



---

**Automated and Feature-Based  
Problem Characterization and Algorithm Selection  
Through Machine Learning**

---

**Inauguraldissertation**

zum Erlangen des Grades eines  
Doktors der Wirtschaftswissenschaften  
durch die Wirtschaftswissenschaftliche Fakultät  
der Westfälischen Wilhelms-Universität Münster

vorgelegt von

**Pascal Kerschke, M. Sc.**

aus Frankfurt (Oder)

Münster, 14. September 2017

- **Dekanin der Wirtschaftswissenschaftlichen Fakultät:**  
Prof. Dr. Theresia Theurl

- **Betreuerin & Erstgutachterin:**  
Prof. Dr. Heike Trautmann

- **Zweitgutachter:**  
Prof. Dr. Thomas H. W. Bäck  
(LIACS, Leiden University, The Netherlands)

- **Tag der mündlichen Prüfung:**  
13. November 2017

# Acknowledgements

*“We must find time to stop and thank the people who make a difference in our lives.”*

---

John F. Kennedy

Without the strong support of my supervisor, colleagues, collaborators, friends and of course my family, this work would have been impossible. Therefore, I want to thank all of you!

First of all, I sincerely thank my supervisor, [Heike Trautmann](#), who gave me the possibility to jointly start this journey with her in Münster. During the last four years, she always provided me with valuable feedback and inspiring ideas, helped me to tackle several obstacles along the way and introduced me to many inspiring people.

Of course, things are much easier, if you are part of a vivid, friendly, cooperative and encouraging (research) group. Therefore, thanks a lot to my great colleagues [Christian Grimme](#), [Kay F. Hildebrand](#), [Jakob Bossek](#), [Mike Preuß](#), [Matthias Carnein](#), [Dennis Assenmacher](#), [Pelin Aspar](#), [Lena Adam](#), [Ingolf Terveer](#) and [Barbara Berger-Mattes](#).

Furthermore, I was lucky enough to frequently collaborate with people from all over the world: members of the [Group of Computational Intelligence](#) from the TU Dortmund University, the [Leiden Institute of Advanced Computer Science \(LIACS\)](#), the [Cinvestav](#) in Mexico City, the [UBC’s Computer Science Department](#) in Vancouver, [Luis Martí](#) in Rio de Janeiro, as well as the many collaborators from the research networks [COSEAL](#), [ERCIS](#), [OpenML](#), [mlr](#) and [NumBBO](#). Aside from those people, who influenced me somewhat more frequently, I feel also very inspired by several smart minds whom I got to know at the many conferences, workshops, seminars and hackathons that I attended in the last years.

Last – but definitely not least – I want to thank my entire family, my beloved wife Laura, and of course all of my friends for their great support, patience and – whenever necessary – positive distractions from work. This way, I always had the possibility to completely recharge myself.

**THANK YOU!**

# Abstract

*“The universe is governed by science. But science tells us that we can’t solve the equations, directly in the abstract.”*

---

Stephen Hawking

One of the many prejudices about Germans is that we are always very structured. However, I would actually say that in today’s world, many things, and especially processes – not only in Germany – are structured. An advantage of having such structures is that we are able to model them. Even in cases, for which one can not come up with an exact mathematical model, one still can measure the influence of the input on the corresponding output. In such a case, we are talking of *black-box problems*.

Interestingly, people tend to optimize everything: we minimize the travel time between home and office, look for the shortest queue in the supermarket, search for the best value for money hotel for the next vacation, or increase the productivity of our machines at work. All of these ‘decisions’ are examples for *optimization problems*. Unfortunately, people tend to make rather poor decisions when optimizing their problems, because most of these ‘optimal solutions’ – at least people often believe that they are optimal – are either based on numerous trial-and-error experiments or (frankly said) ‘gut-decisions’. Instead of these manual approaches, one could use some computational power to run an optimization algorithm. However, as there exists a plethora of optimization algorithms (for each of the different applications) one has to make a sophisticated guess on which of the available algorithms is the best one for the application at hand. This decision is known as the *algorithm selection problem*.

We nowadays have the computational power (and also the necessary tools) for making those selections on the one hand more sophisticated based on problem-specific measures, and on the other hand compute these measures and thus, also the algorithm selections, in an automated fashion. As a result, our machines now are in the position to adjust to the respective optimization problem at hand by selecting and running a promising optimization algorithm, which in turn hopefully provides us with a satisfying – ideally the best possible – solution.

Within this cumulative dissertation, I will present (i) a new set of these aforementioned automatically computable features (which in my case, are able to extract useful information on the structure of single-objective continuous optimization problems at a very low computational budget), (ii) experimental studies that have confirmed the applicability, as well as the good performance of automated and feature-based algorithm selection models, and (iii) several frameworks, which facilitate the research in the fields of single-objective continuous optimization, algorithm selection and machine learning.

# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Benchmarking . . . . .	3
1.2 Exploratory Landscape Analysis . . . . .	4
1.3 Algorithm Selection . . . . .	5
<b>2 Characterizing the Global Structure of Continuous Black-Box Problems</b>	<b>7</b>
2.1 Contributed Material . . . . .	7
2.2 Cell Mapping Techniques for Exploratory Landscape Analysis . . . . .	7
2.3 Detecting Funnel Structures by Means of Exploratory Landscape Analysis . . . . .	9
2.4 Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models . . . . .	11
<b>3 Flacco – A Toolbox for Exploratory Landscape Analysis with R</b>	<b>13</b>
3.1 Contributed Material . . . . .	13
3.2 The R-Package FLACCO for Exploratory Landscape Analysis with Applications to Multi-Objective Optimization Problems . . . . .	13
3.3 flaccogui: Exploratory Landscape Analysis for Everyone . . . . .	15
3.4 Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package flacco . . . . .	17
<b>4 Feature-Based Algorithm Selection from Optimizer Portfolios</b>	<b>18</b>
4.1 Contributed Material . . . . .	18
4.2 Improving the State of the Art in Inexact TSP Solving using Per-Instance Algorithm Selection . . . . .	18
4.3 Leveraging TSP Solver Complementarity through Machine Learning . . . . .	20
4.4 Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning . . . . .	22
<b>5 Platforms for Collaborative Research on Algorithm Selection and Machine Learning</b>	<b>26</b>
5.1 Contributed Material . . . . .	26
5.2 ASlib: A Benchmark Library for Algorithm Selection . . . . .	26
5.3 OpenML: An R Package to Connect to the Machine Learning Platform OpenML	28
<b>6 Summary and Outlook</b>	<b>30</b>

## Appendix: Contributed Publications

<b>A</b>	<b>Characterizing the Global Structure of Continuous Black-Box Problems</b>	<b>42</b>
A.1	Cell Mapping Techniques for Exploratory Landscape Analysis . . . . .	42
A.2	Detecting Funnel Structures By Means of Exploratory Landscape Analysis . . . .	44
A.3	Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models . . . . .	45
<b>B</b>	<b>Flacco – A Toolbox for Exploratory Landscape Analysis with R</b>	<b>46</b>
B.1	The R-Package FLACCO for Exploratory Landscape Analysis with Applications to Multi-Objective Optimization Problems . . . . .	46
B.2	flaccogui: Exploratory Landscape Analysis for Everyone . . . . .	47
B.3	Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package flacco . . . . .	48
<b>C</b>	<b>Feature-Based Algorithm Selection from Optimizer Portfolios</b>	<b>49</b>
C.1	Improving the State of the Art in Inexact TSP Solving Using Per-Instance Al- gorithm Selection . . . . .	49
C.2	Leveraging TSP Solver Complementarity through Machine Learning . . . . .	50
C.3	Automated Algorithm Selection on Continuous Black-Box Problems By Com- bining Exploratory Landscape Analysis and Machine Learning . . . . .	51
<b>D</b>	<b>Platforms for Collaborative Research on Algorithm Selection and Machine Learning</b>	<b>52</b>
D.1	ASlib: A Benchmark Library for Algorithm Selection . . . . .	52
D.2	OpenML: An R Package to Connect to the Machine Learning Platform OpenML	54

# Chapter 1

## Introduction

*“It is not knowledge, but the act of learning, not possession but the act of getting there, which grants the greatest enjoyment.”*

---

Carl Friedrich Gauß

Optimization problems can be found in many different real-world applications. Imagine a mass production of printed circuit boards (PCBs), where for each board with the same arrangement of holes, the roboter follows the same drilling path. Obviously, having a better, i.e., faster, tour for the roboter’s path on the respective board could substantially increase the productivity of the factory. Or think of the print media sector, where a company generates thousands of newspapers, leaflets and magazines per day. Here, many parameters influence the printing performance, which in turn directly affects the company’s profit: the speed of the printer’s rolls, the amount of ink being printed on the paper, the temperature and power of the drying fan, etc.

In either of these scenarios one chases the goal of finding the optimal setting for a given problem. And while domain knowledge can be helpful for finding at least satisfying solutions, the majority of real-world problems is usually too complex for human beings. Even worse, most of these problems are so-called *black-box problems*, i.e., the exact relationship between the controllable inputs and the corresponding outputs is unknown. However, for most of these applications, there exists a variety of optimization algorithms that often find better solutions than the ones that are based on the domain-expert’s gut decisions. Unfortunately, according to the “*no free lunch theorems*” by [Wolpert and Macready \(1997\)](#), there is no single optimization algorithm that is superior to all the other ones for every single problem. In consequence, one has to decide for each problem separately or at least for each group of similar problems, which optimization algorithm one should use.

Of course, one could execute multiple optimization algorithms and afterwards pick the best solution found by any of them. However, while this may be a reasonable approach for scenarios that can be optimized offline – e.g., finding the optimal PCB drilling path can be done via computer simulations – in many real-world problems, each evaluation of a unique parameter configuration can be very costly. For instance, in the print media example from above, a single evaluation stands for a specific combination of rolling speed, fan temperature and amount of ink. Obviously, it is not affordable to try hundreds of configurations (or even more). Therefore, one wants to avoid running several optimization algorithms and instead use only one of them.

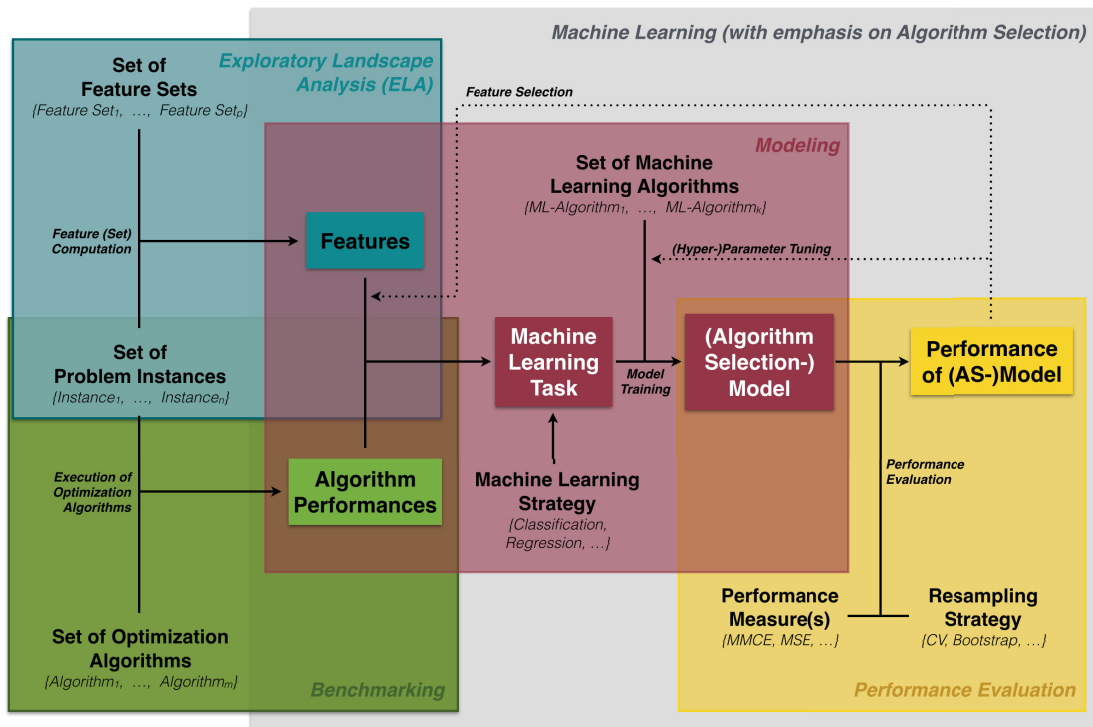


Figure 1.1: Schematic overview of the interlinks between *Exploratory Landscape Analysis* (ELA; blue area in the top left), *Benchmarking* the optimization algorithms (green area in the bottom left), the actual *Modeling* process of the machine learning algorithms (red area in the center) and their *Performance Evaluation* (yellow area in the bottom right). The grey box in the background displays which of the aforementioned topics belong to the field of *Machine Learning* in general.

However, even the evaluations for a single optimizer could already be too expensive – it simply might require too many evaluations until it finds a satisfying solution. Consequently, despite the plethora of optimization algorithms, one remains with the task of solving the following problem:

*How do I know in advance, which optimization algorithm is the best one for my application?*

Within this thesis, I will present a possible solution to this task by means of automated and feature-based algorithm selection. Admittedly, this approach does *not guarantee* to always find the *best* optimization algorithm for every single instance of a problem<sup>1</sup>, but due to the integration of problem-specific features, we usually find competitive algorithms.

A schematic overview of the principle of automated feature-based algorithm selection is given in Figure 1.1. This scheme also shows the links between its key elements, which are the extraction of problem-specific features (highlighted by a blue box), benchmarking a portfolio of optimization algorithms (green box) and training the algorithm selector itself. Note that the latter is distinguished into the actual modeling phase (red area) and its performance assessment (yellow area). Furthermore, the scheme also shows how these elements are embedded within the more general research field of *machine learning* (grey box).

Within the following sections of this introductory chapter, i.e., Sections 1.1 to 1.3, the three aforementioned key elements will be introduced. Afterwards our contributions to each of these

<sup>1</sup>In the example of PCBs, an *instance* would be a different arrangement of the holes, whereas finding the fastest drilling path would be the general optimization *problem*.



areas are described in more detail in the succeeding chapters. More precisely, in Chapter 2 our advances w.r.t. the characterization of the global structure of continuous optimization problems will be presented. Then, in Chapter 3, a toolbox, which enables the computation of numerous problem-specific features for continuous optimization problems, is being introduced. Chapter 4 summarizes several experimental studies, in which we successfully showed the applicability of our approach in two different domains: the *Travelling Salesperson Problem (TSP)* and the single-objective continuous (black-box) optimization problem. As machine learning in general, and algorithm selection in particular, strongly rely on sound experiments, we also contributed to two platforms, which facilitate the exchange of experimental studies and hence, the collaboration among researchers within the respective research domains. Further details on these two platforms are given in Chapter 5. At last, Chapter 6 concludes this thesis with a summary of the presented work and an outlook on promising future extensions.

In order to facilitate the classification of our contributions to the respective areas from Figure 1.1, each of them is listed at the beginning of the respective chapter and marked with boxes that are colored according to the four categories feature computation (■), benchmarking (■), modeling (■) and/or performance assessment (■).

## 1.1 Benchmarking

During the last years, my research projects usually were related to one of the following two types of optimization problems: (1) single-objective continuous optimization problems (e.g., [Boyd and Vandenberghe, 2004](#)), or (2) the *Travelling Salesperson Problem (TSP)* (e.g., [Mersmann et al., 2013](#)). The former one can be defined as follows:

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}) \\ & \text{s.t.} && g_i(\mathbf{x}) \leq b_i, \quad i = 1, \dots, k. \end{aligned}$$

Here,  $\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$  is a  $n$ -dimensional vector from the continuous *search* or *decision space*  $\mathbb{R}^n$ ,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is the single-objective function that is supposed to be minimized<sup>2</sup> and the functions  $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i = 1, \dots, k$ , are the  $k$  inequality constraints with the respective boundaries  $b_i \in \mathbb{R}$ . In addition to these constraints, the optimal value  $\mathbf{x}_{opt} \in \mathbb{R}^n$  also satisfies the condition  $f(\mathbf{x}_{opt}) \leq f(\mathbf{x})$  for all  $\mathbf{x} \in \mathbb{R}^n$ .

