EPiC Series in Computing
Volume 40, 2016, Pages 67–77
IWIL-2015. 11th International Workshop on the Implementation of Logics

On Reducing Clause Database in Glucose

Chu-Min LI[1,2], Fan Xiao[1], and Ruchu XU[1]

[1] Huazhong University of Science and Technology, China
[2] MIS, Universit de Picardie Jules Verne, France

Abstract

Modern CDCL SAT solvers generally save the variable value when backtracking. We present a measure called *nbSAT* based on the saved assignment to predict the usefulness of a learnt clause when reducing clause database in Glucose 3.0. Roughly speaking, the nbSAT value of a learnt clause is equal to the number of literals satisfied by the current partial assignment plus the number of other literals that would be satisfied by the saved assignment. The nbSAT measure is similar to a previous measure called *psm* which is not implemented in Glucose 3.0. We study the nbSAT measure by empirically showing that it may be more accurate than the LBD measure originally used in Glucose. Based on this study, we implement an improvement in Glucose 3.0 to remove half of learnt clauses with large nbSAT values instead of half of clauses with large LBD values. This improvement, together with a resolution method to keep the learnt clauses or resolvents produced using a learnt clause that subsume an original clause, makes Glucose 3.0 more effective for the application and hard combinatorial instances from the SAT 2014 competition.

1 Introduction

In propositional logic, a variable x_i may take values 0 (for false) or 1 (for true). A literal l_i is a variable x_i or its negation $\bar{x}_i$. A clause is a disjunction of literals, and a CNF formula ϕ is a conjunction of clauses. The size of a clause is the number of its literals. An assignment of truth values to the propositional variables satisfies a literal x_i if x_i takes the value 1 and satisfies a literal $\bar{x}_i$ if x_i takes the value 0, satisfies a clause if it satisfies at least one literal of the clause, and satisfies a CNF formula if it satisfies all the clauses of the formula. An empty clause represents a conflict, because it contains no literals and cannot be satisfied. A unit clause contains only one literal that should be satisfied by assigning the appropriate truth value to the variable. An assignment for a CNF formula ϕ is complete if all the variables occurring in ϕ have been assigned; otherwise, it is partial. The SAT problem for a CNF formula ϕ is the problem of finding an assignment of values to propositional variables that satisfies all clauses of ϕ.

Thanks to the progress made in developing CDCL (Conflicting-Driven Clause Learning) SAT solvers, reducing combinatorial problems to SAT becomes a powerful solving strategy. Roughly speaking, in order to solve a SAT problem, a CDCL solver repeatedly makes and propagates a decision, i.e. pick a variable using a heuristic, assign it a truth value and propagate all unit clauses implied by the decision until an empty clause is produced. Then the empty clause is analyzed and a new clause is learnt and added into the clause database. The learnt clause allows the solver to avoid the same conflict in the future and to determine the decision on which the solver should backtrack [5] [8]. If all variables are assigned a truth value without producing any empty clause, the problem is satisfiable and the complete assignment is output as a solution. Otherwise, the solver should learn an empty clause to prove the unsatisfiability of the problem.

So, the learnt clauses are essential for the performance of a CDCL SAT solver. However, the solver is slowed down when there is a large number of learnt clauses, because the solver has to check too many clauses to find unit clauses in this case during the propagation of a decision. Moreover, these learnt clauses can also overflow the memory. In order to remedy these drawbacks, a common practice when designing a CDCL solver is to measure the quality of each learnt clause using a heuristic and to periodically remove half of learnt clauses judged less useful in the future.

In this paper, we study the policies to reduce the learnt clause database in the well-known CDCL solver Glucose [3], and present a measure called nbSAT to predict the usefulness of a learnt clause. The nbSAT measure is similar to a previous measure called *psm* proposed in [2] which is not implemented in Glucose. We study the nbSAT measure by empirically showing that it may be more accurate than the LBD (Literals Blocks Distance) measure originally used in Glucose. Then we combine nbSAT with LBD in Glucose to reduce the learnt clause database. Furthermore, we implement a resolution method that detects if a learnt clause or a resolvent produced using a learnt clause subsumes an original clause, and in this case, the subsuming learnt clause or resolvent is kept and never removed.

