MaxCDCL in MaxSAT Evaluation 2023

1st Chu-Min Li
Université de Picardie Jules Verne,
MIS, Amiens, France
Aix Marseille Univ, *Université de Toulon*
CNRS, LIS, Marseille, France
chu-min.li@u-picardie.fr

2rd Jordi Coll
Artificial Intelligence Research Institute
CSIC, Bellaterra, Spain
jcoll@iiia.csic.es

3rd Shuolin Li, 4th Djamal Habet
Aix Marseille Univ, *Université de Toulon*
CNRS, LIS, Marseille, France
shuolin.li@etu.univ-amu.fr,
Djamal.Habet@univ-amu.fr

5th Felip Manyà
Artificial Intelligence Research Institute
CSIC, Bellaterra, Spain
felip@iiia.csic.es

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

I. Introduction

MaxCDCL is an extension of CDCL (Conflict Driven Clause Learning) based on the Branch-and-Bound (BnB) scheme for unweighted (partial) MaxSAT [1]. Its main distinguishing feature w.r.t. a CDCL SAT solver is a lookahead procedure to underestimate a lower bound (LB) of the number of soft clauses that will be falsified if search continues. If LB reaches the current upper bound (UB), i.e., if LB$\geq$UB, then any solution better than the best solution found so far does not exist. This situation is called a soft conflict, from which an analysis is carried out to derive a hard clause explaining the soft conflict and to guide the backtracking, so that the same soft conflict will not be produced again in the future. When a hard clause is falsified, the conflict is said to be hard, and MaxCDCL analyzes it as in a CDCL SAT solver to learn a hard clause. The soft conflicts are together with the hard conflicts to drive the entire search, given UB.

MaxCDCL already participated in MaxSAT evaluation 2022 and was ranked 6th over 11 competitors in the unweighted partial MaxSAT track [2].

II. New techniques in MaxCDCL 2023

A. Bounded Variable Elimination and Equivalent Literal Substitution

MaxCDCL 2022 already implemented Bounded Variable Elimination (BVE) and Equivalent Literal Substitution (ELS). However, BVE was only performed in preprocessing, and ELS was only performed after the first feasible solution is found. This was a conservative approach.

In fact, MaxCDCL begins search by setting UB=1. While it does not find any feasible solution satisfying all hard clauses and falsifying less than UB soft clauses, it multiplies UB by 2 and restarts search. However, all changes made for UB are not valid for 2$\times$UB. In MaxCDCL 2022, when UB is doubled, we simply removed all learnt clauses and all variables assignments. When a feasible solution satisfying all hard clauses and falsifying less than UB soft clauses is found, let k be the number of falsified soft clauses, MaxCDCL sets k as the new UB. All changes made for UB, including ELS, are also valid for UB$\leftarrow k$, because $k <$UB. So, it is valid to do preprocessing BVE by setting UB$= \infty$ and ELS after finding the first feasible solution.

In 2023, MaxCDCL saves all original clauses and the relevant information for variables, which are restored each time before UB is increased. In this way, inprocessing BVE and ELS are applied for all UBs, because all clauses, including original clauses, possibly modified in search for smaller UB, are simply removed before increasing UB. On the contrary, when UB is decreased, the modified clauses are kept.

Note that variables occurring in soft clauses are never removed. If a soft clause s is equivalent to a literal h such that h or $\neg h$ does not occur in any soft clause, then h is replaced by s in all hard clauses. If two soft clauses s_1 and s_2 are complementary, i.e., s_1 is satisfied if and only if s_2 is falsified, then the constant cost is increased by 1, and both s_1 and s_2 are removed. If s_1 and s_2 are equivalent, nothing is done, which is different from weighted partial MaxSAT for which s_1 and s_2 could be unified by removing one of them and the sum of their weights becomed the weight of the remaining one.

Inprocessing BVE is called when the total number of literals in original hard clauses is decreased by 1% or 1% variables are assigned at level 0. The variables are checked and eliminated in increasing order of their activity, as proposed in [3]. Since MaxCDCL switches between LRB [4] and VSIDS [5] to define variable activity, the activity of a variable is the current LRB (VSIDS) score used for selecting decision variables.

B. Learnt clause deletion based on variable activity

Similar to a CDCL SAT solver, MaxCDCL learns a new clause from each soft conflict and each hard conflict. These learnt clauses are organized into three subsets as in its base SAT solver Maple_CM [6]: CORE, TIER2 and LOCAL according to their LBD as defined in [7], the learnt clauses in LOCAL having large LBD and being useful only locally.

In MaxCDCL 2022, as well as in its base SAT solver Maple_CM, the clauses in LOCAL are sorted periodically in the increasing order of their activity and the first half is removed, the activity of a clause roughly corresponding to the number of times the clause is used in the recent conflicts.

MaxCDCL 2023 re-defines the activity of each clause c in LOCAL as follows. Let minAct (minAct2) be the (second) smallest variable activity in c. The activity of c is defined to be (1000×minAct+minAct2)/(LBD×LBD). This idea is inspired from [3] in which variables are eliminated in the increasing order of their activity in inprocessing BVE, so that variables with high activity are protected from being eliminated. The intuition is that a clause not frequently used in recent conflicts can be useful (i.e., can easily become unit or falsified) in the future and should not be removed if the activities of its variables are high. On the contrary, a clause containing two small activity variables will be unlikely useful and can be removed without hurting search.

