The R-Package FLACCO for Exploratory Landscape Analysis with Applications to Multi-Objective Optimization Problems

Pascal Kerschke
Information Systems and Statistics
University of Münster
48149 Münster, Germany
Email: kerschke@uni-muenster.de

Heike Trautmann
Information Systems and Statistics
University of Münster
48149 Münster, Germany
Email: trautmann@uni-muenster.de

***Abstract*—Exploratory Landscape Analysis (ELA) aims at understanding characteristics of single-objective continuous (black-box) optimization problems in an automated way. Moreover, the approach provides the basis for constructing algorithm selection models for unseen problem instances. Recently, it has gained increasing attention and numerical features have been designed by various research groups. This paper introduces the `R`-Package `FLACCO` which makes all relevant features available in a unified framework together with efficient helper functions. Moreover, a case study which gives perspectives to ELA for multi-objective optimization problems is presented.**

1. Introduction

The selection of the "best" algorithm for a given optimization problem out of a given portfolio has become a very complex challenge especially as the amount of introduced sophisticated optimization algorithms has heavily increased. It is well known that a global best optimizer over all kind of optimization algorithms does not exist, and the discussion of the so-called algorithm selection problem (ASP) dates back to [1] already in 1976. Therefore, we are interested in deriving rules or ideally models linking problem characteristics to algorithm performance, usually based on systematically designed benchmarks. By this means we are able to give algorithm recommendations for certain classes of optimization problems which differ in the kind of exhibited problem characteristics.

Exploratory Landscape Analysis (ELA), focussed on single-objective continuous black-box optimization, addresses these issues by providing cheaply computable features based on an initial small problem sample. Subsequently, models relating features to algorithm performance can be constructed by means of machine learning techniques based on systematic algorithm benchmark experiments. On this basis models can be constructed which accurately predict the best suited algorithm out of a given portfolio for an arbitrary optimization problem making use of features computed prior to optimization.

During the last years we successfully made important steps into this direction [2]–[6] and sophisticated feature sets have been proposed by various research groups, e.g. [7]–[11]. An overview is given in [12]. Section 2 provides a more detailed discussion. However, no algorithm selection or landscape analysis paper so far considered all relevant features within an experimental study as a comprehensive unified framework consisting of all relevant feature sets was not available. The R-package `FLACCO` [13], detailed in Section 3, fills this gap and additionally provides efficient helper functions for extracting relevant information and creating meaningful figures.

Moreover, the urgent need for ELA in the multi-objective optimization context is discussed and a case study is presented in Section 4 which not only illustrates the usage of the `FLACCO` package in terms of coding and parametrization but as well provides a first step towards multi-objective ELA on common multi-objective benchmark problems. Conclusions and an outlook on future work are provided in Section 5.

2. Exploratory Landscape Analysis

The main idea of ELA is to gather as much knowledge of a (usually black-box) optimization problem prior to optimization in order to decide which algorithm will be most appropriate to solve it. In practice, usually function evaluations are quite expensive so that it is impossible to run all candidate algorithms sequentially or even in parallel and to pick the most suitable one afterwards. Common characteristics of continuous optimization problems can be defined by so-called high-level features [14] set by experts, such as the degree of multimodality, global structure, separability, variable scaling, search space homogeneity, basin-sizes, global to local contrast and plateaus. However, those are debatable to a certain extent and, as a key point, cannot be determined numerically in an automated fashion on new problems or problem instances.

In the initial ELA paper [2] experimental low-level features were introduced which aim at numerically characterizing the landscape properties of a problem to be solved based on systematic sampling of the decision space, e.g. using a latin hypercube design. Naturally, the number of design points has to be kept as small as possible to minimize the computational overhead. By combining features with algorithm benchmark data on representative problems, algorithm selection models are constructible which are designed to generalize to unseen problems. Six low-level feature classes, 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), were presented supplemented by local searches 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). Each class comprises a set of sub-features which result from the same experimental data generated from the initial design and the combination of all features gives the complete picture of the desired overall landscape's characteristics.

In recent years, researchers all over the world have developed new landscape features, implemented in many different programming languages, i.e. mainly in R [15], Matlab [16] or Python [17] while almost all approaches are based on an initial function sample. The dispersion features of [9] compare pairwise distances of all points in the initial design with the pairwise distances of the respective best points. Ruggedness of a landscape is investigated by features introduced in [18] based on fitness-distance computations. So-called information content (cf. [11], [19]) is reflected by features measuring the change in the objective values of neighboring points along a tour across the landscape. Hill climbing together with the analysis of random points is focussed in [7].

By discretizing the continuous space Kerschke et al. [4] presented features based on a cell mapping approach. Moreover, barrier trees form the basis of the mathematical landscape analysis in [20]. Most recently, features reflecting funnel structures of optimization problems have been presented in [5] and [6] making use of nearest better clustering, and length scale properties are analysed in [10]. An overview of feature-based approaches to algorithm selection is provided in [12] including a cost-sensitive learning approach based on the initial ELA features in [3].