While the aforementioned optimization problem tries to find the optimal value in a *continuous* decision space, the (Euclidean) Travelling Salesperson Problem is one of the most well-known *combinatorial* optimization problems. Given a set  $\mathcal{C} := \{c_1, \dots, c_n\}$  of  $n$  so-called *cities* with non-negative distances  $d(c_i, c_j) \geq 0$  between all pairs of cities  $c_i$  and  $c_j$ , the goal is to find the shortest tour, which visits each of the  $n$  cities exactly once and afterwards returns to its origin.

For either of these types of optimization problems, there exist several benchmarks. As displayed by the green area in the bottom left of Figure 1.1, the general idea of benchmarking is quite simple: given a set of problem instances, one executes a set of optimization algorithms on each of the instances. Their performances can then be used in two ways: comparing (1) the optimizers against each other and thereby get a better understanding of their strengths and weaknesses, or (2) the performances across the different (2) problem instances and thereby distinguish, for

<sup>2</sup>A maximization problem  $m(\mathbf{x})$  can easily be transformed into a minimization problem via  $f(\mathbf{x}) := -m(\mathbf{x})$ .

instance, the easy from the difficult problems. Such benchmark analyses are important as there can not exist a single algorithm, which is superior to all other algorithms across all problem classes (Wolpert and Macready, 1997).

In the context of single-objective continuous optimization, there already exist such benchmarks, e.g., the COCO platform (Hansen et al., 2016), which summarizes the performances of more than a hundred optimizers across the so-called *Black-Box Optimization Benchmark* (BBOB, Hansen et al., 2009a,b). The latter consists of 24 problem classes, which Hansen et al. (2009b) divided into five groups according to their separability, conditioning, multimodality and global structure. For each of the 24 functions, one can generate multiple variants by rotating, shifting or scaling the respective *original* instance. Each of the transformed versions – as well as many other single- and multi-objective optimization problems – can for example be generated using the R-package `smoof` (Bossek, 2016).

In correspondence with the diversity of optimization problems, there also exist numerous optimization algorithms. In the single-objective continuous optimization domain, the most successful ones usually are variants of *Quasi-Newton methods* (e.g., BFGS; Broyden, 1970), *Covariance Matrix Adaption Evolution Strategies* (CMA-ES; Hansen, 2006), *Simulated Annealing* (SA; Tsallis and Stariolo, 1996), *Genetic Optimization Using Derivatives* (Genoud; Mebane Jr. and Sekhon, 2011), *Multi-Level Methods* (e.g., Rinnooy Kan and Timmer, 1987), *Differential Evolution* (DE; Storn and Price, 1997) and *Particle Swarm* (PSO; Eberhart and Kennedy, 1995) optimization algorithms. However, for many of them, there exists a plethora of variants. For instance, van Rijn et al. (2016) have shown that (considering the different modules that are affecting the solver) there are more than 4,000 variants of the CMA-ES.

Obviously, comparing all of them one-by-one across all problems is simply impossible and also completely unnecessary as one usually requires at most a handful of complementary optimizers, i.e., an optimizer portfolio, for a given set of problems. However, in order to make a sophisticated guess on which solvers might work well on the respective problem(s), it is important to have further information on the respective problems *before* actually optimizing them.

## 1.2 Exploratory Landscape Analysis

The characterization of problem landscapes is very important in order to group the problems according to their similarities. As Mersmann et al. (2010) have shown, such knowledge can be very helpful, because algorithms often perform well on entire classes of (similar) problems rather than just on single instances. That is, if two problem instances  $A$  and  $B$  are similar, it is very likely that an algorithm, which performs well on instance  $A$ , will also perform well on instance  $B$  (and vice versa).

In their work, Mersmann et al. (2010) introduced eight characteristics – which they called *high-level* properties – that could be useful for distinguishing single-objective continuous optimization problems from each other. These properties are the degree of *multimodality*, the underlying *global structure*, the *separability*, the *variable scaling*, the *homogeneity of the search space* and *basin sizes*, the *contrast of global to local optima* and whether the function’s landscapes possesses *plateaus*. Unfortunately, these properties come with the drawback that they require expert knowledge – which is equivalent to having someone (i.e., the expert), who *manually* assigns the

respective attributes for each property and each (new) problem instance.

This issue has been addressed with the introduction of the *Exploratory Landscape Analysis* (ELA) and its so-called *low-level* features (Mersmann et al., 2011). These measures have the advantage, that they are automatically computable. Hence, one has the means for computing sets of feature values in an automated fashion – based on a sample of observations from the given problem instance (see **blue box** in the top left of Figure 1.1). Furthermore, by using these numerical features, one reduces the impact of misclassifications (made by the expert) in case of fuzzy decisions such as the exact degree of a problem’s multimodality (low, moderate or high). Admittedly, Mersmann et al. (2010, 2011) have neither been the first nor the last ones to develop landscape features (as shown by Muñoz Acosta et al., 2015b). However, in their succeeding work, Bischl et al. (2012) showed the importance of automatically computable features for the recently revived research field of algorithm selection (due to advances in the field of machine learning), by using the landscape features for *automated* algorithm selection.

Similar advances were also made in other optimization domains. For instance, Kotthoff et al. (2015) and recently, Kerschke et al. (2017) used problem-specific features to improve the state of the art in (inexact) solving of the *Traveling Salesperson Problem* (TSP) by means of feature-based algorithm selection from a portfolio of heuristic TSP solvers.

It is important to notice that, independent of the research domain, most of these features do not provide intuitively understandable numbers. Therefore, we strongly recommend not to interpret them on their own. Moreover, some of them are stochastic and hence should be evaluated multiple times on an instance and afterwards be aggregated in a reasonable manner. Nevertheless, they definitely provide information that can be of great importance to scientific models, such as machine learning algorithms in general or algorithm selectors in particular.

### 1.3 Algorithm Selection

Although the first formal definition of the *Algorithm Selection Problem* (ASP; Rice, 1976) is already over 40 years old, its general idea still remains the same. Given a set (often denoted *portfolio*) of optimization algorithms  $\mathcal{A}$  and a set of problem instances  $\mathcal{I}$ , we want to find a model – i.e., the *algorithm selector* – which for a new problem instance  $I$  chooses the best-performing algorithm  $A \in \mathcal{A}$  out of the portfolio of optimizers  $\mathcal{A}$ .

However, as already mentioned in Section 1.2, in recent years algorithm selection has strongly benefitted from the advances in *automated* problem characterization via (landscape) features, which – in combination with iteratively enhanced machine learning algorithms – enable automated algorithm selection from a set of various optimization algorithms. Ultimately, these advances could lead to a general-purpose hybrid algorithm, which uses the problem-specific features to automatically adapt itself to the respective problem’s (part of the) landscape, e.g., by independently switching between the algorithms from the considered portfolio while optimizing the given problem.

One essential aspect for tackling the algorithm selection problem is to train different machine learning algorithms as possible algorithm selection models (highlighted by the **red area** in the center of Figure 1.1). However, an accurate evaluation of its performance (visualized by the **yellow area** in the bottom right of Figure 1.1), which ideally should either be assessed via a

separate test set or using a resampling strategy (e.g., crossvalidation or bootstrap sampling), is at least of similar importance. Especially when trying to improve a trained algorithm selector by means of a (sophisticated) feature selection and/or tuning of the algorithm's hyperparameters, its performance provides a very important feedback mechanism for the quality of the generated models.

By all means one has to keep in mind that the generated algorithm selection model only generalizes to unseen instances, which are 'similar enough' to the set of instances that has been used for training the algorithm selector. Or, the other way around, one should ideally use a very representative subset of instances from the space of possible instances of interest.

## Chapter 2

# Characterizing the Global Structure of Continuous Black-Box Problems

*“Measure what is measurable, and make measurable what is not so.”*

---

Galileo Galilei

### 2.1 Contributed Material

- Kerschke, P., Preuss, M., Hernández, C., Schütze, O., Sun, J.-Q., Grimme, C., Rudolph, G., Bischl, B. & Trautmann, H. (2014). *Cell Mapping Techniques for Exploratory Landscape Analysis*. In: *EVOLVE – A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, pages 115 – 131. ■ (see Appendix A.1)
- Kerschke, P., Preuss, M., Wessing, S. & Trautmann, H. (2015). *Detecting Funnel Structures by Means of Exploratory Landscape Analysis*. In: *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 265 – 272. ■ (see Appendix A.2)
- Kerschke, P., Preuss, M., Wessing, S. & Trautmann, H. (2016). *Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models*. In: *Proceedings of the 18th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 229 – 236. ■ (see Appendix A.3)

### 2.2 Cell Mapping Techniques for Exploratory Landscape Analysis

In Kerschke et al. (2014), we continued the work of Mersmann et al. (2011). That is, we designed two new groups of (low-level) features that characterize certain aspects of a problem’s landscape, such as the *basin size homogeneity* or its *global structure*, by means of automatically computable measures. In contrast to the ‘classical’ ELA features from Mersmann et al. (2011), which directly target the continuous landscapes, we discretized the decision space into a grid of cells *prior* to

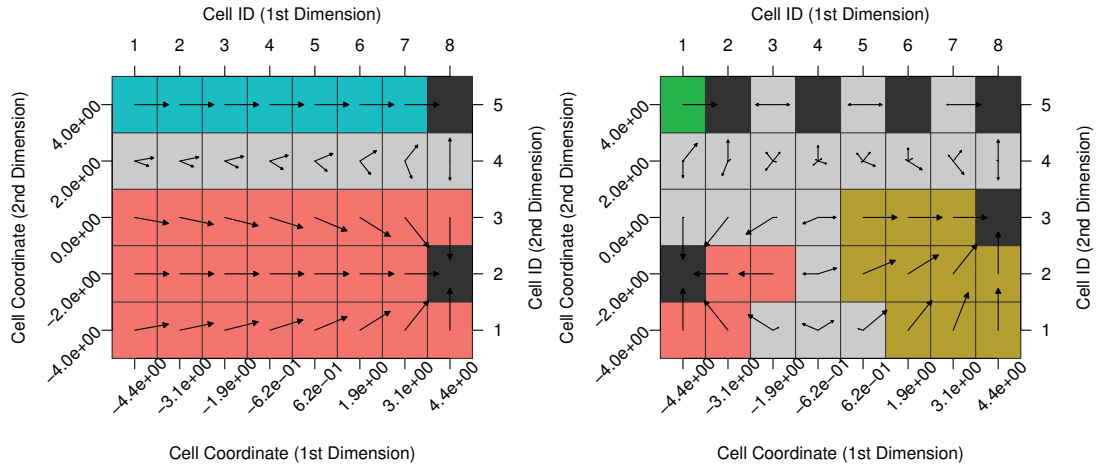


Figure 2.1: Visualizations of the *general cell mapping* idea, exemplarily shown for the *Freudenstein and Roth Function*<sup>3</sup>, displayed for two different cell representation approaches: minimum (left) and average (right). The black and grey boxes show the absorbing and uncertain cells, respectively, whereas the colored cells represent the different basins of attraction.

computing our new landscape features. More precisely, we defined the number of (equidistant) cells per dimension, sampled points random uniformly from the decision space and then assigned each of them to its respective nearest cell.

The first group of proposed measures – denoted “general cell mapping” (GCM) features – use the idea of the so-called *cell-mapping method* (Bursal and Hsu, 1989; Hsu, 1987), whereas the remaining group basically computes feature-like values for each of the cells and aggregates the resulting numbers afterwards.

For the computation of the GCM features, each cell is represented by exactly one observation from the initial design. Within Kerschke et al. (2014), we proposed three different approaches for assigning a *representative* objective value per cell: taking (a) the best objective value (of all samples from the respective cell), (b) the average of the objective values, or (c) the objective value of the observation that is located closest to the respective cell center. The cells were then considered to be absorbing Markov chains, which use the height differences between neighboring cells as *transition ‘probabilities’* for moving from one cell to its neighboring cells. As exemplarily shown within Figure 2.1 for the *Freudenstein and Roth Function*<sup>3</sup>, these probabilities allowed to classify the cells into *absorbing* (depicted by black boxes), *uncertain* (grey boxes) and *transient* cells (colored boxes). The absorbing or *attractor* cells indicate local optima, the colored areas represent the corresponding *basins of attraction* – cells that have the same color belong to the same local optimum – and the uncertain cells can be understood as ridges between at least two basins. That information was used for computing numerous landscape features, such as the number of attractors, the ratio of uncertain cells, and multiple aggregations of the basin sizes.

In addition to the GCM features, we proposed three further feature sets. The *angle features* summarize information on the locations of each cell’s best and worst observation. More precisely, for each cell the angle between the best observation, the cell center and the worst observation,

<sup>3</sup>The *Freudenstein and Roth Function* (Rao, 2009) is defined as  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$  with

$$f(x_1, x_2) = (-13 + x_1 + [(5 - x_2) \cdot x_2 - 2] \cdot x_2)^2 + (-29 + x_1 + [(x_2 + 1) \cdot x_2 - 14] \cdot x_2)^2.$$

as well as the distances from the cell center to the two extreme observations, are measured. The rationale behind this approach is that for simpler problems – e.g., landscapes with a rather low multi-modality and/or a clear trend towards the global optimum – the extreme values often will be located in opposite parts of the cell, whereas the extreme values within more complex problems usually do not show such patterns. The *gradient homogeneity features* also try to capture trends within the cells, but use the information of all sampled points rather than just the two extremes per cell. Our third feature set, the *(cell mapping) convexity features*, were inspired by the convexity features of Mersmann et al., but in contrast to their approach, our features completely rely on the information of the cells and thus, do not require any additional function evaluations.

Within Kerschke et al. (2014), we compared the proposed features to three of the ‘classical’ ELA feature sets – the levelset, meta-model and y-distribution features, i.e., the ones that do not require additional function evaluations – by classifying the high-level properties, introduced by Mersmann et al. (2011), on a set of benchmark problems. More precisely, we considered the two-dimensional versions of all 24 BBOB problems (Hansen et al., 2009b) with ten instances each and tried to predict the correct class label for each property either by using only the cell mapping features, only the ELA features or both groups combined. Our new features led to lower misclassification errors when predicting the *global structure* and *multimodality* properties (compared to the ELA features) and the combination of both feature sets resulted in the best performances on five of the seven considered properties.

In conclusion, we introduced multiple new feature sets that helped to improve the existing feature sets, especially w.r.t. the problem’s global structure, without performing any additional function evaluations. Nevertheless, we are aware of the fact that our proposed features are only useful in case of low-dimensional problems as the discretization into cells is accompanied by an exponential growth of the initial design (in order to provide at least a few points within each of the cells).

## 2.3 Detecting Funnel Structures by Means of Exploratory Landscape Analysis

In Kerschke et al. (2015), we designed five new landscape features, which – in combination with some of the existing ELA features – improve the detection of underlying funnel structures. Here, a “funnel” is defined as “a landscape, whose local optima are aligned close to each other such that they pile up to a mountain (in case of maximization problems) or an ‘upside-down version of a mountain’ (minimization)”. The rationale behind distinguishing such landscapes from each other is the idea that each of these topologies requires a different group of optimization algorithms: While global optimization algorithms are better in exploiting the global structures of funnel-shaped landscapes and thus, are more promising when searching for the global optimum of such structured problems, multimodal optimizers should perform better on non-funnel problems. Therefore, predicting the correct “funnel category” enables to efficiently construct a suitable algorithm portfolio for subsequent algorithm selection studies.

