Optimizing solvers for real-world expensive black-box optimization with applications in vehicle design

Long, F.X.

Optimizing Solvers for Real-World Expensive Black-Box Optimization with Applications in Vehicle Design

Proefschrift

ter verkrijging van
de graad van doctor aan de Universiteit Leiden,
op gezag van rector magnificus prof.dr.ir. H. Bijl,
volgens besluit van het college voor promoties
te verdedigen op donderdag 27 november 2025
klokke 16:00 uur

door

Fu Xing Long

geboren te Pulau Pinang, Malaysia
in 1993

Promotores:
Prof. Dr. T.H.W. Bäck
Prof. Dr.-Ing. M. Gitterle (Munich University of Applied Sciences, Germany)

Co-promotor:
Dr. N. van Stein

Promotiecomissie:
Prof. Dr. M.M. Bonsangue
Prof. Dr. A. Plaat
Prof. Dr. K.J. Batenburg
Dr. E. Raponi
Prof. Dr.-Ing. F. Duddeck (Technical University of Munich, Germany)
Dr. M. Gallagher (University of Queensland, Australia)

This thesis was written as part of the joint project newAIDE under the consortium leadership of BMW AG with the partners Altair Engineering GmbH, divis intelligent solutions GmbH, MSC Software GmbH, Technical University of Munich, TWT GmbH. The project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag.

"The greatest challenge to any thinker is stating the problem in a way that will allow a solution." – Bertrand Russell

Contents

Contents

Chapter 1

Introduction

In general, Black-Box Optimization (BBO) problems refer to a class of optimization problems, where the objective and constraint functions are typically available in the form of black boxes [6]. In real-world applications, for instance, the inputs of BBO problems can be evaluated using laboratory experiments and/or virtual simulations to obtain the corresponding outputs, where an analytical understanding of the underlying process is still lacking. For solving BBO problems, various optimization algorithms have been developed over the years, such as the nature-inspired Evolutionary Algorithm (EA) like Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [45], swarm-based algorithms like Particle Swarm Optimization (PSO) [55], and surrogate-assisted algorithms like Bayesian Optimization (BO) [85]. Given that different algorithms have their own strengths and weaknesses, there is no such universal optimization algorithm that can effectively solve all kinds of BBO problems [156]. Consequently, identifying the most time- and resource-efficient algorithm for specific BBO problems is critical to ensure an optimal optimization performance, which is also commonly known as the Algorithm Selection Problem (ASP) [112]. Nevertheless, selecting the most suitable optimization algorithm can be extremely challenging, especially for practitioners inexperienced in the optimization field and lack of domain knowledge.

One of the ongoing research directions towards automated Algorithm Selection (AS) in the evolutionary computation community focuses on associating the performance of optimization algorithms with the optimization landscape of BBO problems. Essentially, based on some landscape analysis tools, the optimization landscape characteristics of BBO problems can be numerically quantified, such as global struc-

ture and multi-modality [80]. The so-called landscape features can be related with the performance of optimization algorithms, e.g., using Machine Learning (ML) approaches [14, 49, 58]. By employing supervised ML models, for instance, the performance of optimization algorithms on the BBO problems can be roughly estimated based on their landscape characteristics. Subsequently, the most suitable optimization algorithm for unseen BBO problems can be conveniently identified, once their optimization landscape characteristics have been computed.

In previous work like [58], landscape-aware AS approaches have shown encouraging success on BBO benchmark suites, such as the Black-Box Optimization Benchmarking (BBOB) functions [44]. When using benchmark suites, a large set of problem instances necessary for the training of reliable ML models can be easily generated and cheaply evaluated. On the other hand, little work has been attempted to investigate landscape-aware AS for BBO in real-world applications with expensive function evaluations. Compared to benchmark suites, the generation of real-world problem instances can be particularly limited, the landscape characteristics of real-world problems are not well understood, and the function evaluations are prohibitively costly and/or time-consuming. Consequently, generating a large training dataset for ML models is infeasible, and thus, typical AS approaches are not practical for solving real-world expensive BBO problems. To the best of our knowledge, exploration of suitable AS approaches for real-world expensive BBO problems is still lacking.

1.1 Thesis Objective

Motivated to fill the gaps, the primary objective of this thesis is to develop an automated optimization approach for an optimal solving of real-world expensive BBO problems w.r.t. real-world constraints, such as allocated time and resources available for optimization. Precisely, our aim is to investigate an automated approach that can optimally fine-tune optimization algorithms for solving expensive BBO problems using a limited function evaluation budget. Eventually, the proposed optimization pipeline can be deployed to assist practitioners unfamiliar with AS in properly fine-tuning optimization algorithms for their applications.

The general workflow of our proposed optimization pipeline is illustrated in Figure 1.1, consisting of altogether four steps. Fundamentally, by analyzing the optimization landscape characteristics of expensive BBO problems, some test functions that are (i) *cheap-to-evaluate* and (ii) belong to the *same optimization problem class* can be identified. Essentially, these test functions, which we refer to as *representative func-*

tions, can be considered as *surrogates* of the BBO problems. Unlike in a ML context, we are not interested in replacing or approximating the true functions of expensive BBO problems using some representative functions. Instead, we aim to establish a preferably large set of test functions with similar landscape characteristics, which can be considered for the fine-tuning of optimization algorithms, since the performance of optimization algorithms should be comparable between the actual BBO problems and their representative functions. By exploiting these cheap representative functions, near-optimal optimization algorithms for the BBO problems can be identified at a significantly lower computational cost, prior to the actual optimization runs using expensive function evaluations. In other words, a much more flexible function evaluation budget can be afforded for the fine-tuning of optimization algorithms based on some representative functions. Eventually, the actual BBO problems can be effectively solved using the identified near-optimal optimization algorithms.

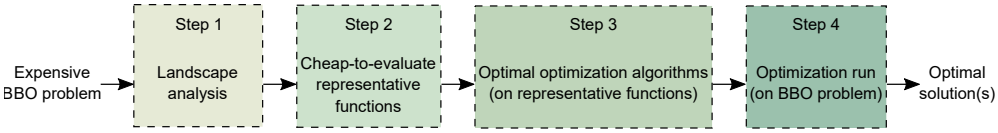


Figure 1.1: Outline of our automated optimization pipeline proposed for solving expensive BBO problems using a limited function evaluation budget. Principally, the optimization landscapes of expensive BBO problems are analyzed (Step 1) to identify cheap-to-evaluate representative functions (Step 2). By exploiting these representative functions, near-optimal optimization algorithms can be cheaply identified (Step 3) and then applied to solve the BBO problems (Step 4).

Within the scope of this thesis, we consider crashworthiness optimization problems in the automotive industry as representative real-world engineering BBO problems with expensive function evaluation. As vehicle design becomes increasingly sophisticated nowadays, solving crashworthiness optimization problems using conventional optimization approaches is getting notoriously challenging and tedious, which start losing their effectiveness. Subsequently, alternatives that can provide better vehicle designs, and thus, accelerate the overall vehicle development progress are gaining more attention in the automotive industry. Compared to conventional optimization approaches for solving automotive crash problems, we aim to achieve a better performance using our proposed optimization pipeline in terms of:

- Using less computational resources w.r.t. expensive function evaluations;
- Being less time-consuming w.r.t. total wall-clock time; and/or

- Finding better solution(s) w.r.t. optimization objective.

Despite automotive crashworthiness optimization problems are being considered as representative real-world applications in this thesis, the potential of our pipeline is not limited to them. In fact, the pipeline is designed and developed with the long-term vision that it can be applied to other expensive BBO problems beyond automotive crashworthiness optimization. Principally, the proposed pipeline could be further extended to other engineering applications with appropriate modifications, such as vehicle control system calibration in the automotive industry [131], ship design in the maritime industry [25], and turbomachinery design in the aerospace industry [103].

Beyond that, another crucial aspect of our proposed optimization pipeline is that optimal optimization algorithms can be identified automatically, requiring minimal input from practitioners. From our perspective, this is essential for real-world applications and can potentially reduce the workload of practitioners, especially for those that are unfamiliar with fine-tuning of optimization algorithms. Interestingly, this is in line with the latest trend, where the so-called *one-click* optimizers are getting introduced in commercial optimization tools [115].

1.2 Research Questions

During the journey to achieve the research objective defined for this thesis, i.e., developing an automated optimization pipeline for optimally solving expensive BBO problems, we focus on addressing the following research questions. In this context, our investigations are primarily based on the widely-used BBOB functions and automotive crashworthiness optimization as a representative example of real-world expensive BBO problems, refer to Section 1.1.

RQ1: Within the feature space defined by optimization landscape features, how are the distributions of real-world expensive BBO problems situated w.r.t. some benchmark functions?

In the landscape-aware AS approach, the optimization landscape characteristics of BBO problems are exploited to identify optimal optimization algorithms. To the best of our knowledge, however, an investigation about the optimization landscape of real-world expensive BBO problems is still lacking. Thus, what are the optimization problem classes of real-world BBO problems? Is there any similarity between real-world BBO problems and benchmark functions in terms of optimization landscape characteristics? To what extent can benchmark

functions be considered as representative functions for real-world BBO problems? (Chapter 3)

RQ2: If none of the benchmark functions can sufficiently represent the optimization problem classes of real-world expensive BBO problems, how to augment the benchmark functions with other test functions that are appropriate to serve as representative functions?

Despite the fact that benchmark functions cover a variety of optimization problem classes, a sufficient coverage of the problem classes in real-world applications might still be lacking. For such cases, how can we complement the set of benchmark functions with more test functions to further improve the overall diversity of optimization problem classes? More importantly, are any of these newly included test functions appropriate to be considered as representative functions for the fine-tuning of optimization algorithms? (Chapter 4)

RQ3: How well can we estimate the actual performance of optimization algorithms on real-world expensive BBO problems based on some cheap-to-evaluate representative functions?

In our optimization approach, we propose to optimally fine-tune optimization algorithms for real-world expensive BBO problems, by exploiting some cheap-to-evaluate representative functions from the same optimization problem classes. In this context, how well can we predict the performance of different optimization algorithms on unseen BBO problems based on some representative functions? More critically, can we identify optimal optimization algorithms for BBO problems in this way? Moreover, are all representative functions appropriate for the fine-tuning of optimization algorithms? Otherwise, how can we properly identify those that are appropriate for this task? (Chapter 4)

RQ4: How to specifically generate test functions that belong to the same optimization problem classes of real-world expensive BBO problems?

Since the hand-crafted benchmark function suites can never fully cover all optimization problem classes, it is valuable to have an alternative that can generate test functions belonging to specific optimization problem classes. In this regard, how can we guide the generation of test functions towards a particular optimization problem class, e.g., based on some optimization landscape characteristics? What is the potential of such approach, and if any, the challenges that must be overcome? (Chapter 4)

RQ5: What is the performance of optimal optimization algorithms identified using the proposed optimization approach (Section 1.1), when tested on some benchmark functions? More crucially, can the optimal optimization algorithms outperform some state-of-the-art approaches for solving real-world expensive BBO problems?

In this thesis, we propose to fine-tune optimization algorithms for real-world expensive BBO problems based on some cheap-to-evaluate representative functions. From this perspective, what is the performance of such optimal optimization algorithms in general? In other words, can we identify well-performing optimization algorithms in this way? Eventually, can we optimally solve real-world expensive BBO problems using the fine-tuned optimization algorithms w.r.t. some real-world constraints? (Chapter 5)

RQ6: What is the potential of pre-trained general purpose predictive models in identifying optimal optimization algorithms for real-world expensive BBO problems?

Similar to a landscape-aware ASP context, we are inspired to explore an alternative that can identify optimal optimization algorithms for real-world expensive BBO problems using some predictive models. In this regard, how can we improve the performance of predictive models to preferably generalize well across different applications? Compared to typical landscape-aware ASP, how can we take it one step further, towards selecting optimal algorithms as well as fine-tuning their hyperparameters? (Chapter 5)

1.3 Thesis Outline

In summary, the structure of this thesis is highlighted in the following:

Chapter 2 briefly presents a literature review and additional information that are essential for comprehending and gaining insights into different research topics that have been explored within the framework of this thesis, such as a gentle introduction to BBO, fine-tuning of algorithm configurations, fitness landscape analysis using Exploratory Landscape Analysis (ELA), and automotive crashworthiness optimization.

Chapter 3 first introduces an approach for capturing the optimization landscape characteristics of BBO problems based on some ELA features. Apart from the

classical ELA features, we also explore a feature-free alternative that can characterize BBO problems based on some latent representations, e.g., computed using deep Neural Network (NN) models. To have a better understanding, we also take a closer look at a large set of BBOB instances w.r.t. their optimization landscape characteristics and optimization performances, since the BBOB suite is heavily utilized in our investigations. Lastly, we analyze the optimization landscape of several real-world automotive crashworthiness optimization problems in terms of ELA features, based on a comparison against those of the BBOB functions.

Chapter 4 summarizes the generation of cheap-to-evaluate representative functions for real-world expensive BBO problems using a tree-based random function generator. Apart from the fact that such representative functions are similar in terms of optimization landscape, we show that they can be exploited for estimating the actual performance of optimization algorithms on unseen BBO problems. Beyond that, we evaluate the potential of guiding the function generation towards specific optimization problem classes based on some ELA features, by extending the random function generator with Genetic Programming (GP).

Chapter 5 describes in detail the automated optimization pipeline proposed for optimally solving real-world expensive BBO problems. For a comprehensive performance assessment, the proposed approach is first evaluated across a wide range of optimization problem classes using the BBOB suite. More importantly, the approach is applied and tested on a real-world automotive crashworthiness optimization problem, using expensive Finite Element (FE) simulation runs. Furthermore, we investigate the potential of training predictive ML models that can efficiently identify optimal algorithms for BBO problems, potentially improving the overall effectiveness of the proposed approach.

Chapter 6 summarizes significant research findings of this thesis, discusses current limitations, and provides potential improvements for future work.

1.4 Author's Contributions

Within the scope of this thesis, all important scientific findings and results have been previously published and contributed as part of the research community, as summarized in the following:

Journal Publications

1. Fu Xing Long, Bas van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Generating Cheap Representative Functions for Expensive Automotive Crashworthiness Optimization. *ACM Trans. Evol. Learn. Optim.*, 4(2), jun 2024.

2. Fu Xing Long, Niki van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Surrogate-based automated hyperparameter optimization for expensive automotive crashworthiness optimization. *Struct. Multidiscip. Optim.*, 68(4), April 2025.

Peer-reviewed Conference Publications

1. Fu Xing Long, Bas van, Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Learning the characteristics of engineering optimization problems with applications in automotive crash. *In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22*, page 1227–1236, New York, NY, USA, 2022. Association for Computing Machinery.*

2. Fu Xing Long, Diederick Vermetten, Bas van Stein, and Anna V. Kononova. BBOB Instance Analysis: Landscape Properties and Algorithm Performance Across Problem Instances. *In Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023*, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings, page 380–395, Berlin, Heidelberg, 2023. Springer-Verlag.†

3. Bas van Stein, Fu Xing Long, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. DoE2Vec: Deep-Learning Based Features for Exploratory Landscape Analysis. *In Proceedings of the Companion Conference on Genetic and Evolutionary Computation, GECCO '23 Companion*, page 515–518, New York, NY, USA, 2023. Association for Computing Machinery.

4. Fu Xing Long, Diederick Vermetten, Anna V. Kononova, Roman Kalkreuth, Kaifeng Yang, Thomas Bäck, and Niki van Stein. Challenges of ELA-Guided Function Evolution Using Genetic Programming. *In Proceedings of the 15th International Joint Conference on Computational Intelligence - Volume 1: ECTA*, pages 119–130. INSTICC, SciTePress, 2023.

*Best Paper Award
†Outstanding Students of 2023

5. Fu Xing Long, Niki van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Surrogate-Based Algorithm Selection and Hyperparameter Tuning for Automotive Crashworthiness Optimization. *In The 15th World Congress of Structural and Multidisciplinary Optimisation*, 2023.

6. Roy de Winter, Fu Xing Long, Andre Thomaser, Thomas H.W. Bäck, Niki van Stein, and Anna V. Kononova. Landscape analysis based vs. domain-specific optimization for engineering design applications: A clear case. *In 2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 776–781, 2024.

7. Fu Xing Long, Moritz Frenzel, Peter Krause, Markus Gitterle, Thomas Bäck, and Niki van Stein. Landscape-Aware Automated Algorithm Configuration Using Multi-output Mixed Regression and Classification. *In Parallel Problem Solving from Nature – PPSN XVIII*, pages 87–104, Cham, 2024. Springer Nature Switzerland.

Chapter 2

Preliminaries

This chapter summarizes some fundamental information that are essential for an in-depth comprehension of this thesis. Firstly, a gentle introduction to BBO is provided in Section 2.1, including state-of-the-art optimization algorithms and benchmark functions. In Section 2.2, an overview about the fine-tuning of optimization algorithms is briefly highlighted. This is followed by an insight into ELA for capturing the optimization landscape of BBO problems in Section 2.3. Lastly, conventional optimization approaches and recent advancements in solving real-world automotive crashworthiness optimization are discussed in Section 2.4.

2.1 Black-Box Optimization

Generally, BBO belongs to a class of optimization problems, where an analytic form is not available, derivative information is not reliable or lacking, and numerical approximation of the derivatives is costly [6, 9]. Without loss of generality, a continuous, single-objective BBO problem can be represented using Equation 2.1 [8]:

$$\begin{aligned} &\text{minimize } f : \Omega \subset \mathbb{R}^d \to \mathbb{R} \\ &\text{subject to } g_i(\boldsymbol{x}) \leq 0 \; \forall i \in \{1, \cdots, p\}; \boldsymbol{x} \in \Omega\,, \end{aligned} \tag{2.1}$$

where f is an objective function, g are some constraint functions, p is the number of constraints, d is the problem dimensionality, $\boldsymbol{x} = (x_1, \ldots, x_d)$ represents an optimization solution, and Ω denotes the search space. Unless otherwise stated, the optimization problems throughout this thesis are defined as minimization problems. In

many real-world BBO problems, the objective and constraint functions are commonly evaluated using laboratory experiments or virtual simulations. Hence, such problems using expensive function evaluations are also known as *expensive* BBO problems.

Due to the fact that derivative information is not available, classical gradient-based optimization approaches are not applicable for BBO problems. Subsequently, various BBO algorithms have been developed over the years, which can perform exceptionally well on benchmark suites. In this regard, an overview for several optimization algorithms investigated in this thesis is provided in Section 2.1.1 and for the BBOB suite in Section 2.1.2.

2.1.1 State-of-the-art Optimization Algorithms

In recent developments of BBO algorithms, rapid advancements in derivative-free EAs and surrogate-based approaches have been witnessed [8]. Basically, most of the bio-inspired EAs are developed based on the principle of survival-of-the-fittest, where a population evolves over generations to produce better individuals through mutation and recombination. For instance, CMA-ES and Differential Evolution (DE) [128] are two powerful EAs for solving BBO problems. On the other hand, surrogate-based approaches like BO iteratively search for promising solutions through resampling, by balancing between exploration of unknown regions within the search space and exploitation of the best-found solution. Principally, these algorithms are performed in a few iterations until a termination condition is fulfilled, such as the function evaluation budget has been elapsed or the improvement in the best-found solutions of the last few iterations is marginal. In this thesis, we primarily focus on the following three BBO algorithms, namely CMA-ES, DE, and BO.

Covariance Matrix Adaptation Evolutionary Strategy

Considered by many as one of the top-performing BBO algorithms, CMA-ES was developed and extended based on Evolutionary Strategy (ES) [107, 108, 116]. In brief, the general framework of ES can be outlined as follows:

1. In the first generation, a parent population is initialized.
2. Next, new offspring are created through recombination of selected parents. To diversify the population, the offspring undergo mutation, which is crucial for escaping from being stuck in local optima.
3. Based on a selection strategy, survivals are then selected for the next generation.

4. Step 2 and 3 are repeated until a termination condition is fulfilled.

Compared to ES, CMA-ES has the advantage that the populations in previous generation are considered for adapting the mutation distribution, which is a covariance matrix [45, 46]. Combined with the information about evolution paths, correlated mutations that are invariant to search space rotations can be employed to update the covariance matrix. Furthermore, the step size in CMA-ES is modified using a cumulative step size adaptation mechanism based on the conjugate evolution paths [40], which is another crucial aspect for its empirical performance.

Since then, various extensions and variants with different improvements for CMA-ES have been progressively introduced, such as active update [51], mirrored sampling [17], threshold convergence [95], and weighted recombination [46]. Subsequently, this motivates the development of a modular framework for CMA-ES [23], which we refer to as Modular Covariance Matrix Adaptation Evolutionary Strategy (ModCMA) in this thesis. In brief, a collection of CMA-ES variants are integrated into ModCMA as modules that can be independently activated or deactivated, offering a high degree of flexibility for a custom instantiation of CMA-ES. Following this, the interactions between different CMA-ES variants as well as between variants and hyperparameters, e.g., population size and learning rates, can be conveniently investigated using ModCMA.

Differential Evolution

Apart from CMA-ES, DE is another widely-used BBO algorithm from the family of EAs, which is a popular choice for its performance, simplicity, and relatively few hyperparameters [128]. The general workflow of DE can be summarized as follows:

1. In the first generation, a population is initialized, and ideally, evenly spread across the search space.

2. Using a mutation strategy, several "donor" solutions (mutants) are created by adding the scaled difference between a pair of solutions to another solution, which are randomly selected.

3. Next, new "trial" solutions (offspring) can be generated through a recombination of donor and "target" solutions (parents).

4. Afterwards, the trial solutions compete against target solutions to determine survivals for the next generation.

5. Step 2 to 4 are repeated until a termination condition is fulfilled.

Fundamentally, the underlying structure of DE is modular by design, where the operations of mutation and recombination are independent, and hence, can be completely separated. Similar to ModCMA, a modular framework consisting of a variety of DE variants has been recently proposed, which we refer to as Modular Differential Evolution (ModDE) [142].

Bayesian Optimization

Unlike EAs, BO is a class of iterative optimization algorithms that explores and searches for better solutions through resampling promising regions within the search space, relying on approximating the true functions using surrogate models [85]. Later, BO was popularized for BBO problems with expensive function evaluations in engineering domain under the name Efficient Global Optimization (EGO) [52]. In fact, BO is widely applied for solving expensive BBO problems using a limited function evaluation budget. A comprehensive explanation of BO and its extensions, e.g., for multi-objective optimization and constraint handling, is available in [153]. For the sake of consistency, we refer to standard BO and EGO simply as BO in the remaining of this thesis. Basically, the general workflow of BO can be described as follows:

1. A set of initial Design of Experiments (DoE) samples is generated using a sampling method, such as Latin Hypercube Sampling (LHS) [81]. Ideally, the DoE samples are evenly distributed across the design space, for a better approximation of the true functions using surrogate models.

2. Based on the DoE samples, a surrogate model is constructed to approximate the true optimization functions with uncertainties. In practice, Gaussian Process Regression (GPR) [61] model is commonly employed as surrogate model in the standard BO approach.

3. By exploiting the information provided by the surrogate model, an acquisition function is utilized to determine promising candidates to be evaluated, which is also known as resampling. Essentially, the acquisition function is maximized through an internal optimization within the BO framework. Some popular choices of acquisition function are Expected Improvement (EI) [52], Probability of Improvement (PI) [84], Lower Confidence Bound (LCB), and Upper Confidence Bound (UCB) [124], each offering a different search strategy in balancing between the explorative and exploitative behavior of BO algorithm.

4. After evaluating the suggested candidates, the new solutions are added to the set of DoE samples. In expensive BBO problems, the candidates are evaluated using expensive function evaluations.

5. Step 2 to 4 are repeated until a termination condition is fulfilled.

One of the weaknesses of BO is that its performance suffers from the curse of dimensionality and tends to deteriorate for high-dimensional problems. Following this, different variants of BO have been introduced over the years to better handle high-dimensional problems, such as Principal Component Analysis assisted Bayesian Optimization (PCA-BO) [106], Kernel Principal Component Analysis assisted Bayesian Optimization (KPCA-BO) [5], Trust Region Bayesian Optimization (TuRBO) [32], Sparse Axis-Aligned Subspace Bayesian Optimization (SAASBO) [31], and Heteroscedastic Evolutionary Bayesian Optimization (HEBO) [21]. An extensive analysis of these BO variants is provided in [113]. Recently, it has been shown that slightly adjusted standard BO can achieve improved performance in high dimensionality, which calls for further investigations [48].

Further Optimization Algorithms

Apart from the aforementioned algorithms, we also briefly employ Genetic Programming (GP) to guide the function evolution towards specific optimization problem classes in this thesis (Section 4.3). Inspired by neo-Darwinian evolution [64], the workflow of standard GP is similar to other EAs, i.e., by evolving a population based on the principle of survival-of-the-fittest and biologically-inspired operators. Previously, GP has been commonly applied for solving symbolic regression problems [130], where GP is employed to construct an explicit mathematical expression based on a given dataset. Following this, human-interpretable surrogate models can be build to analyze the relationship between different decision variables of BBO problems.

Beyond that, we also utilize several other BBO algorithms in this thesis, such as PSO [55], Estimation of Multivariate Normal Algorithm (EMNA) [66], Constrained Optimization by Linear Approximation with Random Restart (RCobyla) [97], Simultaneous Perturbation Stochastic Approximation (SPSA) [123], NGOpt [104], and Random Search (RS). Comprehensive overviews of further state-of-the-art BBO algorithms are available in [6, 8, 9, 10, 96].

2.1.2 Black-Box Optimization Benchmarking Suite

With the exponential growth of BBO algorithms over the years, a fair performance assessment and comparison between them heavily relies on suitable benchmark suites that cover a diverse set of optimization problem classes. For example, the BBOB family of problem suites are some of the most utilized sets of problems considered for benchmarking optimization heuristics [41]. Initially proposed in [44], the BBOB suites have been integrated into the Comparing Continuous Optimizers (COCO) platform [43] and Iterative Optimization Heuristic (IOH) profiler [28]. As one of the most established platforms for benchmarking purposes, COCO facilitates a comparison between algorithms by storing detailed performance statistics, which can be easily retrieved, processed, and compared against a constantly expanding set of publicly accessible data [42]. Due to their popularity, the BBOB suites have become a popular testbed for the benchmarking of algorithm configuration fine-tuning techniques, even though they were never originally designed for this purpose.

In this thesis, we focus on the BBOB suite that consists of 24 single-objective, noiseless, and continuous functions, which we refer to as *the* BBOB functions. According to their high-level properties [82, 83], the 24 BBOB functions can be separated into five main groups, namely (i) separable problems, (ii) low or moderate conditioned problems, (iii) high conditioned and unimodal problems, (iv) multi-modal problems with adequate global structure, and (v) multi-modal problems with weak global structure, as presented in Table 2.1.

One of the main advantages of the BBOB suite is that the functions can be scaled to arbitrary dimensionality, facilitating an investigation of different problem dimensionalities. Furthermore, arbitrarily many problem instances or variants of BBOB functions can be generated through transformations of both the search space and objective values, which are controlled internally by a unique identifier, also known as instance ID (IID). Principally, problem instances of the same BBOB function belong to the same optimization problem class. For most of the BBOB functions, the search space transformations consist of a combination of translation and rotation matrices. Since the objective values can be transformed as well, the optimization performance metrics commonly used are relative to the global optimum, allowing a comparison between different instances. While the BBOB suite was originally intended to be used for unconstrained optimization, in practice, most investigations based on the BBOB functions focus on the search space $[-5, 5]^d$, where the global optimum is located inside $[-4, 4]^d$.

Table 2.1: Classification of 24 BBOB functions into five groups (separated by horizontal lines) based on their high-level properties, such as multi-modality, global structure, separability, variable scaling, homogeneity, basin-sizes, and global to local contrast. *From top to bottom:* Separable problems (F1–F5), low or moderate conditioned problems (F6–F9), high conditioned and unimodal problems (F10–F14), multi-modal problems with adequate global structure (F15–F19), and multi-modal problems with weak global structure (F20–F24). Table taken from [82].

	BBOB Function	**multim.**	**gl.-struc.**	**separ.**	**scaling**	**homog.**	**basins**	**gl.-loc.**
F1	Sphere	none	none	high	none	high	none	none
F2	Ellipsoidal separable	none	none	high	high	high	none	none
F3	Rastrigin separable	high	strong	none	low	high	low	low
F4	Bueche-Rastrigin	high	strong	high	low	high	med.	low
F5	Linear Slope	none	none	high	none	high	none	none
F6	Attractive Sector	none	none	high	low	med.	none	none
F7	Step Ellipsoidal	none	none	high	low	high	none	none
F8	Rosenbrock	low	none	none	none	med.	low	low
F9	Rosenbrock rotated	low	none	none	none	med.	low	low
F10	Ellipsoidal high-cond.	none	none	none	high	high	none	none
F11	Discus	none	none	none	high	high	none	none
F12	Bent Cigar	none	none	none	high	high	none	none
F13	Sharp Ridge	none	none	none	low	med.	none	none
F14	Different Powers	none	none	none	low	med.	none	none
F15	Rastrigin multi-modal	high	strong	none	low	high	low	low
F16	Weierstrass	high	med.	none	med.	high	med.	low
F17	Schaffer F7	high	med.	none	low	med.	med.	high
F18	Schaffer F7 mod. ill-cond.	high	med.	none	high	med.	med.	high
F19	Griewank-Rosenbrock	high	strong	none	none	high	low	low
F20	Schwefel	med.	deceptive	none	none	high	low	low
F21	Gallagher 101 Peaks	med.	none	none	med.	high	med.	low
F22	Gallagher 21 Peaks	low	none	none	med.	high	med.	med.
F23	Katsuura	high	none	none	none	high	low	low
F24	Lunacek bi-Rastrigin	high	weak	none	low	high	low	low

Recently, effort has been invested to further diversify the benchmark problem classes available in the BBOB suite using an affine combination of two selected BBOB functions [26]. In brief, new benchmark functions can be generated through interpolation between the selected BBOB functions, by employing a weighting factor to control the shifting between them. Later, this function generation approach was generalized to affine combinations of multiple BBOB functions, also known as Many-Affine Black-Box Optimization Benchmarking (MA-BBOB) functions [147, 148]. In other words, the affine combination mechanism has been extended to a combination of multiple BBOB functions, thereby allowing the generation of more complex functions.

Beyond that, the so-called Instance Space Analysis (ISA) [121] is another exciting research direction, attempting to improve the diversity of optimization problem classes provided by existing benchmark suites. Essentially, the instance space covered by benchmark suites is analyzed based on some feature-based representations of the problem instances, to identify poorly-covered regions within the instance space. Oftentimes, ISA is performed using a performance-oriented perspective w.r.t. a set of optimization algorithms, ensuring that the newly created functions are discriminative in terms of algorithm performances. For instance, GP can be employed for the generation of new benchmark functions to fill the gaps based on some ELA features [88].

2.2 Fine-tuning of Algorithm Configuration

Since the performance of BBO heuristics highly depends on their hyperparameter setting, an optimal configuration is necessary to achieve the best possible optimization performance w.r.t. the best-found solution, time, and computational resources. In short, the terminology *hyperparameter* refers to a parameter of optimization algorithms that can be defined by practitioners to control the search behavior during optimization runs, such as the learning rates in CMA-ES. The process of identifying optimal algorithm configurations is also commonly known as Hyperparameter Optimization (HPO) [37]. Apart from fine-tuning only the hyperparameters, HPO can be combined with AS, becoming Automated Algorithm Configuration (AAC) or Combined Algorithm Selection and Hyperparameter Optimization (CASH) [133]. Essentially, the main purpose of CASH is to identify the most suitable optimization algorithm from an algorithm portfolio, while simultaneously fine-tuning its hyperparameters. In literature, these terminologies are sometimes used interchangeably, since a clear definition is still lacking. Based on the descriptions provided in [114], we consider the following definitions in this thesis for the sake of consistency:

AS: Selecting an optimal algorithm from an algorithm portfolio, without fine-tuning its hyperparameters.

HPO: Fine-tuning the hyperparameters of a particular optimization algorithm.

AAC / CASH: Selecting an optimal algorithm and fine-tuning its hyperparameters.

In this context, we also refer to the term *algorithm configuration* as a set of hyperparameter settings of an optimization algorithm. To the best of our knowledge, existing HPO approaches typically require a large function evaluation budget, and hence, are less applicable for real-world expensive BBO problems.

Since there is no such a one-for-all optimizer that can perform well on all BBO problems [156], HPO is vital to fine-tune optimization algorithms for maximal optimization efficiency. Nevertheless, this task can be extremely challenging, particularly for practitioners with limited experience and domain knowledge in HPO. Consequently, various optimizers have been developed over the years to facilitate an automated HPO process, such as Tree-structured Parzen Estimator (TPE) [13], Sequential Model-based Algorithm Configuration (SMAC) [69], Iterated Racing (irace) [78], and Mixed-Integer Parallel Efficient Global Optimization (MIP-EGO) [140], where the hyperparameter search space typically consists of continuous, integer, categorical, and conditional variables. Within the scope of this thesis, we primarily focus on the surrogate model-based HPO approaches, namely TPE and SMAC.

TPE: As a variant of the BO algorithm, TPE has a similar framework as the standard BO, but utilizes Parzen estimators as surrogate models, which can handle mixed-integer search space and scale well to high dimensionality. For example, TPE has been previously applied to fine-tune the learning rates of CMA-ES [159].

SMAC: To handle the mixed-integer search space, Random Forest (RF) models are utilized in SMAC as surrogate models. Subsequently, the mean and variance computed from the individual trees in the forest can be considered as uncertainty of RF models, which is necessary for the computation of acquisition functions, as described in Section 2.1.1.

Typically in HPO, the optimal algorithm configuration identified is applied and fixed throughout an optimization run. As optimization runs progress, nevertheless, this algorithm configuration might not always be optimal. In other words, a better optimization performance could be achieved using other configurations. Hence, recent research about HPO has been extended towards dynamic algorithm configuration,

where configurations are fine-tuned and adapted on the fly during an optimization run [1, 145], which is beyond the scope of this thesis.

2.3 Exploratory Landscape Analysis

In evolutionary computation, substantial work has been invested in landscape-aware ASP for the fine-tuning of algorithm configuration based on the landscape characteristics of BBO problems. Subsequently, this encourages the development of abundant techniques for fitness landscape analysis [80]. As one of the well-established landscape analysis methods, ELA facilitates a numerical quantification of the high-level properties of an optimization landscape [83], such as global structure, multi-modality, and separability (refer to Table 2.1), based on some carefully designed low-level features. Apart from the six fundamental classes of low-level features originally proposed in ELA, namely y-distribution, level set, meta-model, local search, curvature, and convexity [82], more complementary features have been gradually included [59], such as dispersion [79], Nearest Better Clustering (NBC) [56], Principal Component Analysis (PCA), linear model, and Information Content of Fitness Sequences (ICoFiS) [89]. Previously, it has been shown that ELA features are indeed sufficiently informative for a reliable classification of the BBOB functions [111], for AS purposes [14, 49, 58, 90], and for HPO tasks [12, 27].

Essentially, ELA features can be computed using a set of DoE samples $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ and the corresponding objective values $\mathcal{Y} = \{y_1, \cdots, y_n\}$, where $f \colon \mathcal{X} \to \mathcal{Y}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, and n represents the DoE sample size. Consequently, the computation of ELA features is dependent on factors, such as DoE sample size, sampling strategy, and problem dimensionality [87, 152]. In fact, concerns regarding the following aspects of ELA features have been previously raised, namely:

- Many of the ELA features are highly correlated and redundant, particularly for those belonging to the same feature class [150];
- Some of the ELA features are insufficiently expressive in distinguishing different problem instances [110];
- The hand-crafted ELA features might suffer from potential bias in capturing certain optimization landscape characteristics [118]. Furthermore, since ELA was developed and evaluated mainly using the BBOB suite, the design and problem classes of BBOB functions might guide the development of ELA to

some extent, raising the questions as to whether ELA can generalize well to problem classes beyond the BBOB suite; and

- ELA features are in general less discriminative for high-dimensional problems [86].

Attempting to overcome the drawbacks of ELA features, feature-free landscape analysis approaches based on latent space representations computed using deep-learning techniques have been actively explored, such as Deep-ELA [118, 119] and DoE2Vec [139] (Section 3.1.2). Further relevant work is available in [2, 99, 102]. Compared to the classical ELA features, however, an interpretation and understanding of these latent space representations are not as straightforward.

Apart from handling automated AS tasks, ELA has shown promising potential in a wide range of applications. For instance, ELA has been applied to understand the optimization landscape of neural architecture search tasks [141], to analyze the problem space of different benchmark problem sets [150], to study multi-objective optimization problems [57], and to examine the landscape properties of engineering problems like vehicle dynamics control systems [131]. Furthermore, a landscape-aware fine-tuning of the acquisition function in BO algorithm based on ELA features has been recently investigated in [12]. As far as we know, no relevant work has been previously attempted to analyze the optimization landscape characteristics of real-world expensive BBO problems, such as automotive crashworthiness optimization.

2.4 Automotive Crashworthiness Optimization

In recent years, the competitive landscape in the automotive industry has changed drastically due to the emergence of competent automobile manufacturers worldwide and the growth of market preference toward Battery Electric Vehicle (BEV). To stay competitive and relevant in the market, automobile manufacturers are facing enormous pressure to improve different aspects, such as enhancing the vehicle designs, broadening the spectra of vehicle models, shortening vehicle development cycles, and effectively lowering production costs [29]. For instance, vehicle designs nowadays are becoming increasingly sophisticated due to the stricter regulations on road safety imposed by authorities and the fact that various features and functionalities must be integrated to better retain customer satisfaction in terms of safety, comfort, driving experience, and entertainment. Subsequently, vehicle development demands a close cooperation between multiple disciplines, such as crashworthiness, structural statics,

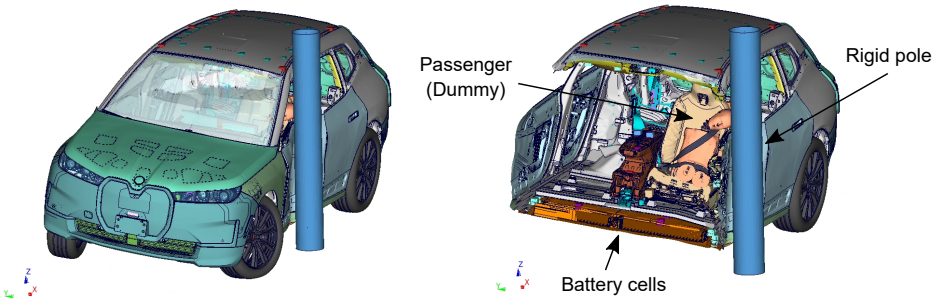
driving dynamics, and acoustics [18, 47]. Hence, an efficient strategy for the vehicle designing process during the early development phase is crucial to optimize time to market and to minimize investment costs. In this context, optimization approaches based on virtual simulations have been heavily integrated to identify vehicle designs that can optimally fulfill requirements imposed by different disciplines.

One of the ongoing, yet demanding research topics regarding vehicle design is the optimization of vehicle design w.r.t. crashworthiness, which is also commonly known as crashworthiness optimization. Belonging to the group of expensive BBO problems, automotive crashworthiness optimization problems are usually solved using numerical FE simulation runs, as shown in Figure 2.1 and Figure 2.2. Within the scope of this thesis, we focus on automotive side crash as a representative crash scenario, where the battery cells in BEV must be additionally protected from serious damages during crash. Basically, the main goal of automotive crashworthiness optimization is to identify vehicle designs that can provide sufficient protection to passengers in the event of a crash, while fulfilling other requirements, such as being durable and lightweight to reduce fuel consumption and production costs. Considering that crash problems are oftentimes strongly nonlinear, discontinuous, and high-dimensional, and the fact that FE simulations are computationally costly, solving automotive crashworthiness optimization can be tremendously challenging and time-consuming.

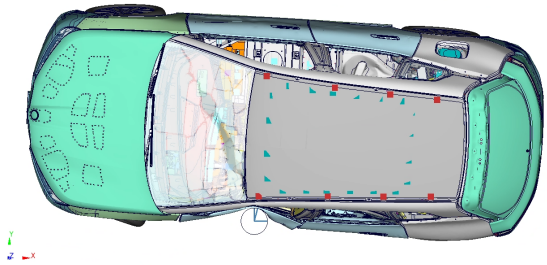
2.4.1 Surrogate-based Optimization Approaches

Classically, an automotive crashworthiness optimization problem is solved using the one-shot optimization approach, where the best vehicle design in a set of DoE samples is simply considered as the optimization solution [15, 16]. The main advantage of this approach is that multiple vehicle designs can be evaluated at once using parallel simulation runs.

Extended from the one-shot optimization approach, the Response Surface Method (RSM) has gained popularity in the automotive industry and is widely applied for automotive crashworthiness optimization, where response surfaces or data-driven surrogate models are employed to find optimal vehicle designs [35]. In short, surrogate models like polynomial function, Radial Basis Function (RBF), and GPR models are trained using a set of DoE samples to approximate the true optimization problems. With sufficiently high approximation accuracy, solutions better than the best DoE sample can be identified using RSM. Over the years, substantial work has been invested to improve the effectiveness of RSM for solving vehicle design problems [34, 93].



(a) Full-view (*left*) and cross-sectioned view (*right*) of the deformed vehicle during crash impact. To protect passengers and battery cells in BEV, the crash impact energy must be sufficiently absorbed through plastic deformation of different components in the crumple zones.



(b) Top view of the deformed vehicle. Depending on the investigation purposes, the position of side pole can be adjusted alongside the vehicle body.

Figure 2.1: Example of FE model developed for an automotive crashworthiness optimization problem with side impact against a rigid pole. Figures taken from [72, 73].

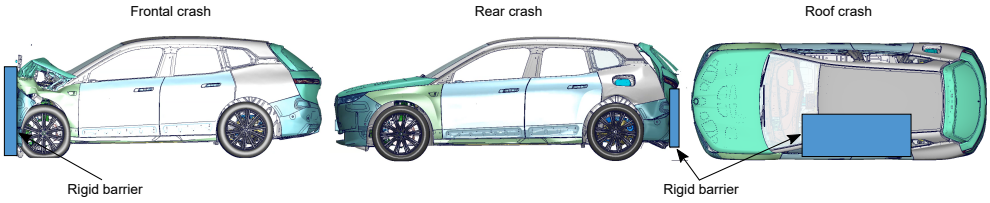


Figure 2.2: Side view of FE models developed for vehicle frontal crash (*left*), rear crash (*middle*), and roof crash (*right*) against a rigid barrier. The roof crash scenario, for instance, is designed for investigating a vehicle rollover or an overturn situation during crash.

