RC2: a Python-based MaxSAT Solver

Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva
Faculty of Sciences, University of Lisbon, Portugal
`{aignatiev,ajmorgado,jpms}@ciencias.ulisboa.pt`

I. Introduction

RC2 is an open-source MaxSAT solver written in Python and based on the PySAT framework[1] [1]. It is designed to serve as a simple example of how SAT-based problem solving algorithms can be implemented using PySAT while sacrificing just a little in terms of performance. In this sense, RC2 can be seen as a solver prototype and can be made somewhat more efficient if implemented in a low-level language. RC2 is written from scratch and implements the OLLITI (or *RC2*, i.e. *relaxable cardinality constraints*) MaxSAT algorithm [2], [3] originally implemented in the MSCG MaxSAT solver [3], [4]. The RC2 algorithm proved itself efficient in the previous editions of the MaxSAT Evaluation: namely in 2014, 2015, and 2016 (see the results of the MSCG solver, which was one of the best complete MaxSAT solvers in the aforementioned competitions).

II. Description

RC2 supports *incrementally* a variety of SAT solvers provided by PySAT, and its competition version uses Glucose 3.0 [5] as an underlying SAT oracle. Two variants of the solver were submitted to the MaxSAT Evaluation 2018 including RC2-A and RC2-B. Both of these versions implement the same algorithm [2], [3] and share most of the techniques used [3]. Their major components and differences are briefly described below.

III. Variants of the Solver

The following heuristics are used by both solver variants submitted to the MaxSAT Evaluation 2018: incremental SAT solving [6], Boolean lexicographic optimization [7] and stratification [8] for weighted instances, unsatisfiable core exhaustion (originally referred to as cover optimization) [8].

Additionally, the following heuristic was used in both variants of RC2: given a set S of soft clauses, a number of subsets $S' \subseteq S$ were identified such that at most one soft clause in S' can be satisfied, i.e. $\sum_{c \in S'} c \leq 1$. Every subset S' can be treated as an unsatisfiable core of cost $|S'| - 1$, which can be represented as a single clause.

The only difference between the solver variants is the policy for unsatisfiable core minimization. In contrast to RC2-A, RC2-B applies heuristic unsatisfiable core minimization done with a simple deletion-based *minimal unsatisfiable subset* (*MUS*) extraction algorithm [9]. During the core minimization phase in RC2-B, all SAT calls are dropped after obtaining 1000 conflicts. Note that core minimization in RC2-B is disabled for large *plain* MaxSAT formulas, i.e. those having no hard clauses but more than 100000 soft clauses. The reason is that having this many soft clauses (and, thus, as many assumption literals) and no hard clauses is deemed to make SAT calls too expensive. Although core minimization is disabled in RC2-A, reducing the size of unsatisfiable cores can be still helpful for weighted instances due to the nature of the OLLITI/RC2 algorithm, i.e. because of the clause splitting applied to the clauses of an unsatisfiable core depending on their weight. Therefore, when dealing with weighted instances RC2-A *trims* unsatisfiable cores at most 5 times (e.g. see [3] for details) aiming at getting rid of unnecessary clauses. Note that core trimming is disabled in RC2-A for unweighted MaxSAT instances and it is not used in RC2-B at all.

[1] `http://pysathq.github.io`

IV. Availability

RC2 is distributed as a part of the PySAT framework, which is available under an MIT license at `https://github.com/pysathq/pysat`. It can also be installed as a Python package from PyPI:

```
pip install python-sat
```

The RC2 solver can be used as a standalone executable `rc2.py` and can also integrated into a complex Python-based problem solving tool, e.g. using the standard *import* interface of Python:

```
from pysat.examples import rc2
```

References

[1] A. Ignatiev, A. Morgado, and J. Marques-Silva, "PySAT: a Python toolkit for prototyping with SAT oracles," in *SAT*, 2018, to appear.
[2] A. Morgado, C. Dodaro, and J. Marques-Silva, "Core-guided MaxSAT with soft cardinality constraints," in *CP*, 2014, pp. 564–573.
[3] A. Morgado, A. Ignatiev, and J. Marques-Silva, "MSCG: Robust core-guided MaxSAT solving," *JSAT*, vol. 9, pp. 129–134, 2015.
[4] A. Ignatiev, A. Morgado, V. M. Manquinho, I. Lynce, and J. Marques-Silva, "Progression in maximum satisfiability," in *ECAI*, 2014, pp. 453–458.
[5] G. Audemard, J. Lagniez, and L. Simon, "Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction," in *SAT*, 2013, pp. 309–317.
[6] N. Eén and N. Sörensson, "Temporal induction by incremental SAT solving," *Electr. Notes Theor. Comput. Sci.*, vol. 89, no. 4, pp. 543–560, 2003.
[7] J. Marques-Silva, J. Argelich, A. Graca, and I. Lynce, "Boolean lexicographic optimization: algorithms & applications," *Annals of Mathematics and Artificial Intelligence (AMAI)*, vol. 62, no. 3-4, pp. 317–343, 2011.
[8] C. Ansótegui, M. L. Bonet, J. Gabàs, and J. Levy, "Improving WPM2 for (weighted) partial maxsat," in *CP*, 2013, pp. 117–132.
[9] J. M. Silva, "Minimal unsatisfiability: Models, algorithms and applications (invited paper)," in *ISMVL*, 2010, pp. 9–14.