Artificial Intelligence 196 (2013) 77–105

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

SAT-based MaxSAT algorithms [☆]

Carlos Ansótegui [a], Maria Luisa Bonet [b], Jordi Levy [c,*]

[a] Universitat de Lleida (DIEI), Jaume II 69, Lleida, Spain
[b] Universitat Politècnica de Catalunya (LSI), J. Girona 1-3, Barcelona, Spain
[c] Artificial Intelligence Research Institute (IIIA, CSIC), Campus UAB, Bellaterra, Spain

ARTICLE INFO

Article history:
Received 19 June 2012
Received in revised form 3 January 2013
Accepted 13 January 2013
Available online 21 January 2013

Keywords:
MaxSAT
SAT
Boolean optimization

ABSTRACT

Many industrial optimization problems can be translated to MaxSAT. Although the general problem is NP hard, like SAT, many practical problems may be solved using modern MaxSAT solvers. In this paper we present several algorithms specially designed to deal with industrial or real problems. All of them are based on the idea of solving MaxSAT through successive calls to a SAT solver. We show that this SAT-based technique is efficient in solving industrial problems. In fact, all state-of-the-art MaxSAT solvers that perform well in industrial instances are based on this technique. In particular, our solvers won the 2009 partial MaxSAT and the 2011 weighted partial MaxSAT industrial categories of the MaxSAT evaluation. We prove the correctness of all our algorithms. We also present a complete experimental study comparing the performance of our algorithms with latest MaxSAT solvers.

© 2013 Elsevier B.V. All rights reserved.

1. Introduction

The MaxSAT problem is a generalization of the satisfiability problem. The idea is that sometimes not all restrictions 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 problem. In this case, we can divide the restrictions in two groups: the clauses 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 clauses, where the weight is the penalty to falsify the clause. The idea is that not all restrictions are equally important. The addition of weights to clauses makes the instance *weighted*, and the separation into hard and soft clauses makes the instance *partial*. Given a weighted partial MaxSAT instance, we want to find the assignment that satisfies the hard clauses, and the sum of the weights of the falsified clauses is minimal. Such an assignment will be optimal in this context.

The weighted partial MaxSAT problem is a natural combinatorial problem, and it can be used in several domains, such as: scheduling [40] and timetabling [9] problems, FPGA routing [41], software package installation [7], design debugging [36], bioinformatics [39], probabilistic reasoning [34], etc. However, state-of-the-art solvers have not yet experienced the same success as SAT solvers for the Satisfiability problem in the industrial field. Weighted Partial MaxSAT is an NP-hard problem, so our aim is to produce efficient solvers to handle real/industrial problems that although generally big in size, are not as hard as the worst case instances.

☆ Research partially supported by the Ministerio de Economía y Competividad research projects ARINF TIN2009-14704-C03-01, TASSAT TIN2010-20967-C04-01/03/04.

* Corresponding author.
E-mail addresses: carlos@diei.udl.cat (C. Ansótegui), bonet@lsi.upc.edu (M.L. Bonet), levy@iiia.csic.es (J. Levy).
URLs: http://www.lsi.upc.edu/~bonet (M.L. Bonet), http://www.iiia.csic.es/~levy (J. Levy).

0004-3702/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.artint.2013.01.002

Originally, MaxSAT solvers (*wmaxsatz* [23], *minimaxsat* [18], *incwmaxsatz* [24]) were based on a depth-first branch and bound solving schema. Recently, the use of SAT solvers for solving MaxSAT problems has emerged as a new paradigm. The core idea is to reduce an optimization problem to a sequence of decision problems which are solved with the appropriated solver. This general approach comes from the observation that many important optimization problems fall into the class of FP^{NP}, which is the class of functions that can be computed polynomially with a SAT oracle [33]. This idea has been applied in [15] to solve scheduling optimization problems through a sequence of Constraint Satisfaction Problems and in [20] to solve planning problems through a sequence of SAT instances.

Fu and Malik [17,16] describe a MaxSAT algorithm in this direction. This algorithm can be considered as our initial source of inspiration. Since then, there has been a development of solvers based on satisfiability testing: *sat4j* [22], *wbo* and *msuncore* [25], *wpm1* [4], *wpm2* [5], *pwbo* [32], *bincd* [19] and *maxhs* [13]. Analyzing the results of the latest MaxSAT evaluations [8] we can conclude that, in general, the branch and bound solvers are more competitive on random problems, while solvers based on calls to a SAT solver are better for industrial or real problems.

