2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

How Far Out of Distribution Can We Go With ELA Features and Still Be Able to Rank Algorithms?

1st Gašper Petelin
Computer Systems Department
Jožef Stefan Institute
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia
gasper.petelin@ijs.si

2nd Gjorgjina Cenikj
Computer Systems Department
Jožef Stefan Institute
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia
gjorgjina.cenikj@ijs.si

***Abstract*—Algorithm selection is a critical aspect of continuous black-box optimization, and various methods have been proposed to choose the most appropriate algorithm for a given problem. One commonly used approach involves employing Exploratory Landscape Analysis (ELA) features to represent optimization functions and training a machine-learning meta-model to perform algorithm selections based on these features. However, many meta-models trained on existing benchmarks suffer from limited generalizability. When faced with a new optimization function, these meta-models often struggle to select the most suitable algorithm, restricting their practical application. In this study, we investigate the generalizability of meta-models when tested on previously unseen functions that were not observed during training. Specifically, we train a meta-model on base COmparing Continuous Optimizers (COCO) functions and evaluate its performance on new functions derived as affine combinations between pairs of the base functions. Our findings demonstrate that the task of ranking algorithms becomes substantially more challenging when the functions differ from those encountered during meta-learning training. This indicates that the effectiveness of algorithm selection diminishes when confronted with problem instances that substantially deviate from the training distribution. In such scenarios, meta-models that use ELA features to predict algorithm ranks do not outperform mere predictions of the average algorithm ranks.**

***Index Terms*—algorithm selection, exploratory landscape analysis, evolutionary computation, black-box optimization**

I. Introduction

Continuous single-objective optimization (SOO) is a process of finding the optimal value of a single objective function in a continuous domain. It is a fundamental problem in optimization and has applications in many different domains [1]. A closely related problem in SOO is the automated algorithm selection (AS) problem [2] where one wants to select the best-performing optimization algorithm for a given problem. In previous research, this issue has received considerable attention [3]. Typically, the process of choosing algorithms involves utilizing specific properties or features of the problem at hand. Some examples of such features include Exploratory Landscape Analysis (ELA) [4], Topological Landscape Analysis (TLA) [5], and Deep learning-based features [6]. This is accompanied by a machine learning meta-model that given a set of features either predicts performance (regression task [7]) or determines the most appropriate algorithm from a given set (classification task [8]).

In the existing body of literature, algorithm selection is predominantly carried out by utilizing established benchmark suites like COmparing Continuous Optimizers (COCO) [9] and IEEE Congress on Evolutionary Computation (CEC) [10]. For our specific case, our focus will solely be on the COCO benchmark, which encompasses 24 classes of single-objective optimization problems. Each problem class within the benchmark consists of multiple instances, created by applying various transformations such as scaling or shifting. Essentially, the COCO collection comprises numerous problem instances, all belonging to one of the 24 problem classes with problem instances belonging to the same class usually having identical or extremely similar properties [11].

In the context of utilizing the COCO benchmark, two prevalent methods for algorithm selection and performance prediction during validation are known as "leave-one-instance-out" (LOIO) (see [12]) and "leave-one-problem-out" (LOPO) (see [3]). By utilizing LOPO validation, the process of automated algorithm selection involves creating a meta-model and evaluating it across various types of problems. Typically, this entails selecting specific instances from one problem class to form a test set, while the remaining instances from the other 23 problem classes are utilized to construct the meta-model. The LOPO validation for AS models can pose significant challenges due to the diverse properties of COCO problems [13]. Consequently, there may be instances during prediction where a problem with similar properties has not been previously encountered. Conversely, in contrast to LOPO validation, LOIO validation tends to be more lenient since the AS model is built using all problem classes. Under this validation approach, some instances from a particular problem class are employed for training, while others are used for testing. Consequently, during the evaluation of the model, functions with comparable properties have already been observed, simplifying the task of selecting the best algorithm or predicting performance. Further information and comparisons between these two strategies can be found in [3].

