ADVISER$^+$: Toward a Usable Web-based Algorithm Portfolio Deviser

Hoong Chuin Lau[1], Mustafa Mısır[2], Xiang Li[1], Lingxiao Jiang[1]

[1] Singapore Management University
School of Information Systems, 178902, Singapore
hclau@smu.edu.sg, xiang.li.2012@sis.smu.edu.sg, lxjiang@smu.edu.sg

[2] Nanjing University of Aeronautics and Astronautics
College of Computer Science and Technology, 211106, Nanjing, Jiangsu, China
mmisir@nuaa.edu.cn

Abstract

The present study offers a more user-friendly and parallelized version of a web-based algorithm portfolio generator, called ADVISER. ADVISER is a portfolio generation tool to deliver a group of configurations for a given set of algorithms targeting a particular problem. The resulting configurations are expected to be diverse such that each can perform well on a certain type of problem instances. One issue with ADVISER is that it performs portfolio generation on a single-core which results in long waiting times for the users. Besides that, it lacks of a reporting system with visualizations to tell more about the generated portfolios. ADVISER is extended as ADVISER$^+$ to overcome both of these issues while introducing new tuning based portfolio generation capabilities.

1 Introduction

Algorithm portfolios [7] have been proposed to build a portfolio of algorithms with varying characteristics in order to effectively solve different instances of a given problem. It achieves this either by choosing a subset of an existing set of algorithms and/or varying configurations of these algorithms. Hydra [23] and ISAC [10] are two successful portfolio methods operating via parameter configuration. After finding an effective portfolio, the next step is how to use it appropriately. Algorithm selection [12, 21] can be used to pick one or more algorithms from a resulting portfolio. OSCAR [18] is a framework that hybridizes the notions of algorithm portfolio and algorithm selection in an online manner for operator selection in a memetic algorithm. Another hybrid approach [5] utilizes parameter tuning based on design of experiments for online algorithm selection.

While algorithm portfolios can potentially be utilized by any designer who has access to a machine, it can be a challenge for novice users, since it is usually not straightforward to work with existing algorithm portfolio tools. One needs to have some background knowledge about how a particular tool works in order to use it effectively. Furthermore, learning how the tool works does not solve the problem entirely since it is essential to have an appropriate environment, i.e. installing a suite of softwares, to run such tools. For instance, LLAMA[1] [11] and auto-sklearn[2] [4] are very easy-to-use tools for algorithm selection and hyper-parameter tuning, yet there is still a barrier for whom without knowledge on the related topics or at least, certain softwares should be installed first. The second issue is on the requirement of high computational power to generate a portfolio. Especially, from a configuration perspective, portfolio generation can require substantial computational resource since evaluating a configuration can be expensive. For instance, an aforementioned portfolio generation method, i.e. Hydra [23], took 70 CPU days for the reported case study. Unavailability of such computational power limits the use of similar tools.

ADVISER[3] [17], i.e. an automated Algorithm portfolio DeVISER, was introduced to address the issue of usability. Since ADVISER runs on remote machines via a web browser, no software installation is needed. Besides that, its usage is straightforward and no prior knowledge on algorithm portfolios is

[1] `https://bitbucket.org/lkotthoff/llama`
[2] `https://github.com/automl/auto-sklearn`
[3] `http://research.larc.smu.edu.sg/adviser/`

required. Its design philosophy follows a web-based algorithm configuration tool, i.e. AutoParTune [4] [13].

In order to take ADVISER to the next level of usability, this paper introduces ADVISER$^+$ that offers enhanced features, both in the front-end and the back-end. For the front-end, a visualization module is provided that enables the user to understand how the portfolio has been constructed visually. The back-end is improved by integrating parallelization during training on multi-core processors for faster response time to the end-users.

In the following, Section 2 explains these added features. Experimental details of running ADVISER$^+$ on two scenarios with different parametric algorithms applied to varying problem instances are briefly discussed in Section 3. Section 4 provides a summary of future research plans.

