Improving CDCL via Local Search

Xindi Zhang, Shaowei Cai*, Zhihan Chen

[1]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2]School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China
{zhangxd,caisw,chenzh}@ios.ac.cn

***Abstract*—This document describes our SAT solvers submitted to the SAT Competition 2021, the solvers are based on Relaxed, CaDiCaL and kissat.**

I. LSTECH_MAPLE

$LStech\_Maple$ (short for $LSTech$) is the improved version of $Relaxed\_LCMDCBDL\_newTech$ (short for $Relaxed$), which proposed the relaxed CDCL method [9]. The relaxed method is to relax the backtracking process for protecting promising partial assignment, where a promising assignment is defined according to its consistency and length. When the CDCL process meets some conditions, the algorithm will enter a non-backtracking stage until it gets a full assignment α. Once it gets α, a local search SAT solver is called immediately.

The differences between $LSTech$ and $Relaxed$ lie in the non-backtracking stage entrance conditions and the local search process entrance conditions. Inspired by the updating method of target phase [4], $LSTech$ enters the non-backtracking phase to construct a full assignment each time the CDCL process reaches a higher trail. And $LSTech$ enters the local search process according to the number of restarts, rather than after each non-backtracking stage. The detailed rules are described as follow.

No-backtracking Stage Entrance Rule: Let p be the size of non-conflict trail that allowing the algorithm enter the non-backtracking stage. $p = 0$ at the beginning. If the CDCL process reaches a higher no-conflict trail with size p', then it will enter the non-backtracking stage, and $p \leftarrow p'$ accordingly. Moreover, $p \leftarrow 0.9 \times p$ after each local search process.

Local Search Entrance Rule: CCAnr [3] is called every δ restarts and the best local search solution (denoted as lb_soln) is updated accordingly. δ is set to 300 initially. And $\delta \leftarrow \delta + 300$ if the lb_soln has not been improved, $\delta \leftarrow \delta - 300$ (keeping $\delta > 300$) if the lb_soln has been improved.

Furthermore, we improve the CCAnr with clause state based CC [5] instead of neighborhood based CC [3], and the latter method need to keep a neighborhood list for each variable, which is time and space consuming. In addition, $LSTech$ remove the distance branching strategy [8] in the beginning.

This work was supported by Beijing Academy of Artificial Intelligence (BAAI), and Youth Innovation Promotion Association, Chinese Academy of Sciences [No. 2017150].

* Corresponding author

II. CADICAL_RP

Phase saving [7] is a powerful and standard technique for modern CDCL solvers, which saves the latest polarity of each variable in a vetecr $polarity$. Rephase [4] is recent proposed to reset $polarity$ with some promising full assignments (which is also called phases). The goal of CaDiCaL_rp is to improve their base-solver $CaDiCaL$ and $kissat$ [4] by selecting the appropriate phases based on probability as $Relaxed$ [9].

III. KISSAT_CF

For better utilize the information of local search, we use the local search conflict frequency to enhance the VSIDS branching strategy [6]. The technique is used in $Relaxed$ [9] and $LSTech$ as well. The conflict frequency for a variable v is denoted as $freq(v)$, which is the number of steps in which it appears in at least one unsatisfied clause divided by the total number of steps of the local search process. $freq(v)$ will be updated after each local search process of kissat, in which a local search solver YalSAT [2] is embedded. Every 20 restarts, the activity of each variable v is bumped by $100\times freq(v)$, unless there are variables eliminated in in-processing.

IV. KISSAT_BONUS

Considering that modern branching strategies like VSIDS [6] only bump activity score for each variable based on the recency, which means that the activity scores of the variable in the conflict clause will be bumped by the same score inc, no matter which clause it is. Thus, we designed a method to take the clause quality (measured by LBD [1]) into account. We set a reward coefficient $bonus$ for each new conflict clause, and the score of each variable related to this conflict will be bumped with $inc * bonus$ instead of inc. $bonus$ is 1 when the LBD of the conflict clause is equal to the global average LBD value, and$bonus$ is 2 when the LBD is unit. And we design the $bonus$ factor as an exponential function negatively related to LBD.

REFERENCES

[1] G. Audemard and L. Simon. Predicting learnt clauses quality in modern SAT solvers. In *IJCAI 2009*, pages 399–404, 2009.
[2] A. Biere. Yet another local search solver and lingeling and friends entering the sat competition 2014. *Sat competition*, 2014(2):65, 2014.
[3] S. Cai, C. Luo, and K. Su. Ccanr: A configuration checking based local search solver for non-random satisfiability. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 1–8, 2015.

[4] A. B. K. F. M. Fleury and M. Heisinger. Cadical, kissat, paracooba, plingeling and treengeling entering the sat competition 2020. *SAT COMPETITION 2020*, page 50, 2020.

[5] C. Luo, S. Cai, W. Wu, Z. Jie, and K. Su. Ccls: an efficient local search algorithm for weighted maximum satisfiability. *IEEE Transactions on Computers*, 64(7):1830–1843, 2014.

[6] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference, DAC 2001, Las Vegas, NV, USA, June 18-22, 2001*, pages 530–535, 2001.

[7] K. Pipatsrisawat and A. Darwiche. A lightweight component caching scheme for satisfiability solvers. In J. Marques-Silva and K. A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 294–299, 2007.

[8] F. Xiao, C. Li, M. Luo, F. Manyà, Z. Lü, and Y. Li. A branching heuristic for SAT solvers based on complete implication graphs. *Sci. China Inf. Sci.*, 62(7):72103:1–72103:13, 2019.

[9] X. Zhang and S. Cai. Relaxed backtracking with rephasing. *SAT COMPETITION 2020*, page 15.