Journal on Satisfiability, Boolean Modeling and Computation 9 (2015) 129-134

MSCG: Robust Core-Guided MaxSAT Solving

SYSTEM DESCRIPTION

Antonio Morgado ajrm@sat.inesc-id.pt
Alexey Ignatiev aign@sat.inesc-id.pt
Joao Marques-Silva jpms@tecnico.ulisboa.pt
INESC-ID, IST
University of Lisbon
Lisbon, Portugal

Abstract

Maximum Satisfiability (MaxSAT) is a well-known optimization version of Propositional Satisfiability (SAT) that finds a wide range of practical applications. This work describes and evaluates the *Maximum Satisfiability using the Core-Guided approach* solver (**MSCG**), which is a robust MaxSAT solver that participated in the MaxSAT Evaluation 2014.

KEYWORDS: *maximum satisfiability, core-guided, Boolean optimization*

Submitted December 2014; revised May 2015; published December 2015

1. Introduction

Maximum Satisfiability (MaxSAT) is the problem of determining a minimum cost assignment to the variables of a given set of clauses, where each falsified clause incurs in a penalty cost. MaxSAT has been applied to relevant applications that include planning, fault localization in C code and design debugging, among others (see [9] for references).

MaxSAT Evaluations have been held since 2006. In recent evaluations a trend has emerged where core-guided algorithms have been particularly successful in industrial categories. **MSCG** is a new state-of-the-art MaxSAT solver that includes different core-guided algorithms. The competition version of **MSCG** participated in the latest MaxSAT Evaluation 2014, where it ranked second place in weighted partial, and third place in partial of the industrial categories (disregarding portfolio solvers). This paper describes the algorithms and techniques included in **MSCG**.

The paper is organized as follows. Section 2 describes the main architectural components of **MSCG**, while Section 3 presents experimental data comparing **MSCG** with the winners of the latest MaxSAT Evaluations. Section 4 concludes the paper.

2. MSCG - Main Components

This section describes the main components of **MSCG**. Figure 1 shows the architecture of **MSCG** with respect to its main components. Each of the components is detailed in the following sections.

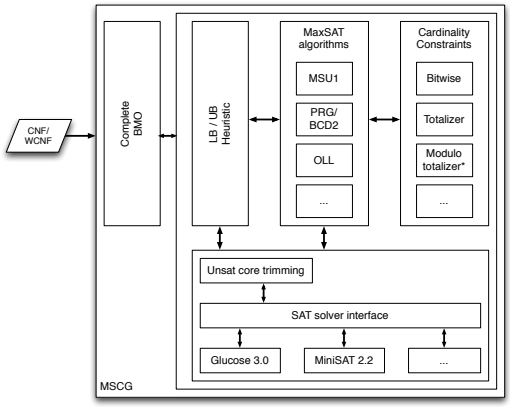

Figure 1. Architecture of **MSCG**

2.1 MaxSAT Algorithms

MSCG gathers different core-guided MaxSAT algorithms. In this paper, we reference the following MaxSAT algorithms (used by **MSCG** in the MaxSAT Evaluation 2014): the MSU1 algorithm [3], the Progression algorithm [9] (with BCD2 as the base algorithm), and an incremental version of the OLL algorithm [8] (see Section 2.4). The use of incrementality allows a rearrangement of the algorithm. Namely, as published in [8], the original OLL asserts the output of the created sums by adding soft unit clauses (of the corresponding output literals). In this case, removing constraints not useful anymore can be practically inefficient. In contrast to the original implementation of OLL [8], the incremental version of OLL asserts the outputs of the created sums with the use of assumption literals. This allows the solver to remove unneeded constraints incrementally *"on the fly"*.

2.2 Cardinality Constraint Encodings

The algorithms in **MSCG** rely heavily on the ability of the solver to encode cardinality constraints into CNF, in particular to transform the atMostK cardinality constraint[1.] $l_1 + \ldots + l_n \leq k$ into a set of clauses. **MSCG** uses different cardinality constraints encodings, in particular in this work we consider the Bitwise encoding [12] (for which $k = 1$), a variant of the Modulo Totalizer encoding [11], the Totalizer encoding [2] (these correspond to the encodings used by **MSCG** in the MaxSAT Evaluation 2014) and its iterative version [6].

