Sensitivity Analysis of RF+clust for Leave-One-Problem-Out Performance Prediction

Ana Nikolikj, Michal Pluháček, Carola Doerr, Peter Korošec, Tome Eftimov

Sensitivity Analysis of RF+clust for Leave-one-problem-out Performance Prediction

Ana Nikolikj, Michal Pluháček, Carola Doerr, Peter Korošec, and Tome Eftimov

***Abstract*—Leave-one-problem-out (LOPO) performance prediction requires machine learning (ML) models to extrapolate algorithms' performance from a set of training problems to a previously unseen problem. LOPO is a very challenging task even for state-of-the-art approaches. Models that work well in the easier leave-one-instance-out scenario often fail to generalize well to the LOPO setting. To address the LOPO problem, recent work suggested enriching standard random forest (RF) performance regression models with a weighted average of algorithms' performance on training problems that are considered similar to a test problem. More precisely, in this RF+clust approach, the weights are chosen proportionally to the distances of the problems in some feature space. Here in this work, we extend the RF+clust approach by adjusting the distance-based weights with the importance of the features for performance regression. That is, instead of considering cosine distance in the feature space, we consider a weighted distance measure, with weights depending on the relevance of the feature for the regression model. Our empirical evaluation of the modified RF+clust approach on the CEC 2014 benchmark suite confirms its advantages over the naive distance measure. However, we also observe room for improvement, in particular with respect to more expressive feature portfolios.**

***Index Terms*—Automated Performance Prediction, AutoML, Single-Objective Black-Box Optimization, Zero-Shot Learning**

I. Introduction

In black-box optimization, supervised machine learning (ML) models are commonly used for automated algorithm selection [1]. The models use representations of the optimization problems in terms of exploratory landscape analysis (ELA) features [2] to predict the algorithm performance on the problems. Typically, regression models are used [3]. While promising results have been achieved, the models may not make accurate predictions when they are trained on problems that are not representative of the new problems, for which the best-performing algorithm shall be selected.

In the majority of previous works, the predictive power of the ML models is evaluated in leave-one-instance-out (LOIO) scenarios [4]–[6], where different instances from the same problem are present in both, training and test data. The problem instances are obtained with transformations of the same base problem class by using shifting, scaling, and/or rotation [7]. In such an evaluation scenario there is a guarantee that the training data covers the feature space also of the unseen test instances. As a result, the learned predictive model performs well on new problems.

The main challenge arises when the evaluation of the predictive model is performed in a leave-one-problem-out (LOPO) manner. That is, all the instances of the same base problem are left out for testing, and no instances of the problem are present during the training of the model. Recent studies [8], [9] show that it is difficult to generalize a predictive model for automated algorithm performance prediction trained on problems from one benchmark suite to problems from another benchmark suite.

The LOPO performance prediction can be stated as an ML task known as zero-shot learning [10]–[12]. Nikolikj et al. introduced RF+clust for LOPO algorithm performance prediction [13]. The approach calibrates the prediction obtained by a Random Forest (RF) model [14] for a given test problem with a weighted mean of the algorithm performance on problems from the training data that are the most similar to the test problem, based on their feature representation. The similarity between problems is measured using cosine similarity [15]. While RF+clust produces promising results, it struggles when problems have very similar feature representations but there is a large difference in algorithm performance on them. In addition, treating all features equally when calculating the similarity between the problem representations can be a weak point of the approach, resulting in similar problems from the training data which are not similar in reality to the test problem.

Our contribution: In this study, we analyze how using a weighted similarity measure to find similar problem instances affects the performance of RF+clust. We evaluate the performance of two weighted RF+clust variants that incorporate feature importance in the similarity measure as weights. We tested two methods for determining the feature importance: an *unsupervised method*, based on clustering, and a *supervised method* based on feature permutation. The results obtained on the CEC 2014 benchmark suite show that using a weighted cosine similarity measure with feature importance as weights can improve the performance of the RF+clust approach. Most of the problems for which new similar problems have been found with a weighted similarity measure and the algorithm

Ana Nikolikj (Email: ana.nikolikj@ijs.si), Peter Korošec (Email: peter.korosec@ijs.si) and Tome Eftimov (Email: tome.eftimov@ijs.si) are with Computer Systems Department, Jožef Stefan Institute, 1000 Ljubljana, Slovenia. Ana Nikolikj is also with the Jožef Stefan International Postgraduate School, 1000 Ljubljana, Slovenia

Michal Pluháček (Email:pluhacek@utb.cz) is with the Faculty of Applied Informatics, Tomas Bata University in Zlin, 760 01 Zlin, Czech Republic.

Carola Doerr (Email: carola.doerr@lip6.fr) is with the Sorbonne Université, CNRS, LIP 1000 Paris, France.

The authors acknowledge the support of the Slovenian Research Agency through program grant No. P2-0098, project grants N2-0239 and J2-4460, and a bilateral project between Slovenia and France grant No. BI-FR/23-24-PROTEUS-001 (PR-12040). Our work is also supported by ANR-22-ERCS-0003-01 project VARIATION.

performance on them is similar yield better results. Previously for several problems for which similar problems are not found, we can now identify them through the weighted similar measure and improve the prediction. For some problems the approach leads to worse predictions, indicating that the most important features are not expressive enough to distinguish between these problems, with significantly different algorithm behavior. Comparing both weighted variants, it follows that the one that uses the permutation feature importance provides close to the uniform contribution of the features. This is the reason it has more similar prediction results to the standard RF+clust approach.

