Journal on Satisfiability, Boolean Modeling and Computation 15 (2024) 1–7

TT-Open-WBO-Inc: An Efficient Anytime MaxSAT Solver

Alexander Nadel alexander.nadel@cs.tau.ac.il
The Faculty of Data and Decision Sciences
Technion – Israel Institute of Technology
Haifa, 3200003
Israel
Intel Corporation
Haifa, 3508409
Israel

Abstract

We present our anytime MaxSAT solver TT-Open-WBO-Inc, focusing on its evolution since the initial version that won both of the weighted incomplete tracks of MaxSAT Evaluation 2019 (MSE19). The solver's MSE20 version claimed victory in these tracks at MSE20 and secured second place in both unweighted incomplete tracks. The major innovation in the MSE20 version was the integration of SAT-based local search. The contributions of this paper include: (1) Introducing a previously unpublished variant of the SAT-based local search algorithm Polosat, Polosat-OBV, applied by default already in the MSE20 version, and showing its superiority for weighted solving; (2) Describing and analyzing TT-Open-WBO-Inc's unweighted component, not studied in previous work; (3) Demonstrating that integrating the local search algorithm SATLike into TT-Open-WBO-Inc as a preprocessor enables it to outperform the winners of MSE20 in all four incomplete tracks.

Keywords: *MaxSAT, anytime MaxSAT, Mrs. Beaver, Polosat, SAT-based Local Search*

Submitted 01 November 2020; revised 27 March 2024; accepted 20 June 2024

1. Introduction

MaxSAT is a widely used extension of SAT to optimizing a linear Pseudo–Boolean (PB) function. Given a set of *hard* propositional clauses H and a *target bit-vector (target)* $T = \{t_n, t_{n-1}, \ldots, t_1\}$, where each *target bit* t_i is a Boolean variable associated with a strictly positive integer weight w_i, MaxSAT finds a model σ to H that minimizes the objective function $\Psi(\sigma) = \sum_{i=1}^{n} \sigma(t_i) \times w_i$. A MaxSAT instance is *unweighted* iff all the weights are 1; otherwise it is *weighted*. *Anytime* MaxSAT solvers, evaluated at MaxSAT Evaluations (MSEs) since 2011 in the so-called incomplete tracks, find an *improving* set of models $\{\mu_1, \mu_2, \ldots, \mu_n\}$ over time, that is, for every $j > i$, they have $\Psi(\mu_j) < \Psi(\mu_i)$.

This paper presents the latest version of our anytime MaxSAT solver TT-Open-WBO-Inc. Our solver's initial release, described in [6], was spawned from the MSE18 version of Open-WBO-Inc [3]. That release won both of the weighted incomplete tracks of MSE19. The major innovations in the next MSE20 version included the integration of the SAT-based local search [8] and the enablement of unweighted solving. The current paper covers:

1) a previously unpublished variant of the SAT-based local search algorithm Polosat [8], Polosat-OBV, applied by default already at MSE20. Section 3 demonstrates Polosat-OBV's superior performance over Polosat for weighted solving, thus

Table 1. Weighted component evolvement in `TT-Open-WBO-Inc` since `Open-WBO-Inc`

MSE	Algorithm	Functionality	Incorporation	Ref.
–	`SATLike`	Classical local search	Preprocess for 15 sec.	[2]
2020	`Polosat-OBV`	SAT-based local search	Replace SAT queries	New
2019	`TORC` & `TSB`	SAT heuristics	Change default heuristics	[7]
2018	`BMO`	SAT-based MaxSAT flow	Baseline (`Open-WBO-Inc`)	[3]

`Polosat-OBV` contributed to the solver winning both weighted incomplete tracks at MSE20.

2) `TT-Open-WBO-Inc`'s unweighted component's architecture, previously undescribed. The solver came in second in both unweighted incomplete tracks at MSE20.
3) showing that by incorporating the local search solver `SATLike` [2] for preprocessing, `TT-Open-WBO-Inc` surpasses the winners of MSE20 in all four incomplete tracks.

The baseline algorithm for both weighted and unweighted solving in `Open-WBO-Inc`, inherited by `TT-Open-WBO-Inc`, is Linear Search SAT-UNSAT (`LSU`) [4]. `LSU` finds the first model μ_1 by applying an incremental SAT solver over the hard clauses only. Then, it blocks all the solutions of weight $\geqslant \Psi(\mu_1)$ using PB constraints (in the weighted case) or cardinality constraints (in the unweighted case) and continues the process of finding and blocking new improving models iteratively. `LSU` stops when the solver returns UNSAT, in which case the last model is an optimal one. A known drawback of `LSU`, which slows it down considerably in practice, is that the cardinality and the PB constraints are too heavy, especially so when the value of the objective function given the initial model is high.

