Detecting Funnel Structures by Means of Exploratory Landscape Analysis

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

Mike Preuss
Computational Intelligence Group
TU Dortmund
mike.preuss@tu-dortmund.de

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

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

ABSTRACT

In single-objective optimization different optimization strategies exist depending on the structure and characteristics of the underlying problem. In particular, the presence of so-called funnels in multimodal problems offers the possibility of applying techniques exploiting the global structure of the function. The recently proposed Exploratory Landscape Analysis approach automatically identifies problem characteristics based on a moderately small initial sample of the objective function and proved to be effective for algorithm selection problems in continuous black-box optimization. In this paper, specific features for detecting funnel structures are introduced and combined with the existing ones in order to classify optimization problems regarding the funnel property. The effectiveness of the approach is shown by experiments on specifically generated test instances and validation experiments on standard benchmark problems.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Fitness landscapes, Working principles of evolutionary computing, Machine learning, Exploratory Landscape Analysis, Funnel Structure, Optimization, Feature Selection

1. INTRODUCTION

Let us assume we need to find a very good solution for an unknown real-valued black-box optimization problem that is most likely multimodal. Then, we can imagine two extreme cases: a) all the peaks are well distributed over the search space so that knowing one of them does not at all help to find another one, or b) the peaks are partly superimposed and pile up to a huge mountain, with the global optimum to be found on top of it. Turning this problem upside down from a maximization to a minimization problem results in a so-called funnel, and we can presume that knowing some optima within a funnel *does* help to find even better ones because this resembles a *global structure* that can be exploited. The Rastrigin function is a well-known benchmark problem that possesses this property. Examples for the possible topologies are shown in Fig. 1.

In this work, we do not investigate how large the benefit of knowing the global structure of a problem actually is. This surely depends on many factors, and especially on the available optimization algorithms. However, when we have to select a suitable optimization algorithm, this knowledge will undoubtedly have an advantage. In order to obtain knowledge about the global structure of an unknown problem, we suggest to apply *Exploratory Landscape Analysis* (ELA).

The basic idea of ELA is to compute many simple, so-called low-level features based on a relatively small sample of evaluated search points, and then to use feature selection and classification methods in order to estimate the high-level properties of an optimization problem. The approach has been introduced in [20] and extended in [7], and it turned out that multimodality and global structure are among the most important high-level properties that help to differentiate between problem classes. A more detailed ELA description is following in §2.

In this work, we review the existing literature on optimization of funnel problems (§1.1), and define some new features that shall be suitable for funnel detection in §2. Employing also existing ELA features, we establish funnel classifiers on a set of generated problems in §3. The new classifiers are then tested on the BBOB problem suite and also on a set of disc packing problems in order to check if they are transferrable in §4. Given they still work, we have evidence that the classifiers can also be applied successfully to other related optimization problems.

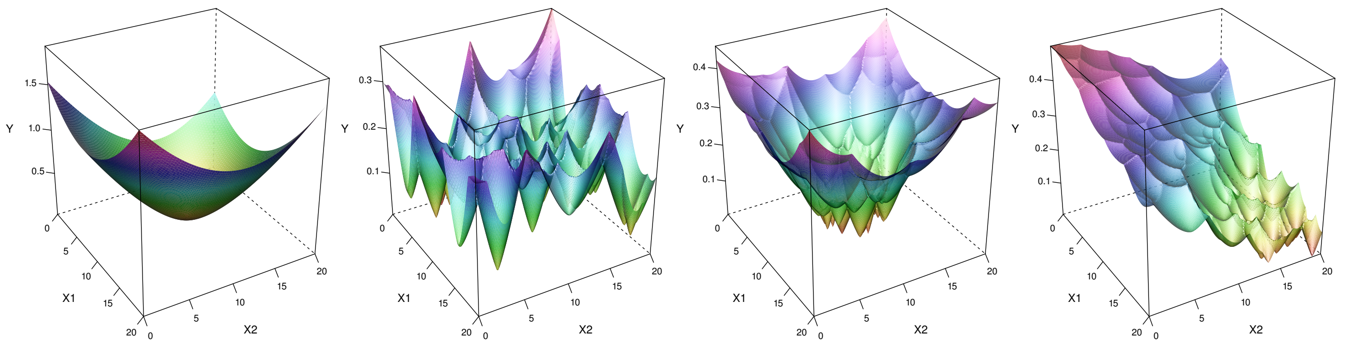

Figure 1: Generated test instances with different topologies. The first panel shows a unimodal problem. The other three have 40 local optima each. The second has a completely random structure. The third resembles a simple funnel, in which the best optima are clustered around the center of the search space. For the last one, there exists a linear trend from $(0, 0)^T$ to $(20, 20)^T$.

