Enhancing Algorithm Performance Prediction in Constrained Multiobjective Optimization Using Additional Training Problems

Andrejaana Andova
Jordan N. Cork
Tea Tušar
Bogdan Filipič
andrejaana.andova@ijs.si
jordan.cork@ijs.si
tea.tusar@ijs.si
bogdan.filipic@ijs.si
Jožef Stefan Institute
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia

ABSTRACT

A research problem studied extensively in recent years is the prediction of optimization algorithm performance. A common approach is using the landscape features of optimization problems to train machine learning models. These models are then used to predict algorithm performance. Due to the small number of constrained multiobjective optimization problems (CMOPs) available for benchmarking, training a machine learning model to predict algorithm performance is a hard task. To address this issue, this study uses the functions from the `bbob` and `bbob-constrained` benchmark problems to generate new CMOPs. These are then used as additional training examples for the machine learning models. Given the large number of generated CMOPs, the experiments in this study are limited to those with two objectives and two variables. The obtained results are promising. Using additional problems in the training phase improves the predictions in half of the defined classification tasks.

CCS CONCEPTS

• **Theory of computation** → *Bio-inspired optimization*; *Nonconvex optimization*; **Numeric approximation algorithms**; • **Computing methodologies** → **Classification and regression trees**.

KEYWORDS

constrained multiobjective optimization, exploratory landscape analysis, algorithm performance prediction

ACM Reference Format:
Andrejaana Andova, Jordan N. Cork, Tea Tušar, and Bogdan Filipič. 2024. Enhancing Algorithm Performance Prediction in Constrained Multiobjective Optimization Using Additional Training Problems. In *Genetic and Evolutionary Computation Conference (GECCO '24), July 14–18, 2024, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3638529.3654098

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0494-9/24/07.
https://doi.org/10.1145/3638529.3654098

1 INTRODUCTION

Optimization algorithms behave differently on different problem landscapes. Therefore, a lot of recent research in evolutionary computation has been focused on developing methodologies that can recommend the best algorithm for a given problem, based on the analysis of its landscape. The task of selecting the best algorithm for a given optimization problem is called algorithm selection. Most of the research on this topic was done on single-objective optimization problems [19, 27]. Typically, researchers approach algorithm selection by extracting exploratory landscape analysis (ELA) features from a population of solutions, which are then used as inputs to machine learning models.

Although there are many single-objective benchmark problems, a substantial effort has been recently put into developing additional single-objective optimization problems as affine combinations of pairs of the available benchmark problems [8]. One of the motivations behind this is to increase the number of problems researchers can use to analyze and assess the performance of optimization algorithms [30].

The set of constrained multiobjective problems (CMOP) available for benchmarking is relatively limited compared to other continuous optimization problems. Moreover, CMOPs are considerably more complex and challenging to solve. For this reason, the algorithm selection task for CMOPs is an ambitious goal, and thus, as a first step towards solving it, researchers have instead proposed methods for predicting algorithm performance [3].

An interesting approach to creating new benchmark problems for multiobjective optimization has been proposed in [6], where combinations of `bbob` problems have been used as separate objectives to generate two suites of multiobjective benchmark problems, `bbob-biobj` and `bbob-biobj-ext`. However, their large size (55 and 92 problems, respectively) is a significant limitation as evaluating algorithms on each of these problems in six dimensions and 15 instances can take a long time. As a solution to this issue, [2] attempted to use ELA features to select a subset of problems from the `bbob-biobj` benchmark suite. However, they came to a number

of limitations in their approach, most of them connected to the differences between instances of the same problem combination.

This paper focuses on predicting algorithm performance on CMOPs. To achieve this, we use state-of-the-art ELA features [1], explicitly designed for CMOPs. These features serve as inputs to traditional machine learning classification models. In our experiments, we use a comprehensive set of baseline CMOPs to evaluate the performance of our methodology. Furthermore, inspired by [6], we create additional bi-objective CMOPs, which we include in the training data. This way, we increase the number of CMOPs in the training data and their diversity. Due to the large number of additional CMOPs, on which we also have to run three selected optimization algorithms, we restrict these initial experiments to 2D CMOPs only.

The structure of this paper is as follows. In Section 2, we present the background of this work and in Section 3, we describe the proposed methodology. Then, Section 4 explains the experimental setup, Section 5 presents the obtained results and Section 6 discusses them. Finally, in Section 7, we give a conclusion to the experiments and outline ideas for future research.

2 BACKGROUND

In this section, we firstly formulate what a CMOP is, and then briefly explain the ELA features and the performance metric used in the methodology. Finally, we give a short introduction of the `bbob` and the `bbob-constrained` benchmarks used to create the the additional CMOPs.

2.1 Constrained Multiobjective Optimization

A CMOP can be formulated as:

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

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

A solution $\mathbf{x}$ is *feasible*, if it satisfies all constraints, i.e., $g_j(\mathbf{x}) \leq 0$, for $j = 1, \ldots, J$ and $h_k(\mathbf{x}) = 0$, for $k = 1, \ldots, K$. A feasible solution $\mathbf{x}$ *dominates* another feasible solution $\mathbf{y}$ 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 $1 \leq m \leq M$. A feasible solution $\mathbf{x}^*$ is a *Pareto-optimal* if there exists no feasible solution $\mathbf{x} \in S$ that dominates $\mathbf{x}^*$. All nondominated feasible solutions form the *Pareto set* S_o, and the image of the Pareto set in the objective space is the *Pareto front*, $P_o = \{f(\mathbf{x}) \mid \mathbf{x} \in S_o\}$.

The point in the objective space with the best objective values is the *ideal point* $z_I = (\min_{\mathbf{x} \in S_o} f_1(\mathbf{x}), \ldots, \min_{\mathbf{x} \in S_o} f_M(\mathbf{x}))$. The *nadir point* represents the point in the objective space with the worst fitness values across all solutions in the Pareto front $z_N = (\max_{\mathbf{x} \in S_o} f_1(\mathbf{x}), \ldots, \max_{\mathbf{x} \in S_o} f_M(\mathbf{x}))$.

