Towards Online Landscape-Aware Algorithm Selection in Numerical Black-Box Optimization

Anja Jankovic

Thèse de doctorat en informatique
de Sorbonne Université

École Doctorale Informatique, Télécommunications et
Électronique

Towards Online Landscape-Aware Algorithm Selection in Numerical Black-Box Optimization

---

par
Anja Jankovic

présentée et soutenue publiquement
pour obtenir le grade de
DOCTEUR de SORBONNE UNIVERSITÉ

Date provisoire de soutenance: le 17 décembre 2021

Directrice de thèse

Dr.-Ing. Carola Doerr
Sorbonne Université, CNRS
Chargée de recherche

Jury

Dr. Marie-Éléonore Kessaci
Université de Lille, CRIStAL
Rapportrice – Maîtresse de conférences

Prof. Dr. Heike Trautmann
University of Münster, Department of Information Systems
Rapportrice – Professeure des universités

Prof. Dr. Jamal Atif
Université Paris-Dauphine, LAMSADE
Examinateur – Professeur des universités

Prof. Dr. Evripidis Bampis
Sorbonne Université, LIP6
Examinateur – Professeur des universités

Dr. Christoph Dürr
Sorbonne Université, CNRS
Examinateur – Directeur de recherche

Dr. Alberto Tonda
INRAE, AgroParisTech, EKINOCS
Examinateur – Chargé de recherche

Abstract

Black-box optimization algorithms (BBOAs) are conceived for settings in which exact problem formulations are non-existent or inaccessible, or in which problems are too complex to be solved analytically, thus requiring users to treat it as a black box. In those scenarios, BBOAs are essentially the only means of finding a good solution to such a problem. Due to their general applicability, BBOAs can exhibit different behaviors when optimizing different types of problems. This yields a meta-optimization problem of choosing the best suited algorithm for a particular problem at hand, called the *algorithm selection* problem.

By reason of inherent bias and limited knowledge about the complex relationship between algorithms, problems, and performance, a manual selection of the algorithms is undesirable. In consequence, the vision of automating the selection process has quickly gained traction in the evolutionary computation community. One prominent way of doing so is via so-called *landscape-aware algorithm selection*, where the choice of the algorithm is based on predicting its performance by means of numerical problem instance representations called *features*. There exists a large body of work in this very domain. However, a key challenge that landscape-aware algorithm selection faces is the computational overhead of extracting the features, a step typically designed to precede the actual optimization, which reduces the computational budget that can be allocated to the optimization algorithm.

In this thesis, we propose a novel *trajectory-based* landscape-aware algorithm selection approach which incorporates the feature extraction step within the optimization process. We show that the features computed using the search trajectory samples can lead to very robust and reliable predictions of algorithm performance, and consequently to powerful algorithm selection models built atop. We also present several preparatory analyses, including a novel perspective of combining two complementary regression strategies that outperforms any of the classical, single regression models, to amplify the quality of the final selector.

Résumé

Les algorithmes d'optimisation de boîte noire (BBOA) sont conçus pour des environnements dans lesquels les formulations exactes de problèmes sont inexistantes ou inaccessibles, ou dans lesquels les problèmes sont trop complexes pour être résolus analytiquement, obligeant ainsi les utilisateurs à les traiter comme une boîte noire. Dans ces scénarios, les BBOA sont essentiellement le seul moyen de trouver une bonne solution à un tel problème. En raison de leur applicabilité générale, les BBOA peuvent présenter des comportements différents lors de l'optimisation de différents types de problèmes. Cela donne un problème de méta-optimisation consistant à choisir l'algorithme le mieux adapté à un problème particulier, appelé problème de *sélection d'algorithmes*.

En raison du biais inhérent et des connaissances limitées sur la relation complexe entre les algorithmes, les problèmes et les performances, une sélection manuelle des algorithmes n'est pas souhaitable. En conséquence, la vision d'automatiser le processus de sélection a rapidement gagné du terrain dans la communauté. Un moyen important de le faire est ce que l'on appelle la *sélection d'algorithmes tenant compte du paysage*, où le choix de l'algorithme est basé sur la prédiction de ses performances au moyen de représentations numériques d'instances de problèmes appelées *caractéristiques*. Il existe un grand nombre de travaux dans ce domaine. Cependant, un défi clé auquel la sélection d'algorithmes tenant compte du paysage est confrontée est le coût de calcul de l'extraction des caractéristiques, une étape qui précède l'optimisation, ce qui réduit le budget pouvant être alloué à l'algorithme d'optimisation.

Dans cette thèse, nous proposons une nouvelle approche de sélection d'algorithmes tenant compte du paysage *basée sur la trajectoire* qui intègre l'étape d'extraction de caractéristiques dans le processus d'optimisation. Nous montrons que les caractéristiques calculées à l'aide des échantillons de trajectoire de recherche peuvent conduire à des prédictions très robustes et fiables des performances des algorithmes, et par conséquent à de puissants modèles de sélection d'algorithmes construits dessus. Nous présentons également plusieurs analyses préparatoires, y compris une nouvelle perspective de combinaison de deux stratégies de régression complémentaires qui surpasse n'importe lequel des modèles classiques de régression simple, pour amplifier la qualité du sélecteur final.

Acknowledgments

My PhD journey has been the most beautiful, exciting, fulfilling, yet challenging experience of my life. This thesis is nothing but the essence of the past three and a half years squeezed into a manuscript. Although I am very proud of myself to have grown as a person during this time and to have accomplished becoming a Doctor of Science, this whole story would never in a million years have been possible if it had not been for the people, old and new, that were by my side every step of the way.

The person whom I am indebted for life is Carola Doerr, my thesis supervisor. I am forever grateful that our paths even crossed in this universe. She helped me shape myself as a researcher with endless patience and kindness. Her sharp mind and warm heart are the best combination one can come across in academia. Carola helped me whenever I was feeling stuck or down, and genuinely celebrated my little achievements throughout this journey. I still to this day cannot believe my luck to have had her as my supervisor, and I am humbled to now be able to call her a true friend.

I thank my collaborators for their enthusiasm, ideas and discussions that have led to very nice publications. I also thank the whole RO team at LIP6 for being a second family to me all these years. Without them, my life would be far more boring, and I would not have experienced what it was like to be a part of a true research environment, like the one we have at LIP6. Thank you from the bottom of my heart for making me feel at home.

Nevertheless, there are two people that I am happy to call friends, albeit first getting to know them as colleagues: Martin Krejca and Elena Raponi. I do not know what I would do without you, guys! Thank you for the precious feedback on my thesis, Martin, for our countless weird discussions in the office, for understanding my unpredictable moods and handling them exquisitely, and for keeping up with me in general. Elena, we share the same soul, but different bodies. I think you already know everything that I would say, and you would even say it much better, so I am not going to get even more pathetic here. I owe you a lot, both of you.

Outside of the research context, my family and friends are the ones that kept me going in the worst of times, making sure to transform it into the best of times. My friends: Stefan, Milica, Saša, Uroš, Dunja, Mia, Emilija (with her baby, my godson!), Anastasija, Vanja… the list does not end here! You know who you are, and you know I love you a lot.

To my family (the whole tribe), I know this period has been hard for you at times, especially when you were witnessing my struggles. I think there is nobody in this world prouder of me than they are. I will not list all of them here, except my brother Lazar

and my sister Marija. They have always been successful in keeping the family dynamics as close as to what it had been when we were children. Thanks to them, I never lost this crucial, child-like part of myself, no matter the circumstances. I love you all to the moon and back.

Finally, I dedicate this thesis to the most important people in my life: my mom and dad. I consider them to be living legends, each in their own right. I would not at all be here today if it was not for them, but much more importantly, I would not be the person I am, today and every day, without their unquestionable love, support, encouragement, critiques, conversations... the whole lot! I am sorry that I have not been around a lot, which is extremely hard for you, but at the same time, you are the ones that opened the doors for me and gave me strength to lead my life the way I want. Thank you for being my parents. *Volim vas.*

Contents

1 Introduction

Optimization is one of the central themes across many scientific fields, notably mathematics, computer science, and related technical disciplines. Optimization is also at the core of many real-world applications. Solving an optimization problem is typically not easy, as problems are often computationally hard or otherwise resource-intensive; finding an exact optimal solution in a reasonable time is thus always a luxury demand. Furthermore, in many concrete cases formal problem modeling is impossible. The nonexistence (or inaccessibility) of an explicit mathematical formula, or the computational complexity of a problem, requires users to treat it as a *black box*.

In order to tackle problems in these circumstances, we resort to so-called **black-box optimization** techniques. They comprise algorithms, also known as *heuristics*, that are able to generate good, but not necessarily optimal, solutions via limited number of operations, such as sampling and evaluating new candidates, and then use the obtained knowledge to steer the search towards more promising alternatives, proceeding in iterations until eventually recommending their best estimate for a good solution. This framework allows for saving precious time and computational resources, and it works surprisingly well for many different types of problems.

A plethora of different black-box optimization algorithms has been developed to this day. Their behaviors typically vary fundamentally from one approach to another. For example, *Bayesian optimization techniques* [FSK08] approximate the true problem instance via a surrogate function which, once optimized, tells the algorithm which solution candidate to consider in the next step; *direct search methods*, such as the Nelder-Mead algorithm [NM65] and Powell algorithm [Pow64], examine candidate solution in a sequential fashion and compare each newly estimated solution quality with the quality of the best found solution so far, selecting the better of the two; *evolution strategies* [ES15], in their own right, apply (stochastic) mutation, recombination, and selection operators to a population of individuals containing candidate solutions in order to evolve iteratively better and better solutions.

Different black-box optimization algorithms have complementary strengths and weaknesses due to their different behaviors. Hence, to be able to achieve peak performance, this large pool of algorithmic options induces a challenging meta-optimization problem: **how does one select the most efficient algorithm for a given problem instance?**

1.1 Algorithm Selection

For a long time, algorithm selection was addressed manually by relying solely on expert knowledge of both the problem and the algorithm. The main disadvantage of this algorithm selection approach is not only that it requires an outstanding level of expertise, but, more importantly, that it can be highly prone to human bias. Therefore, the automation of the selection process has become a vision that quickly gained traction within the research community. Various research routes have since been explored in order to achieve efficient automated techniques.

The most basic approach in **automated algorithm selection** consists of using *a priori* available high-level problem characteristics to recommend the most appropriate algorithm via so-called *passive* selection techniques [MRW+21]. In contrast, *active* selection approaches [MBT+11; ME13; MKH15] operate within a *bet-and-run* framework, where several algorithms are run on an instance for some time and then all but the best one are stopped. Lastly, in recent years, significant research effort has been put towards designing algorithm selection approaches comprising an important *preprocessing* step that consists of extracting knowledge about the problem instance beforehand and using this knowledge to base the decision of which algorithm to use in the particular situation.

This thesis is concerned with the last of these perspectives, also known as **landscape-aware algorithm selection** [BBH+19; KT19; MKH12]. The term *landscape* here stands for the fitness landscape of a function, i.e. a set of function evaluations corresponding to a sample of observations in the search space. The landscape properties of a problem are first automatically computed as low-level features and are then used as a basis for further selecting an algorithm. The standard pipeline of this approach is shown in Figure 1.1. Importantly, we note that the common landscape-aware algorithm selection framework relies on using machine learning tools, notably supervised learning techniques, for designing models that recommend the most appropriate algorithm in the last phase of the pipeline.

1.2 Exploratory Landscape Analysis

Research addressing efficient ways to characterize problem instances via low-level feature approximations is subsumed under the umbrella term **exploratory landscape analysis (ELA)** [MBT+11]. These low-level features are computed based on a (random or carefully chosen) sample of observations and their function evaluations from the given problem instance. Information carried by the features allows to select an algorithm that fits well to the particular problem *instance* we aim to solve. This perspective is known as *per-instance algorithm selection*. Classically, the landscape-aware algorithm selection pipeline requires that this feature extraction is performed in a preliminary

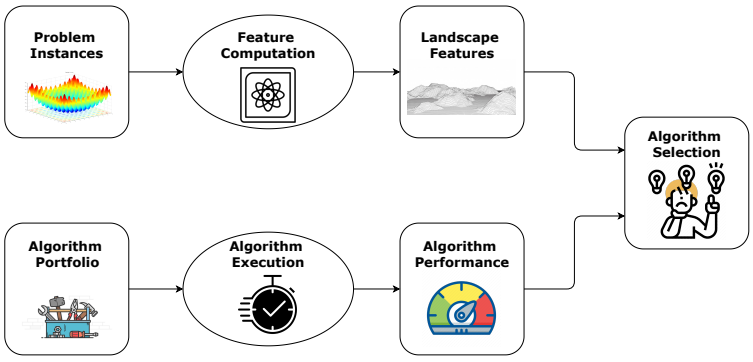


Figure 1.1: High-level schematic representation of the structure of landscape-aware algorithm selection pipeline.

phase, which introduces a significant additional cost in terms of function evaluations on top of the cost needed for the algorithm selector itself.

This overhead cost is of huge concern. A classical recommendation for computing reliable and robust feature values is $50d$ function evaluations dedicated to the preprocessing step [BDS+17; KPW+16], where d denotes the dimensionality of the problem. Nonetheless, this feature extraction budget is still outstandingly large for many practical applications; moreover, it is always considered to be independent from the budget of the optimization process. In turn, these observations create room for additional improvement by reducing this cost to a greater extent.

1.3 Key Objective of the Thesis

This thesis studies how to further save resources in the algorithm selection process. **We balance out the aforementioned cost of the feature extraction step by incorporating it within the optimization process.** We do so by:

1. running a default algorithm on problem instances,
2. extracting features based on the algorithm's search trajectory samples,
3. using this information to predict performances of the different algorithms in the portfolio,
4. switching to a potentially better suited algorithm for the problem at hand, based on the predictions.

We show that this paradigm shift towards **trajectory-based landscape-aware algorithm selection** is indeed highly promising. We note that the terms *trajectory-based* and *online* landscape-aware algorithm selection will be used interchangeably in the remainder of the thesis.

Our work involves a performance prediction step as a foundation upon which we build the algorithm selector. To this end, we opt for regression models (rather than classification ones), as we are interested in targeting concrete performance values. The general pipeline is then expanded by this additional step and is presented in Figure 1.2.

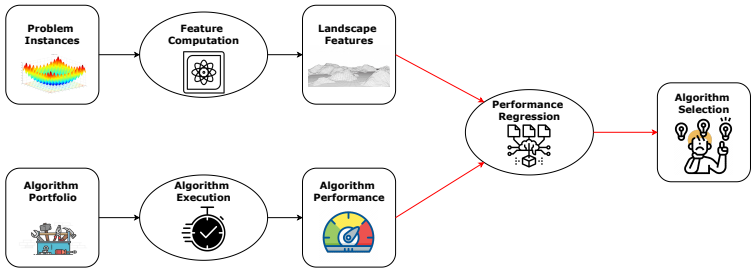


Figure 1.2: High-level schematic representation of the structure of landscape-aware algorithm selection pipeline, where the algorithm selection is based on **predicted performance values**.

Generally, when speaking of algorithm performances, most previous works in the context of landscape-aware algorithm selection have been undertaken in the fixed-target setting, in which the goal is to minimize the number of function evaluations needed to find a solution of a predefined target value. However, we often face a different optimization scenario in practice, where time is of vital importance, and computational and financial costs associated to time can be overwhelming if we aim for some fixed quality solution.

This is precisely the reason why we choose to shift the viewpoint for our work, and we thus place ourselves throughout this thesis in the **fixed-budget context**. Here, algorithm performance is measured as expected solution quality after spending a predetermined budget of function evaluations we have at our disposal. We thus translate the landscape-aware (i.e., ELA-based) algorithm selection approach to the fixed-budget setting, which has not been systematically analyzed so far.

In order to set up the correct trajectory-based framework, the thesis also introduces several preparatory steps. We look into the design of more accurate algorithm selection approaches building on top of the state-of-the-art, by combining supervised models that predict differently with different precision and by personalizing those models to the particular problem instance we are interested in solving. We also provide a detailed

empirical performance analysis of machine learning tools needed for the algorithm selection in the desired context.

1.4 Outline of the Thesis

The general structure of the thesis is presented in this section.

Chapter 2 gives an overview of the contributions of the thesis via summaries of publications that the thesis is based on. The remainder of the thesis is organized in two parts.

In Part I, we are concerned with the conceptual background for each of the components in the algorithm selection pipeline. We first introduce important notions from the black-box optimization domain, notably the core principle of black-box optimization algorithms, their performance measures, and the nature of numerical black-box optimization problems in Chapter 3. Techniques of representing the problem instances through numerical values, which are subsumed under the term of exploratory landscape analysis, are presented in Chapter 4. We then define the algorithm selection problem, giving an overview of the state of the art and highlighting the key challenges in Chapter 5.

The detailed contributions of the thesis are found in Part II. Chapter 6 introduces a combined regression strategy for algorithm performance prediction required to select algorithms in the landscape-aware context. We are concerned with comparing the quality of different regression models based on their parametrization in Chapter 7. A novel performance prediction approach based on personalizing regression models to the problems is proposed in Chapter 8. We then give a brief analysis of how problem landscapes look as seen by the optimizer in Chapter 9. Our novel trajectory-based (i.e., online) algorithm selection framework is finally introduced in Chapter 10.

Lastly, Chapter 11 opens some promising avenues and gives motivation for future work in this line of research.

2 Contributions of the Thesis

This thesis is based on the following publications, listed here in reverse chronological order:

- **Anja Jankovic**, Gorjan Popovski, Tome Eftimov and Carola Doerr. The Impact of Hyper-Parameter Tuning for Landscape-Aware Performance Regression and Algorithm Selection. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21)*, pages 687–696. ACM, 2021. [JPE+21]
- Tome Eftimov, **Anja Jankovic**, Gorjan Popovski, Carola Doerr and Peter Korošec. Personalizing Performance Regression Models to Black-Box Optimization Problems. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21)*, pages 669–677. ACM, 2021. [EJP+21]
- **Anja Jankovic**, Tome Eftimov and Carola Doerr. Towards Feature-Based Performance Regression Using Trajectory Data. In *Proc. of Applications of Evolutionary Computation (EvoApplications'21)*, volume 12694 of *Lecture Notes in Computer Science*, pages 601–617. Springer, 2021. [JED21]
- **Anja Jankovic** and Carola Doerr. Landscape-Aware Fixed-Budget Performance Regression and Algorithm Selection for Modular CMA-ES Variants. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'20)*, pages 841–849. ACM, 2020. [JD20]
- **Anja Jankovic** and Carola Doerr. Adaptive Landscape Analysis. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'19), Companion Material*, pages 2032–2035. ACM, 2019. [JD19]

In Sections 2.1 to 2.5, we give a brief summary of each publication, in the order in which they appear in the thesis.

2.1 Landscape-Aware Fixed-Budget Performance Regression and Algorithm Selection for Modular CMA-ES Variants [GECCO'20]

In this paper, we propose a novel automated algorithm selection approach based on two differently trained and suitably combined supervised learning models which underpin the recommendation of the most appropriate algorithm for the problem instance at hand. A common component in the state-of-the-art algorithm selection techniques are regression models which predict the performance of a given algorithm on a previously unseen problem instance.

In the context of numerical black-box optimization, such regression models build on ELA features, which quantify relevant characteristics of the problem. These measures can be then used in a rather straightforward way by a regression model that maps them to algorithm performances. First steps towards ELA-based performance regression have been made in the context of a fixed-target setting. In many applications, however, the user needs to select an algorithm that performs best within a given budget of function evaluations, as for numerous practical purposes with limited time or computational restrictions, a fixed-target setting is often off-limits.

Adopting this fixed-budget setting, we demonstrate that it is possible to achieve high-quality performance predictions with off-the-shelf supervised learning approaches, by **combining two differently trained regression models**: one trained on the *original* performance values of different algorithms, and the other trained on *log-scale* performance values. In the fixed-budget context, this is especially useful, as log-scale values of the data match the intuitive notion of the *distance level* of the solution to the optimum. We test this approach on a very challenging problem: algorithm selection on a portfolio of very similar algorithms, which we choose from the family of modular CMA-ES algorithms [RWL+16]. We show that the complementary algorithm selection approach achieves better performance quality compared to any of the standalone selectors (based either on the original data or on the log-scale data). The pipeline of the approach is illustrated in Figure 2.1.

2.2 The Impact of Hyper-Parameter Tuning for Landscape-Aware Performance Regression and Algorithm Selection [GECCO'21a]

This paper studies the differences in algorithm selection accuracy depending on the choice of the underlying machine learning model, and in particular depending on its **hyper-parameter configuration**. Despite a significantly growing number of applications, machine learning models are still often chosen in an *ad hoc* manner; this motivates

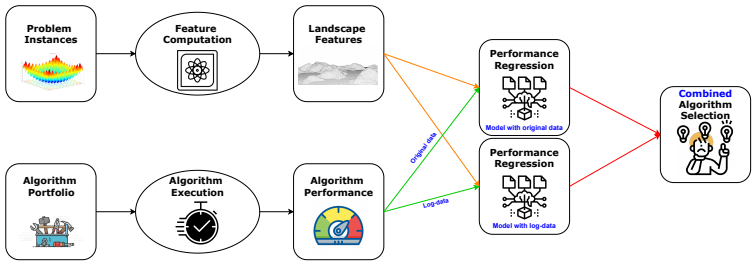


Figure 2.1: High-level schematic representation of the structure of our **novel combined** landscape-aware algorithm selection pipeline.

further investigation of the influence that the choice of the machine learning model can have on the performance prediction and, consequently, on the algorithm selection quality. We show in this work that three classical regression methods are able to achieve meaningful results for ELA-based algorithm selection. For those three model families – random forests, decision trees, and bagging decision trees – the quality of the regression models is highly impacted by the chosen hyper-parameters. This has significant effects also on the quality of the algorithm selectors that are built on top of these regressions.

After a preliminary study which led to discarding a few hundreds of tested models due to their very low accuracy, we restrict our attention to 30 differently parametrized models from the three aforementioned model families. Each of the models is then coupled with 2 complementary regression strategies, as described in Section 2.1. The so-obtained insights allowed us to derive guidelines for the tuning of the regression models and provide general recommendations for a more systematic use of classical machine learning models in landscape-aware algorithm selection. This paper being a first step in this direction, we point out that a number of open questions merit further investigation. Nevertheless, general results indicate that the choice of the machine learning model is a challenging step in the design of the algorithm selection pipeline and needs to be undertaken carefully.

2.3 Personalizing Performance Regression Models to Black-Box Optimization Problems [GECCO'21b]

This paper focuses on analyzing the possibility of **personalizing performance regression models** to specific types of optimization problems. Accurately predicting the performance of different optimization algorithms for previously unseen problem instances is crucial for high-performing algorithm selection techniques. In the context

of numerical black-box optimization, researchers and practitioners typically stick to standard choices of the machine learning models as a basis for algorithms selectors. These are, most frequently, off-the-shelf tools, which can seem rather naïve from the machine learning viewpoint. Thus, we are presented with a risk of not achieving the full potential of a machine learning technique if we disregard proper investigation of its actual suitability for the problems we encounter.

With this study, we bring to the attention of evolutionary computation community one way of increasing the suitability of machine learning tools for the problems at hand. Instead of aiming for a single model that works well across a whole set of possibly diverse problems, our personalized regression approach acknowledges that different models may suit different types of problems. Going one step further, we also investigate the impact of selecting not a single regression model per problem, but personalized ensembles. We test our approach on predicting the performance of numerical optimization heuristics on the BBOB benchmark suite and we show that this approach opens promising avenues for further increasing the regression quality and, consequently, the accuracy of the algorithm selector built atop.

2.4 Adaptive Landscape Analysis [GECCO'19]

This paper takes a deeper dive into the underlying properties of different black-box optimization problems. In order for the optimization process to be as efficient as possible, one must first recognize the nature of the problem at hand and then proceed to choose the algorithm exhibiting the best performance for that type of problem. A standard way of identifying similarities and differences between various problems lies in ELA feature extraction and representation of the problem via these numerical values.

We present here some first steps towards **adaptive landscape analysis**. Our approach is aimed at taking a closer look into how feature values evolve during the optimization process and whether this *local* feature information can be used to discriminate between different problems. The motivation of our work is to understand if and how one could exploit the information provided by local features to get a step closer to dynamic (also known as online) algorithm selection. The idea behind it is fairly simple, yet extremely promising in terms of gaining efficiency and performance quality; the choice of an algorithm is tracked and adapted during the optimization process, allowing for switching from one algorithm to a better suited one. The long-term goal of this analysis is to leverage local landscape information for adjusting the choice of the algorithm on the fly, i.e., during the optimization process itself.

We show that the local feature values differ drastically compared to the global ones, which gives a promising insight into how the problem landscape looks locally, as seen by the algorithm. These insights could turn out to be crucial for ultimately designing an algorithm selector model that operates in an online fashion.

2.5 Towards Feature-Based Performance Regression Using Trajectory Data [EvoAPPs'21]

The final paper investigates a novel framework where the performance regression model is based on the information obtained via a **search trajectory** of a default solver. The pipeline of this trajectory-based approach is shown in Figure 2.2. Existing ELA-based algorithm selection approaches require the approximation of problem features based on a significant number of samples. They are typically selected through uniform sampling or Latin hypercube designs, and they are computed in a problem-specific fashion, independently of the optimization process that occurs later. The evaluation of these points is costly, and the benefit of an ELA-based algorithm selection over a default algorithm must therefore be significant in order to pay it off.

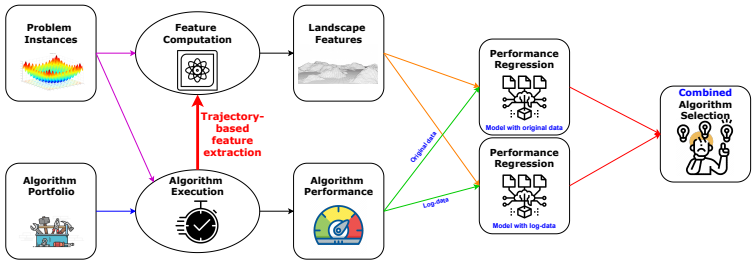


Figure 2.2: High-level schematic representation of the structure of our **novel trajectory-based** landscape-aware algorithm selection pipeline.

One could hope to by-pass the evaluations for the feature approximations by using the samples that a default algorithm would anyway perform, i.e., by using the points of the default algorithm's trajectory. We analyze here how well such an approach can work. Concretely, we test how accurately trajectory-based ELA approaches can predict the final solution quality of the particular default solver (the CMA-ES algorithm) after a fixed budget of function evaluations.

We observe that the loss of trajectory-based predictions can be surprisingly small, even negligible, compared to the classical global sampling approach, if the remaining budget for which solution quality shall be predicted is not too large. This important result fortifies our vision of trajectory-based algorithm selection being the future of dynamic algorithm selection. It goes without saying that a large pool of open questions are yet to be answered. Nonetheless, we note that the untapped potential within this research direction remains the central point of our research efforts in the imminent future.

Part I

The Background

3 Black-Box Optimization

In Chapters 3 to 5 we provide a more detailed overview of the components in the algorithm selection pipeline we adopt throughout the thesis (cf. Figure 1.2). In this first chapter of this background part, we briefly discuss the relevant black-box optimization techniques (cf. Section 3.1), the performance measures in the fixed-budget context that we consider in this thesis (cf. Section 3.2), and the problem collections that we use to evaluate our pipeline (cf. Section 3.3).

3.1 Black-Box Optimization Algorithms

Black-box optimization algorithms (BBOAs) are a wide-ranging family of adaptive sampling-based strategies, which adjust their behavior during the optimization process. There have been various attempts to provide an exhaustive classification of BBOAs in the literature [BLS13; BPS+01; SB18], which proved to be a challenging task in its own right, given the rather loosely defined boundaries between different algorithm classes.

The arguably largest class of BBOAs are iterative optimization heuristics. They are initialized by selecting a set of candidate solutions (i.e., an initial *population*) that are evaluated in a first iteration. Once the quality of these search points is known, a second set of solution candidates is selected and evaluated. The algorithm proceeds in this manner, alternating between selecting and evaluating solution candidates iteration by iteration, until some termination criterion is met, e.g., when a sufficiently good solution has been found, when a time (or function evaluation) budget is exhausted, when no progress has been observed for some time, or when the diversity of the solutions has fallen below some threshold. With each iteration, the algorithm collects more information about the problem instance at hand, which it uses to steer the search towards the most promising regions in the decision space. The simplified blueprint of a black-box algorithm is shown in Algorithm 1.

Many, but not all, iterative optimization heuristics are randomized. Furthermore, most of these heuristics were developed with a particular application in mind, and some have paved their way into a much broader field than what they had been initially conceived for. One of such universally acknowledged strategies has served as a default solver in many use-cases across the field due to its state-of-the-art performance, and was thus also selected for the purpose of this thesis. We introduce it in Section 3.1.1. Due to the many variants of this default solver that have been developed, a novel modular framework based on it has also been proposed recently and presents another important algorithm family that we introduce in Section 3.1.2.

