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	$(x_1, \dots, x_d)$
	`x1`	First variable	x_1
	`xt`	Translated decision variable	$(x_2, \dots, x_d, 0)$
	`xr`	Rotated decision variable	$\mathbf{xr}$
	`index`	Index vector	$(1, \dots, d)$
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	$\lceil x \rceil$
	`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	$(\sum_{i=1}^{1} x_i, \dots, \sum_{i=1}^{d} x_i)$
	`prod`	Product of vector	$\prod_{i=1}^{d} x_i$
	`max`	Maximum value of vector	$\max_{i=1,\dots,d} x_i$

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 $(x_1, \dots, x_d)$ by $(x_1, \dots, x_d) - x_1$	0.05
6	Linkages between each two contiguous variables	Replace $(x_1, \dots, x_d)$ by $(x_1, \dots, x_d) - (x_2, \dots, x_d, 0)$	0.05
7	Complex linkages between all the variables	Replace $(x_1, \dots, x_d)$ by $\mathbf{xr}$	0.05
8	Different optimal values of the variables	Replace $(x_1, \dots, x_d)$ by $(1 \cdot x_1, \dots, d \cdot x_d)$	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-

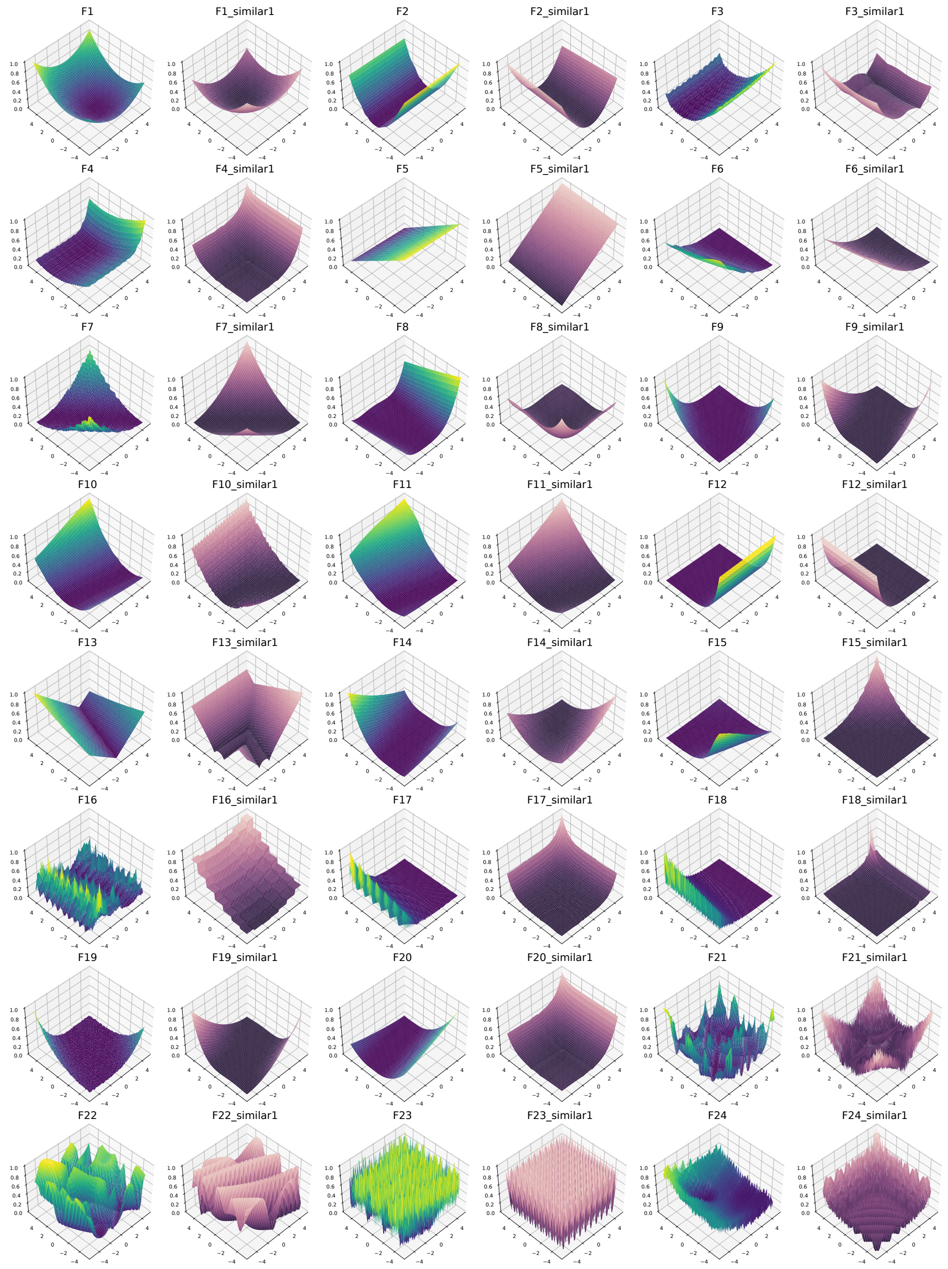


Figure 4.1: Visual comparison of the optimization landscape between pairs of 24 BBOB functions in 2-d (*left; yellow-green*) and RGFs identified as representative functions based on ELA features (*right; beige-purple*). The search space $[-5, 5]^2$ is shown on the in-plane axis, while the min-max normalized 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.

