

Open-WBO @ MaxSAT Evaluation 2020

Ruben Martins
rubenm@cs.cmu.edu
CMU, USA

Norbert Manthey
nmanthey@conp-solutions.com
Dresden, Germany

Miguel Terra-Neves, Vasco Manquinho, Inês Lynce
{neves,vmm,ines}@inesc-id.pt
INESC-ID/IST, Portugal

I. INTRODUCTION

OPEN-WBO [1] is an open source MaxSAT solver that supports several MaxSAT algorithms [2], [3], [4], [5], [6], [7], [8] and SAT solvers [9], [10], [11]. OPEN-WBO is particularly efficient for unweighted MaxSAT and has been one of the best solvers in the MaxSAT Evaluations from 2014 to 2017. Five versions of OPEN-WBO were submitted to different tracks at MaxSAT Evaluation 2020: `open-wbo-res-mergesat-v1`, `open-wbo-res-mergesat-v2`, `open-wbo-res-glucose-v1`, `open-wbo-res-glucose-v2`, and `open-wbo-topk`. The remainder of this document describes the differences between these versions.

II. SAT SOLVERS

OPEN-WBO is based on the data structures of MINISAT 2.2 [9], [12]. Therefore, solvers based on MINISAT 2.2 can be used as a potential back-end solver. For the MaxSAT Evaluation 2019, we use GLUCOSE 4.1 [10], [13], [14] as the back-end SAT solver of the versions that end in `glucose` and MERGESAT [11] as the back-end SAT solver of the versions that end in `mergesat`.

MERGESAT is a new CDCL solver developed by Norbert Manthey and it is based on the SAT competition winner of 2018, MAPLELCMDISTCHRONOBT [15], and adds several known techniques. For restarts, only partial backtracking is used, learned clause minimization is implemented more efficiently, and also applies simplification again in case the first swipe resulted in a simplification. Finally, the time-based decision heuristic switch is made deterministic by using solving steps. Furthermore, subsumption and self-subsuming resolution is used as inprocessing. To support being used inside MaxSAT solvers, the incremental search feature had to be enabled again.

III. MAXSAT ALGORITHMS

In this section, we briefly describe the algorithms used for the complete and top- k tracks at the MSE2020.

A. Complete Unweighted Track

Four versions were submitted to the complete unweighted track: `open-wbo-res-mergesat-v1`, `open-wbo-res-mergesat-v2`, `open-wbo-res-glucose-v1`, `open-wbo-res-glucose-v2`.

All versions use a variant of the unsatisfiability-based algorithm MSU3 [3] and the OLL algorithm [7]. This algorithm

works by iteratively refining a lower bound λ on the number of unsatisfied soft clauses until an optimum solution is found. We use an incremental version of this algorithm by taking advantage of the incremental version of the Totalizer encoding [4]. We also extended the incremental MSU3 algorithm [4] with resolution-based partitioning techniques [8]. We represent a MaxSAT formula using a resolution-based graph representation and iteratively join partitions by using a proximity measure extracted from the graph representation of the formula. The algorithm ends when only one partition remains and the optimal solution is found. Since the partitioning of some MaxSAT formulas may be unfeasible or not significant, we heuristically choose to run either MSU3 with partitions or without partitions. In particular, we do not use partition-based techniques when one of the following criteria is met: (i) the formula is too large ($> 1,000,000$ clauses), (ii) the ratio between the number of partitions and soft clauses is too high (> 0.8), (iii) the sparsity of the graph is too small (< 0.04), or (iv) there exist some at-most-one relations between soft clauses (> 10), i.e. if one soft clause is satisfied it implies that some other soft clauses will be unsatisfied.

The difference between versions `v1` and `v2` is that version `v1` uses the MSU3 algorithm when the partitioning algorithm is not applicable, whereas `v2` uses the OLL algorithm.

B. Top- k Track

For the top- k track, we use a linear search SAT-UNSAT to find the optimal solution. Once the optimal solution is found, we change to the (W)MSU3 algorithm and enumerate solutions until we exhaust the search space or we find k solutions. Each algorithm is incremental but the swap is made in a non-incremental fashion. For the unweighted track, we use the incremental Totalizer encoding for MSU3 [4] and the incremental SWC encoding for WMSU3 [16].

These algorithms are not optimized for the top- k track and should be consider a baseline for future improvements.

