Predicting Algorithm Performance in Constrained Multiobjective Optimization: A Tough Nut to Crack

Andrejaana Andova[1,2(✉)], Jordan N. Cork[1,2], Aljoša Vodopija[1,2], Tea Tušar[1,2], and Bogdan Filipič[1,2]

[1] Jožef Stefan Institute, Ljubljana, Slovenia
{andrejaana.andova,jordan.cork,aljosa.vodopija,tea.tusar,bogdan.filipic}@ijs.si
[2] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia

Abstract. Predicting algorithm performance is crucial for selecting the best performing algorithm for a given optimization problem. While some research on this topic has been done for single-objective optimization, it is still largely unexplored for constrained multiobjective optimization. In this work, we study two methodologies as candidates for predicting algorithm performance on 2D constrained multiobjective optimization problems. The first one consists of using state-of-the-art exploratory landscape analysis (ELA) features, designed specifically for constrained multiobjective optimization, as input to classical machine learning methods, and applying the resulting models to predict the performance classes. As an alternative methodology, we analyze an end-to-end deep neural network trained to predict algorithm performance from a suitable problem representation, without relying on ELA features. The experimental results obtained on benchmark problems with three multiobjective optimizers show that neither of the two methodologies is capable of substantially outperforming a dummy classifier. This suggests that, with the current benchmark problems and ELA features, predicting algorithm performance in constrained multiobjective optimization remains a challenge.

Keywords: Constrained multiobjective optimization · Exploratory landscape analysis · Algorithm performance prediction · Empirical cumulative distribution function · Machine learning · Deep learning

1 Introduction

When attempting to solve an optimization problem, the choice of which optimization algorithm to use is crucial for obtaining satisfying results in a limited time. It is, therefore, necessary to develop a method that identifies which algorithm performs best on a particular optimization problem. The task of selecting a single algorithm that performs best for a given optimization problem is called the algorithm selection task.

S. Smith et al. (Eds.): EvoApplications 2024, LNCS 14635, pp. 310–325, 2024.
https://doi.org/10.1007/978-3-031-56855-8_19

Solving an algorithm selection task requires a collection of algorithms from which to choose. It also requires a collection of diverse problems, which elicit different performance out of the algorithms. Constrained multiobjective problems (CMOPs) are both lesser in quantity and diversity and greater in complexity than unconstrained and/or single objective problems. Therefore, solving the constrained multiobjective algorithm selection task is an ambitious goal. As a first step towards solving it, we aim to develop a method for predicting algorithm performance on a given CMOP.

In recent years, many researchers have tried to predict algorithm performance [21,28]. They generally do so by extracting exploratory landscape analysis (ELA) features from a population of solutions. These are then used as input to a machine learning classifier, which identifies the optimization algorithm that performs best on the given problem. Many ELA features have been proposed for single-objective optimization, and the package `flacco` [11] contains a broad collection of these. However, ELA features for more complex problems, like CMOPs, are still under development, with only a few related works [2,15,30]. This adds to the difficulty of predicting algorithm performance on these problems.

In a previous work [3], we tried to predict algorithm performance on CMOPs by using the state-of-the-art collection of CMOP ELA features proposed in [2]. These features were used as inputs into classical machine learning regression models. We attempted to predict algorithm performance on three benchmark suites, for 2D, 3D, and 5D CMOPs. The target of our prediction task was the area under an algorithm performance curve (explained in Sect. 2.3). However, the obtained results were not encouraging and, therefore, we are trying to improve upon them. In this work, we have increased the number of CMOPs used in the learning process, changed the prediction target and utilized an end-to-end deep neural network (DNN) methodology that does not use ELA features.

The paper is further organized as follows. In Sect. 2, we introduce the background of our study. In Sect. 3, we explain the applied methodology. In Sect. 4 we present the experimental setup and, in Sect. 5, the obtained results. Finally, in Sect. 6, we provide a conclusion and outline ideas for future work.

2 Background

In this section, we introduce constrained multiobjective optimization, explain ELA for this kind of optimization, present the recently proposed performance indicator specifically developed for CMOPs, and outline deep neural networks.

2.1 Constrained Multiobjective Optimization

A CMOP is 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*, respectively. M is the number of objectives, and J and K are the number of inequality and equality constraints, respectively.

A solution $\mathbf{x}$ is *feasible*, if it satisfies all constraints, $g_j(\mathbf{x}) \leq 0$, for $j = 1, \ldots, J$ and $h_k(\mathbf{x}) = 0$, for $k = 1, \ldots, K$. A feasible solution $\mathbf{x}$ is said to *dominate* 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 solution* if there exists no feasible solution $\mathbf{x} \in S$ that dominates $\mathbf{x}^*$. All feasible solutions constitute the *feasible region* F. 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*, $P_o = \{f(\mathbf{x}) \mid \mathbf{x} \in S_o\}$.