2.1. Transfer to Multi-Objective Optimization

Despite the success of ELA in single-objective continuous black-box optimization and significant progress in automated algorithm selection based on the ELA approach the important class of multi-objective optimization problems has not been focussed yet apart from a limited number of studies in multi-objective combinatorial optimization [21]–[24]. Challenging real world optimization problems usually are multi-objective in nature, i.e. consist of multiple, usually conflicting objectives. Evolutionary multi-objective optimization algorithms (EMOA, [25]) have proven to be high-performing, especially in the case of black-box objective functions. A huge number of EMOA with increasing complexity have been presented in the community and usually tested on well-established benchmark sets of objective functions (e.g. the DTLZ [26] or ZDT [27] functions).

In order to understand the results of such benchmark experiments a systematic characterization of the considered benchmark problems is required which can be linked to algorithm performance and will hopefully allow the construction of automated algorithm selection models in the long run. However, besides some straightforward and expert-based characterizations such as the shape of the Pareto front, the number of objective functions, the decision space dimensionality and some intuitions about multimodality of the problems, no meaningful experimental features for numerically characterizing the difficulty of multi-objective optimization problems exist. Once such features have been set up, reasons for difficulty of multi-objective optimization problems can be investigated and strengths as well as weaknesses of specific EMOAs as well as their underlying operators can be analyzed which will have high potential for systematic and sophisticated algorithm design.

As a first approach, the single-objective ELA features can be computed for each objective function individually which will provide sound knowledge about the characteristics of the respective objective functions separately. However, by this means the interaction of the functions is not addressed at all, i.e. specific numerical characteristics of the *multi-objective* landscape are required.

The case study in Section 4 illustrates the functionality and the usage of the `FLACCO`-Package by means of characterizing the functions of the DTLZ and ZDT test sets by means of ratios of ELA features of the individual objectives which can be seen as an initial step towards the multi-objective ELA issue.

3. The R-Package `FLACCO`

As shown in the previous chapter, numerous landscape features have been developed in the past years. Unfortunately, the corresponding implementations are spread over multiple platforms which made a combined usage of different feature sets – i.e. collections of features, which are based on the same approach and only differ in the way, they aggregate the results – rather complicated. However, as single feature sets measure at most a few properties, the usage of multiple feature sets at once is necessary for a careful characterization of a problem instance. This issue has recently been approached with the creation of `FLACCO` [13], an R-package for **f**eature-based **l**andscape **a**nalysis of **c**ontinuous and **c**onstrained **o**ptimization problems. It can nicely be used in combination with the R-packages `smoof` (Single and Multi-Objective Optimization Test Functions, [28]) as well as `mlr` (Machine Learning in R, [29]).

In its current version[1], the package contains a total of more than 300 features, distributed across the following 17 feature sets:

1. The stable release can be found on CRAN, whereas the developer's version is available on GitHub.

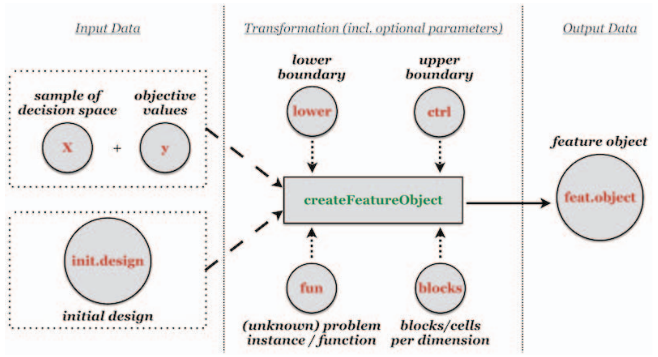


Figure 1. Scheme for creating a feature object: (1) provide the input data, (2) call `createFeatureObject` and pass additional arguments if applicable, (3) receive the feature object, i.e. a list, which is the basis for all further computations and visualizations.

- Initial ELA Features (Convexity, Curvature, y-Distribution, Meta Model, Local Search, Levelset)
- Cell Mapping (Angle, Convexity and Gradient Homogeneity) and Generalized Cell Mapping Features
- Barrier Tree Features
- Nearest Better Clustering Features
- Information Content Features
- Dispersion Features
- Miscellaneous Approaches (Basic, Linear Models, Principal Components)

In either of the feature computations, as well as for all the other functionalities of the package, one first has to create the so-called *feature object* (using `createFeatureObject`, see Fig. 1), a specific R object, which stores the entire data set of the decision space, as well as additional information such as the number of dimensions and observations, the upper and lower bounds and the name of the objective. The following code snippet shows how one can create an initial sample using `flacco`. Each line of code begins with ">" (or "+" if it continues the previous line of code) and comments within the code are introduced with a hash tag (#). The snippet begins with the creation of a control object (stored under the name `ctrl`), which specifies the sample type `"lhs"` (latin hypercube sample, [30]) and the upper bounds for the initial sample. Without specifying the bounds, the following function (`createInitialSample`) would create a sample within $[0,1]^d$ (with d being the number of dimensions). In the given example, the sample consists of 200 three-dimensional observations. In a next step, the corresponding 200 objective values, $y_i = \sum_{j=1}^{3} x_{ij}^4$, $i = 1, \ldots, 200$, are computed.

```
> # define a control object
> ctrl = list(init_sample.type = "lhs",
+   init_sample.upper = c(7, 1, 19))
>
> # create an initial sample and the corresponding
> # objective values
> X = createInitialSample(n.obs = 200, dim = 3,
+   control = ctrl)
> f = function(x) sum(x^4)
> y = apply(X, 1, f)
```