This paper is organized as follows. Section 2 recalls the main features of Glucose 3.0 related to our work. Section 3 presents the nbSAT measure and studies its properties. Section 4 and Section 5 present different uses of the nbSAT measure in reducing clause database in Glucose 3.0. Section 6 presents the resolution method to keep the subsuming learnt clauses or resolvents. Section 7 presents experimental results. Section 8 concludes.

2 Main features of Glucose

Glucose is a very efficient CDCL-based complete SAT solver developed from Minisat [6]. It is always one of the awarded SAT solvers in SAT competitions since 2009. Our work is based on the following features of Glucose 3.0, the last sequential release of Glucose[1].

2.1 LBD: a measure of the usefulness of a learnt clause

In a CDCL SAT solver, each decision is followed by a unit propagation. The decision and all literals fixed (i.e. satisfied or falsified) during the unit propagation belong to the same level. In Glucose, these literals, as well as the decision, are said to form a “Block”. When a clause C is learnt, all literals in C are falsified under the current partial assignment, and the number of different blocks in C is called Literals Blocks Distance (LBD) of C. This measure is static in most cases, i.e., it is recomputed and updated only in special cases. The learnt clauses with LBD=2 are called “Glue Clauses” and are attached a particular importance, because they only contain one variable in the last decision level and all other variables belong to another block. It is expected that these glue clauses are frequently involved in future unit propagations and conflicts, so the glue clauses are never removed in Glucose.

It is well-known that industrial SAT instances usually exhibit a clear community structure, i.e., variables in an industrial instance form communities. The variables inside a community

[1]available at http://www.labri.fr/perso/lsimon/glucose/

have strong relations, but the relations among the variables in different communities are much weaker [1]. In [9], it is shown that LBD can be strongly related to the community structure of the initial formula, which explains why clauses with smaller LBD are more probably used in unit propagations.

2.2 Aggressive learnt clause database reduction

Since Glucose, aggressive clause database reduction policies are essential ingredients of CDCL solvers. Once the number of clauses learnt since the last database reduction reaches 2000 + 300*n, where n is the number of database reductions performed so far, the reducing process is fired: the learnt clauses are sorted in the decreasing order of their LBD, and the first half of learnt clauses are removed except binary clauses, glue clauses, and the clauses that are reasons of the current partial assignment. One consequence of this aggressive clause database reduction policy is that more than 93% of learnt clauses can be removed (see the Glucose webpage). The new measure of learnt clause usefulness we present in this paper is strongly related to the aggressive reduction policy.

2.3 Fast restart and phase saving

A CDCL based SAT solver usually uses the restart mechanism [7] to prevent heavy-tailed phenomena, every restart constructing a search tree from scratch. In Glucose 3.0, the restart policy is very aggressive. It does not depends on the learnt clause database reduction, nor on the number of conflicts reached since the last restart. It depends on the average LBD of the learnt clauses [4]. The consequence is that Glucose 3.0 is restarted only after few hundreds of conflicts in the average.

Together with the aggressive restart policy, Glucose implements phase saving [10] policy: when backtracking, Glucose saves the value of each fixed variable, and when a variable is picked up to make a decision, it is assigned the save value. In this way, Glucose stays in the same search space and benefits from the results of early restarts. The new measure of learnt clause usefulness we present in this paper is strongly related to the aggressive restart policy and the phase saving policy.

3 A Measure of Learnt Clause Usefulness: nbSAT

From a theoretical point of view, all learnt clauses are logical consequences of original clauses of a CNF formula and thus are redundant. From a practical point of view, a learnt clause is useful only if it is involved in at least one unit propagation and helps to reach a conflict. The intuition of the LBD measure in Glucose is based on the tight links between the variables in a block: once a variable is fixed, the other variables in the same block can probably be also fixed in a unit propagation because of the tight links among them, so that a clause with smaller LBD has more chance to become unit or empty. Nevertheless, when the links among the variables in a block are not so tight, other measures of the learnt clause usefulness might be more relevant.

