MaxCDCL and WMaxCDCL in MaxSAT Evaluation 2022

1st Jordi Coll, 2nd Shuolin Li
Aix Marseille Univ,*Université de Toulon*
CNRS, LIS, Marseille, France
jordi.coll@lis-lab.fr, shuolin.li@etu.univ-amu.fr

3rd Chu-Min Li
Université de Picardie Jules Verne, Amiens, France
Aix Marseille Univ,*Université de Toulon*, CNRS, LIS, Marseille, France
chu-min.li@u-picardie.fr

4th Felip Manyà
Artificial Intelligence Research Institute
CSIC, Bellaterra, Spain
felip@iiia.csic.es

5th Djamal Habet
Aix Marseille Univ,
Université de Toulon
CNRS, LIS, Marseille, France
Djamal.Habet@univ-amu.fr

6th Kun He
Huazhong University of Science and Technology
Wuhan,China
brooklet60@hust.edu.cn

I. Introduction

MaxCDCL and WMaxCDCL are two new unweighted and weighted partial MaxSAT solvers respectively. The main solving algorithm is MaxCDCL [1] for the MaxCDCL solver, and a weighted extension of MaxCDCL for the WMaxCDCL solver.

II. MaxCDCL Algorithm

The MaxCDCL algorithm is an extension for MaxSAT of the CDCL algorithm which combines Branch and Bound and clause learning. Similarly as done in CDCL, the MaxCDCL algorithm roughly alternates decisions and unit propagation with conflict analysis and clause learning. Moreover, at some selected nodes of the search tree, MaxCDCL computes a lower bound (LB) of the number of soft clauses that will be falsified in any solution that satisfies the hard clauses. If the bounding procedure detects that the current assignment cannot be extended to a satisfying assignment that improves the best solution found so far, i.e. $LB \geq UB$, a *soft conflict* is detected. Similarly to (hard) conflicts in CDCL, which can also occur in MaxCDCL, and where a hard clause is falsified, MaxCDCL detects an implicit clause that is falsified when a soft conflict occurs. Both after hard and soft conflicts, conflict analysis is used to find the first unique implication point and backtrack. In addition, when the lower bounding procedure does not detect a soft conflict but $LB = UB - 1$, all non-falsified soft clauses can be hardened. This hardening is done by unit propagation after introducing new clauses explaining the reason of the hardening.

The computation of the lower bound is based on the detection of local unsatisfiable cores, i.e. cores that depend on the current partial assignment. Roughly, the detection of a local core is done by assuming soft clauses to be true and applying unit propagation until some conflict is found [2]–[4]. For every detected local core, the lower bound can be increased by one. A detailed description of the unweighted MaxCDCL algorithm can be found in [1], [5].

III. WMaxCDCL Algorithm

There are some adaptations that we do on the MaxCDCL algorithm to deal with weights. Here we describe the main ones. These particularities require the use many more data structures in WMaxCDCL solver. Therefore, although WMaxCDCL can solve unweighted instances, the source codes of MaxCDCL and WMaxCDCL are different.

The contribution to the lower bound for each core is not always one but it is the minimum weight among the soft clauses of the core. Hence, we make virtual copies of the soft clauses by splitting their weights so that every clause can belong to multiple local cores. More precisely, we dynamically decrease the weights of the soft clauses as they appear in new cores.

There are also relevant changes regarding hardening, since the fact that different soft clauses can have different weights implies that hardening can happen more frequently and at different decision levels. Moreover, after hardening soft clauses, usually more unit propagations can be done, which can falsify new soft clauses causing an increase the lower bound of the cost, which at turn can enable more hardening. Therefore, in WMaxCDCL, the unit propagation phase of CDCL is replaced by a fix point propagation loop that alternates unit propagation and hardening.

IV. Implementation details

Both MaxCDCL and WMaxCDCL solvers are implemented on top of MapleCOMSPS_LRB [6]. They include a number of preprocessing, inprocessing, and additional techniques to enhance their performance that we list in this section.

We compute a first upper bound *initUB* and lower bound *initLB* of the optimal cost, in order to limit the search, with a combination of methods. First, MaxHS [7] is run for 5 minutes to find initial bounds. The MaxHS version from [7] has been slightly adapted to deal with the new instance format and set time limits. The binary files submitted to the competition are compiled with IBM ILOG CPLEX version 22.1.

