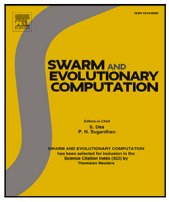
Swarm and Evolutionary Computation 87 (2024) 101534

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo



A cross-benchmark examination of feature-based algorithm selector generalization in single-objective numerical optimization

Gjorgjina Cenikj [a,b,*], Gašper Petelin [a,b], Tome Eftimov [a]

[a] Computer Systems Department, Jožef Stefan Institute, Ljubljana, 1000, Slovenia
[b] Jožef Stefan International Postgraduate School, Ljubljana, 1000, Slovenia

ARTICLE INFO

Keywords:
Algorithm selection
Multi-target regression
Generalization
Benchmarking

ABSTRACT

The task of selecting the best optimization algorithm for a particular problem is known as algorithm selection (AS). This involves training a model using landscape characteristics to predict algorithm performance, but a key challenge remains: making AS models generalize effectively to new, untrained benchmark suites. This study assesses AS models' generalizability in single-objective numerical optimization across diverse benchmark suites. Using Exploratory Landscape Analysis (ELA) and transformer-based (TransOpt) features, the research investigates their individual and combined effectiveness in AS across four benchmarks: BBOB, AFFINE, RANDOM, and ZIGZAG. AS models perform differently based on benchmark suite similarities in algorithm performance distributions and single-best solvers. When suites align, these models underperform against a baseline predicting mean algorithm performance; yet, they outperform the baseline when suites differ in performance distributions and solvers. The AS models trained using the ELA landscape features are better than the models trained using the TransOpt features on the BBOB and AFFINE benchmark suites, while the opposite is true for the RANDOM benchmark suite. Ultimately, the study reveals challenges in accurately capturing algorithm performance through problem landscape features (ELA or TransOpt), impacting AS model applicability.

1. Introduction

The selection of the most appropriate optimization algorithm to solve a given optimization problem instance, known as Algorithm Selection (AS), is driven by the potential to capitalize on the varied performance of different algorithms across sets of different problem instances. AS involves selecting the most appropriate algorithm for a given problem instance from a set of available algorithms. However, determining the optimal algorithm for an unseen instance has been shown to be a challenging task, which has garnered significant attention from researchers in recent years [1,2].

Typically, the best algorithm for a given problem instance is found by considering the properties of the problem instance, which are described using some numerical vector representation, often referred to as problem landscape features [3,4]. Machine Learning (ML) models can use these features to learn the relationship between the problem instances and the algorithm performance and to recommend the best algorithm for a given problem instance.

The majority of works on the topic of AS in the field of single-objective numerical optimization represent problem instances using Exploratory Landscape Analysis (ELA) [5] features, which are a set of mathematical and statistical features calculated on candidate solutions sampled from the decision space of the problem instance. ELA features are useful for characterizing optimization problems, but they also have some limitations, such as being sensitive to the sample size and the sampling method used to sample solutions from the decision space [6,7], and not being invariant to transformations such as scaling and shifting of the problem [7,8].

In most of the available studies [9,10], the AS is typically trained and tested on the same benchmark suite. Due to the lack of a better resource, the Black-box Optimization Benchmarking (BBOB) suite [11] has extensively been used as both a training and testing benchmark, with the generalizability of the performance of the AS rarely being evaluated outside of this suite [9,12–15]. The BBOB benchmark suite contains 24 problem classes, from which problem instances can be generated by applying a transformation (scaling/shifting) on the original function representing the problem class.

An eye-opening study from 2022 [12] has shown that an AS model trained on randomly generated functions [16] has a poor generalization to the BBOB benchmark. This phenomenon may be owed to two factors, which are not mutually exclusive. The first explanation is that the

* Correspondence to: Jamova Cesta 39, 1000 Ljubljana, Slovenia.
E-mail address: gjorgjina.cenikj@ijs.si (G. Cenikj).

https://doi.org/10.1016/j.swevo.2024.101534
Received 1 December 2023; Received in revised form 6 February 2024; Accepted 6 March 2024
Available online 11 April 2024

problem landscape feature values of the training and testing benchmarks are too dissimilar, meaning that the test problem instances are not well represented by the training data. In this regard, recent work has focused on generating new problem instances, either via function generators (random or controlled) [16,17], or through combining the existing BBOB problem instances [18,19]. The second explanation for the low generalization of AS models is that the ELA features fail to capture some characteristics of the problem landscape that are relevant for predicting algorithm performance. This has been observed in [9,15], where some problem instances with similar ELA features did not have similar algorithm performance. To address this issue, several research works in the past few years have proposed the development of alternative representations of continuous problem instances [20–22], however, as of yet, their utility for the AS task and the degree to which they improve the generalizability of an AS model has not been evaluated.

Our contribution: In this work, we first aim to evaluate the generalizability of an AS model across different benchmark suites using the ELA features, as the most commonly used landscape features for single-objective numerical optimization. Secondly, we aim to investigate the degree to which the recently proposed landscape features based on transformers (TransOpt) can complement the ELA features in this task. Additionally, we aim to identify the reasons behind the lack of generalizability of AS models when applied to different benchmarks by observing the distribution of feature values across the benchmarks, and the degree to which similarity in problem representations is consistent with similarity in algorithm performance.

For this purpose, we have trained and evaluated three combinations of feature-based AS models that are different in the features used to describe the landscape characteristics of the problem instance (i.e., ELA, TransOpt, ELA+TransOpt). In all cases, a Random Forest (RF) model has been trained in a multi-target regression scenario predicting the performance of four single-objective optimization algorithms: Differential Evolution (DE) [23], Genetic Algorithm (GA) [24], Particle Swarm Optimization (PSO) [25], and Evolutionary Strategy (ES) [26]. To test the generalizability of the AS models, four benchmark suites have been utilized: BBOB [11], AFFINE [18], RANDOM [16], and ZIGZAG [17]. Our analysis is focused on the performance of the AS trained on one of these benchmarks, and evaluated on the other ones.

Outline: The rest of the paper is organized as follows. In Section 2, we present works that tackle a similar research topic, in Section 3 we present the methodology and experimental setup. Section 4 presents the results of the analysis. Finally, we conclude the paper and provide directions for future work in Section 5.

Reproducibility: The code and the data for the experiments are available at https://github.com/gjorgjinac/opt_generalization.

2. Related work

From the perspective of ML, the AS problem can be modeled using several approaches. A common approach is to formulate it as a multi-class classification problem, where the goal is to select the best algorithm for each problem instance from a given set of algorithms [10,12]. Alternatively, the problem can be addressed using pairwise classification, where a classification model is built for each pair of algorithms to determine which one is better [10,27]. Other possible scenarios are regression, where the ML model estimates the performance of each algorithm [9,10], or algorithm ranking, where the model orders the algorithms based on a specific metric [15]. A benefit of using regression models, also known as automated algorithm performance prediction, over classification models for AS, is the ability to capture the magnitude of performance differences between algorithms, as they predict numerical performance values. In addition, classification-based methods are only interested in predicting the best algorithm, disregarding the performance of the other algorithms. A comparison between the above-mentioned AS approaches has been already conducted on the BBOB benchmark suite [10]. The results have shown that there are no big performance differences in favor of a particular approach, however, this can be also a case due to the design of the BBOB benchmark suite.

Based on the evaluation approaches used in AS studies, the following categorization of evaluation approaches can be made based on how the problem instances are distributed among the training and testing sets:

- **Instance-split evaluation**, which involves training on some instances from all of the problem classes, and testing on other instances from each problem class. This evaluation approach assumes that the test set contains similar instances to the training set since instances of the same problem class are used both for training and testing. This type of evaluation is possible only on the BBOB benchmark suite, where the concepts of a problem instance and problem class are defined. As such, this is the easiest evaluation strategy since the problem instances used in the training and testing phases are relatively similar in terms of features and performance. This evaluation is commonly done in a leave-one-instance-out fashion [10], where a single instance from a problem class is included in the test set, however, multiple instances can also been included.
- **Problem-split evaluation**, which involves removing entire problem classes (all of their problem instances) from the training set and using them for testing the ML model. In this case, the ML model may encounter problem instances in the testing set, which are substantially different than the ones used for training, making it a more challenging evaluation scenario. The most common type of the problem-split evaluation in the community is the leave-one-problem-out evaluation, where instances from a single problem are used for testing. Such an evaluation has been explored for performance prediction models on the BBOB and the benchmark from the Congress on Evolutionary Computation (CEC) special session [28] in [9,29]. In addition, in [9] it was pointed out that when evaluation is done in a problem-split scenario, large errors can be obtained when the AS model is applied to a new problem instance which is not similar to any of the problem instances in the training data. This happens when evaluation is done on the BBOB or CEC benchmarks, since they are limited in the number and diversity of problem instances, introducing the need of including new benchmark suites in the experimental setup.
- **Evaluating the generalization across benchmarks** - Rather than evaluating the performance within one benchmark, this approach involves the training of an AS method on one benchmark, and evaluating it on a completely different benchmark [12,14,15].

In [15], it was observed that an AS trained on the BBOB benchmark has poor generalization to a set of interpolated numerical problems [30]. In this case, the AS is modeled as a ranking problem, where the algorithm portfolio consists of 110 configurations of the Differential Evolution (DE) algorithm. An analysis of the correlation between distances in the feature space and distances in the performance space showed that there is no correlation between the two, which goes against the assumption that ELA features can be used for AS. We build upon this work by including multiple benchmarks, different features for training the AS model, and including a more extensive analysis.

In a recent study [12], all of the previously mentioned types of evaluation have been applied for an AS model which is trained for multi-class classification, where the ELA features are used to represent problem instances. When the model is evaluated using the instance-split evaluation strategy on the BBOB benchmark, a precision of 0.69 and a recall of 0.60 are observed. However, using the problem-split evaluation, the precision score drops to 0.24, while the recall score is 0.32. The authors also investigate the AS generalization across two benchmarks by training the AS on a set of artificial functions and testing on the BBOB benchmark. In this case, a precision of 0.13 and a

recall of 0.14 are achieved, indicating no generalization of the model performance across benchmarks. While the ultimate goal of this paper is aligned with ours, the experimental setup is different in that in [12], instead of a multi-class classification model we are training a multi-target regression model to predict a score of algorithm performance that is relative to the best and worst performing algorithms in the algorithm portfolio. The advantage of our approach is that it takes into account the fact that several algorithms may obtain similar results and can quantify the degree to which one algorithm outperforms another. In addition, our study considers a wider set of problem benchmarks involving four benchmark suites (including one that was recently proposed [18]). Finally, our study also aims to evaluate the performance of the recently proposed transformer-based features for the AS task, in contrast to [12], which is focused only on the ELA features.

3. Methodology

In this section, we first introduce the problem portfolios, i.e., the set of benchmarks for training and evaluating the AS. We present the algorithms included in the algorithm portfolio, as well as the performance metric used to compare them. Subsequently, we explain the features used to represent the problem instances. Finally, we present the training of the ML model used for AS, as well as the metrics used to evaluate it.

3.1. Problem portfolios

Next, the benchmark suites utilized in this study are explained in more detail. These benchmarks were chosen because they had accessible implementations and they allow the creation of problem instances of different dimensions. We need to note that for all benchmarks, the dimension of the problem has been set at 3 and 10 ($d = 3$ and $d = 10$), where d is the problem dimension.

BBOB Benchmark - The Black-box Optimization Benchmarking (BBOB) [11,31] suite consists of 24 single-objective optimization problem classes. Multiple problem instances can be generated from the original problem class by applying a transformation (for instance, scaling or shifting). We use the first 100 problem instances from each of the 24 problem classes from the BBOB benchmark, ending up with 2400 problem instances.