1.1. Weighted Component of `TT-Open-WBO-Inc`

Table 1 tracks the evolution of the weighted component of `TT-Open-WBO-Inc`. The baseline algorithm, `BMO`, is inherited from `Open-WBO-Inc` [3]. `BMO` is an incomplete algorithm that approximates weighted solving by clustering the target bits to groups of roughly similar (possibly relaxed) weight and applying `LSU` over these groups from the highest towards the lowest weights (inspired by Boolean Multilevel Optimization [1]). For MSE19, we incorporated the polarity and variables selection heuristics `TORC` and `TSB`, respectively [6,7], to bias the SAT solver towards both the best solution so far and the optimal solution. Specifically, `TORC` always chooses 0 as the initial polarity for the target bits, whereas, for non-target bits, it picks their value in the best solution so far (if already available). `TSB` boosts the scores of the target bits at the beginning. For MSE20, we enhanced the weighted component by a new variant of `Polosat`, `Polosat-OBV`. Section 2 below reviews `Polosat` and introduces `Polosat-OBV`. Finally, for this paper, we also integrated the (classical) local search solver `SATLike` [2], which we apply for preprocessing similarly to the winner of MSE20 in incomplete unweighted tracks `SATLike-20` as follows: obtain a first solution with SAT, improve it for 15 sec. with `SATLike` and switch to the main SAT-based flow.

1.2. Unweighted Component of `TT-Open-WBO-Inc`

Consider Table 2, which tracks the evolution of the unweighted component of our solver. Its baseline algorithm is `Mrs.Beaver` [5], inherited from `Open-WBO-Inc` [3]. `Mrs.Beaver` approximates MaxSAT by *Optimization Modulo Bit-Vectors (OBV)* [9]. OBV is an optimization

Table 2. Unweighted component evolvement in `TT-Open-WBO-Inc` since `Open-WBO-Inc`

MSE	Algorithm	Functionality	Incorporation	Ref.
–	`Polosat`	SAT-based local search	Supplant `Polosat-OBV`	[8]
	`SATLike`	Classical local search	Preprocess for 15 sec.	[2]
2020	`Polosat-OBV`	SAT-based local search	Replace SAT queries	New
	`GSC & SSCP`	`Mrs. Beaver` heuristics	Modify `Mrs. Beaver`	[7]
	`TORC`	SAT heuristic	Change default heuristic	[7]
2018	`Mrs. Beaver`	SAT-based MaxSAT flow	Baseline (`Open-WBO-Inc`)	[5]

problem akin to MaxSAT, with the only distinction being that OBV's objective is to minimize the *value* of the target T (where T is interpreted as an unsigned number), that is, in OBV, the order of the target bits matters. `Mrs.Beaver` comprises two stages. The first *incomplete* stage tries to quickly improve the best known solution by iteratively applying the `OBV-BS` OBV algorithm. Briefly, `OBV-BS` works as follows. It goes down the target bits from the MSB towards the LSB, where, for the current bit i, it checks if there is a solution with $t_i = 0$ using a SAT query (`OBV-BS` inside `Mrs.Beaver` limits every SAT query by a 1000 conflict threshold). If no solution is found, t_i is fixed to 1. Otherwise, t_i and any subsequent satisfied bits $t_{i+1} \dots t_{i+l}$ are fixed to 0 and the algorithm moves on to testing bit number $i+l+1$. For improving the odds of finding a good unordered solution with `OBV-BS`'s ordered approximation, `Mrs.Beaver` shuffles or reverses the target bits provided to `OBV-BS` in every iteration; some iterations also push any 0-valued bits towards the MSB after a new model is found [5]. The second *complete* stage of `Mrs.Beaver` falls back to `LSU`.