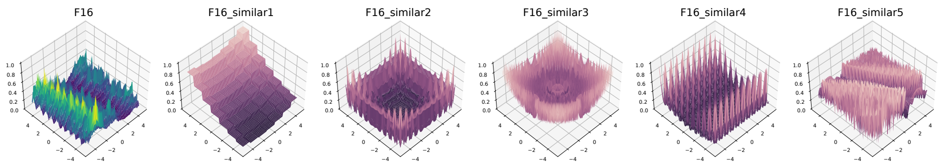


Figure 4.2: Visual comparison of the optimization landscape between F16 in 2-d (*yellow-green*) and the first five RGFs with the most similar landscape characteristics (*beige-purple*). The RGFs are sorted in ascending order from `similar_1` to `similar_5`, starting with the one having the smallest difference 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:

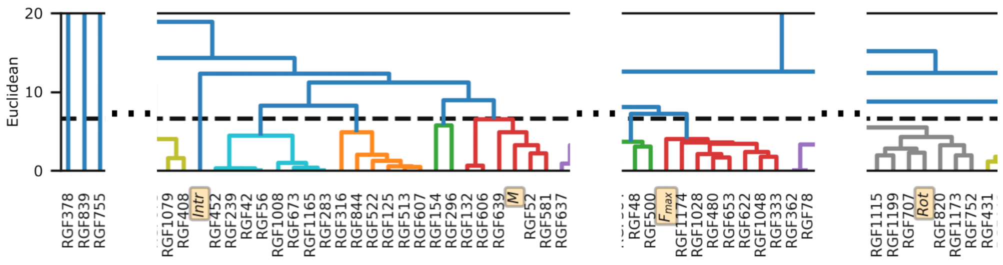


Figure 4.3: Clustering pattern of the four crash functions and 1 000 RGF (labeled from `RGF1` to `RGF1 000`) for the problem instance `Crash_2`. Only relevant sections of the clustering pattern and y-axis are shown here due to the limited space. The labels of crash functions are highlighted (*orange*) 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-