To our knowledge, this is the first attempt that explicitly tries to classify an unknown problem according to its global structure by means of a limited sample of search points.

1.1 Funnel Structure Optimization

It is not exactly defined what constitutes a funnel structure. Lunacek and Whitley [19] survey several possible definitions, which basically all focus on two problem properties, namely the distribution of optima and the correlation of their objective values. The importance of the former aspect for problem difficulty was already recognized by Törn and Zilinskas [27, p. 11], although without using the term "funnel". Addis and Locatelli [3] develop a more detailed characterization of problem landscapes, by defining three levels of local optima, based on a graph structure. However, their theory is not applicable to real-world problems, because the necessary information is not available.

In combinatorial optimization, a common approach to solve funnel problems is iterated local search (ILS) [18]. ILS is a very simple concept, whose core idea is to restart local search from perturbations of previously found local optima. Examples for funnel problems in combinatorial optimization are the traveling salesperson problem, scheduling problems, or MAX-SAT [18].

Addis et al. [4] propose *monotonic basin hopping* (MBH) as optimization algorithm for real-valued funnel problems. This algorithm is basically the real-valued equivalent to ILS. In [2], this approach is extended by embedding the MBH method into a random restart mechanism. As a local search component for their MBH, Addis et al. [4] favor sequential quadratic programming.

Also the CMA-ES is known to exhibit a good performance on funnel problems if the population size μ is chosen large enough [15, 19]. On problems without funnel structure, the large μ is not as helpful, instead a high number of random restarts is more important there. The "increasing population size" heuristic (IPOP) is an attempt to get rid of μ as an external parameter, by starting with a small μ and doubling it at every restart of CMA-ES [5]. Consequently, this approach is especially successful on funnel problems and does not harm too much on unimodal problems. Examples of problems in real-valued optimization – which supposedly are of the funnel type – are disc packing (see §4) or space trajectory planning [2].

2. ELA BASICS AND NEW FEATURES

Exploratory Landscape Analysis aims at characterizing the problem landscape and deriving rules for determining how problem properties influence algorithm performance. The final goal is an accurate prediction of the best suited algorithm for an arbitrary optimization problem based on the computed features. During the last years, important steps into this direction for single-objective optimization problems have been made. Experimental low-level features were introduced based on systematic sampling of the decision space in [20]. Six low-level feature classes were proposed (each containing several subfeatures), i.e. measures related to the distribution of the objective function values (y-Distribution), estimating meta-models such as linear or quadratic regression models on the sampled data (Meta-Model) and the level of convexity (Convexity). Furthermore, local searches are conducted starting at the initial design points (Local Search), the relative position of each objective value compared to the median of all values is investigated (Levelset), and numerical approximations of the gradient or the Hessian represent the class of curvature features (Curvature). It was shown in [20] that these features relate well to expert-defined high-level features such as the degree of multimodality, global structure, separability, variable scaling, search space homogeneity, basin-sizes, global to local contrast, and plateaus. Additional, conceptually similar features were introduced in [22, 1, 21]. In [7], a representative portfolio of four optimization algorithms was constructed from the complete list of BBOB 2009/2010 candidates. Based on the low-level features a sufficiently accurate prediction of the best suited algorithm within the portfolio for each function was achieved.

Recently, the feature set was extended based on the cell mapping concept in [17] by which a finite subdivision of the domain in terms of hypercubes is constructed. Function properties are then reflected by features comparing samples of different cells in order to assess local and global function characteristics. As these features can operate on the same initial design as the ELA features the cell mapping features come at no additional cost. It could be shown that the new properties especially improve the ELA feature set as in [20] in terms of an accurate mapping to a subset of the high-level features, i.e. the degree of multimodality and global structure. However, a meaningful application is so far limited to low-dimensional search spaces.

To detect if a problem exhibits a funnel structure, Lunacek and Whitley proposed to calculate a feature called "dispersion" [19]. As for all the features used in this paper, it is necessary to first draw a sample of the search space, for instance randomly uniform. The idea is to compare the sample's average pairwise distance with the same measure for a subset of the best solutions of this sample. The size of this subset may be determined by a parameter.

2.1 Features for Global Structure Recognition

Nearest-Better Clustering (NBC) is a recent heuristic for recognizing single peaks of a multimodal landscape. It is employed in its current form in [23] within the NEA2 algorithm, and we take over some of its general ideas in order to define features that should be helpful for recognizing funnel problems.