As summarized in Table 2, we upgraded the unweighted component for MSE20 as follows. First, we added `TORC` (omitting `TSB` due to preliminary experiments showing no benefit in unweighted solving). Second, we enhanced `Mrs.Beaver` by the following two heuristics from Sect. IV.1 in [7]: Global Stopping Condition (`GSC`) and Size-based Switching to Complete Part (`SSCP`). With `SSCP`, `Mrs.Beaver` switches from the incomplete to the complete stage whenever the number of clauses expected to be generated by `LSU` (for blocking all the solutions of weight $\geqslant \Psi(\mu_1)$) is not greater than 10^6; otherwise, `Mrs.Beaver` never switches. `GSC` halts any `OBV-BS` iteration at the incomplete stage and proceeds to the next one once (and if) the number of target bits fixed to 1 by `OBV-BS` matches that of the best MaxSAT solution so far (so, no improvement is possible). Third, we incorporated `Polosat-OBV`. For this paper, we integrated `SATLike` [2] similarly to the weighted component, and, based on our results in Section 3, reverted from `Polosat-OBV` to `Polosat` [8] for unweighted solving.

The following Section 2 is about SAT-based local search: it reviews `Polosat` and introduces `Polosat-OBV`. Section 3 is dedicated to experimental results. Section 4 concludes our work.

2. SAT-Based Local Search: `Polosat` and `Polosat-OBV`

Alg. 1 presents the code of both `Polosat` [8] and our new `Polosat-OBV` algorithm.

2.1. `Polosat`

Consider Alg. 1 in `Polosat` mode (with lines 8 to 13 disabled). It implements Alg. 2 in [8]. `Polosat` can be understood as a SAT-based local search algorithm. First, it invokes a SAT solver to get the first model and stashes that model in μ. Then it enters a loop, where each iteration is called an *epoch*. Each epoch tries to improve the best model so far μ. The algorithm finishes and returns μ, when a certain epoch cannot improve μ anymore.

Each epoch goes over the so-called *bad target bits* B, where a target bit is considered *bad* if it has not been assigned 0 in any model from the beginning of the current epoch. The algorithm tries to flip each bad target bit t_i by sending the SAT solver a *flip-query* with $\neg t_i$ as an assumption. Note that if the flip-query finds any model, it must be different from every other model encountered during the current epoch, since the current bad target bit is enforced to 0. If a model better than μ is found, μ is updated. The set of the bad target bits B is refined, whenever *any* new model is found.

2.2. `Polosat-OBV`

The idea behind our new `Polosat-OBV` algorithm is as follows. Let t_i be the current bad target bit, encountered by `Polosat`. We are interested to make an additional SAT query, called the *context-query*, prior to the flip-query. For the context-query, the SAT solver is provided with the assumption $\neg t_i$ (as in `Polosat`) along with a set of assumptions assigning the target bit variables $t_j : 1 \leqslant j < i$ their polarity in μ. The context-query looks for a new model in a more restricted context, induced by the value in μ of the current target prefix. It is expected to come back faster than the flip-query, because of the additional assumptions. If the context-query succeeds to improve the best model, we skip the flip-query for the current bad target bit. Otherwise, the flip-query is applied as usual.

Algorithm 1 `Polosat-OBV` or `Polosat`

1: $\mu := \textsc{Sat}()$ ▷ μ: the best model so far
2: $is\_good\_epoch := 1$
3: **while** is_good_epoch **do** ▷ One loop is an epoch
4: $\quad B := \{t : t \in T, \mu(t) = 1\}$ ▷ B is an order-preserving subset of T
5: $\quad is\_good\_epoch := 0$
6: $\quad$ **while** B is not empty **do**
7: $\quad\quad t_i := B.front(); B.dequeue()$ ▷ $B.front$ () returns $t_i \in B$ with the smallest i
8: $\quad\quad$ **if** `Polosat-OBV` is applied **then**
9: $\quad\quad\quad P := \{t_j \in T : j < i\}$
10: $\quad\quad\quad \sigma := \textsc{Sat}(\{\neg t_i\} \cup \{t : t \in P \wedge \mu(t) = 1\} \cup \{\neg t : t \in P \wedge \mu(t) = 0\})$
11: $\quad\quad\quad$ **if** SAT **then** ▷ Satisfiable
12: $\quad\quad\quad\quad$ **if** $\Psi(\sigma) < \Psi(\mu)$ **then** $\mu := \sigma$ and $is\_good\_epoch := 1$
13: $\quad\quad\quad\quad B := \{t : t \in B, \sigma(t) = 1\}$ ▷ B is an order-preserving subset of T
14: $\quad\quad$ **if** (`Polosat` is applied) or (previous call wasn't SAT or $\Psi(\sigma) \geqslant \Psi(\mu)$) **then**
15: $\quad\quad\quad \sigma := \textsc{Sat}(\{\neg t_i\})$
16: $\quad\quad\quad$ **if** SAT **then** ▷ Satisfiable
17: $\quad\quad\quad\quad$ **if** $\Psi(\sigma) < \Psi(\mu)$ **then** $\mu := \sigma$ and $is\_good\_epoch := 1$
18: $\quad\quad\quad\quad B := \{t : t \in B, \sigma(t) = 1\}$ ▷ B is an order-preserving subset of T
19: **return** μ

