Parameter Prediction Under Ambiguity: Single-Target vs. Multi-Choice Models for IEA Configuration on QAP

Paweł Kolendo[1][0009−0008−3685−280X], Wojciech Chmiel[1][0000−0002−4773−9123], and Joanna Kwiecień[1][0000−0002−8225−7605]

AGH University of Krakow, al. Mickiewicza 30, 30-059 Krakow, Poland {kolendo, wch, kwiecien}@agh.edu.pl

Abstract. Island Evolutionary Algorithms (IEAs) are effective for solving hard problems such as the Quadratic Assignment Problem (QAP), but their performance depends on careful and computationally expensive parameter tuning. While ML has been used to automate solver tuning in SAT (and related constraint-solving problems), its transfer to other problem classes with a different structure is less explored. Moreover, most ML-based approaches assume a single optimal configuration, ignoring the fact that multiple parameter settings can yield comparable performance. In this work, we formulate parameter prediction as an inconsistent learning problem and investigate whether instance features can be used to predict effective IEA configurations for QAP. We compare three strategies: a regression-based performance prediction model, a multiple-choice model that treats near-optimal configurations as valid targets, and a baseline single-label model. The models are trained on effective parameters for 867 QAP instances tuned with SMAC, using instance-level features for training. The regression approach achieved poor predictive performance ($R^2 = 0.16$ on training and -0.16 on the test set). The multiple-choice model improved generalization (test accuracy 0.85 vs. 0.66), but did not improve regression quality ($R^2 \approx 0.23$) or downstream optimization. The baseline model achieved the best results, producing configurations that did not differ significantly from SMAC-tuned parameters according to Wilcoxon paired test. These findings suggest that direct configuration prediction provides a robust approach despite ambiguity in the parameter space.

Keywords: Parameter tuning · Algorithm configuration · Quadratic Assignment Problem · Combinatorial Optimization · Metaheuristics · Hyperparameter Optimization · Performance Prediction · Evolutionary Algorithms

1 Introduction

Optimization algorithms have become indispensable for solving complex computational problems in various domains, ranging from engineering design to

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

machine learning. Among these, Island Evolutionary Algorithms (IEAs) have proven to be powerful tools due to their ability to explore vast solution spaces efficiently. However, the performance of IEAs is highly dependent on their parameter settings, such as population size, mutation rate, and crossover probability [14], as well as other algorithm-specific parameters. Achieving optimal or near optimal performance often requires fine-tuning these parameters, a process that is both challenging and computationally expensive. The tuning phase is critical for optimizing algorithm performance, and omitting this step can significantly hinder algorithm performance [5]. Traditional rules of thumb often lead to inefficient parameter configurations [30], making it extremely challenging to propose parameters that align well with the specific problem at hand.

This study applies machine learning to predict effective parameter configurations for IEAs solving the Quadratic Assignment Problem (QAP). While prior research has demonstrated the effectiveness of ML-based parameter prediction in domains such as SAT [4], its applicability to other problems has received limited attention. In particular, there is a lack of systematic studies that examine whether such approaches generalize to structurally different problems like QAP.

Moreover, existing ML-based parameter tuning methods typically assume a single optimal configuration for each problem instance. However, in practice, multiple parameter configurations often yield comparable performance. In this setting, treating parameter selection as a single-label prediction is problematic, as it penalizes models for predicting valid but unlabeled configurations. This ambiguity in the solution space has not been adequately addressed in the literature.

To bridge these gaps, this study proposes and compares three modeling strategies that differ in how they handle performance equivalence among parameter configurations:

1. **Regression-Based Model:** A neural network that predicts the expected performance of a given parameter configuration and selects the configuration minimizing the objective value.
2. **Multiple-Choice Model:** A model that allows multiple parameter configurations to be considered correct, addressing cases where several near-optimal solutions exist.
3. **Baseline Model:** A baseline approach that learns and predicts only the single best-performing configuration observed in the training data.

This comparison allows us to assess which approach best handles the ambiguity inherent in near-optimal solutions.

This study addresses three main research questions:

(RQ1) Can machine learning models use features of QAP instances to predict better IEA parameter settings than the standard configuration?

(RQ2) Does allowing multiple near-optimal parameter configurations as valid solutions improve model performance compared to selecting a single labeled configuration?

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

(RQ3) Which modeling strategy–regression-based, multiple-choice, or baseline–provides the most robust and effective approach for parameter selection in the presence of solution ambiguity?

Answering these research questions will help determine the feasibility of predicting effective parameter configurations for the QAP and will clarify how multiple near-optimal configurations can be identified and managed within a parameter prediction framework.