Based on RSM, Sequential or Successive Response Surface Method (SRSM) [62, 125, 127] was developed to identify better solutions within the design space through an iterative resampling, as shown in Figure 2.3. Essentially, a new response surface is constructed to approximate the local optimization problem within a subregion of the search space in each iteration. Based on the best-found solution in each iteration, the subregion gradually shrinks and progressively moves towards regions with better solutions. Subsequently, the fitting quality of response surfaces can be improved by successively reducing the size of subregion. On top of that, engineering expertise and domain knowledge can be additionally provided to guide the adaptation of subregion towards regions that are expected to yield better solutions, potentially accelerating the solving of automotive crash problems. Nevertheless, this process is challenging, as such information might be unintentionally biased towards certain assumptions and/or experiences from the past might be less relevant for new vehicle designs. Despite SRSM has demonstrated promising potential in solving crashworthiness optimization problems [65], its application in the automotive industry is still rather limited, e.g., due to the poorly fitted response surfaces, substantial computational expenses using an iterative resampling, and challenges in sharing information between iterations [35].

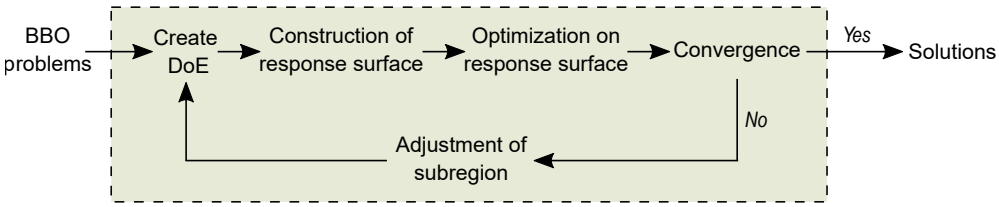


Figure 2.3: Example of typical optimization workflow based on SRSM for BBO problems.

Over the years, a variety of design optimization tools has been developed and offered to solve engineering BBO problems in the automotive industry, such as `Optimus` (Noesis Solutions) [92], `HyperStudy` (Altair) [3], `LS-OPT` (DYNAmore) [30], `optiSLang` (Ansys) [4], and `HEEDS` (Siemens) [120]. While a detailed explanation about the optimization approaches available in these commercial tools is not available due to confidentiality reasons, most of them are implemented based on RSM and/or SRSM with modifications, according to the best of our understanding. Following this, we would expect that their optimization performances would be similar or with some improvements, compared to standard RSM and SRSM.

Due to the rapid developments in vehicle designing process, both RSM and SRSM

are gradually losing their effectiveness in solving complex automotive crashworthiness optimization problems, which can be briefly summarized as follows:

Low fitting quality: The performance of RSM and SRSM is heavily dependent on the fitting quality of response surfaces in approximating the true optimization problems. Since vehicle designs are getting evermore complicated, the strongly nonlinear and discontinuous characteristics in crashworthiness optimization usually lead to a poorly fitted response surface, and eventually, a sub-optimal solution; and

Expensive simulations: As the complexity of vehicle FE models rises exponentially over the years, the solving time and computational resources required for FE simulations have also increased significantly, limiting the FE simulation runs and optimization iterations that can be performed. Consequently, it is often infeasible in the automotive industry to improve the fitting quality of response surfaces in RSM and SRSM by increasing the DoE sample size.

Meanwhile, the BO algorithm that excels for solving expensive BBO problems using a small function evaluation budget (Section 2.1.1) has shown promising potential in solving automotive crashworthiness optimization problems [39, 129]. Despite of that, the problem dimensionalities investigated were relatively low, i.e., in 5-d and 6-d respectively. As the complexity of vehicle design increases nowadays, a higher problem dimensionality must be considered for optimization, e.g., commonly between 15-d and 25-d. Moreover, the hyperparameters of BO algorithm must be properly fine-tuned for an effective optimization performance, e.g., the choices of surrogate model and acquisition function, which can be challenging for practitioners lacking in domain knowledge.

In summary, optimization approaches that can effectively solve crashworthiness optimization problems have gained growing attention in the automotive industry. Beyond that, recent optimization approaches focus on relieving practitioners from the monotonous and exhausting manual tasks associated with conventional optimization approaches. This can be achieved, for example, by integrating artificial intelligence into the optimization workflow. Subsequently, practitioners can redirect their focus and resources towards enhancing other aspects of the vehicle designing process, such as a better definition of the vehicle design problem.

2.4.2 Alternative Optimization Approaches

Apart from the conventional surrogate-based optimization approaches, various alternatives have been actively explored for solving complex automotive crashworthiness optimization problems. Some of the related recent advancements are presented in the following:

- While population-based optimization heuristics, such as CMA-ES, DE, and PSO, are powerful in solving BBO problems, a large function evaluation budget is usually required for an optimization convergence. Unlike benchmark suites, where a budget of $10\,000 \cdot d$ evaluations is still relatively affordable for an optimization run, such an evaluation budget is infeasible for expensive BBO problems. Following this, these algorithms are currently less practical for real-world expensive BBO problems like automotive crashworthiness optimization [158]. Meanwhile, progressive developments have been attempted to overcome the limitation of population-based algorithms for applications in expensive BBO domains [68];

- Multi-fidelity optimization [54] and model order reduction [22, 67] are some other active research areas concerning crashworthiness optimization. Basically, BBO problems are solved at a reduced cost, by utilizing low-fidelity models with an acceptable trade-off between modeling accuracy and computational resources. While these techniques have shown good potential in solving simple crashworthiness optimization problems, an application for complex vehicle designs in the automotive industry is still rather limited;

- Another intriguing research direction is to leverage the potential of transfer learning approaches for automotive crash optimization, by learning and transferring knowledge from the past to future vehicle design problems [20]. One of the main advantages of this learning-based approach is that the historical data and information can be exploited to guide the development of future vehicle designs, potentially reducing the need for expensive simulation runs. Nevertheless, this technique is still in its early developing phase and further investigations are necessary for real-world applications in the automotive industry; and

- From the perspective of numerical modeling, the computation time and resources necessary for simulation runs can be effectively reduced through some simplifications of FE models [155]. In a so-called FE submodel, only parts of a vehicle that are relevant for the crash dynamics are retained, while the rest can be replaced using appropriate substituting mass and boundary conditions. Following

this, the computational resources necessary for a simulation run can be greatly reduced, allowing more simulation runs to be performed. While the usage of FE submodels can be beneficial in reducing computational effort, they might be less practical for complex vehicle design problems, especially when the crash dynamics can be hardly represented and/or the additional effort needed for developing submodels are not well worth it.

Chapter 3

Landscape Analysis of Engineering Optimization Problems in Automotive Crash

As introduced in Section 1.1, we propose considering cheap-to-evaluate representative functions with similar optimization landscape characteristics for the fine-tuning of optimization configurations. Following this, we focus on having a better understanding about the optimization landscape characteristics of real-world expensive BBO problems w.r.t. some benchmark functions. In this context, two different approaches to capture the optimization landscape characteristics of BBO problems are first introduced in Section 3.1, namely based on the widely-used ELA features and the latent space information extracted using deep NN models. Next, our investigation is extended to analyze the function properties of a large set of BBOB problem instances in Section 3.2, since the BBOB functions are heavily considered throughout this thesis. This is followed by an analysis of the optimization problem classes for several real-world automotive crashworthiness optimization problems in Section 3.3. Lastly, Section 3.4 summarizes and concludes this chapter.

3.1 Capturing the Landscape Characteristics of Optimization Problems

To capture the optimization landscape characteristics of expensive BBO problems, we evaluate the potential of two different approaches. Firstly, we consider numerically quantifying the representative landscape characteristics of BBO problems using some ELA features, which has shown promising potential in previous work. As explained in detail in Section 3.1.1, the respective optimization problem classes of BBO problems can be conveniently identified based on their ELA features.

Apart from the hand-crafted ELA features, we investigate a feature-free alternative in capturing the optimization landscape of BBO problems using deep NN models, which we call *DoE2Vec*, as introduced in Section 3.1.2. Essentially, we propose an automated self-supervised representation learning approach to characterize the optimization landscape of BBO problems based on some latent space representations.

3.1.1 Landscape Features using ELA

An overview of our ELA-based approach proposed to quantify the optimization landscape characteristics of real-world expensive BBO problems is visualized in Figure 3.1, consisting of four steps in total. Essentially, by comparing the ELA features of a BBO problem against some test functions, such as the well-known BBOB functions, the optimization problem class of the BBO problem can be identified based on their similarity in terms of ELA features. In other words, a BBO problem belongs to the same optimization problem class as the test functions that have similar ELA features. Subsequently, these test functions with similar optimization landscape characteristics can be considered as representative functions for further usages, e.g., fine-tuning of algorithm configurations for the BBO problem. In the following, our approach is explained in detail:

Input: A set of DoE samples of the BBO problem instance to-be-solved is required as input. Unless otherwise stated, the Sobol' sampling strategy [122] is employed for the generation of DoE samples, as suggested in [109]. Nevertheless, any sampling strategy or a custom DoE can be utilized in practice.

Data pre-processing: The input DoE samples are pre-processed, where duplicated samples or samples with incomplete data are discarded, e.g., missing values due to interrupted FE simulation runs. To facilitate a fair comparison of ELA features between the BBO problem and test functions, the search space is rescaled

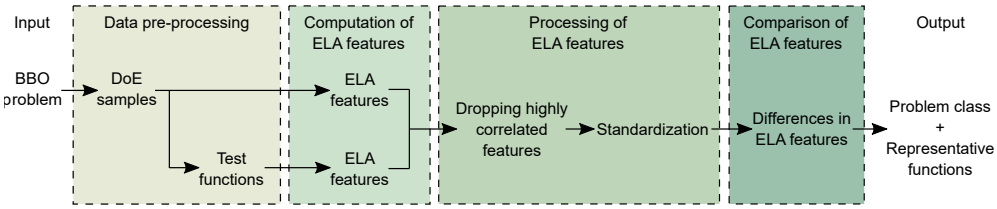


Figure 3.1: Overview of our approach to characterize the optimization landscape of real-world expensive BBO problems, requiring some DoE samples as input and consisting of four steps as marked with boxes. By comparing the ELA features of a BBO problem against those of some test functions, the optimization problem class of the BBO problem can be determined. On top of that, suitable representative functions can be identified based on the ELA features. Figure taken from [73].

to $[-5, 5]^d$ using Equation 3.1, which is commonly considered for the BBOB functions.

$$x_{new} = \frac{x_{orig} - a_{min}}{a_{max} - a_{min}} \cdot (b_{max} - b_{min}) + b_{min} \,, \tag{3.1}$$

where x_{orig} and x_{new} are the design variables before and after rescaling, a_{min} and a_{max} are the original minimum and maximum scale range, and b_{min} and b_{max} are the minimum and maximum scale range after rescaling. Following this, the same DoE samples and problem dimensionality are utilized to compute the objective values of test functions. To minimize potential inherent bias in ELA feature computation, the objective values are min-max normalized before the computation [100].

Computation of ELA features: ELA features are separately computed for the BBO problem and test functions using `pflacco` [98, 101], which was implemented based on `flacco` [59, 60]. Among the more than 300 available ELA features, we only consider features that fulfill the following requirements:

- Features that can be cheaply computed using the input DoE samples, without requiring additional resampling or function evaluations;
- Features that concern only the DoE samples $\mathcal{X}$ are neglected, e.g., some of the PCA features; and
- Features that are not informative about the problem landscape are ignored, e.g., concerning the computational costs.

Since an analytical form is not known and function evaluations are costly in real-world expensive BBO problems, computation of ELA features that require

resampling is particularly limited. Consequently, mainly the eight ELA feature classes summarized in Table 3.1* are considered in this thesis, which can be relatively cheaply computed without resampling.

To improve the reliability of ELA feature computation, particularly for a small DoE sample size in expensive BBO problems, and to minimize the impact of random sampling in ELA feature computation, we consider the mean ELA feature values computed based on a bootstrapping strategy. In brief, the ELA feature computation is repeated for 20 times using a subset of the input DoE samples, consisting of roughly 80% or $0.8 \cdot n$ samples that are randomly selected in each repetition. Eventually, the mean feature values are considered for further analysis. Furthermore, when the BBOB functions are considered as test functions, the ELA feature values are additionally averaged across the first 20 BBOB instances to improve their robustness. In cases where a feature computation fails, e.g., when the sample size is too small for computing the level set or linear model features, it will be skipped. Consequently, fewer ELA features will be computed for such a problem instance.

Processing of ELA features: Before the ELA features between the BBO problem and test functions are compared against each other, two feature processing steps are performed as follows:

1. Since many of the ELA features are highly correlated and redundant [109, 150], only a subset of all ELA features computed is considered for the comparison. Precisely, for each highly correlated feature pair based on Pearson's correlation coefficient (> 0.95), the feature that has a higher mean correlation with other remaining features is removed. Meanwhile, ELA features that have a constant value across all functions are automatically neglected; and
2. To ensure that the computed ELA features are within a comparable scale range, they are standardized by removing the mean and scaling to unit variance.

Both steps are first performed on the ELA features of the test functions, e.g., BBOB functions, and then applied to those of the BBO problem, using the same

*It is worth noting that, `flacco` was initially employed for the ELA feature computation during some early investigations in this thesis, which was later replaced by `pflacco` due to its higher compatibility with our optimization pipeline. As such, the ELA features computed might differ slightly between investigations. However, this variation does not significantly impact our results according to some preliminary testings.

Table 3.1: Brief descriptions of the ELA features from eight feature classes considered in this thesis, with the respective labels for feature classes and ELA features. Using `flacco` (in `R`), more level set features can be computed, as highlighted in gray color, which are not yet implemented in `pflacco` (in `Python`). Table taken from [73].

Feature class	**Description**	**ELA feature**
y-distribution (`ela_distr.*`)	Distribution of function values. 3 features	`skewness` `kurtosis` `number_of_peaks`
Level set (`ela_level.*`)	Measure the performance of different classification methods based on function value thresholds. 9 features (in `pflacco`) (or 18 features in `flacco`)	`mmce_lda_{10,25,50}` `mmce_qda_{10,25,50}` `mmce_mda_{10,25,50}` `lda_qda_{10,25,50}` `lda_mda_{10,25,50}` `qda_mda_{10,25,50}`
Meta-model (`ela_meta.*`)	Fitting quality of linear and quadratic models with and without interactions. 9 features	`lin_simple.{adj_r2,intercept}` `lin_simple.coef.{min,max,max_by_min}` `lin_w_interact.adj_r2` `quad_simple.{adj_r2,cond}` `quad_w_interact.adj_r2`
Dispersion (`disp.*`)	Comparison of dispersion between initial sample points and subset of points based on function value thresholds. 16 features	`ratio_mean_{02,05,10,25}` `ratio_median_{02,05,10,25}` `diff_mean_{02,05,10,25}` `diff_median_{02,05,10,25}`
NBC (`nbc.*`)	Comparison of distance between all sample points towards nearest points and nearest points with better function value. 5 features	`nn_nb.{sd_ratio,mean_ratio,cor}` `dist_ratio.coeff_var` `nb._fitness.cor`
PCA (`pca.*`)	Information based on PCA on initial sample points. 2 features	`expl_var_PC1.{cov_init,cor_init}`
Linear model (`limo.*`)	Measure the average coefficient vectors across multiple linear models. 4 features	`avg_length.{reg,norm}` `length.mean` `ratio.mean`
ICoFiS (`ic.*`)	Measure of smoothness, ruggedness, and neutrality of the landscape through random walk. 5 features	`h.max` `eps.{s,max,ratio}` `m0`

subset of ELA features as well as the same mean and variance for standardization. This is necessary to minimize potential information leakage, similar to the over-fitting problem in a ML context. Moreover, both steps are essential to improve the comparison results based on a distance-based metric in the next step.

Comparison of ELA features: Using the Euclidean distance metric and an equal weighting of all remaining ELA features, the similarity between the BBO problem and test functions w.r.t. ELA features can be quantified. Subsequently, the optimization problem class of the BBO problem can be identified based on its neighboring test functions with a small difference in ELA features.

Output: The landscape characteristics of the BBO problem in terms of ELA features and a set of neighboring test functions with similar optimization landscape characteristics are provided as output. These similar test functions can be potentially considered as representative functions for the BBO problem for further usages.

In our approach, the optimization landscape of real-world expensive BBO problems are characterized based on some ELA features computed using an initial set of DoE samples, without requiring additional costly function evaluations. While the effectiveness of ELA features could be potentially sacrificed, since they are dependent on factors like DoE sample size and problem dimensionality, as discussed in Section 2.3, this step is essential to facilitate an application of our approach for expensive BBO problems. Given that adding more DoE samples can be particularly limited in expensive BBO problems, we propose to consider an initial DoE sample size that is affordable w.r.t. real-world constraints, while the representative landscape characteristics can still be sufficiently captured. Based on our preliminary testing, for instance, a DoE of around $20 \cdot d$ samples seems to be an optimal trade-off for the automotive crashworthiness optimization problems investigated in this thesis (Chapter 5). Nonetheless, a closer investigation on this topic is necessary to have a better understanding.

Apart from considering some test functions with similar optimization landscape characteristics as representative functions, another alternative could be using some data-driven surrogate models trained on some DoE samples for the fine-tuning of algorithm configurations. While this approach seems to be much more straightforward, our approach has the advantage that optimization configurations are fine-tuned for a particular optimization problem class, rather than only for that particular problem instance, similar to a ML over-fitting problem. In other words, optimal configurations identified using our approach could perform equally well on other problem instances

that belong to the same optimization problem class. A similar research topic was briefly explored in [157].

3.1.2 DoE2Vec: A Feature-free Approach using Autoencoder

On the contrary to the hand-crafted ELA features, here we investigate a feature-free approach to capture the optimization landscape characteristics of real-world expensive BBO problems. Principally, informative latent representation vectors are extracted from some DoE samples using deep NN models, e.g., Variational Autoencoder (VAE), and thus, the name DoE2Vec. Subsequently, the low-dimensional representations of the optimization landscape of BBO problems can be captured in an automated, generic, and unsupervised manner. Compared to the classical ELA approach, DoE2Vec has the following advantages, namely:

- Domain knowledge in optimization landscape analysis w.r.t. feature engineering and feature selection is not strictly required;
- Bias toward particular landscape characteristics in the latent representations, if any, can be minimized; and
- Independent of the sampling method.

In the following, a brief introduction to the VAEs employed in DoE2Vec is presented, followed by an overview of the DoE2Vec workflow, and lastly some result discussions and limitations of DoE2Vec are provided. To the best of our knowledge, a similar approach to capture the optimization landscape characteristics of BBO problems using VAEs has not yet been attempted.

Variational Autoencoder in DoE2Vec

Similar to a standard Autoencoder (AE), the VAEs employed in DoE2Vec have a similar architecture, consisting of three components, namely an encoder, a hidden layer, also known as bottleneck, and a decoder, as shown in Figure 3.2. In DoE2Vec, the objective values $\mathcal{Y}$ of some DoE samples are provided as input for a VAE. Essentially, the encoder projects the input space $\mathcal{Y}$ to a representative feature space $\mathcal{H}$, i.e., $v\colon \mathcal{Y} \to \mathcal{H}$, while the decoder transforms the feature space back to the input space $w\colon \mathcal{H} \to \hat{\mathcal{Y}}$ [19]. To better capture crucial representations of the input space, a VAE is constructed using an input layer and an output layer of the same dimension $d_{\mathcal{Y}}$, but a bottleneck

layer of lower dimension $d_{\mathcal{H}}$, i.e., $d_{\mathcal{H}} < d_{\mathcal{Y}}$. By encoding the latent space as a distribution using a mean and variance layer, along with a sampling method, the latent space can be properly regularized to provide more meaningful latent representations.

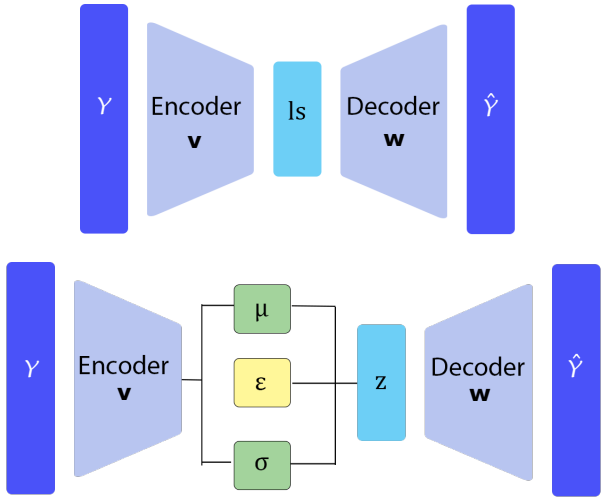


Figure 3.2: Example of standard AE (*top*) and VAE (*bottom*) considered in DoE2Vec. In comparison to an AE, the latent space of a VAE is encoded as a distribution using a mean and variance layer, together with a sampling method. The latent space z is defined by $z = \mu + \sigma \cdot \epsilon$, where ϵ denotes the sampling using a mean μ and a variance σ. Figure taken from [139].

During the unsupervised training process, a VAE attempts to optimally reconstruct the original input space $\mathcal{Y}$, by minimizing the reconstruction error $\mathcal{L}_{\text{VAE}}$ defined in Equation 3.2, which can be considered as a way to estimate the quality of learned representations.

$$\begin{aligned} \mathcal{L}_{\text{VAE}}(\mathcal{Y}, \hat{\mathcal{Y}}) &= \beta \cdot \mathcal{L}_{\text{KL}}(\mathcal{Y}, \hat{\mathcal{Y}}) + \mathcal{L}_{\text{MSE}}(\mathcal{Y}, \hat{\mathcal{Y}}) \,, \\ \mathcal{L}_{\text{KL}}(\mathcal{Y}, \hat{\mathcal{Y}}) &= \frac{1}{2} \sum_{i=1}^{|\mathcal{Y}|} (\exp(\sigma_i) - (1 + \sigma_i) + \mu_i^2) \,, \\ \mathcal{L}_{\text{MSE}}(\mathcal{Y}, \hat{\mathcal{Y}}) &= \sum_{y \in \mathcal{Y}, \hat{y} \in \hat{\mathcal{Y}}} (y - \hat{y})^2 \,, \end{aligned} \tag{3.2}$$

where a weighting factor β is included to control the trade-off between a regularization term and a reconstruction error, μ denotes the mean, and σ denotes the variance latent layers in a VAE model. In DoE2Vec, the regularization term is expressed using

the Kullback–Leibler (KL) divergence $\mathcal{L}_{\text{KL}}$, while the Mean Squared Error (MSE) $\mathcal{L}_{\text{MSE}}$ is used for the reconstruction error. Moreover, the VAEs employed in DoE2Vec have an architecture of seven fully connected layers in total, namely:

- The encoder is composed of four fully connected layers, starting with the input layer size d_y, which is equal to the DoE sample size n. This is followed by two hidden layers with a size of $n/2$ and $n/4$ respectively. For the mean and log variance of the latent space, the latent size ls ($ls < n/4$) is assigned; and
- The decoder is composed of three fully connected layers, having a size of $n/4$, $n/2$, and n for the final output layer.

Furthermore, the Rectified Linear Unit (ReLU) activation function is assigned to the hidden layers, while the sigmoid activation function is used for the final output layer in decoder.

Using a grid search approach, the impact of two parameters on the model performance is briefly analyzed, namely the latent size ls and the KL loss weight β, as shown in Figure 3.3. Based on our analysis, a combination of either 24 or 32 latent size (expressed as `VAE-24` or `VAE-32`) and β of 0.001 seems to be a good compromise for minimizing the loss functions. Subsequently, these parameters are considered in our following investigations. While a fine-tuning of the model parameters could potentially further improve the model performance, this is not considered here, since we are not focusing on reconstructing the optimization landscape. Instead, we consider the reconstruction as a mean to evaluate the quality of learned representations.

The complete specification of different VAEs is available in [138] and the pre-trained models are available in our repository [137].

Workflow of DoE2Vec

Similarly to the ELA-based approach illustrated in Figure 3.1, the workflow of DoE2Vec is visualized in Figure 3.4. Instead of the BBOB functions, the function generator from Section 4.1 is integrated in DoE2Vec to generate a diverse set of test functions in terms of optimization landscape. Precisely, test functions belonging to different optimization problem classes can be randomly generated using this function generator, which we call *random* functions or Randomly Generated Function (RGF). A detailed explanation of RGFs is provided in Section 4.1.

In the first step, the DoE samples $\mathcal{X}$ of a BBO problem is taken as input, which is re-scaled to $[-5, 5]^d$. Using the random function generator, a preferably large set of RGFs

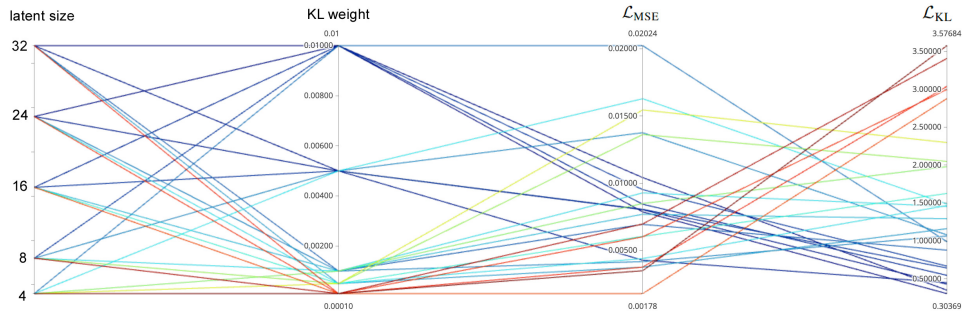


Figure 3.3: Parallel coordinates plot of different latent sizes ls and KL loss weights β in relation to the regularization term $\mathcal{L}_{\text{KL}}$ and reconstruction error $\mathcal{L}_{\text{MSE}}$. Each color or line represents a combination of model parameters. Generally, conflict between both loss terms can be observed, where a parameter combination with a lower $\mathcal{L}_{\text{MSE}}$ tends to have a higher $\mathcal{L}_{\text{KL}}$, and vice versa. *From left to right:* Latent size, KL loss weight, $\mathcal{L}_{\text{MSE}}$, and $\mathcal{L}_{\text{KL}}$.

can be generated and evaluated using the same DoE samples. Next, the corresponding objective values $\mathcal{Y}$ are rescaled to $[0, 1]$, which are then provided for the training of VAEs. Subsequently, the trained VAEs can be employed to compute the latent space representations of the BBO problem. Using a distance-based metric like Euclidean distance, a set of RGFs similar to the BBO problem in terms of latent representations can be identified. Since the latent representations can be readily utilized, a feature processing step is not strictly necessary here, such as normalizing or eliminating highly correlated features.

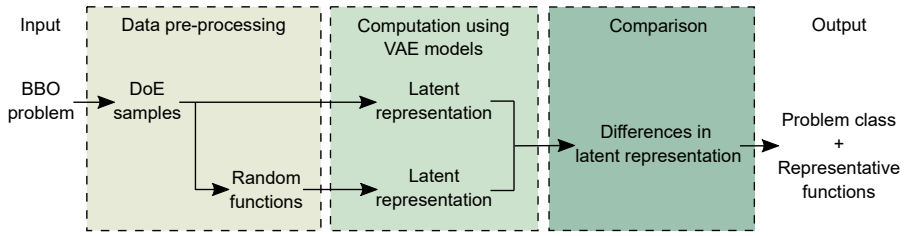


Figure 3.4: Overview of the workflow of DoE2Vec. Basically, VAEs are employed to capture the latent representations of BBO problems, which are trained using a preferably large set of RGFs covering a variety of optimization problem classes.

Performance Assessment of DoE2Vec

To evaluate the performance of DoE2Vec in capturing representative latent representations of BBO problems, two investigations are performed, namely (i) visual inspection of the optimization landscapes and (ii) classification of the BBOB functions. In general, our investigations have the following setup, namely a DoE of 256 samples generated using Sobol' sampling and a set of 250 000 RGFs for the model training.

i. Visual comparison of optimization landscape: Firstly, we evaluate the potential of DoE2Vec in identifying RGFs that have optimization landscape characteristics similar to BBO problems. Since an analytical analysis can be challenging, our investigation instead focuses on visual inspection of the optimization landscape based on 24 BBOB functions (of the first instance) in 2-d. Precisely, the respective RGFs having the most similar latent space representations in terms of the Euclidean distance are identified for each BBOB function. As visually compared in Figure 3.5, a well-fitting RGF can be identified for most BBOB functions, even for highly nonlinear functions like F22. Nevertheless, a clear visual difference in the optimization landscape between some BBOB functions and RGFs can be observed, such as F16 and F24. This indicates that the representative landscape characteristics of such complex functions are insufficiently captured using our DoE2Vec approach.

ii. Classification of the BBOB functions: Secondly, another attractive application of DoE2Vec is to classify optimization problems according to their high-level function properties, such as multi-modality, global structure, and funnel structure [119]. Since these high-level properties often determine the complexity of optimization problems, an appropriate capturing of these properties are crucial for algorithm configuration fine-tuning purposes. For the multiclass classification tasks, a standard RF model with 100 trees from `sklearn` [94] is employed. Using the following setup, we compare the effectiveness of ELA features and/or latent representations computed using different AEs or VAEs for the multiclass classification tasks:

- Standard AE using the same architecture as VAE models, except that the latent space is now a single dense layer. Subsequently, the performance of two AEs and two VAEs is evaluated, consisting of `AE-24`, `AE-32`, `VAE-24`, and `VAE-32`;

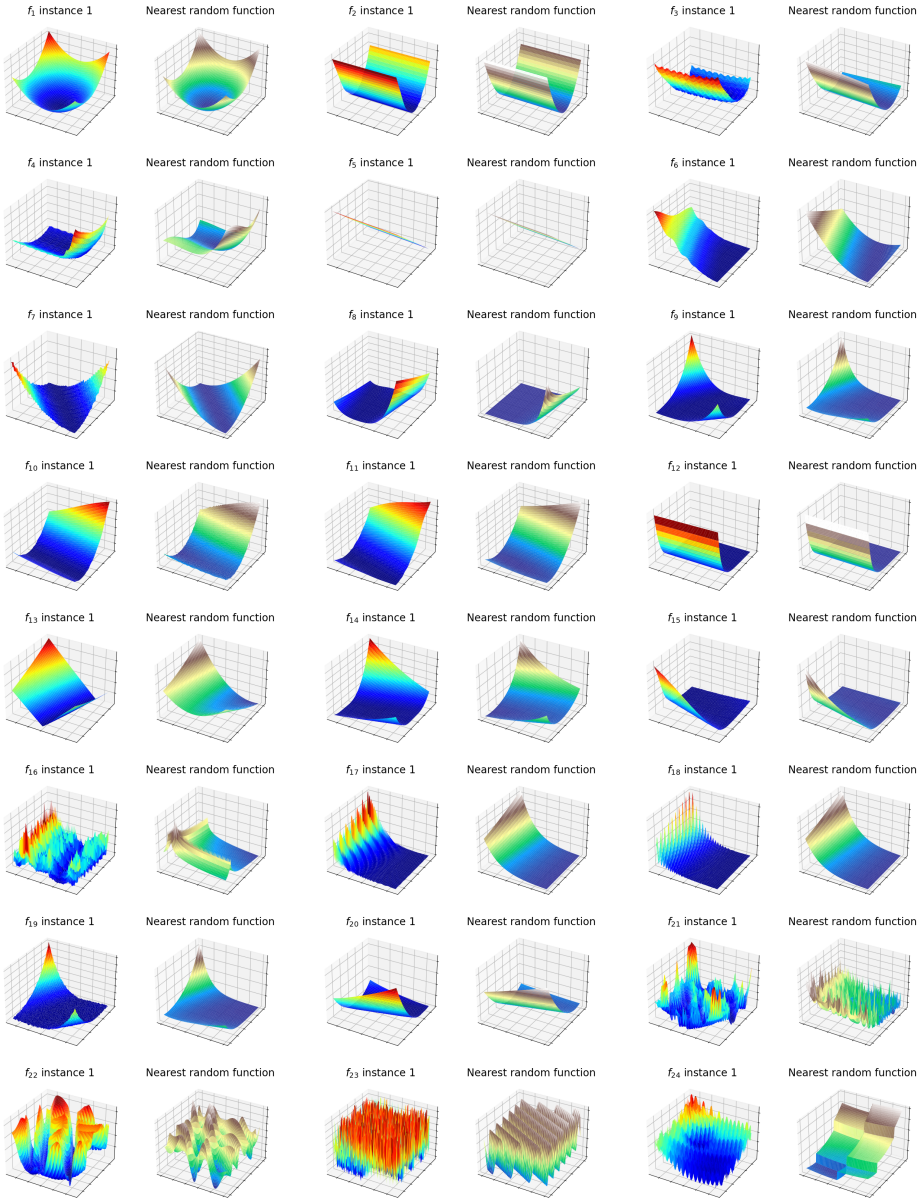


Figure 3.5: Pairs of 24 BBOB functions (*left side*) and their respective nearest RGF (*right side*) identified based on some latent space representations computed using `VAE-24`. Figure taken from [139].

- The classical ELA features, which are specifically designed for such function classification tasks; and
- A combination of ELA features and latent representations captured using `VAE-32` for investigating the complementary effect of different feature sets.

The classification results based on macro F1 scores using different approaches are summarized in Table 3.2. Generally, a good performance can be achieved by DoE2Vec using both AE and VAE, particularly in low problem dimensionality. On the other hand, the ELA approach constantly outperforms DoE2Vec, especially in classifying the global structure and multimodal landscapes, which however comes as no surprise. Fascinatingly, the classification performance can be further improved using a combination of ELA and DoE2Vec, revealing the complementary effect of both feature sets. In other words, the latent representations learned using DoE2Vec can be employed next to the ELA features, for a better handling of classification tasks.

Table 3.2: Classification results based on macro F1 scores averaged across 10 repetitions using a standard RF model. The RF model is first trained using the feature representations (AE, VAE, ELA, and ELA combined with `VAE-32`) of the first 100 instances for each BBOB function, and then validated using instance 101 to 120. *The results using PCA, reduced Multiple Channel (rMC), and Transformer are directly taken from previous work in [119], using an identical experimental setup, but without repetition. Table taken from [139].

d	Task	AE-24	AE-32	VAE-24	VAE-32	ELA	PCA*	rMC*	Transformer*	ELA-VAE-32
2	multimodal	0.875	0.849	0.877	0.856	0.984	0.994	0.971	0.991	**0.991**
	global struct.	0.903	0.904	0.902	0.889	0.983	0.992	0.965	0.991	**0.998**
	funnel	0.985	0.974	0.956	0.978	**1.000**	0.999	0.995	1.000	**1.000**
5	multimodal	0.908	0.903	0.880	0.889	0.963	0.897	0.947	0.991	**0.998**
	global struct.	0.838	0.828	0.810	0.793	**1.000**	0.807	0.859	0.978	**1.000**
	funnel	**1.000**	**1.000**	0.996	0.991	**1.000**	0.990	0.989	1.000	**1.000**
10	multimodal	0.877	0.813	0.844	0.838	**1.000**	0.839	0.952	0.974	**1.000**
	global struct.	0.794	0.737	0.783	0.745	0.902	0.774	0.911	0.963	**0.991**
	funnel	0.998	0.993	0.997	0.993	0.972	0.977	0.991	**1.000**	0.997
20	multimodal	0.726	0.722	0.700	0.694	0.970	-	-	-	**0.991**
	global struct.	0.689	0.621	0.606	0.626	0.972	-	-	-	**0.997**
	funnel	0.993	0.982	0.985	0.982	**1.000**	-	-	-	**1.000**

Limitations of DoE2Vec

Based on our investigations, DoE2Vec has shown motivating potential in capturing representative latent representations of BBO problems, which could be potentially

used to identify representative functions for further usages. The complete result discussions are available in [139]. Instead of replacing the classical ELA features, we propose to extend the ELA feature set with the latent space representations computed using DoE2Vec. Meanwhile, the following limitations of DoE2Vec have been identified, namely:

1. The latent space representations computed using VAEs are a black-box that are extremely difficult to be interpreted;

2. DoE2Vec is scale-invariant, but not rotation- or translation-invariant, which could be improved by using other loss functions; and

3. In cases where no pre-trained VAE is available, e.g., when using a custom DoE, the model must be trained from scratch. Depending on the DoE sample size and number of RGFs, the training of VAEs could be relatively time-consuming compared to the ELA feature computation.

Following this, substantial work is necessary to significantly improve the performance and reliability of DoE2Vec. Thus, this topic is not pursued any further in this thesis. Instead, we focus on the ELA-based approach proposed in Section 3.1.1 to capture the optimization landscape characteristics of real-world expensive BBO problems.

3.2 Analyzing the Function Properties of BBOB Instances

To identify appropriate representative functions for real-world expensive BBO problems, their optimization landscape characteristics are compared against those of some test functions, as proposed in Section 3.1.1. In this context, the BBOB functions could potentially serve as the best candidate for this purpose, and thus, an in-depth understanding is necessary. Commonly in previous work, only the first few BBOB instances are considered for investigations. Nonetheless, this could significantly impact the investigation outcomes, especially if these instances are not representative of the overall space of instances. Following this, we analyze a large set of BBOB instances to better understand the representativeness of different BBOB instances w.r.t. their underlying function properties. Precisely, we investigate the properties of BBOB instances in terms of (i) ELA features in Section 3.2.1, (ii) the performance of optimization algorithms in Section 3.2.2, and (iii) the global function properties in Section 3.2.3.

Here, we focus on the first 500 instances of the BBOB functions in 5-d and 20-d. Furthermore, a confidence level of 99% is considered in all statistical comparisons, i.e., a null hypothesis is rejected, if the p-value is lower than 0.01. Full descriptions of the experimental setup and results are available in [76, 77].

3.2.1 Landscape Characteristics based on ELA Features

Our investigation begins with an analysis of the optimization landscape characteristics in terms ELA features. Using the ELA-based approach proposed in Section 3.1.1, a final set of 65 ELA features is computed, where some of the PCA features that only concern the DoE samples $\mathcal{X}$ are additionally included for a comprehensive analysis. To obtain the ELA feature distributions, a total of 100 DoEs having 1 000 samples each are generated using LHS for each BBOB instance, where the DoEs are identical across instances.

A. Distributions of ELA features: In the first step, the distributions of ELA features between different BBOB instances are compared using the pairwise two-sample Kolmogorov-Smirnov (KS) test [53], using the null hypothesis *the ELA distribution is similar between both problem instances*. This ends up with altogether $\frac{500 \cdot 499}{2} = 124\,750$ comparison pairs for each ELA feature. To mitigate the effect of multiple comparisons, the Benjamini-Hochberg (BH) correction method [11] is additionally applied. For a convenient interpretation, the results are aggregated across all BBOB instances. Figure 3.6 shows the average rejection rate of the null hypothesis, i.e., the fraction of tests that rejects the null hypothesis.

In 5-d, some ELA features clearly differ between the BBOB instances, particularly the `ela_meta.lin_model.intercept`. Nevertheless, this does not necessarily indicate that all instances should be considered different, since some ELA features are not invariant to scaling of the objective function, such as the linear model intercept [152]. On the other hand, barely any test rejections can be observed in some features, such as the PCA features. This is because they are primarily computed using the DoE samples $\mathcal{X}$, which are identical across instances. While the objective values $\mathcal{Y}$ can principally influence some of the PCA features, their impact is relatively marginal.

On a per-function basis, some commonalities in many of the ELA features can be noticed. Interestingly, no difference between instances for almost all ELA

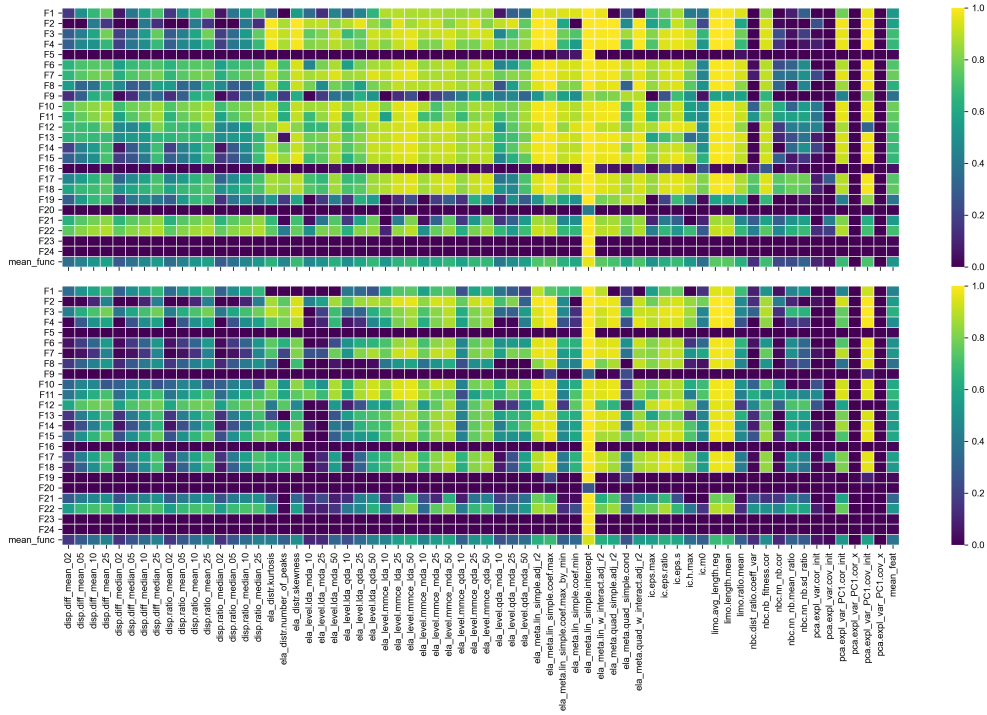


Figure 3.6: Average rejection rate of the null hypothesis *the distribution of ELA feature between instances is similar*, aggregated over 500 BBOB problem instances in 5-d (*top*) and 20-d (*bottom*). A lighter color represents a higher rejection rate. An extra row (*bottom*) is included for the average across all BBOB functions and an extra column (*right*) for the average across all ELA features. Figure taken from [76].

features can be observed in functions like F5, F16, F23, and F24. It is worthwhile to point out that even for a simple function like F1, many features differ between instances. Since translation is the only transformation applied to F1 [44], which uniformly at random moves the global optimum within $[-4, 4]^d$, the high-level landscape properties must be well preserved. While this is true as long as F1 is treated as an unconstrained problem, this is not the case in ELA, where samples are drawn within a bounded domain for feature computation. Following this, translating the function can significantly impact the low-level landscape features, which might explain the huge differences in ELA features across F1 instances.

Overall, a similar pattern can be observed in 20-d BBOB functions with a reduced magnitude. Moreover, functions like F9, F19, and F20 now barely show any statistical difference between instances.

