A Preprocessor for Max-SAT Solvers*

Josep Argelich[1], Chu Min Li[2], and Felip Manyà[1]

[1] Computer Science Department
Universitat de Lleida
Jaume II, 69, E-25001 Lleida, Spain
[2] LaRIA, Université de Picardie Jules Verne
33 Rue Saint Leu, 80039 Amiens Cedex 01, France

Abstract. We describe a preprocessor that incorporates a variable saturation procedure for Max-SAT, and provide empirical evidence that it improves the performance of some of the most successful state-of-the-art solvers on several partial (weighted) Max-SAT instances of the 2007 Max-SAT Evaluation.

1 Introduction

In the last years, there has been an increasing interest in Max-SAT formalisms such as (weighted) Max-SAT and partial (weighted) Max-SAT. Among the most relevant results we highlight the following ones: (i) there exist solvers like ChaffBS [6], Clone [14], Lazy [1], MaxSatz [12], MiniMaxSat [8], ms4 [13], PMS [3], Sat4Jmaxsat, SR(w) [15] and Toolbar [7] which solve many instances that are beyond the reach of the solvers existing just five years ago; (ii) resolution refinements, which preserve the number of unsatisfied clauses, have been incorporated into Max-SAT solvers [9,7,12], as well as good quality underestimations of the lower bound [10,11,14,15], (iii) a resolution-style calculus for Max-SAT has been proven to be complete [4,5], (iv) formalisms like Partial Max-SAT have been investigated for solving problems with soft constraints [2,6,8,3], and (v) two evaluations of Max-SAT solvers have been performed for the first time.

In this paper we present a preprocessor that can be applied to solvers for Max-SAT formalisms, including Max-SAT, weighted Max-SAT, partial Max-SAT and partial weighted Max-SAT solvers. Our preprocessor implements a variable saturation procedure defined in [4,5]. Moreover, we provide empirical evidence that it improves the performance of MiniMaxSat, SR(w) and PMS on several partial (weighted) Max-SAT instances of the 2007 Max-SAT Evaluation. The preprocessor applies the variable saturation procedure defined in [4,5] to a limited number of variables which are selected heuristically, and transforms the input instance into an equivalent instance which does not contain the saturated variables. To the best of our knowledge, this is the first paper that investigates the practical usefulness of the notion of variable saturation in the Max-SAT context.

* This research was funded by the MEC research projects TIN2006-15662-C02-02 and TIN2007-68005-C04-04, and Acción Integrada HP2005-0147.

H. Kleine Büning and X. Zhao (Eds.): SAT 2008, LNCS 4996, pp. 15–20, 2008.

The structure of the paper is as follows. Section 2 contains preliminary definitions about Max-SAT. Section 3 introduces the Max-SAT resolution rule and the notion of variable saturation. Section 4 reports and analyses the experiments.

2 Preliminaries

In propositional logic a variable x_i may take values 0 (false) or 1 (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 is a multiset of clauses. A weighted clause is a pair (C_i, w_i), where C_i is a disjunction of literals and w_i, its weight, is a positive number, and a weighted CNF formula is a multiset of weighted clauses. An assignment of truth values to the propositional variables satisfies a literal x_i ($\bar{x}_i$) if it takes the value 1 (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.

The Max-SAT problem for a CNF formula ϕ is the problem of finding an assignment that maximizes (minimizes) the number of satisfied (unsatisfied) clauses. The weighted Max-SAT problem for a weighted CNF formula ϕ is the problem of finding an assignment that minimizes the sum of weights of unsatisfied clauses. A Partial Max-SAT instance is a CNF formula in which some clauses are *relaxable* or *soft* and the rest are *non-relaxable* or *hard*. Solving a Partial Max-SAT instance amounts to find an assignment that satisfies all the hard clauses and the maximum number of soft clauses. The *weighted Partial Max-SAT* problem is the combination of weighted Max-SAT and Partial Max-SAT.

3 Resolution in Max-SAT

The *Max-SAT resolution* rule is defined as follows:

$$\begin{array}{l} x \vee a_1 \vee \cdots \vee a_s \\ \overline{x} \vee b_1 \vee \cdots \vee b_t \\ \hline\hline a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_t \\ x \vee a_1 \vee \cdots \vee a_s \vee \overline{b_1} \\ x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \overline{b_2} \\ \cdots \\ x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_{t-1} \vee \overline{b_t} \\ \overline{x} \vee b_1 \vee \cdots \vee b_t \vee \overline{a_1} \\ \overline{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \overline{a_2} \\ \cdots \\ \overline{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \cdots \vee a_{s-1} \vee \overline{a_s} \end{array}$$

This inference rule is applied to multisets of clauses, and replaces the premises of the rule by its conclusions. We say that the rule *cuts* the variable x, and the tautologies concluded by the rule are removed from the resulting multiset. In partial Max-SAT, the hard clauses remain and the clauses subsumed by the hard clause are removed (see [3] for details). For the sake of clarity, we did not define the weighted version of the rule (see [5] for details).

