Optimization Modulo Theories
An Introduction

Roberto Sebastiani

Dept. of Computer Science and Engineering, DISI
University of Trento, Italy
roberto.sebastiani@unitn.it
http://disi.unitn.it/rseba

– International SAT/SMT/AR School, Lisbon, PT, July 3-7th, 2019 –
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Satisfiability Modulo Theories SMT($\mathcal{T}$)

**SMT($\mathcal{T}$):** the problem of deciding the satisfiability of a (typically) ground first-order formula wrt some background theory $\mathcal{T}$.

- $\mathcal{T}$ can be a combination of theories $\bigcup_i \mathcal{T}_i$
- Theories of Interest:
  - Linear arithmetic over the rationals ($\mathcal{LRA}$)
    \[(T_\delta \rightarrow (s_1 = s_0 + 3.4 \cdot t - 3.4 \cdot t_0)) \land (\neg T_\delta \rightarrow (s_1 = s_0))\]
  - Linear arithmetic over the integers ($\mathcal{LIA}$)
    \[(x := x_l + 2^{16} x_h) \land (x \geq 0) \land (x \leq 2^{16} - 1)\]
  - Arrays ($\mathcal{AR}$)
    \[(i = j) \lor \text{read(write}(a, i, e), j) = \text{read}(a, j)\]
  - Bit vectors ($\mathcal{BV}$)
    \[x_{16}[15:0] = (y_{16}[15:8] :: z_{16}[7:0]) << w_{16}[3:0]\]
  - Non-linear arithmetic ($\mathcal{NLA}$)
    \[((c = a \cdot b) \land (a_1 = a - 1) \land (b_1 = a + 1)) \rightarrow (c = a_1 \cdot b_1 + 1)\]
    ...
- “Lazy” Approach: SMT solver = CDCL SAT solver + $\mathcal{T}$-solver(s)
Need for Satisfiability Modulo Theories (SMT)

SMT solvers widely used as backend engines in formal verification and many other applications

- SW verification
- verification of Timed and Hybrid Systems
- verification of RTL Circuit designs & of microcode
- static analysis of SW programs
- test-case generation
- program synthesis
- scheduling
- planning with resources
- compiler optimization
- ...

Many SMT-encodable problems require optimum solutions wrt. some objective function. E.g.:

- SW verification
- formal verification of parametric systems
- optimization of physical layout of circuit designs
- scheduling and temporal reasoning
- displacement of tools (e.g. strip-packing problem)
- planning with resources and retrofit planning
- radio link frequency assignment
- machine learning on hybrid domains
- goal modeling in requirement engineering
- ...
Ex.: FV of parametric systems

A (parametric version of a) timed system from [Alur, CAV-99] [8]:

Decision Problem: check safety under fixed choices of the constants (e.g., the delay after which the controller orders the gate to lower the bar) \((M \models G\neg(in \land up))\)

- BMC encodable into a SMT(\(\mathcal{LRA}\)) problem (sat. \(\nrightarrow\) unsafe)
Ex.: FV of parametric systems

A (parametric version of a) timed system from [Alur, CAV-99] [8]:

Optimization Problem: find the minimum “unsafe” delay $D$ after which the controller orders the gate to lower the bar, which doesn’t guarantee safety ($M \not\models G \neg (in \land up)$).

$\implies$ Set the delay $D$ strictly smaller

- BMC encodable into a OMT($\mathcal{LRA}$) problem (min. $D$ s.t. satisf.)
Bounded Model Checking (BMC) looks for an execution path of $M$ of (increasing) length $k$

- satisfying the temporal property $\neg f$ (i.e. $M \models_k E \neg f$)
- minimizing the total elapsed time: $\text{cost} = \min(t^N - t^0)$

BMC is encoded into SMT($\mathcal{T}$) (e.g. $\mathcal{T} = \mathcal{LRA} \cup \mathcal{AR} \cup \ldots$):

- if $\varphi_k$ is satisfiable, then $M \not\models f$

  \[
  \begin{align*}
  DUMP^1 & \rightarrow (A^1 = \text{write}(A^0, i^1, v^1_i)) \\
  \wedge \neg DUMP^1 & \rightarrow (A^1 = A^0) \\
  \wedge DUMP^1 & \rightarrow (t^1 - t^0 = 0) \\
  \wedge \ldots & \\
  \wedge \text{WAIT}^1 & \rightarrow (t^1 - t^0 > 0) \\
  \wedge \ldots & \\
  \wedge DUMP^N & \rightarrow \ldots \\
  \wedge \ldots &
  \end{align*}
  \]
Ex.: Formal Verification of Real-Time Systems

Model Checking: $M \models f$?

Bounded Model Checking (BMC) looks for an execution path of $M$ of (increasing) length $k$

- satisfying the temporal property $\neg f$ (i.e. $M \models_k E \neg f$)
- minimizing the total elapsed time: $\text{cost} = \min(t^N - t^0)$

BMC is encoded into SMT($\mathcal{T}$) (e.g. $\mathcal{T} = \mathcal{LRA} \cup \mathcal{AR} \cup \ldots$):

- if $\varphi_k$ is satisfiable, then $M \not\models f$

\[
\begin{align*}
DUMP^1 &\quad \rightarrow \quad (A^1 = \text{write}(A^0, i^1, v_i^1)) \\
\land \quad \neg DUMP^1 &\quad \rightarrow \quad (A^1 = A^0) \\
\land \quad DUMP^1 &\quad \rightarrow \quad (t^1 - t^0 = 0) \\
\land \quad \ldots \\
\land \quad \text{WAIT}^1 &\quad \rightarrow \quad (t^1 - t^0 > 0) \\
\land \quad \ldots \\
\land \quad \text{DUMP}^N &\quad \rightarrow \quad \ldots \\
\land \quad \ldots
\end{align*}
\]
**Ex.: Planning with Resources [62]**

- SAT-based planning augmented with numerical constraints
- Straightforward to encode into SMT($\mathcal{LRA}$)
- Goal: find a plan minimizing some resource consumption (time, money, gasoline, ...)

**Example (sketch) [62]**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\text{Deliver})$</td>
<td>$\land$ // goal</td>
</tr>
<tr>
<td>$(\text{MaxLoad})$</td>
<td>$\land$ // load constraint</td>
</tr>
<tr>
<td>$(\text{MaxFuel})$</td>
<td>$\land$ // fuel constraint</td>
</tr>
<tr>
<td>$(\text{Move} \rightarrow \text{MinFuel})$</td>
<td>$\land$ // move requires fuel</td>
</tr>
<tr>
<td>$(\text{Move} \rightarrow \text{Deliver})$</td>
<td>$\land$ // move implies delivery</td>
</tr>
<tr>
<td>$(\text{GoodTrip} \rightarrow \text{Deliver})$</td>
<td>$\land$ // a good trip requires</td>
</tr>
<tr>
<td>$(\text{GoodTrip} \rightarrow \text{AllLoaded})$</td>
<td>$\land$ // a full delivery</td>
</tr>
<tr>
<td>$(\text{MaxLoad} \rightarrow (\text{load} \leq 30))$</td>
<td>$\land$ // load limit</td>
</tr>
<tr>
<td>$(\text{MaxFuel} \rightarrow (\text{fuel} \leq 15))$</td>
<td>$\land$ // fuel limit</td>
</tr>
<tr>
<td>$(\text{MinFuel} \rightarrow (\text{fuel} \geq 7 + 0.5 \times \text{load}))$</td>
<td>$\land$ // fuel constraint</td>
</tr>
<tr>
<td>$(\text{AllLoaded} \rightarrow (\text{load} = 45))$</td>
<td>$\land$ //</td>
</tr>
</tbody>
</table>
Ex.: (LGDP/MILP) Strip-packing & Carpet-cutting
[29, 51, 53]

**Strip-packing**: Minimize the length $L$ of a strip of width $W$ while fitting $N$ rectangles (no overlap, no rotation) [29]. **Carpet-cutting**: w. rotation.

$$\varphi \overset{\text{def}}{=} (\text{cost} = L) \land \bigwedge_{i \in N} (L \geq x_i + L_i)$$
$$\land \bigwedge_{i,j \in N, i < j} \left( (x_i + L_i \leq x_j) \lor (x_j + L_j \leq x_i) \lor (y_i - H_i \geq y_j) \lor (y_j - H_j \geq y_i) \right)$$
$$\land \bigwedge_{i \in N} (x_i \leq \text{ub} - L_i) \land \bigwedge_{i \in N} (x_i \geq 0)$$
$$\land \bigwedge_{i \in N} (H_i \leq y_i) \land \bigwedge_{i \in N} (W \geq y_i) \land \bigwedge_{i \in N} (y_i \geq 0)$$
Ex.: (LGDP/MILP) Zero-Wait Jobshop Scheduling [29, 51, 53]

Given a set $I$ of jobs which must be scheduled sequentially on a set $J$ of consecutive stages with zero-wait transfer between them, minimize the makespan $M$ [47].

\[
\varphi \overset{\text{def}}{=} \left( \text{cost} = M \right) \land \bigwedge_{i \in I} \left( M \geq s_i + \sum_{j \in J_i} t_{ij} \right) \land \bigwedge_{i \in I} \left( s_i \geq 0 \right) \\
\land \bigwedge_{j \in C_{ik}, i, k \in I, i < k} \left( s_i + \sum_{m \in J_i, m \leq j} t_{im} \leq s_k + \sum_{m \in J_k, m < j} t_{km} \right)
\lor \left( s_k + \sum_{m \in J_k, m \leq j} t_{km} \leq s_i + \sum_{m \in J_i, m < j} t_{im} \right)
\]
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Optimization Modulo Theories: General Case

Ingredients

- a SMT formula $\varphi$ in some background theory $T = T_{\leq} \cup \bigcup_i T_i$
  - $\bigcup_i T_i$ may be empty
  - $T_{\leq}$ has a predicate $\preceq$ representing a total order
- a $T_{\leq}$-variable/term “cost” occurring in $\varphi$

Optimization Modulo $T_{\leq} \cup \bigcup_i T_i$ (OMT($T_{\leq} \cup \bigcup_i T_i$))

The problem of finding a model $M$ for $\varphi$ whose value of cost is minimum according to $\preceq$.

- maximization dual
Optimization Modulo Theories with \textit{LIRA} costs

**Ingredients**

- an \textit{SMT} formula $\varphi$ on $\textit{LIRA} \cup \mathcal{T}$
  - $\textit{LIRA}$ can be $\textit{LRA}$, $\textit{LIA}$ or a combination of both
  - $\mathcal{T} \overset{\text{def}}{=} \bigcup_i \mathcal{T}_i$, possibly empty
  - $\textit{LIRA}$ and $\mathcal{T}_i$ disjoint Nelson-Oppen theories
- a \textit{LIRA} variable [term] “cost” occurring in $\varphi$
- (optionally) two constant numbers $lb$ (\textit{lower bound}) and $ub$ (\textit{upper bound}) s.t. $lb \leq cost < ub$ ($lb$, $ub$ may be $\mp\infty$)

Optimization Modulo Theories with $\textit{LIRA}$ costs ($\text{OMT}(\textit{LIRA} \cup \mathcal{T})$)

Find a model for $\varphi$ whose value of $\text{cost}$ is minimum.
- maximization dual

We first restrict to the case $\textit{LIRA} = \textit{LRA}$ and $\bigcup_i \mathcal{T}_i = \emptyset$ ($\text{OMT}(\textit{LRA})$).
Optimization Modulo Theories with $\mathcal{LRA}$ costs

Ingredients

- an SMT formula $\varphi$ on $\mathcal{LRA} \cup \mathcal{T}$
  - $\mathcal{LIRA}$ can be $\mathcal{LRA}$, $\mathcal{LIA}$ or a combination of both
  - $\mathcal{T} \overset{\text{def}}{=} \bigcup_i \mathcal{T}_i$, possibly empty
  - $\mathcal{LRA}$ and $\mathcal{T}_i$ disjoint Nelson-Oppen theories
- a $\mathcal{LRA}$ variable [term] “cost” occurring in $\varphi$
- (optionally) two constant numbers $lb$ (lower bound) and $ub$ (upper bound) s.t. $lb \leq \text{cost} < ub$ ($lb$, $ub$ may be $\mp\infty$)

Optimization Modulo Theories with $\mathcal{LRA}$ costs ($\text{OMT(\mathcal{LRA} \cup \mathcal{T})}$)

Find a model for $\varphi$ whose value of cost is minimum.

- maximization dual

We first restrict to the case $\mathcal{LIRA} = \mathcal{LRA}$ and $\bigcup_i \mathcal{T}_i = \{\}$ ($\text{OMT(\mathcal{LRA})}$).
Solving OMT($\mathcal{LRA}$) [52, 53]

**General idea**
Combine standard SMT and LP minimization techniques.

**Offline Schema**
- Minimizer: based on the Simplex $\mathcal{LRA}$-solver by [25]
  - Handles strict inequalities
- Search Strategies:
  - Linear-Search strategy
  - Mixed Linear/Binary strategy
A toy example (linear search)

[w. pure-literal filt. \(\implies\) partial assignments]

- OMT(\(\mathcal{LRA}\)) problem:
  \[
  \varphi \overset{\text{def}}{=} (\neg A_1 \lor (2x + y \geq -2)) \land (A_1 \lor (x + y \geq 3)) \land (\neg A_2 \lor (4x - y \geq -4)) \land (A_2 \lor (2x - y \geq -6)) \land (\text{cost} < -0.2) \land (\text{cost} < -1.0) \land (\text{cost} < -2.0)
  \]

- \(\mu\) = \[
  \begin{align*}
  &A_1, \neg A_1, \ A_2, \neg A_2, \\
  &\ (4x - y \geq -4), \\
  &\ (x + y \geq 3), \\
  &\ (2x + y \geq -2), \\
  &\ (2x - y \geq -6) \\
  &\ (\text{cost} < -0.2) \\
  &\ (\text{cost} < -1.0) \\
  &\ (\text{cost} < -2.0)
  \end{align*}
  \]
A toy example (linear search)

[w. pure-literal filt. $\implies$ partial assignments]

- **OMT($\mathcal{LRA}$) problem:**
  \[
  \varphi \overset{\text{def}}{=} \left( \neg A_1 \lor (2x + y \geq -2) \right) \\
  \land \left( A_1 \lor (x + y \geq 3) \right) \\
  \land \left( \neg A_2 \lor (4x - y \geq -4) \right) \\
  \land \left( A_2 \lor (2x - y \geq -6) \right) \\
  \land (\text{cost} < -0.2) \\
  \land (\text{cost} < -1.0) \\
  \land (\text{cost} < -2.0)
  \]

- **cost** \(\overset{\text{def}}{=} x\)

- **$\mu$** = \[
  \left\{ A_1, \neg A_1, \ A_2, \neg A_2, \\
  (4x - y \geq -4), \\
  (x + y \geq 3), \\
  (2x + y \geq -2), \\
  (2x - y \geq -6) \\
  (\text{cost} < -0.2) \\
  (\text{cost} < -1.0) \\
  (\text{cost} < -2.0) \right\}
  \]

$\implies$ SAT, \(\text{min} = -0.2\)
A toy example (linear search)

[w. pure-literal filt. $\implies$ partial assignments]

- **OMT($LRA$) problem:**
  \[
  \varphi \overset{\text{def}}{=} (\neg A_1 \lor (2x + y \geq -2)) \\
  \land ( A_1 \lor (x + y \geq 3)) \\
  \land (\neg A_2 \lor (4x - y \geq -4)) \\
  \land ( A_2 \lor (2x - y \geq -6)) \\
  \land (\text{cost} < -0.2) \\
  \land (\text{cost} < -1.0) \\
  \land (\text{cost} < -2.0)
  \]

- **cost** $\overset{\text{def}}{=} x$

- **$\mu = \{\}$**
  \[
  \begin{align*}
  A_1, \neg A_1, & \quad A_2, \neg A_2, \\
  (4x - y \geq -4), & \quad (x + y \geq 3), \\
  (2x + y \geq -2), & \quad (2x - y \geq -6) \\
  (\text{cost} < -0.2), & \quad (\text{cost} < -1.0), \\
  (\text{cost} < -2.0)
  \end{align*}
  \]

$\implies$ SAT, $\min = -1.0$
A toy example (linear search)

[w. pure-literal filt. \(\iff\) partial assignments]

- **OMT(\(\mathcal{LRA}\)) problem:**
  \[
  \phi \overset{\text{def}}{=} \neg A_1 \lor (2x + y \geq -2)
  \land ( A_1 \lor (x + y \geq 3))
  \land (\neg A_2 \lor (4x - y \geq -4))
  \land ( A_2 \lor (2x - y \geq -6))
  \land (\text{cost} < -0.2)
  \land (\text{cost} < -1.0)
  \land (\text{cost} < -2.0)
\]

