On the Influence of the Feature Computation Budget on Per-Instance Algorithm Selection for Black-Box Optimization

Koen van der Blom*[1] and Diederick Vermetten*[2]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
kvdb@cwi.nl
[2] Sorbonne Université, CNRS, LIP6, Paris, France
diederick.vermetten@lip6.fr

Abstract. Per-instance algorithm selection (PIAS) takes advantage of complementarity between a set of algorithms by deciding which algorithm to run on a given instance. This decision is based on features of the instances, which, in the context of black-box optimization (BBO), require a part of the optimization budget to be computed. This raises two questions: (a) from which fraction of the budget spent on feature computation does PIAS become worth it for BBO, and (b) which fraction of the budget optimizes the tradeoff between feature accuracy and PIAS performance. To this end, we perform a broad study where PIAS with varying sampling budgets for feature computation is compared to the single best algorithm on a broad range of algorithm selection scenarios. These scenarios consist of two portfolio sizes, three problem sets, 4 dimensionalities, and 10 target budgets. We find that PIAS is viable for the majority of tested scenarios, even when as much as a quarter of the total budget is spent on feature computation. The tradeoff for the fraction of the budget spent on feature computation to maximize the benefit of PIAS is highly dependent on the specific AS scenario. Further, on average 20 percent of PIAS loss to the virtual best solver is explained by the budget spent on feature computation, highlighting the importance of properly accounting for the feature budget.

1 Introduction

For most problems a variety of algorithms is available to solve them, but it is not obvious which one is the best. When there are many instances of the problem, and there is complementarity between the algorithms in terms of for which instances they perform best, this becomes a recurring question. This has resulted in substantial attention for the algorithm selection (AS) problem [28] across different domains, such as SAT [35], MIP [36], and black-box optimization [3] (BBO).

At its core, AS is a learning problem, where given a new problem instance, we aim to select a single method from a range of possibilities. A key question

* Both authors contributed equally.

thus arises in the representation of the problem. For per-instance algorithm selection (PIAS), individual problem instances are typically represented by a set of instance features. In some contexts, these features are inherent from the problem formulation, or can be extracted without incurring a high cost. For example, in SAT solving, there are features that can be computed directly from the instance file [34]. However, when dealing with black-box optimization (BBO), the only a priori available information about an instance is the shape of the search space, e.g., the number and types of decision variables. This is insufficient to differentiate between individual problem instances.

The introduction of exploratory landscape analysis [18] (ELA) provided features that aim to describe the optimization landscape for BBO problems by sampling the objective function.

In turn, this allowed PIAS to be applied in the context of BBO by using part of the total evaluation budget to sample solutions to compute ELA features, while using the remaining budget for the selected algorithm [3,12].

A common baseline for PIAS is the so-called single best solver (SBS), representing the algorithm that performs the best on aggregate across all relevant instances. In many settings, the SBS and the algorithm chosen by PIAS are given the same budget, because the cost of feature computation is negligible. For BBO, however, this cost may be substantial, and, for a fair comparison, the budget for the selected algorithm should be reduced by the budget used for feature computation. The performance of the solutions sampled for feature computation should also be counted towards the selector performance. Although the effect of this may quickly diminish for larger total budgets, it may substantially affect performance in low-budget scenarios. Unfortunately, not all papers in this area make clear whether they account for these factors or not.

In this paper, we compare PIAS and the SBS for various feature computation budgets, in order to investigate the tradeoff between the improved reliability of features when a larger fraction of the total budget is used to compute them, and the loss in performance due to the selected optimizer being left with a smaller budget. By studying a wide variety of AS scenarios with different problem sets, algorithm portfolios, total budgets, and dimensionalities, we aim to provide a more general perspective. Specifically, we aim to answer the following two questions:

Research question A: When is it worth using a fraction of the optimization budget to apply per-instance algorithm selection in black-box optimization?

Research question B: What is the optimal fraction of the total budget to spend on feature computation?

We show that PIAS can reliably be used for most scenarios, even when as much as 25 % of the budget is spent on feature computation. However, the optimal tradeoff varies greatly based on the exact settings of the considered algorithm selection scenario. Finally, we observe that for PIAS on average 11 % to 28 % (depending on the problem set) of the total loss comes from budget spent on feature computation, highlighting the importance of properly accounting for the feature budget.

2 Background and Definitions

To provide the appropriate background, we briefly introduce key concepts in algorithm selection, and notation for the budgets and methods we compare.

The algorithm selection problem [28] considers that for a set of problem instances $\mathcal{I}$, there is often no single algorithm A in a portfolio $\mathcal{A}$ that performs the best on every problem instance. This problem is relevant across domains, but our focus is on BBO. There are some review papers that discuss the particular challenges of PIAS for BBO, but they were done when few practical implementations existed [11,19]. Since then, algorithm selection for BBO has received increased attention primarily in the single-objective continuous setting, including the use of deep-learning models [1,29] and dynamic algorithm selection [16], but has also branched out to, e.g., constrained multi-objective optimization [2]. In terms of features that can be used, a recent survey covered the available feature sets for continuous single-objective BBO [5].

The algorithm selection problem can be formulated as the question of how we can select the best algorithm $A \in \mathcal{A}$ for each instance $i \in \mathcal{I}$, such that we optimize a performance metric $m : \mathcal{A} \times \mathcal{I} \to \mathbb{R}$. To assess the quality of an algorithm selector, two baselines are used most commonly: the single best solver (SBS) and the virtual best solver (VBS). The SBS represents the algorithm that performs the best on aggregate across all instances in $\mathcal{I}$, whereas the VBS represents the hypothetical perfect selector that, for each individual instance in $\mathcal{I}$, chooses the best algorithm in $\mathcal{A}$.

