On the Benefits of Enhancing Optimization Modulo Theories with Sorting Networks for MaxSMT

Roberto Sebastiani and Patrick Trentin

DISI, University of Trento, Italy

Abstract

Optimization Modulo Theories (OMT) is an extension of SMT, which combines SMT with optimization, finding models that make given objectives optimal. OMT has been extended to be incremental and to handle multiple objective functions either independently or with their linear, lexicographic, Pareto, min-max/max-min combinations. OMT applications can be found not only in the domains of Formal Verification, Automated Reasoning and Planning with Resources, but also Machine Learning and Requirement Engineering.
(Partial weighted) MAXSMT–or, alternatively, OMT with Pseudo-Boolean objective functions– is a very-relevant subcase of OMT. Unfortunately, using general OMT algorithm for MAXSMT suffers from some intrinsic inefficiencies in some cases. In this paper we identify the sources of such inefficiencies and address them by enhancing general OMT by means of sorting networks. We implemented this idea on top of the OPTIMATHSAT OMT solver and evaluated them empirically on problems coming from Machine Learning and Requirement Engineering. The empirical results support the effectiveness of this idea.

1 Introduction

Optimization Modulo Theories (OMT) is an extension of SMT, which allows for finding models that make a given objective optimal through a combination of SMT and optimization procedures [18, 11, 12, 20, 21, 14, 15, 9, 10, 23, 22]. Latest advancements in OMT have further broadened its horizon by making it incremental [9, 23] and by supporting objectives defined in other theories than linear arithmetic (e.g. Bit-Vectors) [9, 10, 16]. Moreover, OMT has been extended with the capability of handling multiple objectives at the same time either independently or through their linear, min-max/max-min, lexicographic or Pareto combination [9, 10, 23].

An important sub-case of OMT is (partial weighted [1]) MAXSMT–or alternatively OMT with Pseudo-Boolean objective functions [19]– [18, 11, 12], 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 made True by the model.

Two main approaches for MAXSMT have been adopted in the literature.

One approach is to embed some MAXSAT engine within the SMT solver itself, and use it in combination with dedicated $\mathcal{T}$*-solvers* [4, 12, 9, 10]. Although this approach can be very efficient, it has some limitations that make it an impractical choice in some cases.

The first issue is that, to the best of our knowledge, MAXSAT engines deal with integer weights only. However, some applications, e.g., *Learning Modulo Theories* [25] –a hybrid Machine Learning approach in which OMT is used as an oracle for Support Vector Machines [25]– may require the weight of *soft*-constraints to be high-precision rational values. 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.

[1]Hereafter, when speaking of MAXSAT and MAXSMT, we keep “partial weighted” implicit unless explicitly stated otherwise.

The second drawback is that a MAXSAT engine cannot be used when dealing with an OMT problem with multiple-independent objectives that need to be optimized at the same time, or when the objective function is given by either a linear combination of Pseudo-Boolean and arithmetic terms (like, e.g., for Linear Generalized Disjunctive Programming problems [21]) or a non-trivial combination of several Pseudo-Boolean sums as in [25].

An alternative approach for dealing with MAXSMT is to encode it as a Pseudo-Boolean objective in Optimization Modulo Theories [21] which, for the above reasons, is the approach adopted in OPTIMATHSAT [1].

We notice that, unfortunately, compared with the use of a dedicated MAXSAT engine, this second approach might result in poor performance when dealing with MAXSMT problems in which a large number of *soft*-clauses are assigned the same weight. In fact, this situation entails the existence of symmetries in the solution space that might lead to a combinatorial explosion of the partial truth assignments generated by the CDCL engine during the optimization search.

In this paper we analyze the sources of this inefficiency and describe and evaluate a solution based on sorting networks, which we implemented on top of OPTIMATHSAT. The benefits of applying this technique are shown with an experimental evaluation based on two OMT applications, performed on benchmarks sets coming from the Requirement Engineering and Machine Learning domain.