Now, one can create the corresponding feature object, containing the matrix with the initial sample `X`, the vector of objective values `y`, the corresponding objective function `f` and a vector with the number of `blocks` (= cells) per dimension, which are required for the computation of the cell mapping features. Afterwards one can print the created feature object (just by calling the name of the object, i.e. `feat.object`) and receive a summary of the provided data.

```
> # create a feature object
> feat.object = createFeatureObject(X = X, y = y,
+   fun = f, blocks = c(3, 5, 4))
>
> # have a closer look at the feature object
> feat.object
Feature Object:
- Number of Observations: 200
- Number of Features: 3
- Lower Boundaries: 0.00e+00, 0.00e+00, 0.00e+00
- Upper Boundaries: 7.00e+00, 1.00e+00, 1.90e+01
- Name of Features: x1, x2, x3
- Optimization Problem: minimize y
- Function to be Optimized: function (x) sum(x^4)
- Number of Cells per Dimension: 3, 5, 4
- Size of Cells per Dimension: 2.33, 0.20, 4.75
- Number of Cells:
  - total: 60
- non-empty: 60 (100.00%)
- empty: 0 (0.00%)
- Average Number of Observations per Cell:
  - total: 3.33
- non-empty: 3.33
```

Based on such a feature object, one now can compute feature sets. Each of the feature sets can either be computed in combination with all the other feature sets via the R command `calculateFeatures`, i.e.

```
> all.features = calculateFeatures(feat.object)
```

or independently from the other ones via `calculateFeatureSet`. In the latter case, one has to specify the feature set with the `set` argument. For example:

```
> dispersion.features = calculateFeatureSet(
+   feat.object = feat.object, set = "disp")
> nbc.features = calculateFeatureSet(
+   feat.object = feat.object, set = "nbc")
```

The computation of just a single feature itself is not supported within the package, as it is not computationally efficient. However, one can of course compute the entire feature set and afterwards inspect the single features of that set. Features belonging to the same feature set are based on the same approach and only differ in the (rather cheap) way they aggregate the respective information. The resulting features are supplemented by two additional variables providing the costs for calculating the specific feature set, i.e. the number of required function evaluations as well as the runtime in seconds. Both contain important information w.r.t. algorithm selection approaches which have to take the costs of feature computations into account.

Furthermore, it is possible to adapt each feature to the problem's landscape by configuring feature-specific control parameters. By not explicitly setting those parameters, the features will be employed in their original version, i.e. as implemented by their developers. Note that some features are stochastic, e.g. the local search or the convexity features, so that it is strongly recommended to calculate them multiple times. Afterwards, one has to make a decision on either using all replicates or applying a meaningful aggregation per feature and problem instance. While aggregating the feature replicates usually leads to rather robust values, it also results in a loss of information.

In addition to the feature calculations, FLACCO provides further useful functionalities such as (1) a random sample generator (either using random uniform sampling or a latin hypercube sample creating samples with specified dimensions, numbers of observations and boundaries, (2) visualizations of various feature sets (namely the general cell mapping, barrier trees and information content feature sets), as well as (3) the so-called feature importance plot (cf. section 4) based on the output of a feature selection using resampling strategies, such as cross-validation or bootstrap (e.g. as coming from the mlr [29] package).

Moreover, the package comes with an extensive R documentation, an online tutorial[2], as well as an issue tracker for suggestions of improvements or possible bug reports on the corresponding GitHub developers site[3].

As the number of landscape features will still increase – especially w.r.t. multi-objective problems – it is obvious that flacco needs to be updated / extended on a regular basis. Any user can contribute to the package, e.g. by implementing a new set of features or analysis tools. In order to contribute a feature set, the user needs to write a function in R which computes the new feature set based on a given feature object. This code can then be submitted to the developers site[3] and will eventually be integrated into the package.

4. ELA for Multi-Objective Optimization Problems Using FLACCO: A Case Study

The following case study means to illustrate the working mechanism of the FLACCO package in detail. For this purpose, a first approach for understanding characteristics of bi-objective test problems is presented using a combination of single-objective ELA features, see section 2.1 for the respective background. Here, we assume that the ratios of the individual ELA features influence the characteristics of the multi-objective landscape. Moreover, using this approach we do not have to consider both objective functions individually in the subsequent feature analysis step. We will investigate whether meaningful subgroups of multi-objective test functions can be derived based on this information.

