Swarm and Evolutionary Computation 100 (2026) 102246

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

A methodology for multi-label algorithm selection in constrained multiobjective optimization

Andrejaana Andova, Jordan N. Cork, Tea Tušar, Bogdan Filipič *

Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
Jožef Stefan International Postgraduate School, Jamova cesta 39, Ljubljana, Slovenia

ARTICLE INFO

Dataset link: https://github.com/andrejaana/algorithm_selection_CMO

Keywords:
Algorithm selection
Exploratory landscape analysis
Machine learning
Constrained multiobjective optimization

ABSTRACT

Algorithm selection in optimization is often done by considering a single best-performing algorithm per problem. However, sometimes multiple algorithms perform comparably well on the same optimization problem, and in such cases, it would be appropriate to consider all of them as best performing. Hence, this work proposes an algorithm selection methodology that enables the identification and prediction of multiple algorithms as best performing. More specifically, the methodology involves first identifying the best-performing algorithms using statistical tests that show when the algorithms perform comparably well. Then, these algorithms are set as targets to machine learning models that can predict multiple algorithms as best performing. Finally, an evaluation measure is introduced to assess the performance of the algorithm selection models. The proposed methodology is applied to constrained multiobjective optimization.

1. Introduction

When dealing with a new optimization problem, it is helpful to know which optimization algorithms are most likely to perform best in finding good solutions to the problem. In recent years, much research has been focused on analyzing how algorithms perform on various optimization problems [1,2], predicting the algorithm performance [3–5], and building machine learning models that predict the best-performing algorithm which is known as *algorithm selection*.

When selecting the best algorithm for an optimization problem, prior works [6–9] typically use a budget of solution evaluations for which they choose a single best-performing algorithm. As the best-performing algorithm, they identify the algorithm with the best average of performance indicator values obtained over a number of runs. Consequently, they disregard the distributions of performance values and thus cannot recognize when multiple algorithms perform comparably well.

When multiple algorithms perform similarly well, labeling just one as the best-performing one can mislead the selection model. The main reason is that the model focuses too narrowly on differentiating tiny performance gaps between the algorithms, which are negligible or statistically insignificant, and may not be consistent across different algorithm runs. Moreover, by training the model to consider these tiny performance gaps, the algorithm selection model risks overfitting and may fail to generalize to new problems.

Additionally, if we declare a single algorithm as the best-performing one across a collection of algorithm runs, and other algorithms have statistically comparable performance, a different algorithm might be identified as best-performing in another collection of runs. For example, after running SPEA [10] and NSDE [11] for 5,000 evaluations on the problem MW6 [12] ten sets of 31 independent runs, we found that SPEA2 was identified as the best-performing algorithm in six collections of runs, while NSDE was identified as the best-performing algorithm in four collections of runs. This inconsistency reveals that slight variations in the quality indicator values across a collection of runs could alter the identified best-performing algorithm, making the algorithm selection methodology unstable and non-reproducible. To overcome these shortcomings, the main contribution of this work is a multi-label algorithm selection methodology that allows several algorithms to be identified as best-performing, rather than forcing a single winner.

Furthermore, apart from their performance on a given optimization problem, algorithms can differ in the time complexity and memory usage of their operators. Moreover, the implementation of some algorithms may be more accessible than others, or a user may have a better understanding of the functioning of a specific algorithm. These are additional reasons for choosing one algorithm over another, which are not typically considered when building an algorithm selection model. Therefore, having an algorithm selection methodology that predicts multiple best-performing algorithms can allow the user to select from

* Corresponding author at: Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia.
E-mail address: bogdan.filipic@ijs.si (B. Filipič).

https://doi.org/10.1016/j.swevo.2025.102246
Received 11 July 2025; Received in revised form 11 November 2025; Accepted 2 December 2025
Available online 7 December 2025

Table 1
Summary of statistical testing methods as used in related work and their suitability for algorithm selection.

Work	Statistical method	Advantages	Limitations
[13]	Novel, deep statistical comparison method	Well-defined methodology	Identifies best algorithms over multiple problems
[14]	Robust ranking with multiple objectives	Well-defined methodology	Identifies best algorithms over multiple problems
[15]	Wilcoxon sum-rank test	Considers multiple algorithm runs	Excludes problems where multiple algorithms perform similarly
[16]	Rank-sum test	Considers other best-performing algorithms when evaluating performance	Predicts a single best-performing algorithm

the best-performing algorithms an algorithm that is preferred based on other criteria. These additional criteria would not in any way affect the outcome of the algorithm selection model. However, after the best-performing algorithms have been predicted, the user can use only one of them to solve a specific problem, and they can select that algorithm based on these criteria.

For this reason, when dealing with algorithm selection, it is crucial to use a methodology that allows multiple best-performing algorithms. These can be identified using statistical tests.

In recent years, several works have proposed methodologies for applying statistical testing in optimization. For example, Eftimov et al. [13] have proposed a methodology for statistically comparing algorithms on a set of problems. Later, a similar methodology was proposed that uses multiobjective optimization [14]. In both cases, the authors proposed statistical testing for a large collection of problems. In contrast, in algorithm selection, statistical testing needs to focus on identifying the best-performing algorithms for an individual optimization problem. Thus, the methodologies in [13,14] cannot be used.

A few related studies used statistical tests for algorithm selection. Nonetheless, these works did not propose a well-defined methodology for algorithm selection where multiple algorithms can be selected as best performing. For example, performing algorithm selection on a large collection of single-objective optimization problems, Škvorc et al. [15] excluded problems from their analysis for which statistical tests indicated that multiple algorithms performed comparably. Qiao et al. [16] used statistical tests to determine whether other algorithms perform comparably to the chosen best-performing one. Then, if the algorithm selection model predicts one of these other algorithms as the best-performing one on the test set, they consider the instance correctly classified. An overview of the statistical methods used in the related work is presented in Table 1.

In this work, we propose a methodology for algorithm selection that allows multiple algorithms to be selected as best-performing ones. This methodology involves statistical tests to identify the best-performing algorithms, machine learning models that can predict multiple best-performing algorithms, and an evaluation measure specifically designed for algorithm selection. While the approach demonstrates its usefulness in the context of Constrained Multiobjective Optimization (CMO), it also has limitations regarding its sensitivity to parameter settings, its evaluation scope, and its reliance on artificial benchmark problems, which are discussed in detail later in the paper.

The rest of this paper is organized as follows. Section 2 gives the background for this work. Section 3 introduces the proposed methodology for algorithm selection, and in Section 4 the experimental setup is presented. Section 5 presents and discusses the obtained results as well as the advantages and limitations of the methodology. Finally, in Section 6 we conclude the paper with a summary of the proposed methodology and the obtained results with it.

2. Background

2.1. Constrained multiobjective optimization

In a Constrained Multiobjective Optimization Problem (CMOP), the task is to optimize multiple objectives while satisfying given constraints [17]. It can be defined as:

$$\begin{aligned} &\text{minimize} \quad f_m(\mathbf{x}), \quad m = 1, \dots, M \\ &\text{subject to} \quad g_j(\mathbf{x}) \leq 0, \quad j = 1, \dots, J, \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \dots, x_D)$ is a D dimensional solution, $f_m(\mathbf{x})$ are the objective functions, and $g_j(\mathbf{x})$ are the constraint functions. M is the number of objectives, and J is the number of constraints.

An important concept in constrained optimizations is *constraint violation*, which is defined as $v_j = \max\{0, g_j(\mathbf{x})\}$ for a single constraint. For all constraints that apply to a solution, the *overall constraint violation* is defined as $v(\mathbf{x}) = \sum_{j=1}^{J} v_j(\mathbf{x})$. A solution is *feasible* if it satisfies all constraints ($v(\mathbf{x}) = 0$). A *Feasible Region* (FR) [18] is the set of all feasible solutions.

A feasible solution is said to dominate another feasible solution if $f_m(\mathbf{x}) \leq f_m(\mathbf{y})$ for all $1 \leq m \leq M$, and $f_m(\mathbf{x}) < f_m(\mathbf{y})$ for at least one $1 \leq m \leq M$. *Pareto optimal solutions* are the feasible solutions that no other feasible solution dominates. All nondominated feasible solutions constitute the *Pareto Set* (PS) of solutions. Its image in the objective space is called the *Pareto Front* (PF). If we disregard the constraints, all solutions are feasible and the optimal solutions form the *Unconstrained Pareto Front* (UPF) in the objective space.

The point in the objective space with the best values attainable for each objective across the Pareto front is the *ideal point*. Consequently, the *nadir point* is defined as the point composed of the worst objective values observed among the Pareto optimal solutions.

2.2. Exploratory landscape analysis for constrained multiobjective optimization

Exploratory Landscape Analysis (ELA) is a feature extraction methodology [19] developed to characterize optimization problems by calculating statistical features from a sample of solutions. Many ELA features were proposed for single-objective optimization, resulting in a package with their collection [20]. However, ELA features for unconstrained and constrained multiobjective optimization problems have been researched to a lesser degree [1,2,21]. This presents an additional difficulty in solving the algorithm selection task using ELA features for CMOPs.

