Challenges of ELA-Guided Function Evolution Using Genetic Programming

Fu Xing Long[1][a], Diederick Vermetten[2][b], Anna V. Kononova[2][c], Roman Kalkreuth[3][d], Kaifeng Yang[4][e], Thomas Bäck[2][f] and Niki van Stein[2][g]

[1]*BMW Group, Knorrstraße 147, Munich, Germany*

[2]*LIACS, Leiden University, Niels Bohrweg 1, Leiden, The Netherlands*

[3]*Computer Lab of Paris 6, Sorbonne Université, Paris, France*

[4]*University of Applied Sciences Upper Austria, Softwarepark 11, 4232, Hagenberg, Austria*

Keywords: Function Generator, Genetic Programming, Exploratory Landscape Analysis, Instance Spaces.

Abstract: Within the optimization community, the question of how to generate new optimization problems has been gaining traction in recent years. Within topics such as instance space analysis (ISA), the generation of new problems can provide new benchmarks which are not yet explored in existing research. Beyond that, this function generation can also be exploited for solving expensive real-world optimization problems. By generating fast-to-evaluate functions with similar optimization properties to the target problems, we can create a test set for algorithm selection and configuration purposes. However, the generation of functions with specific target properties remains challenging. While features exist to capture low-level landscape properties, they might not always capture the intended high-level features. We show that it is challenging to find satisfying functions through a genetic programming (GP) approach guided by the exploratory landscape analysis (ELA) properties. Our results suggest that careful considerations of the weighting of ELA properties, as well as the distance measure used, might be required to evolve functions that are sufficiently representative to the target landscape.

1 INTRODUCTION

Benchmark problems play a key role in our ability to efficiently evaluate and compare the performance of optimization algorithms. Well-constructed benchmark suites provide researchers the opportunity to gauge the different strengths and weaknesses of a wide variety of optimization algorithms (Hansen et al., 2010). Following this, carefully handcrafted sets of problems, such as the black-box optimization benchmarking (BBOB) suite (Hansen et al., 2009), have become increasingly popular (Bartz-Beielstein et al., 2020). Beyond benchmarking purposes, the BBOB suite has been intensively used in the research field of algorithm selection problem (ASP) (Rice, 1976), with the focus on identifying computational and time-efficient algorithms for a particular problem instance. Recently, this has been associated with the optimization landscape properties of problem instances, where the landscape properties are exploited to predict the performance of optimization algorithms, e.g., using machine learning models (Kerschke and Trautmann, 2019a).

One inherent limitation of these hand-crafted benchmark suites, however, lies in the fact that they can never cover the full instance space. For instance, it has been shown that real-world automotive problem instances are insufficiently represented by the BBOB functions in terms of landscape properties (Long et al., 2022). Consequently, there is a growing trend in understanding the coverage of problem classes by existing benchmark sets, and in the creation of new benchmarks to fill the gaps (Smith-Miles and Muñoz,

[a] https://orcid.org/0000-0003-4550-5777
[b] https://orcid.org/0000-0003-3040-7162
[c] https://orcid.org/0000-0002-4138-7024
[d] https://orcid.org/0000-0003-1449-5131
[e] https://orcid.org/0000-0002-3353-3298
[f] https://orcid.org/0000-0001-6768-1478
[g] https://orcid.org/0000-0002-0013-7969

Long, F., Vermetten, D., Kononova, A., Kalkreuth, R., Yang, K., Bäck, T. and van Stein, N.
Challenges of ELA-Guided Function Evolution Using Genetic Programming.
DOI: 10.5220/0012206200003595
In *Proceedings of the 15th International Joint Conference on Computational Intelligence (IJCCI 2023)*, pages 119-130
ISBN: 978-989-758-674-3; ISSN: 2184-3236

2023; Muñoz and Smith-Miles, 2020). In this research area, or commonly known as instance space analysis (ISA), a feature-based representation of the problem instances are used to identify functions that are lacking and should be newly created. This is often combined with a performance-oriented view of several optimization algorithms, leading to the creation of new functions, where the benefits of one algorithm over the others can clearly be observed.

The most common approach to generating new benchmark problems is through the use of genetic programming (GP) (Muñoz and Smith-Miles, 2020). Since GP has a long history in the domain of symbolic regression (SR), it is a natural choice for the creation of optimization problems. Essentially, GP is guided towards a target feature vector in a poorly-covered part of the instance space. These features can be generated, for example, using the exploratory landscape analysis (ELA) (Mersmann et al., 2011), which aims to capture low-level information about the problem landscape using a limited number of function evaluations.

While this GP-based approach to function generation has shown considerable promise in ISA, it can also be used to create a set of representative functions for algorithm selection and hyperparameter tuning purposes. This is especially useful for real-world optimization problems with expensive function evaluation, e.g., requiring simulator runs. Indeed, previous work has shown that benchmark problems with similar characteristics as real-world problems can be useful to tune optimization algorithms, leading to performance benefits on the original problems (Thomaser et al., 2023). Therefore, the ability to generate a set of problems with similar optimization properties would be of significant practical importance.

In this work, we focus on investigating how a GP guided by ELA features can be utilized to generate problems which are similar to known benchmark functions. This illustrates the challenges which still need to be overcome to efficiently generate sets of feature-based surrogate problems. In particular, our contributions are as follows:

1. We adapt the random function generator (RFG) from (Tian et al., 2020) into a GP approach and investigate the impact this has on the distribution of ELA features of the generated problems.
2. We investigate the impact of the used distance measure between ELA feature vectors. Our results suggest that the Wasserstein distance metric and equal treatment of all features might not be desirable.

2 RELATED WORK

2.1 Black-Box Optimization Benchmarking

The BBOB family of problem suites are some of the most well-known sets of problems for benchmarking optimization heuristic algorithms (Hansen et al., 2010), particularly the original continuous, noiseless, single-objective suite, which is often referred to as *the* BBOB (Hansen et al., 2009). To facilitate the benchmarking purposes, the BBOB suite has been integrated as part of the comparing continuous optimizers (COCO) platform (Hansen et al., 2021) and iterative optimization heuristics profiler tool (IOHProfiler) (Doerr et al., 2018), where the statistics of algorithm performances stored can be easily retrieved. Due to its popularity, the BBOB suite has also become a common testbed for automated algorithm selection and configuration techniques, even though the suite was never designed with this in mind.

