Natural Max-SAT Encoding of Min-SAT

Adrian Kügel

Faculty of Engineering and Computer Sciences
Ulm University, Ulm, Germany
Adrian.Kuegel@uni-ulm.de

Abstract. We show that there exists a natural encoding which transforms Min-SAT instances into Max-SAT instances. Unlike previous encodings, this natural encoding keeps the same variables, and the optimal assignment for the Min-SAT instance is identical to the optimal assignment of the corresponding Max-SAT instance. In addition to that the encoding can be generalized to the Min-SAT variants with clause weights and hard clauses. We conducted experiments which give evidence that our encoding is practically relevant, as Min-2-SAT instances can be solved much faster by transforming them to Max-SAT and using a Max-SAT solver than by using the best Min-SAT solver directly.

Keywords: Min-SAT, Max-SAT

1 Introduction

The Minimum Satisfiability Problem (Min-SAT) asks for an assignment of Boolean variables which satisfies the minimum number of clauses of a given formula, whereas the Maximum Satisfiability Problem (Max-SAT) seeks to maximize the number of satisfied clauses. Both problems can be seen as a generalization of the Satisfiability Problem (SAT).

Recently, Li et al. [8] presented a Min-SAT solver called MinSatz and showed that a Min-SAT encoding of the MaxClique problem (and related problems) makes MinSatz competitive with the best MaxClique solvers. MinSatz was also tested against Max-SAT solvers using the three different encodings presented in [7]. In these tests, MinSatz was faster than the Max-SAT solvers on the encodings. These experiments indicated that solving a Min-SAT instance can be done faster using MinSatz than encoding it to a Max-SAT instance and using one of the available Max-SAT solvers. We show that when using our *better* Max-SAT encoding of Min-SAT instances it is still possible to outperform MinSatz on several kind of Min-SAT instances by encoding them to Max-SAT and using the Max-SAT solver akmaxsat.

Our paper is structured as follows: in Section 2 we provide basic definitions, then in Section 3 we describe our encoding. In Section 4 we present experimental data and finally in Section 5 we draw our conclusions.

2 Definitions

A *CNF formula* $\mathcal{F}$ is a conjunction of clauses consisting of Boolean variables. A *clause* C is a disjunction of literals and is written as $(l_1 \vee l_2 \vee \cdots \vee l_k)$, where $l_1, \ldots, l_k$ are from the set of variables and their negations. A literal $\overline{x}_i$ is true if the variable x_i is false, and it is false otherwise. We call a clause *satisfied* if at least one of its literals is true, and we call it *unsatisfied* if all its literals are false.

A *hard clause* is a clause which needs to be satisfied, whereas a *soft clause* specifies a clause which may be unsatisfied by the optimal assignment. The *partial Min-SAT problem* and the *partial Max-SAT problem* deal with both soft and hard clauses. Another variant of the Min-SAT problem is the *weighted Min-SAT problem*; in this variant, each clause has a positive weight which indicates the relative importance of the clause, and the sum of the weights of satisfied clauses has to be minimized. Likewise, in the *weighted Max-SAT problem* the sum of the weights of satisfied clauses has to be maximized.

We define the size of a clause to be the number of literals it consists of. A CNF formula which consists only of clauses of size k is also called a k-SAT formula, the corresponding Max-SAT instance Max-k-SAT, and the corresponding Min-SAT instance Min-k-SAT.