B. Dimensionality reduction: Based on the standardized ELA features, i.e., by removing the mean and scaling to unit variance, the distributions of BBOB instances are projected to the 2-d ELA feature space using the t-distributed Stochastic Neighbor Embedding (t-SNE) approach [136], as shown in Figure 3.7. While it is clear that most instances of the same BBOB functions are clustered together, several BBOB instances are spread throughout the feature space, indicating that these outlying instances might be less similar compared to those clustering one. This is particularly noticeable in 5-d, where several functions are somewhat spread across the projected space. On the other hand, the BBOB function clusters seem to be much more stable in 20-d, matching our interpretations earlier in Figure 3.6. In fact, the differences between BBOB functions can be indeed easier detected in higher dimensionality based on ELA features, as shown in [111], which matches the more well-defined problem clusters in Figure 3.7.

C. Representativeness of BBOB instances 1–5: Based on the statistical analysis performed in Figure 3.6, we evaluate the representativeness of the first five BBOB instances that are commonly considered in literature in terms of ELA features. Instead of aggregating the rejections on a per-feature level, we do it now per-function. An instance can be considered as an outlier, and thus, non-representative, if the fraction of test rejections against other instances is high, while the overall fraction of pairwise test rejections is low. The average fraction of rejections across ELA features is visualized in Figure 3.8, where the rejection rates of the first five instances are highlighted. Generally, there is no obvious

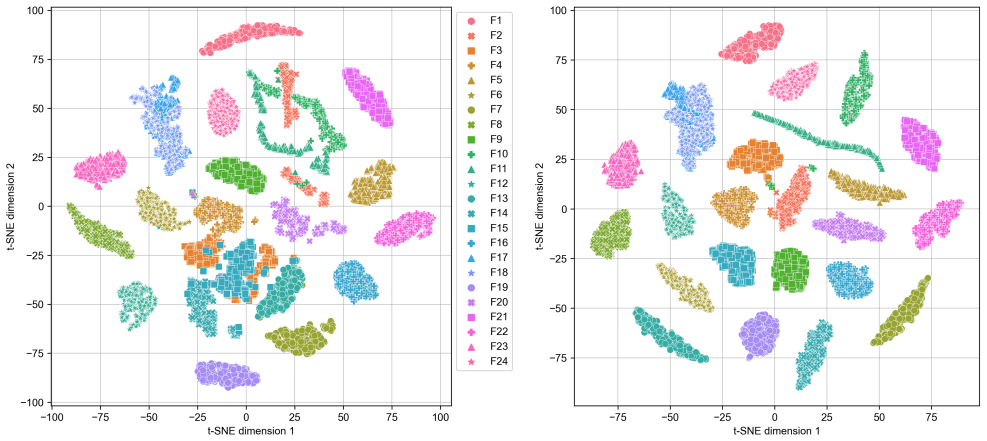


Figure 3.7: Projection of the high-dimensional ELA feature space (altogether 64 features, without `ela_meta.lin_simple.intercept`) to a 2-*d* visualization for the BBOB functions in 5-*d* (*left*) and 20-*d* (*right*) using the t-SNE approach. Each dot represents a BBOB instance and each color represents a BBOB function. Figure taken from [76].

case, where the five instances are all outliers. While some instances might have a slightly different rejection rate than the remaining instances, we could not conclude that the choice of selecting these five would be any better or worse than other instances of the same function.

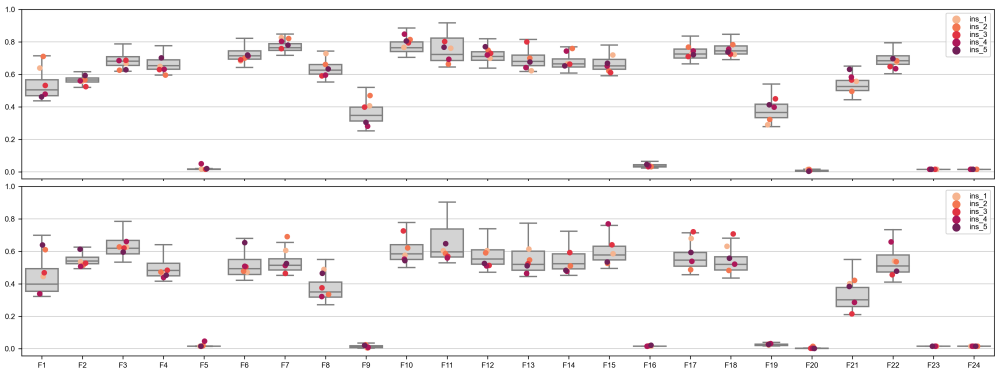


Figure 3.8: Average fraction of rejections of pairwise tests between one instance and each of the remaining ones, aggregated over all ELA features from Figure 3.6. The first five instances of each function are highlighted, while the boxplots (*gray*) show the distribution for all 500 instances in 5-*d* (*top*) and 20-*d* (*bottom*). Figure taken from [76].

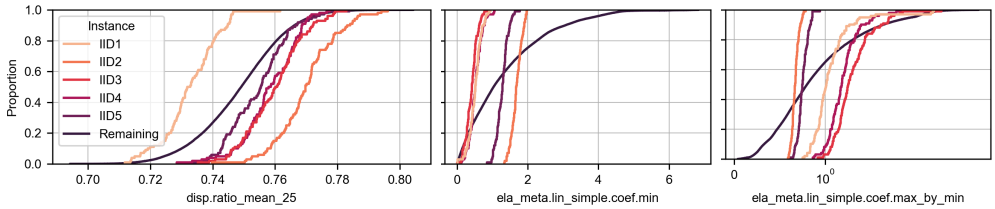


Figure 3.9: ECDF curves of normally (*left*) and non-normally distributed (*middle* and *right*) ELA features for F1 in 5-d, showing instances 1–5 and all remaining instances. Figure taken from [76].

Apart from considering the overall representativeness based on the aggregation of all ELA features, we also analyze each feature in more detail. Using three ELA features of F1 in 5-d as an example, namely `disp.ratio_mean_25`, `ela_meta.lin_simple.coef.min`, and `ela_meta.lin_simple.coef.max_by_min`, where the pairwise statistical analysis shows a large number of rejections, the Empirical Cumulative Distribution Functions (ECDF) curves in Figure 3.9 show the differences in distribution of instances 1–5 against the remaining instances. Despite the differences between instances can be relatively large, there is no evidence to conclude that the first five instances would be less representative than any other instance set. Interestingly, while some ELA features are seemingly normally distributed, this is not the case for all features. In fact, based on our normality tests for each ELA feature distribution in [77], some features are *non-Gaussian*, such as the distribution and meta-model features.

3.2.2 Algorithm Performance across Instances

Apart from analyzing the ELA features, we also investigate the performance of optimization algorithms across different BBOB instances. In this investigation, we consider eight derivative-free optimization algorithms available in `Nevergrad` [104], namely DiagonalCMA (a variant of CMA-ES), DE, EMNA, NGOpt14, PSO, RS, RCobyla, and SPSA, using a budget of 10 000 function evaluations and 50 repetitions. In this context, the algorithm performances are evaluated based on the best-found solution achieved after 1 000 and 10 000 evaluations on 500 BBOB instances in 5-d. To determine whether there are significant differences in algorithm performance between instances, the Mann-Whitney U (MWU) test with the null hypothesis *the algorithm performances are similar across instances* is employed, along with the BH correction

method. On top of this pairwise testing, we additionally consider a one-vs-all comparison using the same procedure, where the algorithm performance of a selected instance is compared against the remaining 499 instances.

Rather than the absolute objective values, we consider the relative performance measure in our investigation, i.e., precision from the global optimum. Subsequently, it is to be expected that RS is invariant across instances, as observed in Figure 3.10. In general, most of the algorithms have a stable performance on all BBOB functions, which mostly matches the results from Figure 3.6. On the opposite side, SPSA shows differences in performance between instances, particularly for F1, F5, F19, F20, F23, and F24. This indicates that SPSA is not invariant to the transformations implemented for the BBOB instance generation, matching with the observation that SPSA displays clear structural bias in [144].

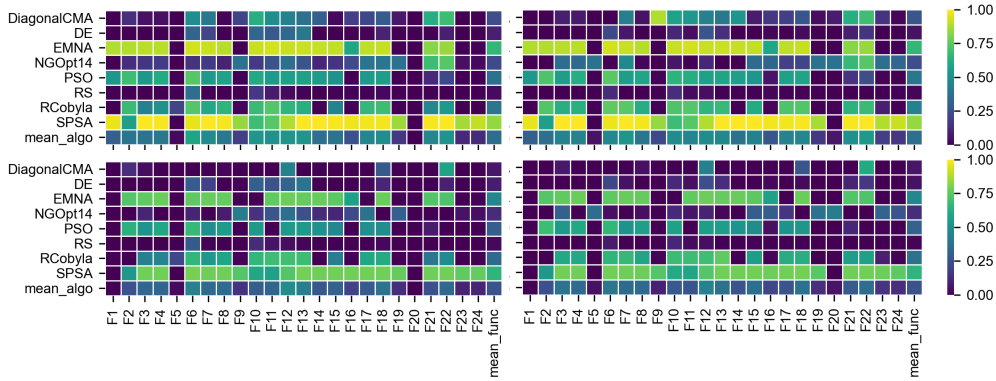


Figure 3.10: Average rejection rate of the null hypothesis *the algorithm performances are similar across instances*, aggregated over BBOB instances per function. The optimization results based on pairwise (*top row*) and one-vs-all (*bottom row*) comparisons are shown, using 1 000 (*left column*) and 10 000 (*right column*) function evaluations. The average values are shown in the last column and last row of each figure. Figure taken from [76].

While some algorithms, specifically DiagonalCMA and DE, are expected to be invariant to the transformations used for the BBOB instance generation, this assumption does not seem to hold for some BBOB functions like F12. This indicates that the instances lead to statistically different performances of these algorithms, which might be explainable considering that these algorithms treat the optimization problems as being box-constrained, while the BBOB function transformations assume that the domain is unconstrained [43]. Furthermore, while the algorithms might be invariant to rotation and transformation, applying these mechanisms can have an effect on the initializa-

tion, which eventually impacts the algorithm performances [143]. This is intended for the BBOB suite, since it is stated that *"If a solver is translation invariant (and hence ignores domain boundaries), this [running on different instances] is equivalent to varying the initial solution"* [43]. While this is true for unconstrained optimization, it is not as straightforward for box-constrained problems like the BBOB functions, since changing the initialization method could significantly influence algorithm behavior.

3.2.3 Global Function Properties

For most BBOB functions, the transformation mechanism generally consists of rotations and translations, which are applied differently to preserve the high-level function properties [44]. Nonetheless, the impact of transformation processes on the low-level features of BBOB functions can not always be as easily interpreted. Consequently, the differences between instances of each function are influenced by the associated transformation procedure, resulting in some functions being much more stable than others.

One aspect of the BBOB instances that is being treated differently is the location of the global optimum. For most of the BBOB functions, the location of the global optimum is uniformly sampled in $[-4, 4]^d$ by construction, where the optimum is moved from the default location $\mathbf{0}^d$ using translation. Nevertheless, a different procedure is used in some BBOB functions, such as F5. In Figure 3.11, the true location of the global optima for the first 500 BBOB instances in 2-d is visualized. While the distribution pattern of the global optimum for most of the BBOB functions is similar to F1, the following exceptions can be noticed:

- The asymmetric pattern for F4 is due to the fact that the even coordinates are being used differently than the odd ones by construction;
- For F8, a scaling transformation is applied before the final translation, resulting in the optimum being confined to a smaller space; and
- For other functions, namely F9, F19, F20, and F24, the problem construction requires a different setup, and thereby, the optima are distributed differently.

3.2.4 Summary

Based on our analysis, statistically significant differences between problem instances in terms of ELA features can be observed in some BBOB functions, which seemingly

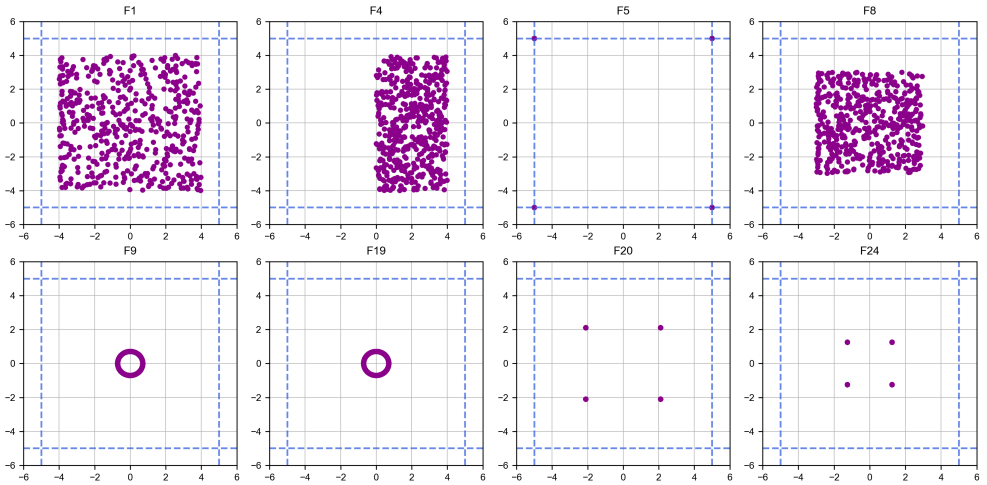


Figure 3.11: Distributions of the global optima for 500 instances of selected BBOB functions in 2-d. The remaining BBOB functions not shown here have a distribution pattern similar to F1, refer to [77]. Each violet dot represents the optimum location of a BBOB instance in the commonly used search space $[-5, 5]^2$, marked using dashed lines. Figure taken from [76].

lessens with increasing dimensionality. Following this, care should be taken when relying on ELA features to represent instances, e.g., for landscape-aware HPO, since the choice of BBOB instances could potentially lead to a different result.

Regarding the performance of optimization algorithms, only RS is close to being fully invariant across different BBOB instances. Meanwhile, differences in performance between instances of the same function can be observed for algorithms like CMA-ES, which are typically considered rotation invariant. This might be related to the impact of instance transformations on the effectiveness of algorithm initialization [143], or it might be due to the fact that we consider the problems to be box-constrained, which seemingly invalidates the assumptions considered in the transformation mechanisms.

Lastly, differences in the distribution pattern of the global optimum between BBOB instances can be observed as well. While the optimum location is confirmed to be uniform at random in $[-4, 4]^d$ for most BBOB functions, this is not always the case, since some problem formulations seemingly require different transformation mechanisms.

3.3 Landscape Characteristics of Automotive Crashworthiness Optimization

To learn the optimization landscape characteristics of automotive crashworthiness optimization, we have collected 20 automotive crashworthiness optimization problems from previous vehicle development projects performed by BMW, a German premium automobile manufacturer. Consisting of different automotive crash scenarios, namely side crash, rear crash, roof crash, and frontal crash, as summarized in Table 3.3, the DoE samples were generated using the Modified Extensible Lattice Sequence (MELS) sampling available in `HyperStudy`, which is a sequential lattice space-filling DoE approach developed based on the Sobol' sequences.

Table 3.3: Summary of 20 automotive crash problem instances analyzed in this thesis, consisting of four different crash scenarios. Table taken from [73].

Problem instance	Scenario	Design variables	Sample size
Crash_1	Side crash	22	59
Crash_2	Side crash	22	309
Crash_3	Side crash	22	309
Crash_4	Side crash	16	150
Crash_5	Side crash	16	102
Crash_6	Side crash	16	132
Crash_7	Side crash	20	329
Crash_8	Side crash	20	330
Crash_9	Side crash	20	333
Crash_10	Side crash	18	530
Crash_11	Side crash	13	150
Crash_12	Side crash	14	99
Crash_13	Rear crash	12	180
Crash_14	Rear crash	12	259
Crash_15	Roof crash	19	100
Crash_16	Roof crash	17	107
Crash_17	Frontal crash	22	487
Crash_18	Frontal crash	20	246
Crash_19	Frontal crash	8	248
Crash_20	Frontal crash	8	246

Due to the expensive FE simulation runs, as is commonly in automotive crash problems, some of the DoE sample sizes might be too small in relation to the high dimensionality for a reliable ELA feature computation, particularly in `Crash_1`. However, adding more DoE samples is ruled out here, since we have no access to these FE simulations. For problem instances from the same crash scenario, they were mainly different in terms of vehicle models and load cases, e.g., different pole positions for

side crashes. Furthermore, the design variables were the thicknesses of different vehicle components to-be-optimized, e.g., the thicknesses of rocker panels in side crash scenarios. During these development projects, the quality of a vehicle design was evaluated using the following five objectives, which were measured and quantified as scalar FE outputs:

1. Mass (M): Weight of components;
2. Maximum force (F_{max}): Maximum impact force during crash;
3. Intrusion ($Intr$): Magnitude of inward structural deformation;
4. Energy absorption (EA): The amount of kinetic energy absorbed during crash; and
5. Rotation (Rot): Rotational deformation of components during crash. This metric was introduced to measure the average vertical deformation of FE nodes between inner and outer side of components.

Principally, the mass objective provides information about the vehicle weight, fuel consumption, and manufacturing costs, while the remaining objectives are about the structural performance of a vehicle design. Throughout this thesis, we also refer to these objectives as *crash functions*. Nonetheless, depending on the purpose of each vehicle development project, not all five objectives were always considered, and therefore, not all of them were available for our study. In fact, out of the maximum of $20 \times 5 = 100$ potentially available ones, only a total of 48 crash functions were available.

3.3.1 Optimization Problem Classes

Using our approach proposed in Section 3.1.1, the ELA features of automotive crash problems are computed and compared against those of the BBOB functions. The differences in ELA features are visualized using agglomerative hierarchical clustering approach [91], where problems are clustered together in a bottom-up fashion. Precisely, this approach starts with each problem as a cluster and progressively merges clusters until one large cluster is left, consisting of all problems. Correspondingly, we consider the Ward's method [154] as a linkage criterion for the cluster merging strategy, by minimizing the within-cluster variance. In other words, clusters are selected for merging based on the smallest increase in the within-cluster sum of squared error

after merging, which is proportional to the Euclidean distance. In such way, the problem class of automotive crash problems can be easily identified based on the BBOB functions within the same cluster.

The clustering results are visualized in Figure 3.12, using the average pairwise distance between different BBOB functions as our reference in determining to which extent a crash function is similar to its neighboring BBOB functions. Generally, we observe that many of the crash functions are rather separated from the BBOB functions, especially the intrusion function in `Crash_8` and `Crash_9`, and the maximum force function in `Crash_16`. The extremely large Euclidean distance of the intrusion function in `Crash_8` is primarily due to the standardized `pca.expl_var_PC1.cov_init` feature, indicating that our standardization approach for ELA features using the mean and variance from BBOB functions might not be appropriate for all crash functions. The fact that some crash functions, such as the rotation function in `Crash_6`, are clustered in the same group as F16 and F23 suggests that they could have similar landscape characteristics, e.g., highly rugged and repetitive landscape. As expected, all mass functions are clustered in the same group as F5, since mass is linearly dependent on the thicknesses of vehicle components. Based on the clustering patterns, we observe that many automotive crash problems are indeed different from the BBOB functions in terms of landscape characteristics, revealing that these automotive crash problems belong to problem classes other than those covered by the BBOB suite. Consequently, the BBOB functions might be insufficiently representative for these automotive crash problems.

For a better understanding of the clustering results, each of the ELA features is closely examined, using `Crash_2` as a representative example due to its high dimensionality and sufficiently large DoE sample size. As visualized in Figure 3.13, several ELA features show remarkable differences in feature values between the crash functions and BBOB functions, e.g., `ela_distr.kurtosis` and `ela_level.mmce_qda_10`. For a fair evaluation, the distribution of ELA features between the crash functions and BBOB functions are compared using the two-sample KS test with the null hypothesis *the distribution of ELA features between them is similar*. Since the null hypothesis is rejected for some ELA features with a confidence level of 95%, the distribution of these ELA features is indeed different between the crash functions and BBOB functions, which might explain the separation in the clustering patterns observed previously.

For a convenient interpretation, the statistical comparison of ELA feature distributions between the crash functions and BBOB functions for all 20 automotive crash problems is summarized in Figure 3.14. While no obvious ELA feature has a com-

3.3 Landscape Characteristics of Automotive Crashworthiness Optimization

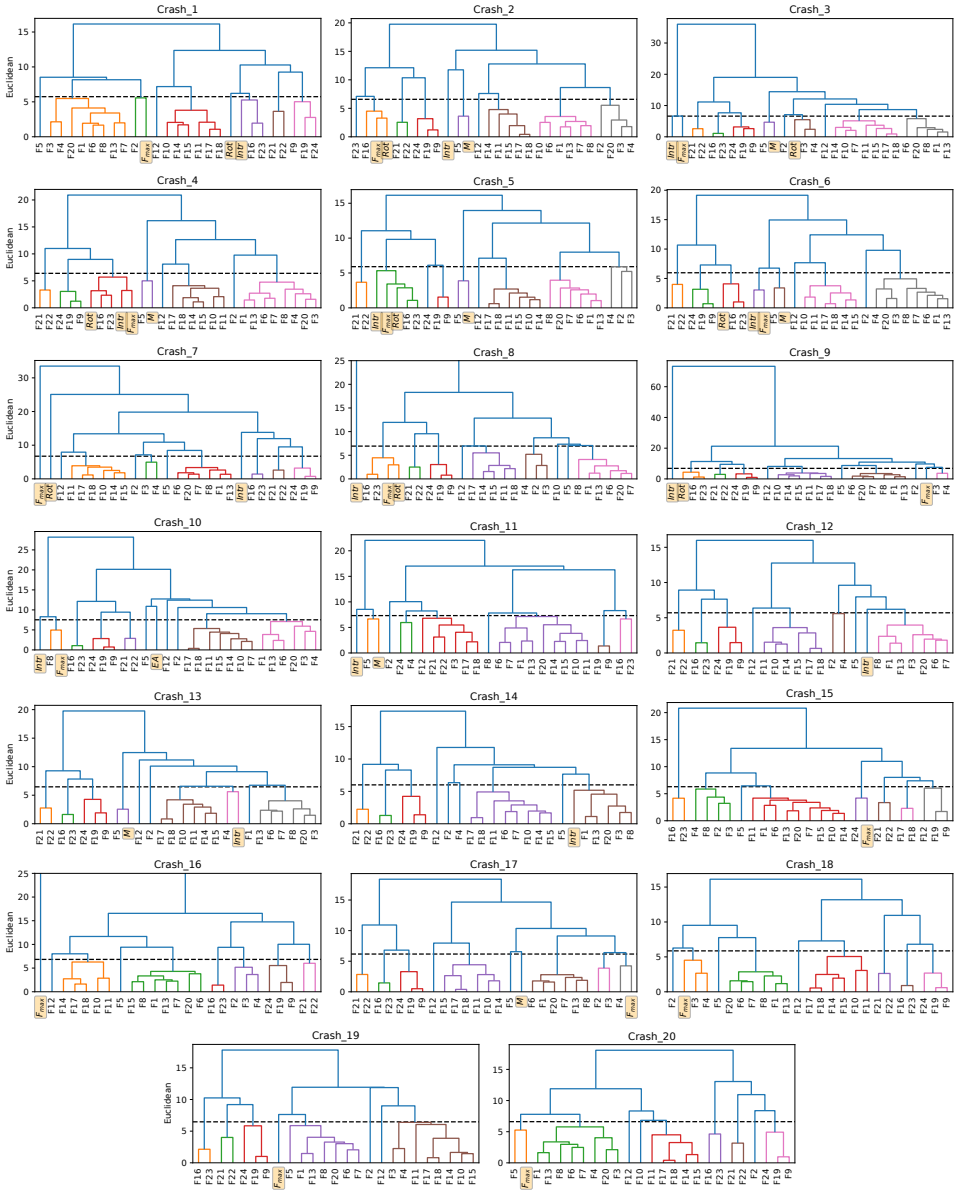


Figure 3.12: Clustering patterns of the crash functions and 24 BBOB functions for all 20 automotive crash problems, where the labels of crash functions are highlighted in orange color. The reference Euclidean distance is marked with a dashed line, and clusters below it are assigned with different colors. The intrusion function in `Crash_8` has a Euclidean distance of around 700 to the main cluster, and the maximum force function in `Crash_16` has a distance of around 300, which are cut-off due to visualization purposes. Figure taken from [73].

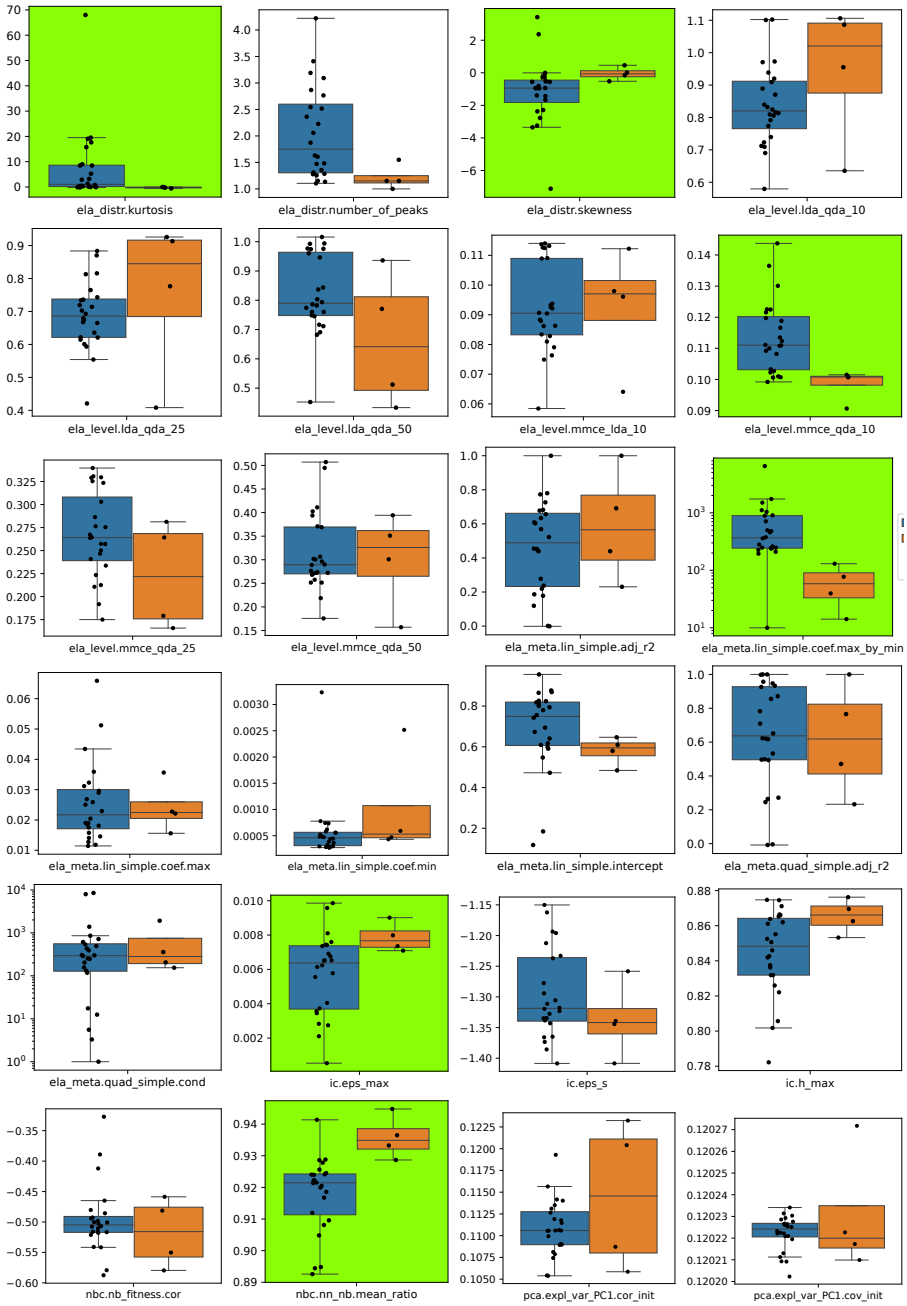


Figure 3.13: Distribution of ELA features in raw feature values considered for the clustering of 24 BBOB functions (*blue color*) and four crash functions (*orange color*) in `Crash_2`. An ELA feature is highlighted with green color, if the null hypothesis *the distribution of ELA features between the BBOB and crash functions is similar* is rejected based on the KS test with a confidence level of 95%. Figure taken from [73].

pletely different distribution between the crash functions and BBOB functions in all cases, the null hypothesis is rejected in many of the dispersion features. Subsequently, the crash functions and BBOB functions might have a different degree of dispersion, which quantifies the size of search space region with better solutions.

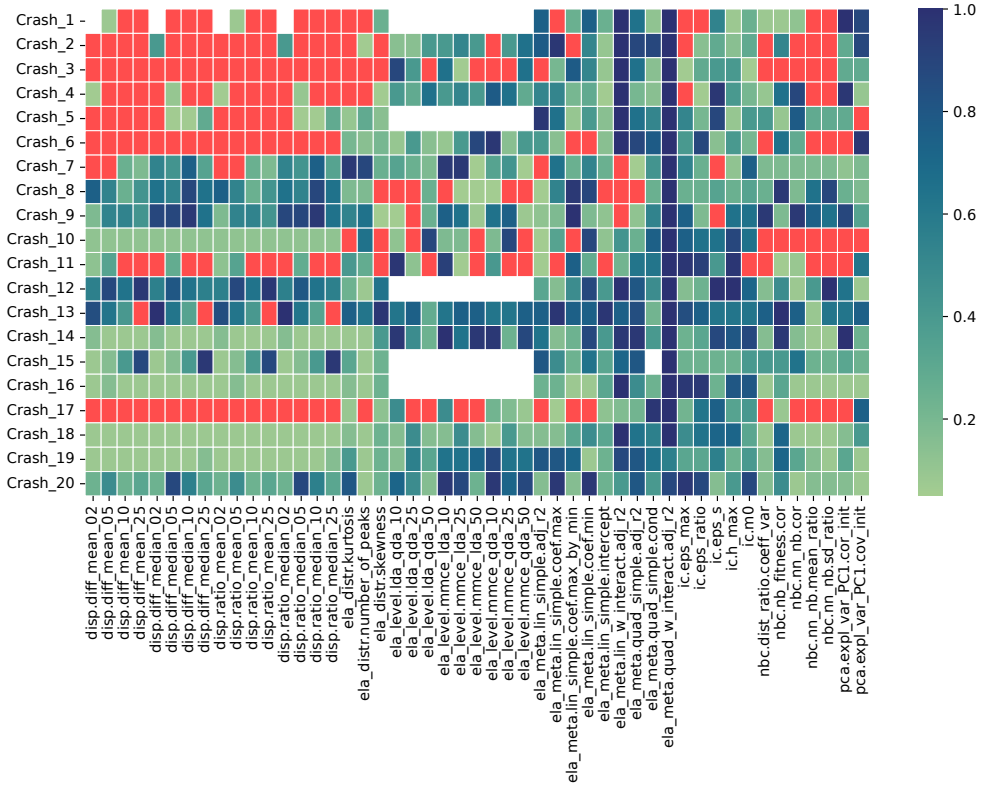


Figure 3.14: Comparison of all 49 ELA feature distributions between the crash and BBOB functions for all 20 automotive crash problems, based on the p-value computed using the KS test. A larger p-value (darker color) indicates that the null hypothesis *the distribution of ELA features between the crash and BBOB functions is similar* is less likely to be rejected, while a smaller p-value (lighter color) for a higher chance of rejection. A red color indicates that the null hypothesis is rejected with a confidence level of 95%, i.e., p-value less than 0.05, while a white color indicates that an ELA feature computation is skipped, e.g., due to a small sample size. Figure taken from [73].

Attempting to verify our hypothesis *all automotive crash problems belong to the same optimization problem class*, where a general one-for-all representative function and/or optimizer for automotive crashworthiness optimization is possible, we further

analyze the problem class of different crash problems. Using the t-SNE approach, the high-dimensional ELA feature space is projected to a 2-d visualization for all crash functions and BBOB functions, as shown in Figure 3.15. In line with our observations earlier, many of the crash functions are separated from the BBOB functions, indicating that these functions belong to different problem classes. Rather than forming one cluster, the crash functions are spread across the ELA feature space, forming their own problem classes, even for crash functions of the same type, e.g., intrusion. As opposed to our hypothesis, this insight suggests that the problem classes of automotive crashworthiness optimization could be different, depending on the problem definition. Consequently, each crash problem must be treated separately, such as the identification of appropriate representative functions and fine-tuning of optimization algorithms. In other words, an *instance*-based fine-tuning of optimization configurations approach is necessary, due to the diverse optimization problem classes across crash problems. Nonetheless, we are fully aware that this experimental setup might be inadequate to reach a decisive conclusion and an in-depth investigation is necessary, since the crash DoEs investigated have different sample sizes and dimensionalities, which could impact the ELA feature computation.

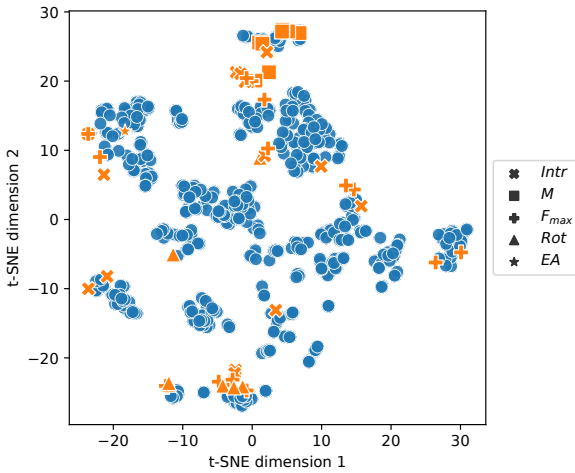


Figure 3.15: Projection of the ELA feature space to a 2d visualization for all crash functions (altogether 48 functions; *orange*) and BBOB functions (altogether $24 \times 20 = 480$ functions; *blue*) using the t-SNE approach. Figure taken from [73].

3.3.2 Comparison against Vehicle Dynamics Problems

Beyond that, we investigate the diversity of optimization problem classes across problem instances from different engineering domains. Precisely, the landscape characteristics of automotive crash problems are compared against those of vehicle dynamics problems, another common BBO problems in the automotive industry. In brief, the vehicle control systems like Anti-lock Braking System (ABS) must be optimally calibrated for maximum vehicle safety and ideal driving experience [24, 132].

Altogether five ABS problems using different tires and vehicle loads are considered, each consisting of two design variables and 10 101 DoE samples. Since such a large set of DoE samples is not available and/or infeasible for crash problems, only crash problems having a DoE sample size of at least $10 \cdot d$ in Table 3.3 are considered for comparison. Subsequently, we end up with a final set of 13 crash problems, consisting of seven side crashes, two rear crashes, and four frontal crashes. For the ELA feature computation, the approach proposed in [131] is employed, which slightly differs from our approach proposed in Section 3.1.1 in the following aspects:

- Instead of standardization, the computed ELA features are rescaled to a similar scale range using min-max scaling; and
- To discard highly correlated ELA features, a PCA approach is applied to the high-dimensional ELA feature vectors.

For a fair comparison of the optimization landscape characteristics, both steps are first performed on the BBOB functions and the identical scaling and transformation are then applied to the engineering problem instances.

As visualized in Figure 3.16, the high-dimensional ELA feature vectors are projected to a 2-d space using the first two PCA components. Notably, a clear separation between the crash and ABS problem instances can be observed, indicating that both domains are indeed different in terms of ELA features. Similar to our observations in Figure 3.15, the problem instances from the same engineering domain are scattered across the ELA feature space as well. Interestingly, the diversity within the same domain can be as large as across different domains.

To gain a deeper understanding about the separation between these two engineering domains, the corresponding ELA features are further analyzed, as shown in Figure 3.17. Here, the level set features are omitted, because the feature computation fails in many of the crash problem instances. Based on visual inspection, many of the ELA features show clear differences between the two engineering domains. For a fair

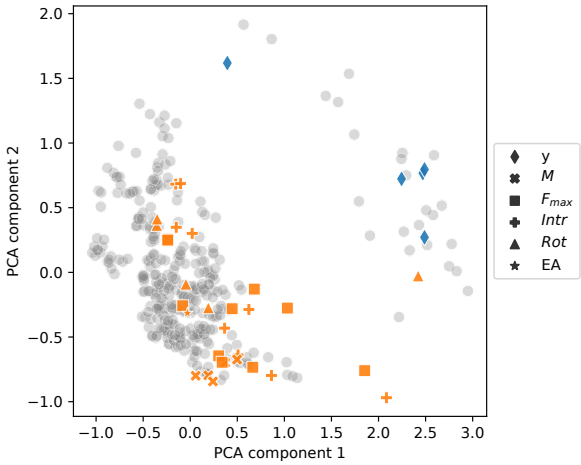


Figure 3.16: Visualization of the ELA feature space for the five ABS functions (*blue*), 30 crash functions (*orange*), and 336 BBOB functions (altogether 24 BBOB × 14 problem instances; *gray*) using the first two PCA components. Each dot represents a problem instance. Figure taken from [24].

evaluation, the distribution of ELA features between them is statistically analyzed using the two-sample KS test with the null hypothesis *the distribution of ELA features is similar*. In fact, many of the ELA features indeed have a different distribution between the crash functions and ABS problem instances, in line with our previous observations. Consequently, our results show that both engineering domains belong to different optimization problem classes. For a full result discussion, we refer to [24].

3.4 Conclusions

In this chapter, we focus on analyzing the optimization landscape characteristics of real-world expensive BBO problems, using automotive crashworthiness optimization as a representative example. Firstly, two approaches for capturing the optimization landscape characteristics are proposed in Section 3.1. On one hand, the optimization landscape characteristics of BBO problems are numerically quantified using the classical ELA features. On the other hand, the DoE2Vec approach is introduced, where the BBO problems are characterized based on some low-dimensional latent space representations computed using deep NN models like VAE. Next, a large set of BBOB instances is extensively analyzed in Section 3.2 from the perspectives of ELA features, optimization performances, and global function properties. Lastly, the optimization

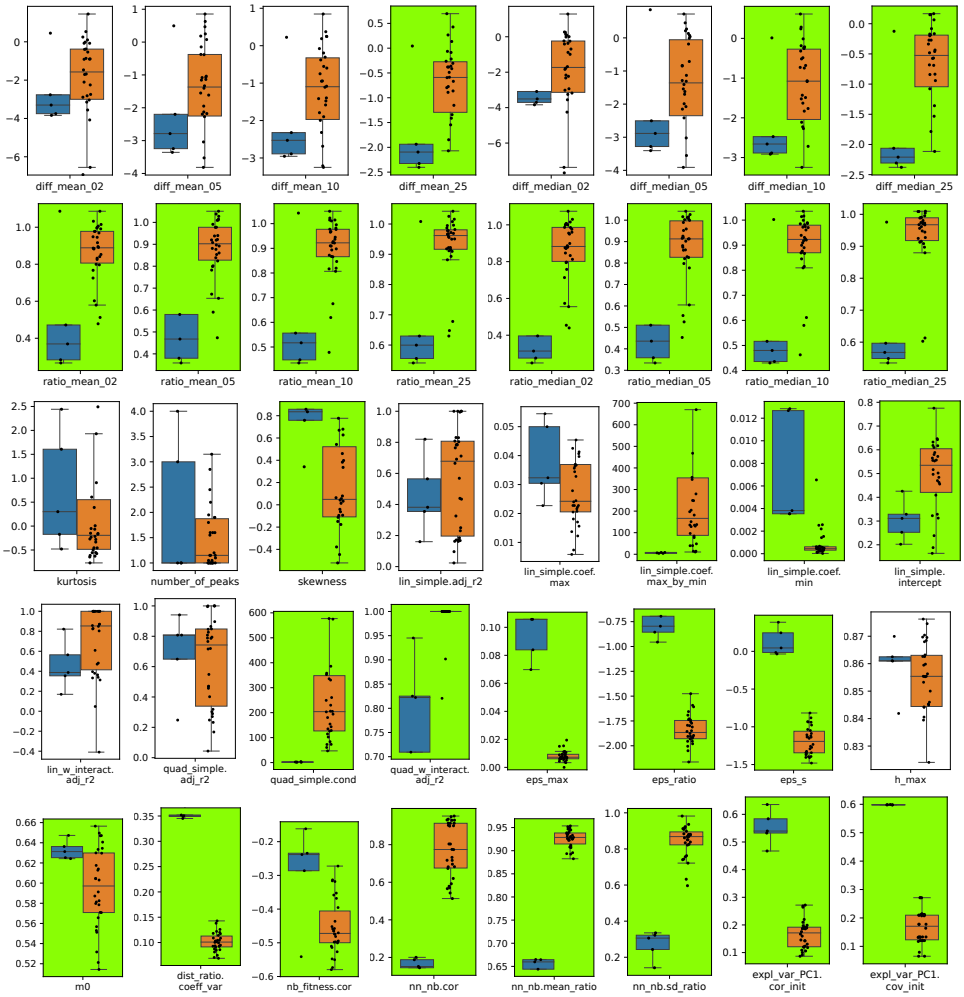


Figure 3.17: Comparison of 40 ELA features in raw feature values between the five ABS functions (*blue*) and 30 crash functions (*orange*). An ELA feature is highlighted with green color, if the null hypothesis *the distribution of ELA features between the ABS and crash functions is similar* is rejected based on the KS test with 95% confidence. Figure taken from [24].

landscape characteristics of some real-world automotive crash problem instances are analyzed in Section 3.3, based on a comparison against those of the BBOB functions and some vehicle dynamics problems.

Essentially, we provide an answer to *RQ1* in detail, by analyzing the optimization landscape characteristics of automotive crashworthiness optimization problems.

RQ1: Within the feature space defined by optimization landscape features, how are the distributions of real-world expensive BBO problems situated w.r.t. some benchmark functions?

Using our proposed ELA-based approach, the landscape characteristics of altogether 20 automotive crashworthiness optimization problems with a dimensionality between 8-d and 22-d for different vehicle crash scenarios are computed. Through hierarchical clustering and visualization of the ELA feature space, many of the crash problems are clustered into different optimization problem classes, which are separated from the BBOB functions. Following this, the BBOB functions are insufficient to represent these crash problems, which belong to distinguishable problem classes. In other words, the BBOB functions are inappropriate to be considered as representative functions for the automotive crash problems, e.g., for HPO purposes.

Beyond that, clear differences in the optimization landscape characteristics among the automotive crash problems can be observed. Instead of forming one cluster, the crash problems are spread throughout the ELA feature space, forming their own clusters, even for crash problems of the same type. Remarkably, the diversity of landscape characteristics within a single engineering domain can be as high as across different domains, when compared against some vehicle dynamic problems. Consequently, an instance-based HPO approach seems to be much more appropriate and effective for solving real-world expensive BBO problems, i.e., the fine-tuning of optimization configurations is performed specifically for a problem instance. In other words, optimization configurations should be fine-tuned according to a particular problem instance, rather than to an engineering domain.