4.1. Experimental Setup

In a first step, we use the smoof R-package [28], to create the twelve bi-objective test functions DTLZ1 to DTLZ7 and ZDT1 to ZDT6 (with the exception of ZDT5, which is

2. kerschke.github.io/flacco-tutorial/
3. https://github.com/kerschke/flacco

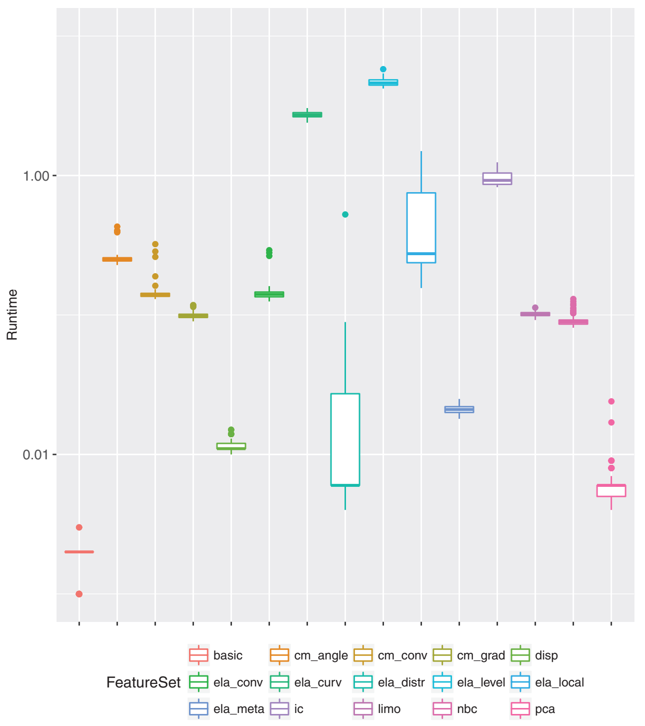


Figure 2. Runtime (in seconds) of computing the considered ELA feature groups (shown on a logarithmic scale).

currently not available in smoof) for 3-dimensional decision vectors. The low decision space dimension is chosen in order to ensure a meaningful applicability of the cell mapping features discretizing the decision space. Of course we usually encounter higher dimensions in practice. The function objectives were generated as follows illustrated by means of two exemplary functions:

```
> dtlz1 = makeDTLZ1Function(dimensions = 3,
>   n.objectives = 2)
> zdt4 = makeZDT4Function(dimensions = 3)
```

The lower and upper bounds of each function were extracted by means of a specific helper function from the smoof package and used to create a latin hypercube sample X of the decision space, consisting of three decision variables and 300 observations with FLACCO:

```
> lower = getLowerBoxConstraints(zdt4)
> upper = getUpperBoxConstraints(zdt4)
> ctrl = list("lhs", lower, upper)
> X = createInitialSample(n.obs = 300, dim = 3,
>   control = ctrl)
```

Of course, the size of the initial design has to be chosen carefully w.r.t. algorithm selection approaches, i.e. as small as possible without losing (much) information. In this case study, we did not encounter substantial differences in the results in case the size of the initial design was decreased from $200 \cdot d$ to $100 \cdot d$ with d denoting the decision space dimension so that we naturally opted for the smaller version.

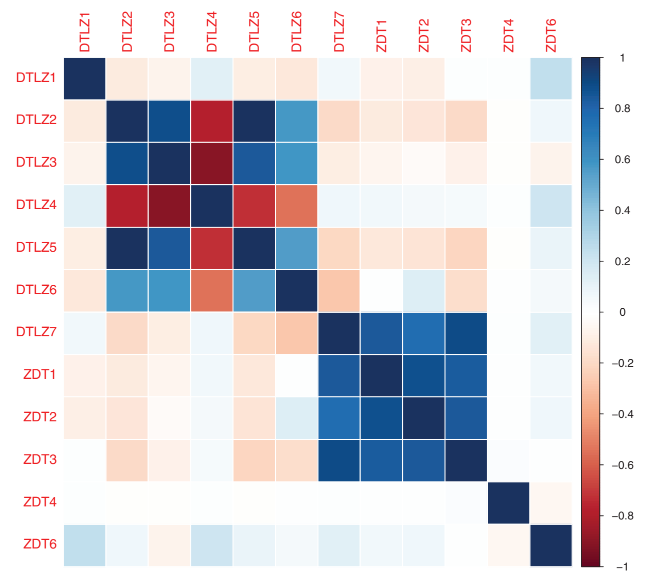


Figure 3. Visualization of the correlation matrix to the twelve multi-objective test problems. This plot suggests two bigger clusters and three to four singleton clusters.

For each of the bi-objective problem instances, two feature objects (one per objective) were created via `createFeatureObject`, resulting in a total of 24 single-objective problems. Afterwards all 15 feature sets contained in `FLACCO`, with the exception of the *general cell mapping* and *barrier tree features*, are computed per scenario. Those features were excluded as preliminary studies did not reveal a substantial information gain in case of consideration. Note that not all discretized features have been discarded in general – the regular *cell mapping* features are part of the 15 considered feature sets.

