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

Long, F.X.

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

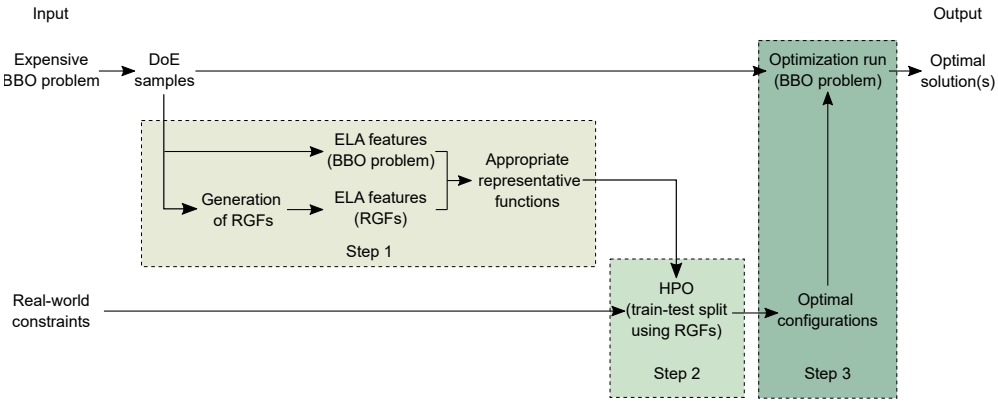


Figure 5.1: Overview of our surrogate-based landscape-aware optimization pipeline proposed for real-world expensive BBO problems, consisting of altogether three main steps. By exploiting some cheap-to-evaluate RGFs with similar optimization landscape characteristics in terms of ELA features as representations 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.