Related Work. The idea of MaxSMT and, more generally, of optimization in SMT was first introduced by Nieuwenhuis & Oliveras [18], 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. [11] introduced the notion of "Theory of Costs" $\mathcal{C}$ to handle Pseudo-Boolean (PB) cost functions and constraints by an ad-hoc and independent "$\mathcal{C}$-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. Ansótegui et al. [4] described the evaluation of an implementation of a MaxSMT procedure based on YICES, although this implementation is not publicly available. Cimatti et al. [12] presented a "modular" approach for MaxSMT, combining a lazy SMT solver with a MaxSAT solver, which can be used as black-boxes, 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 [20, 21] introduced a wider notion of optimization in SMT, namely *Optimization Modulo Theories (OMT) with linear cost functions on the rationals*, 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 [2], and presented novel OMT($\mathcal{LRA} \cup \mathcal{T}$) tools which combine standard SMT with LP minimization techniques. 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 [15, 14, 9, 23, 10, 22]. To the best of our knowledge only four OMT solvers are currently implemented: BCLT [14], OPTIMATHSAT [23, 22], SYMBA [15] and νZ [9, 10]. Remarkably, νZ [9, 10] implements specialized procedures for MaxSMT, leveraging to SMT level various state-of-the-art MaxSAT procedures. In addition to it, νZ features a Pseudo-Boolean $\mathcal{T}$*-solver* which can generate sorting networks on demand for Pseudo-Boolean inequalities featuring sums with small coefficients when a Pseudo-Boolean inequality is used some times for unit propagation/conflicts [10, 8].

Importantly, both partial weighted MAXSMT and SMT with PB objectives can be encoded into OMT($\mathcal{LRA} \cup \mathcal{T}$), whereas the contrary is not possible [20, 21].

Content. The paper is structured as follows. The background and the state of the art are briefly reviewed

[2]Notice that $\mathcal{T}$ can also be a combination of Theories $\bigcup_i \mathcal{T}_i$ [20, 21].

in section §2, whereas section §3 describes the source of inefficiency arising when partial weighted MAXSMT is encoded in OMT as in [21]. Section §4 illustrates a possible solution based on sorting networks, whereas in §5 we provide empirical evidence of the benefits of this approach on two applications of OMT interest. Section §6 concludes the paper with some considerations on the future work.

2 Background and State of the Art

Optimization Modulo Theories 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* [18].

The basic minimization scheme implemented in state-of-the-art OMT solvers, known as *linear-search* scheme [20, 21], requires solving an SMT problem with a solution space that is progressively tightened by means of unit linear constraints in the form $(obj < ub_i)$, 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* of interest that, taken as input a pair $\langle \mu, obj \rangle$, returns the optimal value ub of *obj* for such μ. The OMT search terminates when the latter 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 [20, 21].)

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 [18, 11, 4, 12]. As described in [21], MAXSMT $\langle \varphi_h, \varphi_s \rangle$ can be encoded into a general OMT problem with a Pseudo-Boolean objective. To do so, one first introduces 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 encodes the problem into OMT as a pair $\langle \varphi, obj \rangle$ where φ is defined as

$$\begin{aligned} \varphi &\overset{\text{def}}{=} \varphi^* \wedge \textstyle\bigwedge_i ((A_i \rightarrow (x_i = w_i)) \wedge (\neg A_i \rightarrow (x_i = 0))) \wedge & (2)\\ &\quad \textstyle\bigwedge_i ((0 \leq x_i) \wedge (x_i \leq w_i)) & (3)\\ obj &\overset{\text{def}}{=} \textstyle\sum_i x_i, \; x_i \; fresh & (4) \end{aligned}$$

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 [6]) to detect a possible $\mathcal{LRA}$ inconsistency among the current partial truth assignment over variables A_i and linear cuts in the form $(obj < ub)$ 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 using OMT for MaxSMT

Consider first the case of a MAXSMT-derived OMT problem as in equation (1) s.t. all weights are identical, that is: let $\langle \varphi, obj \rangle$ be an OMT problem, where $obj = \sum_{i=0}^{i=n-1} w \cdot A_i$, A_i being 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 [20, 21], the OMT solver adds a unit clause in the form $(obj < ub)$ 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 $(obj < ub)$ does not only prune the current truth assignment μ from the feasible search space, but it also makes inconsistent any other possible (partial) truth assignment μ' which sets exactly k (or more) A_i variables to True. Thus, each new unit clause in this form prunes at least $\gamma = \binom{n}{k}$ truth assignments from the search space, where γ is equal to the cardinality of the set of possible permutations of μ over the variables A_i.