As some of the feature sets are stochastic, each feature set was computed ten times, resulting in a total of 120 observations (twelve problems, ten replications) and 682 features (two objectives with 341 features each). In a next step, the size of the data set was cut in half by computing the ratio of the feature values between the first and second objective for each feature. Afterwards, all features that contain infinite or non-defined values (e.g. resulting from division by zero), as well as the pure runtimes of the feature computations, were removed from the data base reducing the number of features to 131. As an orientation, the computation times of all considered feature groups are provided in Fig. 2 (on a logarithmic scale) revealing that there are indeed substantial differences.

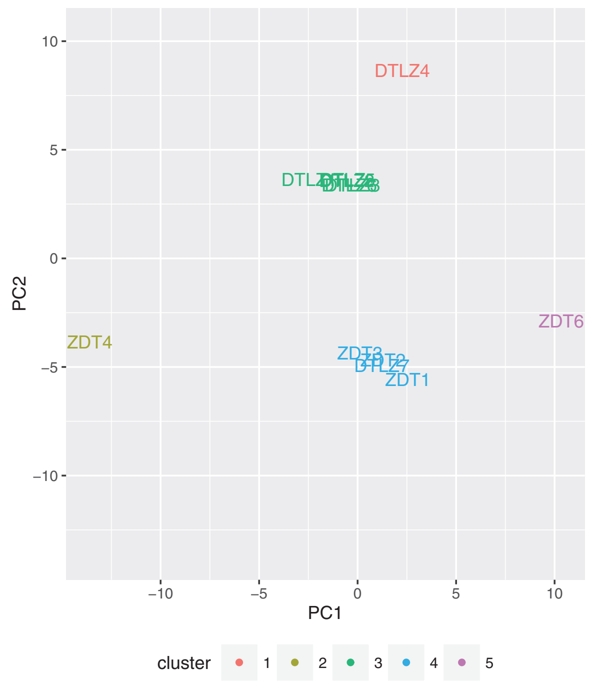
4.2. Results

Fig. 3 investigates the correlation structure of the considered multi-objective test functions w.r.t. the feature levels. For this purpose, we used an aggregated version of the data set, which used the median over the ten replications of a feature (per test function).

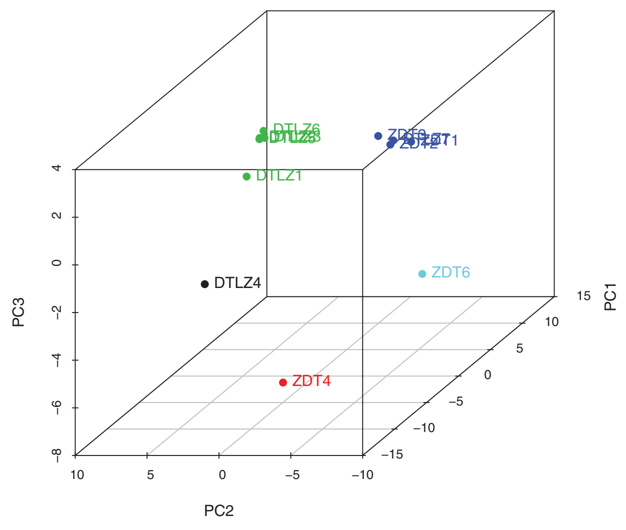
Recall that a feature in the multi-objective setting is represented by the ratio of the two levels of an individual single-objective ELA feature. By this means we aim at capturing the interaction between both function characteristics. The plot indicates the existence of possible five to six groups of functions with similar characteristics, while two big clusters and three separate functions visually emerge. The first block of positive correlations is formed by DTLZ2, 3, 5 and 6, while DTLZ4 is negatively correlated with the latter and thus exhibits different characteristics. DTLZ1 is neither highly positively nor negatively correlated with the remaining functions. The second group consists of the first three ZDT functions, interestingly combined with DTLZ7. ZDT4 as well as ZDT6 do not exhibit strong correlations.

These findings are supported by Fig. 4 which visualizes the first two principal components [31] based on all considered features. Those already explain roughly 65% of the variation. Five clusters of functions can be identified which coincide with the correlation structure reflected in Fig. 3 with the single exception that DTLZ1 is plotted adjacent to DTLZ2, 3, 5 and 6. The 3D-plot of the first three principal components explaining more than 75% of the variance, shows that DTLZ1 is farthest away from the respective cluster center. The k-means clustering algorithm recommends five clusters as well which is reflected by the first "knee" at $k = 5$ of the internal clustering criterion which has to be minimized (see Fig. 5).

