RF+clust for Leave-One-Problem-Out Performance Prediction

ANA NIKOLIKJ, Jožef Stefan International Postgraduate School, Slovenia and Computer Systems Department, Jožef Stefan Institute, Slovenia

CAROLA DOERR, Sorbonne Université, CNRS, LIP6, France

TOME EFTIMOV, Computer Systems Department, Jožef Stefan Institute, Slovenia

Per-instance automated algorithm configuration and selection are gaining significant moments in evolutionary computation in recent years. Two crucial, sometimes implicit, ingredients for these automated machine learning (AutoML) methods are 1) feature-based representations of the problem instances and 2) performance prediction methods that take the features as input to estimate how well a specific algorithm instance will perform on a given problem instance. Non-surprisingly, common machine learning models fail to make predictions for instances whose feature-based representation is underrepresented or not covered in the training data, resulting in poor generalization ability of the models for problems not seen during training. In this work, we study leave-one-problem-out (LOPO) performance prediction. We analyze whether standard random forest (RF) model predictions can be improved by calibrating them with a weighted average of performance values obtained by the algorithm on problem instances that are sufficiently similar to the problem for which a performance prediction is sought, measured by cosine similarity in feature space. While our RF+clust approach obtains more accurate performance prediction for several problems, its predictive power crucially depends on the chosen similarity threshold as well as on the feature portfolio for which the cosine similarity is measured, thereby opening a new angle for feature selection in a zero-shot learning setting, as LOPO is termed in machine learning.

CCS Concepts: • **Algorithm Performance Prediction**; • **AutoML**; • **Zero-Shot Learning**; • **Single-Objective Black-Box Optimization**;

1 INTRODUCTION

Various algorithms for continuous single-objective optimization (SOO) have already been developed and their performance investigated through statistical analyses, in most cases reporting the average performance across a selected set of benchmark problem instances [26]. However, the algorithm instance behavior varies substantially depending on the problem instance that is being solved. For this purpose, there is a predictive task known as automated algorithm selection, where the main goal is selecting the best-performing algorithm from a set of algorithm instances for a given problem instance [1, 5, 11, 17]. To achieve it, automated algorithm performance prediction is a crucial step that should be done. The automated algorithm performance prediction is tackled as a supervised machine learning problem, such as classification (i.e., predicts whether or not the algorithm solves the instance given some precision) or a regression (i.e., predicts the performance of the algorithm as a real value). To train a supervised machine learning algorithm a set of examples, problem instances described by their landscape features used as input data (i.e., benchmark suite) and algorithm performance achieved on them used as target (i.e., solution precision) are required. Nowadays, if we want to generalize a supervised prediction model to another benchmark suite, whose problems were not involved in the training data, the predictive model performance decreases greatly. This occurs because the training data does not cover some regions of the landscape space involved in the test data, or the model is biased toward some over-represented landscape regions that are present in the training data. This is the reason why most of the studies focus on the Black-box Optimization Benchmarking (BBOB) benchmark suite [9] since it involves several instances from the same problem class involved in the training data which makes it a suitable resource for training an ML-supervised model. However, if we remove the instances from the same problem class from the training data (leave all instances from a single problem out for testing), the performance of the model for automated algorithm performance prediction decreases significantly [25]. This issue

makes all approaches for automated algorithm performance prediction that are developed based on the BBOB benchmark suite difficult to generalize on other benchmark suites such as CEC [16], Nevergrad [23], etc. since in their definition there is only a single instance per each problem class. In practice, an application such as algorithm performance prediction requires making predictions for problem instances whose problem class (i.e., landscape properties) has not been seen previously by the underlying model or we have a leave-one-problem-out (LOPO) learning scenario.

Our contribution. In this study, we propose an approach for LOPO automated algorithm performance prediction. Our RF+clust approach calibrates a classic random forest (RF) predictive model with a prediction obtained by a similarity relationship method that aggregates the ground algorithm instance performance for the most similar problem instances from the training data. In contrast to a classic KNN approach, we use a similarity threshold to decide which problems are taken into account for the calibration. The number of considered 'neighbors' can therefore differ between problems. We evaluate our approach on performance data of three differential evolution (DE) variants on the CEC 2014 and the BBOB benchmark suite of the COmparing Continuous Optimizers (COCO) environment [10]. We observe better results for RF+clust than for stand-alone RF performance prediction in a number of cases. However, there are also cases when similar landscape representations can lead to different performances of the algorithm, which can affect the prediction of the RF+clust approach. This further points out that in the future we need to focus on finding problem feature representations with sufficient discriminate power that will be also able to capture the performance of the algorithm.

Outline. The remainder of the paper is organized as follows: Section 2 surveys past work on automated algorithm performance prediction. The proposed LOPO regression method is introduced in Section 3. Section 4 details the benchmark problem suites and algorithms used for the validation of the proposed approach, the problem landscape features, as well as the machine learning algorithm tuning and evaluation. The results and discussion are provided in Section 5. Finally, the conclusions are drawn in Section 6.

2 RELATED WORK

Next, we point out some of the works which are addressing the critical issue of generalization over new problem classes.