The objective of this research is to investigate the general-

izability of meta-models in algorithm selection when utilizing ELA features. Existing studies typically rely on either LOIO or LOPO strategies, which often result in two extreme scenarios: highly accurate meta-models that excel in algorithm selection under LOIO, and underperforming meta-models when the LOPO strategy is employed. Therefore, assessing the effectiveness of specific features and meta-models in algorithm selection becomes challenging. One strategy can be categorized as "extremely easy" because the training and testing instances are nearly identical, while the other strategy is considered "extremely hard" due to the complete dissimilarity between the prediction and training data, making learning a difficult task.

Our contribution: In this paper, we evaluate the ability of ELA features combined with meta-models in determining the appropriate algorithm for unfamiliar problems. Our aim is to construct a meta-model that can effectively map ELA features to algorithm ranks. Subsequently, we assess the performance of the meta-model on new problems generated by combining existing COCO functions. Our results indicate that, in the majority of cases, the meta-model's ability to accurately rank algorithms deteriorates substantially when applied to modified problems that were not encountered during training. This suggests that the current algorithm selection approach based on LOIO methodology is inadequate, as even minor variations in ELA features can result in meta-model producing poor rankings. Furthermore, this highlights the limitations of the LOIO methodology, as it can yield meta-models that perform exceptionally well but fail to generalize to novel problems they may encounter.

Outline: The structure of the remaining sections in the paper is outlined as follows: In Section II, we elaborate on the methodology employed to extract features from optimization functions and utilize them to rank algorithms. Section III shows the outcomes regarding the robustness of algorithms trained on a specific problem and their generalizability to new, unseen problems. Finally, Section IV provides concluding remarks for the paper.

Reproducibility: The code used to carry out the experiments can be found at the Gitlab repository https://repo.ijs.si/gpetelin/affine-ranking.

II. Methodology

In this section, we present a comprehensive explanation of the methodology employed in the research paper. The process involves several steps: *i)* computing ELA features and algorithm rankings for problem instances across all 24 base COCO problem classes; *ii)* training a model that takes as input ELA features and predicts algorithm rankings specifically for problem instances falling under the 24 base problem classes; *iii)* generating new problem instances by combining existing base COCO functions, calculating ELA features for these instances, and running the algorithm to get the ground truth ranks of optimization algorithms; *iv)* utilizing the model trained on the base problem classes to predict the rankings for the newly generated problems; *v)* comparing the ground truth rankings with predicted ones from out-of-distribution functions and assessing the extent to which we can go outside the distribution before the algorithm rankings become unreliable.

A. Affine Function Combinations and ELA Feature Representation

In this study, we make use of affine function transformations as defined in [14]. Specifically, we use the following formula to combine two functions from the COCO suite:

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

In this case $P_{a,b}$ represents the b-th problem instance belonging to the a-th COCO problem class. The $O_{a,b}$ represents the location of the optimum of the function $P_{a,b}$. All of the objective functions created by this equation have the optimal solution of the objective value at zero ($P_{a,b}(O_{a,b}) = 0.0$). Parameter α describes the mixing degree of two functions. To give an example, Figure 1 demonstrates the newly created 2D functions that are a combination of instances belonging to classes 1 and 22 with different values of α.

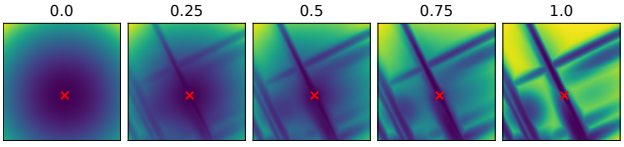

Fig. 1. The red mark indicates the global optimum on the landscape resulting from the combination of $P_{22,1}$ with $\alpha = 1$ and $P_{1,1}$ with $\alpha = 0$, for instance, 1 of both functions. The landscape evolves as α varies between 0.0 and 1.0.

In order to make the objective functions suitable for machine learning applications, we employ Exploratory Landscape Analysis (ELA) [4], [15] features (specifically the Dispersion, ELA Dist, ELA Level, ELA Meta, IC, NBC and PCA feature groups). ELA features are utilized to quantify specific properties of continuous optimization problems, including convexity, curvature, multimodality, and variable scaling. These features are derived from a set of low-level metrics computed using sampled points and corresponding function values within the problem domain. In this paper, we calculate ELA features by employing Latin Hypercube Sampling (LHS) with a sample size of $1000D$. In total, we use 62 ELA features to describe each problem instance. Keep in mind that this represents a relatively large sampling budget when calculating ELA features. However, in this instance, we aimed for the features to be as robust and reliable as possible.

