Patching MiniSat to Deliver Performance of Modern SAT Solvers

Chanseok Oh
New York University
New York, NY, USA

***Abstract*—COMiniSatPS is a patched MiniSat generated by applying a series of small diff patches to the last available version (2.2.0) of MiniSat that was released several years ago. The essence of the patches is to include only minimal changes necessary to make MiniSat sufficiently competitive with modern SAT solvers. One important goal of COMiniSatPS is to provide these changes in a highly accessible and digestible form so that the necessary changes can be understood easily to benefit wide audience, particularly starters and non-experts in practical SAT. As such, the changes are provided as a series of incrementally applicable diff patches, each of which implements one feature at a time. COMiniSatPS has many variations. The variations are official successors to an early prototype code-named SWDiA5BY that saw great successes in the past SAT-related competitive events.**

I. Introduction

It has been shown in many of the past SAT-related competitive events that very simple solvers with tiny but critical changes (e.g, MiniSat [1] hack solvers) can be impressively competitive or even outperform complex state-of-the-art solvers [2]. However, original MiniSat itself is vastly inferior to modern SAT solvers in terms of actual performance. This is no wonder as it has been several years since the last 2.2.0 release of MiniSat. To match the performance of modern solvers, MiniSat needs to be modified to add some of highly effective techniques of recent days. Fortunately, small modifications are enough to bring up the performance of any simple solver to the performance level of modern solvers. COMiniSatPS [1] adopts only simple but truly effective ideas that can make MiniSat sufficiently competitive with recent state-of-the-art solvers. In the same minimalistic spirit of MiniSat, COMiniSatPS prefers simplicity over complexity to reach out to wide audience. As such, the solver is provided as a series of incremental patches to original MiniSat. Each small patch adds or enhances one feature at a time and produces a fully functional solver. Each patch often changes solver characteristics fundamentally. This form of source distribution by patches would benefit a wide range of communities as it is easy to isolate, study, implement, and adopt the ideas behind each incremental change. The goal of COMiniSatPS is to lower the entering bar so that anyone interested can implement and test their new ideas easily on a simple solver guaranteed with exceptional performance.

The patches first transform MiniSat into Glucose [3] and then into SWDiA5BY. Subsequently, the patches implement new techniques described in [2] to generate the current form of COMiniSatPS.

[1]Source is available at http://www.cs.nyu.edu/~chanseok/cominisatps/.

II. Main Sequence and Subdwarf

As a basis, both Main Sequence and Subdwarf implement the three-tiered learned clause management scheme introduced in [2]. However, Subdwarf is a highly experimental proof-of-concept solver that works radically different from the Main Sequence solver. The three-tiered clause management is an improvement to the clause management scheme of the predecessor SWDiA5BY. The new scheme extends the notion of *core* and *local* learned clauses to include mid-tier learned clauses.

A. Main Sequence

The solver implements the hybrid restart strategy, alternating variable decay factors for VSIDS [4], and three-tiered learned clause management [2]. A few minor modifications have been made to the solver, of which two are as follows:

1) The solver starts in the Glucose-restart [5] phase and runs in the phase for the first 10,000 conflicts before beginning the normal cycles of hybrid phases.
2) For learned clauses having their LBD [3] greater than 50, the solver treats such clauses as if their LBD is 50 when computing the global LBD average in the Glucose-restart phase.

B. Subdwarf

This particular solver is presented as a proof-of-concept solver. The criterion to be classified as core learned clauses is to be at most of LBD 1 or size 5. Note that clauses can never be learned with the LBD value of 1 initially. The criterion for clauses to be able to remain in the mid-tier is slightly relaxed: as long as and only if the clauses have been involved within the past 50,000 conflict analyses (increased from 30,000). The solver also disables the hybrid approach to utilize the Glucose-style restarts only. As a consequence of disabling the hybrid approach, Subdwarf is expected to have improved performance on unsatisfiable problem instances with reduced effectiveness on satisfiable instances.

III. Incremental SAT Solving

COMiniSatPS has been adopted to support full preprocessing of MiniSat in the context of incremental SAT solving by implementing incremental variable elimination [6].

However, for the competition versions, incrementally adding clauses that contain already eliminated variables will always revive the eliminated variables instead of attempting to eliminate the variables. Another unique strategy instituted for incremental SAT is the extreme clause cleaning at the end of each incremental solving run. Inspired by the performance of Subdwarf in the previous section, the incremental versions of COMiniSatPS clear the majority of learned clauses after each incremental run. Specifically, all core clauses whose size is greater than 5 and not used within the past 30,000 conflicts are cleaned. All mid-tier and local clauses are removed as well.

A. 1Earth

The 1Earth version presents a reference implementation of incremental variable elimination on top of MiniSat and demonstrates the incremental solving capability of COMiniSatPS with pre-processing enabled. However, the solver does not target to solve a long series of many easy incremental problems. Rather, the solver is suitable for solving reasonably difficult problems in an incremental fashion.

B. 1Sun

The hybrid strategy of COMiniSatPS is suitable for solving hard SAT problems. However, the hybrid strategy has a disadvantage that it usually requires considerably more time when solving very easy problems. This is due to the hybrid nature of phase switching that requires some initial time frame to stabilize the search by completing several full cycles of two alternating phases. For this reason, the 1Sun version disables the hybrid strategy to use the Glucose-style restarts only. Restart blocking [5] as in Glucose is re-introduced.

IV. Availability and License

Actual source code for all versions presented in this paper has been submitted to the competition. COMiniSatPS uses the same MIT license as MiniSat's.

References

[1] N. Eén and N. Sörensson, "An extensible SAT-solver," in *SAT*, 2003.

[2] C. Oh, "Between SAT and UNSAT: The fundamental difference in CDCL SAT," in *SAT*, 2015.

[3] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *IJCAI*, 2009.

[4] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *DAC*, 2001.

[5] G. Audemard and L. Simon, "Refining restarts strategies for SAT and UNSAT," in *CP*, 2012.

[6] J. Ezick, J. Springer, T. Henretty, and C. Oh, "Extreme SAT-based constraint solving with R-Solve," in *HPEC*, 2014.