Chapter 10
Algorithms for Maximum Satisfiability Using Unsatisfiable Cores

Joao Marques-Sila and Jordi Planes

Abstract Many decision and optimization problems in electronic design automation (EDA) can be solved with Boolean satisfiability (SAT). These include binate covering problem (BCP), pseudo-Boolean optimization (PBO), quantified Boolean formulas (QBF), multi-valued SAT, and, more recently, maximum satisfiability (MaxSAT). The first generation of MaxSAT algorithms are known to be fairly inefficient in industrial settings, in part because the most effective SAT techniques cannot be easily extended to MaxSAT. This chapter proposes a novel algorithm for MaxSAT that improves existing state-of-the-art solvers by orders of magnitude on industrial benchmarks. The new algorithm exploits modern SAT solvers, being based on the identification of unsatisfiable subformulas. Moreover, the new algorithm provides additional insights between unsatisfiable subformulas and the maximum satisfiability problem.

10.1 Introduction

Boolean satisfiability (SAT) is used for solving an ever increasing number of decision and optimization problems in electronic design automation (EDA). These include model checking, equivalence checking, design debugging, logic synthesis, and technology mapping [8, 19, 34, 36]. Besides SAT, a number of well-known extensions of SAT also find application in EDA, including pseudo-Boolean optimization (PBO) (e.g., [27]), quantified Boolean formulas (QBF) (e.g., [13]), multi-valued SAT [26], and, more recently, maximum satisfiability (MaxSAT) [33].

MaxSAT is a well-known problem in computer science, consisting of finding the largest number of satisfied clauses in unsatisfiable instances of SAT. Algorithms for MaxSAT are in general not effective for large industrial problem instances, in part

J. Marques-Sila (✉)
University College Dublin, Dublin, Ireland
e-mail: jpms@ucd.ie

Based on Marques-Silva, J.; Planes, J.: "Algorithms for maximum satisfiability using unsatisfiable cores," Design, Automation and Test in Europe, 2008. DATE '08, pp. 408–413, 10–14 March 2008 Doi: 10.1109/DATE.2008.4484715

S.P. Khatri, K. Gulati (eds.), *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, DOI 10.1007/978-1-4419-7518-8_10,

because the most effective SAT techniques cannot be applied directly to MaxSAT [9] (e.g., unit propagation).

Motivated by the recent and promising application of MaxSAT in EDA (e.g., [33]) this chapter proposes a novel algorithm for MaxSAT, `msu4`, that performs particularly well for large industrial instances. Instead of the usual algorithms for MaxSAT, the proposed algorithm exploits existing SAT solver technology and the ability of SAT solvers for finding unsatisfiable subformulas. Despite building on the work of others, on the relationship between maximally satisfiable and minimally unsatisfiable subformulas [6, 16, 20, 21, 24], the approach outlined in this chapter is new, in that unsatisfiable subformulas are used for guiding the search for the solution to the MaxSAT problem. The `msu4` algorithm builds on recent algorithms for the identification of unsatisfiable subformulas, which find other significant applications in EDA [32, 37]. The `msu4` algorithm also builds on recent work on solving PBO with SAT [15], namely on techniques for encoding cardinality constraints as Boolean circuits obtained from BDDs. The `msu4` algorithm differs from the one in [16] in the way unsatisfiable subformulas are manipulated and in the overall organization of the algorithm.

Experimental results, obtained on representative EDA industrial instances, indicate that in most cases the new `msu4` algorithm is orders of magnitude more efficient than the best existing MaxSAT algorithms. The `msu4` also opens a new line of research that tightly integrates SAT, unsatisfiable subformulas, and MaxSAT.

The chapter is organized as follows. The next section provides a brief overview of MaxSAT and existing algorithms. Section 10.3 describes the `msu4` algorithm and proves the correctness of the proposed approach. Section 10.4 provides experimental results, comparing `msu4` with alternative MaxSAT algorithms. The chapter concludes in Section 10.6.

