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

Roberto Sebastiani, Patrick Trentin
roberto.sebastiani@unitn.it
trentin@disi.unitn.it

DISI, University of Trento

SMT Workshop, July 1\textsuperscript{st}, 2016
Contents

1. Background & Motivation

2. Efficiency Issue

3. Solution: OMT with Sorting Networks

4. Experimental Evaluation
Optimization Modulo Theories (OMT) [15, 16, 18]

Optimization Modulo Theories with $\mathcal{L}A$ objectives [15, 16, 6, 7, 17, 13, 4, 5]

problem of finding a model for $\langle \varphi, obj \rangle$, minimizing the value of $obj$:

- $obj$ being a $\mathcal{L}A$ or Pseudo-Boolean cost function
- maximization dual
- extended to multiple objectives (linear combination, min-max, boxed, lexicographic, Pareto) [13, 4, 5, 20, 19]
- incremental [5, 20]

\[ \varphi, \min(obj) \rightarrow \text{OMT} \rightarrow \begin{cases} SAT, M(\varphi) |_{\min(obj)} \\ UNSAT \end{cases} \]
## Formal Verification

- Formal **Verification** of parametric systems [18, 14]
- SW verification & synthesis [12, 14]
  (e.g. BMC, invariant generation, program synthesis, ...)
- Computation of worst-case execution time of loop-free programs [11]
- Computation of optimal structure of undirected Markov network [9]
- Computation abstract transformers [13]
- ...

---

### Other

- **MLMT** [22], a Structured Learning Modulo Theories tool that performs inference and learning in hybrid domains
- **CGM-T** [1], a tool for computing optimal realization of a Goal Model enriched with preferences and resources
## OMT Applications

### Formal Verification
- **Formal Verification** of parametric systems [18, 14]
- SW verification & synthesis [12, 14]
  (e.g. BMC, invariant generation, program synthesis, ...)
- Computation of worst-case execution time of loop-free programs [11]
- Computation of optimal structure of undirected Markov network [9]
- Computation abstract transformers [13]
- ...

### Other

| Machine Learning | **PYLMT** [22], a Structured Learning Modulo Theories [22] tool that performs inference and learning in hybrid domains |
| Requirement Engineering | **CGM-TOOL** [1], a tool for computing optimal realization of a Goal Model enriched with preferences and resources |
A pair \( \langle \varphi_h, \varphi_s \rangle \), where

- \( \varphi_h \): set of “hard” \( \mathcal{T} \)-clauses
- \( \varphi_s \): set of positive-weighted “soft” \( \mathcal{T} \)-clauses

**goal:** find \( \psi \), \( \psi \subseteq \varphi_s \), s.t. \( \varphi_h \cup \psi \) is \( \mathcal{T} \)-satisfiable and \( \psi \) has maximum-weight
Partial Weighted MaxSMT [15, 6, 7, 16, 17]

A pair \( \langle \varphi_h, \varphi_s \rangle \), where

- \( \varphi_h \): set of “hard” \( \mathcal{T} \)-clauses
- \( \varphi_s \): set of \textit{positive-weighted} “soft” \( \mathcal{T} \)-clauses

**goal:** find \( \psi \), \( \psi \subseteq \varphi_s \), s.t. \( \varphi_h \cup \psi \) is \( \mathcal{T} \)-satisfiable and \( \psi \) has maximum-weight

**Approaches**

- MaxSAT engine + SMT’s \( \mathcal{T} \)-Solvers
- encoded as OMT Pseudo-Boolean objective
Partial Weighted $\text{MaxSMT}$ [15, 6, 7, 16, 17]

A pair $\langle \varphi_h, \varphi_s \rangle$, where

- $\varphi_h$: set of “hard” $T$-clauses
- $\varphi_s$: set of positive-weighted “soft” $T$-clauses

**goal:** find $\psi$, $\psi \subseteq \varphi_s$, s.t. $\varphi_h \cup \psi$ is $T$-satisfiable and $\psi$ has maximum-weight

**Approaches**

- $\text{MaxSAT}$ engine + SMT’s $T$-Solvers
- encoded as OMT Pseudo-Boolean objective

**SMT + $\text{MaxSAT}$ engine**

++ very efficient (for pure $\text{MaxSMT}$)

**OMT encoding:**

++ can be used when objective is given by the linear (or min-max) combination of Pseudo-Boolean and Arithmetic terms (e.g. LGDP [17], or [22]).