B. Optimization Algorithm Portfolio

In our study we define portfolio of k optimization algorithms as $\mathcal{A} = \{a_1, \ldots, a_k\}$. The portfolio consists of five algorithms from the *pymoo* toolbox [16], which provides a standardized approach to access different optimization algorithms. The following optimization algorithms are assessed in

our study based on the availability of the code and popularity within the field of optimization:

- Genetic Algorithm (GA) [17]
- Differential Evolution (DE) [18]
- Particle Swarm Optimization (PSO) [19]
- Evolutionary Strategy (ES) [20]
- Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [21]

All the algorithm hyperparameters were configured using the default values defined in the *pymoo* framework version *0.5.0*. All algorithms were assigned a budget of 2000D function evaluations.

C. Algorithm Ranking Metric

A method for algorithm selection based on machine learning aims to identify a mapping that associates specific features with algorithms available in a given portfolio. Various metrics have already been proposed in this regard [3]. In our study, we employ the pairwise ranking error (PRE) to evaluate and compare the quality of different rankings. The pairwise error between two ranked lists is defined as follows:

$$PRE = \frac{1}{|\mathcal{A}|\,|\mathcal{A}|} \sum_{a_i \in \mathcal{A}} \sum_{a_j \in \mathcal{A}} r(a_j, a_i) \tag{2}$$

$$r(a_j, a_i) = \begin{cases} 0 & \text{if } r_p(a_i, a_j) = r_g(a_i, a_j) \\ 1 & \text{if } r_p(a_i, a_j) \neq r_g(a_i, a_j) \end{cases} \tag{3}$$

In this scenario, the function $r_p(a_i, a_j)$ yields -1, 0, or 1 based on whether the predicted rank of algorithm a_i is lower, equal, or higher than the rank of algorithm a_j, respectively. Similarly, the function $r_g(a_i, a_j)$ also provides the values -1, 0, or 1, but in this case, it considers the order of algorithms in relation to the ground truth, which is determined by executing the algorithm on a specific problem instance 30 times. To provide an example, let's consider a set of algorithms [GA, PSO, DE, CMA-ES, ES] and evaluate them three times on a specific problem instance. The obtained ranks for each run are as follows: [5, 2, 3, 1, 4], [5, 3, 2, 1, 4], [5, 2, 3, 1, 4]. By averaging these ranks across all three runs, we obtain the average ranks: [5, 2.33, 2.67, 1, 4]. This indicates that, based on these three runs, GA is ranked the worst while CMA-ES is ranked as the best algorithm. These average ranks are considered the ground truth for algorithm performance on a specific problem instance.

Now, let's assume a meta-model predicts algorithm ranks of [4.8, 2.63, 2.37, 1.1, 4.1]. With this prediction, all the algorithms are still ranked correctly, except for PSO and DE whose rankings differ from the ground truth. To measure the difference in rankings, we can calculate the pairwise error using equation (2). The computed pairwise error, in this case, is 0.1, since the predicted order is relatively close to the ground truth order of algorithms.

The ranking error is a measure used to evaluate the performance of ranking algorithms. It assesses how well a ranking algorithm predicts the correct order of items in a list compared to a ground truth ranking. If the predicted ranking list and ground truth ranking list both rank optimization algorithms in the same order, the ranking error would be 0.0. This indicates a perfect match between the meta-model predictions and the ground truth rankings. A ranking error of 0.5 signifies that the meta-model predictions are equivalent to random ranking. In other words, the meta-model performance is no better than randomly assigning ranks to the optimization algorithms in the portfolio. Lastly, a ranking error of 1.0 indicates that the predictions are completely opposite to the ground truth rankings. This means that the algorithm has reversed the order of all the items in the list. Note that in practice ranking is often performed using more advanced tools such as DSC [22] that might assign the same rank to algorithms where there is no statistical difference in performance.

D. Meta-model

In this study, we use three different meta-models that map between ELA features obtained through sampling an objective function and optimization algorithm ranks. All of the meta-models are essentially multi-target regression models, which take 62 ELA feature values as input and produce predictions of five real numbers that correspond to algorithm rankings. Three meta-models, in this case, are *xgboost*, *mean*, and *random*.