2.2 Exploratory Landscape Analysis for Constrained Multiobjective Optimization

ELA is a methodology whereby features of an optimization problem are extracted from a sample of solutions [19]. These features are generally expertly designed statistical relations between solutions. While many ELA feature sets have been designed for single-objective optimization problems, only a few exist for CMOPs.

For CMOPs, state-of-the-art features were collected by Alsouly et al. [2]. They proposed additional features on top of the fast-computing features for CMOPs from the related work. The combined set of features is divided into three groups that describe: the multiobjective landscape, the violation landscape, and a combination of the two – the multiobjective violation landscape.

Features describing the objectives and their internal relations belong to the *multiobjective landscape group*. Global features in this group include the proportion of unconstrained Pareto optimal solutions, the hypervolume of the unconstrained Pareto front, and the correlation between the objective values, among others. Statistics on the distance between random walk neighbors in the objective space make up the random walk features.

Features describing the problem constraints belong to the *violation landscape group*. Global features in this group are devoted to global constraint violation statistics, while the random walk features consist of constraint violation statistics between random walk neighbors.

Features describing the relations between the objective and the constraints belong to the *multiobjective violation landscape group*. Global features include the proportion of feasible solutions, the proportion of Pareto optimal solutions, the hypervolume, statistics on the correlations between objectives and constraints, and others. Statistics on the dominance relations between random walk neighbors make up the random walk features.

2.3 Empirical Cumulative Distribution Functions

In constrained multiobjective optimization, there is a drawback to using the hypervolume of feasible solutions as the quality indicator, because it does not

record algorithm performance until feasible solutions are reached. However, recently, [29] introduced a new quality indicator for constrained multiobjective optimization, I_{CMOP}, to address the gap in this area. The new indicator generalizes the hypervolume-based quality indicator $I_{\text{HV}+}$ from [10]. Notably, both $I_{\text{HV}+}$ and I_{CMOP} assume that low quality indicator values indicate better sets of solutions and vice versa. I_{CMOP} can be defined as follows:

1. When all solutions in the set are infeasible, the I_{CMOP} quality indicator takes on the smallest constraint violation of all solutions in the set, plus a threshold τ^*.
2. When the set contains at least one feasible solution, the quality indicator equals the value of $I_{\text{HV}+}$ bounded above by the threshold τ^*, i.e., it equals $\min\{I_{\text{HV}+}, \tau^*\}$.

The threshold value τ^* ensures that an infeasible solution will always be deemed worse than a feasible one.

Also, to be able to compare different CMOPs, one first needs to normalize the $I_{\text{HV}+}$ value and the constraint violation value, based on a sample of 100 solutions. The details of how this is done can be found in [29].

For algorithm performance measurement during the algorithm run, we track the number of function evaluations (*runtimes*) needed to reach a particular quality indicator value (*target*). This is carried out for a set of targets and the runtimes are visualized using the Empirical Cumulative Distribution Function (ECDF) [10]. The ECDF shows the proportion of targets achieved by the algorithm at a certain runtime and increases as the algorithm achieves further targets. The maximum value achievable by an algorithm is 1, meaning it reached all targets. One way to express algorithm performance in a single number is by computing the area under the curve of the ECDF – larger values correspond to a better/faster algorithm performance.

2.4 Deep Neural Networks

Deep Neural Networks (DNNs) are one of the most widely used prediction models at the moment. For more details on how they work, refer to [20]. Here, we briefly introduce the three DNN architectures used in our work. They are as follows:

- A *feedforward neural network* (FNN) is a standard deep neural network, consisting of layers whose neurons are fully connected to the neurons from the neighboring layers.
- *Convolutional neural networks* are DNNs consisting of convolutional layers followed by activation layers and, sometimes, pooling layers. They are most often used in computer vision, as they are good at describing the local properties of the images, using filters that can be of different sizes.
- An *autoencoder* is a DNN architecture that consists of an encoder and a decoder part. These parts are usually symmetrical, therefore, the input and the output of an autoencoder neural network have the same shape. The goal of this neural network architecture is to compress the data. Thus, the encoder

compresses the data, and the decoder decompresses it. Essentially, the autoencoder can also be an FNN or a convolutional neural network as long as it performs data compression.

3 Methodology

In this section, we present the methods applied in this study. First, we explain how the ECDF of the I_{CMOP} indicator was used to define three different classification tasks. We then describe various methods for solving these tasks – machine learning methods that predict algorithm performance based on ELA features and the newly proposed end-to-end DNN, which circumvents the ELA features by using the problem landscape samples directly.

3.1 Classification Tasks

The ECDF of the performance indicator, described in Sect. 2.3, shows the number of targets achieved at each evaluation step. As explained in [29], to compare targets between different CMOPs, we normalize the targets using a sample of 100 solutions, and we set $\tau^* = 1$. Also, the authors state that a good set of target precision values corresponds to $\tau^\epsilon = \tau^{\text{ref}} + \epsilon$, 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^{ref} is the hypervolume of the true Pareto front, or an approximation of it. We used the same for our target precision values.

