SELECTOR: Selecting a Representative Benchmark Suite for Reproducible Statistical Comparison

Gjorgjina Cenikj, Ryan Dieter Lang, Andries Petrus Engelbrecht, Carola Doerr, Peter Korošec, Tome Eftimov

SELECTOR: Selecting a Representative Benchmark Suite for Reproducible Statistical Comparison

Gjorgjina Cenikj
Computer Systems Department
Jožef Stefan Institute
Slovenia
gjorgjina.cenikj@ijs.si

Ryan Dieter Lang
Computer Science Division
Stellenbosch University
South Africa
langr@sun.ac.za

Andries Petrus Engelbrecht
Department of Industrial Engineering,
and Computer Science Division
Stellenbosch University
South Africa
engel@sun.ac.za

Carola Doerr
Sorbonne Université
CNRS, LIP6
Paris, France
carola.doerr@lip6.fr

Peter Korošec
Computer Systems Department
Jožef Stefan Institute
Slovenia
peter.korosec@ijs.si

Tome Eftimov
Computer Systems Department
Jožef Stefan Institute
Slovenia
tome.eftimov@ijs.si

ABSTRACT

Fair algorithm evaluation is conditioned on the existence of high-quality benchmark datasets that are non-redundant and are representative of typical optimization scenarios. In this paper, we evaluate three heuristics for selecting diverse problem instances which should be involved in the comparison of optimization algorithms in order to ensure robust statistical algorithm performance analysis. The first approach employs clustering to identify similar groups of problem instances and subsequent sampling from each cluster to construct new benchmarks, while the other two approaches use graph algorithms for identifying dominating and maximal independent sets of nodes. We demonstrate the applicability of the proposed heuristics by performing a statistical performance analysis of five portfolios consisting of three optimization algorithms on five of the most commonly used optimization benchmarks.

The results indicate that the statistical analyses of the algorithms' performance, conducted on each benchmark separately, produce conflicting outcomes, which can be used to give a false indication of the superiority of one algorithm over another. On the other hand, when the analysis is conducted on the problem instances selected with the proposed heuristics, which uniformly cover the problem landscape, the statistical outcomes are robust and consistent.

CCS CONCEPTS

• **Theory of computation → Continuous optimization**.

KEYWORDS

benchmarking, black-box optimization, single-objective optimization, optimization algorithm performance evaluation

1 INTRODUCTION

Reliable and unbiased algorithm performance evaluation plays an indispensable role in the analysis of the strengths and weaknesses of existing algorithms, tracking improvements, and establishing future research efforts [4].

The main task of benchmarking in evolutionary computation involves evaluation of the performance of an algorithm against other algorithms. To do this, three main questions should be taken with great care: i) which problem instances should be selected for the comparison study, ii) how to design the experiments that lead to reproducible and replicable results, and iii) which performance measures should be analyzed and with which statistical approach.

Even though great progress has been achieved by proposing more robust statistical methodologies for benchmarking, those approaches focused only on the third question of the benchmarking theory, with the assumption that the first and the second one are already done with great care. However, the selection of the problem instances (i.e., the benchmark suite) that will be included in the analysis can have a huge impact on the experimental design and statistical analysis performed using the performance data. It can happen that the same algorithm portfolio (i.e., set of algorithm instances) evaluated using some statistical methodology on different sets of problem instances results in different winning algorithms. This means that the selection of the problem instances can lead to biased performance analysis (i.e., selecting problem instances in favor of the winning algorithm). This allows researchers to present results that make their algorithm look superior to the others.

In the field of single-objective optimization, algorithm performance is commonly assessed experimentally, for example on the benchmarks provided by the IEEE Congress on Evolutionary Computation (CEC) Special Sessions and Competitions on Real-Parameter Single-Objective Optimization [20–22, 39] and the Genetic and Evolutionary Computation Conference (GECCO) Black-box Optimization Benchmarking (BBOB) [11, 12] workshops. However, recent studies suggest that the CEC and the BBOB benchmark suites may provide poor coverage of the space of numerical optimization problems [18, 27, 35]. Even more, it has also been shown that the benchmark problem classes are highly correlated from the perspective of the performance space [5, 42].

In this paper, we address the selection of problem instances which should be involved in the comparison of optimization algorithms in order to ensure robust statistical algorithm performance analysis.

One approach to select the problem instances is the Instance Space Analysis (ISA) methodology [36], which uses visualization to assess the effect of instance characteristics on algorithm performance, by finding areas in the problem landscape space where some algorithms perform better than the others. The idea is to

select instances that maximize the performance difference between algorithms to highlight their strengths and weaknesses [33, 40]. Racing has also been explored for the selection of problems that highlight performance differences between algorithms [41]. An alternative approach was proposed in [38] where instances are selected based on clustering applied in *performance space*, i.e., they were selected to maximize the diversity of the comparisons that one could perform between the algorithms. While our approach has a stronger motivation in selecting instances for robust comparisons, the approach in [38] has a stronger motivation in benchmarking for algorithm analysis.

Other studies explore the diversity of single-objective optimization problems [19, 25, 34], without focusing on selection heuristics that can produce more comprehensive benchmark datasets.

The main difference of our approach with the aforementioned approaches is that the selection is done based on landscape characteristics of the problem instances, without investigating the relations with the performance space. Consequently, our approach is not affected by the selection of the algorithm portfolio which will be included in the analysis and it enables the selection of diverse instances based on the landscape characteristics, regardless of which optimization algorithms will be further run and compared.

Our contribution: We evaluate three approaches for the selection of problem instances, which should be involved in the comparison of optimization algorithms in order to ensure robust statistical algorithm performance analysis, all of which first involve generating a numerical representation of the problem instances using exploratory landscape analysis (ELA). The first approach then employs clustering to identify similar groups of problem instances and subsequent uniform sampling from each cluster to construct new benchmarks. The other two approaches are based on graph theory algorithms for finding a dominating set (DS) and a maximal independent set (MIS) of nodes in a graph.