Bischl et al. [3] consider automated algorithm selection as a cost-sensitive classification task using one-sided Support Vector Machines. Problem instance-specific miss-classification costs are defined, unlike standard classification where all the errors in classification are penalized the same by the algorithm. The predefined miss-classification costs represent external information to aid the learning process. The approach was tested on problem instance feature representations consisting of "cheap" and "expensive" ELA features [18], with respect to the sample size required for their calculation. It was shown that the model is able to generalize over new instances, however, the prediction error gets worse for new problem classes, when the prediction is based only on the "cheap" ELA features representation. To discover the source of the larger model errors, analysis of the feature space is performed based on the euclidean distances between the problem instances representations. They conclude that the degree of classification performance tends to correlate with the proximity in feature space for the case of using the entire feature set, however, this was not that straightforward for the "cheap" features.

The work [7] brings to attention the possibility to *personalize* regression models (Decision Tree, Random Forest, and Bagging Tree Regression) to specific optimization problem classes. Instead of aiming for a single model that works well across a whole set of possibly diverse problems, the personalized regression approach acknowledges that different models may suit different problem types.

In [25] a classification-based algorithm selection approach is evaluated on the COCO benchmark suite [9] and artificially generated problems [6]. The results show that such a model has low generalization power between datasets and in the leave-one problem-out cross-validation procedure where each problem class was removed one at a time from the same dataset. However, a model trained and tested in a leave-instance-out scenario achieves much higher accuracy. A correlation analysis using the Pearson correlation coefficient was performed for the problem representations based on the "cheap" ELA features, showing that a large number of both, the COCO and the artificial problems are highly correlated within their own set of problems. i.e., the poor generalization is due to the differences between the two data sets in feature space.

The feasibility of a "per-run" algorithm selection scheme is investigated in [14], based on ELA features that are calculated from the observed trajectory of the algorithm (i.e., the samples the algorithm visits during the optimization procedure). This avoids the usually required additional evaluation of (quasi-)random samples implemented by classic per-instance algorithm selection schemes. Results for the COCO benchmark suite show performance comparable to the per-function virtual best solver. However, these results did not directly generalize to the other benchmark suites used in the experiments, namely the YABBOB suite from the Nevergrad platform [23].

3 LOPO ALGORITHM PERFORMANCE PREDICTION

Let us assume a set of benchmark problem instances P_t^i, $i = 1, \ldots, n$, which are grouped into training problem classes P_t, $t = 1, \ldots, m$, and performance data for an algorithm instance A on the selected set of benchmark problem instances. To predict the performance y_q^i of the algorithm instance A on a problem instance P_q^i from a new problem class P_q that is not involved in the training data, we have proposed the following LOPO approach:

1) Represent the selected benchmark problem instances from the m problem classes by calculating the ELA features and linking them to the performance of the algorithm instance after a certain number of function evaluations.

2) Train a supervised regression model that uses the ELA features as input data and predicts the algorithm instance performance.

3) For a new problem instance P_q^i from a new problem class P_q that is not involved in the training data, use its ELA features as input data into the learned model to make the prediction y_q^i.

4) Select the k-nearest problem instances from the training set that are the most similar to the new problem instance based on their landscape features representation. The similarity is measured by a similarity metric s, and the selection is done by defining a prior similarity threshold. We selected cosine similarity as a similarity measure. Finally, all problem instances from the training data that have a similarity greater or equal to the predefined threshold are selected, from which the ground truth algorithm performance is retrieved $p_1, p_2, \ldots, p_k$. We need to point out here that the number of the nearest problem instances k, differs for different problem instances, so it is found by the selection rule and the predefined threshold.

5) The final prediction of the algorithm instance performance on the new problem instance is made by calibrating the prediction obtained by the learned model y_q^i with the ground truth algorithm instance performance retrieved for the selected nearest problem instances from the training data. This is performed as an aggregation procedure as follows: $\widehat{y_q^i} = \left(y_q^i + F(p_1, p_2, \ldots, p_k) \right) / 2$, where $F(p_1, p_2, \ldots, p_k)$ is an aggregation function, which can be for example weighted mean. 6) If there are no problem instances in the training data to which the new problem instance is similar above the threshold, only the prediction of the model is considered, y_q.

4 EXPERIMENTAL DESIGN

Here, we are going to present all experimental details starting from the data that is involved and the techniques used for the ML learning process.

Problem Benchmark Suites. We evaluate the proposed method by using two of the most currently used benchmark problem suites in the field of numeric single-objective optimization. The first benchmark suite involved in the experiments is the 2014 CEC Special Sessions & Competitions (CEC 2014) suite. The suite consists of 30 problems where only one instance per problem is available. The problems are provided in dimension 10. The full problem list and descriptions of all the problems are available at [16]. The second problem set is the 24 noiseless single-objective optimization problems from the BBOB collection of the COCO benchmarking platform [10]. Different problem instances can be derived by transforming the base problem with predefined transformations to both its domain and its objective space, resulting in a set of different instances for each base problem class, that have the same global characteristics. We consider the first five instances of each BBOB problem, resulting in a dataset of 120 problem instances. In coherence with the CEC problem suite, the problem dimension D was set to 10.