Affine Benchmark - The first five problem instances from the 24 BBOB problem classes have been used to generate affine recombinations as proposed in [18]. The combinations have been done by combining problem instances from different problem classes that have the same instance ID. For example, the first instance of the first problem class is combined with the first instances of the other 23 problem classes. We need to highlight here that instances with different IDs from different problem classes are not combined together. We do this in order to limit the number of generated instances. The recombination is performed with α values of 0.25, 0.50, and 0.75 for all pairs of problem instances. In this way, we obtain 8280 problem instances. The affine transformations have been generated using the equation suggested in [32], also presented below:

$$\begin{aligned} &F(P_{i,m}, P_{j,n}, \alpha)(x) = \\ &\quad \exp(\alpha\, log(P_{i,m}(x) - P_{i,m}(O_{i,m})) + \\ &\quad (1-\alpha)\, log(P_{j,n}(x - O_{i,m} + O_{j,n}) - P_{j,n}(O_{j,n}))). \end{aligned} \tag{1}$$

In this context, $P_{a,b}$ denotes the bth instance of the problem within the ath BBOB problem class. Meanwhile, $O_{a,b}$ signifies the optimum location for the function $P_{a,b}$. The parameter α illustrates the blending extent of the two functions. Every objective function formulated through this equation possesses an optimal solution with an objective value of zero ($P_{a,b}(O_{a,b}) = 0.0$).

Random Functions - We generate 10,000 random problem instances using the random function generator proposed in [16]. The generator constructs a tree representation by randomly combining mathematical operands and operators from a predefined pool, where each operand and operator has a specific probability of being selected. We use the Python implementation of the random function generator, which was originally implemented in Matlab and was re-implemented in Python in [33]. Please note that there are substantial similarities among some of the objective functions in this dataset. This arises from the way function generation is carried out and is an inherent limitation of the proposed generator.

ZIGZAG Benchmark - We generate 5000 random problem instances by randomly initializing the parameters of the four zigzag functions proposed in [17]. The k parameter, controlling the period of the objective function, is randomly sampled as an integer in the range [0,30]. The m parameter, controlling its amplitude, is sampled in the range [0,1]. It has been shown that depending on the m, k, and λ parameters, the generator can create diverse enough objective functions where some algorithms struggle to solve them while others do not.

The rationale for incorporating these benchmark suites is rooted in the diverse needs of the optimization domain, which are designed for different purposes. On the one hand, benchmarks containing carefully designed problems for which we understand their high-level properties and difficulty are better suited for understanding the strengths and weaknesses of different algorithms by observing their behavior on problems with different properties. The BBOB is designed with this goal in mind, while the AFFINE, RANDOM, and ZIGZAG are not. However, the training of an AS model (that involves ML) that will generalize to real-world problems requires a large and diverse set of problems to be used for its training. This introduces a need for a different type of benchmark, which will provide a large coverage of the problem space, with thousands of different problems. Manually designing such a benchmark takes a lot of effort. To the best of our knowledge, there is no benchmark containing real-world problems that are large enough to satisfy the requirements of training an AS model, which is why people are resorting to random problem generators.

3.2. Algorithm portfolio

The algorithm portfolio for performing AS consists of four optimization algorithms Differential Evolution (DE) [23], Genetic Algorithm (GA) [24], Particle Swarm Optimization (PSO) [25], and Evolutionary Strategy (ES) [26]. The *pymoo* [34] python library is used to run them on the problem instances from all of the benchmarks. The algorithms are executed in a fixed-budget scenario with a population size of $10d$. The performance of the algorithm is recorded at budgets of 10, 30, and 50 iterations. For a $10d$ problem, a budget of 50 iterations is equivalent to a total of 5000 function evaluations (50 iterations with a population size of 100). All algorithms use Latin Hypercube Sampling [35] to construct the initial population. The rest of the configuration properties of the algorithms are set to the default values provided in the *pymoo* library, version 0.6.0. We perform ten executions of each algorithm on all of the problem instances. The selected algorithm portfolio has been done only for illustration purposes, however, in the future other algorithms can also be involved. To show this, we have included additional experiments with a different portfolio of algorithms taken from the Nevergrad python library (library established by META). In particular, we take into account the TwoPointsDE, RealSpacePSO, CMA, MetaCMA, and DiagonalCMA algorithms. We include these experiments in the supplementary materials due to the lack of space in the main manuscript.

3.3. Algorithm performance metric

We use a fixed-budget scenario for evaluating the algorithms, meaning that after some budget of algorithm iterations, we record the best-found objective function value. Next, we calculate a relative performance score. The reason for using a relative and not an absolute score is that two of the involved benchmark suites (RANDOM and

Table 1
Example of the calculation of the algorithm performance metric for one problem instance.

Run	DE	ES	GA	PSO
Objective value of best solution found (per run)				
1	−15.10	−14.18	−14.95	−15.03
2	−15.09	−14.13	−14.37	−14.16
3	−15.12	−13.94	−14.89	−15.07
Normalized algorithm score (per run)				
1	0.00	1.00	0.16	0.08
2	0.00	1.00	0.75	0.97
3	0.00	1.00	0.19	0.04
Final algorithm score (median across all runs)				
	0.00	1.00	0.19	0.08

ZIGZAG) do not have a defined optimum for the problem instances. To explain the relative performance score that is further involved as a target in the ML task, let us assume that y_{arp} is the best solution (the lowest objective function value) found by algorithm a in run r for problem instance p. The value y_{arp} is scaled by the range of the solutions found by the other algorithms for the same problem instance and the same initial population (same run). Denoting by b_{rp} and w_{rp} the best and worst solution found for problem instance p in run r by any of the set of algorithms, respectively, we obtain the scaled best objective function value achieved by algorithm a in run r on problem instance p:

$$s_{arp} = \frac{y_{arp} - b_{rp}}{w_{rp} - b_{rp}}. \tag{2}$$

This score captures how much the best solution found by algorithm a in run r was better than the other algorithms executed with the same initial population. We calculate the final value s_{ap}, capturing the performance of algorithm a on problem instance p across all runs of the algorithm, by simply taking the median of all s_{arp}.

To clarify and illustrate how the performance score is calculated, Table 1 demonstrates an example of how the algorithm performance scores are calculated for a single problem instance and a single budget. We should note that within the example, each algorithm is executed three times, so we have three runs, however, in reality, we are executing the algorithms ten times on each problem instance. The table is divided in three parts, each one showing a different step of the process of calculating the algorithm performance metric. The first part of the table shows the objective function values of the best solutions found by each algorithm after a fixed budget of function iterations. In the second step, these values are then normalized within each run, by performing a min–max normalization column wise. This means that for each run, the best algorithm obtains a score of 0, while the worst algorithm obtains a score of 1. In the example, the best algorithm is DE, and the worst is ES. The final algorithm score is calculated by taking the median of the normalized algorithms scores for all of the runs. Please note that in some cases where the algorithm performance is not very stable between different runs, it is possible to get final algorithm scores where none of the algorithms have a score of 0 or 1.

We opt to use such a performance metric for several reasons: (1) As opposed to using multi-class classification and predicting a single best algorithm, this approach captures the performance of all algorithms, circumventing the issue that a model is penalized for predicting one out of two algorithms with very similar performance. (2) In contrast with the ranking approach, where each algorithm is assigned an integer score based on its performance, our approach captures a fine-grained difference in performance, allowing algorithms with similar best-obtained values to have similar scores.

3.4. Problem representations

In this subsection, we describe the calculation of the ELA features and the transformer features. We also consider the merged representations using both ELA and transformer features.

3.4.1. ELA features

We compute ELA features from the following categories using the *flacco* R library [36]: *basic*, *disp*, *ela_distr*, *ela_level*, *ela_meta*, *ic*, *nbc*, and *pca*. In this way, we calculate a total of 93 features for each problem instance. It is important to note that some of these features contain missing values, which we remove before training the AS model. We calculate the ELA features on a sample of the problem instances obtained with Latin Hypercube Sampling in the range of $[-5,5]^d$, exploring sample sizes of $50d$ and $100d$, where d is the problem dimension.

3.4.2. Transformer features

TransOpt features [21] are low-level problem landscape features learned in a supervised manner by training a transformer architecture [37] for BBOB problem classification. Given samples of a BBOB problem instance, the transformer architecture is trained to predict to which of the 24 BBOB problem classes the instance belongs to. 999 instances are used for each of the 24 problem classes. For each problem instance, we generate samples of candidate solutions using Latin Hypercube sampling in the range of $[-5,5]^d$. Within the samples of each problem instance, the objective function values are scaled to be between [0,1]. We label each sample with the corresponding problem class and use it as the target input to the transformer encoder. A classification head is added as the last transformer component, and the entire architecture is trained to predict one of the 24 problem classes from the BBOB benchmarks. The representations generated in the second-to-last layer, before the classification head, are used as representations for the problem instance. These representations are 120-dimensional vectors. We calculate the transformer features using the same samples as the ELA features to ensure a fair comparison.

3.5. Model

We use a Random Forest (RF) model to perform multi-target regression, where the model predicts the performance score of each algorithm. We chose the RF model because it has good performance on tabular data [38] and it provides feature importance values that help with interpretability. The RF model is executed using the default configuration parameters in the *scikit-learn* [39] Python library version 1.2.2. We do not perform parameter tuning in order to evaluate the impact of the proposed features only, on a fixed model configuration. A separate RF model is trained for each algorithm execution budget, each one predicting the scores obtained by the algorithms after this budget. To ensure the robustness of the results, the training of the RF model is repeated 10 times. In each execution, an entire benchmark is used to train the model, and the evaluation is carried out on the remaining three benchmarks.

3.6. Algorithm selector evaluation metrics

Next, the evaluation metrics used to validate the AS will be explained in more detail. We have used two metrics: pairwise ranking score and loss.

- **Pairwise ranking score** - The pairwise ranking score takes into account on how many problem instances in the test set, the pairs of algorithms are correctly ordered according to their score.

$$PRS = \frac{2}{|\mathcal{A}|\,|\mathcal{A}-1|\,|\mathcal{P}|} \sum_{a_i \in \mathcal{A}} \sum_{a_j \in \mathcal{A}, j>i} \sum_{p \in \mathcal{P}} c(a_j, a_i, p) \tag{3}$$

where P is the set of all problems, A is the set of all algorithms, and

$$c(a_j, a_i, p) = \begin{cases} 1 & \text{if } s_p(a_i, a_j, p) = s_g(a_i, a_j, p) \\ 0 & \text{if } s_p(a_i, a_j, p) \neq s_g(a_i, a_j, p) \end{cases} \tag{4}$$

where the function $s_p(a_i, a_j, p)$ yields −1, 0, or 1 based on whether the predicted rank of algorithm a_i is lower, equal, or higher than

Table 2
An example of calculation of the pairwise ranking score and the loss for two problem instances.

Instance	True scores			Predicted scores			Pairwise ranking score	Loss
	DE	GA	ES	DE	GA	ES		
0	0.00	1.00	0.30	0.00	0.20	0.30	0.66	1.00
1	0.05	0.20	1.00	0.40	0.00	0.07	0.33	0.85

the rank of algorithm a_j, respectively, on problem p. Similarly, the function $s_g(a_i, a_j, p)$ also provides the values −1, 0, or 1, but in this case, it considers the order of algorithms in relation to the ground truth.

The equation calculates the pairwise ranking score by comparing the predicted rank and the actual rank of every pair of algorithms on every problem. If the predicted and actual rank are the same, it adds one to the score. Then it divides the score by the total number of possible comparisons. This gives us a fraction between 0 and 1 that tells us how often the predicted order of algorithms is correct. A higher score means a better prediction.

