Improved algorithm of unsatisfiability-based Maximum Satisfiability

Yiyuan Wang, Dantong Ouyang, Liming Zhang*

Symbol Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
College of Computer Science and Technology, Jilin University, Changchun 130012, China

Abstract. Recently, the Maximum Satisfiability (MaxSAT) discovers a rapidly increasing number of practical applications in a wide range of different areas. Unsatisfiability-based MaxSAT algorithms have been put forward aimed at improving the performance of solving MaxSAT problem. In these algorithms, several excellent Boolean Satisfiability (SAT) solvers are employed to iteratively identify unsatisfiable sub-formulas and the additional blocking variables are also applied to relax some of initial clauses or encode Boolean constraints into a CNF formula. Firstly, this paper proposes two novel optimizations to reduce the search space: finding all disjoint unsatisfiable cores with the original clauses removed and no blocking variables added; avoiding generating the largest number of blocking variables' clauses with heuristic strategy. Moreover, in order to reduce the number of satisfying iterations, the partial assignment is introduced into satisfiability-based framework as the tight bound accelerator which could find a smaller number of blocking variables. Experimental results show that the unsatisfiability-based algorithm with these optimizations results in an improved and more effective solver for MaxSAT problem than previous algorithms. In addition,this results in consistent performance gains in most cases and on average 1.5 times speed-up for MaxSAT with state-of-the-art algorithms.

Introduction

The classical NP-complete problem of Boolean Satisfiability (SAT) [8] has been a growth of interest in not just the theoretical computer science community, but also in various areas where feasible solutions to these problems are significantly important in many practical applications. These practical applications have been able to successfully apply SAT as a decision procedure to determine whether these instances are SAT or UNSAT. However, there are more extra demands of the SAT problem that exceed this decision procedure of SAT solvers, such as Maximum Satisfiability problem (MaxSAT) [12, 3] and Pseudo-Boolean Optimization (PBO). Moreover, using the strong relationship between these problems, several new algorithms have been developed [21].

MaxSAT as a well-known problem in Computer Science, consists of identifying the largest number of clauses in a CNF formula that can be satisfied. The recent application of MaxSAT and variants in design debugging and verification of complex designs [4, 10, 20] motivated the development of new MaxSAT algorithms, capable of solving

large structured problem instances common to these application domains. Many exact methods for finding MaxSAT have been developed in recent years. Davis-Putnam-Logemann-Loveland procedure [5] explores a binary search tree and then establishes incrementally truth assignments. The result of several methods based on this procedure is very good, but they are not suitable for MaxSAT practical problems. Some other exact methods generally based on Branch and Bound (B&B) algorithms [2] have been designed to handle MaxSAT, but are unable to solve majority of problem instances. Recently algorithms based on the identification of unsatisfiable sub-formulas [7, 18, 19, 15], with blocking variables introduced into these algorithms and Pseudo Boolean Optimization used to simplify CNF, have also been developed, and are now able to tackle all these Boolean optimization problems and most of unsatisfiable instances.

This paper firstly reviews previous MaxSAT algorithm based on unsatisfiable sub-formulas named $msu4$ [19], and then proposes an improved MaxSAT algorithm based on the method of binary search [11]. Despite the improved algorithm building on $msu4$ based on unsatisfiable sub-formulas and binary search, this paper proposes one improvement as the tight bound accelerator and extra two optimizations to reduce the search space, which are novel, including: (1) reducing the number of SAT solver iterations under the control of partial assignment; (2) developing the heuristic strategy to avoid generating the largest number of blocking clauses, so as to slow down the efficiency of algorithm, when the middle value is half number of blocking variables. (3) speeding up to find all disjoint unsatisfiable cores without blocking variables added into these clauses in the early iterative execution of binary search's preprocess. The new improved algorithm also uses several techniques [6] to utilize BDDs for encoding cardinality constraints as Boolean circuits. Experimental results show that the new MaxSAT algorithm is observably faster than the previous MaxSAT algorithms in a wide range of unsatisfiable industrial instances, demonstrating the robustness of the proposed approach.

