Algorithm Instance Footprint: Separating Easily Solvable and Challenging Problem Instances

ANA NIKOLIKJ, Jožef Stefan International Postgraduate School Computer Systems Department, Jožef Stefan Institute, Slovenia
SAŠO DŽEROSKI, Jožef Stefan Institute & Jožef Stefan International Postgraduate School, Slovenia
MARIO ANDRÉS MUÑOZ, OPTIMA, The University of Melbourne, Australia
CAROLA DOERR, Sorbonne Université, CNRS, LIP6, France
PETER KOROŠEC, Jožef Stefan Institute, Slovenia
TOME EFTIMOV, Computer Systems Department, Jožef Stefan Institute, Slovenia

In black-box optimization, it is essential to understand why an algorithm instance works on a set of problem instances while failing on others and provide explanations of its behavior. We propose a methodology for formulating an algorithm instance footprint that consists of a set of problem instances that are easy to be solved and a set of problem instances that are difficult to be solved, for an algorithm instance. This behavior of the algorithm instance is further linked to the landscape properties of the problem instances to provide explanations of which properties make some problem instances easy or challenging. The proposed methodology uses meta-representations that embed the landscape properties of the problem instances and the performance of the algorithm into the same vector space. These meta-representations are obtained by training a supervised machine learning regression model for algorithm performance prediction and applying model explainability techniques to assess the importance of the landscape features to the performance predictions. Next, deterministic clustering of the meta-representations demonstrates that using them captures algorithm performance across the space and detects regions of poor and good algorithm performance, together with an explanation of which landscape properties are leading to it.

CCS Concepts: • **Computing methodologies → Machine learning**; **Learning latent representations**; **Supervised learning**; • **Theory of computation → Design and analysis of algorithms**.

Additional Key Words and Phrases: algorithm behavior, single-objective optimization, latent representations, supervised machine learning, explainability

1 INTRODUCTION

Many algorithms for solving continuous single-objective optimization (SOO) problems have been created and their effectiveness has mostly been evaluated through statistical analysis [28]. The commonly used approaches of computing average performance across a set of benchmark problem instances [4] or comparing distributions for a chosen performance metric [5] have been criticized for a long time [8]. However, there is still a lack of available methodologies and tools to address this issue which means that these practices continue to be used. As a result, the scientific results from such comparisons often do not generalize to new problem instances hindering the progress toward trustworthy optimization and making it difficult to understand the behavior of algorithms. This practice decreases confidence in the use of the algorithms for solving new optimization problem instances.

The primary issue in the direction of understanding algorithm behavior is that we have a limited understanding of the behavior of these algorithms and thus they are treated as black-box systems. The performance of an algorithm instance can vary significantly based on the optimization problem

Authors' addresses: Ana Nikolikj, ana.nikolikj@ijs.si, Jožef Stefan International Postgraduate School and Computer Systems Department, Jožef Stefan Institute, Slovenia; Sašo Džeroski, Jožef Stefan Institute & and Jožef Stefan International Postgraduate School, Slovenia; Mario Andrés Muñoz, OPTIMA, The University of Melbourne, Australia; Carola Doerr, Carola.Doerr@lip6.f, Sorbonne Université, CNRS, LIP6, France; Peter Korošec, Jožef Stefan Institute, Slovenia; Tome Eftimov, tome.eftimov@ijs.si, Computer Systems Department, Jožef Stefan Institute, Slovenia.

instances it is trying to solve. A deeper understanding of the interaction between the algorithm, the optimization problem, and the performance would allow us to identify properties that make a problem instance easy or challenging for a specific algorithm instance.

Our contribution: We propose a methodology to understand how problem properties and algorithm performance interact. We use meta-representations, which integrate problem properties and algorithm performance into a single vector space, created by training a machine learning regression model and analyzing feature importance. Clustering the meta-representations using a deterministic approach reveals regions of good and poor algorithm performance that define the algorithm instance footprint. Post-hoc analysis of the regions identifies the problem properties causing it. This sheds light on the algorithm's behavior and provides insights into its strengths and weaknesses. Note that this methodology is not for comparing different algorithms but for understanding each algorithm's behavior.

Related Work. The most commonly-used practice in analyzing algorithm behavior is performing a performance assessment which relies on statistical analysis, to compare performance data of different algorithm instances across a selected set of benchmark problem instances [4, 5]. The main drawback in such comparisons is that sometimes algorithm instances can be the best for some problem instances but worse for another set of problem instances, which actually affects the final statistical results. Such comparison results do not provide explanations for which algorithm instance is suitable for which problem instance and why.

Instance Space Analysis (ISA) [21, 27] has been introduced to understand the intricate relationships between the algorithm instance behavior and the problem instance properties. It identifies regions in the problem space where a specific algorithm performs well and solves problems easily, by linking problem landscape features with a single performance metric. For this purpose, ISA uses a dimension reduction technique that encourages linear trends to appear both in the features and algorithm performance distributions to support visualizations. Then, it uses a supervised classifier to identify the areas in which the evidence of good performance is the strongest, defined as whether the instances are solvable or not within some precision of the global optima. The areas are characterized by their size, density, and purity, giving metrics of the algorithm's strength relative to the diversity of the benchmarks. Therefore, ISA represents an improvement in comprehending algorithm behavior, describing complex interactions in a linear manner.

