Exact MinSAT Solving*

Chu Min Li[1,2], Felip Manyà[3], Zhe Quan[2], and Zhu Zhu[2]

[1] Hunan Normal University, Changsha, China
[2] MIS, Université de Picardie Jules Verne, Amiens, France
[3] Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain

Abstract. We present an original approach to exact MinSAT solving based on solving MinSAT using MaxSAT encodings and MaxSAT solvers, and provide empirical evidence that our generic approach is competitive.

1 Introduction

MinSAT is the problem of finding a truth assignment that minimizes the number of satisfied clauses in a CNF formula. Despite that MaxSAT has focused the interest of the SAT community in recent years [LM09], we believe that it is worth studying MinSAT and, in particular, to devise fast exact MinSAT solving techniques. It has both theoretical and practical interest: From the theoretical point of view, we highlight the existing work on approximation algorithms for MinSAT (see [MR96] and the references therein); and from the practical point of view, we emphasize its applicability in areas such as Bioinformatics [GKZ05].

Since there exists no exact MinSAT solver similar to the modern branch-and-bound solvers developed for MaxSAT, we believe that it is worth exploring solving MinSAT using a generic problem solving approach. Our proposal relies on defining efficient and original encodings from MinSAT into MaxSAT, solving the resulting MaxSAT encodings with a modern MaxSAT solver, and then derive a MinSAT optimal solution from a MaxSAT optimal solution. To be more precise, we define three encodings. The first one is a straightforward reduction of MinSAT into Partial MaxSAT. The other two encodings are more involved because they first reduce MinSAT to find an optimal MaxClique solution in a graph that represents the interactions among the clauses of the MinSAT instance, and then reduce MaxClique to Partial MaxSAT. The difference between both encodings is that one uses the usual encoding from MaxClique into Partial MaxSAT, while the other uses a novel encoding which is based on identifying a minimum clique partition. Moreover, we performed an empirical investigation that shows that our approach is competitive.

The paper is structured as follows. In Section 2 we define three encoding from MinSAT into Partial MaxSAT. In Section 3 we report on the empirical evaluation we have conducted in order to evaluate our approach to MinSAT solving. We refer to [LM09] for the basic definitions of MaxSAT, and to [MR96] for the used definitions of graphs.

* Research supported by Generalitat de Catalunya (2009-SGR-1434), and *Ministerio de Ciencia e Innovación* (CONSOLIDER CSD2007-0022, INGENIO 2010,TIN2007-68005-C04-04, Acción Integrada HA2008-0017). Acknowledgements: The MinSAT problem was originally asked by Laurent Simon to the first author.

2 Encodings

Definition 1. *Given a MinSAT instance I consisting of the clause set $C_I = \{C_1, \ldots, C_m\}$ and variable set X_I, the direct MaxSAT encoding of I is defined as follows: (i) The set of propositional variables is $X_I \cup \{c_1, \ldots, c_m\}$, where $\{c_1, \ldots, c_m\}$ is a set of auxiliary variables; (ii) for every clause $C_i \in C_I$, the hard clause $c_i \leftrightarrow C_i$ is added; and (iii) for every auxiliary variable c_i, the unit soft clause $\neg c_i$ is added.*

The hard part establishes that c_i is true iff C_i is satisfied. In the soft part, the number of unsatisfied clauses in C_I is maximized. Therefore, if the auxiliary variables set to true in an optimal solution of the MaxSAT encoding are $\{c_{i_1}, \ldots, c_{i_k}\}$, then $\{C_{i_1}, \ldots, C_{i_k}\}$ is a minimum set of satisfied clauses in the MinSAT instance I.

Once we have defined the encoding based on reducing MinSAT to Partial MaxSAT, we define two encodings based on reducing MinSAT to MaxClique, and then MaxClique to Partial MaxSAT. First, we introduce the concept of auxiliary graph:

Let I be a MinSAT instance consisting of the clause set C_I and variable set X_I. The *auxiliary graph* $G_I(V_I, E_I)$ corresponding to I is constructed as follows: the vertex set V_I is in one-to-one correspondence with the clause set C_I; in the sequel, vertex c_i corresponds to clause C_i. For any two vertices c_i and c_j in V_I, the edge $\{c_i, c_j\}$ is in E_I iff the corresponding clauses C_i and C_j are such that there is a variable $x \in X_I$ that appears in uncomplemented form in C_i and in complemented form in C_j, or vice versa. The *complement of an auxiliary graph* G_I is a graph $\overline{G_I}$ on the same vertices such that two vertices of $\overline{G_I}$ are adjacent iff they are not adjacent in G_I.