Since for BBO, spending part of the evaluation budget on feature computation is a requirement for PIAS, we have to account for this when evaluating PIAS methods. To this end, we define the full evaluation budget B, the budget spent on sampling for ELA B_{ELA}, and the remaining budget available for optimization after sampling $B_{\text{opt}} = B - B_{\text{ELA}}$ (see Figure 1 for a visual representation of this).

These different budgets in turn influence the performance evaluation of PIAS methods. We define:

- ELA_{perf} as the best performance found based on the samples used for ELA with budget B_{ELA};
- A^*_{perf} as the best performance found by the algorithm $A^* \in \mathcal{A}$ chosen by PIAS with budget B_{opt};
- and $\text{PIAS}_{\text{perf}} = \max(\text{ELA}_{\text{perf}}, A^*_{\text{perf}})$ as the performance achieved by PIAS, accounting for both the points sampled for ELA, and the performance of the chosen algorithm, i.e., considering the full budget B.

Combined with this, we consider the following standard baselines, adjusted to explicitly take into account the BBO-specific complexities:

- Single best solver (SBS_{full}): The algorithm with the best mean performance over all *training* instances when using the *full budget* B. This represents the best choice that could be made without using PIAS.
- Virtual best solver for the full budget (VBS_{full}): The best algorithm per instance when using the full budget B.

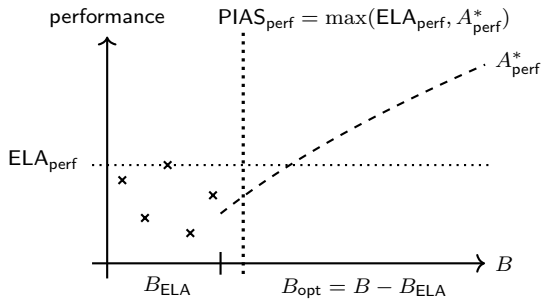


Fig. 1: Schematic visualization showing how the budget is split into feature computation and optimization, and how this relates to selector performance.

– Virtual best solver for the optimization budget (VBS_{opt}): The best algorithm per instance when using the remaining budget after feature computation B_{opt}.

Finally, we note that the performance of an algorithm selector can be measured by looking at the fraction of the so-called VBS-SBS gap that it closes. For this measure, we use the full-budget versions for both VBS and SBS, as this most closely matches the 'oracle' perspective of the VBS. However, in some cases, $\text{PIAS}_{\text{perf}}$ might outperform VBS_{full} if the sampling used for feature calculation finds a better solution than the best algorithm in the portfolio, resulting in a gap-closed measure > 1.

3 Experimental Setup

Since the performance of algorithm selection in continuous black-box optimization settings is highly dependent on both the used problems and algorithms, we make use of a variety of problem suites and algorithm sets in our experiments. From the problem side, we select three sets of problem instances:

– **BBOB** [9]: As the most commonly used benchmark suite for unconstrained single-objective, noiseless optimization, BBOB has naturally been the default for algorithm selection studies [3]. While its choice as a selection testbed has been criticized recently [13], we opt to include it as a setting where PIAS is likely to perform well, and where deeper analysis of the results is possible, by considering, e.g., for which specific problems the selector succeeds or fails.
– The **MA-BBOB** generator [33]: Based on affine recombinations of the base BBOB problems [7], this represents a scenario without the built-in clusters of instances found in BBOB.
– The **RandOptGen** (ROG) generator [30]: This generator represents an alternative scenario without built-in clusters of instances, that is not based on BBOB, but rather creates problems using a tree-based approach.

For each of these problem sets, we consider problem dimensionalities $d \in \{2, 5, 10, 20\}$, and for each dimensionality, we generate a number of instances. For BBOB, we use 10 instances for each of the 24 problems, while for the generators, we generate 250 instances at random.

For our algorithm portfolio, we make use of a set of 22 optimizers from the Nevergrad platform [24].[3] Note that although there is diversity and complementarity among the considered algorithms, the selection is also somewhat arbitrary. This is representative of a realistic scenario, where (exact) relations and complementarity between available algorithms is unknown. We run each algorithm for a total of $2500d$ evaluation, but we construct different algorithm selection settings by taking shorter prefixes of these runs. The different budget settings we consider then become: $B \in \{10d, 15d, 25d, 50d, 100d, 250d, 500d, 1000d, 1500d, 2500d\}$.

Finally, to zoom in on the relative benefits of using some fraction of the budget for feature computation, we use the following feature computation budgets: $B_{\text{ELA}} \in \{5d, 10d, 25d, 50d, 100d, 250d\}$, where we ensure $B_{\text{ELA}} < B$.

For every algorithm, we use 5 independent evaluations for each problem instance. We make use of IOHexperimeter [20] for logging and IOHinspector [32] for processing the performance data. We use the box-constrained versions of all problems, with bounds $[-5, 5]^d$ for BBOB and MA-BBOB and $[-1, 1]^d$ for ROG, following their respective default settings. The full performance data, as well as full reproducibility instructions and additional figures which could not be included here are made available on our Zenodo repository [4].