However, the inconsistency of a truth assignment μ' which sets exactly k variables to True wrt. a unit clause $(obj < ub)$, where $ub = w \cdot k$, can not be determined by simple Boolean Propagation. In fact, $(obj < ub)$ being a $\mathcal{LRA}$ term, the CDCL engine is totally oblivious to this inconsistency until when the $\mathcal{T}$*-solver* for linear arithmetic is invoked, and a conflict clause is generated. Therefore, since the $\mathcal{T}$*-solver* for linear arithmetic is more resource demanding than Boolean Propagation and 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.

To conclude, notice that the performance issue identified with the previous case example can be generalized to any objective obj in the form

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

$$\forall_j \in [1, m].\ (\tau_j = w_j \cdot \sum_{i=0}^{i=k_j} A_{ji}) \ \wedge \ (0 \leq \tau_j) \wedge (\tau_j \leq w_j \cdot k_j) \tag{6}$$

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

4 Combining OMT with Sorting Networks

A solution for improving search efficiency when dealing with MAXSMT and OMT with Pseudo-Boolean objectives in the form

$$obj = w \cdot \sum_{i=0}^{i=n-1} A_i \tag{7}$$

is to reduce the dependency on the expensive $\mathcal{LRA}$-Solver by better exploiting Boolean Constraint Propagation (BCP) with the aid of sorting networks.

A sorting-network relation, depicted in figure 1, takes $A_0, ..., A_{n-1}$ Boolean variables as input and returns $out_0, ..., out_{n-1}$ variables as output s.t., if in the current (partial) truth assignment μ, k variables are set to True, $n - m$ variables are set to False and $m - k$ are unassigned, then by Boolean Constraint Propagation $out_0, ..., out_{k-1}$ are set to True, $out_m, ..., out_{n-1}$ are set to False and $out_k, ..., out_{m-1}$ are not propagated.

For our purposes, it is important that the implementation of the relation is *bidirectional*: e.g., if out_{k-1} is forced to be True (that is, at least k inputs must be True) and $n - k$ inputs A_i are False, then by Boolean Propagation all other unassigned A_is are automatically set to True; vice-versa, if out_{k+1} is forced to be False (that is, at most k inputs can be True) and $n - k$ inputs A_i are True, then all other unassigned A_is are automatically set to False.

Given an OMT problem $\langle \varphi, obj, \rangle$, where obj is as in (7), and a sorting network relation encoding C matching the previous definition, we extend φ in equation (2) as follows:

$$\varphi' = \varphi \wedge C \wedge \bigwedge_{k=0}^{k=n-1} \begin{cases} out_k \rightarrow ((k+1) \cdot w \leq obj) \\ \neg out_k \rightarrow (obj \leq k \cdot w) \\ \neg((k+1) \cdot w \leq obj) \vee \neg(obj \leq k \cdot w) \end{cases} \tag{8}$$

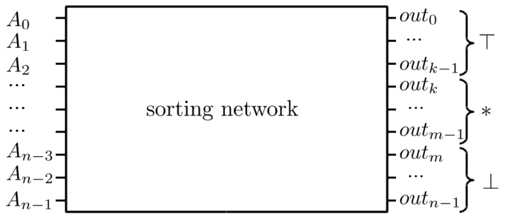


Figure 1: The basic schema of a sorting network relation

and optimize obj over φ'. Notice here that the third line in equation 8 is $\mathcal{T}$-valid, but it allows for assigning the value of some variables out_i by BCP as soon as a unit clause in the form $(obj \leq k \cdot w)$ is added to the formula.

The benefit of this extension is that now, whenever the optimization search finds a new satisfiable truth assignment μ which sets k variables A_i to True, so that a unit clause in the form $(obj < k \cdot w)$ is learned, then as soon as $k-1$ inputs A_i are assigned to True the remaining $n-k+1$ inputs are set to False by BCP. (A dual case occurs when some lower-bound unit clause $(obj > k \cdot w)$ is learned.)