The idea of the variant of the Modulo Totalizer is to use the Modulo Totalizer encoding but approximating it to the technique of k-cardinality. This variant was motivated by the proposed future work in the Conclusions section of [11], where their goal is to do k-cardinality of the Modulo Totalizer instead of approximating it.

1. In **MSCG** only atMostK cardinality constraints are considered, as such, all the other cardinality constraints are disregarded in this description.

2.3 Lower and Upper Bound Heuristics

Before invoking a MaxSAT algorithm, **MSCG** computes both a lower and an upper bound on the MaxSAT solution. The lower bound corresponds to a minimum cost for which it is known that there is no MaxSAT solution smaller than it. The upper bound corresponds to the cost of an assignment satisfying all the hard clauses (if present).

Initially, the solver makes a call to the SAT solver using solely the hard clauses. This call checks whether the CNF formula comprising the hard clauses is satisfiable or not. If it is not, **MSCG** returns an unsatisfiable status. Otherwise, a model is provided, which satisfies all the hard clauses, and may or may not satisfy the soft clauses. The cost of this model represents an initial upper bound on the MaxSAT solution.

When requested, the lower bound in **MSCG** is computed as in the MSU3 algorithm [9], that is the SAT solver is invoked with all the clauses in the formula. If the formula is unsatisfiable then an unsatisfiable core is obtained, all the soft clauses of the core are removed from the formula, and the SAT solver is called again on the new formula. The process continues until the formula becomes satisfiable. At this point, the lower bound is given by the sum of the minimum weights in each of the unsatisfiable cores. Additionally, the model returned by the last SAT call satisfies all the hard clauses (since these are not removed), and thus the cost of this model is compared against the previous upper bound, which may result in a new refined upper bound.

2.4 SAT Solver Interface

A minimal interface has been created in **MSCG** to access all the SAT solver's functionality. This interface builds on top of the SAT solver's API, and is general enough to allow the tool to be able to select the underlying SAT solver used. In order to use this interface with a specific SAT solver, we require the SAT solver to be capable of handling assumptions, to return a model of satisfiable formulas, and to compute unsatisfiable cores of unsatisfiable formulas. In **MSCG** the assumptions are associated with the soft clauses (one-to-one correspondence). Hard clauses do not contain assumptions. Whenever a formula is declared unsatisfiable by the SAT solver, then an unsatisfiable core is obtained from the SAT solver containing the assumptions representing the soft clauses in the core. **MSCG** integrates several SAT solvers including Glucose 3.0[2.] and Minisat 2.2[3.].

All the algorithms in **MSCG** use the SAT solver non-incrementally, i.e. the SAT solver is restarted after each call, the exception being the incremental version of OLL, which maintains clauses between calls to the SAT solver.

2.5 Additional Features

Complete BMO Condition In the case of weighted instances **MSCG** checks if the instance respects the complete BMO condition [5], which consists in the following. Given a MaxSAT formula φ, φ respects the complete BMO condition if for any clause $(c, w) \in \varphi$ its weight w is larger than the sum of weights that are smaller than w, i.e. $w > \sum_{(c_i,w_i)\in\varphi;\ w_i<w} w_i$. If the instance respects the complete BMO condition, then **MSCG**

2. http://www.labri.fr/perso/lsimon/glucose
3. https://github.com/niklasso/minisat

