A Novel Multi-Target Implementation for Highly Parametrized Algorithms Optimization

Marco Di Francesco
University of Twente
m.difrancesco@student.utwente.nl

Jeroen Rook
University of Twente
j.g.rook@utwente.nl

Abstract

This paper investigates the optimization of algorithms with many parameters, a common challenge in areas such as SAT solving, mixed integer programming, AI planning, and machine learning, through Automated Algorithm Configuration (AAC). We compare model-free and model-based AAC methods, with a focus on Sequential Model-Based Optimization (SMBO) and its application in Sequential Model-based Algorithm Configuration (SMAC). We explore multi-objective optimization, analyzing the performance of ParEGO and MO-SMAC. The core contribution of this work is the development of MT-SMAC, a multi-target model using a single surrogate model for all objectives, leveraging a Multi-task Gradient Boosting Machine to achieve a more efficient prediction process by understanding correlations between targets. The paper's empirical section utilizes the YAHPO gym for benchmarking, providing a comparative analysis of the proposed models. Concluding remarks suggest future research directions, including the exploration of Predicted Hypervolume Improvement and the potential of cross-validation to prevent overfitting, aiming to refine the process of algorithm configuration further.

1. Introduction

Algorithms designed for complex computational problems are typically highly parametrized, involving tens of parameters. Algorithms with high numbers of parameters range in many fields, e.g. Boolean Satisfiability Problem (SAT), mixed integer programming (MIP), AI planning, and machine learning. The task for automatically finding parameters for these algorithms is called Automated Algorithm Configuration (AAC).

AAC traditionally focuses on optimizing parameters to achieve a singular target, such as maximizing accuracy in machine learning applications. However, this singular focus often fails to address the multifaceted nature of real-world problem scenarios, where multiple objectives, such as model size, memory requirements, and training time, are equally critical. This limitation has spurred the development of multi-objective models. These models enable a comprehensive evaluation, revealing trade-offs among conflicting objectives. For instance, implementations like ParEGO [5] utilize a weighted sum approach to balance various objectives, while MO-SMAC [12] employs separate surrogate models for each objective. However, MO-SMAC's approach has limitations. Primarily, the loss of correlation information between objectives due to isolated model training. Additionally, increased memory usage, along with training and inference time, given the usage of multiple models.

Contributions. We propose in this paper the implementation of a multi-target surrogate model that overcomes these limitations by useing one surrogate model for all the objectives using a Multi-task Gradient Boosting Machine [14].

2. Background and Related Work

Algorithm configuration (AC) has two major categorizations: model-free and model-based configurations.

Model-free AC methods are considered to be traditional methods and do not rely on surrogate models when predicting the next configuration to test. Generic Genetic Algorithm (GGA, 2007) by Jiao et al. [11] is a genetic-based approach for optimizing algorithm parameters that maintain a population of candidate solutions and evolve over generations; it has shown to have good performance although code was not shared publicly. Friedman Racing (F-RACE, 2009) [2] employs statistical hypothesis tests to evaluate and compare various configurations and by eliminating poorly performing ones; it has shown to have good performance in case the number of configurations that need to be tested is high. Iterated Racing procedure, commonly referred to as i-RACE (2016) [8] is a configurator that generalizes F-RACE, incorporating adaptive mechanisms to refine and focus the search for optimal configurations over successive iterations; it has gained practical application due

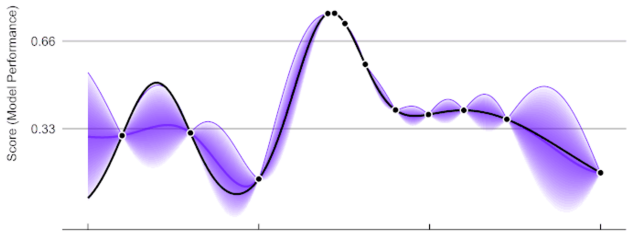

Figure 1. Bayesian optimization of a function (black) with a Gaussian Process (purple) [1].

to its efficiency in handling a large number of configurations. Parametric Iterated Local Search (PARAMILS, 2009) by Hutter et al. [4] is adept at configuring highly parameterized tree search and local search solvers; it can handle categorical parameters directly, but it requires numerical parameters to be discretized; this approach has been effective when the number of configurations is limited and the target algorithm is costly to evaluate. Hyperband (2016) by Li et al. [7] a method that uses the bandit-based approach to adaptively allocate more resources to promising configurations and quickly discard suboptimal ones.

