Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models

Pascal Kerschke
Information Systems and Statistics
University of Münster
kerschke@uni-muenster.de

Mike Preuss
Information Systems and Statistics
University of Münster
mike.preuss@wi.uni-muenster.de

Simon Wessing
Computational Intelligence Group
TU Dortmund
simon.wessing@tu-dortmund.de

Heike Trautmann
Information Systems and Statistics
University of Münster
trautmann@wi.uni-muenster.de

ABSTRACT

When selecting the best suited algorithm for an unknown optimization problem, it is useful to possess some a priori knowledge of the problem at hand. In the context of single-objective, continuous optimization problems such knowledge can be retrieved by means of Exploratory Landscape Analysis (ELA), which automatically identifies properties of a landscape, e.g., the so-called funnel structures, based on an initial sample. In this paper, we extract the relevant features (for detecting funnels) out of a large set of landscape features when only given a small initial sample consisting of $50 \times D$ observations, where D is the number of decision space dimensions. This is already in the range of the start population sizes of many evolutionary algorithms. The new Multiple Peaks Model Generator (MPM2) is used for training the classifier, and the approach is then very successfully validated on the Black-Box Optimization Benchmark (BBOB) and a subset of the CEC 2013 niching competition problems.

CCS Concepts

•Mathematics of computing → Exploratory data analysis; Continuous optimization; •Computing methodologies → Supervised learning by classification;

Keywords

Fitness Landscapes, Working principles of evolutionary computing, Machine Learning, Empirical Study, Multiple solutions / Niching

1. INTRODUCTION

In a recent work [13], an *Exploratory Landscape Analysis* (ELA) based approach was suggested to decide if an unknown, supposedly multimodal problem is a funnel problem or not. In funnel problems (see fig. 2), most interesting optima are clustered in the vicinity of the global optimum. Problems with this property have also been termed *big valley* problems [10]. In any case, for the time being we presume that there is only one funnel (or big valley).

Let us follow the terminology of [13] and call the non-funnel problems random problems. This is not entirely true, as the distribution of optima in non-funnel problems is not necessarily random but could also be very structured, as in a grid. However, for selecting a suitable optimization algorithm, the exact positioning of the optima would make little difference. We presume that it is already sufficient if we know if a problem has the property of a funnel (or global) structure or not. Let us assume a global optimization scenario. Then, knowing that a problem has a funnel structure means that we need an algorithm that quickly focusses onto one specific area, whereas for a random problem we have to search virtually everywhere for new optima. A more detailed discussion of the related work is provided in [13], where a first attempt is described that performs a search space sample, computes some exploratory landscape features, and then classifies the problem as being funnel or non-funnel.

The approach presented there has one decisive shortcoming: it is too expensive in terms of the number of samples needed to classify a problem. A necessary sample size of $500 \times D$ is certainly a huge disadvantage, as it means to sacrifice 1,000 function evaluations even for a 2D problem before the optimization actually starts. Reducing the sample sizes to around $50 \times D$ would mean that, especially for lower dimensions, one would be in the range of the normal starting population sizes, such that no additional fitness evaluations have to be spent in order to detect if we have chosen the right algorithm.

In this work, we suggest to incorporate some modifications that enable to reduce the needed sample sizes dramatically. These changes comprise:

- an improved problem generator as suggested in [24],
- a better sampling procedure that seems to gather the same amount of information from fewer points, and
- an optimal feature selection procedure over a limited set of features that is chosen based on the results reported in [13].

GECCO '16, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4206-3/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908812.2908845

Our results show that we can improve the feature selection procedure such that it is possible to do an accurate problem classification utilizing only very small samples. Thus, we are interested in how small these samples may be without sacrificing too much accuracy, and what the individual effects of the factors enlisted above are concerning the compensation of decreasing the sample size. We also discuss the potential speedup that could be obtained by selecting the right algorithm with given accuracy if we would do algorithm selection based on our results in a very much simplified scenario.

The next sections report on the current situation and recent history in ELA techniques, and on the main constituents for the obtained enhancement, namely the improved LHS (sect. 3), and the new problem generator version MPM2 (sect. 4). Sections 5 and 6 deal with the employed validation problem set and the presentation of our experimental results. In sect. 7, we reason about the performance advantage that successful classification will have when used for algorithm selection.

2. ELA ON FUNNEL STRUCTURES

Exploratory Lanscape Analysis, originally proposed in [16], aims at understanding landscape characteristics of continuous black-box optimization problems. A core purpose is to gain sufficient knowledge of problem characteristics prior to optimization, based on an initial small sample of function evaluations, in order to decide on the best suited optimization algorithm, or at least algorithm class. This is particularly important with regard to expensive fitness function evaluations which are quite common in practical applications. Thus, a sequential or parallel application of several different techniques becomes impossible in these scenarios. Algorithm selection models are required which are constructed by means of linking benchmark data of algorithm candidates on a comprehensive set of test problems to problem features, see e.g. [3] or [19] for an overview about the research field. Feature sets have been proposed by various groups [1, 6, 15, 17, 18, 16, 3, 12, 13]. The R-package `flacco` [11] makes all relevant features available in a unified framework, together with efficient helper functions for aggregating and visualizing results.