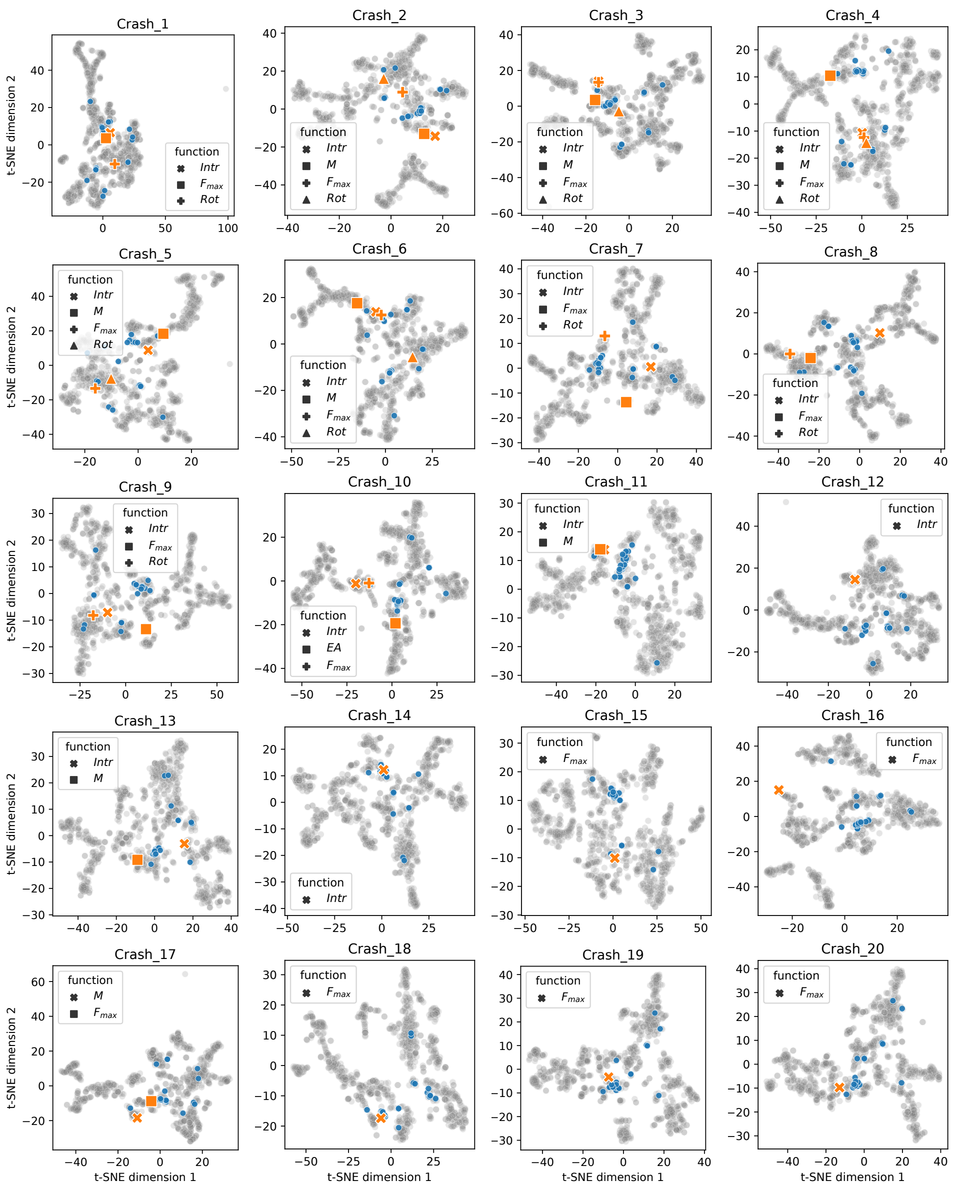


Figure 4.4: Visualization of the 2-d ELA feature space for the crash functions (*orange*), 24 BBOB functions (*blue*), and 1 000 RGFs (*gray*) using the t-SNE approach for all 20 automotive crash problems. Each dot represents a problem instance. Figure taken from [73].

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.

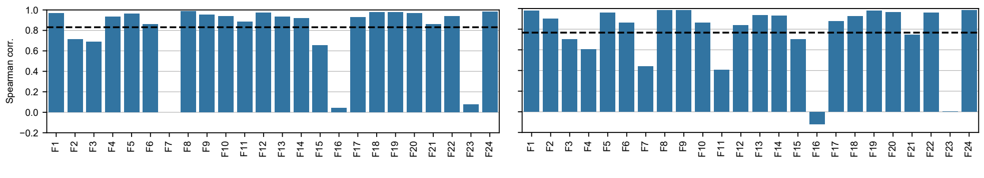


Figure 4.5: Spearman's correlation based on the ranking of ModCMA configurations between 24 BBOB functions and their representative functions, using the best-found optimization solution (*left*) and AUC (*right*) as performance metric. The mean correlation across all BBOB functions is marked with a dashed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

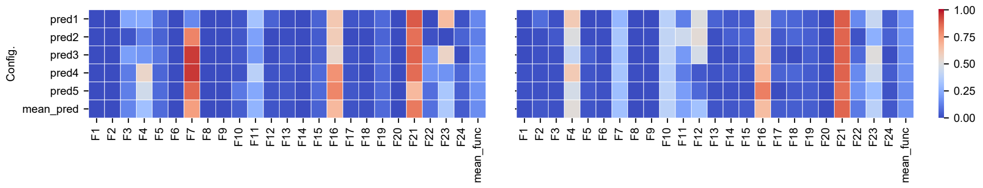


Figure 4.6: Performance of the predicted best five ModCMA configurations on 24 BBOB functions, rescaled between 0 (the best; *blue*) and 1 (the worst; *red*), using the best-found optimization solution (*left*) and AUC (*right*) as performance metric. An extra row is included for the mean across all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