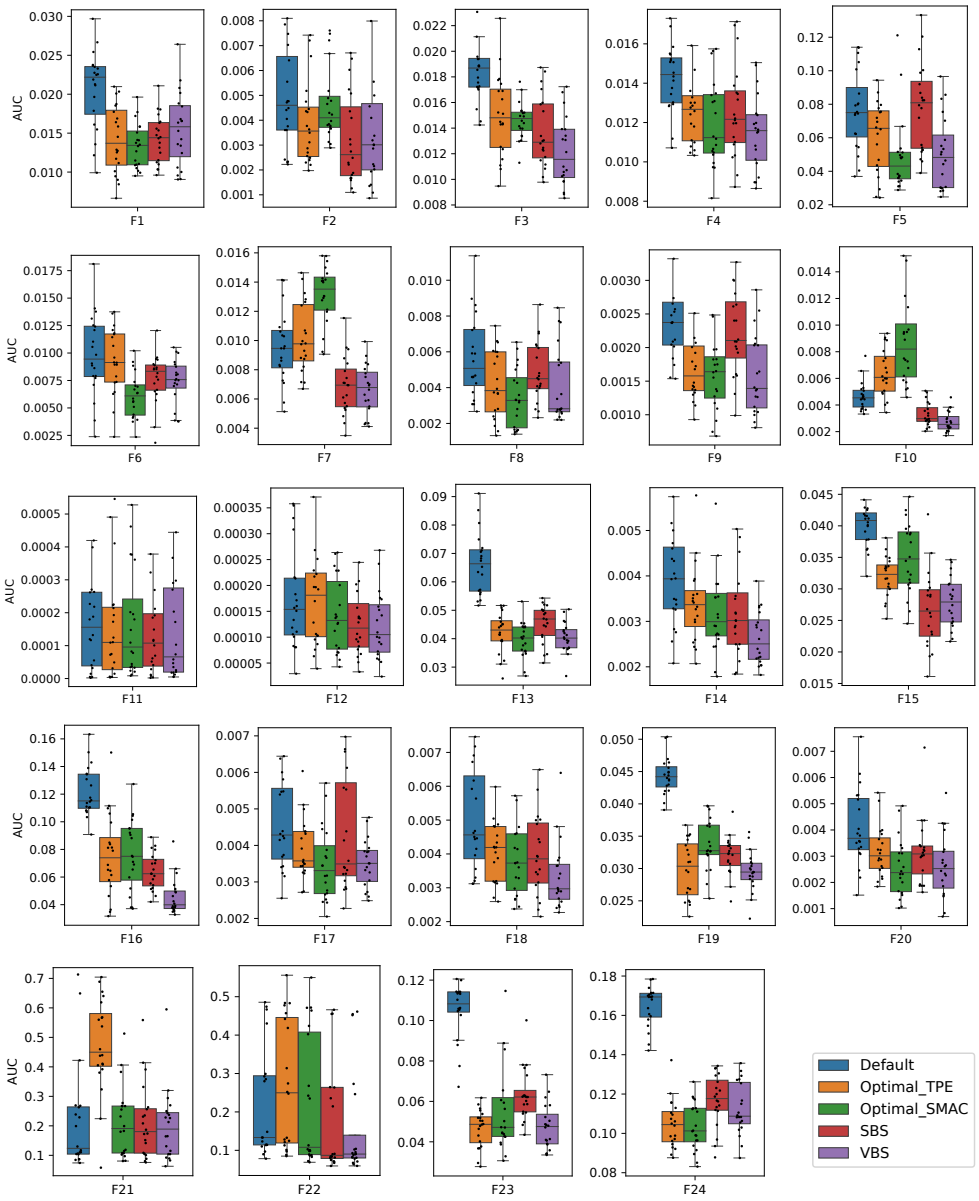


Figure 5.2: Performance of different ModCMA configurations on 24 BBOB functions in 20-d, using a repetition of 20 times for each configuration. The configuration having a smaller AUC is better. *Legend:* Default configuration, optimal configurations identified using TPE or SMAC, SBS, and VBS. Figures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