- **Loss** - To capture the true performance of the algorithm predicted to be the best by the algorithm selector, we calculate the loss of the AS:

$$loss = \frac{1}{|\mathcal{P}|} \sum_{p \in P} 1 - (s_{sp} - s_{bp}) \tag{5}$$

where s_{sp} is the true score of the selected algorithm which was predicted to have the best performance for problem instance p, while s_{bp} is the score of the true best-performing algorithm on problem instance p. This loss would get a value of one if the best algorithm is correctly predicted to be the best for all problem instances, i.e. if the AS model makes no errors. In other cases, the loss will have a value between 0 and 1, depending on how much the performance of the selected algorithm differs from the best-performing one. The intuition behind this metric is that we want to capture the loss in performance when we select an algorithm other than the best-performing one. Additionally, the loss metric is only interested in the algorithm that is predicted to be the best, since we believe this is relevant for a practical application.

Both metrics capture different, complementary aspects of the AS model performance, and we use both in order to have a more extensive evaluation.

Table 2 demonstrates how the pairwise ranking score and the loss are calculated for two problem instances and three algorithms. For the first instance, the true best algorithm is DE with a score of 0, and the model predicts it to be the best. The loss score is 1, because the metric is simply "flipped" so we are maximizing both scores. However, the order of the GA and ES algorithms is incorrect for this instance. Since there are in total three unique pairwise combinations of algorithms ((DE,GA), (DE,ES) and (GA,ES)), and two of them are ordered properly, the pairwise ranking score for the first instance is 2/3, i.e. 0.66. For the second instance, the GA algorithm is predicted to be the best, however, the true score of the GA algorithm is 0.2, and the true best-performing algorithm is DE with a score of 0.05. If one were to use the GA algorithm instead of the DE algorithm in practice, the choice of GA over DE, would get a result that is 0.15 (0.2–0.05) worse than if they were to use the true best algorithm. The loss score in this case is 0.85 (1–0.15). The pairwise ranking score is 0.33, because only the algorithms GA and ES are correctly ordered.

4. Results

In this section, we first analyze the performance of the algorithms across all of the benchmarks. We then evaluate the generalization of the AS models, trained and evaluated on different benchmarks. Subsequently, we aim to identify the reasons behind the lack of generalization between some benchmarks. Specifically, this involves comparing the distributions of the feature values of the benchmarks, analyzing the benchmarks' coverage of the problem landscape, and analyzing the alignment of the problem landscape features and algorithm performance. Our analysis is divided in the following topics:

(i) Section 4.1 gives a short overview of the performance of the four algorithms on the different benchmarks.

(ii) Section 4.2 presents the results of the AS performed with the different types of features, focusing on identifying whether the feature-based models (ELA and transformer) outperform a simple dummy model, and comparing the results obtained with the ELA and transformer features.

(iii) Section 4.3 examines the distributions of feature values, to identify if different ranges of feature values are responsible for the lack of generalization which is sometimes observed when training the AS model is done on one benchmark, but evaluated on another one.

(iv) To gain further insights into how similar one benchmark is to another, Section 4.4 analyzes the benchmark's coverage of the problem landscape. More specifically, we perform a clustering of the problem instances from all benchmarks and observe whether problem instances from different benchmarks lie in the same clusters.

(v) Finally, Section 4.5 investigates the ability of both feature types to capture algorithm performance. In particular, this is done by analyzing whether problem instances with high similarity according to landscape features have similar algorithm performance.

4.1. Analysis of the raw algorithm performance data

Fig. 1 depicts the mean scores of the algorithms obtained with a different execution budget. Fig. 1(a) refers to the 3d problem instances, while Fig. 1(b) refers to the 10d problem instances. Each subplot contains the results for a different benchmark, each row refers to a different algorithm, while each column refers to a different budget. From the figures, we can see that the performance of the algorithms on the BBOB and AFFINE benchmarks is very similar. The reason for this outcome is that AFFINE problem instances are a recombination of the BBOB problem instances.

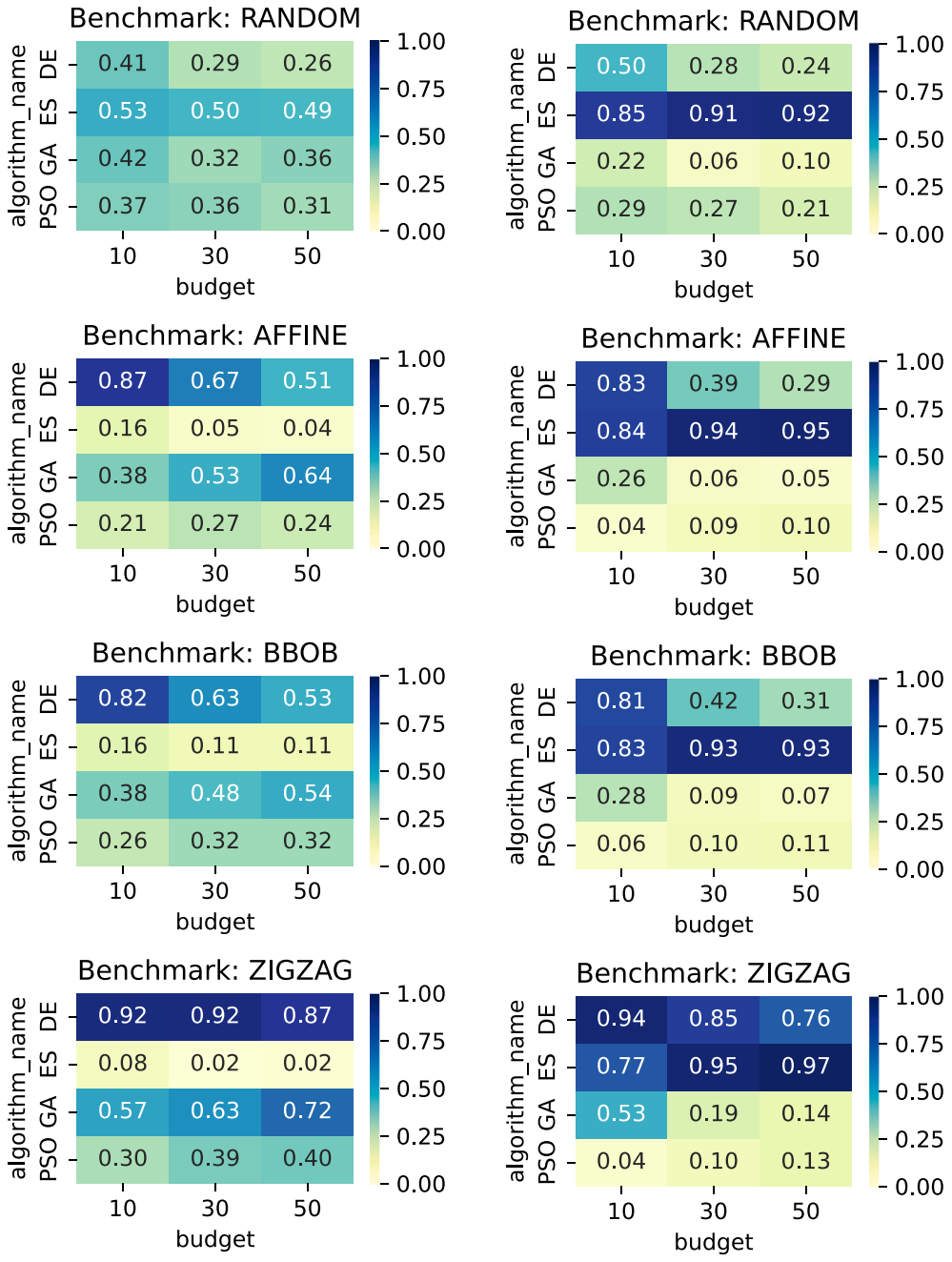
For 3d problems in the case of BBOB, AFFINE and ZIGZAG benchmark suites, the ES algorithm provides the best results across all budgets. On the other hand, on the RANDOM benchmark, the ES algorithm provides the worst results, however, a much higher complimentarity in algorithm performance is observed on this benchmark, with algorithms having similar performance scores across all budgets.

For 10d problems, the GA and PSO algorithms provide the best results on all benchmarks, while the ES algorithm is the worst-performing.

In summary, we find that the BBOB and AFFINE benchmarks rank algorithms in a similar way for both problem dimensions, having the same single-best solver as the ZIGZAG benchmark for 3d problems. For 3d problems, the algorithm performance on the RANDOM benchmark differs from the other benchmarks. This becomes relevant in the next section, as it affects the performance of the baseline model against which the feature-based AS models are compared.

4.2. Generalization of an algorithm selection model across benchmarks

Fig. 2 depicts the results of the model generalizability assessment for the 3d problems. A corresponding figure for the 10d problems can be found in the appendix. The figure is separated into six subplots, with rows of subplots referring to a different budget used for algorithm



(a) $3d$ benchmarks (b) $10d$ benchmarks

Fig. 1. Mean score of each optimization algorithm on $3d$ and $10d$ benchmarks, for budgets of 10, 30 and 50 iterations.

execution, and columns referring to the two evaluation metrics. It is important to note that we aim to maximize both metrics, so a higher score (darker value) indicates better performance. Within each subplot, the row labels indicate the training and testing benchmark, with the first one being the training benchmark. The columns refer to different feature types (ELA, transformer or the merged ELA and transformer features) calculated with different sample sizes ($50d$ or $100d$) used for training the RF model. The first column contains the results of a dummy model, which simply predicts the mean score of each algorithm in the training data and only serves as a baseline against which the other models are compared.

4.2.1. *Comparing AS model performance with a baseline model*

In this subsection, we focus on comparing the performance of the AS model with a simple baseline.

$3d$ analysis: Looking at the pairwise ranking score (in the left column of Fig. 2), we can see that the feature-based models outperform the dummy, except for the cases BBOB-ZIGZAG, AFFINE-ZIGZAG, ZIGZAG-BBOB, and ZIGZAG-AFFINE. The transformer features commonly outperform ELA in the cases AFFINE-BBOB, AFFINE-RANDOM, ZIGZAG-BBOB, while the opposite is true for BBOB-RANDOM, RANDOM-AFFINE, RANDOM-BBOB and RANDOM-ZIGZAG. For the remaining cases, both feature types provide similar results.