Preliminaries

Firstly a finite set of Boolean variables is assumed X=$\{x_1,x_2,x_3,\ldots,x_n\}$. A CNF formula φ is defined on the conjunction of clauses w_i and each clause consists of the disjunction of literals, which is either a variable x_i or its complement $\overline{x}_i$. In the context of search algorithms of SAT, variables can be assigned a logic value, either 0 or 1. Alternatively, variables can defined as a function v: $X \rightarrow\{0,u,1\}$, where u denotes a variable whose value is uncertain. When all of variables are assigned a value in $\{0,1\}$, then v is referred to as a complete assignment. Otherwise it is a partial assignment [17] where some variables do not have values and are assigned u, which will be used in the following improved algorithm.

In this section Maximal Satisfiability and Minimal Unsatisfiability are firstly introduced. Afterwards, the previous MaxSAT algorithm using unsatisfiable cores named $msu4$ is described.

Maximal Satisfiability and Minimal Unsatisfiability

Our improved algorithm and previous methods are based on maximal satisfiability and minimal unsatisfiability. The MaxSAT problem as an optimization problem corresponds to the minimization of the number of false clauses in a CNF formula, which means that MaxSAT problem discovers an assignment to the variables of a CNF formula that maximizes the number of satisfied clauses.

In order to illustrate this problem, Maximally Satisfiable Subset (MSS) is defined as a subset of clauses in a CNF formula, which is satisfiable and then adds any of the other clauses in an original formula to MSS leading new clauses set to unsatisfiable. In this context, Minimal Unsatisfiable Subset (MUS) is an inconsistent subset of the clauses of the original formula and dropping one of its clauses will make it satisfiable. MSS and MUS [14, 7] correspond to the following rules:

$$MSS(\varphi) = \begin{cases} m \quad is \quad satisfiable, & \text{if m}\subseteq\varphi \\ m \cup \{a\} \quad is \quad unsatisfiable, & \forall \text{a}\in(\varphi\text{-m}) \end{cases}$$

$$MUS(\varphi) = \begin{cases} m \quad is \quad unsatisfiable, & \text{if m}\subseteq\varphi \\ m - \{a\} \quad is \quad satisfiable, & \forall \text{a}\in\text{m} \end{cases}$$

Notice that MSS as a satisfied subset of clauses cannot become bigger and MUS as an unsatisfiable subset that removes any of its clauses can render it satisfiable. The relationship between MaxSAT and MSS is subtle. Given an unsatisfiable instance, MaxSAT is the largest subset of clauses that can be satisfiable which means to find MSS with maximal number. However, all MSSes may have different sizes and cannot be all MaxSAT solutions with maximal number. Removing the clauses that are not included in the MSS makes CNF formula satisfiable. Therefore Minimal Correction Set (MCS) is introduced to describe the complement of MSS and the set of MCSes is shown:

$$MCSes(\varphi) = \{m | m \subseteq \varphi \quad and \quad (\varphi - m) \in MSSes(\varphi)\}$$

MCS as the complementary point of MSSes offers the real link to MUSes. Note that any clause of MUS in a CNF formula makes the formula unsatisfiable and the complement formula satisfiable,thus every MSS must contain at least one of clauses of every MUS. This relationship between MCSes and MUSes can usually be described as a solution to a set converting problem. Specifically, each hitting set of the MUSes is an MCS. Given a collection of sets, hitting set intersects all sets in the collection in at least one element and the problem of hitting set is an equivalent reformulation of set covering. A MCS is a hitting set of the MUSes with the additional limit that it cannot become smaller without losing its defining property: it is an irreducible hitting set and each hitting set of MCSes is an MUS. The example formula is shown to explain their relationship between MSSes, MCSes and MUSes.

$$\varphi = (x_1) \wedge (\overline{x}_1 \vee \overline{x}_2) \wedge (x_2) \wedge (\overline{x}_2)$$

$$MSSes(\varphi) = \big\{\{x_1, \overline{x}_1 \vee \overline{x}_2, \overline{x}_2\}, \{\overline{x}_1 \vee \overline{x}_2, x_2\}, \{x_1, x_2\}\big\}$$

$$MCSes(\varphi) = \big\{\{x_2\}, \{x_1, \overline{x}_2\}, \{\overline{x}_1 \vee \overline{x}_2, \overline{x}_2\}\big\}$$

$$MUSes(\varphi) = \big\{\{x_1, \overline{x}_1 \vee \overline{x}_2, x_2\}, \{x_2, \overline{x}_2\}\big\}$$

In the example above, every MUS of a CNF formula is an irreducible hitting set of the MCSes of that CNF formula. In the given CNF formula above, the number of maximal satisfiable clauses is 3. More importantly, MSS and MUS also can tell us the relationship between MUS and MaxSAT problem.

