Algorithm Selection Based on Exploratory Landscape Analysis and Cost-Sensitive Learning

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

Olaf Mersmann
TU Dortmund, Germany
olafm@statistik.tu-dortmund.de

Heike Trautmann
TU Dortmund, Germany
trautmann@statistik.tu-dortmund.de

Mike Preuss
TU Dortmund, Germany
mike.preuss@tu-dortmund.de

ABSTRACT

The steady supply of new optimization methods makes the algorithm selection problem (ASP) an increasingly pressing and challenging task, especially for real-world black-box optimization problems. The introduced approach considers the ASP as a cost-sensitive classification task which is based on Exploratory Landscape Analysis. Low-level features gathered by systematic sampling of the function on the feasible set are used to predict a well-performing algorithm out of a given portfolio. Example-specific label costs are defined by the expected runtime of each candidate algorithm. We use one-sided support vector regression to solve this learning problem. The approach is illustrated by means of the optimization problems and algorithms of the BBOB'09/10 workshop.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global Optimization, Unconstrained Optimization; G.3 [**Probability and Statistics**]: Statistical Computing; I.2.6 [**Learning**]: Knowledge Acquisition

General Terms

Experimentation, Algorithms, Performance

Keywords

algorithm selection, evolutionary optimization, fitness landscape, exploratory landscape analysis, BBOB test set, benchmarking, machine learning

*Corresponding author

GECCO'12, July 7–11, 2012, Philadelphia, Pennsylvania, USA.

1. INTRODUCTION

The *algorithm selection problem* (ASP) (Rice, 1976) resembles the old desire to choose the right optimization algorithm for a not well-known optimization task without enumerating all available techniques. Although there are algorithms coping well with many problems, a cleverly constructed portfolio provides better efficiency than any single algorithm even on a small set of problems as, e.g., the BBOB test set, see the results of the related 2009 and 2010 competitions in Hansen et al. (2010) and Auger et al. (2010). Thus, the ASP can be regarded as still important and largely unsolved.

We suggest to employ *Exploratory Landscape Analysis* (ELA) techniques to approach this problem. The low-level features presented in Mersmann et al. (2011) are a first step to automatically and numerically characterize the landscape of a given fitness function. Section 2 gives an introduction into the ELA background and describes the employed features. The long term goal is to provide a method that is able to select a good algorithm for an arbitrary problem. As a first approach we use the 24 BBOB functions as a representative benchmark set.

Taking into account that some of the tested algorithms are not really competitive on any of the BBOB functions and that a practical portfolio should be as small as possible, we select 4 diverse optimizers from the ones that have been run in 2009 and 2010. Assuming we had an oracle, which would always select the correct algorithm for us given a test function, the chosen subset is optimal among all subsets of the same size. Furthermore, its performance is not far off from the one we would obtain by considering a selection oracle for all available algorithms.

Now a machine learning model has to be constructed that selects a well-performing algorithm from the portfolio based on the extracted ELA features. Usually this problem is tackled either by standard classification (given the features, predict the best algorithm) or regression (given the features, predict the performance measure for each algorithm, then select the algorithm with the best predicted value). But in our opinion both approaches fail to optimally solve the problem at hand: How do we score suboptimal selections and how can this information best be incorporated into the model fitting process? It is neither satisfying to minimize the classification error (because the penalty we incur for a wrong selection should depend on the gap in performance to the

optimal choice), nor do we have to accurately predict performance values given a feature vector (this will in general be impossible and we only care about the order of the predicted performances). For practical applications, we need to minimize the loss in runtime for the predicted algorithm when compared with the optimal available candidate. This implies some form of cost-sensitive learning. As the costs in the discussed scenario are different for each single data point, not many machine learning models are currently available that can solve such tasks with example-specific costs. We propose to use the recently published method of one-sided support vector regression (Tu and Lin (2010)), which is discussed in more detail in Section 3.

We are currently unaware of other work that combines automatically computed, numerical features with a selection strategy that incorporates the relevant costs of wrong selections for optimization.

Some interesting questions can be asked concerning the performance of our approach, and they are answered in the experimental study presented in Section 4:

- Can we generalize over instances? Meaning, can we train our model on only some of the BBOB function instances and predict well-performing algorithms for the remaining ones that we have never seen?
- Can we even generalize over functions? Not only generalizing over instances but transferring knowledge to completely unseen function types would be a much stronger result than generalizing only over instances. If the former can be done successfully but not the latter, this might indicate that the BBOB set is too small and its test functions are too different for effective usage in such learning problems as considered in this study.
- Do we also need the expensive ELA features, e.g., local searches, or can satisfying results also be achieved with only "cheap" features?

We feel that we probably have reached the limit of what can be achieved with the BBOB data alone with respect to algorithm selection and that more extensive experimental data is needed to move beyond the results presented here. This and related aspects are discussed in Section 5. Further discussion and an outlook on future research are given in 6.

2. EXPLORATORY LANDSCAPE ANALYSIS: USED FEATURES

Exploratory Landscape Analysis (ELA) aims at extracting characteristics for a given optimization problem prior to optimization which, in this work, form the basis for an algorithm recommendation. Preliminary approaches date back to measures related to fitness distance correlation (Jones and Forrest (1995)) but did not lead to satisfying results. The reason was mainly that only a single feature was used, which is by far not enough to capture and explain all possible differences in optimization problems. Consequently, the ELA approach is set-based and much more pragmatic: We define many features and then experimentally determine the ones that are actually useful for a given classification problem.

