Loandra in the 2020 MaxSAT Evaluation

Jeremias Berg*, Emir Demirović†, Peter Stuckey‡
*HIIT, Department of Computer Science, University of Helsinki, Finland
†Delft University of Technology, The Netherlands
‡Monash University, Australia

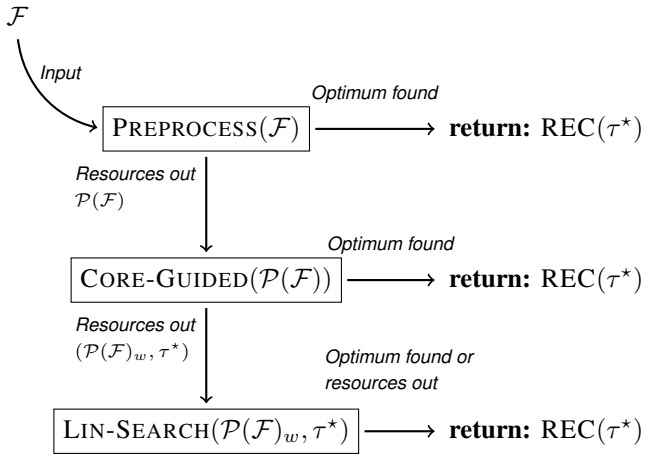

Fig. 1: The structure of Loandra.

I. Preliminaries

We briefly overview the Loandra MaxSAT-solver as it participated in the incomplete track of the 2020 MaxSAT Evaluation, focusing especially on the differences between the 2019 and 2020 versions, more detailed descriptions can be found in [4], [10]. Loandra owes much of its existence to Open-WBO [11], we thank the developers of Open-WBO for their work .

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 $\mathcal{F}$ consists of two CNF formulas, the hard clauses F_h and the soft clauses F_s, as well a weight w_c associated with each $C \in F_s$. A solution to $\mathcal{F}$ is an assignment τ that satisfies F_h. The cost of a solution τ is the sum of weights of the soft clauses falsified by τ. An optimal solution is one with minimum cost over all solutions. An unsatisfiable core κ of $\mathcal{F}$ is a subset of soft clauses s.t. $F_h \wedge \kappa$ is unsatisfiable.

II. Structure of Loandra

Figure 1 overviews the structure of Loandra. In short, Loandra implements core-boosted linear search [4] augmented with tightly integrated MaxSAT preprocessing [3], [9], [10], [2]. More specifically, Loandra consists of three main components: a) Preprocessing, b) Core-guided search, c) Linear search.

a) Preprocessing: On input $\mathcal{F}$, the execution starts by invoking the MaxPre [9] preprocessor on $\mathcal{F}$ using the standard techniques of MaxPre *except for blocked clause elimination (BCE)*. The reason we are not using BCE is that, as detailed in [10], intermediate solutions to instances preprocessed with BCE are more prone to having their costs miss-interpreted by the core-guided and linear search components of Loandra. If MaxPre can not compute the optimal solution to $\mathcal{F}$, the preprocessed instance $\mathcal{P}(\mathcal{F})$ is handed to the core guided phase (CORE-GUIDED in Figure 1), reusing the assumption variables introduced during preprocessing [3].

b) Core-guided search: The core-guided phase is unchanged from the 2019 version; as the instantiation of the core-guided algorithm, we use a reimplementation of PMRES [12] extended with weight aware core extraction (WCE) [5] and clause hardening. If CORE-GUIDED is able to find an optimal solution τ to $\mathcal{P}(\mathcal{F})$, an optimal solution $\text{REC}(\tau)$ to $\mathcal{F}$ is reconstructed and returned. Otherwise i.e. if the core-guided phase runs out of time, the final working instance $\mathcal{P}(\mathcal{F})_w$ and τ^*, the best found solution to it is handed to the linear search component LIN-SEARCH.

c) Linear search: LIN-SEARCH, the linear search phase of Loandra is an implementation of the SAT/UNSAT linear search algorithm [6], extended with solution guided phase saving and varying resolution in the style of LinSBPS [7]. The component is for the most part the same as in the 2019 version. The main difference is, that in the start of each resolution, the currently best known solution τ^* is minimized in order to alleviate the missinterpretation of costs that might happen due to preprocessing in the context of incomplete solving [10].

More specifically, let $\mathcal{P}(\mathcal{F})_w = \mathcal{P}(\mathcal{F}) \cup \text{CARD}$ be the working instance of LIN-SEARCH where $\mathcal{P}(\mathcal{F})$ is the preprocessed instance computed by MaxPre and CARD are the constraints added in the core-guided phase. At the start of each resolution, LIN-SEARCH computes a set $\mathcal{B}_s$ of blocking variables and an upper bound UB over which the PB constraint is built. The upper bound is computed based on τ^*, the current best known solution to $\mathcal{P}(\mathcal{F})_w$. However, as shown in [10], there can be a significant difference between $\text{COST}(\mathcal{P}(\mathcal{F}), \tau^*)$, the cost of τ^* w.r.t to $\mathcal{P}(\mathcal{F})$, and $\text{COST}(\mathcal{F}, \text{REC}(\tau^*))$, the cost of the solution to $\mathcal{F}$ reconstructed from τ^*. This difference might result UB, and as consequence the whole PB constraint, being much larger than actually required. In order to alleviate this issue LIN-SEARCH uses a simple, procedure that iteratively fixes all variables in $\mathcal{B}_s$ in the following manner. In each iteration, all variables in $\mathcal{P}(\mathcal{F})$