Algorithm Performance Data. Performance data is collected for three different randomly selected Differential Evolution (DE) [27] configurations, on both the CEC 2014 and BBOB benchmark suites. The DE hyper-parameters are as presented in Table 1. We indexed the algorithm configurations starting from DE1 to DE3 for easier notation of the results. DE is an iterative population-based meta-heuristic. The population size of DE is set to equal the problem dimension D ($D = 10$ in our study). The three DE configurations were run 30 times on each problem instance, and we extracted the precision after a budget of $500D = 5000$ function evaluations. In our study, we consider the median target precision achieved in these 30 runs. Following the approach suggested in [11], we also consider the logarithm (log10) of the median solution precision. This algorithm performance measure estimates the order of magnitude of the distance of the reached solution to the optimum. Figure 1 presents DE1 performance (log-scale) obtained per benchmark problem on the CEC 2014 and Figure 2 on the BBOB benchmark suite (aggregated for all problem instances).

Table 1. Differential Evolution (DE) configurations.

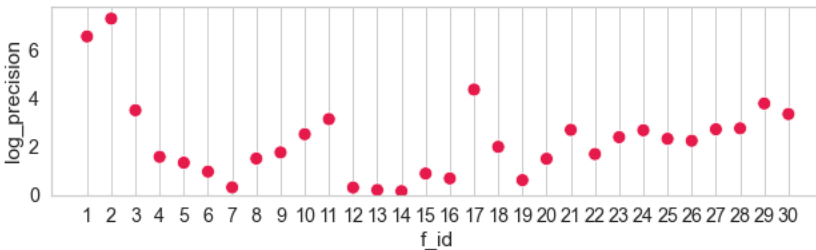| index	strategy	F	Cr
DE_1	Best/3/Bin	0.533	0.809
DE_2	Best/1/Bin	0.617	0.514
DE_3	Rand/Rand/Bin	0.516	0.686

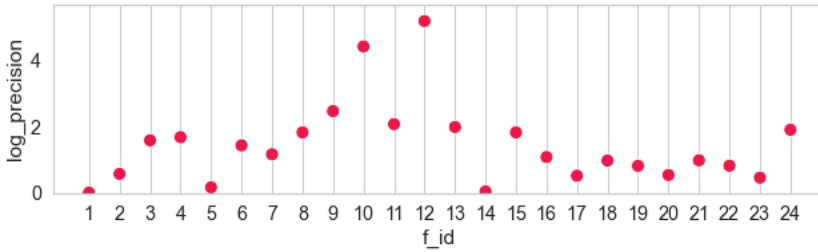Fig. 1. DE1 solution precision (log-scale) per problem instance on the CEC 2014 suite.

Fig. 2. DE1 solution precision (log-scale) on the first instance of each problem in the BBOB suite.

Exploratory Landscape Analysis (ELA). To create a feature representation that encodes problem properties, the static Exploratory Landscape Analysis (ELA) [18] features are used. The features are calculated by the evaluation of a sample of candidate solutions generated by systematic sampling of the decision space of the problem. The corresponding fitness values are then fed to different statistical and mathematical methods to calculate the feature values. The Improved Latin Hypercube Sampling (ILHS) [28] was used as a sampling technique, with a sample size of $800D$ (8000). In reality, this is a really big sample size, however, we are interested in whether the approach works, so we want to reduce the randomness from the feature computation [15]. For each benchmark problem instance, the calculation of the features was repeated 30 times, as it is a stochastic process and the median value was taken as the final feature value that numerically quantifies some property of the problem. The R package "flacco" [13] was utilized for their calculation. We selected all the ELA features which are cheap to calculate with regard to sample size, and do not require additional sampling. This way, a total of 64 features were calculated. The selected features are coming from the following groups: classical ELA (y-distribution measures, level-set, meta-model), Dispersion, Information Content, Nearest Better Clustering, and Principal Component Analysis.

Regression Models for Algorithm Performance Prediction. For the learning process, we considered random forest (RF) regression [2], as it provides promising results for algorithm performance prediction [12] and is one of the most commonly used algorithms for algorithm performance prediction studies in evolutionary computation. The RF algorithm was used as implemented by the scikit-learn package [21] in Python. We have trained single-target regression (STR) models. That is, we have a separate model for predicting the performance of each of the three DE algorithms.

ML Model Evaluation. When splitting the data as described in Section 3 the evaluation of the automated algorithm performance prediction results in leave-one-problem-out fold validation. At each fold, a model was trained using one problem class (including all of its instances) left out for testing, while the remaining are used for training. In order to assess the accuracy of the models, we compute the Mean Absolute Error (MAE). The prediction errors are the absolute distances of the prediction to the true algorithm precision value on the new problem class.

Hyper-parameter Tuning for the Regression Models. The best hyper-parameters are selected for each RF model from the training portion of the fold. The hyper-parameters selected for tuning are $n\ estimators$ - the number of trees in the random forest; $max\ features$ - the number of features used for making the best split; $max\ depth$ - the maximum depth of the trees, and $min\ samples\ split$ - the minimum number of samples required for splitting an internal node in the tree. The ranges of the hyper-parameters have been selected concerning the data set size and the guidelines available in ML to avoid over-fitting. The best hyper-parameters for each problem class are presented in our repository [20].