++ can handle multiple objectives at the same time as in [20]
Given \( \langle \varphi_h, \varphi_s \rangle \), for each \( C_i \in \varphi_s \) introduce fresh Boolean variable \( A_i \)

\[
\varphi \triangleq \varphi_h \cup \bigcup_{C_i \in \varphi_s} \{(A_i \lor C_i)\}; \quad \text{obj} \triangleq \sum_{C_i \in \varphi_s} w_i A_i
\]  

its OMT encoding is a pair \( \langle \varphi, \text{obj} \rangle \) [17]

\[
\varphi \overset{\text{def}}{=} \varphi_h \land \bigwedge_i ((A_i \rightarrow (x_i = w_i)) \land (\neg A_i \rightarrow (x_i = 0))) \land \\
\bigwedge_i ((0 \leq x_i) \land (x_i \leq w_i))^*
\]

\[
\text{obj} \overset{\text{def}}{=} \sum_i x_i, \text{ } x_i \text{ fresh Real variable}
\]

*: Term \( \bigwedge_i \ldots \) + Early Pruning = improved efficiency
OMT encoding

Given $\langle \varphi_h, \varphi_s \rangle$, for each $C_i \in \varphi_s$ introduce fresh Boolean variable $A_i$

$$\varphi \triangleq \varphi_h \cup \bigcup_{C_i \in \varphi_s} \{(A_i \lor C_i)\}; \quad obj \triangleq \sum_{C_i \in \varphi_s} w_i A_i$$

its OMT encoding is a pair $\langle \varphi, obj \rangle$ [17]

$$\varphi \overset{\text{def}}{=} \varphi_h \land \bigwedge_i ((A_i \rightarrow (x_i = w_i)) \land (\neg A_i \rightarrow (x_i = 0))) \land \bigwedge_i ((0 \leq x_i) \land (x_i \leq w_i))$$

$$obj \overset{\text{def}}{=} \sum_i x_i, \; x_i \text{ fresh Real variable}$$

*: Term $\bigwedge_i \ldots$ + Early Pruning = improved efficiency

Problem:

-- Performance bottleneck when dealing with Pseudo-Boolean objectives in the form

$$w_1 \cdot \sum_i A_i + \ldots + w_n \cdot \sum_j A_j$$
Contents

1. Background & Motivation
2. Efficiency Issue
3. Solution: OMT with Sorting Networks
4. Experimental Evaluation
Running Example: efficiency issue

Problem:

- \( \langle \varphi, \min(obj) \rangle \), where \( obj := w \cdot \sum_{i=0}^{n-1} A_i \), currently \( obj = k \cdot w \)
- **OPTIMIZATION STEP**: learn \( \neg (k \cdot w \leq obj) \) and restart/jump to level 0

Example: with \( k = 2 \), \( w = 1 \) and \( n = 4 \)

Learned Clauses

\[ \neg (2 \leq obj) \]
Problem:

- \( \neg(k \leq \text{obj}) \) causes the inconsistency of \( \binom{n}{k} \) truth assignments satisfying exactly \( k \) variables in \( A_0, \ldots, A_{n-1} \)

Example: with \( k = 2 \), \( w = 1 \) and \( n = 4 \)

![Diagram showing learned clauses and inconsistency]

\[ \mu \models \varphi \]
Running Example: efficiency issue

Problem:

- $\neg(k \leq \text{obj})$ causes the inconsistency of $\binom{n}{k}$ truth assignments satisfying exactly $k$ variables in $A_0, \ldots, A_{n-1}$
  
  $\implies$ inconsistency is not revealed by Boolean Propagation

Example: with $k = 2$, $w = 1$ and $n = 4$
Running Example: efficiency issue

Problem:

- up to $\binom{n}{k}$ (expensive) calls to the $\mathcal{LA}$-Solver required

Example: with $k = 2$, $w = 1$ and $n = 4$
1 Background & Motivation

2 Efficiency Issue

3 Solution: OMT with Sorting Networks

4 Experimental Evaluation
Solution: Combine OMT with Sorting Networks

**Idea.** enrich encoding with bi-directional sorting networks [21, 10, 3, 2]

Given \( \langle \varphi, \text{obj} \rangle \), \( \text{obj} := w \cdot \sum_{i=0}^{n-1} A_i \), and a sorting network relation \( C(A_0, ..., A_{n-1}, B_0, ..., B_{n-1}) \) s.t.

- \( k A_i \)'s are \( \top \iff \{ B_0, ..., B_{k-1} \} \) are \( \top \),
- \( m - k A_i \)'s are \( * \iff \{ B_k, ..., B_{m-1} \} \) are \( * \),
- \( n - m A_i \)'s are \( \bot \iff \{ B_m, ..., B_{n-1} \} \) are \( \bot \)

then we encode it as \( \langle \varphi', \text{obj} \rangle \), where

\[
\varphi' := \varphi \land C(A_0, ..., A_{n-1}) \land \bigwedge_{i=0}^{n-1} B_i \leftrightarrow ((k + 1) \cdot w \leq \text{obj}) \land \bigwedge_{i=0}^{n-2} B_{i+1} \rightarrow B_i
\]
OMT with Sorting Network Relation

Properties:

- If \((k \cdot w \leq obj) = \perp\), then by BCP \(\forall i \in [k, n].B_{i-1} = \perp\)

Example: with \(k = 2\), \(w = 1\) and \(n = 4\)
OMT with Sorting Network Relation

Properties:

- if \((k \cdot w \leq obj) = \bot\), then by BCP \(\forall i \in [k, n].B_{i-1} = \bot\)
- as soon as \(k - 1\) \(A_i\) are assigned \(\top\)
  \(\implies\) all others are unit-propagated to \(\bot\)

Dual if \((k \cdot w \leq obj) = \top\).

Example: with \(k = 2\), \(w = 1\) and \(n = 4\)
Running Example: OMT with sorting networks

- **OPTIMIZATION STEP**: learn \( \neg(k \cdot w \leq obj) \) and restart/jump to level 0

Example: with \( k = 2, w = 1 \) and \( n = 4 \)
OPTIMIZATION STEP: learn $\neg (k \cdot w \leq obj)$ and restart/jump to level 0 as soon as $k - 1 A_i$ are assigned $\top$.

$\Rightarrow$ all others are unit-propagated to $\bot$.

Example: with $k = 2$, $w = 1$ and $n = 4$
Solution: Combine OMT with Sorting Networks

Possible encodings for $n \times n$ Boolean relation $C(A_0, \ldots, A_{n-1}, B_0, \ldots, B_{n-1})$ are:

- **Bi-directional Sequential Counter** [21], in $O(n^2)$
  
  sum of $A_i$'s, unary representation

- **Bi-directional Cardinality Network** [10, 3, 2], in $O(n \log^2 n)$
  
  based on *merge-sort* algorithm idea
Solution: Combine OMT with Sorting Networks

Possible encodings for $n \times n$ Boolean relation $C(A_0, \ldots, A_{n-1}, B_0, \ldots, B_{n-1})$ are:

- **Bi-directional Sequential Counter** [21], in $O(n^2)$
  sum of $A_i$’s, unary representation