- *xgboost*: This meta-model performs ranking using the XGBoost [23] machine-learning algorithm (from python package *xgboost* version 2.0.0 with default hyperparamaters). XGBoost is a popular machine-learning algorithm that performs an iterative process of incorporating new decision trees into the ensemble. Noteworthy advantages of XGBoost include its ability to handle missing data, accommodate various data types (categorical, numerical), and typically deliver good results on tabular [24], [25].
- *mean*: This ranking algorithm calculates the average ranking across the entire training set and makes predictions solely based on that, without considering the ELA features of individual problems. Alternatively, one can view this ranking model as the "single best solver" in the ranking context, as it represents the mean rank across the entire training dataset.
- *random*: With this meta-model, optimization algorithms are assigned random rankings without considering any characteristics or features of the problem.

Be aware that the meta-models were trained to predict the rankings of optimization algorithms rather than the actual performance achieved by a particular algorithm. Before training the meta-models, the algorithm performance is converted into ranks meaning that the meta-model uses multi-output regression [26] to predict the ranks of five algorithms in the portfolio.

III. Results

The results section consists of several parts. First, we present the experimental setup III-A followed by the subsection performance of the algorithms on 24 base COCO

problem classes without any modification in III-B. Next, we illustrate some examples of algorithm ranking on problems obtained with affine transformations in III-C. Finally, in III-D we demonstrate how well the ranking meta-model trained on only base functions can rank algorithms accurately on affine combinations of functions.

A. Experimental Setup

In this paper, we exclusively focus on objective functions with a dimensionality $D = 5$. In each algorithm's execution, a fixed budget of 2000D function evaluations is allocated with a population size of 20. In total 62 ELA features are computed using the 1000D LHS sampling strategy. For every problem instance, the algorithms from the portfolio are run 30 times with random seeds. Problem instances 1-5 belonging to COCO base problems are utilized for training, while instances 6-10 are employed to evaluate the performance of the meta-models. The mixing parameters α, always range between 0.0 and 1.0 with increments of 0.1.

B. Algorithm Ranking on Base COCO Problems

To gain a deeper understanding of the performance of various algorithms across different problem classes, Figure 2 presents the average ranks of all five optimization algorithms for all 24 base COCO problem classes on five-dimensional problems. Each problem class comprises multiple instances, and we obtain the average rankings by considering all 10 problem instances and 30 runs for each instance. The figure reveals significant variations in the average rankings of algorithms across different COCO problem classes. For instance, in problem class 1-Sphere function, the PSO and DE algorithms consistently achieve the best rankings, followed by CMA-ES, ES, and finally GA. Conversely, for problem instances in problem class 3-Rastrigin function, the DE algorithm performs the best, while CMA-ES obtains the worst rank. This shows the diversity in algorithm rankings across different problem classes.

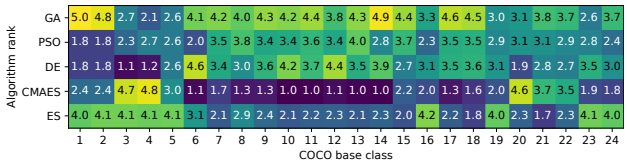

Fig. 2. Algorithm ranks for each of the 24 COCO problems classes aggregated across 10 problem instances per class with 30 runs per instance.

C. Algorithm Ranking on Affine Function Recombinations

One of the goals of the paper is to explore the behavior of algorithms on novel functions that are formed by combining existing functions through affine recombination. To illustrate this, Figure 3 shows the rankings of algorithms when transitioning between problem instances of problem classes 1 and 22 using the described affine function combinations. We observe that when algorithms are ranked for this specific function combination, the following can be expected: if the function deviates slightly from the original COCO base functions ($\alpha < 0.2$ or $\alpha > 0.8$), the algorithm ranks remain relatively stable. However, when a new function differs substantially from the base functions ($0.3 < \alpha < 0.7$), the algorithm ranking can deviate from what would be observed on any of the base functions from which the new one is created. An example of this is the ranking of the DE algorithm, which achieves an average rank of 1.8 and 2.6 on base functions, but obtains a rank of 4.0 on a newly created function with $\alpha = 0.5$. This showcases the difficulty and counterintuitive nature of ranking algorithms when applied to unfamiliar functions derived from base COCO functions. It suggests that for some problem pairs, algorithms that excel with the base functions may not perform effectively with the function created as a combination of base functions thus making the task of predicting the ranks a challenging task.

