GLUCOSE 2.3: Small improvements – Too finish

Gilles Audemard
Univ. Lille-Nord de France
CRIL/CNRS UMR8188
Lens, F-62307
audemard@cril.fr

Laurent Simon
Univ. Paris-Sud
LRI/CNRS UMR 8623 / INRIA Saclay
Orsay, F-91405
simon@lri.fr

Abstract

GLUCOSE is based on a the scoring scheme for clause learning mechanism introduced in [Audemard and Simon, 2009]. This short competition report summarizes the techniques embedded in the SAT challenge 2013 version of GLUCOSE[1].

1 Introduction

In the so-called "modern" SAT solvers [Moskewicz *et al.*, 2001; Eén and Sörensson, 2003], a lot of effort has been put in the design of efficient Boolean Constraint Propagation (BCP), learning mechanisms, and branching heuristics, their three main composants. In [Audemard and Simon, 2009], a new simple measurement of learnt clause usefulness was introduced, called LBD. This measure was no more based on past clauses activities. It was proved so efficient that, this year, we improved the overall architecture of GLUCOSE 1.0 (used in the SAT 2009 competition) to incorporate an even more aggressive database cleanup policy. For this, we had to incorporate a simple auto-adaptative threshold point beyond of which clauses are deleted. We also incorporate a "second chance" mechanism to keep bad clauses alive during one more round of database cleanings, if it shows any interesting improvement in its score.

This new version of GLUCOSE is also based on the version 2.2 of MINISAT [Eén and Sörensson, 2003] (GLUCOSE 1.0 was based on the previous version of MINISAT). For a more comprehensive description of GLUCOSE, please refer to [Audemard and Simon, 2009] and our previous competition (2009, 2011, 2012) reports.

2 Literal Block Distance and Glue Clauses

During search, each decision is often followed by a large number of unit propagations. We called the set of all literals of the same level a "blocks" of literals. Intuitively, at the semantic level, there is a chance that they are linked with each other by direct dependencies. The underlying idea developed in [Audemard and Simon, 2009] is that a good learning schema should add explicit links between independent blocks of propagated (or decision) literals. If the solver stays in the same search space, such a clause will probably help reducing the number of next decision levels in the remaining computation. Staying in the same search space is one of the recents behaviors of CDCL solvers, due to phase-saving [Pipatsrisawat and Darwiche, 2007] and rapid restarts.

Definition 1 (Literals Blocks Distance (LBD)) *Given a clause C, and a partition of its literals into n subsets according to the current assignment, s.t. literals are partitioned w.r.t their decision level. The LBD of C is exactly n.*

From a practical point of view, we compute and store the LBD score of each learnt clause when it is produced. Intuitively, it is easy to understand the importance of learnt clauses of LBD 2: they only contain one variable of the last decision level (they are FUIP), and, later, this variable will be "glued" with the block of literals propagated above, no matter the size of the clause. We suspect all those clauses to be very important during search, and we give them a special name: "Glue Clauses".

The LBD measure can be easily re-computed on the fly when the clause is used during unit propagation. We keep here the strategy used in GLUCOSE 1.0: we change the LBD value of a clause only if the new value becomes smaller.

3 Agressive clauses deletion.. when possible

Before GLUCOSE 1.0, the state of the art was to let the clause database size follow a geometric progression (with a small common ratio of 1.1 for instance in MINISAT). Each time the limit is reached, the solver deleted at most half of the clauses, depending on their score (note that binary and glue clauses are never deleted). In GLUCOSE 1.0, we already chose a very slow increasing strategy. In this new version, we perform a more accurate management of learnt clauses.

3.1 Dynamic threshold

References

[Audemard and Simon, 2009] G. Audemard and L. Simon. Predicting learnt clauses quality in modern sat solvers. In *IJCAI*, 2009.

[Eén and Sörensson, 2003] N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT*, pages 502–518, 2003.

[1] Web page `http://www.lri.fr/~simon/glucose`

[Moskewicz *et al.*, 2001] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff : Engineering an efficient SAT solver. In *DAC*, pages 530–535, 2001.

[Nabeshima *et al.*,] Hidetomo Nabeshima, Koji Iwanuma, and Katsumi Inoue. Glueminisat2.2.5. In *SAT 2011 solvers descriptions*.

[Pipatsrisawat and Darwiche, 2007] K. Pipatsrisawat and A. Darwiche. A lightweight component caching scheme for satisfiability solvers. In *SAT*, pages 294–299, 2007.