On Optimization Modulo Theories, MaxSMT and Sorting Networks

Roberto Sebastiani and Patrick Trentin

DISI, University of Trento, Italy

Abstract. Optimization Modulo Theories (OMT) is an extension of SMT which allows for finding models that optimize given objectives. (Partial weighted) MAXSMT –or equivalently OMT with Pseudo-Boolean objective functions, OMT+PB– is a very-relevant strict subcase of OMT. We classify existing approaches for MAXSMT or OMT+PB in two groups: MAXSAT-*based* approaches exploit the efficiency of state-of-the-art MAXSAT solvers, but they are specific-purpose and not always applicable; *OMT-based* approaches are general-purpose, but they suffer from intrinsic inefficiencies on MAXSMT/OMT+PB problems.
We identify a major source of such inefficiencies, and we address it by enhancing OMT by means of bidirectional sorting networks. We implemented this idea on top of the OPTIMATHSAT OMT solver. We run an extensive empirical evaluation on a variety of problems, comparing MAXSAT-based and OMT-based techniques, with and without sorting networks, implemented on top of OPTIMATHSAT and νZ. The results support the effectiveness of this idea, and provide interesting insights about the different approaches.

1 Introduction

Satisfiability Modulo Theories (SMT) is the problem of deciding the satisfiability of first-order formulas with respect to first-order theories [28, 10] (e.g., the theory of linear arithmetic over the rationals, $\mathcal{LRA}$). In the last decade, SMT solvers –powered by very efficient Conflict-Driven-Clause-Learning (CDCL) engines for Boolean Satisfiability [20] combined with a collection of $\mathcal{T}$-Solvers, each one handling a different theory $\mathcal{T}$– have risen to be a pervasive and indispensable tool for dealing with many problems of industrial interest, e.g. formal verification of hardware and software systems, resource planning, temporal reasoning and scheduling of real-time embedded systems.

Optimization Modulo Theories (OMT) is an extension of SMT, which allows for finding models that make a given objective optimum through a combination of SMT and optimization procedures [25, 15, 16, 29, 30, 18, 19, 13, 14, 32, 31]. Latest advancements in OMT have further broadened its horizon by making it incremental [13, 32] and by supporting objectives defined in other theories than linear arithmetic (e.g. Bit-Vectors) [13, 14, 21]. Moreover, OMT has been extended with the capability of handling multiple objectives at the same time either independently (aka boxed optimization) or through their linear, min-max/max-min, lexicographic or Pareto combination [13, 14, 32].

We focus on an important strict sub-case of OMT, (partial weighted)[1] MAXSMT –or equivalently OMT with Pseudo-Boolean (PB) objective functions [26], OMT+PB–

[1] Hereafter, when speaking of MAXSAT or MAXSMT, we keep "partial weighted" implicit.

which is the problem of finding a model for an input formula which both satisfies all *hard* clauses and maximizes the cumulative weight of all *soft* clauses satisfied by the model [25, 15, 16]. We identify two main approaches which have been adopted in the literature (see related work). One specific-purpose approach, which we call MAXSAT-*based*, is to embed some MAXSAT engine within the SMT solver itself, and use it in combination with dedicated $\mathcal{T}$-*solvers* [8, 13, 14] or with SMT solvers used as black-boxes [16]. One general-purpose approach, which we call *OMT-based*, is to encode MAXSMT/OMT+PB into general OMT with linear-real-arithmetic cost functions [30].

We compare the two approaches and notice the following facts.

The MAXSAT-based approach can exploit the efficiency of state-of-the-art MAXSAT procedures and solvers. Unfortunately it suffers from some limitations that make it impractical or inapplicable in some cases. First, to the best of our knowledge, available MAXSAT engines deal with integer weights only; some applications, e.g., *(Machine) Learning Modulo Theories, LMT* [34] –a hybrid Machine Learning approach in which OMT is used as an oracle for Support Vector Machines [34]– may require the weight of soft constraints to be high-precision rational values. [2] (In this context, it is preferable not to round the weights associated with soft-clauses since it affects the accuracy of the Machine Learning approach; also multiplying all rational coefficients for their lowest common multiple of the denominators is not practical, because such values tend to become huge.) Second, a MAXSAT engine cannot be directly used when dealing with an OMT problem with multiple-independent objectives that need to be optimized at the same time [19],[3] or when the objective function is given by combinations of PB and arithmetic terms –like, e.g., for Linear Generalized Disjunctive Programming problems [27, 30] or LMT problems [34].

The OMT-based approach does not suffer from the above limitations, because it exploits the infinite-precision linear-arithmetic package on the rationals of OMT solvers, and it treats PB functions as any other arithmetic functions. Nevertheless this approach may result in low performances when dealing with MAXSMT/OMT+PB problems.

We analyze the latter fact and identify a major source of inefficiency by noticing that the presence of same-weight soft clauses entails the existence of symmetries in the solution space that may lead to a combinatorial explosion of the partial truth assignments generated by the CDCL engine during the optimization search. To cope with this fact, we introduce and describe a solution based on (bidirectional) sorting networks [33, 9, 6]. We implemented this idea within the OPTIMATHSAT OMT solver [31].

We run an empirical evaluation on a large amount of problems comparing MAXSAT-based and OMT-based techniques, with and without sorting networks, implemented on top of OPTIMATHSAT [31] and νZ [14]. The results are summarized as follows.

(a) Comparing MAXSAT-based wrt. OMT-based approaches on problems where the former are applicable, it turns out that the former provide much better performances, in particular when adopting the maximum-resolution [22, 13] MAXSAT engine.

[2] For example, $\frac{1799972218749879}{2251799813685248}$ is a sample weight value from problems in [34].

[3] One could run a MAXSAT-based search separately on each objective, but doing this he/she would loose the benefits of boxed optimization, see [19, 13, 32].

(*b*) Evaluating the benefits of bidirectional sorting-network encodings, it turns out that they improve significantly the performances of OMT-based approaches, and often also of MAXSAT-based ones.

(*c*) Comparing νZ and OPTIMATHSAT, it turns out that the former performed better on MAXSAT-based approaches, whilst the latter performed seomtimes equivalently and sometimes significantly better on OMT-based ones, in particular when enhanced by the sorting-network encoding.

Related Work. The idea of MaxSMT and of optimization in SMT was first introduced by Nieuwenhuis & Oliveras [25], who presented a general logical framework of "SMT with progressively stronger theories" (e.g., where the theory is progressively strengthened by every new approximation of the minimum cost), and presented implementations for MaxSMT based on this framework. Cimatti et al. [15] introduced the notion of "Theory of Costs" $\mathcal{C}$ to handle Pseudo-Boolean (PB) cost functions and constraints by an ad-hoc "$\mathcal{C}$-solver" in the standard lazy SMT schema, and implemented a variant of MathSAT tool able to handle SMT with PB constraints and to minimize PB cost functions. Cimatti et al. [16] presented a "modular" approach for MaxSMT, combining a lazy SMT solver with a MaxSAT solver, where the SMT solver is used as an oracle generating $\mathcal{T}$-lemmas that are then learned by the MAXSAT solver so as to progressively narrow the search space toward the optimal solution.

