GLUCOSE 1.1: a solver that predicts learnt clauses quality[*].

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

George Katsirelos
Univ. Lille-Nord de France
CRIL/CNRS UMR8188
Lens, F-62307
gkatsi@gmail.com

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

Abstract

GLUCOSE is based on a new scoring scheme for the clause learning mechanism, based on the paper [Audemard and Simon, 2009]. This short competition report summarizes the techniques embedded in the competition 09 version of GLUCOSE. Solver's name comes from *glue clauses*, a particular kind of clauses that GLUCOSE detects and preserves during search. The web page for GLUCOSE is `http://www.lri.fr/~simon/glucose`.

1 Introduction

Since the breakthrough of Chaff [Moskewicz *et al.*, 2001], a lot of effort has been made in the design of efficient Boolean Constraint Propagation (BCP), the heart of all modern SAT solvers. The global idea is to reach conflicts as soon as possible, but with no direct guarantees on the new learnt clause usefulness. Following the successful idea of the Variable State Independent Decaying Sum (VSIDS) heuristics, which favours variables that were often – and recently – used in conflict analysis, future learnt clause usefulness is supposed to be related to its activity in recent conflicts analyses.

In this context, detecting what is a good learnt clause in advance was still considered as a challenge, and from first importance: deleting useful clauses can be dramatic in practice. To prevent this, solvers have to let the maximum number of learnt clauses grow exponentially. On very hard benchmarks, CDCL solvers hangs-up for memory problems and, even if they don't, their greedy learning scheme deteriorates their heart: BCP performances.

If a lot of effort has been put in designing smart restart policies, only a few work targetted smart clause database management. In [Audemard and Simon, 2009], we show that a very simple static measure on clauses can dramatically improves the performances of MINISAT [Eén and Sörensson, 2003], the solver on which GLUCOSE is based. GLUCOSE is based on the last plubicly available version of MINISAT.

[*]supported by ANR UNLOC project n° ANR-08-BLAN-0289-01

2 Identifying good clauses in advance

During search, each decision is often followed by a large number of unit propagations. All literals from the same level are what we call "blocks" of literals in the later. Intuitively, at the semantic level, there is a chance that they are linked with each other by direct dependencies. Our idea 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. This measure is thus static, even if update it during search (LBD score of a clause can be re-computed when the clause is used in unit-propagation). 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".

From a theoretical point of view, it is interesting to notice that LBD of FUIP learnt clauses is optimal over all other possible UIP learning schemas [Jabbour and Sais, 2008]. If GLUCOSE efficiency in the 2009 competition clearly demonstrates our scoring accuracy, this theoretical result will cast a good explanation of the efficiency of First UIP over all other UIP mechanisms: FUIP efficiency would then be partly explained by its ability to produce clauses of small LBD (in addition to its optimality in the size of the backjump [Jabbour and Sais, 2008]).

Property 1 (Optimality of LBD for FUIP Clauses) *Given a conflict graph, any First UIP asserting clause has the smallest LBD value over all other UIPs.*

3 Agressive clauses deletion

Despite its crucial importance, only a few works focus on the learnt clause database management. However, keeping too many clauses may decrease solver BCP performances, but deleting too many clauses may decrease the overall learning benefit. Nowadays, the state of the art is 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 deletes at most half of the clauses, depending on their score (however, binary clauses are never deleted).

In glucose 1.0, we chose the following strategy: every $20000 + 500 * x$ conflicts, we remove at most half of the learnt clause database where x is the number of times this action was already performed before. It can be noticed that this strategy does not take the initial size of the formula into account (as opposite of most current solvers). Our first hope was only to demonstrate that even a static measure on clause usefulness could be *as efficient* as the past-activity one. However, our results were far beyond our initial hope.

In this new version, we perform a more agressive clause reduction. We use the LBD as first criteria and the clause activity as the second one in case of equality. This is the main difference between both versions.

4 Other embedded techniques

The GLUCOSE version submitted to the contest differs from the one used in [Audemard and Simon, 2009] on some very particular points that we review here.

First of all, we upgraded MINISAT for a special handling of binary clauses. We also used a phase-caching schema for variable polarity [Pipatsrisawat and Darwiche, 2007].

4.1 Restarts

One of the best restart strategy in CDCL solver is based on the Luby series, which exactly means that "we don't know when to restart". Recently, first steps have been done to find a dynamic (computed during the search) restart strategy [Biere, 2008; Ryvchin and Strichman, 2008]. Our restart strategy is based on the decreasing of the number of decisions levels during search. If the decreasing is stalling, then a restart is triggered. This is done by a moving average over the last 100 conflicts. If 0.7 times this value is greater than the global average of the number of decision levels, then a restart is forced (at least 100 conflicts are needed before any restart). This strategy should encourage the solver to keep searching at the right place, and to escape from wrong places.

4.2 Reward good variables

The state-of-the-art VSIDS [Moskewicz *et al.*, 2001] heuristic bumps all variables which participated to the resolution steps conducting to the assertive clause. This heuristic favors variables that are often and recently used in conflict analysis. Since we want to help the solver to generate clauses with small LBD values, we propose to reward a second time variables that help to obtain such clauses.

We bump once again all variables from the last decision level, which were used in conflict analysis, but which were propagated by a clause of small LBD (smaller than the new learnt clause).

5 Conclusion

GLUCOSE is based on a relatively old version of MINISAT, which is very well known, and well established. Only a relativaly small amount of changes has been made in MINISAT: we tried to reduce the modifications as much as possible in order to identify what are the crucial techniques to add to a 2006 winning code to win the UNSAT category of the 2009 SAT competition. A lot of improvements can be awaited by more up-to-date datastructures (like the use of blocked literals).

References

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

[Biere, 2008] A. Biere. Adaptive restart strategies for conflict driven sat solvers. In *proceedings of SAT*, pages 28–33, 2008.

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

[Jabbour and Sais, 2008] S. Jabbour and L. Sais. personnal communication, February 2008.

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

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

[Ryvchin and Strichman, 2008] V. Ryvchin and O. Strichman. Local restarts. In *proceedings of SAT*, pages 271–276, 2008.