Decision Diagrams for Sequencing and Scheduling

Willem-Jan van Hoeve

Tepper School of Business
Carnegie Mellon University

www.andrew.cmu.edu/user/vanhoeve/mdd/
What can MDDs do for Combinatorial Optimization?

- *Compact representation* of all solutions to a problem
- Limit on size gives *approximation*
- Control strength of approximation by size limit

MDDs for Discrete Optimization

- 9:00am-10:30am tutorial (John Hooker)
- MDD as discrete relaxation for lower and upper bound
- Exact branch-and-bound search scheme (on MDD states)

MDDs for Sequencing and Scheduling

- MDD-based constraint propagation
- Constraint-based scheduling with MDDs
- State-dependent costs
• Binary Decision Diagrams were introduced to compactly represent Boolean functions \cite{Lee1959, Akers1978, Bryant1986}.
• BDD: merge isomorphic subtrees of a given binary decision tree.
• MDDs are multi-valued decision diagrams (i.e., for arbitrary finite-domain variables).
**Brief background**

- Original application areas: circuit design, verification
- Usually *reduced ordered* BDDs/MDDs are applied
  - fixed variable ordering
  - minimal exact representation
- Application to discrete optimization (exponential-size)
  - cut generation [Becker et al., 2005]
  - 0/1 vertex and facet enumeration [Behle & Eisenbrand, 2007]
  - post-optimality analysis [Hadzic & Hooker, 2006, 2007]
  - set bounds propagation [Hawkins, Lagoon, Stuckey, 2005]
- Scalable variant (polynomial-size)
  - relaxed MDDs
    [Andersen, Hadzic, Hooker & Tiedemann, CP 2007]
Exact MDDs for discrete optimization

\[
\begin{align*}
(1) \quad & x_1 + x_2 + x_3 \geq 1 \\
(2) \quad & x_1 + x_4 + x_5 \geq 1 \\
(3) \quad & x_2 + x_4 \geq 1
\end{align*}
\]
Exact MDDs for discrete optimization

(1) \(x_1 + x_2 + x_3 \geq 1\)
(2) \(x_1 + x_4 + x_5 \geq 1\)
(3) \(x_2 + x_4 \geq 1\)
Exact MDDs for discrete optimization

(1) \( x_1 + x_2 + x_3 \geq 1 \)
(2) \( x_1 + x_4 + x_5 \geq 1 \)
(3) \( x_2 + x_4 \geq 1 \)
Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$

$\begin{align*}
\text{---: 0} \\
\text{---: 1} \\
\end{align*}$
Exact MDDs for discrete optimization

\(1\) \(x_1 + x_2 + x_3 \geq 1\)

\(2\) \(x_1 + x_4 + x_5 \geq 1\)

\(3\) \(x_2 + x_4 \geq 1\)

Each path corresponds to a solution 

\((1,0,1,1,0)\)
Limited-size MDDs

- Exact MDDs can be of exponential size in general
- We can limit the size of the MDD and still have a meaningful representation:
  - First proposed by Andersen et al. [2007] for improved constraint propagation:
    Limit the width of the MDD (the maximum number of nodes on any layer)
MDDs for Constraint Programming
Motivation

Constraint Programming applies
• systematic search and
• inference techniques
to solve combinatorial problems

Inference mainly takes place through:
• **Filtering** provably inconsistent values from variable domains
• **Propagating** the updated domains to other constraints

\[
x_1 > x_2
\]
\[
x_1 + x_2 = x_3
\]
\[
\text{alldifferent}(x_1, x_2, x_3, x_4)
\]

\[
x_1 \in \{1,2\}, \quad x_2 \in \{0,1,2,3\}, \quad x_3 \in \{2,3\}, \quad x_4 \in \{0,1\}
\]

domain propagation can be weak, however...
**Illustrative example**

(alldifferent) \((x_1,x_2,x_3,x_4)\) \hspace{1cm} (1)

\(x_1 + x_2 + x_3 \geq 9\) \hspace{1cm} (2)

\(x_i \in \{1,2,3,4\}\)

(1) and (2) both domain consistent (no propagation)

List of all solutions to **alldifferent**:

\[
\begin{array}{cccc}
 x_1 & x_2 & x_3 & x_4 \\
 1 & 2 & 3 & 4 \\
 1 & 2 & 4 & 3 \\
 1 & 3 & 2 & 4 \\
 \vdots \\
 4 & 3 & 2 & 1
\end{array}
\]

