

# Solving (Weighted) Partial MaxSAT with ILP \*

Carlos Ansótegui<sup>1</sup> and Joel Gabàs<sup>1</sup>

DIEI, Univ. de Lleida  
carlos@diei.udl.cat  
joel.gabas@diei.udl.cat

**Abstract.** Several combinatorial optimization problems can be translated into the Weighted Partial Maximum Satisfiability (WPMS) problem. This is an optimization variant of the Satisfiability (SAT) problem. There are two main families of WPMS solvers based on SAT technology: *branch and bound* and *SAT-based*. From the MaxSAT evaluations, we have learned that SAT-based solvers dominate on industrial instances while branch and bound dominate on random. For crafted instances it depends on the category.

In this work, we study the performance of an Integer Linear Programming approach. In particular, we translate the WPMS problem into ILP and apply the Mixed Integer Programming (MIP) solver, IBM-CPLEX. We present an extensive experimental evaluation showing that this approach clearly dominates on crafted instances.

## 1 Introduction

The Maximum Satisfiability (MaxSAT) problem is the optimization version of SAT and has several application domains [2, 4, 20–24]. The idea behind this formalism is that sometimes not all constraints of a problem can be satisfied, and we try to satisfy the maximum number of them. The MaxSAT problem can be further generalized to the Weighted Partial MaxSAT (WPMS) problem. In this case, we can divide the constraints in two groups: the constraints that must be satisfied (hard), and the ones that may or may not be satisfied (soft). In the last group, we may put different weights to the constraints, where the weight is the penalty to falsify the constraint.

There are two main classes of WPMS algorithms: branch and bound [7, 11, 13–15] and SAT-based [1, 5, 8–10, 17–19]. SAT-based MaxSAT algorithms consist in the reformulation of the problem as a sequence of SAT instances.

From the last international MaxSAT evaluation 2012 [3], we can conclude that SAT-based solvers dominate on industrial instances while branch and bound dominate on random instances. For crafted instances, it is not so clear. Branch and bound solvers have dominated since 2006, but at the last 2012 evaluation, two SAT-based solvers were reported to be the best for crafted instances at the Weighted Partial MaxSAT category.

---

\* This research has been partially founded by the CICYT research projects TASSAT (TIN2010-20967-C04-01/03/04) and ARINF (TIN2009-14704-C03-01).

In this paper, we focus our attention on Mixed Integer Programming (MIP) techniques from Operation Research (OR). They have been very successful on solving optimization problems and it makes sense to study the performance of these techniques on WPMS instances. In particular, we explore the approach which consists in translating the WPMS instances into ILP and then apply a MIP solver. It is easy to see that the *soft* clauses of a WPMS instance represent the objective function of the corresponding ILP instance, while the *hard* clauses represent the region of feasible solutions. Section 3 provides a detailed description.

Our experimental evaluation shows that this approach clearly dominates on (Weighted) Partial crafted instances. This is a surprising result not reported before and worth to be known in the community. On the other hand, we also show that this approach is not competitive on the industrial instances. Therefore, further work is required to identify the strength of ILP for crafted instances and how to take advantage of it.

To our best knowledge, the first work using MIP technology for MaxSAT solving can be found in [6]. However, the experimental results did not show such a good performance on crafted instances. A more recent work, using MIP technology can be found in [5]. The authors present a hybrid approach where a SAT solver and a MIP solver interact. This approach also solves a sequence of SAT instances as SAT-based solvers. These instances are simpler than the ones generated by SAT-based solvers since all the arithmetic constraints are extracted and managed by the MIP solver. This is an interesting approach, but from the experimental evaluation we can see it is not yet competitive enough.

This paper proceeds as follows. Section 2 presents some preliminary concepts. Section 3 presents the translation from WPMS into ILP. Section 4 presents the experimental evaluation. Finally, Section 5 shows the conclusions and the future work.

