CryptoMiniSat 5.6 with YalSAT at the SAT Race 2019

Mate Soos (National University of Singapore), Armin Biere (JKU Linz)

I. Introduction

This paper presents the conflict-driven clause-learning (CLDL) SAT solver CryptoMiniSat v5.6 (*CMS*) augmented with the Stochastic Local Search (SLS) [11] solver YalSAT 03v as submitted to SAT Race 2019.

CryptoMiniSat aims to be a modern, open source SAT solver using inprocessing techniques, optimized data structures and finely-tuned timeouts to have good control over both memory and time usage of inprocessing steps. It also supports, when compiled as such, to recover XOR constraints and perform Gauss-Jordan elimination on them at every decision level. For the competition, this option was disabled. CryptoMiniSat is authored by Mate Soos.

Yet Another Local Search SAT Solver (YalSAT) implements several variants of ProbSAT's [4] algorithm and recent extensions [3]. These variants are selected randomly at restarts, scheduled by a reluctant doubling scheme (Luby). For further details, see [1]. YalSAT is authored by Armin Biere.

A. Composing the Two Solvers

The two solvers are composed together in a way that does *not* resemble portfolio solvers. The system runs the CDCL solver CryptoMiniSat, along with its periodic inprocessing, by default. However, at every N inprocessing step, CryptoMiniSat's irredundant clauses are pushed into the SLS solver (in case the predicted memory use is not too high). The SLS solver is then allowed to run for a predefined number of steps. In case the SLS solver finds a solution, this is given back to the CDCL solver, which then performs all the necessary extension to the solution (e.g. for Bounded Variable Elimination, BVE [5]) and then outputs the solution.

Note that the inclusion of the SLS solver is full in the sense that assumptions-based solving, library-based solver use, and all other uses of the SAT solver is fully supported with SLS solving enabled. Hence, this is not some form of portfolio where a simple shell script determines which solver to run and then runs that solver. Instead, the SLS solver is a full member of the CDCL solver, much like any other inprocessing system, and works in tandem with it. For example, in case an inprocessing step has reduced the number of variables through BVE or increased it through BVA [9], the SLS solver will then try to solve the problem thus modified. In case the SLS solver finds a solution, the main solver will then correctly manipulate it to fit the needs of the "outside world", i.e. the caller.

As the two solvers are well-coupled, the combination of the two solvers can solve problems that neither system can solve on its own. Hence, *the system is more than just a union of its parts* which is not the case for traditional portfolio solvers.

II. Major Improvements

A. Via Negativa

The system has been subjected to a thorough investigation whether all the different systems that have been implemented into it actually make the solver faster. In this spirit, failed literal probing [8], stamping [6], burst searching (random variable picking), and blocked clause elimination [7] have all been disabled.

B. Chronological Backtracking

Chronological backtracking [10] has been implemented into a branch of the solver. However, chronological backtracking (CBT) is a double-edged sword. Firstly, it slows down the solver's normal functionality as it adds a number of expensive checks to both the propagation and the backtracking code. Secondly, it changes the trail of the solver in ways that make it hard to reason about the current state of the solver. Finally, it seems only to help with satisfiable instances which are theoretically less interesting for the author of CryptoMiniSat. These issues make CBT a difficult addition.

Currently, CryptoMiniSat by default does not implement CBT. The SAT Race has two versions submitted, clearly marked, one with, an one without CBT.

C. Cluster Tuning

The author has been generously given time on the ASPIRE-1 cluster of the National Supercomputing Center Singapore[2]. This allowed experimentation and tuning that would have been impossible otherwise. Without this opportunity, CryptoMiniSat would not stand a chance at the SAT Race.

III. General Notes

A. On-the-fly Gaussian Elimination

On-the-fly Gaussian elimination is again part of CryptoMiniSat. This is explicitly disabled for the competition, but the code is available and well-tested. This allows for special uses of the solver that other solvers, without on-the-fly Gaussian elimination, are not capable of.

B. Robustness

CMS aims to be usable in both industry and academia. CMS has over 150 test cases and over 2000 lines of Python just for fuzzing orchestration, and runs without fault under both the ASAN and UBSAN sanitisers of clang. It also compiles and runs under Windows, Linux and MacOS X.

This is in contrast many academic winning SAT solvers that produce results that are non-reproducible, cannot be compiled on anything but a few select systems, and/or produce segmentation faults if used as a library. CryptoMiniSat has extensive fuzzing setup for library usage and is very robust under strange/unexpected use cases.

IV. Thanks

This work was supported in part by NUS ODPRT Grant R-252-000-685-133 and AI Singapore Grant R-252- 000-A16-490. The computational work for this article was performed on resources of the National Supercomputing Center, Singapore[2]. The author would also like to thank all the users of CryptoMiniSat who have submitted over 500 issues and many pull requests to the GitHub CMS repository[12].

References

[1] Anton, B., Daniel, D., Heule, M.J.H., Jarvisalo, M.: Yet another Local Search Solver and Lingeling and Friends Entering the SAT Competition 2014. In: Proceedings of SAT Competition 2014 (2014)

[2] ASTAR, NTU, NUS, SUTD: National Supercomputing Centre (NSCC) Singapore (2018), https://www.nscc.sg/about-nscc/overview/

[3] Balint, A., Biere, A., Fröhlich, A., Schöning, U.: Improving implementation of SLS solvers for SAT and new heuristics for k-SAT with long clauses. In: Sinz, C., Egly, U. (eds.) Theory and Applications of Satisfiability Testing – SAT 2014. pp. 302–316. Springer International Publishing, Cham (2014)

[4] Balint, A., Schöning, U.: Choosing probability distributions for stochastic local search and the role of make versus break. In: Cimatti, A., Sebastiani, R. (eds.) Theory and Applications of Satisfiability Testing – SAT 2012. pp. 16–29. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[5] Eén, N., Biere, A.: Effective preprocessing in SAT through variable and clause elimination. In: Bacchus, F., Walsh, T. (eds.) Theory and Applications of Satisfiability Testing. pp. 61–75. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)

[6] Heule, M.J.H., Järvisalo, M., Biere, A.: Efficient CNF simplification based on binary implication graphs. In: Sakallah, K.A., Simon, L. (eds.) Theory and Applications of Satisfiability Testing - SAT 2011. pp. 201–215. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)

[7] Järvisalo, M., Biere, A., Heule, M.: Blocked Clause Elimination. In: Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Lecture Notes in Computer Science, vol. 6015, pp. 129–144. Springer International Publishing (2010)

[8] Lynce, I., Silva, J.P.M.: Probing-based preprocessing techniques for propositional satisfiability. In: 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2003), 3-5 November 2003, Sacramento, California, USA. p. 105. IEEE Computer Society (2003), https://doi.org/10.1109/TAI.2003.1250177

[9] Manthey, N., Heule, M.J.H., Biere, A.: Automated reencoding of boolean formulas. In: Biere, A., Nahir, A., Vos, T. (eds.) Hardware and Software: Verification and Testing. pp. 102–117. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)

[10] Nadel, A., Ryvchin, V.: Chronological backtracking. In: Beyersdorff, O., Wintersteiger, C.M. (eds.) Theory and Applications of Satisfiability Testing – SAT 2018. pp. 111–121. Springer International Publishing, Cham (2018)

[11] Selman, B., Kautz, H., Cohen, B.: Local search strategies for satisfiability testing. In: DIMACS Series in Discrete Mathematics and Theoretical Computer Science. pp. 521–532 (1995)

[12] Soos, M.: CryptoMiniSat SAT solver GitHub page (2018), https://github.com/msoos/cryptominisat