Algorithm 1: Blueprint of a black-box optimization algorithm

```
choose an initial sampling distribution;
from this distribution, sample the initial set of search points (initial
  population) X;
evaluate the initial population X;
while termination criterion not reached do
    update the sampling distribution based on previous evaluations;
    from this new distribution, sample a new population of solution
     candidates;
    evaluate the new population;
output the best search points seen so far
```

3.1.1 CMA-ES

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [HO01] is one of the most widely applied algorithms for real-valued single-objective optimization problems (the original paper has so far reached more than 4000 citations). Thanks to its invariance properties, some of its default parameter values can be tuned using a rather small set of test functions, while nevertheless providing robust performances on a large variety of problems, from analytical benchmark functions to many real-world applications.

The core ideas behind the CMA-ES are fairly simple. In every iteration, candidate solutions are sampled from a multivariate normal distribution $N(m, C)$, where m stands for the distribution mean, and C denotes the covariance matrix that adapts and rotates the distribution. The best candidates are then selected for shifting the distribution mean, and the distribution is adapted to an ellipsoid with information about evolution paths; i.e., the pairwise dependencies between variables stored in the covariance matrix are continuously updated by incorporating previous evolution paths. Finally, the distribution is centered around the new distribution mean for the next generation. The algorithm runs until a termination criterion is met and returns the best individual observed so far.

The CMA-ES works *derivative-free*, i.e., it does not need access to gradients or higher-order derivatives. Furthermore, it is *comparison-based*, i.e., it only uses the ranking between candidate solutions being exploited for learning the sample distribution. It performs significantly well on ill-conditioned and non-separable problems [HO01], but also on non-convex, highly multi-modal and noisy problems. Figure 3.1 illustrates the behavior of the CMA-ES.

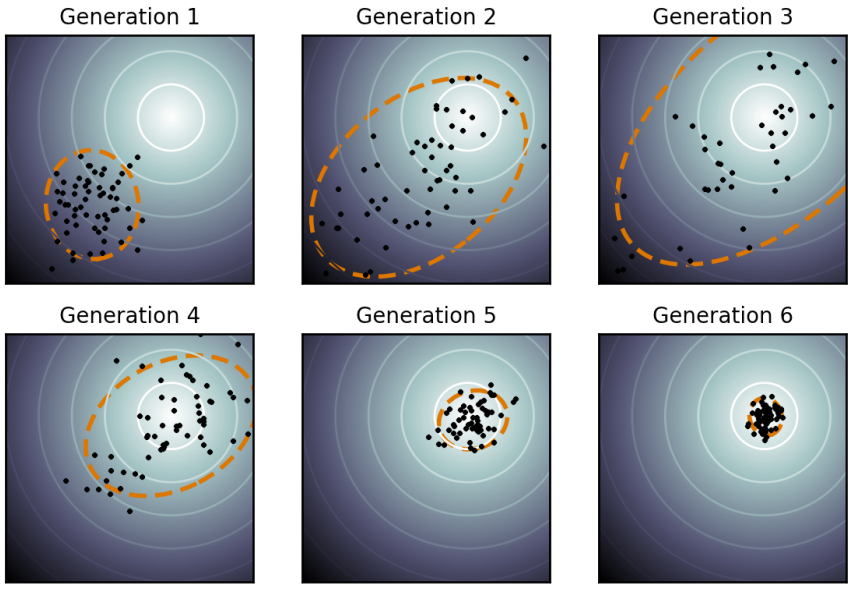


Figure 3.1: Illustration of the CMA-ES algorithm. Taken from Wikimedia Commons [Com08].

3.1.2 Modular CMA-ES

Widespread use of the CMA-ES in the community has naturally led to its several adaptations. For some of them, empirical results have shown that they outperform the original version of the CMA-ES under different circumstances. One perspective that captures multiple alterations of the standard CMA-ES has been proposed in [RWL+16]. The *modular CMA-ES* is an algorithmic framework which allows to extract different functional modules based on underlying mechanisms of existing CMA-ES adaptations. For instance, by enabling or disabling modules such as orthogonal sampling, elitism, or choosing one of the available weighting schemes, we can create different variants of the CMA-ES (cf. Table 3.1 for a complete overview of modules).

Eleven modules have been incorporated within the framework, nine binary and two ternary, hence amounting to 4608 different CMA-ES variants in total. The framework can thus be considered as a family of very similar algorithms.

We note that an extension of the modular CMA-ES, adding in particular new boundary control methods and simplifying the implemented interface, has recently been proposed

in [NVW+21]. Given that it appeared only recently, our work uses only the original framework proposed in [RWL+16].

Table 3.1: Overview of the CMA-ES modules available in the framework. Row 9 specifies formulae for computing each weight w_j.

	Module name	0	1	2
1	Active Update [JA06]	off	on	-
2	Elitism	(μ, λ)	$(\mu + \lambda)$	-
3	Mirrored Sampling [BAH+10]	off	on	-
4	Orthogonal Sampling [WEB14]	off	on	-
5	Sequential Selection [BAH+10]	off	on	-
6	Threshold Convergence [PEB+15]	off	on	-
7	TPA [Han08]	off	on	-
8	Pairwise Selection [ABH11]	off	on	-
9	Recombination Weights [AJT05]	$log(\mu + \frac{1}{2}) - \frac{log(i)}{\sum_j w_j}$	$\frac{1}{\mu}$	-
10	Quasi-Gaussian Sampling	off	Sobol	Halton
11	Increasing Population [AH05; Han09]	off	IPOP	BIPOP

3.1.3 Additional Algorithms

While the abovementioned CMA-ES and its modular extension remain in the spotlight of the thesis, in Chapters 7 and 8 we consider a broader algorithm portfolio, based on the suggestion in [KT19]. We give here a brief overview of different solvers included in this investigation.

The twelve optimization algorithms from the considered portfolio can be grouped into four categories and are summarized below.

Deterministic Optimization Algorithms. The two solvers of this category are variants of the Brent-STEP algorithm [BP15]. The Brent-STEP performs axis-parallel searches and can choose the search dimension of the next iteration either using a round-robin (*BSrr*) [PB15] or a quadratic interpolation strategy (*BSqi*) [PB15].

Multi-Level Approaches. Most algorithms of this category stem from the *multi-level linkage method (MLSL)* [KT87; Pál13b]. MLSL is a stochastic, multi-start algorithm for global optimization that operates using random sampling and local searches. Besides MLSL, our portfolio also consists of several of its variants: *fmincon* [Pál13b] (an interior-point MLSL version for constrained linear problems), *fminunc* [Pál13b] (a quasi-Newton MLSL version which approximates the Hessian matrix using BFGS [Bro70]), and

HMLSL [Pál13b] (a hybrid MLSL variant which incorporates significant improvements within its sampling phase). Finally, this category contains the *multi-level coordinate search (MCS)* [HN99], which splits the search space into smaller portions and that performs local search procedures from the most promising portions. Each of the small portions contains an already known observation.

Variants of the CMA-ES. We include in the portfolio the CMA-ES version with cumulative step-size adaptation (*CMA-ES-CSA*) [Ata15]. Additionally, three more algorithms derived from the standard CMA-ES (presented above) are also found in the portfolio. They are *IPOP400D* [ABH13], a CMA-ES version with restarts, an increasing population size and a maximum of $400 \times (d + 2)$ function evaluations, where d stands for problem dimensionality; *HCMA* [LSS13b], a hybrid CMA which combines three different strategies (BIPOP self-adaptive surrogate-assisted CMA-ES [LSS13a], STEP [SSB94] and NEWUOA [Pow06]) to simultaneously benefit from their complementary strengths; and *BIPOP-CMA-ES* [LSS13a], a CMA-ES variant with special restart strategy switching between two population sizes – a small one (like the default CMA, but with more focused search) and a large one (that is progressively increased as in IPOP). This makes the BIPOP-CMA-ES perform well both on functions with many regularly or irregularly arranged local optima (the latter by frequently restarting with small populations). Note that this last algorithm replaces *SMAC-BBOB* [HHL13], which was a part of the original portfolio, due to reasons of unavailability of raw data.

Other Algorithms. The last solver is a multi-start, commercial heuristic called *OptQuest/NLP (OQNLP)* [Pál13b; ULP+07], which was initially designed to find the global optima of smooth constrained non linear problems (NLPs) and mixed integer non-linear programs (MINLPs). It uses the OptQuest Callable Library (OCL) [LM02] to generate the inital population of candidate solutions for a local NLP solver.

3.2 Algorithm Performance Measures

Throughout Chapters 1 and 2, and so far in Part I, we have mentioned several times that we are interested in *performances* of different algorithms, and yet we did not explicitly state how to assess this performance. The choice of the algorithm heavily depends on the performance measure, since an algorithm that works well with large computational budgets might not necessarily perform very well for small budgets.

A classical approach such as assessing the running time through the number of arithmetic operations cannot work with black-box optimizers; here, the most substantial computational resources are spent on evaluating candidate solutions. It is thus beneficial to use the *number of function evaluations* when speaking of black-box algorithms' running time. Moreover, as iterative optimization heuristics often employ randomization

when generating or selecting candidate solutions, their performance can differ on multiple levels: between different algorithm runs on one problem instance, between different instances of the same problem, let alone between different problems. Therefore, for empirical assessment, multiple algorithm runs need to be aggregated in a suitable way.

With respect to the nature of the performance space, we can essentially distinguish between two main axes. If we are interested in finding a solution of a certain quality, we typically consider the number of function evaluations needed to reach this *fixed target*. On the other hand, if the number of available function evaluations itself is limited, we opt for measuring the quality of the solution that can be obtained within this *fixed budget*. All these measures are then commonly aggregated for multiple runs (or even problem instances) via averaging or some other statistics.

Whereas in practical applications we very frequently encounter the need for setting the budget *a priori*, due to possibly expensive function evaluations that are to be performed, this fixed-budget scenario remains largely unexplored in the context of automated algorithm selection for black-box optimization. We thus adopt this particular setting in the thesis moving forward.

The performance measure that we consequently go for, in accordance to the fixed-budget setting, is the *target precision*, which measures the distance of the best found solution until the budget is exhausted to the optimal solution of a problem instance. Target precision, as we shall see in Part II, carries important intuition about how many so-called *distance levels* there are between some solution and the optimum. We are then able to conveniently incorporate said information into the relevant component of the algorithm selection pipeline to design a novel regression perspective (cf. Figure 2.1).

3.3 Problem Collections

Another important question that arises is how one algorithm's performance generalizes to different types of problems. Indeed, the algorithm may be highly tuned to the specific problem at hand. To reliably evaluate its search behavior, the algorithm needs to be tested on multiple different problem instances and problem classes. Therefore, a procedure called *benchmarking* is applied. A problem collection consists of a fixed set of problems, typically spanning over a wide spectrum of various problem classes. Any new algorithm is tested on exactly the same functions to allow for performance comparisons between different solvers. This thesis also relies of one such benchmark problem collection that is introduced below in more detail.

3.3.1 BBOB Test Suite

In the domain of numerical optimization, a platform called COCO (COmparing Continuous Optimizers) [HAR+20] has been established precisely for this purpose over the

last years. The COCO platform offers interfaces to run solvers on various test suites that consist of multiple benchmark functions. Those functions are used as black boxes for solvers (while still being explicitly known in the community), and they are designed to allow for meaningful interpretation of performance results. Moreover, the COCO platform provides exhaustive methods to trace and process performance data.

This thesis focuses on the COCO test suite called *Black-Box Optimization Benchmark (BBOB)* [HFR+09]. The BBOB problem collection consists of 24 noiseless, single-objective test functions that need to be minimized, with dimensionality (number of problem variables) commonly set to $d \in \{2, 3, 5, 10, 20, 40\}$. The functions are defined in a $[-5, 5]^d$ search space. They are grouped into five different problem classes, spanning across different values for properties such as modality, separability, conditioning and global structure (cf. Table 3.2). Different *instances* of each function are generated by rotating and translating the function in the objective space [Han09]. Figure 3.2 illustrates three BBOB functions in dimension 2.

Table 3.2: Noiseless BBOB functions grouped into five different classes.

Functions	Problem class
F1–F5	separable functions
F6–F9	functions with low conditioning
F10–F14	unimodal functions with high conditioning
F15–F19	multimodal functions with adequate global structure
F20–F24	multimodal functions with weak global structure

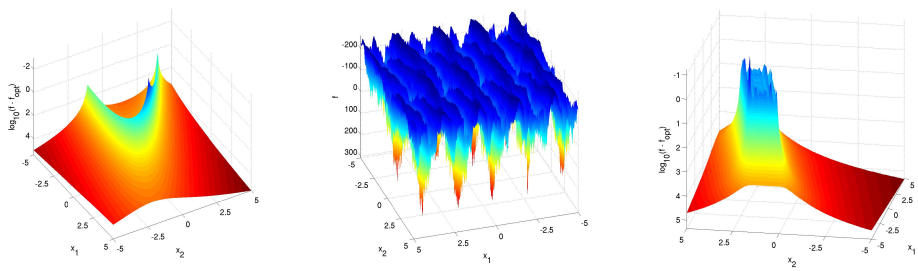


Figure 3.2: Examples of BBOB noiseless test bed function landscapes. Left: F9, the rotated Rosenbrock function (unimodal, low or moderate conditioning). Middle: F16, the Weierstrass function (multimodal, adequate global structure). Right: F20, the Schwefel function (multimodal, weak global structture). Taken from [HAF+10].

4 Exploratory Landscape Analysis

In order to represent the considered optimization problems in a suitable and useful way for the algorithm selection pipeline, we want to quantify different characteristics of the problem instances via appropriate measures. We do this by means of exploratory landscape analysis. In this chapter, we present the concept of exploratory landscape analysis as one of the core pipeline components (cf. Figure 1.2) in greater detail. We explain the notion of features and discuss main challenges with respect to feature computation in Section 4.1, and we present selected feature sets for the purpose of this thesis in Section 4.2. Lastly, we give a brief overview of a tool that allows for an automatic feature computation, which is an indispensable step towards building a landscape-aware algorithm selector (cf. Section 4.3).

4.1 ELA Features

In landscape-aware algorithm selection, algorithm recommendations are based on *features* of the problem instances, which are estimated from a finite set $\{(x, f(x))\}_{x \in D}$ of evaluated samples in a decision space D via so-called *exploratory landscape analysis (ELA)* [MBT+11]. Most research in this area focuses on the definition or the analysis of features that describe certain characteristics of the optimization problem, and their suitability for automated algorithm selection, configuration, and design [LW06; MBT+11; MKH15]. For instance, features can comprise information about problem ruggedness, neutrality, fitness-distance correlation, and presence of funnels [ME13].

For a given problem, ELA aims at measuring problem characteristics through functions that assign to each problem a vector of real numbers. To date, ELA has mostly focused on numerical optimization problems [BDS+17; Mal21], but the concepts have also been very successfully applied to some NP-hard combinatorial problems, such as satisfiability problem, AI planning, travelling salesperson problem and more [FVH+14; HXH+14; MPR12; OV16; PM14; XHH+08]. ELA has also been investigated in multi-objective settings [DLV+17; LDV+20; LVL+21].

These automatically computed features should ideally cover several facets. They should be:

- informative, so that they allow for a sufficient degree of discriminative power between different problem instances;
- interpretable, so that the values they carry enable maximum insight into the instance properties;

- cheaply computable, to ensure that the advantages obtained by ELA-based algorithm selection are not outweighed by the cost of their computation;
- generally applicable to a broad range of problem instances.

Common research questions in ELA concern the number of samples needed to accurately approximate feature values, the design and the selection of features that are descriptive and easy to approximate, and the possibility to use feature values to transfer learned policies from some instances to previously unseen ones.

Diverse sampling strategies can be employed when gathering search points for feature computation, e.g., uniform random sampling, Latin hypercube designs, as well as Sobol sequences. Moreover, feature values are shown to be sensitive to the chosen sampling strategy [RDD+20].

4.2 Choice of Features

In this thesis, we only consider features that do not require additional function evaluations for their computation, also referred to as *cheap features* [BDS+17]. They are computed using the fixed initial sample, while *expensive features*, in contrast, need additional sampling during the run, an overhead that makes them more inaccessible for practical use. Each feature set regroups several relevant features that quantify some information about the landscape. We now give a brief overview of the cheap feature sets we make use of throughout the rest of the thesis.

Classical ELA Feature Sets

Out of the six classical features sets originally proposed in [MBT+11], we opt for *y-Distribution*, *Levelset* and *Meta-Model* sets, as these sets do not make use of additional sampling during the feature computation process.

y-Distribution. This feature set approximates the distribution of the fitness values by means of several measures whose values are then compared to the normal distribution (the skewness (i.e., assymetry) and kurtosis (i.e., sharpness) of the objective function values, as well as the degree of peakedness, i.e., whether the distribution is rather flat or peaked compared to the normal one).

Levelset. This feature set works by splitting the initial data set into two classes via a specific objective level that serves as a threshold, predicting the position of the objective values of the data set with respect to the said threshold, and then extracting the relevant information by cross-validating mean misclassification errors of each classifier. Predictions are performed via different variants of discriminant analysis (LDA, QDA or MDA).

Meta-Model. This feature set fits linear and quadratic regression models (with or without interactions) to the initial data set and the relevant indicators are then carried by the adjusted coefficient of determination R^2 and related metrics.

Other Feature Sets

Dispersion. As ELA was getting traction in the community, many new feature sets have been introduced. One of them is the *dispersion* feature set [LW06], which compares pairwise distances of all points in the initial data set with the pairwise distances of the best points in the initial data set (according to the corresponding objective values). Depending on the size of the best points subset (2%, 5%, 10% or 25%), different comparisons are executed and their relevant values captured by the difference and the ratio of pairwise distance means/medians.

Information Content. Another feature set, proposed in [MKH15], measures the landscape's *Information Content* (i.e., smoothness, ruggedness, or neutrality) based on a random path across the problem's landscape, using the change in the objective values of neighboring points. The random path is constructed starting from an initial observation and then greedily walking to its not yet visited nearest neighbors.

Nearest-Better Clustering. Finally, the last feature set considered in this thesis is the *Nearest-Better Clustering* set [KPW+15]. It distinguishes between so-called "funnel" and "non-funnel" structures by comparing the sets of distances from all observations to their nearest neighbors, as well as their nearest better neighbors.

4.3 Feature Computation

A convenient way to compute landscape features is offered by the *flacco* toolbox [KT16]. This package, available in both R and Python, provides a unified interface to a collection of the majority of feature sets and consequently simplifies their accessibility.

In its default settings, *flacco* computes more than 300 different numerical landscape features, distributed across 17 so-called feature sets, given a set of sampled points and their function evaluations. We make use of this property to import our own trajectory-based sample sets in Chapters 9 and 10. Conveniently, for well-known problem collections such as the BBOB, *flacco* provides interfaces that facilitate feature extraction for those functions and their instances. To this end, it offers different sampling strategies, such as the uniform sampling or generating points by a Latin hypercube construction.

A Web-based graphical user interface of *flacco* is also available, and can be found at [HK17].

5 Algorithm Selection

With the algorithm portfolio, the problem collection and corresponding problem landscape features at hand, we can now discuss the algorithm selection models built on top of performance regression (as illustrated in Figure 1.2), and we outline all the tools needed to achieve a fully functional algorithm selection framework. We also present useful concepts to assess the quality of an algorithm selector, i.e., to measure its performance.

5.1 Per-Instance Algorithm Selection

Many prominent optimization problems have long been considered to be central specimens in active research efforts when it comes to developing new solving strategies. For those well-analyzed and well-understood problems, a range of high-performing algorithms is available, but there is typically no single algorithm that dominates all others on all possible instances of the problem at hand. Instead, different algorithms achieve best performance on different problem instances, which is known as *performance complementarity* [BKK+16; KHN+19; Kot14; Ric76; Smi09]. This long-existing observation has naturally led to the necessity of appropriate algorithm selection for the problem instances we can face. Given an optimization problem, its specific instance that needs to be solved, and a set of algorithms that can be used to solve it, the so-called *per-instance algorithm selection (PIAS)* problem arises: how to determine which of those algorithms can be expected to perform best on that particular instance?

The (per-instance) algorithm selection problem has historically been tackled manually, using expert knowledge to make algorithm recommendations. Addressing the question of reducing bias which is inherent to the manual approach was soon deemed indispensable; it paved the way towards the concept of automating this process by means of various tools available in related fields. To this day, *automated algorithm selection* has been steadily gaining prominence, resulting in a large body of work [CV97; KT19; LHH+15; LNA+03; XHH+08; XHH+12]. From high-level, *passive* strategies that are based on choosing an algorithm as a function of *a priori* available features [BS04; LMP+20; MRW+21], through *active*, bet-and-run approaches that consist of running several algorithms and stopping all but the best-performing one (i.e., per-set algorithm selection) [MBT+11; ME13; MKH15], the research focus has ultimately turned towards establishing efficient *landscape-aware* algorithm selection models that operate using low-level problem features, which are computed beforehand [BMT+12; CDL+21; HKV19; KHN+19; KT19; MKH12].

5.2 From Performance Regression to Algorithm Selection

This thesis makes use of the prediction of algorithm performance as a foundation upon which the algorithm selection models are built. This pipeline component (illustrated in Figure 1.2) relies on machine learning techniques. As briefly mentioned in Chapter 5, general approaches differ based on the type of learning model. Supervised learning methods, such as classification and regression, are the ones coupled with the information about landscape (ELA) features to make predictions about how well certain algorithms will perform on certain problem instances. To this end, we employ *ELA-based regression*, as we are concerned not only in knowing which algorithm is recommended, but also in further interpreting and comprehending the actual behavior of algorithms. Regression models allow for keeping track of precise magnitudes of differences between performances of different algorithms. This interpretability aspect is especially important due to settings that include portfolios of very similar algorithms.

With respect to the available regression techniques, a general state-of-the-art recommendation highlights *random forests* as a method that outperforms other typically used regressors in the context of automated algorithm selection [HXH+14]. Random forests [Bre01] are ensemble-based meta-estimators that fit decision trees on various sub-samples of the data set and use averaging to improve the predictive accuracy and to control over-fitting. We adopt them as a principal regression model going forward, but we also investigate the potential of other models under fixed-budget circumstances (cf. Chapter 7).

5.3 State of the Art

Existing research in automated algorithm selection and closely related topics, such as algorithm configuration, can be roughly positioned on one of the two main axes, depending on whether underlying machine learning techniques are supervised or not. In terms of unsupervised learning, reinforcement learning and its variants are the most predominant approaches [BBE+20; BBH+19; KHN+19; SKL+19]. When it comes to the the supervised learning, strategies building upon exploratory landscape analysis (ELA) (cf. Chapter 4) are central in the field. In particular, ELA-based regression has been applied to assess the effect problem features have on algorithm performance [LDV+20], to configure algorithms' parameters [BDS+17; BKJ+19; HHH+06], as well as to select algorithms from a given portfolio [KT19; MKH12]. See [KHN+19; MSK+15] for comprehensive surveys of automated algorithm selection state-of-the-art methods and results. With regards to dynamic algorithm selection, search behavior or algorithms' state parameters can influence the model recommendations [BPR+19; DLV+19].

5.4 Performance Assessment of Algorithm Selectors

We introduce here two baselines that are essential for assessing the quality of algorithm selectors. The performance of a (hypothetical) perfect per-instance algorithm selector, also known as the *virtual best solver (VBS)* or the *oracle selector*, provides a lower bound on the performance of any realistically achievable algorithm selector. On the other hand, a natural upper bound on the algorithm selector performance is provided by the *single best solver (SBS)*, which is the algorithm with the best performance among all other algorithms in the considered portfolio.

The ratio between VBS and SBS performances, also referred to as the *VBS-SBS gap*, gives an indication of the performance gains that can be obtained by per-instance algorithm selection in the best case. Consequently, the fraction of this gap closed by a certain algorithm selector provides a measure of its quality [LRK17].

To measure both performance regression and algorithm selection accuracy, we use the *Root Mean Square Error (RMSE)* metric. The RMSE is the square root of the mean of the square of all of the errors. It can be thought of as a kind of (normalized) distance between the vector of predicted or selected values and the vector of true values.

When it comes to performance regression, it is the standard deviation of prediction errors; it measures how spread out those errors are, and it serves to aggregate the magnitudes of the errors in predictions for various data points into a single measure of predictive power. The RMSE allows us to compare (and quantify) how well different models predict the performance. In the fixed-budget regression that we focus on it the thesis, the prediction errors are distances of the prediction to the true target value.

We use the same metric to assess the quality of the algorithm selectors we consider; in this case, the selection errors are the difference between performances of the selected algorithm and the true best algorithm. We will thus speak of the regression RMSE and of the selection RMSE in the remainder of the thesis.

Part II

Contributions

6 Combining Fixed-Budget Regression Models

This chapter is based on paper [JD20], which appeared in the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2020, ***Landscape-Aware Fixed-Budget Performance Regression and Algorithm Selection for Modular CMA-ES Variants****.*

6.1 Preliminaries

Summary of Results. In this chapter, we propose a novel combined performance regression model for single-objective numerical problems in the *fixed-budget* setting (i.e., with a limited budget of function evaluations) which aims to solve the algorithm selection problem by combining the use of problem features that can be automatically computed and off-the-shelf supervised machine learning techniques. As commonly carried out in the state of the art algorithm selection, the model learns the mapping between problem features on one hand and algorithm performance (measured by the *target precision* in the fixed-budget context) on the other hand. We allow ourselves a budget of 500 function evaluations.

We show that we are able to achieve high-quality predictions of algorithm performance values by suitably combining two differently trained regression models, one that predicts the actual target precision (the *unscaled* model) and another that predicts the logarithm of the target precision (the *logarithmic* model).

An important conjecture we make here is that the landscape features are expressive enough so that we can rely on information carried by them for the purpose of automatically constructing algorithm selection models for numerical black-box optimization problems.

Experimental Setup

The BBOB Test Suite. In this chapter, we restrict our attention to dimension 5. We take into account only the first 4 instances (IID 1-4) of the 24 noiseless BBOB functions (FID 1-24), which results in 96 black-box problems in total.

The Modular CMA-ES. The algorithm portfolio of choice for this chapter consists of the modular CMA-ES variants (presented in Section 3.1.2). All 4608 modular CMA-ES variants were executed on the 96 problem instances mentioned above (i.e., four instances per each of the 24 BBOB functions), with a budget of 500 function evaluations. We

perform five independent runs per each algorithm and problem instance. After every run, we store the best target precision achieved, computed as $f(x_{\text{best\_tp}}) - f(x_{\text{OPT}})$ (this value is positive, since we assume minimization as objective), as well as the function ID, instance ID, and algorithm ID. We compute the median best target precision over five different runs for all functions and instances and for each algorithm.

From this large median performance dataset, we select the best algorithm per function to create an algorithm portfolio that would act as the target data for our regression models. To identify which algorithm is the best for a certain function, we compute the median performances of algorithms over *instances* as well, which gives us 4608 different target precision values per each of the 24 functions. We then pick the algorithm with the minimum target precision among those 4608 for each function, and end up with a portfolio of 24 best algorithms in total.

Since we operate within the fixed-budget approach, we consequently use the *target precision* after 500 function evaluations as a measure of an algorithm's performance. It is important to note that the information carried by the target precision intuitively stands for the order of magnitude of the actual distance to the optimum. For instance, if a recorded precision value on a certain problem instance is 10^{-2} for one algorithm and 10^{-8} for the other, consequently they differ by 6 orders of magnitude, and so we can interpret that informally as the latter one being "6 levels closer to the optimum" than the first one. This perspective helps immensely in designing the experiment, as we are interested not only in the actual precision values, but also in the "distance levels" to the optimum, which are very conveniently computed as the logarithm of the precision value.

We plot in Figure 6.1 the median target precision values (over five independent runs) of 24 different modular CMA-ES variants for each of the first four instances of the 24 BBOB functions. This is our algorithm portfolio of choice, and we use these values as the target data for our performance regression models. However, we observe that the 24 algorithms are in fact very similar in performance, which makes the algorithm selection problem in this setting rather challenging. As shown in Figure 6.2, each algorithm "wins" on at least one of the 96 considered instances.

Feature Computation. As the predictor variables for our model, we use vectors of landscape feature values per each problem instance. For the feature value computation, we use the *flacco* toolbox, presented in Section 4.3. To compute the features, we evaluate 2000 uniformly sampled search points per function and instance in 5D, and feed the points and their respective fitness values to *flacco*. Since the feature approximations can show low robustness for certain features [RDD+19; SGW19], we replicate this step 50 independent times and take the median feature values per problem instance as a final feature vector. By selecting only those feature sets that do not require further sampling in the search space (all described in Chapter 4), we end up with 56 features in total per problem instance.