Suppose we could evaluate (2) on this list

projection: \(D(x_i) = \{1,2,3,4\}\)
Illustrative example

\[ \text{alldifferent}(x_1, x_2, x_3, x_4) \quad (1) \]
\[ x_1 + x_2 + x_3 \geq 9 \quad (2) \]
\[ x_i \in \{1,2,3,4\} \]

List of all solutions to \textit{alldifferent}:

\[
\begin{array}{cccc}
  x_1 & x_2 & x_3 & x_4 \\
  \checkmark & 2 & 3 & 4 & 1 \\
  \checkmark & 2 & 4 & 3 & 1 \\
  \checkmark & 3 & 2 & 4 & 1 \\
  \checkmark & 4 & 3 & 2 & 1 \\
  \ldots \\
\end{array}
\]

Suppose we could evaluate (2) on this list

\[ D(x_1) = D(x_2) = D(x_3) = \{2,3,4\} \]

projection: \[ D(x_4) = \{1\} \]
Illustrative example (cont’d)

\[ \text{alldifferent}(x_1, x_2, x_3, x_4) \quad (1) \]
\[ x_1 + x_2 + x_3 \geq 9 \quad (2) \]
\[ x_i \in \{1, 2, 3, 4\} \]

List of all solutions: use MDDs

<table>
<thead>
<tr>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Motivation for MDD propagation

• Conventional domain propagation projects all structural relationships among variables onto the domains

• Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation)

We can communicate more information between constraint using MDDs [Andersen et al. 2007]

• Explicit representation of more refined potential solution space

• Limited width defines relaxed MDD

• Strength is controlled by the imposed width
MDD-based Constraint Programming

• Maintain limited-width MDD
  – Serves as relaxation
  – Typically start with width 1 (initial variable domains)
  – Dynamically adjust MDD, based on constraints

• Constraint Propagation
  – Edge filtering: Remove provably inconsistent edges (those that do not participate in any solution)
  – Node refinement: Split nodes to separate edge information

• Search
  – As in classical CP, but may now be guided by MDD
Specific MDD propagation algorithms

- Linear equalities and inequalities [Hadzic et al., 2008] [Hoda et al., 2010]
- *Alldifferent* constraints [Andersen et al., 2007]
- *Element* constraints [Hoda et al., 2010]
- *Among* constraints [Hoda et al., 2010]
- Disjunctive scheduling constraints [Hoda et al., 2010] [Cire & v.H., 2011, 2013]
- *Sequence* constraints (combination of *Amongs*) [Bergman et al., 2014]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]
Example: Among Constraints

- Given a set of variables $X$, and a set of values $S$, a lower bound $l$ and upper bound $u$,

$$\text{Among}(X, S, l, u) := l \leq \sum_{x \in X} (x \in S) \leq u$$

“among the variables in $X$, at least $l$ and at most $u$ take a value from the set $S$”

- Applications in, e.g., nurse scheduling
  - must work between 1 and 2 night shifts each 10 days
Propagating Among Constraints

width 1 vs 16

(Systems of overlapping Among constraints)
Example: Sequence Constraints

Employee must work at most 7 days every 9 consecutive days

<table>
<thead>
<tr>
<th></th>
<th>sun</th>
<th>mon</th>
<th>tue</th>
<th>wed</th>
<th>thu</th>
<th>fri</th>
<th>sat</th>
<th>sun</th>
<th>mon</th>
<th>tue</th>
<th>wed</th>
<th>thu</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1</td>
<td>x2</td>
<td>x3</td>
<td>x4</td>
<td>x5</td>
<td>x6</td>
<td>x7</td>
<td>x8</td>
<td>x9</td>
<td>x10</td>
<td>x11</td>
<td>x12</td>
<td></td>
</tr>
</tbody>
</table>

\[
0 \leq x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 \leq 7
\]
\[
0 \leq x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} \leq 7
\]
\[
0 \leq x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} \leq 7
\]
\[
0 \leq x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{12} \leq 7
\]

\[
=: \text{Sequence}(\{x_1, x_2, ..., x_{12}\}, q=9, S=\{1\}, l=0, u=7)
\]

\[
\text{Sequence}(X, q, S, l, u) := \bigwedge_{|X'|=q} l \leq \sum_{x \in X'} (x \in S) \leq u
\]

\[
\Rightarrow \text{Among}(X, S, l, u)
\]
Performance Comparison for Sequence
A large MDD by itself may not be sufficient!
• MDDs can handle objective functions as well
• Important for many CP problems
  – e.g., disjunctive scheduling
  – minimize makespan, weighted completion times, etc.
• We will develop an MDD approach to disjunctive scheduling
  – combines MDD propagation and optimization reasoning
Handling objective functions