In Mersmann et al. (2010a) and Bartz-Beielstein and Preuss (2012) high-level features specified by human experts such as the level of modality or separability are introduced to reflect the problem characteristics. As these features depend on subjective assessment, experimental low-level features which are presented in Mersmann et al. (2011) can be automatically extracted by using systematic sampling of the decision space using a suitable space filling design. It was shown that these features not only match the expert designed high-level features but that they can be used to successfully predict the predefined function groups of the BBOB'09/10 function set with negligible errors.

The starting point for computing our low-level features is a data set of s different decision variable settings X^s with respective objective values Y^s generated by a random Latin hypercube (LH) design covering the decision space. We will denote the combination of search points and corresponding function values by $D^s = [X^s, Y^s]$. The size of the LH design increases linearly with the dimension d of the problem which is one suitable heuristic possibility to account for the increasing problem complexity, i.e., the number of initial points is chosen as $s = c \cdot d$, where c is a predefined constant.

The features can be grouped into five classes related to characteristics of the distribution of the objective function values (y - Distribution), the relative position of each objective value compared to the median of all values (Levelset), meta-modeling of the initial data set (Meta-Model), the estimated degree of convexity of the function (Convexity) as well as the assessment of multimodality by local searches starting from the initial design points (Local Search). Details are given in Mersmann et al. (2011). Each class contains several lower level sub-features which, in each case, can be generated using the same experimental data pool. All of these sub-features are listed in Table 1. The first three feature classes can be computed directly on the sample without additional evaluations and are denoted as "cheap" features in the following. The last two feature classes require additional function evaluations. In order to limit the required computational effort we strive to restrict the analysis to a low-level feature set which can be computed solely based on the initial design. However, it is currently not clear if using only "cheap" features is feasible when tackling the algorithm selection.

3. COST-SENSITIVE LEARNING AND ONE-SIDED REGRESSION

In this section we briefly introduce a general definition of cost-sensitive learning and present a solution based on one-sided support vector regression proposed by Tu and Lin (2010). The following section will show how to facilitate this modeling technique in order to select an efficient algorithm for continuous black-box optimization.

Supervised classification is concerned with creating predictors mapping from a feature space $\mathcal{X}$ (here $\mathbb{R}^p$) to a set of classes $\mathcal{Y} = \{1, \ldots, k\}$, also called labels. We usually assume a stochastic data-generating process, a distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{Y}$, and a data set $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$, whose observations have been independently drawn from $\mathcal{P}$. This data shall be used to learn the predictor.

Given a parametrized classification model $f(\mathbf{x}, \boldsymbol{\theta})$, where the vector $\boldsymbol{\theta}$ denotes the parameters to be determined, our task can be described as the minimization of the expected number of misclassifications

$$\min_{\boldsymbol{\theta}} \int_{\mathcal{X} \times \mathcal{Y}} [y \neq f(\mathbf{x}, \boldsymbol{\theta})] \, d\mathcal{P}(\mathbf{x}, y) \, . \tag{1}$$

A multitude of different machine learning algorithms exists for multiclass-classification (see Hastie et al. (2001)), and of

Feature group and name	Description
y distribution features:	
4 `distr.skewness_y`	skewness of the distribution of the function values
4 `distr.kurtosis_y`	kurtosis of the distribution of the function values
4 `distr.n_peaks`	estimation of the number of peaks in the distribution of the function values
Levelset features:	
5 `levelset.lda_mmce_{10,25,50}`	mean LDA misclassification error for function values split by 0.1, 0.25, 0.5 quantile (estimated by CV)
5 `levelset.lda_vs_qda_{10,25,50}`	`levelset.lda_mmce_{10,25,50}` divided by `levelset.qda_mmce_{10,25,50}`
6 `levelset.qda_mmce_{10,25,50}`	mean QDA misclassification error for function values split by 0.1, 0.25, 0.5 quantile (estimated by CV)
7 `levelset.mda_mmce_{10,25,50}`	mean MDA misclassification error for function values split by 0.1, 0.25, 0.5 quantile (estimated by CV)
Meta-model features:	
1 `approx.{linear,lineari}_ar2`	adjusted R^2 of the estimated linear regression model without and with interaction
1 `approx.linear_{min,max}_coef`	minimum and maximum value of the absolute values of the linear model coefficients
2 `approx.{quadratic,quadratici}_ar2`	adjusted R^2 of the estimated quadratic regression model without or with interaction
2 `approx.quadratic_cond`	maximum absolute over minimum absolute value of the quadratic term coefficients in the quadratic model
Convexity features:	
3 `convex.{linear,convex}_p`	estimated probability of linearity and convexity
3 `convex.linear_dev`	mean deviation from linearity
Local search features:	
8 `ls.n_local_optima`	number of local optima estimated by the number of identified clusters
8 `ls.best_to_mean_contrast`	minimum value of cluster centers divided by the mean value of cluster centers
8 `ls.{best}_basin_size`	proportion of points in the best cluster
8 `ls.mean_other_basin_size`	mean proportion of points in all clusters but the cluster with the best cluster center
8 `ls.{min,lq,med,uq,max}_feval`	0, 0.25, 0.5, 0.75 and 1 quantile of the distribution of #function evaluations performed during a single local search

Table 1: Low-level features; summary of sub-features within the feature classes. Leading numbers refer to the feature group the low-level feature belongs to. LDA, QDA and MDA denote linear, quadratic resp. mixture discriminant analysis. The two groups below the double line require additional evaluations.

course it is usually impossible to tackle (1) directly. As we do not know $\mathcal{P}$, we have to fit w.r.t. the training data; minimizing the number of misclassified examples is computationally intractable, so we have to approximate the idea expressed in (1) with something more efficient; and we often combine it with regularization to restrict the complexity of the model.