Outline: The rest of the paper is organized as follows: Section II presents the overview of the related work, Section III introduce the two variants of RF+clust approach, the experimental design is explained in Section IV. Section V presents the results with discussion, and finally, the conclusions are presented in Section VI.

II. Related work

A. Common ML approaches for automated algorithm performance prediction

In the black-box optimization context, most of the studies performed in the direction of supervised automated algorithm performance prediction use Exploratory Landscape Analysis (ELA) [2] for calculation of the feature representation of problems. The ELA features are used as input data for the ML models. The ELA features are calculated by using mathematical and statistical techniques on a set of candidate solutions sampled from the problem decision space. Further, they are combined with different supervised ML models (e.g., Random Forest, XGBoost, Neural Networks, etc.) to predict the performance of an algorithm [3], [5], [16]. However, the evaluation of the predictive models in almost all studies on this topic is evaluated in the LOIO scenario.

Few studies that have been published in 2022, analyze models' generalization power in the LOPO scenario. Škvorc et al. [8] showed that a predictive model has lower generalization in such evaluation scenario. They have analyzed this on the real-parameter black-box optimization benchmarking (BBOB) benchmark suite [17]. In addition, they have shown that a model trained on the BBOB benchmark suite provides poor predictive results when it is used to predict the performance of a benchmark suite of artificially generated problems. Kostovska et al. have analyzed a "per-run" algorithm selection scheme [9] by using trajectory-calculated ELA features (i.e., the samples for calculating them are the samples observed by the algorithm during its run) as input data to predict the performance. They have shown that a model learned on the BBOB benchmark suite has poor predictive results for the Nevergad benchmark suite [18] and vice versa.

B. RF+clust for leave-one-problem-out (LOPO) performance prediction

RF+clust is a recently proposed approach for LOPO algorithm performance prediction. The idea behind the approach is to improve the predictions of a standard supervised model when the test problem landscape representation is not present in the training set. The approach consists of the following three steps:

- Training a regression model on a set of training problems represented by their landscape features. As the name suggests, the RF+clust approach uses random forest (RF) regression models to obtain the predicted algorithm performance $\widehat{y}_q$ for the test problem.
- The second step consists of setting a predefined similarity threshold for problem similarity. Based on the feature representation of the problems, the k-nearest problems from the training set whose similarity s with the test problem is greater or equal to the predefined threshold are selected. The number k can be different for different problems and it depends on the predefined threshold. The approach uses cosine similarity as a similarity measure. For the selected problems, the actual performance of the algorithm on them is retrieved, $y_1, y_2, \ldots, y_k$.
- The last step is calibrating the RF prediction $\widehat{y}_q$ with the performance of the algorithm on the selected training problems from the previous step and obtaining the final prediction for the test problem. This is performed by the following aggregation: $\widehat{y}_{q,\text{final}} = \left(\widehat{y}_q^i + F(y_1, y_2, \ldots, y_k)\right)/2$, where $F(\ldots) = \sum_{i=1}^{k} w_i y_i$. The weight indicates how much each of the selected problems contributes to the calibration and it is calculated based on its similarity to the test problem, $w_i = s_i / \sum_{i=1}^{k} s_i$. In cases where there are no similar problems for the predefined threshold from the training set, the prediction is only based on the RF prediction $\widehat{y}_{q,\text{final}} = \widehat{y}_q$.

III. Sensitivity analysis of RF+clust for LOPO performance prediction

In this paper, we perform a sensitivity analysis of RF+clust, where the main difference with the original RF+clust, is to learn the importance of each feature and further incorporate it in the procedure for finding the k-nearest problems from the training set, that are used to calibrate the prediction obtained by the RF model on the test set. The main motivation behind this is, the more important features to have more influence in the similarity score. Let us assume that we have p features.

Here, we use and compare two different methods for learning feature importance, one unsupervised (proposed by us) and one supervised (that is a well-established approach). Details about the approaches are provided below:

- Unsupervised learning - the problems from the training set are clustered with the full feature portfolio into m clusters. Further, the same clustering is performed p times, each time removing one feature and clustering the problems using the $p-1$ features into the same number of previous estimated m clusters. At the same time, we are counting on how many problems the obtained clusters with the full portfolio differ from the new clusters. The cumulative number of problems in which the clustering

differs (n_{diff}) is used to calculate the weight of the omitted feature. A larger number of problems indicates higher importance since omitting the feature places the problems all around the problem landscape space, while a smaller number indicates that this feature is not important in distributing the problems. After we obtain the number of problems that change their placement in the problem landscape space for each feature, we calculate their weight by using $w_i = n_{\text{diff}\,i} / \sum_{i=1}^{p} n_{\text{diff}\,i}$.

- Permutation feature importance - The permutation feature importance, perm, is defined to be the amount by which a model's performance drops when single feature values are randomly shuffled [19]. The main notion behind this procedure is that it breaks the relationship between the feature and the target, thus the drop in the model performance is an indication of how much the model depends on the feature. This technique benefits from being model agnostic and usually, the final feature importance is calculated by shuffling the feature multiple times with different permutations of the feature. The weights are then calculated as $w_i = \text{perm}_i / \sum_{i=1}^{p} \text{perm}_i$.

After obtaining the weights for each feature, the similarity of the test problem with the training problems is calculated using a weighted cosine similarity

$$\text{cosine}(u, v) = \frac{\sum_{i=1}^{p} w_i^2 u_i v_i}{\sqrt{\sum_{i=1}^{p} (w_i u_i)^2} \sqrt{\sum_{i=1}^{p} (w_i v_i)^2}}, \quad (1)$$

where $u = (u_1, \ldots, u_p)$ and $v = (v_1, \ldots, v_p)$ are the feature representations of the problems with p ELA features.