Since appropriate representative functions for the automotive crashworthiness optimization problems cannot be identified from the BBOB suite, we shift our focus towards generating test functions having similar optimization landscape characteristics. Precisely, we evaluate the potential of a random function generator to create

similar test functions that are appropriate to be considered as representative functions for automotive crash problems in the following chapters.

The content of this chapter is mainly based on the author's contribution in the publications [24, 72, 73, 76, 139].

Chapter 4

Representative Functions for Hyperparameter Optimization

As discussed in Chapter 3, the optimization landscapes of automotive crashworthiness optimization are different from those of the BBOB functions in terms of ELA features. In other words, the automotive crash problems belong to problem classes that are not covered by the BBOB suite, and thus, are insufficiently represented by the BBOB functions. Following this, in this chapter we investigate the potential of generating test functions with similar optimization landscape characteristics, which can be considered as representative functions for real-world expensive BBO problems. Precisely, we evaluate the potential of a tree-based function generator that can randomly generate a diverse set of test functions in terms of optimization landscape, as introduced in Section 4.1. Apart from analyzing the landscape characteristics based on ELA features, we further evaluate the potential of considering similar test functions as representative functions for HPO purposes in Section 4.2. Beyond that, we attempt to improve the diversity of test functions that can be generated using the function generator, by guiding the function generation towards specific optimization landscape characteristics using GP, as presented in Section 4.3. Lastly, a conclusion is provided in Section 4.4.

4.1 Tree-based Randomly Generated Functions

Initially proposed in [134], the tree-based function generator can generate a diverse set of random functions or RGFs that belong to different optimization problem classes. In

fact, many of the RGFs have optimization landscape characteristics that are different from the BBOB functions in terms of ELA features, as shown in [151]. In other words, RGFs can potentially complement the BBOB functions to better cover the feature space defined by the ELA features. Inspired by this work, we are motivated to evaluate the potential of this tree-based function generator in generating similar RGFs that can serve as representative functions for real-world expensive BBO problems.

Essentially, a tree-structure function expression is constructed using a predefined pool of mathematical operands and operators, as summarized in Table 4.1, which are randomly selected based on a set of selection probabilities. To improve the diversity of RGFs that can be generated, such as noise, multi-modal landscape, and complex linkage between variables, the so-called *difficulty injection* operation has been implemented to modify the tree expression, as provided in Table 4.2. Furthermore, a *tree-cleaning* operation has been additionally included to simplify the tree representations through elimination of redundant operators, such as simplifying the expression $-x-(-a)$ into $a-x$. An extensive description of these operations is available in [134]. Another advantage of this function generator, apart from the diversity of RGFs, is that an arbitrary number of RGFs can be easily generated.

For a proper integration into our optimization approach, the random function generator, which was originally developed in `Matlab` [135], is reimplemented in `Python` with some minor modifications. Precisely, a RGF is considered invalid and discarded, if any of the following conditions are fulfilled:

1. Error when converting a tree representation to an executable Python expression;

2. Invalid objective values, e.g., missing value or infinity; and

3. A small variance in objective values (< 1.0), to avoid on rare occasions a constant function due to rounding, e.g., objective values are rounded off to a single integer.

Subsequently, the reimplemented random function generator is integrated into our approach in Figure 3.1, where the BBOB functions are replaced with RGFs as test functions. In the following, we take a closer look at the representativeness of RGFs in terms of optimization landscape characteristics, namely based on a visualization of the optimization landscapes in Section 4.1.1 and computation of the respective ELA features in Section 4.1.2.

Table 4.1: Predefined pool of operands and operators considered in the tree-based random function generator. Table taken from [134].

	Notation	Meaning	Syntax
Operands	`a`	A real constant	a
	`rand`	A random number	$rand$
	`x`	Decision vector	$(x_1, \ldots, x_d)$
	`x1`	First variable	x_1
	`xt`	Translated decision variable	$(x_2, \ldots, x_d, 0)$
	`xr`	Rotated decision variable	$\mathbf{xr}$
	`index`	Index vector	$(1, \ldots, d)$
Operators	`add`	Addition	$a + x$
	`sub`	Subtraction	$a - x$
	`mul`	Multiplication	$a \cdot x$
	`div`	Division	a/x
	`neg`	Negative	$-x$
	`rec`	Reciproval	$1/x$
	`multen`	Multiplying by ten	$10x$
	`square`	Square	x^2
	`sqrt`	Square root	$\sqrt{\lvert x \rvert}$
	`abs`	Absolute value	$\lvert x \rvert$
	`exp`	Exponent	e^x
	`log`	Logarithm	$\ln \lvert x \rvert$
	`sin`	Sine	$\sin(2\pi x)$
	`cos`	Cosine	$\cos(2\pi x)$
	`round`	Rounded value	$\lceil x \rfloor$
	`sum`	Sum of vector	$\sum_{i=1}^{d} x_i$
	`mean`	Mean of vector	$\frac{1}{d}\sum_{i=1}^{d} x_i$
	`cum`	Cumulative sum of vector	$(\sum_{i=1}^{1} x_i, \ldots, \sum_{i=1}^{d} x_i)$
	`prod`	Product of vector	$\prod_{i=1}^{d} x_i$
	`max`	Maximum value of vector	$\max_{i=1,\ldots,d} x_i$

Table 4.2: Modifications of a tree-structure function expression ($func$) considered in the difficulty injection operation. Table taken from [134].

	Difficulty	Operation	Probability
1	Noise	Replace $func$ by $func \cdot rand$	0.05
2	Flat landscape	Replace $func$ by $\lceil func \rceil$	0.05
3	Multimodal landscape	Replace $func$ by $func + \sin(2\pi \cdot func)$	0.10
4	Highly multimodal landscape	Replace $func$ by $func + 10\sin(2\pi \cdot func)$	0.05
5	Linkages between all the variables and the first one	Replace $(x_1, \ldots, x_d)$ by $(x_1, \ldots, x_d) - x_1$	0.05
6	Linkages between each two contiguous variables	Replace $(x_1, \ldots, x_d)$ by $(x_1, \ldots, x_d) - (x_2, \ldots, x_d, 0)$	0.05
7	Complex linkages between all the variables	Replace $(x_1, \ldots, x_d)$ by $\mathbf{xr}$	0.05
8	Different optimal values of the variables	Replace $(x_1, \ldots, x_d)$ by $(1 \cdot x_1, \ldots, d \cdot x_d)$	0.05

4.1.1 Visualization of Representative Functions for BBOB

In the first step, we evaluate the potential of RGFs to serve as representative functions for BBO problems based on a visual inspection of the optimization landscapes. Nonetheless, visualizing the optimization landscape is extremely challenging for high-dimensional BBO problems, such as automotive crashworthiness optimization problems. Subsequently, we instead consider the BBOB functions in 2-d, which can be easily visualized. Using the approach proposed in Section 3.1.1, together with a DoE of $150 \cdot d$ samples for the ELA feature computation, a representative function is separately identified from a large set of 10 000 RGFs for each of the BBOB functions. Correspondingly, the optimization landscapes of the BBOB functions and their representative functions are visually compared in Figure 4.1. Generally, the representative functions have a similar optimization landscape in most cases, even for complex functions like F22 and F23. In some cases, it can be observed that the representative functions have a similar topology, yet different orientations, e.g., for F2 and F5.

Nevertheless, the optimization landscapes are visually different between the representative functions and some complex BBOB functions, such as F16. As illustrated in Figure 4.2, the optimization landscape of F16 is compared against the first five RGFs having the most similar landscape characteristics in terms of ELA features. Apart from the first RGF with the smallest difference in ELA features (`similar_1`), the optimization landscape of the remaining RGFs seems to be much more similar to F16, i.e., a highly rugged and repetitive landscape. While a clear explanation remains to be investigated, we suspect that:

1. This might be due to our approach in selecting and processing the ELA features in Section 3.1.1, e.g., perhaps an equal weighting of all ELA features for the distance computation is not optimal for all BBO problems; and/or

2. An ELA feature that can properly capture such complex landscape characteristics is still lacking.

Since the differences in ELA features between F16 and all five RGFs are rather small, we incline towards the first assumption. At the same time, this stresses the importance of considering multiple RGFs as representative functions, to improve the performance and reliability of our approach in fine-tuning algorithm configurations. This topic will be discussed further in the following chapters. While RGFs with a similar optimization landscape can be identified for most of the BBOB functions based on a visual inspection, we are aware that this might not be the case for high-dimensional prob-

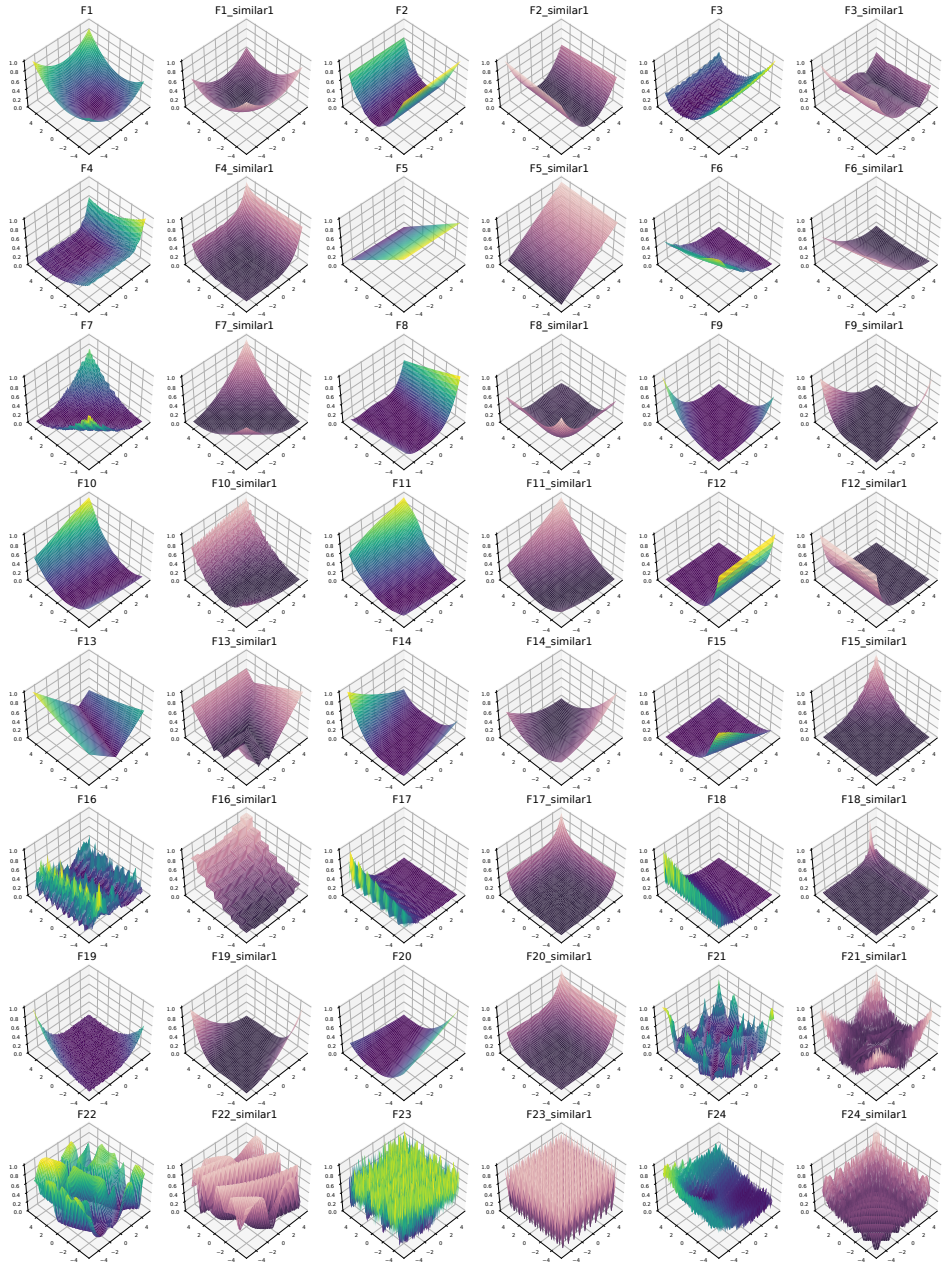


Figure 4.1: Visual comparison of the optimization landscape between pairs of 24 BBOB functions in 2-*d* (*left; yellow-green*) and RGFs identified as representative functions based on ELA features (*right; beige-purple*). The search space $[-5, 5]^2$ is shown on the in-plane axis, while the min-max normalized objective values $[0, 1]$ are on the vertical axis, with 0 being the global optimum. A lighter color represents a larger objective value, while a darker color for a smaller value. Figure taken from [74].

lems. Subsequently, we continue analyzing the representativeness of RGFs based on numerical ELA features in the next step.

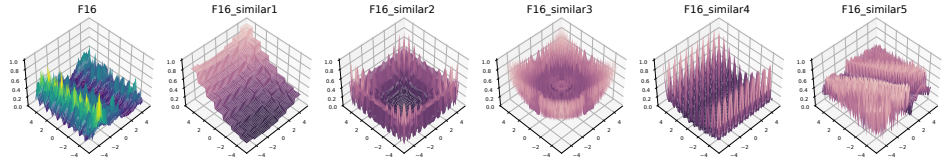


Figure 4.2: Visual comparison of the optimization landscape between F16 in 2-d (*yellow-green*) and the first five RGFs with the most similar landscape characteristics (*beige-purple*). The RGFs are sorted in ascending order from `similar_1` to `similar_5`, starting with the one having the smallest difference in ELA features. The search space $[-5, 5]^2$ is shown on the in-plane axis, while the min-max normalized objective values $[0, 1]$ are on the vertical axis, with 0 being the global optimum. Figure taken from [74].

4.1.2 Representative Functions for Automotive Crash Problems

Next, the potential of RGFs to serve as representative functions for real-world expensive BBO problems is evaluated based on their ELA features. Considering the same experimental setup described in Section 3.3, here we identify representative functions from a set of 1 000 RGFs for the automotive crashworthiness problems. Using the crash problem instance `Crash_2` as a representative example, the crash functions are now clustered in the same groups with several RGFs, as shown in Figure 4.3. Particularly, the maximum force function and rotation function have a relatively small Euclidean distance to their neighboring RGFs, compared to the BBOB functions in Figure 3.12. In other words, these RGFs are indeed much more similar to the automotive crash problems in terms of ELA features, in comparison to the BBOB functions.

By projecting the high-dimensional ELA space to a 2-d visualization using the t-SNE approach, the similarity between automotive crash functions and RGFs can be visualized, as shown in Figure 4.4. In line with our interpretations in Section 3.3, many of the crash functions are clustered into different groups that are separated away from the BBOB functions. On the contrary, the crash functions are closely clustered with some of the RGFs, indicating that these neighboring RGFs are much more similar to the crash functions w.r.t. their optimization landscapes. Following this, such similar RGFs can be considered as representative functions for the automotive crash problems.

Since no neighboring RGF can be identified for the intrusion function, we suspect that:

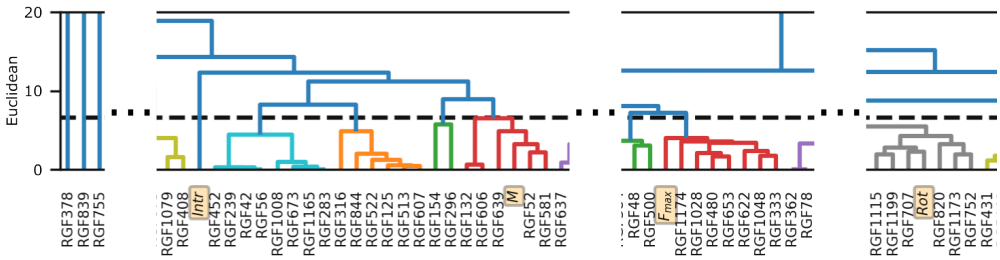


Figure 4.3: Clustering pattern of the four crash functions and 1 000 RGF (labeled from `RGF1` to `RGF1 000`) for the problem instance `Crash_2`. Only relevant sections of the clustering pattern and y-axis are shown here due to the limited space. The labels of crash functions are highlighted (*orange*) and the reference Euclidean distance (same as in Figure 3.12) is marked with a dashed line, where clusters below the reference distance are assigned with different colors. Figure taken from [73].

1. Such a similar RGF is not available in the test function set, which could be easily solved by expanding the function set with more RGFs; or
2. Creating such a similar RGF is currently not possible using the random function generator, which is a limitation in our approach that makes further improvements necessary.

Considering that increasing the number of RGFs does not always lead to finding a more similar RGF, this indicates that some of the optimization problem classes are indeed insufficiently covered by the random function generator. Moreover, the discontinuous nature of automotive crash problems, for instance, is currently not considered in the random function generator, which could be potentially extended using step functions. Based on the fact that a similar clustering pattern can be observed in the remaining automotive crash problem instances, we are confident that RGFs with a similar optimization landscape can be identified for real-world expensive BBO problems, such as automotive crashworthiness optimization.

4.2 Representativeness for Fine-Tuning of Algorithm Configuration

Previously in Section 4.1.2, it has been shown that some of the RGFs have similar optimization landscapes in terms of ELA features, and thus, belong to the same opti-

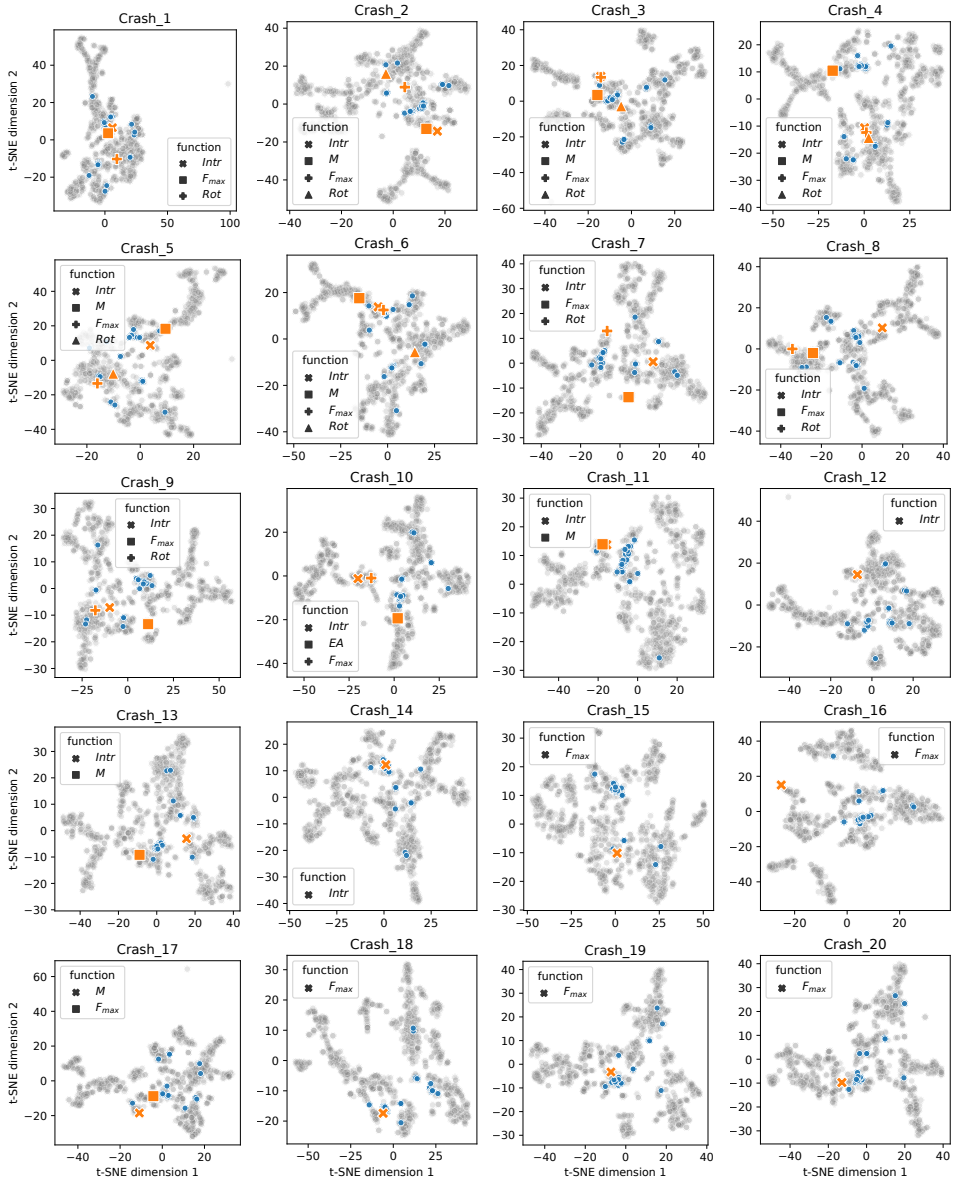


Figure 4.4: Visualization of the 2-*d* ELA feature space for the crash functions (*orange*), 24 BBOB functions (*blue*), and 1 000 RGFs (*gray*) using the t-SNE approach for all 20 automotive crash problems. Each dot represents a problem instance. Figure taken from [73].

mization problem classes as the automotive crash problems. Subsequently, RGFs have promising potential to be considered as representative functions for real-world expensive BBO problems. More importantly, such cheap-to-evaluate RGFs can be exploited for the fine-tuning of optimization configurations to optimally solve expensive BBO problems, as proposed in Section 1.1. To numerically quantify the representativeness of RGFs for HPO purposes, we evaluate the effectiveness of RGFs in estimating the actual performance of different optimization configurations on BBO problems to-be-solved. In other words, we analyze the potential of RGFs in identifying optimization configurations that can perform well on the BBO problems.

Since an investigation based on real-world expensive BBO problems is computationally infeasible and extremely time-consuming, we instead focus on the 24 BBOB functions of the first instance in 20-d, which is a typical problem dimensionality for automotive crash problems. Using the approach proposed in Section 3.1.1, a DoE of $20 \cdot d$ samples generated using the Sobol' sampling, and a bootstrapping of 30 repetitions for the ELA feature computation, several representative functions are separately identified from a large set of 10 000 RGFs for each of the BBOB functions. Unless otherwise stated, the RGF having the smallest difference in ELA features is considered as representative function for each BBOB function.

In this investigation, two state-of-the-art BBO algorithms are considered, namely ModCMA and BO, using the configurations summarized in Table 4.3. Based on an exhaustive grid search approach, a total of 972 configurations for ModCMA and 9 configurations for BO are considered, where each configuration is repeated for 30 times using different random seeds. Considering that the time-complexity of building GPR models rapidly increases with the number of DoE samples, a smaller function evaluation budget is assigned for BO. Moreover, the same DoE samples for the computation of ELA features are considered for the training of GPR models in BO. Overall, a fixed budget of $100 \cdot d$ evaluations is allocated for ModCMA, while only $15 \cdot d$ evaluations for BO. While better optimization configurations could be possibly identified, e.g., using proper HPO instead of the simple grid search, this is not the motivation of this investigation. In fact, we focus on analyzing the potential of RGFs to serve as representative functions in estimating the actual performance of optimization configurations on unseen BBO problems.

Table 4.3: Summary of optimization algorithm configurations considered for ModCMA and BO. Table taken from [73].

Algorithm	Hyperparameter	Value
ModCMA	Number of children	{ 10, 20 }
	Number of parent	{ 3, 5 }
	Initial standard deviation	{ 0.1, 0.3, 0.5 }
	Learning rate step size control	{ 0.1, 0.5, 1.0 }
	Learning rate covariance matrix adaptation	{ 0.1, 0.5, 1.0 }
	Learning rate rank-μ update	{ 0.1, 0.5, 1.0 }
	Learning rate rank-one update	{ 0.1, 0.5, 1.0 }
BO	DoE sample size	{ 50, 150, 250 }
	Acquisition function	{ EI, PI, UCB }

4.2.1 Performance Metric

While a variety of performance metrics have been introduced to evaluate different aspects of an optimization run, such as the expected hitting time [146], many of these metrics are less practical for real-world expensive BBO problems, since the global optimum is oftentimes not known. Subsequently, we mainly consider the following two metrics to evaluate the performance of optimization runs within the scope of this thesis:

Best-found solution: Optimization solutions having a smaller objective value are better; and

Area under the optimization convergence curve: In real-world applications, having a faster optimization convergence is often considered as important as finding the global optimum. In this context, finding an acceptable optimization solution within a shorter wall-clock time and/or using less computational resources can be beneficial in some cases. In fact, identifying the global optimum for real-world expensive BBO problems is extremely challenging, e.g., due to the strongly nonlinear nature of automotive crash problems.

Following this, we propose considering the Area Under the Curve (AUC) of optimization convergence (Figure 4.7) as a performance metric, which is informative about the solutions and convergence speed. By minimizing the AUC metric, e.g., using HPO, we are essentially searching for configurations that have an optimal trade-off between optimization solutions and convergence speed. Furthermore, in cases where multiple configurations perform equally well in terms of the best-found solution, the AUC metric can provide additional information, e.g., for the

ranking of configurations. Throughout this thesis, the AUC of an optimization run is first computed using the min-max rescaled objective values based on the global optimum and worst DoE sample, and then divided by the total function evaluation budget.

4.2.2 Optimization Performance of ModCMA

Firstly, the same set of ModCMA configurations is independently evaluated on both the BBOB functions and their respective representative functions. According to their performances, the optimization configurations are ranked in ascending order, where the configuration having the best performance is assigned with rank one. Meanwhile, the same rank is assigned for configurations having the same performance or ties. To have an understanding about the overall tendency of configuration performances, the rankings of ModCMA configurations for the BBOB functions and representative functions are compared against each other based on Spearman's correlation. Basically, a positive Spearman's correlation indicates that two rankings are similar, i.e., excellent configurations for the representative functions can perform well on the BBOB functions, while a negative correlation for the opposite.

In most of the BBOB functions, a clear positive correlation in configuration performances can be observed for both performance metrics, as illustrated in Figure 4.5. Following this, the optimization performance of ModCMA configurations on the BBOB functions can be estimated using the RGFs as representative functions. In other words, RGFs have promising potential to be exploited as representative functions to identify optimal configurations, e.g., for HPO purposes. Nevertheless, the configuration performance correlations are rather low for complex functions like F7, F16, and F23, especially the slightly negative correlation in the AUC metric for F16.

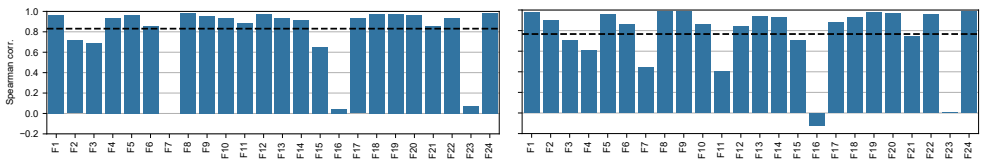


Figure 4.5: Spearman's correlation based on the ranking of ModCMA configurations between 24 BBOB functions and their representative functions, using the best-found optimization solution (*left*) and AUC (*right*) as performance metric. The mean correlation across all BBOB functions is marked with a dashed line. Figure taken from [73].

Eventually, only several top-performing configurations or the best configuration

identified using the representative functions will be applied for solving real-world expensive BBO problems. Following this, we delve into analyzing the competitiveness of the predicted top five ModCMA configurations on the BBOB functions. As shown in Figure 4.6, the predicted best five configurations can generally perform well on most of the BBOB functions, similar to the previous results based on Spearman's correlation. On the other hand, this is not the case for F7, F16, and F21. Considering the low Spearman's correlation in configuration ranking, this is somewhat expected for F7 and F16. Meanwhile, the situation is different for F21 and F23, where the relationship between Spearman's correlation and the performance of predicted top configurations is completely the opposite.

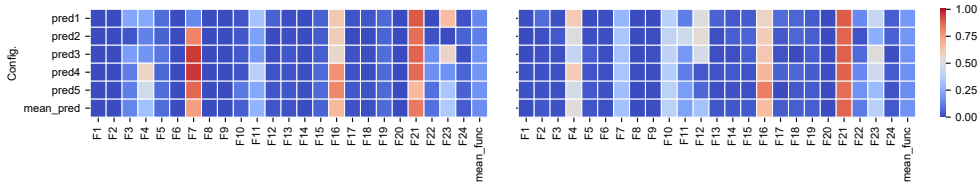


Figure 4.6: Performance of the predicted best five ModCMA configurations on 24 BBOB functions, rescaled between 0 (the best; *blue*) and 1 (the worst; *red*), using the best-found optimization solution (*left*) and AUC (*right*) as performance metric. An extra row is included for the mean across all predicted configurations (*bottom*) and a column for the mean across all BBOB functions (*right*). Figure taken from [73].

To explain this observation, we examine further the optimization runs for these four BBOB functions, as summarized in Figure 4.7. Due to the fact that the same optimization solution can be found for F7 and F23, many of the ModCMA configurations are assigned with the same rank, leading to an ambiguous ranking of configuration performances. Especially for F7, the situation is more extreme, where all configurations are considered as the "best" configuration. For F16, on the other side, the global optimum of the representative functions can only be found occasionally by the optimization runs, resulting in an inconsistent configuration ranking. While the overall configuration performances appear to be similar, the top performing configurations seem to be different for F21 and its representative function.

Consequently, we proceed our investigation with the hypothesis *perhaps not all RGFs are appropriate to serve as representative functions for HPO purposes*, particularly when a clear configuration ranking is not possible. Correspondingly, our analysis is performed again for F7, F16, F21, and F23, but now considering RGFs having the second most similar landscape characteristics in terms of ELA features as represen-

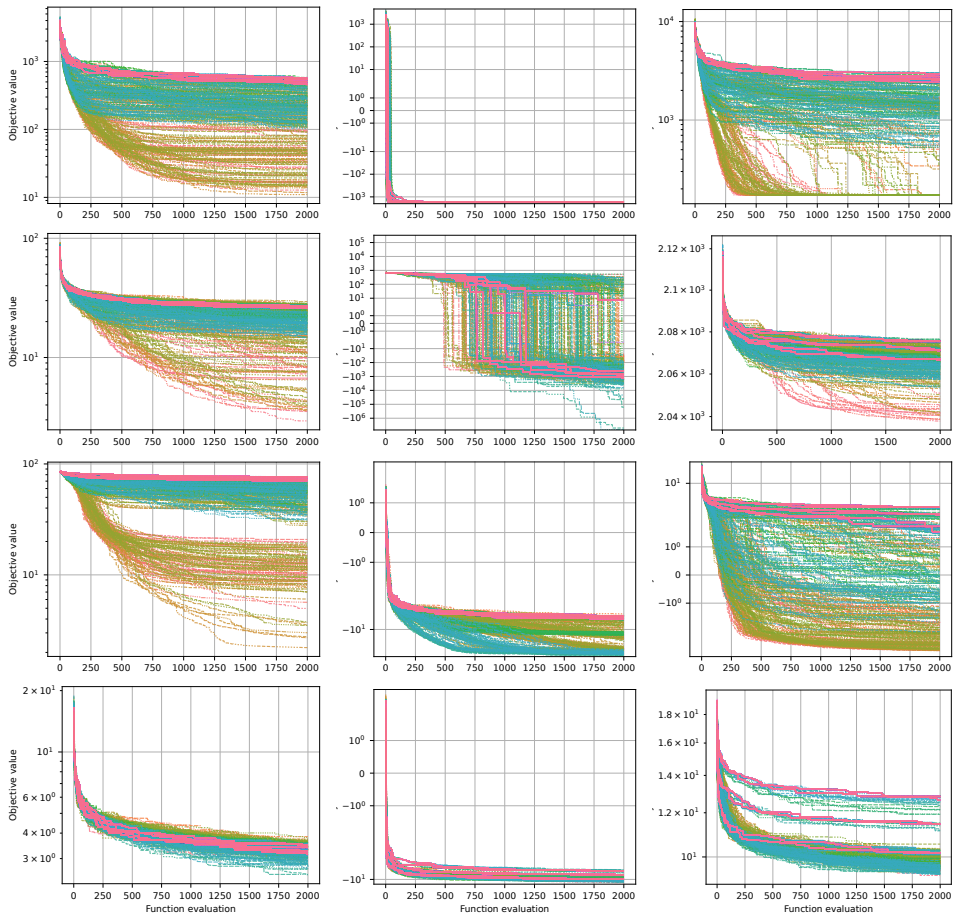
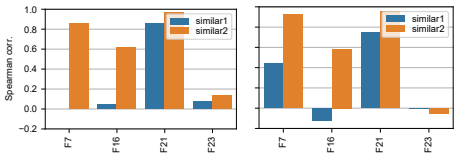
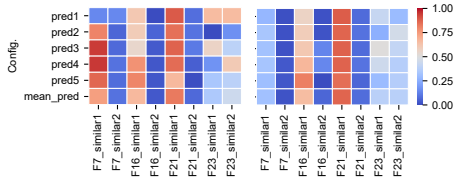


Figure 4.7: Optimization runs of altogether 972 ModCMA configurations for F7, F16, F21, and F23 (*top to bottom*; *left*), their most similar RGFs (*middle*), and their second most similar RGFs (*right*), respectively. The objective values in log-scale are shown on the y-axis, while the number of function evaluations is shown on the x-axis. For the BBOB functions (*left*), the objective values are rescaled to 0 being the true global optimum. Each curve represents the optimization run (median of 30 repetitions) of a configuration and each color represents the same configuration. Figure taken from [73].

tative functions. Generally, significant improvements in both the ranking correlation and performance of predicted best configurations can be noticed for most BBOB functions, as depicted in Figure 4.7 and Figure 4.8. Although having similar optimization landscape characteristics, arguably no improvement can be observed for F23, when using the second most similar RGF as representative function. Hence, it is suspected that this optimization problem class is insufficiently represented and covered by the RGFs, which requires attention in future investigations.



(a) Spearman's correlation of the configuration ranking, using the best-found optimization solution (*left*) and AUC (*right*) as performance metric.



(b) Performance of the predicted best five configurations on the BBOB functions, rescaled between 0 (the best; *blue*) and 1 (the worst; *red*), using the best-found optimization solution (*left*) and AUC (*right*) as performance metric.

Figure 4.8: Comparison of the representativeness between RGF with the most similar (`similar1`) and second most similar (`similar2`) landscape characteristics in terms of ELA features for F7, F16, F21, and F23. Figure taken from [73].

Beyond that, the ranking quality of ModCMA configurations is additionally evaluated using the normalized Discounted Cumulative Gain (nDCG) [50], a common metric utilized in the field of information retrieval. Fundamentally, the quality of a ranking is measured according to the respective relevance scores, where a perfect ranking is assigned with the best score of 1.0. In our investigation, the quality of ModCMA configuration ranking for the BBOB functions is evaluated based on the ranking predicted using representative functions as relevance score. As shown in Figure 4.9, the configuration ranking has a high nDCG score for all BBOB functions, when all configurations are taken into account. On the contrary, when only the top five configurations are taken into consideration, the configuration ranking is below the average for some BBOB functions, e.g., for F3, F15, F16, and F23, using the best-found solution as performance metric.

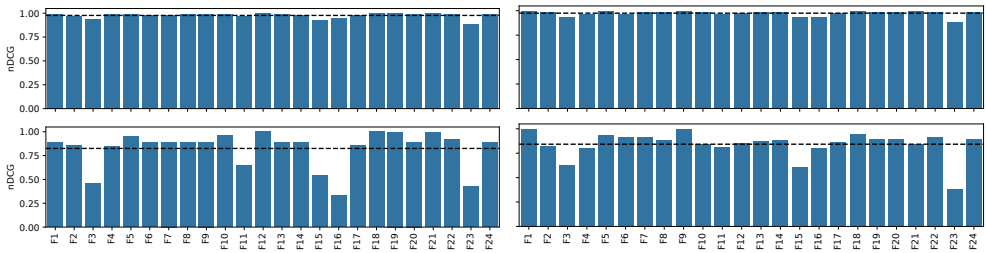
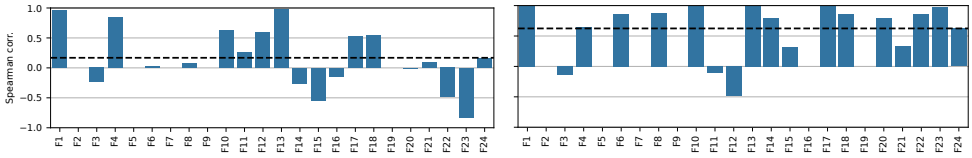


Figure 4.9: Ranking quality of ModCMA configurations based on the nDCG method, when all configurations (*top*) or only the top five configurations (*bottom*) are considered, using the best-found optimization solution (*left*) and AUC (*right*) as performance metric. Here, the second most similar RGFs are considered as representative functions for F7, F16, F21, and F23. The average score across all BBOB functions is marked with a dashed line. Figure taken from [73].

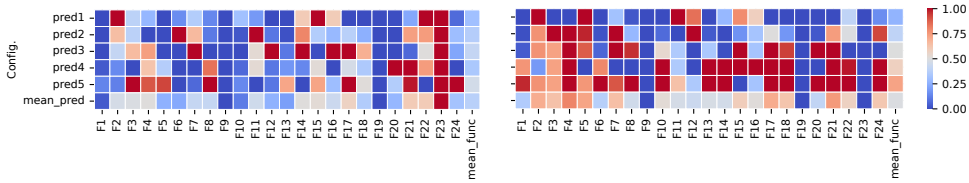
4.2.3 Optimization Performance of BO

Similar to the previous investigations in Section 4.2.2, the representativeness of RGFs in estimating BO configurations is evaluated w.r.t. Spearman's correlation, the performance of predicted best configuration, and the quality of configuration ranking based on the nDCG method, as summarized in Figure 4.10. Compared to ModCMA, the average Spearman's correlation for BO configurations is much lower, particularly for more complex functions like F24. Nevertheless, the top-performing configurations predicted using the representative functions seem to be still competitive for most BBOB functions. It is suspected that the weak ranking correlation and poor performance of predicted best configurations might be partly due to the rather small BO hyperparameter search space (Table 4.3), which could be improved by taking more configurations into consideration.

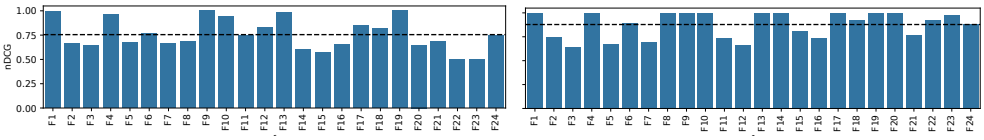
For some of the BBOB functions, such as F2, F5, F7, F9, F12, F15, F16, and F23, the best-found solution within the total evaluation budget belongs to the initial DoE samples, which are considered for the training of GPR models. In other words, no optimization solution better than the best DoE sample can be identified during the optimization runs. Since all BO configurations can find the same solution, they are considered as equally good, resulting in an ambiguous configuration ranking, as clarified in Section 4.2.2. An alternative to overcome this problem, for instance, is by using other sampling methods, such as LHS, where the best-found solution is not part of the initial DoE samples. While the results are not included here, the Spearman's correlation can be improved to around 0.9 for F5, when the LHS method and best-



(a) Spearman's correlation based on the ranking of BO configurations between the BBOB functions and their representative functions. The mean correlation across all BBOB functions is marked with a dashed line.



(b) Performance of the predicted best five BO configurations on the BBOB functions, rescaled between 0 (the best; *blue*) and 1 (the worst; *red*). An extra row is added for the mean across all predicted configurations (*bottom*) and a column for the mean across all BBOB functions (*right*).



(c) Ranking quality of the top five BO configurations based on the nDCG method. The average score across all BBOB functions is marked with a dashed line.

Figure 4.10: Evaluation of the representativeness of RGFs in estimating the performance of BO configurations for 24 BBOB functions, using the best-found optimization solution (*left*) and AUC (*right*) as performance metric. Figures are taken from [73].

found solution as performance metric are considered in our preliminary testing. Apart from this, another alternative is to replace such representative functions with other RGFs, e.g., using the next most similar RGF, since a large set of RGFs can be easily generated. In fact, this approach seems to be more practical for real-world expensive BBO problems, particularly if a modification of the DoE samples is prohibited or computationally infeasible.

4.2.4 Identifying Appropriate Representative Functions

Unlike the widely-used BBOB functions, the global function properties of RGFs, such as the optimization landscape characteristics and global optimum, are not known a priori. Moreover, some of the RGFs are insufficiently discriminative in distinguishing different configurations, despite having similar landscape characteristics in terms of ELA features, as shown in Section 4.2.2. Following this, not all RGFs are appropriate to be considered as representative functions for HPO purposes.

Using the HPO results for three selected RGFs in Figure 4.11 as an illustrative example, we consider RGFs having a similar pattern to `RGF1` as appropriate for HPO purposes, where the performance of different configurations can be clearly distinguished and ranked with only a few ties. More importantly, optimal configurations or the best configuration can be easily identified based on the configuration ranking. On the other side, `RGF2` shows an extreme example, where many or all configurations can perform equally good, resulting in an ambiguous ranking, and thus, difficulty in identifying optimal configurations. It is suspected that such RGFs have a low optimization complexity, and hence, are not appropriate to serve as representative functions. Notably, two RGFs can exhibit the totally opposite HPO patterns, although the difference in their ELA features is relatively small, which remains an open question for future investigations. To improve the reliability of HPO based on representative functions, RGFs similar to `RGF3` are additionally excluded, where the best-found solution is an extreme outlier that can be found occasionally.

To properly identify RGFs that can serve as representative functions for HPO purposes, the following selection process is implemented:

1. **Estimation of the global optimum:** Based on a brute-forcing approach, a HPO using a combination of TPE and ModCMA is directly applied on each RGF, using a similar optimization setup as in Section 5.2.1. Using Equation 4.1, the global optimum y_{opt} of a RGF is then approximated based on the best-found solution during HPO y_{hpo}.

