MDD Propagation for Disjunctive Scheduling

Andre Augusto Cire

Joint work with Willem-Jan van Hoeve

Tepper School of Business, Carnegie Mellon University

ISMP 2012
Motivation

- **Constraint-based scheduling**: *Exploit subproblem structure*
  - High-level, structured constraints (*disjunctive, cumulative…*)
  - Sophisticated inference techniques
  - Process constraints one at a time

- ... but how to pool the results of constraint processing?
  - **Constraint store** - Shared data structure that accumulates *implications* of each constraint
Motivation

- In practice: constraint store is the **domain store**
  - Implications are of the form
    \[ x_i \leq v, \ x_i \geq v, \text{ or } x_i \neq v \ \text{ for } v \in \text{domain}(x_i) \]
  - Propagation: reduce domains as much as possible

- Domain store is a natural relaxation, but may be too **weak**
Motivation

\[
\text{alldiff}(x_1, x_2, x_3),
\]
\[
x_1 + x_2 + x_3 \leq 12,
\]
\[
x_1, x_2, x_3 \in \{1, 9, 10\}.
\]

Problem is infeasible

1. Propagation of \textit{alldiff}
   - No inference.

2. Propagation of \textit{sum}
   - No inference.

Domains remain unchanged!

- Common solution: new global constraint
  - \textit{cost-alldiff}, \textit{cost-sum-weighted-alldiff}, etc …
Motivation

- Other alternative: a **richer** constraint store

- **Proposal:** *Relaxed Multivalued Decision Diagrams* (MDDs)
  - Initial framework by Andersen et al (CP2007).

- **Fundamental questions**
  - How to effectively process MDDs for particular constraints?
  - When does it perform better than domain store?
  - ...

- **Our goal:** application to constraint-based scheduling
Relaxed MDDs

\[
\text{alldiff}(x_1, x_2, x_3), \\
x_1 + x_2 + x_3 \leq 12, \\
x_1, x_2, x_3 \in \{1,9,10\}.
\]

- Compact representation of a search tree
- Ordered Acyclic Digraph
  - *Layers*: variables
  - *Arc labels*: variable assignments
- Paths from *r* to *t*: solutions to the problem
Relaxed MDDs

\[
\text{alldiff}(x_1, x_2, x_3), \quad x_1 + x_2 + x_3 \leq 12, \quad x_1, x_2, x_3 \in \{1,9,10\}.
\]

- Compact representation of a search tree
- Ordered Acyclic Digraph
  - \textit{Layers}: variables
  - \textit{Arc labels}: variable assignments
- Paths from \(r\) to \(t\): solutions to the problem
  - Example: \(x_1=1, x_2=9, x_3=10\)
Relaxed MDDs

```
\text{alldiff}(x_1, x_2, x_3),
\ x_1 + x_2 + x_3 \leq 12,
\ x_1, x_2, x_3 \in \{1,9,10\}.
```

- **Relaxed**
  - It encodes \textit{all} feasible solutions
  - It may encode \textit{infeasible} solutions
Relaxed MDDs

- Relaxation is **adjustable**
  - Controlled by the **width** of the graph

```
{1,9,10}
{1,9,10}
{1,9,10}

{1,9,10}
{9,10}
{9,10}

{1,9,10}
{1,9,10}

{1}
{9,10}
{9,10}

{1}
{9}
{10}

---

Width 1 = Domain Store  
Width 2  
Unlimited width: original problem
```
Relaxed MDDs

- Constraint processing
  - Refine the MDD representation by removing / adding arcs

\[
\text{alldiff}(x_1, x_2, x_3),
\]
\[
x_1 + x_2 + x_3 \leq 12,
\]
\[
x_1, x_2, x_3 \in \{1,9,10\}.
\]

1. Propagation of \textit{alldiff}
Relaxed MDDs

- Constraint processing
  - Refine the MDD representation by removing / adding arcs

\[ \text{alldiff}(x_1, x_2, x_3), \]
\[ x_1 + x_2 + x_3 \leq 12, \]
\[ x_1, x_2, x_3 \in \{1, 9, 10\}. \]

