Detailed Related Work

Chuan Luo[1,2], Holger H. Hoos[2], and Shaowei Cai[3]

[1] Microsoft Research, China
[2] Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands
[3] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China
chuan.luo@microsoft.com, hh@liacs.nl, caisw@ios.ac.cn

1 Related Work

In this document, we give a brief overview of related work, covering the key concepts of configuration checking, programming by optimisation and automated algorithm configuration, which form the basis of *PbO-CCSAT*.

1.1 Configuration Checking

It is well known that the performance of stochastic local search can be seriously degraded by stagnation – situations in which a limited set of candidate solutions is frequently revisited [31]. In order to address this issue, recently, a novel mechanism called configuration checking (CC) has been proposed. Conceptually similar to tabu search [11, 12], the main idea of CC is to prevent SLS solvers from visiting a candidate solution whose context has not changed since it was last visited. CC was first proposed to boost the performance of SLS solvers for the minimum vertex cover problem [8, 7], and subsequently has been shown to be effective in solving a variety of combinatorial problems, including SAT [6, 29, 27], maximum satisfiability [28, 26], maximum clique [35], set cover [10], dominating set [36] and combinatorial auctions [38].

In the context of SAT solving, there are two variants of CC: neighbouring-variables-based configuration checking (NVCC) [7] and clause-states-based configuration checking (CSCC) [27]. These two CC variants are based on different concepts of context for a variable. NVCC defines the context based on neighbouring variables of the corresponding variable, while CSCC defines the context based on the states of clauses where the corresponding variable appears. Recently, a hierarchical combination of these two CC variants with an aspiration mechanism [5] has given rise to *DCCASat* [29], a novel SLS solver that achieves excellent performance on phase-transition random k-SAT instances and several classes of structured SAT instances.

1.2 Programming by Optimisation

The key idea behind programming by optimisation (PbO) is to avoid premature commitment to design choices, especially in early stages of algorithm design, and instead to seek and maintain alternatives for peformance-critical components; the resulting flexibility is later exploited by automatically making design choices in a way that optimises performance for specific classes of inputs [13]. This stands in contrast to traditional software design, which tends to eliminate choices early in the process, based on limited exploration and informal experimentation.

Applied to existing solver architectures, following the PbO paradigm usually involves exposing all design choices as configurable parameters, actively seeking design alternatives for key components, and configuring the resulting flexible solver framework using a state-of-the-art general-purpose automatic configuration procedure that are based on advanced automated optimisation and machine learning techniques. PbO-based solver design has been demonstrated to yield excellent results on a broad range of prominent NP-hard problems, including SAT [14, 20, 33, 21], mixed integer programming [15], AI planning [34], classification problems [32, 22] and minimum vertex cover [**?**].

An early application of PbO to the design of SLS-based solvers for SAT integrated a broad range of high-performance SLS algorithms for SAT into a single, highly configurable framework, with an emphasis on novel hybrids between previously distinct methods; the resulting *SATenstein* framework was shown to perform well on a number of well-known SAT benchmarks, ranging from uniform random 3-SAT to industrial SAT-encoded factoring and software verification instances [20, 21].

1.3 Automated Algorithm Configuration

Many algorithms have parameters whose settings greatly affect performance; this especially holds for heuristic algorithms for solving challenging combinatorial problems, including SAT [17]. Because finding performance-optimising values of these parameters can be difficult and tedious, in recent years, there has been a growing body of work on automatic procedures for determining performance-optimising parameter configurations. This has lead to a number of high-performance, general-purpose automated algorithm configuration procedures, including *Iterated F-RACE* [3, 25, 9], *GGA* [2], *GGA++* [1], *ParamILS* [17] and *SMAC* [16].

Iterated F-RACE combines a racing procedure based on a statistical test for performance differences between candidate configurations with a model-based sampling mechanism for promising configurations [3, 25, 9]. *GGA* and *GGA++* are based on an evolutionary algorithm and employ tournament-based selection mechanisms for filtering out weak configurations [2, 1]. *ParamILS* performs iterated local search in configuration space and utilises an intensification mechanism similar to racing for comparing promising configurations [17]. Finally, *SMAC* is a model-based optimisation procedure that uses random forest models [4] to identify promising configurations, as well as the racing-based intensification mechanism from ParamILS to assess them [16].

SMAC is one of the best-performing and versatile algorithm configuration procedures currently available; we therefore chose it to automatically configure our *PbO-CCSAT* solver framework. In particular, *SMAC* supports conditional parameters (*i.e.,* parameters that only appear when other parameters take certain values), which play an important role in PbO-based solvers, such as *PbO-CCSAT*.

1.4 Portfolio-based Algorithm Selection

In this subsection, we briefly review the portfolio-based algorithm selection, which can be seen as a complementary approach to automated algorithm configuration. Portfolio-based algorithm selection is proposed to address the algorithm selection problem: when there exist a number of solvers for solving the same problem, how to choose the most suitable solver? Considerable attentions have been paid to this research direction, resulting in various effective approaches, including *SATzilla* [37], *ISAC* [19], *3S* [18], *CSHC* [30], *AutoFolio* [23] and *ACPP* [24].

We would like to note that our *PbO-CCSAT* solver is essentially different from portfolio-based algorithm selection. On one hand, portfolio-based algorithm selection only predicts a suitable solver or a scheduling list of solvers, and then directly call existing solver(s) to solve the target benchmarking instance. On the other hand, *PbO-CCSAT* uses the paradigm of PbO, and is built by integrating key components from CC-based SLS solvers and other effective algorithmic techniques. In this sense, in the design space of *PbO-CCSAT*, it is of high possibilities to construct new SLS solvers, assembling CC-based techniques and other algorithmic techniques, which perform much better than current state-of-the-art SLS solvers on a broad range of application SAT instances. Hence, this is the case for our work, and the details can be found in the experiment part of this work.