3.1 Portfolio Size

Since the size of the considered algorithm portfolio can play an important role in the performance of algorithm selection [14], we split our settings into two categories. First, we consider the full portfolio of 22 optimizers. Second, we consider portfolios of size 4, selected to have high **algorithm complementarity**. We use algorithm complementarity as a measure of the potential benefit of algorithm selection, and measure it as the difference between the VBS and SBS performance. Note that this is not intended to create optimal algorithm selection performance (as in [14]), but scenarios in which we can better demonstrate the impacts of the different components of the algorithm selection pipeline.

With this definition of algorithm complementarity, we could find a subset to maximize it for each setting by searching through all possible subsets. However, this is computationally intensive, and since high but non-optimal complementarity is sufficient here, we opt for a heuristic procedure instead. This is a two-stage approach, where we first estimate Shapley values [31] for each algorithm. To this end, we use $\text{VBS}_{\text{full}} - \text{SBS}_{\text{full}}$ as our target measure, and use the marginal contribution of each algorithm to a random set of portfolios to estimate how much

[3] Algorithms considered: RCobyla, DiagonalCMA, NelderMead, OnePlusOne, OnePlusLambda, RLSOnePlusOne, RealSpacePSO, NaiveIsoEMNA, RFMetaModelPSO, DE, RSLSQP, Powell, ManyLN, QODE, CMAsmall, CMAbounded, MultiCMA, CMA, Cobyla, MetaModel, MetaModelOnePlusOne, PMetaModel(CmaFmin2).

each algorithm helps to improve the overall complementarity. These approximated Shapley values are then normalized and used to guide a search for the final portfolio. We use a random search with weighted probabilities for each algorithm based on their Shapley value to find the portfolio of size 4 that maximizes $\mathrm{VBS}_{\mathrm{full}} - \mathrm{SBS}_{\mathrm{full}}$. Separate size-4 portfolios are created for each combination of problem set, budget factor, and dimensionality.

3.2 Algorithm Selection Procedure

Feature Computation While there are a large variety of feature sets available for black-box optimization problems [5], we opt to use ELA, which is most commonly used. Given a budget for feature computation, we use Sobol sampling to get function values, normalize them [22], and calculate the ELA features using the `pFlacco` package [23]. We then perform a filtering stage, where we remove any features that resulted in NaN or infinite values for any instance or are flat across all instances. For each problem instance, feature computation is repeated five times to account for the variance in feature values that may result from different samples.

Performance Measure For our performance measure, we use normalized values in $[0, 1]$ to simplify comparisons. For BBOB and MA-BBOB, we make use of the empirical attainment function [17], with bounds 10^2 and 10^{-8} with a logarithmic scaling between them, following conventional practices from COCO [8]. For ROG, we do not have known optima, so we instead perform a standard normalization per instance using the extrema found by all algorithms between $5d$ and $2500d$ evaluations.[4]

Cross-Validation Throughout our experiments, we employ 5-fold cross-validation, where we train on 4 folds and test on the remaining fold. For the split between folds, we use the instance ID, following the so-called leave-instance-out [3] strategy. We do not consider another popular strategy called leave-problem-out [3], which splits by problem. The reason for this is that the leave-problem-out strategy tests generalization beyond the training distribution [6], which is not the topic of this paper.

Selection Model To construct an algorithm selector, we train a multi-output `RandomForestRegressor` from the `scikit-learn` [21] package. This method was selected for its simplicity, and because earlier work suggests that the type of machine learning model used matters little [15]. Although it is not clear how those results generalize beyond BBOB, for the purposes of this paper a simple selector is sufficient. For the instances in the testing set, the algorithm with the highest predicted performance is selected. For both the training and the testing sets, the mean over all 5 repetitions is used, and matched with each repetition of the feature computation.

[4] One ROG instance for $d = 5$ is excluded from the selector experiments, because function values were too small to reliably measure performance differences.

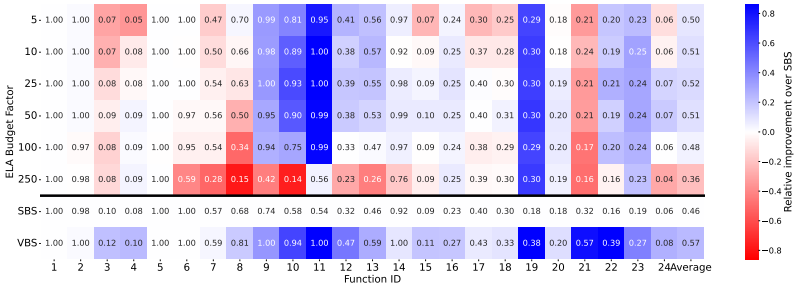


Fig. 2: PIAS performance when trained with different ELA budget factors, compared to the SBS_{full} and VBS_{full} baselines. Each column shows one of the 24 BBOB problems, with results averaged across instances and folds. The total budget factor is 500, $d = 10$, and the portfolio size is 22. Color indicates relative difference to the SBS_{full}, while the numbers indicate the raw performance in terms of attainment.

4 Results

By varying the problem sets, problem dimensionality, algorithm portfolios, and budgets for optimization and feature calculation, we create a wide range of in total 1440 different algorithm selection scenarios. We note that, in principle, each combination represents an independent AS scenario, and care must be taken to avoid drawing too general conclusions based on apparent trends or correlations between results on different scenarios.

4.1 BBOB Results

We start our analysis of the impact of the feature computation budget by looking at the BBOB problem suite in combination with the full algorithm portfolio. By looking at a single algorithm selection scenario, selector performance can be visualized on a per-function basis (averaged over instances), as shown in Figure 2. When also averaging over the functions (shown in the last column in this figure), the tradeoff in terms of feature budget becomes apparent. Performance initially increases as more evaluations are allocated to the ELA part of the selection procedure, but when this factor becomes too large relative to the total budget, it is detrimental to overall performance. This is also visible on individual functions, for example, F10, where the highest ELA budget leads to a substantial drop in performance of the selected algorithm. In this case, using a feature budget of $250d$ leads to a PIAS performance which is clearly below that of the SBS.

