Open-WBO: a Modular MaxSAT Solver[⋆]

Ruben Martins[1], Vasco Manquinho[2], and Inês Lynce[2]

[1]University of Oxford, Department of Computer Science, United Kingdom
ruben.martins@cs.ox.ac.uk
[2]INESC-ID / Instituto Superior Técnico, Universidade de Lisboa, Portugal
{vmm,ines}@sat.inesc-id.pt

Abstract. This paper presents OPEN-WBO, a new MaxSAT solver. OPEN-WBO has two main features. First, it is an open-source solver that can be easily modified and extended. Most MaxSAT solvers are not available in open-source, making it hard to extend and improve current MaxSAT algorithms. Second, OPEN-WBO may use any MiniSAT-like solver as the underlying SAT solver. As many other MaxSAT solvers, OPEN-WBO relies on successive calls to a SAT solver. Even though new techniques are proposed for SAT solvers every year, for many MaxSAT solvers it is hard to change the underlying SAT solver. With OPEN-WBO, advances in SAT technology will result in a free improvement in the performance of the solver. In addition, the paper uses OPEN-WBO to evaluate the impact of using different SAT solvers in the performance of MaxSAT algorithms.

1 Introduction

Maximum Satisfiability (MaxSAT) formulations are currently used for solving many different real-world problems [20,1,16]. This results from the recent improvements of MaxSAT algorithms, which are now able to solve much larger instances than before. These improvements result mainly from (i) an increased performance of the underlying SAT solver and (ii) novel techniques and algorithms proposed for MaxSAT solving.

Currently, MaxSAT solvers more suited for industrial instances may follow different algorithmic approaches, although the common feature is that MaxSAT algorithms rely on successive calls to a SAT solver. In this paper, we present OPEN-WBO, an open-source modular MaxSAT solver that enables an easy replacement of the underlying SAT solver for any MiniSAT-like solver. The OPEN-WBO architecture also allows an easy implementation and extension of different MaxSAT algorithms. Another contribution of the paper is an evaluation of different state of the art SAT solvers in solving MaxSAT. We provide the results of OPEN-WBO for linear search and unsatisfiability-based algorithm when using different underlying SAT solvers.

In what follows we assume that the reader is familiar with MaxSAT [21,26].

[⋆] Partially supported by the ERC project 280053, and FCT grants ASPEN (PTDC/EIA-CCO/110921/2009), POLARIS (PTDC/EIA-CCO/123051/2010), and INESC-ID's multiannual PIDDAC funding PEst-OE/EEI/LA0021/2013.

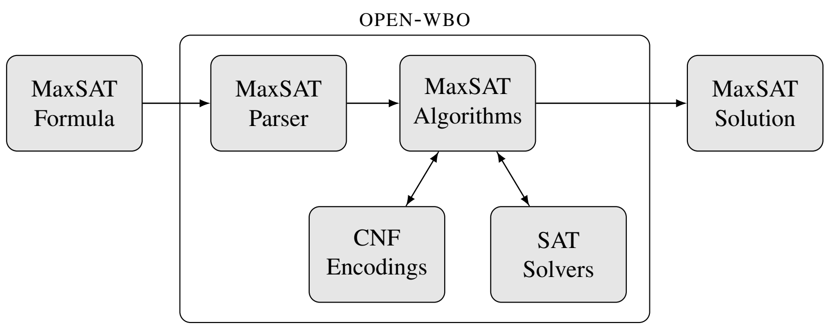

Fig. 1: Overview of the architecture of OPEN-WBO.

2 Solver Description

OPEN-WBO is implemented in C++ and extends the interface of MINISAT [12] for solving MaxSAT formulas. OPEN-WBO is open-source and is available for download at `http://sat.inesc-id.pt/open-wbo`. Figure 1 provides an overview of the OPEN-WBO architecture. The main components are the following: (i) MaxSAT Parser, (ii) MaxSAT Algorithms, (iii) CNF Encodings, and (iv) SAT Solvers. Each of the components of OPEN-WBO is briefly described in this section.