In our work, we have considered two encodings for the sorting network relation: the sequential counter encoding [24] and the cardinality network encoding [3]. Notice that, in contrast with the literature which usually focuses on only one direction of cardinality constraints, in our context we are interested in a bidirectional encoding of sorting circuit so to ensure the correct propagation of the out_k values. This is due to the fact that our OMT solver applies this technique to deal not only with MaxSMT problems but also with Pseudo-Boolean objectives that can be either minimized or maximized, and other Pseudo-Boolean sums subject to simple cardinality constraints outside of an optimization context.

Bidirectional Sequential Counter Encoding

The sequential counter encoding $LT^{n,k}_{SEQ}$ for $\leq k(A_0, ..., A_{n-1})$ presented in [24] requires $O(k \cdot n)$ clauses and variables to be represented, and is arc-consistent. The circuit is given by the composition of n sub-circuits, each of which computes $s_i = \sum_{j=0}^{j=i} A_j$, represented in unary form with the bits $s_{i,j}$. The following formula is the original propositional encoding of $LT^{n,k}_{SEQ}$ presented in [24], with k instantiated to n, which was obtained after discarding one direction of the equations for polarity reasons.

$$\begin{aligned}&(\neg A_0 \vee s_{0,0}) \wedge \bigwedge_{i=1}^{i=n-1} \{(\neg A_i \vee s_{i,0}) \wedge (\neg s_{i-1,0} \vee s_{i,0}) \wedge (\neg A_i \vee \neg s_{i-1,n}) \wedge\} \\ &\bigwedge_{j=1}^{j=n-1} \{(\neg s_{0,j})\} \wedge \bigwedge_{i,j=1}^{i,j=n-1} \{(\neg A_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) \wedge (\neg s_{i-1,j} \vee s_{i,j})\}\end{aligned}$$

We strengthened the original encoding with the following clauses to make it bidirectional:

$$(A_0 \vee \neg s_{0,0}) \wedge \bigwedge_{i=1}^{i=n-1} \{(\neg s_{i,0} \vee A_i \vee s_{i-1,0})\} \wedge \bigwedge_{i,j=1}^{i,j=n-1} (\neg s_{i,j} \vee s_{i-1,j} \vee (s_{i-1,j-1} \wedge A_i))$$

Bidirectional Cardinality Network Encoding

The cardinality network encoding presented in [13, 5, 3], 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, and is arc-consistent. Due to space limitations, we refer the reader to [3] for the encoding of cardinality networks we used in our own work. Notice that, differently than in the previous case, this sorting circuit propagates values in both directions and is thus suitable to be used as is within OMT.

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.

To conclude, note that this approach based on sorting networks can also be applied when dealing with more general Pseudo-Boolean objectives as in equations 5, 6. In this case a separate sorting circuit is generated for each term τ_j, and some (optional) constraints in the form $(obj < w_j \cdot i) \Longrightarrow (\tau_j < w_j \cdot i)$, for $i \in [0, k_j]$, can be added to ensure that the circuit is activated by BCP.

5 Experimental Evaluation

As part of our research work, we extended OPTIMATHSAT with a novel preprocessing step for dealing with `assert-soft` constraints (see [22]) which automatically extends the input formula with a sorting network circuit of choice among the sequential counter and the cardinality network. We recall here that, in OPTIMATHSAT, the `assert-soft` statement can be used not only to encode MAXSMT problems like in νZ, but also generic Pseudo-Boolean terms that can be linearly combined into other constraints or objective functions.

In the following experiments[3] we evaluate whether the speed-up obtained by extending the input formula with a sorting network during the optimization search can outweigh the cost of its generation step within the OMT solver itself, as opposed to demanding the end-user to do it offline.

Each test was performed on a pair of identical Ubuntu Linux machines featuring *8-core Intel-Xeon@2.20GHz* CPU, 64 GB of ram and kernel 3.8-0-29.