Where Figure 2 provides a detailed view of a single algorithm selection scenario, we can aggregate its contents in order to analyze the impact of relative feature budget for a wider range of total budget values. We can subsequently use the fraction of the VBS-SBS gap closed by the selector as the performance measure. With this performance measure, positive values indicate that PIAS

improves over the SBS. In Figure 3, we show the gap closed by PIAS for the BBOB scenarios with the full algorithm portfolio, for each of the 4 selected problem dimensionalities.



Fig. 3: Fraction of the SBS-VBS gap closed (mean over 5 folds) by PIAS for different total budgets, separated by how much budget is spent on feature computation. Each subplot represents a different dimensionality of the BBOB problems, and all use the full algorithm portfolio.

From Figure 3, we can see that the trend observed in Figure 2 seems to hold across most settings: when the feature calculation budget becomes too large relative to the total budget, the selection tends to perform rather poorly. Similarly, we note that the lower feature budgets are insufficient for the smallest total budgets, then become more beneficial, and drop off again relative to larger feature budgets as the total budget increases even further. This shows the inherent tension in allocating a fraction of the budget to feature computation, where too large a fraction leads to poor performance, but too small a fraction is suboptimal for larger total budgets. Although we do not show the results for the portfolio with 4 algorithms on the BBOB suite, in almost all cases, PIAS worked even better.

4.2 Other Problem Suites

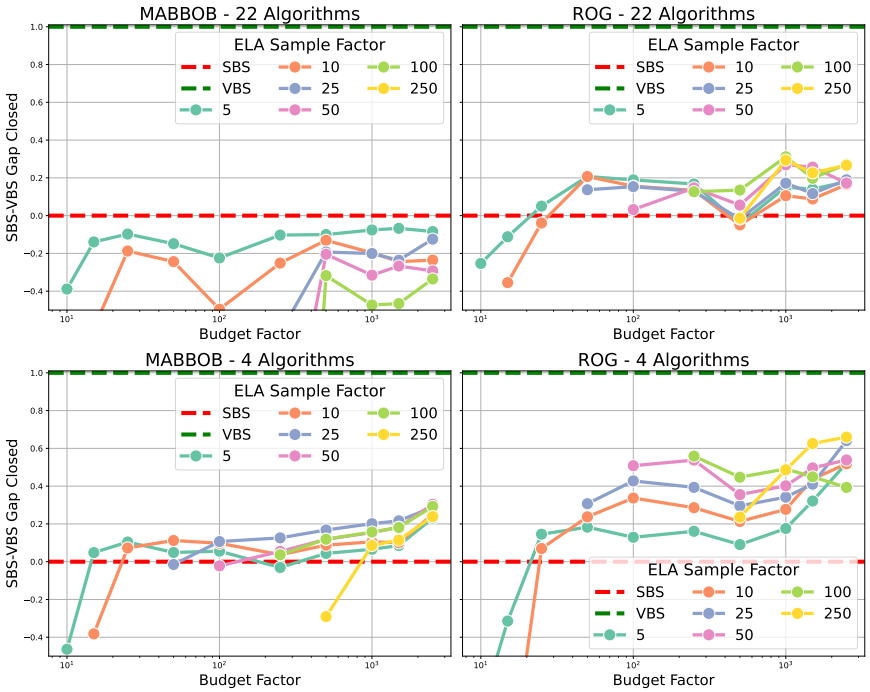


Fig. 4: Fraction of the SBS-VBS gap closed (mean over 5 folds) by PIAS for different total budgets, separated by how much budget is spent on feature computation. Each subplot represents a different problem and algorithm set, as indicated in its title. All plots are using problem dimensionality 5.

We observe that in general, per-instance algorithm selection shows clear benefits over the SBS on the BBOB set when some minimal requirements are met in terms of sampling and total budget. This matches previous observations on BBOB when evaluating PIAS with the leave-instance-out approach [26], which is why it is generally considered a rather simple algorithm selection scenario. To validate whether our findings translate to other problem suites, we show the same gap-closed measure for the MA-BBOB and ROG problems with $d = 5$ in Figure 4. In the first row of this figure, we see the scenarios with the full algorithm portfolio. Here, it is notable that the MA-BBOB scenarios consistently lose out relative to the SBS baseline. In the second row, where the algorithm portfolio has a reduced size, the benefits of PIAS become evident.

The changes of PIAS effectiveness relative to the portfolio size are clear to see in Figure 4. There are two potential factors that influence this. On the one hand, the smaller portfolios have been created in order to maximize algorithm

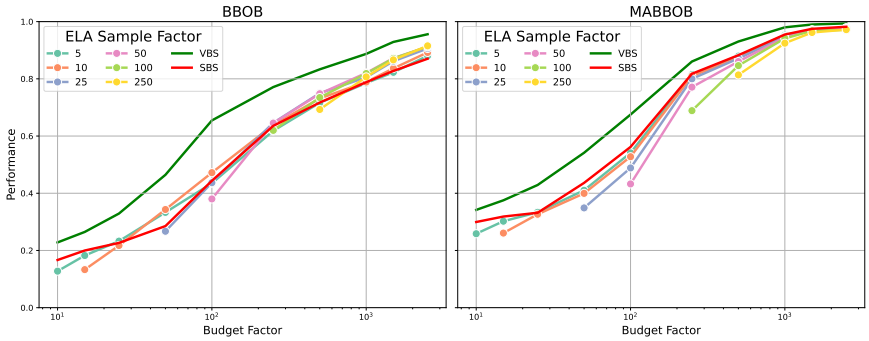


Fig. 5: Performance of PIAS (mean over 5 folds) with different feature computation budgets compared to the VBS and SBS baselines for the 2-dimensional BBOB (left) and MA-BBOB (right) problems. Both plots use the full algorithm portfolios.