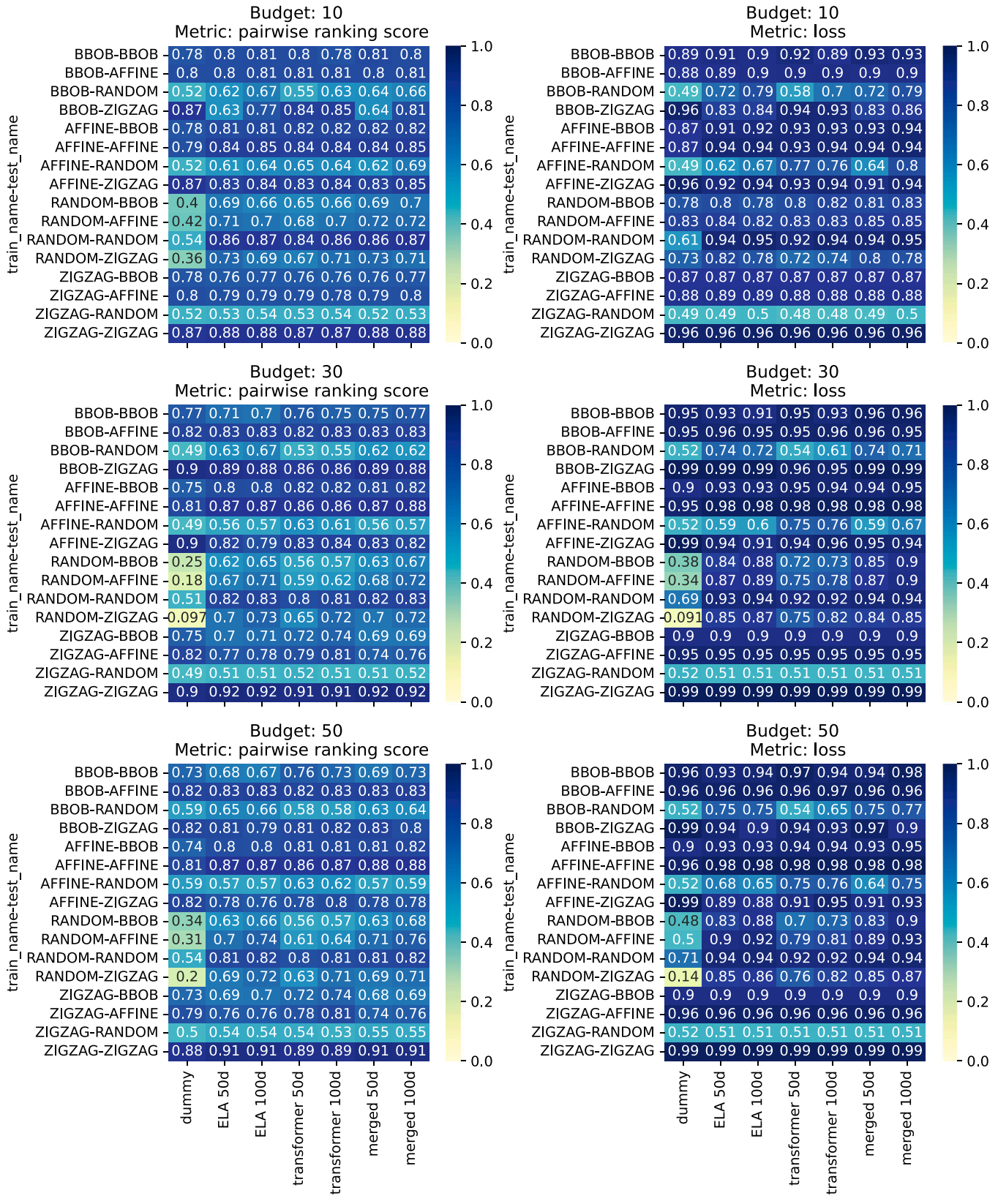


Fig. 2. Pairwise ranking score and loss achieved by the models for 3d problems.

Looking at the loss metric (in the right column of the figure), there are several cases where the feature-based models achieve very good results (above 0.90), but do not outperform the dummy. This indicates that the models perform well, however, the task itself is not very difficult. The reason for this is that when two benchmarks have the same best-performing algorithm (i.e., single best solver), the dummy model is very strong and it is difficult for any of the feature-based models to achieve a better loss. This is the case for the BBOB, AFFINE, and

ZIGZAG benchmarks, on which the ES algorithm largely outperforms the others, resulting in very good performance of the dummy model when the training and testing is executed on a combination of these benchmarks. In these cases, none of the models can outperform the dummy in terms of the loss metric, or outperform it by a very small margin.

The RANDOM benchmark has a different distribution of the algorithm performance scores compared to the other benchmarks. Consequently, the loss of the dummy model trained or tested on the RANDOM benchmark is somewhat lower, and the feature-based models manage to obtain a better loss. The transformer model obtains slightly better loss than the ELA features in the AFFINE-BBOB, AFFINE-RANDOM, and AFFINE-ZIGZAG, while the ELA features perform better in the cases RANDOM-ZIGZAG, RANDOM-AFFINE, RANDOM-BBOB, and BBOB-RANDOM.

10d analysis: In our experiments for 10d problems, available in the appendix, we can again observe the pattern of the dummy outperforming all of the feature-based models for all budgets, which is due to the benchmarks sharing the same single best solver. The transformer features outperform ELA in the case AFFINE-RANDOM, while the ELA features outperform the transformer in the cases ZIGZAG-BBOB, ZIGZAG-AFFINE and RANDOM-ZIGZAG. However, in the majority of cases, both feature types provide comparable results.

4.2.2. Comparison of the feature-based AS models

To summarize the obtained results, Figs. 3(a) and 3(b) show the number of times a specific model outperforms another model type, as well as the mean difference in the loss obtained by the models for 3d and 10d problems, respectively. In this case, we are presenting the generalization results, i.e., when the training and testing benchmark is different, since this is the primary focus of our study. We are only considering the loss metric, since we believe it is more relevant for a practical application of an AS, where we are primarily interested in selecting only the best algorithm, and may not be so interested in the pairwise rankings of the algorithms which is captured by the pairwise ranking score.

The rows in the figures denote which pair of models are compared, while the columns indicate the benchmark that was used for testing the AS. Each cell can have a maximum value of nine, since each model is trained on three different benchmark suites and evaluated on the benchmark suite presented in the column, across three different budgets. The loss results presented are aggregated across different training benchmark suites and different budgets. With this analysis, we aim to analyze which feature types and sample sizes used to calculate the features are better suited for which benchmark.

3d analysis: The first row in Fig. 3(a), labeled as "ELA 100d > dummy (count)" indicates how many times (out of 9) the ELA features calculated with a sample size of 100d outperform the dummy model. In contrast, the second row shows the mean difference in the losses obtained with these models.

From the first six rows of Fig. 3(a), we can see that the ELA, transformer and merged features generally outperform the dummy on the AFFINE, BBOB and RANDOM benchmarks, but not on the ZIGZAG benchmark.

Rows 7–12 indicate that the differences between the losses obtained with a same feature type calculated using different sample sizes are fairly low, meaning that using a larger sample size may not be practically relevant. Evidence of a 100d sample size being preferable over a 50d sample size is only prominent for the transformer features applied on the AFFINE benchmark and the merged features applied on the BBOB benchmark, however, the observed mean differences are less than 0.02.

Considering rows 13–16, which analyze whether the transformer features obtain better results than the ELA features, we can see that the ELA features consistently outperform the transformer for the AFFINE benchmark with a sample size of 50d, while mixed outcomes are obtained for the other benchmarks. The mean difference between the performance obtained with the ELA and transformer features is always less than 0.03.

Finally, the last four rows investigate whether using both the transformer features and the ELA features together provides better results than using only the ELA features. In this case, we can again observe mixed outcomes with mean differences in performance being less than 0.03, which does not allow us to make a firm conclusion.

10d analysis: Looking at Fig. 3(b), summarizing the results for the 10d problems, we can see that the ELA and transformer features do not consistently beat the dummy, with the ELA features beating the dummy more often than the transformer. The performance obtained with different sample sizes used for feature calculation is consistent with the 3d problems, in that the mean differences in performance are quite low, and a strong argument cannot be made for the use of one sample size over another.

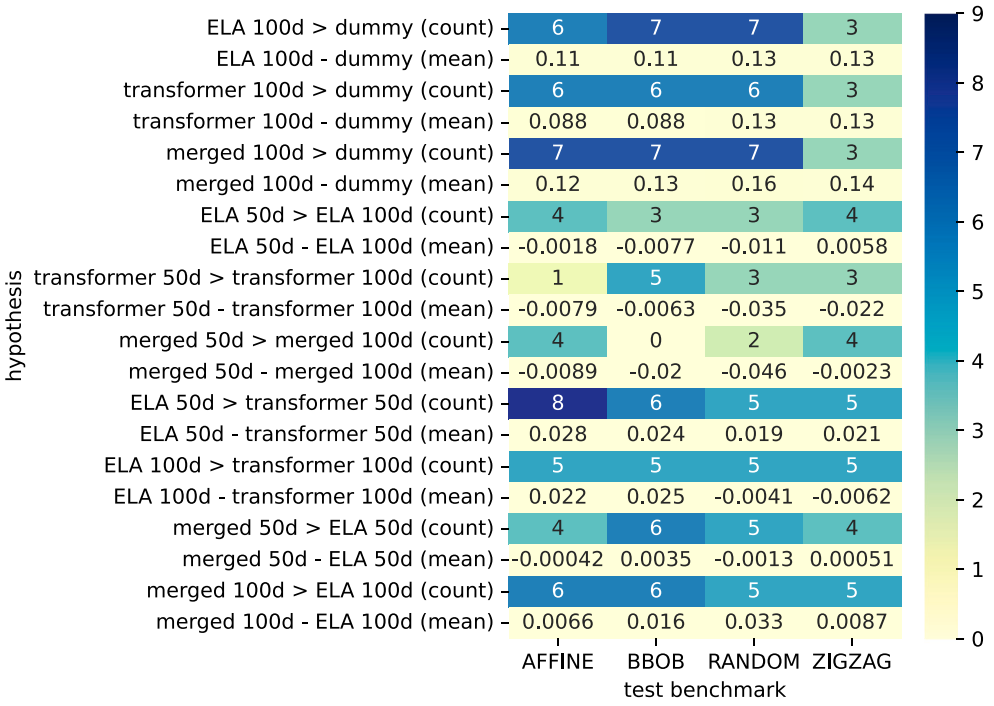
The comparison of the ELA and transformer features is also consistent with the 3d problems, in that ELA features provide better results on the BBOB and AFFINE benchmarks, with mean differences in the range 0.04–0.08. The transformer features provide better results on the RANDOM benchmark, while no clear conclusion can be drawn for the ZIGZAG benchmark.

The analysis of whether the merged features outperform using only the ELA features for 10d problems shows conflicting outcomes for different benchmarks and sample sizes, with low mean differences, which do not allow for a well-founded conclusion.

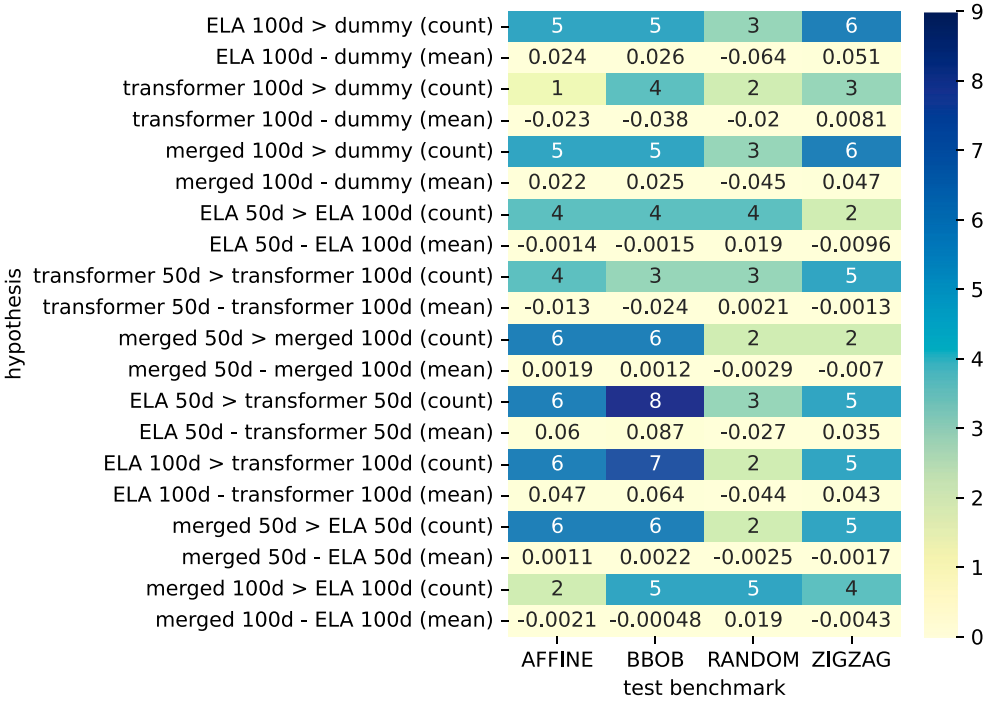
To summarize, in this subsection, we show that there are combinations of training and testing benchmarks for which the feature-based models do not outperform the dummy model. The dummy model is particularly strong when the pairs of benchmarks have the same single-best solver. The comparison of the ELA and transformer features indicates that the ELA features perform better on the BBOB and AFFINE benchmarks, while the transformer features perform better on the RANDOM benchmark. For the ZIGZAG benchmark, conclusive results are not obtained across problem dimensions, regarding the superiority of ELA or transformer features.