Nondomination ranking is a method in multiobjective optimization used for sorting a set of solutions into fronts, based on their dominance. All nondominated solutions get a nondomination rank of 1, solutions that are dominated only by the nondominated solutions get a nondomination rank of 2, and so on.

2.2 Exploratory Landscape Analysis for Constrained Multiobjective Optimization

ELA is a methodology used for extracting features from optimization problems, based on a sample of evaluated solutions [24]. While there are many ELA features for single-objective optimization problems [16], much fewer exist for multi-objective problems with and without constraints [1, 21, 32]. The underdevelopment of the ELA features for these problems presents an additional difficulty in solving the algorithm selection task for them.

In [1], the authors collected state-of-the-art ELA features specifically designed for CMOPs. These features are calculated based on an initial sample of solutions (*sample features*), as well on solutions generated during a random walk (*random walk features*). Moreover, they can be grouped into three distinct groups, each offering insights into different aspects of CMOPs: the multiobjective landscape, the violation landscape, and a hybrid, the multiobjective-violation landscape.

The *multiobjective landscape group* contains ELA features describing the objectives and their internal relations. The sample features in this group include the proportion of unconstrained Pareto optimal solutions, the hypervolume of the unconstrained Pareto front, and the correlation between objective values. The random walk features comprise statistical measures concerning the distance between random walk neighbors in the objective space.

The *violation landscape group* contains ELA features describing the problem's constraints. The sample features in this group comprise statistics related to global constraint violations, while the random walk features comprise constraint violation statistics between random walk neighbors.

The *multiobjective violation landscape group* contains features describing the interactions between the objectives and the constraints. In this group, the sample features include the proportion of feasible solutions, the proportion of Pareto optimal solutions, hypervolume calculations, statistics on correlations between objectives and constraints, and others. The random walk features comprise statistics that describe the dominance relations between random walk neighbors.

2.3 Quality indicator for CMOPs

The hypervolume [13] is the most well-known quality indicator for multiobjective optimization. However, one drawback to it is that it assumes all solutions are feasible, which does not hold in constrained multiobjective optimization. For this reason, [31] proposed a new quality indicator, I_{CMOP}, designed specifically for CMOPs. This indicator extends the concept of a hypervolume-based quality indicator I_{MOP} for unconstrained multiobjective optimization from a previous work [14].

I_{MOP} is defined as follows:

(1) When the solutions in the set are dominated by the nadir point, the I_{MOP} quality indicator takes the value of the distance between the solutions and a region of interest bounded by the ideal and the nadir point.
(2) When at least some of the solutions in the set dominate the nadir point, the quality indicator equals the negative value of the hypervolume, calculated with the reference point being equal to the nadir point.

Next, I_{CMOP} is defined as follows:

(1) When all solutions in the set are infeasible, the I_{CMOP} quality indicator takes on the smallest constraint violation value of all solutions in the set, with the addition of a threshold τ^*.
(2) When the set contains at least one feasible solution, the quality indicator is determined by the value of I_{MOP}, upper bounded by a threshold τ^*, meaning it equals $\min\{I_{\text{MOP}}, \tau^*\}$.

I_{CMOP}, same as I_{MOP}, assumes that lower indicator values signify better sets of solutions and vice versa.

Two preconditions to be able to compare CMOPS using the I_{CMOP} indicator are: (1) the objectives of each CMOP need to be normalized using min-max normalization with the ideal and the nadir point and (2) the constraint violations and the distances between the solutions and the region of interest need to be normalized as well. The normalization in the second precondition can be done using a sample of 100 evaluated solutions.

2.4 The bbob and bbob-constrained benchmark suites

The bbob benchmark suite contains 24 noiseless functions, which can be shifted and transformed into different instances [11]. The functions have known optima and cover various difficulties found in real-world problems.

The bbob-constrained benchmark suite contains nine functions taken from the bbob benchmark suite. Six different types of constraints of various difficulty are applied on each function, resulting in $9 \cdot 6 = 54$ function-constraint combinations. Finally, similarly to the bbob benchmark, the function-constraint combinations are shifted and transformed, resulting in different problem instances.

Both benchmarks can be found in the Comparing Continuous Optimizers (COCO) platform [15].

3 METHODOLOGY

In this section, we present the classification tasks of the experiment and the approach to the automatic creation of additional CMOPs.

3.1 Classification Tasks

To measure algorithm performance, one possibility is to track the number of function evaluations needed to achieve a particular value of a quality indicator (also called a *target*) [14]. An even better understanding of the performance of a given algorithm can be gained by creating a set of targets and analyzing the number of function evaluations needed to achieve a proportion of them.

Following the setup proposed in [31], we use the I_{CMOP} indicator with target precision values of $\tau^{\epsilon} = \tau^* + \epsilon$, where $\epsilon \in \{10^p | p \in \{-5, -4.9, \ldots, 0\}\} \cup \{1 + 10^p | p \in \{-5, -4.9, \ldots, 0\}\}$ and τ^* is a threshold value. This results in a total of 100 targets, where one half of them are values for the I_{MOP} indicator, while the other half deals with cases when none of the solutions are feasible (the first and the second case when describing the I_{CMOP} indicator). To evaluate algorithm performance, we will thoroughly analyze when the algorithm achieved the following target proportions: 50% (equating to finding at least one feasible solution), 60%, 70%, 80%, and 90% of the targets.

Predicting the exact number of function evaluations needed to achieve a certain proportion of targets is a regression task. Sometimes (when the algorithm does not achieve the given percentage of targets during the observed evaluation), this number is unknown. Therefore, to simplify the regression task, we categorize the prediction values into distinct classes and transform the regression task into a classification one.

The number of evaluations depends on the experimental setup. In our scenario, we conduct a maximum of 24,000 evaluations using algorithms with a population size of 200. Therefore, we define the following classes:

- Class 1: The goal is met within 1 to 200 evaluations (during the initial generation),
- Class 2: The goal is met within 201 to 2,000 evaluations.
- Class 3: The goal is met within 2,001 to 8,000 evaluations.
- Class 4: The goal is met within 8,001 to 24,000 evaluations.
- Class 5: The goal is never met.