In a previous work [3], we were predicting algorithm performance using the area under the curve of the ECDF. This turned out to be a very difficult regression task. Therefore, to alleviate it, this work makes two changes to the methodology: (1) instead of the area under the curve, we predict the number of evaluations needed to reach three chosen target proportions, and (2), we predict ranges of values instead of exact numbers, transforming a regression task into a classification one.

More specifically, the target proportions of interest are:

- The number of evaluations needed until a feasible solution is obtained, which due to the choice of targets, corresponds to satisfying 50% of the targets.
- The number of evaluations needed to satisfy 70% of the targets.
- The number of evaluations needed to satisfy 90% of the targets.

Predicting the exact number of evaluations needed to satisfy a given percentage of targets, is difficult. Additional challenges arise from the fact that an algorithm may never reach the most difficult targets on some of the problems, which then requires special handling of such cases. Because of this, we group the number of evaluations into classes and treat their prediction as a classification task, which is expected to be easier to solve.

The number of evaluations of interest depends on the experimental setup. In our case, we will be performing at most 24 000 evaluations and use algorithms with a population size of 200. Therefore, we form the following classes:

- Class 0: The goal is achieved between 1 and 200 evaluations (in the initial generation),
- Class 1: The goal is achieved between 201 and 2 000 evaluations,
- Class 2: The goal is achieved between 2 001 and 8 000 evaluations,
- Class 3: The goal is achieved between 8 001 and 24 000 evaluations,
- Class 4: The goal is never achieved.

3.2 Classical Machine Learning

For the machine learning part, we use the ELA features outlined in Sect. 2.2 as input to three classical machine learning algorithms – Decision Trees [16], Random Forest Classification [5], and C-Support Vector Classification (SVC) [23]. We also include a dummy model in the comparison, which predicts the most frequent class in the training data. We utilize the `scikit-learn` implementations of these methods with default parameter settings [22].

3.3 DNN

Inspired by developments in computer vision, we decided to test whether methodologies from that field could be used for algorithm performance prediction on CMOPs. Some experiments have already been done in single-objective optimization [24,26], but they did not show promising results compared to the results obtained by the well-developed ELA features for single-objective optimization.

For a proof of concept, we limit the dimensionality of the search and objective spaces to 2D. In this way, no additional manipulation, such as dimensionality reduction, is required. More specifically, in our approach, we treat the search space as an image, discretized into 32×32 pixels. Each pixel contains the red, green, and blue color components, representing the two objectives and the overall constraint violation, respectively.

Data Generation. To generate images of the search spaces, we use the following sampling technique. First, we divide the 2D search space into "pixels", by splitting each dimension of the search space into 32 equally sized intervals. Then, for each pixel, we randomly generate a solution within it, and use its objective values and the overall constraint violation value to assign the color to the pixel. A visual representation of a sample generated using this technique is shown in Fig. 1.

DNN Architecture. The architecture of the DNN is composed of a convolutional neural network autoencoder and an FNN. The encoder part of the autoencoder is used as input to the FNN, whose target is the prediction class defined in Sect. 3.1. The DNN architecture is shown in Fig. 2.

Each flat rectangle in the figure represents one layer of the DNN architecture. It contains information about the `keras` library [1] layer class that we used on

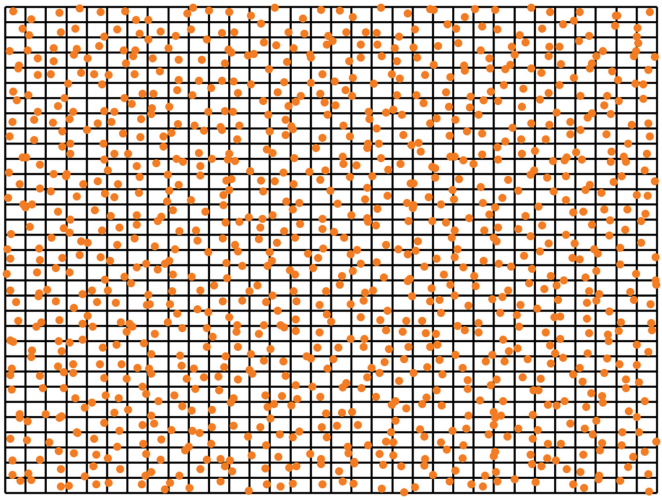

Fig. 1. An example sample of size 32×32 for the DNN.

the left side, and the shape of its input/output on the right side. For example, the first layer in the DNN is an InputLayer, and it takes as input images of size 32×32 with 3 channels.

The top part of the figure presents the encoder, which consists of three pairs of convolutional and max-pooling layers. The bottom part is divided into the decoder (on the left) and the FNN (on the right). The decoder is symmetrical to the encoder, whereas the FNN contains Dense and Dropout layers. The last layer in the FNN has an output of 5 neurons, each one assigned to one of the prediction classes presented in Sect. 3.1.