The k-nearest problems that are selected by applying the similarity threshold to the weighted cosine similarity measure, are the ones used to calibrate the RF prediction.

IV. Experimental design

For better comparison, we build our experiments on the same data as [13].

Problem portfolio. The proposed approach is evaluated on the 2014 CEC Special Sessions & Competitions (CEC 2014) benchmark suite [20]. It consists of 30 benchmark problems. The problem dimension D is set to 10.

Algorithm portfolio. Three randomly selected Differential Evolution (DE) configurations are included in the analysis. Their hyper-parameters are set as presented in [13]. The population size of the algorithm is set equal to the problem dimension (10). Each configuration is run 30 times on each CEC problem and the precision after a budget of $500D = 5000$ function evaluations has been stored. Finally, we report the median precision over all 30 runs for each pair of an algorithm configuration and a CEC problem. In the ML task, we consider the logarithm ($\log_{10}$) of the median precision as a target for prediction. Figure 1 presents $DE1$ performance (log-scale) obtained per benchmark problem on the CEC 2014 benchmark suite. The random selection of the DE configurations is because we focus on presenting the utility of the methodology, which is a method that can be used for any choice of algorithms and their hyperparameters.

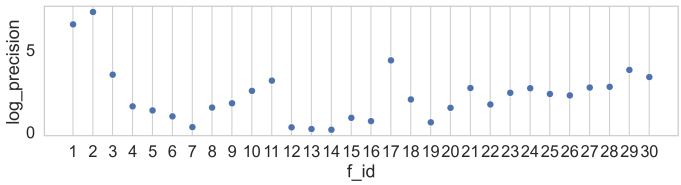


Fig. 1: Best solution precision (log-scale) obtained by DE1, for a budget of 5000 function evaluations, per problem instance in the CEC 2014 benchmark suite.

Exploratory Landscape Analysis (ELA). To extract features that describe the properties of each CEC problem, we utilize the ELA technique which is the most commonly used meta-representation for continuous single-objective optimization problems [2]. 64 ELA features are taken from a previous study [21], where they are calculated using Improved Latin Hypercube Sampling (ILHS) [22] with a sample size of $800D$ (8000) and 30 runs. The large sample size can be at a high cost, however, this was performed to reduce the randomness in the feature extraction process.

For the final analysis, as a feature portfolio, we have selected only the uncorrelated features. Pearson correlation coefficient [23] of 0.9 is used as a threshold to retrieve the highly correlated feature pairs. Next, the correlated features are divided into groups of correlated features, where all the features in the group have been highly correlated with each other. This problem can be translated into a graph problem where all the features are nodes and pairs of features satisfying correlation > 0.9 are edges. The task of finding all "correlated groups" translates into finding all complete sub-graphs in the graph with more than 2 nodes. The implementation is done with the Python package *NetworkX v.2.8.4* [24]. Then the RF model with default parameters was evaluated against every single feature from the group, the feature that resulted with the lowest mean absolute error (MAE) was chosen to be kept and the others were discarded. We need to point out here that the selected feature portfolio is different for different algorithms and even for different folds of the same algorithm.

Table I shows the ML model performance aggregated over all folds, when all features available have been used and when only the uncorrelated features (around 30 depending on the fold) have been used. Comparing the train and test errors for the different feature portfolios, we can see that the performance is only slightly degraded for the uncorrelated feature portfolio, for all the algorithms. For further experiments, we have selected the uncorrelated feature set. Performing this selection we have reduced the risk of overfitting the prediction models.

Feature importance. To learn the weights of the features required to perform the sensitivity analysis of the RF+clust approach, we use i) hierarchical clustering and ii) permutation feature importance. Both methods have been selected to test different variants of RF+clust methodology. In the case of the hierarchical clustering, the number of clusters together

TABLE I: Mean absolute error (MAE) obtained by the RF models when predicting the performance of the three DE configurations, using all and the uncorrelated features. The values in the table represent the MAE over 30 folds.

features	algorithm	MAE_train	MAE_test
uncorrelated	DE1	0.448384	1.267805
all	DE1	0.453909	1.208038
uncorrelated	DE2	0.384077	1.077129
all	DE2	0.392174	1.054833
uncorrelated	DE3	0.375334	1.023077
all	DE3	0.376666	1.018123

with the selected hyper-parameters has been estimated using hierarchical clustering [25] which is often the first option for very small data sets such as the 29 training problems used in this case. Sub-figures in Figure 2 show clustering performance (y-axis) with standard deviation over folds, for different numbers of clusters (x-axis), when using different parameters for the clustering algorithm. The implementation has been done using the *scipy v.1.9.3* Python package with $metric$ set to cosine similarity as a distance measure, $method$ set to "average" and the number of clusters m set to 4. For the permutation feature importance, the implementation has been done using the *scikit-learn v.1.0.2* package in Python by setting the number of permutations, $n\_repeats$, to 15 and $random\_state$ to one for reproducibility of the results.

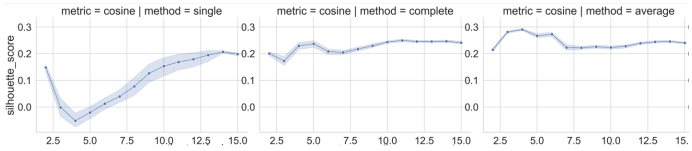


Fig. 2: Estimating the number of clusters and tuning of clustering hyper-parameters for algorithm DE1.