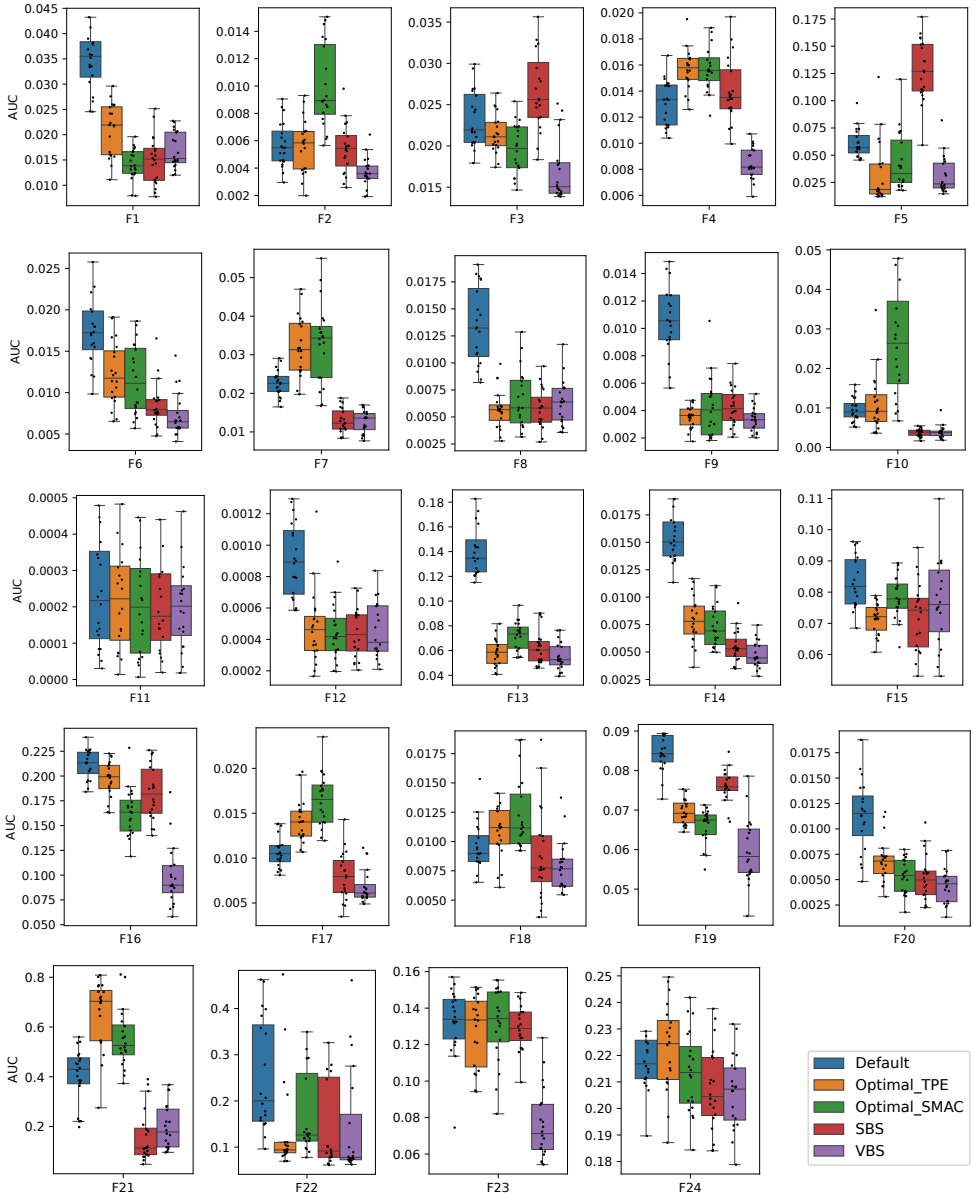


Figure 5.3: Performance of different ModDE configurations on 24 BBOB functions in 20-d, using a repetition of 20 times for each configuration. The configuration having a smaller AUC is better. *Legend:* Default configuration, optimal configurations identified using TPE or SMAC, SBS, and VBS. Figures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

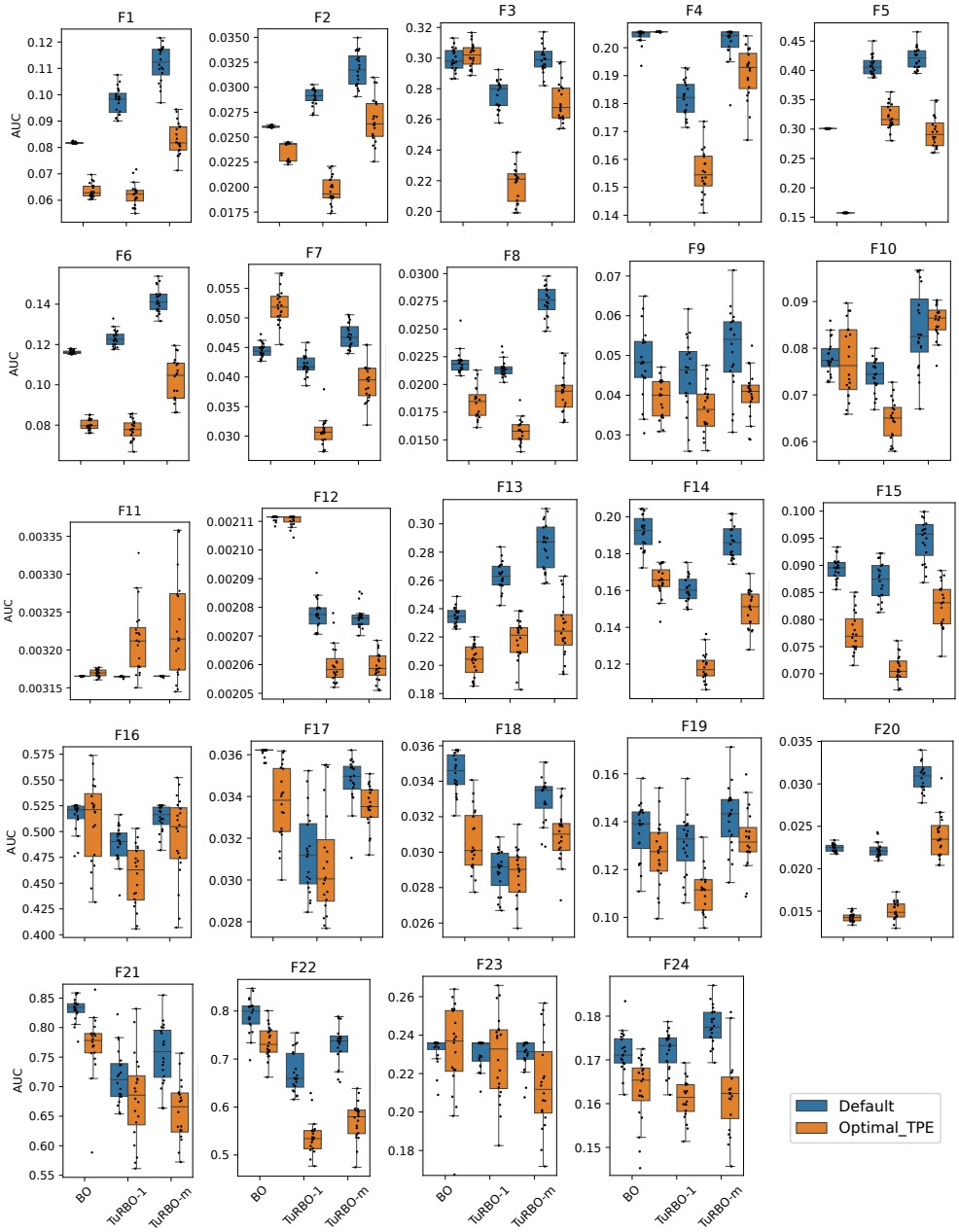


Figure 5.4: Performance of different configurations for three BO variants on 24 BBOB functions in 20-d, using a repetition of 20 times for each configuration. The configuration having a smaller AUC is better. *Legend:* Default configuration and optimal configurations identified using TPE. Figures taken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