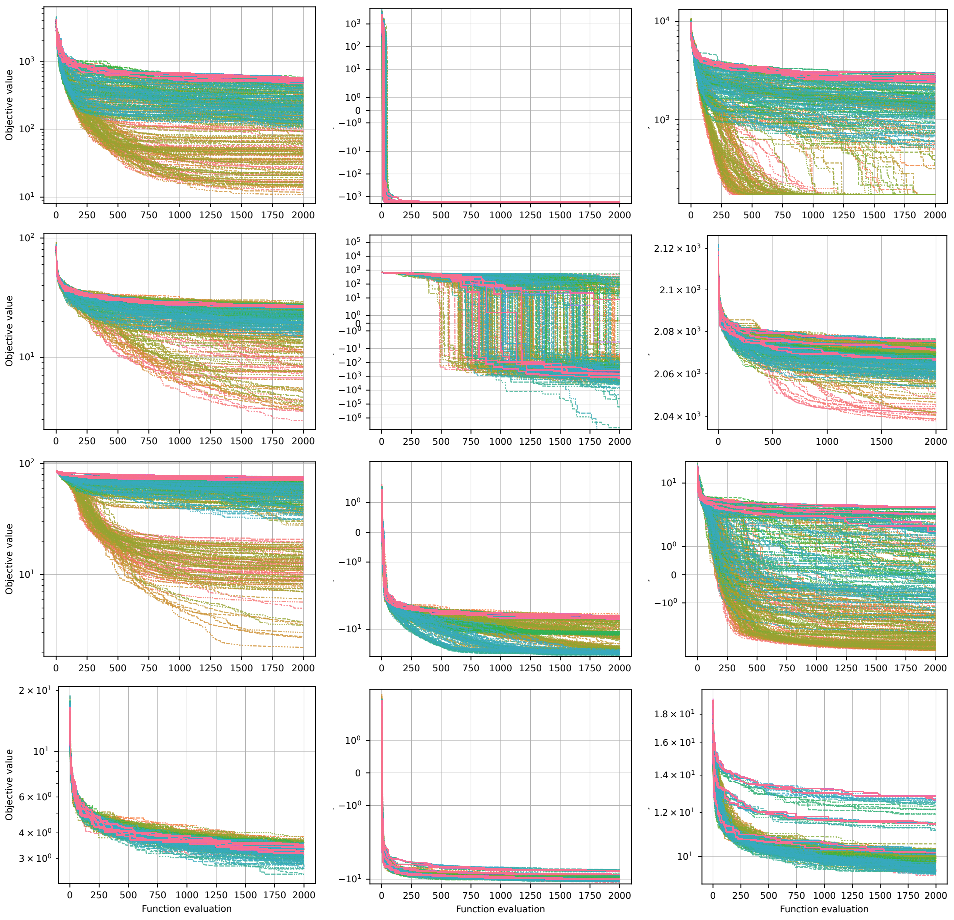
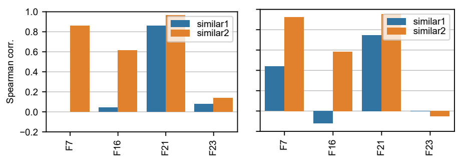
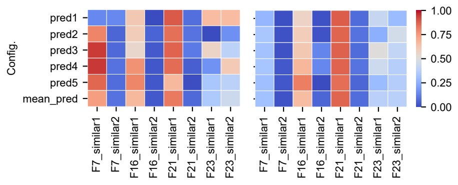


Figure 4.7: Optimization runs of altogether 972 ModCMA configurations for F7, F16, F21, and F23 (*top to bottom*; *left*), their most similar RGFs (*middle*), and their second most similar RGFs (*right*), respectively. The objective values in log-scale are shown on the y-axis, while the number of function 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.

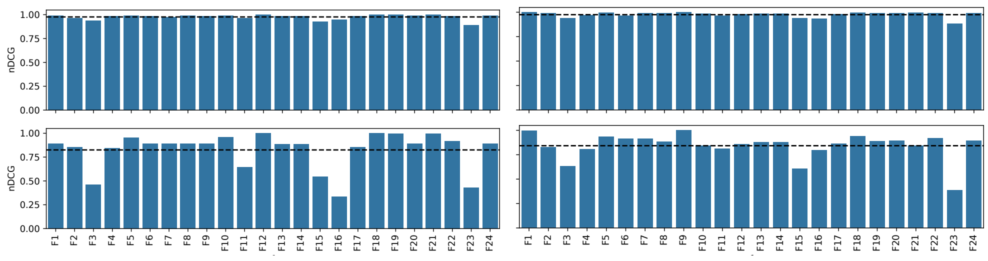
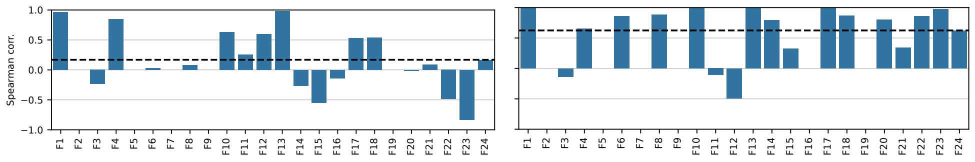


Figure 4.9: Ranking quality of ModCMA configurations based on the nDCG method, when all configurations (*top*) or only the top five configurations (*bottom*) are considered, using the best-found optimization solution (*left*) and AUC (*right*) as performance metric. Here, the second most similar RGFs 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].

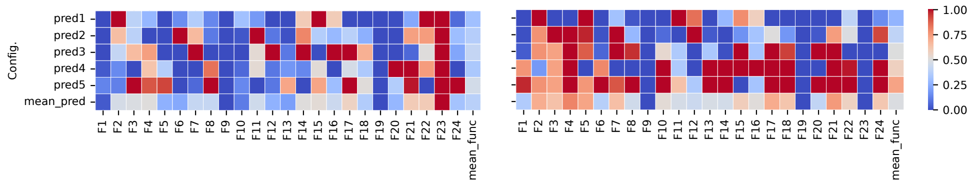
4.2.3 Optimization Performance of BO

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.