Our proposed features – denoted *Nearest Better Clustering (NBC) features* – aggregate information of two sets of distances: the distances of all sampled points to (a) their nearest neighbors

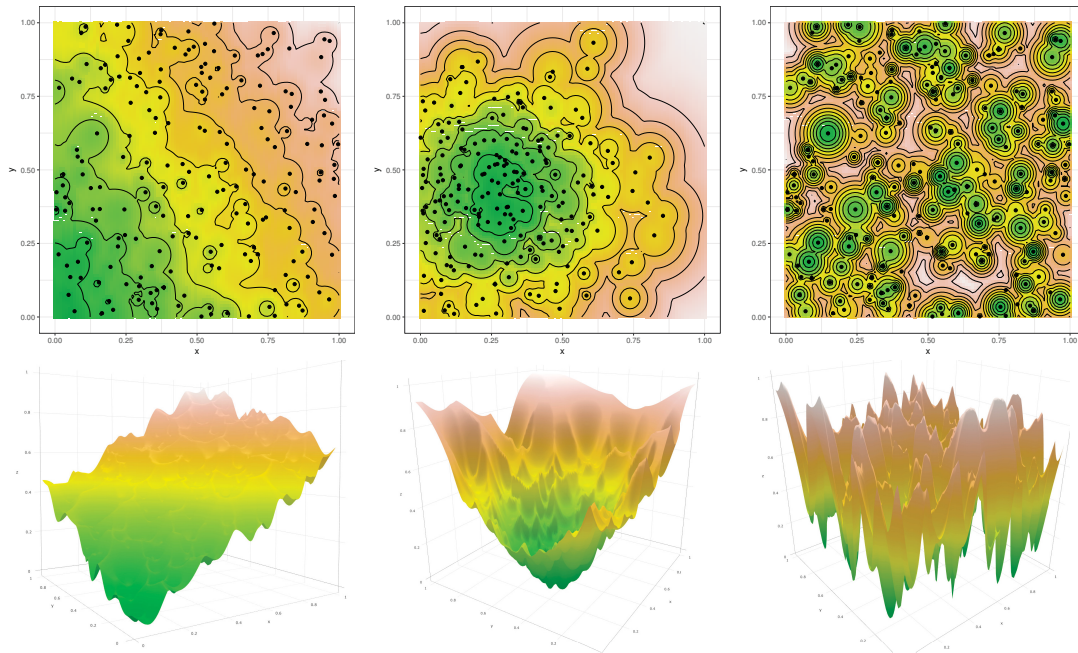


Figure 2.2: Examples of two-dimensional mixed-sphere problems with an underlying linear, funnel or “random” global structure (left to right), and visualized as contour (top) or surface plots (bottom). The problems consist of 200 local optima each (indicated by black dots within the contour plots) and were created using a test problem generator that combined multiple (unimodal) sphere functions into a multimodal landscape (Wessing et al., 2013).

and (b) their nearest better neighbors. While the former is measured straightforward, the latter measures (per observation, i.e., for each sampled point) the distance to its closest neighbor with a better objective value. The first two NBC features are ratios of the standard deviations (or arithmetic means) of the two distance sets and the third one is the correlation between the two sets. In case of non-funnel landscapes the nearest better neighbors are (more) widely spread and thus, the arithmetic mean and/or standard deviation of the nearest better neighbor distances are higher than the ones from the nearest neighbor distance set. Analogously, the (absolute) correlation between the two distance sets should be much lower for non-funnel landscapes than for funnel problems.

We trained various machine learning models – decision trees, random forests, support vector machines and nearest neighbor methods – as possible “funnel detectors”, using a self-made benchmark consisting of 6 000 mixed-sphere problems. Each of them has a specific underlying global structure (linear, funnel or “random”) and a varying number of peaks. Figure 2.2 provides an example for each of the aforementioned landscape categories. Due to the fact that linear problems can be seen as a special case of funnel problems, we considered the funnel detection problem to be binary: funnel (and linear) vs. non-funnel. Each of the machine learning algorithms was trained using the three “cheap” ELA feature sets, as well as our five NBC features. In order to find well-performing funnel detectors, we reduced the number of features by a greedy feature selection strategy and evaluated our models with a nested resampling strategy.

The best-performing version per machine learning algorithm was assessed on two external validation sets: the BBOB problems and a collection of *Disc Packing* problems (Addis et al., 2008). Surprisingly, our most accurate funnel detector was a classification tree, which uses a



small, but plausible group of features: two meta-model (the adjusted model fit, i.e.,  $R_{adj}^2$ , of a quadratic model with interactions, and the intercept of a simple linear model) and two NBC features (the ratio of the standard deviations, and the correlation between the distance sets). This rather simple – and hence, well interpretable – model on average misclassified only 10% of the training data (assessed using a nested 10-fold crossvalidation), 3% of the BBOB problems and also supported the thesis of [Addis et al. \(2008\)](#), who claim that the landscapes of (larger) disc packing problems are funnels.

## 2.4 Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models

After having shown in [Kerschke et al. \(2015\)](#) that we are in general able to detect funnel structures of (black-box) optimization problems, we enhanced our methodology in [Kerschke et al. \(2016a\)](#) allowing us to be competitive when having lower budgets (i.e., less function evaluations) as well. With our new approach, we were able to reduce the size of the initial design – and hence the costs for the computation of our landscape features – by factor ten. That is, instead of  $500 \times d$  function evaluations, where  $d$  is the number of dimensions of the underlying optimization problem, we only require  $50 \times d$  evaluations. These improvements are extremely profitable as they approximate the size of an evolutionary algorithm’s (EA) *initial* population. Considering that an EA has to evaluate its starting population anyway and that it could use our initial design instead, the costs for the computation of the landscape features would be negligible. Another advantage of such a budget decrease is the increased applicability of the ELA approach to real-world problems: (additional) evaluations of such problems often are highly costly and hence, a smaller budget implies much lower costs.

Within our work, the reduction of the budget basically has been achieved by three changes within our experimental setup: (1) generating the training instances with an improved problem generator, (2) using a more sophisticated strategy for sampling the points of the initial design, and (3) selecting a better-performing subset of landscape features.

A weakness of our previously used problem generator was the positioning of the funnel’s global optimum. More precisely, it always located the global optimum within the center of the decision space. Consequently, our funnel detectors had difficulties spotting funnels that were located close to the boundaries of the decision space. However, this issue has been solved with the usage of the *Multiple Peaks Model* generator (MPM2, [Wessing, 2015](#)), which is available in python (within `optproblems`, [Wessing, 2016](#)) and R (`smoof`, [Bossek, 2016](#)). Further *minor* performance improvements – especially w.r.t. detecting non-funnel landscapes – have been achieved by using a *latin hypercube sample* (LHS, [Beachkofski and Grandhi, 2002](#)) instead of a random uniform sampling strategy for constructing the initial design. The final performance improvements have been made by executing a brute force (i.e., exhaustive) feature selection on a pre-selected group of promising landscape features.

In the end, a random forest, consisting of 500 trees and using two NBC features, five meta model features and the dimensionality of the problem itself, was the best performing funnel detector. The results have been assessed using a 10-fold cross validation on our training data (with an accuracy of more than 98%), as well as on two external validation sets: the BBOB

problems (approx. 92% accuracy) and a collection of non-funnel problems from the *CEC 2013 niching competition* (100% accuracy).

All in all, we were able to show that landscape analysis is a powerful methodology for distinguishing optimization problems from each other – at least w.r.t. their global structure in general and the existence of an underlying funnel structure in particular – while only spending a small amount of function evaluations ( $50 \times d$  with  $d$  being the problem’s search space dimensionality) on the computation of the corresponding landscape features.

## Chapter 3

# Flacco – A Toolbox for Exploratory Landscape Analysis with R

*“We have to stop optimizing for programmers and start optimizing for users.”*

---

Jeff Atwood

### 3.1 Contributed Material

- Kerschke, P. & Trautmann, H. (2016). *The R-Package FLACCO for Exploratory Landscape Analysis with Applications to Multi-Objective Optimization Problems*. In: IEEE Congress on Evolutionary Computation (CEC), pages 5262 – 5269. ■  
(see Appendix B.1)
- Hanster, C. & Kerschke, P. (2017). *flaccogui: Exploratory Landscape Analysis for Everyone*. In: Proceedings of the 19th Annual Conference on Genetic and Evolutionary Computation (GECCO) Companion, pages 1215 – 1222. ■ (see Appendix B.2)
- Kerschke, P. (under review). *Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package flacco*. ■  
(see Appendix B.3)

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

Inspired by the multitude of perspective use cases for Exploratory Landscape Analysis, several research groups worldwide have designed new landscape features. In general, this progress should be considered to be very helpful as the different scientific backgrounds of the developers resulted in numerous (often complementary) features, which in turn usually characterized different aspects of a problem’s landscape. However, although a wide variety of feature sets

existed (see, e.g., Muñoz Acosta et al., 2015b, for an overview on some of them), in most of the cases researchers considered only small subsets due to a simple and plausible reason: they did not possess the source code. More precisely, each group implemented its features in its programmer’s favorite language, e.g., `Matlab` (MATLAB, 2013), `python` (VanRossum and The Python Development Team, 2015) or `R` (R Core Team, 2017), and – if at all – published the corresponding code in file bundles or separate packages. In consequence, people who wanted to use and/or compare features that were developed by different groups had to cope with different programming languages, interfaces, etc.

In order to overcome these obstacles, we created `flacco` (Kerschke, 2017), an R-package for feature-based landscape analysis of continuous and constrained optimization problems. It combines more than 300 features (originating from 17 feature sets), as well as several visualization techniques, which should help to make (some of) the feature sets more comprehensible and thereby provide a better understanding of the given optimization problem. Furthermore, we automatically track and report the costs (i.e., the number of function evaluations, as well as the running time in seconds) for computing each of the feature sets and thereby enable a more fair comparison of them, especially when using them for automated algorithm selection.

Given that R-packages are open source and that `flacco`’s development version is hosted publicly accessible on GitHub<sup>4</sup> (including an issue tracker for feedback, bug reports, etc.), the entire source code is available to everybody in the world. Hosting the package on GitHub also makes it easier for other R-developers to extend our framework, e.g., by integrating new feature sets, and thereby making it *the tool* for Exploratory Landscape Analysis. Shortly after launching the package, we also published an online tutorial<sup>5</sup>, which provides further information on the implemented feature sets and visualization techniques, as well as a quick start section that should facilitate the first steps with `flacco`.

In Kerschke and Trautmann (2016), we introduced the R-package to the evolutionary computation community and exemplarily showed how to use it on the basis of a multi-objective case study. To the best of our knowledge, our experiment was the first one to combine landscape features that originate from such a variety of research groups. And while landscape features so far were only used for characterizing single-objective problems, we tested their applicability to multi-objective problems. For this purpose, we combined (three-dimensional) instances from two well-known multi-objective optimization benchmarks, namely DTLZ (Deb et al., 2005) and ZDT (Zitzler et al., 2000), by ratios of objective-wise computed landscape features. That is, we computed each feature on both objectives of an instance and afterwards aggregated them by dividing the value belonging to objective 1 by the respective value of objective 2.

As shown within Figure 3.1, the similarities and dissimilarities between these ‘multi-objective landscape features’ suggested five to six clusters: two bigger groups – consisting of (1) DTLZ2, DTLZ3, DTLZ5 and DTLZ6, and (2) DTLZ7, ZDT1, ZDT2 and ZDT3, respectively – and three to four groups consisting of a single instance each (DTLZ1, DTLZ4, ZDT4 and ZDT6). Based on the right image, which projects the data into the hyperplane spanned by the first two principal components, DTLZ1 might be similar to the problems of the green cluster. With the exception of DTLZ7 one can in general detect a rather high ‘intra-benchmark’ similarity and ‘between-benchmark’ dissimilarity. Thus, although the objective-wise aggregation of the features was

<sup>4</sup><https://github.com/kerschke/flacco>

<sup>5</sup><http://kerschke.github.io/flacco-tutorial/>

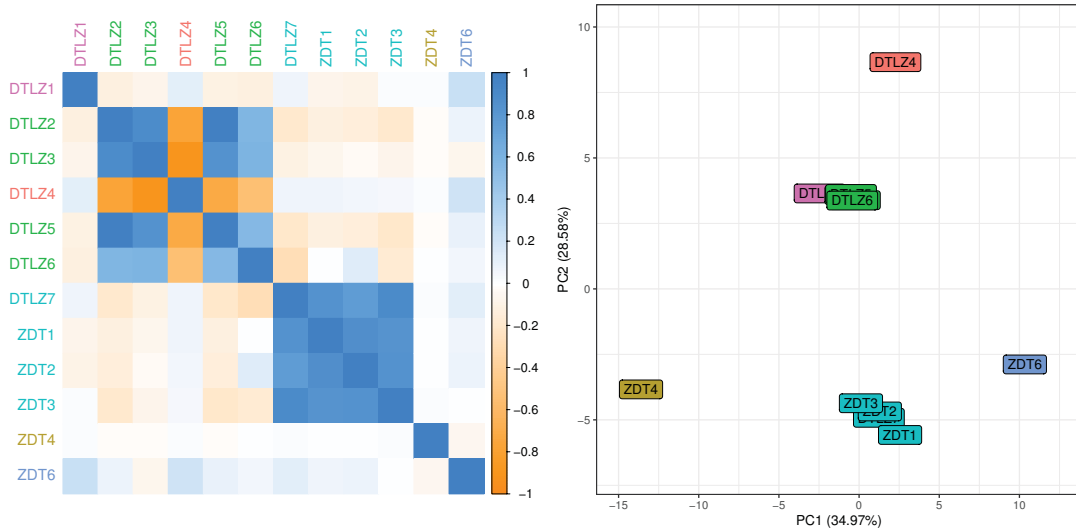


Figure 3.1: Similarities and dissimilarities among the multi-objective optimization problems, according to our ‘multi-objective features’. The heatmap (left) visualizes the correlations between the 12 instances: blue (orange) boxes indicate positive (negative) correlations and their intensities represent the respective magnitude. The plot on the right shows the found clusters based on the first two principal components (which explain roughly 63.6% of the variance).

admittedly quite arbitrary, our multi-objective landscape features were able to find patterns within the benchmarks. Consequently, if one has to rely on artificial test problems, we strongly recommend to use test problems from multiple benchmarks in order to consider a higher variety of problems and thereby avoid a bias towards certain benchmark-specific patterns.

### 3.3 flaccogui: Exploratory Landscape Analysis for Everyone

Although `flacco` provides numerous tools and features for performing Exploratory Landscape Analysis, its usability comes with one essential drawback: it is an R-package and hence will only be used by people who are familiar with that programming language. In [Hanster and Kerschke \(2017\)](#), we therefore introduced a graphical user interface (GUI), which helps to overcome this obstacle. It can either be executed from within R itself (as part of our `flacco`-package) or on an entirely platform-independent web-hosted application<sup>6</sup>. Both versions are identical w.r.t. their appearance and functionalities, but while the web application is hosted on a server and hence, can be accessed from any device (as long as it has access to the internet), the built-in version runs on the user’s local machine and thus neither requires server nor internet access.