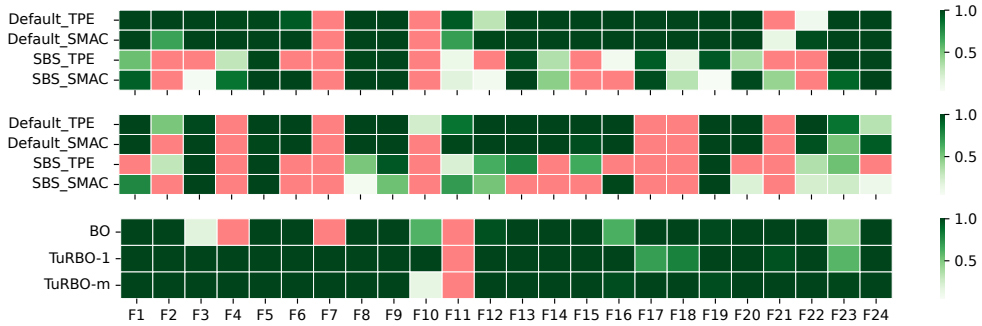


Figure 5.5: Pairwise performance comparison between the optimal configurations identified using TPE or SMAC in our approach against the default configuration and SBS on 24 BBOB functions in 20-d for ModCMA (*top*), ModDE (*middle*), and three BO variants (*bottom*). Based on the Wilcoxon signed-rank 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

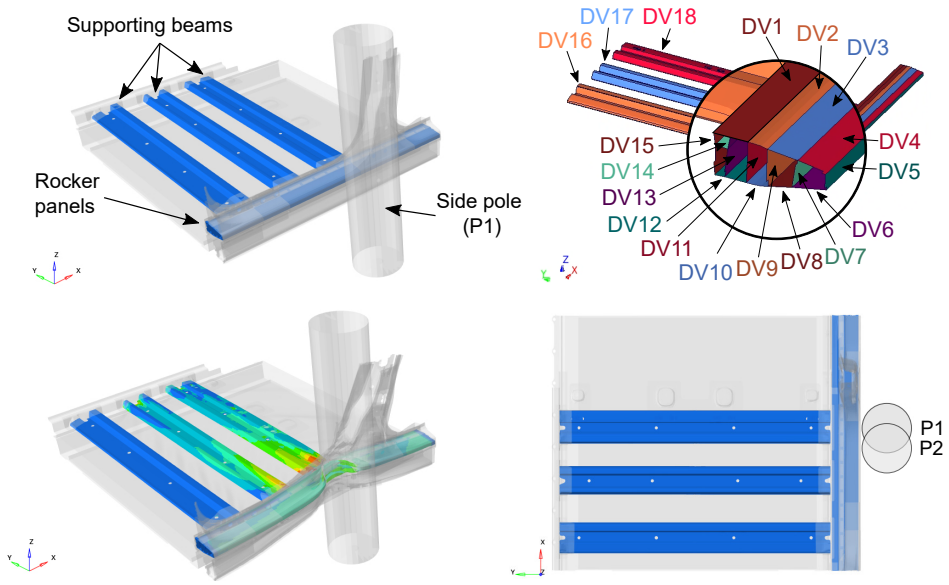


Figure 5.6: Overview of the FE submodel for an automotive side crash against a rigid pole. In BEVs, the battery compartment is commonly installed between the rocker panels on both sides and underneath the supporting beams. *Top left:* The thicknesses of rocker panels and supporting beams are considered 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.