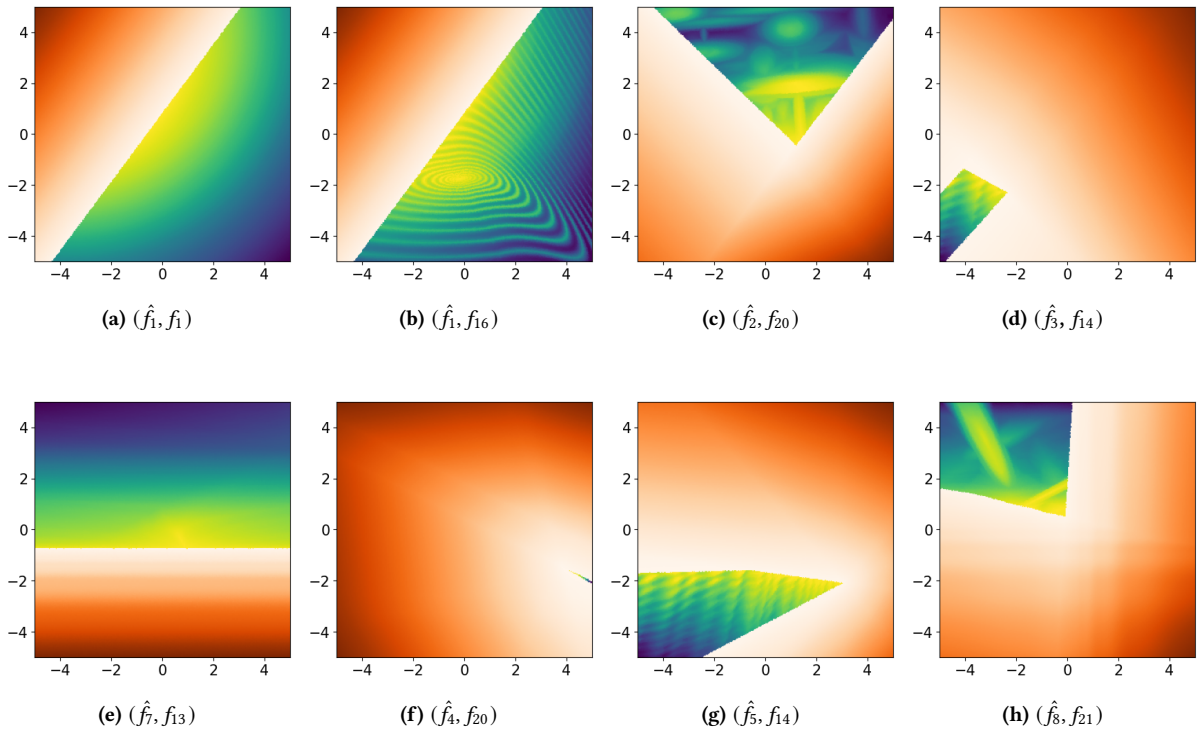
3.2 Additional problem generation

Training a machine learning model requires large amounts of diverse data. However, when predicting the algorithm performance in constrained multiobjective optimization, there are only tens of CMOPs on which we can train the models. In addition to this, it was found in [3] that all ELA features calculated on different samples from one CMOP have similar values and do not add diversity to the data. For this reason, we propose a method to automatically create additional CMOPs, which can be used for training.

To create the additional CMOPs, we use the problems from the bbob and bbob-constrained benchmark suites. We form a single new instance of a constrained bi-objective problem by taking the first instance of a bbob-constrained function as its first objective and the second instance of a bbob function as its second objective. The constraints of the new bi-objective problem instance are equal to those of the bbob-constrained problem. We do so for all possible combinations of the problems from the two suits, resulting in a total of $54 \cdot 24 = 1,296$ additional CMOPs.

By using all problem combinations from the two suites of benchmark problems, we ensure the widest possible selection of problem properties stemming from these suites. Figure 1 presents example landscapes of eight hand-picked problems from the set of the additional CMOPs. In each plot, the problem landscape is approximated with a grid of 300×300 points. The plot shows the dominance rank ratio values of feasible solutions in yellow-green-blue shades (lighter hues denote grid points that are dominated by fewer grid point) [12] and the constraint violation values of infeasible solutions in orange shades (lighter hues denote lower constraint violation values). The eight chosen examples show the variety of problems in the set of additional CMOPs. Note that most of the additional CMOPs have a lower percentage of feasible solutions (see for example problems (d) and (f)) when compared to the baseline CMOPs.

A crucial drawback to creating the additional CMOPs this way is that we do not know their Pareto set and front. While we know the location of one of the two single-objective optima (that of the bbob-constrained function), the other optimum can be shifted since the constraints from the bbob-constrained problem affect



(a) $(\hat{f}_1, f_1)$ **(b)** $(\hat{f}_1, f_{16})$ **(c)** $(\hat{f}_2, f_{20})$ **(d)** $(\hat{f}_3, f_{14})$

(e) $(\hat{f}_7, f_{13})$ **(f)** $(\hat{f}_4, f_{20})$ **(g)** $(\hat{f}_5, f_{14})$ **(h)** $(\hat{f}_8, f_{21})$

Figure 1: Problem landscapes of eight hand-picked additional CMOPs annotated by $(\hat{f}_a, f_b)$, where $\hat{f}_a$, $a \in \{1, \ldots, 54\}$ is the `bbob-constrained` function and f_b, $b \in \{1, \ldots, 24\}$ the `bbob` function. The yellow-green-blue shades present the dominance rank ratio, while the orange shades present the constraint violations.

also the bbob function. Therefore, it is hard to normalize these problems.

To normalize the objectives, as explained in Section 3.1, we need the ideal and the nadir point. These can be retrieved from the true Pareto front or an approximation of it. Given that we do not know the true Pareto front for the additional CMOPs, we compute its approximation for each created additional CMOP by running multiple times three optimization algorithms for a relatively large number of evaluations (see Section 4 for more details). Then, the Pareto front approximation is formed as the set of all nondomianted solutions from all algorithm runs on a problem.