The outcomes of this research will clarify whether instance-level features of the QAP can reliably predict effective IEA parameter settings, or to what extent noise, instance variability, and performance plateaus constrain such predictions. Evidence supporting this relationship would position ML-driven parameter selection as a practical alternative to costly manual tuning, while more limited predictability would point toward the need for adaptive or hybrid approaches. In all cases, the findings will provide empirical insight into the structure of near-optimal parameter spaces and contribute to advancing automated algorithm configuration for complex combinatorial optimization problems.

The project repository[1] provides open access to all source code, generated data, and experimental findings.

2 Quadratic Assignment Problem

The Quadratic Assignment Problem (QAP) was introduced by Koopmans and Beckmann in 1957 as a mathematical model describing assigning a set of economic activities to a set of locations [20]. For the given set $N = \{1, ..., n\}$ we define two non-negative matrices $F = [f_{ik}]_{n\times n}$ and $D = [d_{jl}]_{n\times n}$. In the terminology of facilities location the set N is a set of facilities indexes and $\pi(i) \in N, i = 1, ..., n$ defines the set of location to which the facilities are assigned. The matrix D defines distances between locations, and the matrix F defines the flow (weight, expected value of the flow, number of connections) between pairs of facilities. The matrix B models a linear part of the assignment cost and in most cases is omitted. The solution to the QAP problem (also denoted as QAP(F,D)) can be defined in a permutation form $\pi = (\pi(1), ..., \pi(n))$ of the set of n elements (facilities). The purpose of the optimisation is to find the permutation π^* which minimizes the objective function:

$$f(\pi^*) = \min_{\pi\in S_n} \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} f_{\pi(i)\pi(j)} + \sum_{i=1}^{n} b_{\pi(i)i} \tag{1}$$

where S_n is a set of permutations of the natural numbers $\{1, ..., N\}$. If matrix D and F are symmetric i.e. the distance $d_{ij} = d_{ji}$ and the number of connections $f_{ij} = f_{ji}$ then this version of the QAP problem is called the *symmetric* QAP problem.

[1] https://github.com/NieTrawisz/Parameter-prediction-QAP

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

3 Related works

The performance of IEAs in combinatorial optimization problems such as the QAP is highly sensitive to parameter settings, including mutation probability, crossover rate, and population size. Appropriate parameter choices can significantly affect search efficiency, solution quality, and robustness. However, identifying effective configurations is challenging because optimal settings depend on problem structure and computational budget. This difficulty has motivated extensive research into automated parameter configuration methods.

Algorithm configuration can substantially boost the performance of even state-of-the-art metaheuristics [5]. Early systematic approaches include Design of Experiments (DoE) [25], which analyzes parameter influence by controlled variation. While analytically sound, DoE scales poorly, as the number of required experiments grows exponentially with the number of parameters, making it impractical for high-dimensional or expensive optimization problems.

To address scalability, several automated configuration frameworks have been proposed. Prominent examples include iRace, SMAC, SPO, and GGA.

- **iRace** [22] employs iterated racing with statistical tests to discard inferior configurations early, focusing evaluations on promising candidates while handling noise and conditional parameters.
- **SMAC** [19] extends Sequential Model-Based Optimization by using surrogate models and intensification strategies to guide the search in large, mixed parameter spaces.
- **Sequential Parameter Optimization (SPO)** [3] iteratively builds surrogate models, such as Gaussian processes or random forests, to balance exploration and exploitation, making it suitable for noisy or expensive evaluations.
- **Gender-based Genetic Algorithm (GGA)** [2] combines exploitation of strong configurations with preservation of diversity and modeling parameter dependencies via variable trees.

Despite their effectiveness, these methods share a common limitation: they require repeated evaluations for each new problem or instance. Even though they are sample-efficient, they remain inherently iterative and reactive, leading to substantial computational costs when problem characteristics change.

The optimal configuration of an algorithm strongly depends on the optimization problem at hand [26]. In general, mutation-based evolutionary algorithms perform well on simple problems, whereas crossover-based approaches are more effective in complex search spaces [12]. Statistical studies show that parameter settings have a significant impact on search behavior; however, strong parameter interactions make it difficult to isolate the marginal effect of any single parameter [28]. Recent work has nevertheless shown that carefully chosen mutation, crossover, and selection operators can substantially speed up reinforcement learning hyperparameter optimization [6]. For OneMAX problem instances, exact numerical parameter values (apart from operator choices) can be derived for

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

the $(1 + (\lambda, \lambda))$ genetic algorithm [13]. Despite these advances, general configuration strategies that apply across problem classes are still unknown.