A MaxSAT algorithm using unsatisfiable cores

Recently, several methods based on unsatisfiable cores are proposed. $Msu4$ which applies improved blocking variable into clauses is more effective than other algorithms in the experiment results. More importantly, $msu4$ avoids interacting with a PBO solver and instead is a fully SAT-based solver that relies on the most effective techniques, such as Boolean Constraint Propagation, conflict based learning and conflict-directed backtracking [16]. Finally, the two advantages of $msu4$ lie in adding at most one blocking variable into each clause in a CNF formula which can avoid quickly increasing the number of blocking variables and clauses, and applying linear programming encoding of pseudo boolean constraints with BDDs and sorting networks.

Proposition 1 ***MaxSAT Upper Bound(UB)****: In a CNF formula, φ corresponds to the number of clauses. Let K denote disjoint unsatisfiable cores, then φ-K corresponds to an MaxSAT Upper Bound.*

Proposition 2 ***MaxSAT Lower Bound(LB)****: Let B define as the set of blocking variables assigned value TRUE when clauses become satisfiable, then φ-B denotes a MaxSAT Lower Bound.*

In previous subsection, some relationships between MUS and MaxSAT are briefly introduced and these properties are employed in the algorithm to define MaxSAT Upper Bound and Lower Bound [19]. Some details for $msu4$ are shown in [19].

Improved MaxSAT algorithm with some strategies

As shown previously, unsatisfiable-based algorithm is able to tackle MaxSAT problem. However, $msu4$ works by the redundant lower and upper bounds in some SAT solver iterations. This section proposes an improved algorithm based on $msu4$ with two key optimizations, which are used as the tight bound accelerator. Firstly the thought of binary search [11] (its algorithm named BIN) can be imported into $msu4$ for clearly improving its efficiency by decreasing the redundant calculation. Secondly, based on the idea of binary search, this paper proposes some key optimizations: (1)the use of partial assignment computing a smaller number of blocking variables assigned value

TRUE instead of the middle value of binary search in each satisfying assignment. (2)the heuristic strategy HS_{BS} to avoid generating the largest number of clauses containing blocking variables when the middle value is nearly half number of blocking variables. (3)acceleration to find disjoint unsatisfiable cores by iteratively computing the original formula and temporarily deleting these cores that have already been found.

Unsatisfiable core-based MaxSAT algorithm using binary search

In algorithm 1, compared with the upper bound and lower bound in the $msu4$, two new bounds, which denote the lower and upper bounds of lower bound, are firstly proposed and then used into algorithm 1 and algorithm 2.

Proposition 3 ***Lower Bound of Lower Bound (L_LB)****: φ corresponds to a CNF formula. Remove any L_LB clauses to render new φ^* unsatisfiable. In our algorithm 2, L_LB is initialized to the number of unsatisfiable cores.*

Proposition 4 ***Upper Bound of Lower Bound (L_UB)****:φ corresponds to a CNF formula. Remove the given L_UB clauses to make new φ^* satisfiable. In algorithm 2, L_UB is firstly assigned to the number of blocking variables.*

Proposition 5 *[11] Binary search for MaxSAT executes $\Theta(log(W))$ calls to a SAT oracle in the worst case.*

Compared with $msu4$, algorithm 1 uses the method of binary search to acquire MaxSAT solution and is also based on Proposition 5 defined in the BIN to guide algorithm 1. The pseudo code for $msu5\_bin$ is shown in Algorithm 1.

In $msu4$, lower bound only depends on the value of blocking variables when φ is satisfiable, and its variation is passive. Considering that $msu4$ calculates the size of lower bound by inches, this paper presents the binary search, whose algorithm in computer science finds the position of a specified value within a sorted key. Algorithm 1 employs the binary search to take the initiative to alter two bounds in order to accelerate the growth of two bounds for reducing the number of SAT solver iterations, which will solve the MaxSAT problem quickly.

In algorithm 1, there are several differences between $msu4$ and $msu5\_bin$. L_UB is initialized to the number of initial clauses and L_LB is to 0. Before each call to the SAT solver, BS_B which corresponds to the middle variable, needs to be computed under the method of binary search (line 6). Afterwards $CNF(\sum_{i \in V_B} b_i \leq BS_B)$ is generated and then added to φ_w (line 7 and 8). L_LB is only updated on iterations when the SAT solver returns false. Considering such an iteration, for which the unsatisfiable core does not include initial clauses (line 17), L_LB will be assigned to BS_B. When the unsatisfiable core contains original clauses $I > 0$, this process is the same as $msu4$ from line 13 to 16. Consider now a satisfiable iteration. It occurs when the number of blocking variables assigned 1 is sufficient to make the current CNF formula get a satisfiable assignment, L_UB will be updated(line 21). But it can not guarantee that this number is minimal. When $L\_LB + 1 \geq L\_UB$, it is an answer of the MaxSAT solution.

