, M.; Wolfe, D. A.: *Nonparametric Statistical Methods*. John Wiley & Sons. 1973.

[19] Dunnett, C.: Pairwise Multiple Comparisons in the Unequal Variance Case. *Journal of the American Statistical Association* 75 (1980), S. 796–800.