(1) \( x_1 + x_2 + x_3 \geq 1 \)
(2) \( x_1 + x_4 + x_5 \geq 1 \)
(3) \( x_2 + x_4 \geq 1 \)

Suppose we have an objective:

\[
\begin{align*}
\text{min} & \quad 4x_1 + 3x_2 + x_3 + 2x_4 + 5x_5 \\
\text{s.t.} & \quad 0 \leq x_i \leq 1
\end{align*}
\]

shortest path computation
MDDs for Disjunctive Scheduling

Disjunctive Scheduling
Disjunctive Scheduling in CP

• Sequencing and scheduling of activities on a resource

• Activities
  – Processing time: $p_i$
  – Release time: $r_i$
  – Deadline: $d_i$
  – Start time variable: $s_i$

• Resource
  – Nonpreemptive
  – Process one activity at a time
Extensions

• Precedence relations between activities
• Sequence-dependent setup times
• Various objective functions
  – Makespan
  – Sum of setup times
  – (Weighted) sum of completion times
  – (Weighted) tardiness
  – number of late jobs
  – ...
Inference

• Inference for disjunctive scheduling
  – Precedence relations
  – Time intervals in which an activity can be processed

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

• Examples: $1 \ll 3$
  $s_3 \geq 3$
Assessment of CP Scheduling

• Disjunctive scheduling may be viewed as the ‘killer application’ for CP
  – Natural modeling (activities and resources)
  – Allows many side constraints (precedence relations, time windows, setup times, etc.)
  – Among state of the art while being generic methodology

• However, CP has some problems when
  – objective is not minimize makespan (but instead, e.g., weighted sum of lateness)
  – setup times are present
  – ...

• What can MDDs bring here?
MDDs for Disjunctive Scheduling

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 ‘permutation MDD’

• Every solution can be written as a permutation $\pi$

  $\pi_1, \pi_2, \pi_3, \ldots, \pi_n$ : activity sequencing in the resource

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

  $\text{start}_{\pi_i} \geq \text{start}_{\pi_{i-1}} + p_{\pi_{i-1}} \quad i = 2, \ldots, n$
MDD Representation

Path \{1\} – \{3\} – \{2\}:

\[0 \leq \text{start}_1 \leq 1\]
\[6 \leq \text{start}_2 \leq 7\]
\[3 \leq \text{start}_3 \leq 5\]
Theorem: Constructing the exact MDD for a Disjunctive Instance is an NP-Hard problem

- We work with MDD relaxations instead
- Bounded size in specific cases, e.g. (Balas [99]):
  - TSP defined on a complete graph
  - Given a fixed parameter $k$, we must satisfy
    
    $$i \ll j \quad \text{if} \quad j - i \geq k$$
    
    for cities $i, j$

Theorem: The exact MDD for the TSP above has $O(n2^k)$ nodes
MDD-based propagation

Propagation: remove infeasible arcs from the MDD

We can utilize several structures/constraints:

• *Alldifferent* for the permutation structure
• Earliest start time and latest end time
• Precedence relations

For a given constraint type we maintain specific ‘state information’ at each node in the MDD
  – both top-down and bottom-up
Propagation (cont’d)

- State information at each node $i$
  - labels on all paths: $A_i$
  - labels on some paths: $S_i$
  - earliest starting time: $E_i$
  - latest completion time: $L_i$

- Top down example for arc $(u,v)$
Alldifferent Propagation

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

[Andersen et al., 2007]
Some-paths state: $S_u$

- Labels belonging to some path from node r to node u

$S_u = \{1,2,3\}$

- Identification of Hall sets

- Thus eliminate $\{1,2,3\}$ from $(u,v)$
- Earliest Completion Time: \( E_u \)
  - Minimum completion time of all paths from root to node \( u \)

- Similarly: Latest Completion Time
Propagate Earliest Completion Time

<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>4</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>5</td>
<td>2</td>
<td>10</td>
<td>3</td>
</tr>
</tbody>
</table>

- \( E_u = 7 \)
- Eliminate 4 from \((u,v)\)
Arc with label \( j \) infeasible if \( i \ll j \) and \( i \) not on some path from \( r \).