Altogether, the aforementioned original BBOB suite consists of 24 functions from five problem classes based on their global properties. While the BBOB functions were originally designed for unconstrained optimization, in practice however, they are usually considered within the search domain of $[-5,5]^d$ with their global optimum located within $[-4,4]^d$. Beyond the fact that they can be scaled to arbitrary dimensionality d, the BBOB functions have the advantage that different variants or problem instances can be easily generated through a transformation of the search domain and objective values. This transformation mechanism is internally integrated in BBOB and controlled by a unique identifier, or also called IID.

2.2 Exploratory Landscape Analysis

In landscape-aware ASP, the landscape properties of problem instances are associated with the performance of optimization algorithms. For this, the most common way is by characterizing the landscape characteristics or high-level properties of a problem instance, such as its global structure, multi-modality and separability (Mersmann et al., 2010). Nonetheless, an accurate characterization of these high-level properties is challenging without expert knowledge. To facilitate the landscape characterization, ELA has been introduced to capture the low-level properties of a problem instance, e.g., y-distribution, level set and meta-model (Mersmann et al., 2011). It has been shown that these ELA features are sufficiently expressive in accurately classifying the BBOB functions according to their corresponding problem classes (Renau et al., 2021) and also informative for algorithm selection purposes (Muñoz et al., 2015; Kerschke et al., 2019).

In the ELA, landscape features are computed primarily using a design of experiment (DoE) of some sample points $\mathcal{X} = \{x_1, ..., x_n\}$ evaluated on an objective function h, i.e., $h\colon \mathbb{R}^d \to \mathbb{R}$, with $x_i \in \mathbb{R}^d$, n represents sample size, and d represents function dimensionality. In this work, we compute the ELA features using the `pflacco` package (Prager and Trautmann, 2023b), which was developed based on the `flacco` package (Kerschke and Trautmann, 2019b). With more than 300 ELA features that can be computed, we consider only the ELA features that can be cheaply computed without additional re-sampling, similar to the work in (Long et al., 2022), and disregard the ELA features that only concern the DoE samples (altogether four of the principal component analysis features).

While we are fully aware that the ELA features are highly sensitive to sample size (Muñoz et al., 2022) and sampling strategy (Škvorc et al., 2021a), these aspects are beyond the scope of this work. Throughout this work, we consider the Sobol' sampling technique (Sobol', 1967) based on the results in (Renau et al., 2020).

2.3 Instance Space Analysis

In general, ISA is a methodology of benchmarking algorithms and assessing their strengths and weaknesses based on clusters of problem instances (Smith-Miles and Muñoz, 2023). The instance space refers to the set of all possible problem instances that can be used to evaluate the performance of such an optimization algorithm. The fundamental idea behind ISA is to model the relationship between the structural properties of a given problem instance and the performance of a set of algorithms. Through this approach, footprints can be constructed for each algorithm, which are essentially regions in the instance space where statistically significant performance improvements can be inferred.

In our proposed approach, a similar concept is utilized, namely to find such problem instances that resemble specific expensive real-world problems, to allow the comparison and benchmarking of various algorithms on a highly specific instance space. This differs from other related works, such as the Melbourne algorithm test instance library with data analytics (MATILDA) software (Smith-Miles and Muñoz, 2023), because while other works try to create a space-covering of instances for a general benchmarking purpose and comparison of algorithms, we aim to generate highly specialised benchmarks sets that are fast-to-evaluate and representative for expensive real-world black box problems. Generating such domain-specific benchmarks would allow us to search and optimize specific algorithm configurations that are better applicable to a specific problem domain. Furthermore, it also allows a better understanding of the expensive black-box problem through the analysis of different optimization landscapes that represent the found instance space.

2.4 Genetic Programming

Principally, GP can be considered as a search heuristic for computer program synthesis that is inspired by neo-Darwinian evolution (Koza, 1989). As proposed by Koza, GP traditionally uses trees as program representation. A typical application of GP is to solve optimization problems that can be formulated as: $\arg\min_{\mathbf{t}} \; f(\mathbf{t}), \mathbf{t} \in \mathcal{T}$, where $\mathbf{t} = (t_1, \cdots, t_n)$ represents a decision vector (also known as individual or solution candidate) in evolutionary algorithms (EAs). Similar to other EAs, GP evolves a population of solution candidates by following the principle of the survival of the fittest and utilizing biologically-inspired operators. The feature distinguishing GP from other EAs is the variable-length representation for $\mathbf{t}$, instead of a fixed-length representation.

Throughout the years, GP has been widely used to solve regression problems by searching through a space of mathematical expressions. In fact, GP-based SR (Tackett, 1995) is popular as an interpretable alternative to black-box regression methods, where GP is used to search for an explicit mathematical expression for a given dataset. By producing a mathematical expression that can be easily understood by humans, SR has proven to be a valuable tool in engineering applications, where it is important to comprehend the relationship between different decision variables.

While GP-based SR was mainly used as a surrogate model to either quantify the relationship between different decision variables or replace expensive optimization problems in previous work (e.g., in engineering), we focus on utilizing canonical GP (Tian et al., 2020) to create functions with specific target ELA features in this work.

2.5 Generating Black-Box Optimization Problems

Apart from the expertly designed benchmarking test suites, Tian et al. introduced a SR approach in generating continuous black-box optimization problems (Tian et al., 2020). In their work, a function generator was proposed to generate problem instances of different complexity in the form of tree representations that serve as training samples for a recommendation model. More specifically, the function generator constructs a tree representation by randomly selecting mathematical operands and operators from a predefined pool, where each operand and operator has a specific probability of being selected. In this way, any arbitrary number of functions can be quickly generated. To improve the functional complexity that can be generated, such as noise, multi-modal landscape and complex linkage between variables, a *difficulty injection* operation was included to modify the tree representation. Furthermore, a *tree-cleaning* operation was considered to simplify the tree representation by eliminating redundant operators. In the remainder of this paper, we refer to this function generator as random function generator (RFG).

In fact, functions generated by the RFG indeed have landscape characteristics different from the BBOB test suite and complement the coverage of BBOB functions in the instance space (Škvorc et al., 2021b). Furthermore, it has been shown that, as far as landscape characteristics are concerned, some functions generated by the RFG belong to the same problem class as several automotive crashworthiness optimization problem instances (Long et al., 2022).