Observe that in a CDCL solver with phase saving, a learnt clause has more chance to become unit, if all its literals would be falsified by the saved phases. For example, if the save phase of the three variables x_1, x_2 and x_3 is true, the clause $\neg x_1 \vee \neg x_2 \vee \neg x_3$ becomes unit after two of the three variables are picked as decision variables. More generally, in the search space characterized by the saved phases, a learnt clause has more chance to become unit during search if it is satisfied by fewer variables with the saved value.

Definition 1. *In a CDCL solver with phase saving, the nbSAT (short for number of satisfied literals) of a learnt clause C is the number of literals of C satisfied by the current partial assignment plus the number of literals not fixed but would be satisfied by the saved assignment.*

Proposition 1. *If a CDCL solver with phase saving makes every decision according to the saved assignment, at least one learnt clause with nbSAT=0 will become unit.*

The nbSAT measure is clearly dynamic, contrary to the LBD measure, because the saved assignment is changing upon backtracking. We have to frequently update the nbSAT value for each learnt clause. The definition of nbSAT is similar to the *psm* measure proposed in [2], except that the psm value of a learnt clause, as is described in [2], does not take the current partial assignment into account, and is the number of literals that would be satisfied by the saved assignment, no matter if these literals are actually satisfied or not by the current partial assignment. Note that the value of the variables in the current partial assignment may be different from the saved assignment if these variables are fixed by unit propagation instead of decisions in the current partial assignment. More importantly, the nbSAT measure will be exploited differently in this paper, as will be presented in Section 4.

We now empirically compare nbSAT and LBD in Glucose 3.0 (the psm measure is not compared because it is not implemented in Glucose 3.0). Following [3], we run Glucose 3.0 on the set of hard combinatorial and industrial benchmarks of the SAT competition 2014 on a computer with Intel Westmere Xeon E7-8837 of 2.66GHz and 10GB of memory under Linux with a cutoff time of 5000 seconds as in the competition. Each benchmark contains 300 instances. Before each learnt clause database reduction, the nbSAT value of each learnt clause is computed. For each nbSAT value k in {0, 1, 2, ..., 9, 10, 11+}, where 11+ represents all values equal to or bigger than 11, we measure the total number of times in which all learnt clauses with this nbSAT value are useful in unit propagation #up(k) and in conflict analysis #analyze(k), respectively. Note that the clauses learnt between the m^{th} and the $(m+1)^{th}$ clause database reductions are not counted in #up(k) and #analyze(k) before the $(m+1)^{th}$ reduction. Also note that the real nbSAT value of a learnt clause can be changed during search. However, we use its nbSAT value computed at the m^{th} clause database reduction to compute #up(k) and #analyze(k) between the m^{th} and the $(m+1)^{th}$ clause database reductions. In fact, we want to use this nbSAT value to predict its usefulness between the m^{th} and the $(m+1)^{th}$ clause database reductions.

We compute the cumulative distribution functions for the nbSAT values:

$$f_{nbSATup}(k) = \frac{\sum_{i=0}^{k} \#up(i)}{\sum_{i=0}^{11+} \#up(i)}$$

and

$$f_{nbSATanalyze}(k) = \frac{\sum_{i=0}^{k} \#analyze(i)}{\sum_{i=0}^{11+} \#analyze(i)}$$

Similarly, we compute the cumulative distribution functions for LBD values and clause sizes in {2, 3, ..., 9, 10, 11+}. Again note that clauses learnt between the m^{th} and the $(m+1)^{th}$ clause database reductions are not counted in the functions before the $(m+1)^{th}$ reduction. Generally the LBD value of a clause is not changed. It is changed only when the new LBD value is smaller. In this case, the new LBD value is taken into account when computing the cumulative distribution functions, favoring the set of learnt clauses with small LBD values. Observe that the set of learnt clauses with small LBD values is further favored in the cumulative distribution