The GUI was created using the R-package `shiny` ([Chang et al., 2016](#)), which enables its users to build R-based web applications without any knowledge of web development. Figure 3.2 shows the layout of our application by means of two screenshots. The left one displays the two main panels of the GUI: an input panel on the left side (highlighted by a dark grey background) and an output panel (consisting of the tabs “Feature Calculation” and “Visualization”) on the right side. In the input panel, the user can choose between four options for defining the optimization

<sup>6</sup><http://flacco.shinyapps.io/flacco/>

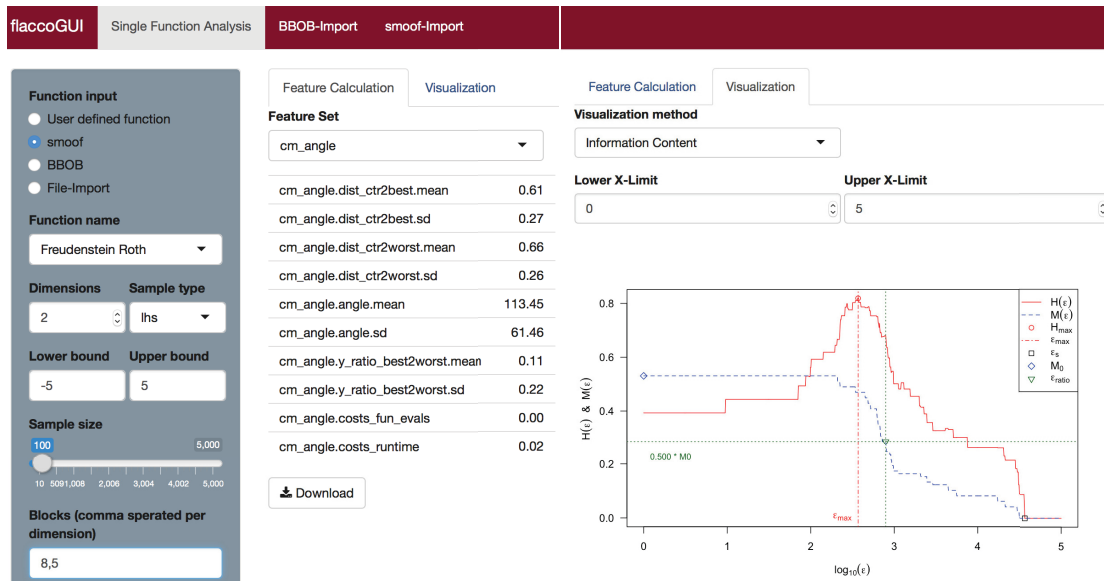


Figure 3.2: The graphical user interface (GUI) of `flacco` basically consists of two panels: (1) an input panel (highlighted by a dark grey area on the very left), where the user can configure the optimization problem, and (2) an output panel that either lists the values for the chosen feature set (the area that is adjacent to the grey input panel, exemplarily shown for the cell mapping angle feature set), or displays one of the package’s various visualization techniques (right half of the image above, which exemplarily shows the *information content plot* as introduced by Muñoz Acosta et al., 2015a).

problem: (1) manually defining it within a text box, (2) selecting any of the single-objective optimization problems from the R-package `smooF` (Bossek, 2016), (3) defining a BBOB problem (via its function and instance IDs), or (4) simply upload an externally evaluated initial design. In the output panel, the user can either compute an entire feature set (as exemplarily shown in the left image of Figure 3.2) or select a visualization of the problem’s landscape<sup>7</sup> or any of its feature sets. For the latter, the user can select between *cell mapping plots* (such as the ones in Figure 2.1), two- or three-dimensional *barrier trees*, or an *information content plot* (e.g., the one shown in the right image of Figure 3.2).

As one of the main purposes of ELA is *automated* feature computation, we also assured that the GUI allows the computation of any user-specified feature set (or all features simultaneously) for multiple problem instances rather than for single instances. The only restriction is that all problems have to belong to the same problem class. That is, all of the problems have to be either BBOB problems or belong to any other single-objective problem class that is implemented in `smooF`. For either of these two options, the application provides a separate screen (“BBOB-Import” or “smooF-Import”), where the user can upload a csv-file with the respective parameter configurations (e.g., function ID, instance ID and problem dimension for the BBOB problems) and in return receive a downloadable table with the computed landscape features. Figure 3.3 shows an example in which the nearest better clustering features are computed for four different BBOB instances. Note that in this example, each feature was computed three times for each of the four instances in order to capture the stochasticity of the features and/or initial design.

<sup>7</sup>The GUI allows to illustrate landscapes of one- or two-dimensional problems and the user can choose between a graph (1D), contour (2D) or surface plot (3D). Examples of the latter two are given within Figure 2.2.

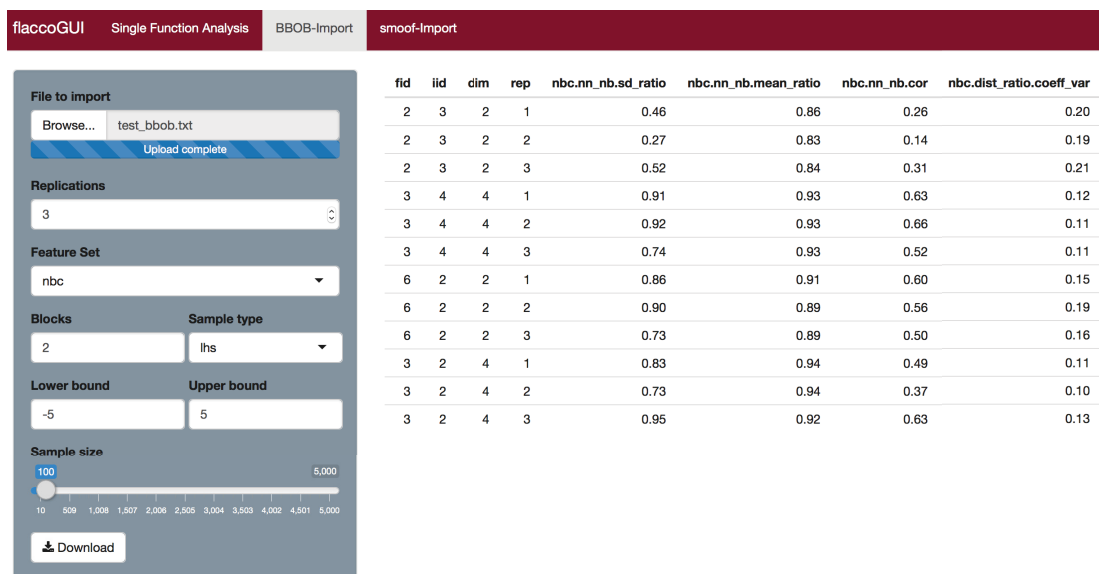


Figure 3.3: Screenshot of the GUI after computing the nearest better clustering features for a set of four different BBOB problems and with three replications each.

### 3.4 Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package `flacco`

After developing the R-package `flacco` and its accompanying GUI, we combined both works within [Kerschke \(under review\)](#) and completed our project by extending the two previous subprojects with a general overview on the existing landscape features (for continuous single-objective optimization), an illustration of the usage of `flacco` by means of a well-known black-box optimization problem<sup>8</sup>, and – most importantly – a detailed description for each of the 17 feature sets that are implemented in the current version (1.7) of our R-package.

In conclusion (of the entire project), we provided a comprehensive toolbox, which unifies a wide collection of landscape features within a convenient, user-friendly and extensible framework. By enhancing it with a graphical user interface, we also have made `flacco` accessible to a much broader group of people – all non-R-users in particular – and thereby enabled them to also benefit from the majority of our framework’s functionalities.

Consequently, many researchers worldwide are now in a position to perform comparative studies with a wide collection of landscape features, which should help to gain more insights into the respective analyzed optimization problem(s). The knowledge derived from these studies can in turn help to perform more sophisticated follow-up actions, as for instance developing new landscape features that aim at detecting specific traits of an optimization problem and have not been addressed so far, or training well-performing (i.e., competitive) automated algorithm selection models as for instance shown in the following chapter.

<sup>8</sup>A two-dimensional instance of *Gallagher’s Gaussian 101-me Peaks* (see, e.g., [Hansen et al., 2009a](#)), i.e., the 21st problem from the Black-Box Optimization Benchmark.

## Chapter 4












# Feature-Based Algorithm Selection from Optimizer Portfolios

*“More data beats clever algorithms, but better data beats more data.”*

---

Peter Norvig

### 4.1 Contributed Material

- Kotthoff, L., Kerschke, P., Hoos, H. H. & Trautmann, H. (2015). *Improving the State of the Art in Inexact TSP Solving using Per-Instance Algorithm Selection*. In: Learning and Intelligent Optimization 9 (LION), pages 202 – 217.     (see Appendix C.1)
- Kerschke, P., Kotthoff, L., Bossek, J., Hoos, H. H. & Trautmann, H. (2017). *Leveraging TSP Solver Complementarity through Machine Learning*. In: Evolutionary Computation Journal (ECJ), pages 1 – 24.     (see Appendix C.2)
- Kerschke, P. & Trautmann, H. (under review). *Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning*.    (see Appendix C.3)

### 4.2 Improving the State of the Art in Inexact TSP Solving using Per-Instance Algorithm Selection

In contrast to the projects described in the previous two sections, which mainly focussed on the characterization of optimization problems by domain-specific features, we went a step further in Kotthoff et al. (2015) and used such features for training per-instance algorithm selectors for the well-known *Travelling Salesperson Problem (TSP)*.

Although there exist two types of TSP algorithms – exact and inexact TSP solvers – for a long time, algorithm selection was not applicable to either one of them as both groups were dominated



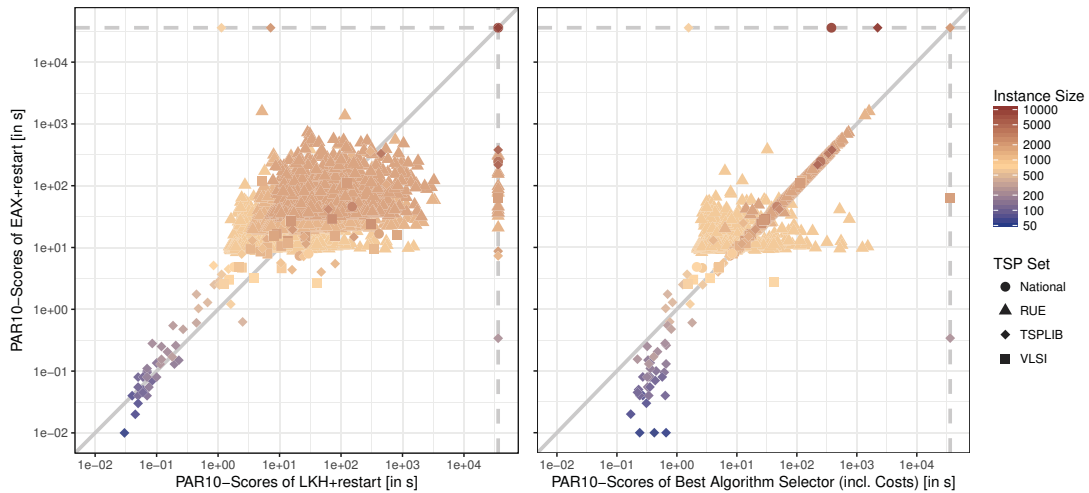


Figure 4.1: Comparison of the portfolio’s single best solver, i.e., EAX+restart (with a mean PAR10-score of 104.01s), with the portfolio’s second best solver, i.e., LKH+restart (422.48s), on the left, as well as our best-performing per-instance algorithm selector, a paired regression MARS model with the corresponding feature costs (95.08s), on the right.

by a single solver each: Concorde (the state of the art among the exact solvers; Applegate et al., 2007) and LKH (inexact; Helsgaun, Keld, 2009). However, with the introduction of the inexact TSP solver EAX, which basically exploits variants of the edge assembly crossover operator, Nagata and Kobayashi (2013) presented an evolutionary algorithm that is competitive to LKH and so, for the first time, made algorithm selection promising within this research domain. By enhancing both heuristics with additional restart-variants, Dubois-Lacoste et al. (2015) introduced two further, very competitive solvers, which perfectly complement our TSP solver portfolio. This competitiveness is in particular observable for the restart version of EAX, which has the best aggregated performance across the entire training set and hence is considered to be the portfolio’s *single best solver (SBS)*. Its complementarity with the restart variant of LKH – which can be seen in the left scatterplot of Figure 4.1 – is a strong indicator for the potential of algorithm selection in this setting.

Within our conducted study, the performances of the four solvers – and thus the ones of the algorithm selectors as well – were measured by means of PAR10-scores (see, e.g., Bischl et al., 2016a). For successful runs, it is identical to the measured runtime, but for unsuccessful runs (usually caused by time- or memouts) it is given a penalty score, which is the tenfold of the largest valid runtime. Here, the walltime was one hour and therefore, the penalized runtime was ten hours (or 36 000s, respectively).

For the purpose of automated algorithm selection, we first created a comprehensive set of TSP problems by collecting instances from four well-known TSP benchmarks: *Random Uniform Euclidean (RUE)* problems, *VLSI* and *National* instances, as well as problems from the *TSPLIB*. While the first set of instances can be created with an artificial problem generator<sup>9</sup>, the other three sets are mostly collections of real-world problems. In order to obtain the corresponding performance values, we executed all four solvers from our portfolio, i.e., EAX, LKH and their restart variants (denoted EAX+restart and LKH+restart, respectively), on each of the col-

<sup>9</sup>The `portgen` generator from the *8th DIMACS Implementation Challenge* (<http://dimacs.rutgers.edu/Challenges/TSP/>).

lected instances. For the same benchmark problems, we also computed two feature sets from the literature (Hutter et al., 2014; Mersmann et al., 2013) and additionally tried a novel set of features, which we denoted *probing* features. The idea of the latter is to monitor the behavior of our portfolio’s *single best solver* (SBS), i.e., EAX+restart, and extract information from its progress across the different generations.

In a first approach, we trained three machine learning models for each of the three supervised learning strategies *classification*, *regression* and *paired regression* as potential algorithm selectors. Due to the fact that the feature costs have a direct impact on the selectors’ performances, each machine learner was trained with different combinations of the feature sets: a subset of 13 rather cheap (i.e., quickly computable) features from Hutter et al. (2014), all 50 features from the same source, the 64 features from Mersmann et al. (2013), as well as the union of both feature sets (114 features). Note that each selector’s performance was assessed with a 10-fold crossvalidation in order to assure reliable performance values.

Our best performing algorithm selector – a paired regression *multivariate adaptive regression spline* (MARS, Friedman, 1991), which was trained with the ‘cheap’ feature subset from Hutter et al. (2014) – achieved a mean PAR10-score of 95.08s (already including the costs for the feature computation) and hence reduced the gap between the single best solver (104.01s) and the *virtual best solver* (VBS; 18.52s) by more than 10%. However, one can also observe the impact of the feature costs by means of the tail in the lower part of Figure 4.1 (right image), which relates to the quickly solvable problems that are usually solved in less than a second. Nevertheless, we successfully showed that – despite the aforementioned overhead for computing problem-specific features – our proposed approach of feature-based algorithm selection still is powerful enough to improve over the current state of the art in inexact TSP solving.

In a second attempt, we used our *probing* features for training the algorithm selection models. The usage of those features comes with the benefit of negligible feature costs on instances that were presolved during the features’ monitoring phase. Even though the probing features themselves might not be as informative as the established sets from the literature, they nevertheless led to algorithm selectors that performed comparable to the SBS. Especially when charging the feature costs only on instances, for which the selected optimization algorithm was different to EAX+restart, our best selector (under this approach), i.e., a regression random forest (Breiman, 2001) with a mean PAR10-score of 103.83s, slightly improved over the SBS. Therefore, our findings based on these (admittedly rather simplistic) online monitoring features indicate that the integration of feature-based per-instance algorithm selection into the optimization process itself could as well be a profitable approach for further research in this area.