10.2 Background

This section provides definitions and background knowledge for the MaxSAT problem. Due to space constraints, familiarity with SAT and related topics is assumed and the reader is directed to the bibliography [10].

10.2.1 The MaxSAT Problem

The maximum satisfiability (MaxSAT) problem can be stated as follows. Given an instance of SAT represented in CNF, compute an assignment that maximizes the number of satisfied clauses. During the last decade there has been a growing interest on studying MaxSAT, motivated by an increasing number of practical applications, including scheduling, routing, bioinformatics, and EDA [33].

Despite the clear relationship with the SAT problem, most modern SAT techniques cannot be applied directly to the MaxSAT problem. As a result, most

MaxSAT algorithms are built on top of the standard DPLL [12] algorithm and so do not scale for industrial problem instances [16, 17, 22, 23].

The usual approach (most of the solvers in the MaxSAT competition [3, 4]) is based on a Branch and Bound algorithm, emphasizing the computation of lower bounds and the application of inference rules that simplify the instance [17, 22, 23]. Results from the MaxSAT competition [3] suggest that algorithms based on alternative approaches (e.g., by converting MaxSAT into SAT) do not perform well. As a result, the currently best performing MaxSAT solvers are based on branch and bound with additional inference rules.

More recently, an alternative, in general incomplete, approach to MaxSAT has been proposed [33]. The motivation for this alternative approach is the potential application of MaxSAT in design debugging and the fact that existing MaxSAT approaches do not scale for industrial problem instances.

10.2.2 Solving MaxSAT with PBO

One alternative approach for solving the MaxSAT problem is to use pseudo-Boolean optimization (PBO) (e.g., [24]). The PBO approach for MaxSAT consists of adding a new (*blocking*) variable to each clause. The blocking variable b_i for clause ω_i allows satisfying clause ω_i independently of other assignments to the problem variables. The resulting PBO formulation includes a cost function, aiming at minimizing the number of blocking variables assigned value 1. Clearly, the solution of the MaxSAT problem is obtained by subtracting from the number of clauses the solution of the PBO problem.

Example 10.1 Consider the CNF formula: $\varphi = (x_1)\,(x_2 + \bar{x}_1)\,(\bar{x}_2)$. The PBO MaxSAT formulation consists of adding a new blocking clause to each clause. The resulting instance of SAT becomes $\varphi_W = (x_1 + b_1)\,(x_2 + \bar{x}_1 + b_2)\,(\bar{x}_2 + b_3)$, where b_1, b_2, b_3 denote blocking variables, one for each clause. Finally, the cost function for the PBO instance is min $\sum_{i=1}^{3} b_i$.

Despite its simplicity, the PBO formulation does not scale for industrial problems, since the large number of clauses results in a large number of blocking variables, and corresponding larger search space. Observe that, for most instances, the number of clauses exceeds the number of variables. For the resulting PBO problem, the number of variables equals the sum of the number of variables and clauses in the original SAT problem. Hence, the modified instance of SAT has a much larger search space.

10.2.3 Relating MaxSAT with Unsatisfiable Cores

In recent years there has been work on relating minimum unsatisfiable and maximally satisfiable subformulas [16, 20, 21, 24]. problem.

This section summarizes properties on the relationship between unsatisfiable cores and MaxSAT, which are used in the next section for developing msu4. Let φ be an unsatisfiable formula with a number of unsatisfiable cores, which may or may not be disjoint. Note that two cores are disjoint if the cores have no identical clauses. Let $|\varphi|$ denote the number of clauses in φ.

Proposition 10.1 (MaxSAT upper bound) *Let φ contain K disjoint unsatisfiable cores. Then $|\varphi| - K$ denotes an upper bound on the solution of the MaxSAT problem.*

Furthermore, suppose blocking variables are added to clauses in φ such that the resulting formula φ_W becomes satisfiable.

Proposition 10.2 (MaxSAT lower bound) *Let φ_W be satisfiable, and let B denote the set of blocking variables assigned value 1. Then $|\varphi| - |B|$ denotes a lower bound on the solution of the MaxSAT problem.*

Clearly, the solution to the MaxSAT problem lies between any computed lower and upper bound.