Model-based AC methods employ a surrogate model that mimics the behavior of the actual algorithm, making an approximation of the target value for a given parameter configuration. This approach is advantageous since running the surrogate model is computationally less expensive than executing the actual algorithm itself. Popular surrogate models are Gaussian Processes (GP) and Random Forests. A GP (sample figure 1) is a probabilistic model that aims to fit the real distribution (black line) by handling uncertainty in predictions (purple area), and in the context of AAC, this is useful because it enables more informed choice of the algorithm parameters. A Random Forest is another category of surrogate models that function based on decision trees which enable the model to predict not only numerical features but also categorical ones.

2.1. SMBO and SMAC

In the context of AAC, Sequential Model-Based Optimization (SMBO) has emerged as a significant method. Introduced by Hutter et al. in their 2011 study [3], SMBO represents a general framework for algorithm configuration. This approach has the ability to handle a wide range of parameters, including both categorical and numerical types. SMBO is designed to optimize algorithm performance across a set of instances. The authors present in the same paper the implementation of SMBO called Sequential Model-based Algorithm Configuration (SMAC).

Specifically, SMBO efficiently assesses configurations during the intensification phase and concurrently constructs a surrogate model updated by these evaluations. This model

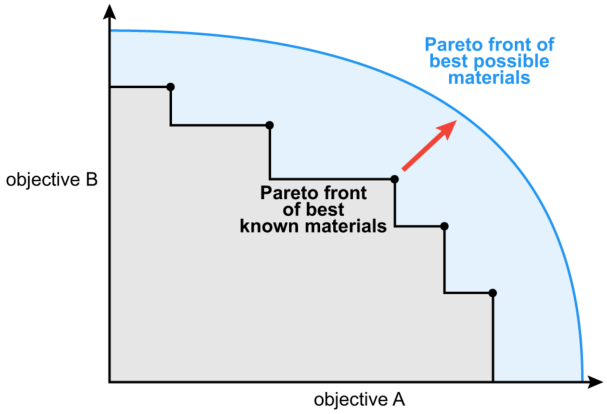

Figure 2. Pareto front example. Source: [9].

is then used within an optimization loop to identify and select new promising configurations for subsequent evaluations. This process involves an alternation between the intensification of existing configurations and the exploration of new ones. More specifically, the *Intensification Mechanism* in SMAC dedicates more computational effort to promising configurations. By evaluating these configurations SMAC aims to estimate their performance, ensuring that the optimization process is directed towards the most promising areas of the configuration space. Moreover, SMAC uses *Expected Improvement* (EI) as a metric to select the next configuration to evaluate. It is based on the surrogate model's predictions and aims to balance the exploration of new configurations with the exploitation of known good ones. EI helps guide the search process efficiently by quantifying the potential benefit of exploring each configuration.

2.2. Multi-Objective Optimization

Multi-Objective optimization extends the principles of single-objective optimization to handle multiple conflicting objectives simultaneously, enabling the algorithm to optimize multiple conflicting objectives simultaneously to find the trade-off between them. One relevant aspect of multi-objective optimization is the *Pareto Frontier*, in the context where multiple conflicting objectives are involved. As shown in the example figure 2 it refers to a situation where one configuration is not dominated by others (black points), which is the case where one configuration is better in all objectives than another. In practice, the aim is to identify a set of configurations that builds an approximation (black line) of the Pareto frontier (blue line).

