Loandra: PMRES extended with preprocessing Entering MaxSAT Evaluation 2017

Jeremias Berg*, Tuukka Korhonen*, and Matti Järvisalo*
*HIIT, Department of Computer Science, University of Helsinki, Finland

I. Preliminaries

We briefly overview the Loandra MaxSAT solver as it participated in the 2017 MaxSAT evaluation. We assume familiarity with conjunctive normal form (CNF) formulas and weighted partial maximum satisfiability (MaxSAT). Treating a CNF formula as a set of clauses a MaxSAT instance consists of two CNF formulas, the hard clauses F_h and the soft clauses F_s, as well a weight function $w\colon F_s \to \mathbb{N}$.

Loandra makes extensive use of SAT-based preprocessing using labels [3], [4]. In order to enable sound application of most SAT-based preprocessing techniques for MaxSAT, each soft clause C is first extended with a fresh label variable l_C. Afterwards the preprocessor is invoked on the clauses in $F_h \cup \{C \vee l_C \mid C \in F_s\}$. During execution the preprocessor is forbidden from resolving on the added labels. Afterwards, the preprocessed instance is converted back to standard MaxSAT by treating all clauses in the preprocessor as hard and introducing a soft clause $(\neg l_C)$ with weight $w(C)$ for each added label.

II. Structure and execution of Loandra

The architecture of Loandra consists of two closely interleaved parts; the *solver* and the *preprocessor*. The solver is a reimplementation of the PMRES MaxSAT algorithm [15] extended with weight-aware core extraction (WCE) as described in [6]. The preprocessor is the recently proposed tool MaxPre [11], modified to support addition of clauses.

In more detail, whenever invoked on an MaxSAT instance (F_h, F_s, w) Loandra first preprocesses the input instance as described in [11]. Afterwards the preprocessed instance is extracted from the preprocessor and given to the solver. When initializing the solver, we follow [5] and do not introduce the soft clauses of form $(\neg l_C)$, but instead reuse the literals as assumption variables to be used in core extraction. Then the solver is invoked on the preprocessed instance. Except for the base algorithm, Loandra also uses stratification and clause hardening [1] as well as clause cloning through assumptions and reusing assumption variables as relaxation variables [6]. This guarantees that the working formula is only modified by adding clauses to it, making it possible to keep the state of the internal SAT solver throughout the solving process. During execution, all cardinality constraints added due to core relaxation are also added to the preprocessor as well. When the working formula is sufficiently modified, the the execution is switched back to the preprocessor which attempts to further simplify the modified formula, i.e. the original clauses with some clauses hardened and the new cardinality constraints. If the preprocessor is successful, the solver is reinitialized on the modified formula. Loandra terminates whenever the solver terminates. At this point the optimal model for the original formula can be reconstructed from the preprocessor.

III. Details on the competition builds

There are three version of Loandra competing in the 2017 MaxSAT evaluation.

- LOANDRA$_I$, which follows the description given above.
- LOANDRA$_P$, which only invokes its preprocessor once and then runs the solver on the preprocessed instance.
- LOANDRA$_S$, which only uses its solver, not invoking the preprocessor at all.

These solvers are built on top of the open source Open-WBO system [13], [14] and use Glucose 3.0 [2] as the internal SAT solver. All preprocessing calls are done with label matching turned off, with the SKIPTECHNIQUE parameter set to 20, and using a technique loop with blocked clause elimination [10], unit propagation, bounded variable elimination [8], subsumption elimination, self-subsuming resolution [8], [9], [12] as well as group-subsumed label elimination [7], [11] and binary core removal [11]. See [11] for more details on the settings of MaxPre. The additional preprocessing step is attempted whenever more than 500 clauses have been hardened since the preprocessing attempt.

IV. Compilation and Usage

Building and using Loandra resembles building and using Open-WBO. A statically linked version of Loandra in release mode can be built from the code by first running MAKE LIB in the maxpre subfolder and then MAKE RS in the base folder. One significant difference to Open-WBO is the need of C++11 features for building Loandra.