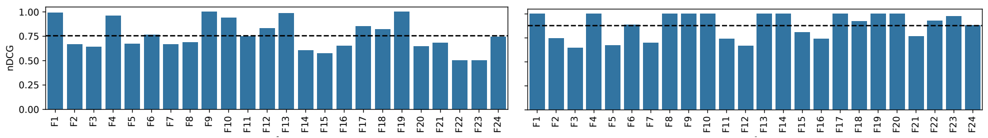
For some of the BBOB functions, such as F2, F5, F7, F9, F12, F15, F16, and F23, the best-found solution within the total evaluation budget belongs to the initial DoE samples, which are considered for the training of GPR models. In other words, no optimization solution better than the best DoE sample 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}.

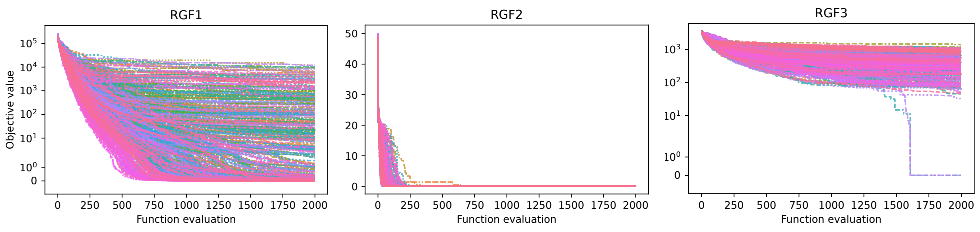


Figure 4.11: Optimization convergence curves of altogether 500 ModCMA configurations evaluated using HPO for three selected RGFs, where a clear configuration ranking is possible (*left*), the configuration ranking is ambiguous (*middle*), and the best-found solution is an outlier that can only be occasionally 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].

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

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	$\lceil x \rfloor$		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.

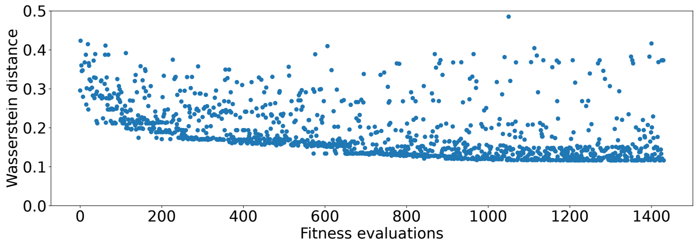


Figure 4.12: Convergence trajectory of GP for F1 in 2-d. The fitness evaluations are shown on the x-axis, while the Wasserstein distance is shown on the y-axis. Figure taken from [75].

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.

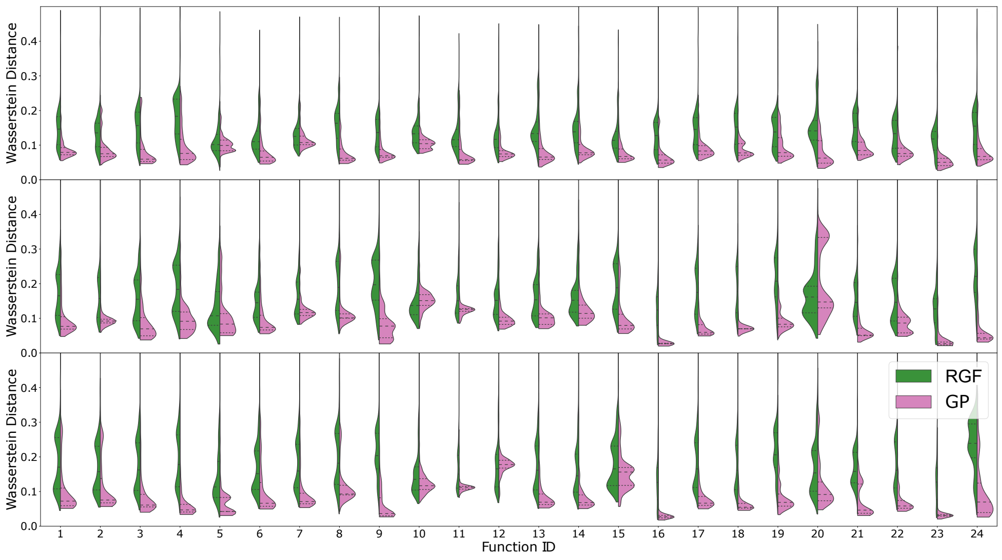


Figure 4.13: Distribution of Wasserstein distances of RGFs (*green*) and GP-generated functions (*pink*) to 24 BBOB functions in 2-d (*top*), 5-d (*middle*), and 10-d (*bottom*) in terms of ELA features. Figure taken from [75].