Then, the solver tries to find an initial feasible cost smaller than *initUB* by solving the problem with a sequence of increasing upper bound values UB starting at *initLB*, until a feasible solution is found or *initUB* is reached. More precisely we update the sequence by $UB{=}UB \times k$, where k is 2 in MaxCDCL and 1.5 WMaxCDCL.

Before starting the search we find incompatible subsets of soft clauses by unit propagation, i.e. sets of soft clauses of which at most one of them can be satisfied according to hard clauses. Then, every set of clauses $c_1, \ldots, c_n$ is replaced by a new unit soft clause d, defined by hard clauses as $d \leftrightarrow c_1 \vee \cdots \vee c_n$, and the cost of any solution is increased by $n-1$. In the weighted case, the weighted clauses (c_i, w_i) have been split into $(c_i, w_i - m)$ and (c_i, m), where m is the minimum among the weights of $c_1, \ldots, c_n$. Then, the weight of d is defined to be m and the cost is increased by $m(n-1)$.

When the number of free soft clauses n and the upper bound UB are small, we add as implied constraints a CNF encoding of cardinality (resp. pseudo-Boolean) constraints in MaxCDCL (resp. WMaxCDCL), expressing that the cost of the solution must be smaller than the best found upper bound. In particular, in MaxCDCL we add the Sequential Counter encoding [8] when $n \times UB \leq 10^4$, and in WMaxCDCL we add the MDD encoding [9] when $n \leq 50$ and $n \cdot K \leq 10^5$, and otherwise the GGPW encoding [10] when $n \leq 5000$ or $n \leq 500$ and $n \cdot K \leq 10^5$.

We include a number of inprocessing algorithms to simplify the formula, namely failed literal detection, equivalent literal detection, and clause simplification. Moreover, only in WMaxCDCL, we try to improve some of the suboptimal solutions that are found by means of a custom implementation of the local search method described in [11].

Acknowledgments

This work has been partially funded by the French Agence Nationale de la Recherche, reference ANR-19-CHIA-0013-01, and project PID2019-111544GB-C21 funded by MCIN/AEI/10.13039/501100011033, and partially supported by Archimedes Institute, Aix-Marseille University. F. Manyà was supported by mobility grant PRX21/00488 of the *Ministerio de Universidades*. We thank the Université de Picardie Jules Verne for providing the Matrics Platform.

References

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

[2] C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes, "Exploiting cycle structures in Max-SAT," in *In Proceedings of SAT 2009*, ser. LNCS, vol. 5584. Springer, 2009, pp. 467–480.

[3] C. M. Li, F. Manyà, and J. Planes, "Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers," in *Proceedings of CP 2005*, ser. LNCS, vol. 3709. Springer, 2005, pp. 403–414.

[4] ——, "Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT," in *Proceedings of AAAI 2006*, 2006, pp. 86–91.

[5] C.-M. Li, Z. Xu, J. Coll, F. Manyà, D. Habet, and K. He, "Boosting branch-and-bound maxsat solvers with clause learning," *AI Communications*, no. Preprint, pp. 1–21, 2021.

[6] J. H. Liang, C. Oh, V. Ganesh, K. Czarnecki, and P. Poupart, "MapleCOMSPS, MapleCOMSPS LRB, MapleCOMSPS CHB," in *Proceedings of SAT Competition 2016: Solver and Benchmark Descriptions*, 2016, pp. 52–53.

[7] F. Bacchus, "MaxHS in the 2021 MaxSAT Evaluation," *MaxSAT Evaluation 2021*, p. 14, 2021.

[8] C. Sinz, "Towards an optimal CNF encoding of Boolean cardinality constraints," in *Proceedings of CP 2005*. Springer LNCS 3709, 2005, pp. 827–831.

[9] M. Bofill, J. Coll, J. Suy, and M. Villaret, "An mdd-based SAT encoding for pseudo-boolean constraints with at-most-one relations," *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5157–5188, 2020.

[10] M. Bofill, J. Coll, P. Nightingale, J. Suy, F. Ulrich-Oltean, and M. Villaret, "SAT encodings for pseudo-boolean constraints together with at-most-one constraints," *Artificial Intelligence*, vol. 302, p. 103604, 2022.

[11] J. Zheng, K. He, J. Zhou, Y. Jin, C.-M. Li, and F. Manya, "Bandmaxsat: A local search MaxSAT solver with multi-armed bandit," in *Proceedings of 31st International Joint Conference on Artificial Intelligence (To appear)*, 2022.