

# TT-Open-WBO-Inc-20: an Anytime MaxSAT Solver Entering MSE'20

Alexander Nadel

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

**Abstract**—This document describes the solver TT-Open-WBO-Inc-20, submitted to the four incomplete tracks of MaxSAT Evaluation 2020. TT-Open-WBO-Inc-20 is the 2020 version of our solver TT-Open-WBO-Inc [5], which came in first at both the weighted, incomplete categories at MaxSAT Evaluation 2019. TT-Open-WBO-Inc-20 has the following two major new features as compared to the previous version: 1) We now support *unweighted* anytime MaxSAT. We apply the Mrs. Beaver algorithm [3], enhanced by several heuristics from the WMB algorithm [4]; 2) We have integrated into TT-Open-WBO-Inc-20 our new Polosat algorithm for solving the problem of generic optimization in SAT [6].

## I. INTRODUCTION

In this document, we assume that a MaxSAT instance comprises a set of *hard* satisfiable clauses  $H$  and a *target bit-vector (target)*  $T = \{t_n, t_{n-1}, \dots, t_1\}$ , 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*  $\mu$  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  $\mu$ . Given a MaxSAT instance, a MaxSAT solver is expected to return a model having the minimum possible weight.

Below, we briefly document: 1) Our new Polosat algorithm for the OptSAT problem of optimizing a generic Pseudo-Boolean function in SAT. Polosat is applied in both the weighted and unweighted components of TT-Open-WBO-Inc-20; 2) The weighted component of TT-Open-WBO-Inc-20; 3) The unweighted component of TT-Open-WBO-Inc-20.

Although we did our best to document our solver as precisely as possible, inevitably, we had to omit some details due to space restrictions. OptSAT, Polosat and the weighted component of our solver are described in full detail in [6].

## II. THE Polosat ALGORITHM FOR GENERIC OPTIMIZATION IN SAT (OPTSAT)

Recall that a Pseudo-Boolean (PB) function is a function that maps every full assignment to a real number.

We now state the OptSAT problem. Given a satisfiable formula  $F(V)$  in CNF and an objective Pseudo-Boolean (PB) function  $\Psi$ , OptSAT returns a model  $\mu$  to  $F$ , such that for every model  $\mu'$  to  $F$ , it holds that  $\Psi(\mu) \leq \Psi(\mu')$ . OptSAT can be thought of as a generalization of MaxSAT, which supports arbitrary PB functions, whereas MaxSAT is restricted to linear PB functions.

In this document, we assume that the objective function  $\Psi$  is strictly monotone in a set of observable variables.

More specifically, let  $Obs \subseteq V$  be a set of *observables* and  $\Psi: [0, 1]^{|V|} \rightarrow \mathbb{R}$  be a PB function. Then: 1)  $\Psi$  is *restrictable* to  $Obs$ , iff for every two assignments  $\theta$  and  $\lambda$ , such that  $\theta(v) = \lambda(v)$  for every  $v \in Obs$ , it holds that  $\Psi(\theta) = \Psi(\lambda)$ ; 2)  $\Psi$  is *strictly monotone in observables*  $Obs$ , iff  $\Psi$  is restrictable to  $Obs$  and for every assignment  $\theta$  and every variable  $v \in Obs$ , such that  $\theta(v) = 1$ , it holds that  $\Psi(\theta^{-v}) < \Psi(\theta)$ .

Note that the objective function  $O(T, \mu)$  in MaxSAT is strictly monotone in the target bits. First,  $O(T, \mu)$  depends only on the target bits. Second, when one of the target bits decreases,  $O(T, \mu)$  decreases too, hence  $O(T, \mu)$ . Hence, an OptSAT algorithm can be applied to solve MaxSAT.

Below, we present our anytime Polosat algorithm for solving OptSAT. It can be used incrementally under assumptions, similarly to modern SAT solvers. Our algorithm is incomplete; it works until a fixed-point, but does not guarantee that the eventual solution is optimal. In this document, we present the strictly monotone version of Polosat. As we shall see, to solve MaxSAT, we integrate Polosat into higher-level MaxSAT algorithms (rather than applying solely Polosat).