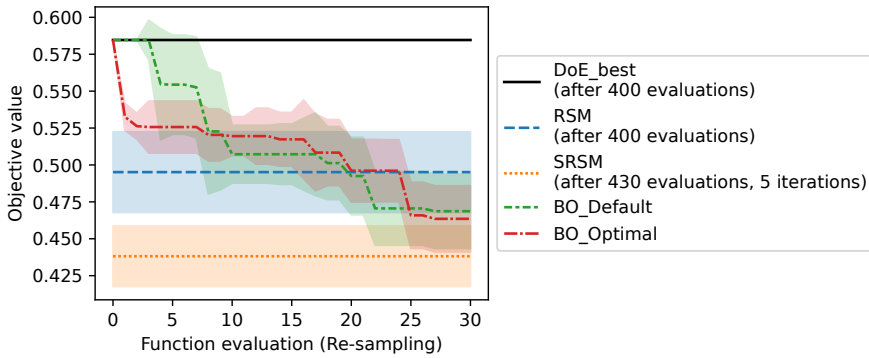


Figure 5.7: Performance comparison between one-shot optimization (*black*), RSM (*blue*), SRSM (*orange*), default BO configuration (*green*), and optimal BO configuration identified using our approach (*red*) for the single pole impact problem (Equation 5.1). While the best-found solution is presented 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).

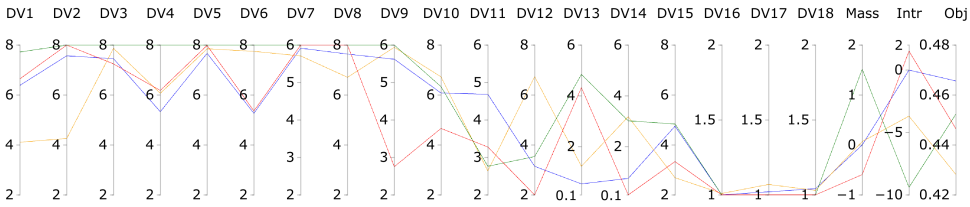


Figure 5.8: Comparison of vehicle designs for the single pole impact problem, focusing on the best optimization run using RSM (*blue*), SRSM (*orange*), default BO configuration (*green*), and optimal BO configuration identified using our approach (*red*). The upper and lower bound of the design space (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.

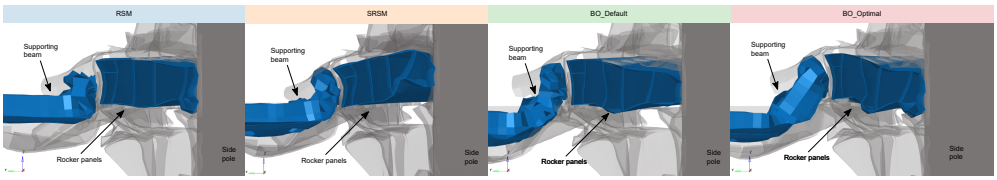


Figure 5.9: Structural deformation of the FE submodel for the single pole impact problem. The cross-section of different vehicle designs identified using RSM, SRSM, default BO, and optimal BO configuration are shown. The same setting is applied for all simulation snapshots, such as deformation scaling,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

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

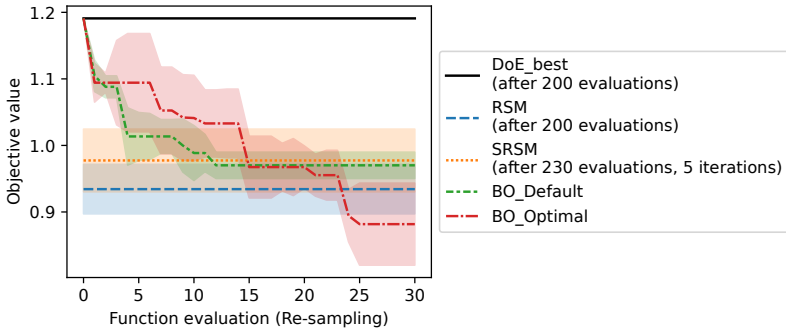
where $\alpha = 2$ for an equal weighting between mass and intrusions and $u'_{i\in\{1,2\}}$ represents the intrusion for impact position `P1` and `P2`, which is computed using the same normalization as in Equation 5.1. For the multi-pole impact, a vehicle design must be separately evaluated using two 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

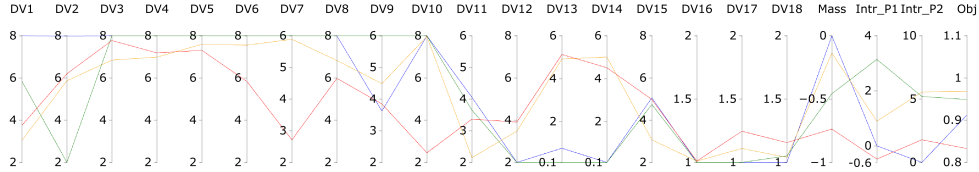
the single pole impact problem, here the performance of SRSM is worse than RSM in identifying a better vehicle design, despite the fact that better solutions could be possibly discovered over iterations. We believe that the complex multi-pole crash function might be poorly approximated in SRSM, considering 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).