- **\(\mu\):**
  \[
  \mu = \left\{ A_1, \neg A_1, A_2, \neg A_2, \\
               (4x - y \geq -4), \\
               (x + y \geq 3), \\
               (2x + y \geq -2), \\
               (2x - y \geq -6) \\
               (\text{cost} < -0.2) \\
               (\text{cost} < -1.0) \\
               (\text{cost} < -2.0) \right\}
  \]

\(\implies\) SAT, \(\min = -2.0\)
A toy example (linear search)

[w. pure-literal filt. \(\Rightarrow\) partial assignments]

- **OMT(\(\mathcal{LRA}\)) problem:**
  \[
  \varphi \overset{\text{def}}{=} (\neg A_1 \lor (2x + y \geq -2)) \land (A_1 \lor (x + y \geq 3)) \land (\neg A_2 \lor (4x - y \geq -4)) \land (A_2 \lor (2x - y \geq -6)) \land (\text{cost} < -0.2) \land (\text{cost} < -1.0) \land (\text{cost} < -2.0)
  \]
  \[
  \text{cost} \overset{\text{def}}{=} x
  \]

\[
\mu = \begin{cases} 
A_1, \neg A_1, & A_2, \neg A_2, \\
(4x - y \geq -4), & (x + y \geq 3), \\
(2x + y \geq -2), & (2x - y \geq -6) \\
(\text{cost} < -0.2) & (\text{cost} < -1.0) \\
(\text{cost} < -2.0) & 
\end{cases}
\]

\[\Rightarrow \text{UNSAT, min} = -2.0\]
Offline Schema: Mixed Linear/Binary-Search Strategy

Input: \( \langle \varphi, \text{cost, lb, ub} \rangle \) // lb can be \(-\infty\), ub can be \(+\infty\)

\( l \leftarrow \text{lb}; u \leftarrow \text{ub}; M \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \neg(\text{cost < lb}, \text{cost < ub}) \);

while \( (l < u) \) do
Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:** \(\langle \varphi, \text{cost}, \text{lb}, \text{ub}\rangle\) // \(\text{lb}\) can be \(-\infty\), \(\text{ub}\) can be \(+\infty\)

\(l \leftarrow \text{lb}; u \leftarrow \text{ub}; \mathcal{M} \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \text{lb}), (\text{cost} < \text{ub})\};\)

while (\(l < u\)) do

  if (BinSearchMode()) then // Binary-search Mode

  else // Linear-search Mode

return \(\langle \mathcal{M}, u \rangle\)
Offline Schema: Mixed Linear/Binary-Search Strategy

\textbf{Input:} \langle \varphi, \text{cost}, \text{lb}, \text{ub} \rangle \ // \text{ub can be } +\infty, \text{lb can be } -\infty

\begin{align*}
l & \leftarrow \text{lb}; \\
u & \leftarrow \text{ub}; \\
M & \leftarrow \emptyset; \\
\varphi & \leftarrow \varphi \cup \{ \neg (\text{cost} < \text{lb}), (\text{cost} < \text{ub}) \};
\end{align*}

\textbf{while} (l < u) \textbf{do}

\begin{align*}
\textbf{if} \ (\text{BinSearchMode}()) \textbf{ then} \ // \text{Binary-search Mode} \\
\langle \text{res}, \mu \rangle & \leftarrow \text{SMT.IncrementalSolve}(\varphi);
\end{align*}

\begin{align*}
\textbf{else} \ // \text{Linear-search Mode} \\
\langle \text{res}, \mu \rangle & \leftarrow \text{SMT.IncrementalSolve}(\varphi);
\end{align*}

\textbf{return} \langle M, u \rangle
Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:** \( \langle \varphi, \text{cost}, \text{lb}, \text{ub} \rangle \) // lb can be \(-\infty\), ub can be \(+\infty\)

\[ l \leftarrow \text{lb}; u \leftarrow \text{ub}; \mathcal{M} \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \text{lb}), (\text{cost} < \text{ub})\}; \]

**while** (\( l < u \)) **do**

**if** (BinSearchMode()) **then** // Binary-search Mode

**else** // Linear-search Mode

\[ \langle \text{res}, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi); \]

**if** (res = SAT) **then**

\[ \langle \mathcal{M}, u \rangle \leftarrow \text{LRA-Solver.Minimize(cost, } \mu); \]

\[ \varphi \leftarrow \varphi \cup \{\text{(cost} < \text{u})\}; \]

**else** {res = UNSAT}

\[ l \leftarrow u; \]

**else**

\[ l \leftarrow \text{pivot}; \]

\[ \varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \text{pivot})\} \cup \{\neg(\text{cost} < \text{pivot})\}; \]

**return** \( \langle \mathcal{M}, u \rangle \)
Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:** \( \langle \varphi, \text{cost}, \text{lb}, \text{ub} \rangle \) // lb can be \(-\infty\), ub can be \(+\infty\)

\( l \leftarrow \text{lb}; u \leftarrow \text{ub}; M \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \text{lb}), (\text{cost} < \text{ub})\} \);

**while** (\( l < u \)) **do**

  **if** (BinSearchMode()) **then** // Binary-search Mode

  **else** // Linear-search Mode

  \( \langle \text{res}, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi); \)

  **if** (\( \text{res} = \text{SAT} \)) **then**

  **else** \( \{\text{res} = \text{UNSAT}\} \)

  \( l \leftarrow u; \)

**return** \( \langle M, u \rangle \)
Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:** \( \langle \varphi, \text{cost, lb, ub} \rangle \) // lb can be \(-\infty\), ub can be \(+\infty\)

\( l \leftarrow \text{lb}; u \leftarrow \text{ub}; M \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \text{lb}), \neg(\text{cost} < \text{ub})\}; \)

**while** \((l < u)\) **do**

**if** (BinSearchMode()) **then** // Binary-search Mode

\( \text{pivot} \leftarrow \text{ComputePivot}(l, u); \)

\( \varphi \leftarrow \varphi \cup \{\text{cost} < \text{pivot}\}; \)

\( \langle \text{res, } \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi); \)

**else** // Linear-search Mode

\( \langle \text{res, } \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi); \)

\( \text{if} \left(\text{res} = \text{SAT}\right) \leftarrow \text{LRA-Solver.Minimize}(\text{cost}, \mu); \)

\( \varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \mu)\}; \)

\( \text{else} \left(\text{res} = \text{UNSAT}\right) \leftarrow \left(\text{cost} < \text{pivot} \right) \not\in \text{SMT.ExtractUnsatCore}(\varphi); \)

\( l \leftarrow u; \)

\( \text{else} \left(\text{cost} < \text{pivot} \right) \in \text{SMT.ExtractUnsatCore}(\varphi); \)

\( l \leftarrow \text{pivot}; \)

\( \varphi \leftarrow \varphi \cup \{\neg(\text{cost} < \text{pivot})\}; \)

**return** \( \langle M, u \rangle \)
Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:** \(\langle \varphi, \text{cost, lb, ub} \rangle\) // lb can be \(-\infty\), ub can be \(+\infty\)

\(l \leftarrow \text{lb} \); \(u \leftarrow \text{ub}\); \(\mathcal{M} \leftarrow \emptyset\); \(\varphi \leftarrow \varphi \cup \{\neg (\text{cost} < \text{lb}), (\text{cost} < \text{ub})\}\);

**while** \((l < u)\) **do**

  **if** (BinSearchMode()) **then** // Binary-search Mode

  pivot \(\leftarrow\) ComputePivot\((l, u)\);

  \(\varphi \leftarrow \varphi \cup \{(\text{cost} < \text{pivot})\};\)

  \(\langle \text{res, } \mu \rangle \leftarrow\) SMT.IncrementalSolve\((\varphi)\);

  **else** // Linear-search Mode

  **if** (res = SAT) **then**

  \(\langle \mathcal{M}, u \rangle \leftarrow\) LRA-Solver.Minimize\((\text{cost, } \mu)\);

  \(\varphi \leftarrow \varphi \cup \{(\text{cost} < u)\};\)

  **else** \{res = UNSAT\}

  return \(\langle \mathcal{M}, u \rangle\)
Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:** \( \langle \phi, \text{cost, lb, ub}\rangle \) // lb can be \(-\infty\), ub can be \(+\infty\)

\[ l \leftarrow \text{lb}; u \leftarrow \text{ub}; M \leftarrow \emptyset; \phi \leftarrow \phi \cup \{\neg (\text{cost} < \text{lb}), (\text{cost} < \text{ub})\}; \]

**while** \((l < u)\) **do**

  **if** (BinSearchMode()) **then** // Binary-search Mode
    \[ \text{pivot} \leftarrow \text{ComputePivot}(l, u); \]
    \[ \phi \leftarrow \phi \cup \{(\text{cost} < \text{pivot})\}; \]
    \[ \langle \text{res}, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\phi); \]
  **else** // Linear-search Mode
    \[ \langle \text{res}, \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\phi); \]
    **if** (\(\text{res} = \text{SAT}\)) **then**
    **else** \(\{\text{res} = \text{UNSAT}\}\)
      **if** ((\(\text{cost} < \text{pivot}\)) \(\notin \text{SMT.ExtractUnsatCore}(\phi)\)) **then**
        \[ l \leftarrow u; \]
      **else**
      **return** \(\langle M, u \rangle\)
Offline Schema: Mixed Linear/Binary-Search Strategy

**Input:** \( \langle \varphi, \text{cost, } lb, ub \rangle \) // \( lb \) can be \(-\infty\), \( ub \) can be \(+\infty\)

\( l \leftarrow lb; u \leftarrow ub; M \leftarrow \emptyset; \varphi \leftarrow \varphi \cup \{ \neg \text{(cost < } lb), (\text{cost < } ub) \}; \)

**while** (\( l < u \)) **do**

  **if** (BinSearchMode()) **then** // Binary-search Mode

    pivot \( \leftarrow \text{ComputePivot}(l, u); \)
    \( \varphi \leftarrow \varphi \cup \{ (\text{cost < pivot}) \}; \)
    \( \langle \text{res, } \mu \rangle \leftarrow \text{SMT.IncrementalSolve}(\varphi); \)

  **else** // Linear-search Mode

    **if** (\( \text{res = SAT} \)) **then**

    **else** \( \{ \text{res = UNSAT} \} \)

      **if** ((\( \text{cost < pivot} \)) \( \notin \text{SMT.ExtractUnsatCore}(\varphi) \)) **then**

      **else**

        \( l \leftarrow \text{pivot}; \)
        \( \varphi \leftarrow (\varphi \setminus \{ (\text{cost < pivot}) \}) \cup \{ \neg (\text{cost < pivot}) \}; \)

  **end**

**return** \( \langle M, u \rangle \)
The Minimizer

Minimizer embedded within the Simplex-based \( \mathcal{LRA} \)-solver by [25]
- Minimization by standard Simplex techniques

Strict Inequalities

Temporally treated as non-strict inequalities:
- if minimum cost \( \text{min} \) lays only on non-strict inequalities, \( \text{min} \) is a solution
- otherwise, for some \( \delta > 0 \) there exists a solution for every cost \( c \in [\text{min}, \text{min} + \delta] \)

If \( \text{min} \) is a non-strict minimum, then \((\text{cost} \leq \text{min})\) is added to \( \varphi \).
Binary vs. Linear search

Beware of Zeno: pure binary search can cause infinite partitioning

E.g. if no solution in $[-1, 0[$, then
$[-1, 0[, [-1/2, 0[, [-1/4, 0[, [-1/8, 0[, \ldots$

SMT solver may find a conflict set $\eta \cup (\text{cost} < \text{pivot})$ even if
$\varphi \backslash \{(\text{cost} < \text{pivot})\}$ is $\mathcal{LRA}$-inconsistent

Solution: Binary-search interleaved with linear-search
(Mixed Linear/Binary Search Strategy)

Note: Binary search not “obviously faster” than linear search

- Binary search: typically smaller number of range-restriction steps
- Linear search: average smaller cost of each range-restriction steps (unsatisfiable calls typically much harder than sat. ones)
Binary vs. Linear search

Beware of Zeno: pure binary search can cause infinite partitioning

-1 - 1/2 - 1/4 - 1/8 - 1/16 0

- E.g. if no solution in \([-1, 0]\), then \([-1, 0], [-1/2, 0], [-1/4, 0], [-1/8, 0], \ldots\)
- SMT solver may find a conflict set \(\eta \cup \{\text{cost } < \text{pivot}\}\) even if \(\varphi \setminus \{\text{cost } < \text{pivot}\}\) is \(\mathcal{LRA}\)-inconsistent
- Solution: Binary-search interleaved with linear-search (Mixed Linear/Binary Search Strategy)

Note: Binary search not “obviously faster” than linear search

- Binary search: typically smaller number of range-restriction steps
- Linear search: average smaller cost of each range-restriction steps (unsatisfiable calls typically much harder than sat. ones)
Binary vs. Linear search

Beware of Zeno: pure binary search can cause infinite partitioning

-1  -1/2  -1/4  -1/8  -1/16  0

- E.g. if no solution in \([-1, 0]\], then \([-1, 0], [-1/2, 0], [-1/4, 0], [-1/8, 0], \ldots\)
- SMT solver may find a conflict set \(\eta \cup (\text{cost} < \text{pivot})\) even if \(\varphi \setminus \{(\text{cost} < \text{pivot})\}\) is \(\mathcal{LRA}\)-inconsistent
- Solution: Binary-search interleaved with linear-search
  (Mixed Linear/Binary Search Strategy)

Note: Binary search not “obviously faster” than linear search

- Binary search: typically smaller number of range-restriction steps
- Linear search: average smaller cost of each range-restriction steps (unsatisfiable calls typically much harder than sat. ones)
Termination & Correctness

Termination

The linear search procedure terminates:
- Finite number of satisfiable truth assignments $\mu_i$
- No truth assignment $\mu_i$ generated twice
  - guaranteed by computing the minimum cost $m_i$ of $\mu_i$ and learning (cost $< m_i$)

$\implies$ also the mixed linear/binary search procedure terminates

Correctness

The procedure returns the minimum cost
- Explores the whole space of satisfiable truth assignments
- For every satisfiable truth assignment, Minimize finds the minimum cost
Some Enhancements [52, 53, 16]

- After invoking the minimizer and learning \((\text{cost} < m_i)\)
  - Invoke \(\mathcal{LRA}\text{-solver.solve}(\mu_i \land (\text{cost} < m_i)) \Rightarrow \text{conflict set } \eta_i\)
    - and learn also \(\lnot \eta_i\)
  - Binary mode: learn also \((\text{cost} < \text{pivot}_i)\) to reuse previously learned clauses in the form \(\lnot (\text{cost} < \text{pivot}_i) \lor C\)

- Tightening of conflicts on binary search [52, 53, 16]
  - when \(\varphi \land (\text{cost} < \text{pivot}_i)\) fails, look for tighter conflict \(\lnot (\text{cost} < M_i)\) s.t. \(M_i > \text{pivot}_i\)

- Adaptive Mixed Linear/Binary-Search Strategy:
  BinSearchMode() chooses according to \(\frac{\Delta_{\text{ub}}}{\Delta \# \text{conflicts}}\)
From OMT($\mathcal{LRA}$) to OMT($\mathcal{LRA} \cup \mathcal{T}$)

OMT($\mathcal{LRA}$) procedure extended for handling $\mathcal{LRA} \cup \mathcal{T}$-formulas $\varphi$:

For free if SMT solver handles $\mathcal{LRA} \cup \mathcal{T}$-solving by *Delayed Theory Combination* [18] or Model-based Combination [23], splitting negated interface equalities $\neg(x_i = x_j)$ into $((x_i < x_j) \lor (x_i > x_j))$:

- Truth assignments $\mu' \overset{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{\text{eid}} \cup \mu_{\mathcal{T}}$ s.t. $\mu' \models \varphi$
  - $\mu_{\text{eid}}$ is a set containing interface equalities $(x_i = x_j)$, disequalities $\neg(x_i = x_j)$ and one inequality in $\{(x_i < x_j), (x_i > x_j)\}$ for every disequality in $\mu_{\text{eid}}$

- $\mathcal{LRA}$-solver.solve invoked on $\mu'_{\mathcal{LRA}}$

  - $\mu'_{\mathcal{LRA}} \overset{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{\text{ei}}$ obtained from $\mu_{\text{eid}}$ by dropping disequalities

$\Rightarrow \mathcal{LRA}$-solver.minimize invoked on $\langle \text{cost}, \mu'_{\mathcal{LRA}} \rangle$
From OMT($\mathcal{LRA}$) to OMT($\mathcal{LRA} \cup T$)

OMT($\mathcal{LRA}$) procedure extended for handling $\mathcal{LRA} \cup T$-formulas $\varphi$:

For free if SMT solver handles $\mathcal{LRA} \cup T$-solving by Delayed Theory Combination [18] or Model-based Combination [23], splitting negated interface equalities $\neg(x_i = x_j)$ into $((x_i < x_j) \lor (x_i > x_j))$:

- Truth assignments $\mu' \overset{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{\text{eid}} \cup \mu_T$ s.t. $\mu' \models \varphi$
  - $\mu_{\text{eid}}$ is a set containing interface equalities $(x_i = x_j)$, disequalities $\neg(x_i = x_j)$ and one inequality in $\{(x_i < x_j), (x_i > x_j)\}$ for every disequality in $\mu_{\text{eid}}$

- $\mathcal{LRA}$-solver.solve invoked on $\mu'_{\mathcal{LRA}}$
  - $\mu'_{\mathcal{LRA}} \overset{\text{def}}{=} \mu_{\mathcal{LRA}} \cup \mu_{\text{ei}}$ obtained from $\mu_{\text{eid}}$ by dropping disequalities

$\Rightarrow$ $\mathcal{LRA}$-solver.minimize invoked on $\langle \text{cost}, \mu'_{\mathcal{LRA}} \rangle$
OMT($\mathcal{LRA} \cup T$) procedures extended to $\mathcal{LIA}$ and mixed $\mathcal{LRA}/\mathcal{LIA}$ costs [16, 55]

$LRA/LIA$-solvers enhanced with ILP minimization techniques (branch & bound, cutting planes, backjumping, ...)