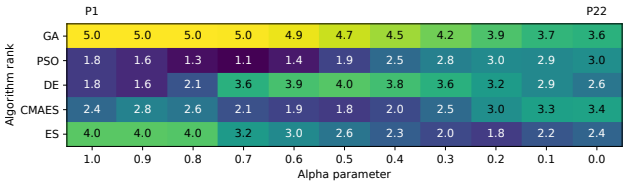

Fig. 3. Ranking of five different algorithms when evaluated on combinations of problems 1 and 22.

D. Out of Distribution Generalization of Meta-models

In this subsection, we investigate the ability of a model, which is specifically trained to rank algorithms for instances belonging to the 24 base COCO problem classes, to generalize to previously unseen instances created by combining objective functions from these base problems. Meta-models (described in II-D are trained solely on problem instances where $\alpha = 0.0$ or $\alpha = 1.0$. The trained meta-models are then evaluated on both the base COCO functions ($\alpha = 0.0$ or $\alpha = 1.0$) and functions created through combination or base functions ($0.0 < \alpha < 1.0$).

Each plot displays three lines representing different ranking techniques. The blue line represents the mean pairwise ranking error of the *xgboost* ranking model. The green line represents the mean ranking error that would result from ranking algorithms randomly i.e. the *random* meta-model. The orange line represents the mean error from the *mean* meta-model obtained by predicting the mean rank from the training data across all problem classes and instances. Specifically for this paper the *mean* ranking model always predicts the ranking [3.87, 2.98, 3.06, 2.18, 2.91] for algorithms [GA, PSO, DE, CMA-ES, ES]. Note that a meta-model that always predicts the correct ranking would obtain a ranking error of 0.0.

Considering the limited space available, we are unable to showcase the performance of meta-models across all conceivable problem combinations. However, we have carefully chosen a few combinations to present. Figure 4 illustrates the behavior of the ranking model for the problem class combinations (P12-P15), (P7-P9), (P6-P9), and (P19-P24) as

representative examples. Each line shows the mean pairwise ranking error computed over problem instances from 6 up to 10 for different meta-models and different values of α. By examining the uppermost plot in the diagram, we can observe the effectiveness of various strategies in achieving a high ranking for a combination of problems P12 and P15. When $\alpha = 0.0$ or $\alpha = 1.0$, the *xgboost* meta-learner demonstrates superior performance compared to the mean model's predicted ranks. This is because the training dataset already contained similar instances of the problem. In such instances, the *xgboost* ranker achieves a ranking error of slightly less than 0.1, whereas the *mean* meta-model experiences an error of 0.2. However, once new problem instances are introduced where $\alpha \neq 0.0$ and $\alpha \neq 1.0$, the performance of the *xgboost* ranking model rapidly declines. Specifically, at $\alpha = 0.6$, the *xgboost* meta-model exhibits a ranking error of approximately 0.3, while *mean* meta-model results in an error of 0.15. This indicates that *xgboost* is proficient at ranking algorithms for problems that closely resemble those in its training data. Conversely, ranking problems that deviate further from the training distribution becomes substantially more challenging for *xgboost*, rendering the *mean* meta-model more effective in such cases.

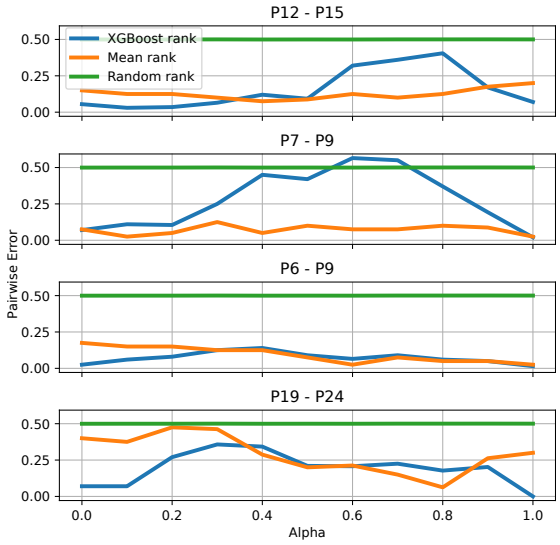

Fig. 4. Generalization of ELA features when performing algorithm ranking for a selected few problem classes. Better meta-model performance is indicated by a lower pairwise error.

