The Pitfalls of Benchmarking in Algorithm Selection: What We Are Getting Wrong

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

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

ABSTRACT

Algorithm selection, aiming to identify the best algorithm for a given problem, plays a pivotal role in continuous black-box optimization. A common approach involves representing optimization functions using a set of features, which are then used to train a machine learning meta-model for selecting suitable algorithms. Various approaches have demonstrated the effectiveness of these algorithm selection meta-models. However, not all evaluation approaches are equally valid for assessing the performance of meta-models. We highlight methodological issues that frequently occur in the community and should be addressed when evaluating algorithm selection approaches. First, we identify flaws with the "leave-instance-out" evaluation technique. We show that non-informative features and meta-models can achieve high accuracy, which should not be the case with a well-designed evaluation framework. Second, we demonstrate that measuring the performance of optimization algorithms with metrics sensitive to the scale of the objective function requires careful consideration of how this impacts the construction of the meta-model, its predictions, and the model's error. Such metrics can falsely present overly optimistic performance assessments of the meta-models. This paper emphasizes the importance of careful evaluation, as loosely defined methodologies can mislead researchers, divert efforts, and introduce noise into the field.

CCS CONCEPTS

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

1 INTRODUCTION

Benchmarking plays a crucial role in the development of algorithms and techniques across various fields, providing a standardized framework for performance evaluation. However, poorly designed evaluation practices can lead to misleading results [3, 9, 10, 19, 33]. In this paper, we focus on algorithm selection (AS), a central topic in continuous black-box optimization [14]. The goal of AS is to determine the most suitable optimization algorithm for a given problem. The most common approach to addressing this challenge begins by representing optimization problems from a benchmark suite using features, such as Exploratory Landscape Analysis (ELA) [17] or one of the other existing feature sets [5]. Next, optimization algorithms from the algorithm portfolio are evaluated on each problem individually to obtain their performance. Subsequently, a machine learning meta-model is trained to predict the most suitable optimization algorithm for a new problem based on its feature representation.

At the time of writing, the practices for performing and evaluating AS are not entirely standardized within the community. As a result, researchers employ various strategies, as there is no single, definitive way to evaluate AS methodologies [31]. Practitioners are free to choose from benchmark suites such as COmparing Continuous Optimizers (COCO) [7] or IEEE Congress on Evolutionary Computation (CEC) Special Sessions and Competitions on Real-Parameter Single-Objective Optimization [32], and often select algorithm portfolios based on the availability of code for each specific optimization algorithm. For the meta-model, standard tabular data models, such as random forests or neural networks, are commonly used. Additionally, practitioners can reformulate the problem as a classification task, aiming to predict the single best algorithm [31], or as a multi-class classification task [29], where the goal is to recommend a set of algorithms. One can even treat AS as a regression task, where the objective is to predict the ranking of algorithms [21], its performance relative to the other algorithms [5] or by how much the optimization algorithms will fall short of optimal values commonly known as target precision [11].

With such a flexible methodology, there is potential for misuse, whether intentional or accidental. Past approaches have reported varying levels of success [1, 5, 6, 11, 14, 15, 18, 21]. It is possible to combine the aforementioned components and develop evaluation methodologies that lack rigor, allowing nonsensical meta-models to significantly outperform baselines and existing models, purely due to flaws in the evaluation approach. In this paper, we highlight two such issues with the methodology commonly used in the community, where incorrect evaluation can lead to wrong or conflicting conclusions.

The first issue we investigate is the use of the COCO benchmark in AS. Originally designed for benchmarking optimization algorithms, COCO consists of 24 problem classes, each with multiple problem instances. These instances are often shifted or translated versions of the original problem to prevent optimization algorithms from overfitting to a specific objective function or search region. Consequently, problem instances within the same problem class most often have properties that are more similar to each other than to instances from different classes [16, 26, 28], resulting in a somewhat comparable performance of optimization algorithms within instances of the same class [16]. Later adopted by the AS community, COCO has become a common benchmark for evaluating the capabilities of AS meta-models. In the context of AS on the COCO benchmark, two main evaluation techniques are used: the "leave-instance-out" (LIO) method and the "leave-problem-out"

(LPO) method. With the LIO method, the train and test sets contain all 24 problem classes, with instances of each problem class being split between the train and test sets. Due to instances of the same class appearing in both the train and test sets, this approach is relatively forgiving. A more challenging evaluation is the LPO evaluation method, where the train set typically contains instances from 23 problem classes, and the test set contains instances from the remaining problem class. In our paper, we demonstrate that the LIO methodology, commonly used with COCO, is flawed when used to benchmark AS meta-models, as it can lead to misleading results. Meta-models achieve high performance due to spurious correlations between features and the target, rather than genuine predictive capability.

The second issue, which we believe is common, is the use of scale-sensitive metrics to measure the performance of optimization algorithms and the subsequent use of these metrics as targets to be predicted by the meta-models. This approach can lead to several issues. For instance, when problem instances are not on the same scale, as is the case with the COCO benchmark, comparing and aggregating these metrics without accounting for scale becomes problematic. Furthermore, scale-sensitive metrics can mislead meta-model construction, as the models may incorrectly attribute importance to features influenced by scale. Finally, developing baselines for meta-models trained on such metrics is challenging and can result in misleading outcomes, where meta-models appear to outperform baselines significantly due to the properties of the metrics rather than genuine model improvements.

Our contribution: In this paper, we identify two significant shortcomings commonly observed in the evaluation of landscape features and meta-models. Specifically, we highlight issues related to the evaluation of LIO and the use of scale-sensitive metrics for meta-model construction. For both methodological shortcomings, we provide comprehensive explanations outlining the severe weaknesses of the current methodologies and propose potential strategies for their improvement. We argue that the existing benchmarking practices for landscape features and meta-models are flawed, leading to potentially erroneous conclusions. Due to these shortcomings, we believe that some evaluations of landscape features and algorithm selection meta-models are likely unreliable due to flawed evaluation techniques.