Note: with $\mathcal{LIA}$
- ILP minimization often expensive
- no “Zeno” problem for binary search
- in principle, if problem is lower-bounded, the ILP minimizer is not necessary

tradeoff between LP, (in)complete ILP minimization, binary search and Boolean Search [16, 55]
Truncated Branch and Bound

Observations:
- branch & bound can be expensive in degenerate cases
- optimality not truly necessary

Idea:
always stop B&B after first iteration, even if cost value is not guaranteed to be optimal.

Trade-off:
- less expensive minimization procedure on Integers
- risk of CDCL generating same $\mu$ multiple times
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Call OMT incrementally
- e.g., in BMC with parametric systems [53]

Intuition
In OMT, all learned clauses are either $T$-lemmas, or derive from $T$-lemmas and the original formulas, or are in the form $(\text{cost} < \text{min})$.

$\implies$ exploit incrementality of SMT solvers, in two alternative ways:

(i) drop the $(\text{cost} < \text{min})$ from one OMT call to the other

(ii) assert fresh variable $S$ at each OMT call, and learn $\neg S \lor (\text{cost} < \text{min})$ instead of $(\text{cost} < \text{min})$

$\implies$ can reuse learned clauses from OMT call to the other,
(included these in the form $\neg (\text{cost} < \text{min}_{\text{old}}) \lor C$ as soon as $\text{min}_{\text{cur}} \leq \text{min}_{\text{old}}$.)
OMT with Independent Objectives (Boxed OMT) [38, 55]

The problem: $\langle \varphi, \{\text{cost}_1, \ldots, \text{cost}_k\} \rangle$ [38]

Given $\langle \varphi, C \rangle$ s.t.:

- $\varphi$ is the input formula
- $C \overset{\text{def}}{=} \{\text{cost}_1, \ldots, \text{cost}_k\}$ is a set of $\text{LIRA}$-terms on variables in $\varphi$,

$\langle \varphi, C \rangle$ is the problem of finding a set of independent $\text{LIRA}$-models $M_1, \ldots, M_k$ s.t. s.t. each $M_i$ makes $\text{cost}_i$ minimum.

Notes

- derives from SW verification problems [38]
- equivalent to $k$ independent problems $\langle \varphi, \text{cost}_1 \rangle, \ldots, \langle \varphi, \text{cost}_k \rangle$
- intuition: share search effort for the different objectives
- generalizes to $\text{OMT}(\text{LIRA} \cup T)$ straightforwardly
OMT with Multiple Objectives [38, 16, 55]

Solution

- Intuition: when a $\mathcal{T}$-consistent satisfying assignment $\mu$ is found,
  
  foreach $\text{cost}_i$
  
  $\min_i := \min\{\min_i, \mathcal{T}\text{solver.minimize}(\mu, \text{cost}_i)\}$;
  
  learn $\bigvee_i (\text{cost}_i < \min_i)$;  // $(\text{cost}_i < -\infty) \equiv \bot$
  
  proceed until UNSAT;

- Notice:
  
  for each $\mu$, guaranteed improvement of at least one $\min_i$
  
  in practice, for each $\mu$, multiple cost$_i$ minima are improved

- Implemented improvements:
  
  (a) drop previous clauses $\bigvee_i (\text{cost}_i < \min_i)$
  
  (b) $(\text{cost}_i < \min_i)$ pushed in $\mu$ first: if $\mathcal{T}$-inconsistent, skip minimization
  
  (c) learn $\neg(\text{cost}_i < \min_i) \lor (\text{cost}_i < \min_i^{\text{old}})$, s.t. $\min_i^{\text{old}}$ previous $\min_i$
  
  $\implies$ reuse previously-learned clauses like $\neg(\text{cost}_i < \min_i^{\text{old}}) \lor C$
Boxed OMT: Example [38, 55]

\[ \varphi = (1 \leq y) \land (y \leq 3) \land (((1 \leq x) \land (x \leq 3)) \lor (x \geq 4)) \land (\text{cost}_1 = -y) \land (\text{cost}_2 = -x - y) \]

\[ \mu_1 = \{ (1 \leq y), (y \leq 3), (1 \leq x), (x \leq 3) \} \implies \text{SAT} \implies [-3, -6] \]
\[ \implies \text{learn} \quad \{ (\text{cost}_1 < -3) \lor (\text{cost}_2 < -6) \} \]

\[ \mu_2 = \{ (1 \leq y), (y \leq 3), (x \geq 4) \} \implies \text{SAT} \implies [-3, -\infty] \]
\[ \implies \text{learn} \quad \{ (\text{cost}_1 < -3) \} \]
\[ \implies \text{UNSAT} \]
OMT with Lexicographic Combination of Objectives

[16]

The problem

Find one optimal model $\mathcal{M}$ minimizing $\textit{costs} \overset{\text{def}}{=} \textit{cost}_1, \textit{cost}_2, \ldots, \textit{cost}_k$ lexicographically.

Solution

Intuition:

\{ \textit{minimize} \textit{cost}_1 \}

\textit{when UNSAT}

\{ \textit{substitute unit clause} (\textit{cost}_1 < \textit{min}_1) \textit{with} (\textit{cost}_1 = \textit{min}_1) \}

\{ \textit{minimize} \textit{cost}_2 \}

\ldots
OMT with Other forms of Objective Combination

OMT with Min-Max [Max-Min] optimization

Given \( \langle \varphi, \{\text{cost}_1, \ldots, \text{cost}_k\} \rangle \), find a solution which minimizes the maximum value among \( \{\text{cost}_1, \ldots, \text{cost}_k\} \). (Max-Min dual.)

- Frequent in some applications (e.g. [53, 59])

\[ \implies \text{encode into OMT} (\mathcal{LIRA} \cup \mathcal{T}) \text{ problem} \]
\[ \{ \varphi \land \land_i (\text{cost}_i \leq \text{cost}), \text{cost} \} \quad \text{s.t. cost fresh.} \]

OMT with linear combinations of costs

Given \( \langle \varphi, \{\text{cost}_1, \ldots, \text{cost}_k\} \rangle \) and a set of weights \( \{w_1, \ldots, w_k\} \), find a solution which minimizes \( \sum_i w_i \cdot \text{cost}_i \).

\[ \implies \text{encode into OMT} (\mathcal{LIRA} \cup \mathcal{T}) \text{ problem} \]
\[ \{ \varphi \land (\text{cost} = \sum_i w_i \cdot \text{cost}_i), \text{cost} \} \quad \text{s.t. cost fresh.} \]

These objectives can be composed with other OMT(\( \mathcal{LIRA} \)) objectives.
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
OMT(\(LRA \cup T\)) vs. SMT with PB costs (& MaxSMT)

SMT + PB costs (& MaxSMT) can be encoded into OMT(\(LRA \cup T\)):

\[
\begin{align*}
&\text{minimize} \quad \sum_j w_j \cdot A_j \quad /\!(\sum_j \text{ite}(A_j, w_j, 0)) \\
&s.t. \quad \varphi \\
\Downarrow \\
&\text{minimize} \quad \sum_j x_j \\
&s.t. \quad \varphi \land \land_j (A_j \rightarrow (x_j = w_j)) \land (\neg A_j \rightarrow (x_j = 0)) \\
&\quad \land \land_j ((x_j \geq 0) \land (x_j \leq w_j))
\end{align*}
\]

but not vice versa!

- SMT + PB costs finds the minimum-cost \(T\)-satisfiable assignment
  \(\Rightarrow\) search for minimum is purely Boolean

- OMT(\(LIRA \cup T\)) finds the \(T\)-satisfiable assignment whose minimum cost is minimum
  \(\Rightarrow\) search for minimum involves two dimensions: Boolean and arithmetical
OMT(\(\mathcal{LRA} \cup T\)) vs. SMT with PB costs (\& MaxSMT)

SMT + PB costs (\& MaxSMT) can be encoded into OMT(\(\mathcal{LRA} \cup T\)):

\[
\begin{align*}
\text{minimize} & \quad \sum_j w_j \cdot A_j \quad /\!\!/ (\sum_j \text{ite}(A_j, w_j, 0)) \\
\text{s.t.} & \quad \varphi \\
\Downarrow \\
\text{minimize} & \quad \sum_j x_j \\
\text{s.t.} & \quad \varphi \land \bigwedge_j (A_j \rightarrow (x_j = w_j)) \land (\neg A_j \rightarrow (x_j = 0)) \\
& \quad \land \bigwedge_j ((x_j \geq 0) \land (x_j \leq w_j))
\end{align*}
\]

but not vice versa!

- SMT + PB costs finds the minimum-cost \(T\)-satisfiable assignment
  \(\implies\) search for minimum is purely Boolean
- OMT(\(\mathcal{LIRA} \cup T\)) finds the \(T\)-satisfiable assignment whose minimum cost is minimum
  \(\implies\) search for minimum involves two dimensions: Boolean and arithmetical
Remark: range constraints “$x_j \geq 0 \land x_j \leq w_j$”

\[ OMT + PB : \quad \sum_j w_j \cdot A_j, \quad w_i > 0 \quad // (\sum_j \text{ite}(A_j, w_j, 0)) \]
\[ \Downarrow \]
\[ \sum_j x_j, \quad x_j \text{ fresh} \]
\[ \text{s.t.} \quad \ldots \land \bigwedge_j (A_j \rightarrow (x_j = w_j)) \land (\neg A_j \rightarrow (x_j = 0)) \land (x_j \geq 0) \land (x_j \leq w_j) \]

Range constraints “$x_j \geq 0 \land x_j \leq w_j$” logically redundant, but essential for efficiency:

- Without range constraints, the SMT solver can detect the violation of a bound only after all $A_i$’s are assigned:
  Ex: $w_1 = 4$, $w_2 = 7$, $\sum_{i=1} x_i < 10$, $A_1 = A_2 = \top$, $A_i = \ast \ \forall i > 2$.
- With range constraints, the SMT solver detects the violation as soon as the assigned $A_i$’s violate a bound $\implies$ drastic pruning of the search

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]
Remark: range constraints “\((x_j \geq 0) \land (x_j \leq w_j)\)”

\[
\text{OMT + PB : } \sum_j w_j \cdot A_j, \ w_i > 0 \ // (\sum_j \text{ite}(A_j, w_j, 0)) \\
\downarrow \\
\sum_j x_j, \ x_j \text{ fresh} \\
\text{s.t. } \ldots \land \bigwedge_j (A_j \rightarrow (x_j = w_j)) \land (\neg A_j \rightarrow (x_j = 0)) \\
\land (x_j \geq 0) \land (x_j \leq w_j)
\]

Range constraints “\((x_j \geq 0) \land (x_j \leq w_j)\)” logically redundant, but essential for efficiency:

- Without range constraints, the SMT solver can detect the violation of a bound only after all \(A_i\)’s are assigned:
  Ex: \(w_1 = 4, \ w_2 = 7, \ \sum_{i=1} x_i < 10, \ A_1 = A_2 = \top, \ A_i = * \ \forall i > 2\).

- With range constraints, the SMT solver detects the violation as soon as the assigned \(A_i\)’s violate a bound \(\implies\) drastic pruning of the search

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]
Remark: range constraints \((x_j \geq 0) \land (x_j \leq w_j)\)

\[
\text{OMT} + \text{PB} : \quad \sum_j w_j \cdot A_j, \quad w_i > 0 \quad \// (\sum_j \text{ite}(A_j, w_j, 0))
\]

\[
\implies \sum_j x_j, \quad x_j \text{ fresh}
\]

s.t. \[\ldots \land \land_j (A_j \rightarrow (x_j = w_j)) \land (\neg A_j \rightarrow (x_j = 0)) \land (x_j \geq 0) \land (x_j \leq w_j)\]

Range constraints \((x_j \geq 0) \land (x_j \leq w_j)\) logically redundant, but essential for efficiency:

- Without range constraints, the SMT solver can detect the violation of a bound only after all \(A_i\)'s are assigned:
  Ex: \(w_1 = 4, \ w_2 = 7, \sum_{i=1} x_i < 10, \ A_1 = A_2 = \top, \ A_i = \ast \ \forall i \geq 2.\)

- With range constraints, the SMT solver detects the violation as soon as the assigned \(A_i\)'s violate a bound \(\implies\) drastic pruning of the search

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]
Remark: range constraints "\((x_j \geq 0) \land (x_j \leq w_j)\)"

\[
\text{OMT} + \text{PB} : \quad \sum_j w_j \cdot A_j, \quad w_i > 0 \quad // (\sum_j \text{ite}(A_j, w_j, 0))
\]

\[
\quad \Downarrow
\]

\[
\sum_j x_j, \quad x_j \text{ fresh}
\]

\[
\text{s.t.} \quad \ldots \land \bigwedge_j (A_j \rightarrow (x_j = w_j)) \land (\neg A_j \rightarrow (x_j = 0))
\]

\[
\land (x_j \geq 0) \land (x_j \leq w_j)
\]

Range constraints "\((x_j \geq 0) \land (x_j \leq w_j)\)" logically redundant, but essential for efficiency:

- Without range constraints, the SMT solver can detect the violation of a bound only after all \(A_i\)'s are assigned:
  Ex: \(w_1 = 4, \ w_2 = 7, \sum_{i=1} x_i < 10, \ A_1 = A_2 = \top, \ A_i = \ast \ \forall i \geq 2\).
- With range constraints, the SMT solver detects the violation as soon as the assigned \(A_i\)'s violate a bound \(\implies\text{drastic pruning of the search}\).

