Influence of CNF Encodings of AtMost-1 Constraints on UNSAT-based PMSAT Solvers

Mohamed El Bachir Menai
Department of Computer Science, College of Computer and Information Sciences
King Saud University, P.O.Box 51178, Riyadh 11543, Saudi Arabia
E-mail: menai@ksu.edu.sa
http://faculty.ksu.edu.sa/menai

Tasniem Nasser Al-Yahya
Department of Computer Science, College of Computer and Information Sciences
King Saud University, P.O.Box 51178, Riyadh 11543, Saudi Arabia
E-mail: talyahya@ccis.imamu.edu.sa

Keywords: artificial intelligence, satisfiability problems, constraint satisfaction, boolean cardinality constraint, CNF encoding, AtMost-1 constraints

Received: July 7, 2012

Partial maximum Boolean satisfiability (Partial MaxSAT or PMSAT) is an optimization variant of Boolean Satisability (SAT). It asks to find a variable assignment that satisfies all hard clauses and the maximum number of soft clauses in a Boolean formula. Several exact PMSAT solvers have been developed since the introduction of the MaxSAT evaluations in 2006, based mainly on the Davis- Putnam-Logemann-Loveland (DPLL) procedure and branch and bound (B&B) algorithms. One recent approach that provides an alternative to B&B algorithms is based on unsatisfiable (UNSAT) core identification. All PMSAT algorithms based on UNSAT identification are dependent on two essential external components: (1) a cardinality constraint encoder for encoding AtMost-1 constraints into conjunctive normal form (CNF); and (2) a SAT solver. Ensuring the effectiveness of both components directly affects the performance of the PMSAT solver. Whereas great advances have been made in PMSAT algorithms based on UNSAT core identification, only a few research work has been conducted to understand the influence of CNF encoding methods on the performance of PMSAT solvers. In this paper, we investigate the influence of three CNF encoding methods for AtMost-1 constraints on an UNSAT-based PMSAT solver. We implement the solver using the pairwise, parallel, and sequential encodings, and evaluate its performance on industrial instances. The experimental results show the impact of the CNF encoding method on the performance of the PMSAT solver. Overall, the best results were obtained with the sequential encoding.

Povzetek: Predstavljena je nova metoda PMSAT, t.j. optimirane variante izpolnjivosti Boolovih enačb (SAT).

1 Introduction

The Boolean satisfiability (SAT) problem is the core of computationally intractable NP-complete problems [7]. Informally, SAT asks if a Boolean expression can be made $True$ by assigning Boolean values to its variables. The maximum satisfiability (MaxSAT) [12] problem is an optimization version of SAT, which consists in finding an assignment that maximizes the number of clauses that are $True$ in a CNF formula. In recent years there has been an increasing interest in designing and implementing MaxSAT solvers. Indeed, state-of-the-art MaxSAT solvers are able to solve large number of instances that were beyond the reach of solvers developed just few years ago.

The partial maximum satisfiability (PMSAT) [5, 22] is a problem between SAT and MaxSAT that is better suited for modeling and solving over constrained problems such as packing, planning, and scheduling. PMSAT instances can be solved using either an exact or stochastic local search method. Exact methods are complete, but are limited to small problem instances. Many exact methods for PMSAT have been introduced in the 2007-2011 SAT conferences, e.g. [10, 14, 15]. Stochastic local search methods for PMSAT can handle very large problem instances, but do not guarantee that an optimal solution will be provided, especially as the instance grows in size [5, 21].

In 2007, the MaxSAT evaluation devoted a special track for exact PMSAT solvers. Most PMSAT solvers submitted to past MaxSAT evaluations are based on B&B methods, e.g. IncWMaxSatz [14, 15] and WMaxSatz+ [13], or DPLL methods, e.g. QMaxSAT and PM2 [1]. The 2007-2011 MaxSAT evaluations show that PMSAT solvers based on B&B methods are superior in the random category, while PMSAT solvers based on DPLL are the best in the

industrial category. DPLL-based solvers for PMSAT are built on top of powerful DPLL-based SAT solvers. Many take advantage of the improvements to DPLL-based SAT solvers, including unsatisfiable (UNSAT) core generation. All PMSAT algorithms based on UNSAT core generation are dependent upon two essential external components: (1) a CNF encoder for Boolean cardinality constraints, which expresses constraints in CNF representation; and (2) a SAT solver, which checks the satisfiability state of the SAT instance and extracts UNSAT cores.