RF+clust model training and evaluation. We follow the previously introduced RF+clust variant, where the RF model (from the *scikit-learn* package in Python) is trained in an STR learning scenario for each algorithm configuration separately. The evaluation is performed in the LOPO scenario (i.e., 29 problems are used for training and one for testing), where the prediction errors are the absolute distances of the prediction of the precision to the true precision value of the algorithm. Since we have 30 problems, the learning process is repeated 30 times, each time one problem is out, so all steps are repeated including feature importance learning and training a regression model only on the training set.

V. Results and discussion

We apply the approach to three random DE configurations on the CEC 2014 benchmark suite. Due to space limitations, we present here some selected results for algorithm DE1 in more detail, while for the other results, similar findings were noticed and are available at [26].

A. Sensitivity analysis

Figure 3 presents the box plots of the distribution of the weights for each of the features obtained for DE1 across all 30 folds. The weights are calculated with the unsupervised feature importance approach. On the x-axis, we have all the selected features and for each one in brackets, we present in how many folds it has been selected as an uncorrelated feature. The y-axis (i.e., value) represents the calculated weight. The figure shows that half of the features selected in the uncorrelated feature portfolio for each fold are not important since they have weights equal to zero in almost all of the folds, which means that they are not used to find similar problems used for the calibration.

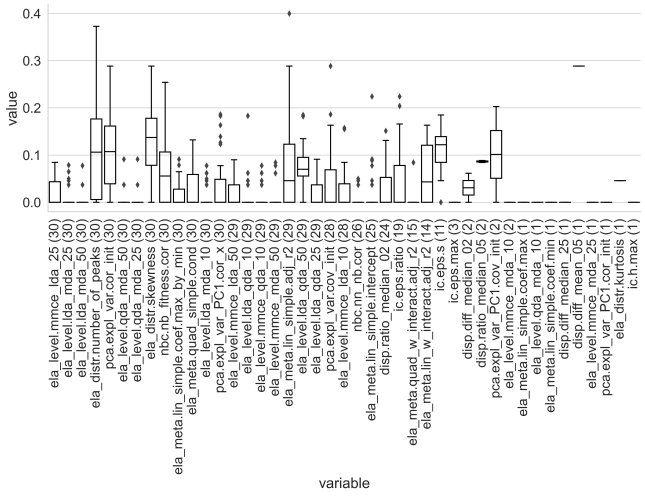


Fig. 3: Feature weights box-plot over all folds for algorithm DE1 obtained with the unsupervised feature importance. The numbers in brackets indicate in how many folds the feature was selected in the feature selection process.

Figure 4 presents the comparison of prediction errors between RF and the initial RF+clust approach for similarity thresholds of 0.5, 0.7, and 0.9 in a LOPO scenario. The first row shows the errors obtained by a standard RF model trained in the LOPO scenario. Each cell of the heatmap represents the mean absolute error obtained by the models on the test set. The numbers under the model error indicate the number of similar problems above the corresponding threshold that have been selected from the training set for the calibration of the prediction. The blank cells in the heatmap are problems for which RF+clust provides the same result as the standard RF model because for those problems we could not find similar problems from the training data to calibrate the prediction. The column names in the heatmap presented below are the problems from the CEC 2014 suite (i.e., f_id).

Figures 5 and 6 demonstrate the results obtained with the weighted RF+clust approach which uses the unsupervised feature importance or permutation feature importance respectively as feature weights to find similar problems used for calibration. The weighted approach calculates the weighted cosine similarity of the problem representations. We can notice

in both figures that the similarity between the problems can be influenced by the weighting, as the number of similar problems changes in many cases as compared to Figure 4.

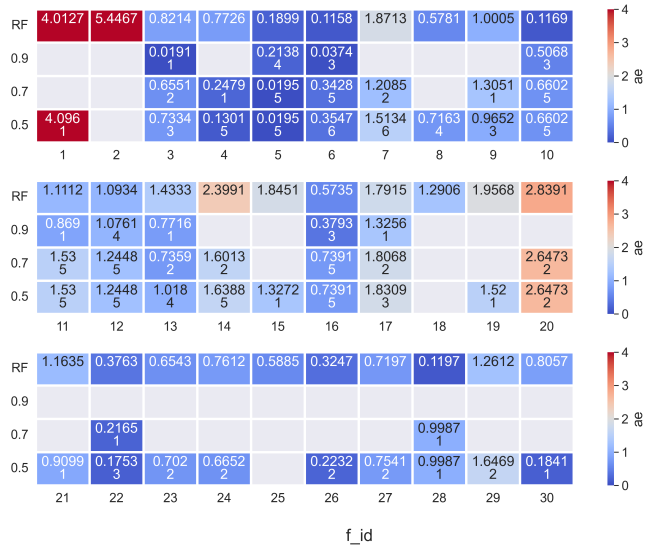


Fig. 4: Error comparison between RF and RF+clust (with **cosine similarity** and similarity threshold of 0.5, 0.7, and 0.9) in predicting the performance of DE1 for each problem instance in the CEC 2014 suite.