Machine learning has been widely applied to improve the design and control of optimization algorithms. Early influential work focused on SAT problems, using learning for performance prediction, satisfiability estimation, and algorithm selection [29]. Parameter prediction has also been studied in SAT; however, existing approaches struggle when many configurations are similarly effective [4]. Beyond SAT, learning-based methods have been explored for combinatorial optimization problems such as the Quadratic Assignment Problem (QAP), including dynamic crossover selection within evolutionary algorithms [24] and, more recently, deep learning approaches that directly learn solution-improvement strategies, such as solution-aware transformers that capture higher-order structure [31]. While these studies highlight the potential of machine learning for guiding optimization algorithms, they primarily address operator selection or solution construction rather than the prediction of effective evolutionary algorithm parameter settings for complex problems such as the QAP. Moreover, to improve the quality of solutions for the QAP, the problem's theoretical properties can be exploited. The formula proposed in [9] for the conditional expectation in the QAP can be widely used in implementing approximation algorithms.

In summary, existing work demonstrates that evolutionary algorithm performance is highly dependent on parameter choices and that automated configuration methods can yield substantial improvements, albeit at significant computational cost and with limited generalization across problem instances. At the same time, machine learning approaches in combinatorial optimization have shown promise for exploiting instance-specific information, yet have rarely been applied to directly predict effective parameter settings for evolutionary algorithms on problems such as the QAP. Moreover, most prior studies implicitly assume a single optimal configuration per instance, despite empirical evidence that multiple near-optimal parameter settings often exist. These gaps motivate the present work, which investigates whether instance features can be used to predict improved IEA parameter configurations, whether treating multiple high-quality configurations as valid targets improves learning performance, and which modeling strategy is most robust in the presence of configuration ambiguity.

4 Research methodology

To investigate whether parameter prediction for QAP is feasible and to determine which model is most suitable, we first implemented the Island Evolutionary Algorithm (IEA). The implementation was optimized for GPU execution to reduce the average runtime [23,17]. Next, we tuned the algorithm configuration using SMAC [21], which allowed us to generate a dataset for training our models: regression-based, multiple-choice, and baseline approaches. Finally, we evaluated the performance of these models on their respective datasets and compared the IEA performance using the predicted parameters. The research methodology and individual steps are described in detail in the following sections.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

4.1 Island Evolutionary Algorithm

To evaluate parameter prediction capabilities, the IEA was selected as a benchmark algorithm for solving designated QAP instances. The IEA is a multipopulation variant of evolutionary algorithms in which several subpopulations (islands) evolve independently and periodically exchange individuals through migration [10]. This mechanism enhances population diversity and reduces the risk of premature convergence. The algorithm was implemented using the CUDA platform to exploit GPU parallelism [23]. Evolutionary operators and objective function evaluation were executed on the GPU, while migration followed a coarse-grained island model.

Control parameters of IEA The implementation incorporates categorical and numerical control parameters subject to tuning and prediction using machine learning. Let $D = \{\text{Roulette, LinearRank, NonlinearRank, Tournament, Threshold}\}$ denote the set of selection mechanisms. The control parameters are:

- $s \in S = \{NoElitism, Elitism, SteadyState\}$: succession type, determining how individuals are preserved between generations,
- $r \in D$: reproduction type, used for selecting parent solutions,
- $m \in D$: migration type, defining how individuals are selected for migration,
- $re \in D$: release type, specifying how individuals are removed and replaced by incoming migrants,
- $\mathcal{N} \in \{2, \ldots, 10\}$: number of islands. A single island makes migration-related parameters redundant, while larger values reduce subpopulation size and learning stability,
- $p_i \in [0, 1],\ i \in O$: probability distribution over pseudo-genetic operators, where $O = \{PMX, OX, CX, SWAP, SHIFT, REV, None\}$ and $\sum_{i \in O} p_i = 1$. The probabilities are normalized during tuning and define a categorical distribution used to sample operators.
- $m_f \in \{1, \ldots, 500\}$: migration frequency. Lower values increase interaction, while higher values approximate independent evolution; the upper bound reflects typical runtime,
- $m_{bs} \in [0, 0.4]$: migration fraction (replacing $2m_{bs}$ per island), values above 0.4 causes migration to dominate selection. It controls the strength of interaction between islands.

Conditionally active parameters were excluded to reduce sparsity and improve learnability, as they are observed only for a subset of configurations and provide limited training signals. The remaining parameters directly influence search dynamics and solution quality, making them suitable for prediction. Parameters related to ranking and threshold selection were fixed, while all others were used in tuning and training.