In addition to GP and RFG approaches, a recent paper proposed to make use of affine combinations of BBOB functions and showed that these new functions can help fill empty spots in the instance space (Dietrich and Mersmann, 2022). Essentially, a new function is constructed via a convex combination of two selected BBOB functions, using a weighting factor to control the interpolation. Extensions of this work have generalized the approach to affine combinations of more functions (not limited to only two functions) and shown their potential for the analysis of automated algorithm selection methods (Vermetten et al., 2023b; Vermetten et al., 2023a).

3 METHODOLOGY

In brief, we develop our GP-based function generator (we simply refer this as GP in the remainder of this paper) based on the RFG and canonical GP approach. Precisely, we consider the mathematical operands and operators similar to those used in the RFG with slight modifications, as summarized in Table 1. Following this, the GP search space consists of the terminal space $\mathcal{S}$ (operands) and function space $\mathcal{F}$ (operators), i.e., $\mathcal{T} = \mathcal{S} \cup \mathcal{F}$. Unlike typical GP-based SR method, where each design variable x_i is separately treated (as terminal), we consider a tree-based math expression, that is, a *vector-based input* $\mathbf{t} = (t_1, \cdots, t_d)$, to facilitate a comparison with the RFG (Azzali et al., 2019).

Regarding the GP aspect, we consider the canonical GP and the distributed evolutionary algorithms in Python (DEAP) package (Fortin et al., 2012). The descriptions of our GP function generator are as follows.

Data. A set DoE samples $\mathcal{X}$ and the ELA features of the target functions are used as input for the GP pipeline.

Objective Function. In the GP-system, our optimization target is to minimize the differences between the ELA features of an individual and the target function. Before the ELA feature computation, we normalize the objective values (by min-max scaling) to remove inherent bias as proposed in (Prager and Trautmann, 2023a). Furthermore, we normalize the ELA features (by min-max scaling) before the distance computation, to ensure that all ELA features are within a similar scale range. For this, we consider the minimum and maximum values from a set of BBOB functions (24 BBOB functions, 5 instances each). Based on the same set of BBOB functions, we identify and filter out ELA features that are highly correlated in a similar fashion to the work in (Long et al., 2022), resulting in a total of 27 remaining features.

Infeasible Solutions. An individual is considered to be infeasible, if any of the four 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 computation, e.g., due to equal fitness in all samples, and
4. Invalid distance, caused by missing value in ELA feature.

All infeasible trees are penalized with a large fitness of 10 000.

Initialize Population. In the first generation, we initialize the initial population (of a population size of 50 for computational reasons) using random sampling, with the tree depth is limited between 3 and 12. Similar to the RFG, we assigned each operand and operator a probability of being selected (Table 1).

Table 1: List of notations and their meaning, syntax, protection rules (if any) and probability of being selected for the GP sampling (only during the first generation).

Notation	Meaning	Syntax	Remark/Protection	Probability
		Operands ($\mathcal{S}$)		
`x`	Decision vector	$(x_1,\ldots,x_d)$		0.6250
`a`	A real constant	a	$\mathcal{U}(1,10)$	0.3125
`rand`	A random number	$rand$	$\mathcal{U}(1,1.1)$	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 $\lvert x\rvert \le 10^{-20}$	0.1098
`neg`	Negative	$-x$		0.0219
`rec`	Reciproval	$1/x$	Return 1, if $\lvert x\rvert \le 10^{-20}$	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 $\lvert x\rvert \le 10^{-20}$	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

Whenever an infeasible tree is generated, we will do re-sampling, meaning that this infeasible tree will be replaced by generating a new tree. As such, we ensure that the initial population is free from infeasible trees (due to errors 1 and 2).

Mating Selection and Variation. We consider the tournament selection with tournaments of size 5, subtree crossover with a crossover probability of 0.5, and subtree mutation with a mutation probability of 0.1. Other remaining hyperparameters are set to default settings. While hyperparameter tuning could potentially improve our results further, we decide to leave it for future work.

Result. An optimal individual as solution of the GP runs.

4 EXPERIMENTAL SETUP

In this work, we test our pipeline based on all 24 BBOB functions (one by one) of three different dimensionalities $d = 2$, 5 and 10 (or simply $2d$, $5d$ and $10d$).

We consider a DoE size of $150d$ samples and the search domain $[-5,5]^d$.

To reliably capture the landscape characteristics, we compute ELA features in a bootstrapping manner (using only 80% of the DoE samples and 5 repetitions with different random seeds). As for the optimization objective or individual fitness in the GP system, we consider minimizing the average Wasserstein distance between the ELA features of the first five BBOB instances and the evaluated individual (5 bootstrapped samples for each feature, for each instance). Due to computational limits, we perform only one run for each target function in each dimensionality.

Reproducibility. The codes (used to generate, process and visualize the experiments), raw data and more figures have been made available in a Zenodo repository (Long et al., 2023).

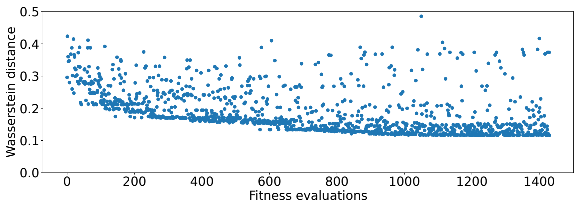

Figure 1: GP convergence for target F1 (sphere), in $2d$, with the fitness evalutions on the x-axis and the Wasserstein distance on the y-axis.

5 RESULTS

5.1 Performance of GP

We start by analysing the functions generated during the GP runs with a $2d$ BBOB function as their target. In Figure 1, we show the convergence trajectory of a single run on F1. From this figure, we clearly see that GP manages to improve over the initial population (first 50 evaluations), as time goes on. Note that, while it has a budget of 50 generations of 50 individuals each, the combination of a crossover rate of 0.5 with a mutation rate of 0.1 means that there is a probability of 0.45 that a selected individual is not modified in any way, and just copied to the next generation without being evaluated. Additionally, Figure 1 does not include the infeasible solutions, which make up 2% of all evaluations in this run.