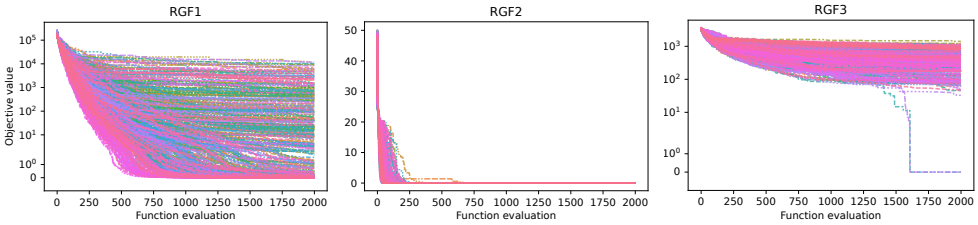


Figure 4.11: Optimization convergence curves of altogether 500 ModCMA configurations evaluated using HPO for three selected RGFs, where a clear configuration ranking is possible (*left*), the configuration ranking is ambiguous (*middle*), and the best-found solution is an outlier that can only be occasionally found (*right*). Each curve and color represents a configuration, showing the median over 20 repetitions. While the function evaluation is shown on the x-axis, the rescaled objective value is shown on the y-axis, where 0 is the best-found solution in all runs. A smaller objective value is better. Figure taken from [74].

$$y_{opt} = \begin{cases} \lfloor y_{hpo} \rfloor \,, & \text{if } 0 \leq |y_{hpo}| < 10 \\ \lfloor y_{hpo}/10 \rfloor \cdot 10 \,, & \text{if } 10 \leq |y_{hpo}| < 100 \\ \lfloor y_{hpo}/10^p \rfloor \cdot 10^p \,, & \text{otherwise} \end{cases} \,, \tag{4.1}$$
$$p = \lfloor \log_{10} |y_{hpo}| \rfloor - 1 \,,$$

where y_{hpo} is either rounded to the nearest lower integer for a small $|y_{hpo}|$, or rounded based on the nearest lower power of 10. In fact, having an estimated global optimum of RGFs is essential in our approach to facilitate an evaluation of configurations across different representative functions with varying scale ranges, e.g., when using multiple representative functions (Section 5.1).

2. **Identifying RGFs appropriate for HPO:** Firstly, all optimization configurations evaluated during HPO from the previous step are assigned with a rank according to their performances, where ties have the same rank. To quantify the ambiguity of a configuration ranking, the ranking is compared against a strict ranking, i.e., a perfect ranking without tie, based on Kendall rank correlation coefficient. Principally, we consider a configuration ranking as ambiguous, if the correlation is lower than 0.9, e.g., due to too many ties. Moreover, the global optimum is considered as an outlier, if the respective standard score or z-score is more than 3 standard deviations away from the mean HPO distribution.

3. **Selection or elimination of RGFs:** A RGF is considered as inappropriate for HPO purposes and eliminated, if any of the aforementioned conditions is fulfilled. Otherwise, the RGF is considered as an appropriate representative function.

In this way, a set of RGFs that are appropriate to serve as representative functions for HPO purposes can be identified. To minimize the overall computational expenses, this selection process is performed only as much as necessary until the desired number of appropriate representative functions is achieved. Precisely, starting with the RGF having the smallest difference in ELA features in ascending order, the RGFs are individually examined and selected/discarded.

4.3 Guiding the Function Generation using Genetic Programming

In Section 4.1, it has been shown that the tree-based random function generator can indeed generate a diverse set of test functions in terms of optimization problem classes. More importantly, RGFs with optimization landscape characteristics that are similar to real-world expensive BBO problems in terms of ELA features can be identified. On top of that, these RGFs have promising potential to be exploited as representative functions for HPO purposes, as demonstrated in Section 4.2. Subsequently, the actual performance of optimization configurations on unseen BBO problems can be estimated, and thus, optimal optimization configurations can be identified. Despite of that, there is still room for improvements in the function generator concerning two aspects, namely:

- In some cases, test functions with similar optimization landscape characteristics still could not be identified for some BBO problems from a large set of RGFs; and
- The efficiency of generating function randomly, which is essentially a RS approach, can be improved.

Following this, we investigate the potential of guiding the function generation towards specific optimization landscape characteristics in terms of ELA features, by integrating GP into the random function generator. Precisely, we attempt to generate test functions with specific optimization landscape characteristics, by minimizing the differences in ELA features based on a GP approach. Unlike the popular ISA, focusing on generating space-covering instances for a general optimization benchmarking purpose,

we aim to create a highly specialized set of test functions for a particular optimization problem class.

4.3.1 ELA-guided Function Evolution using GP

Basically, our investigation is based on the same random function generator employed in Section 4.1, considering a similar set of mathematical operands and operators, as summarized in Table 4.4. Subsequently, the GP search space consists of a terminal space $\mathcal{S}$ (operands) and function space $\mathcal{F}$ (operators), where $\mathcal{T} = \mathcal{S} \cup \mathcal{F}$. Unlike typical GP-based symbolic regression approaches, where each design variable x_i is considered as an individual terminal, we instead focus on tree-based expressions, i.e., *vector-based input* $\mathbf{t} = (t_1, \cdots, t_d)$ [7], to facilitate a comparison against the RGFs.

Table 4.4: List of mathematical operands and operators considered. The respective selection probability for the GP sampling in the first generation, and if any, the protection rules are additionally provided. Table taken from [75].

Notation	Meaning	Syntax	Protection	Probability
		Operands ($\mathcal{S}$)		
`x`	Decision vector	$(x_1, \ldots, x_d)$		0.6250
`a`	A real constant	a		0.3125
`rand`	A random number	$rand$		0.0625
		Operators ($\mathcal{F}$)		
`add`	Addition	$a + x$		0.1655
`sub`	Subtraction	$a - x$		0.1655
`mul`	Multiplication	$a \cdot x$		0.1098
`div`	Division	a/x	Return 1, if $\lvert x\rvert \leq 10^{-20}$	0.1098
`neg`	Negative	$-x$		0.0219
`rec`	Reciprocal	$1/x$	Return 1, if $\lvert x\rvert \leq 10^{-20}$	0.0219
`multen`	Multiplying by ten	$10x$		0.0219
`square`	Square	x^2		0.0549
`sqrt`	Square root	$\sqrt{\lvert x\rvert}$		0.0549
`abs`	Absolute value	$\lvert x\rvert$		0.0219
`exp`	Exponent	e^x		0.0219
`log`	Logarithm	$\ln \lvert x\rvert$	Return 1, if $\lvert x\rvert \leq 10^{-20}$	0.0329
`sin`	Sine	$sin(2\pi x)$		0.0329
`cos`	Cosine	$cos(2\pi x)$		0.0329
`round`	Rounded value	$\lceil x \rceil$		0.0329
`sum`	Sum of vector	$\sum_{i=1}^{d} x_i$		0.0329
`mean`	Mean of vector	$\frac{1}{d}\sum_{i=1}^{d} x_i$		0.0329
`cum`	Cumulative sum of vector	$(\sum_{i=1}^{1} x_i, \ldots, \sum_{i=1}^{d} x_i)$		0.0109
`prod`	Product of vector	$\prod_{i=1}^{d} x_i$		0.0109
`max`	Maximum value of vector	$\max_{i=1,\ldots,d} x_i$		0.0109

Implemented based on `DEAP` [38], we propose a GP-based function generator with the following workflow:

Input: A set of DoE samples $\mathcal{X}$ and the targeted ELA features are required as input.

Optimization objective: Essentially, we aim to minimize the differences in ELA features between the target function and individuals generated using GP, where the differences are quantified using some distance metrics. For the ELA feature computation, a similar approach as proposed in Section 3.1.1 is considered, namely:

- The objective values of individuals are normalized using min-max scaling to reduce inherent bias before the ELA feature computation;
- The computed ELA features are normalized for the distance computation, using min-max scaling based on the minimum and maximum feature values from a set of 24 BBOB functions and 5 instances each; and
- Highly correlated ELA features are removed.

Initial population: In the first generation, an initial population is initialized using random sampling, where the mathematical operands and operators are randomly selected based on a set of probabilities, as summarized in Table 4.4. To minimize the overall computational effort, the initial population consists of 50 individuals, where the depth of a tree expression is limited between 3 and 12 levels. To ensure an optimal initialization of optimization runs, here a resampling is performed whenever a generated individual is infeasible, as explained in the following. In other words, such individuals are discarded and replaced by generating new individuals. Following this, the initial population is free from infeasible individuals due to error 1 and 2.

Mating selection and variation: In our approach, we consider the tournament selection with the size of 5, a subtree crossover probability of 0.5, and a subtree mutation probability of 0.1, while using the default GP setting for the remaining hyperparameters. With the combination of a 0.5 crossover rate and a 0.1 mutation rate, there is a 0.45 probability that a selected individual is not modified in any way, and simply passed to the next generation without being evaluated. Although the optimization performance of GP could be potentially improved using HPO, finding the best GP configuration is not our focus here, and thus, it is not taken into consideration.

Infeasible individual: A GP-generated individual is considered as infeasible, if any of the following conditions is fulfilled:

1. Error when converting the tree representation to an executable Python expression;
2. Bad objective values, e.g., infinity, missing, or single constant value;
3. Error in ELA features computation, e.g., due to equal fitness values in all samples; and
4. Invalid distance, due to missing values in ELA features.

Subsequently, an infeasible individual is penalized with an extremely large fitness value of 10 000.

Output: Individuals with similar ELA features, or ideally, *identical* ELA features as the target function, are provided as solutions of the GP optimization runs.

4.3.2 Result Discussion

In our investigations, the performance of GP-based function generator is evaluated using 24 BBOB functions in 2-d, 5-d, and 10-d as target functions. For the ELA feature computation, we consider a DoE of $150 \cdot d$ samples and 5 bootstrapping repetitions, using a similar approach as introduced in Section 3.1.1. Since an ELA feature can be principally considered as a random variable, a statistical distance measure based on the average Wasserstein distance is considered here. Precisely, we aim to minimize the average Wasserstein distance between the ELA features of the first five BBOB instances and a GP-generated function. To minimize the overall computational expenses, only one GP optimization run is performed for each target function in each dimensionality, where a fixed function evaluation budget of 50 generations with 50 individuals each is allocated, despite of the risk of a premature convergence [117].

GP-generated functions

Firstly, we analyze the performance of GP optimization runs w.r.t. their convergence trajectory. Using F1 in 2-d as a representative example, as shown in Figure 4.12, we can observe that GP improves over the initial population as time goes on, by reducing the distances or differences in ELA features. While not included here, the infeasible individuals sum up to around 2% of the total evaluations in this optimization run.

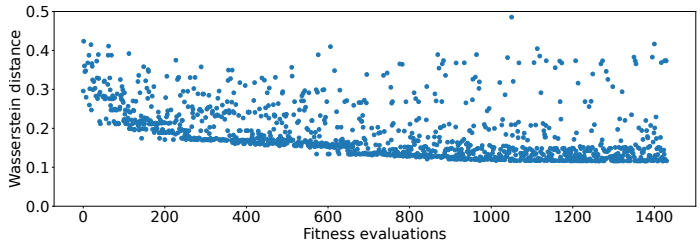


Figure 4.12: Convergence trajectory of GP for F1 in 2-d. The fitness evaluations are shown on the x-axis, while the Wasserstein distance is shown on the y-axis. Figure taken from [75].

Moreover, the functions generated using GP-based function generator are compared against the RGFs from Section 4.1. In this context, altogether 1 000 RGFs are generated for each dimensionality and their respective Wasserstein distances to 24 BBOB functions are computed. Subsequently, these distances of RGFs are compared against those of GP-generated functions, which is about 1 400 functions, as summarized in Figure 4.13. For most BBOB functions, the GP-generated functions seem to be always better compared to RGFs, tending to have a smaller Wasserstein distance to the target functions, apart from some exceptions like F12 in 10-d.

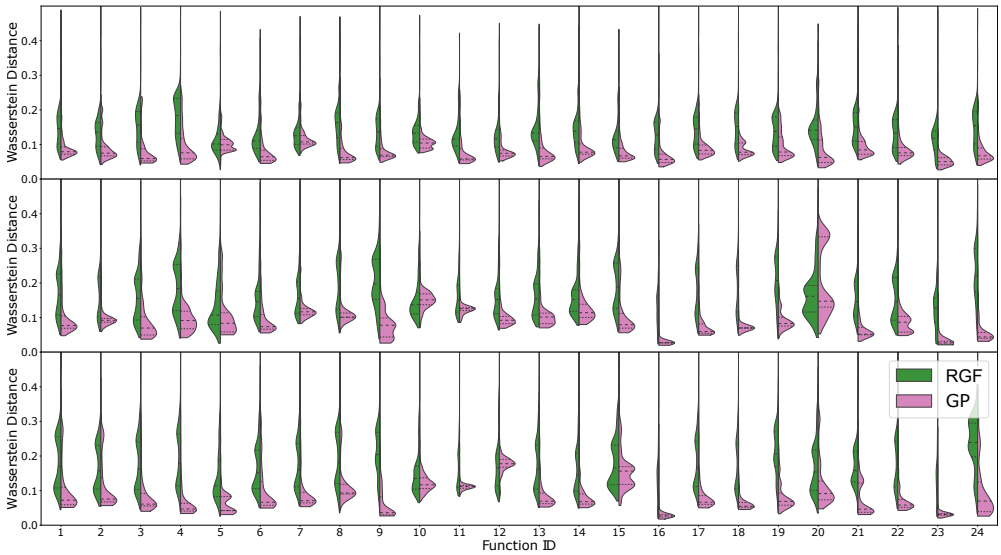


Figure 4.13: Distribution of Wasserstein distances of RGFs (*green*) and GP-generated functions (*pink*) to 24 BBOB functions in 2-d (*top*), 5-d (*middle*), and 10-d (*bottom*) in terms of ELA features. Figure taken from [75].

Beyond that, we evaluate the quality of GP-generated functions based on a visual comparison of optimization landscape. Again using F1 in 2-d as an example in Figure 4.14, the GP-generated function with the smallest Wasserstein distance has a rather low resemblance to a sphere function, unlike our expectation. On the other hand, a much closer visual matching can be observed for F5, as shown in Figure 4.15. Interestingly, some of the GP-generated functions, e.g., the function $\frac{x_0+x_1}{2}$ in row 7 column 5, have a relatively large distance, despite having a similar representation as the target function. Subsequently, this leads to the question *whether the choice of Wasserstein distance based on the target distribution in ELA feature space is appropriate to capture the global function properties.*

Differences in ELA features

Secondly, we take a closer look at the ELA features of the GP-generated functions, focusing on F5 as our target function. As depicted in Figure 4.16, the feature `ela_meta.lin_simple.coef.min` and `ela_meta.lin_simple.coef.max` are clearly different between F5 and the GP-generated function $\frac{x_0+x_1}{2}$, revealing that the function steepness might be different. For a linear slope function like F5, however, the impact should be minimal, since the global properties are mostly preserved. Subsequently, this indicates that different ELA features are critical to represent different types of target functions, which will be discussed further in the following.

To gain a better understanding about the similarities between functions, we project the high-dimensional ELA feature space to a 2-d visualization using the Uniform Manifold Approximation Mapping (UMAP) approach for F5 and the corresponding GP-generated functions. Precisely, the GP-generated functions are projected based on the same mapping defined by the ELA feature representations of 24 BBOB problems, using all five bootstrapped repetitions for each instance. As illustrated in Figure 4.17, most of the GP-generated functions closely cluster around the target functions.

Distances in ELA feature space

The discrepancy between the distances in ELA feature space and our visual understanding of the global function properties might be due to the equal weighting of all ELA features considered in our approach. In other words, ELA features that are more sensitive to a small deviation can have the same impact as features that are crucial to characterize certain function properties. Subsequently, we analyze the relative standard deviation of ELA features within the same BBOB instances, as vi-

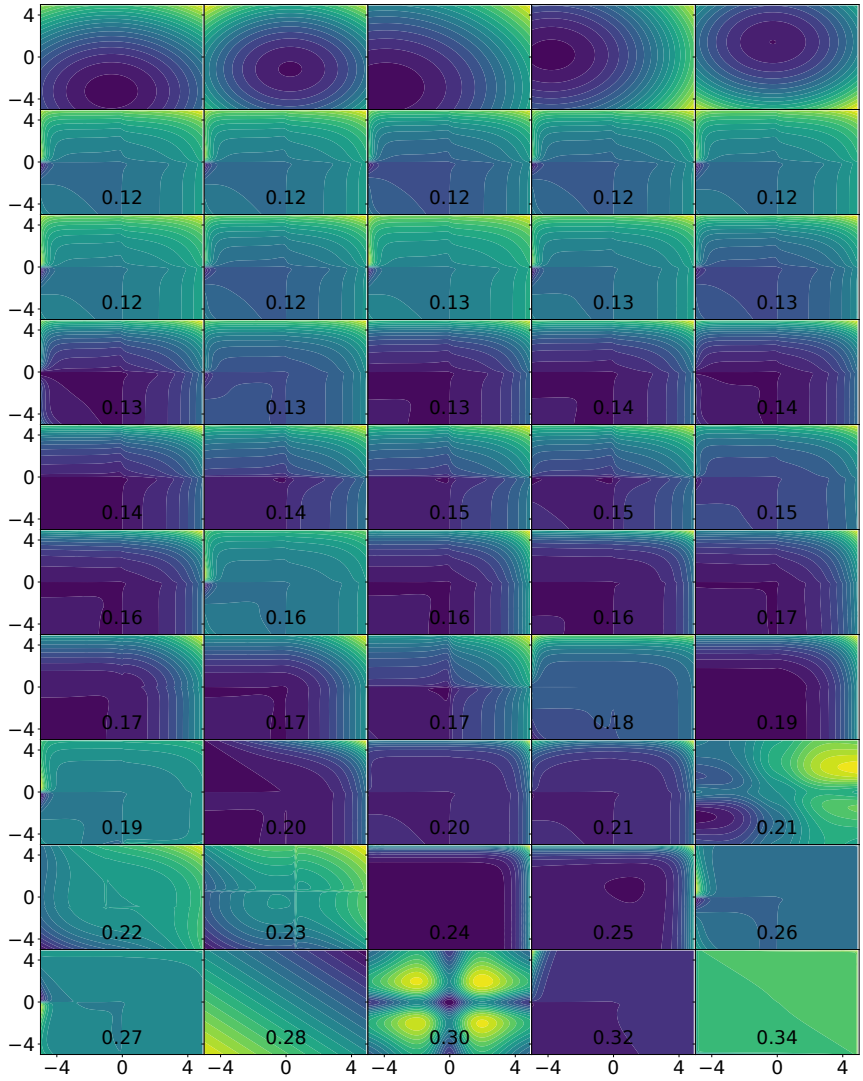


Figure 4.14: Visualization of GP-generated functions for F1 in 2-d, where the 5 BBOB instances are plotted in the first row. This is followed by 45 GP-generated functions, which are selected according to their Wasserstein distances (as indicated by the value) using a linear spacing, starting with the best (*top left*) to the worst function (*bottom right*). Figure taken from [75].

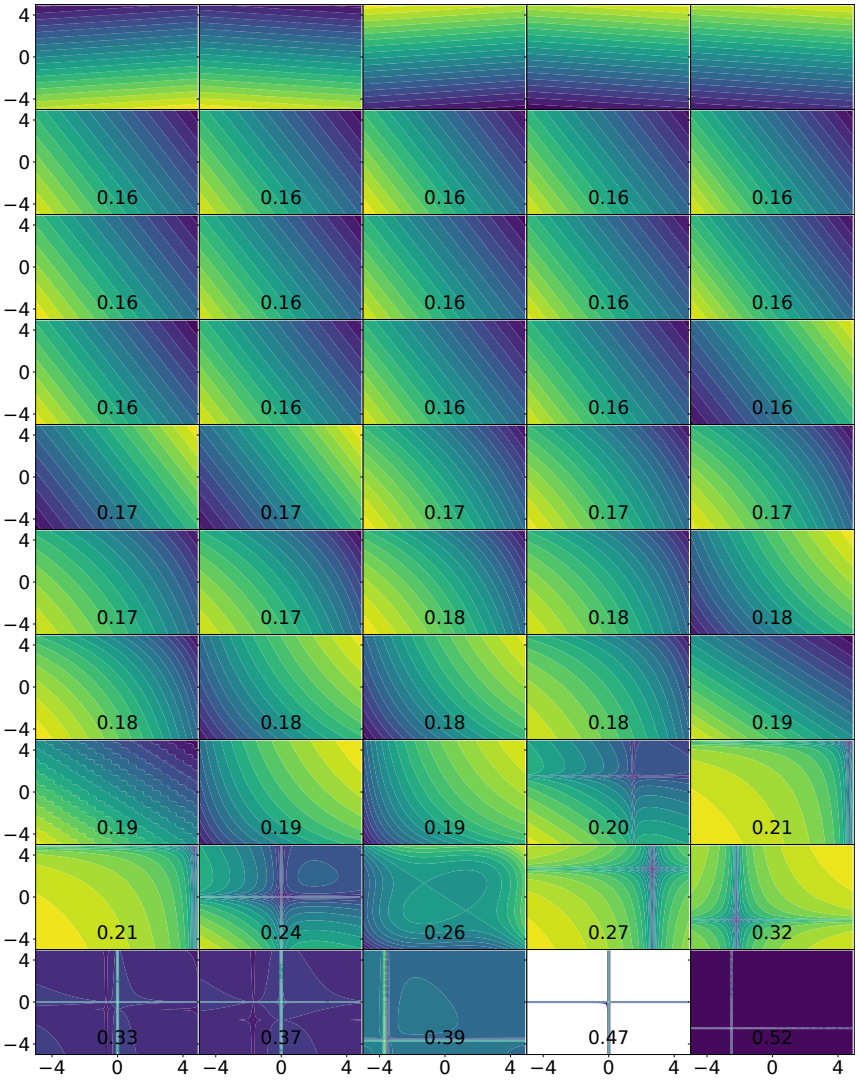


Figure 4.15: Visualization of GP-generated functions for F5 in 2-d, where the 5 BBOB instances are plotted in the first row. This is followed by 45 GP-generated functions, which are selected according to their Wasserstein distances (as indicated by the value) using a linear spacing, starting with the best (*top left*) to the worst function (*bottom right*). Figure taken from [75].

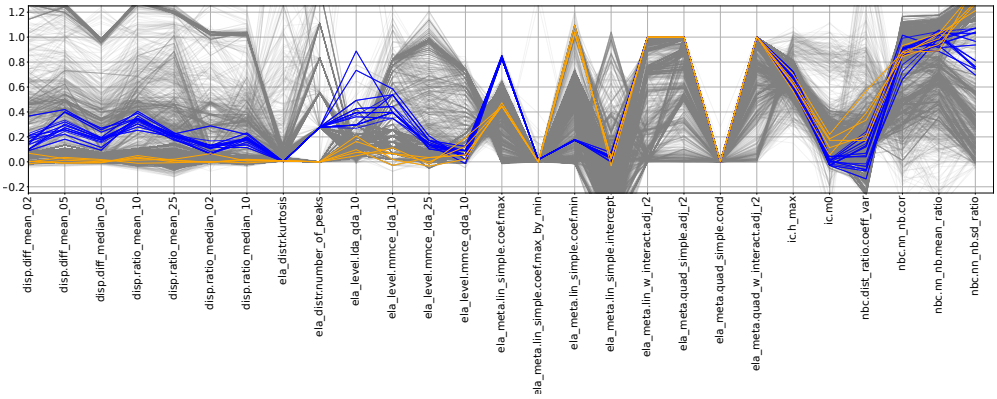


Figure 4.16: ELA features for F5 in 2-d (5 instances; *blue*), the corresponding GP-generated functions $\frac{x_0+x_1}{2}$ (*orange*), and the remaining functions (*gray*). Each line represents a bootstrapped DoE. Figure taken from [75].

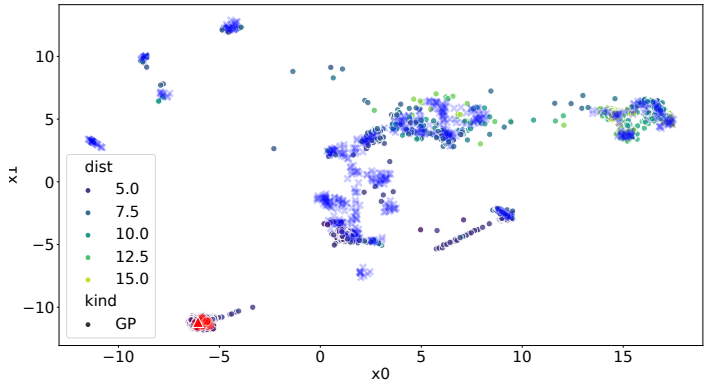


Figure 4.17: Projection of F5 and GP-generated functions to a 2-d visualization space using UMAP. Basically, the projection mapping is defined using only the BBOB instances (*cross; blue*). On one hand, the mean across five instances for the target F5 is shown (*triangle; red*). On the other hand, the GP-generated problems (*dot*) are highlighted according to their cityblock or Manhattan distance to the target, since a coloring based on Wasserstein distance is challenging. Figure taken from [75].

sualized in Figure 4.18, to gain an insight into the relevance and importance of each ELA feature for a given function. Apart from analyzing the ELA feature variances within the BBOB functions, we also attempt to relate this to the deviations within the GP-generated functions. Precisely, we consider the mean standard deviation of ELA features across all GP-generated functions and compute the absolute difference to the corresponding target BBOB functions. Based on our analysis, the deviation in some ELA features is obviously larger in the GP-generated functions than in the BBOB functions, such as `ela_meta.lin_simple.coef.max_by_min`, indicating that these ELA features might play a deciding role for the distance measurement.

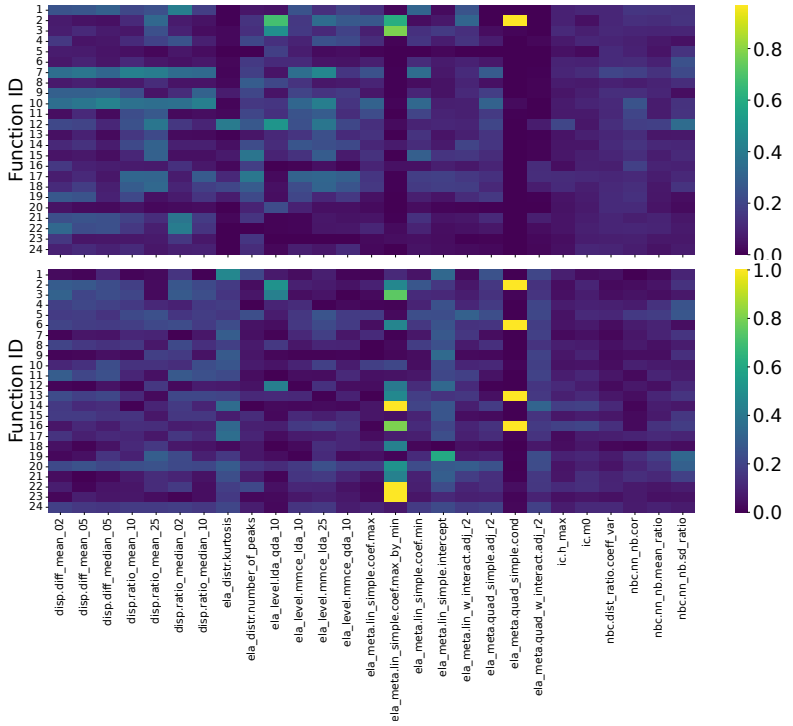


Figure 4.18: Relative standard deviation of normalized ELA features for 24 BBOB function (*top*). Absolute difference in relative standard deviation of normalized ELA features between BBOB functions and GP-generated functions (*bottom*). A lighter color represents a larger deviation, and vice versa. Figures are taken from [75].

Apart from the statistical distance metrics like Wasserstein distance, we investigate the impact of other distance metrics based on the average value of ELA feature distribution. Using the pairwise distances between BBOB instances, we compute the Kendall-tau correlation between a set of six distance metrics, namely Canberra,

cosine, correlation, Euclidean, cityblock, and Wasserstein distance. Generally, a positive correlation between distance metrics can be observed, as shown in Figure 4.19. Nonetheless, the correlations are not perfect, particularly for a comparison between the Wasserstein distance against a vector-based distance.

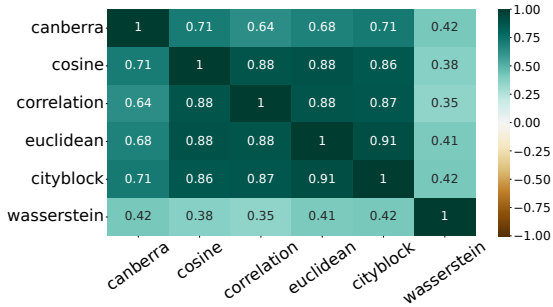


Figure 4.19: Kendall-tau correlation between six distance metrics for the BBOB instances. Figure taken from [75].

To determine which distance metric might be preferable, we additionally compare the distances between instances of the *same* BBOB function against those of *different* functions. As presented in Figure 4.20, it can be noticed that the Wasserstein distance in fact has the weakest distinguishing ability, unlike our initial intuition. On the other hand, a clear trend of smaller distances being assigned to instances of the same BBOB functions can be observed in cosine and correlation distance metric. Attempting to improve the reliability of distance measures, we analyze the differences of each ELA feature between instances of the same function and different functions. As shown in Figure 4.21, some ELA features show a rather limited difference in our comparison, such as `ela_meta.quad_simple.cond`. Consequently, such ELA features are believed to have little contribution to any distance metric, and thus, could be potentially ignored to improve the distance measures by reducing the feature dimensionality.

4.3.3 Limitations of GP-based Function Generator

Based on our analysis, we show that there is some encouraging potential in using GP to guide the function generation towards specific optimization landscape characteristics in terms of ELA features. Especially given the fact that a simple sphere function could not be accurately recreated, however, a few potential pitfalls in this approach have been highlighted that require further attention, namely:

- A proper feature selection step needs to be implemented to identify ELA features

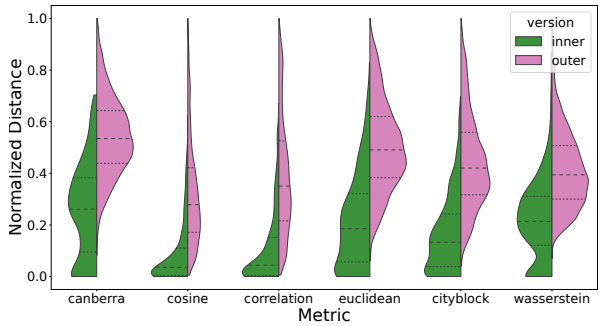


Figure 4.20: Distribution of normalized distances between instances of the same (inner) and different (outer) BBOB functions. Figure taken from [75].

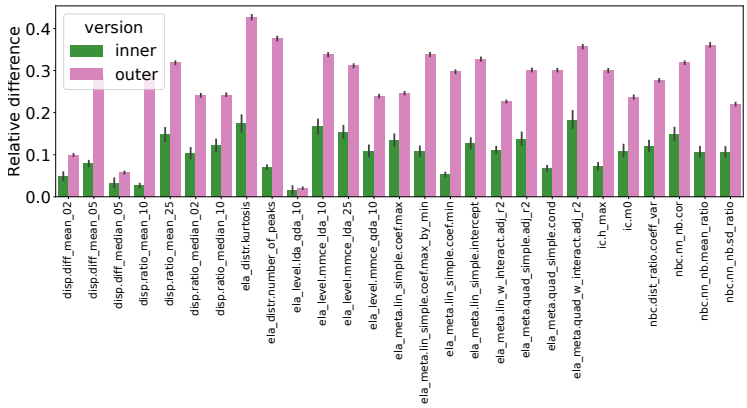


Figure 4.21: Relative differences of normalized ELA features between instances of the same (inner) and different (outer) BBOB functions. Figure taken from [75].

that can accurately capture the properties of different functions;

- Instead of an equal weighting of all ELA features, a different weighting scheme, e.g., based on feature importances, could be considered to better guide the GP search towards relevant function properties;
- An optimal fine-tuning of the GP configuration could potentially lead to the generation of much more diverse solutions; and
- The tree-cleaning operation from [134] could be further improved to increase the efficiency of GP optimization runs, by minimizing the generation of infeasible individuals.

Consequently, a significant amount of work is required to properly improve the performance and reliability of the proposed GP-based function generator. Hence, this topic is not investigated any further in this thesis. Meanwhile, we focus on identifying appropriate representative functions for real-world expensive BBO problems from a set of RGFs created using the random function generator from Section 4.1.

4.4 Conclusions

In summary, the main objective of this chapter is to investigate the potential of considering test functions created using a tree-based random function generator as representative functions for real-world expensive BBO problems. Correspondingly, an introduction of the random function generator and its adaptations are first summarized in Section 4.1. Based on our analysis, test functions with optimization landscape characteristics that are similar to real-world expensive BBO problems can be identified from a set of RGFs. More crucially, such similar RGFs are appropriate to be considered as representative functions for HPO purposes, e.g., estimating the actual performance of optimization configurations and identifying optimal optimization configurations for real-world expensive BBO problems, as discussed in Section 4.2. Beyond that, we evaluate the potential of guiding the function generation using GP towards test functions that belong to specific optimization problem classes in Section 4.3. Based on our in-depth investigations, we present the answer for *RQ2*, *RQ3*, and *RQ4* in the following:

RQ2: If none of the benchmark functions can sufficiently represent the optimization problem classes of real-world expensive BBO problems,

how to augment the benchmark functions with other test functions that are appropriate to serve as representative functions?

Using a tree-based random function generator, a diverse set of RGFs that belong to optimization problem classes different from the BBOB functions can be generated. Based on the same automotive crashworthiness optimization problems investigated in Chapter 3, a group of RGFs having similar optimization landscape characteristics can be identified for most of the crash functions. Compared to the BBOB functions, these RGFs have a smaller difference in terms of ELA features and are closely clustered to the automotive crash problems in the ELA feature space. Subsequently, these RGFs belong to the same optimization problem classes, and thus, can be considered as representative functions for the automotive crash problems. In fact, we propose to consider such RGFs with a similar optimization landscape as scalable and cheap-to-evaluate representations of real-world expensive BBO problems, e.g., for HPO purposes. We believe that RGFs with similar optimization landscape characteristics can be identified for BBO problems, provided that (i) the set of RGFs is sufficiently large and (ii) the optimization problem classes are well covered by the function generator.

RQ3: How well can we estimate the actual performance of optimization algorithms on real-world expensive BBO problems based on some cheap-to-evaluate representative functions?

Based on 24 BBOB in 20-d and two BBO algorithms, namely ModCMA and BO, we show that the optimization performances of optimization configurations are comparable between the BBOB functions and similar RGFs, where optimal configurations on the RGFs can perform well on the BBOB functions as well. In other words, the actual performance of optimization configurations on an unseen BBO problem can be estimated, by exploiting RGFs with similar optimization landscapes characteristics. Following this, RGFs can be considered as cheap-to-evaluate representations of real-world BBO problems, e.g., to estimate actual algorithm performances and to identify optimal optimization configurations. Furthermore, we believe that our approach can generalize well to different BBO problems that belong to similar optimization problem classes covered by the BBOB suite.

Meanwhile, it has been discovered that not all RGFs with similar optimization landscape characteristics are appropriate to serve as representative functions for HPO purposes. Subsequently, a selection process is implemented to iden-

tify RGFs that are sufficiently discriminative in distinguishing the performance of different optimization configurations, allowing a proper selection of optimal configurations. Moreover, instead of just one representative function, our results indicate that considering a set of representative functions could potentially improve the overall robustness in fine-tuning optimal configurations for solving BBO problems.

RQ4: How to specifically generate test functions that belong to the same optimization problem classes of real-world expensive BBO problems?

While we show that GP can be integrated into the random function generator to guide the function evolution towards specific optimization landscape characteristics in terms of ELA features, this process is not as straightforward as expected. In fact, the performance of our proposed GP-based function generator is rather disappointing, e.g., it fails to recreate a simple sphere function. Nevertheless, our investigations have identified several critical challenges, which might be helpful for future developments. For a more vigorously guiding of function evolution towards a target function, for instance, the selection of crucial ELA features and the weighting of ELA features for distance measures are two significant difficulties that must be properly addressed.

In conclusion, cheap-to-evaluate test functions that belong to the same optimization problem classes as real-world expensive BBO problems, such as automotive crashworthiness optimization, can be identified from a sufficiently large set of RGFs. Importantly, they can be considered as representative functions for HPO purposes. As such, all the core elements essential for the proposed automated optimization pipeline (Figure 1.1) are now ready for further investigations. In the following chapter, we focus on evaluating the performance of our optimization pipeline, i.e., identifying optimal configurations for real-world expensive BBO problems based on some cheap-to-evaluate representative functions.

The content of this chapter is mainly based on the author's contribution in publications [72, 73, 75].

Chapter 5

Efficiently Solving Expensive Black-Box Optimization Problems

Based on the findings in Chapter 3 and Chapter 4, here we focus on evaluating the potential of our automated optimization approach proposed for an optimal solving of real-world expensive BBO problems. Essentially, based on some cheap-to-evaluate representative functions, optimal optimization configurations w.r.t some real-world constraints can be identified for BBO problems, prior to the optimization runs using expensive function evaluations. In this regard, a detailed explanation of the proposed optimization approach is first provided in Section 5.1. For a comprehensive analysis, the performance of this optimization approach is evaluated using three state-of-the-art optimization algorithms, consisting of ModCMA, ModDE, and BO, based on two applications, namely the BBOB functions in Section 5.2 and a real-world automotive crashworthiness optimization problem in Section 5.3. Beyond that, we analyze the effectiveness of RGFs and deep NN models for the training of predictive models in Section 5.4, similar to a landscape-aware ASP context. Lastly, Section 5.5 summarizes and concludes this chapter.

5.1 Surrogate-based Landscape-aware Optimization Pipeline

An overview of the complete workflow of our automated optimization pipeline proposed for solving real-world expensive BBO problems is visualized in Figure 5.1. Essentially, optimal optimization configurations for a particular optimization problem class are first identified based on some cheap-to-evaluate representative functions, and then applied for the solving of expensive BBO problems. In general, our optimization pipeline consists of three main steps, namely identifying representative functions that are appropriate for HPO purposes (Step 1), searching for optimal configurations using HPO (Step 2), and optimally solving of the BBO problems using expensive function evaluations (Step 3). Moreover, real-worlds constraints are additionally considered during the HPO process for an optimal solving of expensive BBO problems, such as the wall-clock time and computational resources allocated for optimization runs. Following this, the optimization configurations identified using HPO are optimal *specifically* for this optimization problem class and optimization setup, e.g., function evaluation budget and performance metric. For a different problem class and/or setup, a rerun of the pipeline is necessary to newly identify optimal configurations. In the following, the workflow of our optimization pipeline is described in detail:

Input: A set of DoE samples of the expensive BBO problem instance to-be-solved is required as input for our pipeline. Similar to Section 3.1.1, while any sampling strategy can be used in practice, the Sobol' sampling is preferred for a reliable ELA feature computation. Apart from that, some real-world constraints that are relevant for the optimization runs can be defined as well, to ensure an efficient solving of the BBO problem instance using expensive function evaluations. For automotive crashworthiness optimization problems, for instance, the wall-clock time allocated for optimization runs, the computational resources available for FE simulations, and the availability of commercial solver licenses for potential parallelization are some typical real-world constraints.

Appropriate representative functions (Step 1): Essentially, using the approach proposed in Section 3.1.1, the optimization landscape characteristics of the BBO problem instance are quantified using some ELA features. Next, these ELA features are compared against those of some RGFs created using the tree-based random function generator introduced in Section 4.1. Subsequently, a set of RGFs with similar optimization landscape characteristics in terms of ELA features can

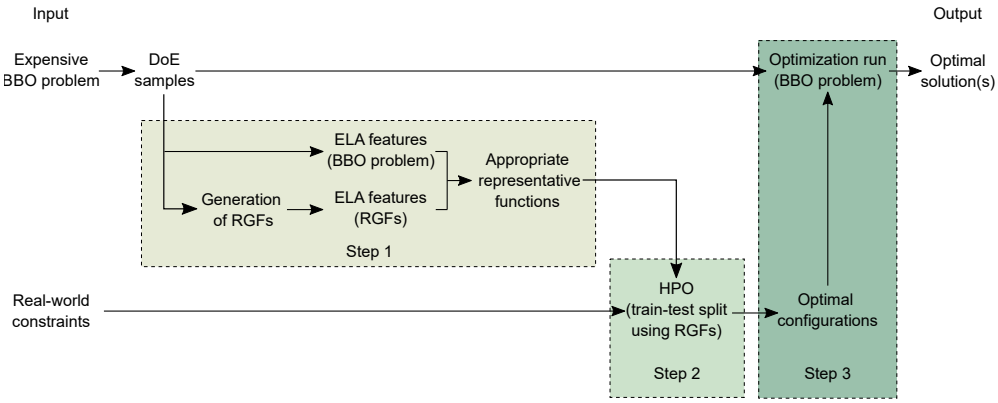


Figure 5.1: Overview of our surrogate-based landscape-aware optimization pipeline proposed for real-world expensive BBO problems, consisting of altogether three main steps. By exploiting some cheap-to-evaluate RGFs with similar optimization landscape characteristics in terms of ELA features as representations of the BBO problem instance (Step 1), optimal optimization configurations can be identified using HPO (Step 2). Subsequently, the optimal optimization configurations can be applied to solve the BBO problem instance using expensive function evaluations (Step 3). Furthermore, real-worlds constraints, such as wall-clock time and computational resources, are taken into consideration for an efficient solving of the BBO problem instance. Figure taken from [74] with modifications.

be identified, similar to Section 3.1.1. Lastly, the selection process described in Section 4.2.4 is employed to identify similar RGFs that are appropriate to serve as representative functions for HPO in the next step.

HPO using training-testing split (Step 2): As shown in Section 4.1.1 and Section 4.2, considering just one single representative function for HPO might not be robust to identify optimal optimization configurations for solving the BBO problem instance. Similar to the over-fitting problem in a ML context, using only one representative function could eventually lead to fine-tuning of optimization configurations towards this particular function. For a better generalization across a particular optimization problem class, we propose to consider several representative functions for HPO. Subsequently, instead of just considering the RGF having the smallest difference in ELA features, the next few RGFs are included in the set of representative functions as well.

To further minimize potential over-fitting problems, the set of representative functions are divided into two groups, consisting of training and testing functions. During HPO, the average performance of each individual configuration is evaluated based on all training functions, where the objective values are rescaled using min-max scaling according to the global optimum and worst DoE sample. Once the HPO process is completed, the top 10% configurations are evaluated again using the testing functions and assigned with a new rank based on their performances. Subsequently, optimal optimization configurations or the best configuration for this particular optimization problem class can be identified. While further investigations are required to determine an ideal ratio of training-testing functions, two training functions and one testing function are considered in this thesis, which can perform well in our preliminary testing, while being computationally affordable.

Optimal solving (Step 3): In the last step, optimal optimization configurations or the best configuration identified can be applied for an efficient solving of the BBO problem instance using expensive function evaluations. It is worth noting that the optimal configurations applied are kept unchanged throughout the optimization runs on expensive BBO problems.

Output: Optimal solution(s) for the expensive BBO problem instance.