2 ADVISER$^+$

ADVISER [17] as a portfolio generator yields a portfolio of algorithm-configuration pairs based on the performance of different algorithm configurations on a given set of training instances. ADVISER offers two major advantages compared to the existing portfolio generation approaches such as Hydra [23] and ISAC [10, 16]. First, ADVISER's web-based nature allows users to run it remotely without any implementation effort and computational resources on the users' side. Second, ADVISER's generic feature extraction approach eliminates the need of the instance features required for the known portfolio methods. In ADVISER, these features represent algorithm-configuration pairs. The latter advantage is particularly helpful for the users who have limited or no knowledge on the target problem domain considering the challenges of feature extraction [6]. ADVISER$^+$ extends the portfolio generation and configuration capabilities by introducing a basic portfolio generator (as provided in ADVISER) as well as a tuning-based portfolio generator.

The basic generator starts by the discretizing parameters of given algorithms with an equal step size like a grid search. Then, all possible parameter configurations using these discretized parameter values run on each training instance. The results, i.e. solution quality, found by all the algorithm-configuration pairs, are used as features. These features are then utilised to cluster these algorithm-configuration pairs after they are being normalized by applying k-means clustering [9]. Afterwords, the best algorithm-configuration pair is picked to build a diverse portfolio. Regarding clustering, similar to ADVISER, ADVISER$^+$ operates without predefined instance features. Despite the benefit of automated feature extraction in ADVISER$^+$, using a grid search can be considered a computationally expensive option for providing those features. However, since the number of configurations to be tested is bounded by a parameter in ADVISER$^+$, the possible computational overhead can be kept low.

For the tuning-based portfolio generator, a set of features are initially extracted in the same way done for the basic generator. Yet, the resulting features are used to characterize the training instances rather than the algorithm-configuration pairs. Thus, the number of features representing each instance is equal to the number of the algorithm-configuration pairs tried. The follow-up portfolio generation steps resemble to ISAC [10] with a more recent parameter tuner. It should be noted that ISAC additionally has an algorithm selection component [1] differently than ADVISER$^+$ which purely focuses on portfolio generation. As in ISAC, the instance features are used to cluster the training instances. Considering that the primary motivation of portfolio generation is diversity, each instance cluster is independently targeted to deliver configurations for the existing algorithms. Thus, a tuning tool is separately applied to the instances from each cluster so that a single configuration for each algorithm is returned. A state-of-the-art, F-Race[5] [2] based parameter tuning method, namely the Post Selection tuner [24] is used as the tuning tool, which makes ADVISER$^+$ comparable to the best tuning methods. After the tuning process, all the unique algorithm-configurations are kept for the portfolio.

For the both portfolio generators, the portfolio size is directly given by the users depending on their available computational resources, e.g. the number of CPU cores. In that respect, by setting a portfolio

[4] `http://research.larc.smu.edu.sg/autopartune/`

[5] from the irace (iterated racing) algorithm configuration package [14]: `http://iridia.ulb.ac.be/irace/`

size of 1 when a single target algorithm is given, the portfolio generators can be used to perform the standard parameter configuration/tuning.

Beside having an additional portfolio generator, the major differences of ADVISER$^+$ to ADVISER are parallelization used in portfolio generation and visualization of the output portfolio. In the following, we will describe these two features in detail.

2.1 Visualization

After the algorithm portfolio has been generated, an advanced algorithm designer user may be interested in knowing how the portfolio has been derived as well. In addition, s/he may be interested in knowing the scenarios under which an algorithm in the portfolio works well and vice versa. To meet these requirement, we propose that the clusters derived by the basic portfolio generator be explicitly visualized to show the similarity and dissimilarity among the algorithm configurations.