Algorithm 1 Improving MaxSAT Algorithm based on binary search

1: ▷ the running condition of algorithm 1 is that φ is unsatisfiable
2: $L\_LB \leftarrow 0$
3: $L\_UB \leftarrow |\varphi|$
4: $\varphi_w \leftarrow \varphi$
5: **while TRUE do**
6: $BS_B \leftarrow (L\_LB + L\_UB)/2$ ▷ BS_B is the middle variable in the binary search
7: $\varphi_{CH} \leftarrow CNF(\sum_{i \in V_B} b_i \leq BS_B)$
8: $(st, \varphi_c) \leftarrow SAT(\varphi_w \cup \varphi_{CH})$
9: **if** st = **UNSAT then**
10: $\varphi_I \leftarrow \varphi_c \cap \varphi$
11: $I \leftarrow \{i | w_i \in \varphi_I\}$
12: $V_B \leftarrow V_B \cup I$
13: **if** $|I| > 0$ **then**
14: $\varphi_N \leftarrow \{w_i \cup \{b_i\} | w_i \in \varphi_I\}$
15: $\varphi_a \leftarrow CNF(\sum_{i \in I} b_i \geq 1)$
16: $\varphi_w \leftarrow (\varphi_w - \varphi_I) \cup \varphi_N \cup \varphi_a$
17: **else**
18: $L\_LB \leftarrow BS_B$
19: **end if**
20: **else**
21: $L\_UB \leftarrow BS_B$
22: **end if**
23: **if** $L\_LB + 1 \geq L\_UB$ **then**
24: **return a satisfiable assignment**
25: **end if**
26: **end while**

Improved algorithm based on binary search

Although the performance of $msu5\_bin$ is superior to previous methods, three novel improvements are proposed in algorithm 2, which can be more efficient to optimize the MaxSAT algorithm.

Firstly, when the SAT solver returns satisfiable, the upper bound only relies on the middle variable in algorithm 1, which may not be the minimal number of blocking variables assigned 1. In detail, after some satisfiable iterations, the assignments of upper bounds are usually redundant. Our algorithm 2 (named $msu6\_bd$) proposes the technology of partial assignment, which could effectively reduce the number of satisfying iterations in algorithm 1. Comparing the number of blocking variables (assigned value TRUE) in partial assignment with the value of middle variable, the smaller value will be assigned to the upper bound. Specifically, a partial assignment might not include some blocking variables called irrelevant variables, which means that SAT solver only needs to analyze several blocking variables and discard some other blocking variables if they are not used for satisfying any clause. Our Algorithm is built on the interface of PicoSAT [1] to eliminate unnecessary SAT tests by simplifying satisfying assignments using standard partial assignment.