A more realistic scenario – especially if we associate specific actions with our predictions – might be to assume individual costs for a predicted class depending on the true label. The standard examples, where this is obviously reasonable and necessary, are labeling patients as "sick" or "healthy" and granting / rejecting credit applications. In such a case we assume a cost matrix G, where $G(i, j)$ expresses the cost of classifying an object of class i as class j. Now we naturally try to minimize

$$\min_{\boldsymbol{\theta}} \int_{\mathcal{X}\times\mathcal{Y}} G(y, f(\mathbf{x}, \boldsymbol{\theta}))\, d\mathcal{P}(\mathbf{x}, y) \tag{2}$$

instead of (1). Optimally solving this cost-sensitive problem for the multiclass case $k > 2$ is quite hard, and although a lot of proposed approaches exist, no consensus seems to have been reached what works best in the general case.

A further generalization allows the costs to depend on the example $\mathbf{x}$ and its features which is the scenario we are interested in. This is sometimes referred to as cost-sensitive learning with example-specific costs, see Turney (2000) for a very extensive taxonomy of cost-sensitive problems. Here, each observed $\mathbf{x}$ has an associated cost vector $\mathbf{c}$, so the available data set has the form $\{(\mathbf{x}^i, \mathbf{c}^i)\}_{i=1}^n$, where c_j^i specifies the cost of labeling example i as class j. Therefore, our distribution $\mathcal{P}$ is now defined on $\mathcal{X} \times \mathbb{R}^k$ and, as our goal still is to obtain a cost-optimal predictor, (2) changes to

$$\min_{\boldsymbol{\theta}} \int_{\mathcal{X}\times\mathbb{R}^k} c(f(\mathbf{x}, \boldsymbol{\theta}))\, d\mathcal{P}(\mathbf{x}, \mathbf{c})\,. \tag{3}$$

Tu and Lin (2010) have proposed to solve the problem (3) via one-sided support vector regression. They compare their method to the most common alternatives (not that many exist) and present convincing empirical results in addition to some theoretical guarantees. For these reasons we select it as our method of choice in this paper. We report the main ideas and formulas of their approach in the following paragraphs.

Construct k regression problem sets $S_j = \{(\mathbf{x}^i, c_j^i)\}_{i=1}^n$ from the observed data where $j \in \{1, \ldots, k\}$. Define $z_j^i = 1$, if c_j^i is the minimal entry in the vector $\mathbf{c}^i$, and therefore labeling $\mathbf{x}^i$ as class j is the best possible action, otherwise set $z_j^i = -1$. Our goal is to construct k regression models f_j, one for each S_j. Then we can estimate the cost vector $\hat{\mathbf{c}} = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}))^T$, given a feature vector $\mathbf{x}$, and select $\operatorname{argmin}_j \hat{c}_j$ as our predicted, hopefully not too pricey, label.

The important observation now is that we do not necessarily have to minimize the difference between $\hat{\mathbf{c}}$ and $\mathbf{c}$, e.g., by minimizing the squared loss $||\mathbf{c} - \hat{\mathbf{c}}||_2^2$ or absolute loss $||\mathbf{c} - \hat{\mathbf{c}}||_1$ in the regression. We only have to be concerned with the property that the true cost of the minimal entry of $\hat{\mathbf{c}}$ is as close as possible to the minimal entry of $\mathbf{c}$, because their difference expresses how much we score worse if we use our predictors f_j instead of directly accessing the unknown true cost vector $\mathbf{c}$ to build our decision. Therefore we choose $L(c_j^i, z_j^i, \hat{c}_j^i) = \max(0, z_j^i(\hat{c}_j^i - c_j^i))$ as our loss function. Meaning, we do not penalize underestimating the true cost of the optimal decision, which can never have a negative effect on selection, and we penalize overestimation linearly. For non-optimal labels we proceed vice versa for a similar reason. Tu and Lin (2010) prove that the true cost of the predicted label for an example $\mathbf{x}^i$ is always bounded from above by $\sum_{j=1}^k L(c_j^i, z_j^i, \hat{c}_j^i) + \min(\mathbf{c})$, further motivating this choice of L. As L is either 0 to the left or right of the origin of the real line, we call a regression based on such L "one-sided". Note, that L is not very different from the hinge loss used in binary support vector classification.

Let us now fix j and construct a regression model for S_j based on the above motivated loss L. Assuming (for now) a

linear model $f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T\mathbf{x}+b$ and that L_2-regularization of $\mathbf{w}$ makes sense, we define the following primal optimization problem:

$$\min_{\mathbf{w},b} \sum_{i=1}^{n} L(c_j^i, z_j^i, \mathbf{w}^T\mathbf{x}^i + b) + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{4}$$

Encoding the non-differentiability of L in constraints results in:

$$\begin{cases} \min\limits_{\mathbf{w},b} & \frac{\lambda}{2}||\mathbf{w}||_2^2 + \sum\limits_{i=1}^{n} \xi_i \\ \text{s.t.} & \xi_i \geq z_j^i(\mathbf{w}^T\mathbf{x}^i + b - c_j^i) \\ & \boldsymbol{\xi} \geq 0\,. \end{cases} \tag{5}$$