Recently, specific features for the detection of funnel structures [13] based on Nearest-Better Clustering (NBC, [20]), were introduced. NBC is a recent heuristic for identifying single basins of a multimodal landscape. The knowledge of possible global structures offers the potential for facilitating the algorithm selection process by allowing to preselect algorithms that are capable of exploiting these structures. A sophisticated problem generator [26] was used to generate test instances with different topologies, i.e. varying in the number of peaks, dimensions and global topology. Classifiers including established and newly introduced features were trained in order to predict whether the considered problem has a funnel structure. Accurate prediction models based on decision trees could be constructed which were subsequently validated on the BBOB test set and an example from disc-packing. Despite of these previous promising results, there appeared to be further potential for reducing the required sample size, i.e. by means of improving the problem generator. We address this expectation in this paper.

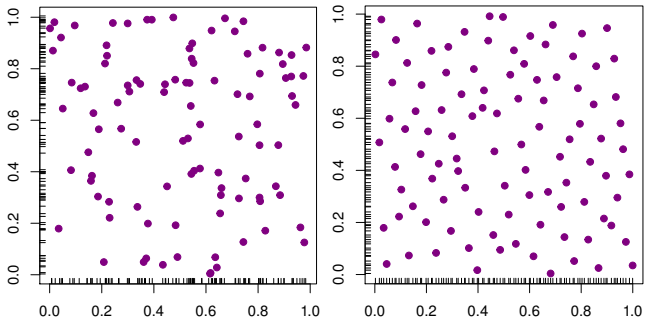

Figure 1: A random uniform (left) and an ILHS sample (right) with 100 points each. One can see that the overall distribution and the one-dimensional projections of ILHS are more uniform.

3. IMPROVED LHS

A classical approach of generating space-filling designs for computer experiments are latin hypercube designs (LHD). Such a design shall obtain a much more regular distribution of points onto a fixed D-dimensional hypercube (see fig. 1 for an example in 2 dimensions).

An LHD is defined as a set of points $\mathcal{D} = \{\vec{z}_1, \ldots, \vec{z}_N\}$, where each set $\{z_{1,j}, \ldots, z_{N,j}\}$, $j = 1, \ldots, D$, is a random permutation of the numbers $1, \ldots, N$. The set of points $\mathcal{D}$ can then be scaled to any cuboidal search spaces. The random LHD construction guarantees a perfectly uniform distribution of one-dimensional projections of the points, but the uniformity in D dimensions is usually not better than that of random uniform points. Thus, it is advisable to use a more sophisticated construction approach.

Beachkofski and Grandhi [2] propose such an algorithm to generate LHDs with improved D-dimensional distribution. The proposal is a greedy heuristic that begins with a single random point and sequentially adds the best point of a random sample until the LHD is complete. We will call it improved latin hypercube sampling (ILHS) here. In each iteration, a number of candidates are considered. The candidates are randomly chosen points not violating the LHD property if added to the design. The selection criterion in this case minimizes the deviation of a point's nearest-neighbor distance to the already chosen points from a supposedly ideal distance $d_{\text{opt}} = N/\sqrt[D]{N}$. Saka et al. [23] observed much better results for ILHS compared to conventional LHS. An implementation of ILHS is, e. g., available in the R-package `lhs` [5].

It was already argued in [24], that an improved sample design also leads to more or better accessible (less noisy) information contained in the point set. On base of this argument, we presume that we can expect the same amount of information in features computed on a smaller sample if the overall sample quality is improved.

4. MULTIPLE PEAKS MODEL 2

In [24], a refined version of the previously used generator [26] was proposed. The main reason for the revision was to improve the construction of the problem. For example, the generator could exceed the requested number of optima slightly, which has been fixed in the MPM2 version. Secondly, the position of the funnel bottom is now random uniformly