Further improvement: Enhance encoding of PB constraints/MaxSMT with sorting networks [56]
### Alternative Solution: conversion into $\text{SMT}(\mathcal{T})$

- SAT + PB can be efficiently encoded into SAT [26]
- $\Rightarrow$ encode SMT($\mathcal{T}$) + PB into SMT($\mathcal{T}$)
- similar idea implemented in [16, 15] for cardinality constraints

### Alternative Solution: Leverage SAT+PB

- develop a “modulo theory” version of your favourite PB-solver
- afiak, no implementation available

### Alternative Solution: $\text{SMT}(\mathcal{T} \cup \mathcal{C})$ [20]

- $\mathcal{C}$ is an ad-hoc “theory of costs”
- a specialized very-fast theory-solver for $\mathcal{C}$ added
  - very fast & aggressive search pruning and theory-propagation
### Alternative Solution: conversion into SMT(\(\mathcal{T}\))
- SAT + PB can be efficiently encoded into SAT [26]
- ∴ encode SMT(\(\mathcal{T}\)) + PB into SMT(\(\mathcal{T}\))
- similar idea implemented in [16, 15] for cardinality constraints

### Alternative Solution: Leverage SAT+PB
- develop a “modulo theory” version of your favourite PB-solver
- afaik, no implementation available

### Alternative Solution: SMT(\(\mathcal{T} \cup \mathcal{C}\)) [20]
- \(\mathcal{C}\) is an ad-hoc “theory of costs”
- a specialized very-fast theory-solver for \(\mathcal{C}\) added
  - very fast & aggressive search pruning and theory-propagation
SMT/OMT with Pseudo-Boolean Constraints & Costs:

Alternative Solution: conversion into SMT\( (\mathcal{T}) \)
- SAT + PB can be efficiently encoded into SAT [26]
- encode SMT\( (\mathcal{T}) \) + PB into SMT\( (\mathcal{T}) \)
- similar idea implemented in [16, 15] for cardinality constraints

Alternative Solution: Leverage SAT+PB
- develop a “modulo theory” version of your favourite PB-solver
- afaik, no implementation available

Alternative Solution: SMT\( (\mathcal{T} \cup \mathcal{C}) \) [20]
- \( \mathcal{C} \) is an ad-hoc “theory of costs”
- a specialized very-fast theory-solver for \( \mathcal{C} \) added
  - very fast & aggressive search pruning and theory-propagation
A “Theory of cost” $\mathcal{C}$

- $M$ variables $cost^i$
- predicate “bound cost” $BC(cost^i, k)$ (“$cost^i \leq k$”)
- predicate “incur cost” $IC(cost^i, j, c^i_j)$ (“the $j$th addend of $cost^i$ is $c^i_j$”)

"$cost^i = \sum_{j=1}^{N^i} c^i_j \cdot A^i_j$, s.t. $cost^i \in (l^i, u^i]$"

encoded as:

$\neg BC(cost^i, l^i) \land BC(cost^i, u^i) \land \bigwedge_{j=1}^{N^i}(A^i_j \leftrightarrow IC(cost^i, j, c^i_j))$
for each $i$, $\mathcal{C}$-solver maintains the current values of the incurred costs

$$cost^i \overset{\text{def}}{=} \sum_{IC(cost^i, j, c^i_j) \leftarrow \top} c^i_j,$$

the total cost of all unassigned IC’s

$$\Delta cost^i \overset{\text{def}}{=} \sum \{ IC(cost^i, j, c^i_j) \text{ unassigned} \} c^i_j,$$

and of the range $[lb^i, ub^i]$

1. \( BC(cost^i, c) \leftarrow \top / \bot \implies \text{update } [lb^i, ub^i] \)
2. \( IC(cost^i, j, c^i_j) \leftarrow \top \implies cost^i \leftarrow cost^i + c^i_j \)
   \quad \quad \quad \quad IC(cost^i, j, c^i_j) \leftarrow \bot \implies \Delta cost^i \leftarrow \Delta cost^i - c^i_j \)
3. \( cost^i > ub^i \implies \text{conflict} \)
4. \( cost^i + \Delta cost^i \leq lb^i \implies \text{conflict} \)
5. \( IC(cost^i, j, c^i_j) \leftarrow \top \text{ causes 3. } \implies \text{propagate } \neg IC(cost^i, j, c^i_j) \)
6. \( IC(cost^i, j, c^i_j) \leftarrow \bot \text{ causes 4. } \implies \text{propagate } IC(cost^i, j, c^i_j) \)

very fast:

- add one constraint & solve: 1 sum + 1 comparison
- theory propagation: linear in the number of propagated literals
for each $i$, $\mathcal{C}$-solver maintains the current values of the incurred costs

$$cost^i \overset{\text{def}}{=} \sum IC(cost^i, j, c^i_j) \leftarrow \top c^i_j,$$

the total cost of all unassigned IC’s

$$\Delta cost^i \overset{\text{def}}{=} \sum \{IC(cost^i, j, c^i_j) \text{ unassigned} \} c^i_j,$$

and of the range $[lb^i, ub^i]$

1. $BC(cost^i, c) \leftarrow \top / \bot \implies \text{update } [lb^i, ub^i]$
2. $IC(cost^i, j, c^i_j) \leftarrow \top \implies cost^i \leftarrow cost^i + c^i_j$
   
   $IC(cost^i, j, c^i_j) \leftarrow \bot \implies \Delta cost^i \leftarrow \Delta cost^i - c^i_j$
3. $cost^i > ub^i \implies \text{conflict}$
4. $cost^i + \Delta cost^i \leq lb^i \implies \text{conflict}$
5. $IC(cost^i, j, c^i_j) \leftarrow \top \text{ causes 3. } \implies \text{propagate } \neg IC(cost^i, j, c^i_j)$
6. $IC(cost^i, j, c^i_j) \leftarrow \bot \text{ causes 4. } \implies \text{propagate } IC(cost^i, j, c^i_j)$

- very fast:
  - add one constraint & solve: 1 sum + 1 comparison
  - theory propagation: linear in the number of propagated literals
MaxSAT Modulo Theories (MaxSMT) I

[Partial Weighted] MaxSMT: The problem

Input: \( \varphi_T^h, \varphi_T^s \): resp. sets of hard and (weighted) soft \( T \)-clauses;

Output: a maximum-weight set of soft \( T \)-clauses \( \psi_T^s \) s.t.
\( \psi_T^s \subseteq \varphi_T^s \) and \( \varphi_T^h \cup \psi_T^s \) is \( T \)-satisfiable

MaxSMT vs. SMT with PB cost functions

MaxSMT \( \langle \varphi_T^h, \varphi_T^s \rangle \) encodable into SMT with PB costs \( \langle \varphi_T', \text{cost} \rangle \):

\[
\varphi_T' \overset{\text{def}}{=} \varphi_T^h \cup \bigcup_{C^T_j \in \varphi_T^s} \{ (A_j \lor C^T_j) \}; \quad \text{cost} \overset{\text{def}}{=} \sum_{C^T_j \in \varphi_T^s} w_j \cdot A_j,
\]

SMT with PB costs \( \langle \varphi_T', \text{cost} \overset{\text{def}}{=} \sum_j w_j \cdot A_j \rangle \) encodable into MaxSMT:

\[
\varphi_T^h \overset{\text{def}}{=} \varphi_T'; \quad \varphi_T^s \overset{\text{def}}{=} \bigcup_j \{ (\neg A_j) \}.
\]
MaxSAT Modulo Theories (MaxSMT) I

[Partial Weighted] MaxSMT: The problem

Input: \( \varphi^T_h, \varphi^T_s \): resp. sets of hard and (weighted) soft \( T \)-clauses;

Output: a maximum-weight set of soft \( T \)-clauses \( \psi^T_s \) s.t. \( \psi^T_s \subseteq \varphi^T_s \) and \( \varphi^T_h \cup \psi^T_s \) is \( T \)-satisfiable

MaxSMT vs. SMT with PB cost functions

MaxSMT \( \langle \varphi^T_h, \varphi^T_s \rangle \) encodable into SMT with PB costs \( \langle \varphi^{T'}, \text{cost} \rangle \):

\[
\varphi^{T'} \overset{\text{def}}{=} \varphi^T_h \cup \bigcup_{C_j^T \in \varphi^T_s} \{(A_j \lor C_j^T)\}; \quad \text{cost} \overset{\text{def}}{=} \sum_{C_j^T \in \varphi^T_s} w_j \cdot A_j,
\]

SMT with PB costs \( \langle \varphi^{T'}, \text{cost} \overset{\text{def}}{=} \sum_j w_j \cdot A_j \rangle \) encodable into MaxSMT:

\[
\varphi^T_h \overset{\text{def}}{=} \varphi^{T'}; \quad \varphi^T_s \overset{\text{def}}{=} \bigcup_{j} \{(-A_j)\} \underbrace{w_j}_{\text{w}};
\]
MaxSAT Modulo Theories (MaxSMT) II

Solution: encode into $\text{OMT}(\mathcal{LRA})$ [44, 52, 53]
- can be composed with other objective functions

Alternative Solution: Leverage MaxSAT
- develop a “modulo theory” version of your favourite MaxSAT solver
- a few implementations available [4, 5, 15]

A “Modular” Approach to MaxSMT [21]
- Idea: Combine an SMT and a MaxSAT solver:
  $\text{MaxSMT} = \text{MaxSAT} + \text{SMT}$
MaxSAT Modulo Theories (MaxSMT) II

Solution: encode into OMT(\(\mathcal{LRA}\)) [44, 52, 53]
- can be composed with other objective functions

Alternative Solution: Leverage MaxSAT
- develop a “modulo theory” version of your favourite MaxSAT solver
- a few implementations available [4, 5, 15]

A “Modular” Approach to MaxSMT [21]
- Idea: Combine an SMT and a MaxSAT solver:
  \[\text{MaxSMT} = \text{MaxSAT} + \text{SMT}\]
MaxSAT Modulo Theories (MaxSMT) II

Solution: encode into OMT(\(\mathcal{LRA}\)) [44, 52, 53]
- can be composed with other objective functions

Alternative Solution: Leverage MaxSAT
- develop a “modulo theory” version of your favourite MaxSAT solver
- a few implementations available [4, 5, 15]

A “Modular” Approach to MaxSMT [21]
- Idea: Combine an SMT and a MaxSAT solver: \(\text{MaxSMT} = \text{MaxSAT} + \text{SMT}\)
A Modular Approach for MaxSMT($\varphi^T_h, \varphi^T_s$) [21]

Input: $\varphi^T_h, \varphi^T_s$ // sets of hard and (weighted) soft $T$-clauses

$\langle \varphi^B_h, \varphi^B_s \rangle \leftarrow T2B (\langle \varphi^T_h, \varphi^T_s \rangle)$;

$\Theta^T \leftarrow \emptyset$; // current set of $T$-lemmas

$\psi^T_s \leftarrow \varphi^T_s$; // current approximation of the result

while (SMT.Solve($\varphi^T_h \cup \psi^T_s \cup \Theta^T$) = UNSAT) do

    $\Theta^T \leftarrow \Theta^T \cup \text{SMT.GetTLemmas}()$; $\Theta^B \leftarrow T2B (\Theta^T)$;

    $\psi^B_s \leftarrow \text{MaxSAT}(\varphi^B_h \cup \Theta^B, \varphi^B_s)$; $\psi^T_s \leftarrow B2T (\psi^B_s)$;

return $\psi^T_s$;

Based on the cyclic interaction of an SMT and a MaxSAT solver:

- SMT.Solve used as a generator of sets of $T$-lemmas $\Theta^T_0, \Theta^T_1, \ldots$ provide the information to rule-out $T$-inconsistent solutions
- MaxSAT used to extract minimum-cost clause sets $\psi^B_{s,0}, \psi^B_{s,1}, \ldots$
  - works on Boolean abstractions $\varphi^B_h, \varphi^B_s$ plus the $T$-lemmas $\Theta^B_i$
A Modular Approach for MaxSMT($\phi_T^h, \phi_T^s$) [21]

Input: $\phi_T^h, \phi_T^s$ // sets of hard and (weighted) soft $T$-clauses

$\langle \phi_B^h, \phi_B^s \rangle \leftarrow T2B (\langle \phi_T^h, \phi_T^s \rangle)$;

$\Theta_T \leftarrow \emptyset$; // current set of $T$-lemmas

$\psi_T^s \leftarrow \phi_T^s$; // current approximation of the result

while ( SMT.Solve($\phi_T^h \cup \psi_T^s \cup \Theta_T$) = UNSAT) do

$\Theta_T \leftarrow \Theta_T \cup$ SMT.GetTLemmas(); $\Theta_B \leftarrow T2B (\Theta_T)$;

$\psi_B^s \leftarrow \text{MaxSAT}(\phi_B^h \cup \Theta_B, \phi_B^s)$; $\psi_T^s \leftarrow B2T (\psi_B^s)$;

return $\psi_T^s$;

Based on the cyclic interaction of an SMT and a MaxSAT solver:

- SMT.Solve used as a generator of sets of $T$-lemmas $\Theta_T^0, \Theta_T^1, ...$ → provide the information to rule-out $T$-inconsistent solutions

- MaxSAT used to extract minimum-cost clause sets $\psi_B^s, 0, \psi_B^s, 1, ...$

- works on Boolean abstractions $\phi_B^h, \phi_B^s$ plus the $T$-lemmas $\Theta_B^i$
A Modular Approach for MaxSMT($\varphi_h^T, \varphi_s^T$) [21]

Input: $\varphi_h^T, \varphi_s^T$ // sets of hard and (weighted) soft $T$-clauses

$\langle \varphi_h^B, \varphi_s^B \rangle \leftarrow T2B (\langle \varphi_h^T, \varphi_s^T \rangle)$;

$\Theta^T \leftarrow \emptyset$; // current set of $T$-lemmas

$\psi_s^T \leftarrow \varphi_s^T$; // current approximation of the result

while (SMT.Solve($\varphi_h^T \cup \psi_s^T \cup \Theta^T$) = UNSAT) do

$\Theta^T \leftarrow \Theta^T \cup \text{SMT.GetTLemmas}()$; $\Theta^B \leftarrow T2B (\Theta^T)$;

$\psi_s^B \leftarrow \text{MaxSAT}(\varphi_h^B \cup \Theta^B, \varphi_s^B)$; $\psi_s^T \leftarrow B2T (\psi_s^B)$;

return $\psi_s^T$;

Based on the cyclic interaction of an SMT and a MaxSAT solver:

- SMT.Solve used as a generator of sets of $T$-lemmas $\Theta^T_0, \Theta^T_1, ...$
  
  \(\Rightarrow\) provide the information to rule-out $T$-inconsistent solutions

- MaxSAT used to extract minimum-cost clause sets $\psi_s^B, 0, \psi_s^B, 1, ...$
  
  works on Boolean abstractions $\varphi_h^B, \varphi_s^B$ plus the $T$-lemmas $\Theta_i^B$
A toy example I

\[ \varphi^T_h \overset{\text{def}}{=} \emptyset \]

\[ \varphi^T_s \overset{\text{def}}{=} \left\{ \begin{array}{l}
C_0 : ((x \leq 0)) \quad [4] \\
C_1 : ((x \leq 1)) \quad [3] \\
C_2 : ((x \geq 2)) \quad [2] \\
C_3 : ((x \geq 3)) \quad [6]
\end{array} \right\} \]

\[ \varphi^T_h \overset{\text{def}}{=} \emptyset \]

\[ \varphi^B_h \overset{\text{def}}{=} \left\{ \begin{array}{l}
(A_0) \quad [4]
\end{array} \right\} \]

\[ \varphi^B_s \overset{\text{def}}{=} \left\{ \begin{array}{l}
(A_0) \quad [3] \\
(A_1) \quad [2] \\
(A_2) \quad [6]
\end{array} \right\} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi^T_h \cup \varphi^T_s \) is:

\[ \Theta^T_* = \left\{ \begin{array}{l}
\theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\
\theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3))
\end{array} \right\} \]

\[ \Theta^B_* = \left\{ \begin{array}{l}
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3)
\end{array} \right\} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta^T_i )</th>
<th>( \psi^T_{s,i} )</th>
<th>Weight(( \psi^T_{s,i} ))</th>
<th>SMT(( \varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{: }</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_4}</td>
<td>{: }</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{\theta_4, \theta_6}</td>
<td>{: }</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{\theta_4, \theta_6, \theta_3}</td>
<td>{: }</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example I

\[ \varphi^T_h \overset{\text{def}}{=} \emptyset \]
\[ \varphi^T_s \overset{\text{def}}{=} \begin{cases} 
C_0 : (x \leq 0) & [4] \\
C_1 : (x \leq 1) & [3] \\
C_2 : (x \geq 2) & [2] \\
C_3 : (x \geq 3) & [6] 
\end{cases} \]
\[ \varphi^B_h \overset{\text{def}}{=} \emptyset \]
\[ \varphi^B_s \overset{\text{def}}{=} \begin{cases} 
(A_0) & [4] \\
(A_1) & [3] \\
(A_2) & [2] \\
(A_3) & [6] 
\end{cases} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi^T_h \cup \varphi^T_s \) is:

\[ \Theta^T_* = \begin{cases} 
\theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\
\theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) 
\end{cases} \]
\[ \Theta^B_* = \begin{cases} 
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3) 
\end{cases} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta^T_i )</th>
<th>( \psi^T_{s,i} )</th>
<th>Weight(( \psi^T_{s,i} ))</th>
<th>SMT(( \varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_4}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{\theta_4, \theta_6}</td>
<td>{C_0, C_1, C_2}</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{\theta_4, \theta_6, \theta_3}</td>
<td>{C_2, C_3}</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example I