Using the same notation as in [24], we denote the distance to the nearest neighbor of a search point $\vec{x}$ from a population $\mathcal{P}$ with

$$d_{\mathrm{nn}}(\vec{x}, \mathcal{P}) = \min(\{\mathrm{dist}(\vec{x}, \vec{y}) \mid \vec{y} \in \mathcal{P} \setminus \{\vec{x}\}\}) \qquad (1)$$

and the distance to the nearest better neighbor ($f(\vec{x})$ stands for the objective function value that corresponds to $\vec{x}$) as

$$d_{\mathrm{nb}}(\vec{x}, \mathcal{P}) = \min(\{\mathrm{dist}(\vec{x}, \vec{y}) \mid f(\vec{y}) < f(\vec{x}) \wedge \vec{y} \in \mathcal{P}\}). \qquad (2)$$

We base our new features on the set of all nearest neighbor distances within the population

$$\mathcal{D}_{\mathrm{nn}} = \{d_{\mathrm{nn}}(\vec{x}, \mathcal{P}) \mid \vec{x} \in \mathcal{P}\} \qquad (3)$$

and, accordingly, the set of all nearest-better distances

$$\mathcal{D}_{\mathrm{nb}} = \{d_{\mathrm{nb}}(\vec{x}, \mathcal{P}) \mid \vec{x} \in \mathcal{P}\} \qquad (4)$$

with $d_{\mathrm{nb}}(\vec{x}^*, \mathcal{P}) := d_{\mathrm{nn}}(\vec{x}^*, \mathcal{P})$ for $\vec{x}^* = \arg\min(\{f(\vec{x}) \mid \vec{x} \in \mathcal{P}\})$, because for the best element in $\mathcal{P}$, no better neighbor exists.

Based on these two sets, we define five features. These are not made up arbitrarily; for each feature, we have an intuition of what it should measure, but at the same time we can assume that the formulation we choose is not unique. Features with approximately the same meaning can be constituted differently.

$$\text{nbf1} = \frac{\mathrm{sd}(\mathcal{D}_{\mathrm{nn}})}{\mathrm{sd}(\mathcal{D}_{\mathrm{nb}})} \qquad (5)$$

divides the two standard deviations of the sets. In case the problem is very multimodal, $\mathcal{D}_{\mathrm{nb}}$ should have a tendency to contain more extreme elements and $\mathrm{sd}(\mathcal{D}_{\mathrm{nb}})$ should be much larger than $\mathrm{sd}(\mathcal{D}_{\mathrm{nn}})$. On a unimodal problem or if a strong global structure exists, the nearest-better distances shall not be very different from the nearest-neighbor distances and the feature should be almost 1.

$$\text{nbf2} = \frac{\mathrm{mean}(\mathcal{D}_{\mathrm{nn}})}{\mathrm{mean}(\mathcal{D}_{\mathrm{nb}})} \qquad (6)$$

inherits the overall idea of nbf1, but uses the mean values instead of the standard deviations.

$$\text{nbf3} = \mathrm{cor}(\mathcal{D}_{\mathrm{nn}}, \mathcal{D}_{\mathrm{nb}}) \qquad (7)$$

measures the Pearson correlation between the two sets. The expectation is that in a funnel setting, the correlation should be high, whereas in a random peaks setting, it should be much lower.

The fourth feature depends on the quotient of an individual's nearest-neighbor and nearest-better distances:

$$\mathcal{Q}_{\mathrm{nn/nb}} = \left\{ \frac{d_{\mathrm{nn}}(\vec{x}, \mathcal{P})}{d_{\mathrm{nb}}(\vec{x}, \mathcal{P})} \,\middle|\, \vec{x} \in \mathcal{P} \right\}. \qquad (8)$$

$$\text{nbf4} = \frac{\mathrm{sd}(\mathcal{Q}_{\mathrm{nn/nb}})}{\mathrm{mean}(\mathcal{Q}_{\mathrm{nn/nb}})} \qquad (9)$$

thereby represents the coefficient of variation of the previously defined set $\mathcal{Q}_{\mathrm{nn/nb}}$, which should be larger for random peak landscapes (because variation of the quotients should be higher).

$$\text{nbf5} = -\mathrm{cor}(\{(\deg^-(\vec{x}), f(\vec{x})) \mid \vec{x} \in \mathcal{P}\}) \qquad (10)$$

relates the indegree $\deg^-(\vec{x})$ of search points in the nearest-better graph (the number of search points for which a certain search point is the nearest better point) to their objective values. The reasoning behind is that in a funnel landscape, the best optima should have a large number of incoming connections whereas in a random peaks landscape, the better optima should have only few.

3. FEATURE SELECTION ON GENERATOR PROBLEMS

3.1 Problem Generator and Methodology

To train our classifier, we need some functions which are guaranteed to exhibit the mentioned structural features. Consequently, we are using a test problem generator to create the attributes artificially. This approach is recommended by Eiben and Jelasity [11]. Important aspects for us are the adjustability of the problem dimension, the number of optima, and of course the global topology. The employed generator combines unimodal functions to obtain a multimodal problem. The exact formulas are described in [28]. Similar approaches also appear in other problem generators [12]. The resulting surfaces are differentiable almost everywhere and the exact locations of the optima are known. Different landscapes can be produced by controlling the correlation between the depth of the unimodal functions and their position. Fig. 1 shows some results of this generator for $n = 2$ decision variables. Note that the generated instances look different every time, because the generating function is stochastic.