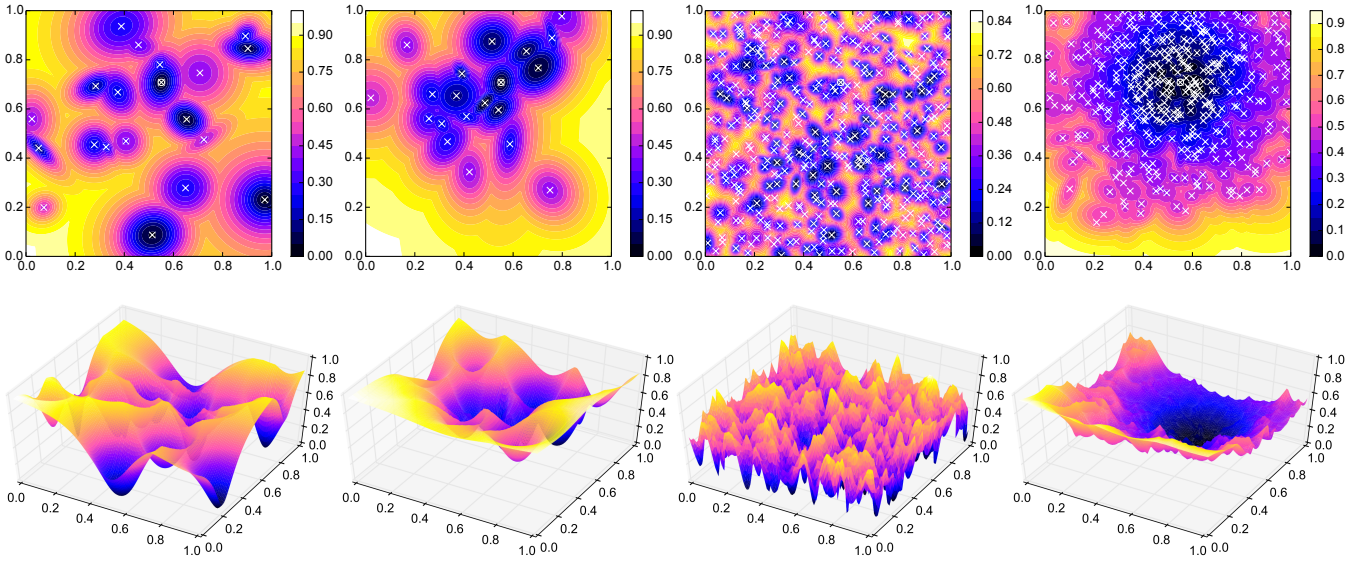

Figure 2: Examples of two-dimensional MPM2 functions (top: two-dimensional contour plots; bottom: three-dimensional perspective plots). The first two functions have 20 optima each, while the next two have 400. The funnel topologies in the second and fourth column can be clearly distinguished from the random topologies in the first and third. Local optima are marked with a cross, global ones are encircled.

distributed in the search space, while it was previously restricted to the vicinity of the centroid of the search space. On the other hand, the positions of local optima around the funnel bottom are now clustered and not random uniform as before. This was done to obtain a greater distinction between funnel and random structures. As this simplifies the classification problem, it shall be easier to learn good classifiers by employing the new generator version.

An MPM2 instance is defined by the following simple formulas:

$$f(\vec{x}) = 1 - \max\{g(\vec{x}, \vec{p}) \mid \vec{p} \in P\} \tag{1}$$

$$g(\vec{x}, \vec{p}) = \frac{h_{\vec{p}}}{1 + \frac{\mathrm{md}(\vec{x}, \vec{p})^{s_{\vec{p}}}}{r_{\vec{p}}}} \tag{2}$$

$$\mathrm{md}(\vec{x}, \vec{p}) = \sqrt{(\vec{x} - \vec{p})^\top \boldsymbol{\Sigma}_{\vec{p}}^{-1} (\vec{x} - \vec{p})} \tag{3}$$

The objective function is given in (1). It takes the minimum of $N_{\text{peaks}} = |P|$ unimodal functions (2) around peak positions $\vec{p} \in P$. This has the advantage that local optima with known positions are created. Rönkkönen et al. [22] criticize the exponential decay of the Gaussians used in [7]. Therefore, we are using the polynomial form in (2). Each of these functions is associated with parameters $h_{\vec{p}}$, $s_{\vec{p}}$, and $r_{\vec{p}}$ for height, shape, and radius, respectively. The idea of random shape and radius parameters is taken from [21]. By slightly deviating from locally quadratic behavior ($s_{\vec{p}} = 2$), we intend to increase the difficulty for local search algorithms. Radii $r_{\vec{p}}$ influence the size of attraction basins. Additionally, a randomly drawn covariance matrix $\boldsymbol{\Sigma}_{\vec{p}}$ belongs to each peak. This matrix is used to create the optima's basins as rotated hyperellipsoids, by calculating the Mahalanobis distance in (3). Overall, the calculation of peaks is now very similar to [7], only with a different, non-Gaussian shape function. All mentioned parameters are drawn randomly during initialization and then stored. By convention, we will always set $\max\{h_{\vec{p}} \mid \vec{p} \in P\} = 1$ and use a box-constrained search space $\mathcal{X} = [0, 1]^D$. The resulting landscapes are illustrated in Figure 2.

An implementation of this generator can be obtained from the Python package `optproblems` [25] and the R-package `smoof` [4].