\[ \varphi^T \overset{\text{def}}{=} \emptyset \]
\[ \varphi^s \overset{\text{def}}{=} \begin{cases} C_0 : ((x \leq 0)) & [4] \\ C_1 : ((x \leq 1)) & [3] \\ C_2 : ((x \geq 2)) & [2] \\ C_3 : ((x \geq 3)) & [6] \end{cases} \]
\[ \varphi^B \overset{\text{def}}{=} \emptyset \]
\[ \varphi^B \overset{\text{def}}{=} \begin{cases} (A_0) & [4] \\ (A_1) & [3] \\ (A_2) & [2] \\ (A_3) & [6] \end{cases} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi^T \cup \varphi^s \) is:

\[ \Theta^T = \begin{cases} \theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\ \theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\ \theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\ \theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\ \theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\ \theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) \end{cases} \]
\[ \Theta^B = \begin{cases} \neg A_0 \lor A_1 \\ \neg A_3 \lor A_2 \\ \neg A_0 \lor \neg A_2 \\ \neg A_3 \lor \neg A_2 \\ \neg A_1 \lor \neg A_2 \\ \neg A_1 \lor \neg A_3 \end{cases} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta^T_i )</th>
<th>( \psi^T_{s,i} )</th>
<th>Weight(( \psi^T_{s,i} ))</th>
<th>SMT(( \varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_4}</td>
<td>{ , C_1, C_2, C_3}</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{\theta_4, \theta_6}</td>
<td>{C_0, C_1, C_2, }</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{\theta_4, \theta_6, \theta_3}</td>
<td>{ , , C_2, C_3}</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example

\[ \varphi^T_h \overset{\text{def}}{=} \emptyset \]
\[ \varphi^T_s \overset{\text{def}}{=} \left\{ \begin{array}{l}
C_0 : ((x \leq 0)) \quad [4] \\
C_1 : ((x \leq 1)) \quad [3] \\
C_2 : ((x \geq 2)) \quad [2] \\
C_3 : ((x \geq 3)) \quad [6]
\end{array} \right\} \]
\[ \varphi^B_h \overset{\text{def}}{=} \emptyset \]
\[ \varphi^B_s \overset{\text{def}}{=} \left\{ \begin{array}{l}
(A_0) \quad [4] \\
(A_1) \quad [3] \\
(A_2) \quad [2] \\
(A_3) \quad [6]
\end{array} \right\} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi^T_h \cup \varphi^T_s \) is:

\[ \Theta^T_* = \left\{ \begin{array}{l}
\theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\
\theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3))
\end{array} \right\} \]
\[ \Theta^B_* = \left\{ \begin{array}{l}
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3)
\end{array} \right\} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta^T_i )</th>
<th>( \psi^T_{s,i} )</th>
<th>Weight(( \psi^T_{s,i} ))</th>
<th>SMT(( \varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{ ( C_0, C_1, C_2, C_3 ) }</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{ ( \theta_4 ) }</td>
<td>{ ( C_0, C_1, C_2, C_3 ) }</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{ ( \theta_4, \theta_6 ) }</td>
<td>{ ( C_0, C_1, C_2 ) }</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{ ( \theta_4, \theta_6, \theta_3 ) }</td>
<td>{ ( C_0, C_1, C_2, C_3 ) }</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example I

\[ \varphi^T_h \overset{\text{def}}{=} \emptyset \]

\[ \varphi^T_s \overset{\text{def}}{=} \begin{cases} 
C_0 : \((x \leq 0)\) & [4] \\
C_1 : \((x \leq 1)\) & [3] \\
C_2 : \((x \geq 2)\) & [2] \\
C_3 : \((x \geq 3)\) & [6] 
\end{cases} \]

\[ \varphi^B_h \overset{\text{def}}{=} \emptyset \]

\[ \varphi^B_s \overset{\text{def}}{=} \begin{cases} 
(A_0) & [4] \\
(A_1) & [3] \\
(A_2) & [2] \\
(A_3) & [6] 
\end{cases} \]

Notice that the set of all (minimal) \(T\)-lemmas on the \(T\)-atoms of \(\varphi^T_h \cup \varphi^T_s\) is:

\[ \Theta^T_* = \begin{cases} 
\theta_1 : \((-x \leq 0) \lor (x \leq 1)) \\
\theta_2 : \((-x \geq 3) \lor (x \geq 2)) \\
\theta_3 : \((-x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : \((-x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : \((-x \leq 1) \lor \neg(x \geq 2)) \\
\theta_6 : \((-x \leq 1) \lor \neg(x \geq 3)) 
\end{cases} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>(i)</th>
<th>(\Theta^T_i)</th>
<th>(\psi^T_{s,i})</th>
<th>Weight((\psi^T_{s,i}))</th>
<th>SMT((\varphi^T_h \cup \varphi^T_s \cup \Theta^T_i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_4}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{\theta_4, \theta_6}</td>
<td>{C_0, C_1, C_2, }</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{\theta_4, \theta_6, \theta_3}</td>
<td>{, C_2, C_3}</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example I

\[ \varphi_T^h \overset{\text{def}}{=} \emptyset \]

\[ \varphi_T^s \overset{\text{def}}{=} \begin{cases} 
C_0 : ((x \leq 0)) & [4] \\
C_1 : ((x \leq 1)) & [3] \\
C_2 : ((x \geq 2)) & [2] \\
C_3 : ((x \geq 3)) & [6] 
\end{cases} \]

\[ \varphi_B^h \overset{\text{def}}{=} \emptyset \]

\[ \varphi_B^s \overset{\text{def}}{=} \begin{cases} 
(A_0) & [4] \\
(A_1) & [3] \\
(A_2) & [2] \\
(A_3) & [6] 
\end{cases} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi_T^h \cup \varphi_T^s \) is:

\[ \Theta_T^* = \{ \theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\
\theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) \} \]

\[ \Theta_B^* = \{ (\neg A_0 \lor A_1) \\
\quad (\neg A_3 \lor A_2) \\
\quad (\neg A_0 \lor \neg A_2) \\
\quad (\neg A_0 \lor \neg A_3) \\
\quad (\neg A_1 \lor \neg A_2) \\
\quad (\neg A_1 \lor \neg A_3) \} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta_T^i )</th>
<th>( \psi_T^s,i )</th>
<th>Weight(( \psi_T^s,i ))</th>
<th>SMT(( \varphi_T^h \cup \psi_T^s,i \cup \Theta_T^i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{( \theta_4 )}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{( \theta_4, \theta_6 )}</td>
<td>{C_0, C_1, C_2, }</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{( \theta_4, \theta_6, \theta_3 )}</td>
<td>{ , C_2, C_3}</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example I

\[ \varphi_T^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_T^s \overset{\text{def}}{=} \begin{cases} 
    C_0 : ((x \leq 0)) & [4] \\
    C_1 : ((x \leq 1)) & [3] \\
    C_2 : ((x \geq 2)) & [2] \\
    C_3 : ((x \geq 3)) & [6] 
\end{cases} \]
\[ \varphi_B^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_B^s \overset{\text{def}}{=} \begin{cases} 
    (A_0) & [4] \\
    (A_1) & [3] \\
    (A_2) & [2] \\
    (A_3) & [6] 
\end{cases} \]

Notice that the set of all (minimal) \( \mathcal{T} \)-lemmas on the \( \mathcal{T} \)-atoms of \( \varphi_T^h \cup \varphi_T^s \) is:

\[ \Theta \overset{\text{def}}{=} \begin{cases} 
    \theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\
    \theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\
    \theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
    \theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
    \theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\
    \theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) 
\end{cases} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta_i^T )</th>
<th>( \psi_{s,i}^T )</th>
<th>Weight(( \psi_{s,i}^T ))</th>
<th>SMT(( \varphi_T^h \cup \psi_{s,i}^T \cup \Theta_i^T ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_4}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{\theta_4, \theta_6}</td>
<td>{C_0, C_1, C_2}</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{\theta_4, \theta_6, \theta_3}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example I

\[ \varphi_T^h \overset{\text{def}}{=} \emptyset \quad \varphi_T^s \overset{\text{def}}{=} \begin{cases} 
C_0 : ((x \leq 0)) & [4] \\
C_1 : ((x \leq 1)) & [3] \\
C_2 : ((x \geq 2)) & [2] \\
C_3 : ((x \geq 3)) & [6] 
\end{cases} \]

\[ \varphi_B^h \overset{\text{def}}{=} \emptyset \quad \varphi_B^s \overset{\text{def}}{=} \begin{cases} 
(A_0) & [4] \\
(A_1) & [3] \\
(A_2) & [2] \\
(A_3) & [6] 
\end{cases} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi_T^h \cup \varphi_T^s \) is:

\[ \Theta_T^* = \begin{cases} 
\theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : (\neg(x \leq 1) \lor (x \geq 2)) \\
\theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) 
\end{cases} \]

\[ \Theta_B^* = \begin{cases} 
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3) 
\end{cases} \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta_i^T )</th>
<th>( \psi_{s,i}^T )</th>
<th>Weight(( \psi_{s,i}^T ))</th>
<th>SMT(( \varphi_T^h \cup \psi_{s,i}^T \cup \Theta_i^T ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>( { C_0, C_1, C_2, C_3 } )</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{ \theta_4 }</td>
<td>{ , C_1, C_2, C_3 }</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{ \theta_4, \theta_6 }</td>
<td>{ C_0, C_1, C_2, }</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{ \theta_4, \theta_6, \theta_3 }</td>
<td>{ , , C_2, C_3 }</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example

$$\varphi^T_h \overset{\text{def}}{=} \emptyset$$
$$\varphi^T_s \overset{\text{def}}{=} \begin{cases} C_0 : ((x \leq 0)) \ [4] \\ C_1 : ((x \leq 1)) \ [3] \\ C_2 : ((x \geq 2)) \ [2] \\ C_3 : ((x \geq 3)) \ [6] \end{cases}$$
$$\varphi^B_h \overset{\text{def}}{=} \emptyset$$
$$\varphi^B_s \overset{\text{def}}{=} \begin{cases} (A_0) \ [4] \\ (A_1) \ [3] \\ (A_2) \ [2] \\ (A_3) \ [6] \end{cases}$$

Notice that the set of all (minimal) $T$-lemmas on the $T$-atoms of $\varphi^T_h \cup \varphi^T_s$ is:

$$\Theta^T_* = \begin{cases} \theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\ \theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\ \theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\ \theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\ \theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\ \theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) \end{cases}$$

$$\Theta^B_* = \begin{cases} (\neg A_0 \lor A_1) \\ (\neg A_3 \lor A_2) \\ (\neg A_0 \lor \neg A_2) \\ (\neg A_0 \lor \neg A_3) \\ (\neg A_1 \lor \neg A_2) \\ (\neg A_1 \lor \neg A_3) \end{cases}$$

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\Theta^T_i$</th>
<th>$\psi^T_{s,i}$</th>
<th>Weight($\psi^T_{s,i}$)</th>
<th>$SMT(\varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>${}$</td>
<td>${C_0, C_1, C_2, C_3}$</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>${\theta_4}$</td>
<td>${, C_1, C_2, C_3}$</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>${\theta_4, \theta_6}$</td>
<td>${C_0, C_1, C_2, }$</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>${\theta_4, \theta_6, \theta_3}$</td>
<td>${, C_2, C_3}$</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example I

\[ \phi^T_h \overset{\text{def}}{=} \emptyset \quad \psi^T_{s,i} \overset{\text{def}}{=} \emptyset \quad \phi^B_h \overset{\text{def}}{=} \emptyset \quad \psi^B_{s,i} \overset{\text{def}}{=} \emptyset \]

\[ \varphi^T_s \overset{\text{def}}{=} \{ C_0 : (x \leq 0) \}^{[4]} \quad \varphi^B_s \overset{\text{def}}{=} \{ (A_0) \}^{[4]} \]

\[ \varphi^T_s \overset{\text{def}}{=} \{ C_1 : (x \leq 1) \}^{[3]} \quad \varphi^B_s \overset{\text{def}}{=} \{ (A_1) \}^{[3]} \]

\[ \varphi^T_s \overset{\text{def}}{=} \{ C_2 : (x \geq 2) \}^{[2]} \quad \varphi^B_s \overset{\text{def}}{=} \{ (A_2) \}^{[2]} \]

\[ \varphi^T_s \overset{\text{def}}{=} \{ C_3 : (x \geq 3) \}^{[6]} \quad \varphi^B_s \overset{\text{def}}{=} \{ (A_3) \}^{[6]} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi^T_h \cup \varphi^T_s \) is:

\[ \Theta^T_* \overset{\text{def}}{=} \{ \theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \} \quad \Theta^B_* \overset{\text{def}}{=} \{ (\neg A_0 \lor A_1) \} \]

\[ \theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \quad (\neg A_0 \lor A_2) \]

\[ \theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \quad (\neg A_0 \lor \neg A_2) \]

\[ \theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \quad (\neg A_0 \lor \neg A_3) \]

\[ \theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \quad (\neg A_1 \lor \neg A_2) \]

\[ \theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) \quad (\neg A_1 \lor \neg A_3) \]

An "unlucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta^T_* )</th>
<th>( \psi^T_{s,i} )</th>
<th>Weight(( \psi^T_{s,i} ))</th>
<th>SMT(( \varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_* ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_4}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>11</td>
<td>UNSAT</td>
</tr>
<tr>
<td>2</td>
<td>{\theta_4, \theta_6}</td>
<td>{C_0, C_1, C_2}</td>
<td>9</td>
<td>UNSAT</td>
</tr>
<tr>
<td>3</td>
<td>{\theta_4, \theta_6, \theta_3}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example II

\[ \varphi_T^h \overset{\text{def}}{=} \emptyset \]

\[ \varphi_T^s \overset{\text{def}}{=} \begin{cases} 
C_0 : ((x \leq 0)) & [4] \\
C_1 : ((x \leq 1)) & [3] \\
C_2 : ((x \geq 2)) & [2] \\
C_3 : ((x \geq 3)) & [6]
\end{cases} \]

\[ \varphi_B^h \overset{\text{def}}{=} \emptyset \]

\[ \varphi_B^s \overset{\text{def}}{=} \begin{cases} 
(A_0) & [4] \\
(A_1) & [3] \\
(A_2) & [2] \\
(A_3) & [6]
\end{cases} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi_T^h \cup \varphi_T^s \) is:

\[ \Theta_T^* = \begin{cases} 
\theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\
\theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3))
\end{cases} \]

\[ \Theta_B^* = \begin{cases} 
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3)
\end{cases} \]

A "lucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta_T^i )</th>
<th>( \psi_T^{s,i} )</th>
<th>Weight(( \psi_T^{s,i} ))</th>
<th>SMT(( \varphi_T^h \cup \psi_T^{s,i} \cup \Theta_T^i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{ }</td>
<td>{ C_0, C_1, C_2, C_3 }</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{ \theta_1, \theta_2, \theta_5 }</td>
<td>{ , C_2, C_3 }</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example II

$$\varphi^T_h \overset{\text{def}}{=} \emptyset$$
$$\varphi^T_s \overset{\text{def}}{=} \begin{cases} C_0 : ((x \leq 0)) & [4] \\ C_1 : ((x \leq 1)) & [3] \\ C_2 : ((x \geq 2)) & [2] \\ C_3 : ((x \geq 3)) & [6] \end{cases}$$

Notice that the set of all (minimal) $T$-lemmas on the $T$-atoms of $\varphi^T_h \cup \varphi^T_s$ is:

$$\Theta^T_* = \begin{cases} \theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\ \theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\ \theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\ \theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\ \theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\ \theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) \end{cases}$$

A "lucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\Theta^T_i$</th>
<th>$\psi^T_{s,i}$</th>
<th>Weight($\psi^T_{s,i}$)</th>
<th>$SMT(\varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>${C_0, C_1, C_2, C_3}$</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>${\theta_1, \theta_2, \theta_5}$</td>
<td>${\ldots, C_2, C_3}$</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example II

\[ \varphi_T^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_T^s \overset{\text{def}}{=} \begin{cases} C_0 : ((x \leq 0)) \ [4] \\ C_1 : ((x \leq 1)) \ [3] \\ C_2 : ((x \geq 2)) \ [2] \\ C_3 : ((x \geq 3)) \ [6] \end{cases} \]

\[ \varphi_B^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_B^s \overset{\text{def}}{=} \begin{cases} (A_0) \ [4] \\ (A_1) \ [3] \\ (A_2) \ [2] \\ (A_3) \ [6] \end{cases} \]