MaxSAT Parser. OPEN-WBO can take as input any MaxSAT formula. The parser reads the MaxSAT formula and stores the hard and soft clauses into two different data structures. Each hard and soft clauses may have private fields that describe properties of those clauses. Currently, to each soft clause we associate an integer weight, a set of relaxation variables, and a selector variable. (In the case of unweighted MaxSAT formulas, weight 1 is associated with each soft clause.) The integer weight is used to store the cost of unsatisfying the soft clause. The set of relaxation variables keeps the fresh variables that have been added to soft clauses during the relaxation procedure of the MaxSAT algorithm. Selector variables are fresh variables that are used to extract an unsatisfiable subformula [5]. For that we first associate each selector variable to the corresponding soft clause. Next, the SAT solver is called with a set of assumptions, where the assumptions correspond to the negation of the selector variables. Assumptions are always picked as the first decision literals by the SAT solver. If the SAT solver returns unsatisfiable, then the solver is able to infer a conflict clause that only contains assumptions. As a result, the subset of soft clauses that contains those assumption variables corresponds to the soft clauses in an unsatisfiable subformula. (Note that unsatisfiability-based algorithms do not need to relax hard clauses. Therefore, we only need to extract the soft clauses from the unsatisfiable subformula.)

If a MaxSAT algorithm requires an additional property of the hard or soft clauses, this can be easily added to the respective data structure. Hence, the proposed architecture allows the working formula to be easily rebuilt at each iteration of the MaxSAT algorithm.

MaxSAT Algorithms. The current implementation of OPEN-WBO has two orthogonal MaxSAT algorithms: (i) unsatisfiability-based algorithm, and (ii) linear search algorithm.

The unsatisfiability-based algorithm is based on the iterated use of a SAT solver to identify unsatisfiable subformulas [13,3,23]. At each MaxSAT iteration, the working formula is relaxed until the formula becomes satisfiable. Therefore, all calls to the SAT solver return unsatisfiable, except for the last one which returns satisfiable. The basic algorithm is improved in OPEN-WBO by considering techniques that have been recently proposed. Symmetry breaking predicates are now added to the formula to break symmetries that arise from having multiple relaxation variables [2]. For weighted partial MaxSAT instances, this algorithm is further improved by considering the weight-based partitioning scheme [25] or the diversity-based heuristic [2].

The linear search algorithm [19,17] starts by adding a new relaxation variable to each soft clause and solving the resulting formula with a SAT solver. Whenever a model is found, a new constraint on the relaxation variables is added such that models with a greater or equal value are excluded. The algorithm terminates when the SAT solver returns unsatisfiable. Therefore, all calls to the SAT solver return satisfiable, except for the last one which returns unsatisfiable. The basic algorithm is improved in OPEN-WBO by considering lexicographic optimization for weighted partial MaxSAT instances where the optimality criterion is lexicographical [24].

CNF Encodings. Most MaxSAT algorithms require the encoding of constraints that are not originally expressed in CNF, such as: (i) at-most-one constraints ($x_1 + \ldots + x_n \leq 1$), (ii) cardinality constraints ($x_1 + \ldots + x_n \leq k$), and (iii) pseudo-Boolean constraints ($a_1 x_1 + \ldots + a_n x_n \leq k$). OPEN-WBO uses the following encodings for these different constraints:

- *Ladder* encoding [4,14] (at-most-one constraints): for a constraint with n literals, this encoding creates $O(n)$ clauses with $O(n)$ auxiliary variables.
- *Cardinality Networks* encoding [6] (cardinality constraints): for a constraint with n literals and with right-hand side k, this encoding creates $O(nlog^2k)$ clauses with $O(nlog^2k)$ auxiliary variables.
- *Sequential* encoding [15] (pseudo-Boolean constraints): for a constraint with n literals and with right-hand side k, this encoding creates $O(nk)$ clauses with $O(nk)$ auxiliary variables.