Outline: The structure of the following sections is organized as follows: Section 2 examines the application of the LIO methodology within the context of AS meta-model evaluation. This is followed by Section 3, which addresses challenges associated with using scale-sensitive metrics in model evaluation. Finally, Section 4 presents concluding observations and remarks.

Reproducibility: The experiments conducted can be replicated using the code available in the Github repository, accessible at [22].

2 LEAVE-INSTANCE-OUT EVALUATION

In this section, we make the following claim: **The "leave-instance-out" evaluation is flawed, as - due to spurious correlations - it can result in non-informative features and meta-models still achieving high performance, producing misleading and over-optimistic results.** To demonstrate this, we adopt an approach where we deliberately construct non-informative features that would be deemed useless by most practitioners for the AS task. Despite this, these features are used to train and evaluate meta-models within the LIO methodology, which achieves relatively good results and significantly outperforms the single best solver. This outcome suggests either that the proposed features are, in fact, useful for AS, or that the LIO methodology itself is flawed and unsuitable for evaluating meta-models.

Before presenting the full methodology, we will illustrate the problem of spurious correlations and correlated subgroups using an example from a different domain. This example will demonstrate how these issues can significantly impact a classifier's perceived accuracy. A well-known example from the computer vision domain highlights the challenges of spurious correlations when building machine learning models. In the "Husky vs. Wolf" dataset, the goal is to classify whether an image contains a wolf or a husky. Initial attempts achieved high accuracy, often surpassing human experts. However, these models struggled to generalize to new images and performed poorly in real-world scenarios, failing to differentiate between wolves and huskies effectively. The reason became evident with the application of explainability techniques: most images of wolves were taken in snowy settings, while husky images rarely featured snow [27]. Consequently, the models learned to identify the presence of snow rather than distinguishing the animals themselves. This example illustrates how spurious correlations, such as background features, can undermine model construction and render any accuracy improvements meaningless. It is important to note that this issue is not unique to convolutional neural networks that were used in the study. Even manually crafted features, such as a ratio of white to non-white pixels, would boost accuracy based on this irrelevant correlation, rather than capturing a meaningful distinction between the breeds.

To demonstrate the issue with the LIO methodology, consider a scenario where a new landscape feature is introduced, and we want to assess whether it is useful for AS, without being interested in other tasks such as classification. Additionally, let us make a mild assumption that the performance of the optimization algorithm differs between problem classes but is to some degree similar for the instances of the same problem class [22]. With LIO, there are four possible cases: (a) the feature is not beneficial for either AS or problem classification, (b) the feature performs well for AS but not for classification, (c) the feature performs well only for classification but not for AS, and (d) the feature performs well for both tasks. The problematic cases in LIO are (c) and (d), where the feature performs well for classification. In such cases, if problem instances are highly similar, the feature may provide a good estimate of an algorithm's performance on a specific instance, simply through memorization (i.e. based on samples identifying to what problem class instance belongs). Additionally, features that encode the problem class of a particular instance are likely to be valuable, as they enable accurate performance estimation by indicating the problem class to which the instance belongs.

2.1 Methodology

In this section, we outline the methodology used to evaluate the claim that spurious correlations, together with LIO, may cause over-optimistic results.

Our methodology begins by splitting problems into training and testing sets based on LIO or LPO approaches. Samples are then obtained from each problem instance to compute features, using three different feature sets: one based on ELA and two additional sets introduced below. Optimization algorithms are evaluated on the training set to assess their performance, which is captured by ranking the algorithms in terms of the objective function value of their best-found solution. Given a set of features and algorithm ranks, meta-models are subsequently trained to map the features to their respective ranks. These meta-models are then evaluated on the test set based on their ability to rank algorithms, as measured by Pairwise Ranking Error, detailed below. This methodology aligns with approaches commonly used in other studies and is similar to standard evaluations in AS. All the details about the exact methodology used are described below.

Benchmark suite used in this study is the COCO benchmark, as introduced earlier. We employ the COCO benchmark suite to highlight potential pitfalls in its application for evaluating AS techniques. Specifically, we use all 24 problem classes, selecting the first 15 problem instances from each class. In this study, we only focus on 5D problems. Note also that other benchmarks do not use the concepts of problem classes and instances, so they may not be applicable.

Optimization algorithm portfolio used in this paper is defined as $\mathcal{A} = \{a_1, \ldots, a_k\}$ and includes the following algorithms from the *pymoo* [4] framework: Genetic Algorithm (GA) [12], Differential Evolution (DE) [25], Particle Swarm Optimization (PSO) [13], Evolutionary Strategy (ES) [2], and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [8]. All the algorithm hyperparameters were configured using the default values defined in the *pymoo* [4] framework version *0.5.0*.

Optimization algorithm performance is measured by running each optimization algorithm 30 times on each problem instance to obtain its mean rank, which is subsequently used as the target of a meta-model. All algorithms are evaluated using 1000 function evaluations per dimension (i.e., fixed-budget). For our 5-dimensional problems, this results in a total of 5000 function evaluations.