## 2 Preliminaries

A *literal* is either a Boolean variable  $x$  or its negation  $\bar{x}$ . A *clause*  $C$  is a disjunction of literals. A *weighted clause* is a pair  $(C, w)$ , where  $C$  is a clause and  $w$  is a natural number or infinity, indicating the penalty for falsifying the clause  $C$ . A *Weighted Partial MaxSAT formula* is a multiset of weighted clauses

$$\varphi = \{(C_1, w_1), \dots, (C_m, w_m), (C_{m+1}, \infty), \dots, (C_{m+m'}, \infty)\}$$

where the first  $m$  clauses are soft and the last  $m'$  clauses are hard. The set of variables occurring in a formula  $\varphi$  is noted as  $\text{var}(\varphi)$ .

A (*total*) *truth assignment* for a formula  $\varphi$  is a function  $I : \text{var}(\varphi) \rightarrow \{0, 1\}$ , that can be extended to literals, clauses, SAT formulas. For MaxSAT formulas is defined as  $I(\{(C_1, w_1), \dots, (C_m, w_m)\}) = \sum_{i=1}^m w_i (1 - I(C_i))$ . The *optimal cost* of a formula is  $\text{cost}(\varphi) = \min\{I(\varphi) \mid I : \text{var}(\varphi) \rightarrow \{0, 1\}\}$  and an *optimal assignment* is an assignment  $I$  such that  $I(\varphi) = \text{cost}(\varphi)$ .

The *Weighted Partial MaxSAT problem* for a Weighted Partial MaxSAT formula  $\varphi$  is the problem of finding an *optimal assignment*.

### 3 Translation of Weighted Partial MaxSAT into ILP

Encodings translating WPMS into ILP can be found in the literature [12, 16]. Here, we describe the precise encoding we used in our evaluation. Given a WPMS formula,  $\{(C_1, w_1), \dots, (C_m, w_m), (C_{m+1}, \infty), \dots, (C_{m+m'}, \infty)\}$ , we can translate it into a ILP instance, as follows:

Let  $s = (\cup_{i=1}^m CNF(\bar{b}_i \leftrightarrow C_i))$  and  $h = (\cup_{j=m+1}^{m+m'} C_j)$ , where  $CNF(\varphi)$  transforms  $\varphi$  into Conjunctive Normal Form and the  $b_i$ 's are new fresh Boolean variables. The  $i$ th element of  $s$  ensures that  $b_i$  is true iff the soft clause  $C_i$  is falsified and  $h$  is the set of hard clauses of the WPMS problem. The soft clauses of a WPMS instance represent the *objective function* of the *equivalent* ILP instance which can be described as follows:

$$\begin{aligned} & \text{Minimize: } \sum_1^m w_i \cdot b_i \\ & \text{Subject to:} \\ & \quad ILP(s \cup h) \\ & \quad 0 \leq x_i \leq 1, x_i \in var(s \cup h) \end{aligned}$$

where function  $ILP(\varphi)$  maps every clause  $C_i \in \varphi$  into a linear inequality with operator  $>$ . The left-hand side of the linear inequality corresponds to the sum of the literals in  $C_i$  once mapped into integer terms, such that, literal  $x(\bar{x})$  is mapped to integer term  $x(1-x)$ . The right-hand side corresponds to constant 0. After moving the constants to the right, the right-hand side corresponds to constant  $-k$ , where  $k$  is the number of negative literals in clause  $C_i$ . Finally, we add the bounding box constraints that ensure that every integer variable in the ILP instance has domain  $\{0, 1\}$ . It can be easily seen that the implication  $C_i \rightarrow \bar{b}_i$  from  $\bar{b}_i \leftrightarrow C_i$  is unnecessary (as we are optimizing).