### 4.3 Leveraging TSP Solver Complementarity through Machine Learning

Inspired by the promising results from Kotthoff et al. (2015), we enhanced our previous experiments by extending the setup with an additional optimization algorithm, an extra set of TSP features, two further artificial problem benchmarks, more sophisticated machine learning approaches, as well as a thorough in-depth analysis of the corresponding performance data (Kerschke et al., 2017). Based on these extensions, we were able to clearly outperform the state of

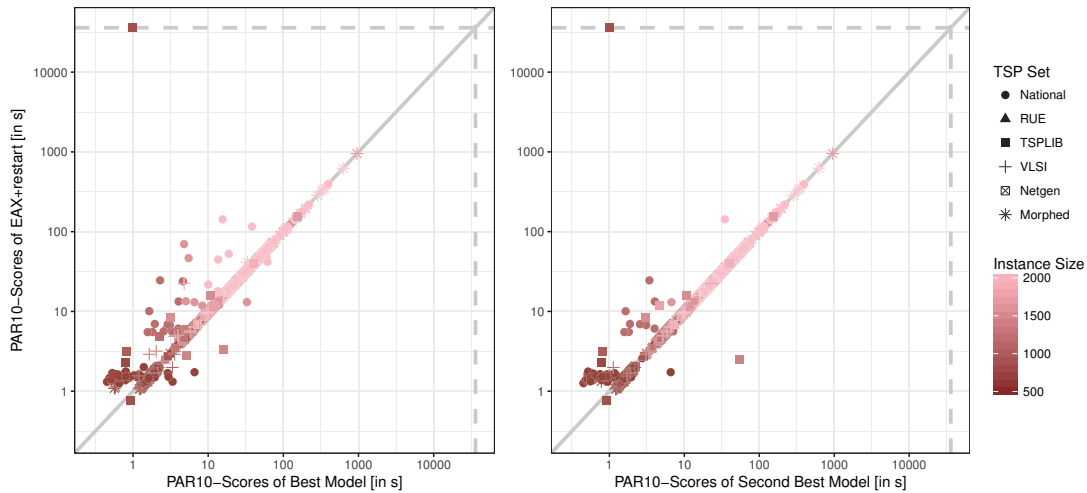


Figure 4.2: Comparison of the single best solver from the portfolio (EAX+restart with a mean PAR10-score of 36.32s) and the two best-performing algorithm selectors, i.e., classification-based SVMs with mean PAR10-scores of 16.75s (left) and 16.93s (right), respectively.

the art in inexact TSP solving by constructing an algorithm selector, which on average required less than half of the resources used by the single best solver from our portfolio.

Within recent works, [Pihera and Musliu \(2014\)](#) introduced a set of 287 TSP features and also showed that the *Multiagent Optimization System (MAOS)* from [Xie and Liu \(2009\)](#) performs competitive to LKH – at least on some of their considered problems – and we therefore added both, MAOS and their proposed features, to our setup. Additionally, we reduced the large impact of the unstructured RUE instances on our experiments – and hence, on the algorithm selectors – by (a) also considering two sets of clustered TSP problems, which were generated with the R-package `netgen` ([Bossek, 2015](#)), and (b) restricting our benchmark to problems of sizes 500 to 2000 cities. While these changes provided us with a much more general data basis, the largest improvements w.r.t. the selectors’ performances were achieved by conducting automated feature selection. Despite the fact that the current implementations of the feature sets did not enable more accurate tracking of the feature costs<sup>10</sup>, using small subsets of the features led to much better performing algorithm selectors. These findings are very plausible, because restricting the number of features obviously reduces the noise and/or redundancy among them.

In the end, we analyzed a total of 280 potential algorithm selectors. The two best of them were classification-based support vector machines ([Vapnik, 1995](#)), which were trained with 16 or 11 features from [Pihera and Musliu \(2014\)](#), respectively. As displayed in Figure 4.2, both models – having average PAR10-scores of 16.75s and 16.93s, respectively – selected for each of the 1845 instances from our benchmark a solver that found an optimal tour within less than 1000s, whereas the single best solver, EAX+restart, found an optimal tour for all but one of the considered problems<sup>11</sup> and had a mean PAR10-score of 36.30s. Therefore, based on these PAR10-scores, both selectors found an optimal tour on average more than twice as fast as the SBS and thus, reduced the gap towards the virtual best solver (10.73s) – i.e., the performance

<sup>10</sup>Independent of the amount of features that were chosen from a specific feature set, we always had to consider the costs for computing *all* features from the respective set.

<sup>11</sup>While EAX, EAX+restart and MAOS failed to solve the TSPLIB-instance `d657` within the given time limit of one hour, LKH and LKH+restart solved it in less than a second.

that one could *theoretically* achieve by always selecting the best-performing solver per instance – by more than 75% (from 25.57s to 6.02s or 6.20s, respectively). Especially when considering that the VBS is measured under very idealistic settings (i.e., perfect oracle-like selections per instance without any additional costs), our algorithm selectors set a very strong and much more *realistic* baseline for the state of the art in inexact TSP solving.

In spite of these very positive results, we still see potential for further improvements. Currently, the source code for the feature computation did not enable a more realistic estimation of the true feature costs when using only (small) subsets of the feature sets. In consequence, the costs that we considered for assessing the selectors’ performances provided an upper bound for the true costs. Also, except for the SVM’s inverse kernel width parameter `sigma`, which strongly influences the SVM’s performance and hence was set upfront to a more reasonable default value, all of our considered machine learning algorithms were executed with default configurations for their respective hyperparameters. Given that the accuracy of a machine learner often relies on its hyperparameter configuration, tuning them could yield even better performances.

## 4.4 Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning

In Kerschke and Trautmann (under review), we transferred our successful strategy of feature-based algorithm selection from the *Travelling Salesperson Problem* (as described in the two previous works) to the domain of single-objective continuous black-box optimization. There, we combined it with our landscape analysis framework `flacco` (see Chapter 3), as well as our findings regarding an optimization problem’s high-level properties, such as the global structure (see Chapter 2).

For this purpose, we first computed all of the more than 300 features that are currently available in `flacco` on a set of 96 problem instances<sup>12</sup> from the Black-Box Optimization Benchmark (BBOB, Hansen et al., 2009b). Due to our recent findings, which showed that low budgets are sufficient for detecting certain problem characteristics (Kerschke et al., 2016a), we again used rather small initial designs of only  $50 \times d$  observations for our experiments. For the same set of instances, we acquired the performance results of several optimization algorithms from the COCO-platform (Hansen et al., 2016). Between the years 2009 and 2015, the latter collected the performances (i.e., the best objective values that were found along with the corresponding number of function evaluations) from a total of 129 optimization algorithms on BBOB. Using this external source enabled us to compare performances from a wide variety of well-established optimization algorithms (without having the burden of making their source code executable on our local machines) and also made our results more comparable for other researchers (as they can simply use the same data base). Therefore, instead of wasting valuable time for dealing with the optimizers’ implementations, we rather invested more resources in a thorough revision of the acquired performance data in order to assure a clean data basis for our experiments.

---

<sup>12</sup>For each of the 24 BBOB functions, we considered four different problem dimensions,  $d \in \{2, 3, 5, 10\}$ , and aggregated the feature and performance data across the respective first five instances,  $IID \in \{1, \dots, 5\}$ .

On the basis of our data analysis, we reduced the set of all 129 solvers to a much smaller, but still very competitive, portfolio of 12 representative optimization algorithms. It consists of two very fast derivative-based solvers, several variants of the well-known *Covariance Matrix Adaption Evolution Strategy* (CMA-ES, Hansen and Ostermeier, 2001), as well as some multi level approaches. Compared to all 129 solvers from COCO, the best solver (per instance) from our portfolio also achieved the best results on 51 of all 96 problems and was at most twice (three times) as slow for ten (three) of the problems. While a hybrid version of the CMA-ES (denoted HCMA, Loshchilov et al., 2013) clearly was the single best solver across the entire benchmark – it also was the only one to approximate the optimal objective value for all 96 problems up to the considered precision level of  $10^{-2}$  – we also confirmed the thesis that the performance of an optimization algorithm heavily relies on the given optimization problem. That is, while the derivative-based approaches on average worked best across the separable problems, more complex problems required much more sophisticated optimizers. Therefore, having a priori knowledge of the problem’s landscape should clearly help to find (per instance) a more appropriate, i.e., better performing, optimization algorithm – compared to sticking to a single algorithm across all problems.

In contrast to the TSP-related projects, in which we measured (the PAR10-score of) the CPU runtime, we here considered the so-called *relative Expected Runtime* (relative ERT) as performance measure. The ‘regular’ *Expected Runtime* (ERT, Hansen et al., 2009a) basically measures the average number of function evaluations (across multiple runs of an algorithm on an instance) that are needed to “solve” an instance. Here, “solving” an instance means that the optimizer found an observation, whose objective value differs at most by a pre-defined precision value  $\varepsilon$  from the landscape’s true global optimum. As the complexity of the optimization problem has a strong impact on the size of the corresponding ERTs<sup>13</sup>, we standardize each optimizer’s ERT per problem and dimension by the respective problem’s best ERT (among the portfolio’s optimization algorithms). Thereby, the performance values of simple and complex problems should become much more comparable to each other across the entire benchmark. Standardizing the ERT per problem and dimension also simplifies the interpretation of the results: the relative ERT of the virtual best solver obviously has to be exactly one on each of the problems and hence, also on the entire benchmark. In consequence, the relative ERTs of the solvers and selectors, basically provide a factor compared to the VBS. For instance, the performance of the single best solver, which achieved a relative ERT of 30.37, implies that HCMA on average requires 30.37 times as many function evaluations for solving an instance as the respective ideal optimizer from our portfolio.

For modeling our algorithm selectors, we considered a similar approach as in Kerschke et al. (2017) and combined several machine learning approaches with different feature selection strategies as potential algorithm selectors. Their performances were assessed by a leave-one-function-out crossvalidation, i.e., we trained each model on 95 of the 96 problems and validated it on the one that was left out. This step is repeated 96 times, so that each problem was used for testing exactly once and afterwards averaged across the resulting 96 relative ERTs. Again, the best performance was achieved by a classification-based support vector machine (Vapnik, 1995). It had a mean relative ERT of 16.67, which means that it (on average) required less than 55% of

<sup>13</sup>Low-dimensional, unimodal, separable functions can be successfully solved with a very small amount of function evaluations, whereas high-dimensional, highly multimodal problems often require millions of function evaluations – if they can be solved at all.

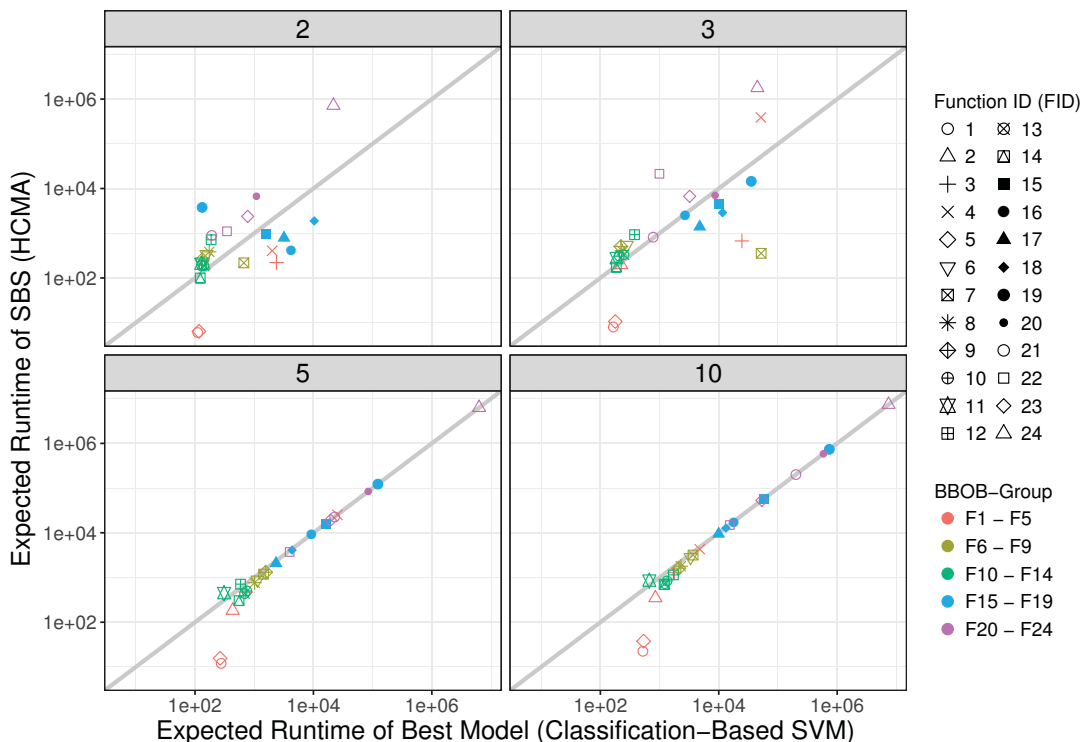


Figure 4.3: Comparison of the HCMA, i.e., the portfolio’s single best solver with a mean relative ERT of 30.37, and the best-performing algorithm selector, a classification-based SVM (14.24). The performances are distinguished by problem dimension and BBOB problem (FID).

HCMA’s number of function evaluations for solving an instance.

Examining the features that were used by our best selector revealed that neither the NBC nor the meta-model features, i.e., features describing a landscape’s global structure, were considered by the found algorithm selector. Instead, it used one levelset and three y-distribution features (Mersmann et al., 2011), two information content features (Muñoz Acosta et al., 2015a), one cell mapping feature (Kerschke et al., 2014) and one of the basic features. However, being aware of the imperfectness of our feature selection strategies – amongst others due to their usage of an either greedy or random approach – we tried a new approach by combining the previously selected eight features with the NBC and meta-model features and performing a second round of feature selection on top. As a result, we found a composition of (three features from the previous model, two meta-model and four NBC) features, which led to an even better algorithm selection model with a mean relative ERT of 14.24. The corresponding performance differences between the portfolio’s best solver, i.e., HCMA, and our best-performing per-instance algorithm selector are displayed in Figure 4.3.

Analyzing our two algorithm selectors in more detail, we detected that both models performed poorly on the separable BBOB problems (FIDs 1 to 5), whereas especially the better (second) model outperformed all single solvers on the group of multimodal problems with a weak global structure (FIDs 20 to 24). However, bringing the size of the initial design in relation to the complexity of the problems, both findings are plausible. The separable problems usually can be solved with a few function evaluations, giving the costs for the initial design – 100 to 500 function evaluations depending on the problem’s dimensionality – a strong negative impact

on the selector’s performance, which is also visible in Figure 4.3. On the other hand, the multimodal problems with a weak global structure are so complex that they require a huge amount of function evaluations – if they can be solved at all. For instance, HCMA had an ERT of 7.3 million function evaluations on the ten-dimensional version of the *Lunacek Bi-Rastrigin Function* (FID 24, Hansen et al., 2009b), but still was the portfolio’s only algorithm to solve this problem at all. In such extreme cases, the cost of at most 500 function evaluations for computing the landscape features becomes completely insignificant compared to the high amount of function evaluations that one would actually *waste* when selecting a solver that either (a) performs very poorly or (b) does *not* even find the optimum for the given problem.

All in all, one can state that the computation of landscape features on very simple problems might cause some overhead, but feature-based algorithm selection has definitely proven to be a very powerful tool, especially on more complex (and thus expensive) problems. Consequently, spending a small amount of function evaluations on the feature computation can in turn be very profitable considering the strong performance improvements that one could achieve with a suitable, i.e., well-performing, optimization algorithm. Furthermore, in order to avoid spending too many function evaluations on these very simple problems, future research could aim at designing new landscape features, which are able to distinguish these very simple problems from all the other ones, by means of an extremely minimalistic budget or initial design.

Finally, considering that the size of the initial design is already in the range of an evolutionary algorithm’s population size, *low-budget* landscape analysis provides essential information for improved continuous black-box optimization basically for free<sup>14</sup>. Instead of using two different sets of observations for both – i.e., the solver’s initial population and the features’ initial design – one could either use the initial population for computing the landscape features or the initial design as the solver’s starting population. Either way, the features would be available for free.

---

<sup>14</sup>Except for the aforementioned very simple problems, which ideally are solved by fast, deterministic (local search) optimization algorithms.