This paper studies the influence of three CNF encoding methods for AtMost-1 constraints on UNSAT-based PMSAT solvers and gives experimental evidence of their impact on the performance of the PMSAT solvers. The rest of the paper is structured as follows. Brief overview of the topics addressed in the paper, namely the CNF encoding problem and PMSAT algorithms using UNSAT core generation are given in Section 2. Related work to CNF encoding methods for AtMost-k constraints on SAT solvers is presented in Section 3. The proposed method and the design of a PMSAT solver with three different encodings for the AtMost-1 constraint are explained in Section 4. Experimental results are presented and analyzed in Section 5. The paper is concluded in Section 6.

2 Background

Boolean cardinality constraints state that at most one Boolean variable (AtMost-1) is allowed to be $True$). Many problems are expressed by CNF clauses and AtMost-1 constraints, such as mixed Horn formulas, planning, and PMSAT [2]. Because it was generally believed that solving such problems through pure CNF encoding is inefficient, many authors have proposed specialized algorithms: Pseudo-Boolean solvers. However, it has been shown [3] that an appropriate CNF encoding method and a robust SAT solver can provide a competitive approach, thus allowing modern SAT techniques, e.g. clause learning, restarts, etc., to be fully functional without the necessity of adapting PMSAT solvers within a mixed ad-hoc solver.

PMSAT was first defined in 1996 by Miyazaki et al. [22], in the context of the optimization of database queries. A PMSAT instance is a CNF formula in which some clauses are soft and the rest are hard. Solving a PMSAT instance consists in finding an assignment that satisfies all the hard clauses and the maximum number of soft clauses.

The following sub-sections discuss the properties of a CNF encoding method and give a quick overview on UNSAT-based PMSAT methods.

2.1 CNF encoding problem

The goal of the CNF encoding problem is the correct and efficient translation of a Boolean cardinality constraint over a set of Boolean variables into a CNF formula. Although there is no general definition of a good encoding technique, such encodings are generally judged in terms of their correctness, consistency enforcing level, and encoding size.

2.1.1 Correctness

Correctness is an essential property that must be preserved in CNF encoding methods. It is formally presented in Definition 1.

Definition 1. *Given a CNF formula F over $X = \{x_1, x_2, ..., x_n\}$, F is said to encode a Boolean cardinality constraint, e.g. $(x_1, x_2, ..., x_n) \leq k$, correctly if and only if, for any complete truth assignment I on X, I satisfies F if and only if I satisfies $(x_1, x_2, ..., x_n) \leq k$.*

2.1.2 Consistency enforcing level

Consistency enforcing methods appeared in the context of constraint satisfaction problems (CSPs) as inference methods to assist searching. It has been shown that applying consistency enforcing methods to CSP instances defined with constraints only in extension, i.e., Boolean cardinality constraints, is equivalent to applying unit propagation to a polynomial SAT encoding of the constraints [11]. Consistency enforcing methods that infer constraints based on pairs of variables are referred to as arc consistency (AC) algorithms. AC ensures that any legal value in the domain of a single variable has a legal match in the domain of any other selected variable. The strength of consistency achieved by the unit propagation rule is reflected in the DPLL performance in many problems. However, achieving stronger consistency is not always beneficial, because of the additional time and space overheads. Thus, there is a trade-off between the effort spent on consistency and that spent on subsequent DPLL search. To date, according to the present research, AC and AC$^+$ are the only consistency enforcing methods incorporated into CNF encoding methods. The AC property is defined below:

Definition 2. *Given a CNF formula F over $X = \{x_1, x_2, ..., x_n\}$, F is said to preserve the AC property when encoding a Boolean cardinality constraint, e.g. $(x_1, x_2, ..., x_n) \leq k$ if and only if, for any partial truth assignment I on X, unit propagation restores AC for $(x_1, x_2, ..., x_n) \leq k$, specifically:*

1. *Unit propagation produces an empty clause when more than k variables in X are assigned to be $True$.*
2. *Unit propagation assigns to $False$ all other variables in X when k variables in X are assigned to be $True$.*
3. *Unit propagation assigns to $True$ all other variables in X when $n - k$ variables in X are assigned to be $False$.*

The definition of the AC$^+$ property is given below:

Definition 3. *Given a CNF formula F over $X = \{x_1, x_2, ..., x_n\}$, F is said to preserve the AC$^+$ property when encoding a Boolean cardinality constraint, e.g.*

$(x_1, x_2, ..., x_n) \leq k$, if and only if, for any complete truth assignment I on X, applying unit propagation on F assigns the same variables of X as restoring AC$^+$ for $(x_1, x_2, ..., x_n) \leq k$, specifically:

1. *Whenever an empty clause is not generated, and*
2. *All the variables of X are assigned.*

Table 1: Summary of CNF encoding methods. The table entry n indicates: Number of variables to be encoded. The columns "Auxiliary variables" and "Auxiliary clauses" indicate the number of new variables and clauses introduced by the encoding, respectively. The "AC/AC$^+$" column shows whether unit propagation for the encoding enforces AC, AC$^+$, both, or neither.

CNF encoding method	Auxiliary variables	Auxiliary clauses	AC /AC$^+$
Pairwise	none	$\frac{n(n-1)}{2}$	Both
Sequential	$n-1$	$2n+n-4$	Both
Parallel	$2n-1$	$\leq 7n - 3\lfloor \log n \rfloor - 6$	none

2.1.3 Encoding size

In practice, reducing the size, i.e., the number of literals or variables, of the resultant CNF formula in an encoding does not guarantee enhanced performance, regardless of how this is measured. Nevertheless, small encoding size is worth aiming for; because computational results can be unpredictable, all else being equal, a smaller encoding is preferable.

The three CNF encoding methods of AtMost-1 cardinality constraints considered in this paper are: pairwise, parallel counter, and sequential counter encoding methods [24]. Table 1 summarizes these methods and their properties. The encodings are presented in the order in which they were introduced.

2.2 UNSAT-based PMSAT solvers

UNSAT-based methods consist in using iteratively SAT solvers to identify and relax UNSAT formulas in PMSAT instances. Any UNSAT subset of clauses in an UNSAT formula is called an UNSAT core. Fu and Malik [10] proposed the first UNSAT-based solver for PMSAT, that consists of the following steps: (1) Identification of UNSAT sub-formulas; (2) Relaxation of each clause in each UNSAT sub-formula by adding a relaxation variable to each clause in an UNSAT sub-formula; and (3) Addition of new Equals-1 constraint indicating that exactly one of these relaxation variables can be assigned the value $True$.

The 2008-2011 MaxSAT evaluations have seen many competitive PMSAT solvers based on the Fu&Malik method, including MSU1.2 [18, 19], MSU4.0 [18], MSUnCore [16, 18-20], PM2 [1], and WPM1 [1].

3 Related work

This section provides some insights into works that compare the performance of various CNF encoding methods for AtMost-k constraints, for different MaxSAT instances.

In [18, 19], sequential counters [24], sorter networks, binary decision diagram (BDD), and bitwise encodings [9] are proposed as alternative encodings to the pairwise encoding used in the Fu&Malik algorithm. The experiments conducted on some selected MaxSAT instances show that the best results were obtained with BDD and bitwise encodings. According to the study in [18], one may conclude that the bitwise encoding is the most appropriate for AtMost-1 constraints.

Marques-Silva and Planes [20], compare BDD and sorting network encoding, in terms of CPU time. Their results show that sorting network encoding is less time consuming than BDD encoding, with a few outliers.