Definition 1. *A multiset of clauses $\mathcal{C}$ is said to be* saturated w.r.t. *x if for every pair of clauses $C_1 = x \vee A$ and $C_2 = \overline{x} \vee B$ of $\mathcal{C}$, there is a literal l such that l is in A and $\overline{l}$ is in B. A multiset of clauses $\mathcal{C}'$ is a* saturation of *$\mathcal{C}$ w.r.t. x if $\mathcal{C}'$ is saturated w.r.t. x and $\mathcal{C} \vdash_x \mathcal{C}'$; i.e., $\mathcal{C}'$ can be obtained from $\mathcal{C}$ applying Max-SAT resolution cutting x finitely many times.*

Lemma 1. *[5] For every multiset of clauses $\mathcal{C}$ and variable x, there exists a multiset $\mathcal{C}'$ such that $\mathcal{C}'$ is a saturation of $\mathcal{C}$ w.r.t. x. Moreover, this multiset $\mathcal{C}'$ can be computed by applying Max-SAT resolution to any pair of clauses $x \vee A$ and $\overline{x} \vee B$ with the restriction that $A \vee B$ is not a tautology, using any ordering of the literals, until we can not apply Max-SAT resolution any longer.*

The completeness proof of Max-SAT resolution [4,5] states that we can get a complete algorithm by successively saturating w.r.t. all the variables as follows: we saturate w.r.t x_1 and then remove all the clauses containing x_1, saturate w.r.t x_2 and then remove all the clauses containing x_2, etc. After saturating this way w.r.t. all the variables we get as many empty clauses as the minimum number of unsatisfied clauses in the original formula.

Solving a Max-SAT instance by successively saturating w.r.t. all the variables is clearly not competitive with solving it with a modern branch and bound solver. Nevertheless, we thought that it would make sense to saturate w.r.t. a limited number of variables as a preprocessing in order to simplify the formula. We select the variables to be saturated, depending on a parameter k, iteratively as follows: We build a graph whose nodes are the Boolean variables occurring in the instance, and add an edge between two vertices if the variables of the vertices occur in the same clause. We select a variable whose vertex has minimal degree if its degree is smaller than k. This process is repeated until no more variables can be selected. The idea is to saturate variables in which the application of variable saturation is not very costly in terms of time and space. We also tried to saturate the variables with a low number of occurrences, but the results were not so good.

4 Experimental Results

To assess the impact of the preprocessor on the performance of branch and bound Max-SAT solvers, we solved partial Max-SAT instances[1] of the 2007 Max-SAT Evaluation (with a timeout of 30 minutes as in the evaluation) on three of the most successful and representative state-of-the-art solvers: MiniMaxSat [8], PMS [3], and SR(w) [15]. For MiniMaxSat and SR(w) we used the same versions as in the evaluation. For PMS we used an improved version (PMS v1.3). We executed the preprocessor with $k = 6, 10, 14$ (remind that k is the parameter for selecting variables). All the experiments were performed on a Linux Cluster where the nodes have a 2GHz AMD Opteron processor with 1Gb of RAM.

[1] We solved only the instances in which the preprocessor detected variables that could be saturated. We also solved (weighted) Max-SAT instances, but the speed-ups were not so good as in (weighted) Partial Max-SAT.

Tables 1 and 2 show the experimental results for PMS. The instances are divided into sets. The first column is the name of the set, the second column shows the number of instances in each set, the third column shows the results for the solver without preprocessing, and the rest of columns show the results with preprocessing for $k = 6, 10, 14$. We display the mean time (in seconds) of the solved instances, as well as the number of solved instances (in brackets). We observe that PMS with preprocessing solves more instances in 5 sets, and reduces considerably the CPU time in most of the other sets. The best improvements are achieved for MaxClique (random), where the preprocessing allows to solve 8 additional instances, and for Auctions (paths), where the preprocessing allows to solve 9 additional instances.

Tables 3 and 4 show the results for MiniMaxSat. In this case, the gains are not so significative as for PMS, although the preprocessing allows to solve 1 additional instance for MaxClique (structured) and for WCSP (spot5 dir).

Tables 5 and 6 show the experimental results for SR(w). In this case, we solve an additional instance for 3 sets (MaxCSP (dense loose), MaxCSP (w-queens) and Auctions (scheduling)), and 185 additional instances for Pseudo (factor). The latter is the best improvement achieved with our preprocessor.