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-

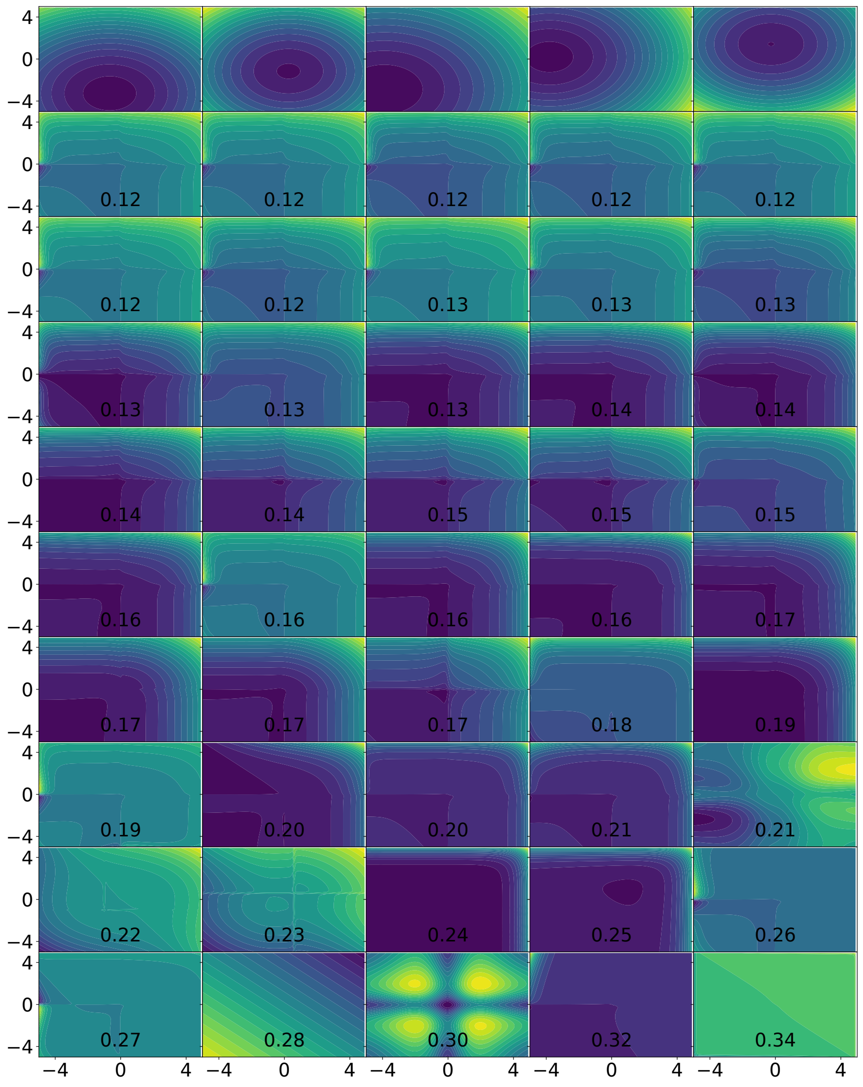


Figure 4.14: Visualization of GP-generated functions for F1 in 2-d, where the 5 BBOB instances are plotted in the first row. This is followed by 45 GP-generated functions, which are selected according to their Wasserstein distances (as indicated by the value) using a linear spacing, starting with the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