Benchmark Set #1: Optimal Realization of Goal Model with Soft-Requirements

In our first experiment, we focused on a benchmark database consisting of 18996 formulas, automatically generated, derived from the problem of computing the optimal realization of a constrained goal model enriched with soft-requirements [17], one of OPTIMATHSAT applications in the context of Requirement Engineering. Each OMT problem contains a combination of up to three MAXSMT objectives sorted with lexicographic priority. In this experiment, we set the timeout at 100 seconds and we verified that all the tested configurations agreed on the optimum value.

As it can be seen in the top table of figure 2, extending the input formula with either of the sorting network circuits increases the number of benchmarks solved within the time-out.

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 circuit is affected by a significant performance hit on a number of benchmarks, as it is witnessed by the bottom-left scatter plot in figure 2. 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 3. Both the table and the scatter plots suggest that the sequential counter encoding can greatly benefit from this simple heuristic, since both the number of solved benchmarks and the running time improve to levels comparable with that achieved using the cardinality network circuit.

As suggested by one of the reviewers, we extended our experimental evaluation to include νZ. The results are shown in the top table of figure 2. Although νZ is able to solve the whole benchmark set in a very short amount of time using the *MaxRes* MAXSMT engine, OPTIMATHSAT and νZ disagree on the optimal solution of a number of benchmarks. Further investigation confirmed that in some cases νZ returns an incorrect optimum model when dealing with multiple MAXSMT combined lexicographically.

[3] All benchmarks, as well as our experimental results, are made available at `http://disi.unitn.it/~trentin/resources/smt16.tar.gz`

encoding	# inst.	# term.	# incorrect	time (s.)
OPTIMATHSAT				
orig. OMT enc.	18996	16316	0	48832
seq. counter enc.	18996	16929	0	90080
card. network enc.	18996	**17191**	0	39215
νZ (maxres)	18996	18996	255	1766
νZ (wmax)	18996	16650	3785	38040

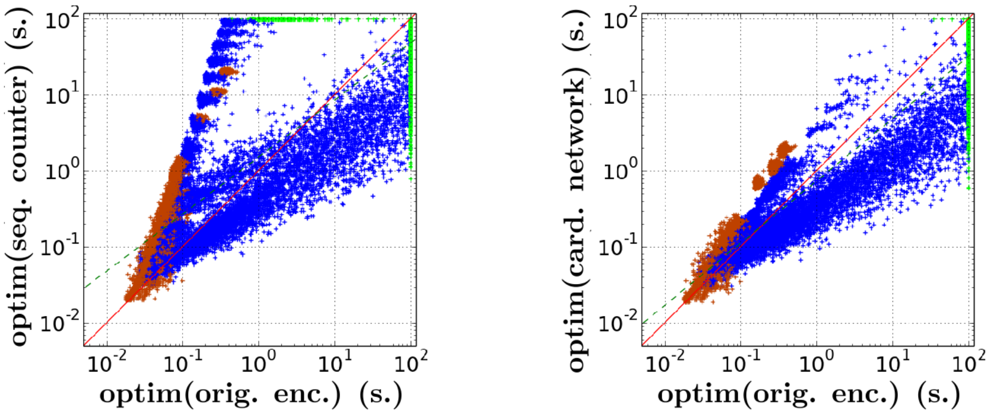


Figure 2: Top: Results on MAXSMT problems using the OMT encoding in [21] (top), paired with the Boolean cardinality constraint encoding in [24] (middle) and in [3] (bottom). Bottom: performance gain obtained by joining the input formula with the sequential counter circuit (left) and the cardinality network encoding (right); brown colour denotes unsatisfiable benchmarks, blue represent satisfiable ones and green represents timeouts.

Benchmark Set #2: Structured Learning Modulo Theories

The second experiment regards a set of problems taken from PYLMT [2], a tool for Structured Learning Modulo Theories [25] which uses OPTIMATHSAT as back-end oracle for performing inference in the context of machine learning in hybrid domains.

