Verifying Optimums of (Partial) Max-SAT Formulas

Mohamed Sami Cherif, Djamal Habet and Cyril Terrioux
Aix-Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
{mohamed-sami.cherif, djamal.habet, cyril.terrioux}@univ-amu.fr

***Abstract*—Checking whether a certain bound holds over a set of relaxation variables is a subproblem which often arises in the context of Maximum Satisfiability (Max-SAT) solving and particularly SAT-based solving. This document describes a collection of SAT instances that have been submitted to the 2021 SAT competition. These instances are derived from Max-SAT formulas whose soft clauses are augmented with relaxation variables. An At-Most-K constraint is then set over these variables to check the validity of a provided bound. We use this process to verify known solutions of (Partial) Max-SAT formulas.**

***Index Terms*—SAT, Max-SAT, At-Most-K constraint**

I. INTRODUCTION

The maximum satisfiability (Max-SAT) problem is an optimization extension of the satisfiability (SAT) problem. For a given formula in Conjunctive Normal Form (CNF), it consists in finding an assignment of the variables which maximizes the number of satisfied clauses. In Partial Max-SAT, clauses are divided into hard and soft clauses and the goal is to find an assignment that satisfies all hard clauses and maximizes the number of satisfied soft clauses. In recent years, Max-SAT solvers have achieved great breakthroughs by relying on SAT technology. In fact, Complete methods for this problem include SAT based approaches which iteratively call SAT solvers making them particularly efficient on industrial instances [5].

Checking whether a certain bound holds over a set of relaxation variables is a subproblem which often arises in the context of Maximum Max-SAT solving and particularly in SAT-based solving. For instance, Linear Search algorithms [2], [3] augment soft clauses with relaxation variables and add a CNF encoding over their sum to specify that the number of falsified soft clauses must be less than a given bound. A SAT solver is then iteratively called and the bound is increased (resp. decreased) until the formula becomes satisfiable (resp. unsatisfiable). Similarly to these algorithms, we rely on the fact that the optimum of a Max-SAT formula is the threshold in which the formula becomes satisfiable to verify the validity of a given optimum. To this end, given a Max-SAT formula and an integer value, we encode two SAT instances to check whether the given value is the threshold, i.e. the optimum of the formula.

II. VERIFYING (PARTIAL) MAX-SAT OPTIMUMS

Let $\phi = H \cup S$ be a Partial Max-Sat formula where H denotes the set of hard clauses and $S = \{c_1, ..., c_m\}$ the set of soft clauses. Let k be an integer value. We define the following formula:

$$\phi_k = H \cup \{c_i \cup \{r_i\} | c_i \in S\} \cup CNF(\sum_{1 \leq i \leq m} r_i \leq k)$$

where $r_1, .., r_m$ are new relaxation variables.

To verify that a given value o is the optimum of a CNF formula ϕ, it is sufficient to check that this value is the threshold in which the formula becomes satisfiable. To this end, we need to encode the formulas ϕ_{o-1} and ϕ_o and verify that ϕ_{o-1} is unsatisfiable and ϕ_o is satisfiable.

III. THE SUBMITTED BENCHMARK

We consider the Single machine scheduling family in the 2020 Max-SAT Evaluation described in [4]. We picked the 18 instances which were solved (by at least one solver) in the 2020 Max-SAT Evaluation and thus for which the optimum is known. For each instance ϕ, we encoded the formulas ϕ_{o-1} and ϕ_o. The submitted benchmark thus comprises 36 instances in total with 18 satisfiable instances and 18 unsatisfiable ones. We maintain the same naming conventions used in [4] except that we add '$\_sat$' or '$\_unsat$' to each formula indicating respectiviely whether it is satisfiable or unsatisfiable. We used the PySAT library [1] to add the cardinality constraints (i.e. At-Most-K constraints) over the relaxation variables. The encoding chosen for these constraints is the sequential counter encoding [6]. Finally, it is important to note that, once the constraints added, the clauses in the resulting formulas are shuffled.

REFERENCES

[1] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.

[2] Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. Qmaxsat: A partial max-sat solver system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 8, 01 2012.

[3] Daniel Le Berre and Anne Parrain. The sat4j library, release 2.2, system description. *Journal on Satisfiability Boolean Modeling and Computation*, 7:59–64, 07 2010.

[4] Xiaojuan Liao and Miyuki Koshimura. Description of Benchmarks on Single-Machine Scheduling. In Fahiem Bacchus, Jeremias Berg, Matti Järvisalo, and Rubens Martins, editors, *MaxSAT Evaluation 2020: Solver and Benchmark Descriptions*, Department of Computer Science Report Series B 2020-2, page 54. University of Helsinki, Department of Computer Science, 2020.

[5] Antonio Morgado, Federico Heras, Mark Liffiton, Jordi Planes, and Joao Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18, 10 2013.

[6] Carsten Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, pages 827–831, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.