Journal on Satisfiability, Boolean Modeling and Computation 8 (2012) 95-100

QMaxSAT: A Partial Max-SAT Solver

SYSTEM DESCRIPTION

Miyuki Koshimura[*] koshi@inf.kyushu-u.ac.jp
Tong Zhang zhang@ar.is.kyushu-u.ac.jp
Hiroshi Fujita fujita@inf.kyushu-u.ac.jp
Ryuzo Hasegawa hasegawa@inf.kyushu-u.ac.jp
Department of Informatics,
Graduate School of Information Science and Electrical Engineering,
Kyushu University, Fukuoka, Japan

Abstract

We present a partial Max-SAT solver QMaxSAT which uses CNF encoding of Boolean cardinality constraints. The old version 0.1 was obtained by adapting a CDCL based SAT solver MiniSat to manage cardinality constraints. It was placed first in the industrial subcategory and second in the crafted subcategory of partial Max-SAT category of the 2010 Max-SAT Evaluation. The new version 0.2 is obtained by modifying version 0.1 to decrease the number of clauses for the cardinality encoding. We compare the two versions by solving Max-SAT instances taken from the 2010 Max-SAT Evaluation.

KEYWORDS: *partial Max-SAT solver, Boolean cardinality constraints, CDCL solver*

Submitted January 2011; revised June 2011; published January 2012

1. Introduction

Max-SAT is an optimization version of SAT which tries to find an assignment that maximizes the number of satisfied clauses [15, 14]. This is a natural extension of SAT if we want to know the degree of unsatisfiability. There are two approaches to solve Max-SAT: approximation and exact algorithms. The former computes near-optimal solutions while the latter computes optimal ones. This paper considers exact solutions.

The exact solvers can be classified into two approaches. The one implements a branch and bound scheme and applies several techniques tailored to Max-SAT [19, 13, 16]. Another makes use of a state-of-the-art SAT solver as an inference engine. We call this approach *SAT-based* approach. QMaxSAT (Q-dai[1] Max-SAT solver) follows SAT-based approach and uses a CDCL (Conflict Driven Clause Learning) SAT solver MiniSat [9] version 2.0 as an inference engine.

SAT-based solvers are further classified into two: *satisfiability-based* [6] and *unsatisfiability-based* [1, 17]. QMaxSAT is a satisfiability-based solver. Satisfiability-based and unsatisfiability-based solvers deal with a sequence of satisfiable and unsatisfiable instances, where each instance is generated from the previous one by adding cardinality constraints, and terminate

∗ This work was supported by JSPS KAKENHI(20240003).
1. Kyushu University is called "Kyushu Daigaku" or "Q-dai" in Japanese.

when unsatisfiable and satisfiable instances are found, respectively. A model of the latest satisfiable instance corresponds to a Max-SAT solution.

The paper is structured as follows. In Section 2, we overview the SAT-based approach. Section 3 and 4 present QMaxSAT version 0.1 and 0.2, respectively. In Section 5, we report our experiments. In Section 6, we present some concluding remarks.

2. SAT-Based Max-SAT

For unsatisfiable SAT instances, usual SAT solvers tell nothing but "unsatisfiable." In order to get more information from unsatisfiable instances or add constraints to them, blocking variables are introduced. Satisfiability-based solvers add the variables to all clauses in a Max-SAT instance while unsatisfiability-based solvers add ones to all clauses in an unsatisfiable subset of the instance.

Satisfiability-based Max-SAT Given a Max-SAT instance $\phi = \{C_1, \ldots, C_n\}$, a new blocking variable b_i is added to each clause C_i $(1 \leq i \leq n)$. Solving the Max-SAT problem for ϕ is reduced to minimize the number of true blocking variables in $\phi' = \{C_1 \vee b_1, \ldots, C_n \vee b_n\}$. A minimal satisfying assignment can readily be found by iterative calls to the solver. First run a SAT solver on ϕ' without constraints to get an initial model and count the number k of true blocking variables in the model, then add the constraint saying that the number of true blocking variables have to be less than k, and run the solver again. If the problem is unsatisfied, k is the optimum solution. If not, the process is repeated with the new smaller solution. This iteration is essentially the same as the one in solvers for pseudo-Boolean optimization [20] and an algorithm for MAXONES [5].