It is also interesting to understand how similar each algorithm configurations are among each other with respect to their performance on the training instances. For this purpose, the best and worst performing instances for each cluster are reported so that the user can learn which algorithm configuration in the portfolio should (or should not) be used when there is a new problem instance. For each target algorithm configuration that runs on each instance, there are several interesting features that can be displayed to the user (such as the quality of solution, running time and cluster number). These information should be displayed in one graph so the user could see an overview of the interaction among algorithm configurations.

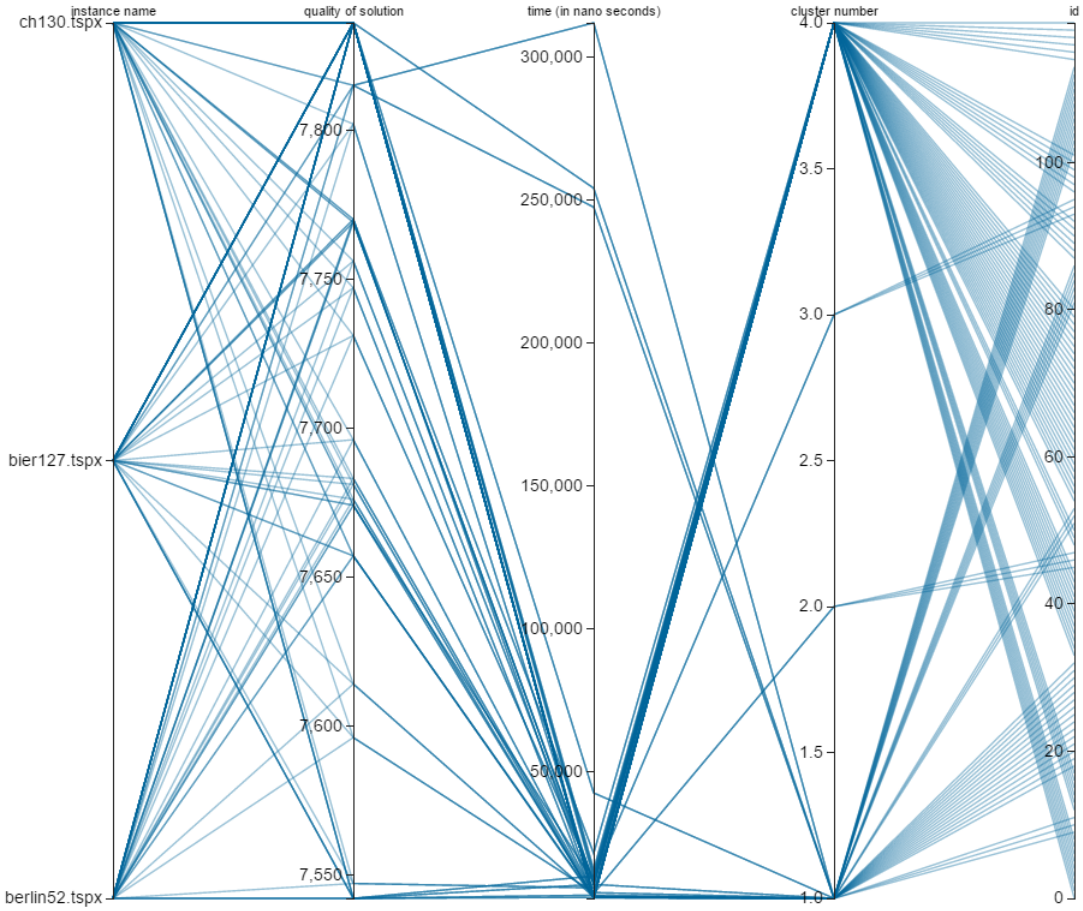


Figure 1: An example parallel coordinate graph

Traditional scatter plot to reveal relationship among variables does not work very well on higher dimensions. In ADVISER$^+$, we make use of the parallel coordinate graph instead, as illustrated in Figure

1. The Parallel coordinate graph is able to show high dimensional data without compromising on the information. For computing the visual representation, parallel coordinate graph has a low complexity compared to traditional scatter plot (O(N) vs O(N^2) where N=number of variables). The parallel coordinate graph also provides the opportunity for further expansion of the visualization as it can be applied to any number of dimensions while not dependent on one single dimension. In the graph, each line represents the target algorithm running on an instance with the specified configuration. The graph shows the execution time, quality of solutions and the cluster number. It provides an overview of the instances, target algorithms and intermediate running results.