Sebastiani and Tomasi [29, 30] introduced a wider notion of optimization in SMT, namely *Optimization Modulo Theories (OMT) with $\mathcal{LRA}$ cost functions*, OMT($\mathcal{LRA} \cup \mathcal{T}$), which allows for finding models minimizing some $\mathcal{LRA}$ cost term –$\mathcal{T}$ being some (possibly empty) stably-infinite theory s.t. $\mathcal{T}$ and $\mathcal{LRA}$ are signature-disjoint– and presented novel OMT($\mathcal{LRA} \cup \mathcal{T}$) tools which combine standard SMT with LP minimization techniques. ($\mathcal{T}$ can also be a combination of Theories $\bigcup_i \mathcal{T}_i$.) Eventually, OMT($\mathcal{LRA} \cup \mathcal{T}$) has been extended so that to handle costs on the integers, incremental OMT, multi-objective, and lexicographic OMT and Pareto-optimality [19, 18, 13, 32, 14, 31]. To the best of our knowledge only four OMT solvers are currently implemented: BCLT [18], νZ (aka Z3OPT) [13, 14], OPTIMATHSAT [32, 31], and SYMBA [19]. Remarkably, BCLT, νZ and OPTIMATHSAT currently implement also specialized procedures for MaxSMT, leveraging to SMT level state-of-the-art MaxSAT procedures; in addition, νZ features a Pseudo-Boolean *$\mathcal{T}$-solver* which can generate sorting circuits on demand for Pseudo-Boolean inequalities featuring sums with small coefficients when a Pseudo-Boolean inequality is used some times for unit propagation/conflicts [14, 12].

Content. The paper is structured as follows. §2 briefly reviews the background; §3 describes the source of inefficiency arising when MAXSMT is encoded in OMT as in [30]; §4 illustrates a possible solution based on bidirectional sorting networks; in §5 we provide empirical evidence of the benefits of this approach on two applications of OMT interest. §6 provides some conclusions with some considerations on the future work.

2 Background

We assume the reader is familiar with the main theoretical and algorithmic concepts in SAT and SMT solving (see [20, 10]). Optimization Modulo Theories (OMT) is an extension of SMT which addresses the problem of finding a model for an input formula φ which is optimal wrt. some objective function obj [25, 29]. The basic minimization scheme implemented in state-of-the-art OMT solvers, known as *linear-search* scheme [25, 29], requires solving an SMT problem with a solution space that is progressively tightened by means of unit linear constraints in the form $\neg(ub_i \leq obj)$, where ub_i is the value of obj that corresponds to the optimum model of the most-recently found truth assignment μ_i s.t. $\mu_i \models \varphi$. The ub_i value is computed by means of a specialized optimization procedure embedded within the $\mathcal{T}$*-solver* which, taken as input a pair $\langle \mu, obj \rangle$, returns the optimal value ub of obj for such μ. The OMT search terminates when such procedure finds that obj is unbounded or when the SMT search is UNSAT, in which case the latest value of obj (if any) and its associated model M_i is returned as optimal solution value. (Alternatively, binary-search schemes can also be used [29, 30].)

An important subcase of OMT is that of MAXSMT, which is a pair $\langle \varphi_h, \varphi_s \rangle$, where φ_h denotes the set of "hard" $\mathcal{T}$-clauses, φ_s is a set of positive-weighted "soft" $\mathcal{T}$-clauses, and the goal is to find the maximum-weight set of $\mathcal{T}$-clauses ψ_s, $\psi_s \subseteq \varphi_s$, s.t. $\varphi_h \cup \psi_s$ is $\mathcal{T}$-satisfiable [25, 15, 8, 16]. As described in [30], MAXSMT $\langle \varphi_h, \varphi_s \rangle$ can be encoded into a general OMT problem with a Pseudo-Boolean objective: first introduce a fresh Boolean variable A_i for each soft-constraint $C_i \in \varphi_s$ as follows

$$\varphi^* \overset{\text{def}}{=} \varphi_h \cup \bigcup_{C_i \in \varphi_s} \{(A_i \vee C_i)\}; \;\; obj \overset{\text{def}}{=} \sum_{C_i \in \varphi_s} w_i A_i \tag{1}$$

and then encode the problem into OMT as a pair $\langle \varphi, obj \rangle$ where φ is defined as

$$\varphi \overset{\text{def}}{=} \varphi^* \wedge \bigwedge_i ((\neg A_i \vee (x_i = w_i)) \wedge (A_i \vee (x_i = 0))) \wedge \tag{2}$$

$$\bigwedge_i ((0 \leq x_i) \wedge (x_i \leq w_i)) \wedge \tag{3}$$

$$(obj = \sum_i x_i), \quad x_i, \; obj \; fresh. \tag{4}$$

Notice that, although redundant from a logical perspective, the constraints in (3) serve the important purpose of allowing early-pruning calls to the $\mathcal{LRA}$-Solver (see [10]) to detect a possible $\mathcal{LRA}$ inconsistency among the current partial truth assignment over variables A_i and linear cuts in the form $\neg(ub \leq obj)$ that are pushed on the formula stack by the OMT solver during the minimization of obj. To this extent, the presence of such constraints improves performance significantly.

3 Problems with OMT-based Approaches

Consider first the case of a MAXSMT-derived OMT problem as in (1) s.t. all weights are identical, that is: let $\langle \varphi, obj \rangle$ be an OMT problem, where $obj = \sum_{i=1}^{n} w \cdot A_i$, where the A_is are Boolean variables, and let μ be a satisfiable truth assignment found by the OMT solver during the minimization of obj. Given $A_T = \{A_i | \mu \models A_i\}$ and $k = |A_T|$, then the upper bound value of obj in μ is $ub = w \cdot k$. As described in [29, 30], the OMT solver adds a unit clause in the form $\neg(ub \leq obj)$ in order to (1) remove the current truth assignment μ from the feasible search space and (2) seek for another μ' which improves the current upper-bound value ub. Importantly, the unit clause $\neg(ub \leq obj)$ does not only prune the current truth assignment μ from the feasible search space, but it

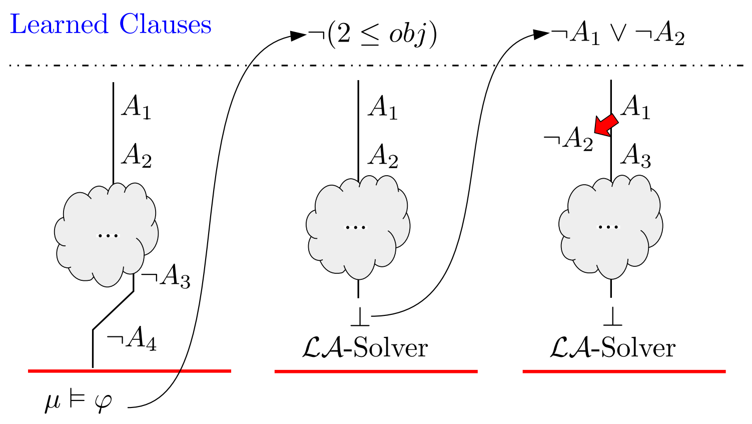


Fig. 1: A simple example of OMT search.

also makes inconsistent any other (partial) truth assignment μ' which sets exactly k (or more) A_i variables to True. Thus, each new unit clause in this form prunes $\gamma = \binom{n}{k}$ truth assignments from the search space, where γ is the number of of possible permutations of μ over the variables A_i. A dual case occurs when some lower-bound unit clause $\neg(obj \le lb)$ is learned (e.g., in a binary-search step, see [29]).