2.3. Polosat and Polosat-OBV Integration into TT-Open-WBO-Inc

We integrated Alg. 1, which supports both Polosat and Polosat-OBV into TT-Open-WBO-Inc as follows. SAT invocations are replaced by Alg. 1 invocations, where the target bits are sorted by their weight in decreasing order and target bits having the same weight are randomly shuffled. In addition, we apply an *adaptive strategy* that stops Alg. 1 forever and falls back to SAT whenever the model generation rate of Alg. 1 is slower than n model per second, where $n = 1$ for the weighted component and $n = 2$ for the unweighted one. We use *conflict threshold* of 1000 for every SAT query. Furthermore, the weighted component also uses the *mutation combination* strategy (see Section 3.B in [8]), which tries to combine solutions to create new ones, although mutation combination's impact is marginal [8].

3. Experimental Results

In this section, we study the performance of both weighted and unweighted components of TT-Open-WBO-Inc and compare it to the state-of-the-art solvers SATLike-cw-20 (weighted) and SATLike-c-20 (unweighted) [2]. We ran the following 6 configurations of TT-Open-WBO-Inc for both the weighted and unweighted cases: TT-{S}-{P}, where $\mathtt{S} \in \{\mathtt{SL}, \mathtt{NoSL}\}$ and $\mathtt{P} \in \{\mathtt{NoPol}, \mathtt{Pol}, \mathtt{PolOBV}\}$. Specifically, SATLike is applied iff $\mathtt{S} = \mathtt{SL}$. In addition, if $\mathtt{P} = \mathtt{Pol}$ then Polosat is applied; if $\mathtt{P} = \mathtt{PolOBV}$ then Polosat-OBV is applied; if $\mathtt{P} = \mathtt{NoPol}$ then neither Polosat nor Polosat-OBV is applied.

We used MSE20 benchmarks. We calculated the score $c \in [0 \ldots 1]$ for solver S and time interval T, given a particular instance, as follows: $c = \sum_i(1 +$ the minimal weight found by any participating solver in 30 minutes) / (1 + the weight found by S in time interval T). We ran the solvers for 30 minutes and measured their average score at different time intervals. We used machines with 32 Gb of memory running Intel® Xeon® processors with 3 Ghz CPU frequency. The results are shown in Fig. 1 and Fig. 2 for weighted and unweighted instances, respectively.[1]

On weighted instances, our new Polosat-OBV algorithm considerably outperforms Polosat, independently of whether SATLike is used. Moreover, combining SAT-based local search and the classical local search yields excellent results. The resulting solver TT-SL-PolOBV is significantly more efficient than both the version without SATLike (TT-NoSL-PolOBV), which won MSE20, and the version without any SAT-based local search (TT-SL-NoPol).

In a striking difference from the weighted case, in the unweighted scenario, the best-performing version over time utilizes Polosat instead of Polosat-OBV, and the overall impact of Polosat is less significant than in the weighted case. The reason might be related to the similarity of the additional context-query in Polosat-OBV to the SAT queries in the OBV-BS-based underlying Mrs.Beaver algorithm. In a sense, Mrs.Beaver already implements a variant of SAT-based local search with context-queries only. Integrating SATLike yields excellent results similarly to the weighted case. The best resulting solver TT-SL-Polosat outeprforms the winner of MSE20, SATLike-c-20, for every time interval.

[1] The code of the solvers we have used and instructions on how to reproduce our experiments are available at https://drive.google.com/file/d/1QIx0oyBZOBXtxXqHPubvfZt0XSXkxjcG/view?usp=sharing.

