

The SAT Solver MXC, Version 0.99

(2009 SAT Competition Version)

David R. Bregman
Simon Fraser University
drb@sfu.ca

March 11, 2009

1. Introduction

MXC is a complete, clause-learning SAT+Cardinality solver, written in C++. MXC is open source, and may be obtained at <http://www.cs.sfu.ca/research/groups/mxp/MXC/>. MXC accepts an extended version of the DIMACS CNF format which contains cardinality constraints interleaved with regular clauses. Unit propagation on cardinality constraints is implemented using a simple counting based method. When a cardinality constraint is used as an antecedent or a conflict, some heuristics are applied to extract a weaker clausal version, which is then used in the standard learning process. The SAT part of the solver is relatively standard, using the two watched literals scheme (with occurrence stacks [5]) for unit propagation, 1-UIP cuts for clause learning, conflict clause minimization [2], activity based variable ordering and clause deletion, progress caching [4], and aggressive nested restarts [5]. When functioning in pure SAT mode (i.e. if no cardinality constraints are present) then the SatELite algorithm [3] is used for preprocessing. The high-level search algorithm itself is implemented as “repeated probing” [6].

2. History

Development of MXC began in 2006, to support the MX project (see: <http://www.cs.sfu.ca/research/groups/mxp/>). The first version, MXC 0.1, received the “best student solver” award at Sat Race '06. The second version, MXC 0.5, received a bronze medal in the “handmade” category at Sat Competition '07, and the third version, MXC 0.75, placed 5th in Sat Race '08, the highest ranking by an open-source solver (as of the date of writing, the top 4 entries remain unavailable for download).

Solver	Solved (out of 100)	Total time (min.)
MXC 0.1 (sat race '06)	49	976.8
MXC 0.5 (sat comp '07)	66	709.0
MXC 0.75 (sat race '08)	82	444.3
MXC 0.99 (sat comp '09)	85	380.3
Minisat 2.0 Beta	71	668.5

Figure 1: performance comparison of MXC 0.99 and previous versions of MXC on the Sat Race '06 benchmark set, with a 15 minute time limit. Instances that time out count the full 15 minutes towards the total time. Minisat 2.0 Beta is included as a point of reference. All tests were run on a 2.4 GHz Opteron 250 with 1MB L2 cache.

3. New in MXC 0.99

Adaptive restart control.

In MXC 0.75 [1], a classification heuristic was introduced to control restart frequency. The classifier was learnt using the Weka machine learning suite on a training set of 300 instances. In MXC 0.99, this is replaced by Biere's ANRFA heuristic [7], which achieves roughly the same result using less computational resources and requiring no training.

Blocking literals.

It has been noticed (independently?) by several authors (e.g. [8,9]) that on typical industrial instances, when scanning a clause for a new watch, the clause is already satisfied by the first literal with high probability - as often as 50-90% of the time. Because visiting the clause requires following a pointer, there will often be an expensive cache miss involved. If a copy of the first literal is stored with the watch, then it can be checked without dereferencing the clause. If it is true, then no additional work needs to be done. This extra literal stored with the watch is called a blocking literal. There is no requirement that it be the same as the first literal in the clause, and it is also possible to have more than one blocking literal. (MXC 0.99 uses a single blocking literal.)

Implementation details.

The source release of MXC 0.99 includes a visual studio project file, and implementation of the `getopt()` function for windows, allowing native compilation on that platform. Version 0.99 is intended to be the last "monolithic" version of MXC. Versions 1.0 and up will be modularized, allowing easy use as an API for interactive solving. Some refactoring towards that end has already been done.

References

- [1] David R. Bregman and David G. Mitchell. The SAT solver MXC, version 0.75. Solver Description for SAT Race 2008.
- [2] Niklas Een and Niklas Sörensson. MiniSat - A SAT solver with conflict-clause minimization. In Proc. SAT'05.
- [3] Niklas Een and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In Proc. SAT'05.
- [4] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In Proc. SAT'07.
- [5] Armin Biere. PicoSAT essentials. JSAT, 2008.
- [6] Jinbo Huang. A case for simple SAT solvers. In Proc. CP'07.
- [7] Armin Biere. Adaptive restart strategies for conflict driven SAT solvers. In Proc. SAT'08.
- [8] Himanshu Jain and Edmund Clarke. Sat solver descriptions: Cmusat-base and cmusat. Solver description for SAT competition 2007.
- [9] Geoffrey Chu and Aaron Harwood and Peter J. Stuckey. Cache Conscious Data Structures for Boolean Satisfiability Solvers. JSAT, 2009. (to be published)