These encodings are compact and maintain generalized arc consistency by unit propagation. For the cardinality and the pseudo-Boolean constraints, the user as a programmer has two options: (i) to encode the constraint into CNF or (ii) to update the right-hand side value of the encoding. If the encoding was already built and the MaxSAT algorithm found a smaller right-hand side value, then the programmer may update the encoding instead of rebuilding it [6]. In this case, all learned clauses from the previous SAT call may be kept in the next call to the SAT solver. With these encodings, a programmer may implement MaxSAT algorithms that are based on successive calls to a SAT solver.

Table 1: Number of instances solved by different SAT solvers in the SAT Competition 2013 in the Application SAT+UNSAT track (out of $150 + 150 = 300$ instances)

	#SAT	#UNSAT	Total
ZENN	113	95	208
SINN	120	86	206
glue_bit	102	102	204
Glucose 2.3	103	98	201
GluH	99	97	196
Glueminisat	100	96	196
Glucored	93	95	188

SAT Solvers. OPEN-WBO can use any MiniSAT-like SAT solver, i.e. any SAT solver that extends and uses the same interface as MINISAT [12]. When compiling OPEN-WBO, the user may choose which MiniSAT-like solver is going to be used. Currently, OPEN-WBO includes the following SAT solvers: (i) `MiniSAT2.0` [12], (ii) `MiniSAT2.2` [12], (iii) `Glucose2.3` [8,9], (iv) `Glucose3.0` [8,7], (v) `ZENN` [33], (vi) `SINN` [32], (vii) `Glueminisat` [27], (viii) `GluH` [29], (ix) `glue_bit` [11], and (x) `GlucoRed` [31]. Note that `Glucose3.0` was modified to support the optimizations on assumptions [7] for MaxSAT. `MiniSAT2.2` extends `MiniSAT2.0` by using Luby restarts [22] and phase saving [30]. The other solvers differ mainly in the strategy employed for deleting learned clauses and for restarting. These strategies are usually based on the Literal Block Distance (LBD) measure [8] or in variations of LBD. For details on the SAT solvers see the proceedings of the SAT Competition 2013 [10].

Table 1 shows the number of instances solved by the different SAT solvers that are being used in OPEN-WBO and also participated in the Application SAT+UNSAT track of the SAT Competition 2013. For a better understanding of the performance of the solvers, we split the number of solved instances into unsatisfiable (#UNSAT) and satisfiable (#SAT). Note that `MiniSAT2.0`, `MiniSAT2.2`, and `Glucose3.0` are not included in Table 1 since they did not participate in the SAT Competition 2013.

Overall, `ZENN` was the best performing MiniSAT-like solver in the SAT Competition 2013. The best solver for satisfiable instances was `SINN`, whereas the best solver for unsatisfiable instances was `glue_bit`. The difference between the number of instances solved by MiniSAT-like solvers is small. For example, `ZENN` only solved 12 more instances than `Glueminisat`. On the other hand, `GlucoRed` was the solver with the worst performance. `GlucoRed` uses two concurrent threads, which are called the solver and the reducer. The solver uses the SAT solver as usual, while the reducer attempts to strengthen the clauses derived by the solver. In practice, `GlucoRed` only has half of the CPU time for the SAT solver since the SAT Competition 2013 enforces a CPU time limit. Hence, it is expected to solve less instances. Nevertheless, we included this solver in our tool since it may solve instances that are not solved by other solvers.

Note that OPEN-WBO is not restricted to these ten solvers and so any MiniSAT-like solver can be easily plugged in. This allows OPEN-WBO to take advantage of the constant improvement in SAT solver technology.