To give some context to the fitness values shown in Figure 1, we compare functions generated via our GP approach with functions generated by the RFG (see Section 2.5). For this purpose, we generate 1 000 feasible functions (using the RFG) for each dimensionality and measure their Wasserstein distance to each of the 24 target BBOB functions. Then, we compare these distances to those of the functions generated during our GP runs ($\sim$ 1 400 functions) to the corresponding target problem. This is visualized in Figure 2, from which we see that the lower end (i.e., generated functions with low distance to target functions) of the GP distribution is almost always better than that of the RFG, with some exceptions, e.g., F12 (Bent Cigar) in $10d$.

For these GP runs, we can also visualize the resulting function landscapes to identify how much they resemble the target BBOB function. This is shown in Figure 3 for F1 (sphere) in $2d$, where the 5 BBOB instances are plotted in the first row, followed by 45 GP-generated functions. These functions are selected by sorting their fitness values and taking a linear spacing in the rank values between the best and worst functions, to show a range of generated functions of varying quality. From visual inspection, we notice that even the best functions (with the smallest Wasserstein distance) do not quite visually represent a sphere as we might have expected.

In a similar fashion, we can create the same visualization for other functions, e.g., F5 (linear slope), as shown in Figure 4, where we observe a much closer matching between the target and generated functions. It is, however, interesting to note that some generated functions, e.g., row 7 column 5 (which represents function $\frac{x_0+x_1}{2}$), appear visually similar to the target, but nonetheless have a relatively large fitness value. *This raises the question of whether the distance to the target distribution in ELA space (using the Wasserstein distance) really captures the intuitive global properties of the linear slope problem.*

5.2 Investigating the ELA Space

To understand why the distance between this linear slope and the target function is relatively large, we need to look at the individual ELA features. This can be done through a parallel coordinate plot, as shown in Figure 5. From this figure, we can see that mostly the *ela_meta.lin_simple.coef.min* and *ela_meta.lin_simple.coef.max* are different between the target function and generated function $\frac{x_0+x_1}{2}$, indicating that the steepness of the function might be different. However, for the linear slope function, this should not have a large impact as the global properties are mostly preserved. This shows that different ELA features are crucial for different types of target functions. One direction to mitigate such problems might be to analyse the spread of ELA feature values over a large set of instances for each BBOB function (see discussion around Figure 7 below).

To better understand the similarities between functions, we visualize the ELA space filled by the newly GP-generated functions relative to the existing BBOB problems, by utilizing the uniform manifold approximation mapping (UMAP) (McInnes et al., 2018) method. For this, we first create the mapping using only the feature representations from the 24 BBOB problems (all five bootstrapped repetitions on each instance). Next, we apply this fixed map on the GP-generated functions (from one run of the GP). The resulting plot for the target function F5 (linear slope) is shown in Figure 6, where we see that most GP-generated functions are indeed clustered together around the target.

5.3 Distances in Feature Space

Our suspicion for why the distances in ELA space do not directly seem to correlate to our visual un-

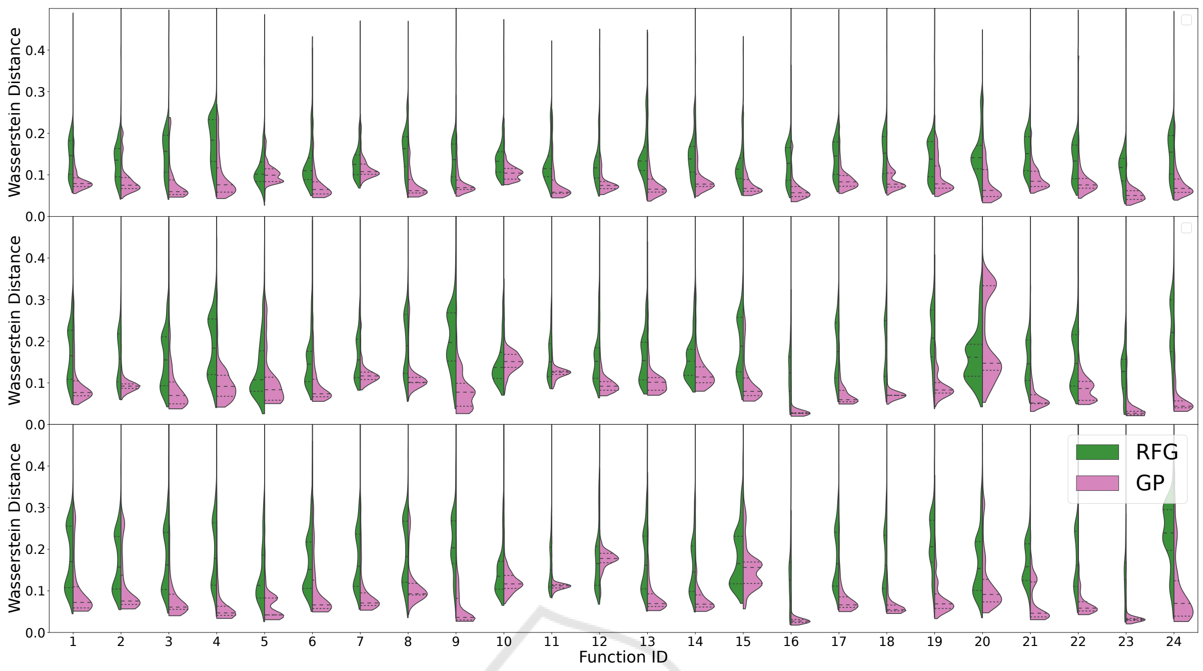

Figure 2: Distribution of fitness values (Wasserstein distance) of the set of functions from the RFG and the GP runs with the specified BBOB target functions (horizontal axis). Rows correspond to dimensionalities: 2*d*, 5*d* and 10*d* (from top to bottom).

derstanding of high-level properties might be that all ELA features are weighted equally. This means that even features, which are more sensitive to small deviations, can have the same impact as features that might be considered crucial to characterize a function e.g., a linear slope. To gain insight into *which features might be more important for a given function*, we analyze the relative standard deviation of each ELA feature within instances of the same BBOB function and visualized the results in Figure 7.