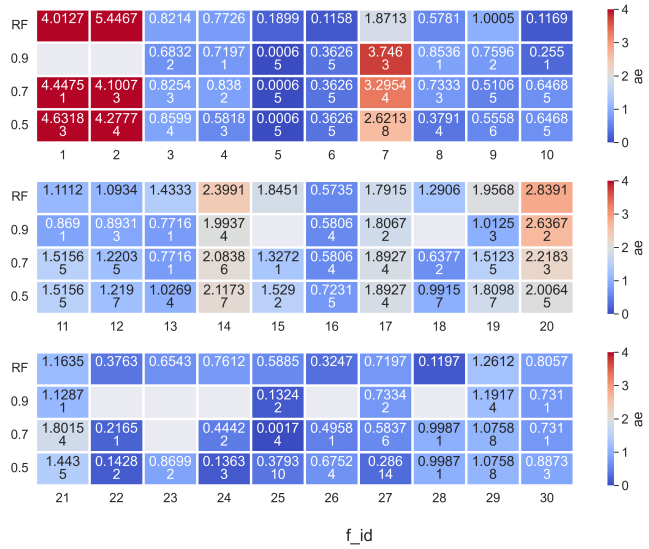


Fig. 5: Error comparison between RF and RF+clust (with **weighted cosine similarity with weights calculated by the unsupervised approach for feature importance** and similarity threshold of 0.5, 0.7, and 0.9) in predicting the performance of DE1 for each problem instance in the CEC 2014 suite

B. In-depth analysis

The heatmap of the weighted RF+clust approach with weights learned by the unsupervised feature importance (see Figure 5), shows lower errors for the following problems: 2,

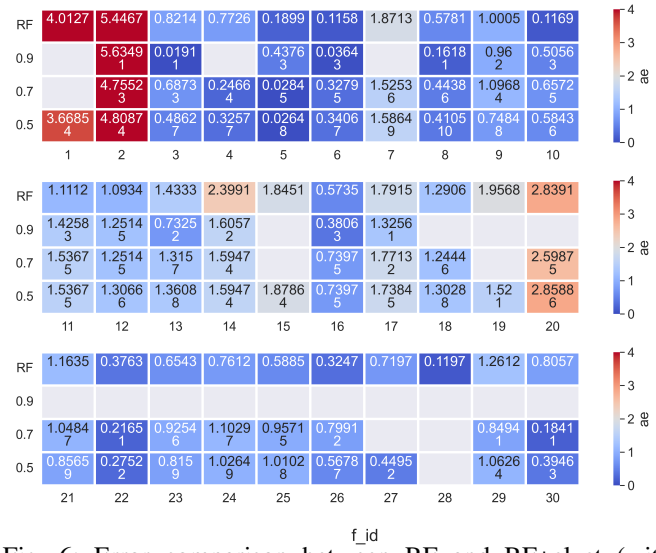
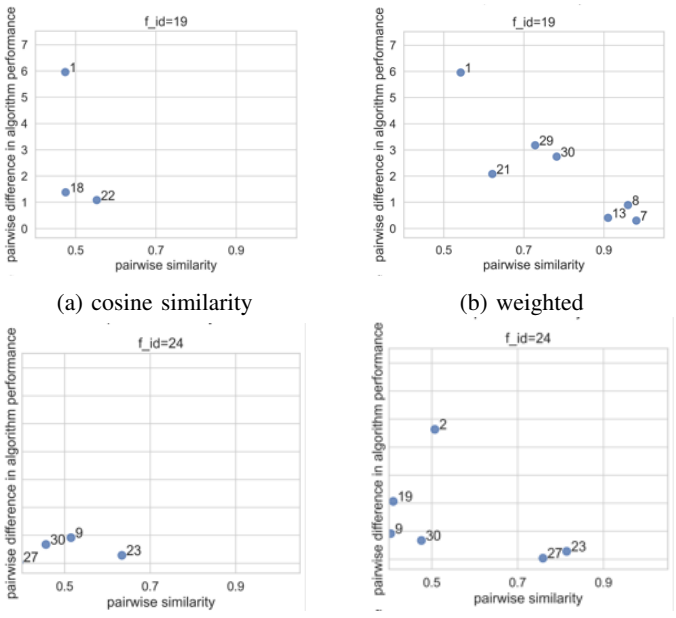


Fig. 6: Error comparison between RF and RF+clust (with **weighted cosine similarity with weights calculated by the with weights calculated by the permutation feature importance** and similarity threshold of 0.5, 0.7, and 0.9) in predicting the performance of DE1 for each problem instance in the CEC 2014 suite.

5, 9, 10, 12, 19, and 24, for all similarity thresholds compared to the initial RF+clust approach. To provide an explanation of why this happens, the 19th problem is analyzed in more detail. With the initial RF+clust, there are no similar problems found above similarity thresholds of 0.7 and 0.9 (see Figure 4). With the weighted RF+clust, we can detect three similar problems from the training data above > 0.9. The result is visible in more detail in Figures 7a and 7b, which show the relationship between the pairwise similarity of the ELA features representation (x-axis) and the absolute pairwise difference in the (ground truth) performance of the optimization algorithm (y-axis) of the 19th problem (as indicated in the plot's title) with the other problems. Figure 7a presents results of using the cosine similarity between the ELA representations, while Figure 7b uses the weighted cosine similarity. Comparing the two plots, the left one shows the initial RF+clust results where it is visible that there are no similar problems with similarity above 0.9, and the right one shows the weighted RF+clust results from which it is visible that three problems have been found as similar with a weighted cosine similarity of above 0.9. In addition, we can see that the difference in ground truth performance of the algorithm on the 19th problem and the three selected similar problems is low. The algorithm has similar behavior on these problems in reality (see also Figure 1), and using them for the calibration helps to obtain lower predictive errors. The same conclusion can be drawn for the 24th problem presented in Figures 7c and 7d with a similarity threshold of 0.7. This result indicates that using the weighted approach can help us to identify and use problems that are more effective in the calibration step.