4 EXPERIMENTAL SETUP

In this section, we first present the used CMOPs, then the algorithms and their parameter setting, and the specifics for calculating the ELA features. Finally, we describe the machine learning models used and explain the two performed experiments.

This research focuses on bi-objective CMOPs with 2D search spaces. The experimental framework includes five well-known benchmark suites: MW [23], C-DTLZ [18], CTP [7], DAS-CMOP [10], and DC-DTLZ [20], as well as three separate CMOPs: BNH [4], TNK [29], and SRN [28]. In total, 36 baseline CMOPs were selected from these suites. In addition to these problems, a total of 1,296 additional CMOPs were created as explained in Section 3.2. A detailed count of the number of CMOPs from each suite is presented in Table 1.

Table 1: The number of bi-objective 2D CMOPs per suite used in this study.

CMOP set	Benchmark	# of problems
Baseline CMOPs	MW	8
	C-DTLZ	5
	CTP	8
	DAS-CMOP	6
	DC-DTLZ	6
	BNH	1
	TNK	1
	SRN	1
Additional CMOPs	/	1,296

For evaluating algorithmic efficacy, three multiobjective optimization algorithms were chosen, each incorporating a distinct technique for handling constraints. These comprise NSGA-III [18], MOEA/D-IEpsilon [9], and C-TAEA [20]. For a fair comparison of the algorithms, the population size was set to 200, and the number of generations to 120, for each algorithm. The rest of the parameters are the same as in the original papers. We execute 31 separate runs for each algorithm on every CMOP to ensure that the algorithm performance can be accurately estimated. Thus, to estimate the number of evaluations needed to achieve a given proportion of

targets, we used the mean of the number of evaluations over all 31 algorithm runs.

Additionally, to calculate the Pareto front approximations needed to normalize the additional problems, we separately run all three algorithms 31 times, each time for 600 generations.

The ELA features consist of sample and random walk features. Sample features are calculated on a sample of size 1,024, retrieved using the Latin hypercube sampling method. The random walk features are computed during a random walk of length 400. Using the sample and the random walk solutions, a total of 77 features were computed. To be able to compare the CMOPs using the ELA features, we normalize the objectives using a min-max normalization, where the ideal and the nadir point of a CMOP are taken as the minimum and maximum values. In addition to this, for ELA features that require hypervolume calculations, we set the reference point to $(100, 100)$. Because of the stochastic nature of ELA features, we repeated the feature calculation process 100 times for each CMOP, resulting in 100 sets of features (i.e., learning instances) per problem.

To predict the evaluation class, we used the ELA features as input to three classical machine learning algorithms – Decision Trees [22], Random Forest Classification [5], and C-Support Vector Classification (SVC) [26], which are the usual choice when predicting algorithm performance from ELA features. We also included a dummy model that predicted the class in a stratified way based on the distribution of the classes in the training data. In our experiments, we used the `scikit-learn` implementations of these methods with default parameter settings [25]. To evaluate the performance of the classifiers, we used the leave-one-problem-out evaluation methodology. This way, we ensured that no information about the target CMOP is available in the training data. Thus, all instances of a problem are used as test data, while the instances from the rest of the problems are used as training data.

We conducted two experiments. In the first, we used only the 36 baseline CMOPs to train and test the classifier, resulting in a total of 3500 training instances, and 100 testing instances. In the second experiment, we added the additional CMOPs when training the classifier, but used only the 36 baseline CMOPs when evaluating it with the leave-one-problem-out evaluation. In total in this experiment, we used 133,100 training instances, and 100 testing instances. By testing only on the baseline CMOPs in the second experiment, we were able to compare the results obtained from the first experiment to the results from the second experiment and decide whether including the additional CMOPs into the training data improved the prediction performance.

5 RESULTS

The classification accuracy of the two experiments described in Section 4 is presented in Tables 2 and 3. In both tables, we show in bold the best model performance for each algorithm and each target percentage. In addition, for an easier comparison of the results from the two experimental setups, the highlighted cells in Table 3 correspond to cases, in which the classification accuracy improved when including the additional CMOPs into the training data.

The results show that, when predicting the evaluation class for achieving 50% of the targets, the best performing models achieve 100% accuracy. This accuracy was achieved for all algorithms on the baseline CMOPs using Decision Tree and Random Forest, and for most of the cases where all CMOPs were used. However, outperforming a model with a 100% accuracy is hard, so in this case, training models on all CMOPs shows that one is able to generalize well and still achieve comparable results.

When predicting the evaluation class for achieving 60% of the targets, the best-performing model in both experiments was the Random Forest. For NSGA-III and C-TAEA, the best-performing model was one trained using all CMOPs achieving 61.2% accuracy in both cases, while the best-performing model for MOEA/D was trained using only the baseline CMOPs, and achieved 62.1% accuracy. The difference in the accuracy between the best performing model for MOEA/D when using the baseline CMOPs and all CMOPs is only 0.1%.

Similarly, when predicting the evaluation class for 70% of the targets, the best-performing model for NSGA-III and MOEA/D was one trained using all CMOPs, while the best-performing model for C-TAEA was trained using only the baseline CMOPs. In this case, the best-performing model was Random Forest, achieving 60.6% and 56.3% accuracy for NSGA-III and MOEA/D, respectively, while for C-TAEA it achieved 54.4% accuracy.

When predicting the evaluation class for 80% of the targets, the best-performing models for NSGA-III and MOEA/D were trained on baseline CMOPs, achieving 64.1% and 58.8% accuracy respectively, while the best-performing model and C-TAEA was trained on the all CMOPs and achieved 60.7% accuracy.

For 90% of the targets, the best-performing model for NSGA-III and C-TAEA was trained on the baseline CMOPs and achieved 62.4% and 57.6% accuracy respectively, while the best-performing model for MOEA/D was trained on all CMOPs and achieved 54.3% accuracy.