3 A Natural Max-SAT Encoding of Min-SAT Instances

The idea of the natural Max-SAT encoding of a Min-SAT instance is quite simple: we replace each original clause C by its negation $\overline{C}$. However we need to transform $\overline{C}$ into conjunctive normal form in order to get a Max-SAT instance. We apply an idea based on Max-SAT resolution rules. Max-SAT resolution rules were developed by Bonet et al. ([2], [3]) and Larrosa et al. ([5]). It is shown that $\overline{C} = \overline{(l_1 \vee l_2 \vee \ldots \vee l_k)}$ (where $l_1, \ldots, l_k$ are literals) can be transformed into a set of k clauses. We have adjusted the more general recursive resolution rule in [5] to our special case:

$$\mathrm{CNF}_{\mathrm{linear}}(\overline{l_1 \vee \ldots \vee l_k}) = \overline{l_1} \wedge (l_1 \vee \overline{l_2}) \wedge (l_1 \vee l_2 \vee \overline{l_3}) \wedge \ldots \wedge (l_1 \vee l_2 \vee \ldots \vee \overline{l_k}) \quad (1)$$

We will illustrate the transformation rule with a small example. Let $C = (x_1 \vee \overline{x_2} \vee x_3)$. $\mathrm{CNF}_{\mathrm{linear}}(\overline{x_1 \vee \overline{x_2} \vee x_3}) = \overline{x_1} \wedge (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3})$. It can be easily verified that only for the assignment for which C is unsatisfied, all new clauses are satisfied, and at most one of the new clauses will be unsatisfied by any assignment.

Using this rule, each clause C of the original Min-SAT instance is replaced by $\mathrm{CNF}_{\mathrm{linear}}(\overline{C})$. The set of new clauses has the property, that any assignment that does not satisfy the original clause C satisfies all clauses of the new clause set, and any assignment that satisfies the original clause satisfies all but one of the new clauses. In this case, trying to maximize the number of satisfied clauses in the encoded instance is equivalent to trying to minimize the number of satisfied clauses in the original Min-SAT instance. Also, for any assignment,

the number of unsatisfied clauses in the encoded instance corresponds to the number of satisfied clauses in the original Min-SAT instance.

As a clause consisting of k literals is replaced by k clauses, for a Min-k-SAT instance with m clauses we get a Max-SAT instance with the same number of variables and with $k \cdot m$ soft clauses. Previous encodings presented in [7] used m variables and up to $\Theta(m^2)$ clauses (including some hard clauses).

Our encoding can be used in the reverse direction too, transforming a Max-SAT instance into a Min-SAT instance. In this case, the number of unsatisfied clauses in the Min-SAT instance corresponds to the number of satisfied clauses of the Max-SAT instance. The encoding can also be used for weighted Min-SAT instances; in that case, each replacement clause gets the weight of the original clause. If a Min-SAT instance contains hard clauses, the hard clauses are not replaced, but kept as they are. It is interesting to note that the transformation of a partial Min-SAT encoded Maximum Clique instance yields the commonly used partial Max-SAT encoding of the Maximum Clique instance (and vice versa). In the next section we will present experimental evidence that our encoding is especially useful for Min-2-SAT instances.

4 Experimental Results

In order to evaluate our natural Max-SAT encoding of Min-SAT, we selected our Max-SAT solver akmaxsat [4] which performed best in several categories of random and crafted Max-SAT instances in the Max-SAT evaluation 2011. Our solver akmaxsat can be found at www.uni-ulm.de/in/theo/m/kuegel. For comparison reasons we also used the Max-SAT solver MaxSatz in its publicly available version from 2009 ([6]). Also, we obtained the solver MinSatz from the authors of [8] (the same version that was used in their tests).

As benchmark instances we generated randomly unweighted Min-2-SAT and Min-3-SAT instances. For each selection of number of variables and clauses-to-variables ratio we generated 30 instances. Each Min-SAT instance was also encoded to the corresponding Max-SAT instance using our natural Max-SAT encoding. Also, we generated the corresponding partial Max-SAT instance using the best encoding E3 of [7]. We ran akmaxsat and MaxSatz on both Max-SAT encodings of each Min-SAT instance and compared its performance with the performance of MinSatz on the corresponding Min-SAT instance.

We ran the experiments on a node of the bwGRiD [1] which provides two Intel Harpertown quad-core CPUs with 2.83 Ghz and 8GB RAM each. The installed operating system was Scientific Linux. We used a timeout of 1 hour for each instance. Instances which were not solved within 1 hour are regarded as unsolved.