We show that the results of the statistical algorithm performance analysis executed on a single benchmark (e.g., BBOB, CEC2013, CEC2014, CEC2015, and CEC2017) are not consistent when a different benchmark is being used. Such results allow researchers to select the benchmark where their algorithm is superior, which are bias to the researchers' preference. On the other hand, the subsets of problem instances produced by the proposed approaches provide a robust, consistent, and reproducible statistical analysis.

Outline: The reminder of the paper is organized as follows: Section 2 provides the background, Section 3 presents three different sampling heuristics to select problem instances that guarantee reproducible statistical outcomes. Our key results are presented in Section 4. We conclude the paper in Section 5.

Reproducibility: The performance data as well as the landscape data used in this study are available online at https://zenodo.org/record/5940558. The code is available at https://anonymous.4open.science/r/SELECTOR-80F1.

2 BACKGROUND

In this section, we present two core components of our analysis: exploratory landscape analysis (Section 2.1, used to characterize the problem instances via numerical representations) and statistical performance assessment (Section 2.2, used to compare the algorithms).

2.1 Exploratory Landscape Analysis

Exploratory landscape analysis (ELA) [24] is an approach to characterize black-box optimization problem instances via numerical measures (*features*) that each describe a different aspect of the problem instances. A convenient way to compute ELA features is provided by the *flacco* R package [17]. This packages offers 343 different feature values split into 17 features groups, including dispersion, information content, meta model, nearest better clustering, principal component analysis.

2.2 Performance Assessment

Once the benchmark problems are selected, the performance assessment can be done in a fixed-budget or fixed-target scenario. The fixed-budget scenario measures the quality of the solution achieved with a given computational budget, while the fixed-target scenario returns the budget required to achieve a certain target.

The quality of a solution in single-objective optimization can be expressed in terms of absolute fitness value or in terms of *target precision*, i.e., the difference of this absolute fitness to that of an optimal solution. After selecting the performance measure, statistical analyses play an important role in the interpretation of the obtained results. A common practice is use a non-parametric test on the mean results obtained from multiple runs of each algorithm on each problem instance [9]. Recent advances involve the use of Deep Statistical Comparisons (DSC) [7]. In comparison to using individual descriptive statistics, such as the mean or median, from multiple independent runs on a benchmark problem, DSC is based on the whole distribution of the independent runs, with the goal to ensure a more robust statistical analysis, as the mean can be affected by outliers (i.e., poor runs) and the medians can be in some small ϵ-neighbourhood of the performance space, which is practically insignificant.

3 METHODOLOGY

Figure 1 gives an overview of the proposed methodology. Given a set of benchmarks, the first step involves generating a vectorized representation for each problem instance using ELA features.

The heuristic which performs instance selection using clustering then proceeds to cluster the representations of all problem instances and perform sampling to identify a set of instances which will uniformly cover the problem landscape. On the other hand, the heuristics which perform instance selection using graph theory algorithms construct a graph based on the similarity of the problem instances, and identify a subset of instances on which the algorithms should be compared, by applying the algorithms for finding DS and MIS in the graph. Once the instances are selected by each heuristic, a statistical algorithm performance analysis is conducted by comparing the algorithms' performance on the instances selected by each heuristic independently. We further show that the selected instances from each heuristic enable a consistent and robust statistical analysis, as opposed to using a single benchmark.

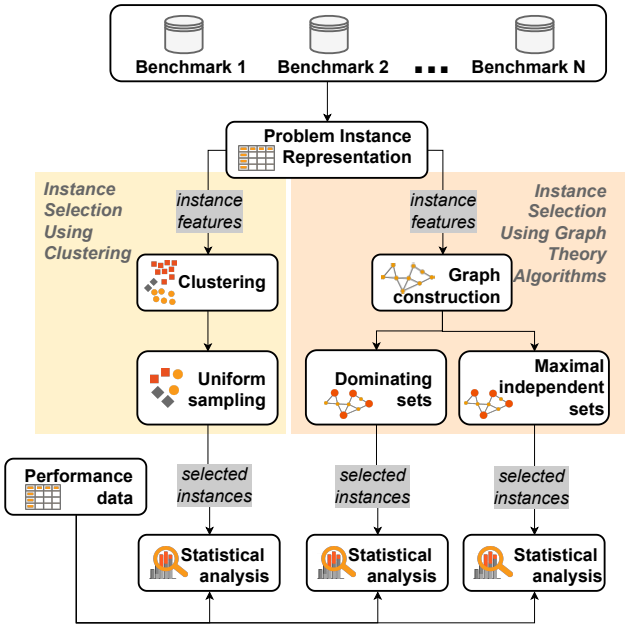


Figure 1: Overview of the proposed methodology.

3.1 Problem Instance Representation

We extract the ELA features using flacco's implementation of the improved Latin Hypercube sampling technique [2]. We use a sample size of $N = 800 \times D = 8000$, because it was shown in [19] that this sample size enables robust generation/calculation of ELA features.

As mentioned in Section 2.1, *flacco* provides a number of ELA feature sets. In this study, a total of **64 features** are included: 16 dispersion measures [23], three y-distribution measures [24], 18 level-set measures [24], nine meta-model measures [24], five information content measures [26], five nearest better clustering measures [16], and eight principal component analysis measures [17]. These ELA features have been selected, as they do not require additional sampling beyond the initial design. Out of these features, the *ic.eps.s* feature is removed, as it contains null values for some of the instances. The features are calculated 30 times, and each problem instance is represented by a 63-dimensional numerical vector, obtained as the median of the features obtained in the 30 runs of the feature calculation.

3.2 Instance Selection Using Clustering

Hierarchical agglomerative clustering is used to partition the entire set of problem instances. The problem instances are represented using the previously described ELA features, and cosine similarity of the ELA features is used to calculate the similarity of two problem instances. Cosine similarity is used as it is dependent only on the direction of vectors and not their length, rendering them less sensitive to outliers compared to other similarity measures [32].