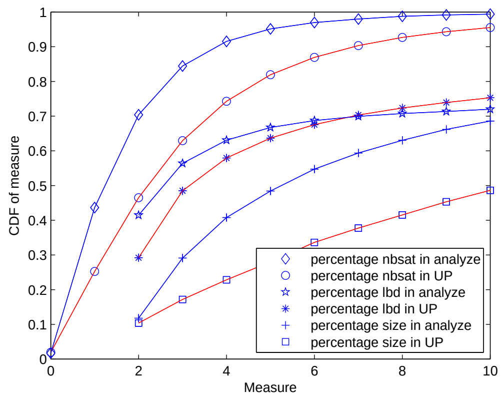

Figure 1. Cumulative distribution functions for nbSAT, LBD and clause size in glucose-3.0 on industrial benchmark of the SAT competition 2014. Names of curves are in the same ordering as the curves. Each point (x, y) represents the percentage y of learnt clauses used in unit propagation or conflict analyses with nbSAT, LBD or clause size$\leq x$.

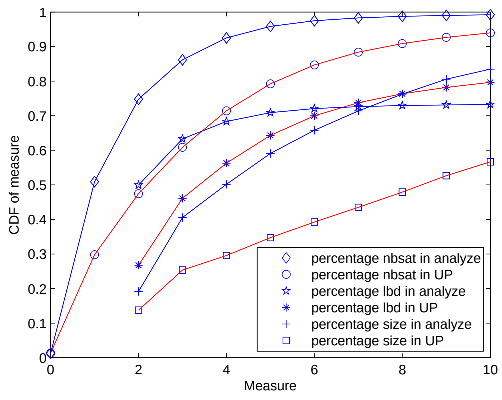

Figure 2. Cumulative distribution functions for nbSAT, LBD and clause size in glucose-3.0 on hard combinatorial benchmark of the SAT competition 2014. Names of curves are in the same ordering as the curves. Each point (x, y) represents the percentage y of learnt clauses used in unit propagation or conflict analyses with nbSAT, LBD or clause size$\leq x$.

functions in Glucose 3.0, because the clauses with large LBD values tend to be removed, which is not the case for the nbSAT measure.

Figure 1 and Figure 2 compare the cumulative distribution functions of the nbSAT and LBD values and the clause sizes for the industrial benchmark and hard combinatorial benchmark, respectively. The two figures clearly show that clauses with small nbSAT values are significantly more frequently used than the clauses with small LBD values in unit propagation and conflict analyses. In particular, more than 95% of learnt clauses used in unit propagation and conflict analyses have nbSAT $\leq$10, which is especially true for the learnt clauses used in conflict analyses.

Figure 3 and Figure 4 show the cumulative distribution functions of the number of learnt clauses with each nbSAT, LBD and clause size for the industrial benchmark and hard combi-

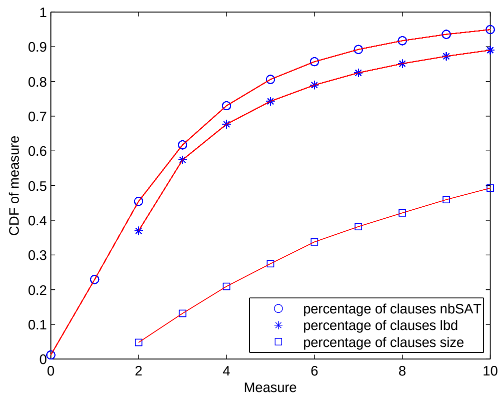

Figure 3. Cumulative distribution functions for number of learnt clauses with different nbSat, LBD and clause size in glucose-3.0 on industrial benchmark of the SAT competition 2014. Names of curves are in the same ordering as the curves. Each point (x, y) represents the percentage y of learnt clauses with nbSAT, LBD or clause size$\leq x$.