*Fixing the polarity* of a variable  $v$  to a Boolean value  $\sigma$  during SAT solver's invocation means assigning  $v$  the value  $\sigma$ , whenever  $v$  is chosen by the solver's decision heuristic. Intuitively, Polosat carries out a purely SAT-based local search algorithm, based on polarity-fixing.

Polosat is shown in Alg. 1. It receives three parameters: 1) A satisfiable CNF formula  $F$  (if the invocation is incremental, assume that  $F$  contains all the clauses, provided by the user so far); 2) A (possibly empty) set of assumptions  $Asmp$ . The assumptions are guaranteed to hold for one particular invocation of the algorithm; 3) The observables  $Obs$ ; 4) The objective PB function to minimize  $\Psi: [0, 1]^{|V|} \rightarrow \mathbb{R}$ .

The algorithm maintains an instance of an incremental SAT solver throughout its execution and the best model so far  $\mu$ . Polosat starts with initializing  $\mu$  with a model by invoking the SAT solver (line 3). Then, it operates in iterations, where each iteration is called an *epoch* (lines 5 to 16). Each epoch tries to improve  $\mu$ . An epoch is *good* if it manages to improve  $\mu$ , otherwise it is *bad*. Our incomplete algorithm finishes whenever a bad epoch is completed.

Each epoch tries to improve the best model so far  $\mu$  in a loop (lines 8 to 16) by looking for a better solution near  $\mu$  when, for each loop iteration, one of the variables is forced to flip its value. In addition, the observables are always fixed to 0. The

loop inside each epoch goes over a set of literals  $B$ , initialized by all the observable variables assigned to 1 by  $\mu$  (line 6). For each variable  $v$ , `Polosat` tries to find a model near  $\mu$  with  $v$  flipped. This is carried out by fixing the polarities of all the variables, except for the observables, to their values in  $\mu$  (line 10), followed by a SAT invocation with  $\neg v$  as a hard assumption (line 11). If the problem is satisfiable and a model  $\sigma$  better than  $\mu$  if found, then: 1)  $\mu$  is updated to  $\sigma$ , 2) the epoch is marked as good. In addition, any observable assigned to 0 in any model is removed from  $B$ .

---

**Algorithm 1** `Polosat`


---

```

1: function SOLVE(CNF  $F$ ; Literals  $Asmp$ ; Variables  $Obs$ ;
    $\Psi: [0, 1]^{|V|} \rightarrow \mathbb{R}$ )
Require:  $F$  is satisfiable and  $\Psi$  is strictly monotone in  $Obs$ 
2:   Fix the polarities of the observables  $Obs$  to 0
3:    $\mu := \text{SAT}(Asmp)$   $\triangleright \mu$ : the best model so far
4:    $is\_good\_epoch := 1$ 
5:   while  $is\_good\_epoch$  do  $\triangleright$  One loop is an epoch
6:      $B := \{v : v \in Obs, \mu(v) = 1\}$ 
7:      $is\_good\_epoch := 0$ 
8:     while  $B$  is not empty do
9:        $v := B.front(); B.dequeue()$ 
10:      Fix the polarities of the observables  $Obs$  to 0
      and all the other variables (that is,  $V \setminus Obs$ ) to  $\mu$ 
11:       $\sigma := \text{SAT}(Asmp \cup \{\neg v\})$ 
12:      if SAT then  $\triangleright$  Satisfiable
13:        if  $\Psi(\sigma) < \Psi(\mu)$  then
14:           $\mu := \sigma$   $\triangleright$  Update the best model so far
15:           $is\_good\_epoch := 1$   $\triangleright$  Good epoch!
16:           $B := \{v : v \in B, \sigma(v) = 1\}$ 

```

---

### III. THE WEIGHTED COMPONENT

We have integrated `Polosat` into the Bounded Multilevel Optimization (BMO)-based anytime MaxSAT algorithm [2], which we call *BMO-based Clustering (BC)*, implemented already in [2], [5].