Notice that the set of all (minimal) \(T\)-lemmas on the \(T\)-atoms of \(\varphi_T^h \cup \varphi_T^s\) is:

\[ \Theta_T^* = \begin{cases} \theta_1 : (\neg(x \leq 0) \lor (x \leq 1)) \\ \theta_2 : (\neg(x \geq 3) \lor (x \geq 2)) \\ \theta_3 : (\neg(x \leq 0) \lor \neg(x \geq 2)) \\ \theta_4 : (\neg(x \leq 0) \lor \neg(x \geq 3)) \\ \theta_5 : (\neg(x \leq 1) \lor \neg(x \geq 2)) \\ \theta_6 : (\neg(x \leq 1) \lor \neg(x \geq 3)) \end{cases} \]

\[ \Theta_B^* = \begin{cases} (\neg A_0 \lor A_1) \\ (\neg A_3 \lor A_2) \\ (\neg A_0 \lor \neg A_2) \\ (\neg A_0 \lor \neg A_3) \\ (\neg A_1 \lor \neg A_2) \\ (\neg A_1 \lor \neg A_3) \end{cases} \]

A "lucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>(i)</th>
<th>(\Theta_T^i)</th>
<th>(\psi_T^{s,i})</th>
<th>Weight((\psi_T^{s,i}))</th>
<th>SMT((\varphi_T^h \cup \psi_T^{s,i} \cup \Theta_T^i))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_1, \theta_2, \theta_5}</td>
<td>{ C_0, C_2, C_3 }</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example II

$$\varphi^T_h \overset{\text{def}}{=} \emptyset$$
$$\varphi^T_s \overset{\text{def}}{=} \left\{ \begin{array}{l}
C_0 : (x \leq 0) \quad [4] \\
C_1 : (x \leq 1) \quad [3] \\
C_2 : (x \geq 2) \quad [2] \\
C_3 : (x \geq 3) \quad [6]
\end{array} \right\}$$

$$\varphi^B_h \overset{\text{def}}{=} \emptyset$$
$$\varphi^B_s \overset{\text{def}}{=} \left\{ \begin{array}{l}
(A_0) \quad [4] \\
(A_1) \quad [3] \\
(A_2) \quad [2] \\
(A_3) \quad [6]
\end{array} \right\}$$

Notice that the set of all (minimal) $\mathcal{T}$-lemmas on the $\mathcal{T}$-atoms of $\varphi^T_h \cup \varphi^T_s$ is:

$$\Theta^T_* = \left\{ \begin{array}{l}
\theta_1 : (\neg (x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg (x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg (x \leq 0) \lor \neg (x \geq 2)) \\
\theta_4 : (\neg (x \leq 0) \lor \neg (x \geq 3)) \\
\theta_5 : (\neg (x \leq 1) \lor \neg (x \geq 2)) \\
\theta_6 : (\neg (x \leq 1) \lor \neg (x \geq 3))
\end{array} \right\}$$

$$\Theta^B_* = \left\{ \begin{array}{l}
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3)
\end{array} \right\}$$

A "lucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>$i$</th>
<th>$\Theta^T_i$</th>
<th>$\psi^T_{s,i}$</th>
<th>Weight($\psi^T_{s,i}$)</th>
<th>$\text{SMT}(\varphi^T_h \cup \psi^T_{s,i} \cup \Theta^T_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\emptyset$</td>
<td>${C_0, C_1, C_2, C_3}$</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>${\theta_1, \theta_2, \theta_5}$</td>
<td>${, C_2, C_3}$</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example II

\[ \varphi_T^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_T^s \overset{\text{def}}{=} \begin{cases} 
C_0 : (x \leq 0) \ [4] \\
C_1 : (x \leq 1) \ [3] \\
C_2 : (x \geq 2) \ [2] \\
C_3 : (x \geq 3) \ [6] 
\end{cases} \]

\[ \varphi_B^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_B^s \overset{\text{def}}{=} \begin{cases} 
(A_0) \ [4] \\
(A_1) \ [3] \\
(A_2) \ [2] \\
(A_3) \ [6] 
\end{cases} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi_T^h \cup \varphi_T^s \) is:

\[ \Theta_T^* = \begin{cases} 
\theta_1 : (\neg (x \leq 0) \lor (x \leq 1)) \\
\theta_2 : (\neg (x \geq 3) \lor (x \geq 2)) \\
\theta_3 : (\neg (x \leq 0) \lor \neg (x \geq 2)) \\
\theta_4 : (\neg (x \leq 0) \lor \neg (x \geq 3)) \\
\theta_5 : (\neg (x \leq 1) \lor \neg (x \geq 2)) \\
\theta_6 : (\neg (x \leq 1) \lor \neg (x \geq 3)) 
\end{cases} \]

\[ \Theta_B^* = \begin{cases} 
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3) 
\end{cases} \]

A "lucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta_T^i )</th>
<th>( \psi_T^s,i )</th>
<th>Weight(( \psi_T^s,i ))</th>
<th>SMT(( \varphi_T^h \cup \varphi_T^s \cup \Theta_T^i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{ ( C_0, C_1, C_2, C_3 ) }</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{ ( \theta_1, \theta_2, \theta_5 ) }</td>
<td>{ ( , C_2, C_3 ) }</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
A toy example II

\[ \varphi_T^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_T^s \overset{\text{def}}{=} \left\{ \begin{array}{ll}
C_0 : ((x \leq 0)) & [4] \\
C_1 : ((x \leq 1)) & [3] \\
C_2 : ((x \geq 2)) & [2] \\
C_3 : ((x \geq 3)) & [6]
\end{array} \right\} \]
\[ \varphi_B^h \overset{\text{def}}{=} \emptyset \]
\[ \varphi_B^s \overset{\text{def}}{=} \left\{ \begin{array}{ll}
(A_0) & [4] \\
(A_1) & [3] \\
(A_2) & [2] \\
(A_3) & [6]
\end{array} \right\} \]

Notice that the set of all (minimal) \( T \)-lemmas on the \( T \)-atoms of \( \varphi_T^h \cup \varphi_T^s \) is:

\[ \Theta_T^* = \left\{ \begin{array}{ll}
\theta_1 : & (\neg(x \leq 0) \lor (x \leq 1)) \\
\theta_2 : & (\neg(x \geq 3) \lor (x \geq 2)) \\
\theta_3 : & (\neg(x \leq 0) \lor \neg(x \geq 2)) \\
\theta_4 : & (\neg(x \leq 0) \lor \neg(x \geq 3)) \\
\theta_5 : & (\neg(x \leq 1) \lor \neg(x \geq 2)) \\
\theta_6 : & (\neg(x \leq 1) \lor \neg(x \geq 3))
\end{array} \right\} \]
\[ \Theta_B^* = \left\{ \begin{array}{ll}
(\neg A_0 \lor A_1) \\
(\neg A_3 \lor A_2) \\
(\neg A_0 \lor \neg A_2) \\
(\neg A_0 \lor \neg A_3) \\
(\neg A_1 \lor \neg A_2) \\
(\neg A_1 \lor \neg A_3)
\end{array} \right\} \]

A "lucky" possible execution of the algorithm is:

<table>
<thead>
<tr>
<th>( i )</th>
<th>( \Theta_T^i )</th>
<th>( \psi_T^s,i )</th>
<th>Weight(( \psi_T^s,i ))</th>
<th>SMT(( \varphi_T^h \cup \psi_T^s,i \cup \Theta_T^i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>{}</td>
<td>{C_0, C_1, C_2, C_3}</td>
<td>15</td>
<td>UNSAT</td>
</tr>
<tr>
<td>1</td>
<td>{\theta_1, \theta_2, \theta_5}</td>
<td>{C_0, \ldots, C_2, C_3}</td>
<td>8</td>
<td>SAT</td>
</tr>
</tbody>
</table>
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
OMT($\text{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

<table>
<thead>
<tr>
<th></th>
<th>Boolean formulas</th>
<th>Sets of $\text{LIRA}$ constraints</th>
<th>SMT($\text{LIRA}$)</th>
<th>SMT($\text{LIRA} \cup \bigcup \mathcal{T}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECISION (Satisfiability)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with PB cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZATION</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with linear cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

<table>
<thead>
<tr>
<th></th>
<th>Boolean formulas</th>
<th>Sets of $\mathcal{LIRA}$ constraints</th>
<th>SMT($\mathcal{LIRA}$)</th>
<th>SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DECISION</strong> (Satisfiability)</td>
<td></td>
<td></td>
<td></td>
<td>SMT($\mathcal{T}$)</td>
</tr>
<tr>
<td><strong>OPTIMIZATION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with PB cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>and constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OPTIMIZATION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with linear cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OMT($\mathcal{LIRA} \cup T$) captures lots of interesting problems

<table>
<thead>
<tr>
<th></th>
<th>Boolean formulas</th>
<th>Sets of $\mathcal{LIRA}$ constraints</th>
<th>SMT($\mathcal{LIRA}$)</th>
<th>SMT($\mathcal{LIRA} \cup \bigcup_i T$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DECISION</strong> (Satisfiability)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OPTIMIZATION</strong> with PB cost function and constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Weighted) MaxSAT</td>
<td>PB Opt.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OPTIMIZATION</strong> with linear cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
$\text{OMT}(\mathcal{LIRA} \cup \mathcal{T})$ captures lots of interesting problems

<table>
<thead>
<tr>
<th>Decision (Satisfiability)</th>
<th>Boolean formulas</th>
<th>Sets of $\mathcal{LIRA}$ constraints</th>
<th>SMT($\mathcal{LIRA}$)</th>
<th>SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTIMIZATION with PB cost function and constraints</td>
<td></td>
<td>MaxSMT and SMT($\mathcal{T}$) with PB cost funct.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZATION with linear cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OMT($\mathcal{LIRA} \cup \mathcal{T}$) captures lots of interesting problems

<table>
<thead>
<tr>
<th></th>
<th>Boolean formulas</th>
<th>Sets of $\mathcal{LIRA}$ constraints</th>
<th>SMT($\mathcal{LIRA}$)</th>
<th>SMT($\mathcal{LIRA} \cup \bigcup_i \mathcal{T}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DECISION (Satisfiability)</td>
<td></td>
<td>LP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZATION with PB cost function and constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZATION with linear cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OMT($\mathcal{LIRA} \cup T$) captures lots of interesting problems

<table>
<thead>
<tr>
<th></th>
<th>Boolean formulas</th>
<th>Sets of $\mathcal{LIRA}$ constraints</th>
<th>SMT($\mathcal{LIRA}$)</th>
<th>SMT($\mathcal{LIRA} \cup \bigcup_i T$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DECISION</strong> (Satisfiability)</td>
<td></td>
<td></td>
<td></td>
<td>ILP, MILP, DP, LGDP</td>
</tr>
<tr>
<td>OPTIMIZATION with PB cost function and constraints</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZATION with linear cost function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OMT(\(\text{LIRA} \cup \mathcal{T}\)) captures lots of interesting problems

<table>
<thead>
<tr>
<th></th>
<th>Boolean formulas</th>
<th>Sets of (\text{LIRA}) constraints</th>
<th>SMT((\text{LIRA}))</th>
<th>SMT((\text{LIRA} \cup \bigcup_i \mathcal{T}))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DECISION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(Satisfiability)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OPTIMIZATION</strong></td>
<td></td>
<td>with PB cost function and constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIMIZATION</td>
<td></td>
<td></td>
<td></td>
<td><strong>OMT((\text{LIRA} \cup \mathcal{T}))</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>with linear cost function</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OMT(\(LIRA \cup T\)) captures lots of interesting problems

<table>
<thead>
<tr>
<th></th>
<th>Boolean formulas</th>
<th>Sets of (LIRA) constraints</th>
<th>SMT((LIRA))</th>
<th>SMT((LIRA \cup \bigcup_i T))</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DECISION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Satisfiability)</td>
<td></td>
<td></td>
<td>LP</td>
<td>ILP, MILP, DP, LGDP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MaxSMT and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SMT((T))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OPTIMIZATION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with PB cost function</td>
<td>(Weighted)</td>
<td></td>
<td>SMT((T))</td>
<td></td>
</tr>
<tr>
<td>and constraints</td>
<td>MaxSAT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB Opt.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MaxSMT and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SMT((T))</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>OPTIMIZATION</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>with linear cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
(Finite Domain) Constraint Programming

**FDCP/MILP**
- Very efficient on (integer) linear arithmetic / combinatorial reasoning
- **Very efficient** handling of global constraints (e.g. all-different)
- Booleans typically represented as 0-1 integers
- (typically) finite precision arithmetic

**SMT/OMT**
- Very efficient on Boolean reasoning
- Supports other theories (*Array*, *Bit-Vectors*, *Strings*, ...)
- Incremental
- infinite precision arithmetic
- Other functionalities: all-smt, proofs, unsat-cores, interpolants, ...
Some OMT tools

- **BCLT** [44, 35]
  http://www.cs.upc.edu/~oliveras/bclt-main.html
- **OptiMathSAT** [52, 53, 55, 54, 57], on top of **MathSAT** [22]
  http://optimathsat.disi.unitn.it
- **SYMBA** [38], on top of **Z3** [24]
  https://bitbucket.org/arieg/symba/src
- **Z3** [16, 15], on top of **Z3** [24]
  http://z3.codeplex.com

More Recently:

- **Hazel** [40]. \( \Rightarrow \) \( \mathcal{BV} \), incremental
- **CEGIO** [7, 9] \( \Rightarrow \) counterexample guided inductive optimization
- **MaxHS-MSAT** [27] \( \Rightarrow \) *MaxSMT* with Implicit Hitting Set (IHS) algorithm
- **Puli** [33]. \( \Rightarrow \) *LIA* cost functions, (based on linear regression)
OMT Applications (OPTIMATHSAT)


**Requirements Engineering.** Constrained Goal Models with resources, preferences and goals [41, 42, 43]. ⇒ OPTIMATHSAT backend engine of CGM-TOOL [1]

**Machine Learning.** Inference & Learning in Hybrid domains [46, 60]. ⇒ OPTIMATHSAT backend engine of LMT tool [2]

**Quantum Annealing.** Solving SAT and MaxSAT with D-Wave 2000Q QAs [12, 13] ⇒ offline used of OPTIMATHSAT to generate optimal QUBO encodings of Boolean functions

**Formal Verification & Model Checking.** Synthesis of Barrier Certificates for Hybrid Dynamical Systems [48] ⇒ OPTIMATHSAT used as oracle to separate safe/unsafe regions starting from a simulation

**Scheduling.** Optimal sleep/wake-up scheduling for WSNs [32, 34, 33] ⇒ OPTIMATHSAT used to deal with increasingly denser WSNs [34]
OMT Applications (Other tools)

**Static Analysis.**
- Generation of Invariants and Proving Termination via *Constraint-based* method [19]
- Finding Inductive Invariants via *Local Policy Iteration* [30, 31]

**Formal Verification & Model Checking.**
- Computing Loop Iterations for Bounded Program Verification [39]

**Scheduling and Planning with Resources.**
- Optimal plans for multi-robot systems [36, 37]
- Task planning for smart factories [14]
- Optimal Job-Shop Scheduling with OMT [50]
- Synthesis Communication Schedules for Time Sensitive Networks [45]

**Software Security Engineering.**
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Field still far from maturity, lots of possible research directions:

- **Improve efficiency!**
- OMT on different theories, e.g.:
  - Bit vectors ([16, 40])
  - $\mathcal{NLA}(\mathbb{R})$
  - $\mathcal{NLA}(\mathbb{Z})$ ([35])
  - Floating point ([61])
- Exploit alternative SMT schemas (e.g., Model-Construction SMT)
- Hybrid techniques, integration with techniques in neighbour fields (MaxSAT, PB, CSP, MILP, CA, ...)
- Extensive empirical comparison wrt. techniques in neighbour fields (MaxSAT, PB, CSP, MILP, ...)
- Bridge SMT/OMT with CSP/COP (Minizinc)
Announcement

PHD POSITION available in Trento on “Advancing Optimization Modulo Theories”
The call will expire in a couple of months.

Please contact me if interested: roberto.sebastiani@unitn.it.
(Se also flier on the desk.)
“That's all Folks!”
References

   www.cgm-tool.eu.

[2] LMT.
   http://disi.unitn.it/~teso/lmt/lmt.tgz.

[3] WCET OMT.
   https://github.com/PatrickTrentin88/wcet_omt.


   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.


   Timed Automata.

   Counterexample guided inductive optimization based on satisfiability modulo theories.

    Cardinality Networks: a theoretical and empirical study.
    Constraints, 16(2):195–221, 2011.
Solving Multi-Objective Workflow Satisfiability Problems with Optimization Modulo Theories Techniques. 

Solving SAT and MaxSAT with a Quantum Annealer: Foundations and a Preliminary Report. 