## Chapter 5

# Platforms for Collaborative Research on Algorithm Selection and Machine Learning

*“Great things in business are never done by one person, they’re done by a team of people.”*

---

Steve Jobs

### 5.1 Contributed Material

- Bischl, B., Kerschke, P., Kotthoff, L., Lindauer, T. M., Malitsky, Y., Frechette, A., Hoos, H. H., Hutter, F., Leyton-Brown, K., Tierney, K. & Vanschoren, J. (2016). *ASlib: A Benchmark Library for Algorithm Selection*. In: Artificial Intelligence Journal (AIJ), pages 41 – 58.   (see Appendix D.1)
- Casalicchio, G., Bossek, J., Lang, M., Kirchhoff, D., Kerschke, P., Hofner, B., Seibold, H., Vanschoren, J. & Bischl, B. (2017). *OpenML: An R Package to Connect to the Machine Learning Platform OpenML*. In: Journal of Computational Statistics (COST), pages 1 – 15.   (see Appendix D.2)

### 5.2 ASlib: A Benchmark Library for Algorithm Selection

As shown in the previous chapter, feature-based per-instance algorithm selection can yield impressive performance improvements within different research domains. And although there has been a steady growth in the number of related research projects (see, e.g., Kotthoff, 2014; Muñoz Acosta et al., 2015b), there exists no common platform for researchers, where they can share and/or compare their experimental data and approaches. Therefore, in order to increase both, comparability and exchangeability of algorithm selection results from different studies, we defined a standardized format (for data sets of algorithm selection scenarios) and introduced the *Algorithm Selection Library*<sup>15</sup>(ASlib, Bischl et al., 2016a).

The ASlib is an online platform, which (a) comes with a repository for sharing various data sets (features, feature costs, algorithm performances, runstatus, etc.) for a wide variety of

---

<sup>15</sup><http://aslib.net/>



algorithm selection scenarios, (b) provides useful insights for each of the uploaded scenarios by means of an automatically generated explorative data analysis, and (c) runs some initial benchmark experiments (with a few standard machine learning algorithms) per scenario. For better usability of the library, we additionally provided interfaces for accessing the platform either from python (`ASlibScenario`<sup>16</sup>) or R (`aslib`<sup>17</sup>, [Bischl et al., 2016b](#)). Thereof we expect a facilitated communication between the users on the one end and the ASlib on the other.

Due to these functionalities, the library mostly simplifies the steps *prior* to actually performing algorithm selection. That is, one neither has to deal with any of the implementations of the different optimization algorithms nor with the feature computations and their costs. Each of the AS scenarios available on the library assured to provide all these required data sets and furthermore the corresponding data sets also have to pass various sanity checks prior to being published on the platform. Also, due to the standardized data format, the user has to cope with the format specifications only once – at the very first encounter with any of the scenarios. In future experiments, users can easily adapt their code from the previous experiments by replacing the links to the ‘old’ data sets with the respective ones from any of the other scenarios.

An additional advantage and at the same time very important contribution of the ASlib is its automatically generated *Exploratory Data Analysis (EDA)*<sup>18</sup>, which provides valuable insights into the performance and feature data of each scenario. On its first page, it summarizes all available scenarios by means of a table with the number of instances, (optimization) algorithms and features. The same table also displays whether the data contains stochastic features or algorithms, respectively. Therefore, a user who is looking for a specific setup for an AS scenario, can quickly scan over the entire list of all scenarios and afterwards explore the relevant ones in more detail. For the latter, our EDA provides a multitude of visualizations, such as boxplots, density or cumulative distribution functions (of the algorithm performances), as well as several tables summarizing the presented performance and feature values.

Aside from the scenario’s raw data, the aforementioned EDA and a readme-file, which provides further information on the source of the respective scenario, the library offers summaries of standard baseline measures – i.e., the virtual best solver (VBS) and the single best solver (SBS) – along with the results of an initial benchmark study based on well-established machine learning algorithms (e.g., random forests and support vector machines). These benchmark tables, which show the respective performances by means of three different measures – success rates, PAR10-scores and misclassification penalties – should give the user some first indications on the size of the gap between VBS and SBS, or VBS and best machine learning algorithm, respectively. Hence, the users get an approximate idea of how much improvement one could expect from more thoroughly trained algorithm selectors.

A further contribution that we have made in [Bischl et al. \(2016a\)](#) is a brief description of the six optimization domains that were covered by the platform’s 17 algorithm selection scenarios (at the time of the release of version 2.0). However, in its current version (3.0), the library contains 21 scenarios (originating from ten optimization domains), as well as an additional repository for unverified scenarios, which have the potential to be integrated in future releases. At last, we also presented an exploratory benchmark study, whose results on the one hand displayed the

---

<sup>16</sup><https://github.com/mlindauer/ASlibScenario>

<sup>17</sup><https://github.com/coseal/aslib-r>

<sup>18</sup><http://coseal.github.io/aslib-r/scenario-pages/index.html>

diversity of the available scenarios and on the other hand provided a nice baseline for further studies on each of them.

In conclusion, the development and introduction of the ASlib finally provided a platform where researchers can access and share the data from various experimental studies related to algorithm selection. Furthermore, we provided various tools which enable a stronger focus on the actual algorithm selection itself as users no longer have to deal with the usually rather time consuming preprocessing steps, such as collecting the data, performing sanity checks, or extracting further information from the data, e.g., by means of an explorative data analysis. Therefore, our platform clearly facilitates collaborations across different applications for algorithm selection, e.g., *Satisfiability (SAT)* or *Travelling Salesperson Problem (TSP)*, as (expert) knowledge of those is – due to the provided data sets – no longer a mandatory prerequisite.

## 5.3 OpenML: An R Package to Connect to the Machine Learning Platform OpenML

While the previously described Algorithm Selection Library aims at experimental studies that are specifically related to the topic of algorithm selection, the online machine learning platform OpenML<sup>19</sup> (Vanschoren et al., 2013) targets a much broader community: machine learning researchers in general.

The main goal of OpenML is to provide a place, where researchers can easily collaborate (online), e.g., by sharing their data, machine learning algorithms, experimental setups, or even entire experiments. Due to the modularized structure of the platform, researchers can directly benefit from advances made by any other member of this network within a specific part of their experimental workflow. In order to enable as many researchers as possible access into the network, OpenML provides interfaces to the majority of machine learning applications, namely WEKA (Hall et al., 2009), MOA (Bifet et al., 2010), RapidMiner (Hofmann and Klinkenberg, 2013), Java (e.g., Arnold et al., 2005), R (R Core Team, 2017) and python (VanRossum and The Python Development Team, 2015).

In Casalicchio et al. (2017b), we introduced the R-package `OpenML` (Casalicchio et al., 2017a), which provides the interface from R to OpenML and vice versa. Our package is closely connected to `m1r` (Bischl et al., 2016c), which probably is the most complex and at the same time most powerful framework for machine learning in R. Within our work, we provide a description of the different building blocks of OpenML, i.e., the *data sets*, *tasks* (i.e., encapsulations of the data sets, enriched by further information such as the supervised learning type, the desired performance measure and resampling strategy), *flows* (i.e., the considered machine learning algorithms along with specifications of their possible hyperparameters) and the *runs* (i.e., results as achieved by a flow with a specific hyperparameter configuration and applied to a certain task).

For facilitating the first steps with OpenML and in particular with our package, we (a) designed a tutorial<sup>20</sup>, and (b) also presented a case study (within our work), in which we demonstrate the usage of our package in combination with OpenML and the `m1r` framework.

---

<sup>19</sup><https://www.openml.org/>

<sup>20</sup><https://cran.r-project.org/web/packages/OpenML/OpenML.pdf> or  
<https://cran.r-project.org/web/packages/OpenML/vignettes/OpenML.html>

Summarizing, one can state that the R-package `OpenML` provides R-users easy access to the very promising machine learning platform OpenML. Due to the many interfaces of the platform, its users are enabled to collaborate with many other scientists worldwide and independent of their favorite programming language. After all, such a powerful network will help its members to jointly tackle complex machine learning problems and thereby push the entire, already very vivid, research domain even further.

# Chapter 6

## Summary and Outlook

*“Realize that everything connects to everything else.”*

---

Leonardo da Vinci

This thesis was structured into four parts in order to represent the different aspects that contributed towards the overall goal of performing *“Automated and Feature-Based Problem Characterization and Algorithm Selection Through Machine Learning”*. As discussed within this work, we have made valuable progress in each of these categories with the most important ones among them being summarized below.

### **Characterizing the Global Structure of Continuous Black-Box Problems**

We proposed new feature sets, which improved the classification of a problem’s degree of multimodality and its global structure. In particular, these features enabled the distinction of funnel-shaped problems from landscapes with rather randomly distributed local optima. More importantly, we showed that our features, despite being computed by means of very low budgets, always classified the aforementioned properties with a high accuracy (of 90% and more).

### **Framework for Exploratory Landscape Analysis**

The two essential contributions of this category have been the development of the R-package `flacco` and its accompanying platform-independent and web-hosted graphical user interface, which even enables non-R-users to benefit from this framework’s functionalities. The package itself provides a collection of more than 300 automatically computable landscape features, as well as several visualization techniques. While the latter should help its users to get a better understanding of the problems at hand, the former is a very important ingredient for performing automated algorithm selection.

### **Experimental Studies on Feature-Based Algorithm Selection**

We have conducted experimental studies to show the applicability of feature-based algorithm selection within different application domains. For the *Travelling Salesperson Problem (TSP)*,

our best algorithm selector completely dominated the *single best solver (SBS)* from our portfolio of state-of-the-art heuristic TSP solvers. More precisely, we reduced the gap between the *virtual best solver (VBS)*, i.e., an oracle like predictor that always predicts the best performing optimization algorithm for every single instance (at no costs), and the aforementioned SBS by more than 75%. Within the second considered application, i.e., single-objective continuous black-box optimization, we performed algorithm selection on a portfolio of 12 complementary solvers that were evaluated across all 24 functions from the well-known Black-Box Optimization Benchmark (BBOB, Hansen et al., 2009a) and across four different problem dimensions. Similar to the TSP experiments, our best algorithm selection model again required less than half of the resources compared to the respective SBS. Especially the ‘funnel features’, which we proposed for characterizing the structures of continuous black-box problems proved to be valuable for our algorithm selector.

### Development of Platforms for Machine Learning and Algorithm Selection

At last, we have made contributions to two online platforms, which encourage collaborations among researchers in the field of machine learning (OpenML) in general and algorithm selection (ASlib) in particular. On either of these platforms, the users can publicly share their data, which in turn allows others to study those in more detail. Consequently, being checked by numerous people, it is more likely that the provided data sets – and hence, the experiments conducted on them – are of better quality. Also, the platforms facilitate contributions to separate parts of the algorithm selection and/or machine learning models and thereby help to improve them without too much additional effort.

Based on these very promising results, we have laid the basis for several new developments within the analyzed research field. Focussing on the automated characterization of problem landscapes, it would be interesting to investigate how much one could reduce the available budget without losing (too much) information and thereby accuracy. While one could obviously try to tackle this problem by directly decreasing the size of the initial designs, one would assume that more sophisticated approaches, such as combining landscape analysis with surrogate-assisted models (e.g., Kriging, Jones et al., 1998), yields more promising results.

Also, a weakness of our feature-based algorithm selection (in case of continuous optimization) was its poor performance on very simple problems. Admittedly, this weakness is negligible as it obviously only affects a few very special problems (such as for instance the *Sphere* and *Linear Slope* functions). Nevertheless, designing features that are able to detect these (usually separable) problems by means of a very limited amount of function evaluations, likely will have a strong positive impact on the overall performance of the algorithm selection models.

So far, all of these landscape features have been designed for tackling single-objective problems. However, as the majority of real-world problems usually have to be optimized w.r.t. more than one objective, we need features that are able to characterize such multi-objective landscapes. Again, one could use naïve approaches, such as computing the features per objective, but then one would lose (probably essential) information on the interaction between the objectives. So, although some initial steps towards characterizing such landscapes have already been made (see, e.g., Kerschke and Grimme, 2017; Kerschke et al., 2016b, under review), this research field is

---

one of the most important ones to address in the near future.

Aside from the continuous landscapes, I also see possible enhancements w.r.t. features characterizing the TSP instances. The implementations of most of the TSP feature sets within this thesis only allow to measure their costs across the entire feature set. However, as our experimental studies have shown, usually only small subsets of these features are sufficient for training well-performing algorithm selectors. Thus, measuring the feature costs at a smaller level would (a) allow the machine learning algorithms to choose from a wider collection of features (currently, cheap features that are part of an expensive feature set, will usually be disregarded by the machine learning algorithm), and (b) provide better estimates for the required runtime (so far, the performances of our algorithm selectors are only upper bounds for the actual costs).

Furthermore, the hyperparameters of our algorithm selection models have always been used within their default configurations. However, as shown in many works of the adjacent research field of *algorithm configuration*, tuning an algorithm to the problem at hand usually yields strong performance improvements as well. Based on those results and the contributions presented within this thesis, there is a lot of potential for a fruitful combination of these two approaches. On the one hand, algorithm configuration could strongly benefit from our automated feature computation as our features could reduce the hyperparameters' search space prior to actually running the algorithm configurator and thereby clearly decrease its required computational budget. On the other hand, the configurators could be used for tuning our machine learning algorithms, which in turn should push them towards even better performances.

As a final enhancement of our results, one could – instead of computing the landscape features offline *before* the optimization – measure them online while actually running the optimization algorithms. By means of such online monitoring features (similar to the ‘probing features’ that we proposed in [Kotthoff et al., 2015](#)) one could either (a) switch between different optimization algorithms from a pre-defined portfolio or (b) tune the optimizer’s hyperparameters so that it focusses on exploiting the landscape’s promising regions – and thereby saves valuable function evaluations.

# Bibliography

*“It’s the knowledge derived from information that gives you a competitive edge.”*

---

Bill Gates

- Bernardetta Addis, Marco Locatelli, and Fabio Schoen. Disk Packing in a Square: A New Global Optimization Approach. *INFORMS Journal on Computing*, 20(4):516 – 524, 2008. doi: 10.1287/ijoc.1080.0263. URL <http://pubsonline.informs.org/doi/abs/10.1287/ijoc.1080.0263>.
- David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA, 2007. URL <http://press.princeton.edu/titles/8451.html>.
- Ken Arnold, James Gosling, and David Holmes. *The Java (TM) Programming Language*. Addison Wesley Professional, 4th edition, 2005. URL <http://dl.acm.org/citation.cfm?id=1051069>.
- Brian Beachkofski and Ramana Grandhi. Improved Distributed Hypercube Sampling. In *Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, pages 1 – 7. American Institute of Aeronautics and Astronautics (AIAA), 2002. doi: 10.2514/6.2002-1274. URL <https://arc.aiaa.org/doi/abs/10.2514/6.2002-1274>.
- Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. MOA: Massive Online Analysis. *Journal of Machine Learning Research (JMLR)*, 11(May):1601 – 1604, 2010. URL <http://www.jmlr.org/papers/v11/bifet10a.html>.
- Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike 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)*, pages 313 – 320. ACM, July 2012. doi: 10.1145/2330163.2330209. URL <http://dl.acm.org/citation.cfm?doid=2330163.2330209>.
- Bernd Bischl, Pascal Kerschke, Lars Kotthoff, Thomas Marius Lindauer, Yuri Malitsky, Alexandre Fréchet, Holger Hendrik Hoos, Frank Hutter, Kevin Leyton-Brown, Kevin Tierney, and Joaquin Vanschoren. ASlib: A Benchmark Library for Algorithm Selection. *Artificial Intelligence Journal (AIJ)*, 237:41 – 58, 2016a. doi: 10.1016/j.artint.2016.04.003. URL <https://www.sciencedirect.com/science/article/pii/S0004370216300388>.