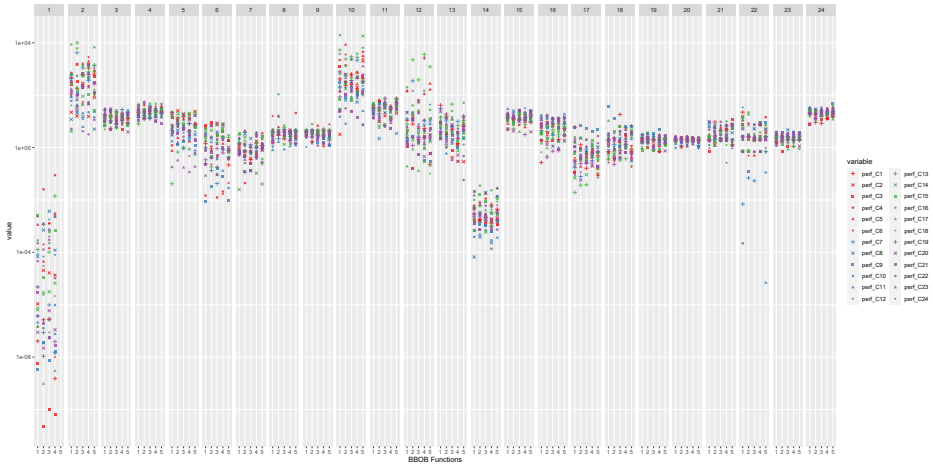


Figure 6.1: Median performance (measured by the target precision) over 5 independent runs of the 24 modular CMA-ES variants (selected out of 4608 as the best variant per BBOB function) on the first 4 instances of all 24 BBOB functions. These 24 algorithms represent the algorithm portfolio from which we want to select the best-performing one for an unseen optimization problem.

Configuration																							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
7	3	10	4	1	2	3	3	4	2	5	3	8	6	3	1	6	1	6	2	5	7	4	2
Number of instances for which the configuration performed best (out of a total of 96 instances)																							

Figure 6.2: Number of problem instances (out of 96 total; second row) for which each of the algorithms (first row) achieved the best performance.

It is worth noting that the sample size of 2000 points is certainly much higher than one would be willing to invest in concrete applications, and we will therefore analyze the sensitivity of the results with respect to the sample size in Section 6.3.3. Note here that – in particular with such a small budget as investigated in our work – the development of features that use the samples of a search trajectory rather than additional samples and hence avoid the specific sampling step for feature value computation [MS17] is very strongly needed. This trajectory-based approach has been analyzed and shown to be successful in Chapter 10.

We will also investigate the effect of the feature portfolio (cf. Section 6.3.1), and show that a smaller set of features does not necessarily lead to worse results.

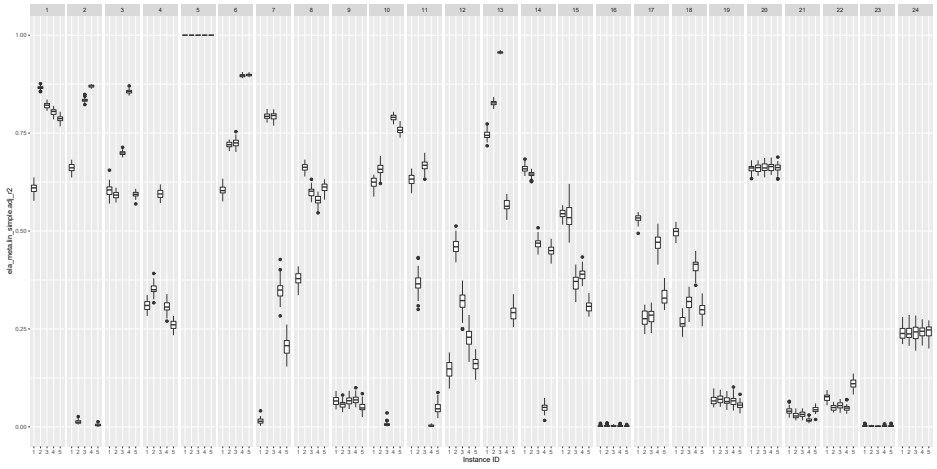


Figure 6.3: Distribution of approximated values for the *ela_meta_lin_simple_adj_r2* feature for the first five instances of the 24 BBOB functions, normalized in the 0-1 range. The values are computed from 2000 samples each, and each box plot shows the distribution of 50 independent runs.

Figure 6.3 shows an example of how the feature values are distributed for different instances. The feature under consideration is one of the classical ELA Meta-model features, a value of adjusted R^2 correlation coefficient for the linear model fitted to the data. We see that the values are overall quite constant for most functions. However, there are functions (e.g., F8, F12) for which this is not the case. All five instances of F5 are correctly identified as linear slope functions; they have a feature at value 1. Importantly, we observe that some features are more expressive than others, and are prone to discriminate between different problems better than others, as suggested by [RDD+19].

Regression Models. For our regression models, we use off-the-shelf random forest regressors with 1000 estimators, available via the *scikit-learn* Python package [PVG+11]. No parameter tuning was involved in our setup, which gives us hope that a further improvement of the accuracy is highly likely to achieve by fine-tuning the machine learning model, but also that a clever design of underlying model mechanism can render the processing itself quite cheap. This is investigated in more detail in Chapter 7.

6.2 Fixed-Budget Performance Regression

Using the elements described in Section 6.1 as key pieces in our approach, we organize the experiment in the following way. For each algorithm in the portfolio, we train two separate random forest regression models to predict the algorithm's performance on all problem instances, given a vector of features per problem instance as the predictor variable. One model is trained to predict the actual target precision values (we call it the *unscaled model* from here onward), while the other predicts logarithms of the target precision data to represent the "distance levels" explained in Section 6.1 (we refer to this model as the *logarithmic model* or simply the *log-model*).

In order to obtain more realistic and reliable estimates of the accuracy of the regression models, we assess them using a K-fold *leave-one-instance-out* cross-validation (we use $K = 4$), i.e., per algorithm, we split the data in such a way that we use three instances per BBOB function for the training (72 problems in total), and we test on the remaining instance (24 problems in total). We do this with each of the four instances to ensure that each instance was used once in the test phase. We consistently store the values of the prediction on the test instance for each algorithm in the portfolio for both the unscaled and the logarithmic model. The full experiment is replicated three independent times and the median values are retained for the analysis.

In Figure 6.4, we plot the true vs. predicted values of a single variant of the modular CMA-ES over 50 independent regression model runs for each of the first four instances of the 24 BBOB functions. The distribution of two different predictions, the unscaled and the log-prediction, are shown. We observe a high stability of the predictions irrespective of the number of replications (i.e., whether we perform three runs of the model, as throughout our experiments, or 50 runs, as shown here), and we can thus conclude that this allows for a significant decrease in the computational cost of creating such a model.

Importantly, Figure 6.4 shows how well the predicted values follow the actual data. As a general trend, the predictions of the unscaled model fit better to larger precision values, while the logarithmic model better predicts small target precisions.

As a measure of model accuracy, the *Root Mean Square Errors* (RMSE) and its logarithmic counterpart (log-RMSE) are computed per each prediction. They allow us to compare and quantify how well different models predict the performance. Within the regression context, while the RMSE corresponds to the unscaled model, the log-RMSE

Table 6.1: Root Mean Square Error (RMSE) and its logarithmic counterpart values (log-RMSE) as a measure for model prediction accuracy for each algorithm in the portfolio in 3 different scenarios, for both the unscaled and logarithmic model. These measures compare how well different models fit the actual target data. The default experiment (the first 2 columns; on the left) consists of the performance regression using the full feature set, where features were computed using 2000 samples. The second 2 columns (in the middle) correspond to the experiment where the regression was based on *9 selected features only*, while the third 2 columns (on the right) describe the case where, again, the full feature set was used, but this time the features were *computed using 50d (= 250) samples*. The values shown in bold represent lower errors when comparing the first 2 scenarios (all features vs. selected features), while the underlined values highlight lower errors when the 1^{st} and the 3^{rd} scenario are compared (features computed with 2000 samples vs. with 250 samples).

	Default		Selected features		$50d$ samples	
Config.	RMSE	log-RMSE	RMSE	log-RMSE	RMSE	log-RMSE
C1	**130.1**	0.808	135.4	**0.663**	123.6	1.066
C2	84.1	0.776	**79**	**0.682**	77.7	0.891
C3	206.5	0.842	**199.7**	**0.663**	189.6	1.308
C4	201.9	0.757	**198.3**	**0.682**	200.7	0.829
C5	1011.2	0.690	**916.8**	**0.564**	1106.6	0.742
C6	**649.9**	**0.986**	660.6	1.018	618.1	1.030
C7	462.8	0.804	**455**	**0.743**	412.8	0.989
C8	61.2	0.830	**58.8**	**0.698**	57.4	0.925
C9	**71.1**	0.770	74.3	**0.623**	77.3	1.133
C10	12.1	0.771	**11.4**	**0.649**	12.2	1.080
C11	71.9	0.794	**55.2**	**0.653**	78.3	1.060
C12	76.7	0.708	**64.9**	**0.601**	78.1	0.789
C13	**1120.4**	0.700	1124.7	**0.696**	1113.1	0.789
C14	**51.1**	0.792	51.4	**0.678**	44.4	0.973
C15	60.5	0.629	**54.7**	**0.519**	56.5	0.748
C16	2306.3	0.621	**2280.6**	**0.604**	2239.0	0.791
C17	114.3	0.781	**98**	**0.631**	111.2	1.134
C18	**130.4**	0.640	149.6	**0.596**	131.6	0.903
C19	85.1	0.710	**82.4**	**0.571**	73.5	1.025
C20	**144.3**	0.760	152.9	**0.618**	138.7	1.032
C21	23.2	0.719	**23**	**0.662**	20.7	1.007
C22	17.0	0.805	**16.4**	**0.714**	16.5	0.919
C23	53.6	0.613	**45.8**	**0.538**	55.0	0.691
C24	**571.9**	**0.803**	604.7	0.86	531.1	0.872

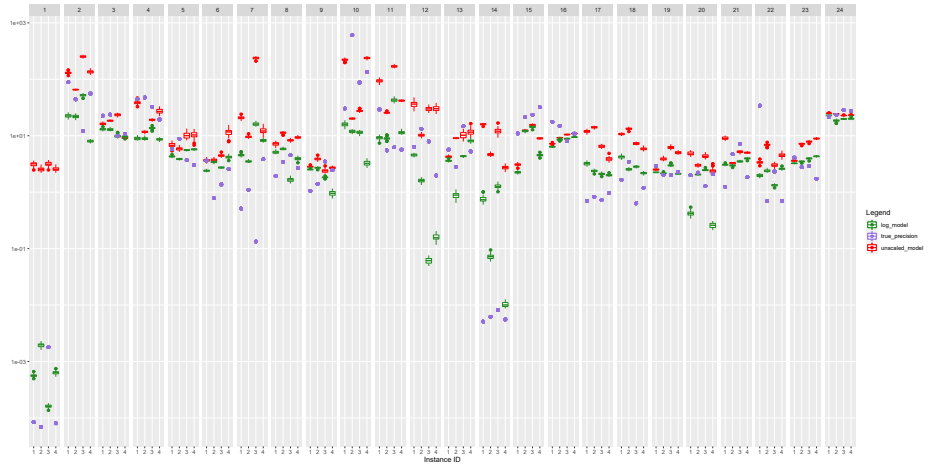


Figure 6.4: Distribution of predicted algorithm performance values by the unscaled (in red) and logarithmic regression models (in green) across 50 independent runs, with the actual algorithm performance values (in purple) as dots. The algorithm performance is measured by the target precision.

corresponds to the logarithmic model. Once computed, we aggregate the relevant RMSE and log-RMSE values per algorithm to estimate the quality of the model at hand.

We report in the first two columns of Table 6.1 how good the regression models (unscaled and logarithmic) were at predicting the performance of each algorithm. We observe that the same algorithm (e.g., C16) can have high RMSE (and thus be very bad in predicting the actual data) all while having one of the lowest log-RMSE (quite the opposite for the log-scale data).

6.2.1 Impact of Feature Selection

We have reported above results for the regression models that make use of 56 *flacco* features. However, it is well known that feature selection can significantly improve the accuracy of random forest regressions. We therefore present a brief sensitivity analysis, which confirms that significantly better results can be expected by an appropriate choice of the feature portfolio. We note that a reduced feature portfolio also has the advantage of faster computation times, both in the feature extraction and in the regression steps. We have performed an *ad hoc* feature selection, which solely builds on a visual analysis of the approximated feature values. This was done by studying the plots as in Figure 6.3 for all computed *flacco* features, which we use to identify features that show a high *expressiveness* [RDD+19], in the sense that they seem very

suitable to discriminate between the different BBOB functions. From this set, we then choose nine features: *disp.diff_mean_02, ela_distr.skewness, ela_meta.lin_simple.adj_r2, ela_meta.lin_simple.coef.max, ela_meta.lin_simple.intercept, ela_meta.quad_simple.adj_r2, ic.eps.ratio, ic.eps.s* and *nbc.nb_fitness.cor*.

The middle two columns of Table 6.1 illustrate how the model accuracy indeed increases with the reduced feature set. 16 out of the 24 RMSE values and 22 of the 24 log-RMSE values are smaller than for the model using all features (which, we recall, are provided in the 2 leftmost columns of the same table). The values of the more accurate model between the two are highlighted in bold in this case. These results support the idea that an appropriate feature selection is likely to result in significant improvements of our regressions, and hence of the algorithm selectors which we shall discuss in the next section.

6.3 Fixed-Budget Algorithm Selection

After examining the regression models' accuracy, we next evaluate the performance of two simple algorithm selectors, which are based on the predictions of the unscaled and the logarithmic models, respectively. The former selects the algorithm for which the unscaled regression model predicted the best performance, and, similarly, the latter bases its decision on the best prediction of the logarithmic regression model. To quantify how well the selectors perform per problem instance, we compare the target precision of the algorithm chosen by the selector for the instance at hand to the target precision of the actual best algorithm for that instance. We are then able to indicate the overall quality of this selector by computing the RMSE and log-RMSE values (aggregated across all problems) using the differences in performance.

Following common practices in algorithm selection [BKK+16], we compare the performances of these two selectors with two different baselines: the *virtual best solver (VBS)* (which gives a lower bound for the selectors), and the *single best solver (SBS)* (which gives an upper bound for the selectors). Since we have two models, we have two different SBS, one for the logarithmic model (SBS_{log}) and one for the unscaled one ($SBS_{unscaled}$). Studying the RMSE values of the 24 different algorithms (Figure 6.5), we find that configuration C10 has the best performance (measured against the VBS), with the RMSE value of 13.65. Configuration C21, in contrast, is seen to have the best log-RMSE value, and is therefore the SBS_{log} of the full portfolio. Its log-RMSE value is 0.733. In the following, for ease of notation, we will not distinguish between the two SBS, and, in abuse of notation, will combine them into one. That is, we simply speak of the SBS, and refer to C10 when discussing RMSE values, while we refer to C21 when discussing log-RMSE values. As we can see in Figure 6.5, our algorithm selectors are able to outperform the SBS in terms of log-RMSE performance. We did not find a way, however, to beat the RMSE values of C10, nor the ones of C8, C14, C21, C22, nor C23.

Performances of single algorithms from the portfolio, as well as those of the two selectors (unscaled and logarithmic selector) are shown in Figure 6.5. For the majority of the portfolio, a lower RMSE value entails a lower log-RMSE value and vice versa. We see that our two selectors already outperform the majority of algorithms from the portfolio on both RMSE- and log-RMSE scales, but we want to make use of the observation reported in Section 6.2 that the unscaled model better predicted higher target precision, while the logarithmic model has better accuracy for small target precision. We therefore aim at combining the two regression models, to benefit from the two complementing strengths. Here again we can define a virtual best solver, which is the one that chooses for each instance the better of the two suggested algorithms from the unscaled and the logarithmic model, respectively. Clearly, in terms of single best solver, the logarithmic model minimizes the log-RMSE, whereas the unscaled model minimizes the RMSE. We compare these three algorithm selectors (unscaled algorithm selector, logarithmic algorithm selector, and VBS algorithm selector) with a selector which combined the two basic selectors in the following way: if the target precision of an algorithm, as predicted by the logarithmic model, is smaller than a certain threshold, we use the logarithmic selector, whereas we use the recommendation of the unscaled selector otherwise. A new optimization sub-problem that arises here is to find the threshold value which minimizes the RMSE and log-RMSE of the combined selector, respectively. A sensitivity analysis with respect to this threshold will be presented in Section 6.3.1.

Once the optimal threshold value found, we measure the performance of our selector and add it, along with the VBS algorithm selector, to Figure 6.5. We clearly see that our algorithm selector performs better than the unscaled algorithm selector and the logarithmic algorithm selector. It is also better than most of the algorithms. Also, Figure 6.5 demonstrates that our selector effectively reduces the gap towards the VBS algorithm selector. Detailed numbers for the RMSE and log-RMSE values of the different algorithm selectors are provided in Table 6.3 (rows for 24 algorithms).

6.3.1 Impact of the Threshold Value and the Feature Portfolio

We now study the influence of the threshold value which determines whether we use the unscaled or the logarithmic algorithm selector. In the previous section, we had chosen this value so that it optimized the log-RMSE measure and the RMSE measure (these are the red triangle and the red diamond in Figure 6.5, respectively). Table 6.2 analyzes the influence of this threshold value, and shows both the RMSE and log-RMSE values for different thresholds. We note that one could formulate an alternative decision rule, in which the selection of the model is not based on the target precision recommended by the logarithmic model, but by the unscaled model. We did not observe significant differences in the performance of these two approaches, and thus omit a more detailed discussion.

We show in Table 6.2 also the results for the algorithm selectors that build on the

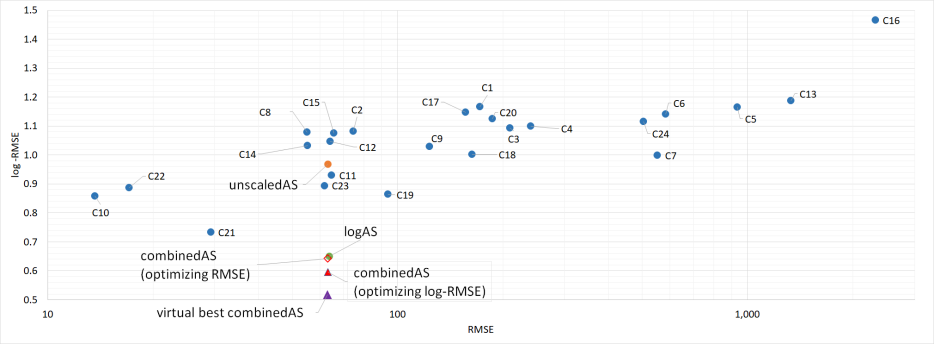


Figure 6.5: RMSE and log-RMSE values as measures for the quality of the configurations (label C*x*) and for the algorithm selectors: the AS using only the predictions from the logarithmic model (green dot, hiding behind red diamond), the AS using predictions from the unscaled model (orange dot), the virtual best combination of these two models (purple triangle), and our two combined algorithm selectors, which optimize for RMSE and log-RMSE (red diamond and red triangle, respectively).

regression models using only the nine selected features. In fact, it turns out that for these selected-feature regression models, the combination of the two different regressions is not beneficial – we were not able to identify means to improve upon the algorithm selector that uses the performances predicted by the logarithmic model.

6.3.2 Impact of the Algorithm Portfolio

Our final analysis concerns the portfolio for which we do the regression. We note that in all the above we have given ourselves a very difficult task: algorithm selection for a portfolio of 24 solvers that all show quite similar performance (i.e., that all stem from a family of very similar algorithms; Figure 6.1). We now study alternative problems, in which we consider only subsets of the 24 CMA-ES configurations considered above. More precisely, we consider in Table 6.3 the portfolio of configurations C13-C24, i.e., the second half of the original portfolio.

We observe that, while the log-RMSE values remain fairly consistent for all the selectors independently of the portfolio size, the RMSE is significantly reduced by reducing the portfolio size for all but one selector, the SBS. It is worth looking further into this aspect of the problem in order to better understand if the observed effects are simply a product of less choice, or, more likely, there are other factors at play, e.g., whether the algorithms chosen for the portfolio are diverse enough in their performance on different problem instances.

Table 6.2: Sensitivity of the RMSE and the log-RMSE with respect to the threshold value at which we switch from choosing the modular CMA-ES configuration suggested by the logarithmic model to the one suggested by the unscaled model. We show results for both models, using the full feature set and the selected feature set as the basis for regression. Optimal values are shown in bold and are underlined.

	RMSE		**log-RMSE**	
Threshold	**All features**	**Selected features**	**All features**	**Selected features**
0.01	63.20	25.87	0.687	0.728
0.1	63.20	25.87	0.676	0.723
0.5	63.19	25.81	0.600	0.637
0.814	63.19	25.81	**0.595**	0.627
1	63.19	25.77	0.624	0.620
2	63.17	25.81	0.643	0.607
2.294	**63.17**	25.80	0.643	0.590
3	63.75	25.82	0.656	0.583
8.525	63.71	**15.50**	0.654	**0.565**
10	63.71	15.55	0.654	0.565
20	63.69	15.55	0.650	0.565
50	63.69	15.55	0.650	0.565

Table 6.3: Comparison of the RMSE and log-RMSE values for the different algorithm selectors for the full portfolio of 24 and for the reduced set of 12 configurations.

	# algos	unscaled	log	the AS	VBS	SBS
RMSE	12	17.25	18.03	16.94	12.78	20.37
	24	63.19	63.69	63.19	63.09	13.65
log-RMSE	12	0.967	0.621	0.608	0.561	0.629
	24	0.968	0.650	0.595	0.517	0.733

6.3.3 Impact of the Sample Size for Feature Extraction

We recall that our feature approximations are based on 2000 samples. As commented above, this number is much larger than what one could afford in practice. Belkhir et al. [BDS+17] showed that sample sizes as small as $30d$-$50d$ can suffice to obtain reasonable results. While their application is in algorithm *configuration*, we are interested in knowing whether we obtain similarly robust performance for algorithm *selection*. As mentioned before, in the long run, one might hope for zero- or low-cost feature extraction mechanisms that simply use the search trajectory samples of a CMA-ES variant (or some other solver) to predict algorithm performances and/or perform a selection task. First steps in this direction have already been made [Mal18; MM19; MS17].

Therefore, in this last section we study the influence of the feature sample size on the performance of our algorithm selector. We compare the results reported in Section 6.3 with the results obtained from the repeated regression experiment, only this time using features computed with $50d$ (250) samples.

Table 6.4: Comparison of the RMSE and log-RMSE values for the different algorithm selectors for the regression based on 250-sample features and 2000-sample features.

	# feature samples	unscaled	log	the AS	VBS	SBS
RMSE	250	23.51	38.74	23.45	23.05	13.65
	2000	63.19	63.69	63.19	63.09	13.65
log-RMSE	250	0.881	0.700	0.660	0.511	0.733
	2000	0.968	0.650	0.595	0.517	0.733

The two rightmost columns of Table 6.1 show the regression model accuracy for this case, and allow for a comparison with the default scenario. We once again conveniently highlight the values of the more accurate model, only this time underlined. We clearly notice that, in terms of RMSE values of the logarithmic model, using a larger sample size is preferable consistently for all the algorithms in the portfolio. On the other hand, the model using a reduced sample size performed better on 18 out of 24 algorithms in terms of RMSE.

In Table 6.4 we report the differences of the algorithm selectors in case the regression was based on features computed using $50d$ (250) samples vs. those computed using the original 2000 samples. The results are comparable in terms of both RMSE and log-RMSE values; the distances between the performance of our combined selector and the VBS are similar in both 250- and 2000-sample experiments. Also, in neither of the two experiments have we been able to beat the SBS. However, we notice a general decrease in the RMSE when using 250 samples to compute the features, which is an interesting observation leading to a conjecture that it might be preferable to use a smaller number

of samples to compute the features for regression purposes, while still maintaining the robustness of the results.

6.4 Conclusions

We have studied in this chapter how to increase the accuracy of ELA-based regression models and algorithm selection by combining a plain, "unscaled" regression with a regression operating on the log-scaled data. While the former achieves higher accuracy for large target precision values, the latter performs better for fine-grained precisions. By combining the two models, we could improve the accuracy of the regression and of the algorithm selector. Our combined algorithm selector reduces the gap towards the VBS, although it does not consistently beat the SBS across all cases. These results, however, still open up a path to further exploit the power of ELA-based regression and algorithm selection in different settings.

In the remainder of this section, we list a few promising avenues for future work.

Cross-Validation of the Trained Algorithm Selector on Other Black-Box Optimization Problems. Our ultimate goal is to train an algorithm selector that performs well on previously unseen problems. We are therefore keen on testing our regression models for the different CMA-ES variants and on testing the trained algorithm selector on other benchmark functions. The current literature is not unanimous with respect to the quality that one can expect from the training on the BBOB functions. While [BDS+17] reported encouraging performance, LaCroix and McCall [LM19] could not achieve satisfactory results.

Feature Selection. The results presented in Section 6.2.1 indicate that a proper selection of the features can improve the quality of the random forest regression quite significantly. Our feature selection was based on an purely visual interpretation of the distribution of the feature value approximations (i.e., plots as in Figure 6.3), which is similar to the analyses made in [MKS18; RDD+19]. A proper feature selection may help to improve the accuracy of our models further. Since feature selection is quite expensive in terms of computational cost, a first step could be a comparison of the accuracy of the two here-presented models with those using the feature sets selected in [KT19].

Fixed-Target Settings. While we have deliberately chosen a fixed-budget setting (which is the setting of our envisaged applications), we are nevertheless confident that the combination of a logarithmic with an unscaled regression model could also prove advantageous in fixed-target settings, in which the goal is to minimize the average time needed to identify a solution of function value at least as good as some user-defined threshold.

Different Algorithm Portfolios. We have chosen a very challenging task in performing algorithm selection on a collection of algorithms that all stem from the same family. A cross-validation of our findings on more diverse portfolios is a straightforward next step for our work.

7 Impact of Hyper-Parameter Tuning

This chapter is based on paper [JPE+21], which appeared in the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2021, ***The Impact of Hyper-Parameter Tuning for Landscape-Aware Performance Regression and Algorithm Selection****.*

7.1 Preliminaires

Summary of Results. We compare in this chapter the performance of 30 different regression models on the algorithm selection task suggested in [KT19]. The 30 selected models are all *tree-based*, and they are configurations of random forest, decision tree and bagging decision tree regression techniques.

We train the 30 models on different data sets, taking as input the ELA features of a problem instance and outputting the fixed-budget performance of an algorithm. Two distinct ELA feature representations (based on different sample sizes for feature computation) describe each of 120 problem instances belonging to 24 BBOB problems. Algorithm performances for a portfolio of twelve algorithms were recorded for different budgets of function evaluations (we consider here budgets of 250, 500, and 1000 function evaluations), measuring the target precision of the best found solution (i.e., the distance to the optimum). On top of that, two complementary regression approaches were adopted for each model and each data set: one that predicts true (unscaled) target precision values and another that predicts the logarithm of target precision, as suggested in Chapter 6.

We then use stratified 5-fold cross-validation to ensure that all problem instances were used in the test phase of our models, and we build, for each regression model and each data set, an algorithm selector, which takes as input the predicted target precision of each of the twelve algorithms and which returns the algorithm with the best predicted performance. The true target precision of this selected algorithm is then compared to that of the actual best algorithm for that problem instance, which defines the loss that we associate to the regression model. Following the approach in Chapter 6, for each regression model and each data set, we also build an algorithm selector which combines the regression for the unscaled target precision with that for the logarithmic precision, favoring the latter for small target precision values, and favoring the former otherwise. In total, we evaluate for each of the 30 regression models, three different algorithm

selectors (unscaled, logarithmic, combined), on 6 different data sets (3 different budgets of function evaluations and 2 different feature sample sizes).

The results in this chapter clearly indicate a need for appropriate tuning of the regression techniques, but more importantly, they raise an important question of how the chosen machine learning model can lead to highly varying results in the algorithm selection step. We argue that the untapped potential of a careful choice of the relevant machine learning model and its hyper-parameter configuration can be only fully exploited when taking into account some preliminary knowledge about the problem classes and the algorithms. This sub-explored area of research merits further investigation.

We see differences of up to several orders of magnitude in the accuracies of the different models, not aggregated across optimization algorithms. Even if we aggregate the errors, the differences between them are still as high as 60%. We further notice that different models perform differently on different types of problems, making it difficult to derive a general recommendation for which model to favor in which scenario. This, however, does not limit the relevance of our work, since training the different regression models is of negligible cost, in particular when compared to the efforts required for setting up the whole algorithm selection pipeline. In practice, the use of several machine learning techniques at the same time ("ensembles") is not uncommon – quite the contrary, in fact [MSJ+12]. We show in Chapter 8 that employing ensemble learning techniques in the context of algorithm performance regression for landscape-aware algorithm selection is indeed promising. Therefore, this chapter also motivates further investigation of those techniques in landscape-aware algorithm selection, configuration and design.

Experimental Setup

Regression Models. Following best practices in applying supervised machine learning techniques to the landscape-aware algorithm selection context, we first perform a preliminary step of testing the quality of the following seven families of regression models with different hyper-parameter values tested via iterative grid search: Random Forests [Bre01], Decision Trees [BFS+84], Bagging Decision Trees [Bre96], Lasso [Tib96], ElasticNet [ZH05], KernelRidge [Mur12], and PassiveAggressive [CDK+06]. We retained only the models that largely outperformed the rest in terms of regression quality.