Similar observations can also be made in the lower three plots shown in Figure 4. For instance, when considering combinations of problems P7 and P9, the *xgboost* meta-model archives algorithm ranking error that is comparable to *mean* meta-model ranking when $\alpha = 0.0$ or $\alpha = 1.0$. However, the performance of *xgboost* declines when predicting algorithm ranks for unfamiliar problems. Around $\alpha = 0.6$, the *xgboost* ranking model achieves an inferior accuracy to randomly ranking algorithms, indicating poor generalization on unseen problems. Conversely, when combining problems P6 and P9, the *xgboost* ranking meta-model achieves satisfactory outcomes even for previously unseen instances of problems compared to the *random* meta-model. Despite being trained only on problem instances with $\alpha = 0.0$ and $\alpha = 1.0$, the algorithm demonstrates the ability to achieve minimal ranking errors on mixed instances combining P6 and P9, surpassing the *mean* ranking strategy in some of the cases. Lastly, we shift our attention to the plot that depicts the model's accuracy on instances involving a combination of problems P19 and P24. In this scenario, the *xgboost* meta-model outperforms both the *random* and the *mean* meta-model for $0.3 < \alpha < 0.9$ in terms of pairwise ranking error. This once again confirms that the ranking meta-model based on *xgboost* can generate precise rankings for problems P19 and P24, which were part of the training set. However, it performs poorly when confronted with new problems that are a combination of both, indicating that these new out-of-distribution problems pose a challenge. In summary, this analysis reveals that meta-models trained to rank algorithms on base COCO problem instances often exhibit limited generalization when faced with new problem instances formed by combinations of base COCO problems. However, it is noteworthy that there are instances, where the *xgboost* meta-model can still generate algorithm ranks that closely align with the ground truth, even for these newly created problems.

Figure 5 illustrates the pairwise ranking results obtained for different α levels, encompassing pairs between all 24 problem classes and 5 problem instances present in the test set. The figure provides several key observations. When utilizing ELA features to map algorithm ranks, the *xgboost* ranking model demonstrates superior performance for problems observed during training or those similar to the training set. Therefore, when a problem is observed during training, the *xgboost* ranking model consistently outperforms the *mean* ranking meta-model in terms of average ranking. Training and evaluation of *xgboost* the meta-model on instances belonging to base problems yield a pairwise ranking error of 0.1 on new instances from the same base problems, which surpasses the *mean* meta-model ranking error of 0.3. However, when encountering new problem instances created as combinations of existing base COCO problems, the performance of *xgboost* meta-models which rely on ELA features substantially deteriorates. In such cases, simply predicting mean ranks across all 24 problems often achieves better results compared to the *xgboost* ranking method or around 0.3.

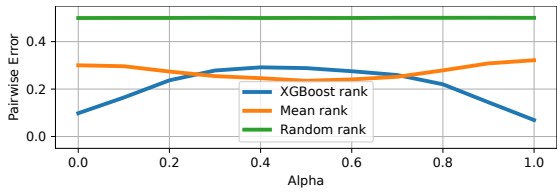

Fig. 5. Generalization of ELA features when performing algorithm ranking across all problem instances and problem classes. Better meta-model performance is indicated by a lower pairwise error.

IV. Conclusion and Future Work

This research paper investigates the effectiveness of ELA features in describing problems and their application in predicting the ranking of optimization algorithms. Specifically, it explores the capabilities of a meta-model that is trained exclusively on base functions and assesses its performance in ranking algorithms when applied to newly generated problems using affine transformations. The results indicate that the meta-model exhibits varying degrees of success in predicting algorithm order. Although it shows relative success in predicting the ranks of algorithms for problem classes whose instances fall within the training distribution, its performance substantially declines as the optimization problems deviate from what the meta-model has seen during training. The model's ability to rank algorithms in these cases becomes comparable to predicting the mean ranking that is obtained over all of the problems. These findings underscore the limitations of meta-models trained exclusively on base functions and highlight the need for further research to enhance their generalization capabilities, particularly for out-of-distribution problems.