The pairwise, bitwise [9], and sequential counters [24] encodings are tested on a wide range of industrial MaxSAT instances [23]. The results indicate that bitwise encoding is the best one, as already suggested in [18], whereas the performance of the sequential counters is considered as quite good.

4 Proposed method

All UNSAT-based PMSAT solvers are built on top of powerful SAT solvers. Thus, the studies in [6, 8, 17] have motivated the investigation of CNF encoding methods of Boolean cardinality constraints on the performance of UNSAT-based PMSAT solvers. The most appropriate combination of: (1) Boolean cardinality constraint type; (2) CNF encoding method; and (3) SAT solver; are based on the arguments below.

Marques-Silva and Planes [19] have shown that clauses introduced by Equals-1 constraint used in the original Fu&Malik algorithm [10] can be reduced by replacing it with an AtMost-1 constraint, while maintaining the correctness of the algorithm. In order to quantify the impact of CNF encoding methods on the performance of UNSAT-based PMSAT solvers, we implement and test a PMSAT solver based on the following CNF encoding methods for AtMost-1 constraint: pairwise, parallel, and sequential encodings [24]. In the literature only few competitive SAT solvers generating UNSAT cores are described. Research and discussions with some developers of SAT solvers led to the selection of PicoSAT-936 [4] as the SAT/UNSAT solver that will be used to extract UNSAT cores for our PMSAT solver. PicoSAT-936 [4] is a state-of-the-art conflict-driven clause learning (CDCL) SAT solver that comes with the PicoMUS utility to compute minimal UNSAT cores.

Fu&Malik(α)
Input : α is PMSAT instance
Output:
  if α *satisfiable* **then**
    Number of soft clauses falsified in α
  else
    ∞
$Optimal = 0$
while $True$ **do**
  (α_{SAT}) = PMSATtoSAT(α) (call PMSATtoSAT to convert PMSAT instance α to SAT instance α_{SAT})
  $State$ = PicoSAT-936(α_{SAT}) (call PicoSAT-936 solver & return $True$ if α satisfiable, otherwise $False$)
  if $State == True$ **then**
    return $Optimal$
  α_{UNSAT} = PicoMUS(α_{SAT}) (call PicoMUS utility & return minimal UNSAT cores α_{UNSAT})
  $\beta = \emptyset$
  foreach $soft\_clause \in \alpha_{UNSAT}$ **do**
    $\beta = \beta \cup a_{new}$ (a_{new} is a new auxiliary variable)
    $\alpha = (\alpha \backslash soft\_clause) \cup (soft\_clause \vee a_{new})$
  if $\beta == \emptyset$ **then**
    return ∞
  $CNF = Encode\_AtMost - 1(\sum_{a \in \beta} a \leq 1)$ (encode AtMost-1 cardinality constraint to CNF clauses using: Pairwise, parallel, or sequentail encoding)
  $\alpha = \alpha \cup CNF$ (add AtMost-1 cardinality constraint)
  $Optimal = Optimal + 1$

Figure 1: General procedure for proposed method

Algorithm 1 shows the general steps of the proposed UNSAT-based PMSAT solver. The algorithm consists in iteratively calling PicoSAT-936 on a converted PMSAT to SAT instance. PicoSAT-936 determines whether the formula is satisfiable or not, and in case the instance is unsatisfiable, it gives an unsatisfiable core by calling the PicoMUS utility. At this point, the algorithm produces new variables, relaxation variables, one for each soft clause in the unsatisfiable core. The new working instance consists in adding the new variables to the soft clause in the unsatisfiable core, adding a cardinality constraint saying that at most one of the new variables should be $True$ using: pairwise, parallel, or sequential encoding [24]. This procedure is applied until PicoSAT-936 returns satisfiable.

5 Empirical evaluation

This section presents an experimental evaluation of the performance of the UNSAT-based PMSAT solver using the three encodings. The main goal is to assess the effect of the CNF encoding methods, i.e., pairwise, parallel, and sequential. Three versions of our UNSAT-based PMSAT solver, related to CNF encoding methods, were tested: PMSAT_UNSAT/PW (pairwise encoding), PMSAT_UNSAT/PC (parallel encoding), and PMSAT_UNSAT/SC (sequential encoding).