Despite of the fact that an additional computational cost is necessary for the HPO

processes in our approach, e.g., to select appropriate representative functions and to identify optimal optimization configurations, we argue that:

1. The optimal optimization configurations identified can improve the overall optimization efficiency in solving expensive BBO problems. In other words, our approach offers an alternative to cheaply fine-tune optimization configurations that can optimally solve expensive BBO problems w.r.t. some real-world constraints, e.g., finding better solutions and/or using less expensive function evaluations;

2. The total additional cost is relatively insignificant compared to the optimization runs on BBO problems using expensive function evaluations. A qualitative comparison of the overall computational effort between a HPO process based on representative functions and solving a real-world expensive BBO problem is provided in Table 5.1, using automotive crashworthiness optimization as a representative example; and

3. The same representative functions and optimal configurations identified can be considered for future BBO problems that belong to the same optimization problem classes, meaning that the HPO processes can be potentially skipped.

Table 5.1: Qualitative comparison of the computational effort required for the HPO in our approach and solving a real-world expensive BBO problem, using automotive crashworthiness optimization as an example. For other BBO problems, the computational cost can be different depending on the problem definition and optimization setup. Nevertheless, a proper run time analysis is necessary for a precise quantification. Table taken from [74].

Estimation	**HPO**	**Automotive crash optimization**
Function evaluation	Representative function	FE simulation
Cost per function evaluation	Within seconds (using 1 CPU core)	$>$ 24 hours (using 192 CPU cores)
Total wall-clock time	A few hours	Several days / weeks
Overall effort	Low / Cheap	High / Expensive

5.2 Fine-Tuning of Optimization Configurations for BBOB Functions

In the first step, the performance of the optimization pipeline introduced in Section 5.1 is evaluated based on the BBOB suite, which covers a wide range of optimization

problem classes. The corresponding experimental setup is summarized in Section 5.2.1, followed by the result discussion in Section 5.2.2.

5.2.1 Experimental Setup

- Altogether 24 BBOB functions of the first instance in 20-d;
- Representative functions are selected from a large set of 10 000 RGFs;
- The ELA features are computed based on a DoE of $20 \cdot d$ samples generated using the Sobol' sampling;
- In this investigation, we focus on fine-tuning the configurations of ModCMA, ModDE, and three BO variants, consisting of BO, TuRBO-1, and TuRBO-m, using the hyperparameter search spaces summarized in Table 5.2, Table 5.3, and Table 5.4, respectively;
- For ModCMA and ModDE, each optimization run is allocated with a budget of $100 \cdot d$ function evaluations and 20 repetitions using different random seeds. On the other hand, only a budget of 250 evaluations is allocated for the BO variants, since the time-complexity of training GPR models increases exponentially with sample size;
- Regarding HPO, two optimizers are employed for ModCMA and ModDE, namely TPE available in `HyperOpt` and SMAC, using a fixed budget of 500 configuration evaluations. Given that the BO hyperparameter search spaces are relatively small, only TPE and 50 evaluations are considered for all BO variants; and
- The performance of optimal ModCMA and ModDE configurations identified is compared against the default configuration, Single Best Solver (SBS), and Virtual Best Solver (VBS). Meanwhile, we only compare against the default configuration for all BO variants, since they are relatively computationally intensive. In this context, the configuration performances are evaluated using the AUC metric from Section 4.2.1.

To fairly evaluate the performance of optimal optimization configurations identified using our approach, we consider the following configurations as a comparison baseline, consisting of the default configuration, SBS, and VBS:

Default configuration: The readily available configuration in its proposed implementation. For practitioners unfamiliar with HPO, the default configuration is

Table 5.2: Overview of the ModCMA hyperparameters considered for HPO. The default configuration is highlighted in bold, where the default learning rates are automatically computed w.r.t. other hyperparameters. Symbol: $\mathbb{Z}$ for integer, $\mathbb{R}$ for continuous variable, and **C** for categorical variable. Table taken from [74].

Num.	Hyperparameter	Type	Search space
1	Number of children	$\mathbb{Z}$	{ 5, . . . , 50 } (**4** + ⌊(**3ln(d)**)⌋)
2	Number of parent (as ratio of children)	$\mathbb{R}$	[0.3, 0.5] (**0.5**)
3	Initial standard deviation	$\mathbb{R}$	[0.1, 0.5] (**0.2**)
4	Learning rate step size control	$\mathbb{R}$	[0.0, 1.0]
5	Learning rate covariance matrix adaptation	$\mathbb{R}$	[0.0, 1.0]
6	Learning rate rank-μ update	$\mathbb{R}$	[0.0, 0.35]
7	Learning rate rank-one update	$\mathbb{R}$	[0.0, 0.35]
8	Active update	**C**	{ True, **False** }
9	Mirrored sampling	**C**	{ **none**, 'mirrored', 'mirrored pairwise' }
10	Threshold convergence	**C**	{ True, **False** }
11	Recombination weights	**C**	{ '**default**', 'equal', '1/2^lambda' }

Table 5.3: Overview of the ModDE hyperparameters considered for HPO, where the default configuration is highlighted in bold. Symbol: $\mathbb{Z}$ for integer, $\mathbb{R}$ for continuous variable, and **C** for categorical variable. Table taken from [74].

Num.	Hyperparameter	Type	Search space
1	Population size	$\mathbb{Z}$	{ 5, . . . , 50 } (**4** + ⌊(**3ln(d)**)⌋)
2	Weighting factor F	$\mathbb{R}$	[0.0, 1.0] (**0.5**)
3	Crossover constant CR	$\mathbb{R}$	[0.0, 1.0] (**0.5**)
4	Base vector	**C**	{ '**rand**', 'best', 'target' }
5	Reference vector	**C**	{ **none**, 'pbest', 'best', 'rand' }
6	Number of differences	**C**	{ **1**, 2 }
7	Weighted F	**C**	{ True, **False** }
8	Crossover method	**C**	{ '**bin**', 'exp' }
9	Eigenvalue transformation	**C**	{ True, **False** }
10	Adaptation of F	**C**	{ **none**, 'shade', 'shade-modified', 'jDE' }
11	Adaptation of CR	**C**	{ **none**, 'shade', 'jDE' }
12	Use JSO caps for F and CR	**C**	{ True, **False** }

Table 5.4: Overview of the hyperparameters of three BO variants considered for HPO, where the default configuration is highlighted in bold. For investigations based on automotive crash problems (Section 5.3), the DoE sample size is fixed and excluded from HPO. Symbol: $\mathbb{Z}$ for integer, $\mathbb{R}$ for continuous variable, and **C** for categorical variable. Table taken from [74].

Variant	Hyperparameter	Type	Search space
BO	DoE size (as ratio of total budget)	$\mathbb{R}$	[0.1, 0.9] (**0.5**)
	Kernel of GPR models	**C**	{ Matérn 3/2, **Matérn 5/2**, RBF }
	Acquisition function	**C**	{ **EI**, PI, LCB }
TuRBO-1	DoE size (as ratio of total budget)	$\mathbb{R}$	[0.1, 0.9] (**0.5**)
	Infill points	$\mathbb{Z}$	{ 1, . . . , 10 } (**1**)
TuRBO-m	DoE size (as ratio of total budget)	$\mathbb{R}$	[0.1, 0.9] (**0.5**)
	Number of trust regions	$\mathbb{Z}$	{ 2, . . . , 6 } (⌊(**D/5**)⌋)
	Infill points	$\mathbb{Z}$	{ 1, . . . , 10 } (**1**)

commonly considered and simply taken off-the-shelf for their applications. In line with our research motivation, i.e., assisting practitioners in automatically fine-tuning optimization configurations, we consider this configuration as our primary comparison reference;

SBS: The configuration that can perform well on average across 24 BBOB functions. Essentially, SBS is identified according to the mean configuration performance on all BBOB functions. In this thesis, the performance of SBS serves as our secondary comparison reference; and

VBS: The best performing configuration for a particular BBOB function. Fundamentally, the performance of VBS is treated as the lower bound.

Typically for ASP in literature, both SBS and VBS are identified from an algorithm portfolio, based on an evaluation of a limited set of algorithms. Nevertheless, a complete evaluation of all possible configurations within the huge hyperparameter search spaces is computationally infeasible, e.g., for ModCMA (Table 5.2). Following this, both SBS and VBS are identified through HPO using TPE and the aforementioned experimental setup. Considering the stochastic nature of TPE, configurations that could outperform the VBS identified might exist, but are not discovered during HPO. While this can be effectively avoided by performing the HPO for a few repetitions, we only consider one repetition here to minimize the overall computational cost. It is worth mentioning that such SBS and VBS are usually not available for real-world expensive BBO problems.

5.2.2 Empirical Assessment of Optimization Performances

The optimization performance for different ModCMA configurations on 24 BBOB functions in 20-d is visualized in Figure 5.2. Compared to the default configuration, optimal configurations identified using our approach generally have a smaller AUC for most of the BBOB functions, even for complex functions like F16 and F23. This indicates that configurations that are superior than the default configuration can be identified based on some representative functions. Meanwhile, the optimal configurations can outperform the SBS on some of the BBOB functions, while being equally competitive on several remaining BBOB functions. This is especially motivating for real-world applications, where such SBS is usually not readily available. Nevertheless, some weaknesses in our approach can be identified that calls for further improvements, e.g., struggling on a few BBOB functions. For instance, the performance of optimal

configurations identified on F10 is obviously worse than the default configuration and SBS.

Basically, a similar tendency can be observed for ModDE, as presented in Figure 5.3, where our optimal configurations can perform well on many of the BBOB functions. Consequently, our approach has a promising potential in optimally fine-tuning algorithm configurations for the BBOB functions, which can work well with different optimization algorithms. Given the fact that the optimal ModCMA configurations can outperform the default configuration and compete against the SBS on a larger set of BBOB functions, our approach seems to be less reliable in consistently identifying well-performing ModDE configurations. We suspect that the relatively weak performance of ModDE might be partly due to its stochastic nature, e.g., randomness in population initialization, which requires further investigations.

Lastly, the optimization performances of different BO variants are summarized in Figure 5.4, where the optimal configurations can outperform the default configuration on most of the BBOB functions. Remarkably, a smaller DoE sample size is preferred in all optimal configurations identified using HPO, which might be partly related to the computation of AUC metric.

For a fair and precise evaluation, the performance of different configurations on the BBOB functions are statistically analyzed further. Precisely, the performance of optimal configurations identified using our approach is compared against the default configuration and SBS based on the Wilcoxon signed-rank test in `scipy` [149], using the alternative hypothesis *the performance of optimal configurations identified by our approach is weaker*. As shown in Figure 5.5 for ModCMA, ModDE, and three BO variants, the optimal configurations identified indeed can outperform the default configuration on many BBOB functions, and even compete against the SBS in some cases, in line with our previous interpretations. While not being the focus of this investigation, SMAC seems to be slightly advantageous in identifying configurations with a better performance, in comparison to TPE. Nonetheless, the effectiveness of our approach is rather lackluster on some BBOB functions, such as F7, F10, and F11, which might be due to the processing of ELA features, as discussed in Section 4.1.1. In summary, our analysis shows that our approach proposed for an optimal fine-tuning of optimization configurations based on some representative functions has an attractive potential in generalizing across different optimization problem classes covered by the BBOB suite.

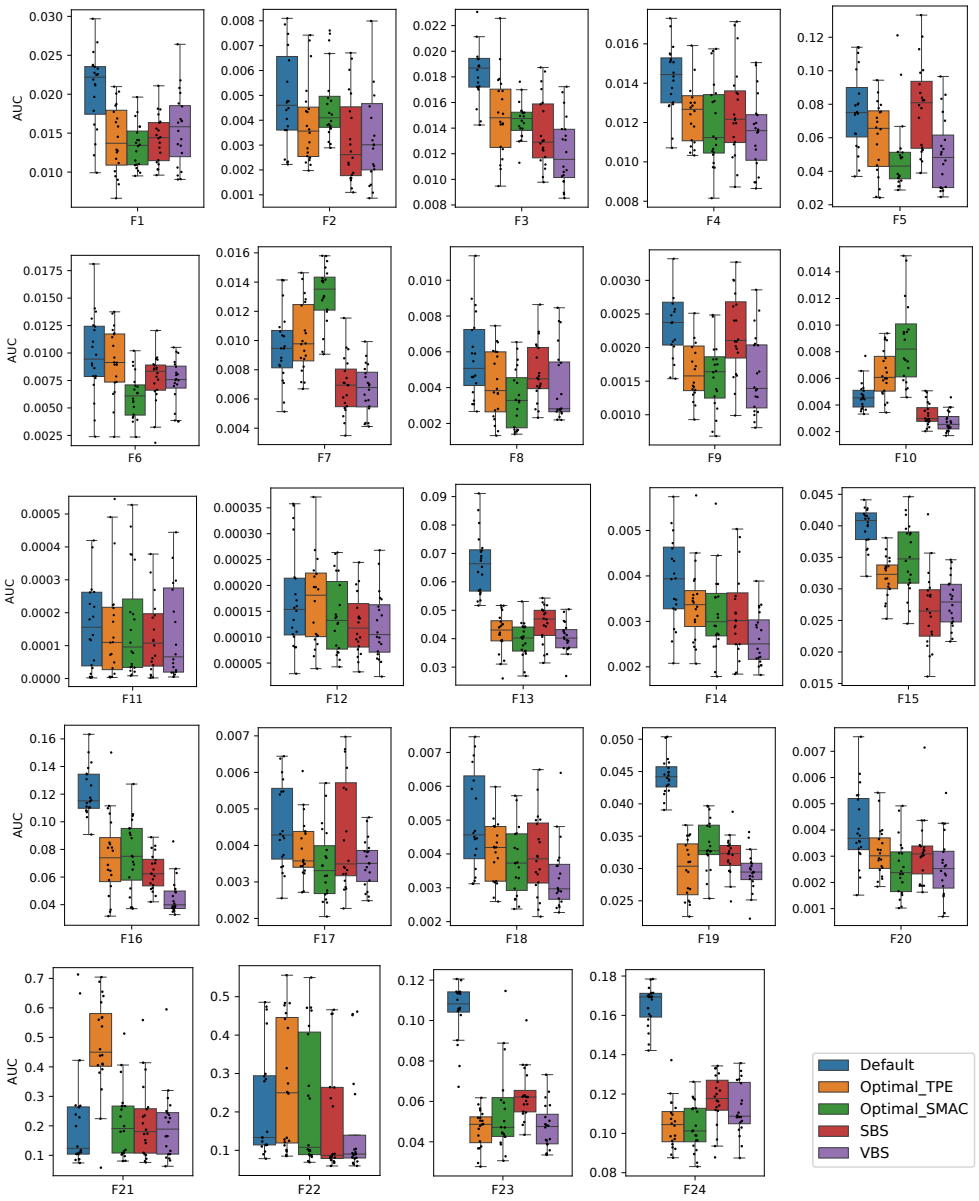


Figure 5.2: Performance of different ModCMA configurations on 24 BBOB functions in 20-d, using a repetition of 20 times for each configuration. The configuration having a smaller AUC is better. *Legend:* Default configuration, optimal configurations identified using TPE or SMAC, SBS, and VBS. Figures taken from [74].

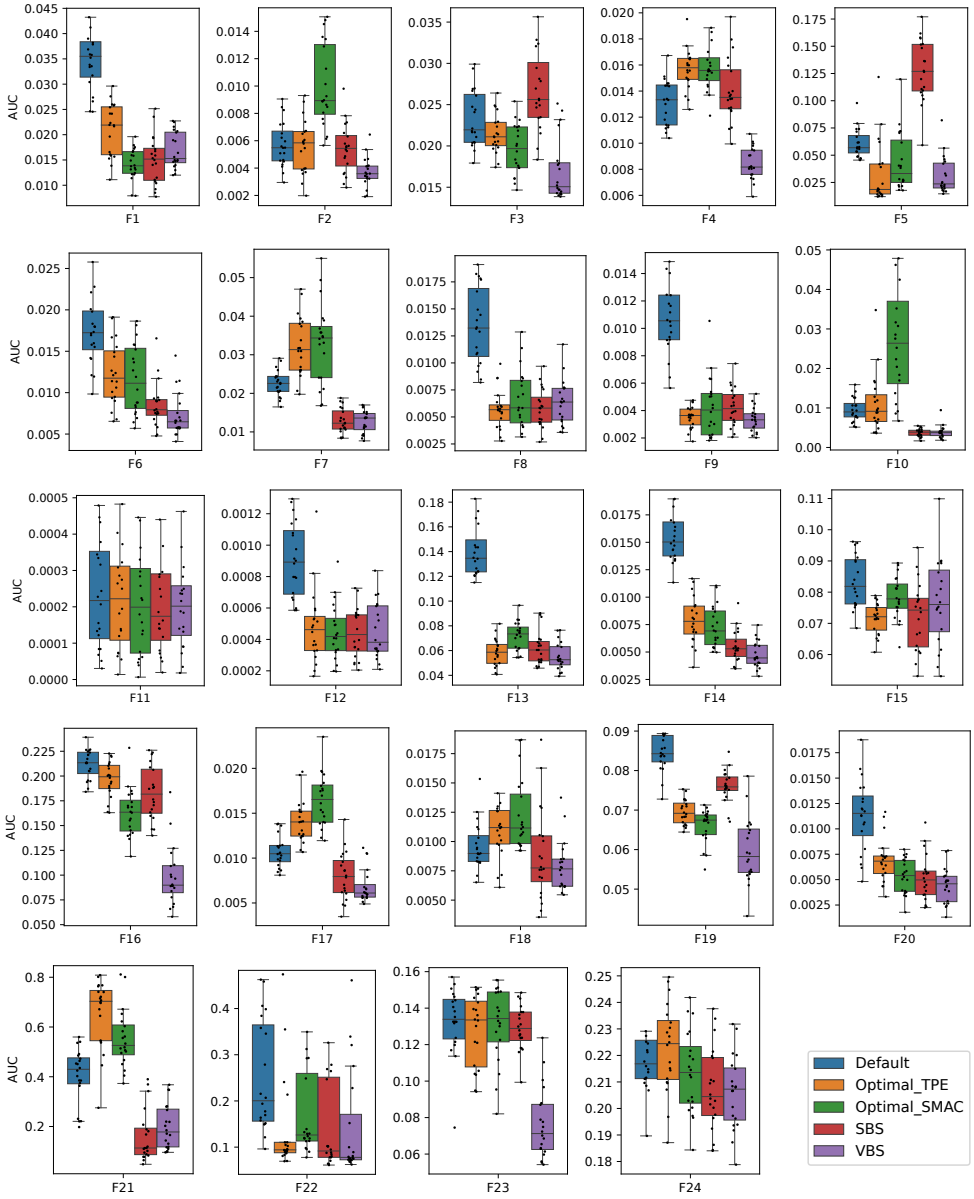


Figure 5.3: Performance of different ModDE configurations on 24 BBOB functions in 20-d, using a repetition of 20 times for each configuration. The configuration having a smaller AUC is better. *Legend:* Default configuration, optimal configurations identified using TPE or SMAC, SBS, and VBS. Figures taken from [74].

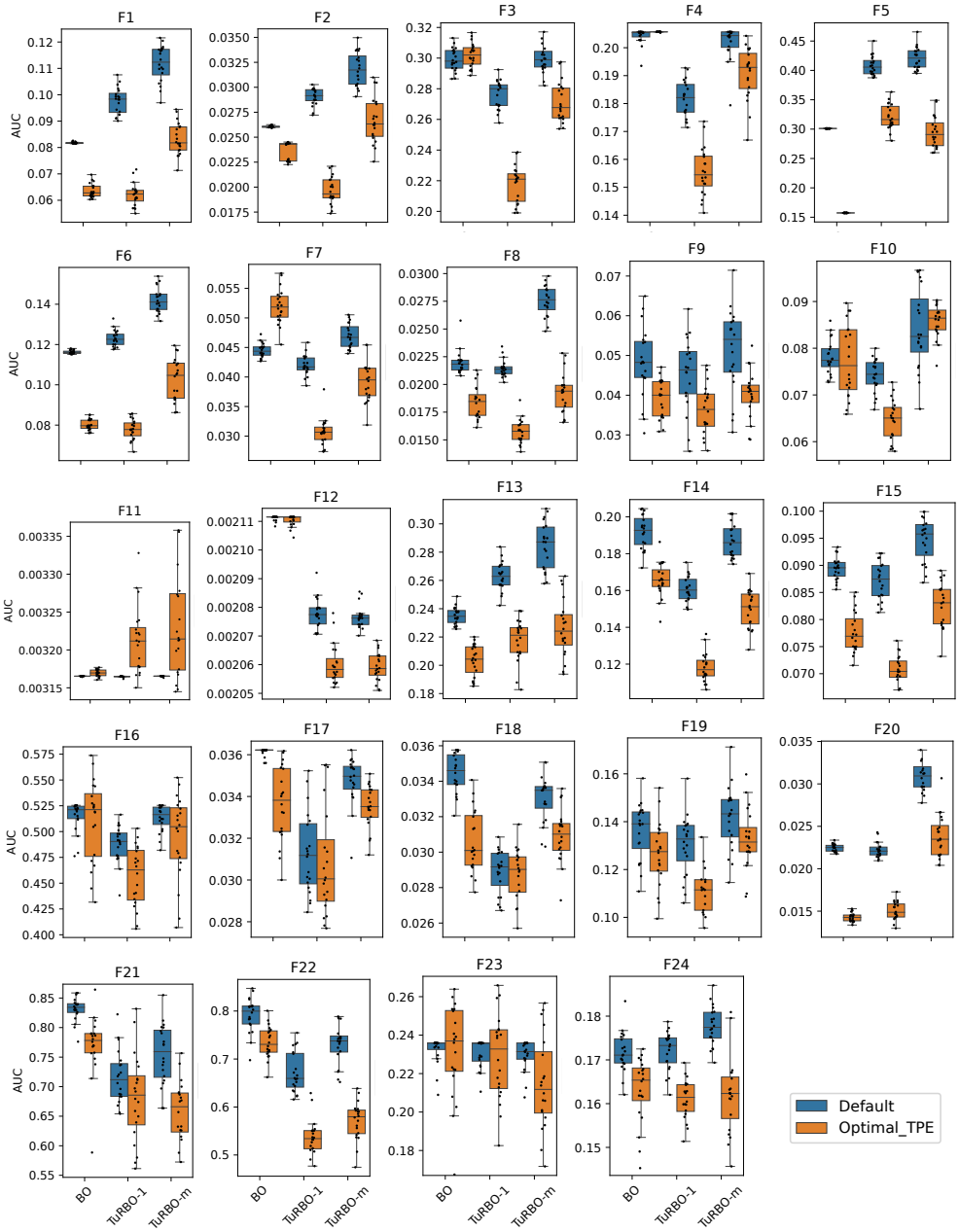


Figure 5.4: Performance of different configurations for three BO variants on 24 BBOB functions in 20-d, using a repetition of 20 times for each configuration. The configuration having a smaller AUC is better. *Legend:* Default configuration and optimal configurations identified using TPE. Figures taken from [74].

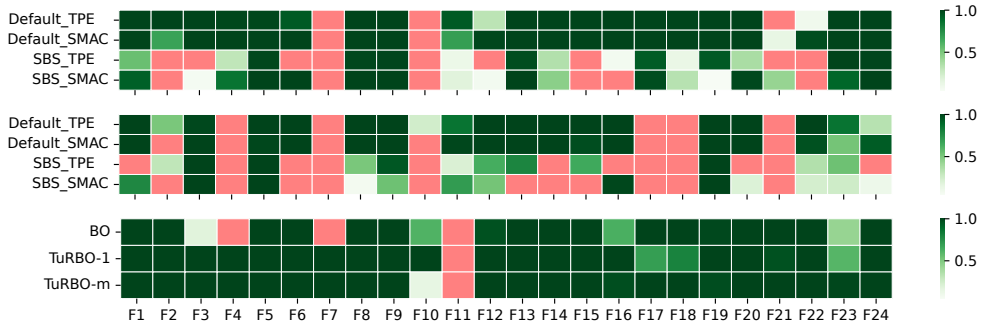


Figure 5.5: Pairwise performance comparison between the optimal configurations identified using TPE or SMAC in our approach against the default configuration and SBS on 24 BBOB functions in 20-d for ModCMA (*top*), ModDE (*middle*), and three BO variants (*bottom*). Based on the Wilcoxon signed-rank test, a red color indicates that there is a statistically significant evidence to support the alternative hypothesis *the performance of our optimal configurations is weaker*, with a p-value smaller than 0.05. On the other hand, a green color indicates that our optimal configurations are equally competitive or better, where a darker green color or a higher p-value indicates that the hypothesis is less likely to be rejected. *Legend:* Comparison of the default configuration or SBS against our optimal configurations identified using TPE or SMAC. Figure taken from [74].

5.3 Real-World Expensive Automotive Crashworthiness Optimization

Apart from the BBOB functions, we also evaluate the potential of our optimization pipeline proposed in Section 5.1 for an optimal solving of real-world expensive BBO problems. Precisely, our investigation is based on an automotive crashworthiness optimization problem for a side crash scenario. In this regard, an overview of the corresponding FE crash model is first provided in Section 5.3.1. Based on this FE model, two load cases are considered in our investigation, as discussed in Section 5.3.2 and Section 5.3.3.

5.3.1 FE Submodel for Automotive Side Crash

Within the scope of this thesis, we focus on the side crash scenario against a rigid pole as a representative automotive crash problem, in line with the current trend in developing new vehicle designs for BEVs, as described in Section 2.4. In fact, this side crash scenario is often considered as critical for BEVs, since both the passengers

and battery cells must be sufficiently protected from serious damage, leading to an increased design complexity. While vehicle design problems cover a broad scope of research topics, within the scope of this thesis we focus on optimizing the structural crashworthiness of vehicle body's frame, also known as Body-in-White (BIW). To minimize the overall computational expenses, we consider a FE submodel developed based on state-of-the-art modeling of BEVs, as shown in Figure 5.6, which has a comparable crash kinematics to a full vehicle FE model. Generally, different aspects of the FE simulation can be summarized as follows:

- The FE submodel consists of altogether 95 000 elements;
- The vehicle is impacted from the sideways at a small angle against a rigid pole at a speed of 32 km/h, similar to the guidelines provided by the European New Car Assessment Programme (EuroNCAP) [33];
- Regarding the computational effort, each FE simulation is terminated after 50 ms and requires a solving time of about 6 minutes. Here, the crash simulations are solved using the explicit solver `LS-DYNA` [70] in its MPP version, which is distributed across 64 CPU cores using high-performance computing clusters;
- In total, 18 thicknesses of rocker panels (made of aluminum alloys) and supporting beams (made of high-strength steels) are the design variables considered for optimization;
- We focus on optimizing the design variables w.r.t. the mass m and structural performance of a vehicle design. Precisely, the structural performance is quantified using the maximum structural deformation towards the battery compartment, which is also known as intrusion u. In practice, finding an optimal trade-off between theses conflicting objectives is challenging, e.g., a smaller intrusion can be achieved by increasing the component thicknesses, which in turns lead to a larger mass; and
- As a baseline, the vehicle design located at the center of design space has a mass of 32.4 kg.

5.3.2 Load Case A – Single Pole Impact

In the first load case, only one impact position is considered, and hence, we call *single* pole impact problem. Precisely, the vehicle is impacted from the side against a rigid

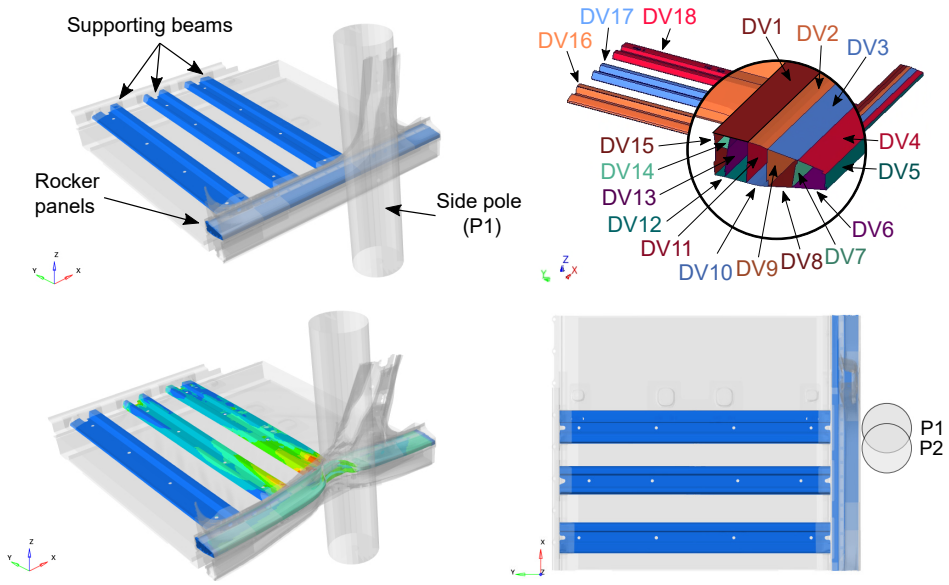


Figure 5.6: Overview of the FE submodel for an automotive side crash against a rigid pole. In BEVs, the battery compartment is commonly installed between the rocker panels on both sides and underneath the supporting beams. *Top left:* The thicknesses of rocker panels and supporting beams are considered as design variables for optimization, as highlighted in blue color. *Bottom left:* An example of FE simulation result, showing the structural deformation after impact and the corresponding plastic strain distribution. *Top right:* The 18 design variables in rocker panels (`DV1` – `DV15`) and supporting beams (`DV16` – `DV18`) considered for optimization, as highlighted using different colors. *Bottom right:* Top view of the FE submodel, showing two possible impact positions, labeled as `P1` and `P2`, by placing the rigid pole differently. Figure taken from [74].

pole positioned at `P1` (Figure 5.6). For this problem, the optimization objective f_{opt} is defined using Equation 5.1.

$$\begin{aligned} \min f_{opt} &= m' + u' \,, \\ m' &= \frac{m - m_{min}}{m_{max} - m_{min}} \,, \\ u' &= \frac{u - u_{min}}{u_{max} - u_{min}} \,, \end{aligned} \tag{5.1}$$

where m' and u' represents the min-max normalized mass and intrusion according to the minimum and maximum of DoE samples. Due to the fact that FE simulations are expensive, only BO is considered for solving the automotive crash problems, which suites well for optimization problems with a limited function evaluation budget. Particularly, we focus on the standard BO, which can outperform the TuRBO variants in our preliminary testing. In brief, the experimental setup consists of an initial DoE of fixed 400 samples (roughly $20 \cdot d$) for the ELA feature computation and training of GPR models in BO, 30 resamplings using FE simulations, and five repetitions using different random seeds, while the remaining setup is similar to Section 5.2.1.

Unlike our previous investigation based on the BBOB functions, however, the SBS and VBS are not available to serve as a comparison baseline for our crash optimization. On the contrary, here the performance of RSM and SRSM are considered as our primary benchmark references, which are described in detail in the following:

RSM: Using a set of DoE samples, seven types of surrogate model are trained to approximate the true crash functions, consisting of polynomial function, support vector regressor, RF, gradient boosting, dense NN, GPR, and k-nearest neighbour [94]. To further improve the fitting quality of surrogate models, the corresponding hyperparameters are fine-tuned using TPE based on a similar framework in [63] and a five-folds cross-validation. Subsequently, based on the surrogate model with the best fitting quality, ModCMA is then applied to identify the global optimum of the approximated function; and

SRSM: In short, we utilized a similar SRSM optimization approach as explained in [62, 105, 125, 126], where the response surfaces are trained in the same way as in RSM. For a fair comparison, the same total function evaluation budget, i.e., the sum of DoE samples and resamplings considered for BO, is equally divided across iterations. To improve the efficiency of SRSM, DoE samples from previous iterations that are within the subregion of current iteration are additionally

taken into account for the construction of response surfaces. Considering the DoE sample size per iteration w.r.t. the problem dimensionality, five iterations are considered for SRSM in this investigation.

The optimization results using different optimization approaches for the single pole impact problem are summarized in Figure 5.7. Generally, better vehicle designs with lower objective values can be identified using RSM, SRSM, and BO, in comparison to the classical one-shot optimization approach. Among them, both SRSM and BO outperform RSM in finding better vehicle designs, with the performance of SRSM being the best. While the results are not included here, no significant improvement can be observed in the optimization performance of RSM in our followed-up testing, despite using an additional 200 DoE samples (+50%). Subsequently, RSM indeed struggles to accurately approximate the highly nonlinear crash functions for an optimal optimization. Meanwhile, this indicates the benefits of using an iterative optimization approach to solve complex automotive crash problems, based on the fact that better solutions can be clearly found using SRSM. On the other side, compared to the default BO configuration, a slightly better optimization performance can be achieved using our optimal BO configuration w.r.t. the best-found solution (roughly −2%) and AUC metric (roughly −3%). In this regard, a faster convergence speed at the beginning of optimization runs can be observed for the optimal BO configuration.

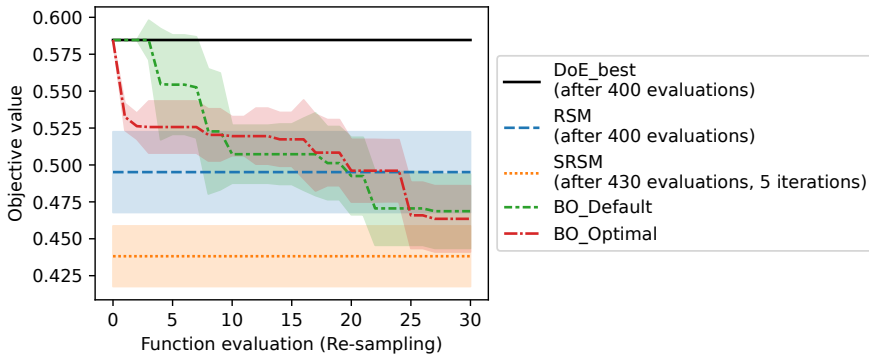


Figure 5.7: Performance comparison between one-shot optimization (*black*), RSM (*blue*), SRSM (*orange*), default BO configuration (*green*), and optimal BO configuration identified using our approach (*red*) for the single pole impact problem (Equation 5.1). While the best-found solution is presented for RSM and SRSM, the best-so-far solution during optimization is shown for BO, starting with the best DoE sample. The median performance and standard deviation over five optimization repetitions are reported. A smaller objective value and/or AUC is better. Figure taken from [74].

Focusing on the best optimization run among all repetitions, i.e., the run with the smallest objective value, we delve into analyzing the vehicle designs w.r.t. the design variables. As shown in Figure 5.8, a general tendency can be observed for most vehicle designs, where thick rocker panels for regions close to the impact (`DV1` – `DV8`) and thin supporting beams (`DV16` – `DV18`) are preferred. For the remaining design variables on the inner side of rocker panels (`DV9` – `DV15`), the thicknesses are relatively flexible. Beyond that, the vehicle designs identified by the optimization algorithms belong to different optima. Using RSM as a baseline, the vehicle design found by SRSM has a smaller intrusion (−3.7 mm), but slightly increased mass (+0.1 kg). On the other side, while the default BO configuration reduces intrusion by increasing the thicknesses of rocker panels, our optimal BO configuration attempts to minimize both vehicle mass and intrusion. Correspondingly, the vehicle design provided by the default BO configuration has a smaller intrusion (−9.4 mm) at the cost of a larger mass (+1.5 kg). Meanwhile, the other way round can be achieved using our optimal BO configuration, namely a smaller mass (−0.6 kg), yet a slightly larger intrusion (+1.5 mm).

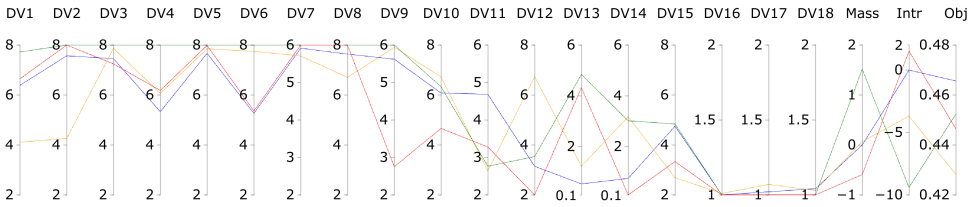


Figure 5.8: Comparison of vehicle designs for the single pole impact problem, focusing on the best optimization run using RSM (*blue*), SRSM (*orange*), default BO configuration (*green*), and optimal BO configuration identified using our approach (*red*). The upper and lower bound of the design space (thickness in mm) are shown at the top and bottom row. Using RSM as a baseline, the relative differences in mass (kg) and intrusion (mm) are shown, while the optimization objective values (Equation 5.1) are included in the last column. Figure taken from [74].

The structural deformation for different vehicle designs is illustrated in Figure 5.9, again focusing on the best optimization run. In line with our previous observations, for instance, a greater intrusion can be indeed observed in the vehicle design identified using our optimal BO configuration, compared to the default BO configuration.

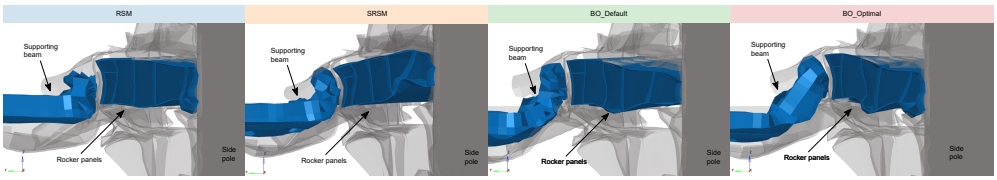


Figure 5.9: Structural deformation of the FE submodel for the single pole impact problem. The cross-section of different vehicle designs identified using RSM, SRSM, default BO, and optimal BO configuration are shown. The same setting is applied for all simulation snapshots, such as deformation scaling, view position, and time frame. Figure taken from [74].

5.3.3 Load Case B – Multi-Pole Impact

In real-world situation, a side impact can happen at any position alongside the vehicle body, which is challenging to be accurately pinpointed. To improve the overall robustness of a vehicle design, a BIW is typically optimized w.r.t. *multiple* pole impact positions, or we also call *multi*-pole impact problem. Nonetheless, optimizing vehicle designs for multi-pole impact problem is challenging, since the structural performance of a vehicle design can vary strongly even for a slight change in the impact position. Based on the same FE submodel introduced in Section 5.3.1 and a similar setup in Section 5.3.2, we investigate the multi-pole impact problem using the pole position `P1` and `P2`, and the optimization objective f_{opt} defined in Equation 5.2.

$$\min f_{opt} = \alpha \cdot m' + u'_1 + u'_2 \,, \tag{5.2}$$

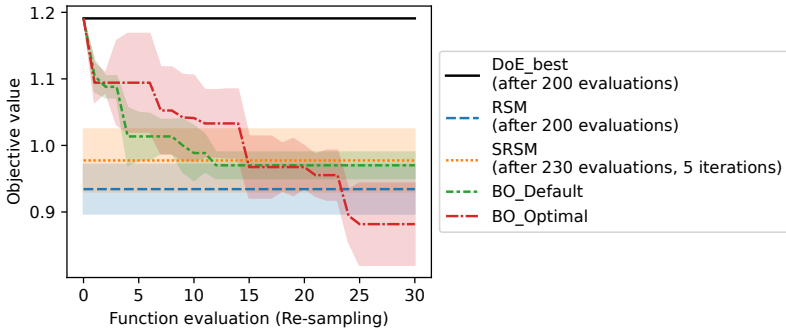
where $\alpha = 2$ for an equal weighting between mass and intrusions and $u'_{i\in\{1,2\}}$ represents the intrusion for impact position `P1` and `P2`, which is computed using the same normalization as in Equation 5.1. For the multi-pole impact, a vehicle design must be separately evaluated using two FE simulations, i.e., one FE simulation for each impact position, e.g., during the generation of DoE samples and resampling. Following this, an experimental setup with a reduced scope is considered here to minimize the computational cost, consisting of an initial DoE of 200 samples (roughly $10 \cdot d$), 30 resamplings, and three optimization repetitions, refer to Section 5.3.2.

The optimization results using different optimization approaches for the multi-pole impact problem is summarized in Figure 5.10(a). In line with our expectations, better vehicle designs with smaller objective values can be identified using RSM, SRSM, and BO, compared to the classical one-shot optimization approach. In contrast to

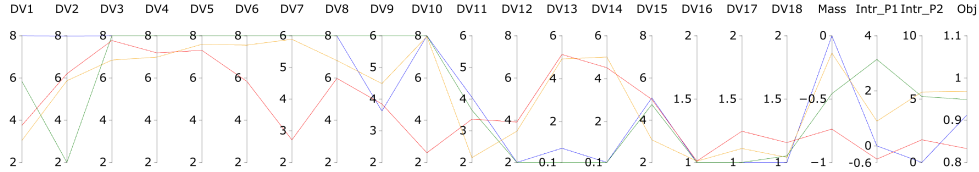
the single pole impact problem, here the performance of SRSM is worse than RSM in identifying a better vehicle design, despite the fact that better solutions could be possibly discovered over iterations. We believe that the complex multi-pole crash function might be poorly approximated in SRSM, considering that a reduced function evaluation budget, and hence, a smaller DoE sample size is available in each iteration for the construction of response surfaces, compared to the single pole impact problem. Meanwhile, the default BO configuration seems to be stuck in a local optimum, having an arguably similar performance as SRSM. On the other side, clearly better vehicle designs can be identified using our optimal BO configuration, outperforming RSM, SRSM, and the default BO configuration. In fact, while the optimal BO configuration converge almost as fast as the default BO configuration based on the AUC metric (roughly +1%), it can find a much better vehicle design (roughly −10%).

When looking at the design variables in Figure 5.10(b), a similar trend in the vehicle designs as previously in the single pole impact can be observed as well, namely a combination of thick rocker panels and thin supporting beams is favorable. On one side, most optimization approaches attempt to push the vehicle design towards the boundary of design space, revealing that thick rocker panels could potentially improve the overall robustness of a vehicle design in terms of structural performance against the multi-pole impacts. On the other side, our optimal BO configuration seems to be more flexible in exploring the design space, given that a combination of thinner rocker panels and slightly thicker supporting beams is provided as solution. Again using RSM as a baseline, the vehicle design found using SRSM has a slightly smaller mass (−0.1 kg), but at the cost of larger intrusions for `P1` (+0.9 mm) and `P2` (+5.6 mm). For the default BO configuration, the corresponding vehicle design is lighter (−0.5 kg), yet having a larger intrusion for `P1` (+3.1 mm) and `P2` (+5.2 mm). In comparison to that, the vehicle design identified using our optimal BO configurations is arguably better, having a smaller mass (−0.7 kg), a lower intrusion for `P1` (−0.4 mm), and a slightly larger intrusion for `P2` (+1.7 mm).