Figure 2 presents the distributions of the evaluation classes for each percentage of targets. The figure shows that the distributions of the evaluation classes of the baseline CMOPs and the additional CMOPs rarely match. Also, the range of covered evaluation classes is wider for the additional CMOPs than for the baseline CMOPs.

To analyze the difference in the performance of the three selected optimization algorithms on all CMOPs, compared to the baseline CMOPs, we generate Venn diagrams in Figure 3 using the visualization tool proposed in [17]. In this figure, we can see how often the three selected algorithms performed similarly on the same set of CMOPs (separately for baseline and additional CMOPs), and how often one or two algorithms performed better than the rest. To determine which algorithm performs best, we use the evaluation classes presented in Section 2.3. Specifically, if one algorithm is assigned a lower evaluation class compared to the other two for a given evaluation target, we say it performs better than the other two. If all three algorithms belong to the same evaluation class, we say that there is no difference between the performance of the three algorithms.

We can notice that, for the baseline CMOPs, all algorithms belong to the same evaluation class when trying to achieve 50% of the targets. However, for the additional CMOPs, there were 43 problems

Table 2: Classification accuracy (of learning models predicting the algorithm performance classes by using the leave-one-problem-out methodology) in the first experiment. We use only the baseline CMOPs for training and testing. Numbers in bold denote the best model performance for each algorithm and each target percentage.

Targets	Classifier	NSGA-III	MOEA/D	C-TAEA
50%	Dummy	0.838	0.838	0.838
	Decision tree	**1.000**	**1.000**	**1.000**
	Random Forest	**1.000**	**1.000**	**1.000**
	SVC	0.916	0.916	0.916
60%	Dummy	0.257	0.252	0.253
	Decision tree	0.425	0.471	0.488
	Random Forest	**0.594**	**0.621**	**0.566**
	SVC	0.444	0.444	0.396
70%	Dummy	0.249	0.261	0.253
	Decision tree	0.363	0.506	0.455
	Random Forest	**0.553**	**0.535**	**0.544**
	SVC	0.166	0.416	0.508
80%	Dummy	0.320	0.258	0.290
	Decision tree	0.604	0.574	0.508
	Random Forest	**0.641**	**0.588**	0.582
	SVC	0.591	0.536	**0.590**
90%	Dummy	0.291	0.367	0.413
	Decision tree	0.478	0.386	0.456
	Random Forest	**0.624**	0.495	**0.576**
	SVC	0.476	**0.503**	0.558

Table 3: Classification accuracy (see also Table 2) in the second experiment. We use all CMOPs for training and only the baseline CMOPs for testing. Numbers in bold denote the best model performance, while highlighted cells correspond to cases, for which the classification accuracy improved when including the additional CMOPs into the training data.

Targets	Classifier	NSGA-III	MOEA/D	C-TAEA
50%	Dummy	0.875	0.860	0.862
	Decision tree	**1.000**	0.996	**1.000**
	Random Forest	0.978	**1.000**	**1.000**
	SVC	0.916	0.916	0.916
60%	Dummy	0.247	0.268	0.251
	Decision tree	0.513	0.481	0.515
	Random Forest	**0.612**	**0.620**	**0.612**
	SVC	0.331	0.387	0.262
70%	Dummy	0.215	0.211	0.281
	Decision tree	0.389	0.555	0.504
	Random Forest	**0.606**	**0.563**	0.513
	SVC	0.415	0.387	**0.526**
80%	Dummy	0.325	0.306	0.371
	Decision tree	0.483	0.512	0.540
	Random Forest	0.605	**0.558**	0.550
	SVC	**0.609**	0.553	**0.607**
90%	Dummy	0.361	0.423	0.479
	Decision tree	0.400	0.462	0.354
	Random Forest	**0.526**	**0.543**	**0.565**
	SVC	0.475	0.486	0.542

on which NSGA-III performed best and 14 problems on which NSGA-III and MOEA/D performed better than C-TAEA.

The algorithm performances for 60% or 70% of the targets did not differ much on the baseline CMOPs. However, a much wider spectrum of performances could be noticed on the additional problems, where $188+55+42 = 285$ of the CMOPs for 60% of the targets, and $349 + 131 + 8 = 488$ of the CMOPs for 70% of the targets were best solved by one algorithm.