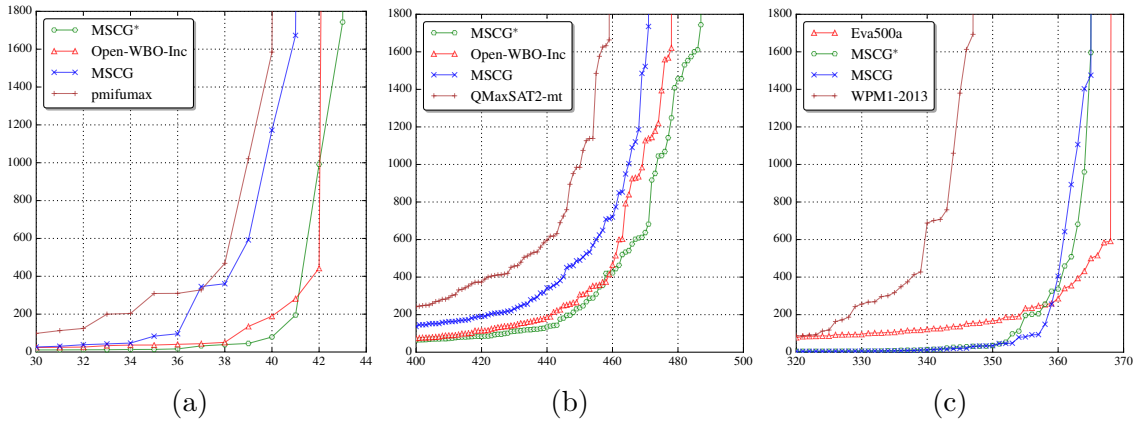

Figure 2. Results for Industrial MaxSAT benchmarks from MaxSAT Evaluation 2014 (a) Plain MaxSAT benchmarks (b) Partial MaxSAT benchmarks (c) Weighted Partial MaxSAT benchmarks

solves the instance by considering different levels of weights starting from the largest weight and disregarding the levels with smaller weights. Each level is solved as a non-weighted MaxSAT instance (since all weights in the level are equal), and when a solution is found for that level, the corresponding clauses of the level are hardened for the following levels. When the last level is processed, a solution of the original weighted MaxSAT formula is found [5].

Unsatisfiable Core Trimming In order to reduce the size of the cores obtained, the technique of *unsatisfiable core trimming* has been implemented in **MSCG**. Whenever a new unsatisfiable core φ_C is extracted, the SAT solver is called again with the soft clauses belonging to φ_C together with all the hard clauses. Since the new CNF instance is unsatisfiable, a new core φ'_C is extracted. Note that $\varphi'_C \subseteq \varphi_C$. If $\varphi'_C \subset \varphi_C$, then φ_C is replaced by φ'_C and the trimming algorithm continues until a fixed point is found (i.e. when $\varphi'_C = \varphi_C$) or a maximum number of trimming attempts are done.

3. Experimental Results

This section presents the experimental data obtained for **MSCG** for all the industrial benchmarks of the MaxSAT Evaluation 2014[4]. The experiments were performed on a cluster with Intel Xeon E5-2630-v2 2.60GHz processors with 64GB of RAM. All the tested solvers were set to run for each instance for 1800 seconds with 3.5GB of memory limit. The best performing solvers in each industrial category of both the 2014 and 2013 MaxSAT Evaluations (disregarding portfolio solvers) were selected for the experiments. As such, pmifumax[5] and Open-WBO-Inc [7] were considered for Plain MaxSAT instances while QMaxSAT2-mt [4] and Open-WBO-Inc [7] were chosen for Partial MaxSAT. Finally, WPM1-2003 [1] and Eva500a [10] were used for Weighted Partial MaxSAT benchmarks. Two versions of **MSCG** (*MSCG* and *MSCG**) were considered and are described in what follows.

4. http://www.maxsat.udl.cat
5. http://sat.inesc-id.pt/~mikolas/sw/mifumax

MSCG *MSCG* is the competition version of **MSCG**, which was submitted to the MaxSAT Evaluation 2014. *MSCG* corresponds to a wrapper made on top of **MSCG** that depending on the formula type sets which components to use. In the case of plain MaxSAT instances, *MSCG* invokes the MSU1 algorithm [3] using the Bitwise cardinality constraint encoding [12]. For partial MaxSAT instances *MSCG* invokes the Progression algorithm [9] with BCD2 as the base algorithm, and using the variant of the Modulo Totalizer cardinality encoding [11]. Finally, for weighted (partial) MaxSAT instances *MSCG* invokes the incremental version of the OLL algorithm [8], using the Totalizer cardinality encoding [2]. Whenever a weighted (partial) instance respect the complete BMO condition [5], then the complete BMO approach is used (see Section 2.5). Additionally, *MSCG* uses Glucose 3.0 in the non-incremental mode (except for the OLL algorithm), trims all cores at most 5 times and always computes both an upper and a lower bound (as described in Section 2.3).