Unfortunately, the inconsistency of a truth assignment μ' which sets exactly k variables to True wrt. a unit clause $\neg(ub \le obj)$, where $ub = w \cdot k$, cannot be determined by simple Boolean Constraint Propagation (BCP). In fact, $\neg(ub \le obj)$ being a $\mathcal{LRA}$ term, the CDCL engine is totally oblivious to this inconsistency until when the $\mathcal{T}$*-solver* for linear rational arithmetic ($\mathcal{LRA}$-Solver) is invoked, and a conflict clause is generated. Therefore, since the $\mathcal{LRA}$-Solver is much more resource-demanding than BCP and it is invoked less often, it is clear that the performance of an OMT solver can be negatively affected when dealing with this kind of objectives.

Example 1. Figure 1 shows a toy example of OMT search execution over the pair $\langle \varphi, obj \rangle$, where φ is some SMT formula and $obj \overset{\text{def}}{=} \sum_{i=1}^{4} A_i$ (i.e., $w_i = 1$ for every i). We assume the problem has been encoded as in (2)-(4), so that the truth assignment $\mu_0 \overset{\text{def}}{=} \cup_{i=1}^{4}\{(0 \le x_i), (x_i \le 1)\} \cup \{(obj = \sum_{i=1}^{4} x_i)\}$ is immediately generated by BCP, and is part of all truth assignments generated in the search. In the first branch (left) a truth assignment $\mu \overset{\text{def}}{=} \mu_0 \cup \{A_1, (x_1 = 1), A_2, (x_2 = 1), \neg A_3, (x_3 = 0), \neg A_4, (x_4 = 0)\}$ is found s.t. $obj = 2$, resulting from the decisions A_1, A_2, $\neg A_3$ and $\neg A_4$. Then the unit clause $\neg(2 \le obj)$ is learned and the Boolean search is restarted in order to find an improved solution. In the second branch (center) A_1 and A_2 are decided, forcing by BCP the assignment $\mu' \overset{\text{def}}{=} \mu_0 \cup \{\neg(2 \le obj), A_1, (x_1 = 1), A_2, (x_2 = 1)\}$ which is $\mathcal{LRA}$-inconsistent. However, it takes a (possibly-expensive) intermediate call to the

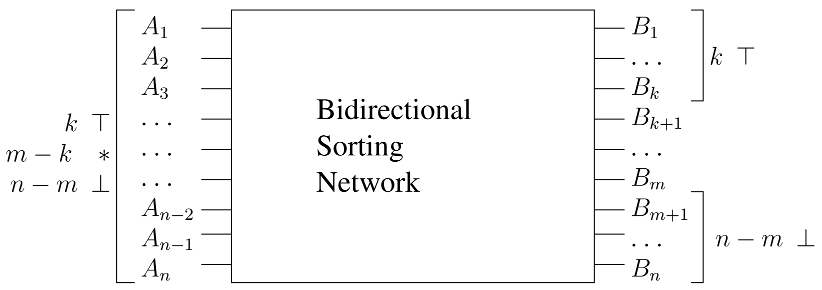


Fig. 2: The basic schema of a bidirectional sorting network.

$\mathcal{LRA}$-Solver to reveal such an inconsistency.[4] If so, a new conflict clause $\neg A_1 \vee \neg A_2$ is learned, forcing the solver to back-jump and toggle the value of A_2 (right). The search continues with the new decision A_3, which is again $\mathcal{LRA}$ inconsistent, causing a new conflict clause as before, and so on. In this way, the solver might uselessly enumerate and check all the up-to $\binom{4}{2}$ assignments that assign two A_i's to true and are consistent with φ, even though they are intrinsically incompatible with $\neg(2 \le obj)$. $\diamond$

The performance issue identified with the previous case example can be generalized to any objective obj in which groups of A_i's share the same weights:

$$obj = \tau_1 + ... + \tau_m, \tag{5}$$

$$\bigwedge_{j=1}^{m} \left((\tau_j = w_j \cdot \textstyle\sum_{i=1}^{k_j} A_{ji}) \ \wedge \ (0 \le \tau_j) \wedge (\tau_j \le w_j \cdot k_j) \right), \tag{6}$$

where the logically-redundant constraints $(0 \le \tau_j) \wedge (\tau_j \le w_j \cdot k_j)$ are added for the same reason as with (3).

4 Combining OMT with Sorting Networks

Notationally, the symbols $\top, \bot, *$ denote respectively "true", "false" and "unassigned". We represent truth assignment as sets (or conjunctions) of literals s.t. a positive [resp. negative] literal denotes the fact that the corresponding atom is assigned to $\top$ [resp. $\bot$]. Given a Boolean formula φ and two truth assignments μ, η on the atoms in φ, "$\langle \varphi, \mu \rangle \vdash_{\mathsf{bcp}} \eta$" denotes the fact that all literals in η are inferred by BCP on φ if all literals in μ are asserted. (Notice that "$\langle \varphi, \mu \rangle \vdash_{\mathsf{bcp}} \eta$" is stronger than "$\varphi \wedge \mu \models \eta$".)

When dealing with MAXSMT and OMT with PB objectives in the form

$$obj = w \cdot \textstyle\sum_{i=1}^{n} A_i \tag{7}$$

a solution for improving search efficiency is to reduce the dependency on the expensive $\mathcal{LRA}$-Solver by better exploiting BCP with the aid of Boolean *bidirectional sorting networks*.

[4] The fact that such call is actually performed depends on the early-pruning strategy implemented in the OMT solver; alternatively, a possibly-expensive $\mathcal{T}$-propagation step on the previous $\mathcal{LRA}$-Solver call has a similar effect. (See e.g. [28, 10].)