Table 2. Mean Absolute Error (MAE) obtained by the RF models for predicting the performance of DE1 with different feature portfolios on the CEC 2014 benchmark problems

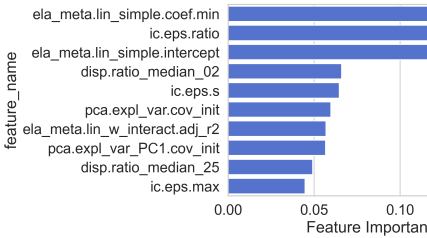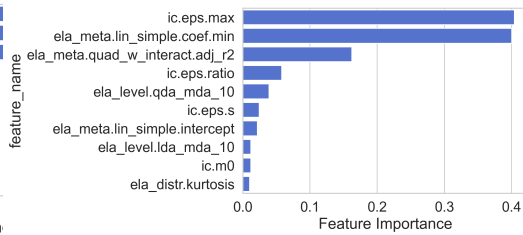| top features	aggregation	mae_train	mae_test
10	mean	0.504464	1.279261
10	median	0.536681	0.991274
30	mean	0.458142	1.318326
30	median	0.439434	0.770265
64	mean	0.498807	1.465239
64	median	0.480457	0.931685

Fig. 3. CEC 2014

Fig. 4. BBOB

Fig. 5. The ten most important ELA features for predicting the performance of DE1 on the CEC 2014 and BBOB benchmark suites.

Feature Selection. Taking into consideration the size of the datasets, in a scenario where 30 data instances (CEC 2014 benchmark suite) are available, and 64 features to describe them, we run the risk of overfitting our model. Therefore, we have performed feature selection. Since we have a LOPO scenario (i.e., leave all instances for a single problem out), in our case we ended with 30 ML predictive models. To select the top most important features for each model, the SHAP method [19] was utilized. Finally, the importance of the features was summarized across all the models, and the 10 and 30 most important features were used to train the models. These two sizes of feature portfolios were tested in order to compare the decrease in ML model performance (if any) when using different feature portfolios (Table 2). Also, the feature portfolio influences the proposed approach as it is based on the pairwise similarity of the features. So different feature portfolios can result in different instances retrieved as similar. Figure 5 shows the feature portfolio of the 10 most important features for CEC 2014 and BBOB accordingly.

5 RESULTS

We apply the approach to three random DE configurations and two benchmark suites (CEC 2014 and COCO). Due to space limitations, however, we present here some selected results for algorithm DE1 and the CEC 2014 benchmark suite, while other results where similar findings are noticed are available at[20].

In Figure 6 we compare a classical supervised RF model and an RF+clust model in a leave-one-problem-out scenario. Figure 6 shows the prediction errors obtained by a standard RF model trained in the LOPO (corresponding to the "RF" denoted row on the heatmap) and errors of the proposed RF+clust approach (for similarity thresholds of 0.5, 0.7, and 0.9, also with corresponding rows on the heatmap). The predictions were obtained by using a feature portfolio of the ten most
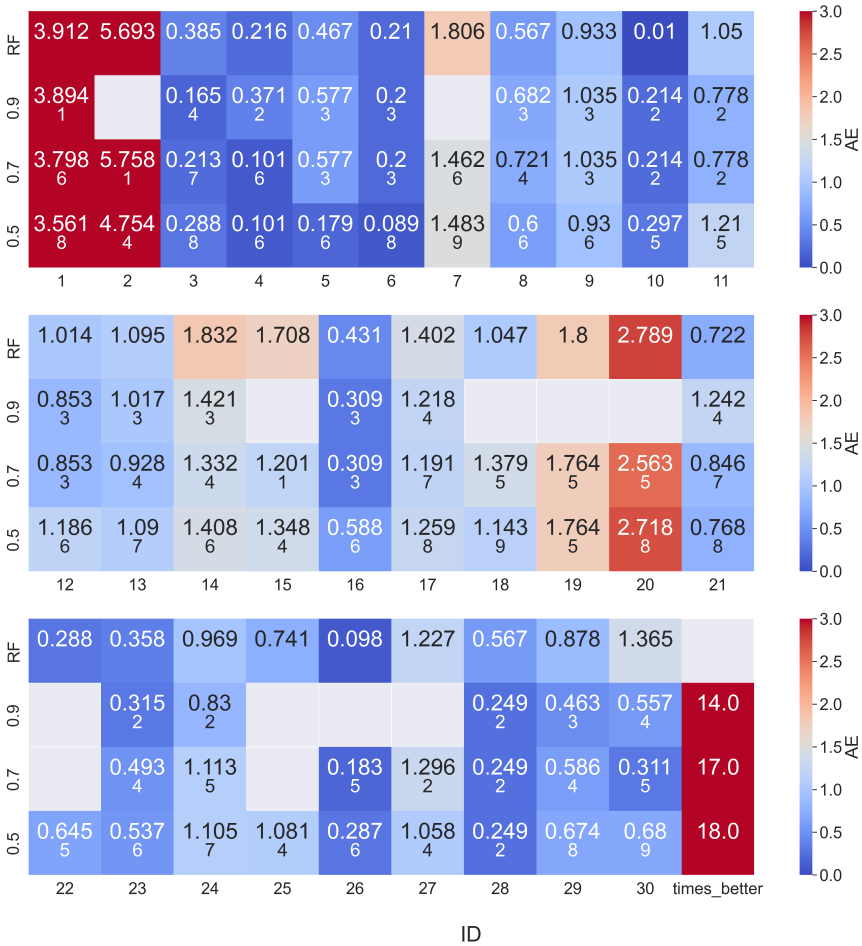
Fig. 6. The mean absolute error of the RF model and the RF+clust approach using a similarity threshold of .5, .7, and .9, for each problem in the CEC 2014 benchmark suite, for DE1 and the feature portfolio of ten most important ELA features.