For our experiments, we used the generator to create a training data set, which contains instances of four values of $n \in \{2, 3, 5, 10\}$, ten different amounts of peaks (20, 40, ..., 200) and three versions of underlying topologies (*linear*, *funnel* and *random*). In addition, each of those instances was generated with five different starting seeds in order to respect the variability of the problem generator. In consideration of the stochasticity of some of the landscape features, we also replicated each instance ten times, resulting in a total of 6,000 instances. Such a duplication of instances is essential when dealing with stochastic features, as this allows for a comparison of the variability of a feature (and thereby reduces the effect of an outlier) on the very same problem instance.

On each of those instances, we computed a total of 29 features: 24 ELA features (12 Levelset, 9 Meta-Model, 3 y-Distribution, [20]), as well as our five NBC features (§2.1). All of those features are based on an initial design, consisting of $500 \times n$ observations, and therefore no additional function

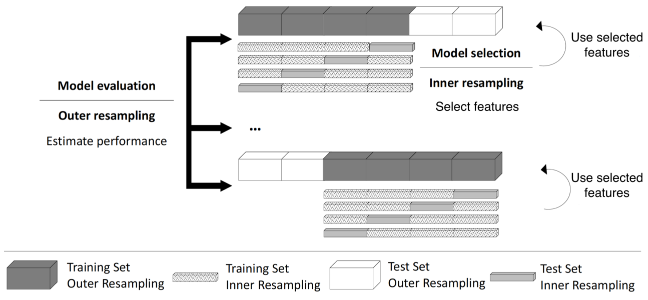

Figure 2: Feature selection based on nested resampling, cf. [17].

evaluations are required. In addition, a second feature set was generated, consisting of the 29 features above, supplemented by 16 dispersion features, whose idea is basically adapted from [19]. We therefore compared the arithmetic mean as well as the median of the pairwise distances of the entire data to the same measurement of the $\alpha\%$ best observations (w.r.t. the objective value), with $\alpha \in \{2, 5, 10, 25\}$. The comparison itself is employed by either using the quotient or the difference of the two measurements.

Based on the two previously introduced training data sets, we were able to train various models, using different classifiers, namely a classification tree (*rpart*, [9]), a random forest (*rf*, [8]), consisting of 500 classification trees, and kernel versions of k-nearest neighbors (*kknn*, [16]) and support vector machines (*ksvm*, [10]).

Although we first introduced the classes *random*, *linear* and *funnel*, we will actually make it a 2-class problem, because *linear* problems can be considered as a small subgroup of *funnel* problems. Therefore, we integrated the former one into the latter class, even though this leads two an uneven set of classes (67% *funnel* and 33% *random*).

In order to avoid overfitting to the data, we reduced the set of features per classifier, by conducting a forward-backward-feature selection, supported by a nested 10-fold cross-validation as shown in Fig. 2. For the outer cross-validation, we divided our data set into ten blocks of 600 observations each, where all ten replications that belong to the same instance were located within the same block. Then, nine of the ten blocks, i.e., 5,400 instances, were used to find the best subset of features over those instances. This is done by starting with an empty set of features and iteratively adding or removing a feature from that set until the performance of the model, which is trained on that set, does not improve. That performance itself is again evaluated with a 10-fold cross-validation – the so-called inner resampling. The feature set, which resulted in the best performance will then be trained on all 5,400 instances and evaluated on the remaining 600 instances. This approach is repeated ten times, so that the model performance was evaluated on each of the ten blocks exactly once. Using that approach, we receive ten different models – one per fold. However, as we are only interested in a single model, we use all 6,000 instances and all the features, which were selected within at least two of the ten folds, to train another model on the complete data. As we were using four different classifiers and two versions of initial feature sets, we received a total of eight models. The entire feature selection, modeling and performance evaluation was performed using the statistical software `R` [25] and the `R`-package `mlr` [6].