Moreover, the performance of the solver under the three encodings is compared with that of another well-known UNSAT-based solver for PMSAT described in the literature. The solver selected for the comparison is MSUnCore [16, 18-20], introduced in the 2009 MaxSAT evaluation and ranked in $5th$ place. The evaluation is based on CPU time, i.e., the time spent executing user code and system functions. Experiments were performed on an Intel core 2/Linux machine running at 2.50GHz with 4GB of memory. The time limit for the experiments with each instance of each data set was set to 30 minutes. Benchmarks drawn from the 2010 MaxSAT evaluation were chosen. Challenging industrial instance sets were carefully selected (see Table 2).

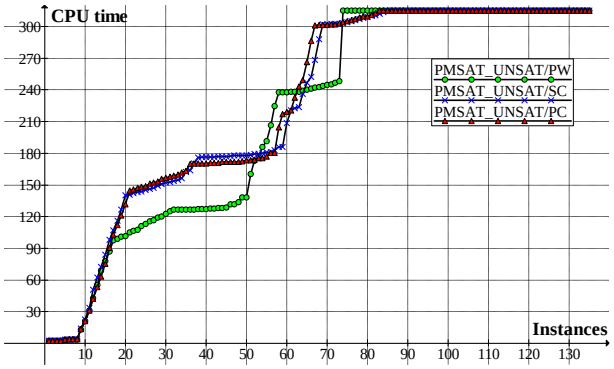

Figure 2: Run time results for selected 2010 industrial instances. The y axis represents the CPU time in seconds.

Figure 1 plots cumulative run time distributions for the solver under the three encodings, for industrial benchmark category. It can be seen that the performance difference of the three encodings considered differs significantly. Figure 1 also shows that PMSAT_UNSAT/PW exhibits a sharp transition in its behavior. Up to a certain point, PMSAT_UNSAT/PW is competitive with the other two encodings. However, as the instances become more complex, the number of clauses added by the pairwise encoding increases quadratically. This produces larger UNSAT cores and increases the computational cost of the pro-

Table 2: Summary of selected industrial instance sets for 2010 benchmarks.

Set Name	No. of instances	Min. nb. of variables - Max. nb. of variables	Min. nb. of clauses - Max. nb. of clauses
Haplotype-Assembly/	6	11642-19918	37730-59200
bcp-fir/	59	54-203287	112-583176
bcp-hipp-yRa1/SU/	38	5352-8334	81318-209643
bcp-hipp-yRa1/simp/	17	166-4477	431-51775
pbo-routing/	15	996-4028	2755-11763

cess. As a result, the performance of PMSAT_UNSAT/PW declines and the computation usually aborts due to excessive memory requirements. As shown in Figure 1, PMSAT_UNSAT/PC and PMSAT_UNSAT/SC exhibit similar behavior, i.e. both have the same transition shape. It can be seen that with the exception of a few outliers, PMSAT_UNSAT/SC outperforms PMSAT_UNSAT/PC, however the differences are essentially negligible. This is explained by the fact that although PMSAT_UNSAT/SC makes use of a sequential, unary counter that requires more clauses for encoding than is the case with PMSAT_UNSAT/PC, PMSAT_UNSAT/SC enforces AC/AC$^+$ properties. Thus, in contrast to PMSAT_UNSAT/PC, a solution can be found faster, and no search is required to check whether or not the constraint is fulfilled.

A summary of the number of solved instances is presented in Table 3. For the industrial category, PMSAT_UNSAT/SC solved two more instances than PMSAT_UNSAT/PC (84 as opposed to 82) and, as expected, both solved significantly more instances than PMSAT_UNSAT/PW. The previous results demonstrate that UNSAT-based solvers for PMSAT are effective in solving problem instances obtained from industrial settings.

