ManySAT 1.1: solver description

Youssef Hamadi\textsuperscript{1,2}, Said Jabbour\textsuperscript{3}, and Lakhdar Sais\textsuperscript{4}

\textsuperscript{1} Microsoft Research, 7 J J Thomson Avenue, Cambridge, United Kingdom
\texttt{youssefh@microsoft.com}
\textsuperscript{2} LIX École Polytechnique, F-91128 Palaiseau, France
\texttt{said.jabbour@inria.fr}
\textsuperscript{3} Microsoft-INRIA joint-lab, 28, rue Jean Rostand, 91893 Orsay, France
\texttt{sais@cril.fr}

Overview

ManySAT is a parallel DPLL-engine which includes all the classical features like two-watched-literal, unit propagation, activity-based decision heuristics, lemma deletion strategies, and clause learning [4, 5]. In addition to the classical first-UIP scheme, it incorporates a new technique which extends the classical implication graph used during conflict-analysis to exploit the satisfied clauses of a formula [1].

When designing ManySat we decided to take advantage of the main weakness of modern DPLLs: their sensitivity to parameter tuning. For instance, changing parameters related to the restart strategy or to the variable selection heuristic can completely change the performance of a solver on a particular problem. In a multi-threading context, we can easily take advantage of this lack of robustness by designing a system which will run different incarnation of a core DPLL-engine on a particular problem. Each incarnation would exploit a particular parameter set and their combination should represent a set of orthogonal strategies.

To allow ManySAT to perform better than any of the selected strategy, conflict-clause sharing was added. Technically, this is implemented through lockless shared data structures. The version 1.1 implements innovative dynamic clause sharing policies [6].

Code

The system is written in C++ and has about 4000 lines of code. It is written on top of minisat 2.02 [3], which was extended to accommodate the new learning scheme, the various strategies, and our multi-threading clause sharing policy. SatElite is systematically applied as a pre-processor [2].

References