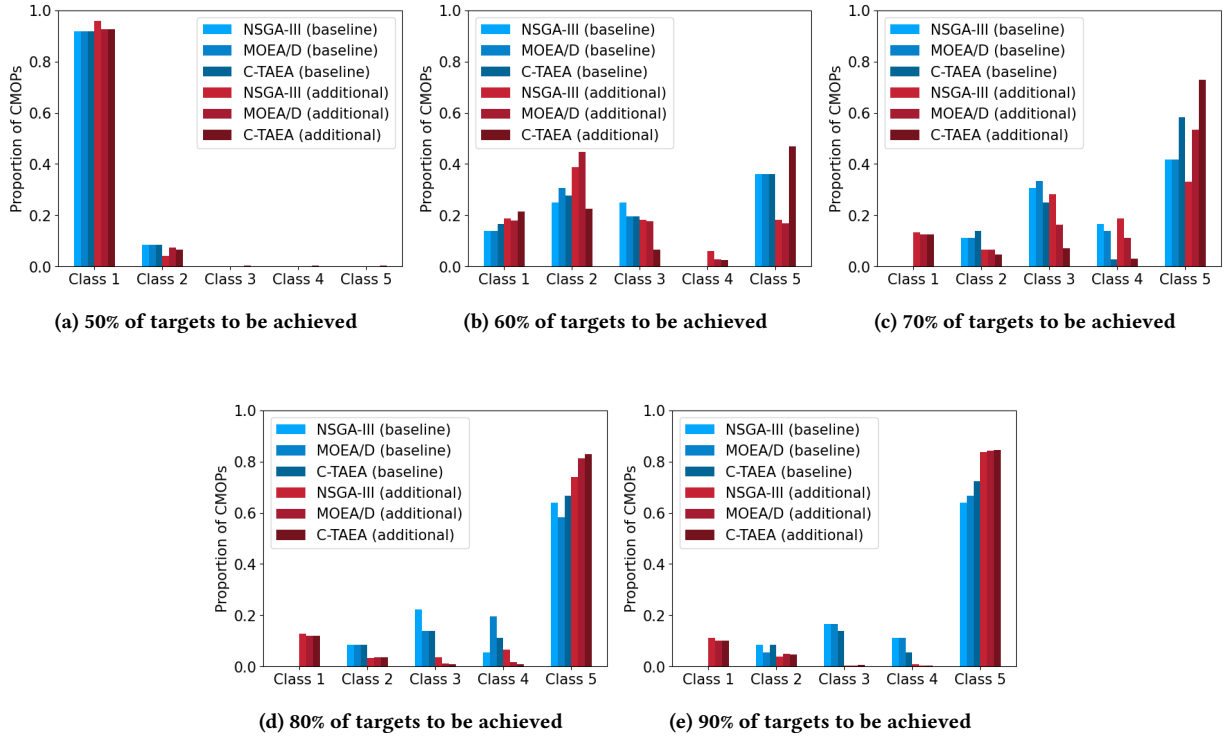
The assigned evaluation classes for achieving 80% and 90% of the targets were not so diverse for the baseline CMOPs or the additional CMOPs. A reason for this can be found in Figure 2, where almost none of the algorithms achieved 80% and 90% of the targets (they were mostly assigned to evaluation Class 5). However, in these cases, there were still CMOPs in which one algorithm performed best. More specifically, there are $2 + 3 = 5$ baseline CMOPs for 80% and $2 + 6 = 8$ baseline CMOPs 90% of the targets for which one algorithm performed best, and $116+18+7 = 141$ and $30+11+7 = 48$ additional CMOPs for 80% and 90% of the targets respectively, in which one algorithm performed best.

6 DISCUSSION

The evaluation class distributions presented in Figure 2 show how efficiently the algorithms achieve the given targets. We can see that for most CMOPs, 50% of the targets are achieved in the first generation, while 80% and 90% of the targets are rarely achieved. For 80% and 90% of the targets, the proportion of the additional CMOPs on which the algorithms never achieve the targets is higher than the proportion of such baseline CMOPs. This shows that for the optimization algorithms, converging close to the Pareto front on the additional CMOPs is much harder than doing so on the baseline CMOPs. In addition, there are some additional CMOPs, on which the algorithms achieve 80% and 90% of the targets in the first generation, while there are no such instances of baseline CMOPs.

These findings show how different in difficulty the baseline CMOPs and the additional CMOPs are. Moreover, as the targets are getting harder to be achieved, the differences between CMOPs also increase. As a consequence, the differences between the baseline CMOPs and the additional CMOPs are more present in the higher target percentages. As a result of this, the classification accuracy drops when using the additional CMOPs in the training data.

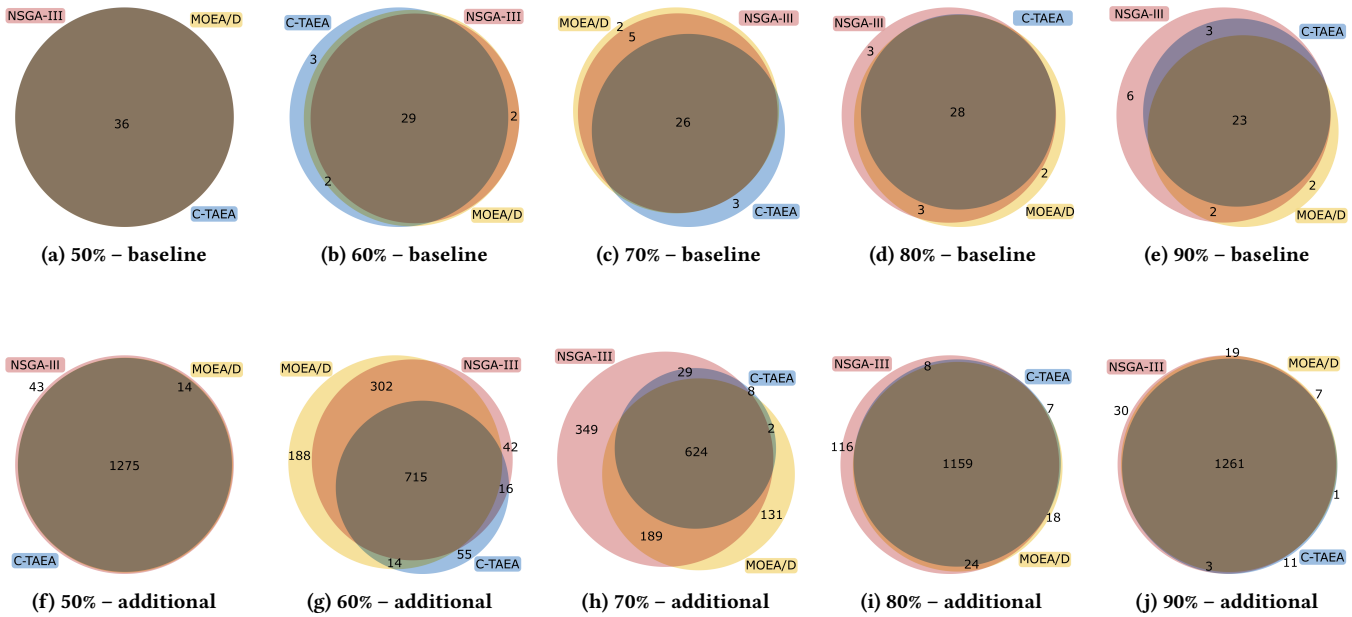
Analyzing the differences in algorithm performance presented in Figure 3, we notice bigger differences among algorithms in the additional CMOPs than in the baseline CMOPs. This applies count-wise for all percentages of targets, as well as proportion-wise for achieving 60% and 70% of the targets. The differences in algorithm performance on the additional CMOPs show that the problem diversity inside this problem set is greater than the baseline CMOP set. Thus, exploring other ways to combine constrained single-objective



(a) 50% of targets to be achieved (b) 60% of targets to be achieved (c) 70% of targets to be achieved

(d) 80% of targets to be achieved (e) 90% of targets to be achieved

Figure 2: Percentage of CMOPs belonging to a given evaluation class per algorithm. The three blue bars represent the proportion of the 36 baseline CMOPs, while the three red bars represent the proportion of the 1296 additional CMOPs.