important features. Each cell of the heatmap represents the absolute error obtained by the models. The columns represent each problem instance separately, while the last column indicates on how many instances the approach showed lower prediction error). The numbers under the model error indicate the number of similar instances set above the corresponding threshold that were present in the training of the model. The blank cells in the heatmap are places where the RF+clust approach provides the equal result as the RF model because for those problems we could not find similar problem instances from the training data using the predefined threshold.

The figure shows that there are problems (1, 3, 6, 11, 12, 13, 14, 16, 17, 23, 24, 28, 29, 30) for which the RF+clust approach shows better predictive results than using a classically trained RF model. We can see that for the high similarity threshold (0.9), calibrating the classical prediction with the ground truth performance of the optimization algorithm of the retrieved similar problems from the training data decreases the predictive error. To provide an explanation for why this happens, Figure 7 presents the relation between the pairwise similarity of the ELA features representation
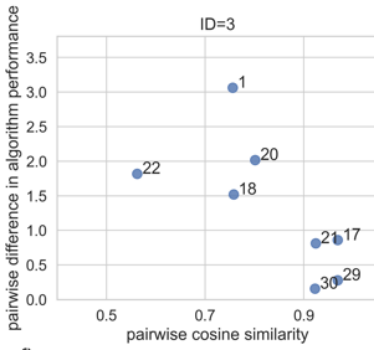
Fig. 7. Pairwise similarity of the ELA features representation (x-axis) and the pairwise difference in the ground truth performance of DE1 (y-axis) for the third problem in CEC 2014.
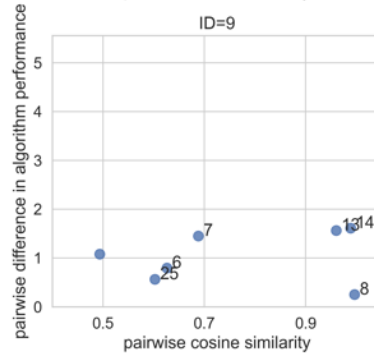

Fig. 8. Pairwise similarity of the ELA features representation (x-axis) and the pairwise difference in the ground truth performance of DE1 (y-axis) for the ninth problem in CEC 2014.

(x-axis) and the pairwise difference in the ground truth performance of the optimization algorithm (y-axis) for the third problem in the CEC 2014, with the other problems. The heatmap shows that the third problem has four similar problem instances over 0.9 (17, 21, 29, 30), as visible in Figure 7. In addition, we can see that the difference in ground algorithm performance of the problem and the similar instances is low, so the algorithm has similar behavior on these problems (see also Figure 1), and using them for the calibration helps to obtain better predictive errors.

There are also problems when the predictions are affected by the RF+clust approach, slightly worse than the prediction obtained from the classical RF model. Figure 8 presents the relation between the pairwise similarity of the ELA features representation (x-axis) and the pairwise difference in the ground truth performance of the optimization algorithm (y-axis) for the ninth problem in the CEC 2014. The heatmap shows that the ninth problem has three similar problem instances (8, 13, 14) according to Figure 8. Here, we can see that one out of three problems is similarly based on the ELA representation and the algorithm has similar behavior on it. However, on the remaining two problems, we can see that even with high similarity in the landscape space, the difference in algorithm performance is larger in reality (see Figure 1), so using the performance to calibrate the prediction yields a larger error. A similar scenario happens for the 21st problems, where the similarity is greater or equal to 0.9 but the difference in performance between them is very large (see Figure 9). This indicates that there are problems for which the ELA features representations are not expressive enough and they could not well describe the problems in such a scenario (i.e., similar ELA landscape representation may not lead to similar algorithm behavior in the performance space).

There are also problems such as the first and the second that are difficult to be solved by the optimization algorithm (see Figure 1 for ground truth performance). Figure 10 presents the relation between the pairwise similarity of the ELA features representation (x-axis) and the pairwise difference in the ground truth performance of the optimization algorithm (y-axis) for the second problem in the CEC 2014. It is visible that this problem has very few similar instances, does not have similar instances over 0.9 at all, and also the difference in algorithm performance with similar instances over 0.5 is very large. This is the case where the test problem class is not covered enough by the train, however, even in such scenarios, we can be slightly better in the prediction results.
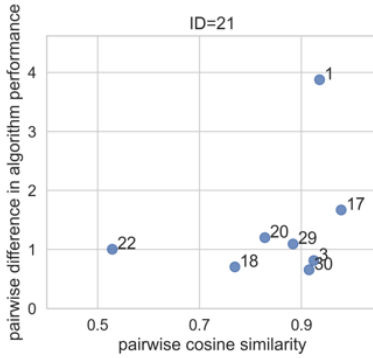
Fig. 9. Pairwise similarity of the ELA features representation (x-axis) and the pairwise difference in the ground truth in the ground truth performance of DE1 (y-axis) for the 21st problem in CEC 2014.
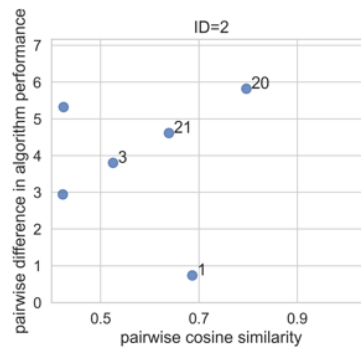
Fig. 10. Pairwise similarity of the ELA features representation (x-axis) and the pairwise difference in the ground truth in the ground truth performance of DE1 (y-axis) for the second problem in CEC 2014.