We selected the hierarchical clustering, since in k-means clustering one starts with a random choice of clusters, and the results produced by running the algorithm many times may differ. Hierarchical clustering thus ensures better reproducibility of our results.

To estimate the number of clusters we used the silhouette score. In addition, when determining the number of clusters and the number of instances that will be sampled from each cluster, one has to make sure that the total number of sampled instances will be enough to perform the statistical analysis.

To construct an unbiased benchmark set of problem instances that will provide a more uniform coverage of the feature space, we sample an equal number of problem instances from each cluster to make each of the landscape they cover represented with the same number of instances. To see if the sampling ensures reproducible statistical outcome, we should repeat the sampling several times and perform the statistical analysis.

3.3 Instance Selection Using Graph Theory Algorithms

In order to select diverse instances from all of the instances in the benchmarks, we explore two graph theory algorithms, which involve finding the dominating and independent sets of nodes in a graph, respectively.

3.3.1 Definition of Dominating and Maximal Independent Sets. Given an undirected, acyclic, finite graph G with no multiple edges, represented with a set of vertices V and a set of edges E, one can define a dominating set (DS), D, as a subset of the nodes in G, such that every node not in D is adjacent to at least one member of D [1]. Formally, a dominating set D is a subset of V such that for all $v \in V - D$ its neighborhood set $N(v)$ has non-empty intersection with D, i.e., $N(v) \cap D \neq \emptyset$.

On the other hand, an independent set I is a set of nodes such that the subgraph of G induced by these nodes contains no edges, i.e., none of the nodes in I are adjacent to each other in G, or formally, $\forall u, v \in I, N(v) \cap \{u\} = \emptyset$. A maximal independent set (MIS) is then defined as an independent set such that no node can be added to it without violating independence [14].

3.3.2 Application of Dominating and Maximal Independent Sets Algorithms for Benchmark Sampling. The first step in our proposed methodology based on graph theory involves the construction of a homogeneous graph where the nodes represent problem instances, and an edge between two nodes indicates that they are similar according to some similarity measure.

We define a pair of problem instances to be similar if the cosine similarity of the ELA features used to represent the two instances exceeds a certain threshold. For this purpose, we evaluate several values for the similarity threshold.

When the graph is constructed in such a manner, the application of the DS and MIS algorithms produces a subset of diverse problem instances from the considered benchmarks. Both algorithms ensure that instances which are not similar to any of the other instances (nodes with no neighbours in the graph) will be part of the extracted set. The DS algorithm selects instances so that all the instances which are not selected, are similar to at least one of the selected instances, while the MIS algorithm selects instances which are not similar to each other.

For our experiments, we use the implementation of the DS and MIS algorithms provided by the *networkx* python library [10]. Since these algorithms are stochastic in nature (in particular due to the

random selection of the initial node set), we repeat the experiment 30 times with different random seeds, and each algorithm produces 30 different sets of problem instances.

3.4 Statistical Analysis

The sets of problem instances produced with both the clustering and the graph theory approaches are evaluated in terms of the robustness and consistency of the statistical algorithm performance analysis which can be executed when the algorithms are compared on the performance they achieve on these instances.

The Deep Statistical Comparison (DSC) raking scheme [7] is used to rank the algorithms on each problem instance selected in the benchmark suite. The rankings determined by DSC are analyzed with the Friedman non-parametric statistical test. The null hypothesis of the Friedman test states that all of the investigated algorithms achieve equivalent performance.

If the null hypothesis of the Friedman test is rejected, i.e. the algorithm performances are not equivalent, the Nemenyi post-hoc test [28] is used to identify the algorithm pairs which provide statistically different performance (i.e., multi-hypothesis correction scenario performed for each selected benchmark suite) [6]. To determine the robustness of the evaluation done on the benchmarks produced in each of the 30 different executions of the sampling process in the clustering approach and the 30 different executions of the DS and MIS algorithms, we keep track of the number of times (out of 30) that the Nemenyi test indicates that each algorithm pair produces an equivalent performance. The counting approach is only an indicator if the statistical results are robust if we repeat it using different problem instances selected by the proposed heuristics.

The Nemenyi test provides a p-value for each pair of algorithms. If the p-value is greater or equal than the significance level, the null hypothesis is not rejected, and (i.e., we translate this to 1, no statistical significance between the performance of the algorithms), otherwise there is a statistical significance between the performance of the compared pair of algorithms (i.e., we translate this to 0, a statistical significance is detected). A significance level of 0.05 is used in the experimental procedure.

We use the implementation of the Friedman test provided by the *scmamp* R package [3] (version 0.2.55) and the Nemenyi post-hoc test from the *PMCMR* R package [30] (version 4.3).

4 RESULTS AND DISCUSSION

This section presents the data used to evaluate the proposed approach, followed by the statistical results obtained when selecting a representative benchmark suite.

4.1 Data

The benchmark suites used in this study are a subset of those utilized in [19]. Specifically, the BBOB [11, 12] and CEC [20–22, 39] benchmark suites from the 2013, 2014, 2015, and 2017 single-objective competitions are used. The BBOB benchmark suite defines 24 base problems, from which different instances can be generated by applying transformations, such as rotation and scaling. The first five instances of the BBOB functions are used, resulting in 120 benchmark problems. The CEC2013 benchmark suite contains 28 problems, CEC2014 contains 30 problems, CEC2015 contains 15 problems, and CEC2017 contains 29 problems. All CEC problems are included only with a single instance. Therefore, a total of 222 benchmark problems are investigated. For the experimental procedure, the dimensionality of the decision variable space is set to 10.