ParEGO. One implementation of multi-objective optimization in the SMAC framework is Pareto Efficient Global Optimization [5], which extends the Efficient Global Optimization (EGO) strategy to handle multiple conflicting objectives. It operates by combining multiple objectives by

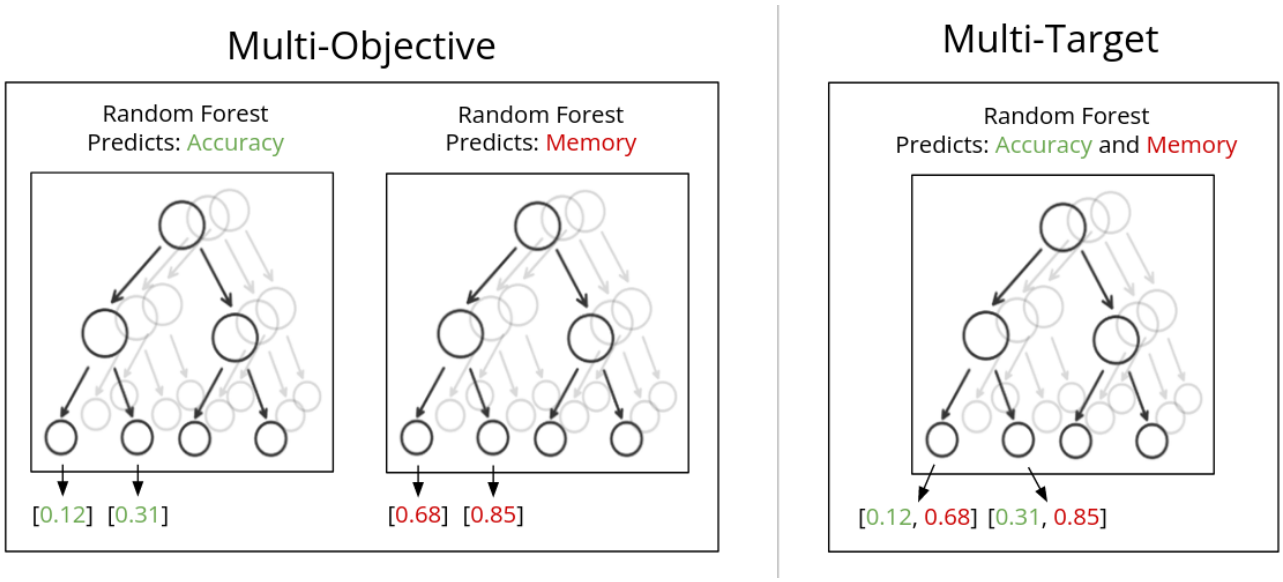

Figure 3. Comparative illustration of MO-SMAC (left) and MT-SMAC (right) framework surrogate model components.

applying a weighted sum to combine different objectives into a single scalar objective. This technique enables SMAC to operate as though addressing a single-objective problem, despite the underlying multi-objective context. However, in situations where the trade-offs between objectives are complex or non-linear, it may be more difficult to represent the ideal balance between them using a linear combination, making this scalarization approach less effective.

MO-SMAC. Multi-Objective SMAC by Rook et al. [12] is an implementation and introduces a novel approach to tackle multi-objective optimization problems without scalarization. In contrast to the Expected Improvement (EI) criterion, MO-SMAC utilizes the concept of *Hypervolume Improvement* (HVI). EI focuses on the potential gain over the current best single objective, Hypervolume Improvement considers the multi-dimensional space formed by all objectives. It measures the increase in the dominated volume of the space dominated by the Pareto frontier after adding a new configuration. The paper further presents two variations for HVI: *Expected Hypervolume Improvement* (EHVI) and *Predicted Hypervolume Improvement* (PHVI). The latter adopts a more straightforward approach, focusing on the predicted improvement in hypervolume based on deterministic predictions, without explicitly modeling uncertainty. On the other hand, EHVI integrates uncertainty into the to expanding the Pareto frontier to estimate the expected value of HVI.