1. Propagation of \text{alldiff}
Relaxed MDDs

- Constraint processing
  - Refine the MDD representation by removing / adding arcs

\[ \text{alldiff}(x_1, x_2, x_3), \]
\[ x_1 + x_2 + x_3 \leq 12, \]
\[ x_1, x_2, x_3 \in \{1,9,10\}. \]

1. Propagation of \textit{alldiff}

2. Propagation of \textit{sum}
   - Detects infeasibility!
Relaxed MDDs and Scheduling

- **Focus:** disjunctive scheduling
  - Highlight of CP, widespread application
  - Still has particular deficiencies

- MDD constraint processing for disjunctive scheduling
Disjunctive Scheduling

- Sequencing and scheduling of activities on a resource

- **Activities**
  - Processing time: \( p_i \)
  - Release time: \( r_i \)
  - Deadline: \( d_i \)

- **Resource**
  - Nonpreemptive
  - Process one activity at a time
Common Side Constraints

- Precedence relations between activities
- Sequence-dependent setup times
- Induced by objective function
  - Makespan
  - Sum of setup times
  - Sum of completion times
  - Tardiness / number of late jobs
  - ...

15
Inference

- Inference for disjunctive scheduling
  - Precedence relations
  - Time intervals that an activity can be processed

- Sophisticated techniques include:
  - Edge-Finding
  - Not-first / not-last rules

- Challenges arise in presence of
  - Sequence-dependent setup times
  - Complex objective functions
MDDs for Disjunctive Scheduling

Our two three main considerations:

- **Representation**
  - How to represent solutions of disjunctive scheduling in an MDD?

- **Construction**
  - How to construct this relaxed MDD?

- **Inference techniques**
  - What can we infer using the relaxed MDD?
MDD Representation

- Natural representation as MDDs
- Every solution can be written as a permutation $\pi$

\[ \pi_1, \pi_2, \pi_3, \ldots, \pi_n: \text{activity sequencing in the resource} \]

- Schedule is *implied* by a sequence, e.g.:

\[ start_{\pi_i} \geq start_{\pi_{i-1}} + p_{\pi_{i-1}} \quad i = 2, \ldots, n \]
MDD Representation

\[ \pi_1 : \text{first activity} \]
\[ \pi_2 : \text{second activity} \]
\[ \pi_n : \text{n-th activity} \]
MDD Representation

Path \{1\} – \{3\} – \{2\}

0 \leq \text{start}_1 \leq 1
6 \leq \text{start}_2 \leq 7
3 \leq \text{start}_3 \leq 5
MDD Construction

- In general, MDDs can grow exponentially
  - Polynomial-width for particular scheduling problems

- We fix a maximum width $W$

- Apply a variation of filter and refinement technique
  - Andersen et al. (CP2007), Hoda et al. (CP2010)
Filter and Refinement

- **Start with a width-1 MDD**
  - Straightforward MDD relaxation

- **Filter infeasible arc values**
  - Top-down/Bottom-up passes

![Diagram]

π₁

π₂

π₃
Filter and Refinement

- **Start with a width-1 MDD**
  - Straightforward MDD relaxation

- **Filter infeasible arc values**
  - Top-down/Bottom-up passes

- **Refinement**
  - Add nodes to improve relaxation
Filter and Refinement

- **Start with a width-1 MDD**
  - Straightforward MDD relaxation

- **Filter** infeasible arc values
  - Top-down/Bottom-up passes

- **Refinement**
  - Add nodes to improve relaxation

- Repeat filtering/refinement until certain conditions are met
Filter: Top-Down Example

- Filter based on a state information at each node

- Example:
  Filtering arcs \((u,v)\)
Filter: Top-Down Example

- **Earliest Completion Time:** $E_u$
  - Minimum completion time of all partial sequences represented by paths from root to node $u$

- **Similarly:** Latest Completion Time
Filter: Top-Down Example

<table>
<thead>
<tr>
<th>Act</th>
<th>r_i</th>
<th>d_i</th>
<th>p_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- $E_u = 7$
- *Eliminate 4 from* $(u,v)$
Filter and Refinement

- Other filters
  - Node edge-finding, not-first/not-last rules
  - Precedence filtering
  - Additional alldiff filters
  - ...