While Figure 7 shows us the variance of each ELA feature within the target functions, it is important to relate this to the deviations observed in the GP-generated functions. Following this, we look at the average standard deviation of each ELA feature across all functions sampled during a single GP run and compute the absolute difference to the values seen on the corresponding BBOB function target. The resulting differences are shown in Figure 8, where we can see that some features, such as *ela_meta.lin_simple.coef.max_by_min*, are significantly more variable in the GP-generated functions than in BBOB. This indicates that these ELA features might be very important for the distance measurement.

In our GP experiments, we consider the average Wasserstein distance to determine the fitness value. This choice was made, since each ELA feature can be considered as a random variable, for which a statistical distance measure would be appropriate. Alternatively, we could make use of a regular distance measure, based on the mean values of these ELA feature distributions. To determine the impact the choice of distance measure might have, we consider the Kendall-tau correlation between a set of six metrics on the pairwise distances between the BBOB instances, consisting of the Canberra, cosine, correlation, Euclidean, cityblock and Wasserstein distance, as visualized in Figure 9.

From Figure 9, we can see that the correlations, while clearly positive, are not perfect. This is especially the case when comparing the statistical distance (Wasserstein) against the vector-based distances. To gauge which distance metric might be preferable, we then compare the distances between instances of the *same* function to the distances between instances of *different* functions, as is done in Figure 10. Based on this comparison, we notice that the Wasserstein distance surprisingly has the lowest distinguishing power (unlike our initial intuition), while the cosine and correlation distances show a clear trend of assigning lower distances to same-function instances.

To further identify potential ways to modify the distance measures, we compare the differences in individual ELA features between same-function and different-function instances. From the results shown in Figure 11, we see that some features, e.g., *ela_meta.quad_simple.cond*, show very limited differ-

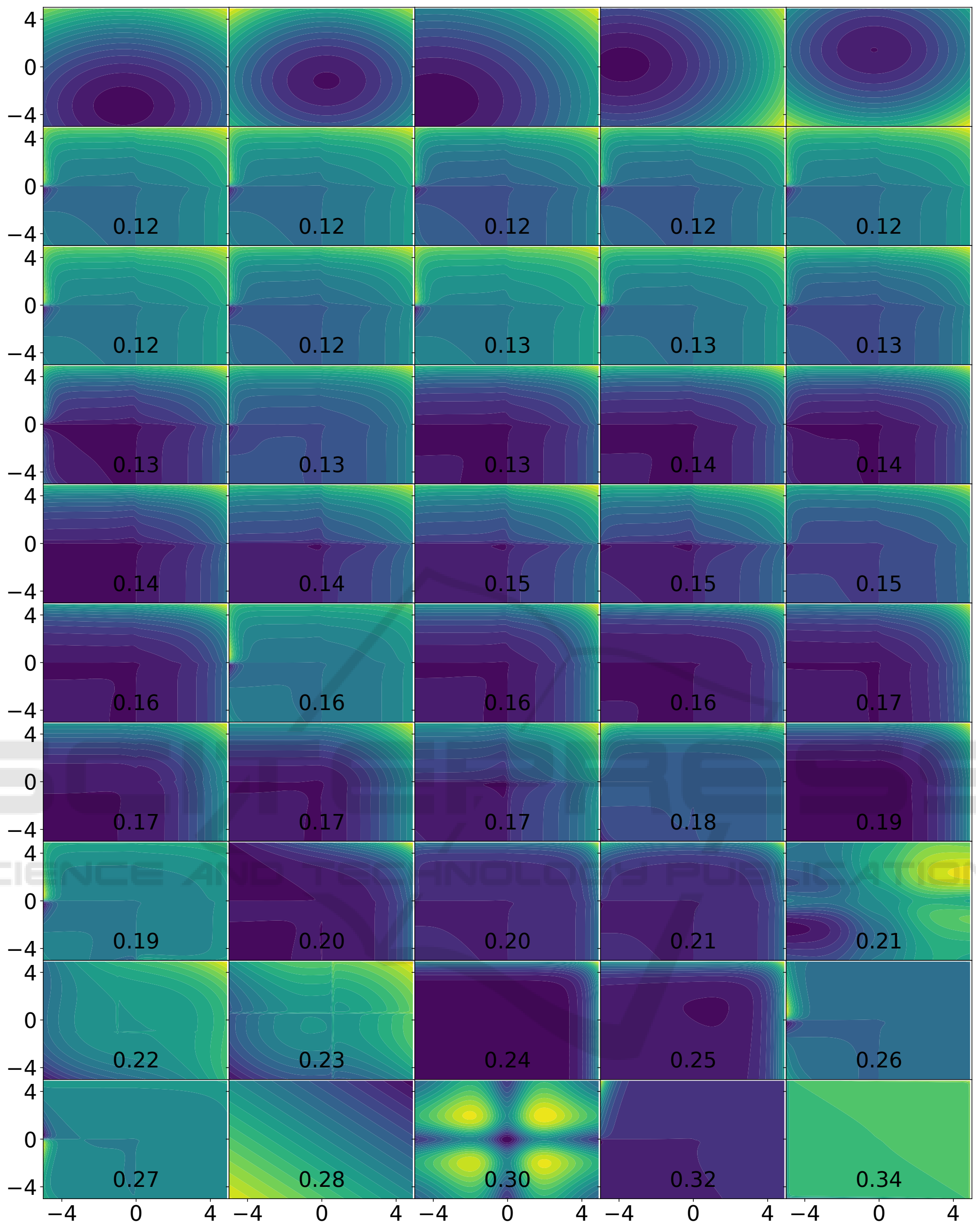

Figure 3: Grid of functions generated by the GP procedure with target function F1 (sphere) in $2d$, of which 5 instances are plotted in the first row. The rows below are GP-generated problems selected by ranking their Wasserstein distance to the target feature vector (indicated by the value in each subfigure) and taking a linear spacing in this ranking from the best (top left) to the worst (bottom right).

ences when comparing instances of the same function relative to different function instances. As such, it is likely that these features contribute very little to any distance measure, and might be potentially removed to improve the stability, as reducing the vector dimensionality can make the distances more reliable.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have shown that GP can be guided by ELA features to find problems with similar high-level characteristics as a set of target problems. How-

