From Grammars to Parameters: Automatic Design of Iterated Greedy Algorithms

Franco Mascia, Manuel López-Ibáñez, Jérémie Dubois-Lacoste and Thomas Stützle

IRIDIA, CoDE, Université Libre de Bruxelles

{fmascia,manuel.lopez-ibanez,jdubois,stuetzle}@ulb.ac.be

Abstract

Recent advances in automatic algorithm configuration make possible to configure flexible algorithmic frameworks in order to fine-tune them for particular problems. This is often done by using automatic methods to set the values of algorithm parameters. A rather different approach instantiates possible algorithms by repeated application of the rules defined by a context-free grammar. An instantiation of the grammar is represented either explicitly by a derivation tree or implicitly by numerical codons, as done in grammatical evolution.

In this work, we show how the process of instantiating such a grammar can be described in terms of parameters. In particular, we show how a grammar that generates iterated greedy (IG) algorithms can be represented in terms of parameters. We compare the quality of the IG algorithms generated by an automatic configuration tool using the parametric representation versus using the codon-based representation of grammatical evolution. The same experiments are repeated for two problems: the permutation flow-shop problem with weighted tardiness minimization and the one-dimensional binpacking problem. In our experiments, the parametric approach leads to significantly better IG algorithms.

Keywords: automatic algorithm configuration, grammatical evolution, iterated greedy, permutation flow-shop problem, binpacking

Despite (or perhaps because) the large number and variety of metaheuristics available for solving hard optimization problems, algorithm designers still rely on their experience and intuition when tackling a specific problem. Automatic algorithm configuration methods help designers to explore a larger number of algorithm designs than it was previously feasible. In addition, automatic methods rely less on human intuition, and, thus, they consider combinations of algorithmic components and parameter settings that would have been discarded by a human designer as supposedly uninteresting.

We identify two conceptually different approaches for automatic algorithm design. *Top-down* approaches consider a flexible framework of fully-defined algorithmic components. These components may be enabled, disabled, or combined with other components by setting appropriate parameters. Algorithm configuration methods are then used for automatically finding the best parameter configuration for a specific problem. Such an approach has been used to automatically design state-of-the-art SAT solvers [2], and multi-objective ant colony optimization algorithms [4].

By contrast, in *bottom-up* approaches, the algorithmic components are loosely defined, and the algorithm is assembled from very simple operations that can be combined with a higher degree of flexibility. Two recent works follow such a bottom-up approach. Vázquez-Rodríguez and Ochoa [5] automatically generate an initial order for the NEH algorithm, a well-known constructive heuristic for the PFSP, by using genetic programming. More recently, Burke et al. [1] automatically generate iterated greedy (IG) algorithms for the one-dimensional bin packing problem using grammatical evolution (GE). Both works instantiate algorithms bottom-up from a context-free grammar.

In this paper, we show that grammars can be also represented in terms of a parametric space, using categorical, numerical and conditional parameters. Such parametric representation allows to exploit the abilities of well-known automatic configuration tools used in top-down approaches. We test our proposed approach on two different problems: the permutation flow-shop problem (PFSP) and the one-dimensional bin packing problem (ODBP). For the PFSP, we devise a grammar that generates IG algorithms, whereas for the ODBP, we consider the grammar proposed by Burke et al. [1], for the sake of comparison.

We compare four methods: randomly generating IG algorithms, grammatical evolution, an automatic configuration tool (`irace` [3]) using the codon-based representation of GE, and `irace`

using our proposed parametric representation. Our experimental results clearly indicate that `irace` produces better IG algorithms than random generation or grammatical evolution. Moreover, `irace` using a parametric representation generates much better IG algorithms than using the codon-based representation in the two scenarios considered. This indicates that the parametric representation can help to avoid disadvantages of grammatical evolution. Finally, our approach is not limited to `irace` and it can be applied using other automatic configuration tools, as long as they can handle categorical and conditional parameters. Further work should examine for which kind of grammars our approach becomes prohibitively expensive and other representations might be more appropriate. Nonetheless, the grammar used in this work is similar in this respect to others that can be found in the literature for the generation of heuristic algorithms.

References

[1] Burke, E.K., Hyde, M.R., Kendall, G.: Grammatical evolution of local search heuristics. IEEE Transactions on Evolutionary Computation 16(7), 406–417 (2012)

[2] KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. In: Boutilier, C. (ed.) Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09). pp. 517–524. AAAI Press, Menlo Park, CA (2009)

[3] López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)

[4] López-Ibáñez, M., Stützle, T.: The automatic design of multi-objective ant colony optimization algorithms. IEEE Transactions on Evolutionary Computation 16(6), 861–875 (2012)

[5] Vázquez-Rodríguez, J.A., Ochoa, G.: On the automatic discovery of variants of the NEH procedure for flow shop scheduling using genetic programming. Journal of the Operational Research Society 62(2), 381–396 (2010)