natorial benchmark, respectively (at each database reduction, after removing the half of learnt clauses with bigger LBD, the number of remaining learnt clauses with each nbSAT (LBD, clause size) value is counted. The total number of learnt clauses with each nbSAT (LBD, clause size) value is obtained by summing up the numbers for all database reductions). On the industrial benchmark, about 46% of learnt clauses are of nbSAT $\leq$2, and these clauses contribute about 71% in conflict analyses (i.e. 71% of the learnt clauses used in conflict analyses are of nbSAT$\leq$2) and 47% in unit propagation; while about 58% of learnt clauses are of LBD$\leq$3, and these clauses contribute about 62% in conflict analyses and about 46% in unit propagation. These data mean that the clauses with small nbSAT are probably more useful for deriving conflicts and for unit propagation during search than the clauses with small LBD. Considering that Glucose 3.0 can remove some clauses with nbSAT$\leq$2 that could otherwise contribute in unit propagation and in conflict analyses, but always keeps the half of learnt clauses with smaller LBD in each database reduction, the effectiveness of the clauses with small nbSAT for deriving conflicts and for unit propagation might be still more important than as suggested in Figure 1. Similar observation can be made by analyzing Figure 2 and Figure 4 on hard combinatorial benchmark.

4 Using nbSAT in Clause Database Reduction

We implement two derived versions of Glucose 3.0 to use the nbSAT measure in the clause database reduction: Glucose_nbSAT+LBD and Glucose_LBD+nbSAT, in which everything is identical to Glucose 3.0, except that nbSAT is used to predict the usefulness of a learnt clause. Concretely, if the number of learnt clauses becomes bigger or equal to 2000+300*n since the last database reduction, we will remove half of the learnt clauses in Glucose_nbSAT+LBD as follows.

1. compute the $nbSAT$ value for each learnt clause;
2. Sort all learnt clauses in the decreasing order of their nbSAT value, breaking ties using the decreasing order of their LBD value. The remaining ties are broken using the clause activity value as in Glucose 3.0.

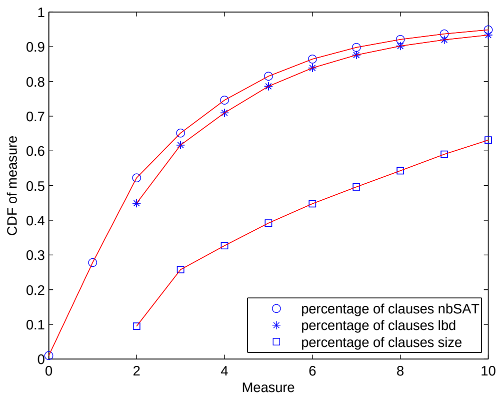

Figure 4. Cumulative distribution functions for number of learnt clauses with different nbSAT, LBD and clause size in glucose-3.0 on hard combinatorial benchmark of the SAT competition 2014. Names of curves are in the same ordering as the curves. Each point (x, y) represents the percentage y of learnt clauses with nbSAT, LBD or clause size$\leq x$.

3. Remove the first half of learnt clauses (i.e. those with bigger nbSAT values), by keeping binary clauses, clauses whose LBD is 2, and the locked clauses (i.e. clauses that are reasons of the current partial assignment).

Glucose_LBD+nbSAT is identical to Glucose_nbSAT+LBD except that all learnt clauses are sorted in the decreasing order of their LBD value, breaking ties using the decreasing order of their nbSAT value.

Note that the saved assignment changes frequently during search. The measurement nbSAT works well only when the learnt clause database reduction is fired frequently, because otherwise, it does not reflect the current search state after many conflicts. This is not a problem with Glucose_nbSAT+LBD and Glucose_LBD+nbSAT, because the two solvers reduces the database frequently as Glucose 3.0, making it relevant to use the nbSAT measure in the database reduction.