*Example 1.* Given the WPMS formula,  $\{(x_1 \vee x_2, 2), \dots, (x_1 \vee \bar{x}_2, 3), (\bar{x}_1 \vee x_2, \infty), (\bar{x}_1 \vee \bar{x}_2, \infty)\}$ , the corresponding ILP formulation is <sup>1</sup>:

$$\begin{aligned} & \text{Minimize: } 2 \cdot b_1 + 3 \cdot b_2 \\ & \text{Subject to:} \\ & \quad x_1 + x_2 + b_1 > 0; \quad \backslash \bar{b}_1 \rightarrow (x_1 \vee x_2) \\ & \quad -x_1 - b_1 > -2; \quad \backslash (x_1 \vee x_2) \rightarrow \bar{b}_1 \\ & \quad -x_2 - b_1 > -2; \quad \backslash \bar{b}_2 \rightarrow (x_1 \vee \bar{x}_2) \\ & \quad x_1 - x_2 + b_2 > -1; \quad \backslash (x_1 \vee \bar{x}_2) \rightarrow \bar{b}_2 \\ & \quad -x_1 - b_2 > -2; \\ & \quad x_2 - b_2 > -1; \\ & \quad -x_1 + x_2 > -1; \quad \backslash \bar{x}_1 \vee x_2 \\ & \quad -x_1 - x_2 > -2; \quad \backslash \bar{x}_1 \vee \bar{x}_2 \\ & \text{Bounds: } 0 \leq x_1 \leq 1; 0 \leq x_2 \leq 1; 0 \leq b_1 \leq 1; 0 \leq b_2 \leq 1; \end{aligned}$$

---

<sup>1</sup> For example, for the first soft clause we get:  $ILP(\{CNF(\bar{b}_1 \leftrightarrow (x_1 \vee x_2))\}) = ILP(\{CNF(\bar{b}_1 \rightarrow (x_1 \vee x_2)), CNF((x_1 \vee x_2) \rightarrow \bar{b}_1)\}) = ILP(\{(x_1 \vee x_2 \vee b_1), (\bar{x}_1 \vee \bar{b}_1), (\bar{x}_2 \vee \bar{b}_1)\}) = \{(x_1 + x_2 + b_1 > 0), ((1-x_1)+(1-b_1) > 0), ((1-x_2)+(1-b_2) > 0)\} = \{(x_1 + x_2 + b_1 > 0), (-x_1 - b_1 > -2), (-x_2 - b_1 > -2)\}$ .

## 4 Experimental Results

In this section we present our intensive experimental investigation on the PMS and WPMS crafted and industrial instances from the 2012 MaxSAT Evaluation. Results on random instances are not included since the translation of WPMS into ILP did not win on any family. We provide results for the ILP translation, the best two solvers for each category of the 2012 MaxSAT Evaluation, and two solvers which did not participate but have been reported to exhibit good performance. We run our experiments on a cluster featured with 2.27 GHz processors, memory limit of 3.9 GB and a timeout of 7200 seconds per instance<sup>2</sup>.

The results are presented in Table 1 following the same criteria as in the 2012 MaxSAT Evaluation. For each solver and family of instances, we present the number of solved instances in parenthesis and the mean solving time. Solvers are ordered from left to right according to the total number of solved instances. The results for the best performing solver in each family are presented in bold. The number of instances of each family is specified in the column under the sign '#'. Since different families may have different number of instances, we also include for each solver the mean ratio of solved instances.

Table 1 shows the results of our experimentation where we compare the following solvers. The solver *ilp* which corresponds to the translation of MaxSAT into ILP (see Section 3)<sup>3</sup>, and the application of the MIP solver IBM-CPLEX studio124 (through C++ API and default parameters). The best two solvers for each category of the 2012 MaxSAT Evaluation: WPMS crafted (*wpm1* [1], *shinms* [9]), WPMS industrial (*pwb02.1* [17, 18], *wpm1*), PMS crafted (*qms0.21* [10], *akms\_ls* [11] and PMS industrial (*qms0.21g2*, *pwb02.1*). Two other solvers that have been reported to exhibit good performance: *bincd2*, which is the new version of the BINCD algorithm [8, 19], with the best configuration reported by authors, and *maxhs* from [5], which is a hybrid SAT-MIP approach.

Table 1(a) presents the results for the PMS crafted instances. The *ilp* approach solves 332 of 372 instances, 35 more than *akms\_ls*. PMS solver *qms0.21* is the third in solved instances but the first in mean ratio with 81.1%.