There are two satisfiability-based solvers participated in the 2010 Max-SAT Evaluation: Sat4j-MaxSAT [6] and QMaxSAT. Sat4j-MaxSAT translated Max-SAT problems into pseudo-Boolean optimization ones and solve them with a pseudo-Boolean solver Sat4j-PB. The minimization of the number of true blocking variables is treated as an objective function $min : \sum_{i=1}^{n} b_i$ in Sat4j-PB. Sat4j-PB manages the objective function natively, while QMaxSAT manages the minimization by a CNF encoding of cardinality constraints as mentioned in Section 3.

Unsatisfiability-based Max-SAT Let φ be a Max-SAT instance. We iterate the following process until φ becomes satisfiable: Run a SAT solver on φ. If φ is unsatisfiable, we extract an unsatisfiable subset $US = \{C_1, \ldots, C_m\}$ from φ and introduce m new blocking variables $b_i (1 \leq i \leq m)$. Then, we build new φ by replacing C_i with $C_i \vee b_i$ $(1 \leq i \leq m)$ and adding a CNF encoding of $\sum_{i=1}^{m} b_i = 1$. If φ is satisfiable, the iteration terminates. The number of iterations indicates the number of falsified clauses in a Max-SAT solution of the original φ. Note that this iteration is based on the original Fu and Malik work [11]. There are several works for improving this work [1, 17].

3. Version 0.1

The Partial Max-SAT (PMS) problem for a CNF formula, in which some clauses are declared to be soft and the rest are declared to be hard, is the problem of finding an assignment that satisfies all the hard clauses and the maximum number of soft clauses.

Let $C = H \cup S$ be a PMS instance consisting of a set H of hard clauses and a set S of soft clauses. Assume that S consists of n soft clauses, that is, $S = \{S_1, \ldots, S_n\}$. We introduce n new blocking variables $b_i (1 \leq i \leq n)$ and construct a new clause set $C^b = H \cup S^b$ where $S^b = \{S_1 \vee b_1, \ldots, S_n \vee b_n\}$. Finding a PMS solution of C is reduced to find the minimal integer k satisfying C^b and $\sum_{i=1}^{n} b_i \leq k$, that is, minimize the number of true blocking variables while satisfying C^b.

The constraint $\sum_{i=1}^{n} b_i \leq k$ is called the *cardinality constraint*. There are several works on encoding cardinality constraints into CNF formulas [4, 21, 10, 18, 3, 8]. QMaxSAT uses the encoding of Bailleux and Boufkhad [4]. In this encoding, we introduce n new variables $v_i (1 \leq i \leq n)$ for n blocking variables $b_i (1 \leq i \leq n)$. Then, we make a CNF formula $C(b_1, \cdots, b_n, v_1, \cdots, v_n)$ saying:

1. If m blocking variables are assigned 1, the first m variables $v_i (1 \leq i \leq m)$ are going to become assigned 1.

2. If m blocking variables are assigned 0, the last m variables $v_{n-i+1} (1 \leq i \leq m)$ are going to become assigned 0.

The encoding needs $O(n \cdot log\ n)$ auxiliary variables and $O(n^2)$ clauses. With the encoding, we encode the constraint $l \leq \sum_{i=1}^{n} b_i \leq k$ by setting the first l variables $v_i (1 \leq i \leq l)$ to 1, and the last $n - k$ variables $v_i (k < i \leq n)$ to 0.

Algorithm 1 QMaxSAT

```
1:  A = C^b; {C^b : PMS instance augmented with blocking variables}
2:  M = ∅; first = true;
3:  while (solve(A)) do
4:    Let M be a model of A;
5:    "count the number k of true blocking variables in M";
6:    if (first) then
7:      first = false;
8:      A = A ∧ C(b_1, ··· , b_n, v_1, ··· , v_n); {augment the constraint to A}
9:    end if
10:   for i = k to n do
11:     v_i = 0;
12:   end for{add the constraint Σ_{i=1}^{n} b_i < k}
13: end while
14: return M;
```

Algorithm 1 shows the QMaxSAT algorithm. The function `solve(A)` denotes the core part of the SAT solver which returns `false` when a SAT instance `A` is unsatisfiable and `true` when `A` is satisfiable. In the latter case, a model M of `A` is obtained through an array from which we count the number k of true blocking variables in M (lines 4,5).