Table 1. Partial Max-SAT benchmarks with PMS

Instance set	#	PMS	PMS(6)	PMS(10)	PMS(14)
MaxClique (random)	96	43.69(80)	69.30(83)	61.04(85)	**53.85(88)**
MaxClique (structured)	62	175.27(23)	183.30(24)	178.03(24)	**171.13(25)**
MaxOne (3-SAT)	50	261.95(50)	122.08(50)	**62.06(50)**	328.07(48)
MaxOne (structured)	60	**177.84(58)**	234.56(56)	223.76(42)	6.57(1)
MaxCSP (dense loose)	20	5.50(20)	5.26(20)	**3.39(20)**	8.31(20)
MaxCSP (dense tight)	20	9.76(20)	9.76(20)	**7.83(20)**	12.95(20)
MaxCSP (sparse loose)	20	16.39(20)	9.18(20)	**4.77(20)**	36.51(19)
MaxCSP (sparse tight)	20	24.02(20)	21.70(20)	**18.07(20)**	84.81(20)
WCSP (w-queens)	7	72.22(6)	72.19(6)	**72.17(6)**	72.18(6)

Table 2. Weighted Partial Max-SAT benchmarks with PMS

Instance set	#	PMS	PMS(6)	PMS(10)	PMS(14)
Auctions (paths)	88	233.56(71)	**178.50(80)**	127.72(77)	266.47(63)
Auctions (regions)	84	**5.24(84)**	5.30(84)	5.52(84)	5.62(84)
Auctions (scheduling)	84	89.70(84)	89.62(84)	89.66(84)	**89.61(84)**
Pseudo (factor)	186	**11.00(186)**	11.64(186)	226.88(186)	924.37(2)
Pseudo (miplib)	16	1.94(4)	**0.96(4)**	190.93(4)	2.34(1)
QCP	25	**199.31(15)**	199.36(15)	199.46(15)	199.52(15)
WCSP (planning)	71	**13.96(71)**	21.97(71)	63.65(70)	233.29(42)
WCSP (spot5 dir)	21	14.86(2)	6.59(5)	**57.96(6)**	13.27(5)
WCSP (spot5 log)	21	18.95(2)	91.03(3)	2.55(4)	**1.46(4)**

Table 3. Partial Max-SAT benchmarks with MiniMaxSat

Instance set	#	MiniMS	MiniMS(6)	MiniMS(10)	MiniMS(14)
MaxClique (random)	96	**2.41(96)**	2.44(96)	2.67(96)	4.38(96)
MaxClique (structured)	62	85.22(36)	82.15(37)	**67.94(37)**	66.43(36)
MaxOne (3-SAT)	50	**0.37(50)**	0.40(50)	0.43(50)	8.87(50)
MaxOne (structured)	60	**31.35(60)**	20.57(54)	65.88(42)	0.78(1)
MaxCSP (dense loose)	20	**0.65(20)**	0.71(20)	0.87(20)	5.11(20)
MaxCSP (dense tight)	20	**0.69(20)**	0.70(20)	0.70(20)	2.87(20)
MaxCSP (sparse loose)	20	**0.35(20)**	0.36(20)	0.57(20)	21.20(20)
MaxCSP (sparse tight)	20	**0.85(20)**	0.87(20)	0.94(20)	27.05(20)
WCSP (w-queens)	7	55.47(7)	55.28(7)	**54.56(7)**	179.13(7)

Table 4. Weighted Partial Max-SAT benchmarks with MiniMaxSat

Instance set	#	MiniMS	MiniMS(6)	MiniMS(10)	MiniMS(14)
Auctions (paths)	88	29.82(88)	**19.44(88)**	13.52(84)	78.21(75)
Auctions (regions)	84	1.63(84)	**1.55(84)**	**1.55(84)**	1.56(84)
Auctions (scheduling)	84	**46.14(84)**	46.24(84)	46.28(84)	46.16(84)
Pseudo (factor)	186	**1.16(186)**	1.79(186)	5.53(186)	905.51(183)
Pseudo (miplib)	16	**41.35(5)**	84.90(5)	398.55(5)	1.43(1)
QCP	25	25.00(20)	26.71(20)	25.28(20)	**24.65(20)**
WCSP (planning)	71	**9.97(71)**	10.11(71)	22.12(71)	235.45(47)
WCSP (spot5 dir)	21	2.63(3)	11.82(3)	8.18(4)	**6.99(4)**
WCSP (spot5 log)	21	**9.07(4)**	5.69(2)	152.16(3)	323.82(4)

Table 5. Partial Max-SAT benchmarks with SR(w)