2.3. From Multi-Objective to Multi-Target

MT-SMAC. Transitioning from Multi-Objective to Multi-Target in algorithmic predictions shifts from using individual Random Forest (RF) models for each objective to a singular RF model to target all objectives concurrently. The intuition can be understood in figure 3. In the work by Krocev et al. [6] they adapt the structure of Random Forests to accommodate multiple targets within a single model, where each leaf now contains a vector of values corresponding to each target, rather than a single value, and they can leverage the inherent correlations between different targets. This enables better predictive performance by considering the interdependencies among the targets, and also potentially reduces memory footprint and computational overhead by using one instead of multiple RFs.

3. Methods

Codebase. The MT-SMAC implementation has been realized, and the corresponding source code is accessible on GitHub at (`https://github.com/MarcoDiFrancesco/MT-SMAC`). This codebase originated from the existing MO-SMAC repository, serving as the framework for our development.

Compared models. Our method is compared against ParEGO, implemented in the official SMAC3 repository [1]; and MO-SMAC implementation by Rook et al. over SMAC3 [2]. Our new approach, named *MT-SMAC* was implemented over the MO-SMAC implementation, using as a model the XDBoost library version 2.0.1 [13] Python APIs. When calling this function model the *multi_output_tree* parameter was specified to obtain multi-output trees.

[1] `https://automl.github.io/SMAC3/main/examples/3_multi_objective/2_parego.html`

[2] `https://github.com/jeroenrook/SMAC3/tree/38b22d4bf4d97dd5351be2ce943a7aa1d88eb607`

Implementation. Building on this foundation, the project involved the introduction of a new facade as an abstraction layer on top of the SMBO backend. The facade structures the components of the Bayesian Optimization process, managing components like the target function and instantiation of the surrogate model. In the implementation, we created a new facade named *ACFacadeMO* starting from the already implemented *ACFacade*. The new surrogate model class was named *RandomForestMO*, where we defined the acquisition function to be PHVI. Inside the *RandomForestMO* surrogate model class, we instantiated the model using the *XGBRegressor* API[3] specifying the multi-output tree strategy. This implementation of the random forest uses regression trees to include both numerical features and categorical features.

Performance Synthesis. In our approach, the operational sequence for utilizing the surrogate model to estimate performance averaged across various instances is as follows: first (1) within each tree, we predict performance for the combinations of the given configuration and each instance; then (2) we compute means and variances across trees; finally (3) we combine these predictions with the arithmetic mean across instances. This was changed from the SMAC implementation that swaps step 2 and 3. That implementation was not possible given that the XGDBoost library still does not give access to the leaf values in the Python APIs.

Research questions. Our study aims to evaluate the performance of our recently proposed MT-SMAC model against existing methods. We want to determine how well our model performs in terms of prediction accuracy when compared to the models mentioned above. Furthermore, we examine the speed at which the models converge the fastest towards identifying optimal incumbents in the search space. In addition, we intend to assess the computational cost with regard to execution time. This is done to understand if the MT-SMAC model provides a faster solution than the MO-SMAC model, in comparable computational circumstances.

Experiments. Given these research questions, we propose three experiments. The first experiment was done to visualize the incumbents and non-dominated points after the training phase of each algorithm. The second experiment involves visualization of the incumbent distribution over multiple runs in order to visualize how the incumbents were generalizing over all the instances. This test was run for 150 trials over all 119 instances, and for each configuration and instance it was run 10 times by changing the seed, each time keeping incumbents separate. Testing these configurations took 35 minutes using one Intel i7-7700HQ CPU. The third experiment was the visualization of the hypervolume for each iteration, to understand which model is converging faster. This was done by saving the incumbent configurations for each iteration and repeating this for each model. Both metrics are then max normalized to get a maximum volume of 1.