Table 3. Number of times the RF+clust is better, equal, or worse than stand-alone RF for predicting the performance of DE variants on the CEC 2014 benchmark suite.

algorithm name	model	# better	# equal	# worse
DE1	0.9	14.0	10.0	6.0
DE1	0.7	17.0	2.0	11.0
DE1	0.5	18.0	0.0	12.0
DE2	0.9	13.0	5.0	12.0
DE2	0.7	18.0	0.0	12.0
DE2	0.5	21.0	0.0	9.0
DE3	0.9	8.0	15.0	7.0
DE3	0.7	14.0	2.0	14.0
DE3	0.5	16.0	0.0	14.0

Looking back at the heatmap (Figure 6), we can see that when the similarity threshold decreases (i.e., going to 0.5), some of the predictions are slightly worse. This was expected since having a lower threshold in the landscape space does not guarantee to capture similar performance in the performance space (see Figures 8, 9, and 10). The heatmaps presenting the results for the other two DE algorithms lead to similar results and explanations.

In Table 3 we summarize how many out of the 30 problems the RF+clust approach provides better, worse, or equal predictions (when similar instances are not found, the prediction is not calibrated) than the classical (stand-alone) RF model on the CEC 2014 benchmark suite.

To investigate the sensitivity with a different feature portfolio, we repeated the experiments by selecting the top 30 most important features for predicting the performance of DE1 on the CEC 2014 benchmark suite. Figure 11 presents the absolute errors of the RF and RF+clust approaches obtained for each problem of the CEC 2014. Focusing on the similarity threshold of 0.9, the results show that the RF+clust approach provides better predictions (i.e., improvements) than the classical RF model for nine problems, worse predictions for five problems, and equal for 16 problems. We need to point out here that increasing the number of the top most important features from 10 to 30,

also affects the similarity of the problem instances. From the heatmap is visible that now we are not able to detect similar problems for some of the problems (e.g., 1, 21, 23, 24, 28, 29, 30) for which we were able to detect similar problems above 0.9 when the feature portfolio of the 10 most important features is used. This further opens a new angle for feature selection that will have discriminate power to capture also differences that happen in the performance space.
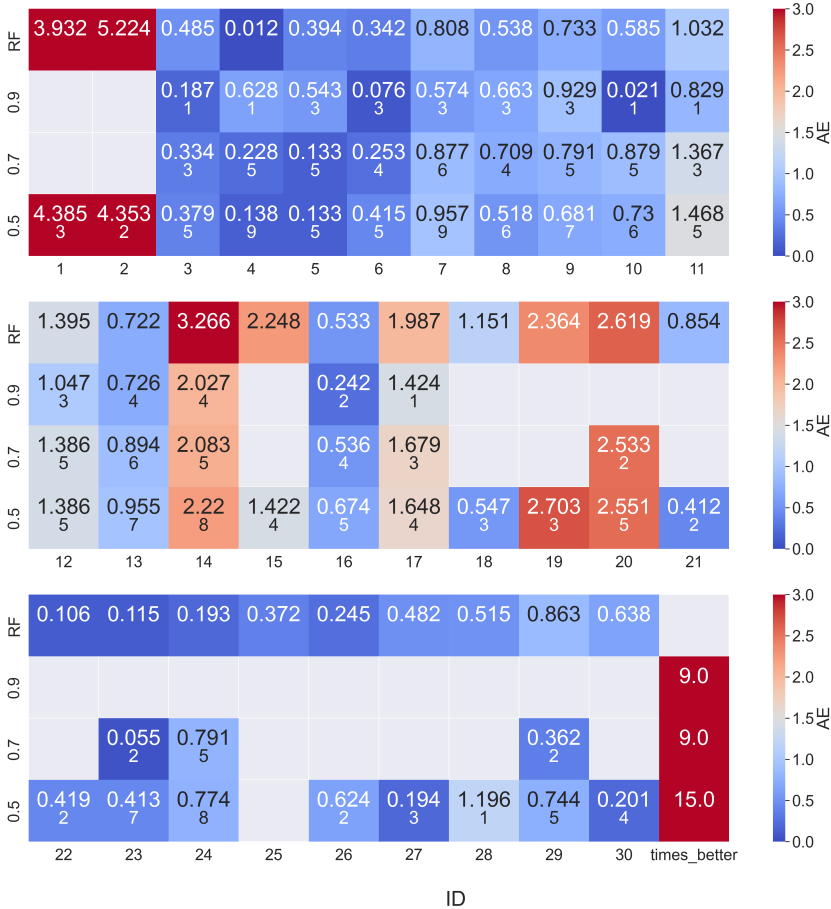

Fig. 11. The mean absolute error of the RF model and the RF+clust approach using a similarity threshold of .5, .7, and .9, for each problem in the CEC 2014 benchmark suite, for DE1 and the feature portfolio of 30 most important ELA features.