Table 1 shows for all three solvers the average runtime in seconds on the solved Min-2-SAT instances of each kind (showing in parentheses the number of instances solved). Our natural Max-SAT encoding is labeled with *NE*. The test results show that akmaxsat (using our encoding) clearly outperforms MinSatz on random Min-2-SAT instances. For clauses-to-variables ratios of at most 3, the

Table 1. Average runtime in seconds on Min-2-SAT instances

		MinSatz	akmaxsat		maxsatz	
C/V	#var		NE	E3	NE	E3
2	160	0.05 (30)	**0.01 (30)**	0.04 (30)	**0.01 (30)**	0.02 (30)
2	180	0.08 (30)	**0.01 (30)**	0.04 (30)	**0.01 (30)**	0.03 (30)
2	200	0.13 (30)	**0.01 (30)**	0.05 (30)	**0.01 (30)**	0.04 (30)
3	160	0.19 (30)	**0.02 (30)**	0.21 (30)	0.07 (30)	0.14 (30)
3	180	0.31 (30)	**0.02 (30)**	0.26 (30)	0.08 (30)	0.17 (30)
3	200	0.52 (30)	**0.04 (30)**	0.48 (30)	0.18 (30)	0.34 (30)
4	160	0.96 (30)	**0.11 (30)**	40.06 (30)	2.59 (30)	52.58 (30)
4	180	1.47 (30)	**0.15 (30)**	44.80 (30)	5.18 (30)	44.94 (30)
4	200	4.09 (30)	**0.28 (30)**	87.73 (30)	25.80 (30)	142.62 (30)
5	160	16.85 (30)	**0.78 (30)**	605.81 (25)	41.41 (30)	1027.26 (21)
5	180	51.80 (30)	**1.29 (30)**	940.13 (19)	175.03 (30)	930.35 (13)
5	200	117.4 (30)	**2.45 (30)**	1474.66 (12)	722.35 (30)	1987.71 (4)
6	160	349.9 (30)	**3.67 (30)**	1533.23 (6)	321.30 (30)	2088.89 (2)
6	180	861.6 (26)	**11.26 (30)**	1617.81 (1)	1136.41 (26)	- (0)
6	200	1330 (16)	**31.96 (30)**	2485.65 (1)	1540.48 (8)	- (0)

solver MaxSatz is faster than MinSatz, too. When comparing the two encodings, the new encoding always leads to a faster runtime for both Max-SAT solvers.

Table 2 shows the average runtime on the Min-3-SAT instances in the same format as in Table 1. On Min-3-SAT instances, the new encoding seems to work better than the encoding E3 for clauses-to-variables ratios above 3. A clauses-to-variables ratio of 3 leads to different results for the two Max-SAT solvers: akmaxsat can handle the new encoding better, whereas MaxSatz is faster on encoding E3. Note that in this case, akmaxsat on the new encoding is faster than MaxSatz on the encoding E3. For small clauses-to-variables ratios of less than 3, the encoding E3 seems to be always superior. Comparing the results of akmaxsat on the new encoding to the results of MinSatz, we can see that in most cases, MinSatz is still faster, but for some instances with a clauses-to-variables ratio of 6, akmaxsat outperforms Minsatz.

5 Conclusions

We have presented a natural Max-SAT encoding of Min-SAT instances that has the following advantages:

1. The encoding keeps the same variables and just increases the number of clauses by a factor of number of variables per clause.
2. The optimal assignment of the encoded instance is identical to the optimal assignment of the Min-SAT instance.
3. The encoding works notably well on Min-2-SAT instances, and the Max-SAT solver akmaxsat on the encoded instance is much faster than the Min-SAT solver MinSatz on the original instance.