This work uses ELA features for CMOPs collected by Alsouly et al. [1]. They are calculated from a sample of solutions and solutions produced from a random walk. The features can be divided into three groups:

- The *multiobjective landscape group* contains ELA features describing properties of a CMOP (most of all, the relations between the objectives) disregarding the constraints.

- The *violation landscape group* contains ELA features describing the problem constraints.
- The *multiobjective-violation landscape group* contains features describing the interactions between the objectives and constraints in a CMOP.

In total, there are 80 ELA features in the three groups.

2.3. Quality indicator for CMO

The most widely used quality indicator in multiobjective optimization is the hypervolume [22]. This quality indicator was proposed for unconstrained multiobjective optimization problems and thus can only deal with feasible solutions. To overcome this limitation, Vodopija et al. [23] proposed a quality indicator for CMOPs, I_{CMOP}. This quality indicator is an extension of a hypervolume-based quality indicator, $I_{\text{HV+}}$, which was initially proposed in [24].

First, $I_{\text{HV+}}$ is defined as follows:

1. When the solutions in a given set are dominated by the nadir point, $I_{\text{HV+}}$ takes the value of the minimal distance between the solutions and the region of interest bounded by the ideal and the nadir point.
2. When at least one of the solutions dominates the nadir point, $I_{\text{HV+}}$ takes the negative value of the hypervolume, with the nadir point as a reference point.

Next, I_{CMOP} extends $I_{\text{HV+}}$:

1. When all solutions in a given set are infeasible, I_{CMOP} takes the value of the smallest total constraint violation value of all solutions, with the addition of a threshold τ^*.
2. When at least one solution in the set is feasible, I_{CMOP} takes the value of $I_{\text{HV+}}$, upper bounded by a threshold τ^*, meaning it equals $\min\{I_{\text{HV+}}, \tau^*\}$.

Both $I_{\text{HV+}}$ and I_{CMOP} assume that lower indicator values signify better sets of solutions and vice versa.

3. Algorithm selection methodology

In this work, we propose a new algorithm selection methodology that allows multiple algorithms to be identified as best-performing ones. After collecting algorithm performance information on a number of problems, we use statistical tests to identify the best-performing algorithms for each problem. Then, multi-label classifiers are used to train models that predict all the best-performing algorithms for a problem. The performance of the multi-label classification models is evaluated using an evaluation measure we propose for algorithm selection, called Algorithm Selection Precision (ASP).

The proposed algorithm selection methodology will be applied in CMO. The steps for performing algorithm selection in CMO are visualized in Fig. 1, and presented in Algorithm 1. In the rest of this section, we will explain in more detail the best-performing algorithm identification method based on statistical tests, the multi-label classification method, and the ASP evaluation measure.

The proposed methodology was implemented in Python, and the code is available at https://github.com/andrejaana/algorithm_selection_CMO.

Algorithm 1 Algorithm selection in CMO

Require:
$\mathcal{A}$: Set of A algorithms
$\mathcal{P}$: Set of P problems
n: Evaluation budget per run
s: Sample size
Φ: Set of ELA features
k: Number of independent runs
Ensure: Average ASP across all problems
▷ Step 1
for each problem $p \in \mathcal{P}$ **do**
  for each algorithm $a \in \mathcal{A}$ **do**
    for $j \leftarrow 1$ **to** k **do**
      $X \leftarrow$ Run a on p for n evaluations
      $\mathcal{I}_{pa}[j] \leftarrow I_{\text{CMOP}}(X)$
    end for
  end for
end for
▷ Step 2
for each problem $p \in \mathcal{P}$ **do**
  $\mathcal{A}_p^* \leftarrow$ BestPerformingAlgorithms($\{\mathcal{I}_{pa}\}_{a\in\mathcal{A}}$)
end for
▷ Step 3
for each problem $p \in \mathcal{P}$ **do**
  $\mathcal{D}_p \leftarrow \emptyset$ ▷ Initialize ML instances
  for $j \leftarrow 1$ **to** s **do**
    $S_j \leftarrow$ SampleSolutions(p)
    $\mathcal{F}_{pj} \leftarrow [\phi(S_j) \forall \phi \in \Phi]$ ▷ ELA features
    $\mathcal{D}_p \leftarrow \mathcal{D}_p \cup \{(\mathcal{F}_{pj}, \mathcal{A}_p^*)\}$
  end for
end for
▷ Step 4
for each problem $p \in \mathcal{P}$ **do**
  $\mathcal{M} \leftarrow$ TrainClassifier($\mathcal{D}_{P\setminus\{p\}}$)
  Initialize $\hat{\mathcal{A}}_p^* \leftarrow$ vector(s)
  for $j \leftarrow 1$ **to** s **do**
    $\hat{\mathcal{A}}_p^*[j] \leftarrow \mathcal{M}(\mathcal{F}_{pj})$
  end for
  scores[p] $\leftarrow$ AS_Precision($\mathcal{A}_p^*, \hat{\mathcal{A}}_p^*$)
end for
return $\frac{1}{P}\sum_{p=1}^{P}$ scores[p]

3.1. Best-performing algorithms assessment

We apply statistical tests to determine which algorithms perform best on an optimization problem. The pseudocode for assessing the best-performing algorithms using statistical tests is given in Algorithm 2.

First, a best-performing algorithm is selected based on the median value of the performance indicator values over multiple runs after a targeted number of solution evaluations. Then, this algorithm is compared to the other algorithms using statistical tests. If there is no significant statistical difference between the best-performing and another algorithm, both algorithms are said to perform equally well.

However, the more statistical tests are performed, the higher the probability of committing a Type I error. Hence, to limit the number of statistical tests performed, we first check if the distributions of the performance indicator values of the best-performing and another algorithm over all runs overlap within one Median Absolute Deviation (MAD) [25] from their respective median values. Specifically, we consider that two distributions overlap if the lower bound (median minus one MAD) of one algorithm's values is less than or equal to the upper bound (median plus one MAD) of the other algorithm's values. If this holds, we add the algorithm to a list of probable best-performing algorithms.

Then, since we want to compare the means of two or more groups by analyzing the variance between groups, we apply ANOVA [26], a standard statistical test for such cases. We set the p-value to 0.005, and applied the test to all algorithms in the list of probable best-performing algorithms. This statistical test gives information on whether there is a significant difference between the algorithms in the list. If the ANOVA test indicates that there is no difference between the algorithms, all are selected as best-performing ones. Otherwise, if it indicates that there are differences between the algorithms, we need to proceed with a post-hoc test.

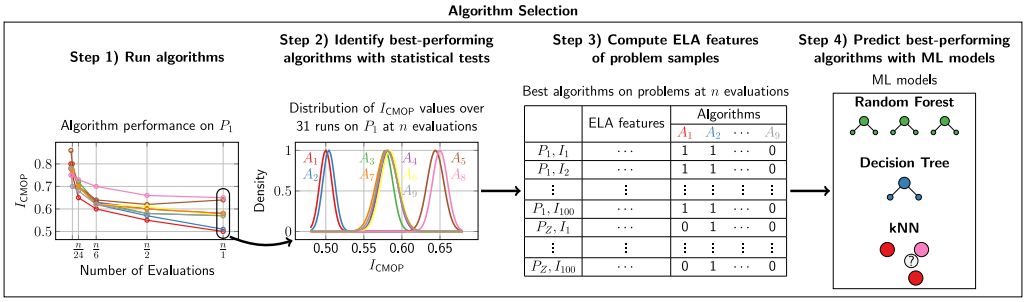


Fig. 1. Steps needed to perform algorithm selection in CMO: (1) Run the algorithms on 2D, 5D, and 10D CMOPs 31 times. (2) Statistically test the obtained performance indicator values to identify the best algorithms for different numbers of evaluations. (3) Extract the ELA features using 100 samples per problem, resulting in 100 machine learning instances per problem. (4) Use the ELA features as inputs and the identified best algorithms as targets to build multi-label classification models for algorithm selection.

We use Tukey's HSD post-hoc test [27] with a small $\alpha = 0.005$ to pairwise compare the best-performing algorithm with the algorithms from the list of probable best-performing algorithms. This post-hoc test is a standard approach for post-hoc analysis of the results obtained from the ANOVA test. Tukey's HSD test assesses whether the mean performance indicator values of two algorithms come from the same distribution. If the null hypothesis of Tukey's HSD post-hoc test is not rejected for a given algorithm pair, it suggests that there is no statistically significant difference in the performance indicator values of the two algorithms. Therefore, we can consider the two algorithms to perform comparably well.

In our experiments, we performed many statistical tests as we compared all algorithms across all problems and for a number of evaluation budgets. Since each additional test increases the likelihood of incorrectly rejecting the null hypothesis of the test, we decided to be strict and set the parameters p and α of the statistical tests lower than their usual values.

To statistically compare the algorithms' performance, we use the SciPy library [28] implementation of the ANOVA test, and the statsmodels library [29] implementation of Tukey's HSD test.

