Weighted version of EvalMaxSAT 2022

Florent Avellaneda*, Carl-Elliott Bilodeau-Savaria†, Lancelot Normand‡
Computer Science Departement, Université du Québec à Montréal
QC, Canada
Email: *avellaneda.florent@uqam.ca, †bilodeau-savaria.carl-elliott@courrier.uqam.ca, ‡normand.lancelot@courrier.uqam.ca,

I. Introduction

EvalMaxSAT[1] is a MaxSAT solver written in modern C++ language mainly using the Standard Template Library (STL). The solver is built on top of the SAT solver CaDiCaL [1], but any other SAT solver can easily be used instead. EvalMaxSAT is based on the OLL algorithm [2] originally implemented in the MSCG MaxSAT solver [3], [4] and then reused in the RC2 solver [5]. This new version of the solver includes support for weighted formulas.

II. Description

The main strategy of the solver to solve weighted formula is to rapidly find non-optimal solutions that allow some soft variables to be transformed to hard variables. Specifically, if an assignment is obtained such that the sum of the weights of the unsatisfied soft variables is smaller than the weight of a soft variable v, then we can deduce that this variable v must be satisfied, and can therefore be considered a hard variable.

To find non-optimal solutions, there is one existing approach, known in the literature as the stratification strategy [6], involves considering only the variables with a weight higher than a certain threshold as soft, then reducing the threshold until all the soft variables are considered.

In this new version of EvalMaxSAT, a second strategy is added in addition to the stratification strategy. The second strategy does not immediately add new constraints when a new core is considered, but accumulates constraints until the formula becomes satisfiable. The accumulated constraints are considered only when the formula becomes satisfiable and a search for soft variables to transform to hard will be performed

Algorithm 1 presents a general view of how the solver functions. Note, that the function ChooseNextMinimumWeight represents a heuristic used to select a threshold value necessary to select a subset of soft variables. The heuristic implemented in the tool consists of reducing the threshold by a minimum step initially, then increasing this step when the computation time of the second loop (line 4) increases.

Algorithm 1

```
Input: A formula $\varphi$
 1: while $true$ do
 2:    $w_{min} \leftarrow ChooseNextMinimumWeight()$
 3:    $\varphi' \leftarrow \{cl \mid cl \in \varphi$ and $weight(cl) \geq w_{min}\}$
 4:    while $true$ do
 5:       $(result, \varphi_{core}) \leftarrow SATSolver(\varphi')$
 6:       if $result$ is a satisfying assignment then
 7:          $curCost \leftarrow CalculateCost(result, \varphi)$
 8:          $\varphi \leftarrow Harden(\varphi, curCost - cost)$
 9:          if $\varphi_{tmp}$ is empty then
10:             break
11:          end if
12:          $\varphi' \leftarrow \varphi' \cup \varphi_{tmp}$
13:          continue
14:       end if
15:       $\varphi_{core} \leftarrow minimize(\varphi', \varphi_{core})$
16:       $cost \leftarrow cost + min_{v \in \varphi_{core}}(weight(v))$
17:       $\varphi' \leftarrow relax(\varphi', \varphi_{core})$
18:       $\varphi_{tmp} \leftarrow \varphi_{tmp} \cup createSum(\varphi_{core}, k)$
19:    end while
20:    if $w_{min} = 1$ then
21:       return $cost$
22:    end if
23: end while
```

References

[1] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger, "CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020," in *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, ser. Department of Computer Science Report Series B, T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, Eds., vol. B-2020-1. University of Helsinki, 2020, pp. 51–53.

[2] A. Morgado, C. Dodaro, and J. Marques-Silva, "Core-guided MaxSAT with soft cardinality constraints," in *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014, Lyon, France, September 8-12, 2014. Proceedings*, ser. Lecture Notes in Computer Science, B. O'Sullivan, Ed., vol. 8656. Springer, 2014, pp. 564–573. [Online]. Available: https://doi.org/10.1007/978-3-319-10428-7_41

[3] A. Morgado, A. Ignatiev, and J. Marques-Silva, "MSCG: robust core-guided maxsat solving," *J. Satisf. Boolean Model. Comput.*, vol. 9, no. 1, pp. 129–134, 2014. [Online]. Available: https://satassociation.org/jsat/index.php/jsat/article/view/127

[4] A. Ignatiev, A. Morgado, V. M. Manquinho, I. Lynce, and J. Marques-Silva, "Progression in maximum satisfiability," in *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, ser. Frontiers in Artificial Intelligence and Applications, T. Schaub, G. Friedrich, and B. O'Sullivan, Eds., vol. 263. IOS Press, 2014, pp. 453–458. [Online]. Available: https://doi.org/10.3233/978-1-61499-419-0-453

[5] A. Ignatiev, A. Morgado, and J. Marques-Silva, "RC2: an efficient maxsat solver," *J. Satisf. Boolean Model. Comput.*, vol. 11, no. 1, pp. 53–64, 2019. [Online]. Available: https://doi.org/10.3233/SAT190116

[6] C. Ansótegui, M. L. Bonet, J. Gabas, and J. Levy, "Improving wpm2 for (weighted) partial maxsat," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2013, pp. 117–132.

[1]See https://github.com/FlorentAvellaneda/EvalMaxSAT