5. TEST PROBLEMS: BBOB / CEC-2013

For validation purposes, we employ a set of test problems that is completely disjoint to the training set produced with the MPM2 generator. In [13], the BBOB problem set of 24 test problems [8] has been partitioned into funnel and random problems. Namely, only problems number 16, 21, 22, and 23 (Weierstrass, Gallagher 101 peaks, Gallagher 21 peaks, and Katsuura) have been classified as random, all others as funnel problems. As this results in a very unbalanced portfolio, we also take some problems from the CEC 2013 niching competition problem set [14] into account. This set may be partitioned into 3 subsets: a) some very simple 1D functions, b) a group of 2D and 3D well-known problems that have been adapted for the all-global setting (F4 to F8), and c) a group of composition functions that are defined by combining several base functions. Here, we concentrate on the group b) problems, namely:

F4 Himmelblau (2D)

F5 Six-Hump Camel Back (2D)

F6 Shubert (2D, 3D)

F7 Vincent (2D, 3D)

F8 Modified Rastrigin - All Global Optima (2D)

All these problems are of the random type, as they have been set up in a way that a number of search space positions with equal, globally optimal function values have to be found, and these locations are not clustered in one area of the search

space but more or less evenly distributed. The test problem set derived from the CEC 2013 niching competition setup (most of which are displayed in fig. 3) thus consists of 7 problems in 2-D and 3-D, whereas the BBOB test set (24 problems in 2, 3, 5, and 10 dimensions each) consists of 96 problems, 80 of which are of the funnel type.

6. EXPERIMENTS

Based on the results of [13], we have the following situation: We know that with $500 \times D$ samples, funnel problem classification is possible with good accuracy, and that some groups of features are much more often selected than others (this basically refers to the meta-model and the nbc features). Also we recognize a certain advantage of `rpart` and `randomForest` classifiers, which are either very simple and interpretable, or very accurate, respectively. In the experiments, we start to decrease the sample sizes and try to compensate the accuracy loss by different measures, as listed in the introduction. Our task is to achieve an accurate classification into funnel or non-funnel, based on a sample of only $50 \times D$.

6.1 Setup

In a first step, an improved latin hypercube sample (cf. section 3) has been used to generate samples of size $200 \times D$, where $D \in \{2, 3, 5, 10\}$ is the number of decision space variables. Then, we created funnel and random topology functions by means of the MPM2 generator (cf. section 4). Each of the functions contains $20, 40, \ldots, 200$ peaks and is created with five different seeds per sample, resulting in a total of 400 training instances. In a next step, the R-package `flacco` [11] has been used to compute the more than 300 available different landscape features per instance. As several ELA features are of stochastic nature, we repeated the feature computation ten times per instance, in order to better capture the variety of those features. All of the resulting features will be used simultaneously for further computations, because the alternative – aggregating them over the replications – would result in a loss of information.

The aforementioned problem instances have been used to train six classification models, which on basis of the feature information predict whether a given problem has an underlying funnel structure or not. As classification methods, we employed the tree-based learners `randomForest` (consisting of 500 trees) and a classification tree (`rpart`), as well as kernel-based versions of the support vector machine (`ksvm`) and the nearest neighbors method (`kknn` with $k \in \{3, 7, 15\}$). In case of the former, we perform a data dependent estimation of the inverse kernel width σ using the `R` function `sigest` from the `kernlab` [9] package. This estimation has been performed, because the choice of that parameter has a strong impact on the support vector machines (SVM) performance. Apart from that parameter, we executed all classifiers using their default values as specified by `mlr`, e.g. using the *Gaussian radial basis function* as kernel for the SVM or the *triangular* kernel for the nearest neighbor approach.

Training classification models on the given training data is not recommended, as it likely results in overfitting due to the high amount of features. Additionally, it is probable that many features capture similar characteristics of a problem and thus lead to a lot of redundancy. Therefore, we employed a feature selection with a greedy forward-backward-selection strategy. That is, one starts with an empty set of features and greedily adds or removes a feature to the feature set. These steps are repeated until no further improvement of the model performance, i.e. a decrease of the misclassification error, can be observed. Note that the misclassification rate has the advantage that it can be computed for any classification model in contrast to model-specific performance measures such as the random forest's out-of-bag-error. In order to receive reasonable performance values, our models were trained using a 10-fold cross-validation. In consequence, the feature selection resulted in a nested cross-validation as suggested by [12]. As a result, one receives a separate feature set for each of the ten folds. Out of those features, all features that were suggested in at least two folds were then used for training our final models.

For a better assessment of the models, each classifier has been applied to problems from other test beds, namely the BBOB test set, as well as the test problems from the CEC 2013 niching competition. As mentioned in the previous chapter, the BBOB test bed consists of 24 problems. By means of scaling, rotation and shifts, 15 different problem instances were created for each of those problems. Considering the four different dimensions, as well as ten replications for each of the instances, the entire test bed consists of 14,400 validation instances. As the majority of BBOB functions are funnel problems, the models were also assessed on 7 niching problems (with ten replications each), whose optima follow a non-funnel structure.

Due to promising results, we decided to split the initial designs in half, i.e. reducing them to $100 \times D$ observations, and running the experiments again on the smaller samples. In addition, the so-called expensive feature sets, i.e. features, which need additional function evaluations (e.g. the *local search* features) were removed from the set of features. In a next step, we followed the suggestions from [13] and reduced the features to a total of 15 features: the dimension of the decision space, the nearest-better clustering features (cf. [13]) as well as the meta-model features (cf. [16]). For each of the three approaches (all features, all non-expensive features and the small feature set), a feature selection has been conducted.