In addition to the results obtained on the CEC 2014 benchmark suite, Figure 12 presents the prediction results obtained for the DE1 algorithm for the first instance of each COCO problem. We selected only one instance here, for visualization purposes (our overall setting remains LOPO, i.e., we omit *all* instances of the left-out problem, and we use data from the five instances of the other problems for training. This also explains why the number of similar problem instances is larger in Figure 12 compared to those for the CEC benchmark presented in Figure 6). The results are in line with those obtained for the CEC 2014 benchmark. We point out that the top 10 most important features to train the prediction model differ from those selected on the CEC 2014 benchmark;

see Figure 5 for details. An indirect outcome of this study is that these two benchmark suites are different, which also supports previously published findings [24].
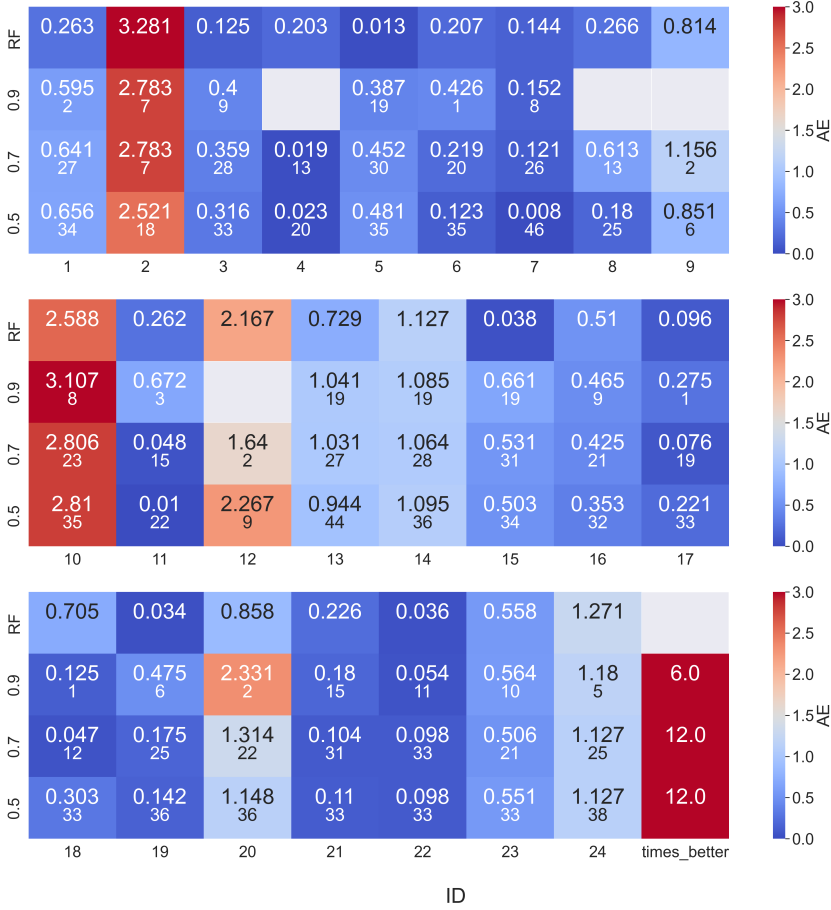

Fig. 12. The mean absolute error of the RF model and the RF+clust approach using a similarity threshold of .5, .7, and .9, for the first instance of each problem in the BBOB benchmark suite, for DE1 and the feature portfolio of ten most important ELA features.

6 CONCLUSIONS

In this study, we have proposed leave-one-problem-out (LOPO) for algorithm performance prediction. The idea behind the approach is to predict the performance of an optimization algorithm by using a supervised ML model on a problem that is not presented in the training data. First, a model is learned from a feature landscape representation of the problem instances from the train problem classes, and a prediction is made for an instance from a new problem class in a supervised manner. Second, based on the similarity relationship between the problem classes based on their feature landscape representation, the prediction for the new problem instance is calibrated by applying an aggregation procedure over the algorithm performance of the k-nearest problem instances from the training data.

The results performed on the CEC 2014 benchmark suite showed promising results and explanations about the strengths and weaknesses of the proposed approach. Better results are achieved for problems for which their landscape feature representation is similar to other problems from the training data and the algorithm behaves similarly on those problems. However, there are also problems for which the proposed approach can lead to slightly worse prediction results. This happens for problems for which the landscape feature representation leads to finding similar problems from the training data, however, the performance of the algorithm significantly differs. Such a result indicates that we need to find an expressive enough landscape feature representation that correlates with the algorithm's performance. Also, there are problems for which there are no similar problems in the training data, which further indicates that we need to enrich the data that is used in ML setup with new problems (e.g., merging different benchmark suites or using artificially problem generators [6]) by taking care that all landscape spaces approximate a uniform distribution in the problem space.