To illustrate the rankings of optimization algorithms on the benchmark problems, an algorithm portfolio of three single-objective optimization algorithms is selected: the Covariance Matrix Adaption Evolutionary Strategy (CMA-ES) [13], RealSpace Particle Swarm Optimization (RSPSO) [15], and Differential Evolution (DE) [37]. This algorithm portfolio was chosen somewhat arbitrarily, as the key interest in this study is in presenting the general landscape-aware instance selection pipeline itself, and in analyzing whether it can provide reproducible statistical outcomes rather than an in-depth study of any particular algorithm portfolio. For validation of the results, we have repeated the experiments for four additional algorithm portfolios of three randomly selected algorithms (results/tables are available in our GitHub repository).

The *Nevergrad* [31] library is used for its implementations of the optimization algorithms with the default hyperparameter configurations provided in the library. All algorithms are run for a fixed budget of 100,000 function evaluations. An algorithm run is terminated either if the function evaluation budget is exhausted, or if the algorithm reaches within $\epsilon = 10^{-8}$ of the function's known global minimum. Due to the stochastic nature of the investigated algorithms, each algorithm is run for 30 independent runs on each problem instance.

4.2 Benchmarking using already established benchmark suites

The best practices of comparing the performance of a newly developed algorithm involve using an already established benchmark suite. However, one of the issues is which benchmark suite to be involved, since some of them can be in favour of the newly developed algorithm. To show the difference between different benchmark suites, we compare the three algorithms separately on each benchmark suite. The results from the comparison, after the Nemenyi post-hoc test, are presented in Table 1. We report the raw p-values, along with a binary indicator of statistical significance, which has a value of 1 when no statistical significance between the difference in performance of the algorithms has been found, and a value of 0, when a statistical significance is detected.

Using the obtained statistical outcomes across the benchmark suites, it is obvious that different statistical outcomes are produced.

On the BBOB benchmark suite, there is no statistical significance between the performance of the CMA and DE, however their performances differ statistically significantly from the performance of RSPSO. When applying the mean DSC ranking on the BBOB and CEC2017 benchmarks, the results suggest that RSPSO is better than CMA and DE, while no statistical significance is found between the performance of CMA and DE.

On the CEC2013 and CEC2014 benchmark suites, there is no statistical significance between the performance of any pair of algorithms. The results obtained on CEC2015 benchmark suite indicate that there is a statistically significant difference between the performance of the pairs of algorithms (DE, RSPSO) and (DE,

Table 1: Statistical comparison of the three algorithms on already established benchmark suites.

	BBOB		CEC 2013		CEC 2014		CEC 2015		CEC 2017		All	
	DE	RSPSO	DE	RSPSO	DE	RSPSO	DE	RSPSO	DE	RSPSO	DE	RSPSO
RSPSO	0.00/0		0.95/1		0.29/1		0.02/0		0.00/0		0.00/0	
CMA	0.33/1	0.00/0	0.47/1	0.65/1	0.75/1	0.07/1	0.04/0	0.98/1	0.97/1	0.00/0	0.06/1	0.00/0

CMA), while there is no statistical significance between CMA and RSPSO.

Using all problem instances from all benchmark suites, the RSPSO is the superior algorithm, and there is no statistical difference between the performances of CMA and DE.

Looking at the results, the crucial question is what one wants to show as the outcome of the statistical comparison. One option is to show that DE is superior and for this purpose we could select CEC2015 benchmark suite. Another option is to show that all algorithms have hardly distinguishable performance, by selecting the CEC2013 or the CEC2014 suite. A third option is to select BBOB or CEC2017 and show that RSPSO is the superior one. This kind of manipulation with the results allows publishing biased statistical results or presenting results based on the researchers' preference.

To overcome this issue, we propose three heuristics for selecting problem instances that provide reproducible statistical outcomes.

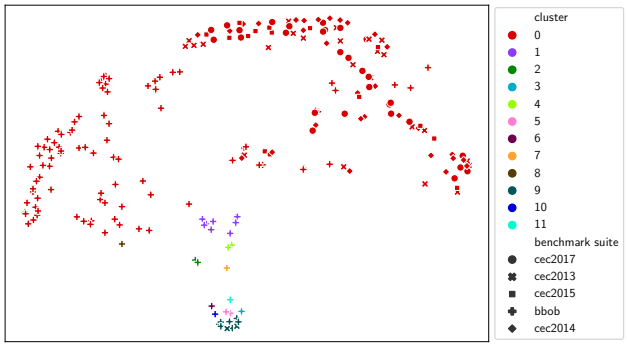
4.3 Instance Selection Using Clustering

To cluster the problem instances, agglomerate hierarchical clustering was applied using the *scikit-learn* python library [29]. We estimate the number of clusters to be 12, based on the silhouette score and the fact that a minimum of 10 samples are needed to execute the statistical analysis.

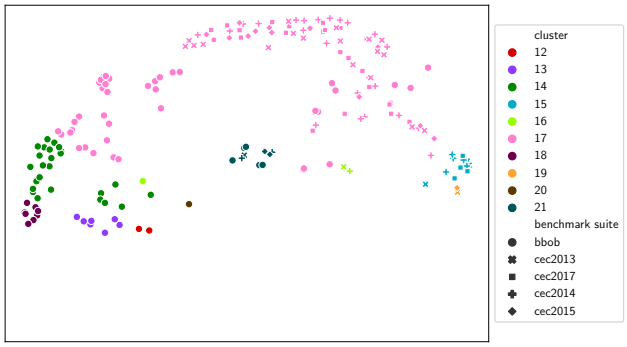
Figure 2a visualizes these 12 clusters, as computed by the agglomerate hierarchical clustering approach. We easily spot that the number of problem instances per cluster is not balanced: There are six clusters that consist of a single instance, while the largest cluster contains 190 instances. We therefore took the instances that belong to the largest cluster and we further clustered them to find subclusters. Looking at the silhouette scores obtained when the large cluster is broken up, we ended up with 10 subclusters, leaving us with **a total number of 21 clusters.** These clusters are visualized in Figure 2b and Figure 3 features the number of instances from each benchmark suite in each cluster.