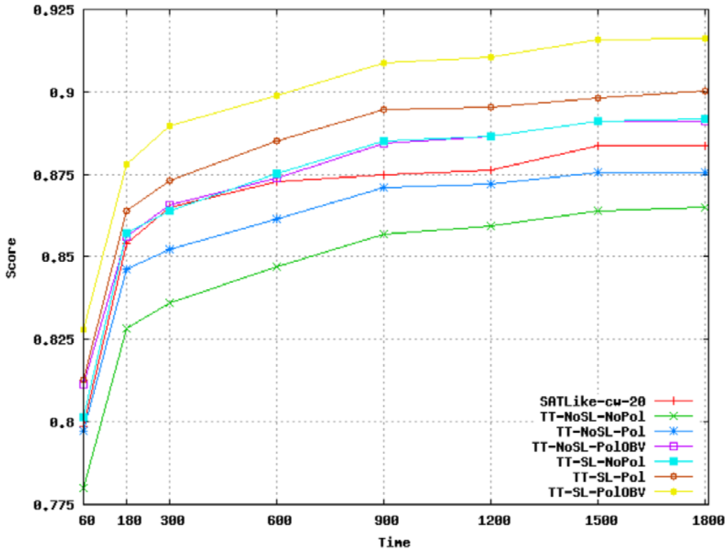

Figure 1. Weighted results.

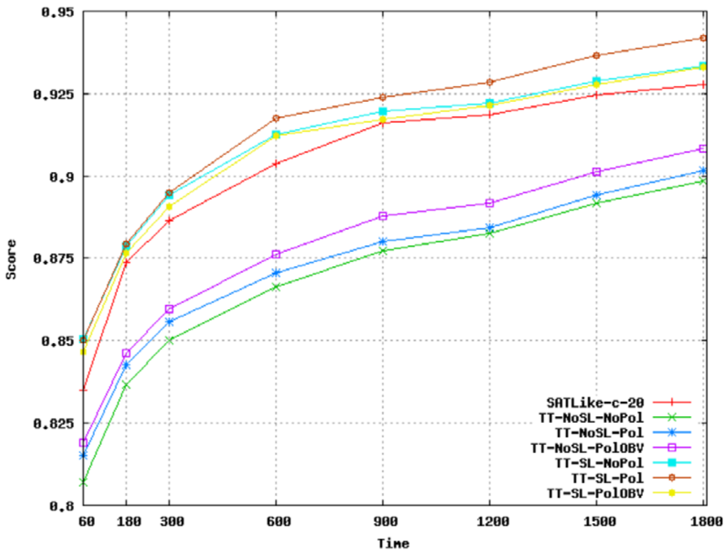

Figure 2. Unweighted results.

4. Conclusion

We presented the latest version of our state-of-the-art anytime MaxSAT solver `TT-Open-WBO-Inc`. Our exposition included: (1) Describing a previously unpublished variant of the SAT-based local search algorithm `Polosat`, `Polosat-OBV`, and concluding that `Polosat-OBV` is superior for weighted solving, while `Polosat` is better for unwegihted solving. (2) Detailing `TT-Open-WBO-Inc`'s unweighted component, which has not been examined in prior work; (3) Showing that incorporating the local search solver `SATLike` [2] for preprocessing allows TT-Open-WBO-Inc to surpass the winners of MSE20 in all four incomplete tracks.

References

[1] J. Argelich, I. Lynce and J.P. Marques Silva, On solving Boolean multilevel optimization problems, in: *IJCAI 2009*, pp. 393–398.

[2] S. Cai and Z. Lei, Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability, *Artif. Intell.* **287** (2020), 103354. doi:10.1016/j.artint.2020.103354.

[3] S. Joshi, P. Kumar, S. Rao and R. Martins, Open-wbo-inc: Approximation strategies for incomplete weighted maxsat, *J. Satisf. Boolean Model. Comput.* **11**(1) (2019), 73–97.

[4] D. Le Berre and A. Parrain, The sat4j library, release 2.2, *JSAT* **7**(2–3) (2010), 59–64.

[5] A. Nadel, Solving MaxSAT with bit-vector optimization, in: *SAT 2018*, 2018, pp. 54–72.

[6] A. Nadel, Polarity and variable selection heuristics for SAT-based anytime MaxSAT, *J. Satisf. Boolean Model. Comput.* **12** (2020), 17–22.

[7] A. Nadel, Anytime weighted MaxSAT with improved polarity selection and bit-vector optimization, in: *FMCAD 2019*, pp. 193–202.

[8] A. Nadel, On optimizing a generic function in SAT, in: *FMCAD 2020*, pp. 205–213.

[9] A. Nadel and V. Ryvchin, Bit-vector optimization, in: *TACAS 2016*, pp. 851–867.