It was proved in [MR96] that the number of clauses of a MinSAT instance I satisfied by an optimal assignment is equal to the cardinality of a minimum vertex cover for the auxiliary graph G_I. On the other hand, the set of vertices not belonging to a maximum clique in the complement graph $\overline{G_I}$ are a minimum vertex cover of G_I. This follows from the fact that, for any graph $G(V, E)$, $V' \subseteq V$ is a vertex cover iff $V - V'$ in a clique in the complement graph of G. Therefore, the number of clauses of a MinSAT instance I satisfied by an optimal assignment is equal to the total number of vertices minus the cardinality of a maximum clique in $\overline{G_I}$.

Moreover, an optimal assignment for the MinSAT instance I can be derived from a minimum vertex cover for G_I as follows [MR96]: The variables occurring in clauses corresponding to vertices not belonging to the minimum vertex cover must be assigned in such a way that these clauses are set to false, which is possible because, by construction of G_I, these clauses do not contain both x and $\bar{x}$ for any $x \in X_I$; the rest of variables are assigned to an arbitrary value. Therefore, we can derive an optimal assignment for the MinSAT instance I from a maximum clique of $\overline{G_I}$: The variables occurring in clauses corresponding to vertices belonging to the maximum clique must be assigned in such a way that these clauses are set to false; the rest of variables are assigned to an arbitrary value. Therefore, in order to find an optimal assignment for the MinSAT instance I, we can search for a maximum clique in $\overline{G_I}$. The MaxClique problem for $\overline{G_I}$ can be naturally encoded into a Partial MaxSAT problem as follows.

Definition 2. *Given a MinSAT instance I consisting of the clause set $C_I = \{C_1, \ldots, C_m\}$, the MaxClique-based MaxSAT encoding of I is defined as follows: (i) The set of propositional variables is $\{c_1, \ldots, c_m\}$; (ii) for every two clauses C_i, C_j in C_I such* *that C_i contains an occurrence of a literal l and C_j contains an occurrence of $\neg l$, the hard clause $\neg c_i \vee \neg c_j$ is added; and (iii) for every propositional variable c_i, the unit soft clause c_i is added.*

Observe that there is a hard clause for every two non-adjacent vertices in $\overline{G_I}$ encoding that the two vertices cannot be in the same clique, and a soft unit clause for every vertex in $\overline{G_I}$. The set of propositional variables evaluated to true in an optimal assignment of the resulting Partial MaxSAT problem forms a maximum clique in $\overline{G_I}$.

The MaxClique-based MaxSAT encoding can be improved by reducing the number of soft clauses by taking into account the following fact: If the auxiliary graph of a MinSAT instance contains a clique $C = \{c_{i_1}, \ldots, c_{i_k}\}$, then the hard part contains the clauses $\neg c_{i_j} \vee \neg c_{i_k}$ for all i, j such that $1 \leq i < j \leq k$. Observe that these clauses encode the following at-most-one condition: There is at most one literal in $\{c_{i_1}, \ldots, c_{i_k}\}$ that evaluates to true. In other words, they encode that any feasible solution assigns to true at most one variable of the subset of variables $\{c_{i_1}, \ldots, c_{i_k}\}$. Therefore, we can replace the k unit soft clauses $c_{i_1}, \ldots, c_{i_k}$ with the soft clause $c_{i_1} \vee \cdots \vee c_{i_k}$ because the number of satisfied clauses is the same in both encodings for any feasible assignment.

Definition 3. *Let I be a MinSAT instance, let $G_I(V_I, E_I)$ be the auxiliary graph of I, and let $V_1, \ldots, V_k$ be a clique partition of G_I. The improved MaxClique-based MaxSAT encoding of I is obtained from the MaxClique-based MaxSAT encoding of Definition 2 by replacing, for every clique $V_i = \{c_{i_1}, \ldots, c_{i_k}\}$ in the partition, the soft unit clauses $c_{i_1}, \ldots, c_{i_k}$ with the soft unit clause $c_{i_1} \vee \cdots \vee c_{i_k}$.*