Feature sets utilized in this paper are derived using three distinct approaches. Candidate solutions are generated through LHS with a sample size of 250D, where fitness values are scaled between 0 and 1 prior to feature extraction. The three techniques employed for feature extraction are:

i) The first is the well-known ELA feature set [24], with a total of 85 features used to construct a meta-model.

ii) The second uses the so-called non-informative features. These features are designed to identify the class of a problem but are not useful for performing AS. We design non-informative features in the following way. First, n candidate solutions $\mathbb{X} = \{\vec{x}_1, \ldots, \vec{x}_n\}$ are obtained and evaluated $\mathbb{Y} = \{f(\vec{x}_1), \ldots, f(\vec{x}_n)\}$, same as with ELA features. In the next step, the vector of fitness values Y is transformed into a feature f_i using the following feature construction template $f_i = agg(tr(sc\mathbb{Y}))$. In this case, sc is a randomly selected scalar, tr is a randomly selected transformation, and agg is an aggregate function. This is repeated to obtain multiple features by repeatedly creating functions by randomly selecting aggregate function $agg \in \{mean(x), median(x), std(x), quantile_5(x), quantile_{25}(x), quantile_{75}(x), quantile_{95}(x)\}$, transformation function $tr \in \{sin(x), cos(x), x^{\frac{1}{6}}, x^{\frac{1}{3}}, x^{\frac{1}{2}}, x^2, log(x+1)\}$, and scalar $sc \in \{0.2, 0.3, 0.5, 0.7, 1, 2, 3, 5, 7, 9\}$. These features do not use any information about where candidate solutions were sampled $\mathbb{X}$ but only the fitness values of candidate solutions $\mathbb{Y}$. Additionally, the first feature uses the same randomly selected scalar, aggregate, and transformation functions across all objective functions, and the same applies to the second feature, etc. Although this methodology allows us to construct as many features as we like, we created the same number of features as were used with ELA.

iii) Lastly, to demonstrate the worst-case scenario, we add an additional feature set. This feature set contains only one feature: the problem class to which each individual instance belongs. This serves mainly as a demonstration of what happens if features are strongly (or perfectly) correlated with the problem class and how a non-informative feature, in the context of AS, can achieve extremely good results with LIO evaluation.

Meta-models in our study minimize the MSE between the ground truth and the predicted ranks. They utilize the random forest regressor from the *scikit-learn* library [20] with default hyperparameters. Meta-models used in this section are the following:

i) The `random` meta-model, which randomly orders optimization algorithms;

ii) The `mean` meta-model, which predicts the mean rank of each optimization algorithm, functioning similarly to a single best solver in the context of ranking;

iii) The `ela` meta-model, where ELA features are used to predict ranks;

iv) The `non-inf` meta-model, where non-informative features capture basic and relatively non-informative statistics about objective function values and use them to predict ranks; and

v) The `class` meta-model, which uses the class of a problem instance as a feature in rank prediction.

Meta-models' error metrics used to evaluate the accuracy of a meta-model is the Pairwise Ranking Error (PRE) defined as:

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

$$r(a_i, a_j) = \begin{cases} 0 & \text{if } r_p(a_i, a_j) = r_g(a_i, a_j) \\ 1 & \text{if } r_p(a_i, a_j) \neq r_g(a_i, a_j) \end{cases} \tag{2}$$

In this context, let a_i and a_j represent algorithms within the portfolio, with $r_p(a_i, a_j)$ and $r_g(a_i, a_j)$ serving as functions that denote the rank differences between these optimization algorithms. Specifically, $r_p(a_i, a_j)$ outputs values of -1, 0, or 1, indicating whether the predicted rank of algorithm a_i is lower than, equal to, or higher than the rank of algorithm a_j, respectively. Similarly, $r_g(a_i, a_j)$ follows the same rule, but instead relies on rankings based on the ground truth. This approach ensures that $r_g(a_i, a_j)$ yields values of -1, 0, or 1 by referencing the actual rankings established through 100 runs of each optimization algorithm. The PRE metric evaluates the effectiveness of ranking algorithms by quantifying the percentage of pairs of algorithms where the order of ranks between them was guessed correctly. A PRE value of 0.0 signifies a perfect match between the predicted and true order. When ranks are assigned randomly, the error value is 0.5, indicating that half of the item

pairs are correctly ordered. A PRE of 1.0 represents the extreme case where all pairs are misordered, reflecting a scenario in which the predicted ranking is entirely reversed from the true order.

2.2 Results

This subsection presents the results of evaluating various meta-models and feature sets for both the LIO and LPO methodologies. Figure 1 show the PRE of different meta-models across two evaluation methodologies. We begin with the LIO evaluation. As expected, the `random` meta-model performs the worst, with a PRE of approximately 0.5, followed by the `mean` meta-model, which outputs the mean ranks of optimization algorithms calculated on the training set, achieving a PRE of around 0.3. The remaining meta-models—`ela`, `non-inf`, and `class`—substantially outperform the baselines, each achieving a PRE below 0.15. Among these three meta-models, the `non-inf` meta-model has the highest PRE at approximately 0.13, followed by the ELA-based meta-model at 0.1. However, the best-performing meta-model is `class`, which uses a single feature—the problem instance class—and achieves a PRE of 0.05. This suggests that even trivial summary statistics in the `non-inf` features can yield performance close to that of meta-models using ELA features. Furthermore, theoretically, if a feature could perfectly identify the problem class of an instance, such a meta-model would outperform the ELA meta-model. This strongly indicates that within the LIO framework, features effective in classification can significantly reduce meta-model error, even if the meta-model has no practical application for AS, as is the case with the problem class feature. For completeness, LPO validation is also presented. With LPO, all meta-models, except the `random`, achieve comparable performance, revealing that while class and non-informative features may show low PRE or even outperform ELA under LIO, this is due to LIO's tendency to reward irrelevant features that correlate with the target—a phenomenon not observed in the LPO evaluation.