As can be seen from Figure 3, the BBOB benchmark covers 19 out of the 21 clusters with at least one instance. On the other hand, the CEC2013, CEC2014, CEC2015 and CEC2017 benchmarks cover 6, 4, 3, and 2 clusters, respectively, and are mostly distributed in the subclusters obtained with the second clustering. This suggests that the BBOB benchmark covers more of the landscape, however it should be combined with some of the CEC benchmark suites to cover a larger portion of the problem landscape.

From Figure 3, we can see that there are six clusters with a single problem instance, indicating that we need to sample one instance from each cluster. The question that arises here is which instance should be selected from the larger clusters. If we select the instance randomly, it can be closer to some of the other clusters and will not represent the cluster structure. For this purpose, we need to find the problem instance that is closest to the centroid of each cluster.



(a) Visualization of the initial 12 clusters



(b) Visualization of the subclusters obtained by re-clustering the largest of the 12 clusters

Figure 2: Visualization of the clustering, obtained by projecting the ELA instance features into 2 dimensions using t-distributed stochastic neighbor embedding with cosine similarity. Different colors indicate different clusters, while different shapes indicate different benchmarks.

To calculate the centroids, we use all instances that belong to each cluster and calculate the mean value of each ELA feature. Next, using cosine similarity we find the closest problem instance to the centroid of each cluster. These problem instances are selected as representatives and further involved in the benchmark suite.

The results of comparing the algorithms using the Friedman and Nemenyi post-hoc tests on the selected problem instances are presented on the left of Table 2, where we can see that only RSPSO and CMA have a statistically significant performance difference.

We repeated the experiment when the largest cluster is split into 15 clusters instead of 10, since the silhouette score is similar. In this case, we ended up with 26 clusters. The results of this experiment

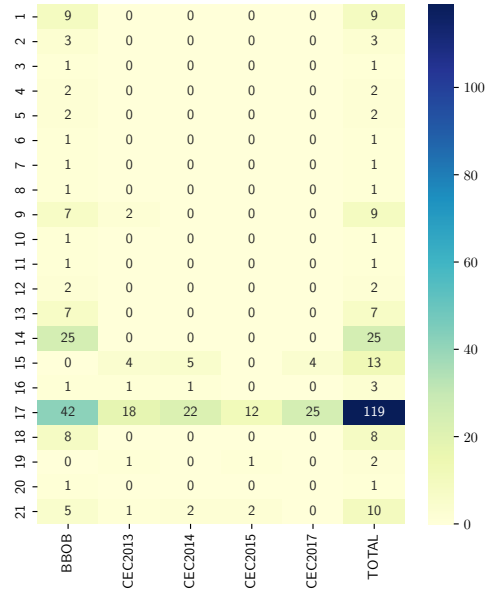


Figure 3: Number of instances from each benchmark suite, and total number of instances in each cluster.

Table 2: Results of the Friedman test and the Nemenyi post-hoc test for the statistical comparison of the three algorithms using the benchmark suites selected from the 21 and the 26 clusters, respectively.

	21 clusters		26 clusters	
	DE	RSPSO	DE	RSPSO
RSPSO	0.24/1		0.28/1	
CMA	0.48/1	0.02/0	0.51/1	0.02/0

are reported on the right part of Table 2. They are consistent with those of the 21 clusters, i.e., statistical significance is reported only between the algorithms RSPSO and CMA.

Furthermore, we have also tested the statistical outcome when median values instead of mean values of the ELA features are used to select the centroids, but the same statistical outcome has been achieved as in the case with the mean values.

To explore the sensitivity and flexibility of selecting the representatives for each cluster, instead of selecting the instance that is the closest to the cluster centroid, we can set a percentage of the instances closest to the cluster centroid that can be used as a representatives for the larger clusters. Furthermore, to create a benchmark suite, we should uniformly select one instance form the representatives per each cluster. To show if this leads to reproducible results, we repeat the selection several times, 15 (not 30 since we can not produce 30 different benchmark suites) and see if involving different representatives for the same cluster changes the statistical outcome. For this purpose, we have tested the following percentages: 12.5%, and 25%. The statistical outcomes are presented in Table 3, where the results show that reproducible statistical outcomes are obtained no matter which percentage is selected to define the representative instances for the larger clusters. Even more, the results are statistically the same as the results obtained when the closest instance to the centroid from each cluster is selected and involved in the benchmark suite.

Table 3: Results of the Friedman test and the Nemenyi post-hoc test for the statistical comparison of the three algorithms using the benchmark suites selected by using different percentage of representatives for the larger clusters.

	12.5% repres.		25% repres.	
	DE	RSPSO	DE	RSPSO
RSPSO	15.00		15.00	
CMA	14.00	0.00	14.00	0.00

4.4 Instance Selection Using Graph Theory Algorithms

Table 4 displays the number of edges in the graphs produced with each similarity threshold, as well as the minimum, maximum and median of the number of problem instances produced by the DS and MIS algorithms in the 30 independent runs, respectively. As evident from the table, increasing the threshold results in a reduction of the number of edges in the graph, and an increase in the number of instances produced by the DS and MIS algorithms. Since the algorithms produce a maximum of 4 and 8 instances when run on the graphs constructed with a similarity threshold of 0.50 and 0.70, respectively, we do not consider these results in the statistical analysis, since they do not provide enough samples for the safe execution of the Friedman test.