Finally, it should be observed that the relationship of unsatisfiable cores and MaxSAT was also explored in [16] in the context of partial MaxSAT. This algorithm, msu1, removes one unsatisfiable core each time by adding a fresh set of blocking variables to the clauses in each unsatisfiable core. A possible drawback of the algorithm of [16] is that it can add multiple blocking variables to each clause, an upper bound being the number of clauses in the CNF formula [30]. In contrast, the msu4 algorithm adds at most one additional blocking variable to each clause. Moreover, a number of algorithmic improvements to the algorithm of [16] can be found in [30], i.e., msu2 and msu3. The proposed improvements include linear encoding of the cardinality constraints and an alternative approach to reduce the number of blocking variables used.

10.3 A New MaxSAT Algorithm

This section develops the msu4 algorithm by building on the results of Section 10.2.3. As shown earlier, the major drawback of using a PBO approach for the MaxSAT problem is the large number of blocking variables that have to be used (essentially one for each original clause). For most benchmarks, the blocking variables end up being significantly more than the original variables, which is reflected in the cost function and overall search space. The large number of blocking variables basically renders the PBO approach ineffective in practice.

The msu4 algorithm attempts to reduce as much as possible the number of necessary blocking variables, thus simplifying the optimization problem being solved. Moreover, msu4 avoids interacting with a PBO solver and instead is fully SAT based.

10.3.1 Overview

Following the results of Section 10.2.3, consider identifying disjoint unsatisfiable cores of φ. This can be done by iteratively computing unsatisfiable cores and adding blocking variables to the clauses in the unsatisfiable cores. The identification and blocking of unsatisfiable cores are done on a working formula φ_W. Eventually, a set of disjoint unsatisfiable cores is identified, and the blocking variables allow satisfying φ_W. From Proposition 10.2, this represents a lower bound on the solution of the MaxSAT problem. This lower bound can be refined by requiring fewer blocking variables to be assigned value 1. This last condition can be achieved by adding a cardinality constraint to φ[1].

The resulting formula can still be satisfiable, in which case a further refined cardinality constraint is added to φ_W. Alternatively, the formula is unsatisfiable. In this case, some clauses of φ without blocking variables may exist in the unsatisfiable core. If this is the case, each clause is augmented with a blocking variable, and a new cardinality constraint can be added to φ_W, which requires the number of blocking variables assigned value 1 to be less than the total number of new blocking clauses. Alternatively, the core contains no original clause without a blocking variable. If this is the case, then the highest computed lower bound is returned as the solution to the MaxSAT problem. The proof that this is indeed the case is given below.

In contrast with the algorithms in [16] and [30], the `msu4` algorithm is not exclusively based on enumerating unsatisfiable cores. The `msu4` algorithm also identifies satisfiable instances, which are then eliminated by adding additional cardinality constraints.

10.3.2 The Algorithm

Following the ideas of the previous section, the pseudocode for `msu4` is shown in Algorithm 5. The `msu4` algorithm works as follows. The main loop (lines 8–33) starts by identifying disjoint unsatisfiable cores. The clauses in each unsatisfiable core are modified so that any clause ω_i in the core can be satisfied by setting to 1 a new auxiliary variable b_i associated with ω_i. Consequently, a number of properties of the MaxSAT problem can be inferred. Let $|\varphi|$ denote the number of clauses, let ν_U represent the number of iterations of the main loop in which the SAT solver outcome is unsatisfiable, and let μ_{BV} denote the smallest of the number of blocking variables assigned value 1 each time φ_W becomes satisfiable. Then, an upper bound for the MaxSAT problem is $|\varphi|-\nu_U$, and a lower bound is $|\varphi|-\mu_{BV}$. Both the lower and the upper bounds provide approximations to the solution of the MaxSAT problem, and the difference between the two bounds provides an indication on the number of iterations. Clearly, the MaxSAT solution will require *at most* μ_{BV} blocking variables to be assigned value 1. Also, each time the SAT solver declares the CNF formula