Note that both (4) and (5) are equal to respective formulations in binary support vector classification, if we use $z_j^i \in \{+1, -1\}$ for positive / negative labels and set all $c_j^i = 1$. Converting (5) into its dual version and employing the kernel trick by substituting all scalar products with a kernel function $k(\cdot,\cdot)$, we finally arrive at

$$\begin{cases} \min\limits_{\mathbf{w},b} & \frac{1}{2}\boldsymbol{\alpha}^T \text{diag}(\mathbf{z}) K \text{diag}(\mathbf{z})\boldsymbol{\alpha} + \mathbf{z}^T\boldsymbol{\alpha} \\ \text{s.t.} & \mathbf{z}^T\boldsymbol{\alpha} = 0 \\ & 0 \leq \boldsymbol{\alpha} \leq \frac{1}{\lambda}\,. \end{cases} \tag{6}$$

Here, K is the kernel matrix with $K(i,j) = k(\mathbf{x}^i, \mathbf{x}^j)$, $\boldsymbol{\alpha}$ is the vector of Lagrange multipliers and diag($\mathbf{z}$) is a diagonal matrix with entries $\mathbf{z} = (z_j^1, \ldots, z_j^n)$. Again, this is very similar to the dual problem of the standard support vector machine. It is a quadratic program that we can solve with the same techniques as the "classic" Support Vector Machine (SVM) problem. And as usual we obtain a formula for the intercept b from the Karush-Kuhn-Tucker conditions. Finally, a prediction for a new point $\mathbf{x}$ can be obtained by

$$f(\mathbf{x}, \boldsymbol{\alpha}, b) = \sum_{i \in \text{SV}} -\alpha_i z_j^i k(\mathbf{x}, \mathbf{x}^j) + b\,, \tag{7}$$

where the set SV indexes the support vectors, i.e., SV $= \{i : \alpha_i > 0\}$. As always, the kernel trick allowed us to transform an essentially linear model into a powerful non-linear one. We use an interior point solver provided by the R package kernlab (Karatzoglou et al. (2004)) to optimize (6).

4. ALGORITHM SELECTION FOR BBOB FUNCTIONS

A detailed description of the BBOB setup can be found in Hansen et al. (2009). It uses a balanced and unbiased sample of the published set of test functions from the field of black-box optimization. To assess the performance of an algorithm, the BBOB team proposes the use of the so called *expected running time* (ERT). This measure estimates the expected number of function evaluations required to achieve an accuracy of $\epsilon > 0$. For a given ϵ the ERT is defined as

$$\mathbf{E}\left\{RT(\epsilon)\right\} := \mathbf{E}\left\{N_{\text{eval}}^{\text{succ}}(\epsilon)\right\} + \frac{1 - \pi^{\text{succ}}(\epsilon)}{\pi^{\text{succ}}(\epsilon)} \mathbf{E}\left\{N_{\text{eval}}^{\text{fail}}(\epsilon)\right\},$$

where $N_{\text{eval}}^{\text{succ}}(\epsilon)$ denotes the number of function evaluations until the algorithm reaches the desired accuracy of ϵ, $N_{\text{eval}}^{\text{fail}}(\epsilon)$ denotes the number of function evaluations until the algorithm terminates without reaching the desired accuracy level (unsuccessful run) and $\pi^{\text{succ}}(\epsilon)$ is the probability of a successful run. Note that the maximum number of function evaluations for each algorithm is set by the resp. BBOB contestant and thus varies within the algorithm set. We will estimate the ERT from the r runs performed by every algorithm on each test function by plugging in the empirical equivalents of the unknown parameters. For a thorough motivation of the ERT see Hansen and Auger (2005).

In order to estimate the ERT, each contestant is required to submit 15 runs of his or her algorithm for each of the 24 test functions. It was required to submit results for 2, 3, 5, 10, 20 and optionally 40 dimensional parameter spaces. The results of each run are automatically stored in a file by the BBOB framework. From this file it is possible to infer the number of function evaluations used for almost any accuracy level ϵ. There is one small difference in the way the 15 runs are composed between the 2009 and 2010 competition. In the 2009 competition the 15 runs were divided among 5 different test function instances[1] for each of which 3 runs had to be performed. In 2010, instead of 5 instances, 15 instances with just one run per instance were required.

4.1 Experimental Setup

The 24 BBOB test functions (Hansen et al. (2009)) are subdivided into four classes, basically with respect to modality, separability and global structure. As already shown in Mersmann et al. (2010a) and Mersmann et al. (2011a) the algorithm performance is not consistent enough within the specified function groups to justify the selection of a reduced set of representative test problems. Therefore, we concentrate on the individual functions rather than selecting a representative function of each algorithm group.

In order to ensure distinguishable performance results of the algorithms the analysis focuses on a precision level of 10^{-3} to be reached. This coincides with the requirements of many practical applications in which a sufficient decrease of the fitness function is acceptable and it is not necessary to discover the global optimum. Furthermore, problem dimensionality is exemplary set to ten dimensions as a representative of the practically relevant dimensions 5-20.

The outline of the analysis is as follows: Based on the raw data, graciously provided by BBOB team on their website[2], and the benchmarking concepts of Mersmann et al. (2010) and Mersmann et al. (2011a), we determine the best algorithm for each BBOB function w.r.t. ERT which is presented in Table 2.