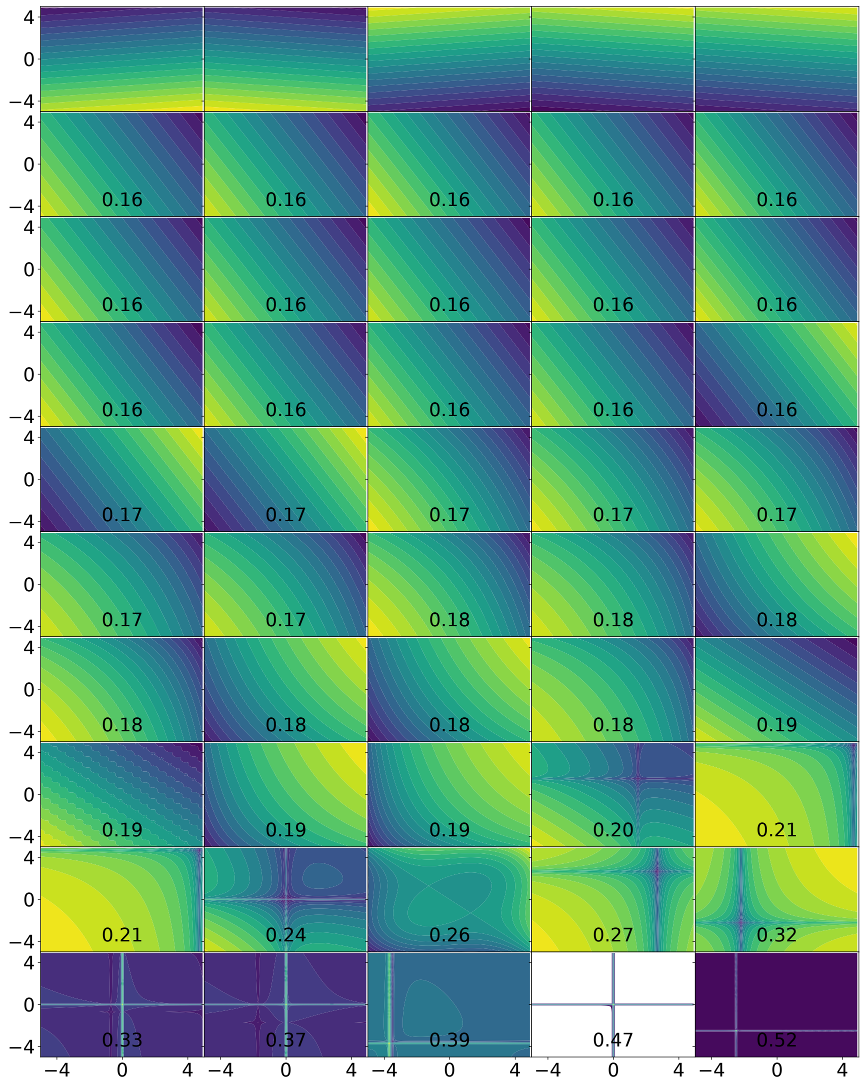


Figure 4.15: Visualization of GP-generated functions for F5 in 2-d, where the 5 BBOB instances are plotted in the first row. This is followed by 45 GP-generated functions, which are selected according to their Wasserstein distances (as indicated by the value) using a linear spacing, starting with the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

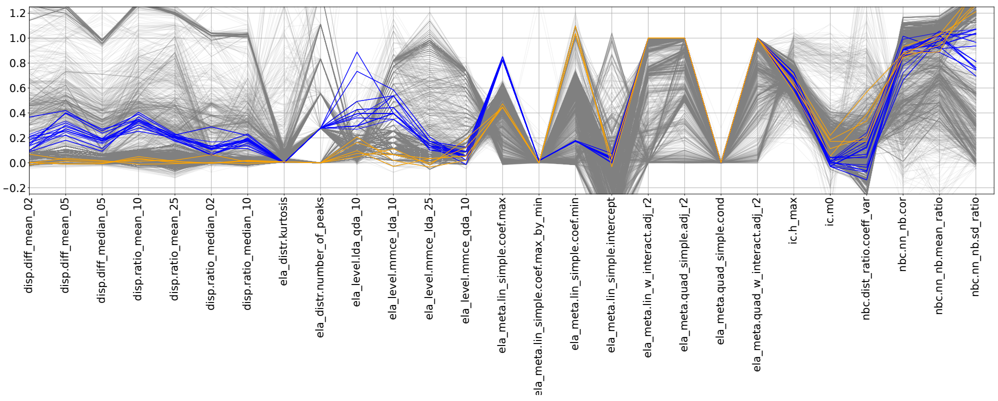


Figure 4.16: ELA features for F5 in 2-d (5 instances; *blue*), the corresponding GP-generated functions $\frac{x_0+x_1}{2}$ (*orange*), and the remaining functions (*gray*). Each line represents a bootstrapped DoE. Figure taken from [75].

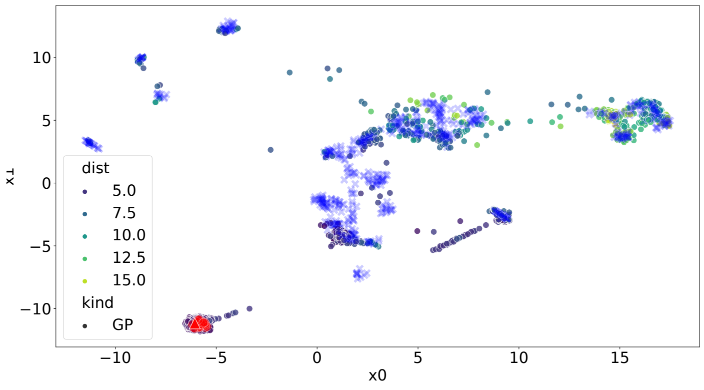


Figure 4.17: Projection of F5 and GP-generated functions to a 2-d visualization space using UMAP. Basically, the projection mapping is defined using only the BBOB instances (*cross; blue*). On one hand, the mean across five instances for the target F5 is shown (*triangle; red*). On the other hand, the 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.