After building, Loandra can be invoked from the terminal. Except for the formula file, Loandra accepts a number of command line arguments; the flag "-inpr" enables execution following LOANDRA$_I$, the flag "-pre" enables execution following LOANDRA$_P$ and the flag "-printM" prints out the optimal model of the instance, and not only its cost. The rest of the flags resemble the flags accepted by Open-WBO; invoke ./loandra_static –help-verb for more information.

References

[1] C. Ansótegui, M. L. Bonet, J. Gabàs, and J. Levy, "Improving SAT-Based Weighted MaxSAT Solvers," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 7514. Springer, 2012, pp. 86–101.

[2] G. Audemard, J.-M. Lagniez, and L. Simon, "Improving Glucose for Incremental SAT Solving with Assumptions: Application to MUS Extraction," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 7962. Springer, 2013, pp. 309–317.

[3] A. Belov, M. Järvisalo, and J. Marques-Silva, "Formula Preprocessing in MUS Extraction," in *Proc. TACAS*, ser. Lecture Notes in Computer Science, vol. 7795. Springer, 2013, pp. 108–123.

[4] A. Belov, A. Morgado, and J. Marques-Silva, "SAT-Based Preprocessing for MaxSAT," in *Proc. LPAR-19*, ser. Lecture Notes in Computer Science, vol. 8312. Springer, 2013, pp. 96–111.

[5] J. Berg, P. Saikko, and M. Järvisalo, "Improving the Effectiveness of SAT-Based Preprocessing for MaxSAT," in *Proc. IJCAI*. AAAI Press, 2015, pp. 239–245.

[6] J. Berg and M. Järvisalo, "Weight-Aware Core Extraction in SAT-Based MaxSAT Solving," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. ????, pp. ??? – ??? (to appear).

[7] J. Berg, P. Saikko, and M. Järvisalo, "Subsumed label elimination for maximum satisfiability," in *Proc ECAI*, ser. Frontiers in Artificial Intelligence and Applications, G. A. Kaminka, M. Fox, P. Bouquet, E. Hüllermeier, V. Dignum, F. Dignum, and F. van Harmelen, Eds., vol. 285. IOS Press, 2016, pp. 630–638.

[8] N. Eén and A. Biere, "Effective Preprocessing in SAT Through Variable and Clause Elimination," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 3569. Springer, 2005, pp. 61–75.

[9] J. Groote and J. Warners, "The propositional formula checker Heer-Hugo," *Journal of Automated Reasoning*, vol. 24, no. 1/2, pp. 101–125, 2000.

[10] M. Järvisalo, A. Biere, and M. Heule, "Blocked Clause Elimination," in *Proc. TACAS*, ser. Lecture Notes in Computer Science, vol. 6015. Springer, 2010, pp. 129–144.

[11] T. Korhonen, J. Berg, P. Saikko, and M. Järvisalo, "MaxPre: An Extended MaxSAT Preprocessor." in *Proc. SAT*, ser. Lecture Notes in Computer Science, S. Gaspers and T. Walsh, Eds., vol. ????, pp. ??? – ??? (to appear).

[12] K. Korovin, "iProver – an instantiation-based theorem prover for first-order logic," in *Proc. IJCAI*, ser. Lecture Notes in Computer Science, vol. 5195. Springer, 2008, pp. 292–298.

[13] R. Martins, S. Joshi, V. Manquinho, and I. Lynce, "Incremental Cardinality Constraints for MaxSAT," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 8656. Springer, 2014, pp. 531–548.

[14] R. Martins, V. Manquinho, and I. Lynce, "Open-WBO: A Modular MaxSAT Solver," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 8561. Springer, 2014, pp. 438–445.

[15] N. Narodytska and F. Bacchus, "Maximum Satisfiability Using Core-Guided MaxSAT Resolution," in *Proc. AAAI*. AAAI Press, 2014, pp. 2717–2723.