TT-Open-WBO-Inc-21: an Anytime MaxSAT Solver Entering MSE'21

Alexander Nadel
Email: alexander.nadel@cs.tau.ac.il

***Abstract*—This document describes the solver TT-Open-WBO-Inc-21, submitted to the four incomplete tracks of MaxSAT Evaluation 2021. TT-Open-WBO-Inc-21 is the 2021 version of our solver TT-Open-WBO-Inc [9], [10] (itself based on Open-WBO-Inc [3]), which came in first in both the incomplete weighted tracks and second in both the incomplete unweighted tracks at MaxSAT Evaluation 2020 (MSE20). TT-Open-WBO-Inc-21 includes the following two major new features as compared to the previous version TT-Open-WBO-Inc-20: 1) integration of SATLike [2] for inprocessing, and 2) further modifications (as compared to TT-Open-WBO-Inc-20 [10]) to the Polosat algorithm [8].**

I. Introduction

Applying the SATLike local search algorithm [2] as a preprocessor, followed by invoking a SAT-based anytime MaxSAT algorithm proved to be a successful strategy for anytime MaxSAT solving, used by both the winner of MSE20 in the two incomplete unweighted tracks SATLike-c-20 [4] and the runner-up in the two incomplete weighted tracks SATLike-cw-20 [5].

Following these results, we have integrated SATLike into TT-Open-WBO-Inc. Moreover, while the weighted component of our solver uses SATLike as a preprocessor, the unweighted component uses it as an *inprocessor*, that is, SATLike is invoked more than once during the solver's lifetime. See Sect. II below for more details.

Polosat algorithm [8] can be understood as a SAT-based local search. Using it *instead* of plain SAT solving significantly improves the performance of anytime MaxSAT solving [8]. TT-Open-WBO-Inc-20 had used Polosat or, more precisely, a modified version of the algorithm, for both weighted and unweighted solving. In our new version TT-Open-WBO-Inc-21, we changed Polosat further, where the modifications differ between the weighted and the unweighted components. Sect. III contains further details.

II. Integrating SATLike

We found in preliminary experiments that it pays off to integrate SATLike in a different manner into the weighted and unweighted components of the solver, respectively.

A. SATLike *in the Weighted Component*

The weighted component of the solver uses SATLike as part of its flow as follows (similarly to the way it is used by SATLike-c-20 and SATLike-cw-20):

1) Run a SAT solver to find an initial model μ.
2) Invoke SATLike for 15 seconds to improve μ, if possible.
3) Switch to an anytime SAT-based algorithm, where μ is used as the initial model. For the anytime SAT-based algorithm, we apply the BMO-based clustering [3] with TORC polarity selection [6], in which SAT invocations are replaced by invocations of our enhanced version of Polosat, discussed in Sect. III-A.

B. SATLike *in the Unweighted Component*

The unweighted component applies SATLike for inprocessing as part of the following flow:

1) Run a SAT solver to find an initial model μ.
2) Invoke SATLike for 15 seconds to improve μ, if possible.
3) Switch to an anytime SAT-based algorithm, where μ is used as the initial model. We apply the Mrs. Beaver algorithm [7], enhanced by TORC polarity selection [6] and two further heuristics from [6]: global stopping condition for OBV-BS and size-based switching to complete part. The SAT invocations are replaced by invocations of our enhanced version of Polosat, discussed in Sect. III-B.
4) Stop the anytime SAT-based algorithm after 60 sec. Let the best model so far be μ.
5) Re-invoke SATLike for 15 seconds to improve μ, if possible. Go to step 3.

III. Polosat Modifications

We assume that a MaxSAT instance comprises a set of *hard* satisfiable 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. The *weight of a variable assignment* μ is $O(T, \mu) = \sum_{i=1}^{n} \mu(t_i) \times w_i$, that is, the overall weight of T's bits, satisfied by μ. Given a MaxSAT instance, a MaxSAT solver is expected to return a model having the minimum possible weight.

Polosat [8] 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. In addition, we apply an *adaptive strategy* that stops Polosat forever and falls back to SAT whenever the model generation rate of Polosat is too slow (1 and 2 models per second for the weighted and unweighted components, respectively).

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 original algorithm in [8] tries to flip each bad target bit t_i by sending the SAT solver the so-called *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 updated, whenever *any* new model is found. In addition, to simulate local search further, `Polosat` applies the `TORC` polarity selection heuristic [6]; see [8] for details.

We modified `Polosat` already in `TT-Open-WBO-Inc-20` as follows. Let t_i be the current bad target bit, encountered by `Polosat`. `TT-Open-WBO-Inc-20` uses an additional SAT query, called the *prefix-query*, prior to the flip-query. For the prefix-query, the SAT solver is provided with the assumption $\neg t_i$ (as in the original `Polosat`) along with a set of assumptions assigning the target bit variables $t_j : 1 \leq j < i$ their polarity in μ. The prefix-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 prefix-query succeeds to improve the best model, the solver skips the flip-query for the current bad target bit. Otherwise, the flip-query is applied as usual.

