SATLike-c: Solver Description

Zhendong Lei[1,2],Shaowei Cai[1,2],Fei Geng[3],Dongxu Wang[3],Yongrong Peng[3],Dongdong Wan[3],Yiping Deng[3] and Pinyan Lu[3,4]
[1]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China*
[2]*School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing, China*
[3]*TCS Lab, Huawei Technologies, Shanghai, China*
[4]*Shanghai University of Finance and Economics, Shanghai, China*
{leizd, caisw}@ios.ac.cn {gengfei12, dongxu.wang, pengyongrong, wandongdong1, yiping.deng, lupinyan}@huawei.com

***Abstract*—This document describes the solver SATLike-c, submitted to the four incomplete tracks of MaxSAT Evaluation 2021.**

I. Introduction

SATLike-c participates in incomplete track. SATLike-c has two main engines, one is local search solver SATLike [1] and the other is SAT-based solver TT-Open-WBO-inc [2].

A. Local Search Algorithm: SATLike

SATLike adopts a dynamic local search framework for SAT and exploits the distinction of hard and soft clauses by a carefully designed clauses weighting scheme. The clauses weighting scheme works on both hard and soft clauses while it puts more increments to hard clauses each time and also sets a limit on the maximum weight that each soft clause can get. As for the variable selection heuristic, it works like a normal local search for SAT which pick a variables with the highest score in each step. The algorithm is thus called SATLike.

The weighting scheme used in SATLike is named **Weighting-PMS**, and works as follows. For each hard clause, we associate an integer number as its weight which is initialized to 1; for each soft clause, we use the original weight (which is 1 for PMS, and is the original weight from the input file for WPMS) as its initial weight. Whenever a "stuck" situation is observed, that is, we cannot decrease the cost by flipping any variable, then clause weights are updated as follows.

- with probability $1 - sp$: for each falsified hard clause c, $w(c) := w(c) + h\_inc$; for each falsified soft clause c, $w(c) := w(c) + 1$ if $w(c) < \zeta$, where ζ limits the maximum value that a soft clause weight can get.
- with probability sp (smoothing probability): for each satisfied hard clause c s.t. $w(c) > 1$, $w(c) := w(c) - h\_inc$; for each satisfied soft clause c s.t. $w(c) > 1$, $w(c) := w(c) - 1$.

SATLike uses scoring function (the score of variables) to guide the search. In SATLike, the score of variable x, denoted by $score(x)$, is the increase of total weight of satisfied clauses (either hard clauses or soft clauses) caused by flipping x.

The main component of SATLike is a loop (lines 3-15), which is executed to iteratively modify the current solution α until a given time limit is reached. During the search, whenever a better feasible solution is found, the best feasible solution is updated accordingly (line 4).

Algorithm 1: SATLike

Input: PMS instance F, *cutoff*
Output: A feasible assignment α of F and its cost, or "no feasible assignment found"

```
1  begin
2      α := an initial complete assignment; α* := ∅;
3      while elapsed time < cutoff do
4          if ∄ falsified hard clauses & cost(α) < cost* then
               α* := α; cost* := cost(α) ;
5          if D := {x|score(x) > 0} ≠ ∅ then
6              v := a variable in D picked by BMS strategy;
7          else
8              update weights of clauses by Weighting-PMS;
9              if ∃ falsified hard cluases then
10                 c := a random falsified hard clause
11             else c := a random falsified soft clause;
12             v :=the variable with highest score in c;
13         α := α with v flipped;
14     if α* is feasible then return (cost*, α*);
15     else return "no feasible assignment found";
```

In each step, if there exits variables with score bigger than 0, SATLike picks a variable with the greatest score and flips it. If there is no such variable, then *SATLike* updates clause weights according to the Weighting-PMS, and picks a variable from a falsified clause.

B. Hybrid Solver: SATLike-c

We combine SATLike with the state of the art SAT-based solvers TT-Open-WBO-inc [2], which leading to the hybrid solver SATLike-c.

The structure of SATLike-c is shown as algorithm 2. First, a SAT solver is executed to find a feasible solution (only works on hard clauses). Then SATLike is executed with this feasible solution as its initial solution. SATLike is executed until there is no improvement over k steps (k is set to 10^7 in our experiment). In most cases of our solver, SATLike can return a high-quality solution in this period, which is even close to the optimal one. But as the execution time increases, it is difficult for SATLike to get a further improved solution. So, the obtained high-quality solution is passed to the SAT-based

Algorithm 2: SATLike-c

Input: PMS instance F, *cutoff*
Output: A feasible assignment α of F and its cost, or "no feasible assignment found"

```
begin
    F'= Hard(F);
    α := SATSOLVER(F');
    α := SATLIKE(α, F);
    α := TTOPENWBOINC(α, F);
    if α* is feasible then return (cost*, α*);
    else return "no feasible assignment found";
```

solver as the initial model, and thus an initial upper bound is also provided. After that, TT-Open-WBO-inc is executed in the rest time.

REFERENCES

[1] Shaowei Cai, Zhendong Lei, "Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability". Artif. Intell. 287: 103354 (2020)

[2] Alexander Nadel."Anytime weighted maxsat withimproved polarity selection and bit-vector optimization" InClark W. Barrett and Jin Yang, editors,2019 Formal Methodsin Computer Aided Design, FMCAD 2019, San Jose, CA, USA,October 22-25, 2019, pages 193–202. IEEE, 2019