For our analysis we have selected three different classes of regression models, namely Decision Tree, Random Forest and Bagging Decision Tree (also referred to as Bagging DT within the chapter). For each model class, the hyper-parameter configurations used are shown in Table 7.1. Note that, due to the rather small size of our data set, we have not considered techniques such as Neural Networks which are powerful for a huge quantity of data. We have not considered any classification techniques either (including Bayes classifier), as we are interested in predicting numerical performance values for each algorithm.

Since the basic component unit of all considered models is a decision tree (both

Random Forests and Bagging DTs are based on them), the hyper-parameter *crit* value can be one of the following three: *mse* (mean squared error), *mae* (mean absolute error) and so-called *friedman mse* - Friedman mean squared error. The *minsplit* hyper-parameter represents the minimum number of data instances a tree node has to contain in order to become a splitting node. Lastly, the *nest* hyper-parameter defines the number of decision trees needed to build a Random Forest or Bagging DT model. In total, we end up with 30 different regression models, with 6 different configurations for Decision Tree, 12 for Random Forest, and 12 for Bagging DT. Table 7.1 summarizes the chosen hyper-parameter values for different regression classes.

Table 7.1: Hyper-parameter values for the regression models.

Model	Hyper-parameters
DecisionTree (6 configs.)	• $crit \in \{"mse", "mae", "friedman\_mse"\}$ • $minsplit \in \{4, 5\}$
RandomForest (12 configs.)	• $crit \in \{"mse", "mae"\}$ • $minsplit \in \{4, 5\}$ • $nest \in \{3, 6, 9\}$
BaggingDT (12 configs.)	• $crit \in \{"mse", "mae"\}$ • $minsplit \in \{4, 5\}$ • $nest \in \{3, 6, 9\}$

Benchmark Problems. We consider the first five instances (IID 1-5) of each of the 24 noiseless functions from the BBOB test suite (FID 1-24) in dimension 5, for a total of 120 problem instances.

Algorithm Portfolio. The algorithm portfolio we choose in this chapter was suggested in [KT19] for its diversity. It consists of the following 12 algorithms: BrentSTEPqi [PB15], BrentSTEPrr [PB15], CMA-ES-CSA [Ata15], HCMA [LSS13b], HMLSL [Pál13a], IPOP400D [ABH13], MCS [HN09], MLSL [Pál13a], OQNLP [Pál13b], fmincon [Pál13b], fminunc [Pál13b], and BIPOP-CMA-ES [Han09]. Note that, due to the unavailability of the raw performance data for one of the algorithms in the original study, the BIPOP-CMA-ES was added instead of the missing one. The performance data of all twelve algorithms can be downloaded at [HAB20], but for our setting it was more convenient to extract the relevant figures from IOHprofiler [DWY+18]. As we focus on the fixed-budget performance throughout the thesis, we consider 3 different budget sizes of 250, 500 and 1000 function evaluations across all algorithms from the portfolio for purpose of sensitivity analysis, and we restrict ourselves to a single algorithm run per problem instance. We show in Figure 7.1 the portfolio's target precisions reached after 1000 evaluations. We note that the algorithm performances are significantly less diverse for functions 15 through 20, 23 and 24 than for the other problems.

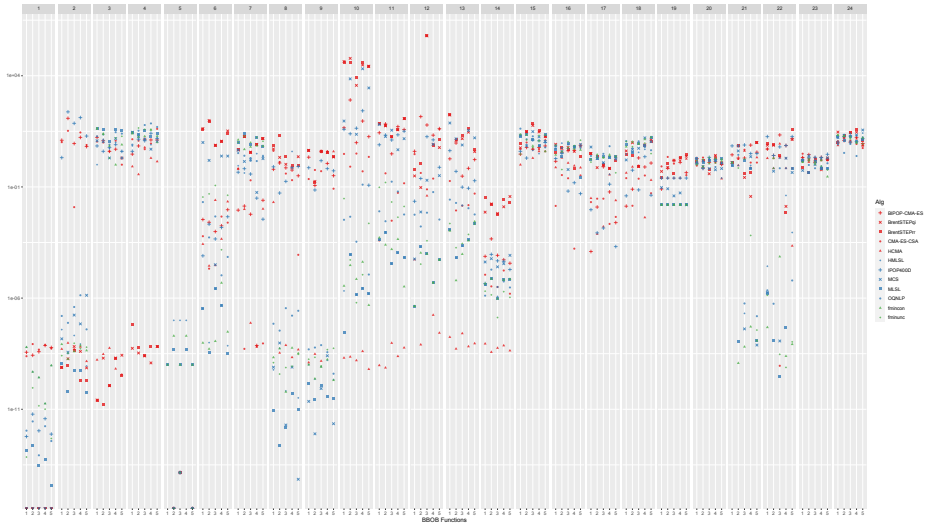


Figure 7.1: Single-run performances (measured by the target precision) of the 12 algorithms on the first five instances of 24 BBOB functions. They represent our portfolio from which we want to select the best-performing algorithm for an unseen problem instance.

Feature Computation. Again, predictor variables for our regression models are vectors of ELA feature values. Feature computation is done using the *flacco* package (presented in Section 4.3) for two distinct sets of uniformly sampled points and their evaluations. Samples are of sizes $50d$ (250) and $400d$ (2000) respectively for purpose of sensitivity analysis. 50 independent feature computations were performed for each sample size, as some of them can show low robustness on certain features [RDD+19], and a median feature value was taken for each one. Following suggestions from [BDS+17; KT19], we choose only those feature sets that do not require additional sampling during their computation; this way, we end up with a total of 56 feature values per problem instance.

7.2 Performance Regression Quality of Different Models

Using the key elements described in Section 7.1, we establish two separate regression (true and logarithmic) approaches for each of the 30 regression models, as suggested in Chapter 6. Following common machine learning practices, we perform a 5-fold

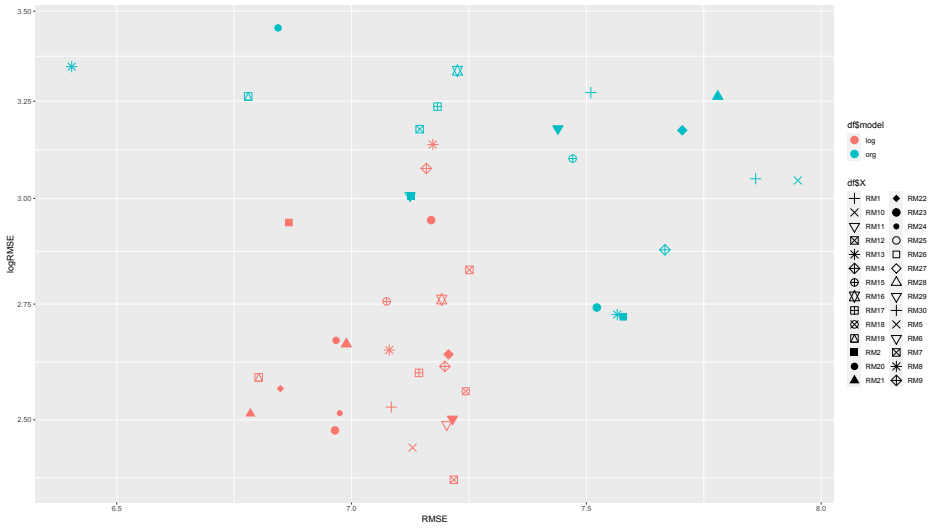


Figure 7.2: Overall prediction quality of different regression models (RMSE vs. log-RMSE), from both the unscaled (in blue) and the log-based approach (in red) on the CMA-ES-CSA algorithm, with the budget of 1000 function evaluations and 2000-sample feature size. Minimizing both error values as a Pareto front, we see that log-based regression models perform significantly better than the unscaled ones.

stratified *leave-one-instance-out* cross-validation when training each model to reduce variability and obtain a higher model accuracy, thus carrying out the training on four out of five instances per each function, testing on the remaining one and combining the results over all folds.

In order to assess and compare the accuracy of different predictions made by the regression models, we compute here again the RMSE and log-RMSE values per prediction, and aggregate them across problems and algorithms. The RMSE is computed based on the actual data values (both for the unscaled and logarithmic approach), whereas the log-RMSE makes use of the log-scaled data for both approaches.

Table 7.2 conveniently shows the best achieved prediction accuracies (RMSE and log-RMSE) of the logarithmic approach for different algorithms in the portfolio, for the 2000-sample feature size, and for all different budgets, aggregated across problems. Different 'Model' columns in the table correspond to the regression model with the best quality in a specific scenario. Note that the RM-labeled models 1–6 refer to the Decision Tree regressors, 7–18 to the Random Forest regressors, and 19–30 to the Bagging DT regressors. This allows us to draw some first conclusions related to selecting the most efficient regressor, with the Decision Tree family being predominantly chosen in the

low-budget setting when optimizing the RMSE, while the presence of the Bagging DT family is stronger when optimizing the log-RMSE in the same setting. Random Forest regressors, however, seem well-performing across the board, which is supported by the fact they are typically an all-round model of choice in related lines of research. Most importantly, we remark that, even when aggregated across problems, the choice of the best regression model is highly dependent on the setting we work in, and varies between algorithms.

Table 7.2: Best quality of the predictions of logarithmic regression models (RMSE vs. log-RMSE) for each algorithm from the portfolio, for feature size of 2000 samples. Note that the labels in columns named 'Model' correspond to the regression model (referred to as *RM* in the table) achieving the said best prediction quality. Different models are the best-performing ones across the portfolio for different budgets of function evaluations.

	Feature size 2000											
	Budget 250				Budget 500				Budget 1000			
Algorithm	RMSE	Model	logRMSE	Model	RMSE	Model	logRMSE	Model	RMSE	Model	logRMSE	Model
BIPOP-CMA-ES	10368.95	RM27	1.65	RM13	1489.86	RM9	1.83	RM20	73.18	RM7	1.94	RM12
BrentSTEPqi	58096.65	RM8	2.22	RM30	57828.22	RM9	2.76	RM12	57040.98	RM9	2.84	RM11
BrentSTEPrr	59475.56	RM3	2.15	RM30	58039.56	RM16	2.70	RM30	57993.93	RM21	2.90	RM12
CMA-ES-CSA	10216.53	RM7	1.62	RM13	280.26	RM13	1.95	RM30	6.78	RM21	2.38	RM12
fmincon	17.73	RM5	1.64	RM18	10.36	RM3	1.96	RM18	6.23	RM20	1.64	RM23
fminunc	254.93	RM21	2.20	RM11	7.05	RM7	2.34	RM29	6.83	RM2	2.35	RM29
HCMA	1651.12	RM13	1.70	RM23	4.29	RM23	2.10	RM18	2.82	RM14	2.77	RM12
HMLSL	16.83	RM2	1.62	RM11	10.45	RM6	1.95	RM23	4.01	RM24	1.87	RM12
IPOP400D	3120.75	RM14	1.72	RM18	1204.75	RM19	1.71	RM11	33.66	RM7	1.99	RM11
MCS	2514.61	RM2	2.40	RM24	2233.27	RM7	2.74	RM11	1992.51	RM24	2.87	RM11
MLSL	19.54	RM7	1.77	RM17	11.45	RM6	2.06	RM23	5.39	RM6	2.07	RM22
OQNLP	26.36	RM13	2.40	RM30	11.34	RM20	2.17	RM17	9.10	RM20	2.29	RM17

As illustrated in the example in Figure 7.2, which is the use-case of CMA-ES-CSA algorithm for budget of 1000 evaluations and 2000-sample feature size, the overall regression quality both in terms of RMSE (on x-axis) and log-RMSE (on y-axis), regarded as a Pareto front, i.e. a two-objective min-min problem, is higher for the log-based approach than for the unscaled one. However, it does not always have to be the case; depending on the algorithm, we can also observe situations in which the unscaled approach yields better results. Nevertheless, a remarkable diversity of regression model accuracies on different problems is easily noticed, and supports the idea of possibly having to resort to multiple regressors (each excelling in prediction of one sub-class of problems, for example) to achieve best results, which further supports the claim that ensemble-based models could be significantly more accurate than the standalone ones.

7.3 ELA-Based Algorithm Selection

After examining the regression accuracy, we proceed to evaluate the performance of two simple algorithm selectors, based on the predictions of the unscaled and the logarithmic approach, respectively, for each of the 30 regression models. The unscaled selector recommends the algorithm for which the unscaled-based model predicted the best performance, and, similarly, the log-selector bases its decision on the best prediction of the logarithmic model. To quantify how well selectors perform per problem instance, we compare the precision of the algorithm chosen by the selector for the instance at hand to the precision of the actual best algorithm for that instance. We then indicate the overall quality of this selector by computing the RMSE and log-RMSE values (aggregated across all problem instances).

Both in unscaled as well as in the logarithmic approach, some algorithms will be selected more often than others per different problem instance, as seen in Figure 7.3. For the budget of 1000 function evaluations and 2000-sample feature size, we observe that HCMA is chosen most consistently across different problem instances, while the choice of BIPOP-CMA-ES is very rare. We also notice that on specific instances, there could be one or two particularly frequently selected (thus good) algorithms which are not selected for other problems in the benchmark set (e.g., CMA-ES-CSA for the function 16 in the unscaled approach, or fmincon for the function 21 in the log-based approach).

Combined Algorithm Selector. We compare the qualities of our two selectors (unscaled and logarithmic) with two standard baselines – the virtual best solver (VBS) and the single best solver (SBS). The SBS baselines vary depending on the budget and feature size, and we can also distinguish between the unscaled approach SBS and the log-based approach SBS.

We plot in Figure 7.4 the quality of the selectors over all problem instances on RMSE and log-RMSE axes, again treating them as a Pareto front with the objective of minimizing both errors, and we see that different unscaled and log-based selectors already outperform the majority of single algorithms from the portfolio on both RMSE and log-RMSE. We also want to incorporate the observation reported in Chapter 6 that the unscaled approach is better in predicting higher target precision, while the log-based approach has better accuracy when targeting smaller performance values.

Again, we establish a combined regression approach for all the models in order to benefit from their two complementing strengths. We can thus define a combined VBS, which is the one that, for each model and problem instance, chooses the better of the two recommended algorithms by the unscaled and the log-based approach, respectively, using the following rule: if the target precision of an algorithm, as predicted by the logarithmic model, is smaller than a certain threshold, we use the log-based approach, whereas we use the recommendation of the unscaled approach otherwise. The threshold value chosen here is 0.9 in order to ensure selecting the log-based approach recommendation

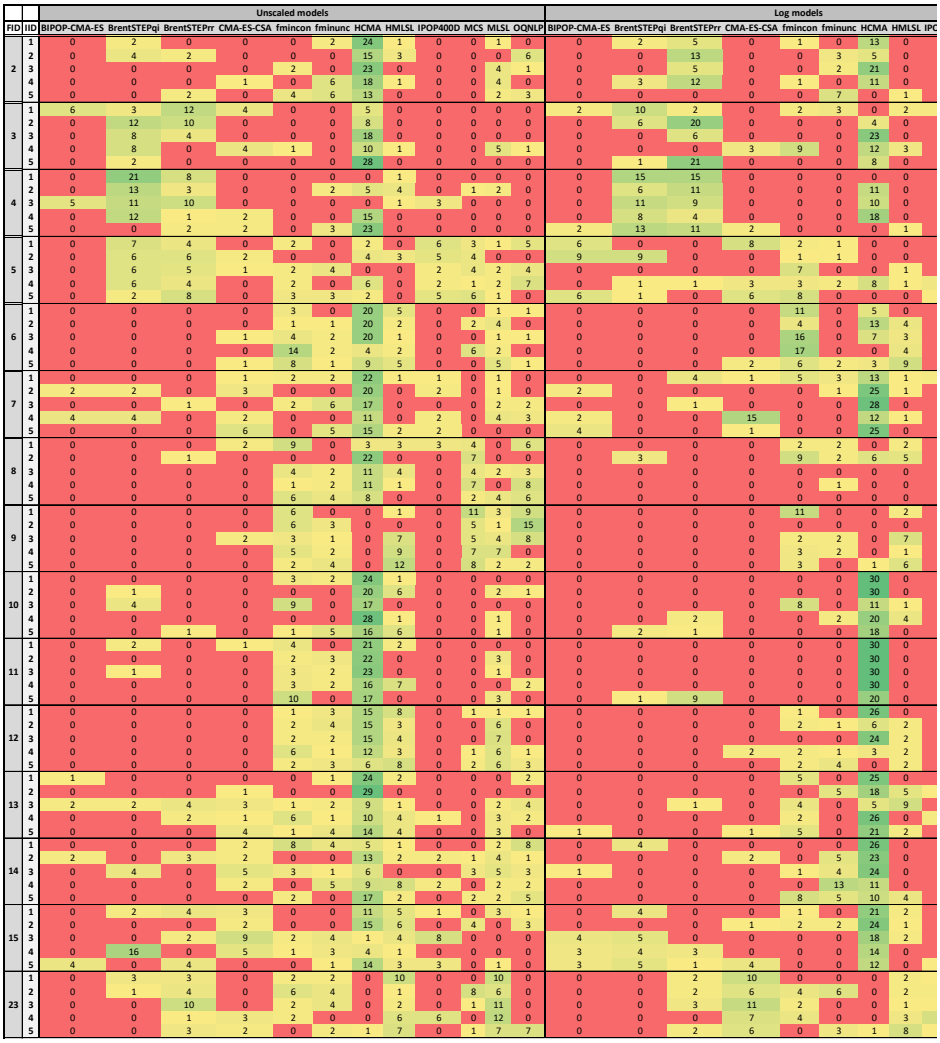


Figure 7.3: Heatmap of the selection frequency of each algorithm from the portfolio per problem instance, both for the unscaled (left) and log-based approach (right), for the budget of 1000 function evaluations and the feature size of 250 samples, showing selected functions only for reasons of space.

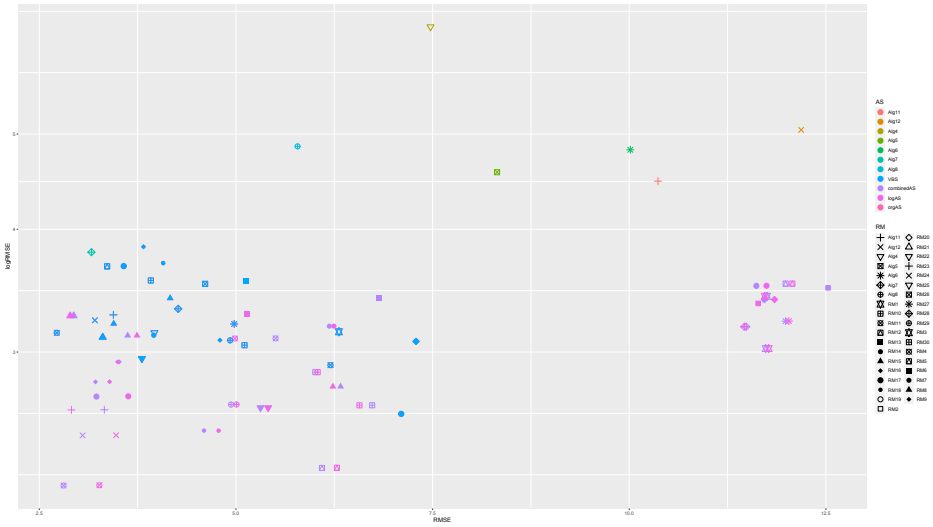


Figure 7.4: Quality of the algorithm selectors (RMSE vs. log-RMSE) for different regression approaches, including the combined selectors, the virtual best combined selectors (combinedVBS), and the selectors based on consistently using a certain algorithm across the board ('Algo' selectors in the legend, among which we can identify the SBS), for the budget of 1000 function evaluations and the 2000-sample feature size. Note that some of the selectors, notably the 'Algo' ones, have been excluded from the figure, as otherwise the visibility of the best ones is disrupted.

for fine-grained precisions and vice versa. We apply this strategy to all 30 regression models, and the results for the budget of 1000 evaluations and the 2000-sample feature size are seen in Figure 7.4. The combined selector clearly outperforms any of the simple regression approaches, the single algorithms and even the combined VBS selectors. This finding highlights the potential of the combined selector, which boosts the quality of its standalone components even in the case when they might not be the optimal regression models for a specific algorithm portfolio and problem set.

7.4 Sensitivity Analyses

Lastly, we highlight the differences in regression quality obtained by using different budgets and feature sizes. We have purposefully randomly selected one regression model (*RandomForest_crit.mse_minsplit.4_nest.9*) and one algorithm (HCMA) to point out the diversity of regression on different problem instances. The data in Table 7.3 corresponds to the log-based approach of said selector, using 2000-sample feature size, in order to

be consistent with the showcased use-case throughout the chapter. We immediately observe extremely large RMSE values for the budget of 250 function evaluations, which drastically decrease for the budgets of 500 and 1000. One possible interpretation of this finding could be that in raw performance data, whereas the budget of 250 evaluations was not big enough to allow for getting closer to the optimum in case of certain functions, increasing the budget seem to facilitate getting a better estimate of the optimal solution, thus making a more straightforward way for the regression to perform better. Finally, this again stresses the possible importance of personalizing the regression models to the actual optimization problems, as the results can vary drastically between problem instances.

Table 7.3: Sensitivity analysis of regression quality for the log-based approach of the *RandomForest_crit.mse_minsplit.4_nest.9* regression model for the HCMA algorithm for the feature size of 2000 samples, with respect to the different algorithm budgets. Note that, for reasons of space, we showcase only values of the first instance of the 24 BBOB functions.

	FID		1	2	3	4	5	6	7	8	9	10	11	12
	IID		1	1	1	1	1	1	1	1	1	1	1	1
Feature size 2000	Budget 250	RMSE	0.12	0.00	4.01	0.66	1.00	229.24	39.73	44127.00	0.01	21357821.80	3243.65	20853295.16
		logRMSE	0.03	15.35	0.01	0.00	14.80	0.40	0.21	1.12	0.00	1.94	0.04	0.65
	Budget 500	RMSE	0.06	0.00	0.16	1.77	1.00	4.83	0.52	10.35	0.13	0.00	0.00	0.00
		logRMSE	0.01	23.83	7.65	0.67	21.68	1.98	4.35	0.28	0.10	1.09	1.50	0.02
	Budget 1000	RMSE	0.01	0.00	2.58	0.66	1.00	0.00	0.41	0.00	0.00	0.00	0.00	0.00
		logRMSE	0.00	0.02	80.68	2.80	11.15	1.28	16.98	0.01	3.85	0.04	0.53	0.26

	FID		13	14	15	16	17	18	19	20	21	22	23	24
	IID		1	1	1	1	1	1	1	1	1	1	1	1
Feature size 2000	Budget 250	RMSE	53.95	1.12	656.09	341.45	7.21	38.76	0.09	6.81	0.49	0.72	3.89	247.95
		logRMSE	0.11	1.19	0.19	0.58	0.40	0.18	0.12	0.15	0.03	0.06	0.09	0.06
	Budget 500	RMSE	0.00	0.01	251.70	72.77	0.11	4.18	0.03	0.84	0.70	0.26	0.55	5.75
		logRMSE	0.67	13.13	6.62	0.39	1.09	2.49	0.06	0.09	0.04	0.02	0.02	0.01
	Budget 1000	RMSE	0.00	0.00	0.99	0.10	0.00	0.18	0.14	2.86	0.10	3.72	0.23	1.65
		logRMSE	0.15	1.01	33.19	0.02	2.96	2.86	0.15	1.48	0.02	3.48	0.01	0.00

7.5 Conclusions

While most landscape-aware algorithm selection, configuration, and design studies in the context of numerical black-box optimization do not pay great attention to the configuration of the machine model, we have demonstrated in this work that both the choice of regression technique and its parametrization can have significant impact on the performance of the trained models. Using a classical experimental design from the context of automated algorithm selection [JD20; KT19], we have analyzed 30 different regression models, applied them to both the logarithmic and to the unscaled performance

data, trained an automated algorithm selector, and analyzed its performance through stratified 5-fold cross validation. We have seen that the regression quality of the different models can vary by several orders of magnitude. This reinforces and justifies the need for meticulously choosing the machine learning model and its hyper-parameter configuration, as we have seen that picking any single model and tuning it might not at all provide good results (e.g., the discarded machine learning models from the preliminary step of this work). Differences in the regression models' quality also lead to very diverse performance portfolios in the algorithm selection task, although the impact there is somewhat less severe, since wrong predictions can still result in a lucky choice of algorithms. Additionally, when considering ensembles, hyper-parameter tuning can be expected to further improve upon performance gains, which will depend on the used data set and learning scenario.

Our study suggests that the selection of the machine learning techniques should be performed with care. It also suggests that the quality of different regression techniques can vary between different types of problems, so that we cannot give a "one-size-fits-all" recommendation for which regression models or which parametrization to favor. As a rule of thumb, different Bagging Decision Tree and Random Forest instances provide better results in terms of log-RMSE, for both feature extraction sample sizes and all three budgets of function evaluations. For the RMSE performance criterion, in contrast, Decision Trees provides best results for some of the algorithms. The important question of choosing the absolute best performing machine learning model for a certain problem set and algorithm portfolio remains open, but this preliminary work stresses the significance of the efforts that should go towards developing more advanced mechanisms to select the most appropriate one and its hyper-parameter configuration.

We note that the computational costs of training different regression models is rather negligible for the data sizes commonly studied in the landscape-aware black-box algorithm selection context, which means a validation step can be added before deciding which model to choose and apply to the real use-cases (i.e., in the *test* phase). In general, we believe that the current common practice of studying stratified 5-fold cross validation on the BBOB functions is too limited to give an accurate impression about the potential of carefully choosing the machine learning model for the automated algorithm selection and configuration in practice. We therefore plan to massively extend our study by adding to our data sets performance data from the Nevergrad platform [RT18], which offers benchmark data for very broad ranges of optimization problems on its frequently updated dashboard.

One problem to overcome in cross-validation across different benchmarking platforms is the fact that one needs to ensure that the data corresponds to the same instances of the algorithms – a CMA-ES implementation in one platform may be much different than a CMA-ES implementation in another. We therefore believe that a common algorithm repository, interfaced with the various benchmarking suites, would be a useful step

towards a better re-usability of results and better training sets for the automated selection, configuration and design of black-box optimization techniques.

8 Personalized Performance Regression

This chapter is based on paper [EJP+21], which appeared in the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2021, ***Personalizing Performance Regression Models to Black-Box Optimization Problems****.*

8.1 Preliminaries

Summary of Results. The main goal of this chapter is to analyze to what extent state-of-the-art landscape-aware algorithm selection models could benefit from a more careful choice of the machine learning tools, and, more concretely, how their complementarity can be leveraged to obtain good predictions for broad sets of optimization problems. We evaluate a way of extending the current practice of deploying a single predictive model by combining ensemble regression and personalized regression.

While ensemble learning is well-known and the de-facto standard in several machine learning applications [ZM12], the idea to personalize the regressions is an original research contribution that we propose here. In a nutshell, the key idea is that different regression models work best for different types of problems, so that we can improve regression quality by automatically selecting the one(s) that showed best performance for similar problems.

We test the impact of each of the suggested extensions on a portfolio of 12 algorithms from the BBOB workshop series [HAB20] suggested in [KT19], for which we task ourselves with predicting the solution quality after a fixed budget of function evaluations.

We find that the regression quality improves for 58%-70% of the tested problems, depending on the comparison scenario, which nicely demonstrates that the current practice in performance regression used within the evolutionary computation community has quite some untapped potential.

The computational overhead for training and applying the personalized models is also negligible for the tasks performed in this study. This happens because we are working only with 120 instances from 24 problems. When moving to larger data sets of several GB or even TB in size (as often considered in machine learning applications), however, we need to consider the impact of data size and data quality on the learning algorithms performances. With larger amounts of data, the so-called offline learning (as used in this paper) can become computationally inefficient. In such cases, one option is to use random or stratified sampling if possible, which can reduce the data size and still preserve the relevant information found in the original data set.

The idea to personalize the regression models is not restricted to performance regression, and not even to optimization. Based on our findings presented here, we consider further applications, for example in personalized medicine, as an exciting avenue for future work, since predictive models that are specifically developed for different genotypes and/or phenotype may allow better recommendations that "one-size-fits-all" predictive models, which are unfortunately known to come with biases that can cause severe harm.

Note that personalized approach is different from training individual models separately, in that the classification which model to use is done in a data-driven way, and not by an external entity. While in this chapter we use classification to assign problem instances to problem classes, our approach can easily be extended to allow for interpolation between personalized models. How much such an additional layer would contribute, however, remains to be evaluated in future work. We compare our results to the approach in which the classification step can be omitted and the ground truth problem class is known (this is the "ensemble-ground" method in Section 8.3.1). Based on the results, the misclassification of the problem class that leads to selecting a personalized model does not have a significant loss. On the contrary, it can even improve the end predictions, however, this happens in cases where landscape representations of the problems instances are quite similar.

8.2 Personalized Machine Learning Models

To introduce our personalized performance regression pipeline, we assume that we face a fixed-budget performance regression task, i.e., we aim at predicting algorithms' solution quality after a fixed budget of function evaluations has been exhausted. The pipeline presented below can be used for other machine learning tasks, but the restriction to a specific use case eases the presentation considerably.