complementarity (size of the SBS-VBS gap), which makes it easier for a decent selection process to improve over the SBS. On the other hand, smaller portfolios tend to be easier for selection methods to handle; the regression-based model we use might be less impacted by compounding errors in predicting too many output nodes.

When we look at the MA-BBOB results in particular, we observed that the SBS on the full algorithm portfolio is very strong. In Figure 5, rather than the gap-closed measure, we show the actual performance of $\text{PIAS}_{\text{perf}}$ relative to the SBS_{full} (for a different scenario, with $d = 2$), and notice the clear differences in the size of the SBS-VBS gap between the two scenarios. For MA-BBOB with the full algorithm portfolio (on the right), there are many problem instances where the SBS performs very well, and a limited number of instances where other generally poor-performing methods are much more suitable. Given the limited training set size, it seems to be challenging for our approach to accurately predict which instances fall in these categories, resulting in poor performance on the testing sets. This also provides additional support for the large influence the algorithm portfolio can have on the difficulty of an AS scenario, in addition to the influence of the considered problem set.

To identify when spending a fraction of the budget on feature computation is worthwhile, we looked at the gap-closed by PIAS, which provides a relative measure of performance. By considering only the comparison of the raw performance of the SBS and PIAS, we get a better view of the practical difference. Since we have many scenarios for each problem suite, we take a high-level view of this tradeoff across all scenarios. In Figure 6, we show the difference in performance between the SBS and PIAS for both the BBOB and MA-BBOB suites. This overview confirms the general view that the BBOB scenarios seem to benefit much more clearly from per-instance selection, while the tradeoff is much more mixed in the MA-BBOB setting. An explanation for this is that the BBOB

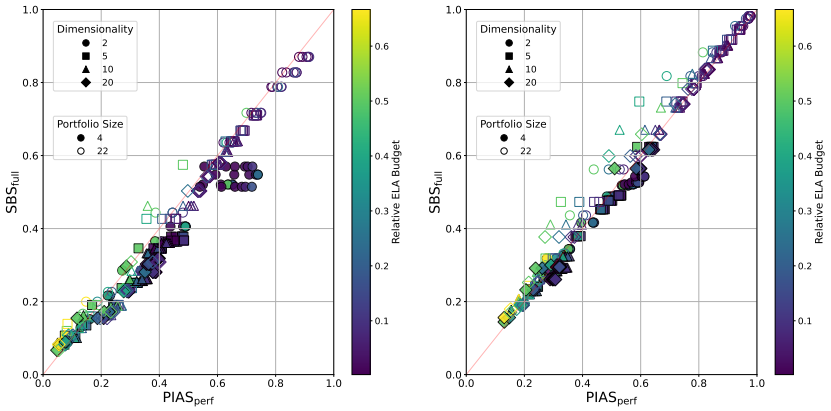


Fig. 6: Comparison of SBS_{full} (y-axis) and $PIAS_{perf}$ (x-axis) results over all settings (mean over 5 folds), for BBOB (left) and MA-BBOB (right) respectively. Color indicates the fraction of budget used for feature calculation. Each dot corresponds to one AS scenario: budget, feature budget, dimensionality. Both portfolio sizes are included. Dots below the diagonal indicate cases where PIAS is beneficial.

problems create a scenario with well-separated clusters of instances belonging to each of the component problems (essentially the prototypical algorithm selection scenario), whereas the MA-BBOB scenarios consist of randomly generated instances without such obvious clusters (potentially a much harder scenario). We also note that the results here in Figure 6 reflect what we previously observed in Figure 4: for MA-BBOB, PIAS almost always failed to be beneficial for the size-22 algorithm portfolio, whereas for the size-4 portfolio (with sufficient budget), it often achieved at least small gains.

4.3 Error Decomposition

To better understand the source of the loss resulting from using different fractions of the budget for feature computation, we lastly look at the two factors that make up the performance of PIAS. On the one side, there is the potential loss in performance the selected algorithm has when it can not use the full budget, because part of it is used for feature computation. The other factor is that the selector should more reliably choose better algorithms as the features more accurately represent the instances when larger feature computation budgets are used [25]. Given this distinction, we decompose the loss of the algorithm selector (how much it loses over the VBS_{full}) into the selection-based loss (how much the selector, $PIAS_{perf}$, loses relative to the VBS_{opt}) and the Budget-based loss (how much the VBS_{opt} loses relative to the VBS_{full}). In Figure 7, we show this decomposition for the BBOB and MA-BBOB problem sets.