2.2 Parallelization

ADVISER$^+$ allows multiple configurations to run multiple test instances in parallel. This has considerably increased the performance of ADVISER, and made the system capable of supporting multiple users simultaneously.

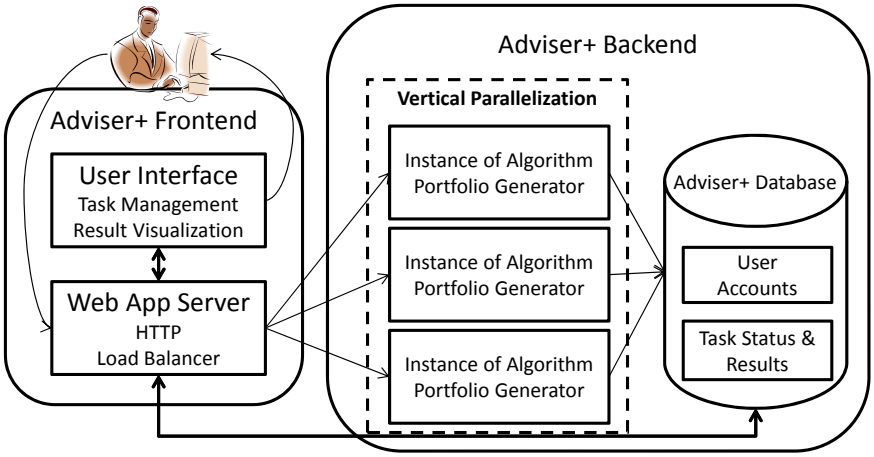


Figure 2: Vertical Parallelization for the Adviser$^+$ Web Application

ADVISER$^+$ employs two levels of parallelization. The first is called *vertical parallelization*. As shown in Figure 2, it utilizes a load balancer together with the http web server to create multiple instances of the algorithm portfolio generator, each of which may be run on a different machine, so that tasks from multiple users may be executed simultaneously without affecting each other. All instances of the generator store their outputs into a common database. Users can via the web user interface check the status and results of their tasks. Different instances of the generator, and the ADVISER$^+$ frontend and database, can in fact all be on the same machine, depending on the machine's capacity and the settings in the load balancer adjustable by the administrator for ADVISER$^+$.

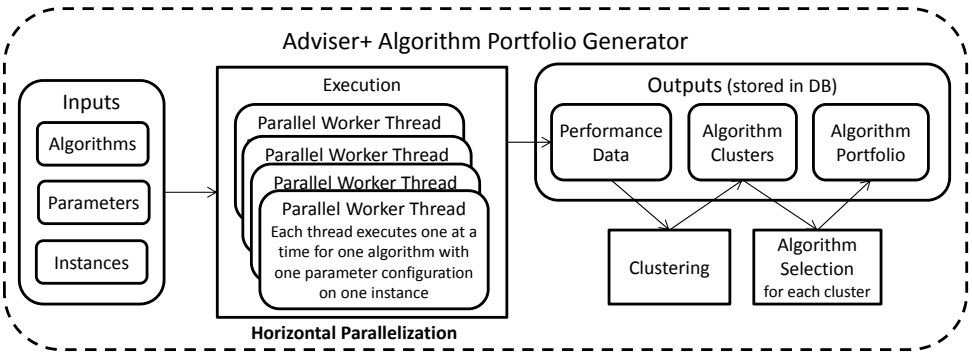


Figure 3: Horizontal Parallelization in Each Instance of Portfolio Generator

The second level called *horizontal parallelization* is within each instance of the algorithm portfolio

generator in ADVISER$^+$. As shown in Figure 3, when the generator takes the inputs from a user's task, it spawns a number of threads based on the available number of hyperthreads or cores or CPUs in the machine, aiming to fully utilize but not overwhelm the computing power of the machine. Each thread executes one algorithm configuration on one instance. The generator will spawn a new thread when a previous thread completes and there are still more configurations or instances to execute. Although the threads may affect each other's performance a bit by competing for the CPUs and memory, such parallelization helps to reduce the completion time for each task significantly.