Figures 2, 3, and 4 compare MSUnCore with the solver under the three encodings, for the industrial category. The close up of Figure 2 shows that PMSAT_UNSAT/PW outperformed MSUnCore for some instances. But overall, MSUnCore performance was better. Figure 3 shows that MSUnCore outperformed PMSAT_UNSAT/PC for almost all problem instances. Figure 4 shows that the performances of MSUnCore and PMSAT_UNSAT/SC are comparable, although MSUnCore is an improved variant of the Fu&Malik [10] procedure that is implemented with the latest techniques for handling hardware requirements. Finally, these results provide experimental evidence of the influence of CNF encodings of AtMost-1 constraints on UNSAT-based PMSAT solvers.

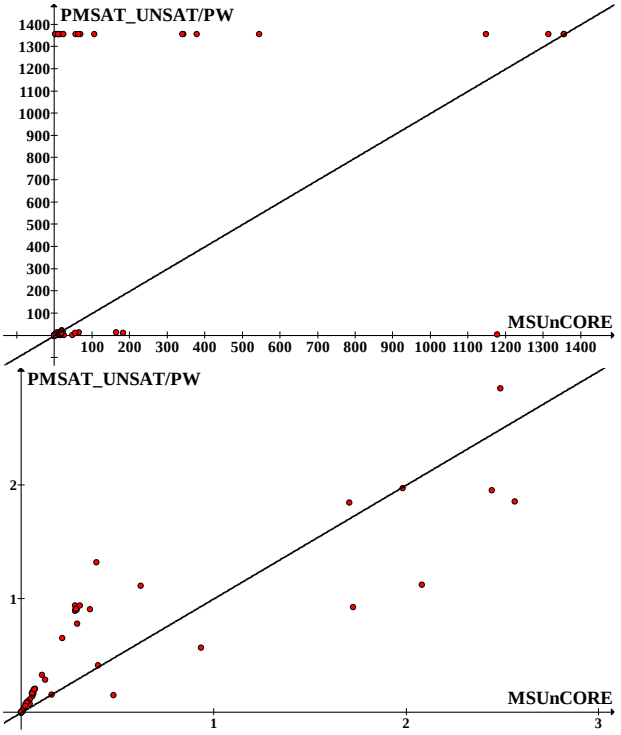

Figure 3: Scatter plots for selected industrial instances: PMSAT_UNSAT/PW versus MSUnCore. x and y axis represent the CPU time in seconds.

6 Conclusions and future work

This paper has presented an empirical study of the influence of the three CNF encoding methods for AtMost-1 constraints on UNSAT-based PMSAT solver: pairwise, parallel, and sequential, with regard to the original Fu&Malik [10] algorithm. An PMSAT solver based on these CNF encoding methods has been implemented and compared to MSUnCore. Overall, empirical results on selected PMSAT instances drawn from the 2010 MaxSAT evaluation have shown that the sequential encoding is the most competitive encoding method among the three, while it was shown to present an unstable behavior on SAT instances [6, 8, 17].

In a future work, we intend to include more CNF encoding methods for AtMost-1 constraints, such as bitwise encoding, binary decision diagram, commander, and sorting network methods. Moreover, we plan to study the effect of different Boolean cardinality constraint types, such as AtLeast-1, Equals-1, and AtMost-k, on UNSAT-based solvers for PMSAT.

Acknowledgement

We thank the anonymous referees for their helpful comments on the manuscript.

Table 3: Performance results on 2010 industrial benchmark. The table's entry of the form $x/y(z)$ corresponds to:x :Number of solved instances, y :Total number of instances, and z : Total CPU time in seconds to solve x instances. Some instances could not be solved for memory issues: 6 instances(*), 4 instances(**), and 1 instance(***).