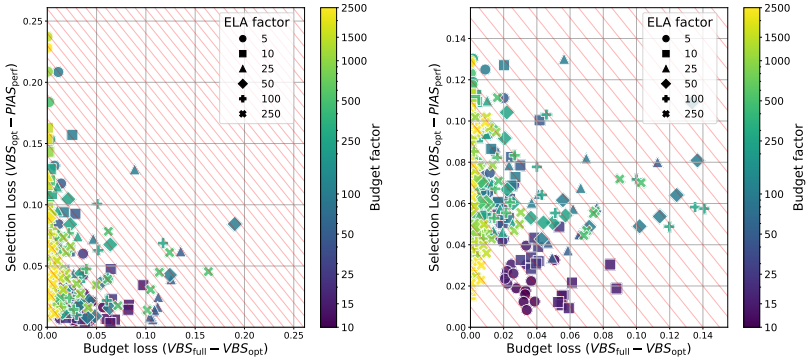


Fig. 7: Decomposition of the $\text{PIAS}_{\text{perf}}$ loss (mean over 5 folds) into the loss that is occurring based on the selection procedure itself ($\text{VBS}_{\text{opt}} - \text{PIAS}_{\text{perf}}$, y-axis), and the loss resulting from the budget needed for feature calculation ($\text{VBS}_{\text{full}} - \text{VBS}_{\text{opt}}$, x-axis). Both the full portfolio and the one with 4 algorithms are included. Each dot represents a single algorithm selection scenario. Left: BBOB. Right: MA-BBOB.

From Figure 7, we see that generally, most of the loss comes from the selection procedure itself. There are also clear patterns present with respect to the optimization budget, which seems to be the main factor that determines budget loss. In order to get a better view of how this relation between budget and selection loss differs based on the budget setting, we can plot it as a function of the relative feature computation budget, as is done in Figure 8. From this figure, we see a clear upward trend as a larger fraction of the budget is spent on feature computation, for all problem settings. The differences between problem suites are also clearly visible. When averaged over all algorithm selection scenarios for each problem suite, we see that the budget-based loss makes up $11, 22$ and $28\,\%$ of the total loss for ROG, MA-BBOB, and BBOB, respectively. This shows that even for settings where the selection works well, the change in optimization budget remains impactful and prevents the algorithm selection from reaching the performance of the VBS_{full}.

5 Conclusions

Per-instance algorithm selection (PIAS) has been successfully applied in domains such as SAT [35], MIP [36], and black-box optimization (BBO) [3]. In BBO, part of the total evaluation budget has to be spent on sampling to compute the instance-features that are needed to apply PIAS. In this work, we investigated the impact on PIAS performance of the fraction of the total budget spent on feature computation. To this end, we also accounted for the performance of the

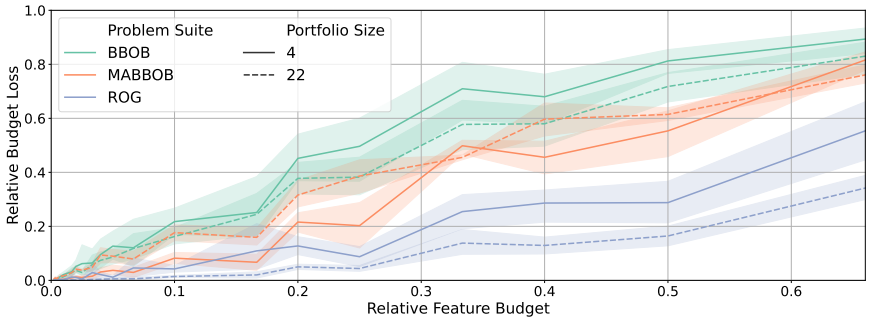


Fig. 8: Average and 95% confidence interval of relative budget loss $(\frac{\mathrm{VBS}_{\mathrm{full}}-\mathrm{VBS}_{\mathrm{opt}}}{\mathrm{VBS}_{\mathrm{full}}-\mathrm{PIAS}_{\mathrm{perf}}})$ across all algorithm selection scenarios and folds, as a function of the relative evaluation budget given to the feature computation.

solutions sampled for feature computation, which may especially have an influence for smaller total budgets. To get a broad view, our experiments covered a wide range of in total 1440 algorithm selection scenarios, including three problem suites (BBOB [9], MA-BBOB [33], RandOptGen [30]), two algorithm portfolio sizes (4, 22), four problem dimensionalities, ten total budgets, and six feature extraction budgets.

We have shown that the budget spent on feature computation can not just be ignored when reporting PIAS performance in the context of BBO. On average across problem suites, we observe that between 11 and 28 % of the final loss of PIAS is caused by having to use a fraction of the evaluation budget for feature computation. While this fraction is highly dependent on the relation between feature computation budget and total budget, it is not negligible, and should thus be properly reported in any algorithm selection study where sampling is required for feature computation.

A core focus of this paper has been to identify when it is worthwhile to use PIAS over sticking with the single best solver (SBS). In this regard, we observed quite some differences between the problem suites used. For scenarios with limited complementarity between optimizers, the SBS can be quite powerful, which leads to the selector performing worse given its lower total budget. However, in the standard BBOB scenario, where problems are complementary by design, we see that even low feature budgets lead to very good selector performance. We believe our results on a broad range of AS scenarios serve as additional evidence that, indeed, BBOB is too easy to meaningfully differentiate between the performance of AS techniques. Even so, we still see value in its use for AS to verify whether new ideas work at all, and, under the right conditions, to perform deeper analysis on specific components of the AS pipeline.

In general, we find that PIAS is quite beneficial in most of the scenarios we considered, with the main determining factors being the complementarity between algorithms in the portfolio and the fraction of the total budget spent

on feature computation. Even with as much as a quarter of the budget spent on feature calculation, PIAS can outperform the SBS. However, without proper evaluation procedures, there is a risk of overstating the benefits of PIAS for BBO. As such, standard benchmarking practices for PIAS need to properly account for this particularity of algorithm selection in a black-box optimization context.