As can be seen in Table 2, there is no global "best" algorithm. In fact, no algorithm is best on more than three functions. In addition, the relative ERT (rel. ERT) values, i.e., the ERT of the specific algorithm divided by the ERT of the optimal algorithm for the respective function, are computed. Table 2 lists the respective median and maximum statistics, computed across all functions. However, the computation of the individual rel. ERT values faces some problems as almost none of the competing BBOB algorithms succeeded in solving all functions for the chosen (minimal) precision level. In many cases even half of the ERT data was missing. The number of solved functions of the considered algorithms together with the unsolved functions are presented in Table 3. The ERT values for a specific unsolved function are imputed by assigning ten times the maximal ERT value of all 14 algo-

[1] Slight reparameterizations of the test function obtained by rescaling, rotating or otherwise transforming the parameter vector before applying the function.

[2] http://coco.gforge.inria.fr/doku.php

Algorithm	Best on Function	med (rel. ERT)	max (rel. ERT)
AVGNEWUOA	8	29.67	58731.09
BFGS	1, 10	31.55	58731.09
BIPOP-CMA-ES	17	3.83	5873.11
DE-F-AUC	18	41.13	11867.07
FULLNEWUOA	6	39.36	58731.09
GLOBAL	12, 21, 22	22.60	58731.09
iAMALGAM	23	10.02	2287.43
IPOP-ACTCMA-ES	7, 11, 13	3.72	2655.55
IPOP-CMA-ES	15, 16, 19	3.40	5743.86
Line Search-fminbnd	2	172.23	58731.09
Line Search - STEP	3, 4	417.47	13494.28
MCS	5, 9	39.55	58731.09
MOS	20, 24	4.27	1186.71
NEWUOA	14	28.01	58731.09

Table 2: All BBOB algorithms which performed best on at least one of the 24 functions w.r.t. ERT. The functions are listed in the 2nd column. In addition, median and maximum rel. ERT values over all functions are given.

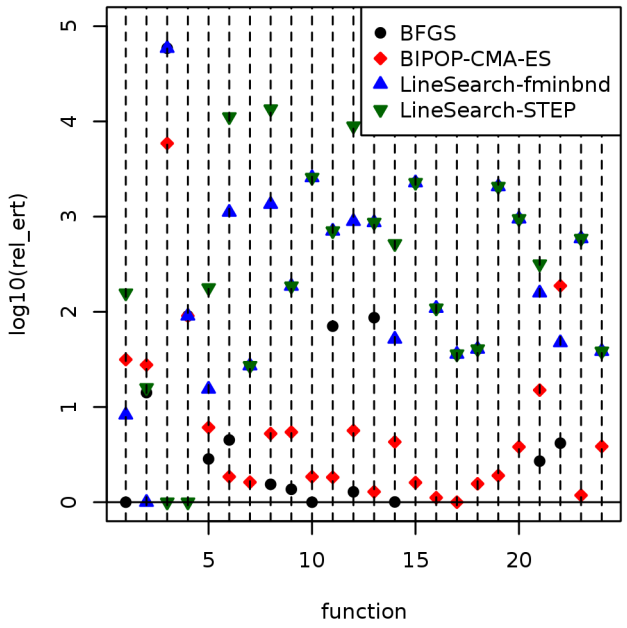

Figure 1: Rel. ERT value (logscale) of the algorithms in the portfolio for the 24 BBOB functions

rithms listed in Table 3 on this function. The worst imputed value equals 58731.09 for the third function.

We can observe that each algorithm performs very badly on at least one function or is not able to solve specific functions at all. This observation underscores the importance of the ASP. However, it turns out that a smaller subset of the 14 algorithms is sufficient to ensure a very good performance of the resulting algorithm portfolio.

We select a representative algorithm pool as follows. We consider an oracle that always selects the optimal algorithm out of the portfolio given a test function. This results in 24 rel. ERT values when the portfolio is applied to all BBOB functions and we use the maximum of these values to describe the worst-case behavior of the portfolio combined with the oracle. We then minimize this value over all possible portfolios. For a portfolio of size four this "oracle performance" drops down to a maximal rel. ERT of 4.169, and further increasing the algorithm set only lowers it insubstantially.

Hence, we select four algorithms for our portfolio: BFGS, BIPOP-CMA-ES, LineSearch-fminbnd and LineSearch-STEP, which also ensure a sufficiently high variety of algorithm concepts. The median rel. ERT value of the portfolio (combined with an oracle) is 1.454, meaning that our portfolio is in principle constructed well enough to allow for an excellent selection strategy: If we would reach the performance of the oracle, we would on average score only 1.5 times worse than the best available algorithm for each function. This value serves as a lower bound and comparison value for our learned selection strategies.

Figure 1 visualizes the rel. ERT values of the selected algorithms for each function. It becomes obvious that in most cases at least one of the algorithms is either optimal or almost optimal within the BBOB algorithm set. On the other hand considerable variance of the algorithms' performance becomes visible.

Two different values of baseline performance of the individual algorithms are given in Table 3 relating to the median rel. ERT value of an individual algorithm applied to all 24 functions. The first column only includes the rel. ERT values of the solved functions while the second one accounts for the whole data with imputed values. Both values are biased but the best option is to refer to the imputed data as the bad performance of an algorithm on an unsolved function has to be taken into account as well. Therefore, the BIPOP-CMA-ES shows the best baseline performance in the portfolio which however is roughly twice as bad as the optimal value which can be reached based on the whole portfolio.

