MDO's Method for NeurIPS 2021 ML4CO's Configuration Task

Mengyuan Zhang
Decision Intelligence Lab
Alibaba DAMO Academy
Hangzhou, China
marvin.zmy@alibaba-inc.com

Abstract

In this report, we describe our approach to the configuration task of NeurIPS 2021 Machine Learning for Combinatorial Optimization (ML4CO) competition. The configuration task aims to search for good parameter configuration of the mixed-integer programming solver SCIP on three benchmark problems sets. To deal with the original high-dimensional configuration space, we rely on the expert knowledge and apply the functional analysis of variance (fANOVA) to narrow down the scope of the parameters being optimized. For each sub-group of parameters, we use the Bayesian optimization package SMAC3 for the task of parameter tuning. To exploit the power of SMAC3, we apply our black-box optimization package, Mindo Black-box Optimizer, to optimize the experiment design of parameter tuning process. We won the **2nd** place in the final leaderboard of configuration task.

1 Introduction

The Machine Learning for Combinatorial Optimization (ML4CO) NeurIPS 2021 competition[1] aims at exploring the possibilities to improve the state-of-the-art MIP solver SCIP[2] by utilizing general machine learning approaches given historical instances. The competition includes three challenges, each of which corresponding to a specific task closely related to the solving procedure of SCIP, and is exposed through OpenAI-gym APIs based on the Python library Ecole.[3]

In the configuration task, the participants need to search for a good parameterization for SCIP that yields the best possible performance while solving a set of mixed-integer linear programming (MILP) problems. Three problem benchmarks: an *item-placement* problems set, a *load-balancing* problems set, and an *anonymous* problems set are used in the challenge, with the training and validation set provided to the participants for preparing their submissions.[4] The performance metric is the primal-dual integral measuring the area between the curve of SCIP's primal bound and the curve of its dual bound over a duration of 15mins. Such a metric captures the quality of the solution as well as the speed of convergence. For each benchmark set, participants submit their policy of configuration setting, which are evaluated by the organizer by calculating the averaged primal-dual integral across all the instances. And the final score is determined based on the individual ranking among all the participants for each of the three benchmarks.

Similar to many other parameter tuning scenarios (e.g., hyper-parameter optimization for machine learning models[1]), the configuration task of MIP solver can be casted as solving a black-box

[1]https://www.ecole.ai/2021/ml4co-competition/
[2]https://www.scipopt.org/
[3]https://www.ecole.ai/
[4]https://github.com/ds4dm/ml4co-competition

optimization (BBO)[2] problem $\min_{x \in \Omega} M(x)$. Specifically, variable x corresponds to the solver's configuration and the objective $M(x)$ indicates the performance metric (e.g., the primal-dual integral). Many meta-heuristic methods including surrogate-based algorithms (e.g., [3, 4, 5]) and evolutionary algorithms (e.g., [6]) have been developed trying to efficiently search for the optimal solution using only zeroth-order information (objective function queries).

In this competition, we jointly apply the state-of-the-art algorithm configuration package SMAC3[7] and our MindOpt Black-Box Optimizer [5] for the parameter tuning process. In addition, some tricks are used to deal with the several challenges such as curse-of-dimensionality of search space, the heterogeneity of problem instances, as well as the high costs of evaluation runs. We won the **2nd** place in the final leaderboard. In the next section, we will describe our approach in more details.

2 Our Approach

As there are more than 2000 tunable SCIP parameters with different types (char, boolean, real, integer), simultaneously tuning all the parameters is inefficient, even impractical. Given some domain knowledge on the modules of SCIP solver (e.g., presolving, node-selection, branching, primal heuristic, etc.), we first pre-process the original configuration space by classifying the parameters according to the modules they are related to, and removing a set of the parameters that are irrelevant to the MILP benchmark problems considered in this task. For the remaining 300+ parameters, we further do the following three steps,

- We identify the conditional relationships among the parameters. For example, we extract those 'freq' parameters that are used to turn on/off a particular heuristic method as well as those 'priority' parameters that determine the orders according to which different heuristic methods are applied.
- We evaluate a set of randomly generated configurations and apply the *functional analysis of variance (fANOVA)* technique [8] to assess the relative importance of each individual parameter.
- We apply the *differential grouping* technique [9] to quantify the interaction between parameters and identify the separable and non-separable parameters (for real-type parameters).

After pre-processing the configuration space, we determine a sequence of parameter tuning sub-tasks, each of which targets on a group of parameters with moderate scale. We choose to use the open-sourced Bayesian optimization tool SMAC3 [7] for each tuning sub-task, as it has a well-designed intensification mechanism that evolves and updates the incumbent configuration without testing each candidate configuration on all the problem instances.

To exploit the power of SMAC3, several customized options on Bayesian optimization algorithm need to be carefully specified, including: (1) the *initial sampling method* which can be chosen between random search, Latin Hypercube design, and Sobol design, (2) the kernel model which can be chosen between Gaussian Process and random forests, and (3) the acquisition function which can be chosen among Expected Improvement (EI) (and its extensions LogEI, EIPS), probability of improvement (PI), Lower Confidence Bound (LCB) and Thompson Sampling (TS). In addition to the specifications of SMAC3, the following experiment design parameters are crucial for improving the efficiency of the parameter tuning under resource limitations.