In our new version of the solver, we modified the queries to the SAT solver further, where the modifications differ for the weighted and unweighted components, respectively.

A. `Polosat` *in the Weighted Component*

The modification is similar to the one we have described above, that is, we apply the prefix-query, followed by the flip-query, except for the following single difference. The flip-query is now skipped whenever the prefix-query finds *any* model (rather than whenever it succeeds to improve the best model so far). The updated algorithm is shown in Fig. 1.

B. `Polosat` *in the Unweighted Component*

Let the *full-query* be a SAT invocation, where the solver is provided with the assumption $\neg t_i$ (as in the original `Polosat`) along with a set of assumptions assigning *all* the target bit variables, but t_i (that is, $t_j : 1 \leq j \neq i \leq n$) their polarity in μ. Note that the search space during the full-query is even more restricted than during the prefix-query.

Our unweighted component sends the solver the flip-query, followed by the full-query, where the full-query is skipped, whenever the flip-query succeeds to improve the best model so far. The algorithm is shown in Fig. 2.

Algorithm 1 `polosat-weighted`

$\mu := \text{SAT}()$ ▷ μ: the best model so far
$g := 1$ ▷ g is 1 iff the current epoch is good
while g **do** ▷ One loop is an epoch
  $B := \{t : t \in T, \mu(t) = 1\}$
  $g := 0$
  while B is not empty **do**
    $t_i := B.front(); B.dequeue()$
    $P := \{t_j \in T : j < i\}$
    $\sigma := \text{SAT}(\{\neg t_i\} \cup \{t : t \in P \wedge \mu(t) = 1\} \cup \{\neg t : t \in P \wedge \mu(t) = 0\})$
    if SAT **then** ▷ Satisfiable
      if $\Psi(\sigma) < \Psi(\mu)$ **then** $\mu := \sigma$ and $g := 1$
      $B := \{t : t \in B, \sigma(t) = 1\}$
    else
      $\sigma := \text{SAT}(\{\neg t_i\})$
      if SAT **then** ▷ Satisfiable
        if $\Psi(\sigma) < \Psi(\mu)$ **then** $\mu := \sigma$ and $g := 1$
        $B := \{t : t \in B, \sigma(t) = 1\}$
return μ

Algorithm 2 `polosat-unweighted`

$\mu := \text{SAT}()$ ▷ μ: the best model so far
$g := 1$ ▷ g is 1 iff the current epoch is good
while g **do** ▷ One loop is an epoch
  $B := \{t : t \in T, \mu(t) = 1\}$
  $g := 0$
  while B is not empty **do**
    $t_i := B.front(); B.dequeue()$
    $\sigma := \text{SAT}(\{\neg t_i\})$
    if SAT **then** ▷ Satisfiable
      if $\Psi(\sigma) < \Psi(\mu)$ **then** $\mu := \sigma$ and $g := 1$
      $B := \{t : t \in B, \sigma(t) = 1\}$
    if not SAT or $\Psi(\sigma) \geq \Psi(\mu))$ **then**
      $P := \{t_j \in T : j \neq i\}$
      $\sigma := \text{SAT}(\{\neg t_i\} \cup \{t : t \in P \wedge \mu(t) = 1\} \cup \{\neg t : t \in P \wedge \mu(t) = 0\})$
      if SAT **then** ▷ Satisfiable
        if $\Psi(\sigma) < \Psi(\mu)$ **then** $\mu := \sigma$ and $g := 1$
        $B := \{t : t \in B, \sigma(t) = 1\}$
return μ

REFERENCES

[1] F. Bacchus, J. Berg, , M. Järvisalo, and R. Martins, editors. *MaxSAT Evaluation 2020: Solver and Benchmark Descriptions*, Department of Computer Science Report Series B 2020-2, Finland, 2020. Department of Computer Science, University of Helsinki.

[2] S. Cai and Z. Lei. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability. *Artif. Intell.*, 287:103354, 2020.

[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):73–97, 2019.

[4] Z. Lei and S. Cai. Satlike-c: Solver description. In Bacchus et al. [1].

[5] Z. Lei and S. Cai. Satlike-c(w): Solver description. In Bacchus et al. [1].

[6] A. Nadel. Anytime weighted MaxSAT with improved polarity selection and bit-vector optimization. In *FMCAD 2019*, pages 193–202.

[7] A. Nadel. Solving MaxSAT with bit-vector optimization. In *SAT 2018*, pages 54–72, 2018.

[8] A. Nadel. On optimizing a generic function in SAT. In *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*, pages 205–213. IEEE, 2020.

[9] A. Nadel. Polarity and variable selection heuristics for SAT-based anytime MaxSAT. *J. Satisf. Boolean Model. Comput.*, 12(1):17–22, 2020.

[10] A. Nadel. TT-Open-WBO-Inc-20: an Anytime MaxSAT Solver Entering MSE'20. In Bacchus et al. [1].