Another way to understand the interactions between algorithm instances and problem instances is through the application of supervised regression models for algorithm configuration (finding the best hyper-parameters of an algorithm instance) [1, 3] and algorithm selection (selecting the best algorithm for a given problem instance) [11, 13, 15]. In this approach, algorithm performance is predicted based on the features extracted from the problem instance landscape, which serve as problem-instance meta-representations. Most studies in this field use meta-learning [31] to train a single regression model to predict the performance of an algorithm instance on all problem instances. Recent studies [29, 30] have explored an explainable workflow for automated algorithm performance prediction. They utilized feature importance analysis to identify the crucial landscape features that predict an algorithm's instance performance on a global scale (i.e., a set of benchmark problem instances) and a local scale (i.e., a specific problem instance).

2 ALGORITHM INSTANCE FOOTPRINT

Let us assume that we have a set of benchmark problem instances represented by their landscape features and linked to the performance of an algorithm instance achieved on them. The set is further split into train and test data sets. The term "algorithm instance footprint" refers to the regions (i.e., sets of problem instances) where an algorithm instance performs well or poorly, with accompanying

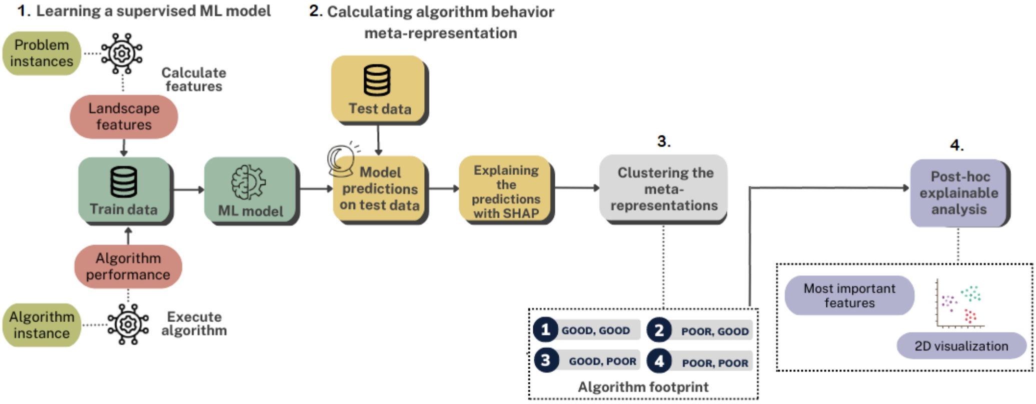


Fig. 1. Flowchart of the methodology for calculating and analyzing algorithm instance footprint.

identification of the problem landscape properties that contribute to this performance variation. The methodology for generating it consists of four steps (see Figure 1):

1) Learning a supervised ML model – Train a supervised ML regression model with the train data, which will predict the performance of the algorithm instance for each problem instance based on its landscape feature representation. Use this model to predict the algorithm instance's performance on the test data set.

2) Calculating algorithm behavior meta-representations – Once the predictions have been made on the test data set, we apply explainable techniques to assess the importance of the landscape features that impact the algorithm's performance prediction for each problem instance independently. These contributions can be used as a meta-representation of the algorithm's behavior on a specific problem instance. We have opted for the SHAP method [26] since it offers localized explanations that incorporate both the problem instance's landscape features and the algorithm instance's performance.

3) Clustering the meta-representations – By computing meta-representations that depict the algorithm instance's behavior on each problem instance, we can categorize the problem instances into two sets: those that are easy for the algorithm instance to solve, and those that are challenging. We have established a deterministic approach to accomplish this, resulting in more straightforward findings. The meta-representations are clustered into four groups using deterministic clustering, based on two factors: i) poor or excellent optimization algorithm instance performance and ii) poor or excellent ML prediction. In both situations, apriori set thresholds are employed to differentiate between poor and excellent performance. For ground truth performance, this implies that an error of no more than a predetermined target t is required for excellent algorithm performance. For the ML model, it is determined that excellent predictive performance is achieved when the error is within $p\%$ of the predicted true precision. Below are explanations of the clusters.

(Good, Good): This pertains to the depiction of algorithmic behavior, which identifies the problem instances where the algorithm instance's discovered solution quality is high, and the ML model accurately predicts the performance with a negligible error. This situation arises when the optimization algorithm easily solves a problem instance, and the ML algorithm recognizes this behavior.

(Poor, Good): This involves behavior observed on problem instances where the algorithm instance's solution quality is low, yet the ML algorithm correctly predicts it with minimal error.

This scenario arises when a problem instance proves to be challenging for the optimization algorithm instance to solve, and the ML algorithm identifies this behavior.

(Good, Poor): This identifies the problem instances where the algorithm instance's discovered solution quality is high, but the ML algorithm failed to predict it.

(Poor, Poor): This encompasses problem instances where the algorithm instance's solution quality is low and the ML algorithm failed to predict this behavior.

4) Post-hoc explainable analysis – Once the clusters have been identified, a post hoc explainable analysis is performed. This analysis clarifies which landscape features make the problem instances easily solved or challenging to solve for the algorithm instance. It is delivered by identifying the most important features of each of the clusters from the previous step, and also by providing 2D visualization of the meta-representations in which the algorithm performance and the most important feature values are visualized across the space.