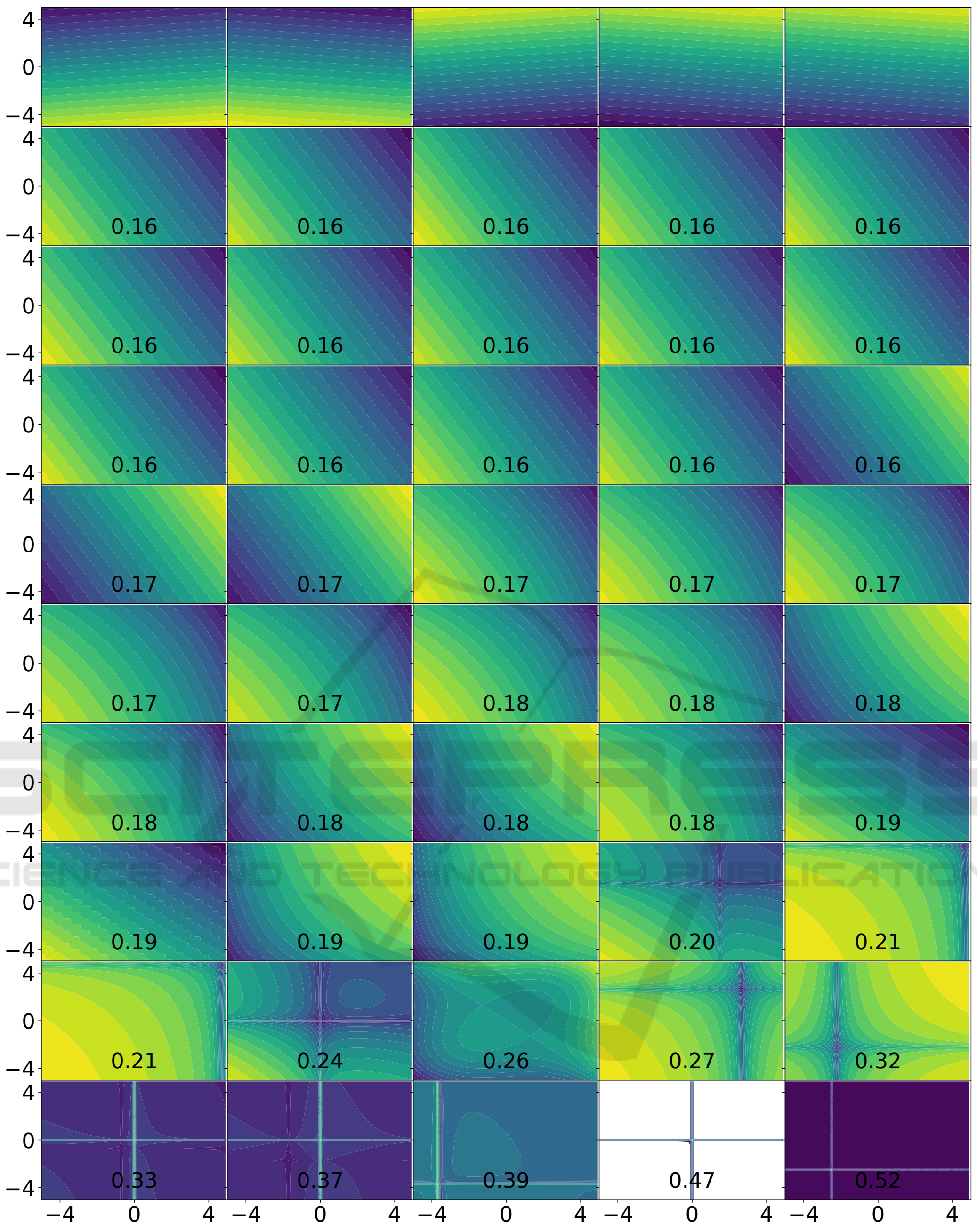

Figure 4: Grid of functions generated by the GP procedure with target function F5 (linear slope) in 2d, of which 5 instances are plotted in the first row. The rows below are GP-generated problems selected by ranking their Wasserstein distance to the target feature vector (indicated by the value in each subfigure) and taking a linear spacing in this ranking from the best (top left) to the worst (bottom right).

ever, through this process, we highlight several potential pitfalls with this approach, illustrated by the fact that we could not accurately recreate a simple sphere problem. Although our results are based on a very limited set of experiments, they reveal that equal weighting of all landscape features on the distance measure leads to difficulties in focusing on the more visual high-level features.

By comparing the differences in ELA features on the BBOB problems both between instances of the same function and instances of different functions, we show that a *feature selection* mechanism should be integrated to make the fitness values more stable. Additionally, a *weighting scheme* based on feature im-

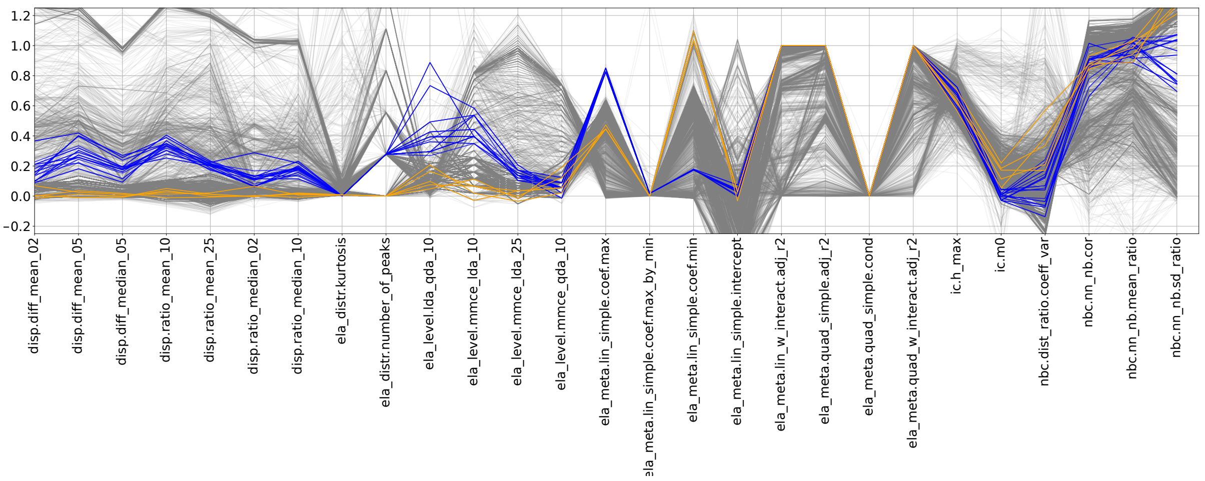

Figure 5: Parallel coordinate plot of the ELA features (one line for each bootstrapped DoE) for the GP-generated functions and target function F5 (linear slope) in $2d$, of which 5 instances are plotted in blue. The orange lines highlight an example of GP-generated function, corresponding to $\frac{x_0+x_1}{2}$, which has a Wasserstein distance of 0.19 to the target.