In the present situation an adequate measure of classification performance is required. Instead of the misclassification rate often used in machine learning, a specific indicator related to the performance loss incurred by suboptimal predictions should be used. In case the best performing algorithm is not correctly identified the classifier is supposed to predict the algorithm with minimal possible loss w.r.t. the optimal ERT. With regard to the cost-sensitive learning approach explained in Section 3 we use the above mentioned four algorithms, therefore $k = 4$. As each x^i relates to a specific test function, we simply define the example-specific cost value c_j^i as the rel. ERT value of algorithm j on this test function. On this learning data we apply one-sided support vector regression with a radial basis function (RBF) kernel. For internal modeling, the rel. ERT values are transformed to logarithmic scale. As the support vector machine is sensitive w.r.t. its hyperparameter settings, both λ and σ (the latter belonging to the RBF kernel) are optimized by grid search on a logarithmic scale by using the values $\{2^{-16}, 2^{-15}, \ldots, 2^{16}\}$.

The low-level features listed in Table 1 are repeatedly computed in five runs for each of the five instances of the 24 functions using an initial budget of 5000 points for the LH design. This results in 600 observations in the data set used for learning the predictor. Note that the feature levels vary for each run and instance whereas the relative ERT value is constant over all runs and instances per function. Only one

Algorithm	# solved	Unsolved Functions	Baseline Performance # solved	Imputed
BFGS	13	3, 4, 7, 15-20, 23, 24	2.70	31.55
BIPOP-CMA-ES	23	4	3.81	3.83
LineSearch-fminbnd	9	3, 7, 9-11, 11, 13, 15-20, 23	51.8	172.23
LineSearch-STEP	6	6-20, 22-24	86.4	417.47

Table 3: Summary of the algorithm portfolio. The number of solved functions, the unsolved functions as well as the baseline performance for the solved functions and included imputed unsolved functions performance are given.

(BBOB 2010) or three (BBOB 2009) repeated algorithm runs on each function are available so that the estimation of the probability of a successful run is only possible in case the individual runs are aggregated for each function. In addition to the whole feature set a subset of the features which do not require additional function evaluations compared to the LH design is considered, and it will be investigated if similar classification accuracy can be achieved.

4.2 Results

Summarizing, the task is to predict a well-performing algorithm within the portfolio for each function based on the extracted low-level features of a specific function instance.

A crucial aspect of classification tasks is the necessity to (cross-) validate the trained model on unseen data. In the considered setting two different options are possible. Ideally, in each sequence of the cross-validation one function is completely discarded for training, and the remaining 25 observations are only used for testing. However, as the BBOB set is rather diverse in feature space (Mersmann et al. (2011)) the characteristics of the discarded functions might be not sufficiently covered by the feature space of the training set. The second option is to cross-validate on function instances, i.e., a five-fold cross-validation is carried out where in each sequence one of the five instances of each function is used for model validation, i.e., in total 120 observations. In both approaches the median of the rel. ERT values of the total 600 observations is used as the overall performance measure. In addition, the maximum rel. ERT value is presented.

Cross-Validation Omitting Instances.

In case of an instance-based cross-validation strategy the cost-sensitive classification approach succeeds in reaching a very high prediction accuracy. Many combinations of λ and σ lead to a median rel. ERT smaller than 1.6. The best classification result is generated based on the settings listed in Table 4. Here, the median of the rel. ERT values is very close to the optimal reference value of 1.454, even if the subset of "cheap" features is used. In case the whole feature set is considered even the worst prediction only takes a rel. ERT value of 86 whereas in the latter situation at least one function, in this case the third, is assigned the worst performing algorithm.

	med(rel. ERT)	max(rel. ERT)
optimum	1.454	4.169
all features	1.540835	86.76449
cheap features	1.606443	58731.08

Table 4: Classification results of cross-validation strategy based on omitting instances depending on the support vector regression parameters in case all features or only the "cheap" features are used.

	med(rel. ERT)	max(rel. ERT)
optimum	1.454	4.169
all features	1.901808	58731.08
cheap features	2.700301	58731.085

Table 5: Classification results of cross-validation strategy based on omitting functions depending on the support vector regression parameters in case all features or only the "cheap" features are used.

Cross-Validation Omitting Functions.

As already discussed before, it is of high interest if the highly qualitative results reported in the previous subsection can also be achieved in case whole functions, not only function instances, are omitted in the training set.

A high number of combinations of λ and σ resulted in a better performance than the best baseline algorithm BIPOP-CMA-ES. The best result out of the tested svm parameterizations for all features as well as the subset of "cheap" features are listed in Table 5.

However, in spite of the different characteristics of all BBOB functions related to the structure of the BBOB test set, the result is quite satisfying. In case all features are used, even 34 different parameterizations generate the same best result reported which is quite close to the optimum achievable value and only 1.9 times worse than the optimum ERT when all 60 BBOB algorithms are considered. The result gets worse in case the prediction is only based on the "cheap" features which, however, is still considerably better than the best baseline algorithm, i.e., the BIPOP-CMA-ES with a median rel. ERT of 3.83.

Figure 2 shows the euclidean distances between the functions in feature space based on all respectively on only the "cheap" features while the median of the scaled feature values over all runs and instances per function form the basis for the computation. Both the median rel. ERT of the predicted as well as the best algorithm of the portfolio for each BBOB function are related to the distance in feature space.

It becomes obvious that the degree of classification performance tends to correlate with the proximity in feature space if it is based on all features which is not that straightforward for the "cheap" features. However, this analysis might be improved by performing feature selection and changing the distance function with regard to the influence that the features have on the classification boundary.

Summarizing, the presented cost-sensitive classification approach proves to be highly successful for the addressed algorithm selection task despite of some limitations due to the structure of the BBOB test set and the results data set. These issues will be discussed in detail in the next section.