4.3. Distributions of feature values

To identify the reasons for the lack of generalization from one benchmark to another (inability of feature-based models to outperform the dummy), we analyze the distributions of the values of the most important features for the models trained on each benchmark. To this end, we first find the top 10 most important features for the AS trained for each budget and each benchmark, by taking the median of the feature importance scores produced by the RF model in each of the 10 training repetitions. Figs. 4 and 5 show the distribution of feature values (scaled in the range [0,1] for visualization purposes only) for the 10 most important features of the AS trained on each training benchmark for 3d problems and a budget of 50 iterations. Fig. 4 refers to the 10 most important ELA features, while Fig. 5 refers to the 10 most important transformer features. The figures are divided into four subplots, each one containing the top features of the AS trained on a different benchmark. The boxplots in different colors reflect the distribution of feature values across the four benchmarks. In Fig. 4, we can see that for all benchmark suites some ELA features appear with no variation (e.g., *ela_meta.quad_simple.cond*, *basic.objective_min*). However, we need to clarify, that they do have different values. The impression that they have no variation in the figure is due to the fact that there are a few problems that have extreme values for these features. Including these problems in the figure makes the variation of the feature values of the other problems look negligible. From Fig. 4, we can see that for some features, the RANDOM benchmark has feature values that are outside of the range of feature



(a) Summary results for $3d$ problems



(b) Summary results for $10d$ problems

Fig. 3. Summary results in terms of the loss metric.

values observed in other benchmarks. Such is the case for the features *ic.eps.max*, *ela_meta.lin_simple.cond*, *ela_meta.lin_simple.coef.max*, *ela_meta.quad_simple.cond*, *basic.objective_max*, *basic.objective_min*, *nbc.nn_nb.mean_ratio*, and *ic.m*0. One thing to point out is that some of the features (most apparent for the features *basic.objective_max*, and *basic.objective_min*, measuring the maximum and minimum value of the objective function values in the sample) are very sensitive to the range of the objective function values. A possible solution to ensuring these features have within-distribution values for all benchmarks is to scale the objective function values before computing the ELA features. While scaling the objective value would solve the issue with the features *basic.objective_max*, and *basic.objective_min*, it is not completely straightforward to say whether it would help with the other features. These results are also aligned with results from a recent study [40],

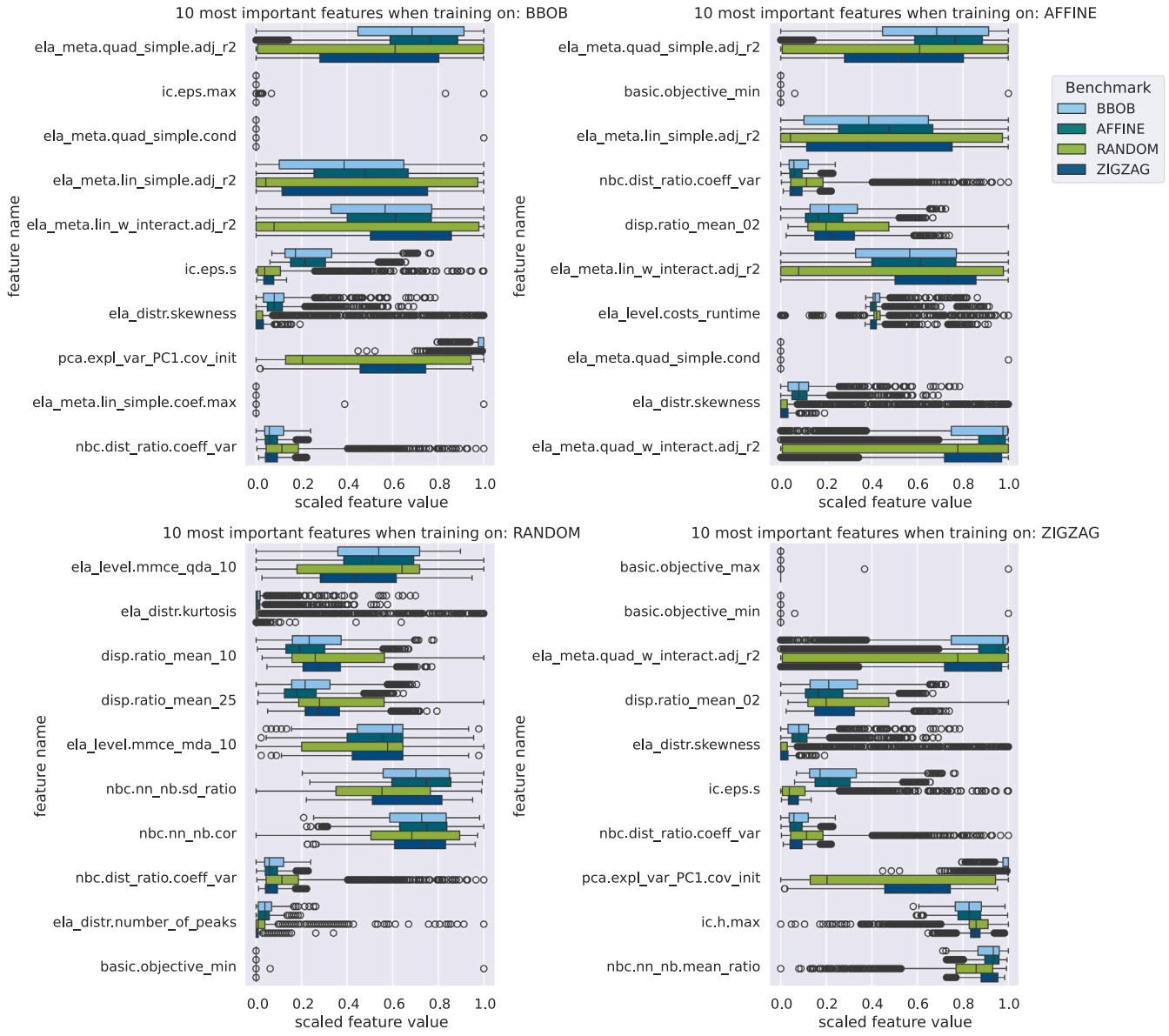


Fig. 4. Distribution of the values of the top most important ELA features for the AS trained for $3d$ problems and a budget of 50 iterations.

where it has been shown that some features are not invariant of shift and scaling if the objective value is not scaled. However, the scaling of the objective function is left for future work as a separate topic and we have performed the experiments with no scaling which is the most common approach presented in the AS studies.

Due to the fact that the transformer features are not explicitly defined, but are generated from a neural network architecture, they do not have dedicated names. Because of this, in Fig. 5, we simply name the features according to their index in the feature vector. In this case, we can also observe that there are several features for which some problem instances from the RANDOM benchmark have values which are out of the distribution of the feature values of other benchmarks. Such is the case, for instance, for features 59, 73, and 63. We should point out here that training a transformer model requires the scaling of the objective function values before computing the features. This means that in this case, the out-of-distribution values obtained for some of the RANDOM problem instances are not simply a result of excessive ranges of objective values, but an indication that there is truly something inherently different about these problems. We can also observe the same for the ZIGZAG benchmark, however, only for feature 49.

In summary, in this subsection, we observe that the RANDOM benchmark has feature values which are outside of the distribution of the other benchmarks, which can explain the poor performance which is sometimes achieved when an AS model is trained on another benchmark, and transferred to the RANDOM benchmark. This behavior is less prominent in case of the transformer features compared to the ELA features, which can also explain why the transformer features outperform ELA when the testing is executed on the RANDOM benchmark, as we saw in the previous section.

4.4. Coverage of the problem landscape

In this subsection, we aim to assess the similarity between problem instances of different benchmarks and analyze the benchmark's coverage of the problem landscape, in order to see if this can explain the generalizability of AS models trained on these benchmarks. To identify the distribution of problem instances from different problem benchmarks, we perform clustering of the problem instances from all benchmarks and analyze the distribution of the benchmarks across clusters. We have performed the clustering twice, using both features

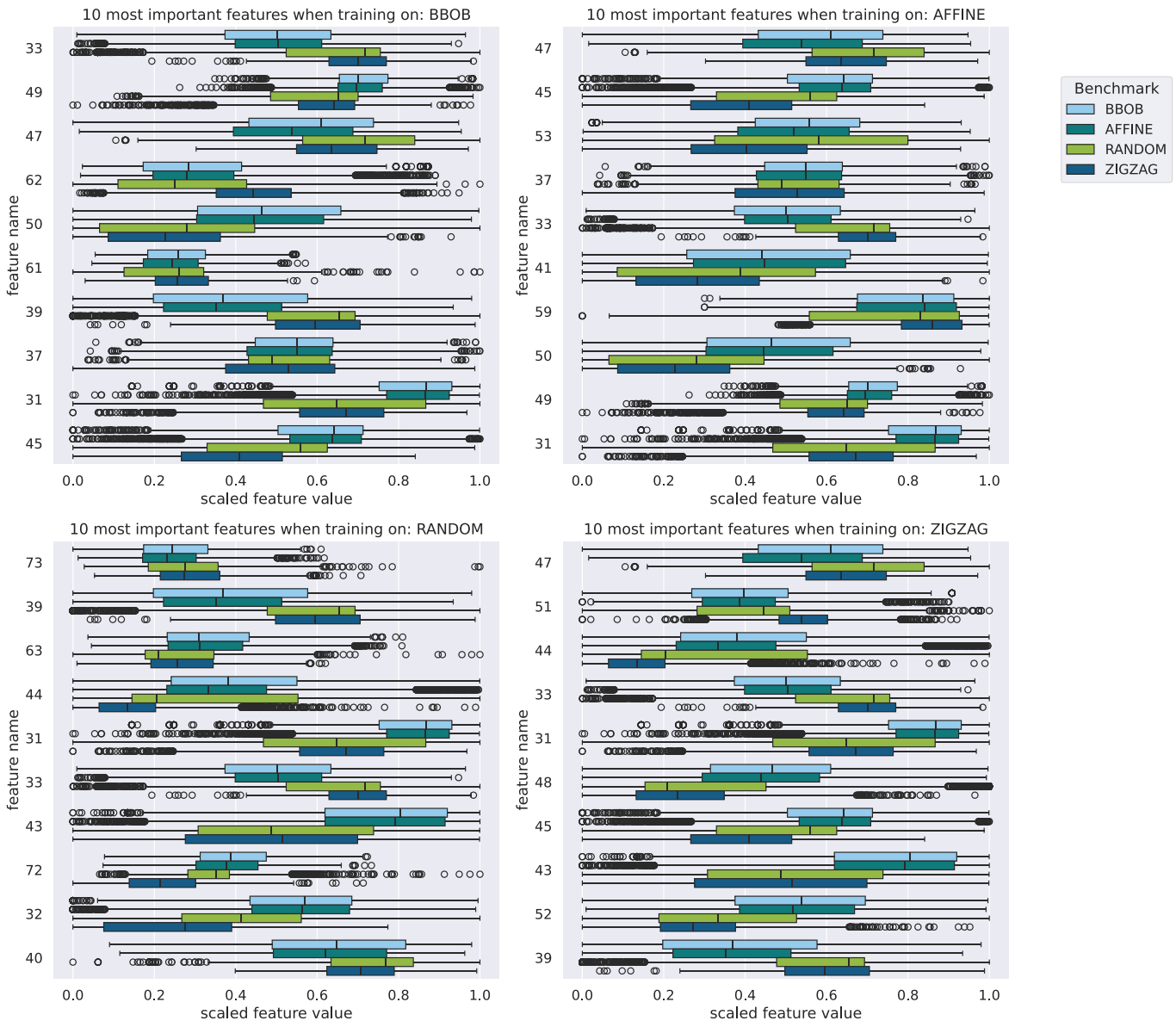


Fig. 5. Distribution of the values of the top most important transformer features for the AS trained for $3d$ problems and a budget of 50 iterations.