Instance set	#	SR-W	SR-W(6)	SR-W(10)	SR-W(14)
MaxClique (random)	96	244.85(55)	219.40(55)	224.65(55)	**218.38(55)**
MaxClique (structured)	62	21.18(9)	**17.56(9)**	22.67(8)	20.17(8)
MaxOne (3-SAT)	50	386.23(41)	**338.69(41)**	718.76(22)	758.61(1)
MaxOne (structured)	60	**471.72(22)**	449.33(19)	618.92(18)	1078.54(1)
MaxCSP (dense loose)	20	697.74(1)	633.31(1)	**1162.49(2)**	0.00(0)
MaxCSP (dense tight)	20	209.22(18)	**199.18(18)**	202.71(18)	350.83(15)
MaxCSP (sparse loose)	20	296.48(16)	**272.89(16)**	408.06(15)	853.86(7)
MaxCSP (sparse tight)	20	235.98(19)	**216.19(19)**	230.31(19)	563.63(12)
WCSP (w-queens)	7	54.00(6)	230.25(7)	**228.06(7)**	258.10(7)

As a conclusion, we could say that variable saturation is an effective preprocessing technique that may produce substantial speed-ups, as well as increase the number of solved instances. As future work we plan to incorporate additional simplification techniques into our preprocessor, and explore the application of variable saturation to a limited number of nodes of the search space because its application at each node is too costly.

Table 6. Weighted Partial Max-SAT benchmarks with SR(w)

Instance set	#	SR-W	SR-W(6)	SR-W(10)	SR-W(14)
Auctions (paths)	88	**173.42(77)**	161.15(76)	169.32(72)	353.90(66)
Auctions (regions)	84	146.54(82)	136.45(82)	126.93(82)	**119.52(82)**
Auctions (scheduling)	84	276.91(56)	240.61(56)	**270.71(57)**	239.26(56)
Pseudo (factor)	186	0.00(0)	2.86(37)	**520.50(185)**	1091.88(1)
Pseudo (miplib)	16	**2.62(5)**	3.04(4)	216.89(4)	4.12(1)
QCP	25	715.58(5)	**572.40(5)**	675.34(5)	674.26(5)
WCSP (planning)	71	**379.57(57)**	371.42(53)	286.88(46)	285.58(25)
WCSP (spot5 dir)	21	2.95(6)	**1.90(6)**	9.27(4)	61.92(3)
WCSP (spot5 log)	21	14.56(6)	**11.53(6)**	25.30(5)	10.83(4)

References

1. Alsinet, T., Manyà, F., Planes, J.: Improved exact solver for weighted Max-SAT. In: SAT-2005, pp. 371–377 (2005)
2. Argelich, J., Manyà, F.: Exact Max-SAT solvers for over-constrained problems. Journal of Heuristics 12(4–5), 375–392 (2006)
3. Argelich, J., Manyà, F.: Partial Max-SAT solvers with clause learning. In: SAT-2007, pp. 28–40 (2007)
4. Bonet, M.L., Levy, J., Manyà, F.: A complete calculus for Max-SAT. In: SAT-2006, pp. 240–251 (2006)
5. Bonet, M.L., Levy, J., Manyà, F.: Resolution for Max-SAT. Artificial Intelligence 171(8–9), 240–251 (2007)
6. Fu, Z., Malik, S.: On solving the partial MAX-SAT problem. In: SAT-2006, pp. 252–265 (2006)
7. Heras, F., Larrosa, J.: New inference rules for efficient Max-SAT solving. In: AAAI-2006, pp. 68–73 (2006)
8. Heras, F., Larrosa, J., Oliveras, A.: MiniMaxSat: A new weighted Max-SAT solver. In: SAT-2007 (2007)
9. Larrosa, J., Heras, F.: Resolution in Max-SAT and its relation to local consistency in weighted CSPs. In: IJCAI-2005, pp. 193–198 (2005)
10. Li, C.M., Manyà, F., Planes, J.: Exploiting unit propagation to compute lower bounds in branch and bound Max-SAT solvers. In: CP-2005, pp. 403–414 (2005)
11. Li, C.M., Manyà, F., Planes, J.: Detecting disjoint inconsistent subformulas for computing lower bounds for Max-SAT. In: AAAI-2006, pp. 86–91 (2006)
12. Li, C.M., Manyà, F., Planes, J.: New inference rules for Max-SAT. Journal of Artificial Intelligence Research 30, 321–359 (2007)
13. Marques-Silva, J., Planes, J.: Algorithms for Maximum Satisfiability using Unsatisfiable Cores. In: DATE-2008 (2008)
14. Pipatsrisawat, K., Darwiche, A.: Clone: Solving weighted max-sat in a reduced search space. In: 20th Australian Joint Conf. on AI, AI-2007, pp. 223–233 (2007)
15. Ramírez, M., Geffner, H.: Structural relaxations by variable renaming and their compilation for solving MinCostSAT. In: CP-2007, pp. 605–619 (2007)