3 EXPERIMENTAL DESIGN

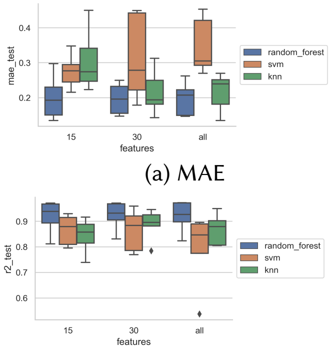
Problem portfolio. The study uses the BBOB (i.e., COCO) benchmark suite [9, 10]. It features 24 noise-free, single-objective optimization problems, which can be altered by scaling and translating in the objective space to create different instances. This work uses the first five instances of each problem, totaling 120 benchmark instances. The problem dimension is set to 10, $D = 10$.

Landscape features. We select the most commonly used landscape features that are used to describe the properties of single-objective optimization problems, known as ELA features [19]. Their calculation has been taken from a previous study [16]. A total of 64 features were selected, including classical ELA features [19], Dispersion [17], Information Content [20], Nearest Better Clustering [12], and Principal Component Analysis [14].

Algorithm portfolio. Three randomly selected Differential Evolution (DE) configurations have been selected as the algorithm portfolio just to present how the proposed methodology works. The configurations (DE1, DE2, and DE3) are taken from a previous study [23], where each configuration is presented in more detail including its strategy, F, and Cr values.

Performance data. The study focuses on the fixed-budget performance scenario, where the target precision of the algorithm (i.e., the distance between the best solution and the estimated optimal solution) is used as a performance metric. The logarithm of the precision is calculated to capture the distance level to the optimum [11]. The budget has been set on $500D$ function evaluations. Each configuration has been run 30 times and the median reached precision is used as an approximation of its performance.

Predictive models. To find a good-performing supervised regression model, we evaluate three different regression families: Random Forest [2], Support Vector Machines (SVM) [24], and K-Nearest Neighbours (KNN) [25]. Each one has been tested with different feature portfolios selected by the SHAP method [26]. To select the feature portfolio, first, a model is trained with all features, then the SHAP feature importance is calculated, and finally the top most important features are selected as indicated by the Shapely scores. The models have been evaluated in stratified five-fold cross-validation, where one instance from each problem has been left for testing and the others four remain in the training data. Box plots depicting model performance on the test data (as measured by MAE and R2 score) for all regression models and different feature portfolios across the five folds are shown in Figure 2. The box plots demonstrate that the RF is robust to the different folds, while the performance of KNN and SVM appears more variable. The result is consistent across all feature portfolios. Using the results, we decide to use the RF model with 30 ELA features to show how the methodology for generating the algorithm footprint works. We need to highlight here that the footprint is calculated five times, for each fold separately. This allows us to investigate the robustness of the algorithm instance performance on the transformations (e.g., shifting or scaling) applied to generate different problem instances.



(a) MAE

(b) R2score

Fig. 2. Box-plot showing the distribution of model performance over the test portion of the five folds: (a) MAE, (b) R2 score, for different feature portfolios of most important features as identified by the SHAP method, when predicting the performance of DE1.

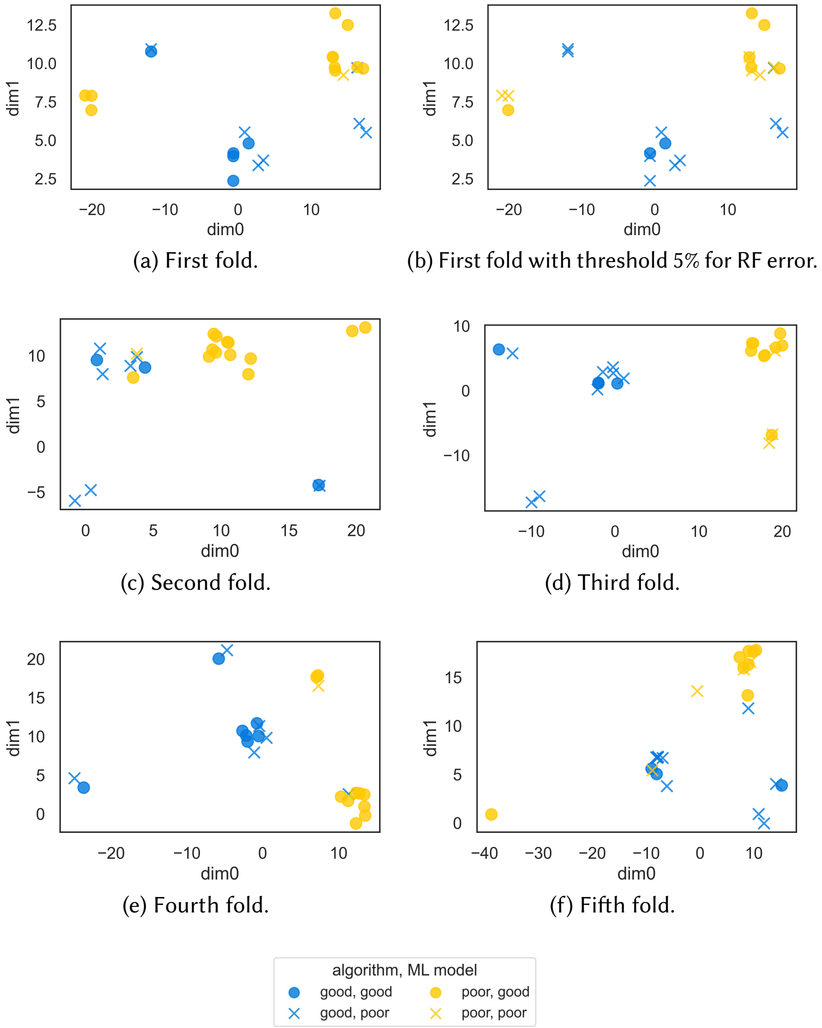
4 RESULTS AND DISCUSSION

Table 1. Distribution of the BBOB problem instances across the deterministic clusters for each fold.