Algorithm 2 Our Improved MaxSAT Algorithm based on binary search

▷ the running condition of algorithm 2 is that φ is unsatisfiable
$IN_C \leftarrow \phi; UN_C \leftarrow \phi; CS_C \leftarrow \phi; V_B \leftarrow \phi$
while $st \neq SAT$ **do**
  $(st, \varphi_c) \leftarrow increSAT(\varphi - IN_C)$
  if st = **UNSAT then**
    $V_B \leftarrow V_B \cup \{i | w_i \in \varphi_c\}$
    $IN_C \leftarrow IN_C \cup \varphi_c$
    $UN_C \leftarrow UN_C \cup \{w_i \cup \{b_i\} | w_i \in \varphi_c\}$
    $CS_C \leftarrow CS_C \cup CNF(\sum_{i \in \varphi_c} b_i \geq 1)$
  end if
end while
$\varphi_w \leftarrow (\varphi - IN_C) \cup UN_C \cup CS_C$
$L\_LB \leftarrow |IN_C|; L\_UB \leftarrow |V_B|$
while TRUE do
  $BS_B \leftarrow (HS_{BS}(L\_LB))?(L\_LB + L\_UB)/2 : (3L\_LB + L\_UB)/4$ ▷ heuristic strategy reduces the size of CNF formula
  $\varphi_{CH} \leftarrow CNF(\sum_{i \in V_B} b_i \leq BS_B)$
  $(st, \varphi_c) \leftarrow increSAT(\varphi_w \cup \varphi_{CH})$
  if st = **UNSAT then**
    $\varphi_I \leftarrow \varphi_c \cap \varphi$
    $I \leftarrow \{i | w_i \in \varphi_I\}$
    $V_B \leftarrow V_B \cup I$
    if $|I| > 0$ **then**
      $\varphi_N \leftarrow \{w_i \cup \{b_i\} | w_i \in \varphi_I\}$
      $\varphi_a \leftarrow CNF(\sum_{i \in I} b_i \geq 1)$
      $\varphi_w \leftarrow (\varphi_w - \varphi_I) \cup \varphi_N \cup \varphi_a$
    else
      $L\_LB \leftarrow BS_B$
    end if
  else
    $V_S \leftarrow$|the number of blocking variables assigned value 1|
    $L\_UB \leftarrow (V_S < BS_B)?V_S : BS_B$ ▷SAT solver selects partial assignment
  end if
  if $L\_LB + 1 \geq L\_UB$ **then**
    return a satisfiable assignment
  end if
end while

Secondly, when the middle variable in algorithm 1 is assigned to half number of blocking variables, it could generate the largest number of clauses for pseudo boolean constraints. Considering that $x_1 + x_2 + \ldots + x_n \geq k$ results in BDDs with $(n - k + 1) \times k$ nodes [6], the number of BDD nodes is maximum when $k = n/2$. Based on above, algorithm 2 proposes the heuristic strategy HS_{BS} to avoid this situation in binary search. The heuristic HS_{BS} function is, when BS_B is approximately equal to $|V_B|/2$, BS_B is assigned to $(3L\_LB + L\_UB)/4$ which is nearly equal to $|V_B|/4$.

Lastly, when algorithm 1 iteratively obtains an unsatisfiable core each time, the CNF formula will be added more and more clauses generated by blocking variables. For this reason, algorithm 2 starts with finding all disjoint unsatisfiable cores as a prepossessing under the smaller search space with original clauses removed and no blocking variables added. In detail, whenever algorithm 2 finds an unsatisfiable core, some clauses in this core will be temporarily removed. When SAT solver can no longer obtain any unsatisfiable core, the CNF formula will be restored and then the method of binary search continues to solve MaxSAT problem. For this unsatisfiable core-solving preprocessing that requires to iterative calls to a SAT solver, the well-known incremental interface of PicoSAT solver can be considered, which is a SAT solver that after running a CNF formula, keeps all clauses and the internal state that can be reused when inserting additional clauses or deleting information learned from clauses. The pseudo code for $msu6\_bd$ is shown in Algorithm 2.

Algorithm 2 begins with finding out all disjoint unsatisfiable cores from line 2 to 11. Afterwards φ_w is set as $(\varphi - IN_C) \cup UN_C \cup CS_C$, which shows that φ removes unsatisfiable cores, and then adds those clauses including blocking variables and pseudo boolean constraints (line 12). L_LB will be initialized to the number of unsatisfiable cores and L_UB is assigned to the number of blocking variables, aiming at reducing the number of SAT solver iterations and speeding up to enhance the performance of algorithm 1 (line 13). In the process of binary search, the function of HS_{BS} is to prevent the number of blocking clauses becoming large when HS_{BS} is nearly equal to $V_B/2$ (line 15). More importantly, L_UB is assigned to the smaller value between V_S and BS_B (line 31).

Experiment Results

This section presents the experimental results of the improved MaxSAT algorithm ($msu6\_bd$). In order to evaluate this new method, pbo [6], $msu4$ [19] and $msu5\_bin$ which is built on the $msu4$ using the binary search [11] are considered as the comparison algorithms. The framework of our algorithm is implemented using a state-of-the-art SAT solver named PicoSAT-V953 [1].

In order to evaluate the improved MaxSAT algorithm, a set of instances are selected. These instances are obtained from existing unsatisfiable subsets of crafted and industrial MaxSAT benchmarks, the SAT competition archives and SATLIB. The majority of the considered industrial instances are originally from EDA applications, including model checking, equivalence checking , and test pattern generation. The total number of unsatisfiable instances considered is 447. All experiments are running on a Core 2 Dual 2.13 Ghz workstation with 4 GB of RAM and the CPU timeout of 1000 seconds for each instance.