The idea behind this architecture is that, by providing the autoencoder with the same image as input and output, we force it to encode the input image so that the least amount of information is lost in the training process. The encoded part can be seen as landscape features that the autoencoder automatically extracts from the input data.

To cause the DNN to encode the properties that are useful for predicting algorithm performance, we use the encoded part as input to an FNN. Both parts of the DNN are trained simultaneously, with a combined loss function (mean absolute error for the decoder, and categorical cross-entropy for the FNN).

Data Preprocessing. The objectives and overall constraint violation have different value ranges across different problems. For this reason, as a preprocessing step, we normalize each of these functions. The min-max normalization procedure applies the normalization over all samples of a given problem, using the minimum and maximum value of the given objective. Furthermore, we normalize the constraints by assigning a 0 value to the feasible solutions, and a 1 value

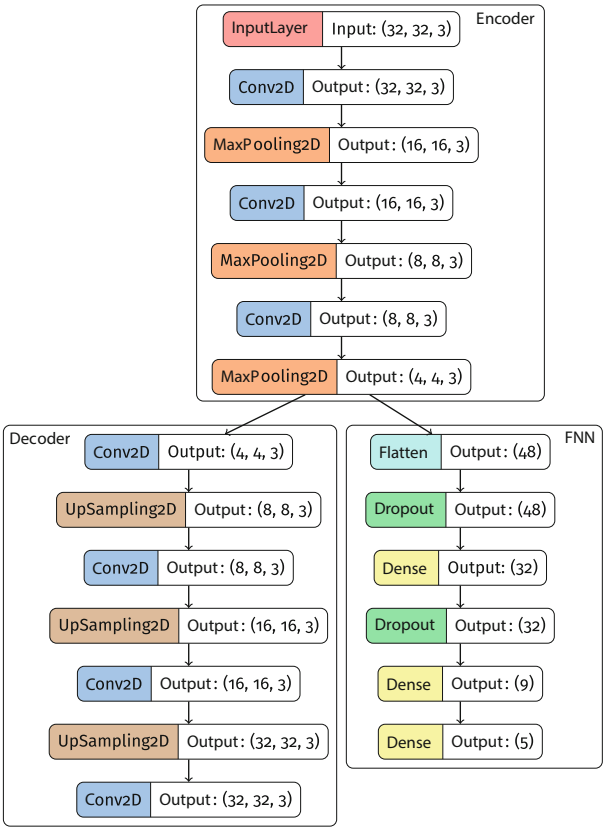

Fig. 2. The applied DNN architecture consisting of the encoder, decoder and FNN.

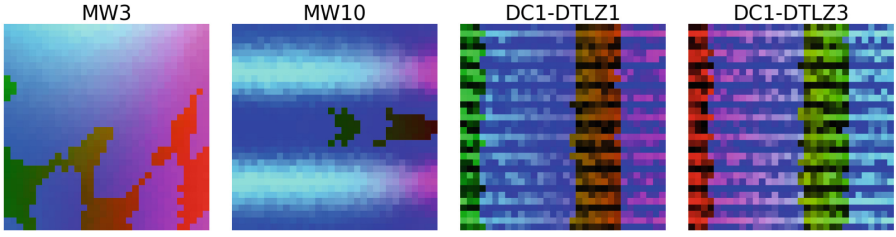

Fig. 3. Four example CMOP inputs, as images, for use with the DNN. Red represents the value of the first objective function, green the value of the second objective function, and blue the constraint violation. Prior to this encoding, the objective values are normalized using the minimum and maximum objective values of the problem samples. (Color figure online)

to the infeasible solutions. Example visualizations of several input images from different CMOPs are presented in Fig. 3.

Note that there are many ways to normalize the functions. For example, one other possibility to normalize the overall constraint violation is to use its the minimum and maximum values. However, after some preliminary experimentation with different normalization techniques, we found that the obtained algorithm performance prediction results were comparable. Thus, in this paper, we only present the results derived from the normalization techniques described in the paragraph above.

In constrained multiobjective optimization, the order of the objectives should not be important. Thus, we generate two images for each input sample – one where the first objective is assigned the red color and the second objective green, and another image where the ordering is reversed. The blue color always encodes the constraint violation.

DNN Settings. We used the ReLu activation function for each hidden layer in the DNN. We set the batch size to 1 000, the number of epochs to 100, and we used the Adam optimizer [13] with a learning rate of 0.0001.

4 Experimental Setup

Our work is focused on bi-objective CMOPs with 2D search spaces. We used six benchmark suites in the experiments: MW [17], C-DTLZ [12], CTP [7], DAS-CMOP [9], and DC-DTLZ [14], as well as three individual benchmark problems: BNH [4], TNK [27], and SRN [25]. The total number of CMOPs with two variables and two objectives from these suites is 36 (see Table 1 for a break-down over problem suites).