Implementation details In the experiments, the total population size was limited to $\mathcal{N} \cdot \lambda \leq 2000$, with the same number of individuals λ on each island.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

The maximum number of iterations without improvement was set to $I_{\max} = 50$. The algorithm terminates if the best solution is not improved for $I_{\max}$ consecutive iterations. The objective function is denoted by $\varphi(\cdot)$. The best solution found during the run is represented as π_{best}, with the objective value φ_{best}.

Algorithm 1 Island Evolution Algorithm

Step 1. For each from n populations (island).

(1. 1) Create randomly λ solutions. Set $I := 0$.
(1. 2) Evaluate the objective function value and sort the population members from best to worse.
(1. 3) Save the best solution in the population $\pi_{best} = arg\min\{\varphi(\pi^i)\}, i = 1, ..., \lambda$, $\varphi_{best} = \varphi(\pi_{best})$.

Step 2. For each from n islands (populations):

(2. 1) $I := I + 1$.
(2. 2) Randomly selected pseudo-genetic operator from set O.
(2. 3) Using selected reproduction type r select one (mutation) or two (crossover) parent solution. On the base selected pseudo-genetic operator create one (mutation) or two offspring (crossover).
(2. 4) Create a new population of size λ on the base selected succession type s. If $s \in \{NoElitism, Elitism\}$ then the number of offspring used to create a new population equals λ. Otherwise, if $s = \{SteadyState\}$ then only two best offspring are used.
(2. 5) Update the best solution $\pi_{best} = arg\min\{\varphi(\pi^i)\}, i = 1, ..., \lambda, \varphi_{best} = \varphi(\pi_{best})$.
(2. 6) If the best solution was not improved during I_{max} iterations, then return $(\pi_{best}, \varphi(\pi_{best}))$ and **stop** the algorithm.
(2. 7) *If* $(I \bmod m_f == 0)$ *goto* **Step 3** *else goto* **Step2**.

Step 3. Migration.

Migrate $m_{bs}\lambda$ individuals (solution) between islands using defined migration type m to select solution to migrate, and re method to release solution from population replaced by immigrants.

In our implementation, each island manages its own population of candidate solutions, evolving them via standard evolutionary operations such as selection, crossover, and mutation. Migration is performed at predefined intervals by exchanging a fraction of individuals between islands using a ring topology. The general scheme of the IEA is presented in Algorithm 1.

4.2 Dataset preparation

Dataset preparation began with the selection of QAP instances. We used 867 instances combined from the QAPLIB [7] and Instance Space Analysis (ISA)

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

datasets [11]. Effective parameter configurations for running the IEA were obtained using SMAC with 2,500 evaluations: 1,500 Latin hypercube samples (with two retries) followed by 1,000 Bayesian optimization evaluations. This process resulted in one parameter set for each of the 867 instances.

The tuning process produced target values for our models. We used ISA descriptors [11] and QAP autocorrelation metrics [8] as features. To expand the dataset, we adopted a dual representation by generating features for both the primal (F, D) and dual (D, F) forms, assuming similar effective configurations, thereby doubling the number of instances. For evaluation, the data were split into 90% training and 10% test sets, with an additional 10% of the training data reserved for validation in the regression-based model, ensuring systematic training and unbiased performance assessment.

4.3 Machine learning models

To investigate how machine learning can be used to predict suitable parameter settings for IEAs applied to the QAP, we compare three distinct modeling approaches. Their performance is evaluated using accuracy and coefficient of determination R^2 metrics for classification and regression tasks, respectively. These approaches are designed to address the inherent ambiguity of parameter tuning, where multiple configurations may yield comparable performance.

Baseline Model The baseline model represents a simplified parameter tuning strategy that ignores ambiguity among near-optimal solutions. For each QAP instance, the model learns to predict the single best-performing parameter configuration observed during training from QAP features. We employ AutoGluon [16] for both classification and regression variants of this task.

This approach effectively reduces parameter prediction to a standard classification and regression task with one target label per instance. While computationally simple and easy to interpret, the baseline model penalizes predictions that differ from the observed best configuration, even if alternative configurations yield comparable performance. As such, it serves as a reference point for evaluating the benefits of the more flexible regression-based and multiple-choice models.

Multiple-Choice Model The multiple-choice model explicitly accounts for the fact that several parameter configurations may lead to near-optimal performance. We select all parameter configurations whose results are at most 1% worse than the best found solution; notably, this inclusive threshold identifies an average of 222 valid configurations per instance that would otherwise be excluded by a strict single configuration choice. By treating this expanded set as equally valid, the model avoids the instability associated with overfitting to a single, potentially noisy, optimal target.