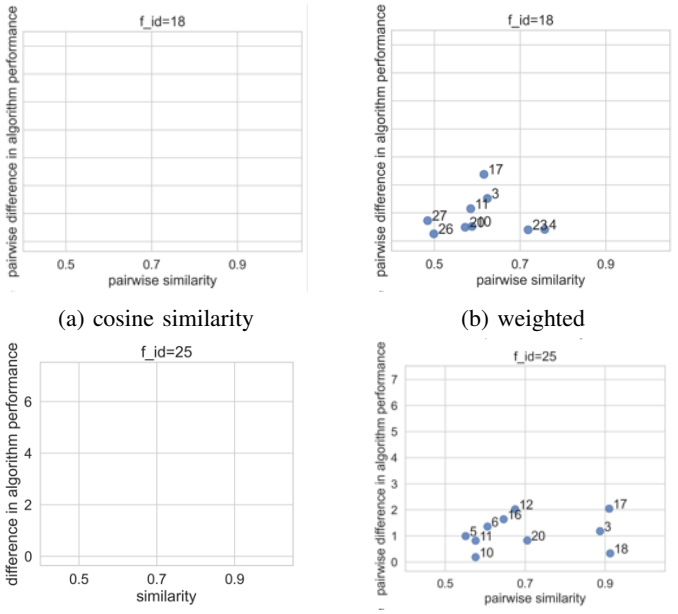
There are also problems such as the 18th and 25th for which



(a) cosine similarity (b) weighted

(c) cosine similarity (d) weighted

Fig. 7: The relationship between the pairwise cosine and weighted cosine similarity by unsupervised feature importance of the feature representations (x-axis) and the difference in DE1 performance (y-axis), for the **19th** and **24th** problem accordingly, with other problems in the CEC 2014 suite.



(a) cosine similarity (b) weighted

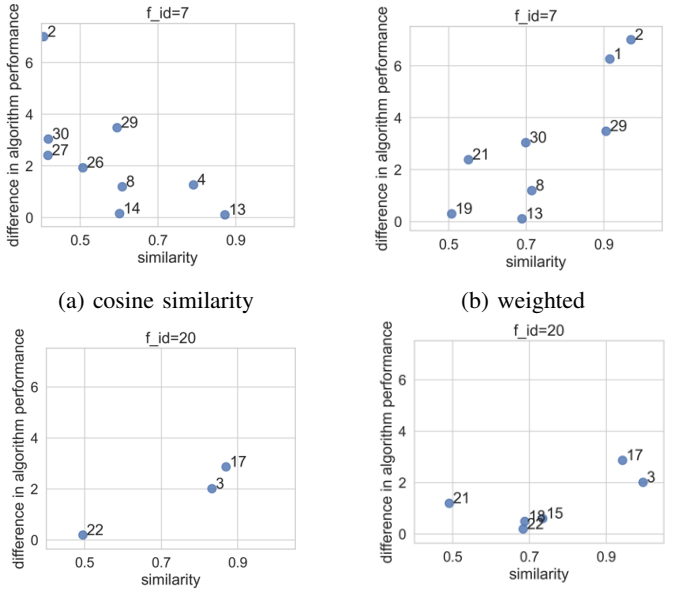
(c) cosine similarity (d) weighted

Fig. 8: The relationship between the pairwise cosine and weighted cosine similarity by unsupervised feature importance of the feature representations (x-axis) and the difference in DE1 performance (y-axis), for the **18th** and **25th** problem accordingly, with other problems in the CEC 2014 suite.

there are no similar problems found using cosine similarity (see Figure 8a and 8c accordingly). In this case, the initial RF+clust has the same prediction error as the standard RF model for all similarity thresholds (0.5, 0.7, 0.9). By applying the weighted RF+clust with the unsupervised feature importance, the 18th and 25th problems are brought closer to some similar problems from the training data in the feature space as demonstrated in Figures 8b and 8d. This helps to reduce the model error for these problems as visible in Figure 5.

For the 20th problem and similarity threshold 0.5 we seem to detect five similar problems (see Figure 5) with weighted cosine similarity and improve the model performance prediction quite a lot, compared to the case when weights are not used and only two problems are there (see Figure 4). From the figures, it is visible that with the weighted approach, three problems (the 15th, 18th, and 22nd) enriched the previous two (the 3rd and 17th) which helped the calibration process. The algorithm has very similar behavior on three problems that are brought closer with the weighted approach, as on the 20th problem. However, on the remaining two problems (3rd and 17th), we can see that even with high similarity in the landscape space, the difference in algorithm performance is larger in reality, so using their performance to calibrate the prediction yields a larger error. This indicates that there are problems for which even the most important ELA features are not expressive enough (i.e., very similar ELA landscape representation but different algorithm performance).

Figures 9a and 9b show another downside of using feature



(a) cosine similarity (b) weighted

(c) cosine similarity (d) weighted

Fig. 9: The relationship between the pairwise a) cosine and b) weighted cosine similarity by unsupervised feature importance of the feature representations (x-axis) and the difference in DE1 performance (y-axis), for the **7th** and **20th** problem accordingly, with other problems in the CEC 2014 suite.

importance as weights for finding similar problems. In cases when all the features have an equal contribution to the cosine similarity, RF+clust provides better prediction than a standard RF model. Here, it is visible that two problems (the 4th and the 13th) help the calibration for a similarity threshold over 0.7, for which the performance of the algorithm is similar to the performance achieved on the 7th problem (see Figure 4). In the case of the weighted variant of RF+clust, the prediction is worse even than a standard RF prediction. This happens since the learned weights by the unsupervised approach for feature importance brought a lot of similar problems (the 1st, 2nd, 8th, 30th) with similarity over 0.7, however, the difference in the performance of the algorithm on those problems with the performance achieved on the 7th problem is higher. This result again points out that similar landscape representation may not always be a guarantee of similar performance, which opens a new research direction of inventing new more robust problem representations that will catch the relation between the feature landscape space and the performance space of the algorithm.