Recall that the definition of nbSAT is similar to the psm measure proposed in [2]. Nonetheless, the psm measure is used in a different way to manage learnt clause database. At each database reduction, if the psm value of a learnt clause is bigger than a dynamically determined threshold, the clause is frozen (i.e., it is no longer used in unit propagation) but not definitely removed, otherwise it is activated (i.e., it can be used in unit propagation). A clause is definitely removed if it is not activated after $k(=7)$ times, or if it is not involved in search for more than k times. The motivation of the approach can be stated as follows. The psm value of a learnt clause can change quickly. Freezing learnt clauses with big psm values instead of definitely removing them allows to keep those learnt clauses of which the psm values are big at a database reduction but will become small (thus useful for unit propagation and conflict analyses) in the near future. See [2] for more details. The utilization of the nbSAT measure appears simpler in our case: at every database reduction, the half of learnt clauses with bigger nbSAT values is simply and definitely removed without using any threshold.

5 Reducing Learnt Clause Database in the Root of the Search Tree

In Glucose, as well as in most CDCL-based solvers, a clause database reducing process can be fired inside a search tree. Two observations can be made about this strategy: (1) there are locked clauses, i.e. clauses that are reasons of the current partial assignment, that cannot be removed, (2) the part of the tree before the reducing and the part of the tree after the reducing are constructed with very different clause database.

We can reduce the learnt clause database always at the beginning of each restart, i.e., at the root of the search tree that is going to be constructed, every time the number of learnt clauses is bigger or equal to 2000 + 300*n since the last database reducing. In this way, clauses satisfied by variables fixed at the root are simply removed, as well as the literals falsified in the remaining clauses. Note that no clause is locked at the root of a search tree. Moreover, since the reduction is not done inside the search tree, the search tree is constructed with the same incremental clause database.

This reduction policy is implemented in Glucose_nbSAT+LBD and Glucose_LBD+nbSAT, giving Glucose_nbSAT+LBD_root and Glucose_LBD+nbSAT_root, respectively. Compared with Glucose 3.0, the database reduction is delayed in Glucose_nbSAT+LBD_root and Glucose_LBD+nbSAT_root, because it is not fired as soon as the number of the newly learnt clauses reaches a limit, but should wait for the next restart. However the delay is not important, because the two solvers perform fast restart as Glucose 3.0. This reducing policy was also used in the parallel version of Glucose. Its effectiveness will be empirically studied in Section 7.

6 Keeping Subsuming Learnt Clauses

When a learnt clause is in the first half after all learnt clauses are sorted in the decreasing order of their nbSAT value, i.e., when it is going to be removed by the database reduction process, we check if it subsumes an original clause or if it can be resolved with an original clause to produce a resolvent that subsumes the original clause.

In the first case, the learnt clause replaces the original clause and will never be removed.

Example. Let $x_1 \vee x_2 \vee x_3 \vee x_4$ be an original clause, and $x_2 \vee x_3 \vee x_4$ be a learnt clause, then the shorter learnt clause is added as an original clause that is never removed, and the original clause $x_1 \vee x_2 \vee x_3 \vee x_4$ is removed.

In the second case, the produced resolvent replaces the original clause and will never be removed.

Example. Let $x_1 \vee x_2 \vee x_3 \vee x_4$ be an original clause, and $\bar{x}_2 \vee x_3 \vee x_4$ be a learnt clause, then the resolvent $x_1 \vee x_3 \vee x_4$ is added as an original clause that is never removed, and $x_1 \vee x_2 \vee x_3 \vee x_4$ is removed.

Glucose_nbSAT+LBD_root_rsltn and Glucose_LBD+nbSAT_root_rsltn are respectively Glucose_nbSAT+LBD_root and Glucose_LBD+nbSAT_root that keep the subsuming learnt clauses. This policy is similar to subsumption elimination (SE) and self-subsuming resolution (SSR) used in preprocessing in some SAT solvers. The difference is that in Glucose_nbSAT+LBD_root_rsltn and Glucose_LBD+nbSAT_root_rsltn, the policy is not limited in the preprocessing, but is dynamically applied in every clause database reduction, which requires a highly efficient implementation, because it is applied for an exponential number of times.