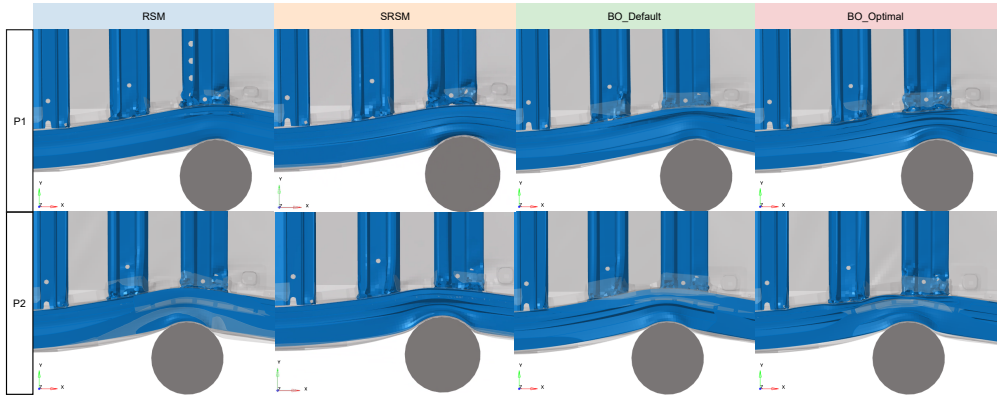
In summary, our optimization approach has promising potential in improving the performance of optimization algorithms for an optimal solving of real-world expensive BBO problems, by fine-tuning optimization configurations based on some representative functions. Particularly when solving complex BBO problems using a limited function evaluation budget, like the multi-pole impact problem, a clear advantage can be obtained using our approach compared to some conventional approaches, in line with our motivations. Nevertheless, we are aware that the optimization performances can be different, depending on the optimization objectives defined for optimization



(a) Performance comparison between one-shot optimization (*black*), RSM (*blue*), SRSM (*orange*), default BO configuration (*green*), and optimal BO configuration identified using our approach (*red*) for the multi-pole impact problem (Equation 5.2). While the best-found solution is presented for RSM and SRSM, the best-so-far solution during optimization is shown for BO, starting with the best DoE sample. The median performance and standard deviation over three optimization repetitions are reported. A smaller objective value and/or AUC is better.



(b) Comparison of vehicle designs for the multi-pole impact problem, focusing on the best optimization run using RSM (*blue*), SRSM (*orange*), default BO configuration (*green*), and optimal BO configuration identified using our approach (*red*). The upper and lower bound of the design space (thickness in mm) are shown at the top and bottom row. Using RSM as a baseline, the relative differences in mass (kg) and intrusions (mm) are shown, while the optimization objective values (Equation 5.2) are included in the last column.



(c) Structural deformation of the FE submodel for the multi-pole impact problem. The cross-section of different vehicle designs identified using RSM, SRSM, default BO, and optimal BO configuration are shown. The same setting is applied for all simulation snapshots, such as deformation scaling, view position, and time frame.

Figure 5.10: Summary of the optimization results using different optimization approaches for the multi-pole impact problem. Figures are taken from [74].

runs, e.g., Equation 5.1 and Equation 5.2, which requires further analysis.

5.4 Landscape-aware HPO using RGFs and deep NN models

Based on our investigations in Section 5.2 and Section 5.3, there is an inspiring potential in exploiting some cheap-to-evaluate RGFs for the fine-tuning of optimization configurations for optimally solving real-world expensive BBO problems. Subsequently, we are inspired to investigate the potential of using some RGFs for the training of predictive models that can predict optimal configurations, similar to a landscape-aware HPO context. In fact, this approach has been previously investigated in [160], where it was reported that the performance of predictive models trained using RGFs was rather lackluster. Independently of this work, our approach mainly differs in the following aspects:

- Instead of simply including any RGF in the training set, the selection process in Section 4.2.4 is employed to identify RGFs that are appropriate for HPO purposes. We believe that this step is essential to improve the model prediction accuracy, which might partly explain the unsatisfied model performances in [160];
- Unlike typically done in ASP, where optimal configurations are identified from a fixed portfolio of limited configurations, our investigation is extended towards HPO and/or CASH, i.e., considering an exploration of the hyperparameter search space; and
- For the prediction of optimal configurations, we propose considering NN-based predictive models, which can properly handle multi-output mixed regression and classification tasks.

An overview of our landscape-aware HPO approach is presented in Figure 5.11. Consisting of a training and testing phase, the ELA features and corresponding optimal configurations identified based on some RGFs are used for the training of predictive models, which can be deployed later for the prediction of optimal configurations for unseen real-world expensive BBO problems. A detailed description is provided in the following:

Training phase:

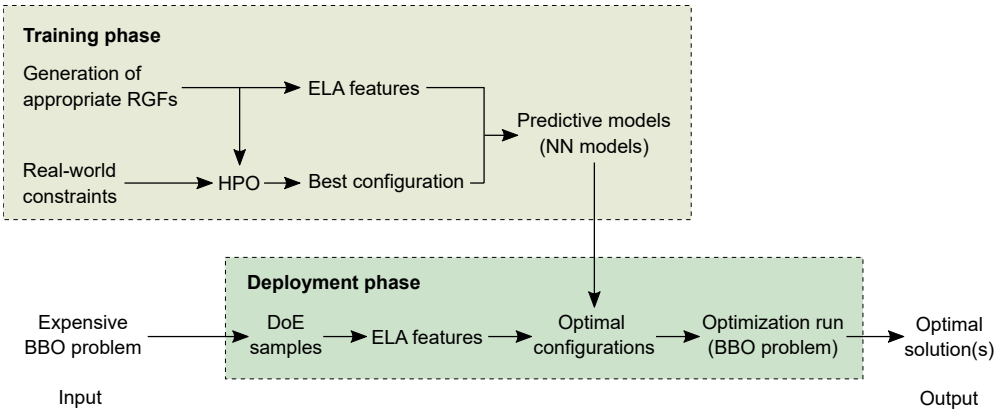


Figure 5.11: Overview of the proposed landscape-aware HPO approach that can identify optimal configurations for real-world expensive BBO problems using some pre-trained predictive models. During the training phase, based on a preferably large set of RGFs, their ELA features and optimal configurations are taken for the training of NN models. Eventually, these NN models can be deployed for the prediction of optimal configurations for unseen BBO problems based on their ELA features. Figure taken from [71] with modifications.

1. Firstly, using the random function generator proposed in Section 4.1, a large set of RGFs is generated. In this regard, the selection process introduced in Section 4.2.4 is utilized to identify RGFs that are appropriate for HPO purposes, which are then included in the training function set.
2. For each of the training function, the corresponding optimization landscape characteristics are quantified using ELA features based on some DoE samples. Similar to Section 3.1.1, (i) the objective values are min-max normalized before the ELA feature computation to reduce inherent bias, (ii) highly correlated ELA features are eliminated, and (iii) the remaining ELA features are rescaled to a comparable scale range using min-max scaling.
3. Meanwhile, the respective optimal configurations are separately identified for each training function using HPO w.r.t. some real-world constraints, such as the allocated world-clock time and computational resources. In preparation for the model training purposes, categorical hyperparameters are one-hot encoded, while continuous hyperparameters are linearly rescaled to $[0, 1]$ using Equation 3.1.
4. Subsequently, deep NN models are trained using the ELA features computed as input and the optimal configurations identified as output. A de-

scription of the NN models is provided in Section 5.4.1.

Deployment phase:

1. A set of DoE samples of a real-world expensive BBO problem to-be-solved is needed as input. Following this, the ELA features of the BBO problem instance can be computed and normalized in a similar way as during the training phase.
2. Based on these ELA features, optimal configurations can be identified using the trained NN models, where the predicted hyperparameters are inversely transformed back to their original search space. To avoid an invalid configuration, e.g., a negative population size for ModCMA, the lower or upper boundary will be assigned for continuous hyperparameters that are predicted outside of the search space.
3. Lastly, the predicted configuration is applied for an optimal solving of the BBO problem instance.

5.4.1 Multi-output Mixed Regression and Classification

In this thesis, we propose considering dense NN models for the prediction of optimal configurations, which is essentially a multi-output mixed regression and classification task, using a similar architecture as illustrated in Figure 5.12.

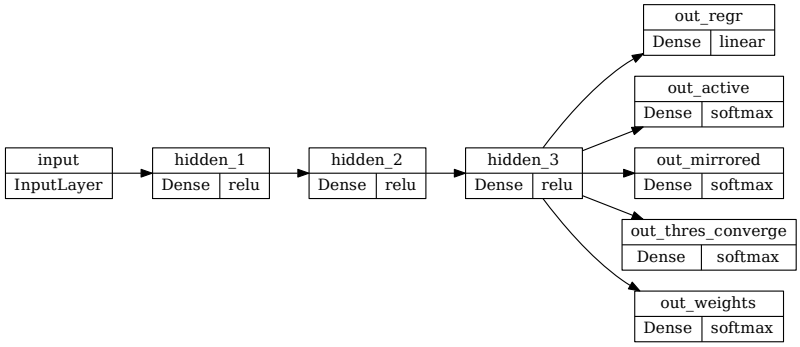


Figure 5.12: Example of the architecture of a dense NN model for mixed regression and classification. *From left to right:* An input layer, three hidden layers, and several output layers, where one output layer is assigned for regression task and four layers for classification tasks. Figure taken from [71].

Input layer: The input layer has a size equal to the number of ELA features available in the training dataset.

Hidden layers: To identify an optimal architecture, different combinations of number of hidden layer $\{1, 2, 3\}$, hidden layer sizes $\{16, 32, 64, 128\}$, and epochs $\{100, 150, 200\}$ are evaluated based on a grid search approach. Using a $80 : 20$ training-testing split of the training dataset and five repetitions, the hidden layers are constructed based on the combination with the smallest validation loss. Here, ReLU is utilized as activation function for all hidden layers.

Output layers: Basically, different output layers are considered for the mixed regression and classification tasks. On one side, a single output layer using linear activation function is assigned for the multi-output regression task. On the other side, the multi-output multi-class classification task is split into multiple classifications tasks using a separated output layer, where softmax activation function is applied for the categorical hyperparameters. Following this, the size of output layers are dependent on the number of hyperparameters.

Loss functions: During the training of NN models, we consider mean squared error and categorical cross entropy as loss functions for the regression and classification tasks, respectively.

To properly evaluate the potential of NN models, we consider the performance of RF models as a baseline, which is a popular choice for landscape-aware ASP. In this context, the configurations of RF models are optimally fine-tuned using an automated CASH tool `auto-sklearn` [36] based on a $80 : 20$ training-testing split of the training dataset. Given that a multi-output multi-class classification task is currently limited in `auto-sklearn`, here we instead consider a multi-target regression task, where the categorical hyperparameters are encoded using numerical labels.

5.4.2 Experimental Setup and Result Discussion

To fairly evaluate the potential of RGFs, a set of MA-BBOB functions is additionally included as training dataset for the predictive models. A summary of the experimental setup is provided in the following, namely:

- Considering 24 BBOB functions of the first instance in 5-d as unseen test problems;

- A set of 1 000 RGFs, 1 000 MA-BBOB functions, and a combination of both functions as training dataset;
- The optimization landscape characteristics in terms of ELA features are computed based on a DoE of $50 \cdot d$ samples;
- In this investigation, we focus on fine-tuning the configurations of ModCMA, as summarized in Table 5.2. Precisely, the corresponding optimal ModCMA configurations are identified for each training function through HPO using TPE and a budget of 500 evaluations. In this context, each optimization run is allocated with a fixed budget of $1\,000 \cdot d$ evaluations and 10 repetitions using different random seeds;
- The optimization performance of a configuration is evaluated using the AUC metric introduced in Section 4.2.1; and
- Apart from that, we also extend our investigation to BBOB functions in 20-d, using a smaller experimental setup to minimize the overall computational effort. This consists of a DoE of $20 \cdot d$ samples for the ELA feature computation, optimization runs using $100 \cdot d$ evaluations, 300 evaluations for TPE, and only seven real-valued ModCMA hyperparameters (Table 5.2) are taken into consideration.

Representativeness of Training Data

Considering that the problem classes of BBOB functions should be sufficiently covered, it is natural to expect that predictive models trained using MA-BBOB functions can have a good performance. While we can indeed observe this on many of the BBOB functions, it is not always the case, e.g., for F7 and F12, which will be discussed later. Looking for an explanation, we delve into analyzing the representativeness of MA-BBOB functions and RGFs w.r.t. the ELA feature space. As shown in Figure 5.13, not all BBOB functions are sufficiently covered by the MA-BBOB functions, which might be related to the mechanism employed for the generation of MA-BBOB functions [147]. Subsequently, we suspect that this insufficient coverage provided by the MA-BBOB functions might explain the poor prediction performances on some BBOB functions. On the contrary, RGFs can cover a larger region of the ELA feature space, showing that RGFs are much more diverse in terms of optimization problem classes. In fact, a combination of the large distribution of RGFs and the more focused distribution of MA-BBOB functions towards some BBOB functions seems to be the best training function set.

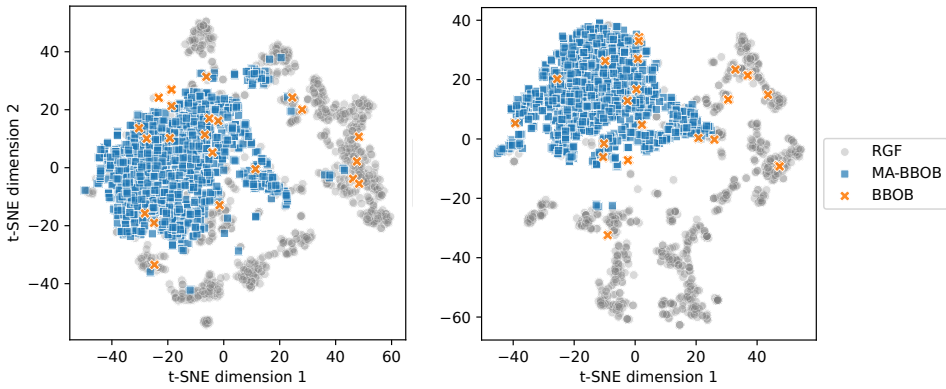


Figure 5.13: Projection of the high-dimensional ELA feature space to a 2-d visualization using the t-SNE approach for 1 000 RGFs (*gray*), 1 000 MA-BBOB (*blue*), and 24 BBOB functions (*orange*) in 5-d (*left*) and 20-d (*right*), based on a similar approach as in Section 4.1.2. Figure taken from [71].

Performance of Predicted Configurations

The optimization performances using different ModCMA configurations for 24 BBOB functions in 5-d are summarized in Figure 5.14. Generally, a better performance can be achieved using optimal configurations identified by the predictive models on most BBOB functions, compared to the default configuration. Meanwhile, the optimal configurations identified can compete against the SBS on some functions, such as F7 and F17. In fact, NN models can identify optimal configurations that outperform the SBS in some cases, e.g., for F5 and F13. For highly multi-modal functions like F16 and F23, on the other hand, the performance of optimal configurations predicted are lackluster. We suspect that an ELA feature that can properly quantify the landscape characteristics of such complex functions is still lacking, indicating that there is still room for improvement in our approach. Interestingly, the optimal configurations predicted sometimes seem to be competitive against the VBS, such as for F21. Principally, similar observations can be made for the BBOB functions in 20-d, as shown in Figure 5.15.

For an unbiased analysis, we statistically compare the performance of different ModCMA configurations, based on the Wilcoxon signed-rank test using the hypothesis *optimal configurations predicted using NN models are equally competitive or better*. Here, we focus on evaluating the performance of optimal configurations identified by NN models trained using RGFs against the default configuration, SBS, and RF mod-

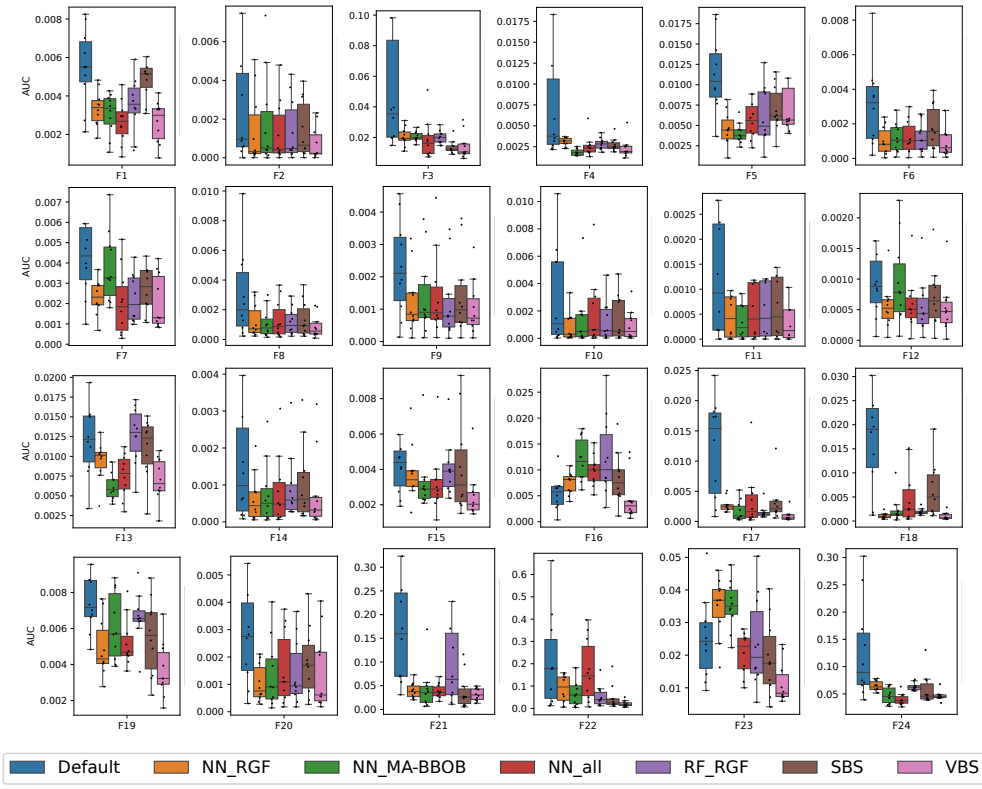


Figure 5.14: Performance of different ModCMA configurations for 24 BBOB functions in 5-*d*, using a repetition of 10 times. The configuration having a smaller AUC is better. *Legend:* Default configuration, configuration predicted by NN models trained using RGFs, by NN models trained using MA-BBOB functions, by NN models trained using both RGFs and MA-BBOB functions, and by RF models trained using RGFs, SBS, and VBS. Figure taken from [71].

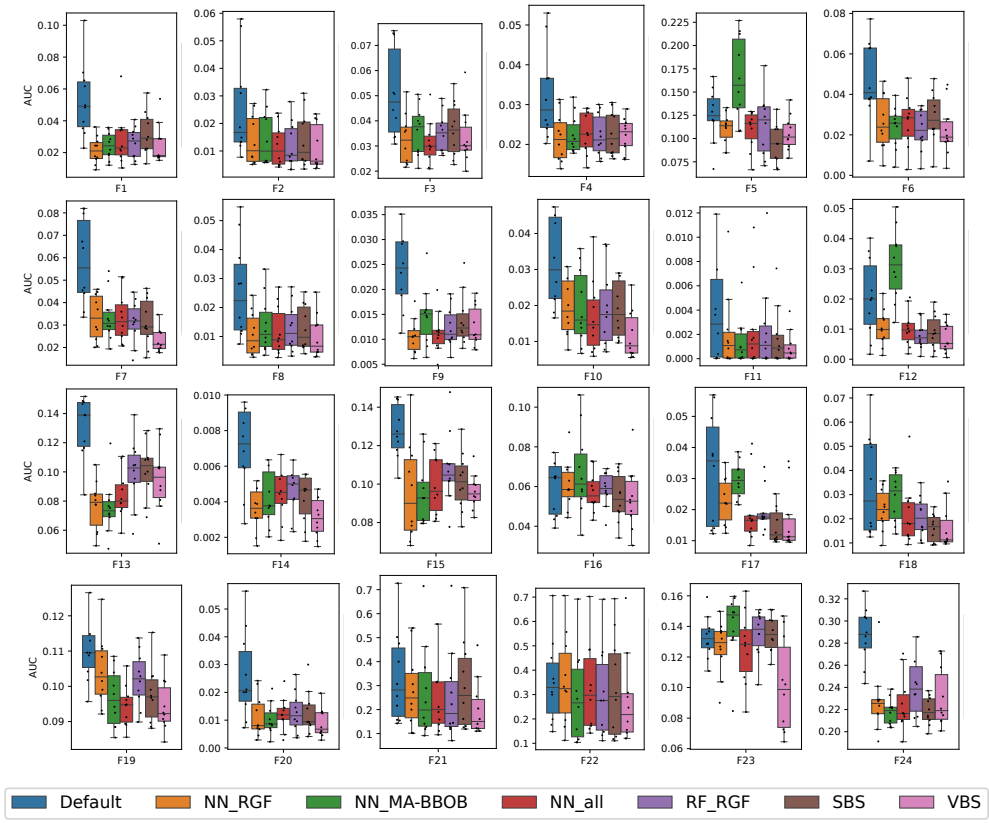


Figure 5.15: Performance of different ModCMA configurations for 24 BBOB functions in 20-d, using a repetition of 10 times. The configuration having a smaller AUC is better. *Legend:* Default configuration, configuration predicted by NN models trained using RGFs, by NN models trained using MA-BBOB functions, by NN models trained using both RGFs and MA-BBOB functions, and by RF models trained using RGFs, SBS, and VBS. Figure taken from [71].

els. As illustrated in Figure 5.16, optimal configurations predicted using NN models can indeed outperform the default configuration on most of the BBOB functions, in line with our previous interpretations. On top of that, the optimal configurations can beat the SBS on many BBOB functions. Meanwhile, the optimal configurations seem to be still competitive against the default configuration and SBS in a few remaining BBOB functions. Remarkably, our analysis reveals that our approach is much more effective in solving simple functions (first half of the BBOB suite), compared to complex functions (second half), which might be related to the effectiveness of ELA features in capturing different landscape characteristics. Beyond that, NN models can perform equally good, or even better than RF models in some cases, especially in 5-d. Meanwhile, a similar trend can be observed for the BBOB functions in 20-d. Nevertheless, the effectiveness of optimal configurations predicted compared to SBS seems to be lower in 20-d.

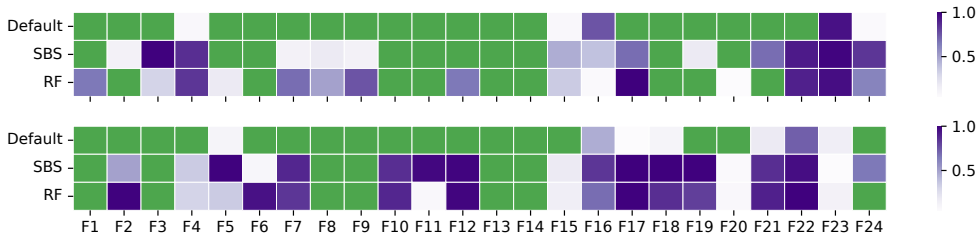


Figure 5.16: Pairwise performance comparison between the optimal configurations identified using NN models against the default configuration, SBS, and RF models for 24 BBOB functions in 5d (*top*) and 20d (*bottom*). Based on the Wilcoxon signed-rank test, a green color indicates that there is statistically significant evidence to support the hypothesis *optimal configurations predicted using NN models are equally competitive or better*, with a p-value smaller than 0.05. On the other hand, a darker purple color or larger p-value indicates that the hypothesis is more likely to be rejected, while a lighter purple color or smaller p-value for a lower chance of rejection. Figure taken from [71].

The performance of optimal configurations identified directly based on RGFs (Section 5.2.2) and using NN models (this Section) seems to be somewhat similar, i.e., outperforming the default configuration on many of the BBOB functions and being competitive against the SBS in some cases. While it is interesting to understand which approach is more effective in identifying better optimization configurations, both results are not directly comparable, since the hyperparameters considered are slightly different. Following this, a further analysis is necessary.

5.5 Conclusions

Motivated to assist practitioners in automatically fine-tuning optimization configurations for their applications, we evaluate the potential of our proposed optimization approach in this chapter. As explained in Section 5.1, we propose considering some cheap-to-evaluate representative functions that belong to the same optimization problem class for the fine-tuning of optimization configurations to optimally solve real-world expensive BBO problems. To improve the reliability of the proposed approach, a selection process for identifying appropriate representative functions and a training-testing split are considered for the HPO process. Subsequently, the performance of our optimization approach is evaluated based on the BBOB functions in Section 5.2 and a real-world automotive crashworthiness optimization problem in Section 5.3. Beyond that, we also investigate the potential of training general purpose predictive models that can identify optimal configurations for real-world expensive BBO problems in Section 5.4, based on a set of appropriate RGFs and deep NN models, similar to a typical landscape-aware ASP context.

In brief, this chapter focuses on answering *RQ5* and *RQ6*, as follows:

RQ5: What is the performance of optimal optimization algorithms identified using the proposed optimization approach (Section 1.1), when tested on some benchmark functions? More crucially, can the optimal optimization algorithms outperform some state-of-the-art approaches for solving real-world expensive BBO problems?

When applied on the BBOB functions in 20-d, optimal configurations identified using our approach can outperform the default configuration on most of the BBOB functions in terms of optimization convergence speed based on the AUC metric. Furthermore, our optimal configurations can compete against or even show a better performance than the SBS on some BBOB functions. This is particularly motivating for real-world applications, where such SBS is usually not available. Apart from that, our approach shows a high flexibility, as it can work well with different BBO algorithms, such as ModCMA, ModDE, and BO, as well as different optimizers for HPO like TPE and SMAC.

Apart from the BBOB functions, we also evaluate the potential of our approach for solving real-world expensive BBO problems, using automotive crashworthiness optimization as a representative example. Precisely, we focus on an automotive side crash problem using two load cases, namely a single pole impact

and a multi-pole impact. Overall, a better optimization performance can be achieved using our optimally fine-tuned BO configurations for both load cases, in comparison to the default BO configuration. Especially for the multi-pole impact problem, our optimal BO configuration outperforms both RSM and SRSM that are considered as state-of-the-art in the automotive industry, finding an arguably better vehicle design w.r.t. mass and intrusions. Following this, we are confident that our approach can be applied for optimally solving real-world expensive BBO problems that require dealing with complex functions and using a limited function evaluation budget, which are two critical aspects in real-world applications.

RQ6: What is the potential of pre-trained general purpose predictive models in identifying optimal optimization algorithms for real-world expensive BBO problems?

Based on our analysis on 24 BBOB functions in 5-d and 20-d, near-optimal ModCMA configurations can be identified based on deep NN models trained using some properly selected RGFs. In fact, these optimal configurations can outperform the default configuration on most of the BBOB functions, and even compete against the SBS in some cases. Compared to the BBOB functions and MA-BBOB functions, it is advantageous to consider RGFs as training dataset, since they cover a broader spectrum of optimization problem classes within the ELA feature space. Subsequently, we believe that our approach can generalize well to real-world expensive BBO problems, provided that their optimization problem classes are sufficiently represented by the RGFs. Overall, well-performing configurations can be best identified using dense NN models trained on a combination of RGFs and MA-BBOB functions.

Based on our analysis, our proposed automated optimization approach has an immense potential in efficiently solving real-world expensive BBO problems w.r.t. some real-world constraints, by optimally fine-tuning optimization configurations based on some cheap representative functions. Although our investigations are limited to unconstrained single objective optimization problems, an improved optimization performance can be achieved using our optimization approach in comparison to conventional methods, in line with our motivations. Once the overall computational effort scales up, e.g., using a full vehicle FE model (Table 5.1), we believe that such a performance improvement could be significant. Meanwhile, we are confident that the proposed optimization approach can generalize well to other real-world BBO domains that are

sufficiently represented by the representative functions.

The content of this chapter is mainly based on the author's contribution in the publications [71, 73, 74].

Chapter 6

Conclusions and Future Work

The contributions of this thesis serve as a significant step in the long journey towards our ultimate vision, i.e., developing an automated optimization pipeline for efficiently solving real-world expensive BBO problems. A summary of this thesis is provided in Section 6.1, followed by a list of future work in Section 6.2.

6.1 Conclusions

Efficiently solving real-world BBO problems w.r.t. real-world constraints, such as wall-clock time and computational cost, has gained increasing attention over the years. Unlike solving benchmark functions, such as the popular BBOB functions, the function evaluation budget available for real-world applications is typically rather limited, particularly when the function evaluation is expensive. In the automotive industry, for instance, the quality of vehicle designs is nowadays commonly evaluated using time-consuming and costly FE simulation runs. Following this, the development of effective optimization approaches for real-world applications is becoming a pressing issue, such as finding better optimization solutions, within a shorter time duration, and/or using less computational resources.

Motivated to fill the gap, we propose an automated optimization pipeline for (near) optimally solving real-world expensive BBO problems within the scope of this thesis. In this regard, we propose considering a set of scalable and cheap-to-evaluate representative functions that belong to the same optimization problem classes as the BBO problems for the fine-tuning of optimization configurations w.r.t. some real-world constraints. Subsequently, optimization configurations can be properly fine-tuned at a

relatively low cost, prior to the optimization runs using expensive function evaluations. Eventually, the optimal configurations identified can be applied for an efficient solving of the expensive BBO problems. Throughout this thesis, we consider automotive crashworthiness optimization as a representative real-world expensive BBO problem, which requires dealing with complex functions and expensive function evaluations.

Firstly, we begin our investigation by analyzing the optimization landscape characteristics of real-world expensive BBO problems in terms of ELA features in Chapter 3, which is still lacking according to the best of our knowledge. Based on our investigations on 20 automotive crash problem instances of different crash scenarios and problem dimensionalities, we show that the optimization landscape characteristics of these automotive crash problems are different from those of the BBOB functions. In other words, **the automotive crash problems belong to optimization problem classes that are different from the BBOB functions (*RQ1*)**. Subsequently, the automotive crash problems are insufficiently represented by the BBOB functions. Beyond that, we investigate a feature-free approach to characterize the optimization landscape of BBO problems based on some latent representations, which are computed using deep NN model like VAE. Interestingly, our results reveal that such latent space representations could potentially complement the classical ELA features in better capturing the optimization landscape of BBO problems.

Since the BBOB functions are insufficiently representative for real-world expensive BBO problems, we shift our focus towards generating test functions that belong to the same optimization problem classes in Chapter 4. **Using a tree-based random function generator, RGFs with similar optimization landscape characteristics in terms of ELA features can be identified for the automotive crash problems (*RQ2*)**. More importantly, **these similar RGFs can be exploited for estimating the actual performance of optimization configurations, and thus, for identifying optimal optimization configurations on expensive BBO problems (*RQ3*)**. Following this, we propose to consider RGFs as scalable and cheap-to-evaluate representations of real-world expensive BBO problems for HPO purposes. Given that a similar RGF could not be identified in some cases, e.g., for some of the BBOB functions, we additionally investigate the potential of GP in guiding the function generation towards specific optimization problem classes based on ELA features. **While some potential of this GP-based function generator can be observed, this approach is not as straightforward as expected (*RQ4*)** and substantial work is necessary to vigorously guide the function evolution, such as an improved feature selection strategy.

Based on the findings in previous chapters, we evaluate the performance of the proposed optimization pipeline for real-world expensive BBOB problems in Chapter 5. When tested on 24 BBOB functions in 20-d, we show that better optimization configurations can be indeed identified based on some representative functions, which can outperform the default configuration on most BBOB functions and even compete against the SBS in some cases. Moreover, the proposed optimization approach has a high flexibility, as it can work well with different combinations of BBO algorithms, e.g., ModCMA, ModDE, and BO, and optimizers for HPO, e.g., TPE and SMAC. More crucially, our results show that **the performance of BO can be improved using optimal configurations identified by our optimization approach, when solving a real-world automotive crashworthiness optimization problem using two load cases, namely a single pole impact and a multi-pole impact (*RQ5*)**. Especially for solving the multi-pole problem, a significant improvement in optimization performance can be achieved using our optimal BO configurations, which clearly outperform the state-of-the-art RSM and SRSM. Subsequently, we are confident that our optimization approach can be applied for an optimal solving of real-world expensive BBOB problems that require dealing with complex functions, while using a limited function evaluation budget. Apart from that, we evaluate the performance of predictive models in identifying optimal configurations for real-world expensive BBO problems, using deep NN models and RGFs as training dataset, similar to a landscape aware HPO context. Based on our investigation on the BBOB functions, **the trained NN models can perform quite well in identifying configurations that can most of the time outperform the default configuration (*RQ6*)**.

In summary, the proposed automated optimization pipeline has a motivating potential for an optimal solving of real-world expensive BBO problems, such as automotive crashworthiness optimization problems. Considering that the proposed approach can perform rather well on the BBOB suite, which covers a wide range of optimization problem classes, we believe that our approach can generalize to other BBO domains that are sufficiently represented by the BBOB functions. For instance, ship design problems in the maritime industry [25] and turbomachinery designs in the aerospace industry [103] are two potential real-world expensive BBO problems that are attractive for an application of our approach with appropriate modifications and extensions.

6.2 Future Work

During our investigations, nevertheless, the following weaknesses and limitations in our proposed optimization approach have been identified, where further analysis and improvements could be worthwhile:

Outlook 1: Within the scope of this thesis, we focus on optimally solving unconstrained single objective optimization problems. To improve the overall applicability of our proposed optimization approach for real-world applications, a proper extension towards constraint handling and/or multi-objective optimization would be valuable;

Outlook 2: Since some DoE samples of real-world expensive BBO problems are required as input, it is essential to investigate an optimal trade-off between the DoE sample size and effectiveness of our approach w.r.t. the problem dimensionality. Ideally, expensive function evaluations are performed only as much as necessary, but as little as possible to minimize computational effort. Meanwhile, an application of the proposed optimization approach for solving real-world BBO problems like automotive crash problems using a full vehicle FE model could be interesting, to properly estimate its value for industrial applications;

Outlook 3: In this thesis, we focus on evaluating the potential of fine-tuning optimization configurations using HPO based on some representative functions. Subsequently, another attractive research direction could be extending our approach towards CASH or AAC, where both the choice of optimization algorithms and their hyperparameters are properly fine-tuned;

Outlook 4: Based on our investigations, we show that BO indeed can perform well in solving real-world expensive BBO problems using a limited function evaluation budget. To further improve the performance of BO, for instance, the development of a configurable BO framework that is similar to ModCMA and ModDE could offer a great functional flexibility for real-world applications. In this regard, the BO hyperparameter search space could be expanded, e.g., including the hyperparameters of internal optimizer in BO and/or combining different BO variants (Section 2.1.1); and

Outlook 5: While a diverse set of RGFs can be generated using the tree-based random function generator (Section 4.1), the optimization problem classes that are being covered is primarily limited by the predefined pool of mathematical

operands and operators. Subsequently, further effort could be invested to improve the diversity of RGFs that can be generated, to cover a broader spectrum of optimization problem classes. The discontinuity in real-world problems, for instance, could be potentially included by considering step functions, as briefly mentioned in Section 4.1.2.

Bibliography

[1] Steven Adriaensen, André Biedenkapp, Gresa Shala, Noor Awad, Theresa Eimer, Marius Lindauer, and Frank Hutter. Automated Dynamic Algorithm Configuration. *Journal of Artificial Intelligence Research*, 75:1633–1699, December 2022. doi:10.48550/arXiv.2205.13881.

[2] Mohamad Alissa, Kevin Sim, and Emma Hart. Algorithm Selection Using Deep Learning without Feature Extraction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 198–206. Association for Computing Machinery, 2019. doi:10.1145/3321707.3321845.

[3] Altair Engineering Inc. Altair HyperStudy, 2022. URL: https://www.altair.com/hyperstudy/.

[4] ANSYS Inc. Ansys optiSLang, 2022. URL: https://www.ansys.com/products/connect/ansys-optislang.

[5] Kirill Antonov, Elena Raponi, Hao Wang, and Carola Doerr. High Dimensional Bayesian Optimization with Kernel Principal Component Analysis. In Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar, editors, *Parallel Problem Solving from Nature – PPSN XVII*, pages 118–131, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-14714-2_9.

[6] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. Springer series in operations research and financial engineering. Springer Nature, Cham, 1st ed. 2017. edition, 2017. doi:10.1007/978-3-319-68913-5.

[7] Irene Azzali, Leonardo Vanneschi, Sara Silva, Illya Bakurov, and Mario Giacobini. A Vectorial Approach to Genetic Programming. In Lukas Sekanina, Ting Hu, Nuno Lourenço, Hendrik Richter, and Pablo García-Sánchez, editors, *Genetic Programming*, pages 213–227, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-16670-0_14.

[8] Thomas H. W. Bäck, Anna V. Kononova, Bas van Stein, Hao Wang, Kirill A. Antonov, Roman T. Kalkreuth, Jacob de Nobel, Diederick Vermetten, Roy

de Winter, and Furong Ye. Evolutionary Algorithms for Parameter Optimization—Thirty Years Later. *Evolutionary Computation*, 31(2):81–122, 06 2023. doi:10.1162/evco_a_00325.

[9] Ishan Bajaj, Akhil Arora, and M. M. Faruque Hasan. *Black-Box Optimization: Methods and Applications*, pages 35–65. Springer International Publishing, Cham, 2021. doi:10.1007/978-3-030-66515-9_2.

[10] Thomas Bartz-Beielstein. A survey of model-based methods for global optimization. *Bioinspired Optimization Methods and Their Applications*, pages 1–18, 2016. URL: https://www.researchgate.net/publication/314207677_A_Survey_of_Model-Based_Methods_for_Global_Optimization.

[11] Yoav Benjamini and Yosef Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995. doi:10.1111/j.2517-6161.1995.tb02031.x.

[12] Carolin Benjamins, Anja Jankovic, Elena Raponi, Koen van der Blom, Marius Lindauer, and Carola Doerr. Towards Automated Design of Bayesian Optimization via Exploratory Landscape Analysis, 2022. arXiv:2211.09678.

[13] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA, 2011. Curran Associates Inc. URL: https://dl.acm.org/doi/10.5555/2986459.2986743.

[14] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuß. Algorithm Selection Based on Exploratory Landscape Analysis and Cost-Sensitive Learning. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, page 313–320, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2330163.2330209.

[15] Jakob Bossek, Carola Doerr, Pascal Kerschke, Aneta Neumann, and Frank Neumann. Evolving Sampling Strategies for One-Shot Optimization Tasks. In Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann, editors, *Parallel Problem Solving from Nature – PPSN XVI*, volume 12269, pages 111–124, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-58112-1_8.

[16] Olivier Bousquet, Sylvain Gelly, Karol Kurach, Olivier Teytaud, and Damien Vincent. Critical Hyper-Parameters: No Random, No Cry, 2017. arXiv:1706.03200.

[17] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, Dirk V. Arnold, and Tim Hohm. Mirrored Sampling and Sequential Selection for Evolution Strategies. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel*

Problem Solving from Nature, PPSN XI, pages 11–21, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-15844-5_2.

[18] Jana Büttner. *Effiziente Lösungsansätze zur Reduktion des numerischen Ressourcenbedarfs für den operativen Einsatz der Multidisziplinären Optimierung von Fahrzeugstrukturen*. Phd thesis, Bergische Universität Wuppertal, Wuppertal, Nordrhein-Westfalen, 2022. Available at https://www.oms.uni-wuppertal.de/fileadmin/maschbau/oms/05_Dissertation_Buettner_bearbeitet.pdf.

[19] David Charte, Francisco Charte, María J. del Jesus, and Francisco Herrera. An analysis on the use of autoencoders for representation learning: Fundamentals, learning task case studies, explainability and challenges. *Neurocomputing*, 404:93–107, sep 2020. doi:10.1016/j.neucom.2020.04.057.

[20] Giada Colella, Volker A Lange, and Fabian Duddeck. Transfer learning for crash design. 01 2024. doi:10.21203/rs.3.rs-3910181/v1.

[21] Alexander Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan-Rhys Griffiths, Alexandre Maravel, Jianye Hao, Jun Wang, Jan Peters, and Haitham Bou Ammar. HEBO: Pushing The Limits of Sample-Efficient Hyperparameter Optimisation. *Journal of Artificial Intelligence Research*, 74, 07 2022.

[22] Catharina Czech, Arne Kaps, and Fabian Duddeck. Robust multi-fidelity optimization approach exploiting data-driven, non-linear model order reduction. In Michael Beer, Enrico Zio, Kok-Kwang Phoon, and Bilal M. Ayyub, editors, *Proceedings of the 8th International Symposium on Reliability Engineering and Risk Management, ISRERM 2022*, Proceedings of the 8th International Symposium on Reliability Engineering and Risk Management, ISRERM 2022, pages 357–363. Research Publishing, 2024. doi:10.3850/978-981-18-5184-1_MS-12-041-cd.

[23] Jacob de Nobel, Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. Tuning as a Means of Assessing the Benefits of New Ideas in Interplay with Existing Algorithmic Modules. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '21, page 1375–1384, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3449726.3463167.

[24] Roy de Winter, Fu Xing Long, Andre Thomaser, Thomas H.W. Bäck, Niki van Stein, and Anna V. Kononova. Landscape analysis based vs. domain-specific optimization for engineering design applications: A clear case. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pages 776–781, 2024. doi:10.1109/CAI59869.2024.00148.

[25] Roy de Winter, Bas van Stein, Matthys Dijkman, and Thomas Bäck. Designing ships using constrained multi-objective efficient global optimization. In Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umeton, and

Vincenzo Sciacca, editors, *Machine Learning, Optimization, and Data Science*, pages 191–203, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-13709-0_16.

[26] Konstantin Dietrich and Olaf Mersmann. Increasing the Diversity of Benchmark Function Sets Through Affine Recombination. In Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar, editors, *Parallel Problem Solving from Nature – PPSN XVII*, pages 590–602, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-14714-2_41.

[27] Carola Doerr, Johann Dreo, and Pascal Kerschke. Making a Case for (Hyper-)Parameter Tuning as Benchmark Problems. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1755–1764, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3319619.3326857.

[28] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics. *arXiv e-prints:1810.05281*, 2018. URL: https://arxiv.org/abs/1810.05281, arXiv:1810.05281.

[29] Fabian Duddeck. Multidisciplinary optimization of car bodies. *Structural and Multidisciplinary Optimization*, 35(4):375–389, 2008. doi:10.1007/s00158-007-0130-6.

[30] DYNAmore GmbH. LS-OPT, 2022. URL: https://www.dynamore.de/en/products/opt/ls-opt?set_language=en.

[31] David Eriksson and Martin Jankowiak. High-Dimensional Bayesian Optimization with Sparse Axis-Aligned Subspaces, 2021. URL: https://arxiv.org/abs/2103.00349, arXiv:2103.00349.