are fixed to the polarities that they are assigned to by τ^*. Additionally an unfixed variable $b \in \mathcal{B}_s$ is fixed to false (i.e. to not incur cost). Then the SAT solver is used to extend these fixings into a satisfying assignment of $\mathcal{P}(\mathcal{F})_w$. If such an assignment τ_b^* can be found, that assignment will have $\text{COST}(\mathcal{P}(\mathcal{F}), \tau_b^*) < \text{COST}(\mathcal{P}(\mathcal{F}), \tau^*)$ while also agreeing with τ^* on all variables in $\mathcal{P}(\mathcal{F})$. The variable b is then fixed to false in subsequent iterations. Otherwise, the variable b is fixed to true in subsequent iterations. Notice that, due to the nature of constraints added by CORE-GUIDED, each individual SAT-solver call is solvable by unit propagation alone (the constraints in CARD are basically cardinality constraints). Even so, preliminary experiments showed that the minimization procedure is too expensive to run for each new solution found, which is why we restrict it to once per resolution.

The linear phase runs until either finding an optimal solution, or running out of time, at which point a reconstruction $\text{REC}(\tau^*)$ of the currently best known solution τ^* to $\mathcal{P}(\mathcal{F})_w$ is returned. Notice that the reconstruction of a solution happens only once, we use the standard, linear time, reconstruction algorithm as implemented by MaxPre.

III. IMPLEMENTATION DETAILS

All algorithms are implemented on top of the publicly available Open-WBO system [11] using Glucose 4.1 [1] as the back-end SAT solver. In order to minimize I/O overhead, we make direct use of the preprocessor interface offered by MaxPre. The linear search algorithm uses the generalized totalizer encoding [8] to convert the PB constraints needed in linear search to CNF. In the evaluation, we set a 30s time limit for the preprocessing phase and a 30 second time limit for the core-guided phase. These limits were chosen based on preliminary experiments. On weighted instances, the core-guided phase is also terminated when the stratification bound would be lowered to 1. On unweighted instances the phase is terminated at the latest after extracting one set of disjoint cores.

IV. COMPILATION AND USAGE

Building and using Loandra resembles building and using Open-WBO. Before building loandra, the maxpre library needs to be built by invoking MAKE LIB in the maxpre subfolder. Afterwards, a statically linked version of Loandra in release mode can be built by running MAKE RS in the base folder.

After building, Loandra can be invoked from the terminal. Except for the formula file, Loandra accepts a number of command line arguments: the flag *-pmreslin-cglim* sets the maximum time that the core-guided phase can run for (in seconds). The rest of the flags resemble the flags accepted by Open-WBO; invoke ./loandra_static –help-verb for more information.

REFERENCES

[1] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern sat solvers," in *Proc IJCAI*. Morgan Kaufmann Publishers Inc., 2009, pp. 399–404.

[2] 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.

[3] 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.

[4] J. Berg, E. Demirovic, and P. J. Stuckey, "Core-boosted linear search for incomplete maxsat," in *CPAIOR*, ser. Lecture Notes in Computer Science, vol. 11494. Springer, 2019, pp. 39–56.

[5] J. Berg and M. Järvisalo, "Weight-aware core extraction in SAT-based MaxSAT solving," in *Proc. CP*, ser. Lecture Notes in Computer Science, 2017, to appear.

[6] D. L. Berre and A. Parrain, "The sat4j library, release 2.2," *J. Satisf. Boolean Model. Comput.*, vol. 7, no. 2-3, pp. 59–6, 2010. [Online]. Available: https://satassociation.org/jsat/index.php/jsat/article/view/82

[7] E. Demirovic and P. J. Stuckey, "Techniques inspired by local search for incomplete maxsat and the linear algorithm: Varying resolution and solution-guided search," in *CP*, ser. Lecture Notes in Computer Science, vol. 11802. Springer, 2019, pp. 177–194.

[8] S. Joshi, R. Martins, and V. M. Manquinho, "Generalized totalizer encoding for pseudo-boolean constraints," in *Proc. CP*, ser. LNCS, vol. 9255, 2015, pp. 200–209.

[9] T. Korhonen, J. Berg, P. Saikko, and M. Järvisalo, "Maxpre: An extended maxsat preprocessor," in *SAT*, ser. Lecture Notes in Computer Science, vol. 10491. Springer, 2017, pp. 449–456.

[10] M. Leivo, J. Berg, and M. Järvisalo, "Preprocessing in incomplete maxsat solving," in *Proc ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. ???? IOS Press, 2020, p. (to appear).

[11] 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.

[12] N. Narodytska and F. Bacchus, "Maximum satisfiability using core-guided MaxSAT resolution," in *Proc. AAAI*. AAAI Press, 2014, pp. 2717–2723.