A short description of the solver EvalMaxSAT

Florent Avellaneda
Computer Research Institute of Montreal
Montreal, Canada
florent.avellaneda@gmail.com

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 Glucose [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].

The OLL algorithm considers all soft variables as hard and attempts to solve the formula. If the formula has no solution, then a conjunction of soft variables that cannot be satisfied (a *core*) is extracted. Each variable constituting this core is then *relaxed* (removed from the list of soft variables or incremented if it is a cardinality) and a new cardinality is added to the list of soft variables encoding the constraint "at most one variable from the core can be false". When the formula is finally satisfied, we obtain a MaxSAT assignment.

In practice, the size of the cores plays an important role in the performance of this algorithm. Indeed, the more variables the cores contain, the more expensive the encoding of cardinalities will be. Thus, once a core is found, a core minimization phase consists of removing unnecessary variables. Although heuristics are generally used to perform this minimization, this phase remains very expensive. EvalMaxSAT performs this minimization several times by calling the solver SAT with a limited number of conflicts. In addition, the algorithm used can easily be adapted to perform the minimization in parallel with the *core* searching.

II. Description

The algorithm used is a modification of the OLL algorithm (see Algorithm 1). The main modification is that when a core is found and minimized, new variables and constraints are not added to the SAT solver immediately. All these new constraints will be added only when the formula becomes satisfiable, or when finding a new solution takes too much time. By doing that, this algorithm tries to reduce the number of implications leading from cardinality to a soft variable.

A second modification made by the EvalMaxSAT solver is in the minimize function. Indeed, this function will perform several minimizations in order to obtain small cores. A first minimization is done by making successive calls to *solver(core)* where solver calls are limited to zero conflicts. Each call to the solver attempts to remove a literal from the core (a literal can be removed if the formulation remains unsatisfiable). After that, we apply the same algorithm with 1000 limited conflicts by considering the variables in differing orders.

Algorithm 1 (Pseudo-code of the sequential algorithm)

```
Input: A formula $\varphi$
 1: $cost \leftarrow extractAM1(\varphi)$
 2: while $true$ do
 3:     $(st, \varphi_c) \leftarrow SATSolver(\varphi)$
 4:     if $st = true$ then
 5:         $\varphi \leftarrow \varphi \cup \varphi_{tmp}$
 6:         $(st, \varphi_c) \leftarrow SATSolver(\varphi)$
 7:         if $st = true$ then
 8:             return $cost$
 9:         end if
10:     end if
11:     $\varphi_c \leftarrow minimize(\varphi, \varphi_c)$
12:     $k \leftarrow exhaust(\varphi, \varphi_c)$
13:     $cost \leftarrow cost + k$
14:     $\varphi \leftarrow relax(\varphi, \varphi_c)$
15:     $\varphi_{tmp} \leftarrow \varphi_{tmp} \cup createSum(\varphi_c, k)$
16: end while
```

III. Implementation Details

Extract AM1 Two algorithms are used to extract AtMost1 constraints from soft variables. The first one uses the mcqd library [6] to find the maximum clique in the incompatibility graph of the soft variables; the second one uses a heuristic.

Cardinality The Totalizer Encoding [7] is used to represent cardinalities. The implementation reuses the code from the PySAT's ITotalizer [8].

Exhaust After the minimization is performed, a *core exhaustion* [5] or *cover optimization* [9] is done.

Timeout Many timeouts are used to stop minimization when they take too much time.

IV. Multicore version

An interesting feature of the algorithm used is that it is very simple to parallelize it. Although the competition does not allow the multi-threaded calculation, this feature has been implemented in the solver **but disabled for the competition**. The architecture of the parallelized algorithm is depicted in Figure 1. The main thread looks for new cores to minimize and when it finds one, it removes all variables

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

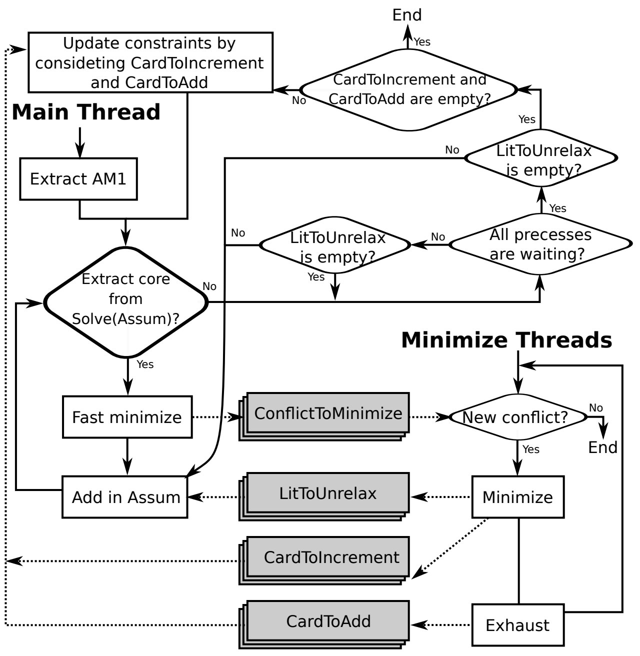

Fig. 1. Algorithm architecture

present in the core from the list of soft variables (Assum). Paralleling this, threads access the cores found by the main thread (ConflictToMinimize), minimize them and share new cardinalities (CardToAdd), unused variables (LitToUnrelax) and cardinalities to be incremented (CardToIncrement). Before searching for new cores, the main thread collects the variables that had previously been removed but were not used in any previous thread.

When the main thread no longer finds a core, all minimization threads have been completed and no variables are to be reconsidered as soft, then the main thread considers the new cardinalities to be added and incremented before restarting the *core* search. If there are no cardinalities to add or increment, then the search is complete and we get a MaxSAT assignment.

References

[1] G. Audemard, J. Lagniez, and L. Simon, "Improving glucose for incremental SAT solving with assumptions: Application to MUS extraction," in *Theory and Applications of Satisfiability Testing - SAT 2013 - 16th International Conference, Helsinki, Finland, July 8-12, 2013. Proceedings*, ser. Lecture Notes in Computer Science, M. Järvisalo and A. V. Gelder, Eds., vol. 7962. Springer, 2013, pp. 309–317. [Online]. Available: https://doi.org/10.1007/978-3-642-39071-5_23

[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] J. Konc and D. Janezic, "An improved branch and bound algorithm for the maximum clique problem," *proteins*, vol. 4, no. 5, 2007.

[7] R. Martins, S. Joshi, V. M. Manquinho, and I. Lynce, "Reflections on "incremental cardinality constraints for maxsat"," *CoRR*, vol. abs/1910.04643, 2019. [Online]. Available: http://arxiv.org/abs/1910.04643

[8] A. Ignatiev, A. Morgado, and J. Marques-Silva, "Pysat: A python toolkit for prototyping with SAT oracles," in *Theory and Applications of Satisfiability Testing - SAT 2018 - 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings*, ser. Lecture Notes in Computer Science, O. Beyersdorff and C. M. Wintersteiger, Eds., vol. 10929. Springer, 2018, pp. 428–437. [Online]. Available: https://doi.org/10.1007/978-3-319-94144-8_26

[9] 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.