Definition 1. *Let* $\mathsf{SN}[\underline{A},\underline{B}]$ *be a CNF Boolean formula on* n *input Boolean variables* $\underline{A} \overset{\text{def}}{=} \{A_1,...,A_n\}$ *and* n *output Boolean variables* $\underline{B} \overset{\text{def}}{=} \{B_1,...,B_n\}$*, possibly involving also auxiliary Boolean variables which are not mentioned.*

We say that $\mathsf{SN}[\underline{A},\underline{B}]$ *is a* **bidirectional sorting network** *if and only if, for every* m *and* k *s.t.* $n \geq m \geq k \geq 0$ *and for every partial truth assignment* μ *s.t.* μ *assigns exactly* k *input variables* A_i *to* $\top$ *and* $n-m$ *variables* A_i *to* $\bot$*:*

$$\langle \mathsf{SN}[\underline{A},\underline{B}], \mu \rangle \vdash_{\mathsf{bcp}} \{\ \ B_1,...,\ \ B_k\}, \tag{8}$$

$$\langle \mathsf{SN}[\underline{A},\underline{B}], \mu \rangle \vdash_{\mathsf{bcp}} \{\neg B_{m+1},...,\neg B_n\}. \tag{9}$$

$$\langle \mathsf{SN}[\underline{A},\underline{B}], \mu \cup \{\neg B_{k+1}\} \rangle \vdash_{\mathsf{bcp}} \{\neg A_i \text{ } s.t. \text{ } A_i \text{ } unassigned \text{ } in \text{ } \mu\}, \tag{10}$$

$$\langle \mathsf{SN}[\underline{A},\underline{B}], \mu \cup \{\ \ B_m\} \rangle \vdash_{\mathsf{bcp}} \{A_i \text{ } s.t. \text{ } A_i \text{ } unassigned \text{ } in \text{ } \mu\}. \tag{11}$$

The schema of a bidirectional sorting network is depicted in Figure 2.

(8)-(9) state that the output values $\underline{B}$ of $\mathsf{SN}[\underline{A},\underline{B}]$ are propagated from the inputs $\underline{A}$ via BCP. (10)-(11) describe how assigning output variables $\underline{B}$ propagates back to input variables $\underline{A}$: (10) states that, when k A_i's are true and B_{k+1} is false, then all other A_i's are forced to be false by BCP; dually, (11) states that, when $n-m$ A_i's are false and B_m is true, then all other A_i's are forced to be true by BCP. (If any of the above BCP assignments conflicts with some previous assignment, a conflict is produced.)

Given an OMT problem $\langle \varphi, obj \rangle$, where obj is as in (7), and a Boolean formula $\mathsf{SN}[\underline{A},\underline{B}]$ encoding a bidirectional sorting network relation as in Definition 1, we extend φ in (2)-(4) as follows:

$$\varphi' = \varphi \wedge \mathsf{SN}[\underline{A},\underline{B}] \wedge \bigwedge_{i=1}^{n} \begin{cases} (\neg B_i \vee (i \cdot w \leq obj)) \wedge \\ (B_i \vee (obj \leq (i-1) \cdot w)) \wedge \\ (\neg(i \cdot w \leq obj) \vee \neg(obj \leq (i-1) \cdot w)) \end{cases} \tag{12}$$

and optimize obj over φ'. Notice here that the third line in equation 12 is $\mathcal{LRA}$-valid, but it allows for implying the negation of $(obj \leq (i-1) \cdot w)$ from $(i \cdot w \leq obj)$ (and vice versa) directly by BCP, without any call to the $\mathcal{LRA}$-Solver.

Consider (8)-(9) and assume that μ assigns k A_is to $\top$ and $n-m$ to $\bot$ as in Definition 1. Then (8) with (12) forces the unit-propagation of $B_1,...,B_k$, and then, among others, of $(k \cdot w \leq obj)$, while (9) with (12) forces the unit-propagation of $\neg B_{m+1},...,\neg B_n$, and then, among others, of $(obj \leq m \cdot w)$. This automatically restricts the range of obj to $[k \cdot w, m \cdot w]$, obtaining the same effect as (2)-(4).

The benefits of the usage of $\mathsf{SN}[\underline{A},\underline{B}]$ are due to both (10) and (11). When the optimization search finds a new minimum $k \cdot w$ and a unit clause in the form $\neg(k \cdot w \leq obj)$ is learned (see e.g. [29]) and $\neg B_k$ is unit-propagated on (12), then as soon as $k-1$ A_is are set to True, all the remaining $n-k+1$ A_is are set to False by BCP (10). A dual case occurs when some lower-bound unit clause $\neg(obj \leq k \cdot w)$ is learned (e.g., in a binary-search step [29]) and B_{k+1} is unit-propagated on (12): as soon as $n-k-1$ A_is are set to False, then all the remaining $k+1$ A_is are set to True by BCP (11).

Example 2. Figure 3 considers the same scenario as in Example 1, in which we extend the encoding with a bidirectional sorting-network relation as in (12). The behaviour is identical to that of Example 1 until the assignment μ is generated, the unit clause $\neg(2 \leq$

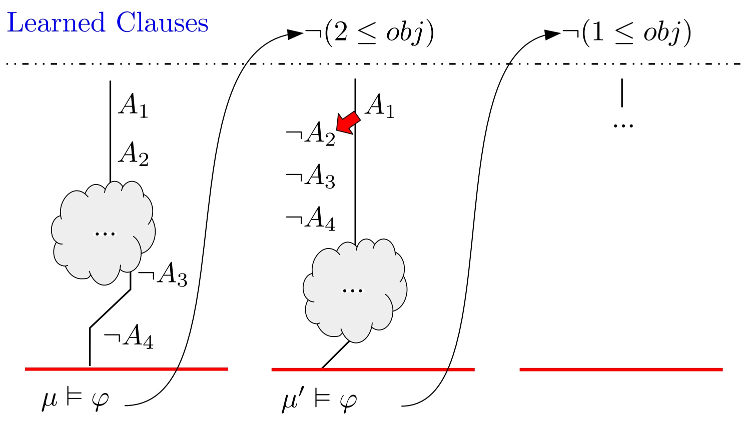


Fig. 3: An example of OMT search with sorting networks.

obj), and the procedure backtracks for the first time (Figure 3 left). This causes the unit-propagation of $\neg B_2$ on (12). As soon as A_1 is picked as new decision, $\neg A_2, \neg A_3, \neg A_4$ are unit propagated (10), saving up to $\binom{4}{2}$ (expensive) calls to the $\mathcal{LRA}$-Solver (Figure 3 center). Then $\neg(1 \leq obj)$ is learned, and the search proceeds (Figure 3 right). $\diamond$

We generalize this approach to deal with the general objectives as in (5)-(6). In this case a separate sorting circuit is generated for each term τ_j, and constraints in the form

$$\bigwedge_{j=1}^{m} \bigwedge_{i=1}^{k_j} (\neg(w_j \cdot i \leq obj) \rightarrow \neg(w_j \cdot i \leq \tau_j)), \tag{13}$$

are added to ensure that the circuit is activated by BCP.