Figure 6 presents the results of the RF+clust variant which uses the permutation importance as weights. We can see that using these weights also changes the number of similar problems retrieved, however the results are similar with the initial RF+clust, with slight changes. Analyzing the weights that were obtained by this approach it seems that they are close to uniform for most of the features, with only a few features showing bigger importance as shown in Figure 10. However, those features are selected as uncorrelated in the feature portfolio only for around half of the folds.

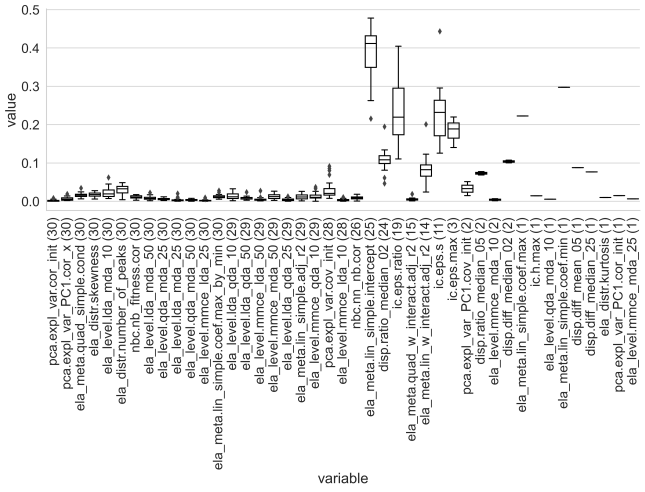


Fig. 10: Feature weights box-plot over all folds for algorithm DE1 obtained with permutation feature importance. The numbers in brackets indicate in how many folds the feature was selected in the feature selection process.

Table II provides the mean absolute prediction error across all 30 problems for a standard RF model and all variants of the RF+clust model for different similarity threshold values (0.5, 0.7, and 0.9) when they are used to predict the performance of three different DE configurations. The RF+clust approach provides better errors (i.e., bold values in Table II) than a standard RF model. Even if there are small improvements on average, from Figures 4, 5, 6, it is obvious that for some problems big improvements are obtained. For DE1, all variants of RF+clust provide a better result than the standard RF model for all similarity thresholds. For DE2, all variants with a similarity of 0.9 provide better prediction results than a standard RF model with the best result achieved when the weights are learned by permutation feature importance. In the case of DE3, the best result has been achieved for the variant when all features have the same contribution. Here, it is obvious that the result is similar to the result achieved by a standard RF model and all RF+clust variants and thresholds. To investigate why this happens, Figure 11 presents the distribution of weights for each feature in the case of DE3. From it, we can see that most of the features have weight zero, so when similar problems are searched for we are deciding only based on a few features. From the results, it follows using a 0.9 similarity threshold can provide better results for all DE configurations. The red values reported in Table II are the models with the smallest MAE for predicting each DE configuration.

TABLE II: MAE for the three DE configurations, where the bold values represent cases when the RF+clust is better than the RF model and the values in red represent the best model.

	s	DE1	DE2	DE3
RF		1.267805	1.077129	1.023077
RF + clust	0.9	**1.199533**	**1.042411**	**1.004657**
	0.7	**1.245611**	1.106662	1.063595
	0.5	**1.209003**	1.128300	1.059814
RF + clust (unsup.)	0.9	**1.223082**	**1.072002**	**1.025545**
	0.7	**1.217473**	1.120936	1.078338
	0.5	**1.221436**	1.078583	1.078182
RF + clust (perm.)	0.9	**1.194667**	**0.971472**	**1.023077**
	0.7	**1.218578**	1.099617	1.048660
	0.5	**1.180002**	**1.052838**	1.106005

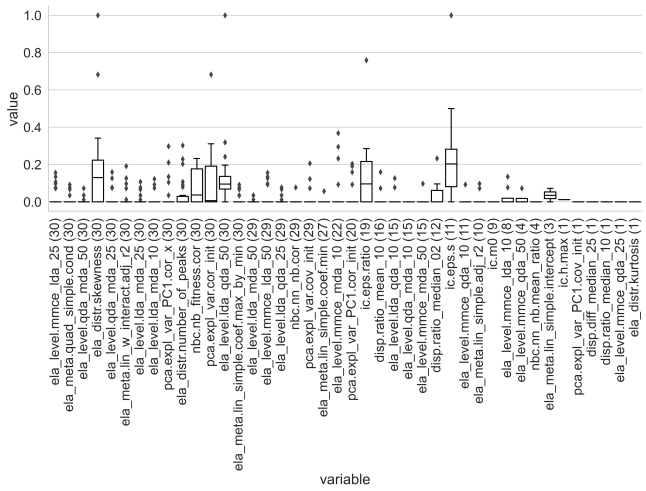


Fig. 11: Feature weights box-plot over all folds for algorithm DE3 obtained by the unsupervised feature importance.