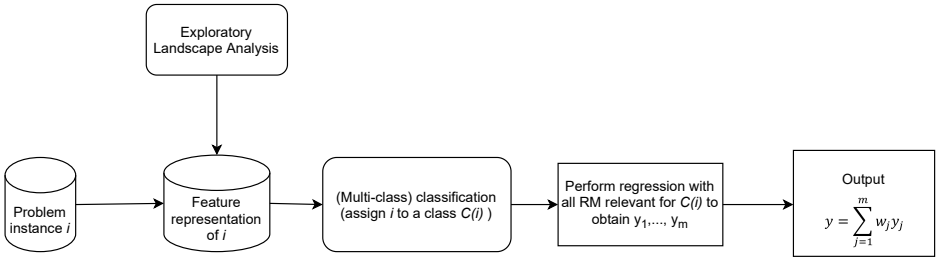


Figure 8.1: Application of the personalized machine learning models. Note that the abbrevation *RM* stands for regression models.

The high-level approach is depicted in Figure 8.1. The main steps to obtain the performance prediction y for a given problem instance i are the following:

1. We first apply a feature extraction method to obtain a description of this instance i. In our context, the features are computed via exploratory landscape analysis (see Section 8.3 for details).
2. We use the instance description to assign instance i to a class $C(i)$ (i.e., we perform a multi-class classification) to obtain a set $\{y_1, \ldots, y_m\}$ of different performance predictions, one per regression model.
3. We combine these values to obtain the end prediction $y = \sum_{j=1}^{m} w_j(i) y_j$, using the weighting scheme $w_1(i), \ldots, w_m(i)$ of the class $C(i)$.

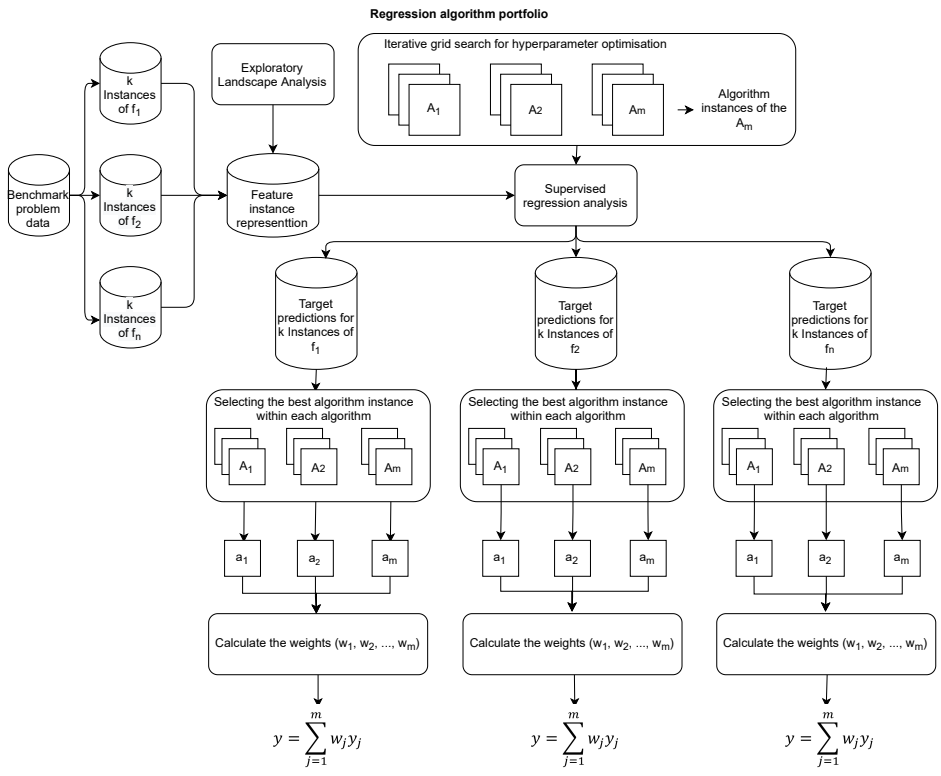


Figure 8.2: Training phase of the personalized problem ensembles.

The association of the regression models to different classes, as well as the computation of the class-specific weights, is handled in a prior (i.e., "offline") **training phase**. Its most relevant steps are illustrated in Figure 8.2.

Assuming that we have a set of training instances which are grouped into n classes $C_1, \ldots, C_n$ (in our case, these are the problem instances), a set of potential regression models, which are grouped into m classes $A_1, \ldots, A_m$,[1], and fixed-budget performance data for an algorithm $\mathcal{A}$, then the training phase comprises the following steps:

1. We compute a representation for each training instance, ideally using the same feature extraction technique that will be used in the applications (i.e., in the *test phase*, in proper machine learning terminology).
2. Each regression model instance uses the problem representation and the algorithm performance data to train a predictive model.
3. Each regression model is evaluated according to its regression performance on the training instances within each optimization problem.
4. For each problem class $C(i)$, we select from each regression model class A_j the configuration $a_j(i)$ which achieved the best performance.
5. We then calculate the importance of each regression model $a_j(i)$ via a min-max normalization. That is, if we denote by $q(i) = (q_1(i), \ldots, q_m(i))$ the vector of performance measures for each of the m selected configurations for the class $C(i)$, the importance of $a_j(i)$ is computed as

$$q_{j,\text{norm.}}(i) = \frac{\max(q(i)) - q_j(i)}{\max(q(i)) - \min(q(i))}, \tag{8.1}$$

   where we assume a performance measure for which lower values are better (typically, a deviation from the ground truth is measured in one way or the other). We then compute the vector $w(i) = (w_1(i), \ldots, w_m(i))$ of weights $w_j = \frac{q_{j,\text{norm.}}(i)}{\sum_{j=1}^{m} q_{j,\text{norm.}}(i)}$, which are used in the third step of the application phase described above.

8.3 Use-Case: ELA-Based Fixed-Budget Performance Regression

We evaluate our personalized machine learning pipeline on a standard fixed-budget regression task, which aims at predicting the final solution quality of a black-box optimization algorithm after a fixed number of function evaluations. The experimental setup is described in Section 8.3.1. In total, we apply our approach to twelve different optimization algorithms. We present here only some selected results (Section 8.3.2). A

1 In this chapter, we group in one class all regression models that differ only in the hyper-parameters, but use the same basic regression technique.

few sensitivity analyses, to test the robustness of our approach, are performed in Section 8.3.2.

8.3.1 Experimental Setup

Algorithm Portfolio, Benchmark Problems, Feature Computation. As in Chapter 7, we aim at predicting the performance of the following 12 algorithms (presented in Chapter 3): BrentSTEPqi [PB15], BrentSTEPrr [PB15], CMA-ES-CSA [Ata15], HCMA [LSS13a], HMLSL [Pál13a], IPOP400D [ABH13], MCS [HN09], MLSL [Pál13a], OQNLP [Pál13b], fmincon [Pál13b], fminunc [Pál13b], and BIPOP-CMA-ES [Han09].

As performance measures of these algorithms, we use their single-run fixed-budget target precision after 250, 500, and 1000 fitness evaluations, respectively, and this for the first five instances of each of the 24 BBOB functions.

The representations of the 120 problem instances are based on exploratory landscape analysis (ELA). The ELA features were computed via the `flacco` package, using the uniform sampling procedure with a budget of $400d$ (a sensitivity analysis for a $50d$ sampling budget will be presented in Section 8.3.2). Following similar reasoning as in Chapters 6 and 7, we used 56 feature values per instance (all presented in Chapter 4). To stick to common practice in the evolutionary computation community, we take the raw feature values, i.e., we do not normalize these values nor do we perform any representation learning prior to feeding the values to our machine learning models.

Personalized Ensembles. To evaluate the personalized ensembles, we use stratified 5-fold cross-validation, where each fold consists of the first, second, third, fourth, and fifth instance for each problem, respectively. That is, we repeat the whole training and testing process described in Section 8.2 five times, each time leaving out one fold for the test phase and using the other four for the training. Note that the personalized ensembles for the same problem can be different across the five different runs, since different training data is used. An example will be presented in Table 8.3.

In a preliminary evaluation of our personalization approach, similarly to Chapter 7, we used seven regression techniques: Lasso [Tib96], Elastic Net [ZH05], Kernel Ridge [Mur12], Passive Aggressive [CDK+06], Decision Tree [BFS+84], Random Forest [Bre01], and Bagging Decision Tree [Bre96]. We apply iterative grid search to each of them in order to test different hyper-parameters. Evaluating the seven regression techniques using the mean absolute errors (MAE) of the test folds from the stratified 5-fold cross-validation, only three regression techniques were selected for further investigation, Decision Tree, Random Forest, and Bagging Decision Tree. All tested hyper-parameter combinations for each of these techniques are summarized in Table 8.1.

Since all selected techniques are based on trees, the *crit* parameter can be one of the following: “mse" (mean squared error), “mae" (mean absolute error), and “friedman_mse" (Friedman mean squared error). With respect to the *minsplit* hyper-parameter, it is

the minimum number of data instances a node contains in order to be split. The *nest* hyper-parameter defines how many decision trees will be built in the Random Forest/Bagging Decision Tree regression model. These range of the hyper-parameters have been selected with respect to the data set size and the general machine learning guidelines in order to avoid overfitting. In total, we end up with 430 different regression models: 30 configurations of Decision Trees, 200 configurations of Random Forests, and 200 configurations of Bagging Decision Trees.

Table 8.1: Hyper-parameter values for each regression model class.

Algorithm	Hyperparameters
Decision Tree (30 configs.)	• $crit \in \{"mse", "mae", "friedman\_mse"\}$ • $minsplit \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$
Random Forest (200 configs.)	• $crit \in \{"mse", "mae"\}$ • $minsplit \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$ • $nest \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$
Bagging DT (200 configs.)	• $crit \in \{"mse", "mae"\}$ • $minsplit \in \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$ • $nest \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$

To select the best regression model from the three selected regression techniques, and to learn the weights for each problem class separately, we use the *mean absolute error (MAE)* of the results obtained on the training fold. We purposefully opt for doing so, since the data set we used is relatively small, and we are not able to split it properly into classical sets for training, validation, and testing. Going forward, when working with much larger data sets, it will be crucial to use validation sets for the weight computation. In the current context, however, this can lead to further overfitting on the training set, but, on the other hand, it can also provide some preliminary insight into how the proposed methodology fits to the new test problem instances.

To associate an instance i to a problem class $C(i)$, we train an ensemble with the majority vote of three multi-class classification algorithms (*BaggingDT_crit-entropy_minsplit-2_nest-9, RandomForest_entropy_nest-9_min-2, and RandomForest_gini_nest-9_min-2*), and we do so on the exact same folds used for building the personalized regression models. The hyper-parameters used for training the classifiers are the same as for the regression models, with the only difference lying in the fact that Gini impurity (*gini*) or the Information Gain (*entropy*) [FHT+01] criteria were used for splitting the nodes in individual trees. Both these criteria measure the impurity of a node.

The comparison is done in the following scenarios, which are summarized in Table 8.2:

- **Ensemble-ground**: this so-called "ground truth" scenario corresponds to personalized ensembles for each problem. The true problem class, $C(i)$, which the test problem instance i belongs to, is known as *a priori* information. That is, we assume in this model that we know which problem class the instance belongs

Table 8.2: Regression models used for evaluation purposes.

Regression model	Personalized	Ensemble	Selection		Classification
			MAE on train	MAE on test	
Ensemble-ground	✓	✓	✓	-	-
Ensemble-class	✓	✓	✓	-	✓
Best-train	-	-	✓	-	-
Best-train-instance	✓	-	✓	-	-
Best-test	-	-	-	✓	-

to, and our key objective is thus to evaluate how appropriate the class-specific ensemble is.

- **Ensemble-class**: this is the approach described in Section 8.2, i.e., we have at our disposal the personalized ensembles for each problem, and the problem class $C(i)$ needs to be guessed by the classifier from the instance representation. When the classifier correctly predicts the true problem class, this prediction is identical to the one of the Ensemble-ground model. If, on the other hand, the instance is misclassified to a different problem class, the noise presented in the classifier will affect the selection of the relevant regression model, which, in turn, influences the end prediction (this can go both ways, as we shall see below).
- **Best-train**: in this scenario, the best regression model from the three regression techniques is selected based on the mean absolute error obtained across all problems from the training folds (i.e., one regression model for all problems).
- **Best-train-instance**: here, the best regression model is selected in the same scenario as the Best-train, but for each problem separately (i.e., not across all problems). Selecting the Best-train-instance model for each problem is a special case of personalized approach (i.e., each problem has its own best regression model, but we do not combine the predictions of several models for the final output).
- **Best-test**: lastly, the best regression model from the three regression techniques is selected based on the mean absolute error obtained across all problems from the test folds (i.e., one regression model for all problems).

Note that the first four abovementioned models are learned without evaluating problem instances from the test folds, whereas testing is needed to select the Best-test regression model.

8.3.2 BIPOP-CMA-ES Performance Prediction

To evaluate the proposed methodology, we explore the scenario of BIPOP-CMA-ES performance prediction. The experiment is performed for a fixed budget of 1000 function

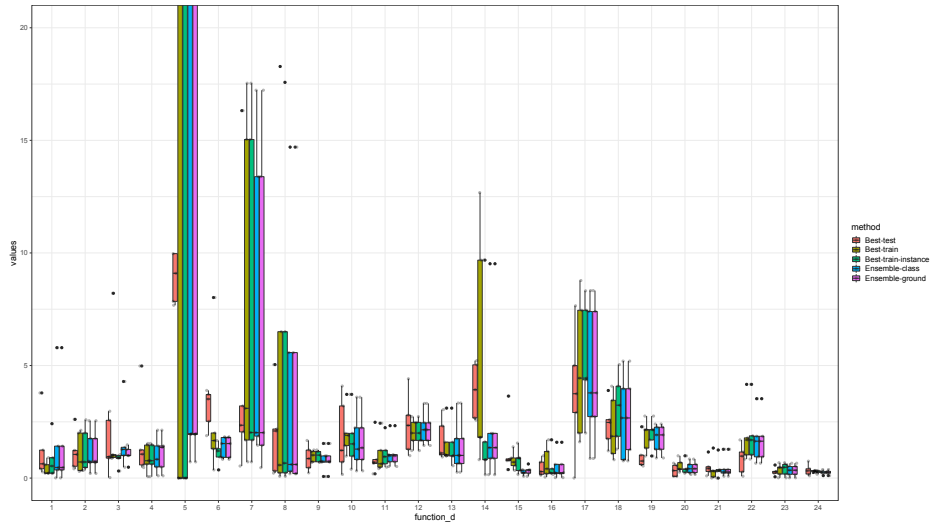


Figure 8.3: Evaluation results of the BIPOP-CMA-ES performance prediction. The y-axis corresponds to the absolute error between the ground truth and predicted target precision (i.e., the logarithm of the target precision), while the x-axis corresponds to each BBOB benchmark problem. The box plots represent the distribution of the absolute error obtained from each test fold for each problem separately in five different scenarios: a) Best-test, b) Best-train, c) Best-train-instance, d) Ensemble-class, and e) Ensemble-ground.

evaluations, and with a feature portfolio calculated using a $400d$ sample size. The personalized ensembles were trained in two scenarios: once to predict the original target precision achieved, and once to predict the logarithm of the target precision. Regardless of how the target is represented, the benefits of using the proposed methodology is the same. We present going forward the logarithmic models in greater detail.

Figure 8.3 shows the distribution of the absolute error obtained from each test fold for each problem separately in five different scenarios: a) Best-test, b) Best-train, c) Best-train-instance, d) Ensemble-class, and e) Ensemble-ground. The Best-test model is *RandomForest_crit-mse_minsplit-6_nest-20*, while the Best-train model is *DecisionTree_crit-mae_minsplit-4*. Table 8.3 highlights the models used in the personalized ensembles for the BBOB problem F6 (i.e., Attractive Sector Function) for each fold.

We note that the performance quality of the tested approaches in the following paragraphs is assessed via the *median absolute error*.

Table 8.3: Regression models used in the personalized ensembles for the BBOB function F6 in each fold.

Fold	Models
1	DecisionTree_crit.mse_minsplit.4 RandomForest_crit.mae_minsplit.2_nest.90 BaggingDT_crit.mae_minsplit.2_nest.10
2	DecisionTree_crit.mae_minsplit.4 RandomForest_crit.mse_minsplit.6_nest.90 BaggingDT_crit.mae_minsplit.6_nest.10
3	DecisionTree_crit.mse_minsplit.4 RandomForest_crit.mse_minsplit.2_nest.20 BaggingDT_crit.mae_minsplit.10_nest.10
4	DecisionTree_crit.mae_minsplit.4 RandomForest_crit.mae_minsplit.4_nest.30 BaggingDT_crit.mae_minsplit.2_nest.10
5	DecisionTree_crit.mae_minsplit.4 RandomForest_crit.mse_minsplit.2_nest.70 BaggingDT_crit.mse_ minsplit.10_nest.10

Comparing a single regression model vs. personalized ensembles. The following analysis involves comparing the results obtained by using a single regression model that works well across all problems (i.e., Best-train or Best-test) with the results obtained using the personalized ensembles (i.e., Ensemble-class and Ensemble-ground).

By comparing the Best-train, Ensemble-class, and Ensemble-ground models, while inspecting the medians from the box plots, it is obvious that the personalized ensembles (i.e., Ensemble-class, and Ensemble-ground) outperform the Best-train for 14 out of 24 BBOB problems (i.e., 6, 7, 9, 10, 13, 14, 15, 16, 17, 19, 21, 22, 23, and 24). This comparison comprises models that have never seen the test instances. On the other hand, Even more promising results are obtained when comparing the Ensemble-class and Ensemble-ground models with the Best-test model. In this case, the personalized ensembles are better in 15 out of 24 BBOB problems (i.e., 1, 2, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 21, and 24). It is important to note that the Best-test model is selected based on the information from the test instances, that, as pointed out above, have never been seen by the personalized models. Table 8.4 presents the median absolute error obtained from each test fold for each problem separately by four different regression models. Comparing the median absolute error values, we observe that the gain achieved by using the personalized ensembles varies between the problems, but we also note that this results from the different target precision ranges between problems.

Comparing the ground-truth personalized ensembles with the personalized ensembles combined with classification. By comparing the median values between the Ensemble-class and Ensemble-ground models, we can actually see the influence of the classifier on the end result. The difference between the end prediction results obtained by both models means that the classifier predicted the wrong problem class. This happens for four BBOB problems (i.e., 3, 4, 10, and 15). In the case of the fourth and the fifteenth problem, the misclassification actually improves the end target prediction. In order to see which regression models are selected and combined to generate the personalized ensemble, we bring the confusion matrix of the classification into play. For the fourth problem, the misclassification happens in the third test fold, where the instance from the fourth problem class (i.e., Büche-Rastrigin function, which is a separable function) is assigned to the third problem (i.e., Rastrigin function, which is also a separable function). For the fifteenth problem (i.e., Rastrigin Function, which is a multi-modal function with adequate global structure), the misclassification happens in the first test fold, where the classifier identifies it as the third problem (i.e., Rastrigin Function, which is a separable function).

These results open up several new directions for future work; instead of training personalized ensembles on the problem level, we can shift our perspective to learn them for a whole group of instances belonging to the same cluster. This can be obtained by clustering the ELA representations of the problem instances.

Comparing a single personalized regression model with personalized ensembles. To delve even deeper into analyzing the potential of the personalized approach, we now compare the Ensemble-class and Ensemble-ground personalized models to the Best-train-instance model. In this scenario, we have restricted ourselves to only the best regression model for each problem separately, which was learned using the performance obtained from the training folds, excluding the information from the test instances in the selection. By looking at the median absolute error across the test folds, we see that the personalized ensembles perform better than the Best-train-instance models for 13 out of 24 BBOB problems (i.e. the problems: 1, 7, 8, 9, 10, 15, 17, 18, 19, 21, 22, 23, and 24). Since the personalized ensembles are based on combining different regression models, the Best-train-instance model for each problem is actually one of the three regression models that are being combined. In most cases, combining the best regression model (i.e., Best-train-instance) with regression models from the other two regression techniques improves the end prediction. However, there are also cases where we do not observe the improvement of the end prediction, which opens up the question of how to select the regression techniques that should be included in the ensemble learning.

Comparison of mean absolute error vs. median absolute error. We now perform a final comparison with respect to the chosen performance quality metric. In

Table 8.4: Median absolute errors obtained from each test fold for each problem separately in four different scenarios.

Problem	Best-test	Best-train	Ensemble-ground	Ensemble-class
1	0.6337	0.2170	0.4718*	0.4718*
2	1.0507	0.7152	0.7478*	0.7478*
3	0.9530	0.9963	1.0053	1.2729
4	1.0627	0.7711	1.3661	0.8353*
5	9.0958	0.0000	1.9736*	1.9736*
6	3.5115	1.6669	1.5341$^{\triangle}$	1.5341$^{\triangle}$
7	2.3472	3.0982	2.0179$^{\triangle}$	2.0179$^{\triangle}$
8	2.0920	0.5789	0.6052*	0.6052*
9	0.8675	1.0208	0.7480$^{\triangle}$	0.7480$^{\triangle}$
10	1.2298	1.8954	1.3431$^{\diamond}$	1.2865$^{\diamond}$
11	0.7064	0.6347	0.9910	0.9910
12	2.3399	1.9997	2.1457*	2.1457*
13	1.1155	1.0443	1.0073$^{\triangle}$	1.0073$^{\triangle}$
14	3.9245	1.8126	1.3522$^{\triangle}$	1.3522$^{\triangle}$
15	0.8055	0.7086	0.3540$^{\triangle}$	0.2452$^{\triangle}$
16	0.2879	0.4122	0.2404$^{\triangle}$	0.2404$^{\triangle}$
17	3.7476	4.4404	3.7838$^{\diamond}$	3.7838$^{\diamond}$
18	2.4728	1.8581	2.6707	2.6707
19	0.7567	2.1484	1.9176$^{\diamond}$	1.9176$^{\diamond}$
20	0.3334	0.4013	0.4151	0.4151
21	0.4166	0.3046	0.2726$^{\triangle}$	0.2726$^{\triangle}$
22	0.9757	1.6932	1.6386$^{\diamond}$	1.6386$^{\diamond}$
23	0.2916	0.4672	0.3494$^{\diamond}$	0.3494$^{\diamond}$
24	0.3209	0.3103	0.2335$^{\triangle}$	0.2335$^{\triangle}$

$^{\diamond}$ Better than Best-train.
* Better than Best-test.
$^{\triangle}$ Better than both, Best-train and Best-test.

the case when the mean absolute error is used instead of the median absolute error, the personalized ensembles (i.e., Ensemble-class, and Ensemble-ground) outperform the Best-train in 17 out of 24 BBOB problems (i.e., 3, 4, 5, 6, 7, 8, 9, 10, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, and 24). When comparing them to the Best-test model, they are better in 11 out of 24 BBOB problems (i.e., 3, 5, 6, 9, 10, 12, 13, 14, 15, 21, and 24). Finally, when the comparison is done to the Best-train-instance, the personalized ensembles are better in 17 out of 24 BBOB problems (i.e., 2, 5, 7, 8, 9, 10, 13, 15, 16, 17, 18, 19, 20, 21, 22, 23, and 24).

Sensitivity Analyses

To investigate the impact of the different steps used by the methodology on the end predictions, we compare the Best-test, Best-train, and Ensemble-class models in three different scenarios:

1. We investigate how different sample sizes required to compute the ELA features influence the prediction of the reached target precision;
2. We compare results for different budgets, for one fixed optimization algorithm and a fixed feature portfolio;
3. We compare results for different optimization algorithms, for a fixed feature portfolio and a fixed budget.

Fixed budget, one optimization algorithm, different sample sizes for feature computation. To investigate the impact that different sample sizes required to compute the ELA features have on the end predictions, the 56 selected ELA features were computed using $50d$ and $400d$ sample sizes. This yields two distinct feature portfolios, which were further used as input data to learn the personalized ensembles for the BIPOP-CMA-ES in the fixed-budget scenario, with the budget of 1000 evaluations. Figure 8.4 presents the relative advantage of the the Ensemble-class-400d model vs. the regression models (Best-train-50d, Best-test-50d, Best-train-400d, Best-test-400d, Ensemble-class-50d) for each problem separately. The suffix $50d$ or $400d$ in the name of each model denotes the sample size of the feature portfolio used for learning. To estimate their advantages, we look into the difference between the median absolute errors. Positive values indicate where the Ensemble-class-400d model performs better than the other models, while negative values indicate the opposite. As seen in the figure, we can safely conclude that the Ensemble-class-400d model outperforms the other models for the majority of the problems. Comparing the Ensemble-class-50d and the Ensemble-class-400d (i.e., the blue line), it follows that using the $400d$-sample-feature portfolio provides much better results in most of the problems. Nevertheless, there are some problems (i.e., 17, 19, 22) for which the opposite is true, with rather insignificant

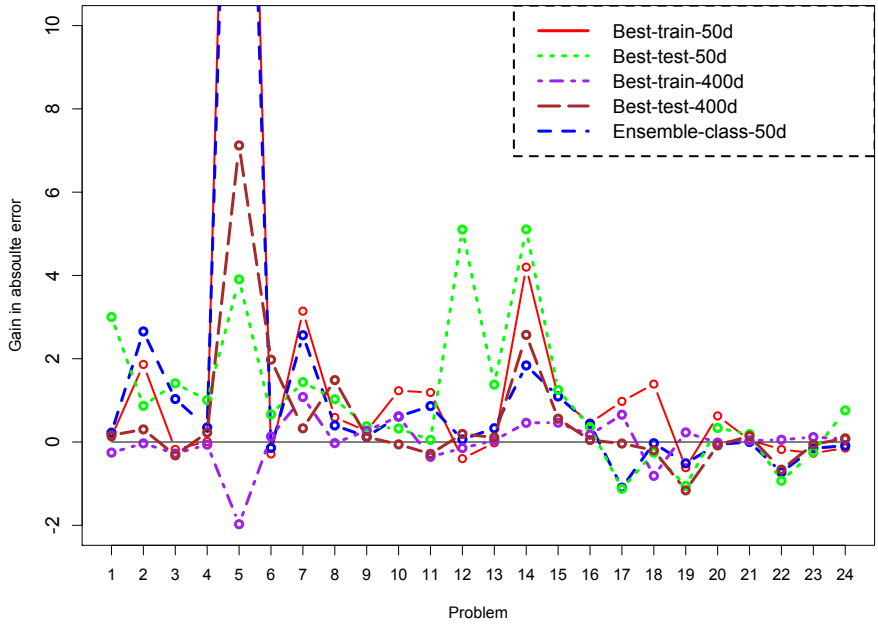


Figure 8.4: Relative advantage of the Ensemble-class-400d model vs. simple regression models (Best-train-50d, Best-test-50d, Best-train-400d, Best-test-400d, Ensemble-class-50d). The y-axis represents the difference between the median absolute errors. Positive values indicate where the Ensemble-class-400d model performs better than the other models, while negative values indicate the opposite.

differences in the median absolute errors. These results indicate that the selection of the sample size required to compute the ELA features can influence the end prediction. The findings also raise a question of the robustness of the ELA features for different samples sizes, which has been already discussed and investigated from another perspective in [RDD+20].

Fixed feature portfolio, one optimization algorithm, different budgets. Figure 8.5 presents the relative advantage of Ensemble-class model vs. Best-train model for each of the 3 budgets (250, 500, 1000), in the case when we analyze the log-scale performance of the BIPOP-CMA-ES. Here, the feature portfolio is fixed and computed using a 400d sample size. Positive values indicate where the Ensemble-class model performs

better than the Best-train model. The figure implies that personalized ensembles work well also for small budgets. By looking at problem 7, it seems that, based on its ELA representation, we can have a good prediction of the target precision reached after 250 and 1000 evaluations, but the Best-train model achieves better performance when the budget is 500. The limitation here is that the ELA representation of a problem instance is static and the same for all budgets, with the only difference in the reached target precision. This further means that the ELA representation does not cover information about the algorithm's behavior (i.e., which parts of the decision space are visited until some budget is exhausted). In order to improve this, additional information about the state of the algorithm should be considered and used as input data to train the personalized ensembles for predicting performance in different budgets. This has been looked into in Chapter 10.

Fixed budget, fixed feature portfolio, different optimization algorithms. To verify the transferability of the proposed methodology to algorithms other than BIPOP-CMA-ES, we highlight here the results obtained by personalized ensembles when trained to predict the performance of CMA-ES-CSA and IPOP400D, with a fixed budget of 1000 evaluations, and with a fixed feature portfolio calculated using $400d$ sample size (Figure 8.6). These insights confirm once again that using personalized ensembles improves the end prediction for most of the problems for both algorithms.

8.4 Conclusions

In this chapter, we present the idea of achieving high-quality performance prediction for optimization algorithms by means of selecting a regression model (or an ensemble) for a problem type. Our results demonstrate that there is quite some untapped potential in moving from "generalist" regression models that work well across broad ranges of optimization problems to more problem-specific, personalized regression models. The sensitivity analyses confirm the robustness of our approach.