Once again, the results were promising and thus the size of the initial designs was reduced to $50 \times D$ observations. As the best results in the previous experiments were achieved by the tree-based learners, we removed the `ksvm` and `kknn` approaches from the following analyses and focussed on the `randomForest` and `rpart`.

In a final step, the eight most important features were manually selected out of the feature set consisting of the aforementioned 15 features (dimension, nearest-better clustering and meta-model features). Afterwards, these features were analyzed with an exhaustive search, i.e. all $2^8 = 256$ feature sets were created, and the two classification models were trained and assessed on each of those feature sets.

6.2 Results

Within the course of our experiments, we have shown that it is sufficient to use initial designs of size $100 \times D$ without losing much quality among the classification models. Unsurprisingly, we detected that models trained on ILHS generated data sets provided similar or better performance than equivalent models based on random uniformly sampled data sets. Therefore, we restrict our focus to the results of ILHS

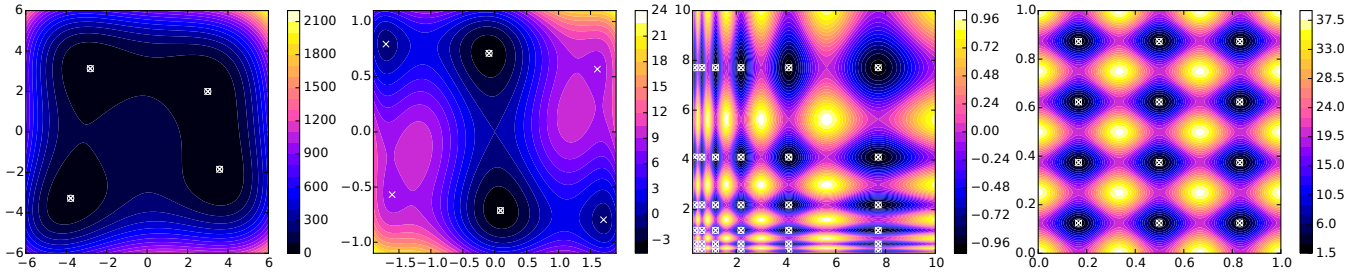

Figure 3: Example problems from the CEC 2013 niching competition. From left to right, F4, F5, F7, and F8 are shown. Local optima are marked with a cross, global ones are encircled.

generated samples in the following.

Whereas reducing the feature number would be a good idea by itself in order to make the resulting models more accessible and understandable, it also enables a more thorough feature selection process. Up to now, we employed a heuristic (forward-backward-selection) that may have missed some very accurate models. Doing a brute force search on all combinations of the 300 available features (this results in 2^{300} possibilities) is simply unfeasible. For that reason, we needed to reduce the size of the initial feature set to a maximum of 15 rather cheap features — i.e. features that only use the information provided by the initial designs and thus do not need any additional function evaluations — which were designed to capture information on the global structure, as designed in the setup (cf. section 6.1).

Training on the $100 \times D$ ILHS generated samples with a restricted feature set led to a random forest which contained only eight features and whose resampled misclassification error on the training data is 1.5% (cf. table 1). Actually, that performance is equal to the *virtual best* performance, which is the arithmetic mean over the best achieved performances within each of the 10 CV-folds. However, one should note that in case of the iterative feature selection, there might exist better performing feature set combinations — i.e. combinations that lead to a smaller virtual best error rate — and which just were not tested due to the greedy selection approach. Therefore, the corresponding values are marked (*) within the table.

While the feature selection strategy suggested feature sets, comprised of nine different features across the 10 CV-folds, only eight of them were used for training the final model. The ninth one occurred only in one of the ten feature sets and therefore has been removed prior to training the final random forest. When validating the forest on the training data and niching problems, it made perfect predictions. However, it had some problems when predicting the funnel problems of the BBOB, resulting in a misclassification error of 10%.

As the results in total were rather promising, we reduced the size of the initial designs to samples with $50 \times D$ observations – which is close to the size of an optimization algorithm's initial population – and repeated the feature selection. The quality of these models deteriorated slightly compared to the previous ones – but mainly during the evaluation of the feature selection itself and on the BBOB funnel problems.