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

MW	C-DTLZ	CTP	DAS-CMOP	DC-DTLZ	BNH	TNK	SRN
8	5	8	6	6	1	1	1

For the purpose of predicting algorithm performance, three multiobjective optimization algorithms were tested, each with a different constraint handling technique. These algorithms were NSGA-III [12], MOEA/D-IEpsilon [8], and C-TAEA [14]. To handle the variation of the results due to the stochastic nature of the algorithms, 31 runs of each algorithm were conducted on each problem. With this approach, algorithm performance can be estimated more accurately. To extract the target classes, we used the mean of the ECDF values over all 31 algorithm runs. Additionally, we applied the same population size and number of generations to all algorithms, allowing for a fair comparison of results. The population size was set to 200, and the number of generations to 120. To generate

reference vectors for NSGA-III and MOEA/D-IEpsilon, we used the Das-Denis approach [6]. The number of reference vectors was 200 for each algorithm.

The ELA features were calculated stochastically, whereby a different sample of solutions was selected each time the feature calculation is begun. This was dealt with by creating 100 samples using Latin hypercube sampling, which resulted in 100 sets of features (i.e., learning instances) for each problem. Similarly, we created 100 samples per problem for the DNN method using the sampling described in Sect. 3.3.

For easier reproducibility of the stochastic learning models, we report that the random number generator was seeded with the value of ten to obtain the results in the following section. Moreover, experiments with alternative seeds resulted in comparable results.

To evaluate the performance of each classifier, we used the leave-one-problem-out evaluation methodology. In this approach, no information about the target problem is available in the training data. Thus, all instances of a problem are used as test data, and the instances from the rest of the problems as training data. This process is repeated for each problem and the average mean absolute error is used as an evaluation metric.

5 Results

The classification accuracy for the desired target percentages for all learning methods is presented in Table 2. From the results we can see that none of the learning models drastically outperforms the dummy classifier. The only exception is the Random forest model. This performs better than the dummy classifier in most cases, except for MOEA/D and C-TAEA when predicting the evaluation class with at least 90% of the targets achieved.

To analyze more thoroughly the predictions by the Random forest model, we provide its confusion matrices for all three classification tasks in Fig. 4. In addition, Fig. 5 shows problem samples in the ELA feature space, reduced to 2D using the t-distributed stochastic neighbor embedding (t-SNE) method [18].

In Table 2, we can see that the classification accuracy is the same across all optimization algorithms when tackling the first classification task, that being to achieve 50% of the targets, i.e., to reach the border between the infeasible and feasible regions. An explanation for this can be derived from the confusion matrices in Fig. 4. These show that most of the optimization algorithms find a feasible solution in the initial population. They are, therefore, labeled with class 0. Otherwise, they achieve a feasible solution in at most 2 000 evaluations.

As shown in Fig. 5, with t-SNE dimension reduction, the instances from the same CMOP form clusters. This means the ELA features from the same problem do not provide the diversity required by the machine learning models. Consequently, during prediction, the learning models usually have either a 100% or 0% accuracy for a given CMOP. This is manifested in the nearly fully rounded results present in the confusion matrices in Fig. 4. A similar behavior can be observed for the DNN method, although this method does not rely on ELA features.

Table 2. Classification accuracy of the learning models predicting the algorithm performance classes.

Targets	Classifier	NSGA-III	MOEA/D	C-TAEA
50%	Dummy	0.916	0.916	0.916
	Decision tree	0.916	0.916	0.916
	Random Forest	0.944	0.944	0.944
	SVC	0.888	0.888	0.888
	DNN	0.916	0.916	0.916
70%	Dummy	0.416	0.416	0.583
	Decision tree	0.446	0.376	0.658
	Random Forest	0.576	0.549	0.674
	SVC	0.406	0.406	0.588
	DNN	0.381	0.304	0.583
90%	Dummy	0.638	0.666	0.722
	Decision tree	0.581	0.668	0.687
	Random Forest	0.677	0.638	0.703
	SVC	0.623	0.650	0.727
	DNN	0.638	0.666	0.722

The DNN, proposed as a novelty in this work, unfortunately never outperforms the dummy classifier and sometimes performs even worse than it, although the loss was observed to decrease during training. Worse performance is, for example, seen when predicting the number of evaluations needed by the NSGA-III and MOEA/D algorithms to achieve 70% of the targets. A reason for the poor DNN performance could be that we used only 35 CMOPs for training. Although we generated 100 samples for each problem, this may still not provide enough diversity and the DNN is not able to learn the patterns of the search space. Namely, it is known that DNN's need huge amounts of data to learn adequately. The classical machine learning models, on the other hand, are designed to be able to handle small amounts of data, but, as stated before, their performance was not found to be promising either.