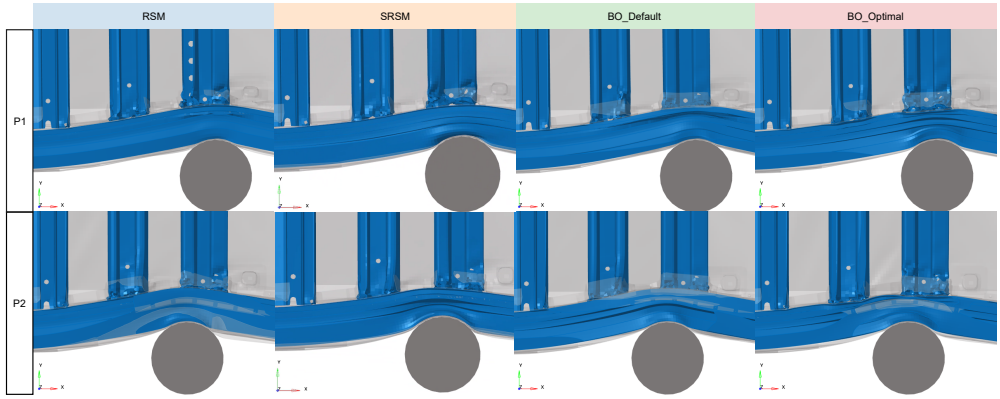
In summary, our optimization approach has promising potential in improving the performance of optimization algorithms for an optimal solving of real-world expensive BBO problems, by fine-tuning optimization configurations based on some representative functions. Particularly when solving complex BBO problems 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:

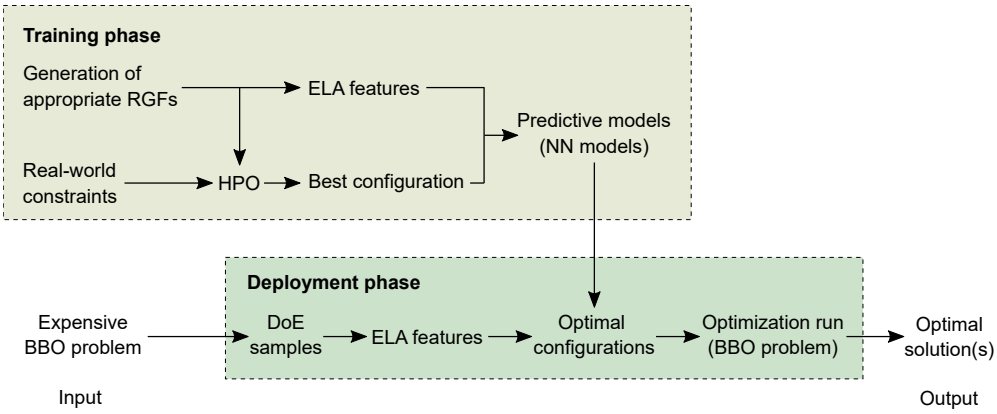


Figure 5.11: Overview of the proposed landscape-aware HPO approach that can identify optimal configurations for real-world expensive BBO problems using some pre-trained predictive models. During the training phase, based on a preferably large set of RGFs, their ELA features and optimal configurations 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.

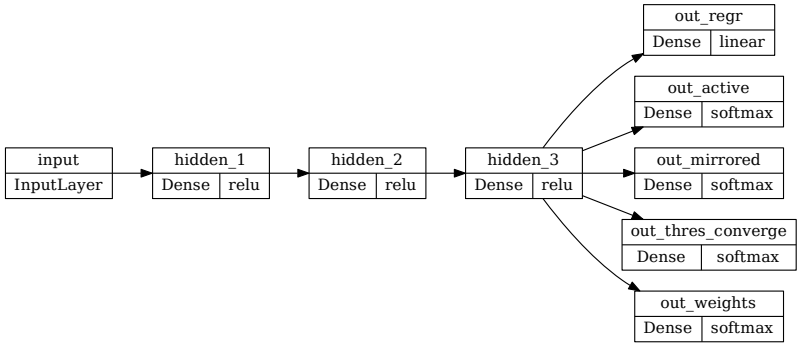


Figure 5.12: Example of the architecture of a dense NN model for mixed regression and classification. *From left to right:* An input layer, three hidden layers, and several output layers, where one output layer is assigned for regression task and four layers for classification tasks. Figure taken from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