- **Bi-directional Cardinality Network** [10, 3, 2], in $O(n \log^2 n)$
  based on *merge-sort* algorithm idea

Generalization

The same performance issue occurs for $\langle \varphi, obj \rangle$, where

$$obj = \tau_1 + \ldots + \tau_m,$$

$$\forall j \in [1, m]. \ (\tau_j = w_j \cdot \sum_{i=0}^{i=k_j} A_{ji}) \land (0 \leq \tau_j) \land (\tau_j \leq w_j \cdot k_j)$$

Solution:

- use a separate sorting circuit for each term $\tau_j$
- add clauses in the form $(w_j \cdot i \leq \tau_j) \rightarrow (w_j \cdot i \leq obj)$
Experimental Evaluation

Test Framework:

- two 8-core 2.20Ghz Xeon Linux machines with 64 GB of RAM

Tools:

- OPTI\textsc{m}ATH\textsc{sat}, OMT(\mathcal{L}A) tool based on MATHSAT5 [8]
- \(\nu Z\), general OMT solver based on Z3 [4]

(Partial) Correctness Check:

- all configurations agree on optimal lexicographic solution
  \[\implies \checkmark\]

- otherwise
  \[\implies\] used Z3 to check model is both \textit{satisfiable} and \textit{optimal}
Experiment #1

Benchmark-Set:

- Structured Learning Modulo Theories [22]: inference in hybrid domain

\[
cover = \sum_i w_i A_i
\]

\[
obj = \sum_j w_j \cdot B_j + cover - \sum_k w_k \cdot C_k - |K - cover|
\]

- 500 formulas, 600 s. timeout
### Experiment #1

<table>
<thead>
<tr>
<th>size</th>
<th># total</th>
<th># solved</th>
<th># time-out</th>
<th>time (s.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nu Z$</td>
<td>500</td>
<td>406</td>
<td>94</td>
<td>2120</td>
</tr>
<tr>
<td>OPTI\textsc{MAT}H\textsc{SAT}</td>
<td>500</td>
<td>424</td>
<td>76</td>
<td>3522</td>
</tr>
<tr>
<td>OPTI\textsc{MAT}H\textsc{SAT} using assert-soft</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orig. OMT enc.</td>
<td>500</td>
<td>421</td>
<td>79</td>
<td>2607</td>
</tr>
<tr>
<td>seq. counter enc.</td>
<td>500</td>
<td>441</td>
<td>59</td>
<td>6381</td>
</tr>
<tr>
<td>card. network enc.</td>
<td>500</td>
<td>442</td>
<td>58</td>
<td>6189</td>
</tr>
</tbody>
</table>

**Observations**

- use of *Sorting Networks*
  - $\rightarrow$ improvement # solved benchmarks
Experiment #1

- use of *Sorting Networks*
  
  $\Rightarrow$ more beneficial on harder benchmarks
Experiment #2

Benchmark-Set:

- Optimal Realization of a Goal Model enriched with Soft-Requirements [1]
  - Multiple Partial Weighted MAXSMT problems
  - 3-levels Lexicographic optimization
- 18996 generated formulas $\Rightarrow$ 100 s. timeout
## Experiment #2

<table>
<thead>
<tr>
<th>encoding</th>
<th># inst.</th>
<th># term.</th>
<th># incorrect</th>
<th>time (s.)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OPTIMATHSAT</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>orig. OMT enc.</td>
<td>18996</td>
<td>16316</td>
<td>0</td>
<td>48832</td>
</tr>
<tr>
<td>seq. counter enc.</td>
<td>18996</td>
<td>16929</td>
<td>0</td>
<td>90080</td>
</tr>
<tr>
<td>card. network enc.</td>
<td>18996</td>
<td><strong>17191</strong></td>
<td>0</td>
<td><strong>39215</strong></td>
</tr>
<tr>
<td><strong>νZ</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxres</td>
<td>18996</td>
<td>18996</td>
<td><strong>255</strong></td>
<td>1766</td>
</tr>
<tr>
<td>wmax</td>
<td>18996</td>
<td>16650</td>
<td><strong>3785</strong></td>
<td>38040</td>
</tr>
</tbody>
</table>