After we obtain the first model of C^b, we build a CNF formula $C(b_1, \cdots, b_n, v_1, \cdots, v_n)$ which encodes a cardinality constraint (line 8). For every model obtained through `solve(A)`, we introduce extra constraint (lines 10,11,12). The k decreases as the procedure progresses and is bounded below by 0. Thus, k converges to a non-negative integer at which we obtain a PMS solution (line 14). We conclude that C is unsatisfiable if `M` keeps the initial value $\emptyset$.

4. Version 0.2

A drawback of version 0.1 arises from the number of clauses for encoding Boolean cardinality constraints. Assuming that there are tens of thousands of soft clauses, the encoding needs hundreds of millions clauses. In our experience, the clauses cannot be held in 4GB memory when the number of soft clauses is greater than nine thousands.

There are several CNF encodings of cardinality constraints with better space complexity than that of version 0.1. For example, Eén and Sörensson propose the use of BDDs, that guarantee a space complexity of $O(n \cdot k_1)$, and sorting networks, that guarantee a space complexity of $O(n \cdot log^2 n)$ [10] where k_1 is the number of true blocking variables in the first model M in the procedure (see Algorithm 1). Sinz also proposes a sequential counter with $O(n \cdot k_1)$ space complexity [21].

Our choice is to improve the Bailleux and Boufkhad's encodings because of its simplicity as follows. The encoding includes the following conjunction of clauses:

$$\bigwedge_{\substack{0 \le \alpha \le \lfloor n/2 \rfloor \\ 0 \le \beta \le \lceil n/2 \rceil \\ \alpha + \beta = \sigma \\ 0 \le \sigma \le n}} (C_1(\alpha, \beta, \sigma) \wedge C_2(\alpha, \beta, \sigma))$$

where n is the number of soft clauses, $C_1(\alpha, \beta, \sigma)$ and $C_2(\alpha, \beta, \sigma)$ are the CNF representations of the relations $\alpha + \beta \le \sigma$ and $\sigma \le \alpha + \beta$, respectively. We can eliminate clauses $C_1(\alpha, \beta, \sigma)$ and $C_2(\alpha, \beta, \sigma)$ satisfying $k_1 < \sigma \le n$ because such clauses are meaningless when $\sum_{i=1}^{n} b_i < k_1$. This elimination reduces the number of clauses for the encoding from $O(n^2)$ to $O(n \cdot k_1)$.

Recently, CNF encodings of cardinality constraints with better space complexity than the above encodings are proposed [3, 8]. They guarantee $O(n \cdot log^2 k_1)$ space complexity. Replacing the current encoding with these new encodings is one of the future works.

5. Experimental Results

We implemented QMaxSAT [2.] based on MiniSat 2.0. We modified only the top-level part of MiniSat to manipulate cardinality constraints. The other parts remain unchanged. All parameters also maintain default values.

Table 1 summarizes the results obtained by running the two versions of QMaxSAT on the PMS problems of the fifth Max-SAT evaluation (Max-SAT 2010) [12, 2]. The problems are divided into three different categories: random, crafted, and industrial. The second column shows the number of instances of the corresponding category. The third and fourth columns show the average cpu time in seconds for instances solved by version 0.1 and 0.2, respectively. The number in parentheses indicates the number of solved instances. The experiments are done on Core i5-750 (4-core 2.66GHz) machine with 4GB memory. We set the timeout to 1800 seconds.

We succeeded in increasing the numbers of solved instances with the new version 0.2 from 294 to 295 for the crafted category and from 379 to 391 for the industrial category.

2. QMaxSAT is available from http://sites.google.com/site/qmaxsat/.

Table 1. Comparison of two versions for partial Max-SAT problems of Max-SAT 2010

	#INS.	v. 0.1	v. 0.2
Random	240	45.0 (31)	30.5 (31)
Crafted	385	101.7 (294)	94.8 (295)
Industrial	497	55.3 (379)	67.1 (391)

Recall that the old version 0.1 was the winner of the industrial category. Thus, the new version 0.2 beats the winner.

There are 13 instances for which the old version fails to generate the cardinality constraints because they have many soft clauses. They all are in the industrial category and the number of their soft clauses are from 9188 to 101248. For these 13 instances, the new version succeeds in generating the cardinality constraints. It also succeeds in solving 8 of 13 within 1800 seconds.