Problem		Settings		Classification method			
group	spec.	CV	disp.	rf	kknn	rpart	ksvm
model	funnel	yes	no	2.4%	5.9%	**5.1%**	3.0%
model	random	yes	no	16.4%	11.9%	**20.9%**	15.0%
model	total	yes	no	7.1%	7.9%	**10.4%**	7.0%
model	funnel	no	no	0.0%	0.0%	**1.2%**	1.4%
model	random	no	no	0.0%	0.0%	**11.5%**	10.9%
model	total	no	no	0.0%	0.0%	**4.7%**	4.5%
BBOB	funnel	no	no	5.2%	6.2%	**2.1%**	0.0%
BBOB	random	no	no	6.1%	68.3%	**7.5%**	86.5%
BBOB	total	no	no	5.4%	16.6%	**3.0%**	14.4%
disc-pack.	2–10	no	no	80/10	30/60	**80/10**	60/30
disc-pack.	11–30	no	no	200/0	0/200	**0/200**	0/200
model	funnel	yes	yes	3.2%	4.7%	5.1%	4.3%
model	random	yes	yes	15.0%	11.1%	17.5%	16.8%
model	total	yes	yes	7.1%	6.8%	9.2%	8.5%
model	funnel	no	yes	0.0%	0.0%	1.2%	1.0%
model	random	no	yes	0.0%	0.0%	12.5%	8.2%
model	total	no	yes	0.0%	0.0%	5.0%	3.4%
BBOB	funnel	no	yes	12.4%	6.3%	14.5%	0.0%
BBOB	random	no	yes	0.8%	67.9%	5.0%	99.6%
BBOB	total	no	yes	10.4%	16.6%	12.9%	16.6%
disc-pack.	2–7	no	yes	90/0	40/50	50/40	80/10
disc-pack.	8–30	no	yes	200/0	20/180	0/200	200/0

Table 1: Comparison of the misclassification rates and predictions (random / funnel) for the four classifiers, based on two different feature sets (with / without dispersion features) and evaluated on all three data sets. Our recommended model is highlighted in red.

3.2 Modelling Results

Tab. 1 summarizes the results of the four classifiers. At this stage, all entries but those related to validation experiments on the BBOB function set and disc packing instances (see §4) are relevant to assess model quality. The column *CV* indicates whether the misclassification errors based on 10-fold cross-validation or for a classifier trained on the whole problem set are reported. The latter functions as a prediction model for so far unknown function instances. The results are further distinguished regarding the integration of the dispersion features [19] into the candidate feature set. Total misclassification errors are supplemented by respective information for the two considered classes individually so that the balance between the prediction accuracies in both classes can be assessed.

The total misclassification rates for all entries are quite moderate, i.e. at most roughly 10 % which indicates that the considered features provide a suitable basis for the considered classification task. In general, the misclassification rates w.r.t. the *random* class exceed the respective ones for the *funnel* class by some orders of magnitude. Thus, the classifiers have a certain bias towards predicting a funnel structure.

We decided against the inclusion of the dispersion features though at first sight no substantial differences in the modeling results become obvious. However, those models perform worse on the validation sets in terms of balanced errors within the two classes and also w.r.t. total misclassification rates (§4). The final classifier recommendation marked in red, i.e. the decision tree *rpart*, as well results from such considerations supplemented by the fact that model complexity and thus computation times are lowest compared to the other classifiers. Additionally, the decision tree can be nicely visualized as shown in Fig. 3. Here, the tree trained on the

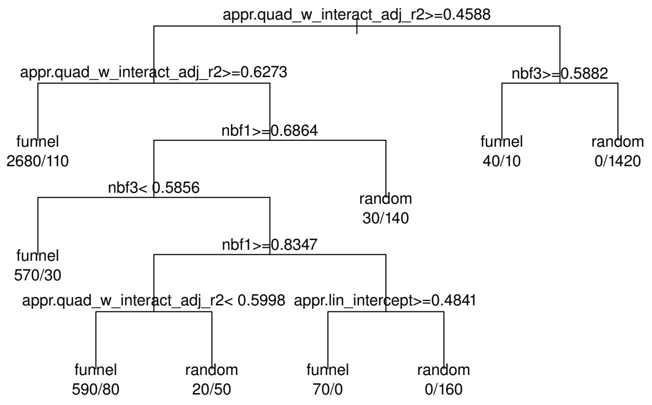

Figure 3: The final rpart-model, i.e. the tree which was built on all observations of the generator problems, but with a reduced feature set. The latter consists of the four features that were selected during the feature selection process, i.e., the features that are shown in the upper panel of Fig. 4.

whole function set without cross-validation is provided as it serves as the final prediction model for the application to BBOB and disc packing later on. Crucial features are those provided by the Meta-Model ELA features (beginning with "appr" in the figures). Especially the one that fits a quadratic regression model with interactions intuitively has a strong relation to global structure. Two nearest-better features play an important role as well, i.e. nbf1 and nbf3. Interestingly, a very small and meaningful subset (four features) out of the whole feature set is sufficient to provide a highly accurate classification.

Fig. 4 visualizes feature importance for the cross-validated *rpart*. The red color reflects features which were used at least seven times in the final classification tree. The displayed feature set is restricted to features which were selected at least twice. Surprisingly, the problem dimensionality is very relevant in the CV-setting while it is not included in the final decision tree.