Figure 4 depicts the graph of instances constructed with a minimum similarity threshold of 0.9, meaning that each of the connected nodes have ELA features with a cosine similarity of at least 0.9. The orange nodes are the instances selected only by the DS algorithm, the green nodes are the instances selected by the MIS algorithm, while the purple nodes are the instances that were selected by both the MIS and the DS algorithm. The graph contains 9 connected components, with the 3 largest components containing 201, 11 and 3 nodes, respectively. There are also 6 zero-degree nodes (nodes

Table 4: Descriptive statistics of the benchmark suites selected using the DS and MIS algorithms.

similarity threshold	edge count	algorithm	min	max	mean
0.50	20821	DS	2	4	3.30
		MIS	3	4	3.37
0.70	19940	DS	5	8	7.07
		MIS	5	8	6.90
0.90	19119	DS	11	12	11.43
		MIS	11	13	11.47
0.95	17460	DS	16	18	17.37
		MIS	16	19	17.27
0.97	15116	DS	20	24	22.13
		MIS	21	24	22.63

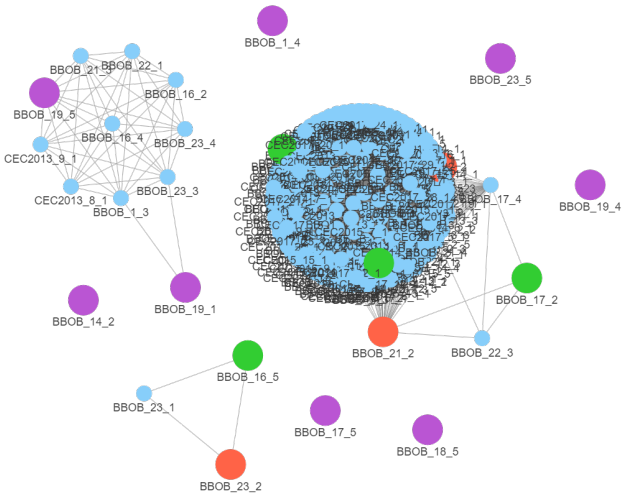


Figure 4: Visualization of the instances selected by the DS and MIS algorithms on the graph constructed with a minimum similarity threshold of 0.9.

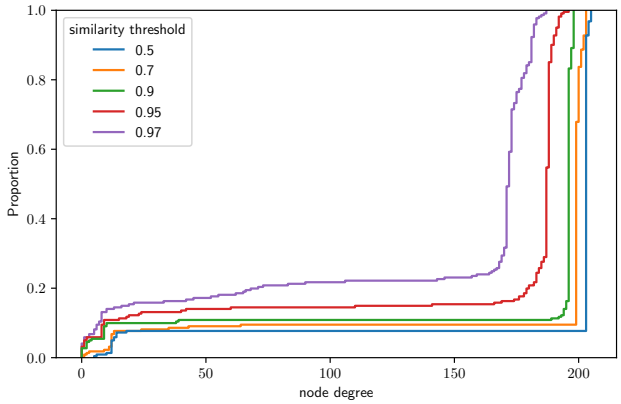


Figure 5: Empirical cumulative distribution plot of the node degrees for the graphs constructed with each similarity threshold, indicating the proportion of nodes having a node degree falling below each value on the x-axis.

Table 5: Results of the Friedman test and the Nemenyi post-hoc test for the statistical comparison of the three algorithms using the benchmark suites selected by the MIS and DS graph algorithms, for different cosine similarity measures. The numbers indicate the number of times in which no statistical significance was identified between the performance of a pair of algorithms, out of 30 independent executions of the statistical analysis, on 30 different subsets of instances produced by 30 runs of the algorithms.

	DS 0.9		DS 0.95		DS 0.97	
	DE	RSPSO	DE	RSPSO	DE	RSPSO
RSPSO	30.00		30.00		30.00	
CMA	27.00	5.00	26.00	3.00	22.00	0.00
	MIS 0.9		MIS 0.95		MIS 0.97	
	DE	RSPSO	DE	RSPSO	DE	RSPSO
RSPSO	30.00		30.00		30.00	
CMA	27.00	3.00	30.00	0.00	24.00	0.00

without have any edges) that form an individual connected component. As can be seen from Figure 4, all of these nodes represent instances from the BBOB benchmark. We can note that all of the zero-degree nodes are selected by both algorithms.

These solutions are based on only one run of the algorithm. The algorithms can produce different results depending on the initially selected nodes. For instance, if we take a look at the connected component of size 3, depicted in the lower left part of Figure 4, in this run of the algorithms, the MIS algorithm selected the instance BBOB_16_5. In another run, the algorithm could select either one of the other two instances in the component, BBOB_23_1 or BBOB_23_2, however, it will ensure that two of these instances will never be selected together, which will prevent redundancy and encourage diversity in the benchmark.

Figure 5 features the distribution of the node degrees in the graphs constructed with each of the thresholds. As can be seen, a large portion of the nodes have degrees in the range (170,200). This indicates that the problem instances are highly similar to each other, and further explains the large cluster obtained with the clustering, and the large connected component obtained in the graphs.

Next, we compare the algorithms using all benchmark suites selected for each combination of a graph heuristic selection (DS or MIS algorithm) and each similarity thresholds 0.90, 0.95, and 0.97. Table 5 presents the results obtained for comparing the algorithms on benchmark suites selected by the DS and MIS algorithms.

We need to point out that in this experiment we do not report the p-values, because we would like to trace the information if the same statistical outcome is achieved if we repeat the comparison on different benchmark suites selected by the same graph heuristic, as in a bootstrapping evaluation process.

The results in Table 5 show that the statistical outcome is reproducible when the benchmark suites are selected by the graph heuristics. For example, when the comparison is performed using problem instances selected with the DS algorithm and a similarity threshold of 0.90 used for generating the graph, in 30 out of 30 independent comparisons (i.e., comparing the algorithms on 30 benchmark suites selected by the DS algorithm with different random seeds) there is no statistical significance between the performance of DE and RSPSO. The same holds for the pair (DE, CMA), since in 27 out of 30 selected benchmark suites the same statistical outcome was achieved. Finally, the statistical outcome of the comparison between CMA and RSPSO is also robust since it shows that in 30-5=25 out of 30 comparisons a statistical significance between their performance was found (resulting in 0 in our counting process, so the end results presented in the table is 5).