Due to the fact that the feature selection strategy might be too greedy at times, we also evaluate the feature sets using an exhaustive combination of the eight most promising

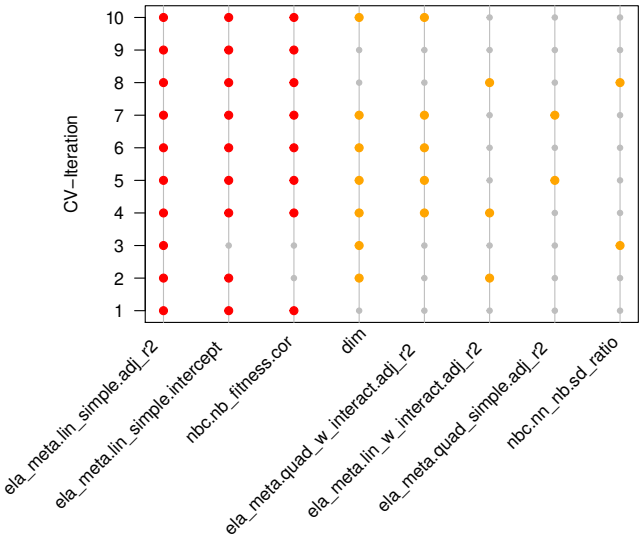

Figure 4: The feature importance plot shows, which feature sets performed best – and thus have been selected by the random forest – in each of the ten cross-validation folds. The features are ordered according to their frequency. Red points indicate features that were selected at least eight times and yellow points indicate features that were selected at least twice.

features. The resulting random forest (highlighted in grey within table 1) is competitive across all categories. Given the fact that it requires the smallest training data (8 features and initial samples of size $50 \times D$), it is our preferred solution.

For better comparison, we also show the performance results of the classification tree, trained on the small initial samples and using an exhaustive feature selection. As one can see, it also performs well on the training data. However, it has even bigger problems with the BBOB funnel problems, resulting in approx. 20% misclassification on these functions.

In order to better understand which features are important for separating funnel and non-funnel problems, we provide the feature importance plot (see fig. 4). This plot shows the selected feature sets within each of the ten cross-validation iterations, e.g. in the third fold, the features `dim`, `ela_meta.lin_simple.adj_r2`, and `nbc.nn_nb.sd_ratio` were selected. The features are ordered w.r.t. their decreasing frequency across the ten iterations. Furthermore, fe-

Table 1: Performance comparison of four different classification models: the first two columns show the quality (misclassification error) of random forests after a feature selection (FS) based on initial designs of sizes $100 \times D$ and $50 \times D$, whereas the last two columns show the quality of the random forest and rpart based on an exhaustive search (Ex.). The upper table focusses on the performance during the training of the model, whereas the lower one shows the performance of the models on the validation data sets. All models have been trained with data created by the MPM2 generator, using an improved latin hypercube sample.

Measure		Subset	random Forest			rpart
			100D (FS)	50D (FS)	50D (Ex.)	50D (Ex.)
Misclassification Error	Resampling	Total	0.015	0.015	0.016	0.035
		Funnel	0.015	0.015	0.018	0.030
		Random	0.015	0.015	0.015	0.040
	Feature Selection	Total	0.023	0.038	0.020	0.040
		Funnel	0.020	0.040	0.025	0.040
		Random	0.025	0.035	0.015	0.040
	Virtual Best	Total	0.015*	0.015*	0.013	0.028
# Selected Features		Total	9	14	8	8
		Model	8	10	8	7

Data Set	Sample Type (Data Set)	Subset	random Forest			rpart
			100D (FS)	50D (FS)	50D (Ex.)	50D (Ex.)
Generator	ILHS	Total	0.000	0.000	0.000	0.025
		Funnel	0.000	0.000	0.000	0.005
		Random	0.000	0.000	0.000	0.045
	Random	Total	0.037	0.036	0.033	0.035
		Funnel	0.000	0.015	0.035	0.015
		Random	0.075	0.058	0.030	0.055
BBOB	ILHS	Total	0.086	0.129	0.083	0.181
		Funnel	0.102	0.153	0.100	0.216
		Random	0.004	0.008	0.000	0.008
	Random	Total	0.036	0.132	0.063	0.181
		Funnel	0.022	0.154	0.073	0.213
		Random	0.108	0.021	0.017	0.017
CEC Niching	ILHS	Total	0.000	0.000	0.000	0.000
	Random	Total	0.000	0.000	0.057	0.000

atures that have been selected in at least eight of the folds are highlighted red, whereas features that were selected at least twice are highlighted yellow. As the plot indicates, the most important features for deciding, whether a problem has an underlying funnel structure, are (1) the adjusted R^2, i.e. the model fit of a simple linear model, (2) the level of the problem (indicated by the intercept of a simple linear model), (3) the so-called *indegree* (i.e. the correlation between the number of observations pointing towards a *nearest better* observation and its fitness value), and (4) the dimension of the decision space. Still, one should note that none of the features has been selected as a stand-alone and thus, the importance of the selected features is more likely the result of interaction effects between the different features.

However, each feature selection strategy has one major drawback: when multiple feature sets lead to the same best performance (i.e. lowest misclassification error) within a fold, one of those sets is drawn randomly. Therefore, we decided to have a closer look at the "pseudo-probabilities" for each feature, being selected in one of the folds. If for instance in the first fold, 20 feature sets resulted in the best misclassification error and the feature `dim` belonged to 15 of them, then it gets receives a value of $15/20 = 0.75$ for that iteration. These values are computed for each feature and each iteration and afterwards averaged across the ten iterations. The resulting ratios – or pseudo-probabilities – are shown within table 2.