Starting from the original set of 500 satisfiable formulas, which used a naive encoding of Pseudo-Boolean objectives by default, we generated a new benchmark set which uses the `assert-soft` statement to encode the following objective function given by a non-trivial combination of several Pseudo-Boolean terms.

$$cover = \sum_i w_i A_i$$
$$obj = \sum_j w_j \cdot B_j + cover - \sum_k w_k \cdot C_k - |K - cover|$$

We ran both νZ and OPTIMATHSAT over the original set of benchmarks, and compared the following three configurations for OPTIMATHSAT over the `assert-soft` based benchmark set: *orig. enc.*, which maps the input weighted MAXSMT formula into an OMT problem with no circuit, *seq. counter enc.*, which extends the input formula with the sequential counter encoding, and *card. network enc.* which uses instead the cardinality network encoding.

encoding	# inst.	# term.	# incorrect	time (s.)
seq. counter enc.				
unbounded	18996	16929	0	90080
10 vars	18996	17033	0	39035
15 vars	18996	17061	0	39264
20 vars	18996	17152	0	43730
card. network enc.				
unbounded	18996	**17191**	0	39215
10 vars	18996	17058	0	36636
15 vars	18996	17133	0	37246
20 vars	18996	17190	0	39492

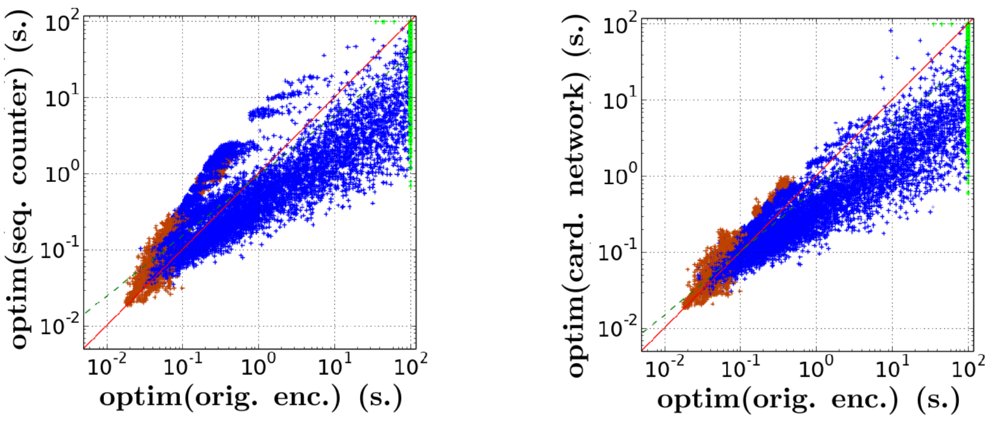


Figure 3: Top: the performance gain obained by the limiting the size of the generated sorting circuit. Bottom: the same comparison as in figure 2, using a circuit size limit of 20; brown colour denotes unsatisfiable benchmarks, blue represent satisfiable ones and green represents timeouts.

We imposed a timeout of 600 seconds for both tools, and verified that all solvers and their configurations –when terminating– agreed on the optimal solution values.

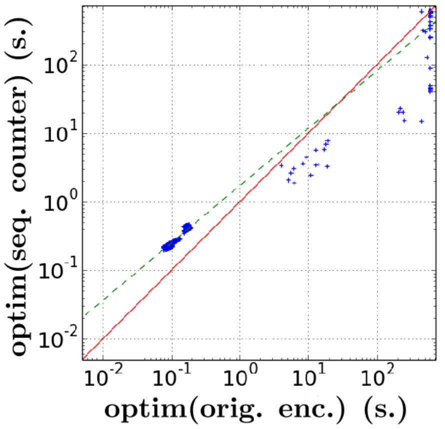
The results, depicted in figure 4, show that extending the input problems with sorting network circuits benefits the number of solved benchmarks within the timeout, with the cardinality network encoding performing better by a small margin. As highlighted by the scatter plots in the bottom part of figure 4, generating these circuit on the fly causes a limited overhead on easy benchmarks, whereas it can significantly improve the running time on difficult ones.

6 Conclusion and Future Work

MaxSMT is an important sub-case of Optimization Modulo Theories for which specialized algorithms and techniques have been developed over the years. Nonetheless, when dealing with non trivial combination of multiple objectives or problems with fine grained weights over soft-clauses, MaxSMT might require being encoded as a Pseudo-Boolean objective and solved with ordinary OMT techniques.