Example 1. Let I be the MinSAT instance $\{C_1, C_2, C_3, C_4, C_5\}$, where $C_1 = a \vee b$, $C_2 = a \vee \neg b$, $C_3 = \neg a \vee b$, $C_4 = \neg a \vee \neg b$, and $C_5 = a \vee c$. The direct MaxSAT encoding of I is formed by the hard clauses $c_1 \leftrightarrow a \vee b, c_2 \leftrightarrow a \vee \neg b, c_3 \leftrightarrow \neg a \vee b, c_4 \leftrightarrow \neg a \vee \neg b, c_5 \leftrightarrow a \vee c$, and the soft clauses $\neg c_1, \neg c_2, \neg c_3, \neg c_4, \neg c_5$. The MaxClique-based MaxSAT encoding of I is formed by the hard clauses $\neg c_1 \vee \neg c_2, \neg c_1 \vee \neg c_3, \neg c_1 \vee \neg c_4, \neg c_2 \vee \neg c_3, \neg c_2 \vee \neg c_4, \neg c_3 \vee \neg c_4, \neg c_3 \vee \neg c_5, \neg c_4 \vee \neg c_5$, and the soft unit clauses c_1, c_2, c_3, c_4, c_5. Since $\{\{c_1, c_2, c_3, c_4\}, \{c_5\}\}$ is a clique partition of the auxiliary graph of I, an improved MaxClique-based MaxSAT encoding of I is obtained by replacing the soft clauses of the previous encoding with the following soft clauses: $c_1 \vee c_2 \vee c_3 \vee c_4, c_5$.

Since finding a minimum clique partition is NP-hard, we propose to apply a heuristic clique partition method for deriving the improved encoding. The heuristic used in our empirical investigation is Algorithm 1. Given a graph G and a list P of disjoint cliques in G, where P is initially empty, let #c(v) denote the number of cliques in P in which vertex v can be inserted to enlarge them. Algorithm 1 selects the vertex v with the minimum #c(v), breaking ties in favor of the vertex with the smallest degree (i.e., with the minimum number of adjacent vertices). Let C be a clique in P, and let v_i and v_j be two non-adjacent vertices that can be respectively inserted into C to get a larger clique, observe that the insertion of v_i into C prevents v_j from being inserted into C, and vice versa. The intuition behind the selection of v in Algorithm 1 is that the most constrained vertex (i.e., with the fewest possibilities in P) is inserted first, avoiding to create a new clique for this vertex after inserting other vertices. The selected v is then inserted into a clique in P to enlarge this clique. If this clique does not exist, a new clique is created and v is inserted there.

Algorithm 1. cliquePartition(G)

```
Input: A graph $G$=($V$, $E$)
Output: A clique partition of $G$
1  begin
2      $P \leftarrow \emptyset$;
3      while $G$ is not empty do
4          For each vertex $v$ in $G$, compute the number #c of cliques in $P$ in which $v$ is
           adjacent to all vertices;
5          $v \leftarrow$ the vertex of $G$ with the minimum #c, breaking ties in favor of the vertex
           with the smallest degree;
6          remove $v$ from $G$;
7          if there is a clique $C$ in $P$ in which $v$ is adjacent to all vertices then
8              insert $v$ into $C$;
9          else
10             create a new clique $C$;
11             insert $v$ into $C$;
12             $P \leftarrow P \cup \{C\}$;
13     return $P$;
14 end
```

3 Experimental Results

We conducted an empirical comparison of the three proposed encodings on several MaxSAT solvers, and also compared our results with the results obtained by solving MinSAT with two of the best performing state-of-the-art exact MaxClique solvers.

The MaxSAT solvers used in our empirical investigation are the versions that that participated in the 2009 MaxSAT Evaluation of the following solvers: WBO [MSP09], MaxSatz [LMP07][1], PM2 [ABL09], and SAT4J-Maxsat[2]. The MaxClique solvers used in our empirical investigation are: MaxCliqueDyn [KJ07][3], and Cliquer [Ost02][4].

The MaxClique solvers MaxCliqueDyn and Cliquer solve a MinSAT instance I by computing a maximum clique in the complement auxiliary graph $\overline{G_I}$. The MaxSAT solvers MaxSatz, WBO, PM2, and SAT4J-Maxsat solve I using the three encodings of MinSAT into Partial MaxSAT defined in the previous sections.

As benchmarks, we used randomly generated Min-2-SAT and Min-3-SAT instances. The number of variables in the instances ranged from 40 to 100, and the clause-to-variable ratios solved for Min-2-SAT were 1 and 3, and the clause-to-variable (C/V)

[1] We introduced a small modification in the heuristic used to select the variables to which failed literal detection is applied in the lower bound computation: instead of applying it to all the variables having both at least two negative and two positive occurrences in binary clauses, it is now applied to all the variables having at least one positive occurrence and at least two negative occurrences in binary clauses, and to all the variables having at least one negative occurrence and at least two positive occurrences in binary clauses.