BC [2] clusters all the target bits to disjoint classes based on their weight. That is, all the targets of the same weight  $w$  belong to the same class. Then, the algorithm sorts the classes according to their weight and goes over them one-by-one starting with the class associated with the highest weight. BC tries to falsify as many target bits in each class as possible with incremental SAT invocations. After BC completes processing one class, it fixes the overall number of falsified target bits in that class.

Our implementation simply replaces every SAT invocation with a `Polosat` invocation in BC with the target  $T$  as the observables and  $O(T, \mu)$  as the objective function. However, there are several subtleties: *a)* We exclude from the set of observables the bits which belong to the fixed classes; *b)* We sort the observables by their weight in decreasing order; *c)* We randomly shuffle the observables within each class before every `Polosat` invocation to diversify `Polosat`'s execution.

### A. `Polosat` Enhancements

We modified `Polosat` to keep track of the number of *Models Per Second* (MPS) throughout its execution starting immediately after the initial SAT invocation. MPS is updated and tested after each SAT invocation. If MPS is lower than 1, the current invocation of `Polosat` is terminated, and the high-level BC algorithm falls back to invoking a plain SAT solver instead of `Polosat` for the rest of its execution. Falling back to SAT makes sense, since SAT/`Polosat` queries tend to become more difficult as the algorithm advances towards the ideal, hence MPS is unlikely to increase.

Furthermore, we use the conflict threshold of 1000 for all the SAT invocations inside `Polosat`, except for the first one.

The two enhancements above are detailed in [6]. On top of that, we have modified `Polosat` to include an additional SAT invocation *before* the one which checks if the current observable  $v$  can be flipped (line 11). Let  $R$  be the subset of the literals in  $\mu$  which correspond to all the observables that appear *before*  $v$  in  $Obs$  (where the polarity of each such literal is determined by  $\mu$ ). We add  $R$  to the assumptions for our additional SAT invocation. We proceed with the original SAT invocation at line 11, only if no model better than  $\mu$  was found by the first invocation. Otherwise, we proceed to the next loop iteration. This adjustment essentially combines `Polosat` and `WMB`.

### IV. THE UNWEIGHTED COMPONENT

The unweighted component uses the `Mrs. Beaver` algorithm [3], enhanced by the following two heuristics from Sect. 4.1 in [4]: global stopping condition for OBV-BS and size-based switching to complete part.

We have integrated `Polosat` into our algorithm by simply replacing SAT invocations with `Polosat` invocations. We apply all the `Polosat` enhancements from Sect. III-A in our unweighted component, where the adapted strategy is used with the threshold of 2 (rather than 1).

In addition, we use chronological backtracking [7] (with the configuration  $\{T = 100, C = 0\}$ ), which we have implemented in the underlying SAT solver `Glucose 4.1` [1]. Furthermore, we take advantage of the `TORC` polarity selection heuristic [4] throughout the algorithm's execution.

### REFERENCES

- [1] G. Audemard and L. Simon. On the glucose SAT solver. *Int. J. Artif. Intell. Tools*, 27(1):1840001:1–1840001:25, 2018.
- [2] 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.
- [3] A. Nadel. Solving maxsat with bit-vector optimization. In *SAT 2018*, pages 54–72, 2018.
- [4] A. Nadel. Anytime weighted maxsat with improved polarity selection and bit-vector optimization. In *FMCAD 2019*, pages 193–202, 2019.
- [5] A. Nadel. TT-Open-WBO-Inc: Tuning Polarity and Variable Selection for Anytime SAT-based Optimization. Department of Computer Science Report Series B, Finland, 2019. Department of Computer Science, University of Helsinki.
- [6] A. Nadel. On optimizing a generic function in SAT. 2020. Under submission.
- [7] A. Nadel and V. Ryvchin. Chronological backtracking. In *SAT 2018*, pages 111–121, 2018.