Table 2: Impact of different SAT solvers in MaxSAT algorithms

(a) Unsat-based algorithm

	ms	pms	wpms	Total
VBS	42	381	331	754
Glucose 3.0	41	365	313	719
Glucose 2.3	40	343	315	698
GluH	40	341	315	696
ZENN	40	341	315	696
Minisat 2.2	41	340	313	694
SINN	41	335	318	694
Glueminisat	39	329	315	683
Minisat 2.0	38	330	310	678
GlucoRed	38	293	295	626
glue_bit	40	286	294	620

(b) Linear search algorithm

	ms	pms	wpms	Total
VBS	10	535	246	791
Glucose 2.3	5	525	242	772
Glucose 3.0	6	521	242	769
ZENN	6	511	242	759
Glueminisat	6	509	238	753
glue_bit	7	510	232	749
GluH	5	505	235	745
GlucoRed	3	509	225	737
SINN	7	484	239	730
Minisat 2.2	7	468	235	710
Minisat 2.0	4	454	233	691

3 Experimental Results

All experiments were run on the unweighted MaxSAT (`ms`, 55 instances), partial MaxSAT (`pms`, 697 instances) and weighted partial MaxSAT (`wpms`, 396 instances) instances from the industrial category of the MaxSAT evaluation of 2013. The evaluation was performed on two AMD Opteron 6276 processors (2.3 GHz) running Fedora 18 with a timeout of 1,800 CPU seconds and a memory limit of 16 GB.

Impact of Different SAT Solvers. Table 2 compares the performance of MaxSAT algorithms when using different SAT solvers. On the left, we present the impact of different SAT solvers in the unsatisfiability-based algorithm, and on the right, the impact of different SAT solvers in the linear search algorithm. For each algorithm, we grouped SAT solvers that performed similarly. The table also includes the Virtual Best Solver (VBS), i.e. the number of instances that were solved by at least one of the SAT solvers.

The performance of SAT solvers in MaxSAT algorithms is very different from the ranking of the SAT solvers at the SAT Competition 2013. This is particularly noticeable for the unsatisfiability-based algorithm. Most SAT solvers have a performance similar to `MiniSAT2.2`, which is the baseline solver for all SAT solvers reported in this paper. Therefore, the remaining solvers are expected to improve the performance of `MiniSAT 2.2`. However, this is not the case for MaxSAT when using the unsatisfiability-based algorithm. It has been observed that the LBD measure is affected when using assumptions [7]. Since each assumption has its own decision level, the LBD measure will be similar to the clause size when learning clauses that contain a large number of assumptions. With the exception of MINISAT, all other solvers use the LBD measure. This may explain why the performance of most SAT solvers is similar to `MiniSAT2.2`. Furthermore, assumptions may also affect other heuristics. For example, `glue_bit` was the best performing MiniSAT-like solver for unsatisfiable instances in the SAT Competition

2013, but is one of the worst solvers when using the unsatisfiability-based algorithm, since it uses a restart strategy based on the depth of the search which is greatly affected by the large number of assumptions. On the other hand, `Glucose3.0` was the best performing solver for the unsatisfiability-based algorithm since it is the only solver with optimizations when using assumptions [7]. The VBS solves 35 more instances than any individual solver.

For the linear search algorithm, all SAT solvers outperform MINISAT. The linear search algorithm does not use assumptions on the SAT calls. Hence, the heuristics of SAT solvers are not affected. `SINN` was the best performing MiniSAT-like solver for satisfiable instances in the SAT Competition 2013, but it is one of the worst solvers in Table 2b. This is due to the formula in the last call of the linear search algorithm being unsatisfiable. Even though `SINN` performs well for satisfiable instances, it does poorly on unsatisfiable instances. `Glucose2.3` was the best solver for the linear search algorithm with results similar to `Glucose3.0`. The VBS solves only 19 more instances than any individual solver. It seems that the performance of the linear search algorithm does not depend much on the SAT solver performance.

State-of-the-art MaxSAT Solvers. We have compared the performance of OPEN-WBO (with `Glucose3.0`) against QMAXSAT [17] and WPM1 [3] (MaxSAT Evaluation 2013 versions) in order to show that OPEN-WBO is competitive.

QMAXSAT uses a linear search algorithm similar to the one implemented in OPEN-WBO and is particularly effective for solving industrial `pms` instances. QMAXSAT solved 641 instances (6 `ms`, 545 `pms`, 90 `wpms`). When compared to the OPEN-WBO linear search algorithm, QMAXSAT solved the same number of `ms` instances, 24 more `pms` instances, and 152 less `wpms` instances. The performance gains on the `pms` instances are mostly due to a new cardinality encoding that has been recently proposed [28]. On the other hand, QMAXSAT is much less efficient than OPEN-WBO when solving`wpms` instances for not using lexicographic optimization.