We employ two strategies to identify robust parameter sets: multi-label classification and regression. In the classification setting, AutoGluon [16] labels configurations within a 101% threshold of the optimum as 1 (otherwise 0). This

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

approach rewards the model for identifying any valid alternative rather than penalizing it for missing a single "best" setting. For regression, NGBoost [15] estimates the objective values of near-optimal configurations, with hyperparameters tuned via Optuna [1] TPE sampler with 100 trials. By allowing for multiple valid outputs, this framework prioritizes flexibility and robustness over a single rigid optimum, better reflecting the practical requirements of IEA tuning.

Regression-Based Model The regression-based approach treats parameter tuning as a supervised performance prediction task. By leveraging all evaluated configurations—rather than only near-optimal ones—the training data increases by nearly 190 times. This allows the model to map the full performance landscape, enabling effective parameter selection by minimizing the predicted objective values.

The input to the model consists of features extracted from the QAP instance combined with an encoding of the IEA parameter configuration. Categorical parameters are represented using one-hot encoding, while numerical features are scaled using adaptive normalization techniques. To facilitate stable neural network training and ensure scale invariance across instances, the target value is normalized. Specifically, a lower bound lb is computed using the Gilmore–Lawler bound [18], and the predicted value is defined as $\sqrt{\frac{\varphi_{best}-lb}{lb}}$. This transformation expresses performance as a relative deviation from the bound, improving comparability across instances of varying difficulty.

Hyperparameters—including network depth, activation functions, dropout rate, and learning rate—are optimized using Optuna [1] with the Tree-structured Parzen Estimator (TPE) sampler [27] over 800 trials. Model selection is performed based on the coefficient of determination R^2 evaluated on a validation dataset.

At inference time, the trained network evaluates all candidate parameter configurations for a given problem instance. The configuration minimizing the predicted normalized objective value is selected. This approach does not explicitly model the existence of multiple near-optimal parameter settings; rather, it relies on precise performance approximation to implicitly guide selection toward high-quality configurations.

4.4 Model performance comparison

To evaluate the practical effectiveness of the proposed models, we assess them in a closed-loop optimization setting. For each QAP instance, the models predict parameter configurations for the IEA, which are subsequently used to solve the instance. Each predicted configuration is executed 30 times with different random seeds to account for the stochastic variability of the algorithm. Performance is measured using the same normalized metric as in the regression-based model, namely $\frac{\varphi_{best}-lb}{lb}$, where lb is the Gilmore–Lawler lower bound [18]. The normalized performance is averaged over the 30 runs to obtain a single robustness-aware performance estimate for each instance and configuration.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

This evaluation protocol is applied to configurations predicted by the three proposed models, as well as to a reference configuration constructed from the most frequently selected categorical parameters and mean values of numerical parameters across the training set. This allows direct comparison between learned parameter selection strategies and a static, data-driven default configuration.

5 Results

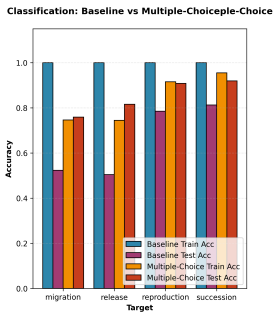
This section presents the experimental results addressing the three research questions: whether learned models outperform a standard configuration (RQ1), whether incorporating multiple near-optimal configurations improves predictive robustness (RQ2), and which modeling strategy provides the most reliable and effective parameter selection framework for QAP instances (RQ3). Recall that the overarching objective of this study is to determine whether machine learning can effectively predict parameter configurations for the Island Evolutionary Algorithm (IEA) when solving instances of the Quadratic Assignment Problem (QAP), and to identify the most suitable modeling strategy in the presence of solution ambiguity.

We evaluate three modeling strategies for predicting IEA parameter configurations on QAP instances: a baseline single-label model, a multiple-choice model that permits sets of near-optimal configurations, and a regression-based performance model, all trained on instance features. Evaluation focuses primarily on downstream optimization performance when the predicted configurations are integrated into the IEA, as predictive metrics alone do not reflect practical effectiveness. Accordingly, balanced accuracy and coefficient of determination (R^2) are reported for the baseline and multiple-choice models, while the regression model is assessed using R^2 on training, validation, and test sets as diagnostic measures. The key comparison is conducted in a closed-loop setting, where the IEA is executed with model-selected parameters, and performance is judged by the resulting optimization outcomes, directly capturing the real-world effectiveness of each approach.