[1] Encodings of cardinality constraints are studied, for example, in [15].

to be unsatisfiable, then the number of blocking variables that must be assigned value 1 can be increased by 1. Each time φ_W becomes satisfiable (line 25), a new cardinality constraint is generated (line 30), which requires the number of blocking variables assigned value 1 to be reduced given the current satisfying assignment (and so requires the lower bound to be increased, if possible). Alternatively, each time φ_W is unsatisfiable (line 12), the unsatisfiable core is analyzed. If there exist initial clauses in the unsatisfiable core, which do not have blocking variables, then additional blocking variables are added (line 17). Formula φ_W is updated accordingly by removing the original clauses and adding the modified clauses (line 18). A cardinality constraint is added to require at least one of the blocking clauses to be assigned value 1 (line 19). Observe that this cardinality constraint is in fact optional, but experiments suggest that it is most often useful. If φ_W is unsatisfiable, and no additional original clauses can be identified, then the solution to the MaxSAT problem has been identified (line 22). Also, if the lower bound and upper bound estimates become equal (line 32), then the solution to the MaxSAT problem has also been identified. Given the previous discussion, the following result is obtained.

Proposition 10.3 *Algorithm 5 gives the correct MaxSAT solution.*

Proof The algorithm iteratively identifies unsatisfiable cores and adds blocking variables to the clauses in each unsatisfiable core that do not yet have blocking variables (i.e., *initial clauses*), until the CNF formula becomes satisfiable. Each computed solution represents an upper bound on the number of blocking variables assigned value 1, and so it also represents a lower bound on the MaxSAT solution. For each computed solution, a new cardinality constraint is added to the formula (see line 30), requiring a smaller number of blocking variables to be assigned value 1. If the algorithm finds an unsatisfiable core containing no more initial clauses without blocking variables, then the algorithm can terminate and the last computed upper bound represents the MaxSAT solution. Observe that in this case the same unsatisfiable core C can be generated, even if blocking clauses are added to other original clauses without blocking clauses. As a result, the existing lower bound is the solution to the MaxSAT problem. Finally, note that the optional auxiliar constraint added in line 19 does not affect correctness, since it solely requires an existing unsatisfiable core not to be re-identified.

10.3.3 A Complete Example

This section illustrates the operation of the `msu4` algorithm on a small example formula.

Example 10.2 Consider the following CNF formula:

$$\begin{aligned}\varphi = {} & \omega_1 \cdot \omega_2 \cdot \omega_3 \cdot \omega_4 \cdot \omega_5 \cdot \omega_6 \cdot \omega_7 \cdot \omega_8 \\ & (x_1)\,(\bar{x}_1 + \bar{x}_2)\,(x_2)\,(\bar{x}_1 + \bar{x}_3)\,(x_3)\,(\bar{x}_2 + \bar{x}_3) \\ & (x_1 + \bar{x_4})\,(\bar{x}_1 + x_4)\end{aligned}$$

Algorithm 5 The `msu4` algorithm

MSU4(φ)