MSCG* After the MaxSAT Evaluation, a new improved version of the solver referred to as *MSCG** was created, which resulted from the following improvements. Independently of the type of instance (that is, for all the categories) the MaxSAT algorithm invoked corresponds to the incremental version of the OLL algorithm (using the SAT solver in incremental mode), and it also uses the iterative version of the Totalizer encoding [6] as cardinality constraint encoding. Similarly to the *weighted configuration* of *MSCG*, Glucose 3.0 (in incremental mode) is used as the underlying SAT solver. Additionally, it computes lower and upper bounds, trims unsatisfiable cores at most 5 times, and uses the complete BMO algorithm whenever possible (clearly, the latter can be used only for some of the weighted formulas).

Figure 2 presents the results obtained for each of the industrial categories. Both versions of **MSCG** clearly improve over the best performing solvers of MaxSAT Evaluation 2013. It can also be seen that in each category the competition version (*MSCG*) is close to the winning solvers of the MaxSAT Evaluation 2014 while *MSCG** is able to outperform the winning solver of 2014 for both plain MaxSAT and for partial MaxSAT benchmarks. It should be noted that *MSCG** is always able to perform the same or better than *MSCG*. Overall the results show that **MSCG** is one of the most robust MaxSAT solvers.

4. Conclusions

This work describes the **MSCG** solver, which comprises different core-guided algorithms as well as some of the state-of-the-art MaxSAT techniques. In this work, we additionally presented two versions of **MSCG**: the competition version *MSCG* submitted to the MaxSAT Evaluation 2014 and *MSCG**, which is currently the best performing configuration of **MSCG**. The experimental results indicate that both versions improve over the best performing solvers for the industrial benchmark categories of the MaxSAT Evaluation 2013, and are comparable with the best performing solvers of the MaxSAT Evaluation 2014. *MSCG** is even able to outperform the best performing solvers in two categories. Overall, it can be seen as one of the most robust state-of-the-art MaxSAT solvers.

Acknowledgments

This work is partially supported by SFI PI grant BEACON (09/IN.1/I2618), FCT grant POLARIS (PTDC/EIA-CCO/123051/2010) and national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC.50021/2013.

References

[1] C. Ansotegui, M. L. Bonet, and J. Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, **196**:77–105, 2013.

[2] O. Bailleux and Y. Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *Principles and Practice of Constraint Programming*, **2833** of *LNCS*, pages 108–122, 2003.

[3] Z. Fu and S. Malik. On solving the partial MaxSAT problem. In *Theory and Applications of Satisfiability Testing*, **4121** of *LNCS*, pages 252–265. 2006.

[4] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa. QMaxSAT: A partial MaxSAT solver. *J. Satisfiability, Boolean Modeling and Computation*, **8**(1/2):95–100, 2012.

[5] J. Marques-Silva, J. Argelich, A. Graca, and I. Lynce. Boolean lexicographic optimization: algorithms & applications. *Annals of Mathematics and Artificial Intelligence (AMAI)*, **62**(3-4):317–343, 2011.

[6] R. Martins, S. Joshi, V. Manquinho, and I. Lynce. Incremental cardinality constraints for MaxSAT. In *Principles and Practice of Constraint Programming*, **8656** of *LNCS*, pages 531–548. 2014.

[7] R. Martins, V. Manquinho, and I. Lynce. Open-WBO: A modular MaxSAT solver. In *Theory and Applications of Satisfiability Testing*, **8561** of *LNCS*, pages 438–445. 2014.

[8] A. Morgado, C. Dodaro, and J. Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *Principles and Practice of Constraint Programming*, **8656** of *LNCS*, pages 564–573. 2014.

[9] A. Morgado, A. Ignatiev, and J. Marques-Silva. MSCG: Robust core-guided MaxSAT solving (extended system description). Technical Report 18/2015, INESC-ID, 2015.

[10] N. Narodytska and F. Bacchus. Maximum Satisfiability using core-guided MaxSAT resolution. In *AAAI Conference on Artificial Intelligence*, pages 2717–2723, 2014.

[11] T. Ogawa, Y. Liu, R. Hasegawa, M. Koshimura, and H. Fujita. Modulo based CNF encoding of cardinality constraints and its application to MaxSAT solvers. In *Tools with Artificial Intelligence*, pages 9–17, 2013.

[12] S. D. Prestwich. Variable dependency in local search: Prevention is better than cure. In *Theory and Applications of Satisfiability Testing*, **4501** of *LNCS*, pages 107–120, 2007.