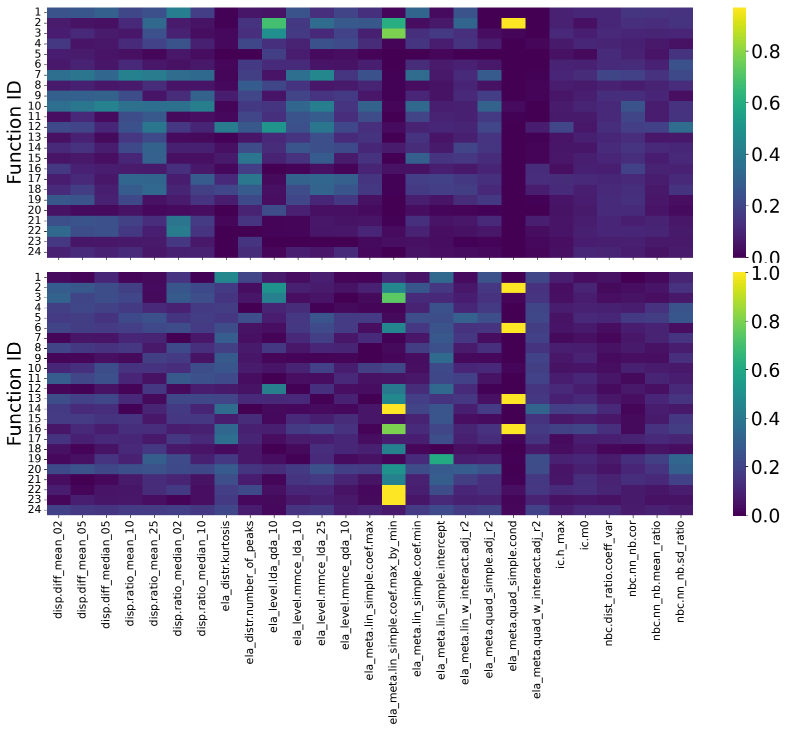
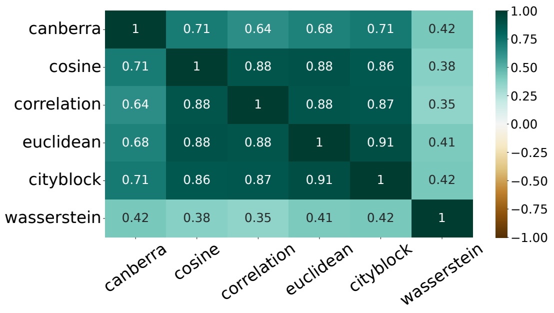


Figure 4.18: Relative standard deviation of normalized ELA features for 24 BBOB function (*top*). Absolute difference in relative standard deviation of normalized ELA features between BBOB functions and GP-generated functions (*bottom*). A lighter color represents a larger deviation, and vice versa.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



	canberra	cosine	correlation	euclidean	cityblock	wasserstein
canberra	1	0.71	0.64	0.68	0.71	0.42
cosine	0.71	1	0.88	0.88	0.86	0.38
correlation	0.64	0.88	1	0.88	0.87	0.35
euclidean	0.68	0.88	0.88	1	0.91	0.41
cityblock	0.71	0.86	0.87	0.91	1	0.42
wasserstein	0.42	0.38	0.35	0.41	0.42	1

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

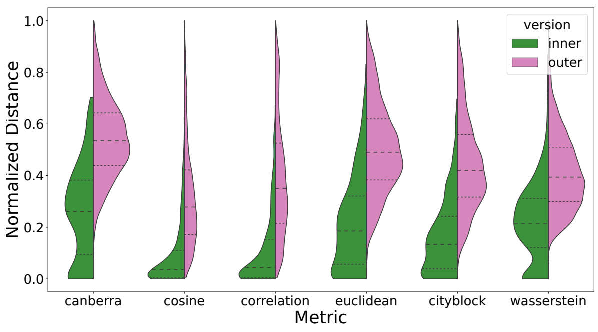


Figure 4.20: Distribution of normalized distances between instances of the same (inner) and different (outer) BBOB functions. Figure taken from [75].

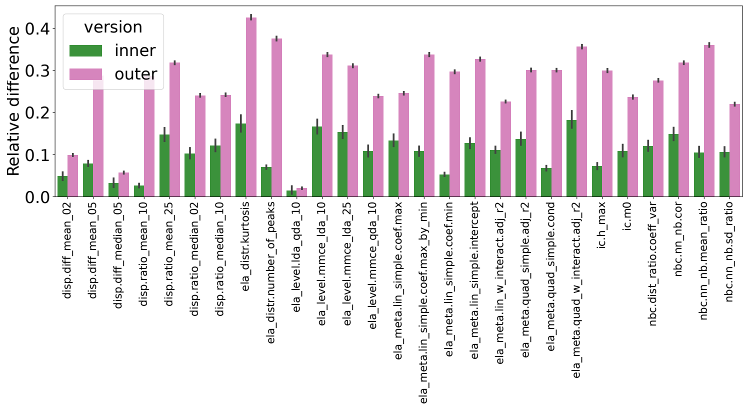


Figure 4.21: Relative differences of normalized ELA features between instances of the same (inner) and different (outer) BBOB functions. Figure taken from [75].

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 identify 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].