Automatic algorithm configuration: instance-specific or not?

Leyman, P.; Hoos, H.H.

Automatic algorithm configuration: Instance-specific or not?

Pieter Leyman
Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands
CODeS, Department of Computer Science, KU Leuven Kulak, Belgium
e-mail: p.leyman@liacs.leidenuniv.nl, pieter.leyman@kuleuven.be

Holger H. Hoos
Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands
Department of Computer Science, University of British Columbia, Canada
e-mail: hh@liacs.nl

Assume that you want to solve a rather difficult, say NP-hard, problem. You could then design a new metaheuristic algorithm, or use (parts of) an existing implementation, with operators or components tailored to the problem you wish to solve. Typically, these operators would be associated with a multitude of, often hard-coded, design choices. Additionally, most if not all metaheuristic algorithms contain parameters (e.g., the population size in a genetic algorithm), which can have major impact on the performance obtained for specific types of problem instances. It is well known that finding good parameter settings can be quite difficult, even for experts, and becomes even harder when additional design choices, such as alternative or optional components have to be made. However, these configuration tasks can be solved automatically, using general-purpose algorithm configuration procedures (see e.g., SMAC, ParamILS, irace).

Programming by Optimization (PbO) [3] states that it is crucial to avoid premature commitment, which can be achieved by incorporating alternatives for each of an algorithm's components, and by employing a (large) number of algorithm parameters to determine which alternative is or should be used. The aforementioned automatic algorithm configuration can then be employed to determine the best combination of components, given a training dataset. Note that algorithms typically are (and should be) configured on a training set, which is different from the evaluation set used to determine the performance of said configuration.

In the context of PbO, we propose an algorithm framework, which exposes all design choices and allows for the conversion to a highly parametrized algorithm (e.g., [4]). Instead of configuring this algorithm on a training dataset and subsequently employing the same configuration to evaluate performance for all instances in an evaluation set, however, we aim to find a configuration model based on instance characteristics. Since it was shown that the largest performance improvement for an algorithm typically comes from tuning 3-4 algorithm parameters [2], and that configuration landscapes are often unimodal and convex [5], we believe the focus of our suggested configuration model can be limited to these most important algorithm parameters.

Specifically, we configure an algorithm with automatic algorithm configuration, and then apply ablation analysis [2] to determine the algorithm parameters with the greatest impact on performance. Afterward, we build an instance-specific model based on this smaller set of algorithm parameters, such that different algorithm configurations can be proposed based on the data parameters of the test instances used. This way, we can obtain a mapping of instance parameters to algorithm parameters, such that different test instances may require different algorithm configurations. A particularly interesting question in this context is then whether our proposed approach would outperform a "standard" automated algorithm configuration (i.e. independent of data parameters), when applied on an evaluation set.

As a possible application, we are currently working on the capacitated vehicle routing problem (CVRP), since this is a well-studied problem in logistics. Recently, [1] proposed a ruin-and-recreate algorithm, which outperformed earlier state-of-the-art algorithms for both the CVRP and several problem extensions. As a result, we believe the algorithm of [1] makes for an excellent application to test our framework with.

Acknowledgements

Pieter Leyman is a Postdoctoral Fellow of the Research Foundation - Flanders.

References

[1] Christiaens, J. and Vanden Berghe, G. (2019). Slack induction by string removals for vehicle routing problems. *Transportation Science*, to appear.

[2] Fawcett, C. and Hoos, H.H. (2016). Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22: 431-458.

[3] Hoos, H.H. (2012). Programming by optimization (2016). *Communications of the ACM*, 55(2): 70-80.

[4] Luo, C., Hoos, H.H., Cai, S., Lin, Q., Zhang, H. and Zhang, D. (2019). Local search with efficient automatic configuration for minimum vertex cover. *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19)*, 1297-1304.

[5] Pushak, Y. and Hoos, H.H. (2018). Algorithm Configuration Landscapes: More Benign than Expected? *Proceedings of the 15th International Conference on Parallel Problem Solving from Nature (PPSN-18)*, 271-283.