6 Future work

Having looked at a large variety of algorithm selection scenarios, we have seen that the difficulty of the selection scenario is influenced by many different factors. This includes, for example, the algorithm portfolio and the instance set. It remains challenging to fully untangle the impact of each of these factors on the selection procedure. As such, continued development of tools to properly capture both the difficulty of algorithm selection scenarios and the performance of algorithm selectors themselves remains an important research direction.

Our results show that in some cases, the performance achieved from sampling for feature calculation can be better than that of the selected algorithms. Such cases highlight the need for properly integrating the feature computation into the analysis of the algorithm selection results. This does not have to be limited to taking the maximum of the two, but could include warm-starting the selected algorithm, as suggested in, e.g., [12].

This aspect of warm-starting then naturally leads to a more dynamic algorithm selection scenario, where instead of relying on random sampling for feature computation, an initial optimization algorithm is run and its trajectory is used for feature computation [10,27]. Initial studies into this *dynamic algorithm selection* setting have shown promising results [16], but further research is needed to validate its usability in a broader context.

Finally, our results indicate a need for the creation of more testbeds for algorithm selection in the black-box optimization context. While the suites used in our paper are able to generate a variety of algorithm selection scenarios, selection-specific settings, such as algorithm complementarity in both problem and algorithm space, are hard to control. Having more robust ways to create algorithm selection scenarios with given properties would be greatly beneficial towards further study of the different components of PIAS in the BBO context.

Acknowledgments. Koen van der Blom acknowledges funding by NWO WISE grant number 24.0742 "Transfer Learning for Evolutionary Algorithms applied to Optimisation Problems". Diederick Vermetten acknowledges funding by the European Union (ERC, "dynaBBO", grant no. 101125586).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alissa, M., Sim, K., Hart, E.: Automated algorithm selection: from feature-based to feature-free approaches. Journal of Heuristics **29**, 1–38 (2023). https://doi.org/10.1007/s10732-022-09505-4
2. Andova, A., Cork, J.N., Tušar, T., Filipič, B.: A methodology for multi-label algorithm selection in constrained multiobjective optimization. Swarm and Evolutionary Computation **100**, 102246 (2026). https://doi.org/10.1016/j.swevo.2025.102246
3. Bischl, B., Mersmann, O., Trautmann, H., Preuß, M.: Algorithm selection based on exploratory landscape analysis and cost-sensitive learning. In: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. p. 313–320. GECCO '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2330163.2330209
4. Van der Blom, K., Vermetten, D.: On the influence of the feature computation budget on per-instance algorithm selection for black-box optimisation - Reproducibility files (Apr 2026). https://doi.org/10.5281/zenodo.19604700
5. Cenikj, G., Nikolikj, A., Petelin, G., Van Stein, N., Doerr, C., Eftimov, T.: A survey of features used for representing black-box single-objective continuous optimization. Swarm and Evolutionary Computation **101**, 102288 (2026). https://doi.org/10.1016/j.swevo.2026.102288
6. Cenikj, G., Petelin, G., Seiler, M., Cenikj, N., Eftimov, T.: Landscape features in single-objective continuous optimization: Have we hit a wall in algorithm selection generalization? Swarm and Evolutionary Computation **94**, 101894 (2025). https://doi.org/10.1016/j.swevo.2025.101894
7. Dietrich, K., Mersmann, O.: Increasing the diversity of benchmark function sets through affine recombination. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) Parallel Problem Solving from Nature – PPSN XVII. pp. 590–602. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_41
8. Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., Brockhoff, D.: COCO: A platform for comparing continuous optimizers in a black-box setting. Optimization Methods and Software **36**(1), 114–144 (2021). https://doi.org/10.1080/10556788.2020.1808977
9. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA (2009), https://inria.hal.science/inria-00362633
10. Jankovic, A., Eftimov, T., Doerr, C.: Towards feature-based performance regression using trajectory data. In: Castillo, P.A., Jiménez Laredo, J.L. (eds.) Applications of Evolutionary Computation. pp. 601–617. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-72699-7_38
11. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated algorithm selection: Survey and perspectives. Evolutionary Computation **27**, 3–45 (2019). https://doi.org/10.1162/evco_a_00242
12. Kerschke, P., Trautmann, H.: Automated algorithm selection on continuous black-box problems by combining exploratory landscape analysis and machine learning. Evolutionary Computation **27**(1), 99–127 (03 2019). https://doi.org/10.1162/evco_a_00236
13. Kononova, A.V., van Stein, N., Mersmann, O., Bäck, T., Bartz-Beielstein, T., Glasmachers, T., Hellwig, M., Krey, S., Kůdela, J., Naujoks, B., Papenmeier, L.,

Raponi, E., Renau, Q., Rook, J., Schäpermeier, L., Vermetten, D., Zaharie, D.: Benchmarking that matters: Rethinking benchmarking in continuous optimisation for practical impact. In: García-Sánchez, P., Díaz Álvarez, J., Murphy, A. (eds.) Applications of Evolutionary Computation. pp. 327–344. Springer Nature Switzerland, Cham (2026). https://doi.org/10.1007/978-3-032-23607-4_20