4.1 Bidirectional Sorting Networks

Unlike the usage of sorting networks in other contexts, which consider only (8) and (10) as relevant properties (e.g. [33]), we are interested in sorting networks which propagate both $\top$ and $\bot$ values in both directions (i.e., which comply with all properties (8)-(11)). To this extent, we have considered two encodings: the sequential counter encoding in [33], which we have extended to comply with all properties (8)-(11), and the cardinality network encoding in [9, 6].

Bidirectional Sequential Counter Encoding. The sequential counter encoding $LT_{SEQ}^{n,k}$ for $\leq k(A_1, ..., A_n)$ presented in [33] consists of $O(k \cdot n)$ clauses and variables and complies with (8) and (10). The circuit is given by the composition of n sub-circuits, each of which computes $S_i = \sum_{j=1}^{i} A_j$, represented in unary form with the bits $S_{i,j}$, i.e., $S_{i,j} = \top$ if $\sum_{r=1}^{i} A_r \geq j$, so that $B_j \overset{\text{def}}{=} S_{n,j}$, $j \in [1..n]$. The (CNF version of the)[5] following formula is the encoding of $LT^{n,k}_{SEQ}$ presented in [33], with $k \stackrel{\text{def}}{=} n$:

$$(A_1 \rightarrow S_{1,1}) \wedge \bigwedge_{i=2}^{n}\{((A_i \vee S_{i-1,1}) \rightarrow S_{i,1})\} \wedge \tag{14}$$
$$\bigwedge_{i=2}^{n}\{(\neg A_i \vee \neg S_{i-1,n})\} \wedge \bigwedge_{j=2}^{n}\{(\neg S_{1,j})\} \wedge \tag{15}$$
$$\bigwedge_{i,j=2}^{n}\{(((A_i \wedge S_{i-1,j-1}) \vee S_{i-1,j}) \rightarrow S_{i,j})\} \tag{16}$$

Notice that, in order to reduce the size of the encoding, in (14)-(16) only right implications "$\rightarrow$" were used to encode each gate in the Boolean sorting circuit [33], so that (14)-(16) complies with (8) and (10) but not with (9) and (11). To cope with this fact, we have added the following part, which reintroduces the left implications "$\leftarrow$" of the encoding of each gate in (14) and (16), making it compliant also with (9) and (11):

$$(A_1 \leftarrow S_{1,1}) \wedge \bigwedge_{i=2}^{n}\{((A_i \vee S_{i-1,1}) \leftarrow S_{i,1})\} \wedge \tag{17}$$
$$\bigwedge_{i,j=2}^{n}(((A_i \wedge S_{i-1,j-1}) \vee S_{i-1,j}) \leftarrow S_{i,j}). \tag{18}$$

Bidirectional Cardinality Network Encoding. The cardinality network encoding presented in [17, 9, 6], based on the underlying sorting scheme of the well-known *merge-sort* algorithm, has complexity $O(n \log^2 k)$ in the number of clauses and variables. Due to space limitations, we refer the reader to [6, 9] for the encoding of cardinality networks we used in our own work. Notice that, differently than in the previous case, this sorting network propagates both $\top$ and $\bot$ values in both directions (i.e., it complies with all properties (8)-(11) [9, 6] and it is thus suitable to be used within OMT without modifications.

Both of the previous encodings are istantiated assuming $k = n$, since the sorting network is generated prior to starting the search. Therefore, the cardinality network circuit looks more appealing than the sequential counter encoding due to its lower complexity in terms of clauses and variables employed.

5 Experimental Evaluation

We extended OPTIMATHSAT with a novel internal preprocessing step, which automatically augments the input formula with a sorting network circuit of choice between the bidirectional sequential counter and the cardinality network, as described in §4. To complete our comparison, we also implemented in OPTIMATHSAT two MAXSAT-based approaches, the max-resolution approach implemented in νZ [22, 13] and (for MAXSMT only) the lemma-lifting approach of [16], using MAXINO [7] as external MAXSAT solver.

Here we present an extensive empirical evaluation of various MAXSAT-based and OMT-based techniques in OPTIMATHSAT [31, 4] and νZ [14, 3]. Overall, we considered $> 20,000$ OMT problems and run $> 270,000$ job pairs. The problems were produced either by CGM-TOOL [2] from optimization of Constrained Goal Models [24, 23] (a modeling and automated-reasoning tool for requirement engineering) or by PYLMT [5] from (Machine) Learning Modulo Theories [34]. We partition these problems into

[5] Here (14)-(18) are written as implications to emphasize the directionality of the encodings.

two distinct categories. In §5.1 we analyze problems which are *solvable by* MAXSAT-*based approaches*, like those with PB objective functions or their lexicographic combination, so that to allow both νZ and OPTIMATHSAT to use their MAXSAT-specific max-resolution engines (plus others). In §5.2 we analyze problems which *cannot* be solved by MAXSAT-based approaches, because the objective functions involve some non-PB components, forcing to restrict to OMT-based approaches only.

The goal of this empirical evaluation is manyfold:

(i) compare the performance of MAXSAT-based approaches wrt. OMT-based ones, on the kind of OMT problems where the former are applicable;
(ii) evaluate the benefits of sorting-network encodings with OMT-based approaches (and also with MAXSAT-based ones);
(iii) compare the performances of OPTIMATHSAT with those of νZ.

For goals (i) and (ii) we used the following configurations of OPTIMATHSAT.

OMT-based: standard, enriched with the bidirectional sequential-counter and cardinality sorting network;
MAXSAT-based: the above-mentioned max-resolution implementation, with and without the cardinality sorting network, and lemma-lifting (for pure MAXSMT only).

For goal (iii) we also used the following configurations of νZ. [6]

OMT-based: standard (encoded as in (2)-(4)).
MAXSAT-based: using alternatively the internal implementations of the max-resolution [22, 13] and WMAX [25] procedures.

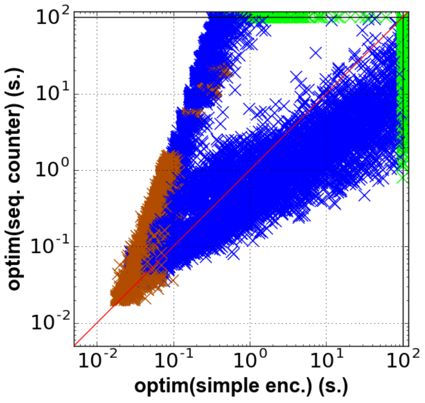
Each job pair was run on one of two identical Ubuntu Linux machines featuring *8-core Intel-Xeon@2.20GHz* CPU, 64 GB of ram and kernel 3.8-0-29. Importantly, we verified that all tools/configurations under test agreed on the results on all problems when terminating within the timeout. (The timeout varies with the benchmark sets, see §5.1 and §5.2.) All benchmarks, as well as our experimental results and all the tools which are necessary to reproduce the results, are available [1].