(ELA and transformer-based) separately. To this end, we first merge the problem instances from all benchmarks into one set. We then scale the feature representations of the entire set in the range [0,1], and estimate the optimal number of clusters using the elbow method. Finally, we cluster the problem instances using KMeans clustering with Euclidean distance. Figs. 6 and 7 show the number of problem instances from each benchmark that belong to each of the clusters. Fig. 6 refers to the results of the clustering performed using ELA features calculated on a $100d$ sample, while Fig. 7 shows the results of the clustering done using the transformer features calculated on a $100d$ sample.

The optimal number of clusters is determined to be seven when the clustering is done using the ELA features, and five when using the transformer features. The difference in the optimal number of clusters obtained could be owed to the fact that the transformer features are trained for the task of classifying BBOB problem instances, and they are thus likely to embed the other benchmarks close to the BBOB problems (i.e., in the same vector space). When the clustering is done using ELA features, we can see that three clusters are almost exclusively populated by RANDOM problem instances, which is consistent with the analysis of the feature distributions, where we saw that the RANDOM benchmark has features with values outside of the distributions of the other benchmarks. On the other hand, the clustering using the transformer features still produces one cluster which is dominated by RANDOM problem instances, however, this cluster also contains two BBOB problem instances.

To summarize, the results presented in this subsection provide another confirmation that the RANDOM benchmark contains problem instances which are substantially different from other benchmarks and occupy their own part of the problem landscape.

4.5. Algorithm performance alignment to problem landscape features

In this subsection, we evaluate if the similarity of the problem landscape features is reflected in the raw algorithm performance. To this end, we analyze the similarity of algorithm performance in problem instances which are deemed to be similar according to their problem landscape features.

Fig. 8 features the algorithm performance metric as defined in Section 3.3 of the first instance of the first BBOB problem class (sphere) when $d = 3$, as well as the scores of the algorithms on the 10 problem instances from the ZIGZAG benchmark, which are most similar to this

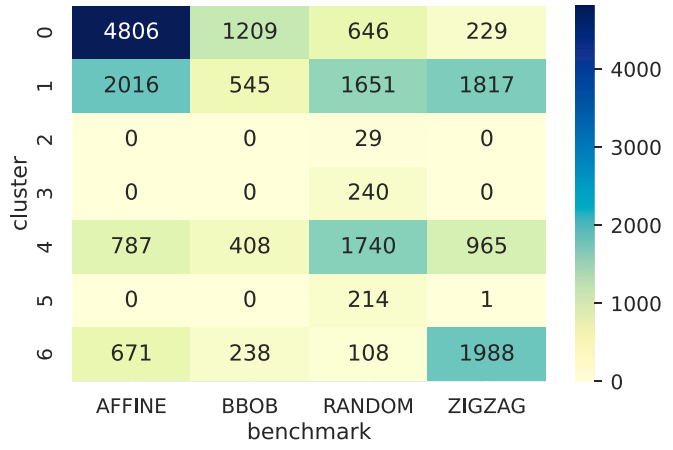


Fig. 6. Number of problem instances from each benchmark in the clusters generated using ELA features with a sample size of $100d$.

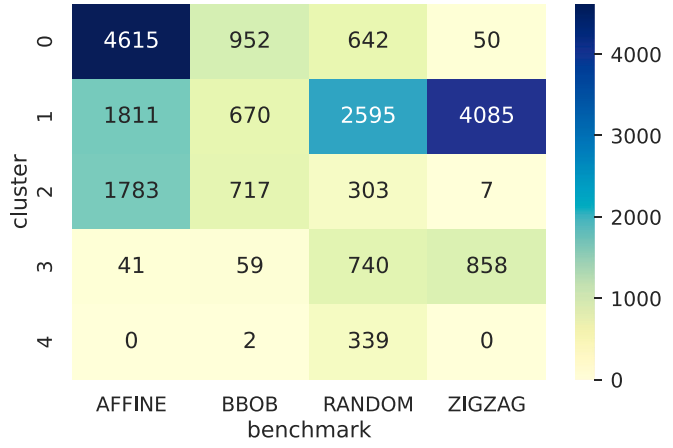


Fig. 7. Number of problem instances from each benchmark in the clusters generated using transformer features with a sample size of $100d$.

problem instance according to the cosine similarity of their problem landscape features. The similarity is calculated separately based on the ELA features, transformer features, and their combination, with a sample size of $100d$. In this case, the algorithm scores are calculated at a budget of 30 iterations for the algorithm execution. In each row, we present a matrix where the rows are different algorithms, the columns are different problem instances, and the values represent the score of the algorithm on the problem instance. The first column represents the query problem instance, i.e. the first instance of the first $3d$ BBOB problem class (Sphere). The remaining 10 columns represent the 10 most similar problem instances from the ZIGZAG benchmark.

The first plot refers to the ELA features, the second suplot refers to the transformer features, and the third subplot refers to the merged ELA and transformer representations.

On the query function, the ES algorithm has the best performance, with the DE algorithm being second-ranked, with a very similar performance. From the figure, we can observe the best-performing algorithm on most of the similar problem instances matches that of the query problem instance. This holds regardless of the type of features used to identify the similar problem instances.

On the other hand, Fig. 9 presents the opposite case, where similar problem landscape features do not result in similar algorithm performance. In this figure, the query problem instance in the first column is the first instance of the fifth $3d$ BBOB problem class (Linear Slope), where the DE, GA, and PSO provide similar results and are ranked much better than the ES algorithm. Looking at the results for the ELA features in the first plot, we can see that the best-performing algorithm is always ES. This is an example of a very poor alignment of the problem landscape features and algorithm performance. The transformer features and the merged features, provide similar results to the ELA features. These results indicate that there are problem instances for which there is no guarantee that similar problem feature distribution will lead to similar algorithm performance, which raises questions about the predictive power of the evaluated features.

Due to page limitations, we cannot present separate figures for each problem instance and each budget for the optimization algorithm execution. Instead, we opt to summarize the information presented in Fig. 8 in the following manner. We calculate the loss value of the algorithm scores of the query function in the first column and the scores of the top 10 most similar functions in the rest of the columns. We then take the mean of these 10 loss values. In this manner, the performance alignment reflected in each subplot of Figs. 8 and 9 is represented by a single number, which indicates how much the algorithm scores of the similar functions differ from the query function.

Fig. 10(a) shows the performance alignment of the first instance of the 24 $3d$ BBOB problem classes and their 10 most similar functions from the ZIGZAG benchmark. The figures depicting the same information for the RANDOM and AFFINE benchmarks can be found in the appendix. Each subplot refers to a different budget used for the algorithm execution. Within each subplot, the rows denote the features and sample size used to represent the problem instances when calculating the top 10 most similar functions, while the columns contain the BBOB problem instance used as a query function. We use the BBOB problem instances as the query problem instances since BBOB is the most well-understood and commonly used benchmark. From the figure, we can see that for most problem instances, the similarity in the problem landscape features is reflected in the similarity of the algorithm performance. There are, however, several problem instances where this does not hold. For instance, such is the case with the Buche-Rastrigin Function (4_1) and the Linear Slope function (5_1).

To see whether the lack of alignment between the similarity of problem landscape features and algorithm performance leads to worse performance of the AS, Fig. 10(b) shows the loss of the AS achieved on the query functions, i.e. the first instance of each BBOB problem class, when the training is done on the ZIGZAG benchmark.

Comparing Fig. 10(a) to Fig. 10(b), showing the alignment of algorithm performance to problem instance features, we can see that the AS performs worse on problem instances where there is a lack of alignment between similarity of problem landscape features and algorithm performance. For instance, we can see that the alignment is low on the Linear Slope function (5_1), and this is exactly where the loss of the algorithm selector is the lowest. These results show that there are problem instances for which the existing landscape features do not properly capture algorithm performance, which is in line with what was found in a previous work [15]. This indicates that there is a need for further development of problem features which will capture complimentary information related to algorithm performance.

5. Conclusion

In this paper, we assessed the generalizability of an algorithm selector across different benchmarks. In particular, we considered four benchmark suites (BBOB, AFFINE, ZIGZAG, and RANDOM) across two problem dimensions. We also evaluated whether the recently proposed transformer features can complement the well-established ELA features for the task of algorithm selection. Our results indicate that ELA features provide better results on the BBOB and AFFINE benchmarks, while the transformer features work better on the RANDOM benchmark. Our analysis points out several explanations for this phenomenon.

Firstly, when benchmark suites share a similar distribution in the raw algorithm performance across all included algorithms, and they

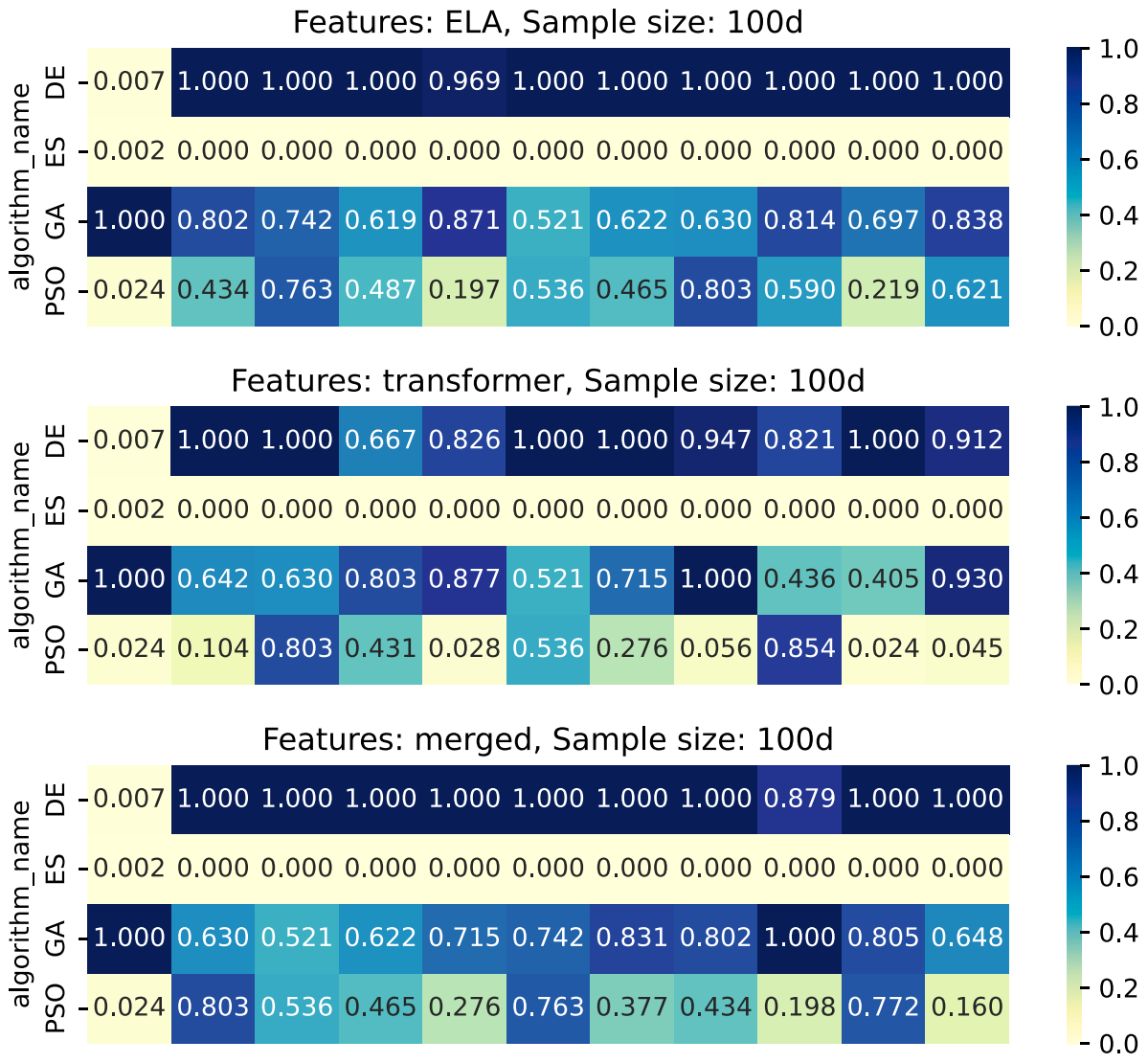


Fig. 8. Performance alignment of the first instance of the first 3*d* BBOB problem class and its 10 most similar problem instances from the ZIGZAG benchmark, for a budget of 30 iterations.

have the same single-best solver, the feature-based AS models exhibit inferior performance. These models, trained on one benchmark suite and tested on others, are outperformed by a baseline model that simply predicts mean performance based on the training data. Next, if the benchmark suite used for training the feature-based AS models differs in raw performance distribution of the algorithms and has a different single-best solver compared to the evaluation suites, these feature-based AS models surpass the baseline dummy model.