model	fold number	(good, good)	(good, poor)	(poor, good)	(poor, poor)
RF	1	16, 19, 20, 21, 22	1, 2, 5, 14, 17, 18, 23	3, 4, 6, 7, 8, 9, 10, 11, 12, 15, 24	13
RF	2	19, 20, 21	1, 2, 5, 14, 17, 22, 23	3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 24	18
RF	3	19, 20, 21, 22	1, 2, 5, 14, 16, 17, 18, 23	3, 4, 6, 8, 9, 12, 13, 15, 24	7, 10, 11
RF	4	5, 16, 18, 19, 20, 21, 22	1, 2, 7, 14, 17, 23	3, 4, 6, 8, 9, 10, 12, 13, 15, 24	11
RF	5	19, 20, 21	1, 2, 5, 7, 14, 16, 17, 22, 23	6, 8, 9, 11, 12, 13, 15, 24	3, 4, 10, 18

To present how the proposed methodology can be used for understanding algorithm instance behavior, we select one DE configuration (DE1) for which we show the results and their interpretation in detail, while the results for the other two configurations (DE2 and DE3) are available at our GitHub repository [22], due to the page limits. We fix a target precision, t, to the median precision calculated over the training problem instances, to define if the algorithm instance can solve or not the problem instance within it, and a percentage of $p = 15\%$ that defines if an ML predictive model provides a good prediction within 15% error. We need to point out here that these values (t and p) are chosen as such only for illustration purposes and should be appropriately set according to the scenario in which the proposed methodology will be used. For example, the t value should be set according to the acceptable optimization accuracy for the specific problem instance and the p value should be set according to the acceptable model prediction accuracy. The acceptable model prediction accuracy depends on the application being solved, which are the tolerance levels for which a good prediction is acceptable. In this way, the methodology can be used to investigate the algorithm instance's strengths and weaknesses as required by the user.

DE1 footprint. Figure 3 presents the 2D visualization of the DE1 footprint, for each fold separately. The footprints consist of four deterministic clusters that are obtained by pairing the ground truth algorithm instance performance and the ML performance. The most interesting are the following two combinations $(good, good)$ and $(poor, good)$. For the problem instances that belong to $(good, good)$ the algorithm instance solves them within the specified target t, and the ML predicts this behavior with an error of $p = 15\%$. In the case of $(poor, good)$ the algorithm instance cannot solve the problem instances within the specified target, however, the ML model predicts its behavior



(a) First fold.

(b) First fold with threshold 5% for RF error.

(c) Second fold.

(d) Third fold.

(e) Fourth fold.

(f) Fifth fold.

Fig. 3. 2D UMAP [18] visualization of the algorithm footprints obtained with the deterministic clustering, on the test portion of each of the five folds. The tolerance error for the RF model is within 15%. The *blue* color represents regions of good algorithm performance, and the *yellow* to regions of poor algorithm performance. The marker shape corresponds to good (O) and poor (X) ML model performance as indicated by the legend at the bottom of the plot.

within an error of 15%. For these two combinations, we can provide additional explanations for the algorithm behavior. For the $(good, good)$, we can see which landscape features make those problem instances easily to be solved, while in the case of $(poor, good)$ which landscape features make those problem instances challenging to be solved. In the case of $(good, poor)$ and $(poor, poor)$, the ML model cannot predict if the algorithm is able to solve the problem instances within the specified target $(good)$ or not $(poor)$ within an error of 15%. Because of this, for these two combinations, we are not able to find which landscape features are relevant to the algorithm behavior.

Table 1 presents the distribution of the BBOB problem instances across the four clusters for each fold separately. From the results, we can conclude that DE1 has stable performance on the

19th (i.e., Composite Griewank-Rosenbrock Function), 20th (i.e., Schwefel Function), and 21st (i.e., Gallagher's Gaussian 101-me Peaks Function) BBOB problem classes. This comes from the fact that no matter the different transformations (e.g., shifting, scaling) that are applied to generate a problem instance of those problems that are parts of different folds, the algorithm instance is able the find a solution with the specified target and the RF model can predict it within an error of 15%. For the 6th, 8th, 9th, 12th, 15th, and 24th BBOB problem classes (please find their names in [7]), the algorithm instance is not able to solve them within the specified target, however, the RF model can predict this behavior within an error of 15%. This result indicates that no matter which transformations are applied to the base problem class to define the problem instances across different folds, the algorithm instance is not able to solve them. In the case of the $(good, poor)$ cluster, the algorithm instance can solve the problem instances from the 1st, 2nd, 14th, 17th, and 23rd problem classes, however, the RF model cannot predict this behavior across all folds.

There are also some interesting problem classes that depending on the transformations used to define the problem instances across different folds, the problem instances change their clusters. For example, the first, third, and fourth problem instances from the 22nd BBOB problem belong to the $(good, good)$ cluster. However, the second and the fifth instances of the same problem are presented in the $(good, poor)$ cluster. This result indicates that the algorithm instances for all problem instances can find a solution within the specified target. The difference is that the RF model cannot predict this behavior for the second and the fifth instance within an error of 15% (the ML errors are 45% and 17% for the 2nd and 5th instances respectively.). Another example is the 5th BBOB problem class. Except for the fourth problem instance from this problem class, for all the remaining the RF error is greater than 15% (for the fourth one is 8%).