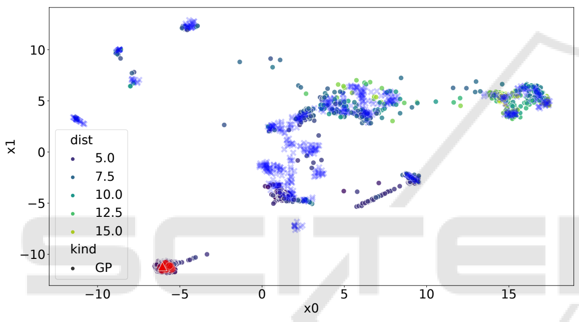

Figure 6: UMAP projection of the GP-generated functions with the target function F5 (linear slope) in $2d$ ELA space. The mapping is based on BBOB instances only, which are highlighted in blue crosses. The target F5 is highlighted in red, with the mean feature vector across the five instances indicated as a red triangle. The dots correspond to the GP-generated problems, where the colour is the cityblock or Manhattan distance to the target vector (here, coloring based on Wasserstein distance is challenging).

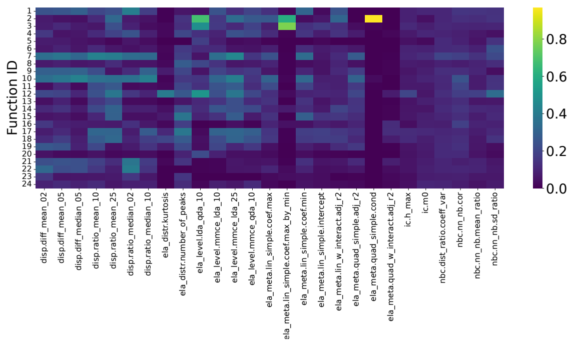

Figure 7: Relative standard deviation (of the normalized values) of each ELA feature for each BBOB function. Lighter color represents a larger deviation.

portances might be used to, in combination with a distance metric, more rigorously guide the GP search

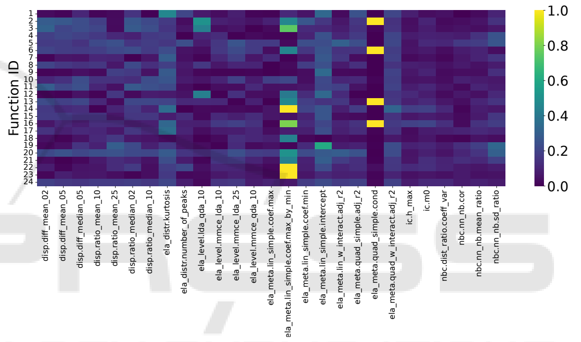

Figure 8: Absolute difference in relative standard deviation (of the normalized values) of each ELA feature for each BBOB function to the functions generated by the GP run with the respective target. Lighter color represents a larger deviation.

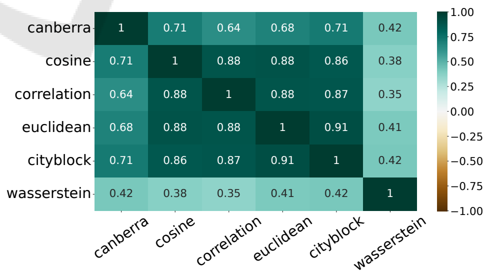

Figure 9: Kendall-tau correlation between six distance measures on the BBOB instances.

towards relevant function characteristics. Further research into ELA and other feature-free approaches are also important to improve the used approach, e.g., models such as DoE2Vec (van Stein et al., 2023) are a starting point in this direction.

Another aspect that should be considered is the specific setting of the GP itself. In this work, we

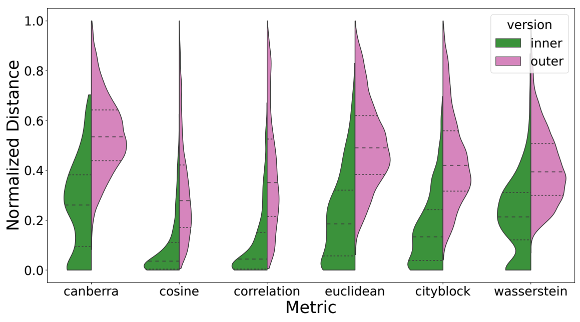

Figure 10: Distribution of normalized distances between BBOB instances of the same problem (inner) and instances of different problems (outer).

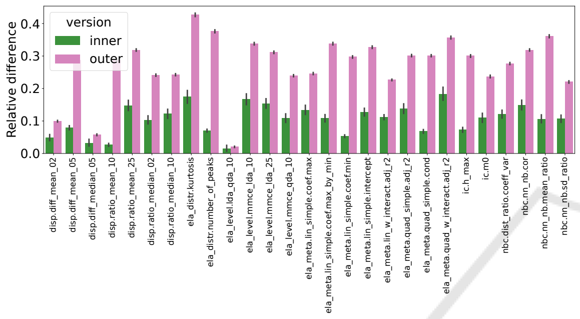

Figure 11: Relative difference (between normalized values) of each of the used ELA features, between BBOB instances of the same problem (inner) and instances of different problems (outer).

made use of default hyperparameter settings with a relatively small population size for computational reasons. This might, however, limit the ability of GP to find diverse solutions, leading to premature convergence (Schweim et al., 2021). Care should also be taken to include a more rigorous tree-cleaning operation, similar to (Tian et al., 2020), to simplify the resulting expressions and prevent infeasible trees from being generated.

In the future, the generation of functions with specific landscape properties has significant potential to help address real-world problems. By generating a diverse set of problems with similar features to a complex target problem, we can create a test set for algorithm selection and configuration pipelines. This has clear benefits over training on a conventional surrogate, since we can avoid overfitting by considering diversified sets of problems.