$\triangleright$ Clauses of CNF formula φ are the *initial* clauses
$\varphi_W \leftarrow \varphi$ $\triangleright$ Working formula, initially set to φ
$\mu_{BV} \leftarrow |\varphi|$ $\triangleright$ Min blocking variables w/ value 1
$\nu_U \leftarrow 0$ $\triangleright$ Iterations w/ unsat outcome
$V_B \leftarrow \emptyset$ $\triangleright$ IDs of blocking variables
$UB \leftarrow |\varphi| + 1$ $\triangleright$ Upper bound estimate
$LB \leftarrow 0$ $\triangleright$ Lower bound estimate
while true
  do (st, φ_C) $\leftarrow$ SAT(φ_W)
    $\triangleright$ φ_C is an unsat core if φ_W is unsat
    if st = **UNSAT**
      then
        $\varphi_I = \varphi_C \cap \varphi$ $\triangleright$ Initial clauses in core
        $I \leftarrow \{i \mid \omega_i \in \varphi_I\}$
        $V_B \leftarrow V_B \cup I$
        if $|I| > 0$
          then $\varphi_N \leftarrow \{\omega_i \cup \{b_i\} \mid \omega_i \in \varphi_I\}$
            $\varphi_W \leftarrow (\varphi_W - \varphi_I) \cup \varphi_N$
            $\varphi_T \leftarrow$ CNF($\sum_{i \in I} b_i \geq 1$)
            $\varphi_W \leftarrow \varphi_W \cup \varphi_T$
          else $\triangleright$ Solution to MaxSAT problem
            return LB
        $\nu_U \leftarrow \nu_U + 1$
        $UB \leftarrow |\varphi| - \nu_U$ $\triangleright$ Refine UB
      else
        $\nu \leftarrow$ | blocking variables w/ value 1 |
        if $\mu_{BV} < \nu$
          then $\mu_{BV} \leftarrow \nu$
            $LB \leftarrow |\varphi| - \mu_{BV}$ $\triangleright$ Refine LB
        $\varphi_T \leftarrow$ CNF($\sum_{i \in V_B} b_i \leq \mu_{BV} - 1$)
        $\varphi_W \leftarrow \varphi_W \cup \varphi_T$
    if $LB = UB$ $\triangleright$ Solution to MaxSAT problem
      then return LB

Initially φ_W contains all the clauses in φ. In the first loop iteration, the core $\omega_1, \omega_2, \omega_3$ is identified. As a result, the new blocking variables b_1, b_2, and b_3 are added, respectively, to clauses ω_1, ω_2, and ω_3, and the CNF encoding of the cardinality constraint $b_1 + b_2 + b_3 \geq 1$ is also (optionally) added to φ_W. In the second iteration, φ_W is satisfiable, with $b_1 = b_3 = 1$. As a result, the CNF encoding of a new cardinality constraint, $b_1 + b_2 + b_3 \leq 1$, is added to φ_W. For the next iteration, φ_W is unsatisfiable and the clauses ω_4, ω_5, and ω_6 are listed in the unsatisfiable core. As a result, the new blocking variables b_4, b_5, and b_6 are added, respectively, to clauses ω_4, ω_5, and ω_6, and the CNF encoding of the cardinality constraint $b_4 + b_5 + b_6 \geq 1$ is also (optionally) added to φ_W. In this iteration, since the lower and the upper bounds become equal, then the algorithm terminates, indicating that two blocking variables need to be assigned value 1, and the MaxSAT solution is 6.

From the example, it is clear that the algorithm efficiency depends on the ability for finding unsatisfiable formulas effectively and for generating manageable cardinality constraints. In the implementation of msu4, the cardinality constraints were encoded either with BDDs or with sorting networks [15].

10.4 Experimental Results

The msu4 algorithm described in the previous section has been implemented on top of MiniSAT [14]. Version 1.14 of MiniSAT was used, for which an unsatisfiable core extractor was available. Two versions of msu4 are considered, one (v1) uses BDDs for representing the cardinality constraints and the other (v2) uses sorting networks [15].

All results shown below were obtained on a 3.0 GHz Intel Xeon 5160 with 4 GB of RAM running RedHat Linux. A time-out of 1000 s was used for all MaxSAT solvers considered. The memory limit was set to 2 GB. The MaxSAT solvers evaluated are the best performing solver in the MaxSAT evaluation [3], maxsatz [23], minisat+ [15] for the MaxSAT PBO formulation, and finally msu4. Observe that the algorithm in [16] targets partial MaxSAT, and so performs poorly for MaxSAT instances [3, 30].

In order to evaluate the new MaxSAT algorithm, a set of industrial problem instances was selected. These instances were obtained from existing unsatisfiable subsets of industrial benchmarks, obtained from the SAT competition archives and from SATLIB [7, 18]. The majority of instances considered was originally from EDA applications, including model checking, equivalence checking, and test-pattern generation. Moreover, MaxSAT instances from design debugging [33] were also evaluated. The total number of unsatisfiable instances considered was 691.