Other examples contain the transition from $(poor, good)$ to $(poor,$
$poor)$: the 3rd, 10th, 11th, and 13th BBOB problem classes. All these problems correspond to problem instances that DE configuration is not able to solve within the specified target. The main difference is the performance of the RF model. For the 11th problem, the RF model can predict the behavior for the first, second, and fifth instances, while for the third and fourth is not able to predict it (i.e., RF errors above 15%). This means that in the case of the first, second, and fifth instances we can further provide explanations of which landscape features make them difficult to be solved, while we cannot provide explanations for the third and fourth instances. Further, by performing an explainable post-hoc analysis, if different features are important between different folds for this problem class, this indicates that the important features for the third and fourth problem instances are "wrong" features that are leading to poor prediction. If there is no difference between the important features across the folds, it means that the RF predictive model does not have enough power (i.e., confidence) to provide explanations for the third and fourth problem instances. Similar explanations are also present for the 3rd (ML error is 17% for the fifth instance of this problem), 10th (RF errors are 18% and 44% for the third and fifth instances of this problem), and 13th (i.e., the RF error for the first instance of this problem is around 17%) problem.

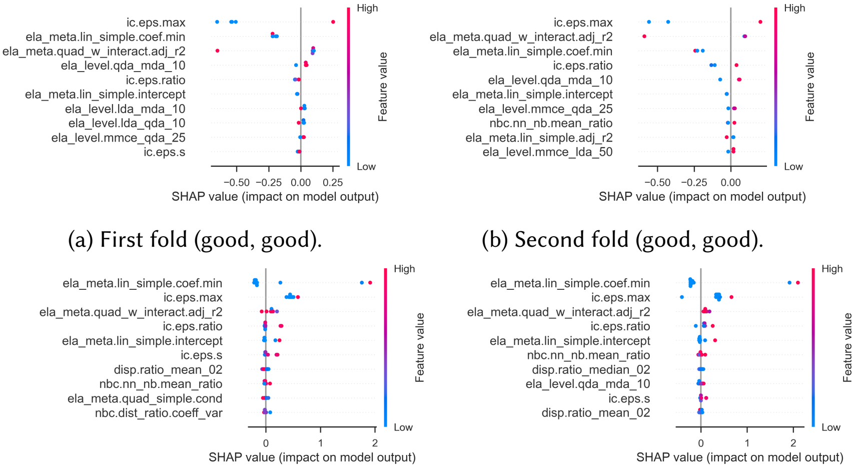
Some of the transitions from $(good, good)$ to $(good, poor)$ and vice-versa, and also from $(poor, good)$ to $(poor, poor)$ and vice versa, are happening only when the RF error is in some close ϵ-neighborhood with the selected percentage, $p = 15\%$.

The problem instances on the 7th and the 18th problem classes are distributed across most of the clusters. This result points out that the algorithm instance does not have stable performance on them. It is either able to solve or not the problem instance within the specified target, which further can be a problem for the ML model to make a prediction.

In general, analyzing the footprints across all folds, we can conclude that the footprints make a clear distinction between *good* vs. *poor* algorithm instance performance (i.e., placing $(good, good)$ to $(good, poor)$ problem instances together vs. $(poor, good)$ to $(poor,$

poor) together). The second dimension, which is the ML model performance, only guarantees confidence in providing further explanations for problem instances that are predicted in the tolerance error.

Sensitivity analysis with regard to the tolerance percentage of the ML model error. To see the influence of the selected percentage for the RF model error, in Figure 3b we present the same footprint as in Figure 3a generated on the data from the first fold with a difference that the percentage for the RF model error is set at $p = 5\%$. As was expected, the main transitions only happen from $(good, good)$ to $(good, poor)$, and from $(poor, good)$ to $(poor, poor)$. If the target precision for the true algorithm instance increases and the RF model is fixed, the possible transitions are from $(good, good)$ to $(poor, good)$, and from $(good, poor)$ to $(poor, poor)$, otherwise, the transition is in the other direction.



(a) First fold (good, good).

(b) Second fold (good, good).

(c) First fold (poor, good).

(d) Second fold (poor, good).

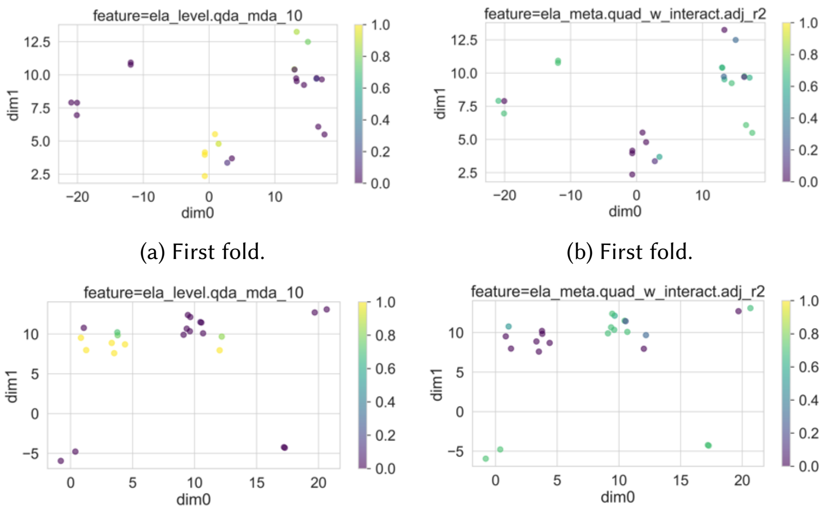
Fig. 4. The 10 most important ELA features and their prediction influence for the test instances of the first and second fold for the (good, good) and (poor, good) clusters. Each point on the plot is a Shapley value for a feature and an instance. Its position on the y-axis is determined by the feature and on the x-axis by the Shapley value. The color represents the value of the feature from low to high.