A reason for poor performance of the classical machine learning models on CMOPs could be that, just like the DNN, they also need more data (although probably less so than the DNN). The small number of CMOPs used for training is certainly a difficulty, but the similarity of some properties across different CMOPs is also a potential reason for the low prediction performance and (likely)

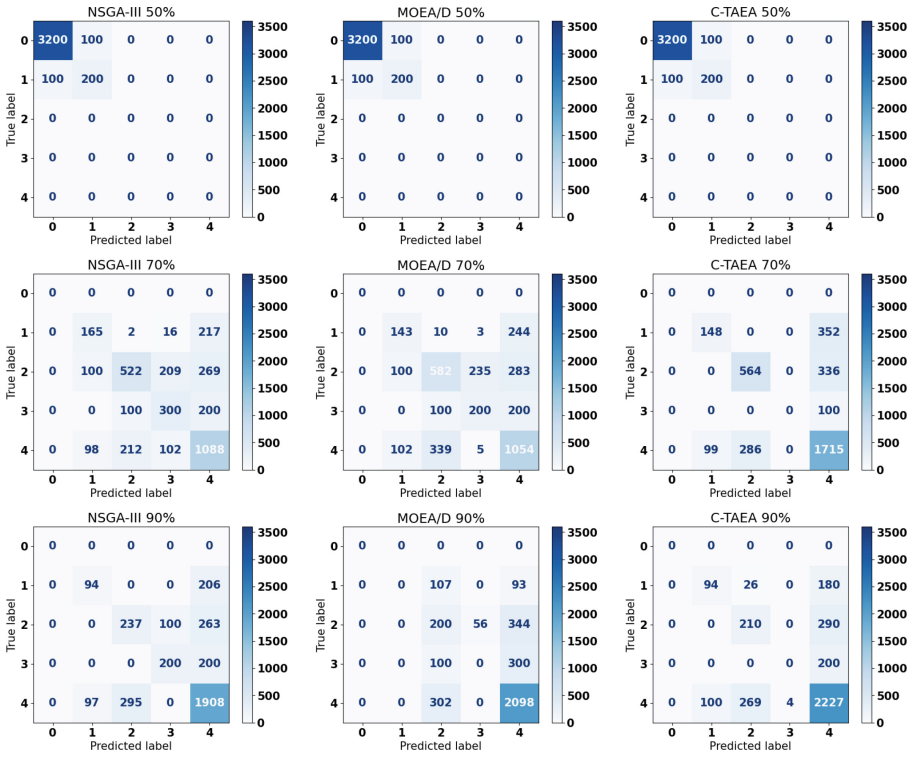

Fig. 4. Confusion matrices of the random forest models for the three desired target percentages. Each confusion matrix refers to the algorithm performance classes explained in Sect. 3.1 in more detail.

overfitting of the data. For example, as shown in Fig. 3, DC1-DTLZ1 and DC1-DTLZ3 have very similar landscapes, and, given that the order of objectives in CMOPs is insignificant, the red and the green sectors may be swapped.

Another reason for the poor performance of feature-based performance prediction might be the recency of research into ELA features for CMOPs. Possibly not all informative characteristics of CMOPs are included in the feature set, as of yet.

