Maxino

Mario Alviano
Department of Mathematics and Computer Science
University of Calabria
87036 Rende (CS), Italy
Email: alviano@mat.unical.it

***Abstract*—Maxino is based on the k-ProcessCore algorithm, a parametric algorithm generalizing OLL, ONE and PMRES. Parameter k is dynamically determined for each processed unsatisfiable core by a function taking into account the size of the core. Roughly, k is in $O(\log n)$, where n is the size of the core. Satisfiability of propositional theories is checked by means of a pseudo-boolean solver extending Glucose 4.1 (single thread).**

A Very Short Description of the Solver

The solver MAXINO is build on top of the SAT solver GLUCOSE [7] (version 4.1). MaxSAT instances are normalized by replacing non-unary soft clauses with fresh variables, a process known as *relaxation*. Specifically, the relaxation of a soft clause ϕ is the clause $\phi \vee \neg x$, where x is a variable not occurring elsewhere; moreover, the weight associated with clause ϕ is associated with the soft literal x. Hence, the normalized input processed by MAXINO comprises hard clauses and soft literals, so that the computational problem amounts to maximize a linear function, which is defined by the soft literals, subject to a set of constraints, which is the set of hard clauses.

The algorithm implemented by MAXINO to address such a computational problem is based on unsatisfiable core analysis, and in particular takes advantage of the following *invariant*: A model of the constraints that satisfies all soft literals is an optimum model. The algorithm then starts by searching such a model. On the other hand, if an inconsistency arises, the unsatisfiable core returned by the SAT solver is analyzed. The analysis of an unsatisfiable core results into new constraints and new soft literals, which replace the soft literals involved in the unsatisfiable core. The new constraints are essentially such that models satisfying all new soft literals actually satisfy all but one of the replaced soft literals. Since there is no model that satisfies all replaced soft literals, it turns out that the invariant is preserved, and the process can be iterated.

Specifically, the algorithm implemented by MAXINO is K, based on the k-ProcessCore procedure introduced by Alviano et al. [2]. It is a parametric algorithm generalizing OLL [3], ONE [2] and PMRES [8]. Intuitively, for an unsatisfiable core $\{x_0, x_1, x_2, x_3\}$, ONE introduces the following constraint:

$$\begin{array}{l} x_0 + x_1 + x_2 + x_3 + \neg y_1 + \neg y_2 + \neg y_3 \geq 3 \\ y_1 \rightarrow y_2 \quad y_2 \rightarrow y_3 \end{array}$$

where y_1, y_2, y_3 are fresh variables (the new soft literals that replace x_0, x_1, x_2, x_3). OLL introduces the following constraints (the first immediately, the second if a core containing y_1 is subsequently found, and the third if a core containing y_2 is subsequently found):

$$\begin{array}{l} x_0 + x_1 + x_2 + x_3 + \neg y_1 \geq 3 \\ x_0 + x_1 + x_2 + x_3 + \neg y_2 \geq 2 \\ x_0 + x_1 + x_2 + x_3 + \neg y_3 \geq 1 \end{array}$$

Concerning PMRES, it introduces the following constraints:

$$\begin{array}{ll} x_0 \vee x_1 \vee \neg y_1 & z_1 \leftrightarrow x_0 \wedge x_1 \\ z_1 \vee x_2 \vee \neg y_2 & z_2 \leftrightarrow z_1 \wedge x_2 \\ z_2 \vee x_3 \vee \neg y_3 & \end{array}$$

which are essentially equivalent to the following constraints:

$$\begin{array}{ll} x_0 + x_1 + \neg z_1 + \neg y_1 \geq 2 & z_1 \rightarrow y_1 \\ z_1 + x_2 + \neg z_2 + \neg y_2 \geq 2 & z_2 \rightarrow y_2 \\ z_2 + x_3 \qquad\quad + \neg y_3 \geq 1 & \end{array}$$

where y_1, y_2, y_3 are fresh variables (the new soft literals that replace x_0, x_1, x_2, x_3), and z_1, z_2 are fresh auxiliary variables.

Algorithm K, instead, introduces a set of constraints of bounded size, where the bound is given by the chosen parameter k, and is specifically $2 \cdot (k+1)$. ONE, which is essentially a smart encoding of OLL, is the special case for $k = \infty$, and PMRES is the special case for $k = 1$. For the example unsatisfiable core, another possibility is $k = 2$, which would results in the following constraints:

$$\begin{array}{lll} x_0 + x_1 + x_2 + \neg z_1 + \neg y_1 + \neg y_2 \geq 3 & z_1 \rightarrow y_1 & y_1 \rightarrow y_2 \\ z_1 + x_3 \qquad\qquad\quad + \neg y_3 \qquad\quad \geq 1 & & \end{array}$$

In this version of MAXINO, the parameter k is dynamically determined based on the size of the analyzed unsatisfiable core: $k \in O(\log n)$, where n is the size of the core.

The analysis of unsatisfiable core is preceded by a *shrink* procedure [1]. Specifically, a reiterated progression search is performed on the unsatisfiable core returned by the SAT solver. Such a procedure significantly reduces the size of the unsatisfiable core, even if it does not necessarily returns an unsatisfiable core of minimal size. Additionally, satisfiability checks performed during the shrinking process are subject to a budget on the number of conflicts, so that the overhead due to hard checks is limited. Specifically, the budget is set to the number of conflicts arose in the satisfiability check that lead to detecting the unsatisfiable core; if such a number is less than 1000 (one thousand), the budget is raised to 1000. The budget is divided by 2 every time the progression is reiterated.

Weighted instances are handled by *stratification* and introducing *remainders* [4]–[6]. Specifically, soft literals are partitioned in strata depending on the associated weight. Initially, only soft literals of greatest weight are considered, and soft literals in the next stratum are added only after a model satisfying all considered soft literals is found. When an unsatisfiable core is found, the weight of all soft literals in the core is decreased by the weight associated with last added stratum. Soft literals whose weight become zero are not considered soft literals anymore.

Finally, a preprocessing step is performed on unweighted instances, which essentially iterates on all hard clauses of the input theory, sorted by length, and checks whether they already witness some unsatisfiable core. Specifically, an hard clause witnesses an unsatisfiable core if all literals in the clause are the complement of a soft literal. If this is the case, the unsatisfiable core is analyzed immediately. The rationale for such a preprocessing step is that hard clauses in the input theory are often small, and the smaller the better for the unsatisfiable core based algorithms.

REFERENCES

[1] Mario Alviano and Carmine Dodaro. Anytime answer set optimization via unsatisfiable core shrinking. *TPLP*, 16(5-6):533–551, 2016.

[2] Mario Alviano, Carmine Dodaro, and Francesco Ricca. A maxsat algorithm using cardinality constraints of bounded size. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2677–2683. AAAI Press, 2015.

[3] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub. Unsatisfiability-based optimization in clasp. In *28th International Conference on Logic Programming*, pages 211–221, Budapest, Hungary, September 2012.

[4] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Solving (weighted) partial maxsat through satisfiability testing. In *SAT 2009*, pages 427–440, Swansea, UK, June 2009. Springer.

[5] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196(0):77–105, March 2013.

[6] Josep Argelich, Inês Lynce, and João P. Marques Silva. On solving boolean multilevel optimization problems. In *21st International Joint Conference on Artificial Intelligence*, pages 393–398, Pasadena, California, July 2009. IJCAI Organization.

[7] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *21st International Joint Conference on Artificial Intelligence*, pages 399–404, Pasadena, California, July 2009. IJCAI Organization.

[8] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2717–2723, Québec City, Canada, July 2014. AAAI Press.