Table 10.1 shows the number of aborted instances for each algorithm. As can be concluded, for practical instances, existing MaxSAT solvers are ineffective. The use of the PBO model for MaxSAT performs better than maxsatz, but aborts more instances than either version of msu4. It should be noted that the PBO approach uses minisat+, which is based on a more recent version of MiniSAT than msu4.

Table 10.1 Number of aborted instances

Total	maxsatz	pbo	msu4 v1	msu4 v2
691	554	248	212	163

Figures 10.1, 10.2, and 10.3 show scatter plots comparing maxsatz, the PBO formulation, and msu4 v1 with msu4 v2. As can be observed, the two versions of msu4 are clearly more efficient than either maxsatz or minisat+ on the MaxSAT formulations. Despite the performance advantage of both versions of msu4, there are exceptions. With few outliers, maxsatz can only outperform msu4 v2 on instances where both algorithms take less than 0.1 s. In contrast, minisat+ can outperform msu4 v2 on a number of instances, in part because of the more recent version of MiniSAT used in minisat+.

Fig. 10.1 Scatter plot: maxsatz vs. msu4-v2

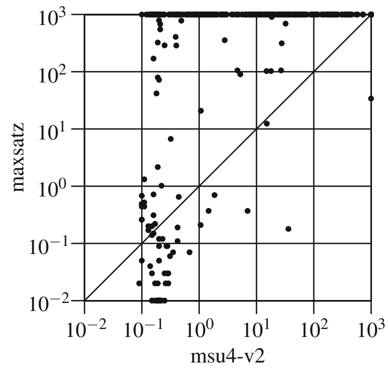


Fig. 10.2 Scatter plot: pbo vs. msu4-v2

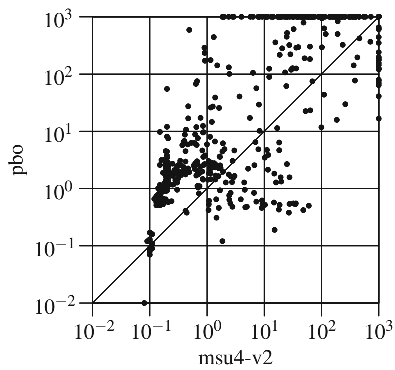
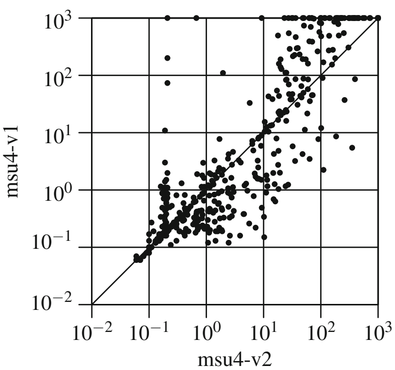


Fig. 10.3 Scatter plot: msu4-v1 vs. msu4-v2



Finally, Table 10.2 summarizes the results for design debugging instances [33]. As can be concluded, both versions of `msu4` are far more effective than either `maxsatz` or `minisat+` on the PBO model for MaxSAT.

Table 10.2 Design debugging instances

Total	maxsatz	pbo	msu4 v1	msu4 v2
29	26	21	3	3

10.5 Related Work

The use of unsatisfiable subformulas for solving (partial) MaxSAT problems was first proposed by Fu and Malik [16]. This algorithm is referred to as `msu1.0`. This work was extended in a number of different ways in our own work [29–31]. `msu4`, the algorithm described in this chapter, was first proposed in [31], whereas `msu3` was first described in [30]. In addition, `msu2` as well as different variations of `msu1.0` (namely, `msu1.1` and `msu1.2`) were proposed in [29]. There has been additional work on unsatisfiability-based MaxSAT [1, 28]. A new algorithm for partial MaxSAT was proposed in [1]. Finally, algorithms for weighted partial MaxSAT were proposed in [1, 2, 28].