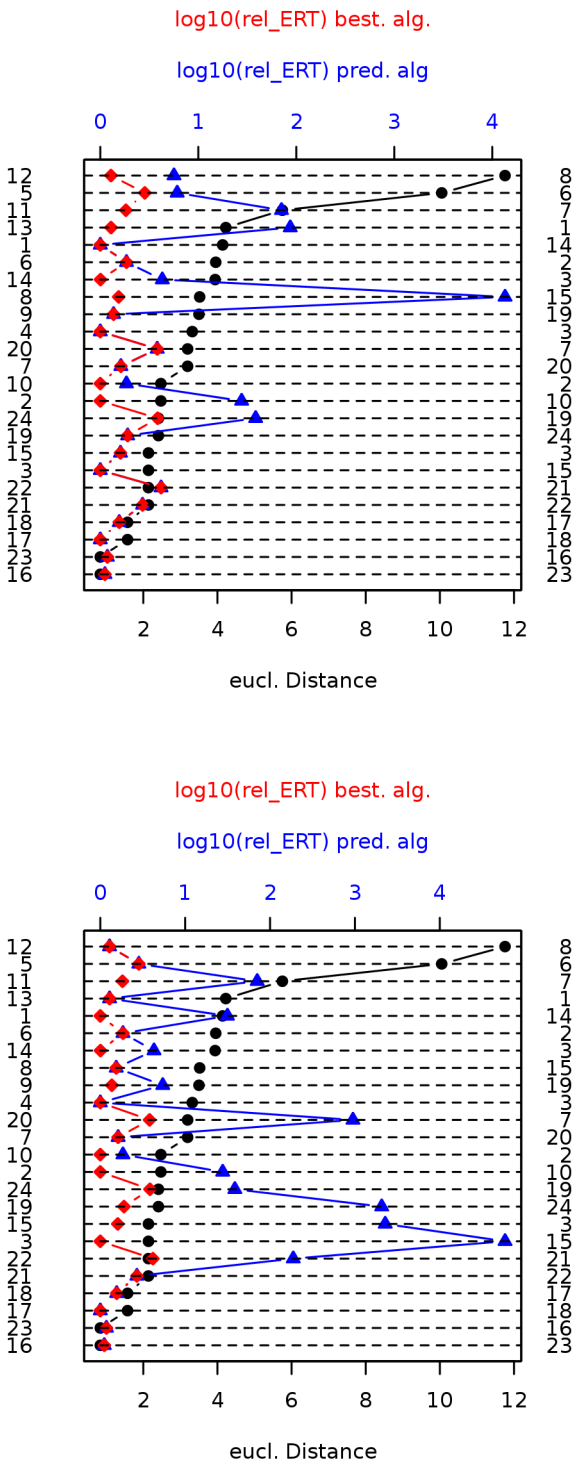

Figure 2: Per function in leave-one-out cross-validation: Median rel. ERT of predicted (blue) and best algorithm (red) of the portfolio and euclidean distance to nearest neighbor in feature space (black). Function removed in leave-one-out is given on the left, its nearest neighbor on the right side. Top: all features, bottom: "cheap features".

5. COMMENTS REGARDING BBOB DATA

One of the achievements of the two BBOB workshops that have been held so far is a wealth of data on a slew of different optimization algorithms and their relative performance. This data is the basis of our ERT calculations and without it, this study would not have been possible. The experimental setup of BBOB has been designed to be as lean as possible to reduce the burden on individual teams. This has lead to a function set that is very heterogeneous, i.e., each function is "very different" from every other function. While good in the context of the BBOB workshop, in the scenario under consideration it makes the machine learning task much more difficult. On the one hand, we have a very limited number of functions to cross-validate over, thus requiring the machine learning algorithm to extrapolate to vastly different functions. While we do want to see that the learning algorithm can generalize, this may in fact be too hard a task for it, given that some functions lie rather isolated in feature space. One way out would be to cross-validate over different instances of the same function instead. The difficulty here is that for each instance we only have three (2009) or one (2010) algorithm results and thus no chance to calculate a decent per-instance ERT estimate.

Another aspect which could be improved is the data storage format. While the BBOB workshop publishes all submitted results, these are stored in either cryptic text files or Python specific binary files. Neither solution is ideal. While both contain all the relevant data, what is missing or hidden in both cases is the relevant metadata. This also includes such simple things as a consistent naming scheme and directory structure for the result files, unique and meaningful algorithm names and a safe way to identify different runs of the same algorithm on the same function instance. Improving this would serve the whole community since the BBOB datasets are also a record of the evolution of the state of the art in our field.

Lastly, the number of algorithms will continue to grow, and this will make it increasingly more difficult to set up the algorithm portfolio. Which CMA-ES variant do I want? Do we want to use FULL_NEWUOA or AVG_NEWUOA? Is there a state of the art Nelder-Mead type algorithm in the dataset? What would help here are two things. First, a taxonomy of the different algorithms as well as a machine-readable classification for each and second, a repository of, again machine-readable, comparison results. The taxonomy would make it much easier for a non-expert to group the different algorithms and to pick a candidate from each group. Furthermore, it would allow us to easily examine hypotheses such as: "Similar algorithms perform similar". The long term goal here would be to arrive at a benchmarking framework in the spirit of the PISA (Bleuler et al., 2003) setup for multi-objective optimization which allows researchers to compare combinations of building blocks of algorithms instead of monolithic algorithm designs. This would greatly aid the understanding of why some algorithms perform well on some functions and others do not. Regarding the suggestion concerning a repository of machine-readable results, this could foster further collaboration between different groups and reduce redundant efforts of duplicated results.