[2] http://www.sat4j.org/

[3] We got the source code of this solver from D. Janezic in January 2010.

[4] We used the last publicly available version of this solver: http://users.tkk.fi/pat/cliquer.html

Table 1. Number of solved instances and mean time (seconds) of MaxSatz, MaxCliqueDyn (Dyn), Cliquer (Clq), PM2, WBO and SAT4J-Maxsat on random Min-2-SAT

instance		*MaxSatz*			*Dyn*	*Clq*	*PM2*			*WBO*			*sat4j-maxsat*		
#var	C/V	$E1$	$E2$	$E3$			$E1$	$E2$	$E3$	$E1$	$E2$	$E3$	$E1$	$E2$	$E3$
90	1.0	0.58	9.38	**0.00**	0.02	**0.00**	0.01	**0.00**	**0.00**	0.02	0.01	**0.00**	2710	1995	0.08
		(50)	*(50)*	***(50)***	*(50)*	***(50)***	*(50)*	***(50)***	***(50)***	*(50)*	*(50)*	***(50)***	*(24)*	*(45)*	*(50)*
100	1.0	0.26	27.34	**0.00**	0.02	**0.00**	0.01	0.01	**0.00**	0.02	0.01	**0.00**	2596	6358	0.11
		(50)	*(50)*	***(50)***	*(50)*	***(50)***	*(50)*	*(50)*	***(50)***	*(50)*	*(50)*	***(50)***	*(14)*	*(12)*	*(50)*
40	3.0	0.12	52.95	0.01	0.03	0.01	0.19	0.51	**0.00**	0.07	0.52	**0.00**	9904	-	0.20
		(50)	*(50)*	*(50)*	*(50)*	*(50)*	*(50)*	*(50)*	***(50)***	*(50)*	*(50)*	***(50)***	*(4)*	*(0)*	*(50)*
50	3.0	0.89	2053	0.01	0.04	1.23	5.25	7.89	**0.00**	42.35	51.12	0.01	-	-	0.43
		(50)	*(48)*	*(50)*	*(50)*	*(50)*	*(50)*	*(50)*	***(50)***	*(50)*	*(50)*	*(50)*	*(0)*	*(0)*	*(50)*
60	3.0	7.42	-	0.02	0.07	15.88	49.35	79.77	**0.01**	207.6	93.17	0.20	-	-	2.41
		(50)	*(0)*	*(50)*	*(50)*	*(50)*	*(50)*	*(50)*	***(50)***	*(50)*	*(50)*	*(50)*	*(0)*	*(0)*	*(50)*
70	3.0	43.25	-	0.03	0.13	68.01	144.3	135.8	**0.01**	225.1	143.9	0.08	-	-	2.42
		(50)	*(0)*	*(50)*	*(50)*	*(50)*	*(50)*	*(50)*	***(50)***	*(50)*	*(50)*	*(50)*	*(0)*	*(0)*	*(50)*
80	3.0	201.2	-	0.05	0.25	302.4	199.9	231.8	**0.02**	416.8	432.9	0.70	-	-	3.35
		(50)	*(0)*	*(50)*	*(50)*	*(50)*	*(50)*	*(50)*	***(50)***	*(50)*	*(50)*	*(50)*	*(0)*	*(0)*	*(50)*
90	3.0	1355	-	0.08	0.91	895.5	353.1	330.3	**0.05**	558.8	295.1	1.37	-	-	5.38
		(49)	*(0)*	*(50)*	*(50)*	*(42)*	*(50)*	*(50)*	***(50)***	*(50)*	*(50)*	*(50)*	*(0)*	*(0)*	*(50)*
100	3.0	3258	-	0.11	1.51	1185	256.7	455.3	**0.09**	548.5	665.9	29.27	-	-	96.91
		(46)	*(0)*	*(50)*	*(50)*	*(44)*	*(10)*	*(50)*	***(50)***	*(50)*	*(50)*	*(50)*	*(0)*	*(0)*	*(50)*

ratios solved for Min-3-SAT were 4.25 and 5. Experiments were performed on a Macpro with a 2.8Ghz intel Xeon processor and 4 Gb memory with MAC OS X 10.5. The cut-off time was set to 3 hours.