Table 1(b) presents the results for the WPMS crafted instances. Again, the *ilp* approach is the best one, solving 332 of 372 instances, 14 more than the second one, *wpm1*. It has a clear impact on the *auc-paths*, *auc-scheduling* and *mini-encoding-warehouses* families, solving 100% of the instances in half a second.

Table 1(c) presents the results for the PMS industrial instances. We can see that, although the *ilp* approach is in general not competitive for industrial instances, it wins for *aes*, *bcp-fir* and *bcp-syn* families. In *bcp-syn* it performs much better than the rest of the solvers.

Table 1(d) presents the results for the WPMS industrial instances. Although we can confirm that the *ilp* approach is not competitive for industrial instances, it performs quite well in *upgradeability-problem* family.

---

<sup>2</sup> We thank the Artificial Intelligence Research Institute (IIA) of the Spanish Research Council (CSIC) for the access to their high-performance computing clusters.

<sup>3</sup> Not considering implication  $C_i \rightarrow \bar{b}_i$  gave similar results.

Family	#	<i>ilp</i>	<i>akms.ls</i>	<i>qms0.21</i>	<i>shinms</i>	<i>bincd2</i>	<i>pwo2.1</i>	<i>wpm1</i>	<i>maxhs</i>
frb	25	1153(13)	160(5)	<b>347(25)</b>	44(23)	0.0(0)	151(15)	0.0(0)	2099(5)
job-shop	3	0.0(0)	0.0(0)	42(3)	<b>36(3)</b>	100(3)	93(1)	83(2)	0.0(0)
mxc_ran	96	45(96)	<b>1.1(96)</b>	270(83)	339(76)	85(71)	80(64)	0.0(0)	2164(12)
mxc_srr	62	326.5(38)	<b>282(41)</b>	800(30)	402(23)	109(21)	37(19)	92(9)	1039(13)
mxo_3st	80	13.1(80)	<b>0.5(80)</b>	198(80)	695(78)	8.3(80)	37(63)	208(78)	234(46)
mxo_str	60	337.6(59)	482(38)	<b>6.4(60)</b>	3.5(59)	57(60)	7.7(60)	618(41)	0.0(0)
min-enc_kt	42	<b>163(42)</b>	3199(34)	248(6)	514(5)	275(6)	307(2)	689(3)	0.0(0)
ps_ml	4	34(4)	259(3)	<b>1.8(4)</b>	3.4(4)	49(4)	93(4)	5.3(3)	92(4)
Total	372	<b>332</b>	297	291	271	245	228	136	80
Ratio		76.5%	63.2%	<b>81.1%</b>	77.0%	65.3%	59.3%	41.1%	26.4%

(a) Partial Crafted

Family	#	<i>ilp</i>	<i>wpm1</i>	<i>shinms</i>	<i>akms.ls</i>	<i>pwo2.1</i>	<i>maxhs</i>	<i>bincd2</i>
auc-pat	86	<b>0.5(86)</b>	484(59)	318(84)	2.6(86)	111(19)	35(86)	1415(12)
auc-sch	84	<b>0.4(84)</b>	5.7(84)	5.8(84)	68(84)	7.7(81)	965(78)	142(81)
min-enc-p	56	297(56)	36(53)	<b>8.2(52)</b>	141(40)	<b>0.5(56)</b>	459(31)	33(54)
min-enc-w	18	<b>0.5(18)</b>	20(14)	0.4(1)	20(2)	3.8(14)	0.2(1)	2.1(1)
ps_ml	12	82.82(3)	845.0(5)	<b>128(5)</b>	0.3(2)	4.0(3)	0.1(1)	1073(4)
ran-net	74	<b>533(59)</b>	160(41)	0.0(0)	4061(8)	42(35)	2771(10)	0.0(0)
wcsp-s5-d	21	43(18)	549(14)	<b>744(21)</b>	1556(6)	62(8)	101(6)	128(12)
wcsp-s5-l	21	323(8)	277(15)	<b>201(17)</b>	109(5)	1.7(6)	357(6)	299(13)
Total	372	<b>332</b>	285	264	233	222	219	177
Ratio		<b>78.6%</b>	72.0%	64.8%	45.3%	54.4%	41.6%	45.6%