C. Preprocessing

We perform identification of unit cores and at-most-one relations between soft clauses by using unit propagation. A similar technique is done in RC2 [17], the winner of the MaxSAT Evaluation 2018.

D. Other

OPEN-WBO now supports printing the certificate in a compact mode using 0's and 1's.

IV. AVAILABILITY

The latest release of OPEN-WBO is available under a MIT license in GitHub at <https://github.com/sat-group/open-wbo>.

ACKNOWLEDGMENTS

We would like to thank Laurent Simon and Gilles Audemard for allowing us to use GLUCOSE 4.1 in the MaxSAT Evaluation. We would also like to thank Niklas Eén and Niklas Sörensson for the development of MINISAT 2.2. Additionally, we would like to thank all the collaborators on previous versions of OPEN-WBO, namely Saurabh Joshi and Mikoláš Janota. Finally, we would like to thank David Chen for his study on the impact of disjoint cores, unit cores, and at-most-one relations between soft clauses that were done in the scope of Independent Studies at CMU.

REFERENCES

- [1] R. Martins, V. Manquinho, and I. Lynce, “Open-WBO: a Modular MaxSAT Solver,” in *SAT*, ser. LNCS, vol. 8561. Springer, 2014, pp. 438–445.
- [2] V. Manquinho, J. Marques-Silva, and J. Planes, “Algorithms for Weighted Boolean Optimization,” in *SAT*. Springer, 2009, pp. 495–508.
- [3] J. Marques-Silva and J. Planes, “On Using Unsatisfiability for Solving Maximum Satisfiability,” *CoRR*, 2007.
- [4] R. Martins, S. Joshi, V. Manquinho, and I. Lynce, “Incremental Cardinality Constraints for MaxSAT,” in *CP*. Springer, 2014, pp. 531–548.
- [5] R. Martins, V. Manquinho, and I. Lynce, “On Partitioning for Maximum Satisfiability,” in *ECAI*. IOS Press, 2012, pp. 913–914.
- [6] R. Martins, V. M. Manquinho, and I. Lynce, “Community-based partitioning for maxsat solving,” in *SAT*. Springer, 2013, pp. 182–191.
- [7] A. Morgado, C. Dodaro, and J. Marques-Silva, “Core-Guided MaxSAT with Soft Cardinality Constraints,” in *CP*. Springer, 2014, pp. 564–573.
- [8] M. Neves, R. Martins, M. Janota, I. Lynce, and V. M. Manquinho, “Exploiting Resolution-Based Representations for MaxSAT Solving,” in *SAT*. Springer, 2015, pp. 272–286.
- [9] N. Eén and N. Sörensson, “An Extensible SAT-solver,” in *SAT*. Springer, 2003, pp. 502–518.
- [10] G. Audemard and L. Simon, “Predicting Learnt Clauses Quality in Modern SAT Solvers,” in *IJCAI*, 2009, pp. 399–404.
- [11] N. Manthey, “Mergesat,” in *Proceedings of SAT Competition 2019: Solver and Benchmark Descriptions*, 2019.
- [12] N. Sörensson, N. Een, and N. Manthey. (2018, May) GitHub repository for MiniSat. <https://github.com/conp-solutions/minisat>.
- [13] G. Audemard, J.-M. Lagniez, and L. Simon, “Improving glucose for incremental sat solving with assumptions: Application to mus extraction,” in *SAT*. Springer, 2013.
- [14] G. Audemard and L. Simon. (2018, May) Glucose’s home page. <http://www.labri.fr/perso/lsimon/glucose>.
- [15] A. Nadel and V. Ryvchin, “Chronological backtracking,” in *SAT*. Springer, 2018, pp. 111–121.
- [16] R. Martins, S. Joshi, V. M. Manquinho, and I. Lynce, “On using incremental encodings in unsatisfiability-based maxsat solving,” *JSAT*, vol. 9, no. 1, pp. 59–81, 2014.
- [17] A. Ignatiev, A. Morgado, and J. Marques-Silva, “PySAT: A Python Toolkit for Prototyping with SAT Oracles,” in *Proc. SAT*, ser. Lecture Notes in Computer Science, O. Beyersdorff and C. M. Wintersteiger, Eds., vol. 10929. Springer, 2018, pp. 428–437.