Table 1 and Table 2 contain the experimental results obtained when solving the Min-2-SAT and Min-3-SAT instances with the MaxSAT and MaxClique solvers. The tables show, for each solver, the number of instances solved within 3 hours (between brackets) in a set of 50 instances at each point, and the mean time needed to solve these solved instances. Table 1 shows results just for 90-variable and 100-variable instances when $C/V = 1$ because they can be easily solved and we have no space. The MaxSAT solvers not included in Table 2 were far from being competitive on Min-3-SAT instances. The three encodings are denoted by $E1$ (Encoding 1), $E2$ (Encoding 2), and $E3$ (Encoding 3).

The experimental results show that the direct encoding of MinSAT into Partial MaxSAT (Encoding 1) is better than the MaxClique based encoding (Encoding 2) for all the MaxSAT solvers. However, when the MaxClique-based encoding is improved using a good clique partition of G_I, then the improved MaxClique-based encoding (Encoding 3) is by far the best performing encoding for all the MaxSAT solvers. More importantly, Encoding 3 even makes MaxSatz significantly better for computing a maximum clique in $\overline{G_I}$ than the state-of-the-art specific MaxClique solvers MaxCliqueDyn and Cliquer.

Table 2. Number of solved instances and mean time (seconds) of MaxSatz, MaxCliqueDyn (Dyn in the table) and Cliquer on random Min-3-SAT

instance		*MaxSatz*			*Dyn*	*Cliquer*
#var	C/V	$E1$	$E2$	$E3$		
40	4.25	4.67*(50)*	992.5*(50)*	0.28*(50)*	**0.12*(50)***	15.94*(50)*
50	4.25	75.6*(50)*	- *(0)*	1.57*(50)*	**0.98*(50)***	945.0*(49)*
60	4.25	1153*(50)*	- *(0)*	**8.31*(50)***	9.94*(50)*	5385*(10)*
70	4.25	5989*(5)*	- *(0)*	**42.77*(50)***	106.4*(50)*	- *(0)*
80	4.25	- *(0)*	- *(0)*	**186.3*(50)***	917.4*(50)*	- *(0)*
90	4.25	- *(0)*	- *(0)*	**760.4*(50)***	4453*(41)*	- *(0)*
100	4.25	- *(0)*	- *(0)*	**2819*(26)***	- *(0)*	- *(0)*
40	5.00	10.87*(50)*	5693*(48)*	0.80*(50)*	**0.30*(50)***	63.98*(50)*
50	5.00	226.8*(50)*	- *(0)*	5.24*(50)*	**3.97*(50)***	3035*(35)*
60	5.00	3803*(48)*	- *(0)*	**39.3*(50)***	60.14*(50)*	- *(0)*
70	5.00	- *(0)*	- *(0)*	**243.4*(50)***	735.2*(50)*	- *(0)*
80	5.00	- *(0)*	- *(0)*	**1512*(50)***	6355*(41)*	- *(0)*
90	5.00	- *(0)*	- *(0)*	**5167*(39)***	- *(0)*	- *(0)*
100	5.00	- *(0)*	- *(0)*	- *(0)*	- *(0)*	- *(0)*

References

[ABL09] Ansótegui, C., Bonet, M.L., Levy, J.: Solving (weighted) partial MaxSAT through satisfiability testing. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 427–440. Springer, Heidelberg (2009)

[GKZ05] Goldstein, A., Kolman, P., Zheng, J.: Minimum common string partition problem: Hardness and approximations. Electr. J. Comb. 12 (2005)

[KJ07] Konc, J., Janezic, D.: An improved branch and bound algorithm for the maximum clique problem. Communications in Mathematical and in Computer Chemistry 58, 569–590 (2007)

[LM09] Li, C.M., Manyà, F.: Max-SAT, hard and soft constraints. In: Biere, A., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, pp. 613–631. IOS Press, Amsterdam (2009)

[LMP07] Li, C.M., Manyà, F., Planes, J.: New inference rules for Max-SAT. Journal of Artificial Intelligence Research 30, 321–359 (2007)

[MR96] Marathe, M.V., Ravi, S.S.: On approximation algorithms for the minimum satisfiability problem. Information Processing Letters 58, 23–29 (1996)

[MSP09] Manquinho, V.M., Silva, J.P.M., Planes, J.: Algorithms for weighted boolean optimization. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 495–508. Springer, Heidelberg (2009)

[Ost02] Ostergard, P.R.J.: A fast algorithm for the maximum clique problem. Discrete Applied Mathematics 120, 197–207 (2002)