6. Conclusion

We have presented QMaxSAT, a partial Max-SAT solver based on a SAT solver MiniSat. Experimental results show that QMaxSAT is good for problems in the industrial and crafted category while it is not so good for those in the random category. This phenomenon coincides with that of MiniSat for SAT problems. Thus, it seems reasonable to claim that the efficiency of QMaxSAT comes from that of MiniSat and the CNF encoding of cardinality constraints.

We implemented QMaxSAT by adapting MiniSat to manage cardinality constraints for soft clauses. The implementation is simple in the sense that we modified only the input routine and the `main` function of MiniSat for introducing blocking variables and manipulating the cardinality constraints on them. The source level difference between QMaxSAT and MiniSat is in a hundred and ten of lines which includes a C++ code for generating cardinality constraints. This means that the syntactical difference is about 5 %.

The QMaxSAT implementation is modular and one could plug in (i) another encoding of cardinality constraints, and (ii) another SAT solver. Replacing the encoding or SAT solver with others is an interesting future work. We also plan to extend QMaxSAT to solve weighted Max-SAT problems.

References

[1] Carlos Ansótegui, María Luisa Bonet, and Jordi Levy. Solving (Weighted) Partial MaxSAT through Satisfiability Testing. In *Proc. of SAT 2009*, pages 427–440, 2009.

[2] Josep Argelich, Chu-Min Li, Felip Manyà, and Jordi Planes. The First and Second Max-SAT Evaluations. *JSAT*, **4**:251–278, 2008.

[3] Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality Networks and Their Applications. In *Proc. of SAT 2009*, pages 167–180, 2009.

[4] Olivier Bailleux and Yacine Boufkhad. Efficient CNF Encoding of Boolean Cardinality Constraints. In *Proc. of CP 2003*, pages 108–122, 2003.

[5] Olivier Bailleux and Yacine Boufkhad. Full CNF Encoding : The Counting Constraints Case. In *Proc. of SAT 2004*, pages 263–268, 2004.

[6] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *JSAT*, **7**:59–64, 2010.

[7] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, **185** of *FAIA*. IOS Press, 2009.

[8] Michael Codish and Moshe Zazon-Ivry. Pairwise Cardinality Networks. In *Proc. of LPAR-16*, pages 154–172, 2010.

[9] Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In *Proc. of SAT 2003*, pages 502–518, 2003.

[10] Niklas Eén and Niklas Sörensson. Translating Pseudo-Boolean Constraints into SAT. *JSAT*, **2**:1–26, 2006.

[11] Zhaohui Fu and Sharad Malik. On Solving the Partial MAX-SAT Problem. In *Proc. of SAT 2006*, pages 252–265, 2006.

[12] Federico Heras, Javier Larrosa, Simon de Givry, and Thomas Schiex. 2006 and 2007 Max-SAT Evaluations: Contributed Instances. *JSAT*, **4**:239–250, 2008.

[13] Federico Heras, Javier Larrosa, and Albert Oliveras. MiniMaxSat: An Efficient Weighted Max-SAT Solver. *J. of Artificial Intelligence Research*, **31**:1–32, 2008.

[14] Katsutoshi Hirayama and Makoto Yokoo. *-SAT: Extentions of SAT. *J. of JSAI*, **25**(1):105–113, 2010. in Japanese.

[15] Chu Min Li and Felip Manyà. *MaxSAT, Hard and Soft Constraints*, chapter 19, pages 613–631. Volume 185 of Biere et al. [7], 2009.

[16] Han Lin, Kaile Su, and Chu-Min Li. Within-Problem Learning for Efficient Lower Bound Computation in Max-SAT Solving. In *Proc. of AAAI-08*, pages 351–356, 2008.

[17] Vasco Manquinho, Joao Marques-Silva, and Jordi Planes. Algorithms for Weighted Boolean Optimization. In *Proc. of SAT 2009*, pages 495–508, 2009.

[18] Joao Marques-Silva and Inês Lynce. Towards Robust CNF Encodings of Cardinality Constraints. In *Proc. of CP 2007*, pages 483–497, 2007.

[19] Knot Pipatsrisawat and Adnan Darwiche. Clone: Solving Weighted Max-SAT in a Reduced Search Space. In *Proc. of AI 2007*, pages 223–233, 2007.

[20] Olivier Roussel and Vasco Manquinho. *Pseudo-Boolean and Cardinality Constraints*, chapter 22, pages 695–733. Volume 185 of Biere et al. [7], 2009.

[21] Carsten Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *Proc. of CP 2005*, pages 827–831, 2005.