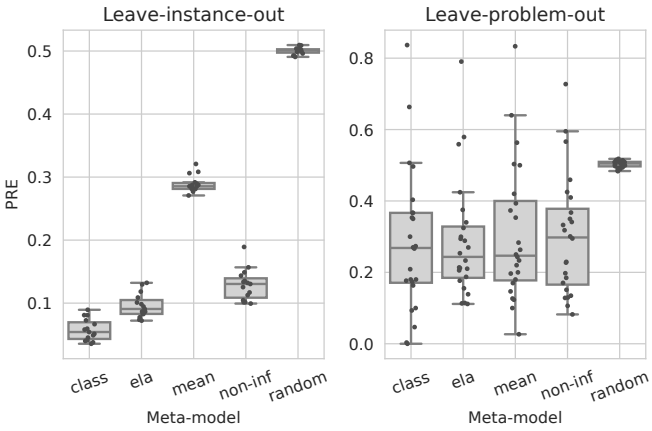

Figure 1: PRE of different meta-models with the LIO evaluation strategy (top) and LPO evaluation strategy (bottom). Each dot represents one train/test split of the data.

2.3 Discussion

Here, we discuss the results from the previous subsection. As expected, we observe entirely contradictory outcomes between LIO and LPO, as one evaluation technique is much more challenging than the other [31]. However, what is unexpected is how well the meta-model using non-informative and class features performs with LIO. We believe the main reason for this is the presence of spurious features—features that reflect some property of a problem which, while not directly helpful for AS, may correlate with performance due to similarities between the problems in the training and test sets. Instances belonging to the same problem class can have extremely similar features, and in most cases, optimization algorithms perform similarly on instances of the same class [22]. With that in mind, there is a high likelihood that spurious correlations substantially improve the performance of meta-models without any actual progress. Any meta-model that achieves accuracy between that of `mean` and `class` cannot be claimed to perform AS due to identifying important problem properties, but rather due to spurious correlations. Although we do not present the results here due to space constraints, training meta-models using a single feature for both ELA and the proposed features, and then evaluating them on LIO, reveals a strong correlation between classification accuracy and the PRE of the meta-model. This suggests a potential similarity between classification and AS in the context of LIO tasks. In other words, it indicates that features deemed important for classification might erroneously be considered important for AS when LIO is used as the evaluation metric.

To draw a parallel with the computer vision example from the beginning of the section: features that are able to perform AS simply because they correlate with the class would be conceptually similar to the ratio of white pixels in the "Husky vs. Wolf" example. With such a feature, differentiating between wolves and huskies is not possible. However, given this feature alongside a flawed evaluation methodology, achieving high performance becomes much easier. In essence, non-informative features can significantly boost performance when faulty evaluation techniques are used.

We would also like to emphasize that the methodology presented here cannot, in general, be used to test whether feature sets are informative or not. If a feature set has similar performance characteristics to our non-informative features, that does not necessarily mean those features are also non-informative. We only show that non-informative features can perform well with the LIO methodology.

We aim to caution researchers against using LIO evaluation and accepting it as a standard, as it will likely:

- Produce overly optimistic results due to the meta-model capturing spurious correlations that may artificially improve performance.
- Allow for the publication of new, non-informative features that appear effective solely due to faulty LIO evaluation techniques, but will never generalize to objective functions that differ from those in the test set.
- Yield misleading conclusions on feature importance that rely solely on LIO evaluation, as these will not accurately indicate which features are crucial for generalization but will reward features that are good at identifying the class of a problem instance. Using explainability techniques and claiming that features identified as

important are actually relevant for algorithm selection in general is misleading. For example, blindly applying feature explainability techniques on a dataset where the goal is to differentiate between wolves and huskies and overgeneralizing could lead to the following conclusion: what differentiates huskies and wolves is not domestication, appearance, or behavior, but the snowy background.

The argument that LIO serves as a simpler stepping stone towards LPO is misguided. While achieving strong performance on LIO may seem like progress, it does not inherently translate to improved LPO outcomes. COCO, with its diverse yet limited set of problems, is not ideally suited as a benchmark for AS in the context of LPO due to insufficient problem quantity and dissimilarity among them. Consequently, treating LIO as an easier benchmark alternative to LPO can yield misleading results, as demonstrated in this section. If LIO is used, the authors should clearly demonstrate why their problem set is unique and justifies the use of LIO methodology, especially in cases where the training and test set instances represent the same objective function but may be slightly shifted, scaled, or rotated. While we do not address other tasks, we believe the same is true for high-level property prediction where exactly the same problems with spurious correlations can arise.

3 SCALE-SENSITIVE METRICS

This section focuses on the metrics used to measure the performance of optimization algorithms and how they affect meta-models. We make the following claim: **Scale-sensitive metrics used to evaluate the performance of optimization algorithms, in combination with certain meta-models, can produce highly misleading results. Using objective functions with different scales may show a substantial improvement in the meta-model's performance, even when this is not the case.**

We will first define a more concrete definition of what we mean by scale-sensitive and scale-independent metrics, as to the best of our knowledge, this has not been defined in the context of AS. We define a scale-sensitive performance metric to be one that changes when the objective function is rescaled. In contrast, a scale-independent metric for comparing algorithm performance remains unchanged even when the scale of the problem changes. For example, target precision [11], computed as the difference between best-found values and the optimal value of the objective function (defined below) is strongly correlated with the scale of the problem, as rescaling the problem will almost certainly alter the difference between the optimum value and the best-found solution. On the other hand, assuming the trajectory of optimization algorithms is not affected by the rescaling of the objective function, a metric such as an algorithm's rank would be much more stable if the problem is rescaled. However, be aware that precise definitions can be tricky. While most optimization problems today, including those discussed in the paper, are relatively invariant to the problem's scale (i.e., differently scaled objective functions do not significantly alter the algorithm's search trajectory), this is not always the case [16]. A metric might be sensitive to scale with one portfolio of algorithms but not with another.