14. Kostovska, A., Cenikj, G., Vermetten, D., Jankovic, A., Nikolikj, A., Skvorc, U., Korosec, P., Doerr, C., Eftimov, T.: PS-AAS: Portfolio selection for automated algorithm selection in black-box optimization. In: Faust, A., Garnett, R., White, C., Hutter, F., Gardner, J.R. (eds.) Proceedings of the Second International Conference on Automated Machine Learning. Proceedings of Machine Learning Research, vol. 224, pp. 11/1–17. PMLR (12–15 Nov 2023), https://proceedings.mlr.press/v224/kostovska23a.html
15. Kostovska, A., Jankovic, A., Vermetten, D., Džeroski, S., Eftimov, T., Doerr, C.: Comparing algorithm selection approaches on black-box optimization problems. In: Proceedings of the Companion Conference on Genetic and Evolutionary Computation. pp. 495–498 (2023)
16. Kostovska, A., Jankovic, A., Vermetten, D., de Nobel, J., Wang, H., Eftimov, T., Doerr, C.: Per-run algorithm selection with warm-starting using trajectory-based features. In: Rudolph, G., Kononova, A.V., Aguirre, H., Kerschke, P., Ochoa, G., Tušar, T. (eds.) Parallel Problem Solving from Nature – PPSN XVII. LNCS, vol. 13398, pp. 46–60. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-14714-2_4
17. López-Ibáñez, M., Vermetten, D., Dreo, J., Doerr, C.: Using the empirical attainment function for analyzing single-objective black-box optimization algorithms. IEEE Transactions on Evolutionary Computation **29**(5), 1774–1782 (2025). https://doi.org/10.1109/TEVC.2024.3462758
18. Mersmann, O., Bischl, B., Trautmann, H., Preuss, M., Weihs, C., Rudolph, G.: Exploratory landscape analysis. In: Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation. pp. 829–836. GECCO '11, Association for Computing Machinery, New York, NY, USA (2011). https://doi.org/10.1145/2001576.2001690
19. Muñoz, M.A., Sun, Y., Kirley, M., Halgamuge, S.K.: Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. Inf. Sci. **317**, 224–245 (2015). https://doi.org/10.1016/J.INS.2015.05.010
20. de Nobel, J., Ye, F., Vermetten, D., Wang, H., Doerr, C., Bäck, T.: IOHexperimenter: Benchmarking platform for iterative optimization heuristics. Evolutionary Computation **32**(3), 205–210 (09 2024). https://doi.org/10.1162/evco_a_00342
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
22. Prager, R.P., Trautmann, H.: Nullifying the inherent bias of non-invariant exploratory landscape analysis features. In: Correia, J., Smith, S., Qaddoura, R. (eds.) Applications of Evolutionary Computation. pp. 411–425. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30229-9_27
23. Prager, R.P., Trautmann, H.: Pflacco: Feature-based landscape analysis of continuous and constrained optimization problems in Python. Evolutionary Computation **32**(3), 211–216 (09 2024). https://doi.org/10.1162/evco_a_00341
24. Rapin, J., Teytaud, O.: Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad (2018)

25. Renau, Q., Dreo, J., Doerr, C., Doerr, B.: Expressiveness and robustness of landscape features. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. pp. 2048–2051. GECCO '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319619.3326913
26. Renau, Q., Dréo, J., Doerr, C., Doerr, B.: Towards explainable exploratory landscape analysis: Extreme feature selection for classifying BBOB functions. In: Castillo, P.A., Jiménez Laredo, J.L. (eds.) Applications of Evolutionary Computation. pp. 17–33. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-72699-7_2
27. Renau, Q., Hart, E.: On the utility of probing trajectories for algorithm-selection. In: Smith, S., Correia, J., Cintrano, C. (eds.) Applications of Evolutionary Computation. pp. 98–114. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-56852-7_7
28. Rice, J.R.: The algorithm selection problem. In: Advances in computers, vol. 15, pp. 65–118. Elsevier (1976). https://doi.org/10.1016/S0065-2458(08)60520-3
29. Seiler, M.V., Kerschke, P., Trautmann, H.: Deep-ELA: Deep exploratory landscape analysis with self-supervised pretrained transformers for single- and multi-objective continuous optimization problems. Evolutionary Computation pp. 1–27 (02 2025). https://doi.org/10.1162/evco_a_00367
30. Seiler, M.V., Preuß, O.L., Trautmann, H.: RandOptGen: A unified random problem generator for single- and multi-objective optimization problems with mixed-variable input spaces. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 76–84. GECCO '25, Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3712256.3726478
31. Shapley, L.S.: A value for n-person games. In: Contributions to the Theory of Games, volume II, pp. 307–317. Princeton University Press. (1953)
32. Vermetten, D., Rook, J., Preuß, O.L., de Nobel, J., Doerr, C., López-Ibañez, M., Trautmann, H., Bäck, T.: MO-IOHinspector: Anytime benchmarking of multi-objective algorithms using IOHprofiler. In: Singh, H., Ray, T., Knowles, J., Li, X., Branke, J., Wang, B., Oyama, A. (eds.) Evolutionary Multi-Criterion Optimization. pp. 242–256. Springer Nature Singapore, Singapore (2025). https://doi.org/10.1007/978-981-96-3506-1_17
33. Vermetten, D., Ye, F., Bäck, T., Doerr, C.: MA-BBOB: A problem generator for black-box optimization using affine combinations and shifts. ACM Trans. Evol. Learn. Optim. **5**(1) (Mar 2025). https://doi.org/10.1145/3673908
34. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Features for SAT. University of British Columbia,, Tech. Rep (2012)
35. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research **32**, 565–606 (2008). https://doi.org/10.1613/jair.2490
36. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming. In: Proceedings of the 18th RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion (RCRA 2011). pp. 16–30 (2011)