Algorithm 2 Identification of best-performing algorithms with statistical testing (BestPerformingAlgorithms)

Require:
$\mathcal{A}$: Set of candidate algorithms
$\{I_a\}_{a\in\mathcal{A}}$: Quality indicator values for 31 runs per algorithm
Ensure: $\mathcal{A}^*$: Subset of statistically equivalent best algorithms
$a^* \leftarrow$ algorithm $a \in \mathcal{A}$ with best median(I_a)
$\mathcal{A}_{\text{cand}} \leftarrow \{a \in \mathcal{A} \mid \text{median}(I_{a^*}) + \text{MAD}(I_{a^*}) \geq \text{median}(I_a) - \text{MAD}(I_a)\}$ ▷ Candidates with overlapping quality indicator values
$p \leftarrow \text{ANOVA}(\{I_a \mid a \in \mathcal{A}_{\text{cand}}\})$
if $p < 0.005$ **then**
  $\mathcal{R} \leftarrow \text{TukeyHSD}(\{I_a \mid a \in \mathcal{A}_{\text{cand}}\}, \alpha = 0.005)$
  $\mathcal{A}^* \leftarrow \{a \in \mathcal{A}_{\text{cand}} \mid \mathcal{R}(a, a^*) = \text{NS}\}$ ▷ NS = not significant
else
  $\mathcal{A}^* \leftarrow \{a \in \mathcal{A}_{\text{cand}}\}$ ▷ No significant differences found
end if
return $\mathcal{A}^*$

3.2. Multi-label classification

We use the ELA features described in Section 2.2 as input to train a machine learning model that predicts the best-performing algorithms. The output of the model is a set of identified best-performing algorithms.

Considering that several algorithms can perform equally well for a given optimization problem, when defining the machine learning task, it is crucial to allow multiple algorithms to be identified as best performing. This way, the machine learning models will be able to generalize better when predicting the best-performing algorithms, as they will treat all algorithms that perform similar equally and will not overfit to the small, insignificant differences between them.

The machine learning task that predicts multiple labels at the same time is called *multi-label classification*. The target of a multi-label classification task is a binary vector.

In algorithm selection, each algorithm is represented as a single element in the multi-label classification target. If an algorithm is identified as one of the best-performing algorithms for a given problem, the corresponding element in the binary vector is assigned a value of 1; otherwise, it is assigned a value of 0.

Solving a multi-label classification problem can be approached in two ways. The most straightforward approach is to train a separate binary classifier for each algorithm and then combine the output of all binary classifiers into a vector representing the predicted binary vector.

The second approach is to use classifier chains for multi-label classification [30]. Here, separate binary classifiers are trained for each algorithm considered. However, in this approach, the predictions of the previous classifiers in the chain are used as additional features to train the next classifier in the chain. Thus, the next classifier can exploit the correlations between the algorithms and learn the dependencies between them.

In the first approach, the models do not consider the relationship between algorithms, while in the second approach, they do. To identify one algorithm as best performing on a given problem, one needs to know how the other algorithms work on the same problem. The process of identifying best-performing algorithms in our algorithm selection methodology involves comparing the algorithms' performances. Therefore, adding a new algorithm to the set of algorithms considered might change the binary values in the target vector. Thus, if the newly added algorithm is identified as the single best-performing algorithm, the other algorithms should all be assigned a value of 0, even though they may have been identified as best-performing before the new algorithm was included.

This shows that the elements in the target binary vector in our machine learning task are strongly dependent on each other. Thus, we choose to utilize the classifier chain approach for multi-label classification, which considers the elements in the target to be interdependent. To further strengthen this decision, we show the correlations between the identified best-performing algorithms in Section 5.1.

As a basis for the classifier chains, we decided to use a kNN, Random Forest, and a Decision Tree classifier provided by the scikit-learn Python library [31]. The classifier chain implementations were also taken from the scikit-learn library. Additionally, for each algorithm, we included a *dummy classifier* from scikit-learn as a baseline. Each dummy classifier predicts the classes for its corresponding algorithm based on the class distribution observed in the training data (i. e., using stratified sampling).

3.3. The ASP evaluation measure

To effectively evaluate a machine learning model, it is essential to choose an appropriate evaluation measure that highlights specific qualities of interest in the model. In algorithm selection, it is crucial for the model to select at least one of the best-performing algorithms and not select an algorithm that does not perform well on a CMOP. To assess these qualities, we propose a measure for algorithm selection similar to the classical precision measure in machine learning.

The proposed measure is based on the so-called Instance Precision (IP) for each machine learning instance i, which is defined with the following formula:

$$\mathrm{IP}_i = \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i} \tag{2}$$

The TP_i (True Positives) are the cases where an algorithm is correctly predicted to be best performing in the machine learning instance i. In contrast, the FP_i (False Positives) are the cases where an algorithm is not the best performing in instance i, but is predicted to be such. This way, the measure punishes false best-performing predictions while promoting detection of at least one best-performing algorithm in the machine learning instance i (if TP_i equals zero, IP_i will also equal zero).

To calculate the value of the proposed ASP measure, we average the IP values over all machine learning instances in the test data:

$$\mathrm{ASP} = \sum_{j=1}^{S} \frac{\mathrm{IP}_i}{S} \tag{3}$$

where S denotes the total number of machine learning instances in the test data, and IP_i denotes the IP value for the ith machine learning instance.

Let us illustrate the calculation of this measure with an example. Let $A_p^* = \{A_1, A_2, A_7\}$ represent the identified best-performing algorithms for problem p. Then, let $\hat{\mathcal{A}}_p^*$ represent the predicted best-performing algorithms for four different machine learning instances, with their corresponding IP values calculated using Eq. (2):

$$\hat{\mathcal{A}}_p^* = \begin{cases} \{A_1, A_2, A_3\}, & \mathrm{IP}_1 = \dfrac{2}{2+1} = \dfrac{2}{3} \\ \{A_1, A_2\}, & \mathrm{IP}_2 = \dfrac{2}{2+0} = 1 \\ \{A_7, A_8\}, & \mathrm{IP}_3 = \dfrac{1}{1+1} = \dfrac{1}{2} \\ \{A_8\}, & \mathrm{IP}_4 = \dfrac{0}{0+1} = 0 \end{cases} \tag{4}$$

In the first machine learning instance, the model correctly predicted algorithms A_1 and A_2 as best-performing ($\mathrm{TP}_1 = 2$) and incorrectly predicted algorithm A_3 as best-performing ($\mathrm{FP}_1 = 1$). Then, following Equation (2), the IP value for this machine learning instance is equal to $\frac{2}{2+1} = \frac{2}{3}$. Iterating over all machine learning instances, we calculate the ASP value for problem p using Eq. (3):

$$\mathrm{ASP}_p = \frac{1}{4} \cdot \left(\frac{2}{3} + 1 + \frac{1}{2} + 0 \right) = \frac{1}{4} \cdot \frac{13}{6} = 0.542 \tag{5}$$

4. Experimental setup

We evaluate the algorithm selection methodology on 2-dimensional (2D), 5-dimensional (5D), and 10-dimensional (10D) bi-objective problems. In the experimental framework, we include seven widely-used benchmark suites: C-DTLZ [32], CF [33], CTP [34], DAS-CMOP [35], DC-DTLZ [36], MW [12], and NCTP [37]. Although the problems in these suites are scalable in the number of objectives and problem dimensionality, there are some limitations in how they can be instantiated. Thus, the total number of CMOPs per problem dimension differs. The number of available CMOPs per benchmark suite and search dimension space is shown in Table 2.

Table 2
The number of CMOPs per dimension.

Benchmark	2D	5D	10D
C-DTLZ	5	6	6
CF	0	7	7
CTP	8	8	8
DAS-CMOP	6	6	6
DC-DTLZ	6	6	6
MW	8	14	14
NCTP	0	18	18
Total	33	65	65

We used nine population multiobjective optimization algorithms: NSGA-II [38], NSGA-III [39], MOEA/D-IEpsilon [40], AGE [41], C-TAEA [36], SPEA2 [10], GDE3 [42], NSDE [11], NSDE-R [43]. For NSGA-III, C-TAEA, NSGA-II, AGE, and SPEA2, we use the implementations from the pymoo library [44], while for GDE3, NSDE, and NSDE-R, the implementations are taken from the pymoode library [45]. For MOEA/D-IEpsilon, we extended the implementation from the pymoo library to match that of the original paper.

The selected algorithms have diverse search and constraint handling techniques. NSGA-II and NSGA-III rely on non-dominated sorting, with NSGA-II using crowding distance for promoting diversity, while NSGA-III uses reference points for scalability to many objectives. Both employ the constraint-domination principle as a constraint handling technique, favoring feasible solutions and ranking infeasible ones by constraint violation. MOEA/D-IEpsilon is a decomposition-based algorithm that solves optimization subproblems in parallel using the epsilon constraint handling technique. AGE is an algorithm that focuses on improving the hypervolume metric and uses the constraint-domination principle to handle constraints. C-TAEA uses a two-archive strategy, maintaining one archive of solutions for convergence towards best feasible solutions, and another archive for diversity and constraint boundary exploration. SPEA2 uses a strength-based fitness assignment and density estimation to maintain a well-distributed external archive, also adopting the constraint-domination principle for constraint handling. GDE3, NSDE, and NSDE-R integrate differential evolution operators with multiobjective principles. They all apply the constraint-domination principle as a constraint handling technique.