7 Experimental Results

We run experiments on the industrial benchmark and the hard combinatorial (crafted) benchmark of the SAT 2014 competition to compare the following solvers:

`Glucose_nbSAT+LBD_root_rsltn:` It is identical to Glucose 3.0, except that it sorts the learnt clauses in the decreasing order of their nbSAT value, breaking ties using the LBD values, when reducing clause database (See Section 4), that clause database reduction is always performed at the root (see Section 5), and that resolution is used to keep subsuming learnt clauses (see Section 6)

`Glucose_LBD+nbSAT_root_rsltn:` It is identical to Glucose_nbSAT+LBD_root_rsltn, except that the it sorts the learnt clauses in the decreasing order of their LBD value, breaking ties using the nbSAT values, when reducing clause database (See Section 4)

`Glucose_nbSAT+LBD_rsltn:` It is identical to Glucose_nbSAT+LBD_root_rsltn, except that the clause database reduction is fired as in Glucose 3.0, i.e., it is not fired in the root of a search tree, but fired as soon as the number of learnt clauses reaches a limit as in Glucose 3.0 (see Section 5)

`Glucose_LBD_root_rsltn:` It is identical to Glucose_LBD+nbSAT_root_rsltn, except that nbSAT is not used to break ties when sorting learnt clauses

`Glucose 3.0:` It is available at www.labri.fr/perso/lsimon/glucose/

All solvers are run on a computer with Intel Westmere Xeon E7-8837 of 2.66GHz and 10GB of memory under Linux. The cutoff time is 5000 seconds as in the competition. Since there can be other users on the machine, each solver is run three times for each instance and the best result is taken into account to minimize the possible perturbation. Table 1 compares the results of each solver.

	Application	Hard combinatorial
solvers	#solved(sat+unsat)(time)	#solved(sat+unsat)(time)
Glucose_nbSAT+LBD_root_rsltn	210(103+107)(995.02s)	174(79+95)(1081.07s)
Glucose_LBD+nbSAT_root_rsltn	207(100+107)(960.02s)	166(78+88)(928.08s)
Glucose_nbSAT+LBD_rsltn	209(97+112)(1002.67s)	176(83+93)(1056.70s)
Glucose_LBD_root_rsltn	201(95+106)(949.60s)	171(79+92)(1064.69s)
Glucose 3.0	206(99+107)(962.35s)	164(77+87)(1087.17s)

Table 1. Number of instances solved within 5000 seconds of each solver, as well as the average runtime to solve an instance, in the set of 300 industrial or hard combinatorial instances of the 2014 SAT competition

The first observation that can be made from Table 1 is that the resolution to keep subsuming learnt clauses in Glucose_LBD_root_rsltn is not very effective for the industrial instances, because Glucose_LBD_root_rsltn solves 5 instances fewer than Glucose 3.0, but it is effective for the crafted instances, because Glucose_LBD_root_rsltn solves 7 instances more than Glucose

3.0. The explanation of this phenomenon is that the resolution is very costly for the industrial instances that contain a large number of clauses, even with a highly efficient implementation. However, The subsuming learnt clauses kept thanks to the resolution becomes effective for both industrial and hard combinatorial instances when the nbSAT measure is used to predict the usefulness of a learnt clause. In fact, the three solvers with nbSAT and the resolution to keep subsuming learnt clauses solve more industrial and hard combinatorial instances, especially when nbSAT is used as the main measure as in Glucose_nbSAT+LBD_root_rsltn and Glucose_nbSAT+LBD_rsltn.

The second observation made from Table 1 is that it does not matter much to reduce clause database at the root of a search tree or inside the search tree, since Glucose_nbSAT+LBD_root_rsltn reduces the clause database only at the root of a search tree, and solves roughly the same number of instances in both industrial and hard combinatorial categories as Glucose_nbSAT+LBD_rsltn does.