Besides dedicated unsatisfiability-based algorithms for MaxSAT, this work has motivated its application in a number of areas. Unsatisfiability-based MaxSAT algorithms motivated the development of similar algorithms for computing the minimal correction sets (MCSes) [25]. The use of CNF encodings in unsatisfiability-based MaxSAT algorithms motivated work on improved encodings for cardinality constraints [5]. Finally, one concrete application area where the best solution is given by unsatisfiability-based MaxSAT algorithms is design debugging of digital circuits [11, 33, 35].

10.6 Conclusions

Motivated by the recent application of maximum satisfiability to design debugging [33], this chapter proposes a new MaxSAT algorithm, `msu4`, that further exploits the relationship between unsatisfiable formulas and maximum satisfiability [6, 16, 20, 21, 24]. The motivation for the new MaxSAT algorithm is to solve large industrial problem instances, including those from design debugging [33]. The experimental results indicate that `msu4` performs in general significantly better than either the best performing MaxSAT algorithm [3] or the PBO formulation of the MaxSAT problem [24].

For a number of industrial classes of instances, which modern SAT solvers solve easily but which existing MaxSAT solvers are unable to solve, `msu4` is able to find solutions in reasonable time. Clearly, `msu4` is effective only for instances for which SAT solvers are effective at identifying small unsatisfiable cores and from which manageable cardinality constraints can be obtained.

Despite the promising results, additional improvements to `msu4` are expected. One area for improvement is to exploit alternative SAT solver technology. `msu4` is based on MiniSAT 1.14 (due to the core generation code), but more recent SAT solvers could be considered. Another area for improvement is considering alternative encodings of cardinality constraints, given the performance differences observed for the two encodings considered. Finally, the interplay between different algorithms based on unsatisfiable core identification (i.e., msu1 [16] and msu2 and msu3 [30]) should be further developed.

References

1. Ansótegui, C., Bonet, M.L., Levy, J.: Solving (weighted) partial MaxSAT through satisfiability testing. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 427–440. Swansea, UK (2009)
2. Ansótegui, C., Bonet, M.L., Levy, J.: A new algorithm for weighted partial MaxSAT. In: National Conference on Artificial Intelligence. Atlanta, USA (2010)
3. Argelich, J., Li, C.M., Manyá, F., Planes, J.: MaxSAT evaluation. http://www.maxsat07.udl.es/ (2008)
4. Argelich, J., Li, C.M., Manyá, F., Planes, J.: The first and second Max-SAT evaluations. Journal on Satisfiability, Boolean Modeling and Computation **4**, 251–278 (2008)
5. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality networks and their applications. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 167–180. Swansea, UK (2009)
6. de la Banda, M.G., Stuckey, P.J., Wazny, J.: Finding all minimal unsatisfiable sub-sets. In: International Conference on Principles and Practice of Declarative Programming, pp. 32–43. Uppsala, Sweden (2003)
7. Berre, D.L., Simon, L., Roussel, O.: SAT competition. http://www.satcompetition.org/ (2008)
8. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: Tools and Algorithms for the Construction and Analysis of Systems, pp. 193–207. (1999)
9. Bonet, M.L., Levy, J., Manyá, F.: Resolution for Max-SAT. Artificial Intelligence **171**(8–9), 606–618 (2007)
10. Bordeaux, L., Hamadi, Y., Zhang, L.: Propositional satisfiability and constraint programming: A comparative survey. ACM Computing Surveys **38**(4) (2006)
11. Chen, Y., Safarpour, S., Veneris, A.G., Marques-Silva, J.: Spatial and temporal design debug using partial MaxSAT. In: ACM Great Lakes Symposium on VLSI, pp. 345–350. Boston, USA (2009)
12. Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Communications of the ACM **5**, 394–397 (1962)
13. Dershowitz, N., Hanna, Z., Katz, J.: Bounded model checking with QBF. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 408–414. St. Andrews, UK (2005)
14. Een, N., Sörensson, N.: An extensible SAT solver. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 502–518. Santa Margherita Ligure, Italy (2003)
15. Een, N., Sörensson, N.: Translating pseudo-Boolean constraints into SAT. Journal on Satisfiability, Boolean Modeling and Computation **2**, 1–26 (2006)
16. Fu, Z., Malik, S.: On solving the partial MAX-SAT problem. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 252–265. Seattle, USA (2006)
17. Heras, F., Larrosa, J., Oliveras, A.: MiniMaxSat: a new weighted Max-SAT solver. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 41–55. Lisbon, Portugal (2007)
18. Hoos, H., Stützle, T.: SAT lib. http://www.satlib.org/ (2008)
19. Kuehlmann, A., Paruthi, V., Krohm, F., Ganai, M.K.: Robust Boolean reasoning for equivalence checking and functional property verification. IEEE Transactions on CAD of Integrated Circuits and Systems **21**(12), 1377–1394 (2002)