In future work, it would be valuable to investigate the underlying factors that contribute to the varying difficulty levels of algorithm ranking for different problems. Understanding the characteristics or patterns that make certain problems more challenging can provide insights into improving algorithm performance. Additionally, exploring methods to quantify the meta-model's confidence in its predictions [27] can further contribute to determining if the meta-model is certain in its predictions and if one should trust the ranks that are predicted. Furthermore, creating new problems that bridge the gaps between existing problems and incorporating them into the training phase could help improve the model's generalization capabilities and ensure better performance on out-of-distribution problems.

Acknowledgment

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

References

[1] A. Slowik and H. Kwasnicka, "Evolutionary algorithms and their applications to engineering problems," *Neural Computing and Applications*, vol. 32, pp. 12 363–12 379, 2020.

[2] J. R. Rice, "The algorithm selection problem," in *Advances in computers*. Elsevier, 1976, vol. 15, pp. 65–118.

[3] R. Tanabe, "Benchmarking feature-based algorithm selection systems for black-box numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 6, pp. 1321–1335, 2022.

[4] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory landscape analysis," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 829–836.

[5] G. Petelin, G. Cenikj, and T. Eftimov, "Tla: Topological landscape analysis for single-objective continuous optimization problem instances," in *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2022, pp. 1698–1705.

[6] B. van Stein, F. X. Long, M. Frenzel, P. Krause, M. Gitterle, and T. Bäck, "Doe2vec: Deep-learning based features for exploratory landscape analysis," *arXiv preprint arXiv:2304.01219*, 2023.

[7] A. Jankovic and C. Doerr, "Landscape-aware fixed-budget performance regression and algorithm selection for modular cma-es variants," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020, pp. 841–849.

[8] U. Škvorc, T. Eftimov, and P. Korošec, "Transfer learning analysis of multi-class classification for landscape-aware algorithm selection. mathematics 10 (3)(2022)," 2022.

[9] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff, "Coco: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, vol. 36, no. 1, pp. 114–144, 2021.

[10] G. Wu, R. Mallipeddi, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 competition and special session on constrained single objective real-parameter optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.*, 10 2016.

[11] Q. Renau, J. Dréo, C. Doerr, and B. Doerr, "Towards explainable exploratory landscape analysis: extreme feature selection for classifying bbob functions," in *Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24*. Springer, 2021, pp. 17–33.

[12] A. Nikolikj, R. Trajanov, G. Cenikj, P. Korošec, and T. Eftimov, "Identifying minimal set of exploratory landscape analysis features for reliable algorithm performance prediction," in *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2022, pp. 1–8.

[13] G. Cenikj, R. Dieter Lang, A. Petrus Engelbrecht, C. Doerr, P. Korošec, and T. Eftimov, "SELECTOR: Selecting a Representative Benchmark Suite for Reproducible Statistical Comparison," in *Proceedings of The Genetic and Evolutionary Computation Conference*, 2022, in Press.

[14] D. Vermetten, F. Ye, and C. Doerr, "Using affine combinations of bbob problems for performance assessment," *arXiv preprint arXiv:2303.04573*, 2023.

[15] Q. Renau, J. Dreo, A. Peres, Y. Semet, C. Doerr, and B. Doerr, "Automated algorithm selection for radar network configuration," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 1263–1271.

[16] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89 497–89 509, 2020.

[17] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, pp. 8091–8126, 2021.

[18] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[20] T. Bäck, "Evolution strategies: An alternative evolutionary algorithm," in *Artificial Evolution: European Conference, AE 95 Brest, France, September 4–6, 1995 Selected Papers*. Springer, 2005, pp. 1–20.

[21] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary computation*, vol. 9, no. 2, pp. 159–195, 2001.

[22] T. Eftimov, P. Korošec, and B. K. Seljak, "A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics," *Information Sciences*, vol. 417, pp. 186–215, 2017.

[23] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

[24] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022.

[25] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, vol. 54, pp. 1937–1967, 2021.

[26] H. Borchani, G. Varando, C. Bielza, and P. Larranaga, "A survey on multi-output regression," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[27] D. Guillory, V. Shankar, S. Ebrahimi, T. Darrell, and L. Schmidt, "Predicting with confidence on unseen distributions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 1134–1144.