From Table 5 we can conclude that both graph heuristics generate a selection of representative benchmark suites that provide reproducible statistical outcomes from comparison studies. We need to point out that in the proposed approach the statistical analysis

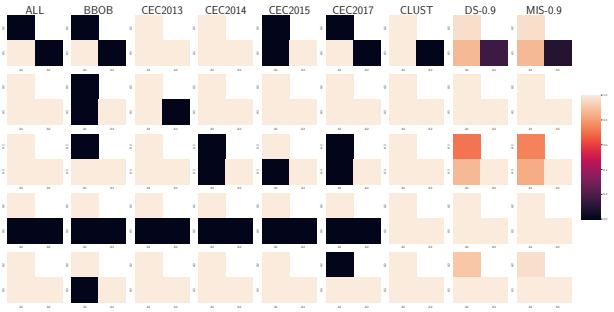


Figure 6: Visualization of the statistical outcomes of the Friedman and Nemenyi tests from the analysis on all five algorithm portfolios. Each row represents a different algorithm portfolio. The columns indicate the set of instances used for comparing the algorithms. Experiments were conducted on all benchmarks (column 1), on each benchmark separately (columns 2-6), on the instances selected by the clustering using 21 clusters (column 7) and the instances selected by the DS and MIS algorithms with a similarity threshold of 0.9 (columns 8-9). The analysis with the instances obtained with the clustering is performed 15 times, while the analysis with the instances from the DS and MIS algorithms are performed 30 times. The binary outcomes from individual runs are summed and normalized in the range [0,1] to enable a comparison with the rest of the approaches for which the analysis is executed once. Lighter values indicate that no statistical significance has been found between the difference in performance of the algorithms, while darker values indicate that the algorithms are statistically different.

should be done several times in order to guarantee the robustness and reproducibility of the statistical outcomes.

4.5 Generalization of the proposed heuristics

In this paper, we demonstrated the applicability of the proposed heuristics with the statistical analysis of the DE, CMA and RSPSO algorithms. To prove the generalization of the proposed heuristics, we conducted the same analysis for four additional portfolios, each containing three arbitrarily chosen optimization algorithms. We used the same approaches for selecting the problem instances independently from the algorithm portfolio involved in the comparison. Further, we compared each algorithm portfolio and for each one we generated all tables that are presented for the portfolio presented in the paper. Figure 6 summarizes the results for all five algorithm portfolios by depicting the statistical outcomes of the pairwise comparisons of the algorithms. The full results can be found in our GitHub repository. For the second portfolio, for which we do not report all of the tables in the paper, we can see that the outcomes obtained on BBOB and CEC2013 are different from the rest. For the third portfolio and fifth portfolio, the evaluations done on a single benchmark produce conflicting results. In all portfolios, the evaluation done using the subsets of instances produced by the proposed heuristics produce consistent and reliable results.

5 CONCLUSION

In this paper, we propose three approaches for selecting benchmark subsets which uniformly cover the problem landscape and can provide a reliable and robust statistical analysis of algorithm performance. We show that when the statistical analysis is conducted on a single benchmark set, one can obtain different outcomes regarding the superiority of one algorithm over another, when a different already established benchmark set is used. The benchmark subsets produced by the three proposed approaches provide a consistent and reliable evaluation.

Our study uses the landscape features computed in [19]. This work includes 118 miscellaneous functions, which we did not include in our study because no performance data was available for them. An extension of our work to include these functions is a straightforward next step, as is the coverage of a larger algorithm portfolio, and testing the sensitivity of the methodology when different similarity measures will be applied instead of cosine similarity and transformation of the ELA representations [8].

The existing optimization benchmarks are developed to understand algorithms' strengths and weaknesses, and not to achieve results that show better transferability to other problems using machine learning (ML) approaches [18, 43]. The proposed approaches can also be used to select problem instances for training ML models with better transferability to other problems in automated performance prediction, selection, or configuration.

ACKNOWLEDGMENTS

This work is based on the research supported by Slovenian Research Agency: research core fundings No. P2-0098 and project No. N2-0239 to TE, and scholarship by the Ad Futura grant for postgraduate study to GC.
This work is also based on the research supported in part by the National Research Foundation (NRF) of South Africa. The findings in this work is that of the authors alone, and the NRF accepts no liability whatsoever in this regard.
Carola Doerr acknowledges financial support from the Paris Ile-de-France Region region.

The authors acknowledge the Centre for High Performance Computing (CHPC), South Africa, for providing computational resources to this research project.

REFERENCES

[1] Robert B. Allan and Renu Laskar. 1978. On domination and independent domination numbers of a graph. *Discrete Mathematics* 23, 2 (1978), 73–76. https://doi.org/10.1016/0012-365X(78)90105-X
[2] B. Beachkofski and R. Grandhi. 2002. Improved Distributed Hypercube Sampling. In *43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2002-1274
[3] Borja Calvo and Guzman Santafe. 2015. scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems. *The R Journal* Accepted for publication (2015).
[4] Gavin C. Cawley and Nicola L. C. Talbot. 2010. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research* 11, 70 (2010), 2079–2107. http://jmlr.org/papers/v11/cawley10a.html
[5] Lee A Christie, Alexander EI Brownlee, and John R Woodward. 2018. Investigating benchmark correlations when comparing algorithms with parameter tuning. In *Proc. of the Genetic and Evolutionary Computation Conference Companion*. 209–210.

[6] Janez Demšar. 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. 7 (dec 2006), 1–30.

[7] Tome Eftimov, Peter Korošec, and Barbara Koroušić Seljak. 2017. A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics. *Information Sciences* 417 (2017), 186–215.

[8] Tome Eftimov, Gorjan Popovski, Quentin Renau, Peter Korošec, and Carola Doerr. 2020. Linear matrix factorization embeddings for single-objective optimization landscapes. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 775–782.

[9] Salvador García, Daniel Molina, Manuel Lozano, and Francisco Herrera. 2009. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. *Journal of Heuristics* 15, 6 (2009), 617–644.

[10] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. (2008), 11 – 15.

[11] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2020. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software* (2020), 1–31.