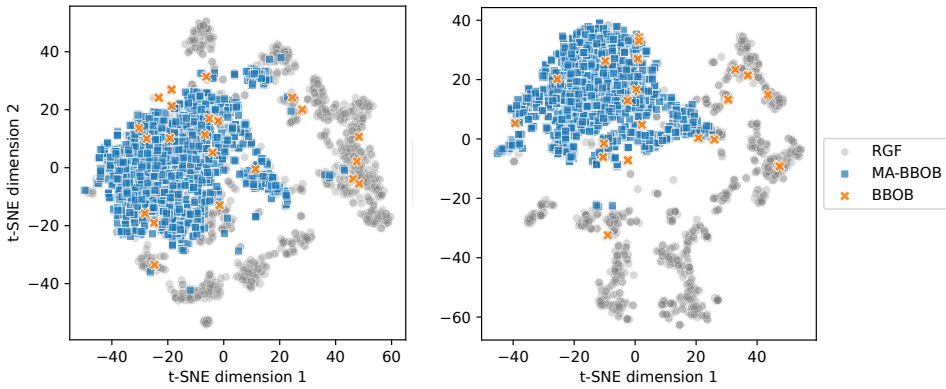


Figure 5.13: Projection of the high-dimensional ELA feature space to a 2-d visualization using the t-SNE approach for 1 000 RGFs (*gray*), 1 000 MA-BBOB (*blue*), and 24 BBOB functions (*orange*) in 5-d (*left*) and 20-d (*right*), based on a similar approach as in Section 4.1.2. Figure taken from [71].

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-

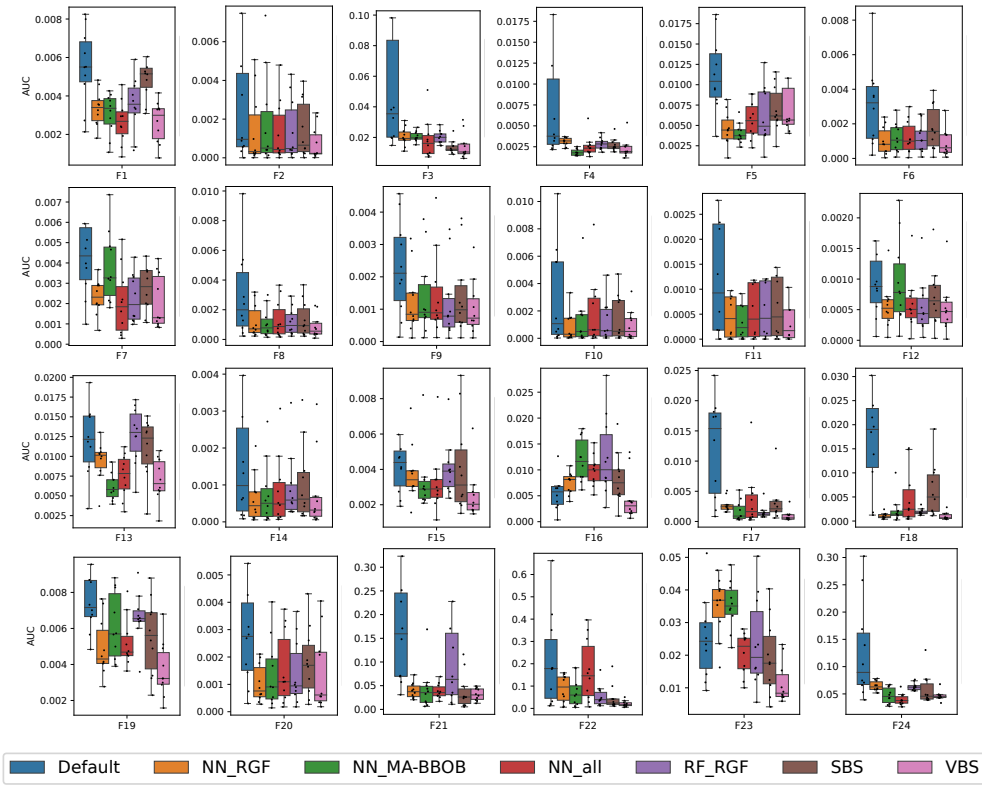


Figure 5.14: Performance of different ModCMA configurations for 24 BBOB functions in 5-*d*, using a repetition of 10 times. The configuration having a smaller AUC is better. *Legend:* Default configuration, configuration predicted by NN models trained using RGFs, by NN models trained using MA-BBOB functions,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