- Refinement
  - Based on earliest completion time of a node
MDD Inference

**Theorem:** Given the exact MDD $M$, we can deduce all implied activity precedences in polynomial time in the size of $M$.

- For a node $v$,
  - $A^\downarrow_v$: values in all paths from root to $v$
  - $A^\uparrow_v$: values in all paths from node $v$ to terminal

- Precedence relation $i \ll j$ holds if and only if $(j \notin A^\downarrow_u)$ or $(i \notin A^\uparrow_u)$ for all nodes $u$ in $M$

- Same technique applies to relaxed MDD
Communicate Precedence Relations

1. Provide precedences inferred from the MDD to solver
   - Update time variables
   - Other inference techniques may utilize them

2. We can filter the relaxed MDD using precedence relations inferred from other (CP) techniques
   - Precedences deduced by this method might not be dominated by other techniques, even for small widths.
Experimental Results

- Implemented in *Ilog CP Optimizer (CPO)*
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation

- Random and structured instances
  - Different classical objective functions

- Tested integration CPO+MDD
Makespan

Random instance
15 jobs
Lex search
MDD width: 16
Makespan

Random instance
15 jobs
Lex search
MDD width: 16
Sum of Completion Times

Random instances
12 jobs
Lex search
MDD width: 16
Sum of Completion Times

Random instances
12 jobs
Lex search
MDD width: 16
TSP with Time Windows

Dumas instances
Ascheuer instances
20-60 jobs
Lex search
MDD width: 16
TSP with Time Windows

Dumas instances  
Ascheuer instances  
20-60 jobs  
Lex search  
MDD width: 16
## Instances Dumas (TSPTW)

<table>
<thead>
<tr>
<th>Instance</th>
<th>Cities</th>
<th>Backtracks</th>
<th>Time (s)</th>
<th>Backtracks</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>n40w40.004</td>
<td>40</td>
<td>480,970</td>
<td>50.81</td>
<td>18</td>
<td>0.06</td>
</tr>
<tr>
<td>n60w20.001</td>
<td>60</td>
<td>908,606</td>
<td>199.26</td>
<td>50</td>
<td>0.22</td>
</tr>
<tr>
<td>n60w20.002</td>
<td>60</td>
<td>84,074</td>
<td>14.13</td>
<td>46</td>
<td>0.16</td>
</tr>
<tr>
<td>n60w20.003</td>
<td>60</td>
<td>&gt; 22,296,012</td>
<td>&gt; 3600</td>
<td>99</td>
<td>0.32</td>
</tr>
<tr>
<td>n60w20.004</td>
<td>60</td>
<td>2,685,255</td>
<td>408.34</td>
<td>97</td>
<td>0.24</td>
</tr>
</tbody>
</table>

- **CPO**: minimize sum of setup times
- **CPO+MDD**: MDDs have maximum width 16
Conclusions

- MDD for disjunctive constraints
  - Strong relation to precedence graph
  - High-level communication between MDD and other inference mechanisms

- Practical perspective
  - Current experiments suggest it is stronger for sums
    - Observed orders of magnitude improvement
Thank you!
**Compilation**

**Theorem:** Constructing the exact MDD for a Disjunctive Instance is an NP-Hard problem

Nevertheless, some interesting restrictions, e.g. (Balas [99]):

- TSP defined on a complete graph
- Given a fixed parameter $k$, we must satisfy

\[ i \ll j \quad \text{if} \quad j - i \geq k \quad \text{for cities } i, j \]

**Corollary:** The exact MDD for the TSP above has $O(n2^k)$ nodes
Filter: Top-Down Example

- **All-paths state:** $A_u$
  - Labels belonging to all paths from node $r$ to node $u$
  - $A_u = \{3\}$
  - Thus eliminate $\{3\}$ from $(u,v)$

- Introduced for Alldifferent constraint in [Andersen et al 2007]
Outline

1. Disjunctive Scheduling
2. Multivalued Decision Diagram (MDD) Representation
3. Filtering and Precedence Relations
4. Experimental Results
5. Conclusion