C. Learnt clause vivification based on variable activity

The authors of [6], [8] asked and answered three questions when vivifying clauses: (1) when should the clause vivification be performed? (2) what are the clauses to be vivified in a clause vivification? (3) What is the order to propagate the literals when vivifying a clause? However, there is a question that is never asked nor answered in our knowledge: in a set of clauses, which clause should be vivified first?

This question is very important, because a clause simplified, i.e., one or more literals are removed by vivification, can help vivify other clauses. The intuition is that the clauses having greater probability to be simplified should be vivified first. The issue then becomes how to estimate the probability with which a clause can be simplified. Nevertheless, we do not need the real value of this probability. What we need in clause vivification is the order of this probability w.r.t. other clauses.

We use the minimum variable activity in a clause to compare the probability with which the clause can be simplified. Intuitively, if the variables of a clause all have high activities, the clause might have great probability to be simplified in a clause vivification. Consequently, MaxCDCL 2023 vivifies the clauses in a set in the decreasing order of their minimum variable activity. This idea is also inspired from [3].

D. Failed literal detection based on variable activity

Periodically, failed literal detection is performed before a slected restart at level 0. Since this detection is time-consuming, only a small subset of literals is detected. A question is then how to select this subset of literals to detect. MaxCDCL 2023 selects the 500 variables with higher activity and for each variable x detects two literals x and $\neg x$. Intuitively, these literals have high probability to be failed. In addition, even if x, as well as $\neg x$, is not failed, we can discover easily those literals l such that $x \rightarrow l$ and $\neg x \rightarrow l$, or $x \rightarrow l$ and $\neg x \leftarrow \neg l$. In the former case, l can be satisfied immediately, and in the second case, x and l are equivalent literals to be treated by equivalent literal substitution.

III. Versions of MaxCDCL in MaxSAT evaluation 2023

We submit three versions of MaxCDCL to MaxSAT evaluation 2023. The first version is pure MaxCDCL without using any third-party solver. The second version first calls SCIP [9] for 10 minutes, then MaxHS [10] for 15 minutes, and finally pure MaxCDCL for 35 minutes. If the instance is not solved by SCIP in 10 minutes, nor by MaxHS in 15 minutes, MaxCDCL reads the best UB obtained by MaxHS (if any) and uses it as the initial UB (initUB). Recall that MaxCDCL begins search by setting UB=1, then while UB is not feasible, sets UB$\leftarrow min(2 \times UB, initUB + 1)$, and then while UB is feasible and k is the number of falsified soft clauses under UB, sets UB$\leftarrow k$. The last infeasible UB is the optimal solution. The third version is the same as the second one but giving 20 minutes to MaxHS and 30 to pure MaxCDCL.

Acknowledgments

This work has been partially supported by AI CHAIR reference ANR-19-CHIA-0013-01 (MASSAL'IA) and project ANR-20-ASTR-0011 (POSTCRYPTUM) funded by the French Agence Nationale de la Recherche, and projects PID2019-111544GB-C21 and TED2021-129319B-I00 funded by MCIN/AEI/10.13039/501100011033, and partially supported by Archimedes Institute, Aix-Marseille University. 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] J. Coll, S. Li, C.-M. Li, F. Manya, D. Habet, and K. He, "Maxcdcl and wmaxcdcl in maxsat evaluation 2022," in *MaxSAT Evaluation 2022 : Solver and Benchmark Descriptions*. Department of Computer Science, University of Helsinki, 2022, pp. 15–16.

[3] S. Li, C.-M. Li, M. Luo, J. Coll, D. Habet, and F. Manya, "A new variable ordering for in-processing bounded variable elimination in sat solvers," in *Proceedings of the 32nd International Joint Conference on Artificial Intelligence*, 2017, pp. 703–711.

[4] J. H. Liang, V. Ganesh, P. Poupart, and K. Czarnecki, "Learning rate based branching heuristic for sat solvers," in *Theory and Applications of Satisfiability Testing–SAT 2016: 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings 19*. Springer, 2016, pp. 123–140.

[5] L. Zhang, C. F. Madigan, M. H. Moskewicz, and S. Malik, "Efficient conflict driven learning in a Boolean satisfiability solver," in *Proccedings of the IEEE/ACM International Conference on Computer Aided Design, ICCAD*, 2001, pp. 279–285.

[6] C.-M. Li, F. Xiao, M. Luo, F. Manya, Z. Lu, and Y. Li, "Clause vivification by unit propagation in cdcl sat solvers,," *Artificial Intelligence, volume 279*, p. 103197, 2020.

[7] L. Simon and G. Audemard, "Predicting learnt clauses quality in modern sat solvers," in *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2009, pp. 399–404.

[8] M. Luo, C.-M. Li, F. Xiao, F. Manya, and Z. Lü, "An effective learnt clause minimization approach for cdcl sat solvers," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 703–711.

[9] K. Bestuzheva et al., "The SCIP Optimization Suite 8.0," Optimization Online, Technical Report, December 2021.

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