Solving SAT and maxsat with a quantum annealer: Foundations, encodings, and preliminary results. 
Under submission for journal publication.

SMarTplan: a Task Planner for Smart Factories. 

νz - an optimizing SMT solver. 

νZ - Maximal Satisfaction with Z3. 
http://www.easychair.org/publications/?page=862275542.

Z3 - An Optimizing SMT Solver. 
References III

Efficient Theory Combination via Boolean Search.

Speeding up the Constraint-Based Method in Difference Logic.

Satisfiability modulo the theory of costs: Foundations and applications.

A Modular Approach to MaxSAT Modulo Theories.

The MathSAT 5 SMT Solver.

Z3: An efficient smt solver.

Z3: an efficient SMT solver.

A Fast Linear-Arithmetic Solver for DPLL(T).
In *CAV*, volume 4144 of *LNCS*, 2006.
References IV


[34] G. Kovásznai, B. Erdélyi, and C. Biró. 
Investigations of graph properties in terms of wireless sensor network optimization. 
In *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*, pages 1–8, Jan 2018.

Minimal-Model-Guided Approaches to Solving Polynomial Constraints and Extensions. 


[38] Y. Li, A. Albarghouthi, Z. Kincaid, A. Gurfinkel, and M. Chechik. 
Symbolic optimization with smt solvers. 

Computing Exact Loop Bounds for Bounded Program Verification. 

[40] A. Nadel and V. Ryvchin. 
Bit-Vector Optimization. 

Multi-objective reasoning with constrained goal models. 
In print. Published online 24 December 2016. DOI: [http://dx.doi.org/10.1007/s00766-016-0263-5](http://dx.doi.org/10.1007/s00766-016-0263-5).
Requirements Evolution and Evolution Requirements with Constrained Goal Models.

Modeling and Reasoning on Requirements Evolution with Constrained Goal Models.

On SAT Modulo Theories and Optimization Problems.

IEEE 802.1Qbv Gate Control List Synthesis Using Array Theory Encoding.

Learning Modulo Theories.

Modelling and computational techniques for logic based integer programming.

Simulation Based Computation of Certificates for Safety of Dynamical Systems.


OptiMathSAT: A Tool for Optimization Modulo Theories.  

[58] C. Sinz.  
Towards an Optimal CNF Encoding of Boolean Cardinality Constraints.  

Structured Learning Modulo Theories.  
To appear.

Structured learning modulo theories.  

Optimization Modulo the Theory of Floating-Point Numbers.  
To appear.

The LPSAT Engine & its Application to Resource Planning.  
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Solving OMT (LRA) [52, 53]

General idea
Combine standard SMT and LP minimization techniques.

Offline Schema
SMT solver and LP minimizer used as blackbox procedures.
⇒ no need to hack the code of the SMT solver

Inline Schema
Search for minimum integrated inside the CDCL loop of the SMT solver.
Solving OMT ($LRA$) [52, 53]

General idea
Combine standard SMT and LP minimization techniques.

Offline Schema
SMT solver and LP minimizer used as blackbox procedures.
\[\Rightarrow\] no need to hack the code of the SMT solver

Inline Schema
Search for minimum integrated inside the CDCL loop of the SMT solver.
Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of LRA-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment LRA-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion
Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of \texttt{LRA}-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment \texttt{LRA}-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion
Search for optimum integrated inside CDCL search schema

*Minimizer called incrementally (no restarting of $\mathcal{LRA}$-solver)*

- Learned clauses drive backjumping up to level 0
- Intermediate-assignment $\mathcal{LRA}$-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion
Search for optimum integrated inside CDCL search schema
Minimizer called incrementally (no restarting of LRA-solver)
Learned clauses drive backjumping up to level 0
Intermediate-assignment LRA-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion
Search for optimum integrated inside CDCL search schema
- Minimizer called incrementally (no restarting of $LRA$-solver)
- Learned clauses drive backjumping up to level 0
- Intermediate-assignment $LRA$-checking (early-pruning) plays the role of “bounding” in a Branch & Bound fashion
Range-minimization loop embedded within CDCL search schema

- Level 0: update pivot \( j \) and decide (\( \text{cost} < \text{pivot}_j \))
Inline Version: Binary-Search Strategy

- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot \( j \) and decide \( \text{cost} < \text{pivot}_j \)
Range-minimization loop embedded within CDCL search schema

Level 0: update pivot\_j and decide (cost < pivot\_j)
Range-minimization loop embedded within CDCL search schema

Level 0: update $\text{pivot}_j$ and decide $(\text{cost} < \text{pivot}_j)$
inline version: binary-search strategy

\[ \phi \land (\text{cost} < m_{i+1}) \]

- Range-minimization loop embedded within CDCL search schema
- Level 0: update pivot \( j \) and decide (cost < pivot \( j \))
Range-minimization loop embedded within CDCL search schema

Level 0: update pivot<sub>j</sub> and decide (cost < pivot<sub>j</sub>)
Range-minimization loop embedded within CDCL search schema

Level 0: update $pivot_j$ and decide $(cost < pivot_j)$
Inline Version: Binary-Search Strategy

- Range-minimization loop embedded within CDCL search schema
- Level 0: update \( \text{pivot}_j \) and decide \((\text{cost} < \text{pivot}_j)\)
Range-minimization loop embedded within CDCL search schema

Level 0: update pivot$_j$ and decide (cost < pivot$_j$)
1 Motivations
2 Optimization Modulo Theories with Linear-Arithmetic Objectives
3 OMT with Multiple and Combined Objectives
4 Relevant Subcases: OMT+PB & MaxSMT
5 Status of OMT
6 Current and Future Research Directions
7 Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Minimization of an unsigned Bit-Vector

Given a pair \( \langle \varphi, \text{cost} \rangle \), where \( \text{cost} \overset{\text{def}}{=} [\text{cost}[0], \ldots, \text{cost}[n - 1]] \) is an unsigned \( \mathcal{BV} \) of \( n \) bits:

- **Reduction to:**
  - Lexicographic OMT: \( \langle \varphi, \{ \text{cost}[0] \neq 0, \ldots, \text{cost}[n - 1] \neq 0 \} \rangle_{\mathcal{L}} \)
  - MaxSMT [16, 17]: \( \langle \varphi, \bigcup_{i=0}^{n-1} \langle \text{cost}[i] \neq 0, 1 \rangle \rangle \)

**OMT-based Approach:** linear-search, binary-search and adaptive-search

**Ad-Hoc Algorithms:**

- **OBV-WA [40]**
  - each cost[\( i \)] transformed into a *high-priority* decision variable
  - the *phase-saving* of each cost[\( i \)] initialized to 0

- **OBV-BS [40]**
  - binary search over the bits [cost[0], \ldots, cost[n - 1]]
  - at most \( n \) incremental calls to the underlying SMT solver

**Question:**

How to deal with other \( \mathcal{BV} \) goals?

- signed vs. unsigned
- maximization vs minimization
Example: encoding of a 8-bits Bit-Vector

**Unsigned:**

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Signed:** (Two’s complement)

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Attractor** *attr* for cost: when minimizing, it’s the **smallest** $BV$-value of the same sort of cost.

- it’s the ideal result of the optimization search
- depends on signed/unsigned

[Dual for Maximization]
OMT($\mathcal{BV}$) - Signed/Unsigned $\mathcal{BV}$ [61]

Reduction to unsigned $\mathcal{BV}$ (minimization)

Given an *attractor* $\text{attr}$ for cost, both $\mathcal{BV}$s of $n$ bits, replace cost with

$$\text{cost } \text{xor}_n \text{ attr}$$

Example: maximization of a signed 8-bits Bit-Vector

**Before:**

\[
\begin{array}{cccccccc}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
\vdots \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
127 \\
126 \\
\vdots \\
-1 \\
-2 \\
\end{array}
\]

**After:**

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\vdots \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
127 \\
126 \\
\vdots \\
1 \\
0 \\
\end{array}
\]

**Positive**

**Negative**

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
-127 \\
-128 \\
\end{array}
\]

**Positive**

**Negative**
Goal: find a model $M$ of $\varphi$ for which the value of cost is minimum.

Simplification: $\exists M$ s.t. $M \models \varphi$ and $M(\text{cost}) \neq \text{NAN}$. 
$\implies$ replace $\varphi$ with $\varphi \land \text{cost} \neq \text{NAN}$

FP Minimization Approaches

- Reduction to Bit-Vector Optimization:
  - $\mathcal{BV}$ and $\mathcal{FP}$ are not Nelson-Oppen disjoint!
  - $\implies$ can only use eager $\mathcal{BV}/\mathcal{FP}$ SMT-solving approach

- OMT-based Approach: linear-search, binary-search and adaptive-search

- Ad-Hoc Algorithms:
  - OFP-BS (based on OBV-BS [40])
    - binary search over the bits $[\text{cost}[0], \ldots, \text{cost}[n-1]]$
    - at most $n$ incremental calls to the underlying SMT solver
OMT(\(\mathcal{FP}\)) [61]

Example: Encoding of a \(\mathcal{FP}_{\langle 3,5 \rangle}\)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Minimization in the

- **Positive Domain**, go towards
  \[
  \begin{array}{cccccccc}
    0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  \end{array}
  \quad +0
  \]

- **Negative Domain**, go towards
  \[
  \begin{array}{cccccccc}
    1 & 1 & 1 & 1 & 1 & 1 & 1 & N\text{A}N \\
  \end{array}
  \]

  unless the exponent is all 1s, then go towards
  \[
  \begin{array}{cccccccc}
    ? & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
  \end{array}
  \quad +\infty/\infty
  \]

Dynamic Attractor \(\text{attr}_{\tau_k}\) for cost: given an assignment \(\tau_k\) to the first \(k\) bits of cost, it's the **smallest** \(\mathcal{FP}\)-value different from \(\text{NAN}\) s.t.

\[
\forall i = k - 1, \text{attr}_{\tau_k}[i] = \tau_k[i]
\]

- The ideal result of the optimization wrt. **current** search horizon
Idea: Use $\text{attr}_{\tau_k}$ as look-ahead.

- if $(\mathcal{M}(\text{cost}[k]) \neq \text{attr}_{\tau_k}[k])$ then
  - SMT.INCREMENTAL_CHECK($\varphi \land \tau_k \land \text{cost}[k] = \text{attr}_{\tau_k}[k]$) // try improve cost
    - UNSAT $\Rightarrow$ update $\tau_k$ and $\text{attr}_{\tau_k}$
    - SAT $\Rightarrow$ update $\tau_k$ and $\mathcal{M}$
- otherwise: skip

Disclosure: based on OBV-BS [40].

Example: minimization of a $FP_{\langle 3,5 \rangle}$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\mathcal{M}(\text{cost})$</th>
<th>$\tau_k$</th>
<th>$\text{attr}_{\tau_k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 0 1 1 1 1  31/2</td>
<td></td>
<td>1 1 1 1 0 0 0 0  $-\infty$</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 1 1 1 1  31/2</td>
<td>0</td>
<td>0 0 0 0 0 0 0 0  +0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 1 0  1/32</td>
<td>0</td>
<td>0 0 0 0 0 0 0 0  +0</td>
</tr>
<tr>
<td>3</td>
<td>0 0 0 0 0 0 1 0  1/32</td>
<td>0</td>
<td>0 0 0 0 0 0 0 0  +0</td>
</tr>
<tr>
<td>4</td>
<td>0 0 0 0 0 0 1 0  1/32</td>
<td>0</td>
<td>0 0 0 0 0 0 0 0  +0</td>
</tr>
<tr>
<td>5</td>
<td>0 0 0 0 0 0 1 0  1/32</td>
<td>0 0 0 0 0 0 0 0  +0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 0 1 0  1/32</td>
<td>0 0 0 0 0 0 0 0  +0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 0 1 0  1/32</td>
<td>0 0 0 0 0 0 0 0  +0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 0 0 1 0  1/32</td>
<td>0 0 0 0 0 0 0 0  +0</td>
<td></td>
</tr>
</tbody>
</table>

$\Rightarrow$ UNSAT $\Rightarrow$ SAT $\Rightarrow$ skip $\Rightarrow$ UNSAT $\Rightarrow$ skip $\Rightarrow$ end.
Idea: Use $\text{attr}_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq \text{attr}_{\tau_k}[k])$ **then**
  - SMT.INCREMENTAL_CHECK$(\varphi \land \tau_k \land \text{cost}[k] = \text{attr}_{\tau_k}[k])$ // try improve cost
    - UNSAT $\implies$ update $\tau_k$ and $\text{attr}_{\tau_k}$
    - SAT $\implies$ update $\tau_k$ and $\mathcal{M}$
- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

<table>
<thead>
<tr>
<th>k</th>
<th>$\mathcal{M}(\text{cost})$</th>
<th>$\tau_k$</th>
<th>$\text{attr}_{\tau_k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 0 1 1 1 1 $31/2$</td>
<td>0</td>
<td>1 1 1 1 0 0 0 0 $-\infty$</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 1 1 1 1 $31/2$</td>
<td>0</td>
<td>0 0 0 0 0 0 0 0 $+0$</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 1 0 $1/32$</td>
<td>0 0</td>
<td>0 0 0 0 0 0 0 0 $+0$</td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 0 1 0 $1/32$</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 $+0$</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 0 1 0 $1/32$</td>
<td>0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 0 1 0 $1/32$</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 0 0 1 0 $1/32$</td>
<td>0 0 0 0 0 1 0</td>
<td>0 0 0 0 0 0 0 1 0 $1/32$</td>
</tr>
</tbody>
</table>

"end."
Idea: Use $\text{attr}_{\tau_k}$ as look-ahead.

- if $(\mathcal{M}(\text{cost}[k]) \neq \text{attr}_{\tau_k}[k])$ then
  
  SMT.INCREMENTAL_CHECK$(\varphi \land \tau_k \land \text{cost}[k] = \text{attr}_{\tau_k}[k])$  // try improve cost
  
  - UNSAT $\implies$ update $\tau_k$ and $\text{attr}_{\tau_k}$
  - SAT $\implies$ update $\tau_k$ and $\mathcal{M}$

- otherwise: skip

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$
Idea: Use \( \text{attr}_{\tau_k} \) as look-ahead.

- if \((\mathcal{M}(\text{cost}[k]) \neq \text{attr}_{\tau_k}[k])\) then
  SMT.INCREMENTAL_CHECK(\( \varphi \land \tau_k \land \text{cost}[k] = \text{attr}_{\tau_k}[k] \)) // try improve cost
  - UNSAT \(\implies\) update \(\tau_k\) and \(\text{attr}_{\tau_k}\)
  - SAT \(\implies\) update \(\tau_k\) and \(\mathcal{M}\)

  otherwise: skip

Disclosure: based on OBV-BS [40].

Example: minimization of a \(\mathcal{FP}_{\langle 3, 5 \rangle}\)

<table>
<thead>
<tr>
<th>(k)</th>
<th>(\mathcal{M}(\text{cost}))</th>
<th>(\tau_k)</th>
<th>(\text{attr}_{\tau_k})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 1 0 1 1 1 1 31/2</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>1 1 1 1 0 0 0 0 −(\infty)</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 1 1 1 1 31/2</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 +0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 +0</td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 +0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 +0</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 1/32</td>
<td>0 0 0 0 0 0 0 1 0 +0</td>
</tr>
</tbody>
</table>

\(\implies\) UNSAT \(\implies\) SAT \(\implies\) skip \(\implies\) UNSAT \(\implies\) skip \(\implies\) end.
Idea: Use $\text{attr}_{\tau_k}$ as look-ahead.

- **if** $(\mathcal{M}(\text{cost}[k]) \neq \text{attr}_{\tau_k}[k])$ **then**
  
  SMT.INCREMENTAL_CHECK($\phi \land \tau_k \land \text{cost}[k] = \text{attr}_{\tau_k}[k]$) // try improve cost

  - **UNSAT** $\implies$ update $\tau_k$ and $\text{attr}_{\tau_k}$
  - **SAT** $\implies$ update $\tau_k$ and $\mathcal{M}$

- otherwise: **skip**

Disclosure: based on OBV-BS [40].