A similar concept of metric sensitivity to scaling has been thoroughly explored in other domains such as time series forecasting [30]. For example, let us consider the case of a bank with two individuals forecasting the price of oil. One is based in Europe and makes forecasts in euros, while the other is based in Japan and makes forecasts in yen. Both individuals use an identical forecasting model to predict the future price of oil. Their future predictions are evaluated based on the mean absolute error (MAE). Since 100 yen is approximately 0.6€ (assuming this exchange rate remains fixed), the same forecasting model will yield different MAE values depending on the currency used for evaluation. A 1% forecasting error might translate to a difference of a few hundred yen, but when measured in euros, it would correspond to only a few euros. Such an evaluation technique presents several challenges. Since both time series operate on completely different scales, their errors will likely reflect this disparity. For instance, an MAE of 20 might be negligible for forecasts in yen but substantial for forecasts in euros. Additionally, simply aggregating the errors without accounting for the scale of the time series would produce meaningless results. Using scale-sensitive metrics like MAE makes it impractical to compare errors across time series with different scales. How should a bank aim to maximize profit, allocate further funds, and measure the success of multiple individuals forecasting different securities? If MAE is used without accounting for differences in exchange rates, all funds would likely be allocated to the European individual simply because their MAE appears lower. Such a performance metric is clearly flawed in this context. When hundreds of securities and currencies are involved, the bank must develop a strategy to incorporate this knowledge into its decision-making process effectively. The same principle applies to AS meta-models combined with scale-sensitive metrics, where similar issues could arise due to the inappropriate handling of scale differences.

When a meta-model is trained to predict a scale-sensitive performance metric, such as the MAE, using a time series represented by a set of features, its task becomes quite tricky. A high predicted MAE could either indicate that the time series is hard to forecast or simply that the series has a larger scale. In this case, the scale-sensitive nature of the performance metric means its value can vary significantly, not only based on forecasting difficulty but also on the scale of the series itself. Thus, a time series that is harder to forecast might still have a smaller MAE than a series that is trivially forecastable but has a larger scale. This makes the meta-model's job more complex, as it must learn to separate the effects of scale from the inherent forecasting difficulty.

Additionally, a consideration with scale-sensitive metrics and time series of different scales is the need to carefully construct the baseline for a meta-model. But what would be a reasonable baseline meta-model in our case of predicting the performance of forecasting algorithms? The simplest baseline we could construct is a meta-model that predicts the mean MAE observed during the training phase. But is such a baseline really reasonable? If MAE accounts for both the scale of the time series and its difficulty in being forecasted, then such a baseline is rather meaningless. Any meta-model that can learn how a time series is scaled will outperform the scale-agnostic baseline that only predicts the mean MAE independent of the problem scale. For that reason, scale-sensitive metrics for measuring the performance of forecasting algorithms are almost never used for meta-model construction [23].

3.1 Methodology

To demonstrate how the use of scale-sensitive metrics for evaluating optimization algorithm performance can influence the construction of meta-models and highlight how simply interpreting meta-model errors can be misleading, the following methodology is employed: a meta-model is trained to predict an optimization algorithm's target precision using a single feature that is correlated with the scale of the objective function. For the LPO evaluation, this meta-model is compared against a baseline model that predicts the mean target precision to determine whether it outperforms the baseline. Additionally, alongside meta-models predicting target precision, another set of meta-models is trained to predict the ranks of optimization algorithms. All the meta-models are then compared to evaluate if there are any discrepancies between the use of scale-sensitive and scale-free metrics. Any substantial differences between the metrics would indicate that, in one case, meta-models might achieve smaller errors simply due to learning the strong correlation between scale and target precision through the use of scale-sensitive features.

The methodology is largely the same as in the previous section, with the **Benchmark suite** and **Optimization algorithm portfolio** remaining the same. Note that at no stage of our methodology did we intentionally rescale the COCO benchmark problems. We keep them as they are in the COCO framework with large differences in scale between problem classes. The major difference with this approach lies in the way we measure performance and the features used to train the meta-models.

Optimization algorithm performance is measured using a metric called target precision, which evaluates how close the optimization algorithm gets to the optimal value. In other words, optimization algorithms are assessed in terms of target precision, defined as $f(\vec{x}_{best\_found}) - f(\vec{x}_{opt})$, where $f(\vec{x}_{best\_found})$ represents the best value found by a specific algorithm, and $f(\vec{x}_{opt})$ denotes the optimum value for the given problem. This value is then used as a target for meta-model training. All the other metrics are the same as in the previous subsection.

Feature set in this section consists of only a single feature named f_{scale}, constructed in the following way. Given candidate solutions that are evaluated to obtain $Y = \{f(\vec{x}_1), \ldots, f(\vec{x}_n)\}$, the feature f_{scale} is defined as $f_{scale} = max(\mathbb{Y}) - min(\mathbb{Y})$. This feature essentially captures how the problem is scaled based on the initial set of samples.

Meta-models used in this section are as follows: `mean-precision`, which predicts the mean precision obtained on the training data without using any features; `mean-rank`, which is identical to the previous model but predicts ranks instead of precision; and `rf-precision` and `rf-rank`, which predict precision and rank, respectively, based on a single feature defined above. All meta-models are set up to minimize MSE between predicted target precision and ground truth target precision, or, in the case of ranks, between ground truth ranks and predicted ranks.

Meta-models' error metric is defined as follows. Meta-models trained to predict target precision (`rf-precision` and `mean-precision`) are evaluated using MSE on raw predictions and PRE when target precision values are transformed into ranks. On the other hand, rank prediction meta-models (`rf-rank` and `mean-rank`) are evaluated solely using PRE.

3.2 Results

The first part of the results section addresses the sensitivity of metrics to scaling and how this affects scale-sensitive and scale-free metrics. Figure 2 illustrates this characteristic of the metrics. We plot target precision and rank relative to scale for two of the five optimization algorithms on the first instances of problems classes 4, 13, and 24. Clear patterns emerge, showing that, in this scenario, the precision metric is highly correlated with the scale of the instance, while rank remains relatively consistent, regardless of scale. This points to the fact that one can arbitrarily increase/decrease the target precision singly by rescaling the objective function, while the same is not true for ranks. Additionally, using a target precision can be a poor indicator of problem difficulty as comparisons in terms of target precision between problems on different scales (as is the case with COCO problems) are meaningless.