For a fair comparison of the algorithms, we set the population size to 200 for each algorithm (100 · number of objectives). The remaining parameters are set to their default values, and are presented in Table 3. Then, we execute each algorithm for $300 \cdot D$ generations, where D represents the problem dimensionality, or a total of $n = 200 \cdot 300 \cdot D$ solution evaluations. We decided on a larger number of generations to ensure that the algorithms achieve convergence. However, when predicting the best-performing algorithms, as explained later, we also used a smaller number of generations.

We execute 31 separate runs for every algorithm in every CMOP to enable a statistical comparison between the algorithms, which is a standard practice when comparing evolutionary algorithms. Finally, we utilize the I_{CMOP} indicator introduced in Section 2.3 to compare the algorithms. Following the proposal from the original paper [23], we set τ^* equal to 1.

Finding the best-performing algorithms for a given problem depends heavily on the number of evaluations executed. Thus, we set four target numbers of solution evaluations for which the best-performing algorithms are predicted. More specifically, we predict the best-performing algorithms after $\frac{n}{24}$, $\frac{n}{6}$, $\frac{n}{2}$, and n evaluations.

When calculating the ELA features, the problem objectives are first normalized with min–max normalization using the ideal and the nadir points as minimum and maximum vectors. The ELA features can be separated into sample features and random walk features. The parameters for calculating the ELA features were set as in the paper proposing the ELA features [1]. In particular, the sample features are

Table 3
Default parameter settings for all evolutionary algorithms used in this study.

Parameter	NSGA-II	NSGA-III	MOEA/D-IEpsilon	AGE	C-TAEA	SPEA2	GDE3	NSDE	NSDE-R
Population size	200	200	200	200	–	200	200	200	200
Reference type	–	energy	energy	–	energy	–	–	–	energy
Reference points	–	200	200	–	200	–	–	–	200
Crossover probability	0.9	1.0	0.9	0.9	1.0	0.9	–	–	–
Crossover prob. per var.	0.5	0.5	0.5	0.5	0.5	0.5	–	–	–
Crossover dist. index	15	30	15	15	30	15	–	–	–
Mutation probability	0.9	0.9	0.9	0.9	0.9	0.9	–	–	–
Mutation dist. index	20	20	20	20	20	20	–	–	–
DE selection type	–	–	–	–	–	–	random	random	random
# difference vectors	–	–	–	–	–	–	1	1	1
DE crossover type	–	–	–	–	–	–	binary	binary	binary
DE crossover rate	–	–	–	–	–	–	0.7	0.5	0.5

computed on a sample produced by the Latin hypercube sampling method, with a sample size of $1{,}000 \cdot D$, where D represents the problem dimensionality. The random walk features are computed from a random walk of length $(\frac{D}{W}) \cdot 10^3$, where W is a neighborhood parameter equal to $W = (2 \cdot D) + 1$. Because of the stochastic nature of the ELA features, the feature calculation process was repeated 100 times, resulting in 100 feature sets (or machine learning instances) per problem.

As it can be seen from the total number of evaluations an algorithm is run, the number of evaluations used for calculating the ELA features is only a small portion of the total number of evaluations (for example, for 2D CMOPs, the features are calculated based on 2,400 evaluations, and each algorithm is run for 120,000 evaluations). Thus, although calculating the ELA features requires some computational time, these evaluations can still serve as a starting point for generating the initial population of the selected algorithm, after the best-performing algorithms have been predicted.

To evaluate the performance of the models, we used the leave-one-problem-out evaluation methodology, which uses a single CMOP as testing data and the rest of the CMOPs as training data. The ASP values obtained from all test CMOPs were then averaged.

Table 4
Average ASP values from multi-label classification models, calculated using the leave-one-problem-out evaluation strategy on 2D, 5D, and 10D CMOPs. Numbers in bold denote the best model performance for each dimension and each target number of evaluations.

# evaluations	Classifier	2D	5D	10D
$\frac{n}{24}$	Dummy	0.612	0.331	0.240
	Decision Tree	0.726	0.698	0.642
	Random Forest	**0.798**	0.727	**0.714**
	kNN	0.724	**0.746**	0.659
$\frac{n}{6}$	Dummy	0.496	0.278	0.233
	Decision Tree	0.602	0.492	0.562
	Random Forest	**0.658**	0.570	0.600
	kNN	0.611	**0.609**	**0.638**
$\frac{n}{2}$	Dummy	0.464	0.221	0.219
	Decision Tree	0.535	0.444	0.488
	Random Forest	**0.670**	**0.483**	0.527
	kNN	0.654	0.427	**0.562**
$\frac{n}{1}$	Dummy	0.432	0.217	0.195
	Decision Tree	0.481	0.477	0.472
	Random Forest	**0.612**	**0.491**	**0.524**
	kNN	0.572	0.450	0.508

5. Results and discussion

5.1. Identifying best-performing algorithms

The best-performing algorithms for the 2D, 5D, and 10D CMOPs, identified by statistical tests, are presented in Fig. 2. In the figure, the CMOPs are on the horizontal axis (not annotated due to lack of space), and the algorithms are on the vertical axis. The horizontal axis is separated into four sections, representing the four target numbers of solution evaluations we analyze. A cell is colored blue if a given algorithm is selected as best performing for a given CMOP. Otherwise, it is white. If the algorithm is selected as the only best-performing algorithm for a given CMOP, the corresponding cell is colored dark blue. Otherwise, it is colored light blue.

The results show that at the beginning of the run ($\frac{n}{24}$ evaluations), more algorithms are identified as the best performing. However, as the number of evaluations increases, fewer algorithms are identified as best performing, and more CMOPs have a single algorithm identified as the only best-performing.

In the figures, we can observe that the single best-performing algorithms vary by dimensionality. For example, for 2D CMOPs, the algorithm most frequently identified as the single best-performing is SPEA2, whereas for 5D and 10D CMOPs, this algorithm is MOEA/D-IEpsilon.

Most importantly, the results show that multiple algorithms are identified as the best-performing even after converging towards the optimal solutions (after n evaluations). This demonstrates the importance of employing the proposed methodology for algorithm selection, which allows multiple algorithms to be considered best-performing.

We also analyzed how the identified best-performing algorithms would change if different, less strict, parameter values were used for the statistical tests (for example $\alpha = 0.05$ and $p = 0.05$). We found that the α and p-value parameters result in at most 5% of changes in the identified best-performing algorithms (see Appendix A for details).

To further analyze how the algorithms relate to each other, Fig. 3 presents the correlation matrices for all three dimensions (rows), as well as for each number of evaluations (columns). Each matrix entry quantifies the degree to which two algorithms are simultaneously identified as best-performing across the considered problems.

The strong positive correlations show that the algorithms tend to be successful under similar conditions. For example, the algorithms NSDE and NSDE-R typically exhibit a positive correlation across all problem dimensionalities. Additionally, in 5D and 10D, the algorithms NSGA-II, NSGA-III, and SPEA2 exhibit a high positive correlation. This indicates that specific patterns can be observed in the algorithms' behavior, which contributes to the reasoning behind incorporating classifier chains into the algorithm selection methodology.

5.2. Predicting best-performing algorithms

After identifying the best-performing algorithms for each CMOP, we tried to predict them by conducting the leave-one-problem-out algorithm selection experiment mentioned in Section 4. The results are presented in Table 4, where the results from the best-performing model per each dimension and number of evaluations are shown in bold.

The results show that the dummy models achieve higher ASP performance at the beginning of the runs, when compared to later in the run. For example, for $\frac{n}{24}$ evaluations, the dummy models achieve 0.612 for 2D, 0.331 for 5D, and 0.240 for 10D. Later in the run, after n evaluations, the dummy models achieve ASP values of 0.432 for 2D,