In the future, we are planning to test the approach on a more comprehensive algorithm portfolio. Next, instead of exploratory landscape features calculated by a global sampling, we are planning to calculate them using the trajectory data that was observed by the algorithm during the run (i.e., to capture also information about the algorithm behavior) [14]. We are also going to try different problem feature representations such as topological data analysis [22]. Last but not least, we are planning to merge different benchmark suites to select representative problem instances [4, 8] that will allow us to represent all possible landscape spaces from the problem space with the same number of problems that will further help the LOPO approach to have better prediction results.

ACKNOWLEDGMENTS

REFERENCES

[1] Nacim Belkhir, Johann Dréo, Pierre Savéant, and Marc Schoenauer. 2017. Per instance algorithm configuration of CMA-ES with limited budget. In *GECCO*. 681–688.

[2] Gérard Biau and Erwan Scornet. 2016. A random forest guided tour. *Test* 25, 2 (2016), 197–227.

[3] Bernd Bischl, Olaf Mersmann, Heike Trautmann, and Mike Preuß. 2012. Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. 313–320.

[4] Gjorgjina Cenikj, Ryan Dieter Lang, Andries Petrus Engelbrecht, Carola Doerr, Peter Korošec, and Tome Eftimov. 2022. SELECTOR: Selecting a Representative Benchmark Suite for Reproducible Statistical Comparison. *arXiv preprint arXiv:2204.11527* (2022).

[5] Bilel Derbel, Arnaud Liefooghe, Sébastien Vérel, Hernan Aguirre, and Kiyoshi Tanaka. 2019. New features for continuous exploratory landscape analysis based on the SOO tree. In *FOGA*. 72–86.

[6] 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*. Springer, 590–602.

[7] Tome Eftimov, Anja Jankovic, Gorjan Popovski, Carola Doerr, and Peter Korošec. 2021. Personalizing performance regression models to black-box optimization problems. In *GECCO*. 669–677.

[8] Tome Eftimov, Gašper Petelin, Gjorgjina Cenikj, Ana Kostovska, Gordana Ispirova, Peter Korošec, and Jasmin Bogatinovski. 2022. Less is more: Selecting the right benchmarking set of data for time series classification. *Expert Systems with Applications* 198 (2022), 116871.

[9] Nikolaus Hansen, Anne Auger, Steffen Finck, and Raymond Ros. 2010. *Real-parameter black-box optimization benchmarking 2010: Experimental setup*. Ph. D. Dissertation. INRIA.

[10] 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 and Software* 36, 1 (2021),

114–144.

[11] Anja Jankovic and Carola Doerr. 2020. Landscape-aware fixed-budget performance regression and algorithm selection for modular CMA-ES variants. In *GECCO*. ACM, 841–849.

[12] Anja Jankovic, Gorjan Popovski, Tome Eftimov, and Carola Doerr. 2021. The impact of hyper-parameter tuning for landscape-aware performance regression and algorithm selection. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 687–696.

[13] Pascal Kerschke and Heike Trautmann. 2016. The R-Package FLACCO for exploratory landscape analysis with applications to multi-objective optimization problems. In *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 5262–5269.

[14] Ana Kostovska, Anja Jankovic, Diederick Vermetten, Jacob de Nobel, Hao Wang, Tome Eftimov, and Carola Doerr. 2022. Per-run Algorithm Selection with Warm-starting using Trajectory-based Features. *arXiv preprint arXiv:2204.09483* (2022).

[15] Ryan Dieter Lang and Andries Petrus Engelbrecht. 2021. An exploratory landscape analysis-based benchmark suite. *Algorithms* 14, 3 (2021), 78.

[16] Jing J Liang, Bo Y Qu, and Ponnuthurai N Suganthan. 2013. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Technical report Zhengzhou, China* 635 (2013), 490.

[17] Katherine M Malan and Andries P Engelbrecht. 2014. Fitness landscape analysis for metaheuristic performance prediction. In *Recent advances in the theory and application of fitness landscapes*. Springer, 103–132.

[18] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *GECCO*. 829–836.

[19] Christoph Molnar. 2020. *Interpretable machine learning*. Lulu. com.

[20] Ana Nikolikj. 2023. *RFclustgit*. https://github.com/anikolik/RF-clust

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[22] Gašper Petelin, Gjorgjina Cenikj, and Tome Eftimov. 2022. TLA: Topological Landscape Analysis for Single Objective Continuous Optimization Problem Instances. In *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, In Press.

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

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

[25] Urban Škvorc, Tome Eftimov, and Peter Korošec. 2022. Transfer Learning Analysis of Multi-Class Classification for Landscape-Aware Algorithm Selection. *Mathematics* 10, 3 (2022), 432.

[26] Jörg Stork, Agoston E Eiben, and Thomas Bartz-Beielstein. 2020. A new taxonomy of global optimization algorithms. *Natural Computing* (2020), 1–24.

[27] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.

[28] Qingshan Xu, Yang Yang, Yujun Liu, and Xudong Wang. 2017. An improved latin hypercube sampling method to enhance numerical stability considering the correlation of input variables. *IEEE Access* 5 (2017), 15197–15205.