Table 2: "Pseudo-probability" showing how often a feature belongs to the best feature set per fold.

Feature	random Forest	rpart
ela_meta.lin_simple.adj_r2	0.922	0.500
nbc.nb_fitness.cor	0.889	0.550
ela_meta.lin_simple.intercept	0.878	0.550
dim	0.665	0.513
ela_meta.quad_w_interact.adj_r2	0.643	0.480
nbc.nn_nb.sd_ratio	0.556	0.490
ela_meta.lin_w_interact.adj_r2	0.534	0.337
ela_meta.quad_simple.adj_r2	0.257	0.583

For the random forest, the table mainly agrees with the results shown in the feature importance plot. Interestingly, the feature `nbc.nn_nb.sd_ratio` has only been selected in two of the ten folds, but was part of 55.6% of the best performing feature sets. Also notable is the fact that the feature which has been in the fewest best performing feature sets of the random forest (`ela_meta.quad_simple.adj_r2`), was most often part of the best performing feature sets of the decision trees.

7. ALGORITHM SELECTION POTENTIAL

It is clear that our validation problems are only placeholders for real-world problems we eventually want to tackle. But the obtained results lay a solid foundation for expecting an accuracy level of about 90% (or 10% misclassification error) by means of a `randomForest` or 80% with a simple decision tree. What does this mean for the speedup we may obtain by using the obtained prediction for algorithm selection?

Let us embark on a very simple thought experiment and assume that we have a classifier with an accuracy of p_t and that it is used to decide which of two available optimization algorithms is applied to a specific problem of unknown characteristics in order to find the global optimum (or any point of very good quality). Let us further assume that the situation is symmetric, meaning that algorithm A is faster on *its* problem type by a factor of $s > 1$ while delivering performance 1 on the other problem type, and that the situation is exactly reversed for algorithm B. Given that problems of either type are equally likely, what performance gain can we expect for varying p_t and s in relation to a randomly chosen algorithm (due to the absence of a reliable classifier)?

We can compute the speedup for the random selection case s_r by simply averaging the better case s and the normal case 1, as both are equally likely:

$$s_r = \frac{s}{2} + \frac{1}{2} = \frac{s+1}{2} \tag{4}$$

For the classifier case with probability p_t to choose the right algorithm we have the speedup s_c:

$$s_c = p_t \cdot s + (1 - p_t) \cdot 1 = p_t(s-1) + 1 \tag{5}$$

Taking both together, we can compute the expected speedup gain s_g that results from actually employing an accurate classifier as:

$$s_g = \frac{s_c}{s_r} = \frac{2p_t(s-1)+2}{s+1} \tag{6}$$

Figure 5 visualizes the outcome of our thought experiment for speedup values s from 1 to 2, and accuracy values p_t from 0.5 to 1.0. We can see that for high accuracy classification, approximately half of the single algorithm speedup remains. Of course, this advantage gets larger if there are more algorithms to choose from. However, the classification problem difficulty increases accordingly.

It is clear that this thought experiment relies on a number of assumptions, and that we have opted for the simpler possibility if there was a choice (i.e., symmetric speed-up properties of A and B). We expect the real outcome to be much more complex, but only want to provide an idea of the potential speedup given that we could recognize what kind of problem we tackle without spending much effort and had identified matching algorithms beforehand.

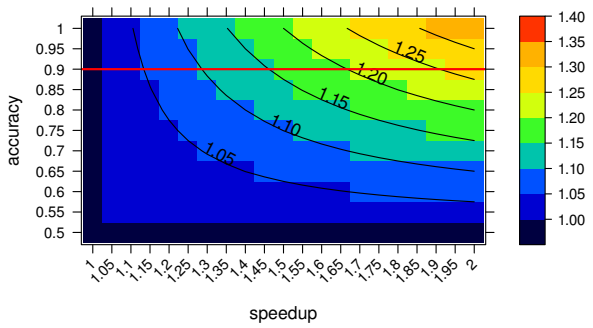

Figure 5: Speedup gain over random algorithm selection that can be achieved by selecting the correct of two algorithms in our very simplified scenario. The red line marks the approximate quality of the randomForest classifiers we have learned.

8. CONCLUSIONS

Summarizing, a small but extremely informative feature set reflecting characteristics of continuous black-box optimization problems could be identified which allows for a very accurate prediction whether the considered problem at hand has a funnel topology, i.e. a specific global structure. A sophisticated problem generator for multiple peaks models was used for this purpose and the resulting models were successfully validated on the BBOB test function set as well as test problems of the CEC niching competition. The core contribution, however, is that only a very small initial sample of function evaluations of 50 $\times$ decision space dimension is required so that we stay in the range of common population sizes of evolutionary optimizers. We additionally emphasize the benefit of using an improved latin hypercube sample instead of the common approach of using random sampling of the decision space.

Detecting a funnel topology prior to optimization on unknown problems is of crucial importance for selecting an appropriate efficient optimization algorithm or at least the respective algorithm class. Thus, we see huge potential of integrating a funnel feature in a sequential algorithm selection process. Moreover, the knowledge that no funnel structure is present enables the decision among possible multimodal optimization approaches. Thus, benefits are expected in terms of both improving global as well as multimodal optimization.