Table 1 shows the number of solved instances for all selected benchmark sets. The improvements from $msu4$ and $msu5\_bin$ to $msu6\_bd$ are very clear. The overall number of solved instances ($msu6\_bd$) is vastly improved as it now solves more instances than $msu5\_bin$, $msu4$ and pbo. When solving MaxSAT crafted instances, the performances of $msu5\_bin$ and $msu6\_bd$ are parallel. Nevertheless, $msu6\_bd$ is not effective enough to solve all the selected instances. The MaxSAT algorithm based on the unsat-

Table 1. Number of Solved Instances

Ben. set	#I	pbo	msu4	msu5_bin	msu6_bd
ms_ind	321	111	178	230	**288**
ms_cra	126	24	68	**91**	90
Total	447	135	246	321	**378**

isfiable cores needs to use SAT solver to iteratively execute, which is not suitable for the time-out instances.

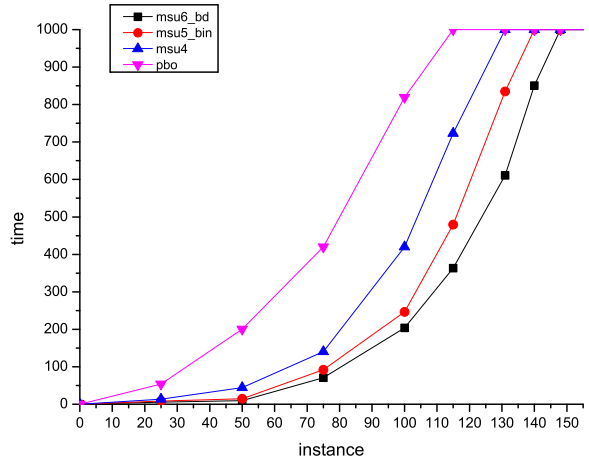

Fig. 1. Run times of algorithms

Figure 1 shows a plot by increasing run times of the test instances for each algorithm. Firstly some instances in the time limits are selected and sorted by the computing time increasingly. Clearly, pbo is inefficient. For most of test cases, it can hardly figure out the answer. In additional, $msu6\_bd$ increases in a slower rate than $msu4$ and $msu5\_bin$. It is beneficial because $msu6\_bd$ uses the binary search and three additional optimization strategies: speeding up to find all disjoint unsatisfiable cores in reduced search space, using the tight bound accelerator to reduce the number of SAT solver iterations, and applying HS_{BS} to reduce the search space of binary search.

Figure 2 plots the run times of our approach $msu6\_bd$ versus the previous algorithm $msu4$, along with the 1x, 2x, 3x and 10x lines, clearly showing the superiority of the proposed method. The time of all selected instances in two algorithms is in time limits. As can be observed, $msu6\_bd$ is clearly faster than $msu4$. $msu4$ has already proved

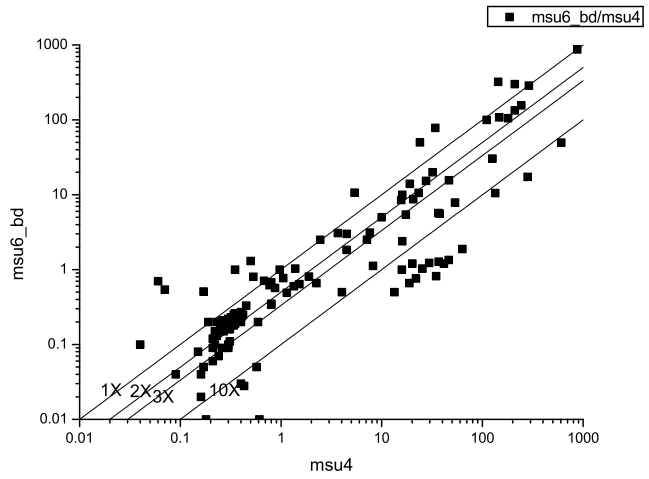

Fig. 2. Scatter plot: msu6_bd vs. msu4

that it is more effective than pbo. Despite the performance advantage of the versions of $msu6\_bd$, there are some exceptions.

