The SAT Solver MXC, Version 0.5

(2007 SAT Solver Competition Version)

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

January 31, 2007

Introduction

MXC is a complete clause-learning SAT solver. The first version of MXC [1] was submitted to SAT-Race 2006 (see http://fmv.jku.at/sat-race-2006/). It received the prize for best student solver, in spite of an unfortunate glitch in the interaction between MXC and the SAT-Race platform that resulted in much poorer performance in the race than on local machines. The current version of MXC is significantly modified from the SAT-Race 2006 version, and also incorporates two non-standard features: a multi-threaded mode and extension of the input language to CNF plus cardinality constraints. MXC is open-source, and may be obtained from http://www.cs.sfu.ca/research/groups/mxp/MXC/.

Design of MXC relative to MiniSat

When run as a single-threaded solver for CNF formulas, MXC Version 0.5 is very similar to MiniSat Version 2.0 (See www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/ for details of MiniSat), with the primary differences being:

1. **Pre-processing:** MXC does no pre-processing of the formula.
2. **Randomization:** MXC has no randomization.
3. **Breadth first strengthening:** The strengthening procedure (also known as conflict clause minimization) is performed breadth-first, by using a queue instead of a stack. We have not carefully evaluated the effects of this difference yet. (It may be interesting see if it affects the number of "cache hits" to the history vector, which is used to determine if a node in the reason graph has already been visited.)
4. **Rapid decay of learnt clause scores:** The constant for decaying learnt clause scores is about 10 times that used in MiniSat. This has the effect of throwing away very different learnt clauses than MiniSat during the learnt purge cycle. This appears to be a good choice for MXC, for reasons that are unclear but perhaps related to the combination with (5).
5. **Literal sorting:** Literals in newly learnt clauses are sorted by decision level. This improves MXC performance. We hypothesize that this, combined with details of our unit propagation, forms some sort of effective heuristic. Scanning a clause for a new watch is done linearly from one end to the other, in the direction of highest to lowest decision level, so a new watch is likely to be from a variable that is recent in the decision tree.
6. **Options:** Several options can be enabled:
   (a) **Sign-scores:** An extension to VSIDS heuristic in which a score is kept for each literal rather than each variable. The score of a variable v is defined to be the sum of its sign-scores: score(v) + score($\neg v$). After choosing a decision variable, it is assigned to the sign with the highest score.

(b) **Pseudo-heap:** A heap structure with a standard array-based implementation, except the remove-max operation is replace by a linear scan through the array for the first unassigned literal. The effect of this is that near the top it behaves very much like a heap, and toward the bottom it devolves into randomness.

(c) **Propagation ordered by scores:** As in RSAT [4], instead of propagating units in the order in which they are discovered, they may be propagated in priority order by their VSIDS score. This heuristic appears to be very effective on certain types of instances.

(d) **Unbounded strengthening:** Allowing the learnt clause strengthening procedure to run to completion. This is very expensive in general, but may sometimes be desirable.

Each of these options have been found to significantly increase performance on specific types of instances. The options under which MXC performs best overall on the SAT-Race 2006 instances are those used in MiniSat: decisions are always negative; a true heap is used for the decision heuristic; units are propagated in the order discovered; and strengthening is bounded. These are the parameters used in the 2007 SAT Solver Competition version of MXC.

Non-standard Capabilities of MXC Version 0.5

The primary goal in developing MXC is to provide a propositional engine to support a general framework for representing and solving search problems [2, 3] (see http:/www.cs.sfu.ca/research/groups/mxp). For this purpose, we want a high-performance ground solver which has a number of features not found in standard SAT solvers, including a more general input language than CNF formulas. It can be a challenge to extend a solver in such ways without losing the good performance of the core algorithm. While allowing that the extensions implemented in MXC 0.5 are not very radical, one might measure how well we are meeting this challenge so far by the performance of MXC as a pure SAT solver. The version submitted to the 2007 SAT Solver Competition has the following non-standard features.

1. **Multi-threaded MXC** MXC can be be run in multi-threaded mode, on either multi-processor machines of multi-core processors. In the current version, each thread is a copy of the MXC algorithm. All threads share all learned unit clauses via a blackboard.

2. **MXC Cardinality Constraints** MXC extends DIMACS CNF format with cardinality constraints of the following syntactic form:

$$l \; u \; l_1 \; l_2 \ldots l_k \; 0$$

with the semantics that the number of true literals among $l_1 \ldots l_k$ must be at least l and at most u.

Reports on the performance of these features of MXC will be forthcoming.

References

[1] David R. Bregman and David G. Mitchell. The SAT solver MXC, v.1, 2006. Solver description for SAT-Race 2006.

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

[3] David Mitchell, Eugenia Ternovska, Faraz Hach, and Raheleh Mohebali. Model expansion as a framework for modelling and solving search problems. Technical Report TR 2006-24, Simon Fraser University, School of Computing Science, December 2006.

[4] Thammanit Pipatsrisawat and Adnan Darwiche. SAT solver description: Rsat, 2006. Solver description for SAT-Race 2006.