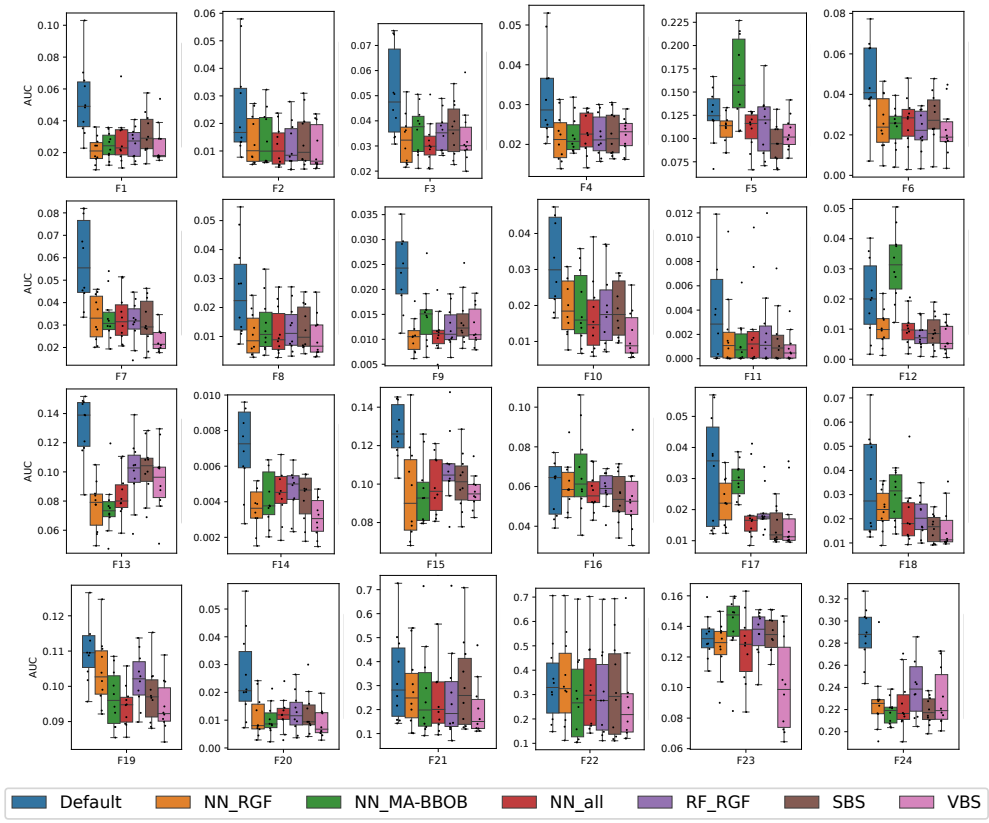


Figure 5.15: Performance of different ModCMA configurations for 24 BBOB functions in 20-d, using a repetition of 10 times. The configuration having a smaller AUC is better. *Legend:* Default configuration, configuration predicted by NN models trained using RGFs, by NN models trained using MA-BBOB functions,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

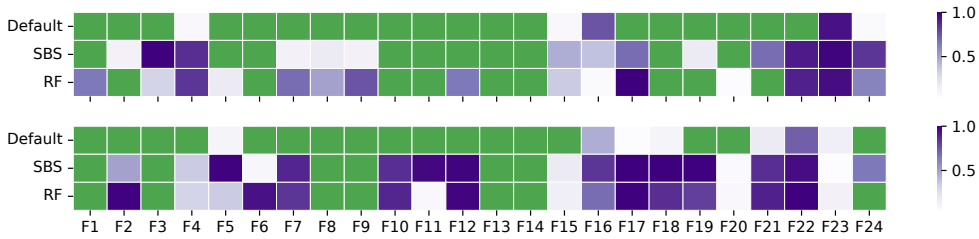


Figure 5.16: Pairwise performance comparison between the optimal configurations identified using NN models against the default configuration, SBS, and RF models for 24 BBOB functions in 5d (*top*) and 20d (*bottom*). Based on the Wilcoxon signed-rank test, a green color indicates that there is statistically 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].