Post-hoc explainable analyses. Next, we provide post-hoc analysis to estimate the landscape features that make the problem instances easy to be solved $(good, good)$) and or challenging to be solved ($(poor, good)$). Figure 4 presents the 10 most important landscape features, for the first and second fold. The plots in this figure also depict both positive and negative relations with the target precision that is being predicted. The dots shown in the plots represent all instances from the selected folds. The ELA features are ordered based on their importance, with the most important feature being listed first. The color coding used reflects the magnitude of the ELA feature value, with higher values represented in red and lower values in blue. The effect of the ELA feature value on the target variable prediction can be seen by its horizontal placement. From the figure, it is obvious that the features for the $(good, good)$ cluster across both presented folds are overlapping for seven out of 10 ELA features and the same patterns of influence are presented. In the case of the $(poor, good)$ cluster, both folds are overlapping in eight out of 10 ELA features with similar patterns of influence. Comparing the $(good, good)$ vs. $(poor, good)$ in both folds separately, we can see that the overlapping is in a few ELA features, however, even the influence patterns of those which are overlapping are different (e.g., ic.eps.max, ela_meta.lin_simple.coef.min).

To go into more detail, we randomly selected two ELA features (*ela_level.qda_mda_10* and *ela_meta.quad_w_interact.adj_r2*) and present their distributions across the algorithm instance footprint (see Figure 5). The distributions of the other ELA features are available in our GitHub repository [link omitted during the review].

In the case of the algorithm footprint generated on the first test fold, it is obvious that *ela_level.qda_mda_10* has higher values for the problem instances that belong to $(good, good)$ cluster and lower values for the problem instances that belong to $(poor, good)$ cluster. The opposite is true for the *ela_meta.quad_w_interact.adj_r2*, where lower values are for problem instances that belong to $(good, good)$ and higher values are related to problem instances that belong to the $(poor, good)$ cluster.

For the algorithm footprint generated on the second test fold, the *ela_level.qda_mda_10* has higher values for the problem instances from the $(good, good)$ cluster and lower values for those that belong to the $(poor, good)$ cluster. In the case of the *ela_level.qda_mda_10* feature, lower values are associated with the problem instances from the $(good, good)$ and medium values are related to the instances from the $(poor, good)$ cluster.



(a) First fold.

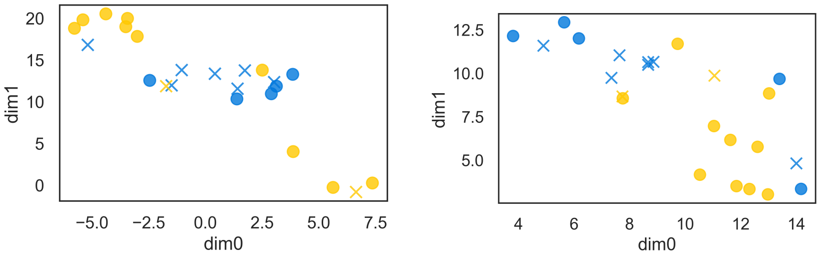
(b) First fold.

(c) Second fold.

(d) Second fold.

Fig. 5. The distribution of two randomly selected (from the top 10) ELA features across the algorithm instance footprint. The color in the plots represents the normalized feature values.

DE1 footprints generated by different ML models. Figure 6 presents the footprints generated for DE1 and the first fold by using different ML predictive models, KNN and SVM. The tolerance error for both ML models is set at 15% in order to benchmark the footprints with the footprint generated by the RF model. Comparing the footprints with the footprint presented in Figure 3a, it is obvious that the distribution of the problem instances in the space is different since the meta-representations are model-specific and generated by using different supervised ML models. Table 2 presents the distribution of the BBOB problem instances across the footprints generated for all folds by RF, KNN, and SVM. All ML models provided similar results on the first fold which is also visible by the distribution of the problem instances across all four deterministic clusters. No matter which model is used, the clusters are almost the same. The distributions of the problem instances slightly change for the other remaining folds, which indicates the model-specific aspect and the importance



(a) First fold of KNN.

(b) First fold of SVM.

Fig. 6. 2D UMAP visualization of the algorithm footprints obtained with the deterministic clustering on the test portion of the first fold. The tolerance error for the KNN and SVM model is within 15%. The *blue* color represents regions of good algorithm performance, and the *yellow* to regions of poor algorithm performance. The marker shape corresponds to good (O) and poor (X) ML model performance.

of selecting a good-performing ML model. However, similar distributions that are achieved across the folds support the fact that the proposed methodology can be used to analyze the strengths and weaknesses of algorithm instance behavior. Currently, the explanations (i.e., important landscape features) are model-specific, they depend on the selection of the supervised regression model. In the future, we are planning to find the intersection of important landscape features across footprints generated with different ML models in order to go through a model-agnostic approach.