Conclusions

This paper proposes an improved unsatisfiable-based MaxSAT algorithm. At first partial assignment can be employed into binary search-based framework as the tight bound accelerator: partial assignment could make upper bound select a smaller number of clauses than previous algorithms. Moreover, HS_{BS} avoids generating the largest number of clauses for pseudo boolean constraints, which can reduce the search space of binary search. Finally, another improvement strategy as the preprocessing of binary search can find as much disjoint unsatisfiable cores as possible to speed up the execution speed of MaxSAT algorithm. With the original clauses removed and no blocking clauses added, this optimization can significantly reduce search space in these number of unsatisfiable iterations. Preliminary experimental results show that these techniques significantly improve the performance of our unsatisfiability-based algorithm in solving industrial instances of MaxSAT problem. As a result, our algorithm maintains its competitive advantage over other algorithms for MaxSAT problem. Despite the promising results, additional application to $msu6\_bd$ is expected, which is to exploit structure information [9, 13] to accelerate MaxSAT algorithm in design debugging.

Acknowledgment

The authors are thankful to the editor and anonymous reviewers for their valuable comments that help us improve the quality of the paper.

References

1. Biere, A.: Picosat essentials. Journal on Satisfiability, Boolean Modeling and Computation 4(75-97), 45 (2008)
2. Borchers, B., Furman, J.: A two-phase exact algorithm for max-sat and weighted max-sat problems. Journal of Combinatorial Optimization 2(4), 299–306 (1998)
3. Cai, S., Luo, C., Thornton, J., Su, K.: Tailoring local search for partial maxsat. In: Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)
4. Chen, Y., Safarpour, S., Marques-Silva, J., Veneris, A.: Automated design debugging with maximum satisfiability. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 29(11), 1804–1817 (2010)
5. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM 5(7), 394–397 (1962)
6. Eén, N., Sörensson, N.: Translating pseudo-boolean constraints into sat. Journal on Satisfiability, Boolean Modeling and Computation 2(3-4), 1–25 (2006)
7. Fu, Z., Malik, S.: On solving the partial max-sat problem. Theory and Applications of Satisfiability Testing-SAT 2006 pp. 252–265 (2006)
8. Gary, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-completeness. WH Freeman and Company, New York (1979)
9. Guo, T., Li, Z., Guo, R., Zhu, X.: Large scale diagnosis using associations between system outputs and components. In: Twenty-Fifth AAAI Conference on Artificial Intelligence (2011)
10. Heras, F., Larrosa, J., Oliveras, A.: Minimaxsat: A new weighted max-sat solver. Theory and Applications of Satisfiability Testing–SAT 2007 pp. 41–55 (2007)
11. Heras, F., Morgado, A., Marques-Silva, J.: Core-guided binary search algorithms for maximum satisfiability. In: Proceedings of the AAAI National Conference (AAAI). pp. 36–41 (2011)
12. Johnson, D.: Approximation algorithms for combinatorial problems. Journal of computer and system sciences 9(3), 256–278 (1974)
13. Le, B., Mangassarian, H., Keng, B., Veneris, A.: Non-solution implications using reverse domination in a modern sat-based debugging environment. In: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012. pp. 629–634 (2012)
14. Liffiton, M., Sakallah, K.: On finding all minimally unsatisfiable subformulas. In: Theory and Applications of Satisfiability Testing. pp. 34–42 (2005)
15. Manquinho, V., Martins, R., Lynce, I.: Improving unsatisfiability-based algorithms for boolean optimization. Theory and Applications of Satisfiability Testing–SAT 2010 pp. 181–193 (2010)
16. Manquinho, V., Marques-Silva, J.: Search pruning techniques in sat-based branch-and-bound algorithms for the binate covering problem. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 21(5), 505–516 (2002)
17. Marques-Silva, J., Lynce, I., Malik, S.: Conflict-driven clause learning sat solvers. SAT Handbook pp. 131–154 (2009)
18. Marques-Silva, J., Planes, J.: On using unsatisfiability for solving maximum satisfiability. arXiv preprint arXiv:0712.1097 (2007)
19. Marques-Silva, J., Planes, J.: Algorithms for maximum satisfiability using unsatisfiable cores. In: Proceedings of the conference on Design, automation and test in Europe. pp. 408–413 (2008)
20. Zhang, L., Bacchus, F.: Maxsat heuristics for cost optimal planning. In: Proceedings of the AAAI National Conference (AAAI) (2012)
21. Zhu, Z., Li, C., Manyà, F., Argelich, J.: A new encoding from minsat into maxsat. In: Principles and Practice of Constraint Programming. pp. 455–463 (2012)