In order to understand the characteristics of each function cluster, resp. the features which allow to differentiate between them, a random forest (cf. [32], [33]) was trained using the default settings from `R`, e.g. 500 trees, based on a 12-fold cross-validation using the `mlr` package. However, as some clusters only contain a single function we could not remove the features of an entire function completely as one fold. Instead, as some features are noisy and were repeatedly evaluated, each function consists of ten observations of ELA features. It is made sure that a subset of these features is contained in each fold for each function. It turned out that the quality of the random forest is very good as it resulted in a mean misclassification error of 0. That is, over all folds of the cross validation, the model always predicted the correct cluster. Usually, this result is very suspicious as a perfect prediction could be due to an overfitted classification model. However, in order to avoid such an overfitting, a greedy forward-backward feature selection has been employed within each fold reducing the number of included features to a minimum. That is, within each fold we start with an empty set of features and iteratively try to add (or remove) a feature from that set as long as the misclassification error decreases. Following this approach, we reduced the number of features from originally 131 features to at most two per fold (as shown in Fig. 6) and a total of eleven different features across the twelve folds. In addition to avoid overfitting, the feature selection allows to assess whether a feature actually contributes useful information for the characterization of a problem.



(a) 2D



(b) 3D

Figure 4. Clustering of the multi-objective test problems represented by the first two (a) or three (b) principal components.

Fig. 6 lists the features, which have been selected by the internal feature selection. The most important feature, i.e. the feature that has been selected in each of the folds (depicted in red within the *feature importance plot*), is the ratio of the maximum objective values of a problem's two objectives over the sample points. In each of the twelve folds, this feature was combined with exactly one further feature, in order to perfectly model the cluster. In a quarter of the cross-validation folds, that supporting feature has been `cm_conv.convex.hard`, one of the *cell mapping* features, which basically measures the convexity of the problem. The remaining "supporting features" either measure the dispersion or characterize the problems with some information derived from simple linear or quadratic models.

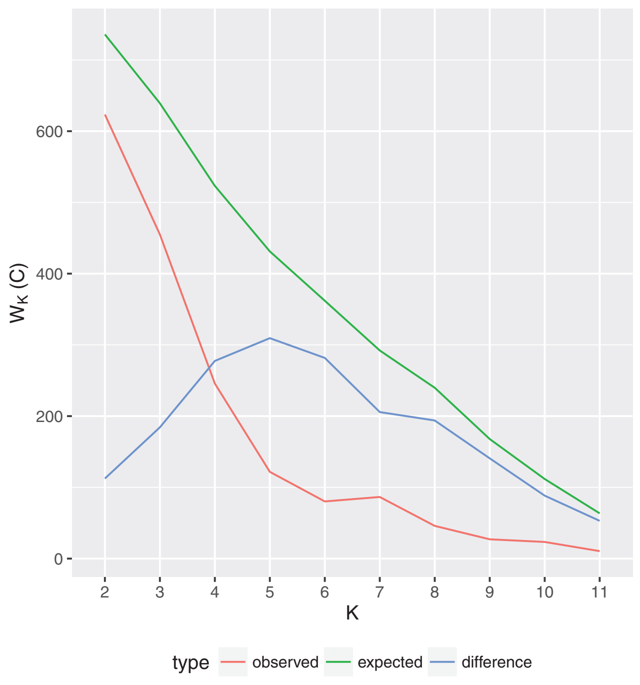


Figure 5. Gap K-Means

This behavior can be seen in Fig. 7, a parallel coordinate plot of the selected features across the multi-objective test functions. Note that all features have been normalized prior to generating this plot, allowing to better compare the magnitude of the features. Feature `basic.objective_max` perfectly divides the twelve test functions into two groups: cluster 1 (orange, DTLZ4) and 2 (blue, DTLZ1, 2, 3, 5, 6) in one group and clusters 3 (red, DTLZ7, ZDT1, 2, 3), 4 (light blue, ZDT4) and 5 (green, ZDT6) within the other group. The remaining features then can be used to split the two groups into the correct clusters.

5. Conclusions and Outlook

The paper presents a unified comprehensive framework for exploratory landscape analysis by providing the most relevant numerical landscape features for continuous black-box optimization within the R-package `FLACCO`. By this means future studies on landscape analysis as well as on automated algorithm selection techniques can be carried out on the same basis. Those therefore become comparable and