Example: minimization of a $\mathcal{FP}_{\langle 3,5 \rangle}$

<table>
<thead>
<tr>
<th>k</th>
<th>$\mathcal{M}(\text{cost})$</th>
<th>$\tau_k$</th>
<th>$\text{attr}_{\tau_k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 0 1 1 1 1</td>
<td>31/2</td>
<td>1 1 1 1 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 1 1 1 1</td>
<td>31/2</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

$\rightarrow$ **UNSAT**

$\rightarrow$ **SAT**

$\rightarrow$ **skip**

$\rightarrow$ **UNSAT**

$\rightarrow$ **skip**

$\rightarrow$ **end.**
**OMT(\(\mathcal{FP}\)) - OFP-BS [61]**

**Idea:** Use \(\text{attr}_{\tau_k}\) as look-ahead.

- if \((\mathcal{M}(\text{cost}[k]) \neq \text{attr}_{\tau_k}[k])\) then
  - SMT.INCREMENTAL_CHECK(\(\varphi \land \tau_k \land \text{cost}[k] = \text{attr}_{\tau_k}[k]\)) // try improve cost
    - UNSAT \(\rightarrow\) update \(\tau_k\) and \(\text{attr}_{\tau_k}\)
    - SAT \(\rightarrow\) update \(\tau_k\) and \(\mathcal{M}\)
- otherwise: skip

**Disclosure:** based on OBV-BS [40].

**Example:** minimization of a \(\mathcal{FP}_{\langle 3,5 \rangle}\)

<table>
<thead>
<tr>
<th>k</th>
<th>(\mathcal{M}(\text{cost}))</th>
<th>(\tau_k)</th>
<th>(\text{attr}_{\tau_k})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 0 1 1 1 1</td>
<td>0 0 0 0 0 0 0 0</td>
<td>1 1 1 1 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 1 1 1 1</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 0 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 0 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 0 0 1 0</td>
<td>0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

\(\rightarrow\) UNSAT, SAT, skip, end. 

Disclosures and data visualizations are also included.
Idea: Use \( \text{attr}_{\tau_k} \) as look-ahead.

- if \((\mathcal{M}(\text{cost}[k]) \neq \text{attr}_{\tau_k}[k])\) then
  SMT.INCREMENTAL_CHECK(\(\varphi \land \tau_k \land \text{cost}[k] = \text{attr}_{\tau_k}[k]\)) // try improve cost
  - UNSAT \(\implies\) update \(\tau_k\) and \(\text{attr}_{\tau_k}\)
  - SAT \(\implies\) update \(\tau_k\) and \(\mathcal{M}\)
- otherwise: skip

Disclosure: based on OBV-BS [40].

Example: minimization of a \(\mathcal{FP}_{\langle 3,5 \rangle}\)

<table>
<thead>
<tr>
<th>(k)</th>
<th>(\mathcal{M}(\text{cost}))</th>
<th>(\tau_k)</th>
<th>(\text{attr}_{\tau_k})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0 1 1 0 1 1 1 1</td>
<td>31/2</td>
<td>1 1 1 1 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 1 1 0 1 1 1 1</td>
<td>31/2</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>6</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>8</td>
<td>0 0 0 0 0 0 1 0</td>
<td>1/32</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
Outline

1 Motivations
2 Optimization Modulo Theories with Linear-Arithmetic Objectives
3 OMT with Multiple and Combined Objectives
4 Relevant Subcases: OMT+PB & MaxSMT
5 Status of OMT
6 Current and Future Research Directions
7 Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Running Example: performance bottleneck

Problem:

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

Example: with $k = 2$, $w = 1$ and $n = 4$
Running Example: performance bottleneck

Problem:

- \( \neg(k \leq \text{cost}) \) 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 \( n = 4 \)

Learned Clauses

\[ \mu \models \varphi \]
Running Example: performance bottleneck

Problem:
- \( \neg (k \leq \text{cost}) \) 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 Constraint Propagation

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

Problem:

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

Example: with \( k = 2 \), \( w = 1 \) and \( n = 4 \)
Solution: OMT + sorting networks [56]

Contribution:
Enriched OMT encoding with bidirectional **sorting networks** [58, 10].

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

- $k A_i$'s are $\top$ $\iff$ $\{B_0, \ldots, B_{k-1}\}$ are $\top$,
- $m - k A_i$'s are $\ast$ $\iff$ $\{B_k, \ldots, B_{m-1}\}$ are $\ast$,
- $n - m A_i$'s are $\bot$ $\iff$ $\{B_m, \ldots, B_{n-1}\}$ are $\bot$

then we encode it as $\langle \varphi', \text{cost} \rangle$, where

$$
\varphi' := \varphi \land C(A_0, \ldots, A_{n-1}, B_0, \ldots, B_{n-1}) \land \bigwedge_{i=0}^{n-1} B_i \leftrightarrow ((i + 1) \cdot w \leq \text{cost}) \land \bigwedge_{i=0}^{n-2} B_{i+1} \rightarrow B_i
$$
Properties: OMT + sorting networks [56]

Properties:
- if \((k \cdot w \leq \text{cost}) = \bot\), then by BCP \(\forall i \in [k, n].B_{i-1} = \bot\)

Example: with \(k = 2\), \(w = 1\) and \(n = 4\)
Properties: OMT + sorting networks [56]

Properties:

• if \((k \cdot w \leq \text{cost}) = \bot\), 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 \((k \cdot w \leq \text{cost}) = \top\).

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

- **Optimization Step:** learn \( \neg(k \cdot w \leq \text{cost}) \) and restart/jump to level 0

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

Learned Clauses

\( \neg(2 \leq \text{obj}) \)

\( \mu \models \varphi \)
Example: OMT with sorting networks

- **Optimization Step:** learn $\neg (k \cdot w \leq \text{cost})$ and restart/jump to level 0
- as soon as $k - 1 \ A_i$ are assigned $\top$
  $\implies$ all others are unit-propagated to $\bot$

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

Learned Clauses

$\mu \models \varphi$

$\mu' \models \varphi$
Solution: Combine OMT with Sorting Networks

**OPTIMATHSAT**: sorting networks implemented

- **Bi-directional Sequential Counter** [58], in $O(n^2)$ but incremental sum of $A_i$'s, unary representation
- **Bi-directional Cardinality Network** [10, 6], in $O(n \log^2 n)$ based on merge-sort algorithm

**Generalization**

The same performance issue occurs for $\langle \varphi, \text{cost} \rangle$, where

$$\text{cost} = \tau_1 + \ldots + \tau_m,$$

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

**Solution:**

- use a separate sorting circuit for each term $\tau_j$
- add clauses in the form $(w_j \cdot i \leq \tau_j) \rightarrow (w_j \cdot i \leq \text{cost})$
Outline

1. Motivations
2. Optimization Modulo Theories with Linear-Arithmetic Objectives
3. OMT with Multiple and Combined Objectives
4. Relevant Subcases: OMT+PB & MaxSMT
5. Status of OMT
6. Current and Future Research Directions
7. Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Idea: given a MaxSMT $\langle \varphi_h, \varphi_s \rangle$, treat both $\varphi_h$ and $\varphi_s$ as hard clauses. 

Analyze conflict $\tau$, where $\tau \stackrel{\text{def}}{=} \tau_h \cup \tau_s$, $\tau_h \subseteq \varphi_h$ and $\tau_s \subseteq \varphi_s$

- if $\tau_s = \emptyset \implies$ input problem is unsatisfiable
- else let $w_{\text{min}} \stackrel{\text{def}}{=} \min(w_i \mid \langle C_i, w_i \rangle \in \tau_s)$ and relax the problem:
  - Learn conflict-clause and replace soft-clauses
    
    $$
    \varphi_h := \varphi_h \cup \bigvee_{\langle C_i, w_i \rangle \in \tau_s} \neg C_i
    $$
    
    $$
    \varphi_s := \varphi_s \setminus \tau_s \cup \bigcup_{\langle C_i, w_i \rangle \in \tau_s} \langle C_i, w_i - w_{\text{min}} \rangle \text{ if } w_i - w_{\text{min}} > 0
    $$

  - if $|\tau_s| > 1 \implies$ add compensation clauses
    
    $$
    \varphi_h := \varphi_h \cup \bigcup_{\langle C_i, w_i \rangle \in \tau_s} \cdot B_i \rightarrow (B_{i-1} \land C_i)
    $$
    
    // $B_0 := \top$, $\forall_{i>0}.B_i$ is fresh Boolean var
    
    $$
    \varphi_s := \varphi_s \cup \bigcup_{\langle C_i, w_i \rangle \in \{\tau_s \setminus \langle C_1, w_1 \rangle\}} \cdot \langle B_{i-1} \lor C_i, w_{\text{min}} \rangle
    $$

No Conflict: optimal solution
Outline

1 Motivations
2 Optimization Modulo Theories with Linear-Arithmetic Objectives
3 OMT with Multiple and Combined Objectives
4 Relevant Subcases: OMT+PB & MaxSMT
5 Status of OMT
6 Current and Future Research Directions
7 Appendix
   - Inline OMT schema
   - OMT for Bit-vector and Floating-point theories
   - Improving OMT+PB by sorting networks
   - The MaxRES MaxSMT Procedure
   - Extended SMT-LIB language
   - Pareto Optimization (hints)
Extended SMT-LIBv2 Interface [57]

(minimize <term> [:id <string>] [:signed]
    [:lower <const_term>] [:upper <const_term>])
(maximize <term> [:id <string>] [:signed]
    [:lower <const_term>] [:upper <const_term>])

(minmax <term> ... <term> [:id <string>] [:signed]
    [:lower <const_term>] [:upper <const_term>])
(maxmin <term> ... <term> [:id <string>] [:signed]
    [:lower <const_term>] [:upper <const_term>])

(assert-soft <term> [:id <string>] [:weight <const_term>])
(check-sat)
(check-allsat (<const_term> ... <const_term>))

(get-objectives)
(load-objective-model <numeral>)
Outline

1 Motivations
2 Optimization Modulo Theories with Linear-Arithmetic Objectives
3 OMT with Multiple and Combined Objectives
4 Relevant Subcases: OMT+PB & MaxSMT
5 Status of OMT
6 Current and Future Research Directions
7 Appendix
   • Inline OMT schema
   • OMT for Bit-vector and Floating-point theories
   • Improving OMT+PB by sorting networks
   • The MaxRES MaxSMT Procedure
   • Extended SMT-LIB language
   • Pareto Optimization (hints)
Pareto OMT

**Definitions:**

- A model $\mathcal{M}$ **Pareto-dominates** $\mathcal{M}'$ iff
  \[ \forall i. \mathcal{M}(\text{cost}_i) \leq \mathcal{M}'(\text{cost}_i) \]
  and
  \[ \exists j. \mathcal{M}(\text{cost}_j) < \mathcal{M}'(\text{cost}_j) \]
  *(dual for maximization)*

- $\mathcal{M}$ is **Pareto-optimal** iff it is not Pareto-dominated by any $\mathcal{M}'$.

**Example:** $\langle \varphi, \{\text{cost}_1, \text{cost}_2\} \rangle_P$

**Goal:** given a pair $\langle \varphi, \mathcal{O} \rangle_P$, where $\mathcal{O} \overset{\text{def}}{=} \{\text{cost}_1, \ldots, \text{cost}_N\}$

- find the set of Pareto-optimal models $\{\mathcal{M}_1, \ldots, \mathcal{M}_M\}$ (i.e. the **Pareto front**).
Pareto OMT: Guided Improvement Algorithm (GIA)

Guided Improvement Algorithm [49, 16]

Given a pair $\langle \varphi, \mathcal{O} \rangle_P$, where $\mathcal{O} \overset{\text{def}}{=} \{ \text{cost}_1, \ldots, \text{cost}_N \}$:

- start from random model $\mathcal{M}$ of $\varphi$
- **loop**: look for a model $\mathcal{M}'$ of $\varphi$ that Pareto-dominates $\mathcal{M}$
  $\implies$ if any, replace $\mathcal{M}$ with $\mathcal{M}'$ and keep looking
- block solutions Pareto-dominated by $\mathcal{M}$
- **repeat**

Infinite Loop:

- some cost $i$ is unbounded
- some cost $j$ can always be improved by an *infinitesimal value* (e.g. OMT($\mathcal{LRA}$))

Also: $\mathcal{T}$-minimization procedure not used
$\implies$ the same $\mu$ may be visited multiple times by CDCL/SAT engine
Observation. If model $\mathcal{M}$ is Lexicographic-optimal for $\langle \varphi, \{\text{cost}_1, \ldots, \text{cost}_N\} \rangle_L$, then $\mathcal{M}$ is also Pareto-optimal for $\langle \varphi, \{\text{cost}_1, \ldots, \text{cost}_N\} \rangle_P$.

Idea:
- Shuffle $\{\text{cost}_1, \ldots, \text{cost}_N\}$  
  $\implies$ explore from different directions
- Extract Lexicographic-optimal $\mathcal{M}$
- Learn
  \[
  \bigvee_{i=1}^{N} (\text{cost}_i < \mathcal{M}[\text{cost}_i])
  \]
  to block Pareto-dominated solutions
- repeat

Example: $\langle \varphi, \{\text{cost}_1, \text{cost}_2\} \rangle_P$
Observation. If model $\mathcal{M}$ is Lexicographic-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\} \rangle_\mathcal{L}$, then $\mathcal{M}$ is also Pareto-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\} \rangle_\mathcal{P}$.

Idea:

- Shuffle $\{\text{cost}_1, ..., \text{cost}_N\}$
  $\implies$ explore from different directions
- Extract Lexicographic-optimal $\mathcal{M}$
- Learn
  $$\bigvee_{i=1}^{N} (\text{cost}_i < \mathcal{M}[\text{cost}_i])$$
  to block Pareto-dominated solutions
- repeat

Example: $\langle \varphi, \{\text{cost}_1, \text{cost}_2\} \rangle_\mathcal{P}$

$$\varphi' := \varphi \land ((\text{cost}_1 < -6) \lor (\text{cost}_2 < -1))$$
Observation. If model $\mathcal{M}$ is Lexicographic-optimal for $\langle \varphi, \{cost_1, \ldots, cost_N\}\rangle_L$, then $\mathcal{M}$ is also Pareto-optimal for $\langle \varphi, \{cost_1, \ldots, cost_N\}\rangle_P$.

Idea:
- Shuffle $\{cost_1, \ldots, cost_N\}$
  $\implies$ explore from different directions
- Extract Lexicographic-optimal $\mathcal{M}$
- Learn
  $$\bigvee_{i=1}^{N} (cost_i < \mathcal{M}[cost_i])$$
  to block Pareto-dominated solutions
- repeat

Example: $\langle \varphi, \{cost_1, cost_2\}\rangle_P$

$\varphi' := \varphi \land ((cost_1 < -6) \lor (cost_2 < -1))$
**Pareto OMT: Lexicographic GIA**

**Observation.** If model $M$ is Lexicographic-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\}\rangle_L$, then $M$ is also Pareto-optimal for $\langle \varphi, \{\text{cost}_1, ..., \text{cost}_N\}\rangle_P$.

**Idea:**
- Shuffle $\{\text{cost}_1, ..., \text{cost}_N\}$
  $\implies$ explore from different directions
- Extract Lexicographic-optimal $M$
- Learn

\[
\bigvee_{i=1}^{\text{cost}_i < M[\text{cost}_i]}
\]

- to block Pareto-dominated solutions
- repeat

**Example:** $\langle \varphi, \{\text{cost}_1, \text{cost}_2\}\rangle_P$

**Problem:** how to deal with unbounded objectives?
Pareto OMT: dealing with unbounded objectives

1. Sort objectives:
   - lower-bounded first
   - lower-unbounded last

before Lex. OMT.
Pareto OMT: dealing with unbounded objectives

1. Sort objectives:
   - lower-bounded first
   - lower-unbounded last

before Lex. OMT.
Pareto OMT: dealing with unbounded objectives

1. Sort objectives:
   - lower-bounded first
   - lower-unbounded last

before Lex. OMT.
Pareto OMT: dealing with unbounded objectives

1. Sort objectives:
   - lower-bounded first
   - lower-unbounded last

   before Lex. OMT.

2. If Lex. OMT unbounded, (temporarily) learn:

   \[ \bigwedge_{i=1}^{N} (\text{cost}_i \leq M[\text{cost}_i]) \]

   and try again.
1. Sort objectives:
   - lower-bounded first
   - lower-unbounded last

before Lex. OMT.

2. If Lex. OMT unbounded, (temporarily) learn:
   \[ \bigwedge_{i=1}^{i=N} (\text{cost}_i \leq M[\text{cost}_i]) \]

and try again.
Pareto OMT: dealing with unbounded objectives

1. Sort objectives:
   - lower-bounded first
   - lower-unbounded last
   before Lex. OMT.

2. If Lex. OMT unbounded, (temporarily) learn:
   \[
   \bigwedge_{i=1}^{N} (\text{cost}_i \leq \mathcal{M}[\text{cost}_i])
   \]
   and try again.

3. If Lex. OMT still unbounded, give up.