We find that the four algorithms (DE, GA, ES, and PSO) are ranked in a mostly consistent manner on the BBOB and AFFINE benchmarks, but rankings differ on the ZIGZAG and RANDOM benchmarks for different problem dimensions. These benchmarks also have similar feature distributions across the problem space. However, the algorithm rankings on the RANDOM benchmark deviate from the rankings on the other benchmarks, and the ELA feature values for some of the problem instances in this benchmark fall outside the ranges observed in the other benchmarks. This implies that the RANDOM benchmark contains some problem instances that are dissimilar to any problem instances from the other benchmarks in terms of ELA features. This is also the case for the transformer features, however, to a smaller extent. The values of the transformer features of the RANDOM benchmark are more consistent with the other benchmarks. A possible reason for this is that the transformer features require the normalization of the objective function values before feature extraction. Further investigation is needed to determine whether normalizing the objective function values before computing ELA features could improve their generalizability across benchmarks.

Our evaluation highlights a common challenge faced by AS models that there are problem instances for which there is no guarantee that similar problem features will lead to similar algorithm performance. This implies that the tested features might lack the necessary discriminatory power to accurately gauge algorithm performance in certain instances. This challenge remains consistent across both ELA and TransOpt features, which brings about the need for novel features to complement existing ones.

For future work, our first direction is to evaluate the ELA features by scaling the objective value before calculating them and see the effect on the AS task. We also plan to involve more landscape feature techniques that have been published recently including topological landscape features (TLA) [22] and features based on convolution neural networks [41]. Next, to find features that capture also algorithm performance, we are planning to evaluate trajectory-based features such as DynamoRep [42], and trajectory-based ELA [14,43]. Last but not least, we are planning to merge all problem instances from different benchmark suites, selecting only representative ones that cover the

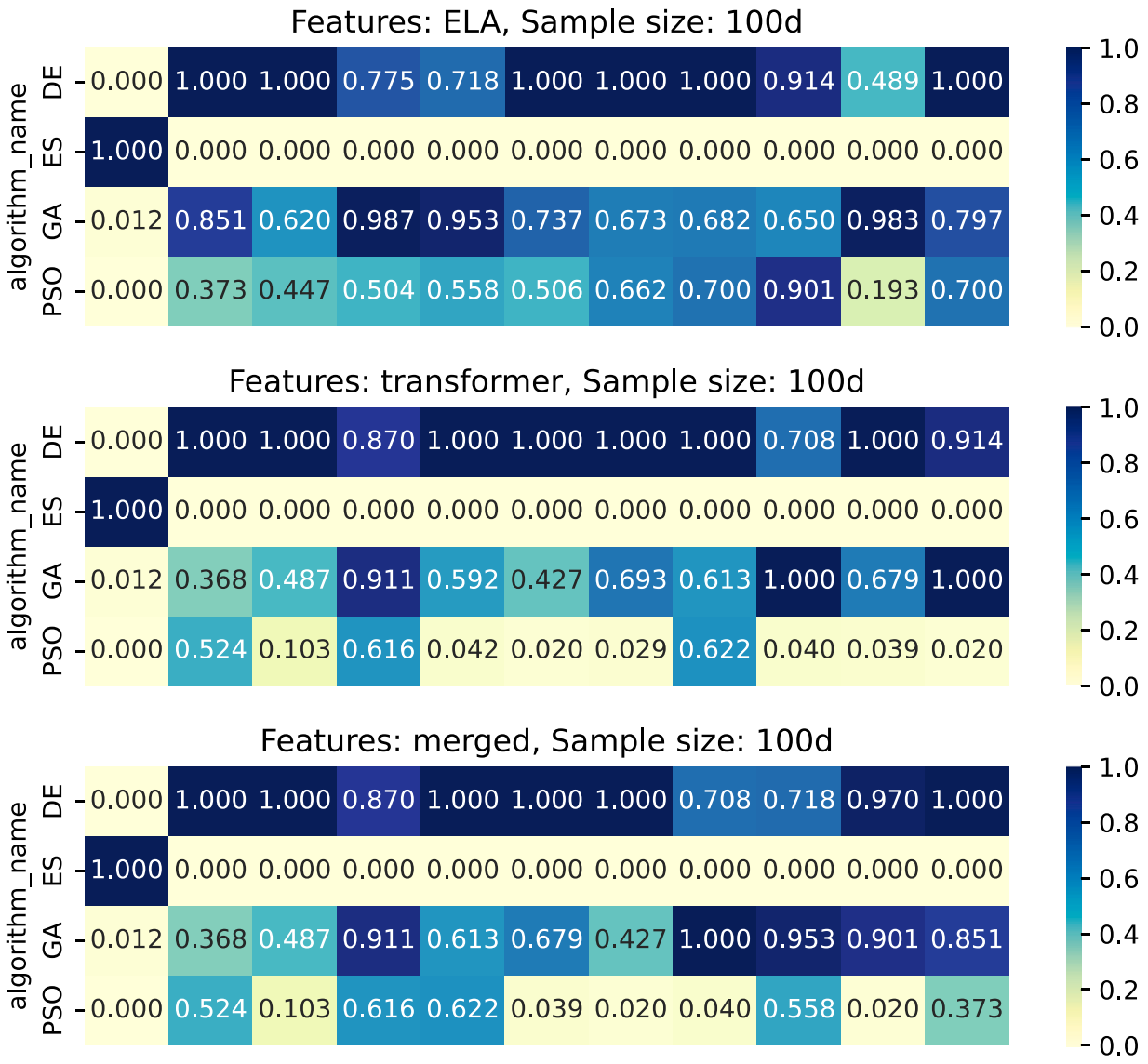


Fig. 9. Performance alignment of the first instance of the fifth 3*d* BBOB problem class and its 10 most similar problem instances from the ZIGZAG benchmark, for a budget of 30 iterations.

problem space as uniformly as possible [44,45], train an AS model, and further use transfer learning techniques to fine-tune it if new problem landscape spaces appear in the future that were not observed in the training phase.

CRediT authorship contribution statement

Gjorgjina Cenikj: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization, Funding acquisition. **Gašper Petelin:** Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Funding acquisition. **Tome Eftimov:** Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing, Resources, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

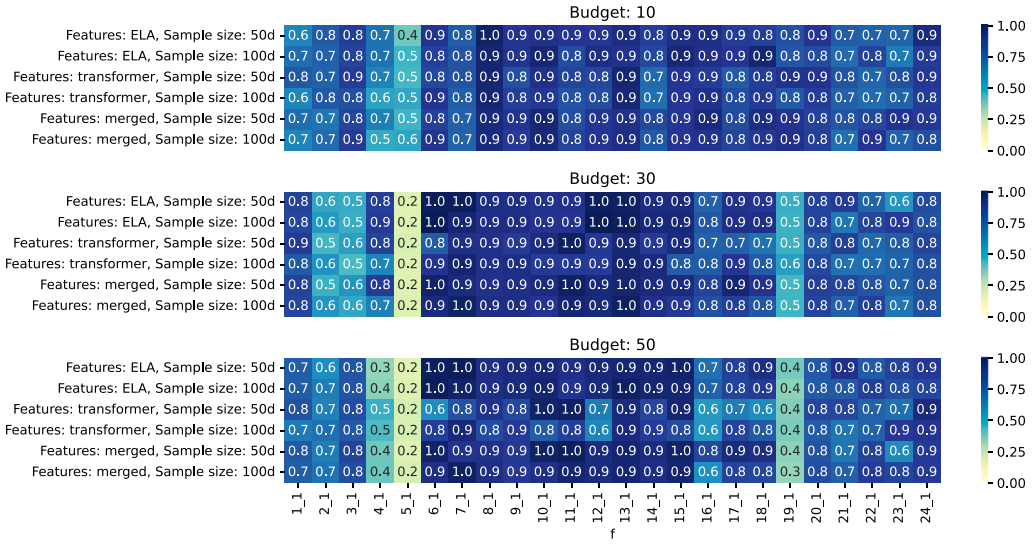
The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gjorgjina Cenikj reports financial support was provided by Slovenian Research and Innovation Agency. Tome Efimov reports financial support was provided by Slovenian Research and Innovation Agency. Gasper Petelin reports financial support was provided by Slovenian Research and Innovation Agency. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

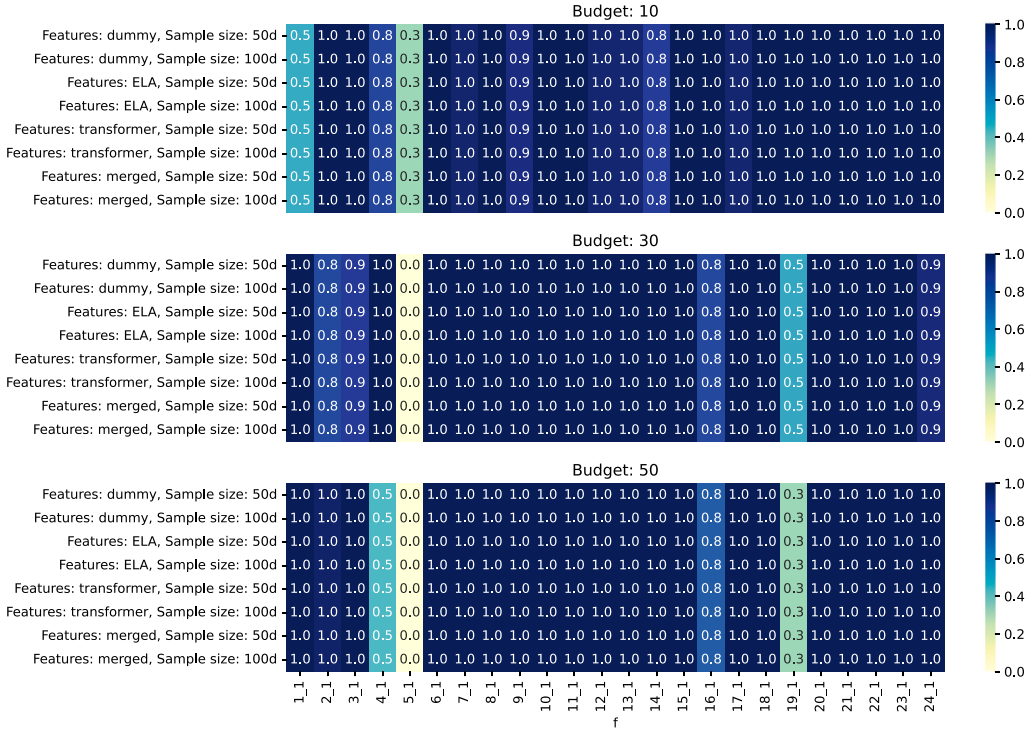
The data is publically available via shared linked in the repository.

