RSat 2.0: SAT Solver Description

Knot Pipatsrisawat and Adnan Darwiche
{thammakn,darwiche}@cs.ucla.edu

University of California, Los Angeles

1 Introduction

RSat 2.0 is a DPLL-based complete SAT solver that employs many modern techniques such as those used in MiniSat [3] and Chaff [8]. RSat 2.0 is an improved version of RSat [10], which won the third place in the SAT-Race 2006 competition [11]. While RSat 2.0 is designed to perform best on industrial SAT instances, we expect the solver to exhibit reasonable efficiency on other categories of instance as well. This document briefly describes the important techniques used in RSat 2.0.

2 Basic Algorithms

RSat 2.0 uses the 2-watched literal scheme [8], which is very critical to the efficient Boolean constraint propagation. Moreover, Boolean constraint propagation mechanism of RSat 2.0 is boosted with a heuristic for ordering implications to be processed. This mechanism helps reduce the amount of work the solver has to do to detect each conflict. The idea is largely based on the work in [6].

RSat 2.0 utilizes conflict clause learning with conflict clause minimization technique [1] that helps reduce the size of learned clauses. Learned clauses are deleted based on their usefulness in recent resolution proofs. Learn clause deletion is performed once the number of clauses reaches a slowly increasing limit.

RSat 2.0 periodically restarts to counter the heavy-tailed effects common in combinatorial search [4]. The precise restarting policy used will be described in more details in Section 4.

3 New Component Caching Scheme

Similar to RSat, RSat 2.0 utilizes a lightweight component caching scheme called progress saving, which is described in great details in [9]. This component caching scheme helps reduce the amount of work repetition that is inherent in backtracking SAT solvers.

The caching scheme used by RSat 2.0 is a refinement of the original technique used by RSat in the SAT-Race 2006 competition. In particular, progress saving is occasionally disabled in the hope to prevent the caching scheme from getting stuck in a cycle of bad assignments.

4 Improved Restarting Policy

An improved restarting policy is used in RSat 2.0. This new policy is largely motivated by the work in [5]. According to the policy, the limit on the number of conflicts experienced by the solver before it has to restart is set according to the Luby's series [7]. In RSat 2.0, the Luby's unit is set to 512.

5 Preprocessor

RSat 2.0 takes advantage of the SatElite preprocessor [2] used by the winner of the last SAT competition [12]. Empirical results have shown that RSat 2.0 benefits from the integration with SatElite, even though SatElite shows some difficulties on instances with too many clauses and literals. In the event that SatElite cannot properly handle the input, RSat 2.0 continues to solve the original input.

References

1. Beame, P., Kautz, H., and Sabharwal, A. Understanding the power of clause learning. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003)* (2003).
2. Eén, N., and Biere, A. Effective preprocessing in sat through variable and clause elimination. In *International Conference on Theory and Applications of Satisfiability Testing (SAT)* (2005).
3. Eén, N., and Sörensson, N. Minisat a sat solver with conflict-clause minimization. *The International Conference on Theory and Applications of Satisfiability Testing* (2005).
4. Gomes, C. P., Selman, B., and Crato, N. Heavy-tailed distributions in combinatorial search. In *Principles and Practice of Constraint Programming* (1997), pp. 121–135.
5. Huang, J. The effect of restarts on the efficiency of clause learning. In *AAAI-06 Workshop on Learning for Search* (2006).
6. Lewis, M. D. T., Schubert, T., and Becker, B. W. Early conflict detection based bcp for sat solving. *The International Conference on Theory and Applications of Satisfiability Testing* (2004).
7. Luby, M., Sinclair, A., and Zuckerman, D. Optimal speedup of las vegas algorithms. In *Israel Symposium on Theory of Computing Systems* (1993), pp. 128–133.
8. Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., and Malik, S. Chaff: Engineering an efficient sat solver. *39th Design Automation Conference (DAC)* (2001).
9. Pipatsrisawat, K., and Darwiche, A. A lightweight component caching technique for satisfiability solvers. Paper submitted to SAT'07.
10. Rsat sat solver homepage. http://reasoning.cs.ucla.edu/rsat.
11. SAT-RACE'06 homepage. http://fmv.jku.at/sat-race-2006/.
12. SAT'05 Competition Homepage, http://www.satcompetition.org/2005/.