For illustration purposes, the respective importance plot in case the feature set is supplemented by the dispersion features is given in the lower part of the figure. It can be seen that problem dimensionality clearly is the most important feature now, while the dispersion features are ranked next. Nearest-better features are partly selected within the CV iterations. Thus, it can be concluded that in case the dispersion features are neglected, the combination of meta-model features and nearest-better features completely capture the corresponding information and additionally also diminish the role of problem dimensionality.

4. VALIDATION EXPERIMENTS

In order to assess the quality of our models, we evaluate their performances on two other well-known research applications: the benchmark for continuous black-box optimization problems (BBOB [13, 14]) and disc packing problems [4].

4.1 Experiment on BBOB Problems

The BBOB set consists of 24 continuous functions, which claim to cover a broad variety of function types. In previous works (e.g. [7]), the functions were examined w.r.t. various properties, such as the multimodality and global structure. We used that information to manually assign the topology of the BBOB functions, as shown in Tab. 2. Afterwards, we employed our classifiers to predict that topology. Therefore, a test set of the BBOB functions, consisting of all 24 functions and four dimensions $n \in \{2, 3, 5, 10\}$, was created. For each function, 15 instances were generated, which differ from each other due to rotation, scaling and shifts. Finally, all instances were replicated ten times in order to handle the stochasticity of some of the features. Then, the eight models were used to predict for each of those 14,400 instances, whether it is rather *random* or *funnel*.

The previously discussed Tab. 1 also presents the classification errors of our classifiers on the BBOB function set in relation to Tab. 2. Only four of the 24 functions shall be considered as of random topology, all others (including the many unimodal functions) possess one or several (functions 20 and 24) funnels. Given that we only apply classifiers that have been trained on a different set of test cases, the error rates are very good. We can observe two trends:

- Except for the *kknn* method (where they are comparable), the results for the feature set without dispersion features are clearly better than for the one including those.
- For both feature sets, the *kknn* and *ksvm* based classifiers fail badly in recognizing random peak topologies.

This naturally narrows our choice down to *random forest* or *rpart* based classifiers on the set without dispersion features. The misclassification rates for both are similar with a slight

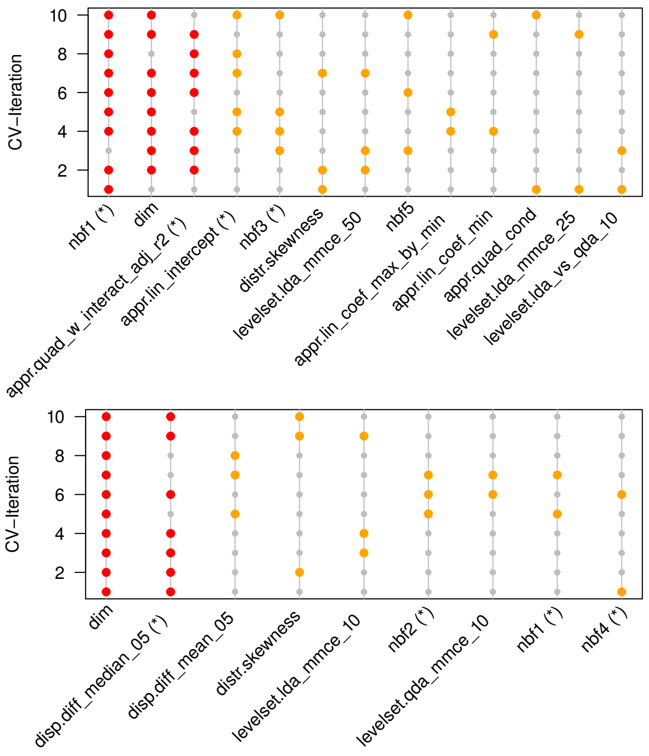

Figure 4: Feature importance plots for the 10-fold CV of the classification tree (top: without dispersion; bottom: with dispersion). Only features that were chosen at least twice are plotted. The ones selected at least seven times are shown in red, those of the final classification tree are marked with (*).

#	Function	multi-modality	global structure	funnel/ random
1	Sphere	none	strong	funnel
2	Ellipsoidal separable	none	strong	funnel
3	Rastrigin separable	high	strong	funnel
4	Bueche-Rastrigin	high	strong	funnel
5	Linear Slope	none	strong	funnel
6	Attractive Sector	none	strong	funnel
7	Step Ellipsoidal	none	strong	funnel
8	Rosenbrock	low	strong	funnel
9	Rosenbrock rotated	low	strong	funnel
10	Ellipsoidal high-conditioned	none	strong	funnel
11	Discus	none	strong	funnel
12	Bent Cigar	none	strong	funnel
13	Sharp Ridge	none	strong	funnel
14	Different Powers	none	strong	funnel
15	Rastrigin multi-modal	high	strong	funnel
16	Weierstrass	high	med.	random
17	Schaffer F7	high	med.	funnel
18	Schaffer F7 mod. ill-conditioned	high	med.	funnel
19	Griewank-Rosenbrock	high	strong	funnel
20	Schwefel	med.	deceptive	funnel
21	Gallagher 101 Peaks	med.	none	random
22	Gallagher 21 Peaks	low	none	random
23	Katsuura	high	none	random
24	Lunacek bi-Rastrigin	high	weak	funnel