[3] `https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBRegressor`

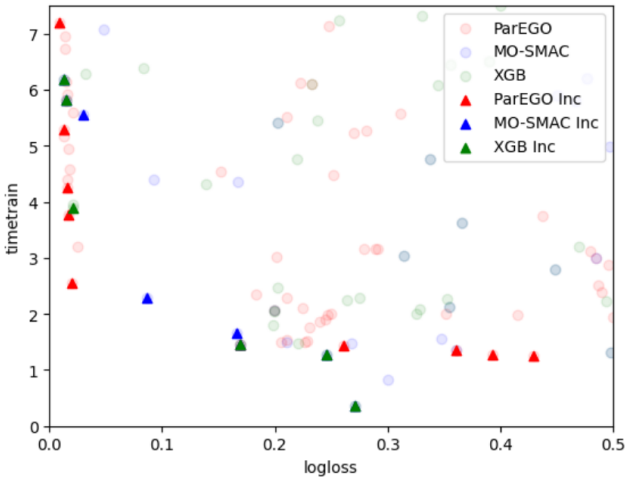

Figure 4. Plot showing configurations for training. Non-dominated points (circles) and incumbents (triangles) over the models ParEGO (red), MO-SMAC (blue), MT-SMAC (green).

Experiment implementation. Experiments are implemented in the *evaltemplate-yahpo* python notebook included in the repository. These experiments were conducted using *YAHPO gym* [10] version 1.0, a collection of problem sets for benchmark hyperparameter and black-box optimization. The tests were performed using the *rbv2_ranger* scenario featuring an 8 dimension search space (e.g. number of trees, minimum size of a node), and 119 instances. Target metrics in this experiment include performance (balanced accuracy, F1 score, log loss), running time (train time, prediction time), and memory footprint measured in RAM consumption. Our experiment included all dimensions and all instances beforementioned, considering two objectives: training time and Log Loss. These metrics were chosen given the nature of being conflicting, to look at the tradeoff.

4. Results and discussion

(1) Incumbents after training. Results for the visualization of the incumbents (figure 4) show the ability of all the algorithms to find good incumbent configurations. A distinction is observed in ParEGO's strategy, which repeatedly targets comparable configurations, leading to a clustering of data points. This approach results in considerable time being expended on minor enhancements, in contrast to the more efficient strategies employed by the other two models.

(2) Incumbents on multiple runs. Results for the in-

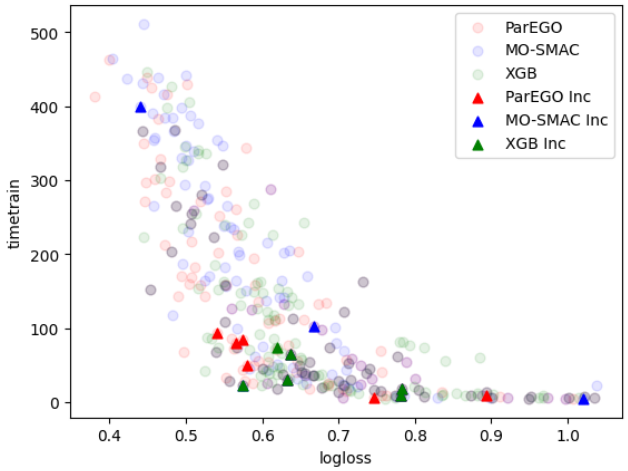

Figure 5. Plot comparing non-dominated points (circles) and incumbents (triangles) over the models ParEGO (red), MO-SMAC (blue), MT-SMAC (green)

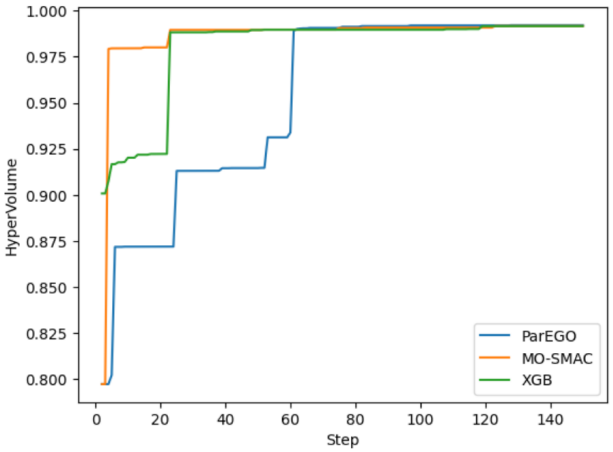

Figure 6. Plot with hypervolume (y-axis) and step (x-axis) over the models ParEGO (blue), MO-SMAC (orange), MT-SMAC (green)

cumbent distribution over multiple runs (figure 5) show the generalization ability of the models. In these iterations, MO-SMAC was less effective in identifying optimal candidates relative to the other two models.