Set Name	PMSAT_ UNSAT/PW	PMSAT_ UNSAT/PC	PMSAT_ UNSAT/SC	MSUnCore
bcp-hipp-yRa1/simp/	8/17 (3.77)	8/17 (4.09)	8/17 (3.97)	8/17 (5.39)
bcp-hipp-yRa1/SU/	9/38 (93.54)	13/38 (140.68)	12/38 (136.01)	12/38 (3190.4)
bcp-fir/	36/59 (79.0)*	41/59 (87.9)**	44/59 (96.3)***	51/59 (4470.9)
Haplotype-Assembly/	5/6 (61.1)	5/6 (68.4)	5/6 (65.6)	5/6 (72.8)
pbo-routing/	15/15 (10.8)	15/15 (11.4)	15/15 (11.4)	15/15 (4.6)
Total	73/135 (248.21)	82/135 (312.47)	84/135 (318.68)	91/135 (7744.09)

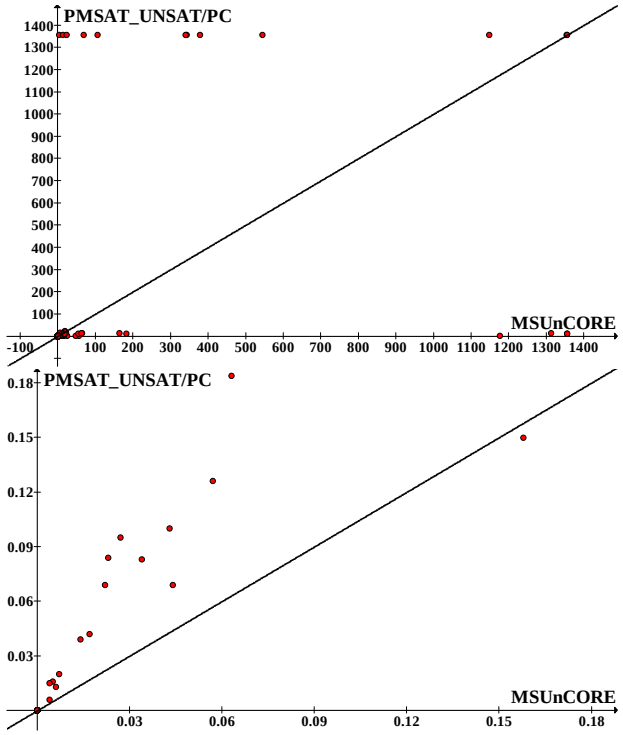

Figure 4: Scatter plots for selected industrial instances: PMSAT_UNSAT/PC versus MSUnCore. x and y axis represent the CPU time in seconds.

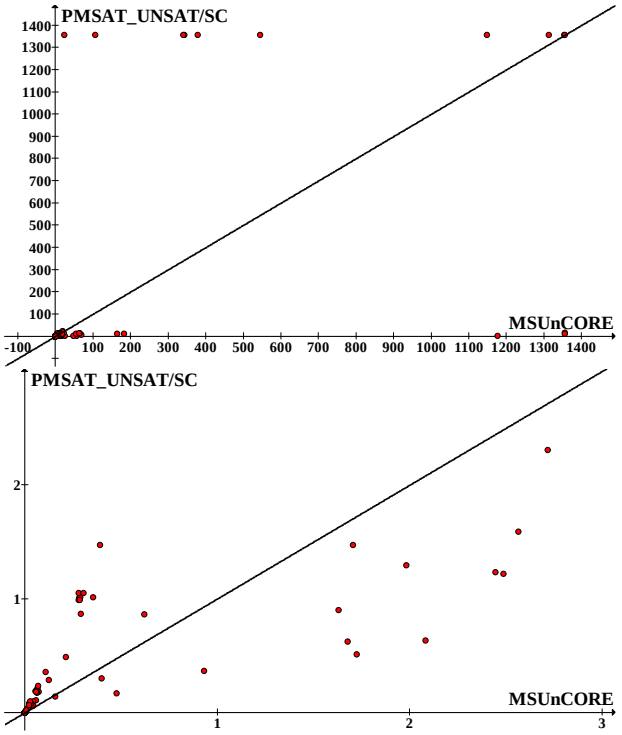

Figure 5: Scatter plots for selected industrial instances: PMSAT_UNSAT/SC versus MSUnCore. x and y axis represent the CPU time in seconds.

References

[1] C. Ansótegui, M. L. Bonet, and J. Levy (2009). Solving (Weighted) Partial MaxSAT through satisfiability testing. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 427–440.