(a) 50% – baseline (b) 60% – baseline (c) 70% – baseline (d) 80% – baseline (e) 90% – baseline

(f) 50% – additional (g) 60% – additional (h) 70% – additional (i) 80% – additional (j) 90% – additional

Figure 3: Area-proportional Venn diagrams showing the number of CMOPs on which each algorithm performs best. In each subplot, the pink, yellow, and blue circles represent the CMOPs for which the algorithms NSGA-III, MOEA/D, and C-TAEA, respectively, are faster in reaching the given percentage of targets. The intersections of circles represent the CMOPs on which multiple algorithms perform similarly (for example, see (g), the purple area represents the CMOPs on which both NSGA-III and C-TAEA perform best).

problems into CMOPs is a relevant approach forward to creating a much needed CMOP benchmark.

In these experiments, we separately predicted algorithm performance for the three algorithms. However, when dealing with a real-world problem, it would be most useful to have a method that suggests the most appropriate algorithm for that problem, based on the ELA features. A method like this was previously not possible due to the small number of available CMOPs, and the similar performance of the algorithms on them. By using the additional CMOPs as introduced in this work, we can get a larger set of CMOPs, on which algorithms perform differently. Consequently, we might want to redefine the algorithm performance prediction question into a more natural one – which algorithm should we use when dealing with a new CMOP (the algorithm selection task)?

7 CONCLUSION

In this work, we made predictions about the performance of three well-known multiobjective optimization algorithms on benchmark CMOPs. For this purpose, we used ELA features designed specifically for CMOPs as input to three classical machine learning algorithms. For a fair evaluation, we used the leave-one-problem-out evaluation methodology, where data from one CMOP is used for testing and the data from the rest of the CMOPs is used for training.

To analyze algorithm performance, we set different performance indicator values (targets) for the algorithms to achieve. These were then used to make predictions about the number of evaluations an algorithm requires to achieve a proportion of the targets. The percentages of targets that the algorithm needs to achieve were 50%, 60%, 70%, 80%, and 90%. In addition to this, to simplify the prediction task, we categorized the number of evaluations into five evaluation classes, with the last class representing the case where none of the specified percentages of targets were achieved.

We focused on two distinct experiments. In the first experiment, we applied the standard approach for algorithm performance prediction, where we used publicly available (baseline) CMOPs for training and testing the classifier. In the second experiment, we added to the classifier training set additional bi-objective CMOPs created by using the functions of the `bbob` and `bbob-constrained` benchmark suites as the two objectives. The constraints of the created CMOPs were the same as those in the corresponding `bbob-constrained` problems. Because of the large number of possible problem combinations (which considerably increases with more objectives), we decided to focus only on bi-objective 2D CMOPs in this initial study.

The comparison between the results of the two experiments has shown when and how much using the additional CMOPs helps in predicting algorithm performance on the baseline CMOPs. Such a comparison was made possible by using only the baseline CMOPs as test data in both experiments.

It was observed that, when training on all CMOPs, the predictions of the evaluation classes were improved for lower percentages of targets. However, for higher percentages of targets (like 80% and 90%), better results were obtained when training only on the baseline CMOPs. As an explanation for this, we presented the distribution of the evaluation classes and showed that for most of the additional CMOPs, the algorithms never reached the more challenging targets. This shows that the additional CMOPs are more difficult to solve than the baseline CMOPs, and because of this, they cannot be used to improve the predictions of the baseline CMOPs for the more challenging targets.

Additionally, we analyzed how many CMOPs have a single best-performing algorithm. The results showed that, on the lower target percentages, there is much more algorithm performance diversity in the additional CMOPs than in the baseline CMOPs. Also, the number of additional CMOPs for which a single algorithm performs best is now sufficiently large to redefine the algorithm performance prediction task into an algorithm selection task, a task which is more meaningful when dealing with a real-world CMOP.

In future work, we plan to improve the set of additional CMOPs by analyzing a larger number of problem combinations. The goal is to collect a set of additional CMOPs with proportional problem difficulties. However, to achieve this, the main obstacle is still the long time required for running the algorithms on a large number of problem combinations. To overcome this obstacle, we will consider two possible options. The first one is to use the ELA features for CMOPs to select the most diverse problem combinations. The second option is to use the publicly available performances of single-objective algorithms to select single-objective problems on which the algorithms performed differently, and combine them to create a set of CMOPs that will likely have diverse algorithm performance. Finally, we also plan to analyze CMOPs of higher dimensions in future studies.

ACKNOWLEDGMENTS

The authors acknowledge the financial support from the Slovenian Research and Innovation Agency (young researcher program, research core funding No. P2-0209, and project No. N2-0254 "Constrained Multiobjective Optimization Based on Problem Landscape Analysis"). This publication is also based upon work from COST Action CA22137 "Randomised Optimisation Algorithms Research Network" (ROAR-NET), supported by COST (European Cooperation in Science and Technology).