Table 2: Classification of the noiseless BBOB functions into the two high-level properties multimodality and global structure. In addition, each function was assigned the label funnel or random – based on expert knowledge.

advantage for the *rpart* based classifier. The advantage of the latter lies in the much faster computation and simpler structure. The choice made in §3.2 therefore also holds true for the BBOB problem classification task.

4.2 Application to Disc Packing

The general task of sphere packing problems is to arrange a given number of non-intersecting spheres with equal radius in a container so that the radius becomes maximal. A special case of this problem in $\mathbb{R}^2$ is to arrange discs in a square. If d is the number of discs, the dimensionality of the corresponding optimization problem is $n = 2d$. There is only one problem instance per dimension.

Globally optimal solutions for this problem, taken from [26], are shown in Fig. 5. These optima are not unique, as the arrangement of discs can be mirrored or rotated without changing the objective value. Additionally, discs which are not in contact with others or the boundary can also be moved without affecting the objective value. Addis et al. argue that disc packing problems possess a funnel structure, similar to the closely related molecular conformation problems [4]. Apart from the symmetries, the funnel property probably stems from the fact that the optimal arrangements contain some regularity, causing locally optimal solutions to have a lot in common with the global ones. We therefore use these problems as an additional test case to validate our classifiers.

For this test problem, we created 290 instances – ten replications for any number of discs between two and 30. Then we used all eight classifiers to predict the topology. Interestingly, all replications that belong to the same number of discs were predicted into the same category. The exact amount of predictions into *random* / *funnel* for each of the models is shown in Tab. 1.

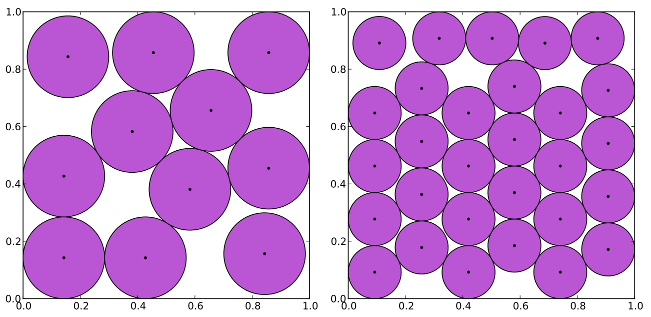

Figure 5: Optimal solutions for the disc packing problems with 11 and 29 discs. The coordinates of the disc centers determine the maximally possible radius.

Discs	random forest	kknn	rpart	ksvm
2	funnel	funnel	funnel	funnel
3	random	random	random	random
4	random	random	random	random
5	random	random	random	random
6	random	funnel	random	random
7	random	funnel	random	random
8	random	funnel	random	random
9	random	funnel	random	funnel
10	random	funnel	random	funnel
11–30	random	funnel	funnel	funnel

Table 3: Class predictions of our classifiers on the disc packing problems (without dispersion features).

In contrast to the previous experiments, we did not look at the misclassification rates, as we are not certain how many discs are needed to make this problem a funnel problem. However, it seems that the funnel topology only emerges in higher dimensions as shown in Tab. 3, which shows the predictions of the classifiers that do not use the dispersion features. The 2-disc problem was always classified as a *funnel* problem, whereas the next problems usually were classified as *random*. With the exception of the random forest, all higher-dimensional problems were predicted to be *funnel* problems. In general, the models based on dispersion features tend to have a strong bias regarding predicting random structures in this setting, especially the random forest and *ksvm*. These mixed results do not contradict the findings of Addis et al. [4], because they only regard problems with 50 discs or more.

5. CONCLUSIONS

As main result of this work we can state that we have established classifiers for experimentally detecting from a small sample, if an unknown black-box optimization problem has the funnel property, a previously unsolved problem. This resembles the first, necessary step for performing (optimization) algorithm selection as we assume that this information can be exploited to speed up the optimization (see the discussion in §1.1). The recommended classifier employs the *rpart* method, which results in a very simple tree with very few features (only four in our case). It is thus easy to interpret the model, and at the same time it provides us with highly accurate and well balanced predictions.

The newly introduced nearest-better features obviously capture important information concerning the problem type (*funnel* or *random*) and therefore seem to be at least as valuable as the dispersion features. All these results have been validated on two additional test sets (BBOB functions and disc packing problems) that have not been used during training. We thus have reason to assume that our classifiers can also be of use for other real-valued optimization problems.