2.3 Usage

ADVISER$^+$ can be accessed from `http://research.larc.smu.edu.sg/adviserplus/`. The input to the system is exactly the same as ADVISER. It includes the algorithms and their respective configuration space, as well as a set of training instances and the maximum portfolio size (K). All the algorithms should be in the form of .exe, accepting parameters as follows:

```
algorithm.exe -I instance_file -S seed ...  OtherParameters
```

A parameter space file should also be provided. In this file, each parameter should be specified with a parameter name (e.g. `INITIAL_TEMPERATURE`), a parameter argument (e.g. `"-T"`), a parameter type (`i`: integer, `r`: continuous, `c`: categorical) and its configuration space, illustrated as follows:

```
INITIAL_TEMPERATURE "-T" i [4000, 6500]
```

3 Case Study

For evaluating ADVISER$^+$, two test domains including the Quadratic Assignment Problem (QAP) and Traveling Salesman Problem (TSP) are used. For the QAP, a hybrid simulated annealing-tabu search meta-heuristic (SA-TS) [19] is used as the target algorithm, while an iterated local search (ILS) implementation is used for the TSP. Table 1 presents all the parameter configuration details.

Table 1: Configuration spaces

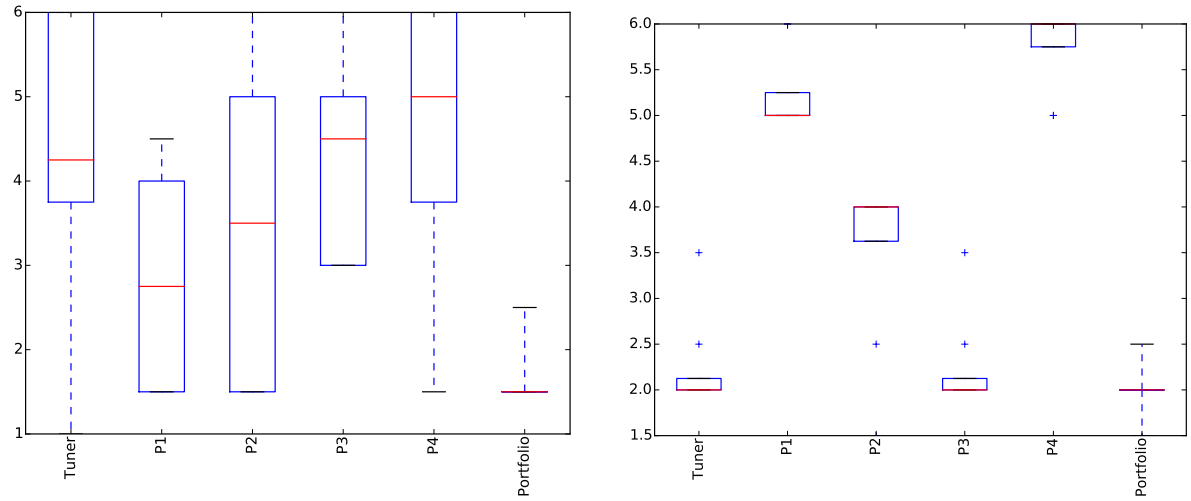
	Method	Type	Parameter	Range
QAP	SA-TS	Integer	Initial Temperature (T)	[4000, 6500]
		Continuous	Cooling Factor (C)	[0.85, 0.95]
		Integer	Tabu List Length (L)	[5, 10]
TSP	ILS	Integer	Perturbation Strength (P)	[1,10]
		Integer	Better Acceptance Criteria (B)	[0,1]
		Categorical	Non-improving Tolerance (N)	[1,2]
		Categorical	OptChoose (O)	[3,4]