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

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

[14] Peter Jeavons, Alex D. Scott, and Lei Xu. 2016. Feedback from nature: simple randomised distributed algorithms for maximal independent set selection and greedy colouring. *Distributed Computing* 29 (2016), 377–393.

[15] J. Kennedy and R. Eberhart. 1995. Particle swarm optimization. In *Proc. of ICNN'95 - International Conference on Neural Networks*, Vol. 4. 1942–1948 vol.4. https://doi.org/10.1109/ICNN.1995.488968

[16] P. Kerschke, M. Preuss, S. Wessing, and H. Trautmann. 2015. Detecting Funnel Structures by Means of Exploratory Landscape Analysis. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*. ACM, 265–272. https://doi.org/10.1145/2739480.2754642

[17] Pascal Kerschke and Heike Trautmann. 2019. Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-package flacco. In *Applications in Statistical Computing – From Music Data Analysis to Industrial Quality Improvement*, Nadja Bauer, Katja Ickstadt, Karsten Lübke, Gero Szepannek, Heike Trautmann, and Maurizio Vichi (Eds.). Springer, 93 – 123. https://doi.org/10.1007/978-3-030-25147-5_7

[18] Benjamin Lacroix and John McCall. 2019. Limitations of Benchmark Sets and Landscape Features for Algorithm Selection and Performance Prediction. In *Proc. of Genetic and Evolutionary Computation (GECCO'19, Companion)*. ACM, 261–262. https://doi.org/10.1145/3319619.3322051

[19] Ryan Dieter Lang and Andries Petrus Engelbrecht. 2021. An Exploratory Landscape Analysis-Based Benchmark Suite. *Algorithms* 14, 3 (2021). https://doi.org/10.3390/a14030078

[20] J.J. Liang, B. Qu, and Ponnuthurai Suganthan. 2013. 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.* (12 2013).

[21] J.J. Liang, B.Y. Qu, P.N. Suganthan, and Q. Chen. 2014. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.* (2014).

[22] J.J. Liang, B. Qu, Ponnuthurai Suganthan, and Alfredo Hernández-Díaz. 2013. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.* (01 2013).

[23] M. Lunacek and D. Whitley. 2006. The dispersion metric and the CMA evolution strategy. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'06)*. ACM, 477. https://doi.org/10.1145/1143997.1144085

[24] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'11)*. ACM, 829–836. https://doi.org/10.1145/2001576.2001690

[25] Laurent Meunier, Herilalaina Rakotoarison, Pak Kan Wong, Baptiste Roziere, Jeremy Rapin, Olivier Teytaud, Antoine Moreau, and Carola Doerr. 2021. Black-box optimization revisited: Improving algorithm selection wizards through massive benchmarking. *IEEE Transactions on Evolutionary Computation* (2021).

[26] M.A. Muñoz, M. Kirley, and S.K. Halgamuge. 2015. Exploratory Landscape Analysis of Continuous Space Optimization Problems Using Information Content. *IEEE Transactions on Evolutionary Computation* 19, 1 (2015), 74–87. https://doi.org/10.1109/TEVC.2014.2302006

[27] Mario A. Muñoz and Kate Smith-Miles. 2020. Generating New Space-Filling Test Instances for Continuous Black-Box Optimization. *Evolutionary Computation* 28, 3 (09 2020), 379–404. https://doi.org/10.1162/evco_a_00262 arXiv:https://direct.mit.edu/evco/article-pdf/28/3/379/1858988/evco_a_00262.pdf

[28] P. Nemenyi. 1963. *Distribution-free Multiple Comparisons*. Princeton University. https://books.google.si/books?id=nhDMtgAACAAJ

[29] 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, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[30] Thorsten Pohlert. 2014. *The Pairwise Multiple Comparison of Mean Ranks Package (PMCMR)*. https://CRAN.R-project.org/package=PMCMR R package.

[31] J. Rapin and O. Teytaud. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.

[32] Ali Seyed Shirkhorshidi, Saeed Aghabozorgi, and Teh Ying Wah. 2015. A comparison study on similarity and dissimilarity measures in clustering continuous data. *PloS one* 10, 12 (2015), e0144059.

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

[34] Urban Škvorc, Tome Eftimov, and Peter Korošec. 2020. Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis. *Applied Soft Computing* 90 (2020), 106138.

[35] Urban Skvorc, Tome Eftimov, and Peter Korosec. 2021. A Complementarity Analysis of the COCO Benchmark Problems and Artificially Generated Problems. *CoRR* abs/2104.13060 (2021). arXiv:2104.13060 https://arxiv.org/abs/2104.13060

[36] Kate Smith-Miles, Davaatseren Baatar, Brendan Wreford, and Rhyd Lewis. 2014. Towards objective measures of algorithm performance across instance space. *Computers & Operations Research* 45 (2014), 12–24.

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

[38] Thomas Weise, Yan Chen, Xinlu Li, and Zhize Wu. 2020. Selecting a diverse set of benchmark instances from a tunable model problem for black-box discrete optimization algorithms. *Appl. Soft Comput.* 92 (2020), 106269. https://doi.org/10.1016/j.asoc.2020.106269

[39] Guohua Wu, Rammohan Mallipeddi, and Ponnuthurai Suganthan. 2016. Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.* (10 2016).

[40] Estefania Yap, Mario A. Munoz, Kate Smith-Miles, and Arnaud Liefooghe. 2020. Instance Space Analysis of Combinatorial Multi-objective Optimization Problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. https://doi.org/10.1109/cec48606.2020.9185664

[41] Bo Yuan and Marcus Gallagher. 2004. Statistical Racing Techniques for Improved Empirical Evaluation of Evolutionary Algorithms. In *Parallel Problem Solving from Nature - PPSN VIII*, Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 172–181.

[42] Yong-Wei Zhang and Saman K Halgamuge. 2019. Similarity of Continuous Optimization Problems from the Algorithm Performance Perspective. In *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2949–2957.

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