Declaration of Generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to improve language and readability of their own writing. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.



(a) Performance alignment of the first instance of each 3*d* BBOB problem class to its most similar 10 problem instances from the ZIGZAG benchmark



(b) Loss achieved by the AS trained on the ZIGZAG benchmark, when tested on the first instance of each of the 3*d* BBOB problem classes

Fig. 10. Performance alignment of the first instance of each of the 3*d* BBOB problem classes to the ZIGZAG benchmark, and loss achieved on these instances when the model is trained on the ZIGZAG benchmark.

Acknowledgments

Funding in direct support of this work: Slovenian Research Agency: research core funding No. P2-0098, young researcher grants No. PR-12393 to GC and No. PR-11263 to GP, and project No. J2-4460.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.swevo.2024.101534.

References

[1] L. Kotthoff, Algorithm selection for combinatorial search problems: A survey, in: Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach, Springer International Publishing, Cham, 2016, pp. 149–190, http://dx.doi.org/10.1007/978-3-319-50137-6_7.

[2] P. Kerschke, H.H. Hoos, F. Neumann, H. Trautmann, Automated algorithm selection: Survey and perspectives, Evol. Comput. 27 (1) (2019) 3–45, http://dx.doi.org/10.1162/evco_a_00242, arXiv:https://direct.mit.edu/evco/article-pdf/27/1/3/1552398/evco_a_00242.pdf.

[3] K.M. Malan, A.P. Engelbrecht, A survey of techniques for characterising fitness landscapes and some possible ways forward, Inform. Sci. 241

(2013) 148–163, http://dx.doi.org/10.1016/j.ins.2013.04.015, URL https://www.sciencedirect.com/science/article/pii/S0020025513003125.

[4] K.M. Malan, A survey of advances in landscape analysis for optimisation, Algorithms 14 (2) (2021) http://dx.doi.org/10.3390/a14020040, URL https://www.mdpi.com/1999-4893/14/2/40.

[5] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, G. Rudolph, Exploratory landscape analysis, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO, 2011, pp. 829–836.

[6] Q. Renau, C. Doerr, J. Dreo, B. Doerr, Exploratory landscape analysis is strongly sensitive to the sampling strategy, in: Proc. of Parallel Problem Solving from Nature, PPSN, 2020, pp. 139–153.

[7] U. Škvorc, T. Eftimov, P. Korošec, The effect of sampling methods on the invariance to function transformations when using exploratory landscape analysis, in: 2021 IEEE Congress on Evolutionary Computation, CEC, 2021, pp. 1139–1146, http://dx.doi.org/10.1109/CEC45853.2021.9504739.

[8] U. Škvorc, T. Eftimov, P. Korošec, Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis, Appl. Soft Comput. 90 (2020) 106138, http://dx.doi.org/10.1016/j.asoc.2020.106138, URL https://www.sciencedirect.com/science/article/pii/S1568494620300788.

[9] A. Nikolikj, C. Doerr, T. Eftimov, RF+clust for leave-one-problem-out performance prediction, in: Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2023, pp. 285–301, http://dx.doi.org/10.1007/978-3-031-30229-9_19.

[10] A. Kostovska, A. Jankovic, D. Vermetten, S. Džeroski, T. Eftimov, C. Doerr, Comparing algorithm selection approaches on black-box optimization problems, in: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, in: GECCO '23 Companion, Association for Computing Machinery, New York, NY, USA, 2023, pp. 495–498, http://dx.doi.org/10.1145/3583133.3590697.

[11] N. Hansen, S. Finck, R. Ros, A. Auger, Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions, Report RR-6829, INRIA, 2009, URL https://hal.inria.fr/inria-00362633.

[12] U. Škvorc, T. Eftimov, P. Korošec, Transfer learning analysis of multi-class classification for landscape-aware algorithm selection, Mathematics 10 (3) (2022) http://dx.doi.org/10.3390/math10030432, URL https://www.mdpi.com/2227-7390/10/3/432.

[13] A. Nikolikj, G. Cenikj, G. Ispirova, D. Vermetten, R.D. Lang, A.P. Engelbrecht, C. Doerr, P. Korošec, T. Eftimov, Assessing the generalizability of a performance predictive model, 2023, arXiv:2306.00040.

[14] A. Kostovska, A. Jankovic, D. Vermetten, J. de Nobel, H. Wang, T. Eftimov, C. Doerr, Per-run algorithm selection with warm-starting using trajectory-based features, in: Parallel Problem Solving from Nature, PPSN, in: LNCS, vol. 13398, Springer, 2022, pp. 46–60, http://dx.doi.org/10.1007/978-3-031-14714-2_4.

[15] B. Lacroix, J. McCall, Limitations of benchmark sets and landscape features for algorithm selection and performance prediction, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 261–262, http://dx.doi.org/10.1145/3319619.3322051.

[16] Y. Tian, S. Peng, X. Zhang, T. Rodemann, K.C. Tan, Y. Jin, A recommender system for metaheuristic algorithms for continuous optimization based on deep recurrent neural networks, IEEE Trans. Artif. Intell. 1 (1) (2020) 5–18, http://dx.doi.org/10.1109/TAI.2020.3022339.

[17] J. Kudela, R. Matousek, New benchmark functions for single-objective optimization based on a zigzag pattern, IEEE Access 10 (2022) 8262–8278, http://dx.doi.org/10.1109/ACCESS.2022.3144067.

[18] D. Vermetten, F. Ye, T. Bäck, C. Doerr, MA-BBOB: Many-affine combinations of BBOB functions for evaluating automl approaches in noiseless numerical black-box optimization contexts, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO, GECCO '23, Association for Computing Machinery, New York, NY, USA, 2023, pp. 813–821.

[19] K. Dietrich, O. Mersmann, Increasing the diversity of benchmark function sets through affine recombination, in: G. Rudolph, A.V. Kononova, H. Aguirre, P. Kerschke, G. Ochoa, T. Tušar (Eds.), Parallel Problem Solving from Nature – PPSN XVII, Springer International Publishing, Cham, 2022, pp. 590–602.

[20] B. van Stein, F.X. Long, M. Frenzel, P. Krause, M. Gitterle, T. Bäck, DoE2Vec: Deep-learning based features for exploratory landscape analysis, in: Proceedings of the Companion Conference on Genetic and Evolutionary Computation, in: GECCO '23 Companion, Association for Computing Machinery, New York, NY, USA, 2023, pp. 515–518, http://dx.doi.org/10.1145/3583133.3590609.

[21] G. Cenikj, G. Petelin, T. Eftimov, TransOpt: Transformer-based representation learning for optimization problem classification, in: 2023 IEEE Symposium Series on Computational Intelligence, 2023, URL http://arxiv.org/abs/2311.18035.

[22] G. Petelin, G. Cenikj, T. Eftimov, TLA: Topological landscape analysis for single-objective continuous optimization problem instances, in: Proceedings of IEEE Symposium Series on Computational Intelligence, 2022, pp. 1698–1705, http://dx.doi.org/10.1109/SSCI51031.2022.10022126.

[23] R. Storn, K. Price, Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.

[24] V. Chahar, S. Katoch, S. Chauhan, A review on genetic algorithm: Past, present, and future, Multimedia Tools Appl. 80 (2021) http://dx.doi.org/10.1007/s11042-020-10139-6.

[25] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968, vol.4.

[26] H. Beyer, H. Schwefel, Evolution strategies - a comprehensive introduction, Nat. Comput. 1 (1) (2002) 3–52, http://dx.doi.org/10.1023/A:1015059928466.

[27] R. Tanabe, Benchmarking feature-based algorithm selection systems for black-box numerical optimization, IEEE Trans. Evol. Comput. 26 (6) (2022) 1321–1335.

[28] J. Liang, B. Qu, P. Suganthan, Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2013.

[29] A. Nikolikj, M. Pluhacek, C. Doerr, P. Korosec, T. Eftimov, Sensitivity analysis of RF+clust for leave-one-problem-out performance prediction, in: IEEE Congress on Evolutionary Computation, CEC 2023, Chicago, IL, USA, July 1-5, 2023, IEEE, 2023, pp. 1–8, http://dx.doi.org/10.1109/CEC53210.2023.10254146.

[30] B. Lacroix, L.A. Christie, J.A.W. McCall, Interpolated continuous optimisation problems with tunable landscape features, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 169–170, http://dx.doi.org/10.1145/3067695.3076045.

[31] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, D. Brockhoff, COCO: A platform for comparing continuous optimizers in a black-box setting, Optim. Methods Softw. (2020) 1–31.

[32] D. Vermetten, F. Ye, C. Doerr, Using affine combinations of BBOB problems for performance assessment, 2023, arXiv preprint arXiv:2303.04573.

[33] F.X. Long, B. van Stein, M. Frenzel, P. Krause, M. Gitterle, T. Bäck, Learning the characteristics of engineering optimization problems with applications in automotive crash, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 1227–1236, http://dx.doi.org/10.1145/3512290.3528712.

[34] J. Blank, K. Deb, Pymoo: Multi-objective optimization in python, IEEE Access 8 (2020) 89497–89509.

[35] J. Menčík, Latin hypercube sampling, in: J. Mencik (Ed.), Concise Reliability for Engineers, IntechOpen, Rijeka, 2016, http://dx.doi.org/10.5772/62370, Ch. 16.

[36] P. Kerschke, H. Trautmann, Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco, in: N. Bauer, K. Ickstadt, K. Lübke, G. Szepannek, H. Trautmann, M. Vichi (Eds.), Applications in Statistical Computing: From Music Data Analysis to Industrial Quality Improvement, Springer, 2019, pp. 93–123, http://dx.doi.org/10.1007/978-3-030-25147-5_7.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[38] R. Shwartz-Ziv, A. Armon, Tabular data: Deep learning is not all you need, Inf. Fusion 81 (2022) 84–90.

[39] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.

[40] R.P. Prager, H. Trautmann, Nullifying the inherent bias of non-invariant exploratory landscape analysis features, in: Applications of Evolutionary Computation, Springer-Verlag, Berlin, Heidelberg, 2023, pp. 411–425, http://dx.doi.org/10.1007/978-3-031-30229-9_27.

[41] M.V. Seiler, R.P. Prager, P. Kerschke, H. Trautmann, A collection of deep learning-based feature-free approaches for characterizing single-objective continuous fitness landscapes, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO, GECCO '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 657–665, http://dx.doi.org/10.1145/3512290.3528834.

[42] G. Cenikj, G. Petelin, C. Doerr, P. Korošec, T. Eftimov, DynamoRep: Trajectory-based population dynamics for classification of black-box optimization problems, 2023, pp. 813–821, http://dx.doi.org/10.1145/3583131.3590401.

[43] A. Janković, C. Doerr, Adaptive landscape analysis, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO, GECCO '19, Association for Computing Machinery, 2019, pp. 2032–2035, http://dx.doi.org/10.1145/3319619.3326905.

[44] G. Cenikj, R. Dieter Lang, A. Petrus Engelbrecht, C. Doerr, P. Korošec, T. Eftimov, SELECTOR: Selecting a representative benchmark suite for reproducible statistical comparison, in: Proc. of Genetic and Evolutionary Computation Conference, GECCO, 2022.

[45] T. Eftimov, G. Petelin, G. Cenikj, A. Kostovska, G. Ispirova, P. Korošec, J. Bogatinovski, Less is more: Selecting the right benchmarking set of data for time series classification, Expert Syst. Appl. 198 (2022) 116871, http://dx.doi.org/10.1016/j.eswa.2022.116871, URL https://www.sciencedirect.com/science/article/pii/S0957417422003189.