

MaxCDCL in MaxSAT Evaluation 2024

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*

2nd Shuolin Li*

*Aix Marseille Univ, Université de Toulon
CNRS, LIS, Marseille, France
shuolin.li@etu.univ-amu.fr*

3rd Jordi Coll

*Universitat de Girona
Girona, Spain
jordicoll@udg.edu*

4th Djamal Habet

*Aix Marseille Univ, Université de Toulon
CNRS, LIS, Marseille, France
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 evaluations 2022 [2] and 2023 [3]. It was ranked 6th over 11 competitors in the unweighted partial MaxSAT track in 2022, and was ranked 2nd in 2023.

II. NEW TECHNIQUES IN MAXCDCL 2024

A. Reinforcement learning for lower bounding

MaxCDCL represents each soft clause using a soft literal. A local core w.r.t. a partial assignment α is a subset of soft literals that cannot be satisfied at the same time without falsifying a hard clause under α . Thus, an important property of a core w.r.t. α is that it contains at least a falsified soft literal if α is extended to a complete assignment. So, the number of disjoint local cores w.r.t. α is a lower bound (LB) of the number of falsified soft literals for any assignment extended from α .

Unfortunately, the detection of a local core is time-consuming, and if the computed LB does not reach UB, i.e., if the LB computation is not successful, the effort spent to compute LB is lost. Consequently, MaxCDCL does not detect

local cores for all partial assignments. Instead, the previous versions of MaxCDCL use a probing technique to estimate the average a and standard deviation d of the number of disjoint local cores detected in a successful LB detection, and decides whether the LB computation should be hired for a partial assignment based on a and d . Concretely, let k be the number of disjoint local cores to detect for a successful LB computation, and c be a parameter. If $k \leq a + c * d$, then the LB computation should be hired. The intuition is as follows. Let n be the number of disjoint local cores to be detected in a successful LB computation. A reasonable hypothesis is that n is a random number with normal distribution. Then 95% (99%) of its values are between $a - c * d$ and $a + c * d$ when $c = 2$ (3) according to a well-known property in statistics. In practice, c is adjusted to maintain a successful rate between 60% and 75%.

In MaxCDCL2024, we use reinforcement learning with 2 armed-bandit mechanism to decide if LB computation should be hired. For this purpose, two actions, *yes* and *no*, are associated with each k , where k is the number of disjoint local cores for a LB computation to be successful. Initially, the score of *yes* and *no* is initialized to 0. If a LB computation is hired and is successful, then the $score(yes)$ and $score(no)$ are updated as follows.

$$score(yes) \leftarrow score(yes) + stepSize * (1 - score(yes))$$

$$score(no) \leftarrow score(no) * stepSize * (1 - score(no))$$

If the LB computation is not successful, then

$$score(yes) \leftarrow score(yes) * stepSize * (1 - score(yes))$$

$$score(no) \leftarrow score(no) + stepSize * (1 - score(no))$$

where $stepSize$ is a parameter initialized to 0.4 and is decreased by 10^{-5} before every LB computation until it reaches 0.04.

*Corresponding author

In summary, the conditions to hire LB computation in MaxCDCL2024 are as follows. For each new conflict, with a small probability (0.02), LB computation is always hired for exploration. Otherwise, LB computation should be hired if $c * score(yes) \geq score(no)$, where c is initialized to 1 and is adjusted to maintain the success rate of LB computation to be between 0.6 and 0.75.

B. Incremental lower bounding

In previous versions of MaxCDCL, the disjoint local cores are discarded immediately after a LB computation, whether the computation is successful or not. In MaxCDCL2024, if the LB computation is not successful, the detected cores are kept until backtracking or the next LB computation, which has two consequences:

- Let k be the number of disjoint local cores, and $k < UB$. If next branchings make n new falsified soft literals that do not belong to these cores, and $k + n \geq UB$, then MaxCDCL should backtrack, avoiding a new LB computation.
- Let $\{s_1, \dots, s_d\}$ be a local core, if $d - 1$ literals, say, $s_2, \dots, s_d$, are assigned true and s_1 is not assigned, then s_1 should be assigned false and propagated. Furthermore, let $\{h_1, \dots, h_r\}$ be the reason of the core. The clause $\neg s_1 \vee h_1 \vee \dots \vee h_r \vee \neg s_2 \vee \dots \vee \neg s_d$ is created as the reason of $\neg s_1$.

C. Instances with few soft conflicts

MaxCDCL encounters very few soft conflicts when solving some particular instances. In other words, the time-consuming lookahead procedure is not effective for these instances and can learn clauses of low quality from the few soft conflicts. In MaxCDCL2024, the normal conditions to hire the lookahead procedure are executed for 200s. If $\#softConflicts / (\#softConflicts + \#hardConflicts) < 0.1$ after 200s, then the conditions for hiring lookahead become more restrictive for the rest of solving: The random call to lookahead at each new conflict is canceled, and at most one call to lookahead is hired after a new conflict even if the call is not successful, allowing to save time by avoiding the ineffective lookaheads.

III. VERSIONS SUBMITTED TO MSE2024

We submit two versions of MaxCDCL2024 to MSE2024. The first version is MaxCDCL2024 itself. The second version combines MaxCDCL2024 with Open-WBO [4] (version MSE2023) in a portfolio in which Open-WBO is first executed 300s, followed by MaxCDCL2024 for 3300s. Note that Open-WBO, WMaxCDCL2023 and MaxCDCL2023 are the only exact solvers of MSE2023 which pass successfully the regression test required in MSE2024.

ACKNOWLEDGMENTS

This work is 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 Matrices 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. Manyà, 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] C.-M. Li, J. Coll, S. Li, D. Habet, F. Manyà, and K. He, "Maxcdcl in maxsat evaluation 2023," *MaxSAT Evaluation 2023*, pp. 14–15, 2023.
- [4] R. Martins, V. M. Manquinho, and I. Lynce, "Open-WBO: A modular MaxSAT solver," in *Proceedings of SAT 2014*, ser. LNCS, vol. 8561. Springer, 2014, pp. 438–445.