Verifiable MaxSAT solutions

Dieter Vandesande Wolf De Wulf

January 1, 2022

1 Introduction

This paper is part of the first project of the course Discrete Modeling, Optimisation and Search taught at the VUB. The assignment consists out of creating a ProofLogged MaxSAT-solver. In SAT, proof logging is already well established. When a SAT-solver returns a model, and thus says the theory is satisfiable, checking if the returned model is indeed a model of the theory can be seen as proving satisfiability. However, when a SAT-solver returns unsatisfiable, this response can either come from the theory being unsatisfiable or from the solver having a bug. In the SAT-competition of 2013, a SAT-solver named BreakIDGlucose returned unsatisfiable on some hard instances. Unfortunately, this response was due to a bug. After having resolved the bug, these instances were not solvable anymore by BreakIDGlucose. Therefore, proof logging for SAT-solvers is mandatory in the latest SAT-sompetitions (Heule et al., 2019). A proof, also called a certificate, is written by a solver and should be written efficiently, as it should not influence the performance of the SAT-solver. Moreover, it should be verifiable in an efficiently manner. VeriPB (Gocht et al., 2020) is such a proof verifier. It expresses the problems as pseudo-Boolean (PB) problems and combines Reverse Unit Propagation (RUP) and Cutting Planes proof system rules.

MaxSAT (Biere et al., 2009; C. M. Li & Manya, 2009) is the problem for which it is tried to make as much as possible (soft) clauses satisfied. When it is possible to also have hard clauses, which have to be satisfied in all possible models, it is said to be a partial MaxSAT-problem. When the clauses are weighted and the problem is to find the assignment that maximizes the sum of the weights of the satisfied soft clauses, it is said to be a weighted MaxSAT-problem. There are multiple approaches to solve a MaxSAT-instance. While core-guided MaxSAT-solvers and MaxSAT-solvers starts with unsatisfiable cores and try to manipulate these cores until a satisfiable subset of the clauses is found, iterative (Koshimura et al., 2012) and Branch and Bound (BnB) (Abramé & Habet, 2014; C.-M. Li et al., 2021) solvers start by finding a model and keep on trying to find a better model until no better model can be found. Except the branch and bound (BnB) solvers, most of the MaxSAT-solvers translate the MaxSAT-instance into a SAT-instance and repeatedly call a SAT-solver as an oracle.

As part of the assignment, we received a yet unpublished extension on VeriPB to allow to verify proofs for optimisation problems. Apart from adding some new proof rules, this extension keeps track of an upper bound respectively a lower bound when minimizing respectively maximizing. When having found a better value, it will update the bound and try to find a better solution by adding a constraint that the value to optimize will be better than the current best value. If this pose a contradiction, the last found best solution is the actual optimal solution to the problem. For the rest of this paper, when we talk about VeriPB, we mean this extension of VeriPB.

Since the idea of the VeriPB proof system for optimisation is using the same ideology to optimisation as both the BnB and the iterative MaxSAT solvers (starting with a satisfying model and keep on improving until no better one can be found), we decided to start working on an iterative MaxSAT-solver. We have chosen to use QMaxSAT (Koshimura et al., 2012). QMaxSAT was the best performing MaxSAT-solver in the partial MaxSAT industrial competition of 2010[1]. QMaxSAT adds a relaxation variable to all soft clauses. When a relaxation variable of a soft clause is falsified, that means that the clause has to be satisfied. Therefore, the optimisation is done by minimising the number of satisfied relaxation variables. QMaxSAT iteratively calls MiniSat (Eén & Sörensson, 2004) with the same theory. Therefore, MiniSat can keep its learned clauses, which improves the performance of later MiniSat-calls. Every call of MiniSat, QMaxSAT constraints the maximum number of relaxation variables to be satisfied. To do so, QMaxSAT uses the Totalizer encoding (Bailleux & Boufkhad, 2003) of the cardinality constraint.

The rest of this paper is organised as follows: ...

2 References

Abramé, A., & Habet, D. (2014). Ahmaxsat: Description and evaluation of a branch and bound max-sat solver. *Journal on Satisfiability, Boolean Modeling and Computation*, *9*(1), 89–128.

Bailleux, O., & Boufkhad, Y. (2003). Efficient CNF Encoding of Boolean Cardinality Constraints. In F. Rossi (Ed.), *Principles and Practice of Constraint Programming – CP 2003* (pp. 108–122). Springer. https://doi.org/10.1007/978-3-540-45193-8_8

Biere, A., Heule, M., & van Maaren, H. (2009). *Handbook of satisfiability* (Vol. 185). IOS press.

[1] http://www.maxsat.udl.cat/10/results/

Eén, N., & Sörensson, N. (2004). An Extensible SAT-solver. In E. Giunchiglia & A. Tacchella (Eds.), *Theory and Applications of Satisfiability Testing* (pp. 502–518). Springer. https://doi.org/10.1007/978-3-540-24605-3_37

Gocht, S., McCreesh, C., & Nordström, J. (2020). *Veripb: The easy way to make your combinatorial search algorithm trustworthy* [Presented at the workshop *From Constraint Programming to Trustworthy AI* at the *26th International Conference on Principals and Practice of Constraint Programming (CP2020)*]. https://www.csc.kth.se/~jakobn/research/VeriPB_CPTAI2020.pdf

Heule, M. J., Järvisalo, M., & Suda, M. (2019). Sat competition 2018. *Journal on Satisfiability, Boolean Modeling and Computation*, *11*(1), 133–154.

Koshimura, M., Zhang, T., Fujita, H., & Hasegawa, R. (2012). QMaxSAT: A Partial Max-SAT Solver. *Journal on Satisfiability, Boolean Modeling and Computation*, *8*(1-2), 95–100. https://doi.org/10.3233/SAT190091

Li, C. M., & Manya, F. (2009). Maxsat, hard and soft constraints. *Handbook of satisfiability* (pp. 613–631). IOS Press.

Li, C.-M., Xu, Z., Coll, J., Manyà, F., Habet, D., & He, K. (2021). Combining clause learning and branch and bound for maxsat. *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*.