(3) Hypervolume convergence. The final experiment (figure 6) demonstrates the speed with which the models identify effective incumbents, thereby indicating their convergence rate. Here convergence threshold is established at a hypervolume of 0.98. Both the MO-SMAC and MT-SMAC models achieved this level within 24 iterations, whereas the ParEGO model reached this point in 64 iterations.

(4) Model training time. The CPU time required for the fitting phase of each model, as shown in figure 7, was

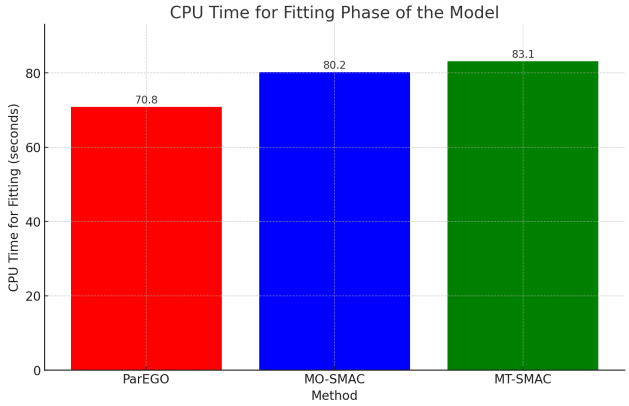

Figure 7. CPU time required to fit the models in three optimization methods: ParEGO, MO-SMAC, and MT-SMAC.

70.8 seconds for ParEGO, 80.2 seconds for MO-SMAC, and 83.1 seconds for MT-SMAC. These results indicate that the hypothesis suggesting the use of a Multi-task Gradient Boosting Machine to accelerate training speed does not hold in this context.

5. Conclusions and Future Works

(1). The experiment for the visualization of the incumbents during training (figure 4) shows that all models successfully identified viable incumbents. One noticeable strategy difference between MT/MO-SMAC and ParEGO is the tendency of the latter to try different configurations very close to each other, creating clusters of (red) points that result in a high usage of resources for small improvement, compared to MT/MO-SMAC that keep a high variance in the distance of each configuration.

(2). The experiment incumbent distribution over multiple runs (figure 5) shows the generalization ability of the incumbents found by each model, and here the models MT-SMAC and ParEGO are able to find good incumbents, compared to the incumbents MO-SMAC found that is far from the approximated Pareto front.

(3). Experiment to visualize hypervolume for each iteration (figure 6) shows the ability of MT-SMAC and MO-SMAC to reach a hypervolume of 0.98 in 26 steps compared to 64 steps of ParEGO. This signals that MO-SMAC and MT-SMAC are capable of finding incumbents faster. This finding underscores the efficiency of these models in rapidly converging to optimal solutions.

(4). Model training time plot (figure 7) does not show noticeable improvement in training time for the Multi-Target model compared to Multi-Objective and instead found that MT-SMAC took approximately 15% more time than ParEGO and 3.5% more than MO-SMAC. This may be given since this test did not consider any memory usage

differences, thus below we propose an improvement to this method.

5.1. Future work on Empirical Evaluation

(3). One future step will be to try out the experiment of hypervolume convergence for each training step with more than two objectives. This is likely to converge slower and would give the chance to see which model converges faster in case of more dimensions.

(4). The difference in the MO-SMAC and MT-SMAC training time should be further investigated by training multiple times the models and averaging the performance. This can be achieved by repeatedly training the models and computing the average performance. Such an investigation is necessary due to the observed high variance in training times across different runs (not reported here), where MT-SMAC frequently outperformed MO-SMAC. Additionally, it is important to ensure that both models utilize equivalent memory and CPU resources, especially considering the potential use of multi-processing in their implementations.