ACKNOWLEDGEMENTS

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

This work was performed using the ALICE compute resources provided by Leiden University.

REFERENCES

Azzali, I., Vanneschi, L., Silva, S., Bakurov, I., and Giacobini, M. (2019). A vectorial approach to genetic programming. In *European Conference on Genetic Programming*, pages 213–227. Springer.

Bartz-Beielstein, T., Doerr, C., Berg, D. v. d., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., et al. (2020). Benchmarking in optimization: Best practice and open issues. *arXiv preprint arXiv:2007.03488*.

Dietrich, K. and Mersmann, O. (2022). Increasing the diversity of benchmark function sets through affine recombination. In *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I*, pages 590–602. Springer.

Doerr, C., Wang, H., Ye, F., van Rijn, S., and Bäck, T. (2018). IOHprofiler: A Benchmarking and Profiling Tool for Iterative Optimization Heuristics. *arXiv e-prints:1810.05281*.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., and Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.

Hansen, N., Auger, A., Ros, R., Finck, S., and Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*, GECCO '10, page 1689–1696. ACM.

Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., and Brockhoff, D. (2021). Coco: a platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144.

Hansen, N., Finck, S., Ros, R., and Auger, A. (2009). Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA.

Kerschke, P., Hoos, H. H., Neumann, F., and Trautmann, H. (2019). Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45.

Kerschke, P. and Trautmann, H. (2019a). Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary Computation*, 27(1):99–127.

Kerschke, P. and Trautmann, H. (2019b). *Comprehensive Feature-Based Landscape Analysis of Continuous* *and Constrained Optimization Problems Using the R-Package Flacco*, pages 93–123. Studies in Classification, Data Analysis, and Knowledge Organization. Springer International Publishing.

Koza, J. R. (1989). Hierarchical genetic algorithms operating on populations of computer programs. In Sridharan, N. S., editor, *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89*, volume 1, pages 768–774. Morgan Kaufmann.

Long, F. X., van Stein, B., Frenzel, M., Krause, P., Gitterle, M., and Bäck, T. (2022). Learning the characteristics of engineering optimization problems with applications in automotive crash. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1227–1236.

Long, F. X., Vermetten, D., Kononova, A. V., Kalkreuth, R., Yang, K., Bäck, T., and van Stein, N. (2023). Reproducibility files and additional figures. https://doi.org/10.5281/zenodo.7896138.

McInnes, L., Healy, J., Saul, N., and Grossberger, L. (2018). Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861.

Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., and Rudolph, G. (2011). Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 829–836. ACM.

Mersmann, O., Preuss, M., and Trautmann, H. (2010). Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In Schaefer, R., Cotta, C., Kołodziej, J., and Rudolph, G., editors, *Parallel Problem Solving from Nature, PPSN XI*, pages 73–82. Springer Berlin Heidelberg.

Muñoz, M. A., Kirley, M., and Smith-Miles, K. (2022). Analyzing randomness effects on the reliability of exploratory landscape analysis. *Natural Computing*, 21(2):131–154.

Muñoz, M. A. and Smith-Miles, K. (2020). Generating new space-filling test instances for continuous black-box optimization. *Evolutionary computation*, 28(3):379–404.

Muñoz, M. A., Sun, Y., Kirley, M., and Halgamuge, S. K. (2015). Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245.

Prager, R. P. and Trautmann, H. (2023a). Nullifying the inherent bias of non-invariant exploratory landscape analysis features. In *Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings*, pages 411–425. Springer.

Prager, R. P. and Trautmann, H. (2023b). Pflacco: Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems in Python. *Evolutionary Computation*, pages 1–25.

Renau, Q., Doerr, C., Dreo, J., and Doerr, B. (2020). Exploratory landscape analysis is strongly sensitive to the sampling strategy. In Bäck, T., Preuss, M., Deutz, A., Wang, H., Doerr, C., Emmerich, M., and Trautmann, H., editors, *Parallel Problem Solving from Nature – PPSN XVI*, pages 139–153. Springer International Publishing.

Renau, Q., Dréo, J., Doerr, C., and Doerr, B. (2021). Towards explainable exploratory landscape analysis: extreme feature selection for classifying bbob functions. In *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*, pages 17–33. Springer.

Rice, J. R. (1976). The algorithm selection problem. volume 15 of *Advances in Computers*, pages 65–118. Elsevier.

Schweim, D., Wittenberg, D., and Rothlauf, F. (2021). On sampling error in genetic programming. *Natural Computing*, pages 1–14.

Smith-Miles, K. and Muñoz, M. A. (2023). Instance space analysis for algorithm testing: Methodology and software tools. *ACM Computing Surveys*, 55(12):1–31.

Sobol', I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112.

Tackett, W. A. (1995). Mining the genetic program. *IEEE Expert*, 10(3):28–38.

Thomaser, A., Vogt, M.-E., Kononova, A. V., and Bäck, T. (2023). Transfer of multi-objectively tuned cma-es parameters to a vehicle dynamics problem. In *Evolutionary Multi-Criterion Optimization: 12th International Conference, EMO 2023, Leiden, The Netherlands, March 20–24, 2023, Proceedings*, pages 546–560. Springer.

Tian, Y., Peng, S., Zhang, X., Rodemann, T., Tan, K. C., and Jin, Y. (2020). A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks. *IEEE Transactions on Artificial Intelligence*, 1(1):5–18.

van Stein, B., Long, F. X., Frenzel, M., Krause, P., Gitterle, M., and Bäck, T. (2023). Doe2vec: Deep-learning based features for exploratory landscape analysis.

Vermetten, D., Ye, F., Bäck, T., and Doerr, C. (2023a). MA-BBOB: Many-affine combinations of BBOB functions for evaluating AutoML approaches in noiseless numerical black-box optimization contexts. *AutoML 2023*.

Vermetten, D., Ye, F., and Doerr, C. (2023b). Using affine combinations of BBOB problems for performance assessment. *CoRR*, abs/2303.04573.

Škvorc, U., Eftimov, T., and Korošec, P. (2021a). The effect of sampling methods on the invariance to function transformations when using exploratory landscape analysis. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 1139–1146.

Škvorc, U., Eftimov, T., and Korošec, P. (2021b). *A Complementarity Analysis of the COCO Benchmark Problems and Artificially Generated Problems*, page 215–216. ACM.