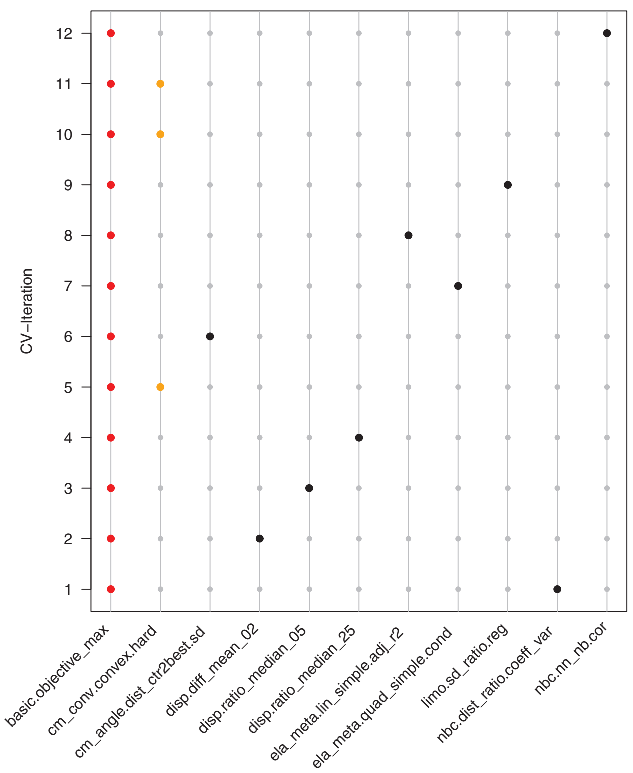


Figure 6. Feature importance plot as derived from a 12-fold cross-validated (forward-backward) feature selection. The first feature was selected in each of the CV folds (shown in red), the second feature in three of the twelve folds (orange) and the remaining features were selected once during the folds (black) for training a classification model.

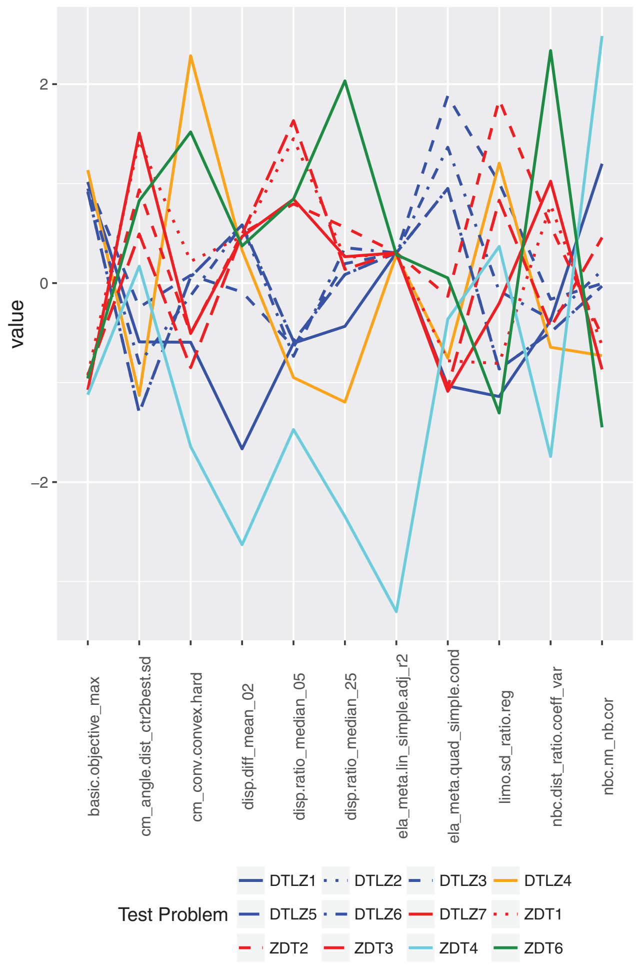


Figure 7. Parallel coordinate plot of a subset of features allowing the detection of characteristics among the twelve multi-objective test problems, as well as the previously found clusters. The employed features are the subset of landscape features, which has been found by a feature selection.

do not have to be carried out using individual source code across several platforms. Specific helper functions facilitate the experimental analyses. The package will be continuously updated w.r.t. the progress in the research community regarding the ELA issue. Moreover, it technically forms the basis for extending ELA to multi-objective optimization problems. Specific features characterizing the multi-objective landscape are required and can be easily integrated into the existing framework. An initial case study motivating the need for ELA in the multi-objective setting and simultaneously providing application details of the FLACCO package are presented.

Acknowledgments

The authors would like to thank Carlos Hernández (from the Department of Computer Science at the CINVESTAV in Mexico City, Mexico) for sharing his MATLAB code of the *general cell mapping* and *barrier tree* features, as well as Jan Dageförde (from the Department of Information Systems at the University of Münster, Germany) for translating the MATLAB code of multiple feature sets into R.

Additionally, the authors acknowledge support by the European Center of Information Systems (ERCIS).

References

[1] J. Rice, “The algorithm selection problem,” *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[2] 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*, ser. GECCO ’11. New York, NY, USA: ACM, 2011, pp. 829–836.

[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*, ser. GECCO ’12. New York, NY, USA: ACM, 2012, pp. 313–320.

[4] 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*. Springer, 2014, pp. 115–131.

[5] 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. ACM, 2015, pp. 265–272.

[6] ——, “Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models,” in *Proceedings of the 18th Annual Conference on Genetic and Evolutionary Computation*. ACM, 2016.

[7] T. Abell, Y. Malitsky, and K. Tierney, “Features for Exploiting Black-Box Optimization Problem Structure,” in *Learning and Intelligent Optimization*. Springer, 2013, pp. 30–36.

[8] M. Gallagher, “Multi-Layer Perceptron Error Surfaces: Visualization, Structure and Modelling,” Ph.D. dissertation, University of Queensland, 2000.