Future work therefore consists of linking funnel features to algorithm benchmark results, both on common test problems as well as real-world optimization tasks, and constructing algorithm selection models specifically adjusted to the kind of global structure. Moreover, we will investigate if and under which conditions a further reduction of the initial sample size is possible to ideally dispose of any computational overhead caused by feature computations.

9. REFERENCES

[1] T. Abell, Y. Malitsky, and K. Tierney. Features for Exploiting Black-Box Optimization Problem

Structure. In *Learning and Intelligent Optimization*, pages 30–36. Springer, 2013.

[2] B. Beachkofski and R. Grandhi. Improved distributed hypercube sampling. In *Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. AIAA paper 2002-1274, American Institute of Aeronautics and Astronautics, 2002.

[3] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuss. Algorithm Selection Based on Exploratory Landscape Analysis and Cost-Sensitive Learning. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 313–320, New York, NY, USA, 2012. ACM.

[4] J. Bossek. *smoof: Single and Multi-Objective Optimization Test Functions*, 2015. R package version 1.0.9000.

[5] R. Carnell. *lhs: Latin Hypercube Samples*, 2016. R package version 0.13.

[6] M. Gallagher. *Multi-Layer Perceptron Error Surfaces: Visualization, Structure and Modelling*. PhD thesis, University of Queensland, 2000.

[7] M. Gallagher and B. Yuan. A general-purpose tunable landscape generator. *IEEE Transactions on Evolutionary Computation*, 10(5):590–603, 2006.

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

[9] A. Karatzoglou, A. Smola, K. Hornik, and A. Zeileis. kernlab – An S4 Package for Kernel Methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004.

[10] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[11] P. Kerschke and J. Dagefoerde. *flacco: Feature-Based Landscape Analysis of Continuous and Constraint Optimization Problems*, 2015. R package version 1.2.

[12] P. Kerschke, M. Preuss, C. Hernández, O. Schütze, J.-Q. Sun, C. Grimme, G. Rudolph, B. Bischl, and H. Trautmann. Cell Mapping Techniques for Exploratory Landscape Analysis. In *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, pages 115–131. Springer, 2014.

[13] P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann. Detecting Funnel Structures by Means of Exploratory Landscape Analysis. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation*, pages 265–272. ACM, 2015.

[14] X. Li, A. Engelbrecht, and M. G. Epitropakis. Benchmark functions for CEC'2013 special session and competition on niching methods for multimodal function optimization. Technical report, RMIT University, Evolutionary Computation and Machine Learning Group, Australia, 2013.

[15] M. Lunacek and D. Whitley. The Dispersion Metric and the CMA Evolution Strategy. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 477–484. ACM, 2006.

[16] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. Exploratory Landscape Analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 829–836, New York, NY, USA, 2011. ACM.

[17] R. Morgan and M. Gallagher. Analysing and Characterising Optimization Problems Using Length Scale. *Soft Computing*, pages 1 – 18, 2015.

[18] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. Exploratory Landscape Analysis of Continuous Space Optimization Problems using Information Content. *Evolutionary Computation, IEEE Transactions on*, 19(1):74–87, 2015.

[19] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224 – 245, 2015.

[20] M. Preuss. Improved topological niching for real-valued global optimization. In *Applications of Evolutionary Computation*, volume 7248 of *Lecture Notes in Computer Science*, pages 386–395. Springer, 2012.

[21] M. Preuss and C. Lasarczyk. On the importance of information speed in structured populations. In X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervos, J. A. Bullinaria, J. E. Rowe, P. Tino, A. Kaban, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 91–100. Springer, 2004.

[22] J. Rönkkönen, X. Li, V. Kyrki, and J. Lampinen. A generator for multimodal test functions with multiple global optima. In X. Li, M. Kirley, M. Zhang, D. Green, V. Ciesielski, H. Abbass, Z. Michalewicz, T. Hendtlass, K. Deb, K. Tan, J. Branke, and Y. Shi, editors, *Simulated Evolution and Learning*, volume 5361 of *Lecture Notes in Computer Science*, pages 239–248. Springer, 2008.

[23] Y. Saka, M. Gunzburger, and J. Burkardt. Latinized, improved LHS, and CVT point sets in hypercubes. *International Journal of Numerical Analysis and Modeling*, 4(3-4):729–743, 2007.

[24] S. Wessing. *Two-stage methods for multimodal optimization*. PhD thesis, Technische Universität Dortmund, 2015.

[25] S. Wessing. *optproblems: Infrastructure to define optimization problems and some test problems for black-box optimization*, 2016. Python package version 0.6. https://pypi.python.org/pypi/optproblems.

[26] S. Wessing, M. Preuss, and G. Rudolph. Niching by multiobjectivization with neighbor information: Trade-offs and benefits. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 103–110, 2013.