The most important observation made from Table 1 is that the nbSAT measure appears to be more accurate than the LBD measure in predicting the usefulness of a learnt clause, especially for the hard combinatorial instances. In fact, Glucose_nbSAT+LBD_root_rsltn solves 10 hard combinatorial instances more than Glucose 3.0, and Glucose_nbSAT+LBD_rsltn solves 12 hard combinatorial instances more than Glucose 3.0. One possible explanation is that industrial instances exhibit a community structure that can be captured by LBD, while the community structure is not so clear in hard combinatorial instances. Therefore, the impact of nbSAT is more important for the hard combinatorial instances.

The solver Glucose_nbSAT+LBD_root_rsltn participated in the SAT Race'2015 under the name Glucose_nbSatRsltn and solves 4 instances more than Glucose in the set of 300 application instances (see http://baldur.iti.kit.edu/sat-race-2015/index.php?cat=results).

8 Conclusion

We have presented a measure called *nbSAT* to predict the usefulness of a learnt clause when reducing learnt clause database in Glucose 3.0. The nbSAT measure is similar to a previous measure called *psm*, but is exploited in a different way. It is closely associated with the aggressive clause database reduction and fast restart policies in Glucose 3.0. Experimental results on instances from the SAT 2014 competition suggest that the learnt clauses with small nbSAT values are more useful than the learnt clauses with small LBD values. We then implemented an improvement in Glucose 3.0 to remove half of learnt clauses with large nbSAT values, instead of half of clauses with large LBD clauses, when periodically reducing the learnt clause database. In addition, we implemented a resolution method in Glucose 3.0 to keep the learnt clauses or resolvents produced using a learnt clause that subsume an original clause of the instance to solve. The experimental results on benchmarks from the SAT 2014 competition show that Glucose 3.0 with these two improvements solve significantly more application and crafted instances than Glucose 3.0, no matter if the clause database reductions are fired at the root of a search tree or not.

Since the nbSAT value of a learnt clause can be changed frequently during search, one way to make the nbSAT measure more effective might be to use a yet more aggressive database reduction policy in Glucose 3.0, so that the nbSAT value can be more frequently re-computed to reflect its actual usefulness. We will investigate this research line in the future.

Acknowledgements We thank the anonymous reviewers for insightful comments. This work is supported by National Natural Science Foundation of China (NSFC, Grant No.61370184, Grant No.61070235 and Grant No.61272014) and by the MeCS platform of university of Picardie Jules Verne.

References

[1] Carlos Ansótegui, Jesús Giráldez-Cru, and Jordi Levy. The community structure of sat formulas. In *Theory and Applications of Satisfiability Testing–SAT'12*, pages 410–423. Springer, 2012.

[2] Gilles Audemard, Jean-Marie Lagniez, Bertrand Mazure, and Lakhdar Sais. On freezeing and reactivating learnt clauses. In *14th International Conference on Theory and Applications of Satisfiability Testing–SAT'11*, pages 188–200, 2011.

[3] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern sat solvers. *International Joint Conference on Artificial Intelligence Proceedings-IJCAI'09*, 38(4):399–404, 2009.

[4] Gilles Audemard and Laurent Simon. Refining restarts strategies for sat and unsat formulae. *in Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming–CP'12*, 2012.

[5] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. Conflict-driven clause learning sat solvers. *Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications*, pages 131–153, 2009.

[6] Niklas Eén and Niklas Sörensson. An extensible sat-solver. pages 502–518, 2004.

[7] Carla P. Gomes, Bart Selman, and Kautz Henry. Boosting combinatorial search through randomization. *in Proc. AAAI-98*, Madison, WI, July 1998.

[8] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[9] Zack Newsham, Vijay Ganesh, Sebastian Fischmeister, Gilles Audemard, and Laurent Simon. Impact of community structure on sat solver performance. In *Theory and Applications of Satisfiability Testing–SAT'14*, pages 252–268. Springer, 2014.

[10] Knot Pipatsrisawat and Adnan Darwiche. A lightweight component caching scheme for satisfiability solvers. In *Theory and Applications of Satisfiability Testing–SAT'07*, pages 294–299. Springer, 2007.