References

1. Ansótegui, C., Malitsky, Y., Samulowitz, H., Sellmann, M., Tierney, K.: Model-based genetic algorithms for algorithm configuration. In: Proceedings of IJCAI 2015. pp. 733–739 (2015)
2. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. In: Proceedings of CP 2009. pp. 142–157 (2009)
3. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-Race and iterated F-Race: An overview. In: Experimental Methods for the Analysis of Optimization Algorithms, pp. 311–336. Springer (2010)
4. Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001)
5. Cai, S., Su, K.: Configuration checking with aspiration in local search for SAT. In: Proceedings of AAAI 2012. pp. 434–440 (2012)
6. Cai, S., Su, K.: Local search for Boolean satisfiability with configuration checking and subscore. Artificial Intelligence **204**, 75–98 (2013)
7. Cai, S., Su, K., Luo, C., Sattar, A.: NuMVC: An efficient local search algorithm for minimum vertex cover. Journal of Artificial Intelligence Research **46**, 687–716 (2013)
8. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence **175**(9-10), 1672–1696 (2011)

9. Dang, N., Cáceres, L.P., Causmaecker, P.D., Stützle, T.: Configuring irace using surrogate configuration benchmarks. In: Proceedings of GECCO 2017. pp. 243–250 (2017)
10. Gao, C., Yao, X., Weise, T., Li, J.: An efficient local search heuristic with row weighting for the unicost set covering problem. European Journal of Operational Research **246**(3), 750–761 (2015)
11. Glover, F.: Tabu search – Part I. INFORMS Journal on Computing **1**(3), 190–206 (1989)
12. Glover, F.: Tabu search – Part II. INFORMS Journal on Computing **2**(1), 4–32 (1990)
13. Hoos, H.H.: Programming by optimization. Communications of the ACM **55**(2), 70–80 (2012)
14. Hutter, F., Babić, D., Hoos, H.H., Hu, A.J.: Boosting verification by automatic tuning of decision procedures. In: Proceedings of FMCAD 2007. pp. 27–34 (2007)
15. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Automated configuration of mixed integer programming solvers. In: Proceedings of CPAIOR 2010. pp. 186–202 (2010)
16. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proceedings of LION 2011. pp. 507–523 (2011)
17. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: An automatic algorithm configuration framework. Journal of Artificial Intelligence Research **36**, 267–306 (2009)
18. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: Proceedings of CP 2011. pp. 454–469 (2011)
19. Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K.: ISAC - instance-specific algorithm configuration. In: Proceedings of ECAI 2010. pp. 751–756 (2010)
20. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. In: Proceedings of IJCAI 2009. pp. 517–524 (2009)
21. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search SAT solvers from components. Artificial Intelligence **232**, 20–42 (2016)
22. Kotthoff, L., Thornton, C., Hoos, H.H., Hutter, F., Leyton-Brown, K.: Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. Journal of Machine Learning Research **18**, 25:1–25:5 (2017)
23. Lindauer, M., Hoos, H.H., Hutter, F., Schaub, T.: AutoFolio: An automatically configured algorithm selector. Journal of Artificial Intelligence Research **53**, 745–778 (2015)
24. Lindauer, M., Hoos, H.H., Leyton-Brown, K., Schaub, T.: Automatic construction of parallel portfolios via algorithm configuration. Artificial Intelligence **244**, 272–290 (2017)
25. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
26. Luo, C., Cai, S., Su, K., Huang, W.: CCEHC: An efficient local search algorithm for weighted partial maximum satisfiability. Artificial Intelligence **243**, 26–44 (2017)
27. Luo, C., Cai, S., Su, K., Wu, W.: Clause states based configuration checking in local search for satisfiability. IEEE Transactions on Cybernetics **45**(5), 1014–1027 (2015)
28. Luo, C., Cai, S., Wu, W., Jie, Z., Su, K.: CCLS: An efficient local search algorithm for weighted maximum satisfiability. IEEE Transactions on Computers **64**(7), 1830–1843 (2015)

29. Luo, C., Cai, S., Wu, W., Su, K.: Double configuration checking in stochastic local search for satisfiability. In: Proceedings of AAAI 2014. pp. 2703–2709 (2014)
30. Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm portfolios based on cost-sensitive hierarchical clustering. In: Proceedings of IJCAI 2013. pp. 608–614 (2013)
31. Michiels, W., Aarts, E.H.L., Korst, J.H.M.: Theoretical aspects of local search. Springer (2007)
32. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of KDD 2013. pp. 847–855 (2013)
33. Tompkins, D.A.D., Balint, A., Hoos, H.H.: Captain Jack: New variable selection heuristics in local search for SAT. In: Proceedings of SAT 2011. pp. 302–316 (2011)
34. Vallati, M., Fawcett, C., Gerevini, A., Hoos, H.H., Saetti, A.: Automatic generation of efficient domain-optimized planners from generic parametrized planners. In: Proceedings of SOCS 2013. pp. 184–192 (2013)
35. Wang, Y., Cai, S., Yin, M.: Two efficient local search algorithms for maximum weight clique problem. In: Proceedings of AAAI 2016. pp. 805–811 (2016)
36. Wang, Y., Cai, S., Yin, M.: Local search for minimum weight dominating set with two-level configuration checking and frequency based scoring function. Journal of Artificial Intelligence Research **58**, 267–295 (2017)
37. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. Journal of Artificial Intelligence Research **32**, 565–606 (2008)
38. Zhang, H., Cai, S., Luo, C., Yin, M.: An efficient local search algorithm for the winner determination problem. Journal of Heuristics **23**(5), 367–396 (2017)