For each problem, 22 instances are considered, all from QAPLIB [3] and TSPLIB [20]: 14 of those instances are used for training, including portfolio generation and separate tuning, and the remaining 8 instances are utilized for testing. Each configuration returned is run on each instance for 10 times due to the stochastic nature of both SA-TS and ILS. The portfolio size is set to 5, meaning that the number of configurations in a resulting portfolio can be 5 at most.

Table 2 illustrates both the configuration portfolio derived and the single configuration determined by the aforementioned parameter tuner on each problem. Each portfolio is composed of 4 configurations. Thus, the resulting portfolios require 4 CPU cores to run each of its constituent algorithm-configuration pairs.

Table 2: Portfolios suggested by ADVISER^{+} for each problem domain (*-Tuner indicates the single configuration found by the tuner when it is applied to the complete training set)

	Method	Portfolios of Configurations
QAP	SA-TS	-T 5300 -C 0.95 -L 7 -T 5400 -C 0.94 -L 7 -T 6000 -C 0.90 -L 5 -T 4000 -C 0.93 -L 9
	SA-TS-Tuner	-T 5100 -C 0.89 -L 8
TSP	ILS	-P 4 -B 1 -N 2 -O 3 -P 2 -B 1 -N 1 -O 3 -P 1 -B 1 -N 2 -O 3 -P 3 -B 1 -N 1 -O 4
	ILS-Tuner	-P 1 -B 1 -N 2 -O 3



(a) QAP (b) TSP

Figure 4: Average ranks on the test instances (P1, P2, P3 and P4 are the constituent configurations of the resulting portfolios as in Table 2)

Figure 4 presents the results in terms of average ranks. Pure parameter tuning is compared against the portfolio and each configuration in the portfolio, P1 $\rightarrow$ P4. The existing portfolio generation methods aren't considered for comparison since they require instance features to be able run, as discussed earlier. For the QAP, the performance of the portfolio is significantly better than the pure tuning case (Tuner) in terms of solution quality on the test instances. Considering the performance of each individual configuration in the portfolio, P1 and P2 return the best solutions for 3 instances, P4 comes up with the best solution only on one instance. The best solution for the last instance is provided by Tuner. For the average rank performance, SA-TS-Tuner outperforms only P4, yet not significantly.

For the TSP, the results are rather different than the QAP case. ILS-Tuner delivers the same average rank performance compared to one of the configurations in the portfolio, i.e. P3. ILS-Tuner and P3 achieve the best performance compared to the other configurations in the portfolio. Yet, the portfolio itself is able to surpass ILS-Tuner, mainly by the help of P2 that finds the best solutions on two instances. Still, there is no statistical performance difference. One possible reason behind the high performance of ILS-Tuner is related to the configuration space. The number of configuration for ILS is limited to only 80 configurations.

4 Conclusion

ADVISER$^+$ is an enhancement of ADVISER by introducing a useful reporting system with various visualization tools, and improving the response time for performing portfolio generation processes across multiple cores/CPUs. For improving ADVISER$^+$ further, integrating other existing portfolio generation related tools will need to be considered. The system will be extended as a web-service such that it can be reached directly from different programming platforms, like AzureML[6]. Related to that, online and interactive capabilities will be added. Additionally, a parameter importance module, like fANOVA [8], will be integrated to determine which parameters of a given algorithm matters most. Besides that, a feature generation through deep learning [15] option will be provided as an alternative to the grid search. Finally, algorithm scheduling [22] will be incorporated to be able to efficiently use algorithm portfolios even on a single core.