[9] 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*. ACM, 2006, pp. 477–484.

[10] R. Morgan and M. Gallagher, “Analysing and Characterising Optimization Problems Using Length Scale,” *Soft Computing*, pp. 1 – 18, 2015.

[11] 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*, vol. 19, no. 1, pp. 74–87, 2015.

[12] M. A. Muoz, Y. Sun, M. Kirley, and S. K. Halgamuge, “Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges,” *Information Sciences*, vol. 317, pp. 224 – 245, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025515003680

[13] P. Kerschke and J. Dagefoerde, *flacco: Feature-Based Landscape Analysis of Continuous and Constraint Optimization Problems*, 2016, R package version 1.3. [Online]. Available: https://github.com/kerschke/flacco

[14] O. Mersmann, M. Preuss, and H. Trautmann, “Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis,” in *PPSN XI: Proceedings of the 11th International Conference on Parallel Problem Solving from Nature*, ser. Lecture Notes in Computer Science 6238, R. Schaefer *et al.*, Eds. Springer, 2010, pp. 71–80.

[15] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2015. [Online]. Available: http://www.R-project.org/

[16] MATLAB, *Version 8.2.0 (R2013b)*. Natick, Massachusetts: The MathWorks Inc., 2013.

[17] G. VanRossum and The Python Development Team, *The Python Language Reference – Release 3.5.0*. Python Software Foundation, 2015.

[18] C. L. Müller and I. F. Sbalzarini, “Global characterization of the CEC 2005 fitness landscapes using fitness-distance analysis,” in *Applications of Evolutionary Computation*. Springer, 2011, pp. 294 – 303.

[19] M. A. Muñoz, M. Kirley, and S. K. Halgamuge, “Landscape Characterization of Numerical Optimization Problems using Biased Scattered Data,” in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[20] C. Flamm, I. L. Hofacker, P. F. Stadler, and M. T. Wolfinger, “Barrier Trees of Degenerate Landscapes,” *Zeitschrift für Physikalische Chemie International Journal of Research in Physical Chemistry and Chemical Physics*, vol. 216, no. 2/2002, p. 155, 2002.

[21] D. Garrett and D. Dasgupta, “Multiobjective landscape analysis and the generalized assignment problem,” in *Learning and Intelligent Optimization*, ser. Lecture Notes in Computer Science, V. Maniezzo, R. Battiti, and J.-P. Watson, Eds. Springer Berlin Heidelberg, 2008, vol. 5313, pp. 110–124. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-92695-5_9

[22] S. Vérel, A. Liefooghe, and C. Dhaenens, “Set-based multiobjective fitness landscapes: a preliminary study,” in *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011*, 2011, pp. 769–776. [Online]. Available: http://doi.acm.org/10.1145/2001576.2001681

[23] S. Verel, A. Liefooghe, L. Jourdan, and C. Dhaenens, “On the structure of multiobjective combinatorial search space: MNK-landscapes with correlated objectives,” *European Journal of Operational Research*, vol. 227, no. 2, pp. 331 – 342, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221712009630

[24] A. Liefooghe, S. Verel, F. Daolio, H. Aguirre, and K. Tanaka, “A feature-based performance analysis in evolutionary multiobjective optimization,” in *Evolutionary Multi-Criterion Optimization*, ser. Lecture Notes in Computer Science, A. Gaspar-Cunha, C. Henggeler Antunes, and C. C. Coello, Eds. Springer International Publishing, 2015, vol. 9019, pp. 95–109. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-15892-1_7

[25] C. C. Coello, G. Lamont, and D. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd ed., ser. Genetic and Evolutionary Computation. Berlin, Heidelberg: Springer, 2007.

[26] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable test problems for evolutionary multiobjective optimization,” in *Evolutionary Multiobjective Optimization*, ser. Advanced Information and Knowledge Processing, A. Abraham, L. Jain, and R. Goldberg, Eds. Springer London, 2005, pp. 105–145.

[27] E. Zitzler, K. Deb, and L. Thiele, “Comparison of multiobjective evolutionary algorithms: Empirical results,” *Evol. Comput.*, no. 2, pp. 173–195, Jun. 2000.

[28] J. Bossek, *smoof: Single and Multi-Objective Optimization Test Functions*, 2015, R package version 1.0.9000. [Online]. Available: https://github.com/jakobbossek/smoof

[29] B. Bischl, M. Lang, J. Richter, J. Bossek, L. Judt, T. Kuehn, E. Studerus, and L. Kotthoff, *mlr: Machine Learning in R*, 2015, R package version 2.5. [Online]. Available: https://github.com/mlr-org/mlr

[30] M. D. McKay, R. J. Beckman, and W. J. Conover, “A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code,” *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[31] W. Härdle and L. Simar, *Applied Multivariate Statistical Analysis*. 3rd edition, Springer, Berlin, Heidelberg, 2012.

[32] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[33] A. Liaw and M. Wiener, “Classification and regression by randomForest,” *R news*, vol. 2, no. 3, pp. 18–22, 2002.