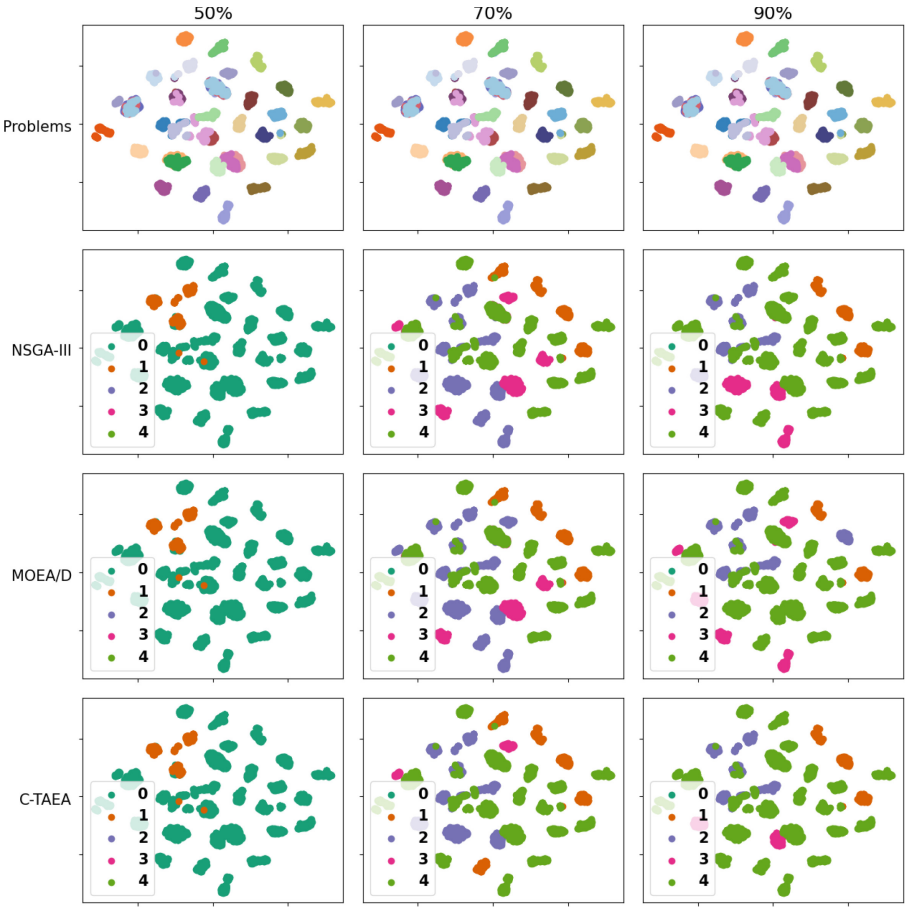

Fig. 5. Visualizations of ELA features for the three desired target percentages, reduced in dimensionality using the t-SNE method. The colors in the first row of the plots represent the problems included in the experiment. In the remaining rows, the colors identify the classes representing the number of evaluations needed to achieve a percentage of targets. (Color figure online)

6 Conclusion

In this work, we tried to improve upon our previous attempt at algorithm performance prediction for three widely used multiobjective optimization algorithms, NSGA-III, MOEA/D-IEpsilon, and C-TAEA, on 2D, 3D, and 5D CMOPs. Previously, we worked on predicting the area under the curve of the ECDF for the I_{CMOP} quality indicator proposed in [29]. We used classical machine learning regression models, whose inputs were the ELA features proposed in [2]. Unfortunately, the obtained results were not encouraging. Consequently, in this work, we focused on 2D CMOPs. We used a total of nine benchmark suites and problems,

which resulted in 36 CMOPs. This is significantly larger than in the previous work where only 13 were used. Furthermore, we changed the prediction task – in this work, we were predicting the number of evaluations needed to achieve 50%, 70%, and 90% of the ECDF targets. Moreover, because predicting the number of evaluations is a hard task, we discretized the number of evaluations needed into five classes.

The results from the previous work left questions as to whether the prediction performance was poor because of the small number of CMOPs used for training, or the underdevelopment of the CMOP ELA feature set. To eliminate the second issue, we proposed an end-to-end DNN, that does not include ELA features to predict algorithm performance. As far as we are aware, this is the first time an end-to-end DNN has been used to predict algorithm performance on CMOPs.

Unfortunately, the newly proposed method did not outperform the dummy prediction model. Nonetheless, the reason for this might be that using merely 36 CMOPs is not enough for training a DNN. Thus, this left us with the dilemma of poor algorithm performance prediction – are more CMOPs required to predict algorithm performance, or better ELA features? Moreover, the tested evolutionary algorithms performed comparably on the benchmark problems. This calls for involving a larger set of algorithms that would potentially show different performance.

In the future, we plan to extend our research on end-to-end DNNs for algorithm performance prediction by applying publicly available pretrained DNNs. The idea is to enhance the performance of the proposed architecture. This is a standard practice in deep learning when dealing with small datasets and thus, although none of the pretrained models was trained on problem landscapes, their learned patterns might still help with our prediction task.

Another way forward is to utilize a larger CMOP benchmark suite. This can be constructed by combining the objectives and constraints of constrained single-objective problems from various benchmark suites. This way, we could include a much larger number of CMOPs in the data, possibly helping both the classical machine learning methods and the deep learning methods better predict algorithm performance. A drawback to this approach is that running the algorithms 31 times for each problem combination would be a time-consuming task. It is possible, however, that already a small proportion of the problem combinations would contribute diversity to the extended benchmark suite.

Ideally, in future work, both the ideas stated above would be combined. Tests on an extended CMOP benchmark suite are sure to answer whether more CMOPs are needed to better predict algorithm performance, while the inclusion of knowledge from pretrained DNNs is likely to provide insights into the possibility of improving the current ELA features for CMOPs.

Acknowledgements. 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"). The publication is also based upon work from COST Action CA22137 "Randomised Optimisation Algorithms Research

Network" (ROAR-NET), supported by European Cooperation in Science and Technology (COST).