However, the fact that the more sophisticated classifiers exhibit a worse performance than the simple ones is a hint that the training of the classifiers could be still improved. In the future, it probably would be advisable to extend the training set for the feature selection task, to rule out any overfitting. Ways to do this would be extending the test problem generator to model a broader class of functions or including other test problems from different sources.

Although the nearest-better features seemed superior so far, it may be also promising to investigate into further variants of the dispersion feature. Especially for measuring the diversity of the involved sets, several other options exist [24]. Additionally, it would of course be nice to further reduce the employed sample size for computing any features, which is currently $500n$, but we did not investigate this issue, yet.

Regarding the final goal of generating an algorithm selection model for unknown test problems, the funnel property could be included into the respective feature set. By this means, the modeling procedure becomes sequential in nature. At first our proposed model will be used to predict if a funnel structure is present for all candidate training instances. Afterwards, this feature can be integrated into the algorithm selection modeling phase (see e.g. [7]). Therefore, promising perspectives regarding the analysis of the influence on algorithm performance exist.

6. ACKNOWLEDGMENTS

Heike Trautmann and Pascal Kerschke acknowledge support by the European Center of Information Systems (ERCIS). In addition, this work was supported by the French national research agency (ANR) within the project ”NumBBO – Analysis, Improvement and Evaluation of Numerical Blackbox Optimizer” (ANR-12-MONU-0009-03).

References

[1] T. Abell, Y. Malitsky, and K. Tierney. Features for exploiting black-box optimization problem structure. In G. Nicosia and P. Pardalos, editors, *Learning and Intelligent Optimization*, Lecture Notes in Computer Science, pages 30–36. Springer, 2013.

[2] B. Addis, A. Cassioli, M. Locatelli, and F. Schoen. A global optimization method for the design of space trajectories. *Computational Optimization and Applications*, 48(3):635–652, 2011.

[3] B. Addis and M. Locatelli. A new class of test functions for global optimization. *Journal of Global Optimization*, 38(3):479–501, 2007.

[4] B. Addis, M. Locatelli, and F. Schoen. Disk packing in a square: A new global optimization approach. *INFORMS Journal on Computing*, 20(4):516–524, 2008.

[5] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *IEEE Congress on Evolutionary Computation (CEC)*, volume 2, pages 1769–1776, 2005.

[6] B. Bischl, M. Lang, J. Richter, J. Bossek, L. Judt, T. Kuehn, E. Studerus, and L. Kotthoff. *mlr: Machine Learning in R*. R package version 2.3.

[7] 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. ACM, 2012.

[8] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[11] A. E. Eiben and M. Jelasity. A critical note on experimental research methodology in EC. In *IEEE Congress on Evolutionary Computation (CEC)*, volume 1, pages 582–587, 2002.

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

[13] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2009: Experimental setup. Technical Report RR-6828, INRIA, 2009.

[14] N. Hansen, A. Auger, S. Finck, and R. Ros. Real-parameter black-box optimization benchmarking 2010: Experimental setup. Technical Report RR-7215, INRIA, 2010.

[15] N. Hansen and S. Kern. Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature – PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 282–291. Springer, 2004.

[16] K. Hechenbichler and K. Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. 2004.

[17] P. Kerschke, M. Preuss, C. Hernández, O. Schütze, J. Sun, C. Grimme, G. Rudolph, B. Bischl, and H. Trautmann. Cell mapping techniques for exploratory landscape analysis. In A. Tantar et al., editors, *EVOLVE — A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V*, volume 288 of *Advances in Intelligent Systems and Computing*, pages 115–131. Springer, 2014.

[18] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search: Framework and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 363–397. Springer, 2010.

[19] 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*, GECCO '06, pages 477–484. ACM, 2006.

[20] 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. ACM, 2011.

[21] R. Morgan and M. Gallagher. Using landscape topology to compare continuous metaheuristics: A framework and case study on EDAs and ridge structure. *Evolutionary Computation*, 20(2):277–299, 2012.

[22] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. A meta-learning prediction model of algorithm performance for continuous optimization problems. In C. A. C. Coello et al., editors, *Parallel Problem Solving from Nature – PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pages 226–235. Springer, 2012.

[23] 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.

[24] M. Preuss and S. Wessing. Measuring multimodal optimization solution sets with a view to multiobjective techniques. In *EVOLVE – A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IV*, volume 227 of *Advances in Intelligent Systems and Computing*, pages 123–137. Springer, 2013.

[25] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[26] E. Specht. `http://www.packomania.com`, Retrieved: 2 Feb 2015.

[27] A. Törn and A. Zilinskas. *Global Optimization*, volume 350 of *Lecture Notes in Computer Science*. Springer, 1989.

[28] 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.