New gym settings. Further testing should also extend to other scenarios provided by YAHPO gym, as well as different configuration budgets. This broader range of tests will help in assessing the versatility and adaptability of the models across various settings.

RMSE for each model. An additional experiment that can be conducted in future work involves evaluating the localized performance of the surrogate model by computing the Root Mean Square Error (RMSE) for each model. This assessment would involve randomly selecting a configuration and using the gym objective function to get the actual value. This real value would be compared with the model's predicted value. The RMSE would be determined by averaging these differences across a large number of configurations (e.g. 1000). This approach would provide a quantitative measure of the model's accuracy in predicting outcomes for individual configurations.

5.2. Future work on Method

(3). An enhancement that could have resulted in more distinct convergence speed outcomes is the implementation of a restricted range of results when measuring the hypervolume at each step. For example cropping 400 seconds of training time down to 7 from 3.1 log loss to 0.5, thus reducing the volume by a factor of 354. This would have discarded some non-dominated points, but would have made the plot much more readable since right now it's hard to notice a hypervolume improvement after step 64, and in these conditions, an improvement of 0.001 at step 130 would be still very relevant.

EHVI implementation. One future step will be to use Expected Hypervolume Improvement (EHVI) instead of Predicted Hypervolum Improvement (PHVI). EHVI is designed to take into account the uncertainty in the predictions of the optimization model. The application of EHVI within MT-SMAC was precluded by the inability to extract the marginalized variance over instances from the XGB surrogate model, thus necessitating the use of PHVI, which does not account for variance. The development of EHVI's implementation is ongoing, with updates accessible in GitHub repostiroy[4].

Cross-validation. A future direction for our methodology could involve partitioning the instances into training and testing sets through cross-validation. This approach would be instrumental in mitigating the risk of overfitting, ensuring that our model's performance is robust and generalizable across different data segments.

References

[1] AnotherSamWilson. Bayesian optimization of a function (black) with a gaussian process (purple). three acquisition functions (blue) are shown at the bottom.

[2] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated f-race: An overview. In Thomas Bartz-Beielstein, Marco Chiarandini, Luís Paquete, and Mike Preuss, editors, *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336. Springer Berlin Heidelberg.

[3] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration (extended version).

[4] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stuetzle. ParamILS: An automatic algorithm configuration framework. 36:267–306.

[5] J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. 10(1):50–66.

[6] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Tree ensembles for predicting structured outputs. 46(3):817–833.

[7] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization.

[8] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. 3:43–58.

[9] Benjamin P. MacLeod, Fraser G. L. Parlane, Connor C. Rupnow, Kevan E. Dettelbach, Michael S. Elliott, Thomas D. Morrissey, Ted H. Haley, Oleksii Proskurin, Michael B. Rooney, Nina Taherimakhsousi, David J. Dvorak, Hsi N. Chiu, Christopher E. B. Waizenegger, Karry Ocean, Mehrdad Mokhtari, and Curtis P. Berlinguette. A self-driving laboratory advances the pareto front for material properties. 13(1):995. Number: 1 Publisher: Nature Publishing Group.

[10] Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. YAHPO gym – an efficient

[4] https://github.com/dmlc/xgboost/issues/9043

multi-objective multi-fidelity benchmark for hyperparameter optimization.

[11] Jianxin (Roger) Jiao, Yiyang Zhang, and Yi Wang. A generic genetic algorithm for product family design. 18(2):233–247.

[12] Rook, Jeroen et al. MO-SMAC: Multi-objective sequential model-based algorithm configuration.

[13] Tianqi Chen. XGBRegressor python API reference — version 2.0.2.

[14] ZhenZhe Ying, Zhuoer Xu, Zhifeng Li, Weiqiang Wang, and Changhua Meng. MT-GBM: A multi-task gradient boosting machine with shared decision trees.