References

1. Keras. https://github.com/fchollet/keras (Accessed 27 September 2023)
2. Alsouly, H., Kirley, M., Muñoz, M.A.: An instance space analysis of constrained multi-objective optimization problems. IEEE Trans. Evol. Comput. **27**(5), 1427–1439 (2023). https://doi.org/10.1109/TEVC.2022.3208595
3. Andova, A., Vodopija, A., Cork, J., Tušar, T., Filipič, B.: 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 (2023)
4. Binh, T.T., Korn, U.: Mobes: a multiobjective evolution strategy for constrained optimization problems. In: Proceedings of the 3rd International Mendel Conference on Genetic Algorithms, MENDEL 1997, pp. 176–182 (1997)
5. Breiman, L.: Random forests. Mach. Learn. **45**, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
6. Das, I., Dennis, J.E.: Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J. Optim. **8**(3), 631–657 (1998). https://doi.org/10.1137/S1052623496307510
7. Deb, K., Pratap, A., Meyarivan, T.: Constrained test problems for multi-objective evolutionary optimization. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 284–298. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44719-9_20
8. Fan, Z., et al.: An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions. Soft. Comput. **23**, 12491–12510 (2019). https://doi.org/10.1007/s00500-019-03794-x
9. Fan, Z., et al.: Difficulty adjustable and scalable constrained multiobjective test problem toolkit. Evol. Comput. **28**(3), 339–378 (2020). https://doi.org/10.1162/evco_a_00259
10. Hansen, N., Auger, A., Brockhoff, D., Tušar, T.: Anytime performance assessment in blackbox optimization benchmarking. IEEE Trans. Evol. Comput. **26**(6), 1293–1305 (2022). https://doi.org/10.1109/TEVC.2022.3210897
11. Hanster, C., Kerschke, P.: flaccogui: exploratory landscape analysis for everyone. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2017, pp. 1215–1222. ACM (2017). https://doi.org/10.1145/3067695.3082477
12. Jain, H., Deb, K.: 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. Evol. Comput. **18**(4), 602–622 (2013). https://doi.org/10.1109/TEVC.2013.2281534
13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2014). https://doi.org/10.48550/arXiv.1412.6980 arXiv preprint arXiv:1412.6980
14. Li, K., Chen, R., Fu, G., Yao, X.: Two-archive evolutionary algorithm for constrained multiobjective optimization. IEEE Trans. Evol. Comput. **23**(2), 303–315 (2018). https://doi.org/10.1109/TEVC.2018.2855411
15. Liefooghe, A., Daolio, F., Verel, S., Derbel, B., Aguirre, H., Tanaka, K.: Landscape-aware performance prediction for evolutionary multiobjective optimization. IEEE

Trans. Evol. Comput. **24**(6), 1063–1077 (2019). https://doi.org/10.1109/TEVC.2019.2940828
16. Loh, W.Y.: Classification and regression trees. Wiley Interdisciplinary Rev. Data Mining Knowl. Dis. **1**(1), 14–23 (2011). https://doi.org/10.1002/widm.8
17. Ma, Z., Wang, Y.: Evolutionary constrained multiobjective optimization: test suite construction and performance comparisons. IEEE Trans. Evol. Comput. **23**(6), 972–986 (2019). https://doi.org/10.1109/TEVC.2019.2896967
18. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(11), 2579–2605 (2008). https://jmlr.org/papers/v9/vandermaaten08a.html
19. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 829–836. ACM (2011). https://doi.org/10.1145/2001576.2001690
20. Nielsen, M.A.: Neural Networks and Deep Learning. Determination Press (2015)
21. Nikolikj, A., Doerr, C., Eftimov, T.: Rf+clust for leave-one-problem-out performance prediction. In: Applications of Evolutionary Computation: 26th International Conference, pp. 285–301. Springer (2023). https://doi.org/10.1007/978-3-031-30229-9_19
22. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011). https://www.jmlr.org/papers/v12/pedregosa11a.html
23. Platt, J.C.: Probabilities for SV machines. In: Smola, A.J., Bartlett, P.L., Schölkopf, B., Schuurmans, D. (eds.) Advances in Large Margin Classifiers, pp. 61–73. MIT Press (2000)
24. Prager, R.P., Seiler, M.V., Trautmann, H., Kerschke, P.: 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 2022. pp. 3–17. Springer (2022). https://doi.org/10.1007/978-3-031-14714-2_1
25. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Comput. **2**(3), 221–248 (1994). https://doi.org/10.1162/evco.1994.2.3.221
26. van Stein, B., Long, F.X., Frenzel, M., Krause, P., Gitterle, M., Bäck, T.: Doe2vec: Deep-learning based features for exploratory landscape analysis. arXiv preprint arXiv:2304.01219 (2023). https://doi.org/10.48550/arXiv.2304.01219
27. Tanaka, M., Watanabe, H., Furukawa, Y., Tanino, T.: 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, pp. 1556–1561 (1995). https://doi.org/10.1109/ICSMC.1995.537993
28. Vermetten, D., Wang, H., Bäck, T., Doerr, C.: Towards dynamic algorithm selection for numerical black-box optimization: Investigating bbob as a use case. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2020. pp. 654–662. ACM (2020). https://doi.org/10.1145/3377930.3390189
29. Vodopija, A., Tušar, T., Filipič, B.: Characterization of constrained continuous multiobjective optimization problems: A performance space perspective. arXiv preprint arXiv:2302.02170 (2023). https://doi.org/10.48550/arXiv.2302.02170
30. Vodopija, A., Tušar, T., Filipič, B.: Characterization of constrained continuous multiobjective optimization problems: a feature space perspective. Inf. Sci. **607**, 244–262 (2022). https://doi.org/10.1016/j.ins.2022.05.106