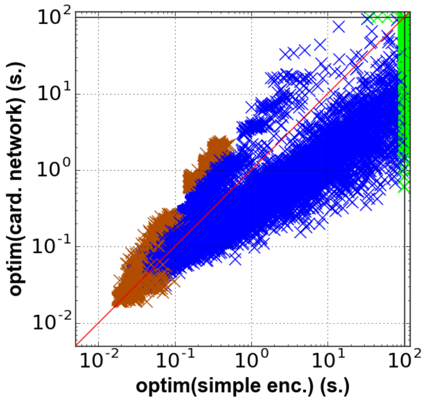
5.1 Problems suitable for MaxSAT-based approaches

Test Set #1: CGMs with lexicographic PB optimization. In our first experiment we consider the set of all problems produced by CGM-TOOL [2] in the experimental evaluation in [23]. They consist of 18996 automatically-generated formulas which encode the problem of computing the lexicographically-optimum realization of a constrained goal model [23], according to a prioritized list of (up to) three objectives $\langle obj_1, obj_2, obj_3\rangle$.

[6] Notice that, unlike OPTIMATHSAT, νZ selects automatically its presumably-best configuration for a given input problem. In particular, when MAXSMT-encodable problems are fed to νZ –like, e.g., those in §5.1– νZ forces automatically the choice of the MAXSAT-based configuration, allowing the user only the choice of the MAXSAT algorithm. Thus we could not test νZ also with OMT-based configuration for the problems in §5.1. Alternatively, we should have disguised the input problem, with the risk of affecting the significance or the result.

tool, configuration & encoding	**inst.**	**term.**	**timeout**	**time (s.)**
OPTIMATHSAT (OMT-based)	18996	16316	2680	48832
OPTIMATHSAT (OMT-based + seq. counter)	18996	16929	2067	90080
OPTIMATHSAT (OMT-based + card. network)	18996	**17191**	1805	39215
OPTIMATHSAT (MAXSAT-based w. maxres)	18996	17933	1063	24369
OPTIMATHSAT (MAXSAT-based w. maxres + seq. counter)	18996	18180	816	49088
OPTIMATHSAT (MAXSAT-based w. maxres + card. netw.)	18996	**18197**	799	26489
νZ (MAXSAT-based w. maxres)	18996	**18996**	0	1640





partitioned sequential-counter encoding				
vars	**inst.**	**term.**	**timeout**	**time (s.)**
∞	18996	16929	2067	90080
10	18996	17033	1963	39035
15	18996	17061	1935	39264
20	18996	**17152**	1844	43730

partitioned cardinality-network encoding				
vars	**inst.**	**term.**	**timeout**	**time (s.)**
∞	18996	**17191**	1805	39215
10	18996	17058	1938	36636
15	18996	17133	1863	37246
20	18996	17190	1806	39492

Fig. 4: [Top, table] Results of various solvers, configurations and encodings on all the problems encoding CGM optimization with lexicographic PB optimization of [24, 23]. (Values in **boldface** denote the best performance of each category; values in **blue** denote the absolute best performance.)
[Middle, scatterplots]. Pairwise comparison on OPTIMATHSAT (OMT-based) with/out sequential-counter encoding (left) and with/out cardinality-network encoding (right). (Brown points denote unsatisfiable benchmarks, blue denote satisfiable ones and green ones represent timeouts.)
[Bottom, tables] Effect of splitting the PB sums into chunks of maximum variable number (no split, 10, 15, 20 variables) with the sequential-counter encoding (left) and the cardinality-network encoding (right).

A solution *optimizes lexicographically* $\langle obj_1, ..., obj_k\rangle$ if it optimizes obj_1 and, if more than one such obj_1-optimum solutions exists, it also optimizes obj_2,..., and so on; both OMT-based and MAXSAT-based techniques handle lexicographic optimization, by optimizing $obj_1, obj_2, ...$ in order, fixing the value of each obj_i to its optimum as soon as

tool, configuration & encoding	**inst.**	**term.**	**timeout**	**time (s.)**
OptiMathSAT (OMT-based)	2499	1794	705	11178
OptiMathSAT (OMT-based + seq. counter)	2499	**2451**	48	18033
OptiMathSAT (OMT-based + card. network)	2499	2186	313	10633
OptiMathSAT (MaxSAT-based w. maxres)	2499	**2499**	0	**128**
OptiMathSAT (MaxSAT-based w. maxres + seq. counter)	2499	2499	0	1638
OptiMathSAT (MaxSAT-based w. maxres + card. netw.)	2499	2499	0	257
OptiMathSAT (lemma-lifting w. Maxino)	2499	2497	2	343
νZ (MaxSAT-based w. maxres)	2499	**2499**	0	**119**
νZ (MaxSAT-based w. wmax)	2499	1799	733	10549

Fig. 5: Results of various solvers, configurations and encodings on CGM-encoding problems of [24, 23] with single-objective weight-1.

it is found [13, 14, 32, 31]. In this experiment, we set the timeout at 100 seconds. The results are reported in Figure 4 (top and middle).

As far as OptiMathSAT (OMT-based) is concerned, extending the input formula with either of the sorting networks increases the number of benchmarks solved within the timeout. Notably, the cardinality network encoding –which has the lowest complexity– scores the best both in terms of number of solved benchmarks and solving time. On the other hand, the sequential counter network is affected by a significant performance hit on a number of benchmarks, as it is witnessed by the left scatter plot in figure 4. This not only affects unsatisfiable benchmarks, for which using sorting networks appears to be not beneficial in general, but also satisfiable ones.

A possible strategy for overcoming this performance issue is to reduce the memory footprint determined by the generation of the sorting network circuit. This can be easily achieved by splitting each Pseudo-Boolean sum in smaller sized chunks and generating a separate sorting circuit for each splice. The result of applying this enhancement, using chunks of increasing size, is shown in Figure 4 (bottom). The data suggest that the sequential counter encoding can benefit from this simple heuristic, but it does not reach the performances of the cardinality network, which are not affected by this strategy. (In next experiments this strategy will be no more considered.)

As far as OptiMathSAT (MaxSAT-based) is concerned, we notice that it significantly outperforms all OMT-based techniques. Remarkably, extending the input formula with the sorting networks improves the performance also of this configuration.

As far as νZ (MaxSAT-based) is concerned, we notice that when using the max-resolution algorithm it outperforms all other techniques by solving all problems.

Test Set #2: CGMs with weight-1 PB optimization. Our second experiment is a variant of the previous one, in which we consider only single-objective optimizations and we fix all weights to 1, so that each problem is encoded as a plain un-weighted MaxSMT problem. We set the timeout to $100s$. The results are reported in Figure 5.

As far as OptiMathSAT (OMT-based) is concerned, extending the input formula with either of the sorting networks increases the number of benchmarks solved within the timeout. Surprisingly, this time the sequential counter network performs significantly better than the cardinality network, despite its bigger size. (We do not have a clear explanation of this fact.)

As far as OPTIMATHSAT (MAXSAT-based) is concerned, we notice that it significantly outperforms all OMT-based techniques, solving all problems. Extending the input formula with the cardinality networks slightly worsens the performances. Also the lemma-lifting techniques outperforms all OMT-based techniques, solving only two problem less than the previous MAXSAT-based techniques.