We note that our study should be seen as a first prototype only. Several extensions are not only possible, but also strongly needed. For example, we need to evaluate our methodology on much bigger data sets, to allow for a proper split into training, validation, and test sets. Within such a setting, the training instances would be used to train the regression models, the validation instances to select and to evaluate the regression models which are to be included in the ensembles (this comprises the association of the importance weights that are used to calibrate the predictions of the different models). The test instances are then used to assess the performance of the overall pipeline.

With respect to combining the output of different regression models into one prediction, we plan on evaluating different approaches to derive the weighting schemes. In particular, we believe that a multi-criteria approach to combine different regression

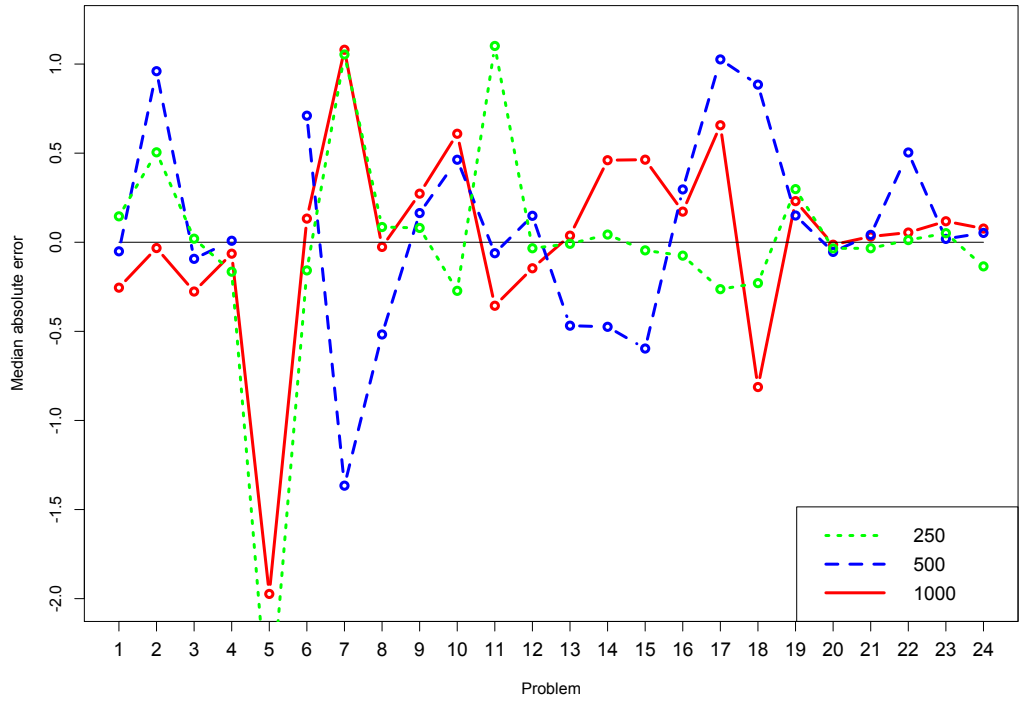
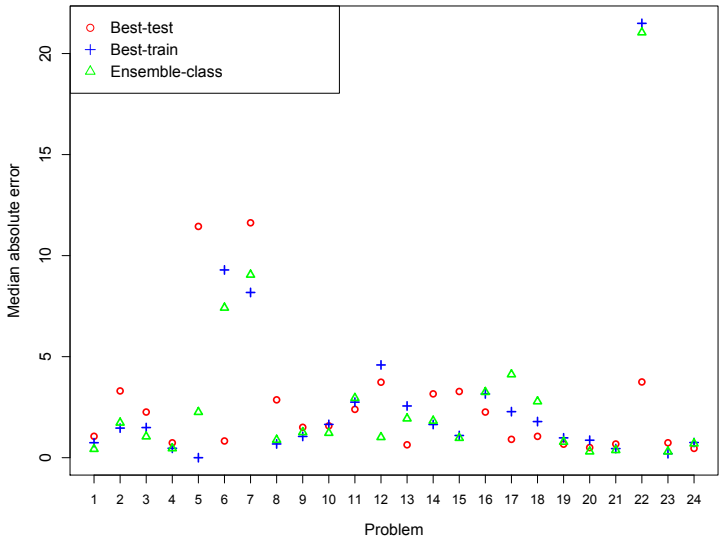


Figure 8.5: Relative advantage of the Ensemble-class model vs. the Best-train model for each of the 3 budgets (250, 500, 1000 function evaluations). The y-axis represents the difference between the median absolute errors. Positive values indicate where the Ensemble-class model outperforms the Best-train model.

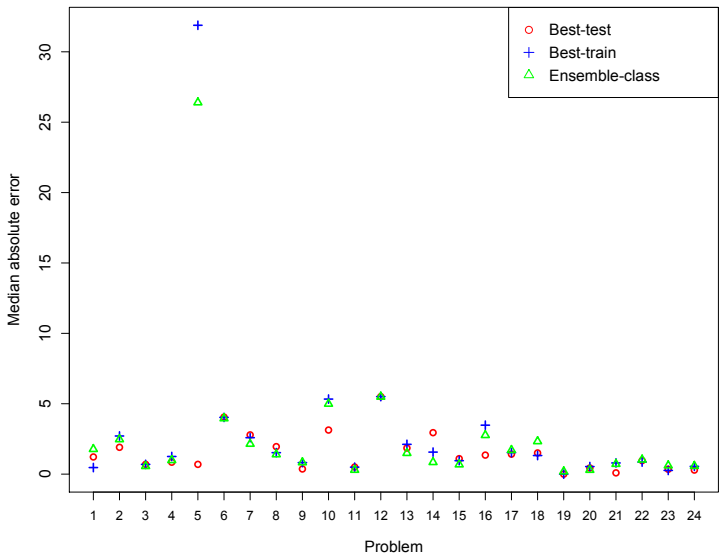
performance measures (such as mean root square error, correlation coefficients, etc.) could be promising, to balance the complementary information obtained through each of these statistics.

We used here a multi-class classification to assign problem instances to problem classes. In practice, instances may stem from problem classes that are not used in the training phase, so that the classifier cannot assign it to one of the present classes. In such cases, the classification step can be exchanged with clustering, which returns its k closest problem instances. The personalized ensembles for the selected problem instances would then be used to compute the performance prediction, which would be further merged with some heuristic of choice to generate the end prediction.

Finally, we plan to evaluate the personalized ensembles trained on one benchmark suite (e.g., the BBOB functions) on other benchmark suites (e.g., Nevergrad [RT18]), in order to investigate the transferability of the models between the different benchmark collections.



(a) CMA-ES-CSA



(b) IPOP400D

Figure 8.6: Median absolute error between the ground truth and predicted target precision (i.e., the logarithm of the target precision) for each BBOB benchmark problem, for fixed budget 1000, fixed feature portfolio calculated using $400d$ sample size, and two optimization algorithms: CMA-ES-CSA and IPOP400D.

9 Adaptive Landscape Analysis

This chapter is based on paper [JD19], which appeared in the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Companion 2019, ***Adaptive Landscape Analysis****.*

9.1 Preliminaries

Summary of Results. This chapter presents a preliminary study of *adaptive landscape analysis*, which is concerned by how problem features change locally depending on the quality of the already reached solution(s), in hope to understand if there is an underlying connection between how the fitness landscape looks locally and the performance of a certain optimization algorithm. We "zoom in" into the local feature structure, and focus in particular on gauging what features values tell us about the nature of the problem, and subsequently what information we can extract about how to adapt the algorithm choice during the optimization process, following the dynamic change in the fitness landscape. These are the first steps towards extension of the so-called per-instance algorithm configuration (PIAC) approach in numerical optimization (successfully applied to the configuration of the well-known CMA-ES algorithm in [BDS+17]) to online landscape-aware algorithm selection. Apart from parameter control, we can also target automated *algorithm design* with the same approach, such as for the modular CMA-ES presented in Section 3.1.2.

Experimental Setup

Problem Benchmark, Choice of the Algorithm, Feature Computation. Again, we consider as the algorithm of choice the standard CMA-ES, presented in Section 3.1.1. For our analysis, we consider the BBOB noiseless test suite (cf. Section 3.3.1. From the 24 BBOB functions, we select three (namely F1, F2 and F6), all in dimension 5, and we focus on the first five instances for each of them.

The experimental code was built upon the original CMA-ES implementation provided in [HAB19]. The feature computation was done using the *flacco* library [KT16].

Structure of the Experiment. The experiment is designed in a following way: while the algorithm is running, we track the precision of the sampled points. Whenever a new target value $10^i, i = \{-2, ..., 9\}$ is reached, 2000 additional points are sampled

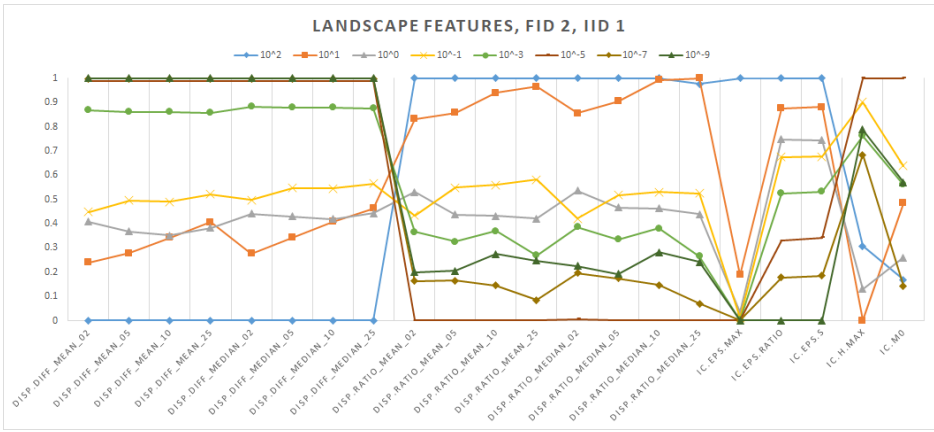


Figure 9.1: Normalized feature values for the Dispersion and Information Content feature sets for the first instance of the BBOB function F2 (ellipsoid function).

from the current distribution and stored along with their evaluations. These additional samples do not influence the behavior of the algorithm and are used only to compute the feature values of the fitness landscape currently seen by the CMA-ES. We run each optimization process five times, with no restarts and with fixed default population size for the CMA-ES.

Landscape feature values are then computed at each target precision level and for each function and instance. To this end, we again only consider features that do not require additional function evaluations for feature computation, introduced in Chapter 4, having at our disposal 56 features in total. For a comparison of these *local* features with the ones typically computed using points sampled globally from the $[-5, 5]$ range (i.e., the domain of definition of the BBOB functions), we have also computed the latter, *global* features for the respective BBOB functions, using the same number of 2000 samples for each of the first five instances in dimension 5. For the purpose of this chapter, we consider average feature values computed over five independent runs, and from hereon we focus exclusively on the first instance of each function.

9.2 "Zooming In" into the Landscapes

In this section we present some of our preliminary findings for the adaptive landscape analysis, obtained through the lens of three selected BBOB functions. The accompanying figures illustrate how the feature values evolve during the optimization process, with different features on x-axis, feature values on y-axis and target values as different lines within a figure. The scale has been normalized to the range $[0, 1]$, for a better

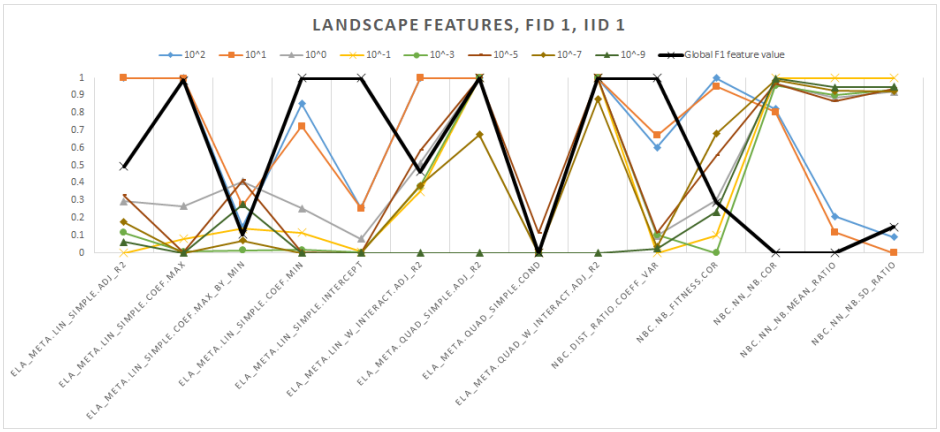


Figure 9.2: Normalized feature values for the Meta-model and Nearest-Better Clustering feature sets for the first instance of the BBOB function F1 (sphere function).

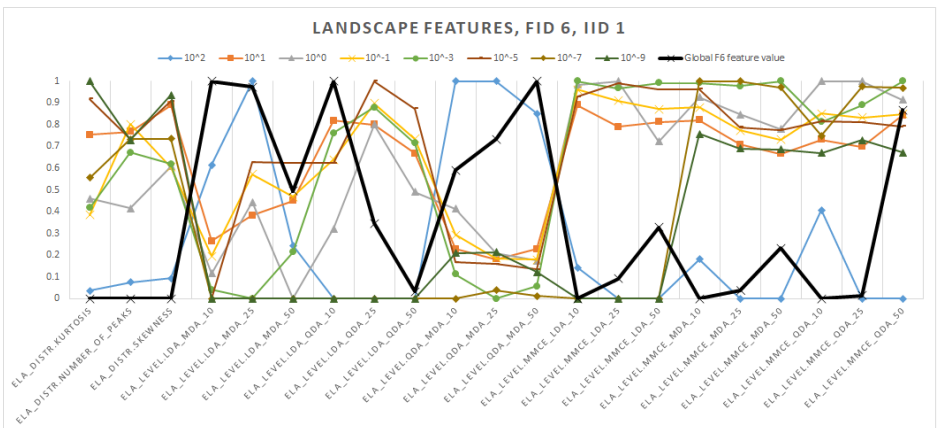


Figure 9.3: Normalized feature values for the y-Distribution and Levelset feature sets for the first instance of the BBOB function F6 (attractive sector function).

visualization of the results. Lastly, in order to visualize the data more clearly and make the charts more accessible for understanding, we have purposefully omitted some target values (namely 10^{-2}, 10^{-4}, 10^{-6} and 10^{-8}) from all the figures.

Figure 9.1 shows Dispersion and Information Content feature sets for the function F2 (ellipsoid function). We observe monotonicity in the relationship between feature values and target values for certain features, which is most prominent in the case of features *IC:eps.ratio* and *IC:eps.s* from the Information Content feature set. Information Content feature set is closely related to the measure of ruggedness of the fitness landscape, and the monotonicity could be explained by the fact that F2 is a locally smooth function. The Dispersion feature set exhibits an overall similar monotonic behavior, although not as consistent at every feature; this set translates the notion of hardness of the problem and quantifies the proximity of more interesting regions of the search space. We remark that these 2 feature sets behave very similarly in the other 2 functions, albeit not shown here. This is in line with the previous comment about the monotonicity of Information Content features, as F1 and F6 are both smooth functions as well. It is worth noting that the curve of the global feature values is excluded from Figure 9.1 for reasons of scale. However, the general trend seems to be that global feature values usually differ significantly from local feature values, indicating that the fitness landscape as seen by the algorithm differs from that of uniform sampling.

On the other hand, both Figure 9.2 and Figure 9.3 display plots of their respective global feature values along with locally observed values, which for the most part show stark contrast between the two, as previously mentioned. Moreover, both of the figures demonstrate rather chaotic and inconsistent local behavior for most features. In Figure 9.2 we observe Meta-model and Nearest-Better Clustering feature sets for the function F1 (sphere function). Meta-model feature set aims to measure the ability to approximate the objective function with a linear, quadratic or regression model, while Nearest-Better Clustering feature set deals with recognizing single peaks within a multimodal landscape. Lastly, Figure 9.3 shows y-Distribution and Levelset feature sets for the function F6 (attractive sector function): the former contains features that measure ruggedness, symmetry and multimodality of the problem at hand, while the latter is especially useful when dealing with multimodal functions.

At this stage, the available data does not yet allow for an more complete intuitive interpretation of the dynamic change of feature values. Some of the important questions that guide our ongoing research activities are why these values evolve in such a fashion, whether these features capture the important knowledge about the problem instance and if yes, how to exploit that knowledge for the purpose of recognizing different problems.

9.3 Conclusions

Motivated by the quest to design landscape-aware online algorithm selection and configuration techniques, we have analyzed in this work to what extent the fitness landscape, as seen by iterative black-box optimization heuristics, changes during the optimization process. To this end, we have computed ELA features as locally seen by the CMA-ES at different target values 10^{-i}, $i = -2, ..., 9$. Our preliminary analysis focuses on 3 selected benchmark problems of the BBOB testbed. In an ongoing work we are extending our approach to the full set of the 24 noiseless BBOB functions.

Our next step towards an online algorithm selection model will be coupling feature information to performance of continuous black-box optimizers. Apart from studying the algorithm selection problem on standard solvers such as CMA-ES, Differential Evolution, EDAs etc., it would be of great importance to build an online configurator for the modular CMA-ES proposed in [RWL+16]. A first indication that a dynamic configurator of this meta-model is likely to give additional performance gains has been demonstrated in [VRB+19].

10 Trajectory-Based Performance Regression

This chapter is based on paper [JED21], which appeared in the Proceedings of the Applications of Evolutionary Computation Conference (EvoAPPs) 2021, held as part of EvoStar 2021, ***Towards Feature-Based Performance Regression Using Trajectory Data****.*

10.1 Preliminaries

Summary of Results. This chapter introduces a novel *trajectory-based* landscape-aware algorithm selection strategy. With the long-term goal to obtain well-performing dynamic ELA-based algorithm selection and configuration techniques, we analyze in this chapter a first, rather cautious task: ELA-based performance prediction using the trajectory samples of the algorithm under investigation. More precisely, we consider the Covariance Matrix Adaptation Evolution Strategy (CMA-ES [HO01]), and we aim at predicting its solution quality (measured as target precision, i.e., the distance to an optimal solution in quality space) after a fixed budget of function evaluations using the landscape features extracted from the samples of the first part of the search trajectory. Concretely, we use the first 250 samples evaluated by the CMA-ES and we aim at predicting its performance after additional 250 evaluations, doing so for 20 independent CMA-ES runs.

We then take into account that problem characteristics cannot only be described via classic ELA features, but that internal states of the search heuristics can also be used to derive information about the problem instance at hand. Such approaches have in the past been used, for example, for local surrogate modeling [PRH19]. We analyze the accuracy gains when using the same state information as in [PRH19], that is, the values of the CMA-ES internal variables that mainly carry information about the current probability distribution from which the CMA-ES samples candidates for the new generation. In our experiments, the advantage of using this state information over using only the ELA features, however, is only marginal. Concretely, the average difference between true and predicted solution quality decreases from 14.4 to 12.1 when adding the state variables as features (where the average error reported here is taken over all 24 BBOB noiseless benchmark problems, and over all performed CMA-ES runs).

We observe in the experiments above that some CMA-ES runs are drastic outliers in terms of performance, at times with the target precision differing from the target precision of all the other runs by up to 10 orders of magnitude. We therefore also consider an intentionally more "friendly" setting, in which we analyze the regression

quality only for the run achieving median performance on a given problem instance. Conclusions for combining trajectory-based and state variable features remain almost identical to those stated above.

We then compare these median trajectory-based predictions to the classical approach using globally sampled features. Here, we pessimistically assume that the samples were computed for free. That is, we couple 2 separate sets of the global feature values approximated from 250 and 2000 uniformly sampled points each to the target precision achieved by the CMA-ES after 500 function evaluations. Interestingly, the difference in prediction accuracy compared to our trajectory-based predictions is rather small. The global predictions still remain, however, more accurate, with an average absolute prediction error of 4.7 vs. 6.2 for the trajectory approach (where again the average is taken over all 24 BBOB functions).

Furthermore, we also use this median setting to analyze the influence of feature selection on prediction accuracy. Different state-of-the-art methods were applied, using a transfer learning scenario, to select features estimated to be the most important and to have highest discriminative power. Here again, the differences in prediction accuracy were small, with feature selection surprisingly leading to an overall slightly worsened solution quality than the full feature portfolio.

As suggested in Chapter 6, all our experiments are based on two independently trained models: one which aims to predict target precision after 500 evaluations, and one which predicts the logarithm of this target precision. While the former is better in guessing the broader "ball park", the latter is more suitable for fine-grained performance prediction, i.e., when the expected performance of the algorithm is very good. As in Chapter 6, we also build a combined regression, which uses either one of the two models, depending on whether the predicted performance is better or worse than a certain threshold. The optimal thresholds differ quite drastically between different feature sets. However, a sensitivity analysis reveals that their influence on overall performance is rather small. Also, the ranking of the different feature portfolios remains almost unaffected by the choice of the threshold. In line with the results in Chapter 6, the combined models perform consistently better than any of the two standalone ones, albeit slightly.

Experimental Setup

Regression Models. In this chapter, we used once again an off-the-shelf random forest regressor from the Python *scikit-learn* package [PVG+11], without parameter tuning and using 1000 estimators.

Algorithm of Choice. We restrict this chapter to a single heuristic, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES, presented in Section 3.1.1). For the purpose of our experiments, we used its standard version, available in the Python

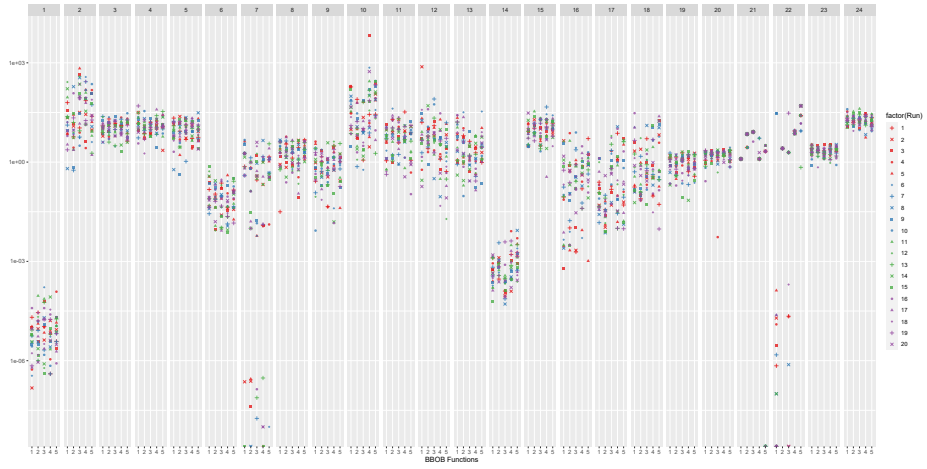


Figure 10.1: Target precision achieved by the CMA-ES with a budget of 500 function evaluations, for each of the first five instances of all 24 BBOB functions. Differently colored and shaped points represent 20 independent CMA-ES runs.

pycma package [HAB19], which uses a fixed population size and no restarts during the optimization process.

Benchmark Problems. As our benchmark, we use the first five instances of all 24 noiseless BBOB functions. We point out that transformations carried out while generating different instances of each function do not affect the performance of CMA-ES, but they do influence some of the feature values, especially those which are not transformation-invariant [ŠEK20] (for the invariant features, the boundary handling can have an effect on the feature values). We focus exclusively on dimension 5 here.

Structure of the Experiment. For our first experiments, we perform 20 independent runs of the CMA-ES on these 120 problem instances, while keeping track of the search trajectories and the internal state variables of the algorithm itself. Throughout this chapter, we fix the budget of 500 function evaluations, after which we stop the optimization and record the target precision of the best found solution within the budget. In order to predict those recorded target precisions after 500 function evaluations, we compute the trajectory-based landscape features using the first 250 sampled points and their evaluations from the beginning of each trajectory, and couple them with the values of the internal CMA-ES state variables extracted at the 250^{th} function evaluation.

Figure 10.1 summarizes the target precision achieved by CMA-ES in each of the 20 runs. We see that the results are more or less homogeneous across different runs

and across different instances of the same problem. However, we also observe several outliers, e.g., for functions 7 (outlier for all instances), function 10 (instance 4), function 12 (instance 1). It is important to keep in mind that the randomness of these performances are entirely caused by the randomness of the algorithm itself – the problem instance does not change between different runs.

Problem Features: Landscape and Algorithm State. We compute the ELA features using *flacco* (presented in Section 4.3. We again restrict ourselves to those feature sets that do not require additional function evaluations for computing the features. Namely, in this chapter we use 2 original ELA feature sets (*y-Distribution* and *Meta-Model*), as well as *Dispersion*, *Nearest-Better Clustering* and *Information Content* feature sets. This gives us a total of 38 landscape features per problem instance.

In addition, we follow up on an idea previously used in [PRH19] and consider a set of internal CMA-ES state variables as features:

- *Step-size*: its value indicates how large is the approximated region from which the CMA-ES samples new candidate solutions.
- *Mahalanobis mean distance*: it represents the measure of suitability of the current sampled population for model training from the point of view of the current state of the CMA-ES algorithm.
- *C evolution path length*: it indicates the similarity of landscapes among previous generations.
- σ *evolution path ratio*: it provides information about the changes in the probability distribution used to sample new candidates.
- *CMA similarity likelihood*: it is a log-likelihood of the set of candidate solutions with respect to the CMA-ES distribution and may also represent a measure of the set suitability for training.

As suggested in Chapter 6, and using the elements described above, we establish two separate regression approaches. One model is trained to predict the actual, true value of the target precision data (we refer to it as the *unscaled model* in the remainder of the chapter), while the other predicts the logarithm of the target precision data (the *logarithmic model*). Moreover, to reduce variability, we estimate both models' prediction accuracy through performing a 5-fold *leave-one-instance-out* cross-validation, making sure to train on four out of 5 instances per BBOB function, test on the remaining instance and combine the results over the rounds.

	Unscaled model			Log model		
FID	SV	ELA	ELA+SV	SV	ELA	ELA+SV
1	4.6	2.6	1.9	0.1	0.0	0.0
2	60.4	64.2	63.7	65.4	58.0	57.1
3	6.4	5.4	5.2	10.1	7.2	6.9
4	9.0	7.0	6.9	12.4	8.7	8.6
5	86.4	6.6	26.2	5.5	4.6	4.5
6	9.1	9.4	4.0	1.1	0.5	0.6
7	6.5	1.0	0.8	0.9	0.7	0.7
8	6.8	1.8	1.7	1.7	1.2	1.2
9	9.2	1.1	1.0	0.9	0.7	0.7
10	126.2	171.1	125.1	125.9	119.7	114.8
11	37.2	12.7	23.9	6.8	5.4	5.6
12	15.2	101.8	24.5	15.3	14.3	13.0
13	9.8	5.6	5.1	4.1	3.9	3.7
14	4.8	2.5	1.1	0.4	0.1	0.0
15	8.8	10.7	5.5	9.6	6.9	6.5
16	4.7	4.8	5.0	0.8	1.7	1.7
17	6.4	1.9	1.8	1.4	0.7	0.7
18	7.8	2.4	2.3	2.7	2.3	2.3
19	11.7	2.3	1.6	2.9	0.5	0.5
20	4.6	0.5	0.6	1.4	0.6	0.6
21	3.8	3.5	3.7	3.5	3.6	3.6
22	10.1	9.0	8.6	9.1	8.6	8.6
23	13.4	1.8	2.0	2.6	0.8	0.8
24	9.4	5.2	5.1	14.9	8.3	8.5

Figure 10.2: Absolute prediction errors for both regression models aggregated per BBOB function in 3 different scenarios depending on the feature set used. The *SV* column stands for the CMA-ES state variables, the *ELA* for the landscape features, and the third one is the combination of both.

10.2 Supervised Machine Learning for Performance Regression

Adopting our two regression models, we trained them separately in the following three scenarios: using as predictor variables the landscape features only, using the internal CMA-ES state variables only, and using the combination of the two. We trained the random forests 3 independent times and took a median of the 3 runs to ensure the robustness of the results.

Figure 10.2 highlights the absolute prediction errors per BBOB function using two regression models, the unscaled and the logarithmic one, when trained with 3 different feature sets: using only the trajectory landscape data, only the CMA-ES state variable data, and the combination of the two. For the majority of the functions, using the combination of the trajectory data and the state variable data seems to help in improving the performance prediction accuracy, compared to the scenarios which use only one of those two feature sets.

We also confirm that the logarithmic model is indeed better at predicted fine-grained target precision (e.g., in the case of F1 (sphere function) or F6 (linear slope function),

we know that those functions do not require many function evaluations to converge to the global optimum, and their recorded target precision values are already quite small as they are very near the optimal solution). On the other hand, the unscaled model performs better where the target precision values are higher (e.g., for the functions such as F3, F15 (two versions of Rastrigin function), and also F24 (Lunacek bi-Rastigin), which are all highly multimodal, the number of function evaluations in our budget was not nearly enough to allow for finding a true optimum).

We also notice that using only the state variables for the unscaled model does not suffice for an accurate prediction in the most cases. The reverse situation is nevertheless also possible: we see that for F12, using only the state variables yields the best accuracy in the unscaled model. Furthermore, there are also exceptions where using only the landscape data results in a higher accuracy than using the combined features (e.g., F11 for both models, F5 for the unscaled model).

