GlueMiniSat2.2.7

Hidetomo Nabeshima
University of Yamanashi, JAPAN

Koji Iwanuma
University of Yamanashi, JAPAN

Katsumi Inoue
National Institute of Informatics, JAPAN

***Abstract*—GlueMiniSat is a SAT solver based on MiniSat 2.2 and the LBD-based evaluation criteria of learned clauses. The new features of the version 2.2.7 are (1) on-the-fly lazy simplification techniques based on binary resolvent, (2) probing-based preprocessing, (3) a new restart strategy based on conflict-generation speed and (4) a minor modification of the evaluation criteria of learned clauses.**

I. Introduction

GlueMiniSat is a SAT solver based on MiniSat 2.2 [1] and the LBD-based evaluation criteria of learned clauses [2]. GlueMiniSat shows good performance for unsatisfiable SAT instances. The previous version 2.2.5 [3] took the first and second places for UNSAT and SAT+UNSAT classes in CPU time evaluation respectively and won the second place for UNSAT class of the category in wall clock time evaluation.

To enhance the UNSAT performance, we introduced some new features to GlueMiniSat: (1) on-the-fly lazy simplification techniques based on binary resolvents, (2) probing-based preprocessing [4], [5], (2) a new restart strategy based on conflict-generation speed and (4) a minor modification of the LBD-based evaluation criteria of learned clauses.

II. Main Techniques

Simplification of a given CNF formula is one of important techniques to decide the satisfiability of the formula efficiently. The simplification techniques are used both before and during the search process. GlueMiniSat has the both simplification techniques. For preprocessing, we have implemented probing-based techniques which consist of false-literal probing, necessary assignment probing, equivalent variable probing [4] and binary clause probing [5], besides variable and subsumption elimination [6] which are implemented in MiniSat 2.2.

For in-processing, GlueMiniSat executes the above probing techniques on-the-fly. To reduce the checking cost, we utilize binary resolvents extracted from unit propagation process. For example, let $\phi = \{x \to y, x \to z, y \wedge z \to v, v \wedge w \to u\}$ and w is assigned as true. If x is selected as a decision variable and assigned as true, then y, z, v, w are propagated. The cause of the propagation of y, z, v is x. This means $\phi \models (x \to y) \wedge (x \to z) \wedge (x \to v)$. However, w is not propagated from x only. It requires x and w as premise literals. The checking of whether a propagated literal has a single cause or not can be done with a constant order at the unit propagation process. We can extract a large number of binary resolvents with very low overhead. This extraction approach is similar to dominator detection algorithm in [7]. Our algorithm detects the earliest dominator (decision literal), whereas [7] uses immediate dominators. The earliest dominator can be detected with O(1), whereas the computation of the immediate dominator sometimes requires linear search between two nodes in a implication graph.

For each literal, GlueMiniSat holds only *one* of premise literals. We prepare an array named *premise*. Each entry of the array is indexed by each literal. The value of *premise*$[x]$ is a literal which denotes one of premise literals of x, that is, $\phi \models$ *premise*$[x] \to x$. Initially, *premise*$[x] = x$. The value of *premise*$[x]$ is updated when x is propagated and x has a single cause of the propagation.

We can execute probing techniques with a constant order by using the array *premise*. For example, the necessary assignment probing can be represented as follows: suppose that ϕ is a formula and x, y are literals. If $\phi \models x \to y$ and $\phi \models \neg x \to y$, then $\phi \models y$. This probing technique requires two premise literals of y. We can get two premise literals of y, that is, the old value of *premise*$[y]$ before updating of it and the new value of it. We denote the old and new values as $\textit{oldpremise}_y$ and $\textit{newpremise}_y$, respectively. Then, we can execute the necessary assignment probing as follows: if $\textit{oldpremise}_y = \neg\textit{newpremise}_y$, then $\phi \models y$ holds. The checking cost is $O(1)$. Other probing techniques can be executed in the same way. GlueMiniSat executes these on-the-fly probing techniques when an entry of the array *premise* is changed. The array *premise* represents a set of binary resolvents. These binary resolvents are also used to shrink clauses by self-subsumption checking.

We hold only one premise literal for each literal. However, the value of *premise*$[y]$ often changes since CDCL solver execute unit propagations very frequently. This variation of premise literals contributes the realization of effective and low cost simplification techniques.

III. Other Techniques

GlueMiniSat uses an aggressive restart strategy: if one of the following conditions is satisfied, then a restart is forced.

1) an average of *LBDs* over the last 50 conflicts is greater than the global average $\times$ 0.8.
2) the number of decisions for producing last 50 conflicts is greater than the global average $\times$ 0.95.

The former condition is same as GlueMiniSat 2.2.5 and Glucose 2.1. The latter one is a new condition which intends to generate conflicts quickly. The parameters 0.8 and 0.95 were determined by experiments on benchmark instances of past SAT competitions.