- Bernd Bischl, Lars Kotthoff, and Pascal Kerschke. *aslib: Interface to the Algorithm Selection Benchmark Library*, November 2016b. URL <https://CRAN.R-project.org/package=aslib>. R-package version 0.1.
- Bernd Bischl, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. mlr: Machine Learning in R. *Journal of Machine Learning Research (JMLR)*, 17(170):1 – 5, 2016c. URL <http://jmlr.org/papers/v17/15-066.html>.
- Jakob Bossek. *netgen: Network Generator for Combinatorial Graph Problems*, 2015. URL <https://CRAN.R-project.org/package=netgen>. R-package version 1.2.
- Jakob Bossek. *smoof: Single and Multi-Objective Optimization Test Functions*, April 2016. URL <https://CRAN.R-project.org/package=smoof>. R-package version 1.5.
- Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi: 10.1017/CBO9780511804441. URL <https://web.stanford.edu/~boyd/cvxbook/>.
- Leo Breiman. Random Forests. *Machine learning*, 45(1):5 – 32, 2001. doi: 10.1023/A:1010933404324. URL <https://link.springer.com/article/10.1023/A:1010933404324>.
- Charles George Broyden. The Convergence of a Class of Double-Rank Minimization Algorithms 2. The New Algorithm. *Journal of Applied Mathematics (JAM)*, 6(3):222 – 231, September 1970. doi: 10.1093/imamat/6.3.222. URL <https://doi.org/10.1093/imamat/6.3.222>.
- Faruk Halil Bursal and Chieh Su Hsu. Application of a Cell-Mapping Method to Optimal Control Problems. *International Journal of Control (IJC)*, 49(5):1505 – 1522, 1989. doi: 10.1080/00207178908559722. URL <http://www.tandfonline.com/doi/abs/10.1080/00207178908559722>.
- Giuseppe Casalicchio, Bernd Bischl, Dominik Kirchhoff, Michel Lang, Benjamin Hofner, Jakob Bossek, Pascal Kerschke, and Joaquin Vanschoren. *OpenML: Exploring Machine Learning Better, Together*, June 2017a. URL <https://CRAN.R-project.org/package=OpenML>. R-package version 1.4.
- Giuseppe Casalicchio, Jakob Bossek, Michel Lang, Dominik Kirchhoff, Pascal Kerschke, Benjamin Hofner, Heidi Seibold, Joaquin Vanschoren, and Bernd Bischl. OpenML: An R package to connect to the machine learning platform OpenML. *Journal of Computational Statistics (COST)*, pages 1 – 15, June 2017b. doi: 10.1007/s00180-017-0742-2. URL <https://link.springer.com/article/10.1007/s00180-017-0742-2>.
- Winston Chang, Joe Cheng, Joseph J. Allaire, Yihui Xie, and Jonathan McPherson. *shiny: Web Application Framework for R*, 2016. URL <https://CRAN.R-project.org/package=shiny>. R package version 0.14.1.
- Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization*, Advanced Information



- and Knowledge Processing, pages 105 – 145. Springer, 2005. doi: 10.1007/1-84628-137-7\_6. URL [https://link.springer.com/chapter/10.1007/1-84628-137-7\\_6](https://link.springer.com/chapter/10.1007/1-84628-137-7_6).
- J eremie Dubois-Lacoste, Holger Hendrik Hoos, and Thomas St utzle. On the Empirical Scaling Behaviour of State-of-the-art Local Search Algorithms for the Euclidean TSP. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 377 – 384, New York, NY, USA, 2015. ACM. doi: 10.1145/2739480.2754747. URL <http://dl.acm.org/citation.cfm?doid=2739480.2754747>.
- Russell Eberhart and James Kennedy. A New Optimizer Using Particle Swarm Theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS)*, pages 39 – 43. IEEE, October 1995. doi: 10.1109/MHS.1995.494215. URL <http://ieeexplore.ieee.org/document/494215/>.
- Jerome Harold Friedman. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, pages 1 – 67, 1991. URL <http://www.jstor.org/stable/2241837>.
- Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10 – 18, 2009. doi: 10.1145/1656274.1656278. URL <http://dl.acm.org/citation.cfm?id=1656278>.
- Nikolaus Hansen. The CMA Evolution Strategy: A Comparing Review. *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*, 192:75 – 102, 2006. doi: 10.1007/3-540-32494-1\_4. URL [https://link.springer.com/chapter/10.1007/3-540-32494-1\\_4](https://link.springer.com/chapter/10.1007/3-540-32494-1_4).
- Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation Journal (ECJ)*, 9(2):159 – 195, 2001. URL <http://www.mitpressjournals.org/doi/10.1162/106365601750190398>.
- Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. Real-Parameter Black-Box Optimization Benchmarking 2009: Experimental Setup. Technical Report RR-6828, INRIA, 2009a. URL <https://hal.inria.fr/inria-00362649v3/document>.
- Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Technical Report RR-6829, INRIA, 2009b. URL <https://hal.inria.fr/inria-00362633/document>.
- Nikolaus Hansen, Anne Auger, Olaf Mersmann, Tea Tu sar, and Dimo Brockhoff. COCO: A Platform for Comparing Continuous Optimizers in a Black-Box Setting. *arXiv preprint*, August 2016. URL <http://arxiv.org/abs/1603.08785v3>.
- Christian Hanster and Pascal Kerschke. flaccogui: Exploratory Landscape Analysis for Everyone. In *Proceedings of the 19th Annual Conference on Genetic and Evolutionary Computation (GECCO) Companion*, pages 1215 – 1222. ACM, July 2017. doi: 10.1145/3067695.3082477. URL <http://dl.acm.org/citation.cfm?doid=3067695.3082477>.
- Helsgaun, Keld. General k-opt submoves for the Lin-Kernighan TSP heuristic. *Mathematical Programming Computation*, 1(2-3):119 – 163, 2009. doi: 10.1007/s12532-009-0004-6. URL <https://link.springer.com/article/10.1007/s12532-009-0004-6>.

- Markus Hofmann and Ralf Klinkenberg. *RapidMiner: Data Mining Use Cases and Business Analytics Applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery. CRC Press, 2013. URL <http://dl.acm.org/citation.cfm?id=2543538>.
- Chieh Su Hsu. *Cell-to-Cell Mapping: A Method of Global Analysis for Nonlinear Systems*, volume 64. Springer, 1987. doi: 10.1007/978-1-4757-3892-6. URL <https://link.springer.com/book/10.1007/978-1-4757-3892-6>.
- Frank Hutter, Lin Xu, Holger Hendrik Hoos, and Kevin Leyton-Brown. Algorithm Runtime Prediction: Methods & Evaluation. *Artificial Intelligence Journal (AIJ)*, 206:79 – 111, 2014. doi: 10.1016/j.artint.2013.10.003. URL <http://www.sciencedirect.com/science/article/pii/S0004370213001082>.
- Donald R. Jones, Matthias Schonlau, and William James Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455 – 492, December 1998. doi: 10.1023/A:1008306431147. URL <https://link.springer.com/article/10.1023/A:1008306431147>.
- Pascal Kerschke. *flacco: Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems*, June 2017. URL <https://cran.r-project.org/package=flacco>. R-package version 1.7.
- Pascal Kerschke. Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package flacco. under review. Current version available at arXiv (<https://arxiv.org/abs/1708.05258>).
- Pascal Kerschke and Christian Grimme. An Expedition to Multimodal Multi-Objective Optimization Landscapes. In Heike Trautmann, Günter Rudolph, Klamroth Kathrin, Oliver Schütze, Margaret Wiecek, Yaochu Jin, and Christian Grimme, editors, *Proceedings of the 9th International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, pages 329 – 343. Springer, March 2017. ISBN 978-3-319-54157-0. doi: 10.1007/978-3-319-54157-0\_23. URL [http://link.springer.com/chapter/10.1007/978-3-319-54157-0\\_23](http://link.springer.com/chapter/10.1007/978-3-319-54157-0_23).
- Pascal Kerschke and Heike Trautmann. The R-Package FLACCO for Exploratory Landscape Analysis with Applications to Multi-Objective Optimization Problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 5262 – 5269. IEEE, July 2016. doi: 10.1109/CEC.2016.7748359. URL <http://ieeexplore.ieee.org/document/7748359/>.
- Pascal Kerschke and Heike Trautmann. Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning. under review. Current version available at arXiv (<https://arxiv.org/abs/1711.08921>).
- Pascal Kerschke, Mike Preuss, Carlos Ignacio Hernández Castellanos, Oliver Schütze, Jian-Qiao Sun, Christian Grimme, Günter Rudolph, Bernd Bischl, and Heike 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, July 2014. doi: 10.1007/978-3-319-07494-8\_9. URL [https://link.springer.com/chapter/10.1007/978-3-319-07494-8\\_9](https://link.springer.com/chapter/10.1007/978-3-319-07494-8_9).

- Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Detecting Funnel Structures by Means of Exploratory Landscape Analysis. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 265 – 272. ACM, July 2015. doi: 10.1145/2739480.2754642. URL <http://dl.acm.org/citation.cfm?doid=2739480.2754642>.
- Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models. In *Proceedings of the 18th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 229 – 236. ACM, July 2016a. doi: 10.1145/2908812.2908845. URL <http://dl.acm.org/citation.cfm?doid=2908812.2908845>.
- Pascal Kerschke, Hao Wang, Mike Preuss, Christian Grimme, André H. Deutz, Heike Trautmann, and Michael T. M. Emmerich. Towards Analyzing Multimodality of Multiobjective Landscapes. In Julia Handl, Emma Hart, Peter Richard Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *Proceedings of the 14th International Conference on Parallel Problem Solving from Nature (PPSN XIV)*, volume 9921 of *Lecture Notes in Computer Science (LNCS)*, pages 962 – 972. Springer, September 2016b. doi: 10.1007/978-3-319-45823-6\_90. URL [https://link.springer.com/chapter/10.1007/978-3-319-45823-6\\_90](https://link.springer.com/chapter/10.1007/978-3-319-45823-6_90).
- Pascal Kerschke, Lars Kotthoff, Jakob Bossek, Holger Hendrik Hoos, and Heike Trautmann. Leveraging TSP Solver Complementarity through Machine Learning. *Evolutionary Computation Journal (ECJ)*, pages 1 – 24, August 2017. doi: 10.1162/evco.a\_00215. URL [https://www.mitpressjournals.org/doi/abs/10.1162/evco.a\\_00215](https://www.mitpressjournals.org/doi/abs/10.1162/evco.a_00215).
- Pascal Kerschke, Hao Wang, Mike Preuss, Christian Grimme, André H. Deutz, Heike Trautmann, and Michael T. M. Emmerich. Search Dynamics on Multimodal Multi-Objective Problems. under review.
- Lars Kotthoff, Pascal Kerschke, Holger Hendrik Hoos, and Heike Trautmann. Improving the State of the Art in Inexact TSP Solving Using Per-Instance Algorithm Selection. In Clarisse Dhaenens, Laetitia Jourdan, and Marie-Éléonore Marmion, editors, *Proceedings of 9th International Conference on Learning and Intelligent Optimization (LION)*, volume 8994 of *Lecture Notes in Computer Science (LNCS)*, pages 202 – 217. Springer, January 2015. doi: 10.1007/978-3-319-19084-6\_18. URL [https://link.springer.com/chapter/10.1007/978-3-319-19084-6\\_18](https://link.springer.com/chapter/10.1007/978-3-319-19084-6_18).
- Lars Kotthoff. Algorithm Selection for Combinatorial Search Problems: A Survey. *AI Magazine*, 35(3):48 – 60, 2014. doi: 10.1609/aimag.v35i3.2460. URL <https://aaai.org/ojs/index.php/aimagazine/article/view/2460>.
- Ilya Loshchilov, Marc Schoenauer, and Michèle Sebag. Bi-Population CMA-ES Algorithms with Surrogate Models and Line Searches. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1177 – 1184. ACM, July 2013. doi: 10.1145/2464576.2482696. URL <http://dl.acm.org/citation.cfm?doid=2464576.2482696>.
- MATLAB. *Version 8.2.0 (R2013b)*. The MathWorks Inc., Natick, MA, USA, 2013.

- Walter Mebane Jr. and Jasjeet Sekhon. Genetic Optimization Using Derivatives: The rgenoud Package for R. *Journal of Statistical Software (JSS)*, 42(11):1 – 26, June 2011. doi: 10.18637/jss.v042.i11. URL <https://www.jstatsoft.org/v042/i11>.
- Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature (PPSN XI)*, Lecture Notes in Computer Science (LNCS), pages 71 – 80. Springer, September 2010. doi: 10.1007/978-3-642-15844-5\_8. URL [https://link.springer.com/chapter/10.1007/978-3-642-15844-5\\_8](https://link.springer.com/chapter/10.1007/978-3-642-15844-5_8).
- Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory Landscape Analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 829 – 836. ACM, July 2011. doi: 10.1145/2001576.2001690. URL <http://dl.acm.org/citation.cfm?doid=2001576.2001690>.
- Olaf Mersmann, Bernd Bischl, Heike Trautmann, Markus Wagner, Jakob Bossek, and Frank Neumann. A Novel Feature-Based Approach to Characterize Algorithm Performance for the Traveling Salesperson Problem. *Annals of Mathematics and Artificial Intelligence*, 69:151 – 182, October 2013. doi: 10.1007/s10472-013-9341-2. URL <https://link.springer.com/article/10.1007/s10472-013-9341-2>.
- Mario Andrés Muñoz Acosta, Michael Kirley, and Saman K. Halgamuge. Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content. *IEEE Transactions on Evolutionary Computation (TEVC)*, 19(1):74 – 87, 2015a. doi: 10.1109/TEVC.2014.2302006. URL <http://ieeexplore.ieee.org/document/6719480/>.
- Mario Andrés Muñoz Acosta, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. Algorithm Selection for Black-Box Continuous Optimization Problems: A Survey on Methods and Challenges. *Journal of Information Sciences (JIS)*, 317:224 – 245, October 2015b. doi: 10.1016/j.ins.2015.05.010. URL <http://www.sciencedirect.com/science/article/pii/S0020025515003680>.
- Yuichi Nagata and Shigenobu Kobayashi. A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem. *INFORMS Journal on Computing*, 25(2): 346 – 363, 2013. doi: 10.1287/ijoc.1120.0506. URL <http://dl.acm.org/citation.cfm?id=2466704>.
- Josef Pihera and Nysret Musliu. Application of Machine Learning to Algorithm Selection for TSP. In *Proceedings of the IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, November 2014. doi: 10.1109/ICTAI.2014.18. URL [https://www.dbai.tuwien.ac.at/staff/musliu/art\\_ictai\\_cam.pdf](https://www.dbai.tuwien.ac.at/staff/musliu/art_ictai_cam.pdf).
- R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL <http://www.R-project.org/>.
- Singiresu S. Rao. *Nonlinear Programming II: Unconstrained Optimization Techniques*, pages 301 – 379. John Wiley & Sons, 2009. doi: 10.1002/9780470549124.ch6. URL <http://dx.doi.org/10.1002/9780470549124.ch6>.

- John Rischard Rice. The Algorithm Selection Problem. *Advances in Computers*, 15:65 – 118, 1976. doi: 10.1016/S0065-2458(08)60520-3. URL <http://www.sciencedirect.com/science/article/pii/S0065245808605203>.
- Alexander Hendrik George Rinnooy Kan and Gerrit Theodoor Timmer. Stochastic Global Optimization Methods Part II: Multi Level Methods. *Mathematical Programming*, 39(1): 57 – 78, 1987. doi: 10.1007/BF02592071. URL <https://link.springer.com/article/10.1007/BF02592071>.
- Rainer Storn and Kenneth Price. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization (JOGO)*, 11(4): 341 – 359, December 1997. doi: 10.1023/A:1008202821328. URL <https://link.springer.com/article/10.1023/A:1008202821328>.
- Constantino Tsallis and Daniel Adrián Stariolo. Generalized Simulated Annealing. *Physica A: Statistical Mechanics and its Applications*, 233(1-2):395 – 406, November 1996. doi: 10.1016/S0378-4371(96)00271-3. URL <http://www.sciencedirect.com/science/article/pii/S0378437196002713>.
- Sander van Rijn, Hao Wang, Matthijs van Leeuwen, and Thomas H. W. Bäck. Evolving the Structure of Evolution Strategies. In *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1 – 8. IEEE, December 2016. doi: 10.1109/SSCI.2016.7850138. URL <http://ieeexplore.ieee.org/document/7850138/>.
- Guido VanRossum and The Python Development Team. *The Python Language Reference – Release 3.5.0*. Python Software Foundation, 2015.
- Joaquin Vanschoren, Jan Nicolaas van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked Science in Machine Learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49 – 60, 2013. doi: 10.1145/2641190.2641198. URL <http://dl.acm.org/citation.cfm?doid=2641190.2641198>.
- Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995. doi: 10.1007/978-1-4757-2440-0. URL <http://www.springer.com/de/book/9781475724400>.
- Simon Wessing. *Two-Stage Methods for Multimodal Optimization*. PhD thesis, Technische Universität Dortmund, 2015. URL <http://hdl.handle.net/2003/34148>.
- Simon Wessing. *optproblems: Infrastructure to define optimization problems and some test problems for black-box optimization*, 2016. URL <https://pypi.python.org/pypi/optproblems>. Python-package version 0.6.
- Simon Wessing, Mike Preuss, and Günter Rudolph. Niching by Multiobjectivization with Neighbor Information: Trade-offs and Benefits. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 103 – 110. IEEE, June 2013. doi: 10.1109/CEC.2013.6557559. URL <http://ieeexplore.ieee.org/document/6557559/>.
- David Hilton Wolpert and William G. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation (TEVC)*, 1(1):67 – 82, April 1997. doi: 10.1109/4235.585893. URL <http://ieeexplore.ieee.org/document/585893/>.

