Glucose and Syrup in the SAT Race 2015

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

Laurent Simon
Univ. Bordeaux
LABRI
lsimon@labri.fr

I*Abstract*—**Glucose is a CDCL solver heavily based on Minisat, with a special focus on removing useless clauses as soon as possible, and an original restart scheme. Syrup is the parallel version of Glucose, with a lazy clauses exchanges policy. In the 2015 version of these solvers, we proposed a genuine version and an "adaptative" version of each of them. The adaptative versions use a set of particular parameters and techniques to adress some outliers benchmarks that can be found in typical competitions sets.**

I. Introduction

Since 2009, Glucose enters SAT competitions/races [1], [2]... Glucose is based on minisat [3] and depends heavily on the concept of Literal Block Distance, a measure that is able to estimate the quality of learnt clauses [4], and when to fire a restart. Indeed, learnt clauses removal, restarts, small modifications of the VSIDS heuristic are based on the concept of LBD. The core engine of Glucose (and Syrup) is 6 years old.

This year, we proposed two versions of each solvers (sequential and parallel). The default ones have no novelty. They are exactly the same solvers as last year. The main contribution come from the "adapt" version of both solvers. The typical set of benchmarks have greatly changed since 2009. It contains now a high diversity of different problems (BMC, Crypto, ...) that does not seem to share any structural properties. We observed that some techniques are well suited for some families of benchmarks only.

II. Adaptative solver

Selected benchmarks of the SAT competition come from many distinct domains. For example, in 2014, industrial benchmarks could be assigned in (at least) nine families like argumentation, io, crypto, diagnosis... It seems unrealistic to design one strategy that will be efficient on all the benchmarks. For instance, Glucose is known to perform better on UNSAT than on SAT instances. On the other side, it is known that long runs (without restarts) are efficient on some families of SAT instances. The works of [5] and [6] are also the motivations of these adaptations.

A. Adaptation in Glucose

A number of recent solvers includes, directly or not, automatic adaptations to benchmarks. In our approach, we used our set of experimental data to classify some strategies adapted to outliers benchmarks. We took 2632 benchmarks from all the competitions, and selected only 1164 interesting ones (benchmarks that needed at least one minute to be solved). We ran a set of Glucose "hacks" on this set of problems and tried to detect some simple measures that identified families of problems. We tried to consider only some "semantic" measure instead of syntactic measures on the initial formula. The adaptation is run after 10,000 conflicts during which Glucose run with its default parameters. We then may switch to some particular behavior if our indicators say so. We searched for simplicity, and we identifed 4 outliers signatures.

- The number of decisions divided by the number of conflicts. This allows us to identify 123 problems over the 1164, containing bivium, hitag, gss, homer, ctl and longmult series of problems. If this number is low, we switch the reduction learnt clauses strategy by using the one proposed by Chanseok Oh [5].
- The number of conflicts without decision (when a conflict is directly reached after a conflict analysis). If this number is low, this is typically a nossum crypto problem. We identified 66 problems from the 1164 ones like that. For these problems, we used a Luby restart policy, and a much less agressive var decay. In the contrary, if this number is important, then we use the Chanseok Oh policy [5] to reduce the learnt clause database, a much less agressive var decay, and a limited randomization on the first descent after each restart (in the spirit of [6]). In this last case, we typically identified vmpc problems.
- The number of "pure" glue clauses (glue clauses of size > 2). A large number is a typical signature of SAT_dat problems (we identified 31 of them with that, over the 1164). In this case, we observed that a much more aggressive var decay may pay.

We observed an important increasing of Glucose performances on the last competitions by using this. In the SAT competition 2014, among the 300 instances of the application category, glucose adjust its parameters on 58 instances and benefits are clears.

III. Specificities of the parallel version

We decided to run syrup on eight cores in the first phase and limit to 32 cores in the second phase. Furthermore, in this case, we decided to limit clauses sharing in order to keep memory (the previous version of Syrup was only tested up to 8 cores). The adaptative version of Syrup only changes the behavior of half of its cores. This was made without real experimental

evidences: we had no access to clusters with 32 cores to check our solutions.

IV. Incremental track

Glucose also entered the incremental part of the SAT-Race. In this case, it uses dedicated data-structures and techniques introduced in [7]. Unfortunately, in the incremental track, the rules were not in favor of our specialized data structure. It was not possible to know the initial variables and the variables added for the search (commonly called the "assumptions", for example variables added to simulate clauses removals). Thus, all the strategies proposed in [7] are useless here.

V. Algorithm and Implementation details

Glucose uses a special data structure for binary clauses, and a very limited self-subsumption reduction with binary clauses, when the learnt clause is of interesting LBD.

References

[1] G. Audemard and L. Simon, "Glucose: a solver that predicts learnt clauses quality," *SAT Competition*, pp. 7–8, 2009.

[2] ——, "Glucose 2.3 in the sat 2013 competition," *Proceedings of SAT Competition*, pp. 42–43, 2013.

[3] N. Eén and N. Sörensson, "An extensible SAT-solver," in *SAT*, 2003, pp. 502–518.

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

[5] C. Oh, "gluh: Modified version of gluclose 2.1," *SAT COMPETITION 2013*, p. 48, 2013.

[6] J. Chen, "A bit-encoding phase selection strategy for satisfiability solvers," in *Theory and Applications of Models of Computation - 11th Annual Conference, TAMC 2014, Chennai, India, April 11-13, 2014. Proceedings*, 2014, pp. 158–167.

[7] G. Audemard, J.-M. Lagniez, and L. Simon, "Improving glucose for incremental sat solving with assumptions: Application to mus extraction," in *16th International Conference on Theory and Applications of Satisfiability Testing(SAT'13)*, 2013, pp. 309–317.