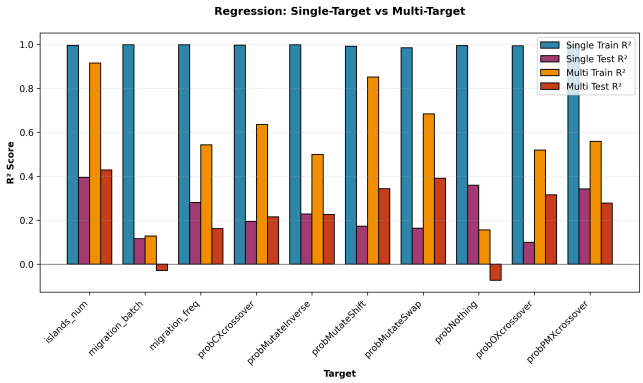
Model performance was evaluated on training and testing sets. While baseline models achieved near-perfect training accuracy and R^2 scores (≈ 1.0), their significantly lower test scores indicate overfitting (Fig. 1). Conversely, the multiple-choice model demonstrated superior generalization, outperforming the baseline on the test set despite lower training scores (Fig. 1a). Notable exceptions included regression targets such as migration batch and migration frequency, where baseline models performed better despite generally poor results from both architectures on the regression task (Fig. 1b). The results indicate that baseline models outperform others during training; however, the multiple-choice architectures demonstrate better generalization on the test set.

Benchmarking the regression model against prior iterations was challenging due to substantial performance discrepancies. The model achieved an R^2 of 0.16 on the training set, which decreased to -0.10 and -0.16 on the validation and test sets, respectively, indicating poor generalization. The architecture comprised

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22



(a) Baseline vs. multiple choice models classification accuracy.



(b) Baseline vs. multiple choice models regression R^2 scores.

Fig. 1: Comparative performance of baseline and multiple-choice models. Colors: baseline train (blue), baseline test (pink), multiple choice train (orange), multiple choice test (red).

three layers with 482, 207, and 221 neurons, using ReLU, tanh, and ReLU activation functions. Dropout rates were 0.04, 0.05, and 0.27, with a learning rate of 2×10^{-3}.The regression model exhibited poor generalization, achieving moderate training performance, but negative validation and test scores, despite using a three-layer architecture with ReLU and tanh activations.

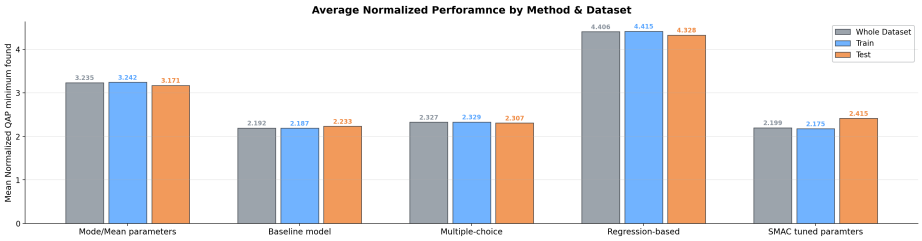


Fig. 2: Mean performance normalized using the Gilmore–Lawler lower bound [18], obtained with running algorithm using given set of parameters across whole QAP dataset, training set, and test set.

Performance comparison provides a rigorous evaluation of the models using the Wilcoxon paired test ($\alpha > 0.05$). Parameters optimized with SMAC [21] significantly outperformed configurations derived from the most common or mean values, with $\alpha = 1.1 \times 10^{-68}$. IEA runs with model-predicted parameters achieved performance comparable to directly tuned configurations and were statistically indistinguishable on the test set, with $\alpha = 7.34 \times 10^{-1}$ for the baseline

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

and $\alpha = 7.64 \times 10^{-1}$ for the multichoice model, indicating strong generalization. Furthermore, the baseline and multiple-choice models produced statistically similar results on both training and test sets, with $\alpha = 1.16 \times 10^{-1}$ and $\alpha = 5.35 \times 10^{-1}$, respectively. The only exception was the regression model, where predicted parameters underperformed relative to mode/mean-based configurations, with $\alpha = 5.29 \times 10^{-67}$. In summary, for the pairwise comparisons of the *baseline vs multichoice* and *multichoice vs tuned* models, there is no evidence to reject the null hypothesis. In the remaining comparisons, the null hypothesis can be rejected.

6 Discussion

This study evaluated learning-based parameter selection as a decision-making component within a closed optimization loop. The results show that, for the QAP and the considered IEA, instance features can be used to predict parameter configurations that achieve competitive performance on unseen instances. Predicted configurations consistently outperformed simple aggregate settings and approached the performance of configurations obtained via computationally expensive tuning, supporting the feasibility of supervised parameter prediction in this setting (RQ1).

A central aspect of this work was modeling parameter prediction as an inconsistent learning problem, where multiple configurations may yield near-optimal performance. The multiple-choice model improved generalization in predictive metrics by treating such configurations as equally valid. However, this did not translate into improved optimization performance compared to the single-label baseline. This suggests that, despite the existence of multiple near-optimal configurations, a single representative configuration is sufficient for effective parameter selection in the considered setting (RQ2).