Xiao-Feng Xie and Jiming Liu. Multiagent Optimization System for Solving the Traveling Salesman Problem (TSP). *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 39(2):489 – 502, April 2009. doi: 10.1109/TSMCB.2008.2006910. URL <http://ieeexplore.ieee.org/document/4717264/>.

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation Journal (ECJ)*, (2):173 – 195, June 2000. doi: 10.1162/106365600568202. URL <http://dl.acm.org/citation.cfm?id=1108876>.

**Appendix:**  
**Contributed Publications**

# Chapter A

## Characterizing the Global Structure of Continuous Black-Box Problems

### A.1 Cell Mapping Techniques for Exploratory Landscape Analysis ■

- Authors:
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Mike Preuss,  
Information Systems and Statistics, University of Münster, Germany
  - Carlos Ignacio Hernández Castellanos,  
Department of Computer Science, CINVESTAV, Mexico-City, Mexico
  - Oliver Schütze,  
Department of Computer Science, CINVESTAV, Mexico-City, Mexico
  - Jian-Qiao Sun,  
School of Engineering, University of California, Merced, USA
  - Christian Grimme,  
Information Systems and Statistics, University of Münster, Germany
  - Günter Rudolph,  
Department of Computer Science, TU Dortmund University, Germany
  - Bernd Bischl,  
Department of Statistics, TU Dortmund University, Germany
  - Heike Trautmann,  
Information Systems and Statistics, University of Münster, Germany
- Published in “EVOLVE – A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation V”, pages 115 – 131, Springer, 2014.
- Presented at the “EVOLVE 2014 International Conference” held from July 1 to 4, 2014 in Beijing, China.



- Abstract:

Exploratory Landscape Analysis is an effective and sophisticated approach to characterize the properties of continuous optimization problems. The overall aim is to exploit this knowledge to give recommendations of the individually best suited algorithm for unseen optimization problems. Recent research revealed a high potential of this methodology in this respect based on a set of well-defined, computable features which only requires a quite small sample of function evaluations. In this paper, new features based on the cell mapping concept are introduced and shown to improve the existing feature set in terms of predicting expert-designed high-level properties, such as the degree of multimodality or the global structure, for 2-dimensional single objective optimization problems.

- Keywords:

exploratory landscape analysis, cell mapping, black-box optimization, continuous optimization, single objective optimization, algorithm selection

## A.2 Detecting Funnel Structures By Means of Exploratory Landscape Analysis ■

- Authors:
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Mike Preuss,  
Information Systems and Statistics, University of Münster, Germany
  - Simon Wessing,  
Department of Computer Science, TU Dortmund University, Germany
  - Heike Trautmann,  
Information Systems and Statistics, University of Münster, Germany
- Published in “Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation (GECCO)”, pages 265 – 272, ACM, 2015.
- Presented at the “Genetic and Evolutionary Computation Conference (GECCO)” held from July 11 to 15, 2015 in Madrid, Spain.
- 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.
- Keywords:

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

### A.3 Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models ■

- Authors:
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Mike Preuss,  
Information Systems and Statistics, University of Münster, Germany
  - Simon Wessing,  
Department of Computer Science, TU Dortmund University, Germany
  - Heike Trautmann,  
Information Systems and Statistics, University of Münster, Germany
- Published in “Proceedings of the 18th Annual Conference on Genetic and Evolutionary Computation (GECCO)”, pages 229 – 236, ACM, 2016.
- Presented at the “Genetic and Evolutionary Computation Conference (GECCO)” held from July 20 to 24, 2016 in Denver, CO, USA.
- 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.
- Keywords:

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

## Chapter B

# Flacco – A Toolbox for Exploratory Landscape Analysis with R

### B.1 The R-Package FLACCO for Exploratory Landscape Analysis with Applications to Multi-Objective Optimization Problems ■

- Authors:
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Heike Trautmann,  
Information Systems and Statistics, University of Münster, Germany
- Published in “IEEE Congress on Evolutionary Computation (CEC)”, pages 5262 – 5269, IEEE, 2016.
- Presented at the “IEEE World Congress on Computational Intelligence (WCCI)” held from July 24 to 29, 2016 in Vancouver, BC, Canada.
- 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.

## B.2 flaccogui: Exploratory Landscape Analysis for Everyone ■

- Authors:
  - Christian Hanster,  
University of Münster, Germany
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
- Published in “Proceedings of the 19th Annual Conference on Genetic and Evolutionary Computation (GECCO) Companion”, pages 1215 – 1222, ACM, 2017.
- Presented at the “Genetic and Evolutionary Computation Conference (GECCO)” held from July 15 to 19, 2017 in Berlin, Germany.
- Abstract:

Finding the optimal solution for a given problem has always been an intriguing goal and a key for reaching this goal is sound knowledge of the problem at hand. In case of single-objective, continuous, global optimization problems, such knowledge can be gained by *Exploratory Landscape Analysis* (ELA), which computes features that quantify the problem’s landscape prior to optimization. Due to the various backgrounds of researches that developed such features, there nowadays exist numerous implementations of feature sets across multiple programming languages, which is a blessing and burden at the same time.

The recently developed R-package `flacco` takes multiple of these feature sets (from the different packages and languages) and combines them within a single R-package. While this is very beneficial for R-users, users of other programming languages are left out. Within this paper, we introduce `flaccogui`, a graphical user interface that does not only make `flacco` more user-friendly, but due to a platform-independent web-application also allows researchers that are not familiar with R to perform ELA and benefit of the advantages of `flacco`.
- Keywords:

Exploratory Landscape Analysis, Graphical User Interface, R-Package, Continuous Optimization, Automated Algorithm Selection

## B.3 Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package `flacco` ■

- Authors:

- Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany

- This manuscript is currently under review.

However, a [preprint of the above manuscript](#) is available on arXiv.

- Abstract:

Choosing the best-performing optimizer(s) out of a portfolio of optimization algorithms is usually a difficult and complex task. It gets even worse, if the underlying functions are unknown, i.e., so-called *Black-Box problems*, and function evaluations are considered to be expensive. In the case of continuous single-objective optimization problems, *Exploratory Landscape Analysis (ELA)* – a sophisticated and effective approach for characterizing the landscapes of such problems by means of numerical values before actually performing the optimization task itself – is advantageous. Unfortunately, until now it has been quite complicated to compute multiple ELA features simultaneously, as the corresponding code has been – if at all – spread across multiple platforms or at least across several packages within these platforms.

This article presents a broad summary of existing ELA approaches and introduces `flacco`, an R-package for feature-based landscape analysis of continuous and constrained optimization problems. Although its functions neither solve the optimization problem itself nor the related *Algorithm Selection Problem (ASP)*, it offers easy access to an essential ingredient of the ASP by providing a wide collection of ELA features on a single platform – even within a single package. In addition, `flacco` provides multiple visualization techniques, which enhance the understanding of some of these numerical features, and thereby make certain landscape properties more comprehensible. On top of that, we will introduce the package’s build-in, as well as web-hosted and hence platform-independent, graphical user interface (GUI), which facilitates the usage of the package – especially for people who are not familiar with R – making it a very convenient toolbox when working towards algorithm selection of continuous single-objective optimization problems.

- Keywords:

Exploratory Landscape Analysis, R-Package, Graphical User Interface, Single-Objective Optimization, Continuous Optimization, Algorithm Selection, Black-Box Optimization

## Chapter C

# Feature-Based Algorithm Selection from Optimizer Portfolios

### C.1 Improving the State of the Art in Inexact TSP Solving Using Per-Instance Algorithm Selection

- Authors:
  - Lars Kotthoff,  
Insight Centre for Data Analytics, Cork, Ireland
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Holger Hendrik Hoos,  
Department of Computer Science, University of British Columbia, Vancouver, Canada
  - Heike Trautmann,  
Information Systems and Statistics, University of Münster, Germany
- Published in “Learning and Intelligent OptimizatiON (LION)”, pages 202 – 217, Springer, 2015.
- Presented at the “Learning and Intelligent OptimizatiON Conference (LION 9)” held from January 12 to 15, 2015 in Lille, France.
- Abstract:

We investigate per-instance algorithm selection techniques for solving the Travelling Salesman Problem (TSP), based on the two state-of-the-art inexact TSP solvers, LKH and EAX. Our comprehensive experiments demonstrate that the solvers exhibit complementary performance across a diverse set of instances, and the potential for improving the state of the art by selecting between them is significant. Using TSP features from the literature as well as a set of novel features, we show that we can capitalise on this potential by building an efficient selector that achieves significant performance improvements in practice. Our selectors represent a significant improvement in the state-of-the-art in inexact TSP solving, and hence in the ability to find optimal solutions (without proof of optimality) for challenging TSP instances in practice.

## C.2 Leveraging TSP Solver Complementarity through Machine Learning

- Authors:
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Lars Kotthoff,  
Department of Computer Science, University of British Columbia, Vancouver, Canada
  - Jakob Bossek,  
Information Systems and Statistics, University of Münster, Germany
  - Holger Hendrik Hoos,  
Department of Computer Science, University of British Columbia, Vancouver, Canada
  - Heike Trautmann,  
Information Systems and Statistics, University of Münster, Germany
- Published in “*Evolutionary Computation*” Journal, pages 1 – 24, MIT Press, 2017.
- Abstract:

The Travelling Salesperson Problem (TSP) is one of the best-studied NP-hard problems. Over the years, many different solution approaches and solvers have been developed. For the first time, we directly compare five state-of-the-art inexact solvers – namely, LKH, EAX, restart variants of those, and MAOS – on a large set of well-known benchmark instances and demonstrate complementary performance, in that different instances may be solved most effectively by different algorithms. We leverage this complementarity to build an algorithm selector, which selects the best TSP solver on a per-instance basis and thus achieves significantly improved performance compared to the single best solver, representing an advance in the state of the art in solving the Euclidean TSP. Our in-depth analysis of the selectors provides insight into what drives this performance improvement.
- Keywords:

Travelling Salesperson Problem, automated algorithm selection, performance modeling, machine learning



## C.3 Automated Algorithm Selection on Continuous Black-Box Problems By Combining Exploratory Landscape Analysis and Machine Learning

- Authors:

- Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
- Heike Trautmann,  
Information Systems and Statistics, University of Münster, Germany

- This manuscript is currently under review.

However, a [preprint of the above manuscript](#) is available on arXiv.

- Abstract:

In this paper, we build upon previous work on designing informative and efficient *Exploratory Landscape Analysis* features for characterizing problems' landscapes and show their effectiveness in automatically constructing algorithm selection models in continuous black-box optimization problems.

Focussing on algorithm performance results of the COCO platform of several years, we construct a representative set of high-performing complementary solvers and present an algorithm selection model that manages to outperform the single best solver out of the portfolio by factor two. Acting on the assumption that the function set of the *Black-Box Optimization Benchmark* is representative enough for practical applications the model allows for selecting the best suited optimization algorithm within the considered set for unseen problems *prior* to the optimization itself based on a small sample of function evaluations. Note that such a sample can even be reused for the initial algorithm population so that feature costs become negligible.

- Keywords:

Automated Algorithm Selection, Black-Box Optimization, Exploratory Landscape Analysis, Machine Learning, Single-Objective Continuous Optimization

# Chapter D

## Platforms for Collaborative Research on Algorithm Selection and Machine Learning

### D.1 ASlib: A Benchmark Library for Algorithm Selection



- Authors:
  - Bernd Bischl,  
Department of Statistics, LMU Munich, Germany
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Lars Kotthoff,  
Department of Computer Science, University of British Columbia, Vancouver, Canada
  - Marius Lindauer,  
Department of Computer Science, University of Freiburg, Germany
  - Yuri Malitsky,  
IBM Research, United States
  - Alexandre Fréchette,  
Department of Computer Science, University of British Columbia, Vancouver, Canada
  - Holger Hendrik Hoos,  
Department of Computer Science, University of British Columbia, Vancouver, Canada
  - Frank Hutter,  
Department of Computer Science, University of Freiburg, Germany
  - Kevin Leyton-Brown,  
Department of Computer Science, University of British Columbia, Vancouver, Canada
  - Kevin Tierney,  
Department of Business Information Systems, University of Paderborn, Germany
  - Joaquin Vanschoren,  
Mathematics & Computer Science, Eindhoven University of Technology, Netherlands

- Published in “Artificial Intelligence” Journal, Volume 237, pages 41 – 58, Elsevier, 2016.
- Abstract:

The task of algorithm selection involves choosing an algorithm from a set of algorithms on a per-instance basis in order to exploit the varying performance of algorithms over a set of instances. The algorithm selection problem is attracting increasing attention from researchers and practitioners in AI. Years of fruitful applications in a number of domains have resulted in a large amount of data, but the community lacks a standard format or repository for this data. This situation makes it difficult to share and compare different approaches effectively, as is done in other, more established fields. It also unnecessarily hinders new researchers who want to work in this area. To address this problem, we introduce a standardized format for representing algorithm selection scenarios and a repository that contains a growing number of data sets from the literature. Our format has been designed to be able to express a wide variety of different scenarios. To demonstrate the breadth and power of our platform, we describe a study that builds and evaluates algorithm selection models through a common interface. The results display the potential of algorithm selection to achieve significant performance improvements across a broad range of problems and algorithms.
- Keywords:

Algorithm selection, Machine learning, Empirical performance estimation

## D.2 OpenML: An R Package to Connect to the Machine Learning Platform OpenML

- Authors:
  - Giuseppe Casalicchio,  
Department of Statistics, LMU Munich, Germany
  - Jakob Bossek,  
Information Systems and Statistics, University of Münster, Germany
  - Michel Lang,  
Department of Statistics, TU Dortmund University, Germany
  - Dominik Kirchhoff,  
Dortmund University of Applied Sciences and Arts, Germany
  - Pascal Kerschke,  
Information Systems and Statistics, University of Münster, Germany
  - Benjamin Hofner,  
Section of Biostatistics, Paul-Ehrlich-Institut, Langen, Germany
  - Heidi Seibold,  
Epidemiology, Biostatistics & Prevention Institute, University of Zurich, Switzerland
  - Joaquin Vanschoren,  
Mathematics & Computer Science, Eindhoven University of Technology, Netherlands
  - Bernd Bischl,  
Department of Statistics, LMU Munich, Germany
- Published in *Journal of “Computational Statistics”*, pages 1 – 15, Springer, 2017.
- Abstract:

OpenML is an online machine learning platform where researchers can easily share data, machine learning tasks and experiments as well as organize them online to work and collaborate more efficiently. In this paper, we present an R package to interface with the OpenML platform and illustrate its usage in combination with the machine learning R package `mlr` (Bischl et al., 2016c). We show how the `OpenML` package allows R users to easily search, download and upload data sets and machine learning tasks. Furthermore, we also show how to upload results of experiments, share them with others and download results from other users. Beyond ensuring reproducibility of results, the OpenML platform automates much of the drudge work, speeds up research, facilitates collaboration and increases the users’ visibility online.
- Keywords:

Databases, Machine learning, R, Reproducible Research