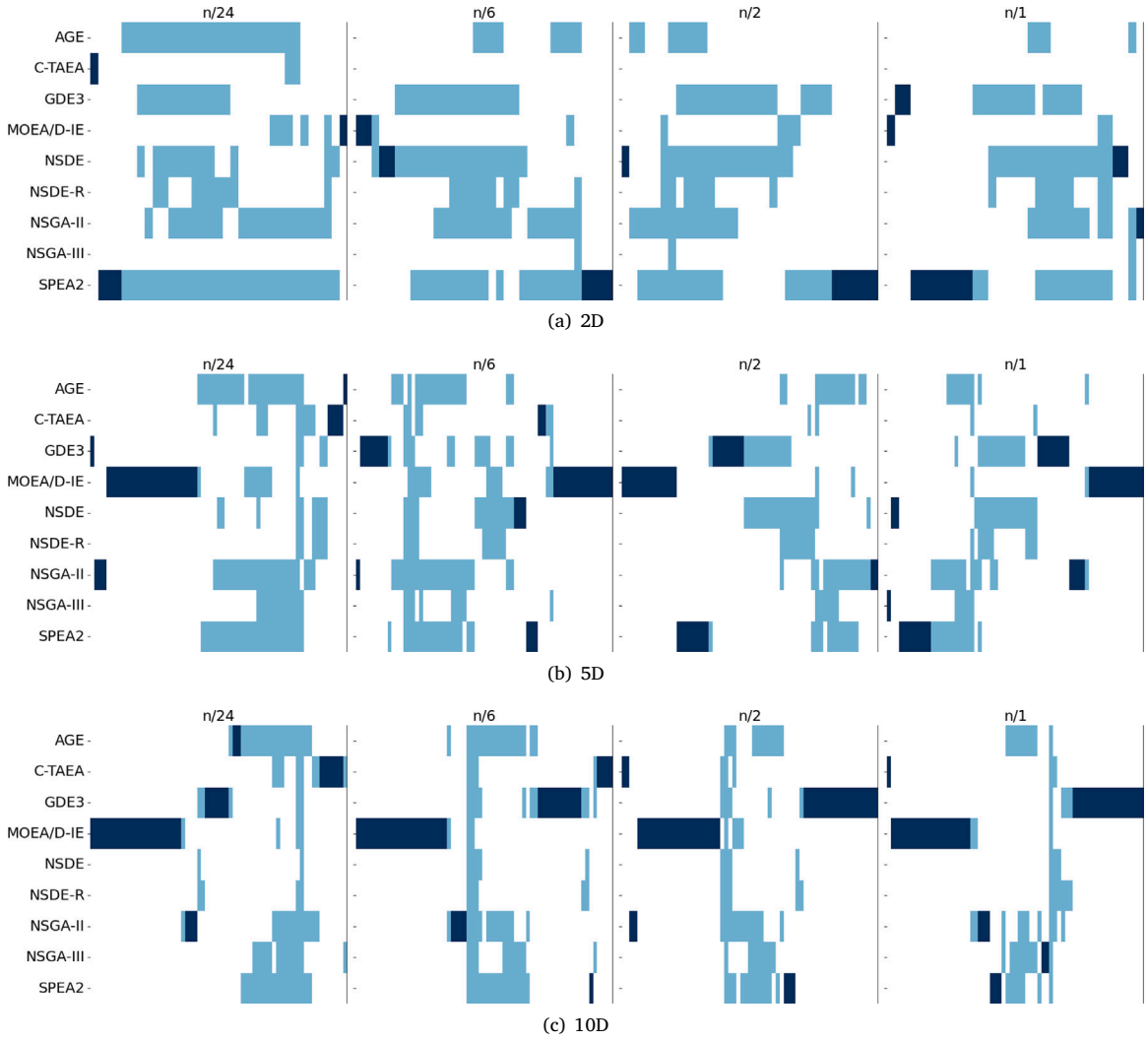


Fig. 2. Best-performing algorithms for 2D, 5D, and 10D CMOPs after $\frac{n}{24}$, $\frac{n}{6}$, $\frac{n}{2}$, and n evaluations. The rows present the nine algorithms, and the columns present the CMOPs. For better visibility, the problems are not listed in the same order. Each cell is colored blue if a given algorithm is selected as the best-performing one for a given CMOP. If the algorithm is selected as the only best-performing algorithm for a given CMOP, it is colored dark blue, otherwise, it is colored light blue.

0.217 for 5D, and 0.195 for 10D. This shows that predicting the best-performing algorithms at the beginning of the run is an easier machine learning task.

The algorithm selection models always outperform the dummy models. The differences between the dummy and the algorithm selection models are more evident when the dummy models have a lower ASP value. For example, in 10D, after n evaluations, the dummy model achieves an ASP value of 0.195 and the best-performing model achieves an ASP value of 0.524 (an increase of 0.329).

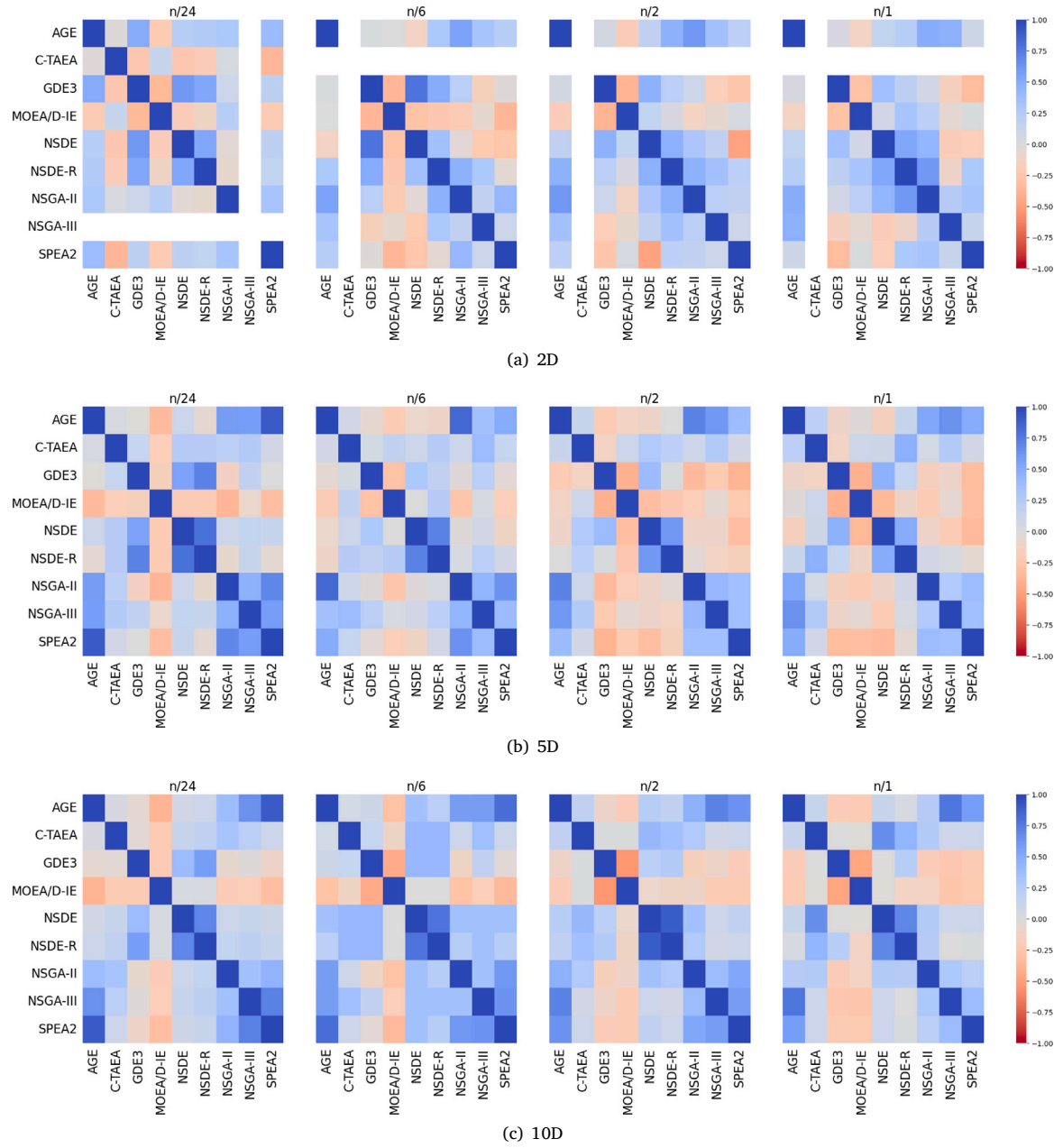
We can also observe that the Random Forest and the kNN classifier models are the best for predicting the best-performing algorithms.

Finally, for comparison, we used additional evaluation measures commonly used in multilabel classification (see Appendix B). However, since the evaluation measure proposed in Section 3.3 was specifically designed for algorithm selection, we emphasize that it remains the most appropriate choice for evaluating algorithm selection models in future work.

5.3. Advantages and limitations

The proposed algorithm selection methodology has several advantages over other algorithm selection methodologies:

- *Better algorithm selection models.* Even with our strict parameters for statistically testing the algorithms, the results from the statistical tests, shown in Fig. 2, indicate that in most CMOPs, multiple algorithms are identified as best-performing. This proves the importance of employing a methodology that considers multiple algorithms as best-performing when selecting an algorithm. Namely, as explained earlier, ignoring the fact that multiple algorithms can be identified as best performing can cause the machine learning models to overfit to the small differences between the algorithms, which are otherwise statistically irrelevant and should not be considered.
- *Multiple algorithm choice.* By using an algorithm selection methodology that produces a list of multiple best-performing algorithms as an output, one can further utilize personal preferences to select the algorithm they prefer for solving a given optimization problem. These preferences include the speed of the algorithms, the availability of the algorithm implementation, and the user's understanding of the algorithm, among other factors. In practice, these criteria are important when deciding which algorithm to use to solve an optimization problem.
- *The ASP measure.* Deciding which performance measure to use when evaluating a machine learning task is one of the most



(a) 2D

(b) 5D

(c) 10D

Fig. 3. Correlation matrices for 2D, 5D, and 10D CMOPs across the considered numbers of evaluations, showing how often pairs of algorithms are identified as best-performing. Blue cells indicate positive correlation, while red cells indicate negative correlation. Darker shades correspond to stronger correlation (positive or negative), whereas lighter shades indicate weaker correlation.

crucial steps. In this work, we proposed the ASP measure because it directly reflects the performance criteria most relevant to algorithm selection. Future works can also use this performance measure to compare the performance of algorithm selection models.