10.3 Comparison with Global Feature Values

We then proceeded to compare the differences in the prediction accuracy from the sets described in the Section 10.2 with the prediction accuracy using the global feature data, both alone and combined with the same CMA-ES state variable data as above. To be able to perform a fair comparison, for the trajectory data we selected from the 20 executed CMA-ES runs those runs with the median target precision value per problem instance and their corresponding features and re-trained the unscaled and the logarithmic model. Global features-wise, both models were also trained using features computed from 2000 and 250 globally uniformly sampled points (the median value of 50 independent feature computations) for each function and instance.

Figure 10.3 shows the absolute errors in prediction when the trajectory-based approach is compared with the results using the global features. The highest accuracy is reported in cases when only the global landscape features were used, across almost all problems, with 2000-sample features yielding the best results. Here, we do not observe a huge improvement when combining the global landscape features with the state variable data. It seems that the number of samples used to compute the features can be crucial in reducing the errors in prediction, as global sampling could be linked to a potential higher discriminative power of features thus computed. Again, for certain functions such as F2 and F10 (both of which are different variants of the ellipsoidal function), we observe an overall low accuracy.

10.4 Sensitivity Analyses

Feature Selection. To provide a sensitivity analysis based on the features used for the performance regression, we performed feature selection in the scenario of transfer

	Unscaled model							Log model						
FID	SV	ELA	ELA +SV	GLOB2k	GLOB2k +SV	GLOB250	GLOB250 +SV	SV	ELA	ELA +SV	GLOB2k	GLOB2k +SV	GLOB250	GLOB250 +SV
1	4.4	2.2	2.0	0.9	1.3	1.8	2.0	0.29	0.00	0.00	0.00	0.00	0.00	0.00
2	31.2	17.4	16.7	18.5	17.9	18.9	17.7	38.89	31.56	31.86	25.65	26.58	28.98	30.20
3	8.3	5.2	5.5	1.6	2.6	3.6	4.9	10.14	7.08	6.97	4.93	5.69	8.33	8.80
4	9.3	3.5	3.8	2.9	3.7	3.9	4.6	11.78	4.45	4.99	3.59	4.87	7.64	8.31
5	1.7	1.4	1.0	1.3	1.1	1.2	0.9	7.16	8.19	8.17	4.97	5.22	5.93	6.21
6	5.0	6.7	6.1	1.4	1.6	2.5	2.5	0.66	1.45	1.33	0.23	0.25	0.30	0.34
7	3.8	4.9	4.7	13.1	12.9	4.0	3.2	0.68	0.31	0.33	2.42	2.05	1.00	1.01
8	3.2	1.6	1.9	1.9	2.2	1.8	2.0	0.98	0.98	0.96	0.81	0.76	0.91	0.71
9	3.2	1.1	0.7	0.5	0.4	1.0	0.8	0.73	0.26	0.27	0.45	0.38	0.40	0.41
10	37.0	24.4	24.3	19.5	19.3	24.8	26.9	37.98	30.84	31.48	29.15	29.94	28.75	30.17
11	14.4	8.6	10.2	2.6	3.0	6.4	8.4	2.18	5.18	5.36	3.09	3.52	2.57	3.05
12	2.2	24.9	20.2	2.3	2.5	4.4	3.7	4.55	1.76	1.40	3.43	3.57	3.65	3.55
13	9.7	2.2	3.6	2.4	2.4	3.2	4.8	1.95	1.62	1.70	1.73	1.81	1.49	1.65
14	2.8	0.7	0.6	3.5	3.5	5.5	5.4	0.45	0.00	0.00	0.05	0.06	0.23	0.25
15	5.5	3.6	3.5	4.3	5.5	5.6	5.8	7.23	7.48	6.23	6.04	6.57	8.23	8.40
16	2.8	5.6	5.5	0.6	1.1	0.5	1.1	0.35	4.26	4.08	0.27	0.33	0.21	0.23
17	3.3	1.3	1.2	2.4	2.2	3.3	2.8	0.89	0.26	0.25	0.27	0.30	0.27	0.27
18	3.3	0.9	1.0	3.0	2.7	3.6	3.0	0.44	0.34	0.27	0.29	0.25	0.39	0.41
19	18.1	2.7	3.5	1.4	2.3	1.4	3.2	2.91	0.24	0.27	0.43	0.53	0.40	0.43
20	3.3	5.8	5.7	0.2	1.1	0.6	1.9	1.27	0.67	0.80	0.67	0.93	0.64	0.75
21	3.1	1.9	2.6	4.1	3.6	2.9	3.0	3.27	3.16	3.15	3.96	3.98	4.11	3.96
22	7.2	7.0	7.0	9.1	9.2	9.6	9.3	7.87	7.45	7.45	7.66	7.85	7.61	7.66
23	6.9	3.0	3.6	0.2	0.9	0.5	2.5	2.53	1.11	1.19	0.61	0.63	0.65	0.68
24	6.7	4.0	4.2	1.5	2.4	2.2	3.3	12.65	8.35	9.22	4.41	7.24	8.07	9.87

Figure 10.3: Absolute prediction errors for both regression models for the median trajectory-based prediction (the first 3 columns of each block) and the median global feature prediction (the middle two columns of each block represent the errors when using the 2000-sample features, and the last two columns correspond to using the 250-sample features).

learning, i.e., between different supervised tasks, where the features selected for the problem classification task have been evaluated on the performance regression task.

To do this, we have explored four state-of-the-art feature selection techniques: *Boruta* [KJR10] is a feature selection and ranking algorithm based on random forests algorithm, which only selects features that are statistically significant. *Recursive feature elimination (rfe)* [GFB+06] learns a model assessing different sets of features by recursively eliminating features per loop until a good model is learnt. It requires an machine learning algorithm for evaluation, and here we use a random forest. *Stepwise forward and backward selection (swfb)* [DK92] tries to fit the best regression model by iteratively selecting and removing features. In our experiments, we used it in both directions simultaneously. *Correlation analysis with different threshold values (cor)* [BCH+09] is based on the correlation analysis done only using the features (i.e., excluding the target). The result is a feature set where highly correlated features are omitted. In our case, we tested three different correlation thresholds: 0.50, 0.75, and 0.90. Note that while the

	boruta	swfb	rfe	cor0.5	cor0.75	cor0.9
# selected ELA features	37	1	7	4	9	15
# selected state variable features	2	0	0	3	3	5

Table 10.1: Number of ELA and state variable features for each selected feature portfolio. Details are available in Table 10.3.

first three feature selection methods require a supervised machine learning task, the last one is completely unsupervised and does not depend on the target.

Our experimental design has been done using stratified 5-fold cross-validation. For a fair feature selection, we used the aforementioned methods on each training fold separately, then selected the intersection of the features returned by each training fold in the end. These features are further evaluated in the performance regression task.

Table 10.1 summarizes how many features were selected per portfolio, from the whole set of 38 ELA landscape features and 5 CMA-ES state variable features.

Combined Selector Model and Sensitivity Analysis. We measure the regression accuracy in terms of *Root Mean Squared Error* (RMSE). Table 10.2 summarizes the RMSE values for the different feature portfolios when using (1) the unscaled model, (2) the logarithmic model, and (3) a combination of the two models (see last three rows of Table 10.2). The threshold τ at which the predictive model changes is optimized for each feature portfolio individually, the obtained thresholds are summarized at the top of Table 10.2. That is, we select the prediction of the logarithmic model when the predicted precision (according to the logarithmic model) is smaller than the threshold value τ, and we use the prediction of the unscaled model otherwise. Note that the optimal threshold value τ varies significantly between the different feature portfolios.

When comparing all the different portfolios (initial trajectory-based, global and selected trajectory-based ones), the good performance of the global feature sets is not surprising. Differences from the initial trajectory-based predictions are marginal for sets such as *boruta*, *cor0.75* and *cor0.9*, whereas *swbf* and *cor0.5* perform constantly worse than *ELA+SV*. Using the *rfe* set, on the other hand, led to better results than using the original feature set. *SV* alone does not achieve good accuracy, but its contribution to ELA-only feature portfolio is around 3% at the best threshold for the combined model, which is $\tau = 4.901$. The absolute errors per instance are plotted in Figure 10.4.

10.5 Conclusions

We analyzed in this final chapter the accuracy of predicting the CMA-ES solution quality after given budget based on the features computed from the samples on the

			SV	ELA	ELA +SV	GLOB2k	GLOB2k +SV	GLOB250	GLOB250 +SV	boruta	cor 0.5	cor 0.75	cor 0.9	rfe	swfb
								Best threshold τ							
FID	min_tp	max_tp	1.336	3.99	4.742	14.497	9.46	0.694	2.605	3.63	1.813	4.901	1.717	7.388	20
1	0	0	0.43	0	0	0	0	0	0	0	0	0	0	0	0.21
2	9.25	87.47	44.69	25.47	24.46	24.49	23.65	28.96	27.98	25.16	36.78	35.28	34.62	24.51	53.85
3	10.34	14.63	9.73	6.62	7.02	5.62	6.25	4.43	9.39	6.73	8.08	7.7	5.63	7.95	8.82
4	10.29	14.53	11.86	5.08	5.08	4.81	5.73	5.1	7.49	5.18	9.26	6.71	5.59	7.53	8.41
5	8.53	11.34	4.59	8.24	8.15	6.02	5.52	1.48	1.06	8.21	2.36	6.76	4.91	4.52	7.74
6	0.04	0.11	0.78	3.87	2.15	0.24	0.27	0.33	0.37	4.46	1.02	1.19	1.65	0.89	1.43
7	0.13	1.83	3.84	0.36	0.39	4.49	4.07	4.62	1.13	0.42	0.86	0.6	0.65	0.67	7.94
8	1.22	2.59	3.02	1.06	1.1	1.25	1.22	2.03	0.84	0.95	2.02	0.93	1.04	1.42	4.88
9	0.68	1.36	3.05	0.34	0.34	0.59	0.53	0.77	0.48	0.33	0.52	0.26	0.29	0.86	3.47
10	8.44	84.69	43.33	32.06	32.85	21.39	23.16	29.29	33.35	32.13	42.18	32.73	35.01	29.64	47.68
11	4.97	8.83	17.48	10.37	12.63	3.46	2.99	8.26	10.18	11.17	25.11	21.61	20.86	2.43	4.77
12	2.53	6.32	4.76	19.34	15.89	4.03	4.16	5.76	4.13	19.45	4.38	2.31	9.69	19.32	3.56
13	1.09	5.92	13.62	2.07	2.13	2.29	2.32	3.51	2.21	2.11	2.45	2.04	5.37	2.03	3.5
14	0	0	1.65	0.01	0.01	0.07	0.08	1.89	0.37	0.01	0.61	0	0	0	5.18
15	8.49	12.93	7.44	6.79	6.66	6.08	6.81	6.24	8.12	5.77	7.07	6.11	5.88	6.73	9.12
16	0.18	0.83	0.87	6.31	5.85	0.32	0.35	0.25	0.25	6.45	0.48	4.42	4.77	2.43	1.21
17	0.03	0.64	2.63	0.36	0.33	0.32	0.33	0.28	0.31	0.38	4.25	0.41	0.38	0.27	2.75
18	0.16	0.69	1.54	0.34	0.34	0.33	0.3	4.96	0.45	0.34	0.54	0.28	0.3	0.54	1.24
19	0.75	1.19	18.75	0.34	0.36	0.65	0.67	1.66	0.49	0.38	9.95	0.59	0.5	0.46	8.54
20	1.7	1.79	3.77	0.76	0.84	0.99	1.17	0.99	0.93	0.69	3.2	1.22	3.16	1.77	5.16
21	0	8.12	4.86	4.55	4.52	5.02	5.02	5.2	4.95	4.53	4.72	4.84	4.83	4.83	3.72
22	0	25.48	12.04	11.91	11.91	11.9	11.91	12.88	11.94	11.91	11.89	11.93	11.7	11.92	5.34
23	1.99	2.35	7.49	3.67	3.61	0.69	0.77	0.64	0.91	3.72	7.89	2.73	4.53	2.46	3.04
24	15.73	20.73	10.06	4.76	4.94	5.15	6.52	2.95	4.29	4.81	11.13	9.49	7.22	7.43	11.65
Overall RMSE, combined			**15.05**	**10.41**	**10.25**	**7.74**	**7.92**	**9.48**	**10.05**	**10.43**	**13.66**	**11.67**	**11.86**	**9.77**	**15.73**
Overall RMSE, unscaled			15.08	11.18	10.88	9.21	9.30	9.58	10.19	11.16	14.11	11.80	12.00	10.93	17.03
Overall RMSE, log			15.63	13.05	13.21	11.46	11.88	12.05	12.87	13.14	14.65	13.89	14.29	12.61	15.73

Table 10.2: RMSE values of the combined selector in three scenarios: when the prediction is based on the search trajectory landscape features and state variables (first 3 columns), on global features (next 4 columns), and finally on selected feature portfolios (last 6 columns).

CMA-ES search trajectory using two complementary regression models, the unscaled and the logarithmic model. Adding information obtained from the CMA-ES internal state variables does not improve the prediction accuracy drastically compared to the trajectory-based data only. Those results were then contrasted to the regression using the global features, where using the latter ones, especially those computed using a higher number of samples, yielded a consistently better accuracy.

Next, we tested whether we would achieve further gains in accuracy through feature selection. Although the overall results are comparable to the ones from initial trajectory-based portfolios, several selected feature sets resulted in worse accuracy than in the initial approach. We ultimately pointed out the advantages of using our combined selector model over relying separately on predictions of the standalone unscaled or logarithmic model across all different feature portfolios in all 3 scenarios.

In terms of **future work**, we plan on continuing this research by considering the following questions and tasks:
(0) Performance prediction of **other solvers**: How accurately can we use trajectory-based features of one algorithm to predict the performance of another algorithm? In this work, we have only tried to predict performance for the same algorithm from whose trajectory the feature values have been computed. A next step would be to test if models

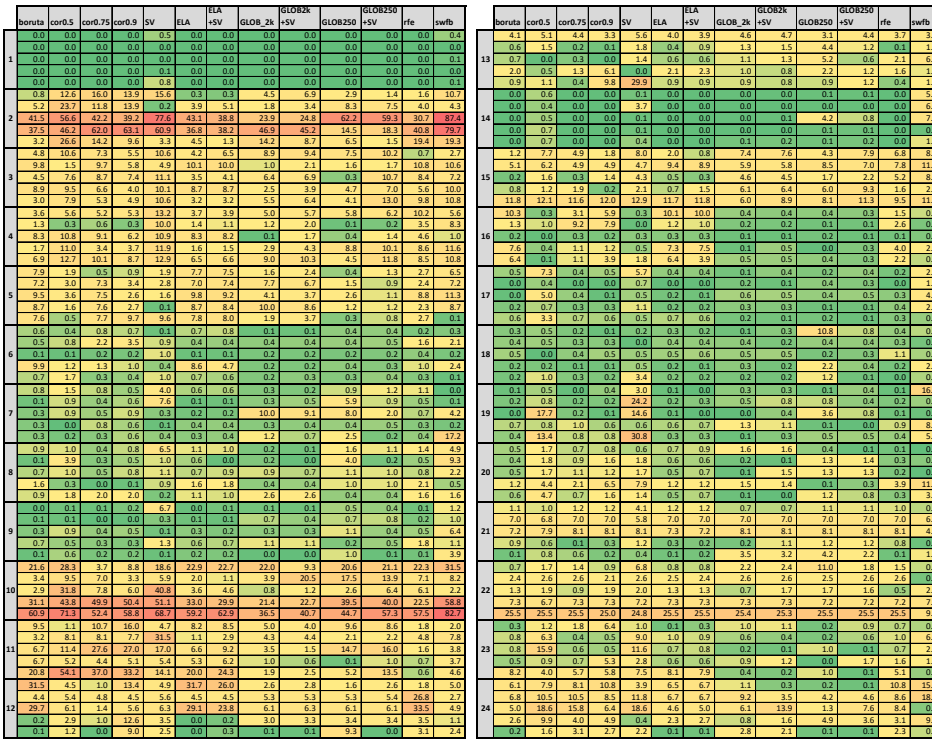


Figure 10.4: Absolute prediction errors of the combined models using portfolio-specific optimal thresholds τ.

for configuring the same algorithm can be trained. When this is successful, transfer learning from one algorithm to another one can be considered.

(1) How can we more efficiently capture the **temporal component**, i.e., the information which sample was evaluated *when* during the search? Using such longitudinal data, both in terms of extracted feature values and in terms of state variable evolution could possibly be done using recurrent neural networks [ZFW+19].

(2) **Combining global and trajectory-based sampling:** In our work, we only considered the case in which *either* global sampling *or* trajectory-based sampling is used. The accuracy of the models based on global sampling was better than that of the trajectory-based features. Even if we keep in mind that this comparison was unfair in that we provided the global feature values "for free", the results nevertheless suggest that a combination of global and trajectory-based feature computations could be worthwhile to investigate. How we can optimally balance the budget between global sampling,

trajectory-based sampling, and remaining optimization budget is a challenging question in this context.
(3) **Warm-starting** the CMA-ES such that it starts the optimization process with the covariance matrix and other parameters that are extrapolated from the (uniformly or otherwise) distributed global samples might significantly improve the overall accuracy, as the CMA-ES will have a better overview of the whole problem instance "from the get-go". A similar approach has been suggested in [MRT15] when switching from a Bayesian optimization algorithm to CMA-ES.
(4) **Feature selection and ranking:** Instead of using transfer learning for feature selection between two different supervised machine learning tasks, feature selection within the same supervised task has not been considered in this chapter. We also plan on making better use of variable importance estimations provided by feature ranking algorithms such as those based on ensemble of predictive clustering trees [PKD20] and those based on ReliefF and RReliefF [RK03].
(5) **Feature design:** The work [DLV+19] suggests several algorithm-specific features for the SOO tree algorithm [Mun11]. Such specific features can much more explicitly capture the characteristics of the algorithm-problem instance interaction. It could be worthwhile to study whether, possibly in addition to the longitudinal data mentioned in (1), such specific features can be identified for other common solvers, such as the CMA-ES.
(6) **Feature portfolio:** We note that our work above is based on the features available in the *flacco* package (cf. Section 4.3). Since the design of *flacco*, however, several new feature sets have been suggested. Another straightforward way to extend our analyses would be in the inclusion of these feature sets, with the hope to improve the overall regression accuracy. In this respect, we find in particular the Search Trajectory Networks suggested in [OMB20] worth investigating.
(7) **Representation learning of landscapes:** The feature data will be additionally explored by applying representation learning methods that automatically learn new data representations by reducing the dimension of the data, automatically detecting correlations, and removing bias and redundancies presented in the feature data. The work presented in [EPR+20] showed that linear matrix factorization representations of the ELA features values significantly detects better correlation between different problem instances.
(8) **Hyperparameter tuning of regression models:** Last, but not least, we are planning to explore algorithm portfolio that consists of different regression methods in order to find the most suitable one, together with finding its best hyperparameters for achieving better performance. In this study, we have used random forest for regression without tuning its parameters, since we have been interested in the contribution of different feature portfolios.

Table 10.3: Feature portfolios that were selected per each of the selection methods used; the number of landscape and state variable features used per selected set, and their frequency of appearance in each selected set.

Feature	# sets	boruta	swfb	rfe	cor0.5	cor0.75	cor0.9
ela_distr.skewness	4	x	x	x			x
ela_distr.kurtosis	4	x		x	x	x	
ela_distr.number_of_peaks	4	x			x	x	x
ela_meta.lin_simple.adj_r2	2	x					x
ela_meta.lin_simple.intercept	1	x					
ela_meta.lin_simple.coef.min	2	x					x
ela_meta.lin_simple.coef.max	2	x		x			
ela_meta.lin_simple.coef.max_by_min	3				x	x	x
ela_meta.lin_w_interact.adj_r2	1	x					
ela_meta.quad_simple.adj_r2	2	x		x			
ela_meta.quad_simple.cond	4	x			x	x	x
ela_meta.quad_w_interact.adj_r2	3	x		x			x
disp.ratio_mean_02	1	x					
disp.ratio_mean_05	1	x					
disp.ratio_mean_10	1	x					
disp.ratio_mean_25	1	x					
disp.ratio_median_02	2	x					x
disp.ratio_median_05	1	x					
disp.ratio_median_10	1	x					
disp.ratio_median_25	1	x					
disp.diff_mean_02	1	x					
disp.diff_mean_05	1	x					
disp.diff_mean_10	1	x					
disp.diff_mean_25	1	x					
disp.diff_median_02	3	x				x	x
disp.diff_median_05	1	x					
disp.diff_median_10	1	x					
disp.diff_median_25	1	x					
nbc.nn_nb.sd_ratio	2	x					x
nbc.nn_nb.mean_ratio	1	x					
nbc.nn_nb.cor	2	x					x
nbc.dist_ratio.coeff_var	3	x				x	x
nbc.nb_fitness.cor	2	x		x			
ic.h.max	3	x				x	x
ic.eps.s	1	x					
ic.eps.max	1	x					
ic.eps.ratio	4	x		x		x	x
ic.m0	3	x				x	x
step_size	4	x			x	x	x
mahalanobis_dist	2				x		x
c_evol_path	3				x	x	x
sigma_evol_path	2					x	x
cma_simil_lh	2	x					x

11 General Conclusions and Outlook

In this thesis, we have proposed a novel approach within the context of landscape-aware algorithm selection. The key objective of our *online, trajectory-based landscape-aware algorithm selection* approach is to balance out the cost of the feature extraction step by incorporating it into the optimization process, with the goal to save precious computational resources that can then be allocated to the optimization routine instead. We investigated this approach in the fixed-budget setting. Since this setting has been largely neglected in the context of landscape-aware algorithm selection prior to our work, we also present various preparatory steps to set up this fixed-budget trajectory-based framework. Notably, we point out the potential of suitably combining two complementary regression strategies in the fixed-budget context. We complete the algorithm selection pipeline by building our selector on top of those strategies.

We hope that our work inspires more research in related promising avenues. In particular, we believe that the following topics merit further attention.

Predicting Multiple Diverse Solvers Our approach in this thesis represents a stripped-down use-case that serves as a form of a proof-of-concept for the trajectory-based algorithm selection, with only one switch from a solver to another. Recent efforts in the fixed-target context also point out the potential of switching the algorithm once [VRB+19]. A more detailed analysis of prediction of another solver's performance based on the features computed using the base solver's samples is thus very strongly required.

In this context, the problem of *warm-starting* the next solver(s) is paramount, i.e., initializing them appropriately by making the best use out of the information collected by the base solver [MRT15; VRB+19].

Automating the Switching between Algorithms Very closely related to the former, the switching process needs to be automated. That is, it needs to track and adapt the choice of the current algorithm on its own. In particular, we only considered a single switch of algorithms, whereas in practical applications it may be beneficial to change the algorithm more often. To this end, several aspects of the algorithm selection pipeline would need further adaptation to be able to make recommendations "on the fly".

Effects of Feature Handling We majorly reduce the cost of feature extraction by integrating it into the optimization process. With the long-term aim to build a fully

online algorithm selection model, feature *computation* will also need to be performed in a more efficient way. For now, we rely on *flacco* whenever necessary. To further reduce the overhead of feature extraction, in particular when aiming for an "on-the-fly" detection of suitable moments to switch to another algorithm, feature computation should ideally also be done "on the fly".

Feature *selection* is another aspect that should be carefully considered. In this thesis, selecting the most informative and descriptive features is restricted for the purpose of sensitivity analysis. However, it has recently been shown that a small number of features suffices to distinguish the problem instances from the BBOB set, which our experiments are based on [RDD+20]. We therefore advise to investigate the feature selection on a broader collection of problem instances.

Selection of the Base Solver In this thesis, we predominantly considered the CMA-ES and its extension into a modular algorithm framework. This choice was based on their versatility across a wide range of optimization scenarios. However, choosing an appropriate base solver is a challenging problem in its own right. Depending on the setting we operate in – determined by the problem(s) to be solved and the resources that we have at our disposal to solve them – an algorithm exhibiting a different behavior might be a more suitable option for the base solver. Therefore, a systematic analysis of our trajectory-based approach on a broader spectrum of base algorithms is needed.

Extending the Approach to Real-World Scenarios We opted for the BBOB problem collection as our benchmark, but the question remains of how representative it is of more practical scenarios. Real-world application of our ideas would indispensably need more tailored training, and extension of the benchmark variety is another direction worth undertaking.

Bibliography

[ABH11] A. Auger, D. Brockhoff, and N. Hansen. **Mirrored Sampling in Evolution Strategies with Weighted Recombination**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'11)*. ACM, 2011, 861–868. DOI: 10.1145/2001576.2001694 (see page 15).

[ABH13] A. Auger, D. Brockhoff, and N. Hansen. **Benchmarking the local metamodel CMA-ES on the noiseless BBOB'2013 test bed**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'13), Companion Material*. ACM, 2013, 1225–1232. DOI: 10.1145/2464576.2482701 (see pages 16, 42, 56).

[AH05] A. Auger and N. Hansen. **A restart CMA evolution strategy with increasing population size**. In: *Proc. of Congress on Evolutionary Computation (CEC'05)*. 2005, 1769–1776. DOI: 10.1109/CEC.2005.1554902 (see page 15).

[AJT05] A. Auger, M. Jebalia, and O. Teytaud. **Algorithms (X, sigma, eta): Quasi-random Mutations for Evolution Strategies**. In: *Artificial Evolution*. Springer, 2005, 296–307. DOI: 10.1007/11740698_26 (see page 15).

[Ata15] A. Atamna. **Benchmarking IPOP-CMA-ES-TPA and IPOP-CMA-ES-MSR on the BBOB Noiseless Testbed**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15), Companion Material*. ACM, 2015, 1135–1142. DOI: 10.1145/2739482.2768467 (see pages 16, 42, 56).

[BAH+10] D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, and T. Hohm. **Mirrored Sampling and Sequential Selection for Evolution Strategies**. en. In: *Proc. of Parallel Problem Solving from Nature (PPSN'10)*. Springer, 2010, 11–21. DOI: 10.1007/978-3-642-15844-5_2 (see page 15).

[BBE+20] A. Biedenkapp, H. F. Bozkurt, T. Eimer, F. Hutter, and M. Lindauer. **Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework**. In: *Proc. of European Conference on Artificial Intelligence (ECAI'20)*. Vol. 325. Frontiers in Artificial Intelligence and Applications. IOS Press, 2020, 427–434. DOI: 10.3233/FAIA200122 (see page 23).

[BBH+19] A. Biedenkapp, H. F. Bozkurt, F. Hutter, and M. Lindauer. **Towards White-box Benchmarks for Algorithm Control**. *CoRR* abs/1906.07644 (2019). arXiv: 1906.07644 (see pages 2, 23).

[BCH+09] J. Benesty, J. Chen, Y. Huang, and I. Cohen. "Pearson correlation coefficient." In: *Noise reduction in speech processing*. Springer, 2009, 1–4. DOI: 10.1007/978-3-642-00296-0_5 (see page 80).

[BDS+17] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. **Per instance algorithm configuration of CMA-ES with limited budget**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*. ACM, 2017, 681–688. DOI: 10.1145/3071178.3071343 (see pages 3, 19, 20, 23, 37, 38, 43, 69).

[BFS+84] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. **Classification and regression trees**. CRC Press, 1984. ISBN: 9781315139470 (see pages 41, 56).

[BKJ+19] A. Blot, M.-E. Kessaci, L. Jourdan, and H. Hoos. **Automatic Configuration of Multi-Objective Local Search Algorithms for Permutation Problems**. *Evolutionary Computation* 27:1 (2019), 147–171. DOI: 10.1162/evco_a_00240 (see page 23).

[BKK+16] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, and J. Vanschoren. **ASlib: A benchmark library for algorithm selection**. *Artificial Intelligence* 237 (2016), 41–58. DOI: 10.1016/j.artint.2016.04.003 (see pages 22, 33).

[BLS13] I. Boussaïd, J. Lepagnot, and P. Siarry. **A survey on optimization metaheuristics**. *Information Sciences* 237 (2013), 82–117. DOI: 10.1016/j.ins.2013.02.041 (see page 12).

[BMT+12] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuss. **Algorithm selection based on exploratory landscape analysis and cost-sensitive learning**. In: *Proc. of Genetic and Evolutionary Computation Conference, GECCO'12*. ACM, 2012, 313–320. DOI: 10.1145/2330163.2330209 (see page 22).

[BP15] P. Baudiš and P. Pošík. **Global Line Search Algorithm Hybridized with Quadratic Interpolation and Its Extension to Separable Functions**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*. ACM, 2015, 257–264. DOI: 10.1145/2739480.2754717 (see page 15).

[BPR+19] L. Bajer, Z. Pitra, J. Repický, and M. Holena. **Gaussian Process Surrogate Models for the CMA Evolution Strategy**. *Evolutionary Computation* 27:4 (2019), 665–697. DOI: 10.1162/evco_a_00244 (see page 23).