Table 2. Average runtime in seconds on Min-3-SAT instances

		MinSatz	akmaxsat		maxsatz	
C/V	#var		NE	E3	NE	E3
2	80	**0.01 (30)**	0.09 (30)	0.04 (30)	1.01 (30)	0.05 (30)
2	90	**0.02 (30)**	0.22 (30)	0.10 (30)	2.77 (30)	0.10 (30)
2	100	**0.05 (30)**	0.68 (30)	0.31 (30)	16.65 (30)	0.28 (30)
3	80	**0.15 (30)**	0.71 (30)	1.55 (30)	3.93 (30)	1.93 (30)
3	90	**0.45 (30)**	2.89 (30)	8.87 (30)	20.43 (30)	10.70 (30)
3	100	**1.41 (30)**	9.34 (30)	22.85 (30)	77.29 (30)	29.16 (30)
4	80	**1.71 (30)**	4.67 (30)	5.51 (30)	18.07 (30)	84.11 (30)
4	90	**8.54 (30)**	24.17 (30)	395.7 (30)	115.44 (30)	582.39 (30)
4	100	**37.06 (30)**	80.61 (30)	961.9 (27)	460.72 (30)	1372.18 (25)
5	80	**14.43 (30)**	21.20 (30)	952.7 (29)	76.97 (30)	1226.06 (26)
5	90	**112.2 (30)**	134.2 (30)	1762 (13)	412.77 (29)	1684.11 (6)
5	100	**439.0 (30)**	587.4 (30)	1554 (1)	1824.86 (22)	3460.76 (1)
6	80	**68.77 (30)**	73.06 (30)	2139 (8)	264.24 (30)	3549.32 (1)
6	90	552.9 (30)	**537.0 (30)**	- (0)	1519.72 (27)	- (0)
6	100	1551 (24)	**1514 (25)**	- (0)	2983.00 (2)	- (0)

4. Our encoding can also be used to transform Max-SAT instances into Min-SAT instances.

As the tests in [8] have shown, the MinSatz solver performs much better than Max-SAT solvers on encoded Maximum Clique instances. There might be other optimization problems where this could be true, and our encoding could be used to automatically transform Max-SAT encoded optimization problems into Min-SAT encoded optimization problems.

Acknowledgments. We gratefully thank the bwGRiD project [1] for the computational resources. Also we thank Zhu Zhu for sending us an executable of the solver MinSatz, and we thank the reviewers for helpful comments.

References

1. bwGRiD, member of the German D-Grid initiative, funded by the Ministry for Education and Research and the Ministry for Science, Research and Arts Baden-Wuerttemberg, `http://www.bw-grid.de`
2. Bonet, M.L., Levy, J., Manyà, F.: A Complete Calculus for Max-SAT. In: Biere, A., Gomes, C.P. (eds.) SAT. Lecture Notes in Computer Science, vol. 4121, pp. 240–251. Springer (2006)
3. Bonet, M.L., Levy, J., Manyà, F.: Resolution for Max-SAT. Artif. Intell. 171, 606–618 (2007)
4. Kuegel, A.: Improved exact solver for the weighted max-sat problem. Workshop Pragmatics of SAT (2010)
5. Larrosa, J., Heras, F., de Givry, S.: A logical approach to efficient Max-SAT solving. Artif. Intell. 172(2-3), 204–233 (2008)

6. Li, C.M., Manyà, F., Mohamedou, N.O., Planes, J.: Exploiting Cycle Structures in Max-SAT. In: Kullmann, O. (ed.) SAT. Lecture Notes in Computer Science, vol. 5584, pp. 467–480. Springer (2009)
7. Li, C.M., Manyà, F., Quan, Z., Zhu, Z.: Exact MinSAT solving. In: Strichman, O., Szeider, S. (eds.) SAT. Lecture Notes in Computer Science, vol. 6175, pp. 363–368. Springer (2010)
8. Li, C.M., Zhu, Z., Manyà, F., Simon, L.: Minimum Satisfiability and Its Applications. In: Walsh, T. (ed.) IJCAI. pp. 605–610. IJCAI/AAAI (2011)