(b) Weighted Partial Crafted

Family	#	<i>bincd2</i>	<i>qms0.21g2</i>	<i>pwo2.1</i>	<i>shinms</i>	<i>wpm1</i>	<i>ilp</i>	<i>maxhs</i>
aes	7	453(1)	3155(1)	0.0(0)	0.0(0)	0.0(0)	<b>1311(3)</b>	453(2)
bcp-fir	59	44(58)	108(56)	68(56)	14(22)	9.6(55)	<b>63(59)</b>	481(25)
bcp-h-y_si	17	171(16)	<b>358(17)</b>	175(15)	41(16)	137(16)	667(6)	277(11)
bcp-h-y_su	38	245(32)	<b>106(35)</b>	98(25)	282(34)	310(24)	0.0(0)	307(21)
bcp-msp	64	<b>214(38)</b>	452(30)	96(26)	<b>281(22)</b>	320(9)	856(37)	120(1)
bcp-mtg	40	1.2(40)	<b>0.2(40)</b>	0.6(40)	0.6(40)	13(40)	769(29)	115(6)
bcp-syn	74	29(43)	284(35)	22(39)	87(33)	32(41)	<b>19(71)</b>	86(61)
cir-tra-com	4	109(4)	<b>45(4)</b>	200(2)	52(4)	544(4)	6922(1)	0.0(0)
hap-ass	6	728(5)	153(5)	<b>9.1(5)</b>	0.0(0)	289(4)	2125(5)	14(5)
pbo-mqc_ne	84	279(84)	<b>59(84)</b>	222(68)	146(84)	486(35)	1101(6)	400(34)
pbo-mqc_nl	84	79(84)	<b>24(84)</b>	72(82)	180(79)	423(49)	508(6)	364(37)
pbo-rou	15	<b>1.1(15)</b>	3.6(15)	28(15)	4.8(15)	1.2(15)	20(15)	28(14)
pro-ins	12	314(3)	<b>129(12)</b>	0.1(1)	207(4)	0.3(1)	<b>2.7(1)</b>	7.6(1)
Total	504	<b>423</b>	418	374	353	293	239	218
Ratio		78.2%	<b>83.0%</b>	66.3%	63.6%	61.2%	48.9%	43.0%

(c) Partial Industrial

Family	#	<i>wpm1</i>	<i>pwo2.1</i>	<i>bincd2</i>	<i>maxhs</i>	<i>ilp</i>	<i>shinms</i>
hap-ped	100	<b>203(95)</b>	123(87)	545(73)	1089(39)	1892(18)	1204(47)
tim	26	<b>1168(11)</b>	672(7)	169(8)	1250(6)	0.0(0)	2261(5)
upg-pro	100	16(100)	33(100)	76(100)	<b>13(100)</b>	19(100)	0.0(0)
Total	226	<b>206</b>	194	181	145	118	52
Ratio		<b>79.1%</b>	71.3%	67.9%	54.0%	39.3%	22.1%

(d) Weighted Partial Industrial

**Table 1.** Experimental results of ILP translation compared with other solvers.

## 5 Conclusions and Future Work

From the experimentation, we conclude that the translation of WPMS into ILP has the best performance on crafted instances. This is a quite remarkable result since branch and bound solvers, like *akms.ls*, have always dominated this category since 2006. The ILP translation is however not competitive on industrial and random instances. This is something to be studied in depth, and may constitute the seed for new hybrid approaches of ILP and other WPMS algorithms.