VI. Conclusion

In this study, we performed a sensitivity analysis of RF+clust, a method for leave-one-problem-out (LOPO) performance prediction for black-box optimization algorithms. The main idea behind the RF+clust approach is to calibrate the prediction of a standard RF model with the performance achieved by the algorithm on similar problems in the training data. In the original RF+clust approach, the similarity between problems is measured by the cosine similarity of the problem landscape features, with all features contributing equally to the similarity measure. For our sensitivity analysis, we tested two new weighted RF+clust variants that use a weighted contribution of each feature to the distance measure. The weights are calculated using a feature importance method. We evaluated two feature-importance approaches: an unsupervised one, based on clustering, and a supervised one, based on permutation. In the future, other feature importance measures can be included in the analysis.

The results performed on the CEC 2014 benchmark suite indicate that RF+clust performance can be further improved by using feature importance as weights. Better results are achieved for problems for which more or new problems (compared to the original RF+clust) were found similar based on the most important features for which also the algorithm behaves similarly. For problems for which there were no similar problems, we could now successfully find problems with similar feature representations. However, there are also problems for which the proposed approach led to worse prediction results. Such results indicate that even the most important features were not expressive enough to discriminate between these problems on which the algorithm the behavior of the algorithm significantly differs.

Our results open several directions for future research. First, we are going to focus on selecting different feature portfolios for different sets of problems that can lead to robust landscape representation. Next, we are going to test problem representations that are learned by the algorithm behavior and capture the relation between the problem and the performance space. Last, but not least, we are going to test the approach using different ML models. That is, we plan to evaluate the advantage of the RF+clust approach when combined with different regression models (as opposed to the random forest models considered so far).

References

[1] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann, "Automated algorithm selection: Survey and perspectives," *Evolutionary Computation*, vol. 27, no. 1, pp. 3–45, 2019. [Online]. Available: https://doi.org/10.1162/evco_a_00242

[2] O. Mersmann, B. Bischl, H. Trautmann, M. Preuss, C. Weihs, and G. Rudolph, "Exploratory landscape analysis," in *GECCO*, 2011, pp. 829–836.

[3] P. Kerschke and H. Trautmann, "Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning," *Evolutionary computation*, vol. 27, no. 1, pp. 99–127, 2019.

[4] A. Jankovic and C. Doerr, "Landscape-aware fixed-budget performance regression and algorithm selection for modular CMA-ES variants," in *GECCO*. ACM, 2020, pp. 841–849.

[5] A. Jankovic, G. Popovski, T. Eftimov, and C. Doerr, "The impact of hyper-parameter tuning for landscape-aware performance regression and algorithm selection," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 687–696.

[6] A. Jankovic, T. Eftimov, and C. Doerr, "Towards feature-based performance regression using trajectory data," in *EvoApplications*. Springer, 2021, pp. 601–617.

[7] U. Škvorc, T. Eftimov, and P. Korošec, "Understanding the problem space in single-objective numerical optimization using exploratory landscape analysis," *Applied Soft Computing*, vol. 90, p. 106138, 2020.

[8] U. Skvorc, T. Eftimov, and P. Korosec, "Transfer learning analysis of multi-class classification for landscape-aware algorithm selection," *Mathematics*, vol. 10, no. 3, p. 432, 2022.

[9] A. Kostovska, A. Jankovic, D. Vermetten, J. de Nobel, H. Wang, T. Eftimov, and C. Doerr, "Per-run algorithm selection with warm-starting using trajectory-based features," in *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I*. Springer, 2022, pp. 46–60.

[10] J. Reis and G. Gonçalves, "Hyper-process model: a zero-shot learning algorithm for regression problems based on shape analysis," *arXiv preprint arXiv:1810.10330*, 2018.

[11] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks." in *AAAI*, vol. 1, no. 2, 2008, p. 3.

[12] W. Wang, V. W. Zheng, H. Yu, and C. Miao, "A survey of zero-shot learning: Settings, methods, and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–37, 2019.

[13] A. Nikolikj, C. Doerr, and T. Eftimov, "Rf+clust for leave-one-problem-out performance prediction," *https://arxiv.org/abs/2301.09524*, 2023.

[14] G. Biau and E. Scornet, "A random forest guided tour," *Test*, vol. 25, no. 2, pp. 197–227, 2016.

[15] A. Singhal *et al.*, "Modern information retrieval: A brief overview," *IEEE Data Eng. Bull.*, vol. 24, no. 4, pp. 35–43, 2001.

[16] R. Trajanov, S. Dimeski, M. Popovski, P. Korošec, and T. Eftimov, "Explainable landscape analysis in automated algorithm performance prediction," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2022, pp. 207–222.

[17] N. Hansen, A. Auger, S. Finck, and R. Ros, "Real-parameter black-box optimization benchmarking 2010: Experimental setup," Ph.D. dissertation, INRIA, 2010.

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

[19] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[20] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," *Technical report Zhengzhou, China*, vol. 635, p. 490, 2013.

[21] R. D. Lang and A. P. Engelbrecht, "An exploratory landscape analysis-based benchmark suite," *Algorithms*, vol. 14, no. 3, p. 78, 2021.

[22] Q. Xu, Y. Yang, Y. Liu, and X. Wang, "An improved latin hypercube sampling method to enhance numerical stability considering the correlation of input variables," *IEEE Access*, vol. 5, pp. 15 197–15 205, 2017.

[23] I. Cohen, Y. Huang, J. Chen, J. Benesty, J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," *Noise reduction in speech processing*, pp. 1–4, 2009.

[24] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[25] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012.

[26] A. Nikolikj. (2023) Rf+clust sensitivity analysis. [Online]. Available: https://github.com/anikolik/RFclust-sensitivity-analysis