[BPS+01] M. Birattari, L. Paquete, T. Stützle, and K. Varrentrapp. **Classification of Metaheuristics and Design of Experiments for the Analysis of Components**. Tech. rep. Intellektik, Darmstadt University of Technology, 2001 (see page 12).

[Bre01] L. Breiman. **Random forests**. *Machine Learning* 45:1 (2001), 5–32. DOI: 10.1023/A:1010933404324 (see pages 23, 41, 56).

[Bre96] L. Breiman. **Bagging predictors**. *Machine Learning* 24:2 (1996), 123–140. DOI: 10.1007/BF00058655 (see pages 41, 56).

[Bro70] C. G. Broyden. **The Convergence of a Class of Double-rank Minimization Algorithms: 2. The New Algorithm**. *IMA Journal of Applied Mathematics* 6:3 (1970), 222–231. DOI: 10.1093/imamat/6.3.222 (see page 15).

[BS04] N. Baskiotis and M. Sebag. **C4.5 Competence Map: A Phase Transition-Inspired Approach**. In: *Proc. of International Conference on Machine Learning (ICML'04)*. ACM, 2004, 10. DOI: 10.1145/1015330.1015398 (see page 22).

[CDK+06] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. **Online Passive-Aggressive Algorithms**. *Journal of Machine Learning Research* 7 (2006), 551–585. DOI: 10.5555/1248547.1248566 (see pages 41, 56).

[CDL+21] R. Cosson, B. Derbel, A. Liefooghe, H. E. Aguirre, K. Tanaka, and Q. Zhang. **Decomposition-Based Multi-objective Landscape Features and Automated Algorithm Selection**. In: *Proc. of Evolutionary Computation in Combinatorial Optimization (EvoCOP'21)*. Vol. 12692. LNCS. Springer, 2021, 34–50. DOI: 10.1007/978-3-030-72904-2_3 (see page 22).

[Com08] Wikimedia Commons. *Concept of directional optimization in CMA-ES algorithm*. 2008. URL: https://upload.wikimedia.org/wikipedia/commons/d/d8/Concept_of_directional_optimization_in_CMA-ES_algorithm.png (see page 14).

[CV97] D. J. Cook and R. C. Varnell. **Maximizing the Benefits of Parallel Search Using Machine Learning**. In: *Proceedings of AAAI Conference on Artificial Intelligence (AAAI'97)*. AAAI Press / The MIT Press, 1997, 559–564 (see page 22).

[DK92] S. Derksen and H. J. Keselman. **Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables**. *British Journal of Mathematical and Statistical Psychology* 45:2 (1992), 265–282. DOI: 10.1111/j.2044-8317.1992.tb00992.x (see page 80).

[DLV+17] F. Daolio, A. Liefooghe, S. Vérel, H. E. Aguirre, and K. Tanaka. **Problem Features versus Algorithm Performance on Rugged Multiobjective Combinatorial Fitness Landscapes**. *Evolutionary Computation* 25:4 (2017). DOI: 10.1162/evco_a_00193 (see page 19).

[DLV+19] B. Derbel, A. Liefooghe, S. Vérel, H. E. Aguirre, and K. Tanaka. **New features for continuous exploratory landscape analysis based on the SOO tree**. In: *Proc. of ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA'19)*. ACM, 2019, 72–86. DOI: 10.1145/3299904.3340308 (see pages 23, 84).

[DWY+18] C. Doerr, H. Wang, F. Ye, S. van Rijn, and T. Bäck. **IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics**. *CoRR* (2018). The BBOB datasets from [HAR+20] are available in the web-based interface of IOHanalyzer at http://iohprofiler.liacs.nl/. arXiv: 1810.05281 (see page 42).

[EJP+21] T. Eftimov, A. Jankovic, G. Popovski, C. Doerr, and P Korošec. **Personalizing performance regression models to black-box optimization problems**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21)*. ACM, 2021, 669–677. DOI: 10.1145/3449639.3459407 (see pages 6, 52).

[EPR+20] T. Eftimov, G. Popovski, Q. Renau, P. Korošec, and C. Doerr. **Linear Matrix Factorization Embeddings for Single-objective Optimization Landscapes**. In: *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI'20)*. IEEE, 2020, 775–782. DOI: 10.1109/SSCI47803.2020.9308180 (see page 84).

[ES15] A. E. Eiben and J. E. Smith. **Introduction to Evolutionary Computing**. 2nd. Springer, 2015. ISBN: 3662448734 (see page 1).

[FHT+01] J. Friedman, T. Hastie, R. Tibshirani, et al. **The elements of statistical learning**. Vol. 1. 10. Springer, 2001. ISBN: 9780387848587 (see page 57).

[FSK08] A. I. J. Forrester, A. Sobester, and A. J. Keane. **Engineering Design via Surrogate Modelling - A Practical Guide**. Wiley, 2008, I–XVIII, 1–210. ISBN: 9780470060681 (see page 1).

[FVH+14] C. Fawcett, M. Vallati, F. Hutter, J. Hoffmann, H. H. Hoos, and K. Leyton-Brown. **Improved Features for Runtime Prediction of Domain-Independent Planners**. In: *Proc. of International Conference on Automated Planning and Scheduling*. AAAI Press, 2014, 355–359. DOI: 10.5555/3038794.3038837 (see page 19).

[GFB+06] P. M. Granitto, C. Furlanello, F. Biasioli, and F. Gasperi. **Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products**. *Chemometrics and Intelligent Laboratory Systems* 83:2 (2006), 83–90 (see page 80).

[HAB19] N. Hansen, Y. Akimoto, and P. Baudiš. *CMA-ES/pycma on Github*. Zenodo. 2019. DOI: 10.5281/zenodo.2559634 (see pages 69, 76).

[HAB20] N. Hansen, A. Auger, and D. Brockhoff. *Data from the BBOB workshops*. https://coco.gforge.inria.fr/doku.php?id=algorithms-bbob. 2020 (see pages 42, 52).

[HAF+10] N. Hansen, A. Auger, S. Finck, and R. Ros. **Real-Parameter Black-Box Optimization Benchmarking: Experimental Setup**. RR-7215. INRIA, 2010 (see page 18).

[Han08] N. Hansen. **CMA-ES with Two-Point Step-Size Adaptation**. *CoRR* (2008). arXiv: 0805.0231 (see page 15).

[Han09] N. Hansen. **Benchmarking a BI-Population CMA-ES on the BBOB-2009 Function Testbed**. In: *Proc. of Genetic and Evolutionary Computation (GECCO'09), Companion Material*. ACM, 2009, 2389–2396. DOI: 10.1145/1570256.1570333 (see pages 15, 18, 42, 56).

[HAR+20] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. **COCO: a platform for comparing continuous optimizers in a black-box setting**. *Optimization Methods and Software* 36 (2020), 1–31. DOI: 10.1080/10556788.2020.1808977 (see pages 17, 91).

[HFR+09] N. Hansen, S. Finck, R. Ros, and A. Auger. **Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions**. Tech. rep. RR-6829. INRIA, 2009 (see page 18).

[HHH+06] F. Hutter, Y. Hamadi, H. H. Hoos, and K. Leyton-Brown. **Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms**. In: *Proc. of Principles and Practice of Constraint Programming (CP'06)*. Vol. 4204. LNCS. Springer, 2006, 213–228. DOI: 10.1007/11889205_17 (see page 23).

[HHL13] F. Hutter, H. Hoos, and K. Leyton-Brown. **An Evaluation of Sequential Model-Based Optimization for Expensive Black-Box Functions**. In: *Proc. of Genetic and Evolutionary Computation (GECCO'13), Companion Material*. ACM, 2013, 1209–1216. DOI: 10.1145/2464576.2501592 (see page 16).

[HK17] C. Hanster and P. Kerschke. **Flaccogui: Exploratory Landscape Analysis for Everyone**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17), Companion Material*. ACM, 2017, 1215–1222. DOI: 10.1145/3067695.3082477. URL: https://flacco.shinyapps.io/flacco/ (see page 21).

[HKV19] F. Hutter, L. Kotthoff, and J. Vanschoren, eds. **Automated Machine Learning - Methods, Systems, Challenges**. The Springer Series on Challenges in Machine Learning. Springer, 2019. DOI: 10.1007/978-3-030-05318-5 (see page 22).

[HN09] W. Huyer and A. Neumaier. **Benchmarking of MCS on the noiseless function testbed**. *Online* (2009). URL: https://mat.univie.ac.at/~neum/ms/mcs_exact.pdf (see pages 42, 56).

[HN99] W. Huyer and A. Neumaier. **Global optimization by multilevel coordinate search**. *Journal of Global Optimization* 14:4 (1999), 331–355. DOI: 10.1023/A:1008382309369 (see page 16).

[HO01] N. Hansen and A. Ostermeier. **Completely Derandomized Self-Adaptation in Evolution Strategies**. *Evolutionary Computation* 9:2 (2001), 159–195. DOI: 10.1162/106365601750190398 (see pages 13, 74).

[HXH+14] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. **Algorithm runtime prediction: Methods & evaluation**. *Artificial Intelligence* 206 (2014), 79–111. DOI: 10.1016/j.artint.2013.10.003 (see pages 19, 23).

[JA06] G. A. Jastrebski and D. V. Arnold. **Improving Evolution Strategies through Active Covariance Matrix Adaptation**. In: *Proc. of Congress on Evolutionary Computation (CEC'06)*. 2006, 2814–2821. DOI: 10.1109/CEC.2006.1688662 (see page 15).

[JD19] A. Jankovic and C. Doerr. **Adaptive Landscape Analysis**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'19), Companion Material*. ACM, 2019, 2032–2035. DOI: 10.1145/3319619.3326905 (see pages 6, 69).

[JD20] A. Jankovic and C. Doerr. **Landscape-Aware Fixed-Budget Performance Regression and Algorithm Selection for Modular CMA-ES Variants**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'20)*. ACM, 2020, 841–849. DOI: 10.1145/3377930.3390183 (see pages 6, 26, 49).

[JED21] A. Jankovic, T. Eftimov, and C. Doerr. **Towards Feature-Based Performance Regression Using Trajectory Data**. In: *Proc. of Applications of Evolutionary Computation (EvoApplications'21)*. Vol. 12694. LNCS. Springer, 2021, 601–617. DOI: 10.1007/978-3-030-72699-7_38 (see pages 6, 74).

[JPE+21] A. Jankovic, G. Popovski, T. Eftimov, and C. Doerr. **The Impact of Hyper-Parameter Tuning for Landscape-Aware Performance Regression and Algorithm Selection**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21)*. ACM, 2021, 687–696. DOI: 10.1145/3449639.3459406 (see pages 6, 40).

[KHN+19] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann. **Automated Algorithm Selection: Survey and Perspectives**. *Evolutionary Computation* 27:1 (2019), 3–45. DOI: 10.1162/evco_a_00242 (see pages 22, 23).

[KJR10] M. B. Kursa, A. Jankowski, and W. R. Rudnicki. **Boruta – a system for feature selection**. *Fundamenta Informaticae* 101:4 (2010), 271–285 (see page 80).

[Kot14] L. Kotthoff. **Algorithm Selection for Combinatorial Search Problems: A Survey**. *AI Mag.* 35:3 (2014), 48–60. DOI: 10.1609/aimag.v35i3.2460 (see page 22).

[KPW+15] P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann. **Detecting Funnel Structures by Means of Exploratory Landscape Analysis**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*. ACM, 2015, 265–272. DOI: 10.1145/2739480.2754642 (see page 21).

[KPW+16] P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann. **Low-Budget Exploratory Landscape Analysis on Multiple Peaks Models**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'16)*. 2016, 229–236. DOI: 10.1145/2908812.2908845 (see page 3).

[KT16] P. Kerschke and H. Trautmann. **The R-Package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems**. In: *Proc. of IEEE Congress on Evolutionary Computation (CEC'16)*. IEEE, 2016, 5262–5269. DOI: 10.1109/CEC.2016.7748359 (see pages 21, 69).

[KT19] P. Kerschke and H. Trautmann. **Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning**. *Evolutionary Computation* 27:1 (2019), 99–127. DOI: 10.1162/evco_a_00236 (see pages 2, 15, 22, 23, 38, 40, 42, 43, 49, 52).

[KT87] A. H. G. Rinnooy Kan and G. T. Timmer. **Stochastic global optimization methods part II: Multi level methods**. *Mathematical Programming* 39:1 (1987), 57–78. DOI: 10.1007/BF02592071 (see page 15).

[LDV+20] A. Liefooghe, F. Daolio, S. Vérel, B. Derbel, H. E. Aguirre, and K. Tanaka. **Landscape-Aware Performance Prediction for Evolutionary Multi-objective Optimization**. *IEEE Transactions on Evolutionary Computation* 24:6 (2020), 1063–1077. DOI: 10.1109/TEVC.2019.2940828 (see pages 19, 23).

[LHH+15] M. Lindauer, H. H. Hoos, F. Hutter, and T. Schaub. **AutoFolio: An Automatically Configured Algorithm Selector**. *Journal of Artificial Intelligence Research* 53 (2015), 745–778. DOI: 10.1613/jair.4726 (see page 22).

[LM02] M. Laguna and R. Martí. "The OptQuest Callable Library." In: *Optimization Software Class Libraries*. Ed. by S. Voss and D. L. Woodruff. Springer, 2002, 193–218. DOI: 10.1007/0-306-48126-X_7 (see page 16).

[LM19] B. Lacroix and J. A. W. McCall. **Limitations of benchmark sets and landscape features for algorithm selection and performance prediction**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'19), Companion Material*. ACM, 2019, 261–262. DOI: 10.1145/3319619.3322051 (see page 38).

[LMP+20] J. Liu, A. Moreau, M. Preuss, J. Rapin, B. Roziere, F. Teytaud, and O. Teytaud. **Versatile Black-Box Optimization**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'20)*. ACM, 2020, 620–628. DOI: 10.1145/3377930.3389838 (see page 22).

[LNA+03] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham. **A Portfolio Approach to Algorithm Selection**. In: *Proc. of International Joint Conference on Artificial Intelligence (IJCAI'03)*. Morgan Kaufmann, 2003, 1542. DOI: 10.5555/1630659.1630927 (see page 22).

[LRK17] M. Lindauer, J. N. van Rijn, and L. Kotthoff. **Open Algorithm Selection Challenge 2017: Setup and Scenarios**. In: *Proc. of Open Algorithm Selection Challenge*. Vol. 79. PMLR, 2017, 1–7 (see page 24).

[LSS13a] I. Loshchilov, M. Schoenauer, and M. Sebag. **Bi-population CMA-ES agorithms with surrogate models and line searches**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'13), Companion Material*. 2013, 1177–1184. DOI: 10.1145/2464576.2482696 (see pages 16, 56).

[LSS13b] I. Loshchilov, M. Schoenauer, and M. Sebag. **Intensive Surrogate Model Exploitation in Self-adaptive Surrogate-assisted Cma-es (Saacm-es)**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'13)*. ACM, 2013, 439–446. DOI: 10.1145/2463372.2463427 (see pages 16, 42).

[LVL+21] A. Liefooghe, S. Vérel, B. Lacroix, A.-C. Zavoianu, and J. A. W. McCall. **Landscape features and automated algorithm selection for multi-objective interpolated continuous optimisation problems**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21)*. ACM, 2021, 421–429. DOI: 10.1145/3449639.3459353 (see page 19).

[LW06] M. Lunacek and D. Whitley. **The dispersion metric and the CMA evolution strategy**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'06)*. ACM, 2006, 477–484. DOI: 10.1145/1143997.1144085 (see pages 19, 21).

[Mal18] K. M. Malan. **Landscape-Aware Constraint Handling Applied to Differential Evolution**. In: *Proc. of Theory and Practice of Natural Computing (TPNC'18)*. Vol. 11324. LNCS. Springer, 2018, 176–187. DOI: 10.1007/978-3-030-04070-3_14 (see page 37).

[Mal21] K. M. Malan. **A Survey of Advances in Landscape Analysis for Optimisation**. *Algorithms* 14:2 (2021), 40. DOI: 10.3390/a14020040 (see page 19).

[MBT+11] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph. **Exploratory Landscape Analysis**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21)*. ACM, 2011, 829–836. DOI: 10.1145/2001576.2001690 (see pages 2, 19, 20, 22).

[ME13] K. M. Malan and A. P. Engelbrecht. **A survey of techniques for characterising fitness landscapes and some possible ways forward**. *Information Sciences* 241 (2013), 148–163. DOI: 10.1016/j.ins.2013.04.015 (see pages 2, 19, 22).

[MKH12] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. **A Meta-learning Prediction Model of Algorithm Performance for Continuous Optimization Problems**. In: *Proc. of Parallel Problem Solving from Nature (PPSN'12)*. Vol. 7491. LNCS. Springer, 2012, 226–235. DOI: 10.1007/978-3-642-32937-1_23 (see pages 2, 22, 23).

[MKH15] M. A. Muñoz, M. Kirley, and S. K. Halgamuge. **Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content**. *IEEE Transactions on Evolutionary Computation* 19:1 (2015), 74–87. DOI: 10.1109/TEVC.2014.2302006 (see pages 2, 19, 21, 22).

[MKS18] M. A. Muñoz Acosta, M. Kirley, and K. Smith-Miles. **Reliability of Exploratory Landscape Analysis** (2018). DOI: 10.13140/RG.2.2.23838.64327 (see page 38).

[MM19] K. M. Malan and I. Moser. **Constraint Handling Guided by Landscape Analysis in Combinatorial and Continuous Search Spaces**. *Evolutionary Computation* 27:2 (2019), 267–289. DOI: 10.1162/evco_a_00222 (see page 37).

[MPR12] M. Maratea, L. Pulina, and F. Ricca. **Applying Machine Learning Techniques to ASP Solving**. In: *Proc. of International Conference on Logic Programming (ICLP'12)*. Vol. 17. LIPIcs. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012, 37–48. DOI: 10.4230/LIPIcs.ICLP.2012.37 (see page 19).

[MRT15] H. Mohammadi, R. Le Riche, and E. Touboul. **Making EGO and CMA-ES Complementary for Global Optimization**. In: *Proc. of Learning and Intelligent Optimization (LION'15)*. Vol. 8994. LNCS. Springer, 2015, 287–292. DOI: 10.1007/978-3-319-19084-6_29 (see pages 84, 86).

[MRW+21] L. Meunier, H. Rakotoarison, P.-K. Wong, B. Rozière, J. Rapin, O. Teytaud, A. Moreau, and C. Doerr. **Black-Box Optimization Revisited: Improving Algorithm Selection Wizards through Massive Benchmarking**. *IEEE Transactions on Evolutionary Computation* (2021). DOI: 10.1109/TEVC.2021.3108185 (see pages 2, 22).

[MS17] M. A. Muñoz and K. A. Smith-Miles. **Performance Analysis of Continuous Black-Box Optimization Algorithms via Footprints in Instance Space**. *Evolutionary Computation* 25:4 (2017). DOI: 10.1162/evco_a_00194 (see pages 29, 37).

[MSJ+12] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. de Sousa. **Ensemble Approaches for Regression: A Survey**. *ACM Computing Surveys* 45:1 (2012). DOI: 10.1145/2379776.2379786 (see page 41).

[MSK+15] M. A. Muñoz, Y. Sun, M. Kirley, and S. K. Halgamuge. **Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges**. *Information Sciences* 317 (2015), 224–245. DOI: 10.1016/j.ins.2015.05.010 (see page 23).

[Mun11] R. Munos. **Optimistic optimization of a deterministic function without the knowledge of its smoothness**. In: *Proc. of Neural Information Processing Systems (NIPS'11)*. 2011, 783–791 (see page 84).

[Mur12] K. P. Murphy. **Machine learning: a probabilistic perspective**. MIT Press, 2012. ISBN: 9780262018029 (see pages 41, 56).

[NM65] J. A. Nelder and R. Mead. **A Simplex Method for Function Minimization**. *The Computer Journal* 7:4 (Jan. 1965), 308–313. DOI: 10.1093/comjnl/7.4.308 (see page 1).

[NVW+21] J. de Nobel, D. Vermetten, H. Wang, C. Doerr, and T. Bäck. **Tuning as a means of assessing the benefits of new ideas in interplay with existing algorithmic modules**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'21), Companion Material*. 2021, 1375–1384. DOI: 10.1145/3449726.3463167 (see page 15).

[OMB20] G. Ochoa, K. M. Malan, and C. Blum. **Search Trajectory Networks of Population-Based Algorithms in Continuous Spaces**. In: *Proc. of Applications in Evolutionary Computation (EvoApplications'20)*. Springer, 2020, 70–85. DOI: 10.1007/978-3-030-43722-0_5 (see page 84).

[OV16] G. Ochoa and N. Veerapen. **Additional Dimensions to the Study of Funnels in Combinatorial Landscapes**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'16)*. ACM, 2016, 373–380. DOI: 10.1145/2908812.2908820 (see page 19).

[Pál13a] L. Pál. **Benchmarking a hybrid multi level single linkage algorithm on the BBOB noiseless testbed**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'13)*. 2013, 1145–1152. DOI: 10.1145/2464576.2482692 (see pages 42, 56).

[Pál13b] L. Pál. **Comparison of multistart global optimization algorithms on the BBOB noiseless testbed**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'13), Companion Material*. 2013, 1153–1160. DOI: 10.1145/2464576.2482693 (see pages 15, 16, 42, 56).

[PB15] P. Posík and P. Baudis. **Dimension Selection in Axis-Parallel Brent-STEP Method for Black-Box Optimization of Separable Continuous Functions**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15), Companion Material*. ACM, 2015, 1151–1158. DOI: 10.1145/2739482.2768469 (see pages 15, 42, 56).

[PEB+15] A. Piad-Morffis, S. Estévez-Velarde, A. Bolufé-Röhler, J. Montgomery, and S. Chen. **Evolution strategies with threshold convergence**. In: *Proc. of Congress on Evolutionary Computation (CEC'15)*. 2015, 2097–2104. DOI: 10.1109/CEC.2015.7257143 (see page 15).

[PKD20] M. Petković, D. Kocev, and S. Džeroski. **Feature ranking for multi-target regression**. *Machine Learning* 109:6 (2020), 1179–1204. DOI: 10.1007/s10994-019-05829-8 (see page 84).

[PM14] J. Pihera and N. Musliu. **Application of Machine Learning to Algorithm Selection for TSP**. *Proc. of IEEE International Conference on Tools with Artificial Intelligence (ICTAI'14)* (2014), 47–54. DOI: 10.1109/ICTAI.2014.18 (see page 19).

[Pow06] M. J. D. Powell. "The NEWUOA software for unconstrained optimization without derivatives." In: *Large-Scale Nonlinear Optimization*. Ed. by G. Di Pillo and M. Roma. Springer, 2006, 255–297. DOI: 10.1007/0-387-30065-1_16 (see page 16).

[Pow64] M. J. D. Powell. **An efficient method for finding the minimum of a function of several variables without calculating derivatives**. *The Computer Journal* 7:2 (Jan. 1964), 155–162. DOI: 10.1093/comjnl/7.2.155 (see page 1).

[PRH19] Zbynek Pitra, Jakub Repický, and Martin Holena. **Landscape analysis of Gaussian process surrogates for the covariance matrix adaptation evolution strategy**. In: *GECCO*. ACM, 2019, 691–699 (see pages 74, 77).

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. **Scikit-learn: Machine Learning in Python**. *Journal of Machine Learning Research* 12 (2011), 2825–2830 (see pages 30, 75).

[RDD+19] Q. Renau, J. Dréo, C. Doerr, and B. Doerr. **Expressiveness and Robustness of Landscape Features**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'19), Companion Material*. ACM, 2019, 2048–2051. DOI: 10.1145/3319619.3326913 (see pages 27, 29, 32, 38, 43).

[RDD+20] Q. Renau, C. Doerr, J. Dréo, and B. Doerr. **Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy**. In: *Proc. of Parallel Problem Solving from Nature (PPSN'20)*. Vol. 12270. LNCS. Springer, 2020, 139–153. DOI: 10.1007/978-3-030-58115-2_10 (see pages 20, 64, 87).

[Ric76] J. R. Rice. **The Algorithm Selection Problem**. *Advances in Computers* 15 (1976), 65–118. DOI: 10.1016/S0065-2458(08)60520-3 (see page 22).

[RK03] M. Robnik-Šikonja and I. Kononenko. **Theoretical and empirical analysis of ReliefF and RReliefF**. *Machine Learning* 53:1-2 (2003), 23–69. DOI: 10.1023/A:1025667309714 (see page 84).

[RT18] J. Rapin and O. Teytaud. *Nevergrad - A gradient-free optimization platform*. https://GitHub.com/FacebookResearch/Nevergrad. 2018 (see pages 50, 67).

[RWL+16] S. van Rijn, H. Wang, M. van Leeuwen, and T. Bäck. **Evolving the Structure of Evolution Strategies**. In: *Proc. of IEEE Symposium Series on Computational Intelligence (SSCI'16)*. IEEE, 2016, 1–8. DOI: 10.1109/SSCI.2016.7850138 (see pages 7, 14, 15, 73).

[SB18] A. E. Stork J.and Eiben and T. Bartz-Beielstein. *A new Taxonomy of Continuous Global Optimization Algorithms*. 2018. arXiv: 1808.08818 (see page 12).

[ŠEK20] U. Škvorc, T. Eftimov, and P. Korošec. **Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis**. *Applied Soft Computing* 90 (2020), 106138. DOI: 10.1016/j.asoc.2020.106138 (see page 76).

[SGW19] S. Saleem, M. Gallagher, and I. Wood. **Direct Feature Evaluation in Black-Box Optimization Using Problem Transformations**. *Evolutionary Computation* 27:1 (2019), 75–98. DOI: 10.1162/evco_a_00247 (see page 27).

[SKL+19] M. Sharma, A. Komninos, M. López-Ibáñez, and D. Kazakov. **Deep reinforcement learning based parameter control in differential evolution**. In: *Proc. of Genetic and Evolutionary Computation Conference, (GECCO'19)*. ACM, 2019, 709–717. DOI: 10.1145/3321707.3321813 (see page 23).

[Smi09] K. A. Smith-Miles. **Cross-Disciplinary Perspectives on Meta-Learning for Algorithm Selection**. *ACM Computing Surveys* 41:1 (2009). DOI: 10.1145/1456650.1456656 (see page 22).

[SSB94] S. Swarzberg, G. Seront, and H. Bersini. **S.T.E.P.: the easiest way to optimize a function**. In: *Proc. of IEEE Congress on Evolutionary Computation (CEC'94)*. Vol. 1. 1994, 519–524. DOI: 10.1109/ICEC.1994.349896 (see page 16).

[Tib96] R. Tibshirani. **Regression Shrinkage and Selection via the Lasso**. *Journal of the Royal Statistical Society. Series B (Methodological)* 58:1 (1996), 267–288. URL: http://www.jstor.org/stable/2346178 (see pages 41, 56).

[ULP+07] Z. Ugray, L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Martí. **Scatter Search and Local NLP Solvers: A Multistart Framework for Global Optimization**. *INFORMS Journal on Computing* 19:3 (2007), 328–340. DOI: 10.1287/ijoc.1060.0175 (see page 16).

[VRB+19] D. Vermetten, S. van Rijn, T. Bäck, and C. Doerr. **Online selection of CMA-ES variants**. In: *Proc. of Genetic and Evolutionary Computation Conference (GECCO'19)*. ACM, 2019, 951–959. DOI: 10.1145/3321707.3321803 (see pages 73, 86).

[WEB14] H. Wang, M. Emmerich, and T. Bäck. **Mirrored Orthogonal Sampling with Pairwise Selection in Evolution Strategies**. In: *Proc. of ACM Symposium on Applied Computing (SAC'14)*. ACM, 2014, 154–156. DOI: 10.1145/2554850.2555089 (see page 15).

[XHH+08] L. Xu, F. Hutter, H.H. Hoos, and K. Leyton-Brown. **SATzilla: Portfolio-based Algorithm Selection for SAT**. *Journal of Artificial Intelligence Research* 32 (2008), 565–606. DOI: 10.1613/jair.2490 (see pages 19, 22).

[XHH+12] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. **Evaluating Component Solver Contributions to Portfolio-Based Algorithm Selectors**. In: *Proc. of Theory and Applications of Satisfiability Testing (SAT'12)*. Vol. 7317. LNCS. Springer, 2012, 228–241. DOI: 10.1007/978-3-642-31612-8_18 (see page 22).

[ZFW+19] J. Zhao, Q. Feng, P. Wu, R. A. Lupu, R. A. Wilke, Q. S. Wells, J. C. Denny, and W.-Q. Wei. **Learning from longitudinal data in electronic health record and genetic data to improve cardiovascular event prediction**. *Scientific Reports* 9:1 (2019), 1–10. DOI: 10.1038/s41598-018-36745-x (see page 83).

[ZH05] H. Zou and T. Hastie. **Regularization and variable selection via the elastic net**. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67:2 (2005), 301–320. URL: https://www.jstor.org/stable/3647580 (see pages 41, 56).

[ZM12] C. Zhang and Y. Ma. **Ensemble machine learning: methods and applications**. Springer, 2012. DOI: 10.1007/978-1-4419-9326-7 (see page 52).