In this paper we reviewed the encoding of MaxSMT in OMT proposed in [21] and described a possible inefficiency problem that might arise when dealing with problems in which a large number of *soft*-clauses share the same weight value, due to the resulting symmetries in the solution space. In order

size	# total	# solved	# time-out	time (s.)
original encoding				
νZ	500	406	94	2120
OptiMathSAT	500	424	76	3522
OptiMathSAT using assert-soft				
orig. OMT enc.	500	421	79	2607
seq. counter enc.	500	441	59	6381
card. network enc.	500	**442**	58	6189



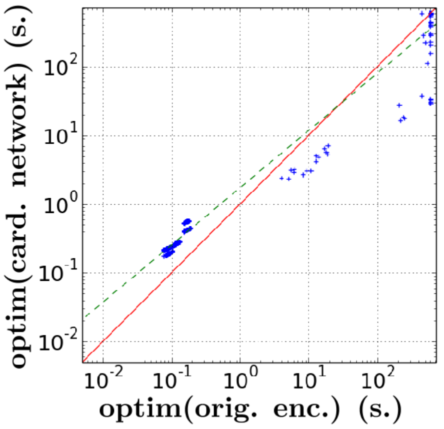


Figure 4: Top: comparison among various OptiMathSAT configurations and νZ on a set of benchmarks based on the Structured Learning Modulo Theories domain [25]. Bottom: scatter plots for the same experimental data showing a comparison among OptiMathSAT solve time over the original problems wrt. the `assert-soft` encoding enriched with a sorting network circuit generated on the fly.

to overcome this issue, we proposed a transparent solution based on extending the input problem with two different sorting networks: the sequential counter circuit and the cardinality constraint encoding.

The benefits of this technique have been demonstrated through an experimental evaluation on a couple of benchmark sets coming from the domain of Requirement Engineering and Machine Learning.

Albeit of the positive results attained, this is still work in progress, and future work should focus on how to extend this technique to better handle similar performance issues when dealing with more heterogeneous sets of weights values.

References

[1] OptiMathSAT. http://optimathsat.disi.unitn.it.

[2] PyLMT. www.bitbucket.org/stefanoteso/pylmt.

[3] 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.

[4] 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.

[5] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.

[6] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability Modulo Theories. In *Handbook of Satisfiability*, chapter 26, pages 825–885. IOS Press, 2009.

[7] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*. IOS Press, February 2009.

[8] N. Bjorner. personal communication, 02 2016.

[9] 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.

[10] N. Bjorner, A.-D. Phan, and L. Fleckenstein. νZ - An Optimizing SMT Solver. In *Proc. TACAS*, volume 9035 of *LNCS*. Springer, 2015.

[11] 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.

[12] 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.

[13] N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *JSAT*, 2(1-4):1–26, 2006.

[14] D. Larraz, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. Minimal-Model-Guided Approaches to Solving Polynomial Constraints and Extensions. In *SAT*, pages 333–350, 2014.

[15] Y. Li, A. Albarghouthi, Z. Kincad, A. Gurfinkel, and M. Chechik. Symbolic Optimization with SMT Solvers. In *POPL*, 2014.

[16] A. Nadel and V. Ryvchin. Bit-vector optimization. In M. Chechik and J. Raskin, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9636 of *Lecture Notes in Computer Science*, pages 851–867. Springer, 2016.

[17] C. M. Nguyen, R. Sebastiani, P. Giorgini, and J. Mylopoulos. Multi object reasoning with constrained goal model. *CoRR*, abs/1601.07409, 2016. Under journal submission. Available at http://arxiv.org/abs/1601.07409.

[18] 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.

[19] O. Roussel and V. Manquinho. *Pseudo-Boolean and Cardinality Constraints*, chapter 22, pages 695–733. In Biere et al. [7], February 2009.

[20] 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.

[21] R. Sebastiani and S. Tomasi. Optimization Modulo Theories with Linear Rational Costs. *ACM Transactions on Computational Logics*, 16(2), March 2015.

[22] 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.

[23] 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.

[24] C. Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In P. van Beek, editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005.

[25] S. Teso, R. Sebastiani, and A. Passerini. Structured Learning Modulo Theories. *Artificial Intelligence Journal*, 2015. To appear.