As far as νZ (MAXSAT-based) is concerned, we notice that using the max-resolution MAXSAT algorithm it is the best scorer, although the differences wrt. OPTIMATHSAT (MAXSAT-based) are negligible, whilst by using the `wmax` engine the performances decrease drastically.

5.2 Problems unsuitable for MaxSAT-based approaches

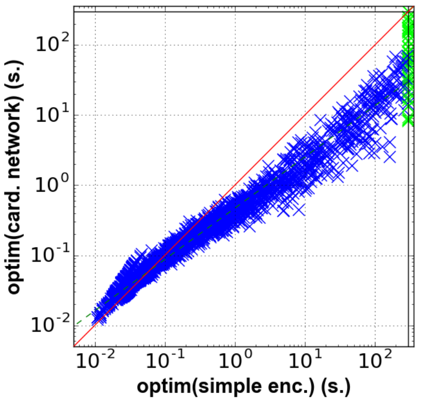
Here we present a couple of test sets which cannot be supported by any MAXSAT-based technique in OPTIMATHSAT or νZ and, to the best of our knowledge, no encoding of these problem into MAXSMT has ever been conceived. Thus the solution is restricted to OMT-based techniques. (To this extent, with OPTIMATHSAT we have used the linear-search strategy rather than the default adaptive linear/binary one to better compare with the linear strategy adopted by νZ.)

Test Set #3: CGMs with max-min PB optimization. In our third experiment we consider another variant of the problems in Test Set #1, in which the three PB/MAXSMT objectives $\langle obj_1, obj_2, obj_3 \rangle$ are subject to a max-min combination: each objective obj_j is normalized so that its range equals $[0, 1]$ (i.e., it is divided by $\sum_i w_{ji}$), then $\bigwedge_{j=1}^{3}(obj_j \le obj)$ s.t. obj is a fresh $\mathcal{LRA}$ variable is added to the main formula, and the solver is asked to find a solution making obj minimum (see [31]). Notice that max-min optimization guarantees a sort of "fairness" among the objectives $obj_1, ..., obj_3$. Since the problem is more complex than the previous ones and the most-efficient MAXSAT-based techniques are not applicable, we increased the timeout to 300 seconds. The results are shown in Figure 6. (Unlike with Figure 4, since the difference in performance between OPTIMATHSAT with the two sorting networks is minor, here and in Figure 7 we have dropped the scatterplot with the sequential-counter encoding and we introduced one comparing with νZ instead.)

Looking at the table and at the scatterplot on the left, we notice that enhancing the OMT-based technique of OPTIMATHSAT by adding the cardinality networks improves significantly the performances. Also, looking at the table and at the scatterplot on the right, we notice that OMT-based technique of OPTIMATHSAT, with the help of sorting networks, performs equivalently or slightly better than that of νZ.

Test Set #4: LMT with mixed complex objective functions. In our fourth experiment we consider a set of 500 problems taken from PYLMT [5], a tool for Structured Learning Modulo Theories [34] which uses OPTIMATHSAT as back-end oracle for performing inference in the context of machine learning in hybrid domains. The objective

tool, configuration & encoding	**inst.**	**term.**	**timeout**	**time (s.)**
OPTIMATHSAT (OMT-based)	2399	2340	59	20841
OPTIMATHSAT (OMT-based + seq. counter)	2399	2394	5	9511
OPTIMATHSAT (OMT-based + card. network)	2399	**2395**	4	8275
νZ (OMT-based)	2399	2390	9	8076



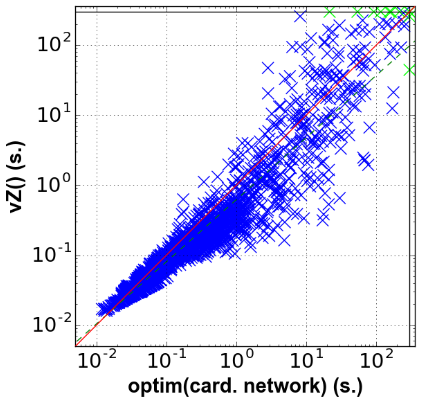


Fig. 6: [Table:] results of various solvers with OMT-based configurations on CGM-encoding problems of [24, 23] with max-min objective functions.
[Left scatterplot:] OPTIMATHSAT + card. network vs. plain OPTIMATHSAT.
[Right scatterplot:] νZ vs. OPTIMATHSAT + card. network.

functions obj are complex combinations of PB functions in the form:

$$obj \overset{\text{def}}{=} \sum_j w_j \cdot B_j + cover - \sum_k w_k \cdot C_k - |K - cover|, \tag{19}$$

$$s.t.\ cover \overset{\text{def}}{=} \sum_i w_i A_i, \tag{20}$$

A_i, B_j, C_k being Boolean atoms, w_i, v_j, z_k, K being rational constants. We imposed a timeout of 600 seconds. The results are presented in Figure 7.

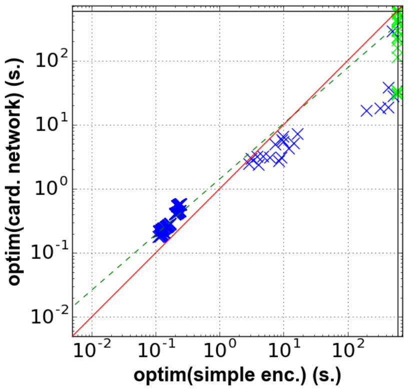
Looking at the table and at the scatterplot on the left, we notice that enhancing the OMT-based technique of OPTIMATHSAT by adding the cardinality networks improves the performances, although this time the improvement is not dramatic. (We believe this is due that the values of the weights w_i, v_j, z_k, K are very heterogeneous, not many weights share the same value.) Also, looking at the table and at the scatterplot on the right, we notice that OMT-based technique of OPTIMATHSAT performs significantly better than that of νZ, even without the help of sorting networks.

Discussion. We summarize the results as follows.

(a) When applicable, MAXSAT-based approaches performed much better than OMT-based ones, in particular when adopting Maximum-Resolution as MAXSAT engine.

(b) Bidirectional sorting-network encodings improved significantly the performances of OMT-based approaches, and often also of MAXSAT-based ones.

tool, configuration & encoding	inst.	term.	timeout	time (s.)
OPTIMATHSAT (OMT-based)	500	421	79	2607
OPTIMATHSAT (OMT-based + seq. counter)	500	441	59	6381
OPTIMATHSAT (OMT-based + card. network)	500	**442**	58	6189
νZ (OMT-based)	500	406	94	2120



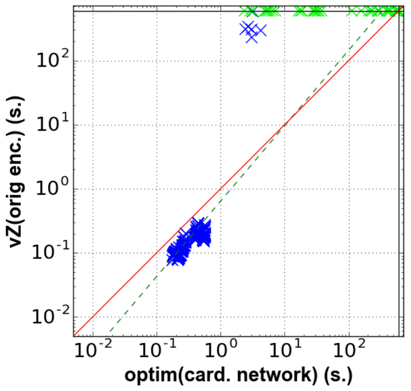


Fig. 7: [Table:] results of various solvers with OMT-based configurations on LMT-encoding problems of [34] with complex objective functions.
[Left scatterplot:] OPTIMATHSAT + card. network vs. plain OPTIMATHSAT.
[Right scatterplot:] νZ vs. OPTIMATHSAT + card. network.