Table 2. Distribution of the BBOB problem instances across the deterministic clusters for each fold.

model	fold number	(good, good)	(good, poor)	(poor, good)	(poor, poor)
RF	1	16, 19, 20, 21, 22	1, 2, 5, 14, 17, 18, 23	3, 4, 6, 7, 8, 9, 10, 11, 12, 15, 24	13
KNN	1	2, 16, 18, 19, 20, 21, 23	1, 5, 14, 17, 22	4, 6, 7, 8, 9, 10, 11, 12, 15, 24	3, 13
SVM	1	16, 19, 20, 21, 22	1, 2, 5, 14, 17, 18, 23	3, 4, 6, 7, 8, 9, 10, 11, 12, 24	13, 15
RF	2	19, 20, 21	1, 2, 5, 14, 17, 22, 23	3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 24	18
KNN	2	5, 17, 19, 20, 23	1, 2, 14, 21, 22	3, 4, 6, 7, 8, 9, 10, 11, 12, 16, 24	13, 15, 18
SVM	2	19, 21	1, 2, 5, 14, 17, 20, 22, 23	6, 8, 9, 12, 15, 16, 18, 24	3, 4, 7, 10, 11, 13
RF	3	19, 20, 21, 22	1, 2, 5, 14, 16, 17, 18, 23	3, 4, 6, 8, 9, 12, 13, 15, 24	7, 10, 11
KNN	3	1, 16, 18, 19, 20, 21, 22, 23	2, 5, 14, 17	6, 8, 9, 10, 12, 13, 24	3, 4, 7, 11, 15
SVM	3	6, 22	1, 2, 5, 14, 17, 18, 19, 20, 21, 23	9, 12, 13, 24	3, 4, 6, 7, 8, 10, 11, 15
RF	4	5, 16, 18, 19, 20, 21, 22	1, 2, 7, 14, 17, 23	3, 4, 6, 8, 9, 10, 12, 13, 15, 24	11
KNN	4	1, 7, 16, 19, 20, 21, 22, 23	2, 5, 14, 17, 18	4, 6, 8, 9, 10, 12, 24	3, 11, 13, 15
SVM	4	16, 20, 21	1, 2, 5, 7, 14, 17, 18, 19, 22, 23	6, 9, 11, 13, 24	3, 4, 8, 10, 12, 15
RF	5	19, 20, 21	1, 2, 5, 7, 14, 16, 17, 22, 23	6, 8, 9, 11, 12, 13, 15, 24	3, 4, 10, 18
KNN	5	1, 14, 16, 19, 20, 21, 22, 23	2, 5, 7, 17	4, 6, 8, 9, 11, 12, 15, 24	3, 10, 13, 18
SVM	5	5, 16, 19, 21, 22	1, 2, 7, 14, 17, 20, 23	3, 8, 9, 10, 11, 12, 18, 24	4, 6, 13, 15

Discussion. Our methodology is proposed as an exploratory tool for analyzing and understanding the strengths and weaknesses of a selected algorithm instance. It cannot be used to benchmark different algorithm instances based on their footprints. This comes with the fact that the footprint of each algorithm is generated using explanations of a single-target regression model, which means that each footprint is generated in its own vector space. As future work, we are going to purpose footprints that can be used for transparent benchmarking where the algorithm performance prediction will be done as multi-task learning where the performance of different algorithm instances will be treated as multiple learning tasks that are solved at the same time while exploiting commonalities and differences across tasks.

In our experiments, the predefined target precision for the algorithm instance performance, t, and the tolerance percentage of the ML error, p, were set only for illustration purposes. Those parameters can be set by the researchers/users depending on their application requirements. They can also be changed in order to explore the sensitivity of the footprints. In addition, when training the ML predictive model for algorithm instance performance prediction, more advanced techniques based on AutoML [6] are recommended to find the best ML predictive model (i.e., in our experiments we use RF with default parameters for illustration purposes).

We illustrate the generation of the algorithm instance footprint using $10D$ problem instances. In the future, footprints for the same algorithm instance can be generated for a different dimension to explore the distribution of the problem instances across the footprint and also to explore the importance of the landscape features when the dimension of the problem instances increases.

We use ELA features that provide some information but are still low-interpretable. In the future, this methodology can also be performed using another portfolio of landscape features (e.g., high-level features) in order to provide more human-interpretable explanations. Currently, the explanations are model-specific, they depend on the selection of the supervised regression model. In the future, we are planning to generate footprints for the same algorithm instance with regard to different regression models and try to find the intersection between them in order to go through a model-agnostic approach.

While sharing philosophical foundations, the proposed methodology differs in approach to ISA [21]. For example, ISA attempts to give both an indication of the diversity of the benchmark instances and serve as a tool for developing hypotheses on the strengths and weaknesses of the algorithms. For this, ISA finds a common space for all algorithms, by selecting features that, after a linear projection, are the most predictive of performance on average across the portfolio. Moreover, ISA constructs and measures the footprints as regions of the space through a combination of clustering and geometric methods [27], to account for the diversity of the instances. Nevertheless, no investigation of multiple algorithm instances through ISA has been made in SOO. The interpretation in ISA is human-driven, with an analysis of the visualizations by the researchers being an important step, with a standardized post-hoc analysis not being part of the ISA yet. A comparison between the results of the proposed methodology and ISA is left for further research.