REFERENCES

[1] Hanan Alsouly, Michael Kirley, and Mario Andrés Muñoz. 2023. An instance space analysis of constrained multi-objective optimization problems. *IEEE Transactions on Evolutionary Computation* 27, 5 (2023), 1427–1439.
[2] Andrejaana Andova, Tobias Benecke, Harald Ludwig, and Tea Tušar. 2023. Towards constructing a suite of multi-objective optimization problems with diverse landscapes. In *26th European Conference on Applications of Evolutionary Computation, EvoApplications 2023*. Springer, Cham, 442–457.
[3] Andrejaana Andova, Aljoša Vodopija, Jordan Cork, Tea Tušar, and Bogdan Filipič. 2023. An attempt at predicting algorithm performance on constrained multiobjective optimization problems. In *Slovenian Conference on Artificial Intelligence: Proceedings of the 26th International Multiconference Information Society, IS 2023*. Jožef Stefan Institute, Ljubljana, 44–47.
[4] To Thanh Binh and Ulrich Korn. 1997. Mobes: A multiobjective evolution strategy for constrained optimization problems. In *Proceedings of the 3rd International Mendel Conference on Genetic Algorithms, MENDEL '97*. 176–182.
[5] Leo Breiman. 2001. Random forests. *Machine Learning* 45 (2001), 5–32.
[6] Dimo Brockhoff, Anne Auger, Nikolaus Hansen, and Tea Tušar. 2022. Using well-understood single-objective functions in multiobjective black-box optimization test suites. *Evolutionary Computation* 30, 2 (2022), 165–193.
[7] Kalyanmoy Deb, Amrit Pratap, and T Meyarivan. 2001. Constrained test problems for multi-objective evolutionary optimization. In *International Conference on Evolutionary Multi-criterion Optimization, EMO 2001*. Springer, Berlin, 284–298.
[8] Konstantin Dietrich and Olaf Mersmann. 2022. Increasing the diversity of benchmark function sets through affine recombination. In *International Conference on Parallel Problem Solving from Nature (PPSN)*. Springer, Cham, 590–602.
[9] Zhun Fan, Wenji Li, Xinye Cai, Han Huang, Yi Fang, Yugen You, Jiajie Mo, Caimin Wei, and Erik Goodman. 2019. An improved epsilon constraint-handling method

in MOEA/D for CMOPs with large infeasible regions. *Soft Computing* 23 (2019), 12491–12510.

[10] Zhun Fan, Wenji Li, Xinye Cai, Hui Li, Caimin Wei, Qingfu Zhang, Kalyanmoy Deb, and Erik Goodman. 2020. Difficulty adjustable and scalable constrained multiobjective test problem toolkit. *Evolutionary Computation* 28, 3 (2020), 339–378.

[11] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. 2009. *Real-parameter black-box optimization benchmarking 2010: Presentation of the noiseless functions.* Technical Report 2009/20. Research Center PPE. http://coco.gforge.inria.fr/downloads/download16.00/bbobdocfunctions.pdf Version from February 2019.

[12] C. M. Fonseca. 1995. *Multiobjective genetic algorithms with application to control engineering problems.* Ph. D. Dissertation. University of Sheffield.

[13] Andreia P Guerreiro, Carlos M Fonseca, and Luís Paquete. 2021. The hypervolume indicator: Computational problems and algorithms. *Comput. Surveys* 54, 6 (2021), 662–671.

[14] Nikolaus Hansen, Anne Auger, Dimo Brockhoff, and Tea Tušar. 2022. Anytime performance assessment in blackbox optimization benchmarking. *IEEE Transactions on Evolutionary Computation* 26, 6 (2022), 1293–1305.

[15] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. 2021. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods Software* 36, 1 (2021), 114–144. https://doi.org/10.1080/10556788.2020.1808977

[16] Christian Hanster and Pascal Kerschke. 2017. flaccogui: Exploratory landscape analysis for everyone. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '17.* ACM, New York, NY, USA, 1215–1222.

[17] Tim Hulsen. 2022. DeepVenn – a web application for the creation of area-proportional Venn diagrams using the deep learning framework Tensorflow. js. arXiv:2210.04597 [cs.HC]

[18] Himanshu Jain and Kalyanmoy Deb. 2013. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation* 18, 4 (2013), 602–622.

[19] Pascal Kerschke and Heike Trautmann. 2019. Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. *Evolutionary Computation* 27, 1 (2019), 99–127.

[20] Ke Li, Renzhi Chen, Guangtao Fu, and Xin Yao. 2018. Two-archive evolutionary algorithm for constrained multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 23, 2 (2018), 303–315.

[21] Arnaud Liefooghe, Fabio Daolio, Sébastien Verel, Bilel Derbel, Hernan Aguirre, and Kiyoshi Tanaka. 2020. Landscape-aware performance prediction for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation* 24, 6 (2020), 1063–1077.

[22] Wei-Yin Loh. 2011. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1, 1 (2011), 14–23.

[23] Zhongwei Ma and Yong Wang. 2019. Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons. *IEEE Transactions on Evolutionary Computation* 23, 6 (2019), 972–986.

[24] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '11.* ACM, New York, NY, USA, 829–836.

[25] 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. https://www.jmlr.org/papers/v12/pedregosa11a.html

[26] John C. Platt. 2000. Probabilities for SV Machines. In *Advances in Large Margin Classifiers*, Alexander J. Smola, Peter L. Bartlett, Bernhard Schölkopf, and Dale Schuurmans (Eds.). MIT Press, 61–73.

[27] Raphael Patrick Prager, Moritz Vinzent Seiler, Heike Trautmann, and Pascal Kerschke. 2022. 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, 3–17.

[28] N. Srinivas and Kalyanmoy Deb. 1994. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computing* 2, 3 (1994), 221–248. https://doi.org/10.1162/evco.1994.2.3.221

[29] M. Tanaka, H. Watanabe, Y. Furukawa, and T. Tanino. 1995. GA-based decision support system for multicriteria optimization. In *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, Vol. 2. IEEE, 1556–1561.

[30] Diederick Vermetten, Furong Ye, and Carola Doerr. 2023. Using affine combinations of BBOB problems for performance assessment. arXiv:2303.04573 [cs.NE]

[31] Aljoša Vodopija, Tea Tušar, and Bogdan Filipič. 2024. Characterization of constrained continuous multiobjective optimization problems: A performance space perspective. *IEEE Transactions on Evolutionary Computation* (2024). https://doi.org/10.1109/TEVC.2024.3366659 (early access).

[32] Aljoša Vodopija, Tea Tušar, and Bogdan Filipič. 2022. Characterization of constrained continuous multiobjective optimization problems: A feature space perspective. *Information Sciences* 607 (2022), 244–262.