Loandra in the 2022 MaxSAT Evaluation

Jeremias Berg
Department of Computer Science, University of Helsinki, Finland

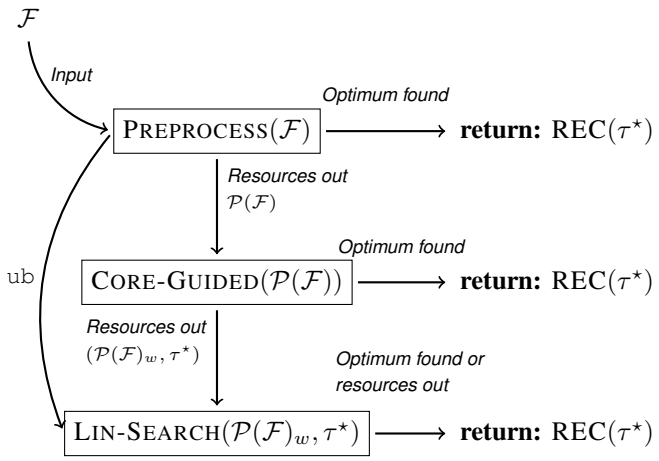

Fig. 1: The structure of Loandra.

I. Preliminaries

We briefly overview the any-time Loandra MaxSAT-solver as it participated in the incomplete track of the 2022 MaxSAT Evaluation, focusing especially on the differences to the 2019 and 2020 versions. All of the new changes to Loandra relate to the preprocessing phase of the algorithm. In particular, the solver now employs a recent extension of MaxPRE (named MaxPRE 2.0) capable of stronger reasoning as well as outputting an upper bound $\texttt{ub}$ on the optimal cost. More detailed descriptions can be found in [4], [11], [10].

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 $\text{COST}(\mathcal{F}, \tau)$ 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.

Loandra is implemented on top of Open-WBO [12]. We thank the developers of Open-WBO for their work.

II. Structure of Loandra

Figure 1 overviews the structure of Loandra. The solver implements core-boosted linear search [4] augmented with tightly integrated MaxSAT preprocessing [3], [10], [11], [2]. More specifically, Loandra consists of three main components: a) *Preprocessing*, b) *Core-guided search* and c) *Linear search*.

a) Preprocessing: On input $\mathcal{F}$, the execution starts by invoking the MaxPre 2.0 [10] preprocessor on $\mathcal{F}$. MaxPre 2.0 is run with the technique string `[u]#[uvsrgVGc]`, enforcing a 30s time-limit on and a skip technique value of 20. In more detail, the preprocessor runs the same "base" techniques as in previous years (unit propagation, bounded variable elimination, subsumption elimination, self-subsuming resolution, group subsumed label elimination and binary core removal) as well as the so called intrinsic at-most-one and TrimMaxSAT techniques [8], [15]. The TrimMaxSAT technique is extended to all literals rather than only literals appearing in soft clauses.

In addition to the more expressive preprocessing rules, another novelty of applying MaxPRE 2.0 is the possibility of obtaining and upper bound $\texttt{ub}$ on $\text{COST}(\mathcal{F})$. The bound is supplied to the linear search phase. Unless MaxPre can compute an optimal solution to $\mathcal{F}$, the preprocessed instance $\mathcal{P}(\mathcal{F})$ is then handed to the core guided phase, reusing the assumption variables introduced during preprocessing [3].

b) Core-guided search: CORE-GUIDED, the core-guided phase is unchanged from previous versions of Loandra. As the instantiation of the core-guided algorithm, we use a reimplementation of PMRES [14] 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 the final working instance $\mathcal{P}(\mathcal{F})_w$ and the best found solution τ^* are handed to the linear search component.

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. As the pseudo-Boolean encoding, we use the so called generalized totalizer [9]. The initial bound $B = \min\{\texttt{ub}, \text{COST}(\mathcal{F}, \tau^*)\}$ on PB-encoding is set to the minimum of the upper bound found by the preprocessor and the cost of the best solution found by the core-guided phase. Note that the linear search phase operates on the working instance of the core-guided search. As such, the range over which it searches is $[\texttt{lb}, B]$ where $\texttt{lb}$ is the lower bound obtained by the core-guided phase. The lower bound is implicitly maintained in the transformed formula, meaning that in practice, the PB constraint is built over the range $[0, B - \texttt{lb}]$.

In the beginning of each resolution (i.e. invocation of linear search on a subset of the soft clauses), the 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 [11]. The minimization procedure re-

sembles ideas proposed in MaxSAT solving algorithms based on bit-vector optimization [13]. In short, the procedure loops over all literals in the objective funciton, attempting to assign an increasing number of them to false (i.e. to not incur cost).

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 [12] 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 [9] 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. 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 and MaxPRE; invoke ./loandra_static –help-verb for more information.

V. Acknowledgments

The algorithmic techniques underlying Loandra have been developed in collaboration with a number of different people. The initial work on the solver was done together with Emir Demirović and Peter Stuckey [4]. Other contributors to Loandra include Matti Järvisalo, Marcus Leivo, Tuukka Korhonen, and Hannes Ihalainen [11], [10]. The primary developer is supported by the Academy of Finland under grant 342145. My sincerest thanks to everyone who has contributed to the algorithmic ideas underlying Loandra.

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] 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.

[9] 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.

[10] 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.

[11] M. Leivo, J. Berg, and M. Järvisalo, "Preprocessing in incomplete maxsat solving," in *ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. 325. IOS Press, 2020, pp. 347–354.

[12] 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.

[13] A. Nadel, "Anytime weighted maxsat with improved polarity selection and bit-vector optimization," in *FMCAD*. IEEE, 2019, pp. 193–202.

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

[15] T. Paxian, P. Raiola, and B. Becker, "On preprocessing for weighted maxsat," in *VMCAI*, ser. Lecture Notes in Computer Science, vol. 12597. Springer, 2021, pp. 556–577.