The proposed algorithm selection methodology also has its limitations. These include:

- *Sensitivity to parameters*. To statistically compare the algorithms, we first need to set the parameters of the statistical tests. This is the bottleneck of the proposed methodology because, with a different parameter setting, the statistical tests might identify a different set of best-performing algorithms, which could change the whole task of the algorithm selection problem.
- *Methodology evaluation*. To evaluate the proposed algorithm selection methodology, we needed to restrict ourselves to the problems for which we evaluated the proposed methodology, the considered algorithms, the considered problem dimensionalities, and the number of evaluations on which we analyzed and predicted the best-performing algorithms. However, all of these settings can be changed, and if changed, they could alter the results and their interpretation.
- *Higher problem dimensionalities not included*. The study is limited to 2D, 5D, and 10D CMOPs. However, as shown in Fig. 2, increasing problem dimensionality tends to reduce the number of algorithms identified as best-performing. This suggests that for higher-dimensional problems, a standard algorithm selection approach focused on selecting a single best algorithm might be sufficient. Still, publicly available real-world CMOPs with high

dimensionality are rare. For example, in [46], the highest dimensionality among real-world problems is 34D, while the majority of the problems have dimensionalities lower than 10D. Therefore, despite this limitation, the proposed methodology remains applicable to many real-world CMOPs.

- *Real world problems not included.* Only artificial CMOPs were used to evaluate the proposed methodology. However, their characteristics may differ from those of real-world problems, making the results presented in this work less applicable in the real world. Nonetheless, a large number of publicly available real-world problems are hard to find, making this limitation extremely hard to overcome. Moreover, artificial benchmark CMOPs were designed to represent real-world CMOPs and are used in various fields of algorithm analysis. For this reason, in the lack of a better option, we decided to evaluate the algorithm selection methodology on these problems.
- *Methodology evaluated only on CMO.* While the present study focuses on CMO due to our expertise and interest, the proposed algorithm selection methodology is general and can be extended to any type of optimization.

6. Conclusion

In this work, we proposed a novel methodology for algorithm selection. The methodology assumes that multiple algorithms can perform comparably and thus can be selected as the best-performing algorithm on a single optimization problem. In the methodology, we focused on finding the best-performing algorithms after a given number of solution evaluations.

The first step is identifying which algorithms perform best on a given problem. We do this by applying statistical tests on the performance indicator values obtained from 31 runs of each algorithm. If the statistical tests show that an algorithm's performance indicator values come from the same distribution as the best-performing algorithm's indicator values, both algorithms perform comparably and are said to be best performing.

By introducing the possibility of multiple algorithms being selected as the best-performing, the machine learning problem can no longer be a classical classification task with a single algorithm as its target. For this reason, we employ multi-label classification, where each optimization problem can be assigned multiple labels, each corresponding to one of the best-performing algorithms. We use Decision Tree, Random Forest, and kNN classifiers. In addition, we use a dummy classifier to evaluate the performance of the obtained machine learning models.

To evaluate the models, we proposed a more specialized measure for algorithm selection. Roughly explained, the measure rewards a model for recognizing at least one best-performing algorithm and penalizes the model for misclassifying a non-best-performing algorithm as best-performing.

The proposed algorithm selection methodology was tested in the field of constrained multiobjective optimization. We evaluated our algorithm selection methodology on 2D, 5D, and 10D CMOPs from seven benchmark suites. In addition, we included nine multiobjective optimization algorithms. We checked the algorithm selection methodology for $\frac{n}{24}$, $\frac{n}{6}$, $\frac{n}{2}$, and n evaluations, where n is the total number of solution evaluations performed in a single algorithm run. We used ELA features designed for CMOPs as input to the algorithm selection models. We used the leave-one-problem-out evaluation methodology to evaluate the models' performance.

The results showed that the dummy model had better ASP values after a smaller number of evaluations when compared to the ASP values after a higher number of evaluations. This indicates that the algorithm selection task is easier earlier in the algorithm run. Moreover, the algorithm selection models always outperformed the dummy models, with Random Forest models achieving the best results in most cases.

Table A.5
The change in the number of identified best-performing algorithms between the setting where $\alpha = p = 0.005$ and $\alpha = p = 0.05$.

Number of evaluations	2D	5D	10D
$\frac{n}{24}$	$\frac{14}{279} = 5.0\%$	$\frac{6}{585} = 1.0\%$	$\frac{0}{585} = 0.0\%$
$\frac{n}{6}$	$\frac{14}{279} = 5.0\%$	$\frac{18}{585} = 3.0\%$	$\frac{2}{585} = 0.3\%$
$\frac{n}{2}$	$\frac{14}{279} = 5.0\%$	$\frac{11}{585} = 1.8\%$	$\frac{0}{585} = 0.0\%$
$\frac{n}{1}$	$\frac{7}{279} = 2.5\%$	$\frac{8}{585} = 1.3\%$	$\frac{6}{585} = 1.0\%$

However, the most crucial finding was that even at the end of the runs, in most CMOPs analyzed, multiple algorithms are identified as best performing by the statistical tests. This stresses the importance of employing the proposed algorithm selection methodology to build general and reliable algorithm selection models.

CRediT authorship contribution statement

Andrejaana Andova: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. **Jordan N. Cork:** Writing – review & editing, Software, Data curation. **Tea Tušar:** Writing – review & editing, Visualization, Methodology, Conceptualization. **Bogdan Filipič:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors acknowledge the financial support from the Slovenian Research and Innovation Agency (young researcher program and research core funding No. P2-0209, and projects No. N2-0254 "Constrained Multiobjective Optimization Based on Problem Landscape Analysis", and GC-0001 "Artificial Intelligence for Science"). This publication is also based upon work from COST Action CA22137 "Randomized Optimization Algorithms Research Network" (ROAR-NET), supported by COST (European Cooperation in Science and Technology). We are grateful to Junoš Lukan for helping with the decisions regarding the statistical testing of the algorithms.

Appendix A. Testing different parameter values in statistical tests

To illustrate how the identified best-performing algorithms change with parameter values, we change the p-value from 0.005 to 0.05, and α from 0.005 to 0.05. The best-performing algorithms identified with the new parameter values are shown in Fig. A.4. This figure should be compared to Fig. 2 in the main part of the paper.

To quantify how many algorithms changed their values, we provide Table A.5, where the proportion of changes is shown. In the table it can be seen that by changing the p-value and the α parameters of the statistical tests, we obtain only minor changes (a maximum of 5%) in the set of identified best-performing algorithms. In some cases, such as for the 10D CMOPs after $\frac{n}{24}$ and $\frac{n}{2}$ evaluations, no changes are observed at all. These findings indicate that the methodology for identifying the best-performing algorithms is relatively robust to variations in the p-value and α parameters.