WPM1 uses an unsatisfiability-based algorithm similar to the one implemented in OPEN-WBO and is particularly effective for solving weighted partial MaxSAT instances. WPM1 solved 772 instances (22 `ms`, 405 `pms`, 345 `wpms`). When compared to OPEN-WBO, WPM1 solved 19 less `ms` instances, 40 more `pms` instances, and 32 more `wpms` instances. The current version of WPM1 uses a SMT solver instead of a SAT solver, and is able to keep some learned clauses between iterations of the MaxSAT algorithm. This may explain the performance gains of WPM1 when compared to OPEN-WBO for the `pms` and `wpms` instances. On the other hand, OPEN-WBO is more efficient for solving `ms` instances. These instances have a very large number of soft clauses, which results in a considerable overhead to the current version of WPM1.

4 Conclusions

In this paper we presented OPEN-WBO, an extensible and modular open-source MaxSAT solver. OPEN-WBO implements state of the art linear search and unsatisfiability-based algorithms and can use any MiniSAT-like SAT solver. An experimental evaluation was conducted to show the performance of OPEN-WBO when using different SAT solvers.

References

1. Achá, R.A., Nieuwenhuis, R.: Curriculum-based course timetabling with SAT and MaxSAT. Annals of Operations Research pp. 1–21 (2012)
2. Ansótegui, C., Bonet, M.L., Gabàs, J., Levy, J.: Improving SAT-Based Weighted MaxSAT Solvers. In: Milano, M. (ed.) International Conference on Principles and Practice of Constraint Programming. LNCS, vol. 7514, pp. 86–101. Springer (2012)
3. Ansótegui, C., Bonet, M.L., Levy, J.: Solving (Weighted) Partial MaxSAT through Satisfiability Testing. In: Kullmann [18], pp. 427–440
4. Ansótegui, C., Manyà, F.: Mapping Problems with Finite-Domain Variables into Problems with Boolean Variables. In: International Conference on Theory and Applications of Satisfiability Testing (2004)
5. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Practical algorithms for unsatisfiability proof and core generation in SAT solvers. AI Communications 23(2-3), 145–157 (2010)
6. Asín, R., Nieuwenhuis, R., Oliveras, A., Rodríguez-Carbonell, E.: Cardinality Networks: a theoretical and empirical study. Constraints 16(2), 195–221 (2011)
7. Audemard, G., Lagniez, J.M., Simon, L.: Improving Glucose for Incremental SAT Solving with Assumptions: Application to MUS Extraction. In: Järvisalo, M., Gelder, A.V. (eds.) International Conference on Theory and Applications of Satisfiability Testing. LNCS, vol. 7962, pp. 309–317. Springer (2013)
8. Audemard, G., Simon, L.: Predicting Learnt Clauses Quality in Modern SAT Solvers. In: Boutilier, C. (ed.) International Joint Conference on Artificial Intelligence. pp. 399–404 (2009)
9. Audemard, G., Simon, L.: Glucose 2.3 in the SAT 2013 Competition. In: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions [10], pp. 42–43
10. Balint, A., Belov, A., Heule, M., Järvisalo, M.: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions. Tech. rep., Department of Computer Science Series of Publications B, vol. B-2013-1, University of Helsinki, Helsinki (2013)
11. Chen, J.: Solvers with a Bit-Encoding Phase Selection Policy and a Decision-Depth-Sensitive Restart Policy. In: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions [10], pp. 44–45
12. Eén, N., Sörensson, N.: An Extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) International Conference on Theory and Applications of Satisfiability Testing. LNCS, vol. 2919, pp. 502–518. Springer (2003)
13. Fu, Z., Malik, S.: On Solving the Partial MAX-SAT Problem. In: Biere, A., Gomes, C.P. (eds.) International Conference on Theory and Applications of Satisfiability Testing. LNCS, vol. 4121, pp. 252–265. Springer (2006)
14. Gent, I.P., Nightingale, P.: A new encoding of All Different into SAT. In: International Workshop on Modelling and Reformulating Constraint Satisfaction Problems (2004)
15. Hölldobler, S., Manthey, N., Steinke, P.: A Compact Encoding of Pseudo-Boolean Constraints into SAT. In: Glimm, B., Krüger, A. (eds.) KI 2013: Advances in Artificial Intelligence. LNCS, vol. 7526, pp. 107–118. Springer (2012)
16. Janota, M., Lynce, I., Manquinho, V., Marques-Silva, J.: PackUp: Tools for Package Upgradability Solving. Journal on Satisfiability, Boolean Modeling and Computation 8(1/2), 89–94 (2012)
17. Koshimura, M., Zhang, T., Fujita, H., Hasegawa, R.: QMaxSAT: A Partial Max-SAT Solver. Journal on Satisfiability, Boolean Modeling and Computation 8, 95–100 (2012)
18. Kullmann, O. (ed.): International Conference on Theory and Applications of Satisfiability Testing, LNCS, vol. 5584. Springer (2009)