6. DISCUSSION AND OUTLOOK

Notwithstanding the above remarks, we have introduced a novel solution to the ASP and shown that it works in different scenarios. In the first scenario, we showed that we can generalize to new instances of the same function for all functions in the BBOB test set. Depending on which ELA features are used, we can predict the optimal or close to optimal portfolio candidate (w.r.t. the median rel. ERT). It is noteworthy that such a portfolio would be a valid entry to the BBOB workshop and might outperform single algorithms (but see the remark about feature costs below). It could even trump a perfect oracle for the two algorithms recommended in Hansen and Ros (2010) (NEWUOA and BIPOP-CMA-ES).

On the other hand, we showed that the approach works remarkably well if we wish to generalize to new functions. This is astonishing given that the BBOB functions were designed to be quite different and unique. The poor worst case performance (max(rel. ERT)) of our approach does mean that the BBOB function set is, for our purposes, too small and would benefit from more functions. Given such a larger set our approach will likely generalize even better and provide further insights into how "landscape" features can be mapped to efficient algorithms.

Of course, for a proper evaluation when comparing such a selection procedure against single algorithms the costs for calculating the features must now be added to the function evaluations of the selected algorithm. For the "cheap" feature set in 10 dimensions this would result in 5000 evaluations. Note that here the median ERT of all the algorithms in the BBOB data is about 10^5, so the additional costs for the feature computation might be amortized in most cases.

However, both the selection of algorithms for the portfolio as well as the selection of the features used could be improved. Ideally, we would like to perform both selections at the same time. This could be done using a binary coded GA that selects both features and algorithms while minimizing the rel. ERT of the selection strategy. If one also considers the additional feature costs, this feature and portfolio optimization might be approached in a multi-criteria fashion similar to what we have already presented in Mersmann et al. (2011). But for all such endeavors a larger benchmark data set is required as well.

In the future we would like to explore ways to generate more test functions in a systematic fashion. This would allow us to apply experimental design methodologies to the problem to generate "unbiased" sets of test functions. For such an analysis it would be beneficial to have a learner that allows some more insight into the selection strategy.

Acknowledgements.
This work was partly supported by the Collaborative Research Center SFB 823 (B4), the Graduate School of Energy Efficient Production and Logistics and the Research Training Group "Statistical Modelling" of the German Research Foundation.

References

AUGER, A., FINCK, S., HANSEN, N., AND ROS, R. 2010. BBOB 2010: Comparison Tables of All Algorithms on All Noiseless Functions. Technical Report RT-388, INRIA. 09.

BARTZ-BEIELSTEIN, T. AND PREUSS, M. 2012. Experimental analysis of optimization algorithms: Tuning and beyond. In *Theory and Principled Methods for Designing Metaheuristics*, Y. Borenstein and A. Moraglio, Eds. Springer, New York.

BLEULER, S., LAUMANNS, M., THIELE, L., AND ZITZLER, E. 2003. PISA — a platform and programming language independent interface for search algorithms. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, Eds. Lecture Notes in Computer Science. Springer, Berlin, 494 – 508.

HANSEN, N. AND AUGER, A. 2005. Performance evaluation of an advanced local search evolutionary algorithm. In *In Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press, 1777–1784.

HANSEN, N., AUGER, A., FINCK, S., AND ROS, R. 2009. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Tech. Rep. RR-6828, INRIA.

HANSEN, N., AUGER, A., ROS, R., FINCK, S., AND POŠÍK, P. 2010. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *GECCO '10: Proceedings of the 12th annual conference comp on Genetic and evolutionary computation*. ACM, New York, NY, USA, 1689–1696.

HANSEN, N., FINCK, S., ROS, R., AND AUGER, A. 2009. Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Tech. Rep. RR-6829, INRIA.

HANSEN, N. AND ROS, R. 2010. Black-box optimization benchmarking of NEWUOA compared to BIPOP-CMA-ES: on the bbob noiseless testbed. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. GECCO '10. ACM, New York, NY, USA, 1519–1526.

HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. 2001. *The Elements of Statistical Learning*. Springer, New York.

JONES, T. AND FORREST, S. 1995. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 184–192.

KARATZOGLOU, A., SMOLA, A., HORNIK, K., AND ZEILEIS, A. 2004. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software 11*, 9, 1–20.

MERSMANN, O., BISCHL, B., TRAUTMANN, H., PREUSS, M., WEIHS, C., AND RUDOLPH, G. 2011. Exploratory landscape analysis. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. GECCO '11. ACM, New York, NY, USA, 829–836.

MERSMANN, O., PREUSS, M., AND TRAUTMANN, H. 2010a. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *PPSN XI: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, R. Schaefer et al., Eds. Lecture Notes in Computer Science 6238. Springer, 71–80.

MERSMANN, O., PREUSS, M., TRAUTMANN, H., BISCHL, B., AND WEIHS, C. 2011a. Analyzing the BBOB Results by Means of Benchmarking Concepts. *Evolutionary Computation Journal, accepted*.

MERSMANN, O., TRAUTMANN, H., NAUJOKS, B., AND WEIHS, C. 2010. Benchmarking evolutionary multiobjective optimization algorithms. In *Congress on Evolutionary Computation (CEC)*, H. Ishibuchi et al., Eds. IEEE Press, Piscataway NJ.

RICE, J. R. 1976. The algorithm selection problem. *Advances in Computers 15*, 65–118.

TU, H.-H. AND LIN, H.-T. 2010. One-sided support vector regression for multiclass cost-sensitive classification. In *ICML*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 1095–1102.

TURNEY, P. 2000. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*. 15–21.