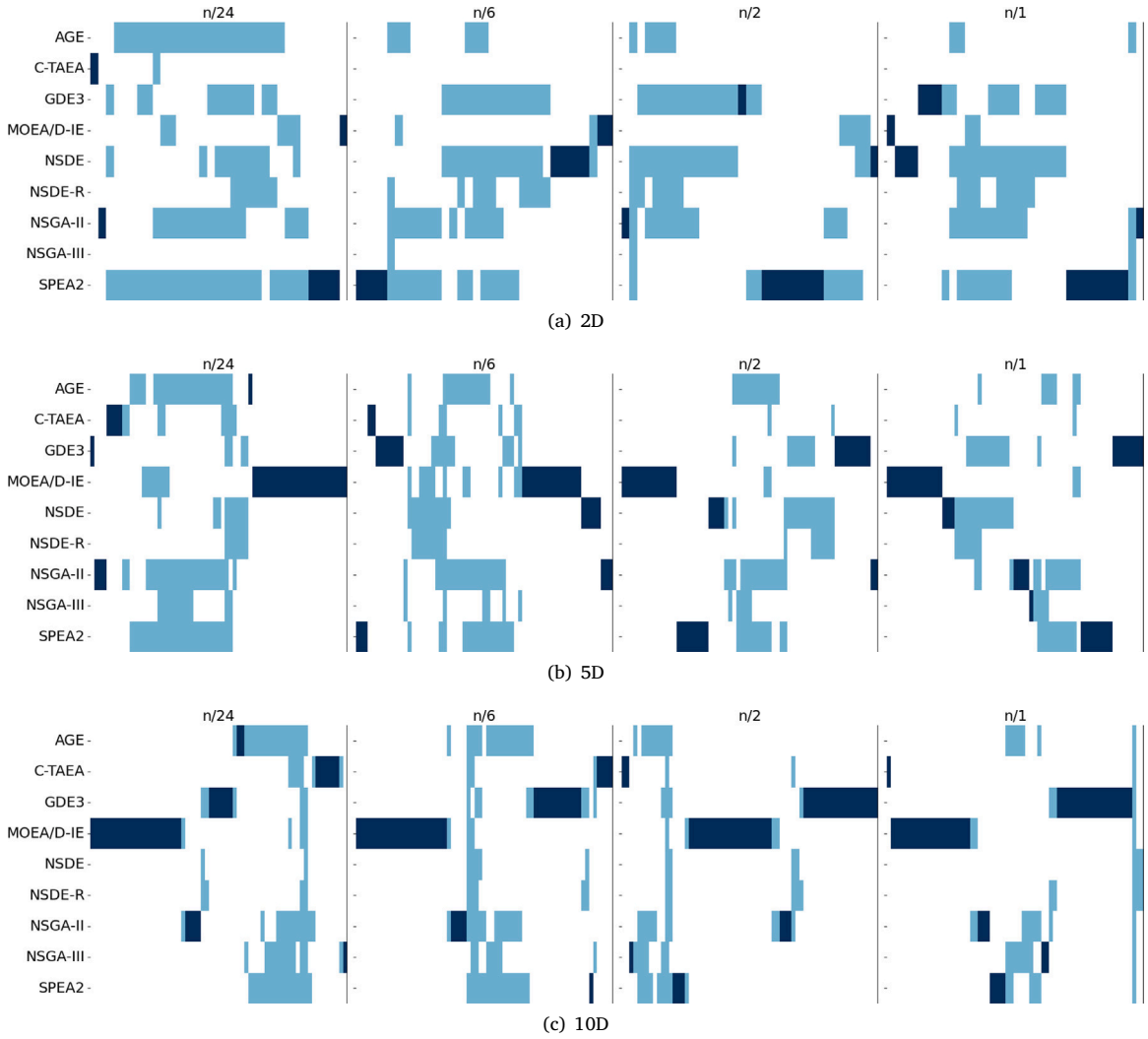


Fig. A.4. An alternative version of Fig. 2, where the parameters of statistical tests have been set to less strict values of $\alpha = 0.05$, and $p = 0.05$.

Appendix B. Using other evaluation measures to assess the algorithm selection models

To see how other popular multilabel evaluation measures would perform on the algorithm selection task, we also calculate the Label Ranking Average Precision (LRAP), Hamming Loss, and Subset Accuracy [47,48]. The implementations of these evaluation measures were taken from the scikit-learn library [31].

LRAP is an evaluation measure that assumes that the predictions are expressed as probabilities of the labels. Then, it checks the true positives of each instance to see how far they are ranked, and whether there are false positives that are ranked higher than them. This measure is the most similar to ASP, as it focuses on the true positives on each sample.

Hamming loss is the fraction of labels that are incorrectly predicted across all samples and all possible labels. Therefore, a lower Hamming loss score means better model performance. The main drawback of this evaluation measure is that it treats all labels equally, while in algorithm selection we want to focus on the true positives and the false positives of a sample.

Subset accuracy is a measure in multilabel classification that checks in how many instances the predicted binary vectors exactly match the true binary vector. This measure is very strict and, again, treats all labels equally.

Table B.6
Average LRAP for the same experiment as the one shown in Table 4.

# evaluations	Classifier	2D	5D	10D
$\frac{n}{24}$	Dummy	0.574	0.373	0.311
	Decision Tree	0.663	0.690	0.643
	Random Forest	**0.732**	0.708	**0.711**
	kNN	0.678	**0.732**	0.668
$\frac{n}{6}$	Dummy	0.488	0.343	0.312
	Decision Tree	0.589	0.534	0.615
	Random Forest	**0.626**	0.577	0.634
	kNN	0.575	**0.578**	**0.658**
$\frac{n}{2}$	Dummy	0.464	0.298	0.306
	Decision Tree	0.540	0.469	0.538
	Random Forest	**0.659**	**0.502**	0.584
	kNN	0.606	0.446	**0.594**
$\frac{n}{1}$	Dummy	0.441	0.292	0.284
	Decision Tree	0.458	0.488	0.531
	Random Forest	**0.596**	**0.536**	**0.565**
	kNN	0.556	0.480	0.548

The results obtained with the additional evaluation measures are shown in Tables B.6, B.7, and B.8.

Table B.7
Average Hamming Loss for the same experiment as the one shown in Table 4.

# evaluations	Classifier	2D	5D	10D
$\frac{n}{24}$	Dummy	0.360	0.355	0.317
	Decision Tree	0.274	0.184	0.167
	Random Forest	**0.219**	0.183	**0.125**
	kNN	0.280	**0.181**	0.177
$\frac{n}{6}$	Dummy	0.378	0.376	0.323
	Decision Tree	0.320	0.309	0.228
	Random Forest	**0.260**	0.267	**0.199**
	kNN	0.314	**0.263**	0.206
$\frac{n}{2}$	Dummy	0.369	0.319	0.284
	Decision Tree	0.277	0.252	**0.196**
	Random Forest	**0.234**	**0.212**	0.202
	kNN	0.287	0.264	0.209
$\frac{n}{1}$	Dummy	0.349	0.308	0.262
	Decision Tree	0.342	0.208	0.188
	Random Forest	**0.263**	**0.183**	**0.174**
	kNN	0.304	0.228	0.196

Table B.8
Average Subset Accuracy for the same experiment as the one shown in Table 4.

# evaluations	Classifier	2D	5D	10D
$\frac{n}{24}$	Dummy	0.023	0.022	0.031
	Decision Tree	0.110	0.333	0.403
	Random Forest	**0.225**	0.359	**0.485**
	kNN	0.216	**0.419**	0.450
$\frac{n}{6}$	Dummy	0.020	0.018	0.033
	Decision Tree	**0.140**	0.224	0.380
	Random Forest	0.137	**0.272**	0.409
	kNN	0.089	0.264	**0.422**
$\frac{n}{2}$	Dummy	0.020	0.025	0.056
	Decision Tree	0.097	0.203	0.336
	Random Forest	**0.239**	**0.236**	0.393
	kNN	0.166	0.182	**0.398**
$\frac{n}{1}$	Dummy	0.032	0.028	0.056
	Decision Tree	0.040	0.252	0.347
	Random Forest	**0.239**	**0.326**	**0.390**
	kNN	0.183	0.243	0.367

The results indicate that all key insights derived using the ASP evaluation measure also hold for the LRAP measure. However, comparing the dummy models across the other two measures, we observe a somewhat counterintuitive finding: the classification task appears more difficult for problems with lower dimensionality and fewer evaluations. This may be because a greater number of algorithms are identified as best-performing early in the run, as well as for lower-dimensional problems. Nevertheless, across all measures, the models consistently outperform the dummy models, with the Random Forest and kNN models typically achieving the best results.

Data availability

The data and the code from this work are available in a GIT repository at https://github.com/andrejaana/algorithm_selection_CMO.

References

[1] H. Alsouly, M. Kirley, M.A. Muñoz, An instance space analysis of constrained multi-objective optimization problems, IEEE Tran. Evolut. Comput. 27 (5) (2023) 1427–1439, http://dx.doi.org/10.1109/TEVC.2022.3208595.

[2] A. Vodopija, T. Tušar, B. Filipič, Characterization of constrained continuous multiobjective optimization problems: A feature space perspective, Inf. Sci. 607 (2022) 244–262, http://dx.doi.org/10.1016/j.ins.2022.05.106.

[3] A. Nikolikj, C. Doerr, T. Eftimov, RF+clust for leave-one-problem-out performance prediction, in: Applications of Evolutionary Computation: 26th International Conference, Springer, Cham, 2023, pp. 285–301, http://dx.doi.org/10.1007/978-3-031-30229-9_19.

[4] A. Andova, J.N. Cork, A. Vodopija, T. Tušar, B. Filipič, Predicting algorithm performance in constrained multiobjective optimization: A tough nut to crack, in: Applications of Evolutionary Computation: 27th International Conference, Springer, Cham, 2024, pp. 310–325, http://dx.doi.org/10.1007/978-3-031-56855-8_19.

[5] A. Andova, J.N. Cork, T. Tušar, B. Filipič, Enhancing algorithm performance prediction in constrained multiobjective optimization using additional training problems, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO, ACM, New York, 2024, pp. 458–466, http://dx.doi.org/10.1145/3638529.3654098.

[6] M. Seiler, U. Škvorc, G. Cenikj, C. Doerr, H. Trautmann, Learned features vs. classical ELA on affine BBOB functions, in: Parallel Problem Solving from Nature – PPSN, Springer Nature Switzerland, Cham, 2024, pp. 137–153, http://dx.doi.org/10.1007/978-3-031-70068-2_9.

[7] R.P. Prager, M.V. Seiler, H. Trautmann, P. Kerschke, Automated algorithm selection in single-objective continuous optimization: A comparative study of deep learning and landscape analysis methods, in: International Conference on Parallel Problem Solving from Nature, PPSN, Springer, Cham, 2022, pp. 3–17, http://dx.doi.org/10.1007/978-3-031-14714-2_1.

[8] P. Kerschke, H. Trautmann, Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning, Evol. Comput. 27 (1) (2019) 99–127, http://dx.doi.org/10.1162/evco_a_00236.