(*c*) νZ performed better than OPTIMATHSAT on MAXSAT-based approaches, whilst the latter performed sometimes similarly and sometimes significantly better on OMT-based ones, in particular when enhanced by the sorting-network encodings.

6 Conclusion and Future Work

MAXSMT and OMT with Pseudo-Boolean objective functions are important sub-cases of OMT, for which specialized techniques have been developed over the years, in particular exploiting state-of-the-art MAXSAT procedures. When applicable, these specialized procedures seem to be more efficient than general-purpose OMT. When they are not applicable, OMT-based technique can strongly benefit from the integration with bidirectional sorting networks to deal with PB components of objectives.

OMT is a young technology, with large margins for improvements. Among others, one interesting research direction is that of integrating MAXSAT-based techniques with standard OMT-based ones for efficiently handling complex objectives and constraints, so that to combine the efficiency of the former with the expressivity of the latter.

References

1. `http://disi.unitn.it/trentin/resources/tacas17.tar.gz`.
2. CGM-Tool. `http://www.cgm-tool.eu`.
3. μZ. `http://rise4fun.com/z3opt`.
4. OptiMathSAT. `http://optimathsat.disi.unitn.it`.
5. PyLMT. `http://www.bitbucket.org/stefanoteso/pylmt`.
6. I. Abío, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. A Parametric Approach for Smaller and Better Encodings of Cardinality Constraints. In *19th International Conference on Principles and Practice of Constraint Programming*, CP'13, 2013.
7. M. Alviano, C. Dodaro, and F. Ricca. A maxsat algorithm using cardinality constraints of bounded size. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2677–2683. AAAI Press, 2015.
8. C. Ansótegui, M. Bofill, M. Palahí, J. Suy, and M. Villaret. Satisfiability Modulo Theories: An Efficient Approach for the Resource-Constrained Project Scheduling Problem. In *SARA*, 2011.
9. R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.
10. C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. *Satisfiability Modulo Theories*, chapter 26, pages 825–885. In Biere et al. [11], February 2009.
11. A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*. IOS Press, February 2009.
12. N. Bjorner. personal communication, 02 2016.
13. N. Bjorner and A.-D. Phan. νZ - Maximal Satisfaction with Z3. In *Proc International Symposium on Symbolic Computation in Software Science*, Gammart, Tunisia, December 2014. EasyChair Proceedings in Computing (EPiC). `http://www.easychair.org/publications/?page=862275542`.
14. N. Bjorner, A.-D. Phan, and L. Fleckenstein. νZ - An Optimizing SMT Solver. In *Proc. TACAS*, volume 9035 of *LNCS*. Springer, 2015.
15. A. Cimatti, A. Franzén, A. Griggio, R. Sebastiani, and C. Stenico. Satisfiability modulo the theory of costs: Foundations and applications. In *TACAS*, volume 6015 of *LNCS*, pages 99–113. Springer, 2010.
16. A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. A Modular Approach to MaxSAT Modulo Theories. In *International Conference on Theory and Applications of Satisfiability Testing, SAT*, volume 7962 of *LNCS*, July 2013.
17. N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *JSAT*, 2(1-4):1–26, 2006.
18. D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. Minimal-Model-Guided Approaches to Solving Polynomial Constraints and Extensions. In *SAT*, 2014.
19. Y. Li, A. Albarghouthi, Z. Kincad, A. Gurfinkel, and M. Chechik. Symbolic Optimization with SMT Solvers. In *POPL*, 2014.
20. J. P. Marques-Silva, I. Lynce, and S. Malik. *Conflict-Driven Clause Learning SAT Solvers*, chapter 4, pages 131–153. In Biere et al. [11], February 2009.
21. A. Nadel and V. Ryvchin. Bit-vector optimization. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2016*, volume 9636 of *LNCS*. Springer, 2016.
22. N. Narodytska and F. Bacchus. Maximum satisfiability using core-guided maxsat resolution. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 2717–2723. AAAI Press, 2014.
23. C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos. Multi-object reasoning with constrained goal models. *Requirements Engineering*, 2016. In print. Published online 24 December 2016. DOI: `http://dx.doi.org/10.1007/s00766-016-0263-5`.

24. C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos. Requirements Evolution and Evolution Requirements with Constrained Goal Models. In *Proceedings of the 37nd International Conference on Conceptual Modeling - ER16*, LNCS. Springer, 2016.
25. R. Nieuwenhuis and A. Oliveras. On SAT Modulo Theories and Optimization Problems. In *Proc. Theory and Applications of Satisfiability Testing - SAT 2006*, volume 4121 of *LNCS*. Springer, 2006.
26. O. Roussel and V. Manquinho. *Pseudo-Boolean and Cardinality Constraints*, chapter 22, pages 695–733. In Biere et al. [11], February 2009.
27. N. W. Sawaya and I. E. Grossmann. A cutting plane method for solving linear generalized disjunctive programming problems. *Computing Chemical Engineering*, 29(9):1891–1913, 2005.
28. R. Sebastiani. Lazy Satisfiability Modulo Theories. *Journal on Satisfiability, Boolean Modeling and Computation, JSAT*, 3(3-4):141–224, 2007.
29. R. Sebastiani and S. Tomasi. Optimization in SMT with LA(Q) Cost Functions. In *IJCAR*, volume 7364 of *LNAI*, pages 484–498. Springer, July 2012.
30. R. Sebastiani and S. Tomasi. Optimization Modulo Theories with Linear Rational Costs. *ACM Transactions on Computational Logics*, 16(2), March 2015.
31. R. Sebastiani and P. Trentin. OptiMathSAT: A Tool for Optimization Modulo Theories. In *Proc. International Conference on Computer-Aided Verification, CAV 2015*, volume 9206 of *LNCS*. Springer, 2015.
32. R. Sebastiani and P. Trentin. Pushing the Envelope of Optimization Modulo Theories with Linear-Arithmetic Cost Functions. In *Proc. Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'15*, volume 9035 of *LNCS*. Springer, 2015.
33. C. Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In P. van Beek, editor, *CP*, volume 3709 of *LNCS*, pages 827–831. Springer, 2005.
34. S. Teso, R. Sebastiani, and A. Passerini. Structured Learning Modulo Theories. *Artificial Intelligence Journal*, 2015. In print. Published online 29 April 2015. `http://dx.doi.org/10.1016/j.artint.2015.04.002.`