The purpose of this paper is to summarize all our contributions to the development of SAT-based MaxSAT solvers [2–5], as well as to present some new results. They include detailed proofs of the correctness of all the algorithms, the introduction of the notion of MaxSAT reducibility and some of its properties, and a complete experimental comparison of our solvers with the present best performing solvers.

We present our algorithm WPM1 [4], which is the weighted version of the Fu and Malik algorithm [17,16], designed originally for Partial MaxSAT. Recently, in [2] we show how we can improve its performance by breaking the symmetries introduced by the reformulation of the MaxSAT problem into a sequence of SAT problems. We also show how we can force the SAT solver to prioritize the discovery of higher quality unsatisfiable cores to boost the overall search process.

We also present our algorithm PM2 [4,3] for Partial MaxSAT, which essentially follows the same search strategy as the Fu and Malik algorithm, but using just one blocking variable per clause. Finally, we present our WPM2 [5] algorithm which is almost a generalization of the PM2 algorithm for Weighted MaxSAT. The WPM2 algorithm was the first algorithm of its class. Although we provide an implementation of WPM2, we mostly focus on the description of a general architecture that can be parametrized to obtain more competitive solvers.

Our solvers based on PM2 and WPM1 solvers won the 2009 Partial MaxSAT category and 2011 Weighted Partial MaxSAT industrial categories of the MaxSAT evaluation, respectively. In this paper we present a complete experimental study with the best performing solvers of the MaxSAT 2011 evaluation and other solvers which have been recently published and did not take part.

This paper proceeds as follows. Section 2 presents some preliminary concepts. Section 3 briefly introduces the main basic ideas behind SAT-based MaxSAT solvers. Section 4 describes the concepts of cost-preservation, MaxSAT equivalence and MaxSAT reducibility. These notions play the same roles for MaxSAT formulas as equisatisfiability and logical equivalence for SAT formulas. Section 5 presents the Fu and Malik's algorithm [17,16] and proves its correctness.

Section 6 describes the problem of symmetries and shows how to break them. Section 7 presents the WPM1 algorithm, proves its correctness and describes the problem of the quality of the cores. Section 8 introduces a stratified approach to come up with higher quality cores in WPM1. This stratified approach is generalized, for any MaxSAT solver preserving MaxSAT reducibility, in Section 9. Sections 10 and 11 present the PM2 and WPM2 algorithms, respectively, and prove their correctness. Finally, Section 12 presents the experimental evaluation.

2. Preliminaries

We consider an infinite countable set of *boolean variables* $\mathcal{X}$. A *literal* l is either a variable $x_i \in \mathcal{X}$ or its negation $\neg x_i$. A *clause* C is a finite set of literals, denoted as $C = l_1 \vee \cdots \vee l_r$, or as $\Box$ for the empty clause. A *SAT formula* φ is a finite set of clauses, denoted as $\varphi = C_1 \wedge \cdots \wedge C_m$.

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 C. A clause is called *hard* if the corresponding weight is infinity, otherwise the clause is called *soft*.

A (*weighted partial*) *MaxSAT formula* is a multiset of weighted clauses

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

where the first m clauses are soft and the last m' clauses are hard. Given a weighted partial MaxSAT formula φ, we define the SAT formulas $\varphi_{soft} = \{C_1, \ldots, C_m\}$, $\varphi_{hard} = \{C_{m+1}, \ldots, C_{m+m'}\}$ and $\varphi_{plain} = \varphi_{soft} \cup \varphi_{hard}$.

The set of variables occurring in a formula φ is noted as $\mathrm{var}(\varphi)$.

Definition 1 *(Partial and total truth assignment).* A (*partial*) *truth assignment* is a function $I : X \to \{0, 1\}$, where $X \subset \mathcal{X}$. This function can be extended to variables from $\mathcal{X} \setminus X$, literals, clauses, SAT formulas and MaxSAT formulas, resulting into a function from formulas to formulas, as follows.

For variables $x_i \notin X$, $I(x_i) = x_i$.

For literals, $I(\neg x_i) = 1 - I(x_i)$, if $x_i \in X$; otherwise, $I(l) = l$.

For clauses, $I(l_1 \vee \cdots \vee l_r) = I(l_1) \vee \cdots \vee I(l_r)$ and $I(\Box) = 0$, simplified considering $1 \vee C = 1$ and $0 \vee C = C$.

For SAT formulas $I(C_1 \wedge \cdots \wedge C_r) = I(C_1) \wedge \cdots \wedge I(C_r)$ and $I(\emptyset) = 1$, simplified considering $1 \wedge \varphi = \varphi$ and $0 \wedge \varphi = 0$.

Download English Version:

https://daneshyari.com/en/article/377015

Download Persian Version:

https://daneshyari.com/article/377015

Daneshyari.com