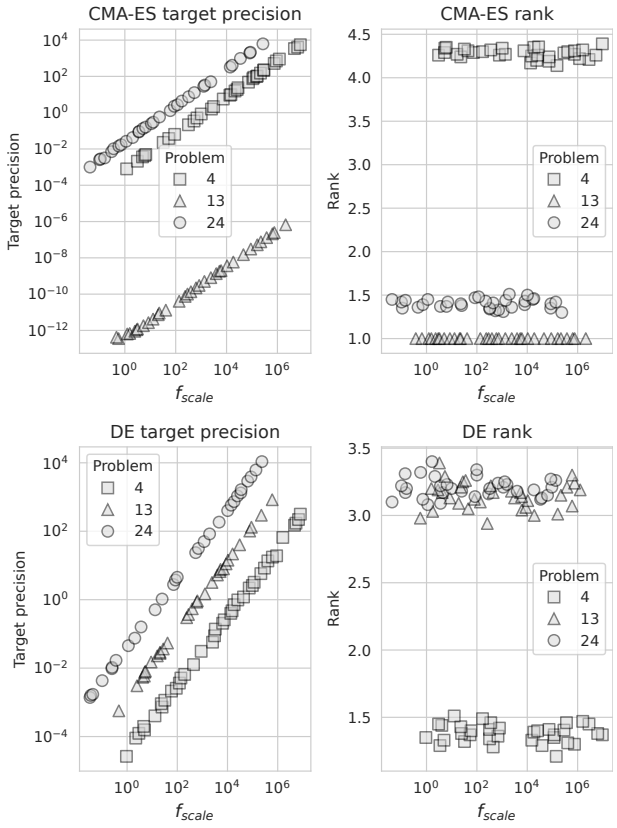

Figure 2: Comparison of target precision and rank metrics (y-axis) for quantifying the performance of algorithms for the first instance of COCO problems 4, 13, and 24. All instances are artificially rescaled and their scale is measured with f_{scale} feature (x-axis). Results are reported for only two out of five algorithms in the portfolio.

In the second part of this section, we focus on the discrepancy between scale-sensitive and scale-free metrics and why setting up a baseline is so difficult with scale-sensitive metrics. First, we plot the MSE of `mean-precision` and `rf-precision` meta-models

in Figure 3 using the LPO methodology. The MSE is aggregated over all five optimization algorithms, and each individual point shows the error of a meta-model on one of the 24 splits. From this, we can conclude that the `rf-precision` meta-model, using only a single feature f_{scale} defined above, gives extremely good results. Its accuracy is an order of magnitude better than the baseline `mean-precision`. This is confirmed with the Wilcoxon signed-rank test ($p < 0.05$), where we observe a statistically significant difference in the performance of the two meta-models. But how is it possible to achieve such impressive results using only a single feature, when even more advanced meta-models [5, 6, 31] struggle to achieve this? And are these results really as impressive as they appear? Surely, such a good target prediction meta-model should be capable of correctly ranking optimization algorithms from best to worst, even for the LPO methodology.

The next step is to use the `rf-precision` and `mean-precision` meta-models to rank optimization algorithms. This is performed as follows: among the five algorithms, the one with the lowest predicted target precision is ranked as the best, the second lowest target precision as the second best, and so on for all five. The obtained ranks are then compared with the PRE metric to measure the error between the predicted and ground truth ranks of the optimization algorithms. Figure 3 shows the PRE of all four meta-models. To our surprise, the `rf-precision` meta-model is no longer effective when evaluated with a scale-free metric. Even more surprisingly, there is no longer any difference in performance between the four meta-models. Using the Friedman test for statistical comparison at a significance level of 0.05, the performances do not differ in any significant way. So how could `rf-precision` meta-model go from significantly outperforming the baseline when evaluated with the MSE when predicting a scale-sensitive target precision metric to showing no difference between meta-models on a scale-free metric? We discuss this in the next section.

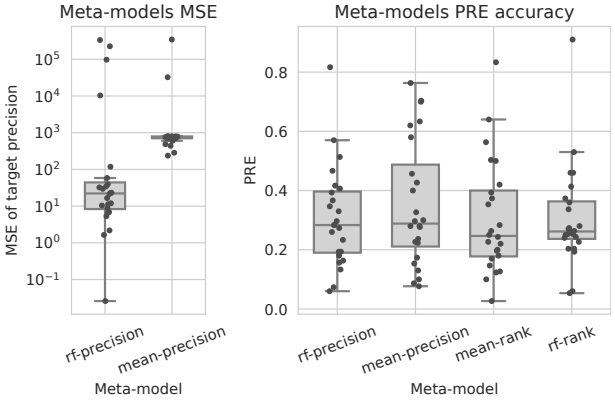

Figure 3: Left: MSE calculated between true and predicted target precision for two meta-models aggregated across all five optimization algorithms. With only a single feature, the `rf-precision` meta-model outperforms the `mean-precision` by orders of magnitude. Right: Evaluating four different meta-models to determine their success in ranking five optimization algorithms in the LPO setting. Two of the meta-models directly predict the ranks, while the remaining two meta-models predict target precision, which is then transformed into ranks. The large differences between the meta-models are now nonexistent.

3.3 Discussion

Achieving a target precision of 0.03 in an optimization problem is hard to evaluate without knowing the objective function's scale. Precision depends on scaling; 0.03 might be negligible for a function ranging from 0 to 10,000 but unacceptably high for one ranging from 0 to 1. Thus, target precisions from differently scaled functions cannot be directly compared without normalization, as they represent varying relative accuracies and risk misleading conclusions.