[9] M. Zuo, D. Gong, Y. Wang, X. Ye, B. Zeng, F. Meng, Process knowledge-guided autonomous evolutionary optimization for constrained multiobjective problems, IEEE Trans. Evolut. Comput. 28 (1) (2023) 193–207, http://dx.doi.org/10.1109/TEVC.2023.3243109.

[10] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the strength Pareto evolutionary algorithm, Technical Report TIK-103, Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, 2001.

[11] R. Angira, B. Babu, Non-dominated sorting differential evolution (NSDE): An extension of differential evolution for multi-objective optimization, in: 2nd Indian Conference on Artificial Intelligence, IICAI, 2005, pp. 1428–1443.

[12] Z. Ma, Y. Wang, Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons, IEEE Trans. Evolut. Comput. 23 (6) (2019) 972–986, http://dx.doi.org/10.1109/TEVC.2019.2896967.

[13] T. Eftimov, P. Korošec, B. Koroušić Seljak, A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics, Inf. Sci. 417 (2017) 186–215, http://dx.doi.org/10.1016/j.ins.2017.07.015.

[14] J. Rook, H.H. Hoos, H. Trautmann, Multi-objective ranking using bootstrap resampling, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO, ACM, New York, 2024, pp. 155–158, http://dx.doi.org/10.1145/3638530.3654436.

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

[16] K. Qiao, K. Yu, B. Qu, J. Liang, C. Yue, X. Ban, Feature extraction for recommendation of constrained multiobjective evolutionary algorithms, IEEE Trans. Evolut. Comput. 27 (4) (2023) 949–963, http://dx.doi.org/10.1109/TEVC.2022.3186667.

[17] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Chichester, 2001.

[18] J. Liang, X. Ban, K. Yu, B. Qu, K. Qiao, C. Yue, K. Chen, K.C. Tan, A survey on evolutionary constrained multiobjective optimization, IEEE Trans. Evolut. Comput. 27 (2) (2023) 201–221, http://dx.doi.org/10.1109/TEVC.2022.3155533.

[19] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, G. Rudolph, Exploratory landscape analysis, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO, ACM, New York, 2011, pp. 829–836, http://dx.doi.org/10.1145/2001576.2001690.

[20] C. Hanster, P. Kerschke, flaccogui: Exploratory landscape analysis for everyone, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO, ACM, New York, 2017, pp. 1215–1222, http://dx.doi.org/10.1145/3067695.3082477.

[21] A. Liefooghe, F. Daolio, S. Verel, B. Derbel, H. Aguirre, K. Tanaka, Landscape-aware performance prediction for evolutionary multiobjective optimization, IEEE Trans. Evolut. Comput. 24 (6) (2020) 1063–1077, http://dx.doi.org/10.1109/TEVC.2019.2940828.

[22] A.P. Guerreiro, C.M. Fonseca, L. Paquete, The hypervolume indicator: Computational problems and algorithms, ACM Comput. Sur. 54 (6) (2021) 662–671, http://dx.doi.org/10.1145/3453474.

[23] A. Vodopija, T. Tušar, B. Filipič, Characterization of constrained continuous multiobjective optimization problems: A performance space perspective, IEEE Trans. Evolut. Comput. 29 (1) (2025) 275–285, http://dx.doi.org/10.1109/TEVC.2024.3366659.

[24] N. Hansen, A. Auger, D. Brockhoff, T. Tušar, Anytime performance assessment in blackbox optimization benchmarking, IEEE Trans. Evolut. Comput. 26 (6) (2022) 1293–1305, http://dx.doi.org/10.1109/TEVC.2022.3210897.

[25] C. Leys, C. Ley, O. Klein, P. Bernard, L. Licata, Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median, J. Exp. Soc. Psychol. 49 (4) (2013) 764–766, http://dx.doi.org/10.1016/j.jesp.2013.03.013.

[26] L. Ståhle, S. Wold, Analysis of variance (ANOVA), Chemom. Intell. Lab. Syst. 6 (4) (1989) 259–272, http://dx.doi.org/10.1016/0169-7439(89)80095-4.

[27] J.W. Tukey, Comparing individual means in the analysis of variance, Biometrics 5 (2) (1949) 99–114, http://dx.doi.org/10.2307/3001913.

[28] P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., SciPy 1.0: Fundamental algorithms for scientific computing in Python, Nat. Methods 17 (3) (2020) 261–272, http://dx.doi.org/10.1038/s41592-019-0686-2.

[29] S. Seabold, J. Perktold, Statsmodels: Econometric and statistical modeling with Python, in: Proceedings of the 9th Python in Science Conference, SciPy, 2010, pp. 92–96.

[30] J. Read, B. Pfahringer, G. Holmes, E. Frank, Classifier chains for multi-label classification, in: Machine Learning and Knowledge Discovery in Databases, Springer, Berlin, 2009, pp. 254–269, http://dx.doi.org/10.1007/s10994-011-5256-5.

[31] 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.

[32] H. Jain, K. Deb, An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach, IEEE Trans. Evolut. Comput. 18 (4) (2013) 602–622, http://dx.doi.org/10.1109/TEVC.2013.2281534.

[33] J.P. Li, Y. Wang, S. Yang, Z. Cai, A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization, in: IEEE Congress on Evolutionary Computation, CEC, IEEE, 2016, pp. 4175–4182, http://dx.doi.org/10.1109/CEC.2016.7744320.

[34] K. Deb, A. Pratap, T. Meyarivan, Constrained test problems for multi-objective evolutionary optimization, in: International Conference on Evolutionary Multi-Criterion Optimization, EMO, Springer, Berlin, 2001, pp. 284–298, http://dx.doi.org/10.1007/3-540-44719-9_20.

[35] Z. Fan, W. Li, X. Cai, H. Li, C. Wei, Q. Zhang, K. Deb, E. Goodman, Difficulty adjustable and scalable constrained multiobjective test problem toolkit, Evol. Comput. 28 (3) (2020) 339–378, http://dx.doi.org/10.1162/evco_a_00259.

[36] K. Li, R. Chen, G. Fu, X. Yao, Two-archive evolutionary algorithm for constrained multiobjective optimization, IEEE Trans. Evolut. Comput. 23 (2) (2018) 303–315, http://dx.doi.org/10.1109/TEVC.2018.2855411.

[37] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition, Technical Report CES-487, The School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK, 2009.

[38] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evolut. Comput. 6 (2) (2002) 182–197, http://dx.doi.org/10.1109/4235.996017.

[39] J. Blank, K. Deb, P.C. Roy, Investigating the normalization procedure of NSGA-III, in: International Conference on Evolutionary Multi-Criterion Optimization, EMO, Springer, Berlin, 2019, pp. 229–240, http://dx.doi.org/10.1007/978-3-030-12598-1_19.

[40] Z. Fan, W. Li, X. Cai, H. Huang, Y. Fang, Y. You, J. Mo, C. Wei, E. Goodman, An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions, Soft Comput. 23 (2019) 12491–12510, http://dx.doi.org/10.1007/s00500-019-03794-x.

[41] A. Panichella, An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO, ACM, New York, 2019, pp. 595–603, http://dx.doi.org/10.1145/3321707.3321839.

[42] S. Kukkonen, J. Lampinen, GDE3: The third evolution step of generalized differential evolution, CEC, in: IEEE Congress on Evolutionary Computation, vol. 1, IEEE, 2005, pp. 443–450, http://dx.doi.org/10.1109/CEC.2005.1554717.

[43] S.R. Reddy, G.S. Dulikravich, Many-objective differential evolution optimization based on reference points: NSDE-R, Struct. Multidisc. Optim. 60 (4) (2019) 1455–1473, http://dx.doi.org/10.1007/s00158-019-02272-0.

[44] J. Blank, K. Deb, Pymoo: Multi-objective optimization in Python, IEEE Access 8 (2020) 89497–89509, http://dx.doi.org/10.1109/ACCESS.2020.2990567.

[45] B. Leite, A.O.S. da Costa, E.F. da Costa Junior, Multi-objective optimization of adiabatic styrene reactors using Generalized Differential Evolution 3 (GDE3), Chem. Eng. Sci. 265 (2023) 118196, http://dx.doi.org/10.1016/j.ces.2022.118196.

[46] A. Kumar, G. Wu, M.Z. Ali, Q. Luo, R. Mallipeddi, P.N. Suganthan, S. Das, A benchmark-suite of real-world constrained multi-objective optimization problems and some baseline results, Swarm Evolut. Comput. 67 (2021) 100961, http://dx.doi.org/10.1016/j.swevo.2021.100961.

[47] G. Tsoumakas, I. Katakis, Multi-label classification: An overview, Int. J. Data Warehous. Min. 3 (3) (2007) 1–13, http://dx.doi.org/10.4018/jdwm.2007070101.

[48] M.L. Zhang, Z.H. Zhou, A review on multi-label learning algorithms, IEEE Trans. Knowl. Data Eng. 26 (8) (2013) 1819–1837, http://dx.doi.org/10.1109/TKDE.2013.39.