19. Le Berre, D., Parrain, A.: The Sat4j library, release 2.2. Journal on Satisfiability, Boolean Modeling and Computation 7(2-3), 59–6 (2010)
20. Le Berre, D., Rapicault, P.: Dependency Management for the Eclipse Ecosystem: Eclipse P2, Metadata and Resolution. In: International Workshop on Open Component Ecosystems. pp. 21–30. ACM (2009)
21. Li, C.M., Manyà, F.: MaxSAT, Hard and Soft Constraints. In: Handbook of Satisfiability, pp. 613–631. IOS Press (2009)
22. Luby, M., Sinclair, A., Zuckerman, D.: Optimal Speedup of Las Vegas Algorithms. Information Processing Letters 47(4), 173–180 (1993)
23. Manquinho, V., Marques-Silva, J., Planes, J.: Algorithms for Weighted Boolean Optimization. In: Kullmann [18], pp. 495–508
24. Marques-Silva, J., Argelich, J., Graça, A., Lynce, I.: Boolean lexicographic optimization: algorithms & applications. Annals of Mathematics and Artificial Intelligence 62(3-4), 317–343 (2011)
25. Martins, R., Manquinho, V., Lynce, I.: On Partitioning for Maximum Satisfiability. In: Raedt, L.D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., Lucas, P.J.F. (eds.) European Conference on Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 242, pp. 913–914. IOS Press (2012)
26. Morgado, A., Heras, F., Liffiton, M., Planes, J., Marques-Silva, J.: Iterative and core-guided MaxSAT solving: A survey and assessment. Constraints 18(4), 478–534 (2013)
27. Nabeshima, H., Iwanuma, K., Inoue, K.: GLUEMINISAT2.2.7. In: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions [10], pp. 46–47
28. Ogawa, T., Liu, Y., Hasegawa, R., Koshimura, M., Fujita, H.: Modulo Based CNF Encoding of Cardinality Constraints and Its Application to MaxSAT Solvers. In: International Conference on Tools with Artificial Intelligence. pp. 9–17. IEEE (2013)
29. Oh, C.: gluH: Modified Version of glucose 2.1. In: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions [10], p. 48
30. Pipatsrisawat, K., Darwiche, A.: A Lightweight Component Caching Scheme for Satisfiability Solvers. In: Marques-Silva, J., Sakallah, K.A. (eds.) International Conference on Theory and Applications of Satisfiability Testing. LNCS, vol. 4501, pp. 294–299. Springer (2007)
31. Wieringa, S.: GlucoRed. In: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions [10], pp. 40–41
32. Yasumoto, T., Okugawa, T.: SINNminisat. In: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions [10], p. 85
33. Yasumoto, T., Okugawa, T.: ZENN. In: Proceedings of SAT Competition 2013 : Solver and Benchmark Descriptions [10], p. 95