[32] David Eriksson, Michael Pearce, Jacob R Gardner, Ryan Turner, and Matthias Poloczek. *Scalable global optimization via local Bayesian optimization*. Curran Associates Inc., Red Hook, NY, USA, 2019. URL: https://dl.acm.org/doi/10.5555/3454287.3454780.

[33] European New Car Assessment Programme (Euro NCAP). Far Side Occupant Test & Assessment Protocol, May 2023. URL: https://cdn.euroncap.com/media/77295/euro-ncap-far-side-test-and-assessment-protocol-v24.pdf.

[34] H. Fang, Masoud Rais-Rohani, Z. Liu, and Mark Horstemeyer. A comparative study of metamodeling methods for multiobjective crashworthiness optimization. *Computers & Structures*, 83(25-26):2121–2136, 2005. doi:10.1016/j.compstruc.2005.02.025.

[35] Jianguang Fang, Guangyong Sun, Na Qiu, Nam-Ho Kim, and Qing Li. On design optimization for structural crashworthiness and its state of the art. *Structural and Multidisciplinary Optimization*, 55(3):1091–1119, 2017. doi:10.1007/s00158-016-1579-y.

[36] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-Sklearn 2.0: Hands-free AutoML via Meta-Learning, 2022. arXiv:2007.04074.

[37] Matthias Feurer and Frank Hutter. *Hyperparameter Optimization*, pages 3–33. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-05318-5_1.

[38] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner Gardner, Marc Parizeau, and Christian Gagné. DEAP: evolutionary algorithms made easy. *J. Mach. Learn. Res.*, 13(1):2171–2175, July 2012. URL: https://dl.acm.org/doi/10.5555/2503308.2503311.

[39] Karim Hamza and Mohamed Shalaby. A framework for parallelized efficient global optimization with application to vehicle crashworthiness optimization. *Engineering Optimization*, 46(9):1200–1221, 2014. doi:10.1080/0305215X.2013.827672.

[40] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial. ArXiv e-prints, arXiv:1604.00772, 2016, pp.1-39, 2005. URL: https://inria.hal.science/hal-01297037.

[41] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, page 1689–1696, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1830761.1830790.

[42] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. COCO data archives. https://numbbo.github.io/data-archive/. Last accessed: May 15, 2024.

[43] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. COCO: a platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, 2021. doi:10.1080/10556788.2020.1808977.

[44] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA, 2009. URL: https://hal.inria.fr/inria-00362633.

[45] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996. doi:10.1109/ICEC.1996.542381.

[46] Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001. doi:10.1162/106365601750190398.

[47] Julian Happian-Smith. *An Introduction to Modern Vehicle Design*. Reed Educational and Professional Publishing Ltd., Oxford, UK, 2002. URL: https://edisciplinas.usp.br/pluginfile.php/6577230/mod_resource/content/1/An_Introduction_to_Modern_Vehicle_Design.pdf.

[48] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. Vanilla Bayesian Optimization Performs Great in High Dimensions, 2024. arXiv:2402.02229.

[49] Anja Jankovic and Carola Doerr. Landscape-Aware Fixed-Budget Performance Regression and Algorithm Selection for Modular CMA-ES Variants. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 841–849, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3377930.3390183.

[50] Kalervo Järvelin and Jaana Kekäläinen. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, oct 2002. doi:10.1145/582415.582418.

[51] Grahame A. Jastrebski and Dirk V. Arnold. Improving Evolution Strategies through Active Covariance Matrix Adaptation. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2814–2821, 2006. doi:10.1109/CEC.2006.1688662.

[52] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455, 1998. doi:10.1023/A:1008306431147.

[53] Frank J. Massey Jr. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68–78, 1951. doi:10.2307/2280095.

[54] Arne Kaps, Catharina Czech, and Fabian Duddeck. A hierarchical kriging approach for multi-fidelity optimization of automotive crashworthiness problems. *Structural and Multidisciplinary Optimization*, 65(4):114, 2022. doi:10.1007/s00158-022-03211-2.

[55] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995. doi:10.1109/ICNN.1995.488968.

[56] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Detecting Funnel Structures by Means of Exploratory Landscape Analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, page 265–272, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2739480.2754642.

[57] Pascal Kerschke and Heike Trautmann. The R-Package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 5262–5269. IEEE, 2016. doi:10.1109/CEC.2016.7748359.

[58] Pascal Kerschke and Heike Trautmann. Automated Algorithm Selection on Continuous Black-Box Problems by Combining Exploratory Landscape Analysis and Machine Learning. *Evolutionary Computation*, 27(1):99–127, 2019. doi:10.1162/evco_a_00236.

[59] Pascal Kerschke and Heike Trautmann. *Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package Flacco*, pages 93–123. Studies in Classification, Data Analysis, and Knowledge Organization. Springer International Publishing, Cham, 2019. doi:10.1007/978-3-030-25147-5_7.

[60] Pascal Kerschke and Heike Trautmann. flacco: Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems, 2019. Last accessed 15 January 2022. URL: https://github.com/kerschke/flacco.

[61] Jack P.C. Kleijnen. Kriging metamodeling in simulation: A review. *European Journal of Operational Research*, 192(3):707–716, 2009. doi:10.1016/j.ejor.2007.10.013.

[62] Schalk Kok and Nielen Stander. Optimization of a sheet metal forming process using successive multipoint approximations. *Structural optimization*, 18:277–295, 1999. doi:10.1007/BF01223312.

[63] Brent Komer, James Bergstra, and Chris Eliasmith. Hyperopt-Sklearn: Automatic Hyperparameter Configuration for Scikit-Learn. In *SciPy*, 2014. doi:10.25080/Majora-14bd3278-006.

[64] John R. Koza. Hierarchical genetic algorithms operating on populations of computer programs. In N. S. Sridharan, editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, volume 1, pages 768–774, Detroit, MI, USA, 20-25 August 1989. Morgan Kaufmann. URL: http://dl.acm.org/citation.cfm?id=1623755.1623877.

[65] Hasan Kurtaran, Azim Eskandarian, D. Marzougui, and N. Bedewi. Crashworthiness design optimization using successive response surface approximations. *Computational Mechanics*, 29(4):409–421, 2002. doi:10.1007/s00466-002-0351-x.

[66] Pedro Larrañaga and Jose A. Lozano. *Estimation of distribution algorithms: A new tool for evolutionary computation*, volume 2. Springer Science & Business Media, 2001. doi:10.1007/978-1-4615-1539-5.

[67] Mathias Lesjak and Fabian Duddeck. Dimensional reduction for parametric projection-based reduced-order models in crash. *PAMM*, 23(2), 2023. doi:10.1002/pamm.202300063.

[68] Jian-Yu Li, Zhi-Hui Zhan, and Jun Zhang. Evolutionary Computation for Expensive Optimization: A Survey. *Machine Intelligence Research*, 19(1):3–23, February 2022. doi:10.1007/s11633-022-1317-4.

[69] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022. URL: http://jmlr.org/papers/v23/21-0888.html.

[70] Livermore Software Technology Corporation. LS-DYNA Theory Manual, July 2019. URL: https://ftp.lstc.com/anonymous/outgoing/jday/manuals/DRAFT_Theory.pdf.

[71] Fu Xing Long, Moritz Frenzel, Peter Krause, Markus Gitterle, Thomas Bäck, and Niki van Stein. Landscape-Aware Automated Algorithm Configuration Using Multi-output Mixed Regression and Classification. In Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tušar, Penousal Machado, and Thomas Bäck, editors, *Parallel Problem Solving from Nature – PPSN XVIII*, pages 87–104, Cham, 2024. Springer Nature Switzerland. doi:10.1007/978-3-031-70068-2_6.

[72] Fu Xing Long, Bas van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Learning the characteristics of engineering optimization problems with applications in automotive crash. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '22, page 1227–1236, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3512290.3528712.

[73] Fu Xing Long, Bas van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Generating Cheap Representative Functions for Expensive Automotive Crashworthiness Optimization. *ACM Trans. Evol. Learn. Optim.*, 4(2), jun 2024. doi:10.1145/3646554.

[74] Fu Xing Long, Niki van Stein, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. Surrogate-based automated hyperparameter optimization for expensive automotive crashworthiness optimization. *Struct. Multidiscip. Optim.*, 68(4), April 2025. doi:10.1007/s00158-025-03989-x.

[75] Fu Xing Long, Diederick Vermetten, Anna V. Kononova, Roman Kalkreuth, Kaifeng Yang, Thomas Bäck, and Niki van Stein. Challenges of ELA-Guided Function Evolution Using Genetic Programming. In *Proceedings of the 15th International Joint Conference on Computational Intelligence - Volume 1: ECTA*, pages 119–130. INSTICC, SciTePress, 2023. doi:10.5220/0012206200003595.

[76] Fu Xing Long, Diederick Vermetten, Bas van Stein, and Anna V. Kononova. BBOB Instance Analysis: Landscape Properties and Algorithm Performance Across Problem Instances. In *Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings*, page 380–395, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-031-30229-9_25.

[77] Fu Xing Long, Diederick Vermtten, Bas van Stein, and Anna V. Kononova. Reproducibility files and additional figures, November 2022. Code and data repository: https://figshare.com/s/9aecdd3e4e2e0e4c12c5. Figure repository: https://figshare.com/s/dec915a84dca01bce781.

[78] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. doi:10.1016/j.orp.2016.09.002.

[79] Monte Lunacek and Darrell Whitley. The Dispersion Metric and the CMA Evolution Strategy. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 477–484, New York, NY, USA, 2006. Association for Computing Machinery. doi:10.1145/1143997.1144085.

[80] Katherine Mary Malan. A Survey of Advances in Landscape Analysis for Optimisation. *Algorithms*, 14(2):40, 2021. doi:10.3390/a14020040.

[81] M. D. McKay, R. J. Beckman, and W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics*, 21(2):239–245, 1979. URL: http://www.jstor.org/stable/1268522.

[82] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory Landscape Analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 829–836, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/2001576.2001690.

[83] Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis. In Robert Schaefer, Carlos Cotta, Joanna Kołodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, pages 73–82, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:10.1007/978-3-642-15844-5_8.

[84] Jonas Močkus. On bayesian methods for seeking the extremum. In G. I. Marchuk, editor, *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, pages 400–404, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg. doi:10.1007/3-540-07165-2_55.

[85] Jonas Mockus. The Bayesian approach to global optimization. In R. F. Drenick and F. Kozin, editors, *System Modeling and Optimization*, pages 473–481, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. doi:10.1007/BFb0006170.

[86] Mario A. Muñoz and Kate A. Smith-Miles. Performance Analysis of Continuous Black-Box Optimization Algorithms via Footprints in Instance Space. *Evol. Comput.*, 25(4):529–554, dec 2017. doi:10.1162/evco_a_00194.

[87] Mario Andrés Muñoz, Michael Kirley, and Kate Smith-Miles. Analyzing randomness effects on the reliability of exploratory landscape analysis. *Natural Computing: An International Journal*, 21(2):131–154, jun 2022. doi:10.1007/s11047-021-09847-1.

[88] Mario A. Muñoz and Kate Smith-Miles. Generating New Space-Filling Test Instances for Continuous Black-Box Optimization. *Evolutionary Computation*, 28(3):379–404, 09 2020. doi:10.1162/evco_a_00262.

[89] Mario Andrés Muñoz, Michael Kirley, and Saman K. Halgamuge. Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87, 2015. doi:10.1109/TEVC.2014.2302006.

[90] Mario Andrés Muñoz, Yuan Sun, Michael Kirley, and Saman K. Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015. doi:10.1016/j.ins.2015.05.010.

[91] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012. doi:10.1002/widm.53.

[92] Noesis Solutions. Optimus, 2022. URL: https://www.noesissolutions.com/our-products/optimus.

[93] Feng Pan and Ping Zhu. Design optimisation of vehicle roof structures: benefits of using multiple surrogates. *International Journal of Crashworthiness*, 16(1):85–95, 2011. doi:10.1080/13588265.2010.514773.

[94] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[95] Alejandro Piad-Morffis, Suilan Estévez-Velarde, Antonio Bolufé-Röhler, James Montgomery, and Stephen Chen. Evolution strategies with thresheld convergence. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 2097–2104, 2015. doi:10.1109/CEC.2015.7257143.

[96] Petr Pošík, Waltraud Huyer, and László Pál. A Comparison of Global Search Algorithms for Continuous Black Box Optimization. *Evolutionary computation*, 20(4):509–541, 2012. doi:10.1162/EVCO_a_00084.

[97] Michael J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, editor, *Numerical Analysis*, pages 144–157, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg. doi:10.1007/BFb0067703.

[98] Raphael Patrick Prager. pflacco: The R-package flacco in native Python code, 2022. Last accessed 03 May 2023. URL: https://github.com/Reiyan/pflacco.

[99] Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. Automated Algorithm Selection in Single-Objective Continuous Optimization: A Comparative Study of Deep Learning and Landscape Analysis Methods. In Günter Rudolph, Anna V. Kononova, Hernán Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tušar, editors, *Parallel Problem Solving from Nature – PPSN XVII*, pages 3–17, Cham, 2022. Springer International Publishing. doi:10.1007/978-3-031-14714-2_1.

[100] Raphael Patrick Prager and Heike Trautmann. Nullifying the Inherent Bias of Non-invariant Exploratory Landscape Analysis Features. In *Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings*, page 411–425, Berlin, Heidelberg, 2023. Springer-Verlag. doi:10.1007/978-3-031-30229-9_27.

[101] Raphael Patrick Prager and Heike Trautmann. Pflacco: Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems in Python. *Evolutionary Computation*, pages 1–25, 07 2023. doi:10.1162/evco_a_00341.

[102] Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. Towards feature-free automated algorithm selection for single-objective continuous black-box optimization. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2021. doi:10.1109/SSCI50451.2021.9660174.

[103] Lisa Pretsch, Ilya Arsenyev, Catharina Czech, and Fabian Duddeck. Interdisciplinary design optimization of compressor blades combining low- and high-fidelity models. *Struct. Multidiscip. Optim.*, 66(4), mar 2023. doi:10.1007/s00158-023-03516-w.

[104] Jeremy Rapin and Olivier Teyaud. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[105] Elena Raponi, Dario Fiumarella, Simonetta Boria, Alessandro Scattina, and Giovanni Belingardi. Methodology for parameter identification on a thermoplastic composite crash absorber by the Sequential Response Surface Method and Efficient Global Optimization. *Composite Structures*, 278:114646, 2021. doi:10.1016/j.compstruct.2021.114646.

[106] Elena Raponi, Hao Wang, Mariusz Bujny, Simonetta Boria, and Carola Doerr. High Dimensional Bayesian Optimization Assisted by Principal Component Analysis. In Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann, editors, *Parallel Problem Solving from Nature – PPSN XVI*, pages 169–183, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-58112-1_12.

[107] Ingo Rechenberg. Cybernetic solution path of an experimental problem. *Roy. Aircr. Establ., Libr. transl.*, 1122, 1965.

[108] Ingo Rechenberg. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. 47, 1973.

[109] Quentin Renau, Carola Doerr, Johann Dreo, and Benjamin Doerr. Exploratory Landscape Analysis is Strongly Sensitive to the Sampling Strategy. In Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann, editors, *Parallel Problem Solving from Nature – PPSN XVI*, pages 139–153, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-58115-2_10.

[110] Quentin Renau, Johann Dreo, Carola Doerr, and Benjamin Doerr. Expressiveness and robustness of landscape features. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 2048–2051, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3319619.3326913.

[111] Quentin Renau, Johann Dreo, Carola Doerr, and Benjamin Doerr. Towards Explainable Exploratory Landscape Analysis: Extreme Feature Selection for Classifying BBOB Functions. In Pedro A. Castillo and Juan Luis Jiménez Laredo, editors, *Applications of Evolutionary Computation*, pages 17–33, Cham, 04 2021. Springer International Publishing. doi:10.1007/978-3-030-72699-7_2.

[112] John R. Rice. The Algorithm Selection Problem. volume 15 of *Advances in Computers*, pages 65–118. Elsevier, 1976. doi:10.1016/S0065-2458(08)60520-3.

[113] Maria Laura Santoni, Elena Raponi, Renato De Leone, and Carola Doerr. Comparison of High-Dimensional Bayesian Optimization Algorithms on BBOB. *ACM Trans. Evol. Learn. Optim.*, 4(3), July 2024. doi:10.1145/3670683.

[114] Elias Schede, Jasmin Brandt, Alexander Tornede, Marcel Wever, Viktor Bengs, Eyke Hüllermeier, and Kevin Tierney. A Survey of Methods for Automated Algorithm Configuration. *Journal of Artificial Intelligence Research*, 75:425–487, 2022. doi:10.1613/jair.1.13676.

[115] David Schneider. Optimize CFD Simulations With Just a Click, 2024. Last accessed 15 November 2024. URL: https://www.ansys.com/blog/optimize-cfd-simulations-just-click.

[116] Hans-Paul Schwefel. Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik. *Diploma thesis, Technical Univ. of Berlin*, 1965.

[117] Dirk Schweim, David Wittenberg, and Franz Rothlauf. On sampling error in genetic programming. *Natural Computing*, 21:173–186, 2021. doi:10.1007/s11047-020-09828-w.

[118] Moritz Vinzent Seiler, Pascal Kerschke, and Heike Trautmann. Deep-ELA: Deep Exploratory Landscape Analysis with Self-Supervised Pretrained Transformers for Single- and Multi-Objective Continuous Optimization Problems, 2024. arXiv:2401.01192.

[119] Moritz Vinzent Seiler, Raphael Patrick Prager, Pascal Kerschke, and Heike Trautmann. A Collection of Deep Learning-based Feature-Free Approaches for Characterizing Single-Objective Continuous Fitness Landscapes, 2022. arXiv:2204.05752.

[120] Siemens Industry Software Inc. HEEDS, 2022. URL: https://plm.sw.siemens.com/de-DE/simcenter/integration-solutions/heeds/.

[121] Kate Smith-Miles and Mario Andrés Muñoz. Instance Space Analysis for Algorithm Testing: Methodology and Software Tools. *ACM Computing Surveys*, 55(12), March 2023. doi:10.1145/3572895.

[122] Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967. doi:10.1016/0041-5553(67)90144-9.

[123] James C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. doi:10.1109/9.119632.

[124] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA, 2010. Omnipress. URL: https://dl.acm.org/doi/10.5555/3104322.3104451.

[125] Nielen Stander and Kenneth J. Craig. On the robustness of a simple domain reduction scheme for simulation-based optimization. *Engineering Computations*, 19(4):431–450, 2002. doi:10.1108/02644400210430190.

[126] Nielen Stander and Kenneth J. Craig. Response surface and sensitivity-based optimization in LS-OPT: A benchmark study. In *7th International LS-DYNA Users Conference, Dearborn, MI*, 2002. URL: https://www.dynalook.com/conferences/international-conf-2002/Session_13-2.pdf.

[127] Nielen Stander, Willem Roux, Mathias Giger, Marcus Redhe, Nelya Fedorova, and Johan Haarhoff. *A Comparison of Metamodeling Techniques for Crashworthiness Optimization*, page 4489. 2004. doi:10.2514/6.2004-4489.

[128] Rainer Storn and Kenneth Price. Differential Evolution–A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997. doi:10.1023/A:1008202821328.

[129] Guangyong Sun, Ye Tian, Ruoyu Wang, Jianguang Fang, and Qing Li. Parallelized multiobjective efficient global optimization algorithm and its applications. *Structural and Multidisciplinary Optimization*, 61(2):763–786, 2020. doi:10.1007/s00158-019-02417-1.

[130] Walter A. Tackett. Mining the genetic program. *IEEE Expert*, 10(3):28–38, 1995. doi:10.1109/64.393140.

[131] André Thomaser, Anna V. Kononova, Marc-Eric Vogt, and Thomas Bäck. One-shot optimization for vehicle dynamics control systems: towards benchmarking and exploratory landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '22, page 2036–2045, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3520304.3533979.

[132] André Thomaser, Marc-Eric Vogt, Thomas Bäck, and Anna V. Kononova. Real-World Optimization Benchmark from Vehicle Dynamics: Specification of Problems in 2D and Methodology for Transferring (Meta-)Optimized Algorithm Parameters. In *Proceedings of the 15th International Joint Conference on Computational Intelligence*, pages 31–40. SCITEPRESS - Science and Technology Publications, 2023. doi:10.5220/0012158000003595.

[133] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, page 847–855, New York, NY, USA, 2013. Association for Computing Machinery. doi:10.1145/2487575.2487629.

[134] Ye Tian, Shichen Peng, Xingyi Zhang, Tobias Rodemann, Kay Chen Tan, and Yaochu Jin. A Recommender System for Metaheuristic Algorithms for Continuous Optimization Based on Deep Recurrent Neural Networks. *IEEE Transactions on Artificial Intelligence*, 1(1):5–18, 2020. doi:10.1109/TAI.2020.3022339.

[135] Ye Tian, Shichen Peng, Xingyi Zhang, Tobias Rodemann, Kay Chen Tan, and Yaochu Jin. Algorithm-Recommendation, 2020. Last accessed 15 January 2022. URL: https://github.com/BIMK/Algorithm-Recommendation.

[136] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[137] Bas van Stein. doe2vec. https://huggingface.co/models?other=doe2vec. Last accessed: October 13, 2024.

[138] Bas van Stein. doe2vec: paper release, September 2022. doi:10.5281/zenodo.7043301.

[139] Bas van Stein, Fu Xing Long, Moritz Frenzel, Peter Krause, Markus Gitterle, and Thomas Bäck. DoE2Vec: Deep-Learning Based Features for Exploratory Landscape Analysis. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, GECCO '23 Companion, page 515–518, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3583133.3590609.

[140] Bas van Stein, Hao Wang, and Thomas Bäck. Automatic Configuration of Deep Neural Networks with Parallel Efficient Global Optimization. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2019. doi:10.1109/IJCNN.2019.8851720.

[141] Bas van Stein, Hao Wang, and Thomas Bäck. Neural Network Design: Learning from Neural Architecture Search. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1341–1349. IEEE, 2020. doi:10.1109/SSCI47803.2020.9308394.

[142] Diederick Vermetten, Fabio Caraffini, Anna V. Kononova, and Thomas Bäck. Modular Differential Evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 864–872, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3583131.3590417.

[143] Diederick Vermetten, Fabio Caraffini, Bas van Stein, and Anna V. Kononova. Using structural bias to analyse the behaviour of modular CMA-ES. In Jonathan E. Fieldsend and Markus Wagner, editors, *GECCO '22: Genetic and Evolutionary Computation Conference, Companion Volume, Boston, Massachusetts, USA, July 9 - 13, 2022*, pages 1674–1682. ACM, 2022. doi:10.1145/3520304.3534035.

[144] Diederick Vermetten, Bas van Stein, Fabio Caraffini, Leandro L. Minku, and Anna V. Kononova. BIAS: A Toolbox for Benchmarking Structural Bias in the Continuous Domain. *IEEE Transactions on Evolutionary Computation*, 26(6):1380–1393, 2022. doi:10.1109/TEVC.2022.3189848.

[145] Diederick Vermetten, Hao Wang, Thomas Bäck, and Carola Doerr. Towards dynamic algorithm selection for numerical black-box optimization: investigating BBOB as a use case. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 654–662, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3377930.3390189.

[146] Diederick Vermetten, Hao Wang, Carola Doerr, and Thomas Bäck. Integrated vs. sequential approaches for selecting and tuning CMA-ES variants. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 903–912, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3377930.3389831.

[147] Diederick Vermetten, Furong Ye, Thomas Bäck, and Carola Doerr. MA-BBOB: A Problem Generator for Black-Box Optimization Using Affine Combinations and Shifts. *ACM Trans. Evol. Learn. Optim.*, June 2024. Just Accepted. doi:10.1145/3673908.

[148] Diederick Vermetten, Furong Ye, and Carola Doerr. Using Affine Combinations of BBOB Problems for Performance Assessment. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, page 873–881, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3583131.3590412.

[149] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

[150] Urban Škvorc, Tome Eftimov, and Peter Korošec. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Applied Soft Computing*, 90:106138, 2020. doi:10.1016/j.asoc.2020.106138.

[151] Urban Škvorc, Tome Eftimov, and Peter Korošec. *A Complementarity Analysis of the COCO Benchmark Problems and Artificially Generated Problems*, page 215–216. Association for Computing Machinery, New York, NY, USA, 2021. doi:10.1145/3449726.3459585.

[152] Urban Škvorc, Tome Eftimov, and Peter Korošec. The Effect of Sampling Methods on the Invariance to Function Transformations When Using Exploratory Landscape Analysis. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1139–1146, 2021. doi:10.1109/CEC45853.2021.9504739.

[153] Hao Wang and Kaifeng Yang. *Bayesian Optimization*, pages 271–297. Springer International Publishing, Cham, 2023. doi:10.1007/978-3-031-25263-1_10.

[154] Joe H. Ward Jr. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, 1963. doi:10.1080/01621459.1963.10500845.

[155] Sven Wielens. *Automatische Erstellung von Submodellen für die Crashoptimierung von Fahrzeugkarosserien.* Phd thesis, Bergische Universität Wuppertal, Wuppertal, Nordrhein-Westfalen, 2022. Available at https://www.oms.uni-wuppertal.de/fileadmin/maschbau/oms/images/Publikationen/Dissertation_Wielens.pdf.

[156] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. doi:10.1109/4235.585893.

[157] Ziyu Wu. Identifying Functions in an Automotive Crashworthiness Optimization Pipeline. Master's thesis, Technical University of Munich, Munich, Bavaria, Dec 2023.

[158] Ali R. Yildiz and Kiran N. Solanki. Multi-objective optimization of vehicle crashworthiness using a new particle swarm based approach. *The International Journal of Advanced Manufacturing Technology*, 59:367–376, 2012. doi:10.1007/s00170-011-3496-y.

[159] Meng Zhao and Jinlong Li. Tuning the hyper-parameters of CMA-ES with tree-structured Parzen estimators. In *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*, pages 613–618, 2018. doi:10.1109/ICACI.2018.8377530.

[160] Urban Škvorc, Tome Eftimov, and Peter Korošec. Transfer Learning Analysis of Multi-Class Classification for Landscape-Aware Algorithm Selection. *Mathematics*, 10(3), 2022. doi:10.3390/math10030432.

Acronyms

AAC Automated Algorithm Configuration.

ABS Anti-lock Braking System.

AE Autoencoder.

AS Algorithm Selection.

ASP Algorithm Selection Problem.

AUC Area Under the Curve.

BBO Black-Box Optimization.

BBOB Black-Box Optimization Benchmarking.

BEV Battery Electric Vehicle.

BH Benjamini-Hochberg.

BIW Body-in-White.

BO Bayesian Optimization.

CASH Combined Algorithm Selection and Hyperparameter Optimization.

CMA-ES Covariance Matrix Adaptation Evolutionary Strategy.

COCO Comparing Continuous Optimizers.

DE Differential Evolution.

DoE Design of Experiments.

EA Evolutionary Algorithm.

ECDF Empirical Cumulative Distribution Functions.

EGO Efficient Global Optimization.

EI Expected Improvement.

ELA Exploratory Landscape Analysis.

EMNA Estimation of Multivariate Normal Algorithm.

ES Evolutionary Strategy.

EuroNCAP European New Car Assessment Programme.

FE Finite Element.

GP Genetic Programming.

GPR Gaussian Process Regression.

HEBO Heteroscedastic Evolutionary Bayesian Optimization.

HPO Hyperparameter Optimization.

ICoFiS Information Content of Fitness Sequences.

IOH Iterative Optimization Heuristic.

irace Iterated Racing.

ISA Instance Space Analysis.

KL Kullback–Leibler.

KPCA-BO Kernel Principal Component Analysis assisted Bayesian Optimization.

KS Kolmogorov-Smirnov.

LCB Lower Confidence Bound.

LHS Latin Hypercube Sampling.

MA-BBOB Many-Affine Black-Box Optimization Benchmarking.

MELS Modified Extensible Lattice Sequence.

MIP-EGO Mixed-Integer Parallel Efficient Global Optimization.

ML Machine Learning.

ModCMA Modular Covariance Matrix Adaptation Evolutionary Strategy.

ModDE Modular Differential Evolution.

MSE Mean Squared Error.

MWU Mann-Whitney U.

NBC Nearest Better Clustering.

nDCG normalized Discounted Cumulative Gain.

NN Neural Network.

PCA Principal Component Analysis.

PCA-BO Principal Component Analysis assisted Bayesian Optimization.

PI Probability of Improvement.

PSO Particle Swarm Optimization.

RBF Radial Basis Function.

RCobyla Constrained Optimization by Linear Approximation with Random Restart.

ReLU Rectified Linear Unit.

RF Random Forest.

RGF Randomly Generated Function.

rMC reduced Multiple Channel.

RS Random Search.

RSM Response Surface Method.

SAASBO Sparse Axis-Aligned Subspace Bayesian Optimization.

SBS Single Best Solver.

SMAC Sequential Model-based Algorithm Configuration.

SPSA Simultaneous Perturbation Stochastic Approximation.

SRSM Successive Response Surface Method.

t-SNE t-distributed Stochastic Neighbor Embedding.

TPE Tree-structured Parzen Estimator.

TuRBO Trust Region Bayesian Optimization.

UCB Upper Confidence Bound.

UMAP Uniform Manifold Approximation Mapping.

VAE Variational Autoencoder.

VBS Virtual Best Solver.

Samenvatting

Het zo optimaal oplossen van problemen met beperkingen in de echte wereld (BBO), die duur zijn om te evalueren ofwel in tijd ofwel in rekenkosten, is zeer uitdagend. Hoewel er in de loop der jaren verschillende BBO-algoritmen zijn geïntroduceerd, zoals Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) en Differential Evolution (DE), is er meestal een aanzienlijk functie-evaluatiebudget nodig voor een optimalisatieconvergentie, wat minder praktisch is voor echte toepassingen die dure functie-evaluaties gebruiken. Het optimaliseren van de botsveiligheid van auto's met behulp van dure Finite Element (FE) simulaties in de auto-industrie is bijvoorbeeld een representatief voorbeeld van echte BBO problemen, die typisch zeer niet-lineair, discontinu en hoog-dimensionaal zijn. Door de strengere regels voor verkeersveiligheid, opgelegd door de overheid, en het feit dat verschillende functies en functionaliteiten geïntegreerd moeten worden om de klanttevredenheid beter te behouden, is het efficiënt oplossen van voertuigontwerpproblemen cruciaal, maar wel een uitdaging. Ondertussen verliezen geavanceerde optimalisatiebenaderingen zoals Response Surface Method (RSM) en Successive Response Surface Method (SRSM) geleidelijk hun effectiviteit in het oplossen van de steeds geavanceerdere voertuigontwerpproblemen. Optimalisatiebenaderingen die op een efficiënte manier botsveiligheids problemen efficiënt kunnen oplossen krijgen daarom steeds meer aandacht in de auto-industrie.

Gemotiveerd om de bestaande limitaties op te lossen, onderzoeken we een geautomatiseerde optimalisatie aanpak voor het efficiënt oplossen van echte dure BBO problemen met behulp van een beperkt functie-evaluatie budget binnen het bestek van dit proefschrift. In wezen willen we praktijkmensen helpen bij het afstemmen van optimalisatieconfiguraties voor het optimaal oplossen van hun toepassingen met behulp van beperkingen in de echte wereld. Door gebruik te maken van enkele goedkoop te evalueren representatieve functies die behoren tot dezelfde optimalisatieprobleemklassen als de op te lossen BBO problemen, kunnen bijna optimale optimalisatie-

configuraties geïdentificeerd worden tegen relatief lage rekenkosten, bijvoorbeeld met behulp van Hyperparameter Optimization (HPO), voordat de eigenlijke optimalisatieruns worden uitgevoerd met behulp van dure functie-evaluaties. Uiteindelijk kunnen de dure BBO problemen efficiënt opgelost worden met behulp van de verfijnde optimalisatieconfiguraties.

Om representatieve functies te identificeren die tot dezelfde optimalisatieprobleemklassen behoren, overwegen we om de kenmerken van het optimalisatielandschap van BBO problemen te kwantificeren met behulp van Exploratory Landscape Analysis (ELA). In principe behoren BBO problemen tot dezelfde optimalisatieprobleemklassen als testfuncties met vergelijkbare ELA-kenmerken. Gebaseerd op onze analyse van 20 auto-ongeluk probleeminstanties van verschillende crashscenario's en probleemdimensies, laten we zien dat de auto-ongeluk problemen behoren tot optimalisatie probleemklassen die verschillen van de veelgebruikte Black-Box Optimization Benchmarking (BBOB) functies. Met andere woorden, de BBOB-functies zijn onvoldoende representatief voor echte auto-ongelukken en zijn dus ongeschikt om als representatieve functies te dienen. Aan de andere kant lijken de auto-ongelukken het meest op optimalisatieprobleemklassen van Randomly Generated Function (RGF) gegenereerd met behulp van een boomgebaseerde willekeurige functiegenerator in termen van ELA-kenmerken. In feite stellen we voor om dergelijke RGF's te gebruiken als schaalbare en goedkoop te evalueren representatieve functies voor echte dure BBO-problemen, bijvoorbeeld voor HPO-doeleinden. Verder onderzoek toont aan dat RGF's veelbelovende mogelijkheden hebben in het schatten van de werkelijke prestaties van verschillende optimalisatieconfiguraties op ongeziene BBO problemen. Anders gezegd, goed presterende optimalisatieconfiguraties voor echte dure BBO problemen kunnen inderdaad geïdentificeerd worden door gebruik te maken van RGFs als representatieve functies.

Bij evaluatie op de BBOB-suite kunnen we beter presterende optimalisatieconfiguraties identificeren dan de standaardconfiguratie met behulp van de voorgestelde optimalisatieaanpak, in overeenstemming met onze motivatie. Opmerkelijk is dat de optimale optimalisatieconfiguraties zelfs concurrerend zijn met de Single Best Solver (SBS) op sommige BBOB-functies. Hier worden een selectieproces van geschikte representatieve functies en een opsplitsing van training en testen tijdens HPO geïmplementeerd om de algehele robuustheid bij het afstemmen van optimalisatieconfiguraties te verbeteren. Nog belangrijker is dat de configuraties die met onze aanpak zijn verfijnd, beter presteren dan de standaard BO-configuratie en RSM voor het oplossen van een zijwaarts botsprobleem bij auto's in termen van de best gevonden oplossing en convergentiesnelheid. Vooral voor het oplossen van complexe crashfuncties met een

beperkt evaluatiebudget kunnen aanzienlijk betere oplossingen worden gevonden met onze aanpak, bijvoorbeeld in vergelijking met SRSM.

Concluderend tonen onze resultaten aan dat de voorgestelde geautomatiseerde optimalisatiepijplijn een bemoedigend potentieel heeft voor het efficiënt oplossen van echte dure BBO-problemen, zoals voertuigontwerpproblemen. Aangezien de voorgestelde optimalisatiebenadering relatief goed kan presteren op de BBOB-suite, die een verscheidenheid aan optimalisatieprobleemklassen omvat, zijn we ervan overtuigd dat onze benadering goed kan veralgemenizeren naar andere BBO-domeinen, op voorwaarde dat ze voldoende worden gerepresenteerd door de BBOB-functies. Met verdere aanpassingen denken we dat onze aanpak uitgebreid kan worden naar het efficiënt oplossen van andere dure BBO problemen in de echte wereld, zoals scheepsontwerpen.

Summary

Optimally solving real-world expensive Black-Box Optimization (BBO) problems w.r.t. real-world constraints, such as wall-clock time and computational costs, is extremely difficult and tedious. While a variety of BBO algorithms have been introduced over the years, such as Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) and Differential Evolution (DE), a considerable function evaluation budget is usually necessary for an optimization convergence, which is less practical for real-world applications using expensive function evaluations. Automotive crashworthiness optimization using expensive Finite Element (FE) simulations in the automotive industry, for instance, is a representative example of real-world BBO problems, which is typically highly nonlinear, discontinuous, and high-dimensional. Due to the stricter regulations on road safety imposed by authorities and the fact that various features and functionalities must be integrated to better retain customer satisfaction, an efficient solving of vehicle design problems is critical, yet challenging. Meanwhile, state-of-the-art optimization approaches like Response Surface Method (RSM) and Successive Response Surface Method (SRSM) are gradually losing their effectiveness in solving the increasingly sophisticated vehicle design problems. Consequently, optimization approaches that can efficiently solve crashworthiness optimization problems have gained growing attention in the automotive industry.

Motivated to fill the gaps, we investigate an automated optimization approach for optimally solving real-world expensive BBO problems using a limited function evaluation budget within the scope of this thesis. Essentially, we aim to assist practitioners in automatically fine-tuning optimization configurations for optimally solving their applications w.r.t. real-world constraints. By exploiting some cheap-to-evaluate representative functions that belong to the same optimization problem classes as the BBO problems to-be-solved, nearly optimal optimization configurations can be identified at a relatively low computational expense, e.g., using Hyperparameter Optimization

(HPO), prior to the actual optimization runs using expensive function evaluations. Eventually, the expensive BBO problems can be efficiently solved using the fine-tuned optimization configurations.

To identify representative functions that belong to the same optimization problem classes, we consider quantifying the optimization landscape characteristics of BBO problems using Exploratory Landscape Analysis (ELA). Principally, BBO problems belong to the same optimization problem classes as test functions with similar ELA features. Based on our analysis on 20 automotive crash problem instances of different crash scenarios and problem dimensionalities, we show that the automotive crash problems belong to optimization problem classes that are different from the widely-used Black-Box Optimization Benchmarking (BBOB) functions. In other words, the BBOB functions are insufficiently representative for real-world automotive crash problems, and thus, they are inappropriate to serve as representative functions. On the other hand, the automotive crash problems are most similar to the optimization problem classes of Randomly Generated Function (RGF) created using a tree-based random function generator in terms of ELA features. In fact, we propose considering such RGFs as scalable and cheap-to-evaluate representative functions for real-world expensive BBO problems, e.g., for HPO purposes. Further investigations demonstrate that RGFs have promising potential in estimating the actual performance of different optimization configurations on unseen BBO problems. Put differently, well-performing optimization configurations for real-world expensive BBO problems can be indeed identified using RGFs as representative functions.

When evaluated on the BBOB suite, we can identify better performing optimization configurations than the default configuration using the proposed optimization approach, in line with our motivation. Remarkably, the optimal optimization configurations are even competitive against the Single Best Solver (SBS) on some BBOB functions. Here, a selection process of appropriate representative functions and a training-testing split during HPO are additionally implemented, to improve the overall robustness in fine-tuning optimization configurations. More importantly, the Bayesian Optimization (BO) configurations fine-tuned using our approach can outperform the BO default configuration and RSM for solving an automotive side crash problem in terms of the best-found-solution and convergence speed. Especially for solving complex crash functions using a limited function evaluation budget, such as a multi-pole impact problem, significantly better solutions can be found using our approach, e.g., compared to SRSM.

In conclusion, our results show that the proposed automated optimization pipeline

has an encouraging potential for an efficient solving of real-world expensive BBO problems, such as vehicle design problems. Since the proposed optimization approach can perform relatively well on the BBOB suite, covering a variety of optimization problem classes, we are confident that our approach can generalize well to other BBO domains that are sufficiently represented by the BBOB functions. With further modifications, we believe that our approach can be extended to efficiently solving other real-world expensive BBO problems, such as ship designs.

Acknowledgements

Embarking on the journey of a PhD has been one of the most transformative experiences of my life. It has been marked by challenges and hardships, all made possible by the constant support and guidance of countless remarkable individuals. With great pleasure and honor, I am proud to share my achievements together with them.

First and foremost, I would like to express my heartfelt gratitude to my supervisors, namely Prof. Dr. Thomas Bäck, Prof. Dr.-Ing. Markus Gitterle, Dr. Niki van Stein, Moritz Frenzel, and Peter Krause. It is their wisdom, encouragement, and mentorship that helped me navigate numerous challenges and achieve our shared goals, fortifying my determination and allowing me to push the boundaries of my research work. I still vividly recall the immense effort we dedicated to writing my first conference paper, which eventually earned the Best Paper Award.

I am also profoundly grateful to my collaborators, whose innovative ideas and insightful discussions have enriched my academic journey. Working with Dr. Anna V. Kononova, Dr. Diederick Vermetten, Dr. Roy de Winter, Dr. Andre Thomaser, Dr. rer. nat. Roman Kalkreuth, and Dr. Kaifeng Yang has been a enjoyable experience. Their contributions have not only enhanced the quality of my research work, but also helped me develop essential academic skills.

Furthermore, I am sincerely thankful to BMW Group for their generous support, providing the infrastructure, resources, and financial backing necessary for my research and conference travels. The trust that I received from my mangers has been invaluable, especially Gerhard Fichtinger, Matthias Nossek, and Vitor Cores Finotto, enabling me to transform theoretical insights into practical applications.

My gratitude extends to Prof. Dr.-Ing. Fabian Duddeck, Dr. Elena Raponi, and Arne Kaps, whose expertise and dedication were pivotal as we supervised two master students, Raúl San Miguel Peñas and Ziyu Wu, at the Technical University of Munich. Their passion for research and commitment to mentoring young scholars have been

truly inspiring, and I am thankful for the opportunity to learn and grow alongside them.

While the list of individuals who have positively influenced my journey could extend much further, I wish to acknowledge a few more exceptional people: Prof. Dr.-Ing. Axel Schumacher, Dr. Hao Wang, Sylvia Maas, Hestia Tamboer, Jacob de Nobel, Jiawen Kong, Sophie Cram, Sonja Schlenz, Andrea Bonfanti, Christian Büttner, and Christian Bohnenberger.

Lastly, I am eternally grateful to my parents, family, and friends, whose moral support and encouragement have been the bedrock of my perseverance. Their belief in me and unwavering love have been a constant source of comfort and strength.

This PhD journey has been a defining chapter in my life, filled with intellectual growth and personal development, which will always hold a special place in my heart. To all the wonderful people mentioned above, thank you for being an integral part of this wonderful experience.

Curriculum Vitae

Fu Xing Long was born in Pulau Pinang, Malaysia in 1993. After finishing his high-school education, Fu Xing continued his higher education in Germany in the engineering field. Granted with a scholarship award sponsored by the Malaysia government, Fu Xing has completed his bachelor's degree in mechanical engineering at Hochschule Karlsruhe in 2018 and master's degree in computational engineering at Technische Universität Darmstadt in 2021. To pursue his dream and passion for scientific research with strong industrial applications, Fu Xing decided to join the ProMotion Program offered by BMW Group as PhD candidate since May 2021, under the supervision of Prof. Dr. Thomas Bäck, Prof. Dr.-Ing. Markus Gitterle (Munich University of Applied Sciences, Germany), and Dr. Niki van Stein. His research interests mainly lie on black-box optimization, fitness landscape analysis, and automated fine-tuning of optimization algorithms. During his PhD studies, he took courses in scientific conduct, among others.