The SAT Solver MXC, version 0.75

(2008 SAT Race Version)

David R. Bregman and David G. Mitchell
Simon Fraser University
drb@sfu.ca, mitchell@cs.sfu.ca

April 15, 2008

1. Introduction

MXC is a complete, clause-learning SAT solver, written in C++. Development of MXC began in 2006, primarily to have an in-house solver to support the research project described in [6, 7]. Since then, we have been keeping MXC up to date with recent developments in "industrial" SAT solver algorithms, and submitting to the competitions. The first released version, MXC v. 0.1 [2], was entered in the 2006 Sat Race where it received the "Best Student Solver" award. The second version, MXC v. 0.5 [3], was submitted to the 2007 Sat Solver Competition, and received a Bronze medal in the handmade category. MXC is open-source, and may be obtained from our website: *http://www.cs.sfu.ca/research/groups/mxp/MXC/*.

2. New in version 0.75

Since the Sat 2007 competition, MXC has been significantly improved. The high-level algorithm is now implemented as "repeated probing" [5], which is perhaps more natural than DPLL as a way to think about the operation of modern clause-learning SAT solvers. SatELite preprocessing [4] has been introduced, as well as progress caching [8] and Biere's version of "nested rapid restarts" (See [1] and the discussion of nested restart strategies on the Sat Gory Details web site [9]). The nested rapid restarts are very effective on many "industrial" instances, but often hurt performance on others. Noticing this, were motivated to introduce a way to decide at runtime whether or not to enable them.

3. A Simple Classification Heuristic

One of our goals for MXC is to have it perform well over a wide variety of instances from applications. We have observed that the best setting of MXC parameters, and especially the restart strategy, varies for families of instances with different properties. In MXC v. 0.75 we implemented a simple strategy which chooses between "fast" (PicoSAT style) and "slow" (MiniSat style) restarts, for each instance. The strategy was obtained by using Weka [10], to learn an instance categorization function based on simple syntactic and density-related features. The training set consisted of some 300 "industrial" instances, on which the learned function is about 80% correct.

References

[1] A. Biere. A short history on sat solver technology and what is next? Invited Talk, 10th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT'07). See http://fmv.jku.at/biere/talks/., 2007.

[2] David R. Bregman and David G. Mitchell. The SAT solver MXC, version 0.1. Solver Description for the SAT Race 2006 solver competition., 2006.

[3] David R. Bregman and David G. Mitchell. The SAT solver MXC, version 0.5. Solver Description for the 2007 SAT Solver Competition., 2007.

[4] N. Een and A. Biere. Effective preprocessing in SAT through variable and clause elimination. In *In, Proc. 8th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT'05)*. Springer, 2005. Lecture Notes in Computer Science (LNCS), volume 3569.

[5] Jinbo Huang. A case for simple SAT solvers. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP '07)*, pages 839–46, 2007. Lecture Notes in Computer Science (LNCS), volume 4741.

[6] D. Mitchell and E. Ternovska. A framework for representing and solving NP search problems. In *Proc. AAAI'05*, pages 430–435, 2005.

[7] D. Mitchell, E. Ternovska, F. Hach, and R. Mohebali. Model expansion as a framework for modelling and solving search problems. *SFU technical report*, TR 2006-24, 2006.

[8] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In *Proc., Theory and Applications of Satisfiability Testing (SAT'07)*, pages 294–299, 2007. Lecture Notes in Computer Science Volume 4501.

[9] SAT gory details. http://www.satcompetition.org/gorydetails/, 2007.

[10] Ian H. Witten and Eibe Frank. Data mining: Practical machine learning tools and techniques. See: http://www.cs.waikato.ac.nz/ml/weka/, 2005.