20. Kullmann, O.: Investigations on autark assignments. Discrete Applied Mathematics **107**(1–3), 99–137 (2000)
21. Kullmann, O.: Lean clause-sets: generalizations of minimally unsatisfiable clause-sets. Discrete Applied Mathematics **130**(2), 209–249 (2003)
22. Li, C.M., Manyá, F., Planes, J.: Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In: National Conference on Artificial Intelligence, pp. 86–91. Boston, USA (2006)
23. Li, C.M., Manyá, F., Planes, J.: New inference rules for Max-SAT. Journal of Artificial Intelligence Research **30**, 321–359 (2007)
24. Liffiton, M.H., Sakallah, K.A.: On finding all minimally unsatisfiable subformulas. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 173–186. (2005)
25. Liffiton, M.H., Sakallah, K.A.: Generalizing core-guided max-sat. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 481–494. Swansea, UK (2009)
26. Liu, C., Kuehlmann, A., Moskewicz, M.W.: CAMA: A multi-valued satisfiability solver. In: International Conference on Computer-Aided Design, pp. 326–333. San Jose, USA (2003)
27. Mangassarian, H., Veneris, A.G., Safarpour, S., Najm, F.N., Abadir, M.S.: Maximum circuit activity estimation using pseudo-Boolean satisfiability. In: Design, Automation and Testing in Europe Conference, pp. 1538–1543. Nice, France (2007)
28. Manquinho, V., Marques-Silva, J., Planes, J.: Algorithms for weighted Boolean optimization. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 495–508. Swansea, UK (2009)
29. Marques-Silva, J., Manquinho, V.: Towards more effective unsatisfiability-based maximum satisfiability algorithms. In: International Conference on Theory and Applications of Satisfiability Testing, pp. 225–230. Guangzhou, China (2008)
30. Marques-Silva, J., Planes, J.: On using unsatisfiability for solving maximum satisfiability. http://arxiv.org/corr/ (2007)
31. Marques-Silva, J., Planes, J.: Algorithms for maximum satisfiability using unsatisfiable cores. In: Design, Automation and Testing in Europe Conference, pp. 408–413. Munich, Germany (2008)
32. McMillan, K.L.: Interpolation and SAT-based model checking. In: Computer-Aided Verification, pp. 1–13. Boulders, USA (2003)
33. Safarpour, S., Mangassarian, H., Veneris, A.G., Liffiton, M.H., Sakallah, K.A.: Improved design debugging using maximum satisfiability. In: Formal Methods in Computer-Aided Design, pp. 13–19. Austin, USA (2007)
34. Smith, A., Veneris, A.G., Ali, M.F., Viglas, A.: Fault diagnosis and logic debugging using Boolean satisfiability. IEEE Transactions on CAD of Integrated Circuits and Systems **24**(10), 1606–1621 (2005)
35. Sülflow, A., Fey, G., Bloem, R., Drechsler, R.: Using unsatisfiable cores to debug multiple design errors. In: ACM Great Lakes Symposium on VLSI, pp. 77–82. Orlando, USA (2008)
36. Wang, K.H., Chan, C.M.: Incremental learning approach and SAT model for Boolean matching with don't cares. In: International Conference on Computer-Aided Design, pp. 234–239. San Jose, USA (2007)
37. Zhang, L., Malik, S.: Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In: Design, Automation and Testing in Europe Conference. Munich, Germany (2003)