- **Size of training instances set.** The size of training instances set should be properly determined for the joint consideration of tuning's time cost and the generalization of the obtained configuration.
- **Time limit for SCIP's solving process.** A shorter running time limit for SCIP (instead of the 15mins used in evaluation) can be used in parameter tuning to improve the time efficiency.
- **Number of SCIP calls in each sub-task.** The total number of SCIP calls allowed in a sub-task, i.e., the query budget of BBO, need to be properly set to balance the performance and the time cost of parameter tuning.

[5]The tool will be officially released soon this year.

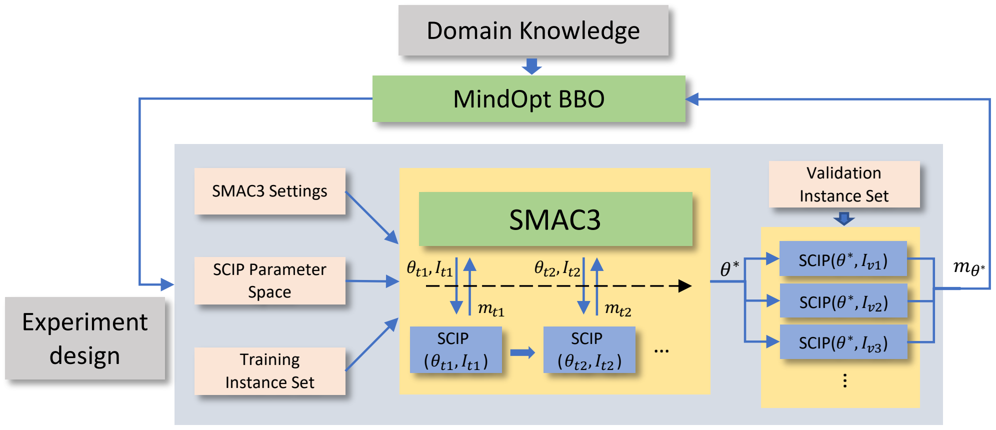

Figure 1: An illustration of our framework. The MindOpt BBO iteratively suggests new experiment design parameters (including the SMAC3 settings) relying on the observed tuning performance of previous experiment designs.

Since the type and distribution of benchmark problems are fixed and consistent between public training set and hidden testing set, we decide to optimize the SMAC3's specifications as well as the above-mentioned experiment design parameters for each benchmark set. To this end, we apply our MindOpt Black-Box Optimizer[6] to iteratively search for the optimal experiment settings as illustrated in Fig.1.

Finally, we introduce some of the tricks we used and experiences we gained in the competition.

- For each benchmark set, we firstly checked the SCIP's default parameter settings on "Presolve", "Heuristics", " Separating" and "Branching" plugin by setting each of them to "DEFAULT", "AGGRESSIVE", "FAST" and "OFF" mode alternatively.
- We learned the characteristics of a benchmark set as well as a particular parameter setting by un-muted the statistics and logs of SCIP during its solving process.
- For a large training instance set, we run multiple tuning experiments using different subset of training instances. The incumbent configuration of previous runs would be added into the initial points set of later runs for the purpose of warm-start.
- In the validation stage, we exploited the power of parallelization by evaluating a configuration over the validation instances using a cluster of machines, which greatly increased the evaluation efficiency.
- For the anonymous benchmark set, we divide the entire instances set into several subsets according to the instance features such as the number and type of variables and constraints. And separate parameter tunings are conducted for different subsets of problem instances.

3 Conclusion

We gained a lot of experiences on configuration optimization for SCIP solver by attending the NeurIPS 2021 ML4CO competition. We were satisfied with our final grades although we might be able to perform better if we had participated into the competition earlier. We believed that our relatively low ranking for the anonymous dataset is due to our negligence on the impact of the random seed (we used single seed during the tuning process, while the evaluation used 5 different seeds). In the future, we will keep exploring the technique of configuration optimization for solvers and develop our own configuration optimization tools.

[6]The optimizer has integrated several of our black-box algorithms such as ZO-BCD[10] and CobBO[11].

Acknowledgments and Disclosure of Funding

We would like to thank our team leader Dr. Wotao Yin, and our team members Tao Li, Yangbin Shen, Mengchang Wang for their great supports.

References

[1] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1), 2015.

[2] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. 2017.

[3] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Sam Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. In *Advances in Neural Information Processing Systems 33*, 2020.

[4] Alberto Nei Carvalho Costa and Giacomo Nannicini. Rbfopt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation*, 10:597–629, 2018.

[5] David Eriksson, Michael Pearce, Jacob R. Gardner, Ryan D. Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. In *Advances in Neural Information Processing Systems 32*, 2019.

[6] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.*, 11(1):1–18, 2003.

[7] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, René Sass, and Frank Hutter. SMAC3: A versatile bayesian optimization package for hyperparameter optimization, 2021.

[8] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, page I–754–I–762, 2014.

[9] Yuan Sun, Mohammad Nabi Omidvar, Michael Kirley, and Xiaodong Li. Adaptive threshold parameter estimation with recursive differential grouping for problem decomposition. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 889–896, 2018.

[10] HanQin Cai, Yuchen Lou, Daniel Mckenzie, and Wotao Yin. A zeroth-order block coordinate descent algorithm for huge-scale black-box optimization. In *ICML*, 2021.

[11] Jian Tan, Niv Nayman, Mengchang Wang, and Rong Jin. Cobbo: Coordinate backoff bayesian optimization. *CoRR*, abs/2101.05147, 2021.