A similar issue arises when predicting target precision errors using a meta-model. For objective functions of different scales, one can expect the errors to follow a similar pattern, meaning the meta-model will have larger errors for objective functions with larger scales. This implies that reporting metrics like MSE or MAE for a meta-model's predictions of target precision compared to the true target precision would also be meaningless without considering scale. When performing k-fold cross-validation, different train/test splits could result in errors that differ by order of magnitude, making the evaluation inconsistent and unreliable without proper scaling.

The second issue we discuss is the contradictory results between using scale-sensitive and scale-free metrics to measure the performance of AS. Both use similar baselines (mean precision and mean rank), the same features, and a similar method of constructing the meta-model (random forest), yet the results are completely different. Why is this the case? We believe the main reason is the following: the target precision metric concurrently measures two aspects. When a particular problem is harder to solve (for example, in terms of multimodality), the optimization algorithm will, in most cases, find inferior solutions compared to an unimodal problem, thereby resulting in worse performance in terms of target precision. However, this is not the sole factor contributing to an increase in error for scale-sensitive metrics. Even if the problem's difficulty remains constant, scaling the problem can also increase or decrease the precision error. Consequently, such errors have two components that are often challenging to untangle. If a meta-model is evaluated on its ability to predict target precision, there are two ways to reduce the error. The first is to learn what makes a problem more or less difficult, and adjust the expected error that the optimization algorithm will make based on that. The other way to improve the performance of such a model is to include features that capture the scale of the problem and adjust the prediction accordingly.

Suppose we have two objective functions for which we need to predict performance, measured by target precision, without any additional information. The best we could do in this case would be to predict the mean precision calculated from the training set. However, if information on the problem's scale is provided, we can significantly improve precision simply by predicting the mean error conditioned on the objective function's scale. This approach can substantially reduce the error without actually improving the ability to select the best algorithm. The same is not true for scale-independent metrics like the PRE metric, since the rank of the optimization algorithm does not change with scale. This points

to a rather alarming fact that even if the meta-model is good at predicting target precision, this will not necessarily translate into a good AS model.

One might argue that the scale-sensitive feature used here is not relevant in practice and that no one would actually use it. While this is true, many ELA features, for instance, are also sensitive to the problem's scale [34], which would similarly introduce scale-dependent information into the meta-model. To create a proper baseline for evaluating precision, the baseline must account for the problem's scale; otherwise, it is too weak to be useful.

We believe the following recommendations should be applied when evaluating AS meta-models:

• Using scale-sensitive metrics to measure algorithm performance, such as target precision, can be interpreted in multiple ways. A large error may indicate that a particular objective function is either more difficult to solve or is differently scaled. Comparing algorithms based on scale-sensitive metrics is tricky for differently scaled objective functions. Aggregating such metrics should be avoided as it can lead to misleading results.

• A meta-model trained to predict a scale-sensitive metric of optimization algorithm performance might struggle to learn how the algorithm's performance relates to the difficulty and scale of the problem. As a result, it may only capture how the features are related to the scale of the objective function.

• Trained meta-models should be evaluated using appropriate metrics. Metrics like classification accuracy, PRE, or similar alternatives are more suitable in such cases, as the meta-model only needs to learn which algorithm to select, independent of scale. Reporting the success of meta-models in predicting scale-sensitive metrics (in terms of MSE, MAE, or similar metrics) is largely meaningless, as such meta-models may not necessarily perform well for the task of algorithm selection.

• Using any sort of explainability techniques together with meta-models trained on scale-sensitive metrics will produce misleading feature importance values, as scale-sensitive features will have their importance artificially inflated. This indicates that the larger the differences in scale, the more important the scale-sensitive features will appear.

4 CONCLUSION

This study highlights some common but often overlooked pitfalls in algorithm selection evaluation methodologies that can lead to overly optimistic and misleading results. We presented two examples where improper evaluation practices can result in incorrect conclusions. First, we demonstrated how the use of the LIO methodology applicable when using COCO benchmark can produce meaningless meta-models with seemingly good performance. This occurs due to strong correlations between features and algorithm performances when similar problem instances appear in both the training and test sets. Second, we addressed the issue of using scale-sensitive metrics, where improvements over baseline models may seem significant but are artifacts of flaws in the metrics themselves. For both issues, we provided a detailed analysis explaining their causes and outlined strategies to mitigate them. We acknowledge that no evaluation methodology is perfect. However, we believe that current approaches to algorithm selection are inadequate, with significant effort being invested in methodologies that are ineffective.

In this work, we have highlighted key methodological challenges and addressed specific gaps in the current landscape of AS. We believe that the presented pitfalls are not simply "small" errors that are to some degree expected when writing a paper. Examples of such minor issues include incorrectly reporting some of the default hyperparameter values of an algorithm or applying inappropriate statistical tests. The pitfalls addressed in this paper, if not properly handled, can invalidate many of the core claims that authors in the community often make. This discussion is not intended as a critique of any specific papers, and especially not to question the value of widely recognized benchmarks like COCO or techniques such as ELA features, both of which remain perfectly valid and have significantly advanced the field, even though they are sometimes used within the faulty methodologies we discuss. We deliberately chose not to list specific papers where the approaches we described as problematic have been employed, focusing instead on fostering constructive discussions about methodological improvements. We acknowledge that we, the authors, have also previously employed some of the approaches we now advocate against using.

Future research should focus on refining benchmarking practices through more rigorous and accurate evaluation methodologies. While this paper primarily points out the weaknesses of current approaches used in the community, further work is needed to establish best practices for reliable assessments of AS meta-models and landscape features.