TABLE I
THE NUMBER OF SOLVED INSTANCES

Solver	#Solved (SAT + UNSAT)
GLUEMINISAT 2.2.5	199 (81 + 118)
GLUEMINISAT 2.2.7	**220** (93 + **127**)
GLUCOSE 2.1	216 (**94** + 122)

The literal blocks distance (LBD) [2] is an evaluation criteria to predict learnt clauses quality in CDCL solvers. The effectiveness of LBD was shown at past competitions by GLUCOSE and GLUEMINISAT. The LBD value of a clause is computed when the learned clause is produced from a conflict, and re-computed when the clause is used for unit propagations. As the results, the LBD values may become less than the original ones. In 2.2.7, we never remove learned clauses whose updated LBD value is *one*, that is, a learned clause is never removed when every literal of the clause are assigned at the same level once.

IV. EXPERIMENTAL RESULTS

We evaluated 3 solvers for 300 instances in the application category of SAT 2011 competition. The solvers are GLUEMINISAT 2.2.5, 2.2.7 and GLUCOSE 2.1. The experiments were conducted on a Core i7 (2GHz) with 8GB memory. We set a timeout for solvers to 5000 CPU seconds. Table I is the experimental results and Fig 1 is cactus plots of the results.

For SAT instances, GLUEMINISAT solves almost the same number of instances by GLUCOSE 2.1. For UNSAT instances, GLUEMINISAT shows the best result.

V. AVAILABILITY

GLUEMINISATis developed based on MINISAT 2.2. Permissions and copyrights of GLUEMINISAT are exactly the same as MINISAT. GLUEMINISAT can be downloaded at http://glueminisat.nabelab.org/.

ACKNOWLEDGMENT

This research is supported in part by Grant-in-Aid for Scientific Research (No. 24300007) from Japan Society for the Promotion of Science and by Artificial Intelligence Research Promotion Foundation.

REFERENCES

[1] N. Eén and N. Sörensson, "An extensible sat-solver," in *Proceedings of SAT-2003*, 2003, pp. 502–518.
[2] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *Proceedings of IJCAI-2009*, 2009, pp. 399–404.
[3] H. Nabeshima, K. Iwanuma, and K. Inoue, "GLUEMINISAT 2.2.5," 2011, SAT Competition 2011 Solver Description.
[4] D. L. Berre, "Exploiting the real power of unit propagation lookahead," *Electronic Notes in Discrete Mathematics*, vol. 9, pp. 59–80, 2001.
[5] I. Lynce and J. P. M. Silva, "Probing-based preprocessing techniques for propositional satisfiability," in *ICTAI*. IEEE Computer Society, 2003, pp. 105–110.
[6] N. Eén and A. Biere, "Effective preprocessing in sat through variable and clause elimination," in *SAT*, ser. Lecture Notes in Computer Science, F. Bacchus and T. Walsh, Eds., vol. 3569. Springer, 2005, pp. 61–75.
[7] H. Han, H. Jin, and F. Somenzi, "Clause simplification through dominator analysis," in *DATE*. IEEE, 2011, pp. 143–148.

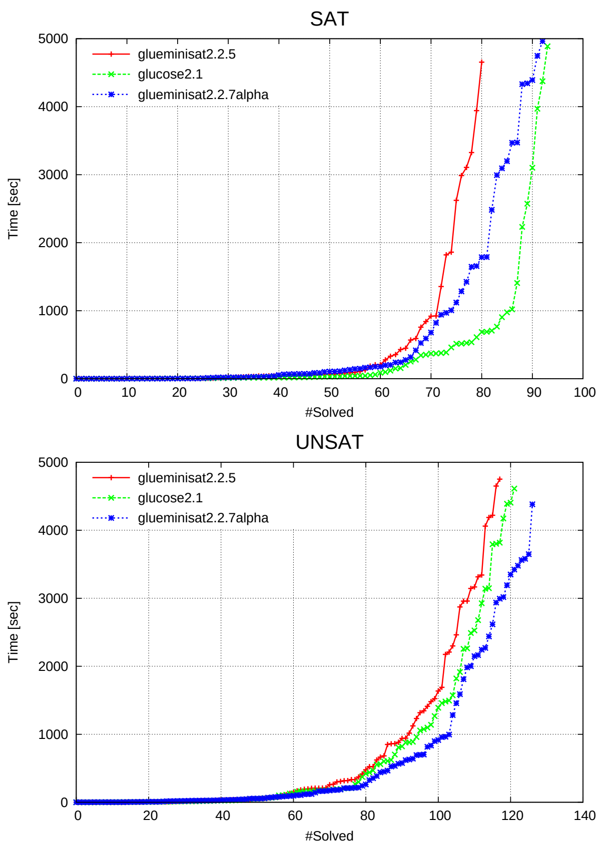

Fig. 1. A cactus plot for application category of SAT 2011 competition