## References

1. C. Ansótegui, M. L. Bonet, J. Gabàs, and J. Levy. Improving sat-based weighted maxsat solvers. In *Proc. of the 18th Int. Conf. on Principles and Practice of Constraint Programming (CP'12)*, pages 86–101, 2012.
2. J. Argelich, D. L. Berre, I. Lynce, J. P. M. Silva, and P. Rapicault. Solving linux upgradeability problems using boolean optimization. In *Proceedings First International Workshop on Logics for Component Configuration (LoCoCo)*, pages 11–22, 2010.
3. J. Argelich, C. M. Li, F. Manyà, and J. Planes. Maxsat evaluations, 2011, 2012. <http://www.maxsat.udl.cat>.
4. R. Asín and R. Nieuwenhuis. <http://www.lsi.upc.edu/~rasin/timetabling.html>, 2010.
5. J. Davies and F. Bacchus. Solving maxsat by solving a sequence of simpler sat instances. In *Proc. of the 17th Int. Conf. on Principles and Practice of Constraint Programming (CP'11)*, pages 225–239, 2011.
6. Z. Fu and S. Malik. On solving the partial max-sat problem. In *Proc. of the 9th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'06)*, pages 252–265, 2006.
7. F. Heras, J. Larrosa, and A. Oliveras. MiniMaxSat: A new weighted Max-SAT solver. In *Proc. of the 10th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'07)*, pages 41–55, 2007.
8. F. Heras, A. Morgado, and J. Marques-Silva. Core-guided binary search algorithms for maximum satisfiability. In *Proc. the 25th National Conference on Artificial Intelligence (AAAI'11)*, 2011.
9. K. Honjyo and T. Tanjo. Shinmaxsat. A Weighted Partial Max-SAT solver inspired by MiniSat+, Information Science and Technology Center, Kobe University.
10. M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. Qmaxsat: A partial max-sat solver. *JSAT*, 8(1/2):95–100, 2012.
11. A. Kügel. Improved exact solver for the weighted max-sat problem. To appear.
12. C. M. Li and F. Manyà. Maxsat, hard and soft constraints. In *Handbook of Satisfiability*, pages 613–631. 2009.
13. C. M. Li, F. Manyà, N. O. Mohamedou, and J. Planes. Exploiting cycle structures in Max-SAT. In *Proc. of the 12th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'09)*, 2009.
14. H. Lin and K. Su. Exploiting inference rules to compute lower bounds for Max-SAT solving. In *IJCAI'07*, pages 2334–2339, 2007.
15. H. Lin, K. Su, and C. M. Li. Within-problem learning for efficient lower bound computation in Max-SAT solving. In *Proc. the 23th National Conference on Artificial Intelligence (AAAI'08)*, pages 351–356, 2008.
16. V. Manquinho, J. Marques-Silva, and J. Planes. Algorithms for weighted boolean optimization. In *Proc. of the 12th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'09)*, pages 495–508, 2009.
17. R. Martins, V. M. Manquinho, and I. Lynce. Exploiting cardinality encodings in parallel maximum satisfiability. In *ICTAI*, pages 313–320, 2011.
18. R. Martins, V. M. Manquinho, and I. Lynce. Clause sharing in parallel maxsat. In *LION*, pages 455–460, 2012.
19. A. Morgado, F. Heras, and J. Marques-Silva. Improvements to core-guided binary search for maxsat. In *Proc. of the 15th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'12)*, pages 284–297, 2012.

20. J. D. Park. Using weighted max-sat engines to solve mpe. In *Proc. the 24th National Conference on Artificial Intelligence (AAAI'10)*, pages 682–687, 2002.
21. S. Safarpour, H. Mangassarian, A. G. Veneris, M. H. Liffiton, and K. A. Sakallah. Improved design debugging using maximum satisfiability. In *FMCAD*, pages 13–19. IEEE Computer Society, 2007.
22. D. M. Strickland, E. Barnes, and J. S. Sokol. Optimal protein structure alignment using maximum cliques. *Oper. Res.*, 53(3):389–402, May 2005.
23. M. Vasquez and J. kao Hao. A "logic-constrained" knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite. *Computational Optimization and Applications*, 20:137–157, 2001.
24. H. Xu, R. A. Rutenbar, and K. A. Sakallah. sub-sat: a formulation for relaxed boolean satisfiability with applications in routing. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(6):814–820, 2003.