Among the evaluated approaches, the baseline model achieved the best optimization performance, despite overfitting in predictive metrics. In contrast, the regression-based model performed poorly both in prediction and in downstream optimization. This indicates that accurate performance prediction is difficult in the presence of noise and performance plateaus, where many configurations yield similar outcomes. As a result, errors in predicted performance can lead to suboptimal parameter choices.

This observation has implications for automated algorithm configuration. Many approaches rely on performance prediction to guide parameter search [19,22,3], yet our results suggest that such strategies may be ineffective for problems with flat or noisy parameter-performance landscapes. In contrast, directly predicting good configurations appears more robust in this setting (RQ3).

7 Conclusions and future work

This work investigated machine learning for parameter prediction in IEAs applied to QAP instances. The results demonstrate that the learned models can

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

predict parameter configurations whose performance is comparable to that obtained through expensive tuning, and that they consistently outperform a fixed reference configuration.

Among the considered approaches, direct parameter prediction using a single-label model yielded the best optimization performance, while the multiple-choice model improved predictive generalization without improving downstream results. The regression-based approach, which relies on performance prediction, performed poorly, suggesting that accurately modeling performance landscapes is challenging in this domain. These findings indicate that direct parameter prediction can be an effective alternative to performance-based configuration methods, and that increased model complexity does not necessarily improve optimization outcomes. Furthermore, modeling parameter selection as an inconsistent learning problem did not yield practical benefits in this setting, despite improving predictive metrics.

Future work should evaluate the generality of these findings across other problem domains and algorithms. It would also be valuable to investigate alternative formulations of performance modeling, such as ranking-based approaches, and to develop novel algorithm configuration strategies that do not rely exclusively on explicit performance prediction models.

Acknowledgments. This research was supported by the Polish Ministry of Education and Science funds assigned to AGH University of Krakow, and further by program "Excellence initiative—research university" for the AGH University of Krakow as well as the ARTIQ project: UMO-2021/01/2/ST6/00004 and ARTIQ/0004/2021. We extend our gratitude to the Faculty of Drilling, Oil, and Gas in AGH University of Krakow Authorities for providing access to a computing cluster that significantly enhanced the quality of the calculations used in this article.

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

References

1. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework (2019), https://arxiv.org/abs/1907.10902
2. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. In: Gent, I.P. (ed.) Principles and Practice of Constraint Programming - CP 2009. pp. 142–157. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
3. Bartz-Beielstein, T., Lasarczyk, C., Preuss, M.: Sequential parameter optimization. In: 2005 IEEE Congress on Evolutionary Computation. vol. 1, pp. 773–780 Vol.1 (2005). https://doi.org/10.1109/CEC.2005.1554761
4. Beskyd, F., Surynek, P.: Parameter setting in sat solver using machine learning techniques. In: International Conference on Agents and Artificial Intelligence (2022), https://api.semanticscholar.org/CorpusID:246911645