References

[1] Carlos Ansótegui, Joel Gabàs, Yuri Malitsky, and Meinolf Sellmann. MaxSAT by improved instance-specific algorithm configuration. *Artificial Intelligence*, 235:26–39, 2016.

[2] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-Race and iterated F-Race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer, 2010.

[3] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. QAPLIB–a quadratic assignment problem library. *Journal of Global optimization*, 10(4):391–403, 1997.

[4] M. Feurer, A. Klein, K. Eggensperger, J.T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. 2015.

[5] A. Gunawan, H.C. Lau, and M. Mısır. Designing and comparing multiple portfolios of parameter configurations for online algorithm selection. In *Proceedings of the 10th Learning and Intelligent OptimizatioN Conference (LION)*, volume 10079 of *LNCS*, pages 91–106, Naples, Italy, 2016.

[6] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[7] B.A. Huberman, R.M. Lukose, and T. Hogg. An economics approach to hard computational problems. *Science*, 275(5296):51, 1997.

[8] F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proceedings of the 31st International Conference on Machine Learning*, pages 754–762, 2014.

[9] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[10] S. Kadioglu, Y. Malitsky, M. Sellmann, and K. Tierney. ISAC–instance-specific algorithm configuration. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*, pages 751–756, 2010.

[11] L. Kotthoff. LLAMA: leveraging learning to automatically manage algorithms. Technical Report arXiv:1306.1031, 2013.

[12] L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 35(3):48–60, 2014.

[6] `https://studio.azureml.net/`

[13] Lindawati, Z. Yuan, H.C. Lau, and F. Zhu. Automated parameter tuning framework for heterogeneous and large instances: Case study in quadratic assignment problem. In *Proceedings of the 7th Learning and Intelligent OptimizatioN Conference (LION)*, volume 7997 of *LNCS*, pages 423–437. Springer, 2013.

[14] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[15] Andrea Loreggia, Yuri Malitsky, Horst Samulowitz, and Vijay A Saraswat. Deep learning for algorithm portfolios. In *Proceedings of the 13th Conference on Artificial Intelligence (AAAI)*, pages 1280–1286, 2016.

[16] Y. Malitsky and M. Sellmann. Instance-specific algorithm configuration as a method for non-model-based portfolio generation. In *Proceedings of the 9th International Conference on Integration of AI and OR Techniques in Contraint Programming for Combinatorial Optimzation Problems (CPAIOR)*, pages 244–259. Springer, 2012.

[17] M. Mısır, S.D. Handoko, and H.C. Lau. ADVISER: A web-based algorithm portfolio deviser. In Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eleonore Marmion, editors, *Learning and Intelligent Optimization*, volume 8994 of *LNCS*, pages 23–28. Springer, 2015.

[18] M. Mısır, S.D. Handoko, and H.C. Lau. OSCAR: Online selection of algorithm portfolios with case study on memetic algorithms. In Clarisse Dhaenens, Laetitia Jourdan, and Marie-Eleonore Marmion, editors, *Learning and Intelligent Optimization*, volume 8994 of *LNCS*, pages 59–73. Springer, 2015.

[19] K.M. Ng, A. Gunawan, and K.L. Poh. A hybrid algorithm for the quadratic assignment problem. In *Proceedings of international Conference on Scientific Computing*, Nevada, USA, 2008.

[20] G. Reinelt. Tspliba traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[21] J.R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.

[22] M. Streeter and S.F. Smith. New techniques for algorithm portfolio design. *arXiv preprint arXiv:1206.3286*, 2012.

[23] L. Xu, H.H. Hoos, and K. Leyton-Brown. Hydra: Automatically configuring algorithms for portfolio-based selection. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 210–216, 2010.

[24] Z. Yuan, T. Stutzle, M.A. Montes de Oca, H.C. Lau, and M. Birattari. An analysis of post-selection in automatic configuration. In *Proceeding of the 15th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)*, pages 1557–1564. ACM, 2013.