[2] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà (2009). Sequential encodings from Max-CSP into Partial Max-SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 161–166.

[3] O. Bailleux and Y. Boufkhad (2003). Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003*, LNCS 2833, pages 108–122.

[4] A. Biere (2008). PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling, and Computation*, 4(2-4):75–97.

[5] B. Cha, K. Iwama, Y. Kambayashi, and S. Miyazaki (1997). Local search algorithms for Partial MAXSAT. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Conference on Innovative applications of Artificial Intelligence, AAAI'97/IAAI'97*, pages 263–268.

[6] J.-C. Chen (2010). A new SAT encoding of the At-Most-One constraint. In *Proceedings of the 10th*

Workshop of Constraint Modelling and Reformulation.

[7] S. A. Cook (1971). The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM symposium on Theory of computing, STOC'71*, pages 151–158, New York, NY, USA.

[8] A. Frisch and P. Giannaros (2010). SAT encoding of boolean cardinality, some old, some new, some fast, some slow. *Manuscript*, pages 195–201.

[9] A. Frisch, T. Peugniez, A. Doggett, and P. Nightingale (2005). Solving non-boolean satisfiability problems with stochastic local search: A study of encodings. *Journal of Automated Reasoning*, 35:143–179.

[10] Z. Fu and S. Malik (2006). On solving the Partial MAX-SAT problem. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing, SAT'06*, LNCS 4121, pages 252–265.

[11] I. P. Gent. Arc consistency in SAT (2002). In Frank van Harmelen, editor, *Proceedings of the 15th Eureopean Conference on Artificial Intelligence*, Eureopean Conference on Artificial Intelligence, pages 121–125, Lyon, France.

[12] D. S. Johnson (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278.

[13] C.-M. Li, F. Manyà, N. Mohamedou, and J. Planes (2009). Exploiting cycle structures in Max-SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 467–480.

[14] H. Lin and K. Su (2007). Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *Proceedings of the 20th International joint Conference on Artificial Intelligence*, pages 2334–2339, San Francisco, CA, USA.

[15] H. Lin, K. Su, and C.-M. Li (2008). Within-problem learning for efficient lower bound computation in Max-SAT solving. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, volume 1, pages 351–356.

[16] V. Manquinho, J. Marques-Silva, and J. Planes (2009). Algorithms for weighted boolean optimization. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 495–508.

[17] J. Marques-Silva and I. Lynce (2007). Towards robust CNF encodings of cardinality constraints. In *International Conference on Principles and Practice of Constraint Programming, CP 2007*, LNCS 4741, pages 483–497.

[18] J. Marques-Silva and V. Manquinho (2008). Towards more effective unsatisfiability-based maximum satisfiability algorithms. In *Proceedings of the 11th international conference on Theory and applications of satisfiability testing, SAT'08*, LNCS 4996, pages 225–230.

[19] J. Marques-Silva and J. Planes (2007). On using unsatisfiability for solving maximum satisfiability. *Computing Research Repository*, abs/0712.1097.

[20] J. Marques-Silva and J. Planes (2008). Algorithms for maximum satisfiability using unsatisfiable cores. In *Proceedings of the Conference on Design, Automation and Test in Europe, DATE'08*, pages 408–413, New York, NY, USA.

[21] M. B. Menai (2005). A two-phase backbone-based search heuristic for Partial MAX-SAT: An initial investigation. In M. Ali and F. Esposito, editors, *Innovations in Applied Artificial Intelligence, 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2005*, Bari, Italy, June 22-24, 2005, LNAI 3533, pages 681–684.

[22] S. Miyazaki, K. Iwama, and Y. Kambayashi (1996). Database queries as combinatorial optimization problems. In *Proceeding of 1st International Symposium on Cooperative Database Systems for Advanced Applications*, pages 477–483.

[23] F. Morgado and J. Marques-Silva (2011). The MSUnCore MAXSAT solver. Pragmatics of SAT 2011, Workshop of the SAT'11 conference, Ann Arbor, USA.

[24] C. Sinz (2005). Towards an optimal CNF encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, LNCS 3709, pages 827–831.