### Observations

- Use of *Sorting Networks*  
  \[\rightarrow\] Larger # solved benchmarks

- **OPTIMATHSAT** + SN with *Cardinality Network* encoding  
  \[\rightarrow\] Also faster than **OPTIMATHSAT**
**Sequential Counter** is expensive: $O(n^2)$

solution $\implies$ upper bound to circuit size
Experiment #2

<table>
<thead>
<tr>
<th>circuit bound</th>
<th># inst.</th>
<th># term.</th>
<th>time (s.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>seq. counter enc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unbounded</td>
<td>18996</td>
<td>16929</td>
<td>90080</td>
</tr>
<tr>
<td>10 vars</td>
<td>18996</td>
<td>17033</td>
<td>39035</td>
</tr>
<tr>
<td>15 vars</td>
<td>18996</td>
<td>17061</td>
<td>39264</td>
</tr>
<tr>
<td>20 vars</td>
<td>18996</td>
<td>17152</td>
<td>43730</td>
</tr>
<tr>
<td>card. network enc.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unbounded</td>
<td>18996</td>
<td>17191</td>
<td>39215</td>
</tr>
<tr>
<td>10 vars</td>
<td>18996</td>
<td>17058</td>
<td>36636</td>
</tr>
<tr>
<td>15 vars</td>
<td>18996</td>
<td>17133</td>
<td>37246</td>
</tr>
<tr>
<td>20 vars</td>
<td>18996</td>
<td>17190</td>
<td>39492</td>
</tr>
</tbody>
</table>

- **Sequential Counter**
  \[\rightarrow\text{improved speed & # solved benchmarks}\]

- **Cardinality Network**
  \[\rightarrow\text{slightly worse}\]
Conclusions & future work

Conclusions:

OMT can benefit from *Sorting Networks* when dealing with

- Partial Weighted $\text{MaxSMT}$
- other Pseudo-Boolean objectives

Works also with SMT with PB objectives (no empirical data yet)

Future Work:

- extend this technique to better handle heterogeneous sets of weights values.
Thanks for listening!

Questions..?

www.cgm-tool.eu.


A Parametric Approach for Smaller and Better Encodings of Cardinality Constraints.


Cardinality networks: a theoretical and empirical study.

*Constraints*, 16(2):195–221, 2011.


$\nu$Z - Maximal Satisfaction with Z3.


http://www.easychair.org/publications/?page=862275542.
νZ - An Optimizing SMT Solver.

Satisfiability modulo the theory of costs: Foundations and applications.

A Modular Approach to MaxSAT Modulo Theories.
In International Conference on Theory and Applications of Satisfiability Testing, SAT, volume 7962 of LNCS, July 2013.

The MathSAT 5 SMT Solver.

Learning chordal markov networks by constraint satisfaction.


Translating pseudo-boolean constraints into SAT.


How to compute worst-case execution time by optimization modulo theory and a clever encoding of program semantics.


Constraints as control.


Symbolic Optimization with SMT Solvers.

In *POPL*, 2014.


Symbolic optimization with smt solvers.


On SAT Modulo Theories and Optimization Problems.


Optimization in SMT with LA(Q) Cost Functions.


Optimization Modulo Theories with Linear Rational Costs.

*ACM Transactions on Computational Logics, 16(2), March 2015.*


Optimization Modulo Theories with Linear Rational Costs.

*ACM Transactions on Computational Logics, 16(2), March 2015.*


OptiMathSAT: A Tool for Optimization Modulo Theories.


Pushing the Envelope of Optimization Modulo Theories with Linear-Arithmetic Cost Functions.


Towards an optimal cnf encoding of boolean cardinality constraints.


Structured Learning Modulo Theories.


To appear.