5 CONCLUSIONS

In the context of black-box optimization, it's crucial to comprehend the reasons why an algorithm instance works well on some problem instances and fails on others. We introduce a method for creating an algorithm instance footprint, which consists of dividing the problem instances into two sets: those that are easily solvable and those that are challenging. This is achieved by linking the algorithm's behavior to the properties of the problem landscape, providing explanations of why some problem instances are easier or harder. Our methodology employs meta-representations, which embed the properties of the problem instances and the performance of the algorithm into the same vector space. These meta-representations are obtained through training a supervised machine learning regression model and examining feature importance. Additionally, deterministic clustering of the meta-representations reveals regions of good and poor algorithm performance, along with an explanation of which landscape properties are responsible. This analysis enables us to gain insight into the strengths and weaknesses of the algorithm instance and move away from treating it as a black-box.

ACKNOWLEDGMENTS

The authors acknowledge the support of the Slovenian Research Agency through program grant No. P2-0098 and P2-0103 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), as well as the EC through grant No. 952215 (TAILOR). Our work is also supported by ANR-22-ERCS-0003-01 project VARIATION.

REFERENCES

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

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

[3] Marcelo de Souza and Marcus Ritt. 2022. Improved regression models for algorithm configuration. In *Proceedings of the Genetic and Evolutionary Computation Conference*. Association for Computing Machinery, New York, NY, USA, 222–231.

[4] Joaquín Derrac, Salvador García, Daniel Molina, and Francisco Herrera. 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation* 1, 1 (2011), 3–18.

[5] Tome Eftimov, Peter Korošec, and Barbara Koroušić Seljak. 2017. A novel approach to statistical comparison of meta-heuristic stochastic optimization algorithms using deep statistics. *Information Sciences* 417 (2017), 186–215.

[6] Matthias Feurer, Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. *Hyperparameter Optimization*. Springer International Publishing, Cham, 3–33.

[7] Steffen Finck, Nikolaus Hansen, Raymond Ros, and Anne Auger. 2010. *Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions*. Technical Report. Citeseer.

[8] Nicholas G Hall and Marc E Posner. 2010. The generation of experimental data for computational testing in optimization. In *Experimental methods for the analysis of optimization algorithms*. Springer, Berlin, Heidelberg, 73–101.

[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, Tome Eftimov, and Carola Doerr. 2021. Towards feature-based performance regression using trajectory data. In *EvoApplications*. Springer International Publishing, Cham, 601–617.

[12] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. 2015. Detecting funnel structures by means of exploratory landscape analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. Association for Computing Machinery, New York, NY, USA, 265–272.

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

[14] Pascal Kerschke and Heike Trautmann. 2019. Comprehensive feature-based landscape analysis of continuous and constrained optimization problems using the R-package flacco. In *Applications in Statistical Computing*. Springer International Publishing, Cham, 93–123.

[15] 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. In *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I*. Springer International Publishing, Cham, 46–60.

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

[17] Monte Lunacek and Darrell Whitley. 2006. The dispersion metric and the CMA evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. Association for Computing Machinery, New York, NY, USA, 477–484.

[18] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[19] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In *GECCO*. Association for Computing Machinery, New York, NY, USA, 829–836.

[20] Mario A Muñoz, Michael Kirley, and Saman K Halgamuge. 2014. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE transactions on evolutionary computation* 19, 1 (2014), 74–87.

[21] Mario A Muñoz and Kate A Smith-Miles. 2017. Performance analysis of continuous black-box optimization algorithms via footprints in instance space. *Evolutionary computation* 25, 4 (2017), 529–554.

[22] Ana Nikolikj. 2023. *Algorithm Footprints*. https://github.com/anikolik/algorithm-footprints

[23] Ana Nikolikj, Ryan Lang, Peter Korošec, and Tome Eftimov. 2022. Explaining Differential Evolution Performance Through Problem Landscape Characteristics. In *Bioinspired Optimization Methods and Their Applications: 10th International Conference, BIOMA 2022, Maribor, Slovenia, November 17–18, 2022, Proceedings*. Springer, Cham, 99–113.

[24] William S Noble. 2006. What is a support vector machine? *Nature biotechnology* 24, 12 (2006), 1565–1567.

[25] Leif E Peterson. 2009. K-nearest neighbor. *Scholarpedia* 4, 2 (2009), 1883.

[26] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. 2022. The shapley value in machine learning. *arXiv preprint arXiv:2202.05594* (2022).

[27] Kate Smith-Miles and Mario Andrés Muñoz. 2022. Instance Space Analysis for Algorithm Testing: Methodology and Software Tools. *ACM Comput. Surv.* (nov 2022). https://doi.org/10.1145/3572895 Just Accepted.

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

[29] Risto Trajanov, Stefan Dimeski, Martin Popovski, Peter Korošec, and Tome Eftimov. 2021. Explainable landscape-aware optimization performance prediction. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 01–08.

[30] Risto Trajanov, Stefan Dimeski, Martin Popovski, Peter Korošec, and Tome Eftimov. 2022. Explainable Landscape Analysis in Automated Algorithm Performance Prediction. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, Cham, 207–222.

[31] Joaquin Vanschoren. 2019. Meta-learning. *Automated machine learning: methods, systems, challenges* (2019), 35–61.