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

5. Biedrzycki, R.: Revisiting cec 2022 ranking: A new ranking method and influence of parameter tuning. Swarm and Evolutionary Computation **89**, 101623 (2024). https://doi.org/https://doi.org/10.1016/j.swevo.2024.101623, https://www.sciencedirect.com/science/article/pii/S2210650224001615
6. Brzęk, B., Probierz, B., Kozak, J.: Exploration-driven genetic algorithms for hyperparameter optimisation in deep reinforcement learning. Applied Sciences (2076-3417) **15**(4) (2025)
7. Burkard, R., Çela, E., Karisch, S., Rendl, F., Anjos, M., Hahn, P.: Qaplib quadratic assignment problem library: Problem instances and solutions (2022), https://doi.org/10.7488/ds/3428
8. Chicano, F., Luque, G., Alba, E.: Autocorrelation measures for the quadratic assignment problem. Applied Mathematics Letters **25**(4), 698–705 (2012). https://doi.org/https://doi.org/10.1016/j.aml.2011.09.053, https://www.sciencedirect.com/science/article/pii/S0893965911004654
9. Chmiel, W.: Evolutionary algorithm using conditional expectation value for quadratic assignment problem. Swarm and Evolutionary Computation **46**, 1 – 27 (2019). https://doi.org/https://doi.org/10.1016/j.swevo.2019.01.004, http://www.sciencedirect.com/science/article/pii/S2210650217307150
10. Chmiel, W., Szwed, P.: Bees algorithm for the quadratic assignment problem on CUDA platform. In: Man-Machine interactions 4 : 4th International Conference on Man-Machine Interactions, ICMMI 2015. Poland (2015)
11. Christiansen, J., Smith-Miles, K.: Instance space analysis for the quadratic assignment problem (2025), https://arxiv.org/abs/2506.20172
12. Deb, K., Agrawal, S.: Understanding interactions among genetic algorithm parameters. In: Banzhaf, W., Reeves, C.R. (eds.) Proceedings of the Fifth Workshop on Foundations of Genetic Algorithms, Amsterdam, The Netherlands, September 24-28, 1998. pp. 265–286. Morgan Kaufmann (1998)
13. Doerr, B.: Optimal parameter settings for the $(1 + (\lambda, \lambda))$ genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. p. 1107–1114. GECCO '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2908812.2908885, https://doi.org/10.1145/2908812.2908885
14. Doerr, C., Wagner, M.: Sensitivity of parameter control mechanisms with respect to their initialization. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) Parallel Problem Solving from Nature – PPSN XV. pp. 360–372. Springer International Publishing, Cham (2018)
15. Duan, T., Anand, A., Ding, D.Y., Thai, K.K., Basu, S., Ng, A., Schuler, A.: NGBoost: Natural gradient boosting for probabilistic prediction. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 2690–2700. PMLR (13–18 Jul 2020), https://proceedings.mlr.press/v119/duan20a.html
16. Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M., Smola, A.: Autogluon-tabular: Robust and accurate automl for structured data. arXiv preprint arXiv:2003.06505 (2020)
17. Garland, M., Grand, S.L., Nickolls, J., Anderson, J., Hardwick, J., Morton, S., Phillips, E., Zhang, Y., Volkov, V.: Parallel computing experiences with cuda. IEEE Micro **28** (2008). https://doi.org/10.1109/MM.2008.57
18. Gilmore, P.C.: Optimal and suboptimal algorithms for the quadratic assignment problem. Journal of the Society for Industrial and Applied Mathematics **10**(2), 305–313 (1962). https://doi.org/10.1137/0110022, https://doi.org/10.1137/0110022

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22

19. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) Learning and Intelligent Optimization. pp. 507–523. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
20. Koopmans, T., Beckmann, M.J.: Assignment problems and the location of economic activities. Econometrica **25**(1) (1957)
21. Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., Hutter, F.: Smac3: A versatile bayesian optimization package for hyperparameter optimization. Journal of Machine Learning Research **23**(54), 1–9 (2022), http://jmlr.org/papers/v23/21-0888.html
22. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016). https://doi.org/https://doi.org/10.1016/j.orp.2016.09.002, https://www.sciencedirect.com/science/article/pii/S2214716015300270
23. Memeti, S., Pllana, S., Binotto, A., Kołodziej, J., Brandic, I.: Using meta-heuristics and machine learning for software optimization of parallel computing systems: a systematic literature review. Computing **101**, 893–936 (8 2019). https://doi.org/10.1007/s00607-018-0614-9
24. Mısır, M.: Algorithm selection on adaptive operator selection: A case study on genetic algorithms. In: Simos, D.E., Pardalos, P.M., Kotsireas, I.S. (eds.) Learning and Intelligent Optimization. pp. 237–251. Springer International Publishing, Cham (2021)
25. Mosayebi, M., Sodhi, M.: Tuning genetic algorithm parameters using design of experiments. pp. 1937–1944. Association for Computing Machinery, Inc (7 2020). https://doi.org/10.1145/3377929.3398136
26. Nannen, V., Smit, S.K., Eiben, A.E.: Costs and benefits of tuning parameters of evolutionary algorithms. In: Rudolph, G., Jansen, T., Beume, N., Lucas, S., Poloni, C. (eds.) Parallel Problem Solving from Nature – PPSN X. pp. 528–538. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
27. Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M.: Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. p. 533–541. GECCO '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3377930.3389817, https://doi.org/10.1145/3377930.3389817
28. Rojas, I., González, J., Pomares, H., Merelo, J.J., Castillo, P.A., Romero, G.: Statistical analysis of the main parameters involved in the design of a genetic algorithm (2002)
29. Shavit, H., Hoos, H.H.: Revisiting satzilla features in 2024. In: 27th International Conference on Theory and Applications of Satisfiability Testing (SAT 2024). pp. 27–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2024)
30. Smit, S.K., Eiben, A.E.: Comparing parameter tuning methods for evolutionary algorithms. In: 2009 IEEE congress on evolutionary computation. pp. 399–406. IEEE (2009)
31. Tan, Z., Mu, Y.: Learning solution-aware transformers for efficiently solving quadratic assignment problem (2024), https://arxiv.org/abs/2406.09899

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29909-3_22