ACKNOWLEDGMENTS

This work received financial support from the following sources: the Slovenian Research Agency through research core funding (No. P2-0098), a young researcher grant (No. PR-11263) awarded to GP, and a young researcher grant (No. PR-12393) awarded to GC, as well as the European Union's Horizon Europe program under grant agreement No. 101077049 (CONDUCTOR). We would also like to thank the reviewers for their valuable comments.

REFERENCES

[1] Andrejaana Andova, Jordan N Cork, Aljoša Vodopija, Tea Tušar, and Bogdan Filipič. 2024. Predicting Algorithm Performance in Constrained Multiobjective Optimization: A Tough Nut to Crack. In International Conference on the Applications of Evolutionary Computation (Part of EvoStar). Springer, 310–325.

[2] Thomas Bäck. 2005. Evolution strategies: An alternative evolutionary algorithm. In Artificial Evolution: European Conference, AE 95 Brest, France, September 4–6, 1995 Selected Papers. Springer, 1–20.

[3] Vahid Beiranvand, Warren Hare, and Yves Lucet. 2017. Best practices for comparing optimization algorithms. Optimization and Engineering 18 (2017), 815–848.

[4] J. Blank and K. Deb. 2020. pymoo: Multi-Objective Optimization in Python. IEEE Access 8 (2020), 89497–89509.

[5] Gjorgjina Cenikj, Gašper Petelin, and Tome Eftimov. 2024. A cross-benchmark examination of feature-based algorithm selector generalization in single-objective numerical optimization. Swarm and Evolutionary Computation 87 (2024), 101534.

[6] Gjorgjina Cenikj, Gašper Petelin, Moritz Seiler, Nikola Cenikj, and Tome Eftimov. 2025. Landscape Features in Single-Objective Continuous Optimization: Have We Hit a Wall in Algorithm Selection Generalization? arXiv preprint arXiv:2501.17663 (2025).

[7] 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.

[8] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. Evolutionary computation 9, 2 (2001), 159–195.

[9] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. 2023. Forecast evaluation for data scientists: common pitfalls and best practices. Data Mining and Knowledge Discovery 37, 2 (2023), 788–832.

[10] John N Hooker. 1995. Testing heuristics: We have it all wrong. Journal of heuristics 1 (1995), 33–42.

[11] Anja Jankovic and Carola Doerr. 2020. Landscape-aware fixed-budget performance regression and algorithm selection for modular CMA-ES variants. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference. 841–849.

[12] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications 80 (2021), 8091–8126.

[13] James Kennedy and Russell Eberhart. 1995. Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks, Vol. 4. IEEE, 1942–1948.

[14] 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.

[15] Benjamin Lacroix and John McCall. 2019. Limitations of benchmark sets and landscape features for algorithm selection and performance prediction. In Proceedings of the Genetic and Evolutionary Computation Conference Companion. 261–262.

[16] Fu Xing Long, Diederick Vermetten, Bas van Stein, and Anna V Kononova. 2023. BBOB instance analysis: Landscape properties and algorithm performance across problem instances. In International Conference on the Applications of Evolutionary Computation (Part of EvoStar). Springer, 380–395.

[17] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. 2011. Exploratory landscape analysis. In Proceedings of the 13th annual conference on Genetic and evolutionary computation. 829–836.

[18] Mario Andrés Muñoz and Michael Kirley. 2021. Sampling effects on algorithm selection for continuous black-box optimization. Algorithms 14, 1 (2021), 19.

[19] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. 2021. Pervasive label errors in test sets destabilize machine learning benchmarks. arXiv preprint arXiv:2103.14749 (2021).

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12 (2011), 2825–2830.

[21] Gašper Petelin and Gjorgjina Cenikj. 2024. On Generalization of ELA Feature Groups. In Proceedings of the Genetic and Evolutionary Computation Conference Companion. 419–422.

[22] Gašper Petelin and Gjorgjina Cenikj. 2025. Benchmarking Pitfalls. https://github.com/gasperpetelin/benchmarking-as-problems. Accessed: 2025-01-20.

[23] Gašper Petelin, Gjorgjina Cenikj, and Tome Eftimov. 2023. Towards understanding the importance of time-series features in automated algorithm performance prediction. Expert Systems with Applications 213 (2023), 119023.

[24] Raphael Patrick Prager and Heike Trautmann. 2023. Pflacco: Feature-based landscape analysis of continuous and constrained optimization problems in Python. Evolutionary Computation (2023), 1–25.

[25] Kenneth Price, Rainer M Storn, and Jouni A Lampinen. 2006. Differential evolution: a practical approach to global optimization. Springer Science & Business Media.

[26] Quentin Renau, Johann Dréo, Carola Doerr, and Benjamin Doerr. 2021. Towards explainable exploratory landscape analysis: extreme feature selection for classifying BBOB functions. In Applications of Evolutionary Computation: 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7–9, 2021, Proceedings 24. Springer, 17–33.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 1135–1144.

[28] 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.

[29] 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.

[30] Eivind Strøm and Odd Erik Gundersen. 2024. Performance metrics for multi-step forecasting measuring win-loss, seasonal variance and forecast stability: an empirical study. Applied Intelligence 54, 21 (2024), 10490–10515.

[31] Ryoji Tanabe. 2022. Benchmarking Feature-Based Algorithm Selection Systems for Black-Box Numerical Optimization. IEEE Transactions on Evolutionary Computation 26, 6 (2022), 1321–1335.

[32] Guohua Wu, Rammohan Mallipeddi, and Ponnuthurai Suganthan. 2016. Problem Definitions and Evaluation Criteria for the CEC 2017 Competition and Special Session on Constrained Single Objective Real-Parameter Optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore. (10 2016).

[33] Renjie Wu and Eamonn J Keogh. 2021. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. IEEE transactions on knowledge and data engineering 35, 3 (2021), 2421–2429.

[34] 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. https://doi.org/10.1016/j.asoc.2020.106138