Suppose \( 4 \ll 5 \):
- \( S_u = \{1, 2, 3\} \)
- Since \( 4 \) not in \( S_u \), eliminate \( 5 \) from \((u, v)\).

Similarly: Bottom-up for \( j \ll i \).
More 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 $u$,
  - $A_u^\downarrow$: values in all paths from root to $u$
  - $A_u^\uparrow$: values in all paths from node $u$ to terminal

- Precedence relation $i \ll j$ holds if and only if $(j \not\in A_u^\downarrow)$ or $(i \not\in A_u^\uparrow)$ for all nodes $u$ in $M$

- Same technique applies to relaxed MDD
Extracting precedence relations

- Build a digraph $G = (V, E)$ where $V$ is the set of activities
- For each node $u$ in $M$
  - if $j \in A_u^\downarrow$ and $i \in A_u^\uparrow$ add edge $(i,j)$ to $E$
  - represents that $i \ll j$ cannot hold
- Take complement graph $\overline{G}$
  - complement edge exists iff $i \ll j$ holds

\[
\begin{aligned}
3 & \ll 1 \\
3 & \ll 2 \\
3 & \ll 4 \\
2 & \ll 4
\end{aligned}
\]

$G$

$\overline{G}$
Extracting precedence relations

- Build a digraph $G = (V, E)$ where $V$ is the set of activities
- For each node $u$ in $M$
  - if $j \in A_u^\downarrow$ and $i \in A_u^\uparrow$ add edge $(i,j)$ to $E$
  - represents that $i \ll j$ cannot hold
- Take complement graph $\overline{G}$
  - complement edge exists iff $i \ll j$ holds
- Time complexity: $O(|M|n^2)$
- Same technique applies to relaxed MDD
  - add an edge if $j \in S_u^\downarrow$ and $i \in S_u^\uparrow$
  - complement graph represents subset of precedence relations
Comparison to other methods

- Existing CP inference methods may not dominate the MDD propagation, even for small widths

<table>
<thead>
<tr>
<th>Act</th>
<th>rᵢ</th>
<th>dᵢ</th>
<th>pᵢ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>27</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>35</td>
<td>5</td>
</tr>
</tbody>
</table>

[Vilim, 2004]

- Edge finding and not-first/not-last deduce that $1 \ll 3$ and $2 \ll 3$, but no changes in time bounds
- MDD finds the same precedences, *and* deduces that $s₃ \geq 10 + 11 = 21$
1. Provide precedence relations from MDD to CP
   - update start/end time variables
   - other inference techniques may utilize them
   - (some of the precedence relations found by the MDD may not be detected by existing CP methods)

2. Filter the MDD using precedence relations from other (CP) techniques
Top-down MDD compilation

To refine the MDD, we generally want to identify equivalence classes among nodes in a layer

- NP-hard, but can be based on state information in practice, e.g., EST, LCT, alldifferent constraint ($A_i$ and $S_i$ states), ...
Computational Evaluation

- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation

- Three different variants
  - CPO (only use CPO propagation)
  - MDD (only use MDD propagation)
  - CPO+MDD (use both)
Problem classes

• Disjunctive instances with
  – sequence-dependent setup times
  – release dates and deadlines
  – precedence relations

• Objectives
  – minimize makespan
  – minimize sum of setup times
  – minimize total tardiness

• Benchmarks
  – Random instances with varying setup times
  – TSP-TW instances (Dumas, Ascheuer, Gendreau)
  – Sequential Ordering Problem
Importance of setup times

Random instances
- 15 jobs
- lex search
- MDD width 16
- min makespan

CPO Backtracks / MDD Backtracks

Importance of setup times
(increasing average length of setup times)
TSP with Time Windows

Dumas/Ascheuer instances
- 20-60 jobs
- lex search
- MDD width: 16
Minimize Total Tardiness

• Consider activity \(i\) with due date \(\delta_i\)
  – Completion time of \(i\): \(c_i = s_i + p_i\)
  – Tardiness of \(i\): \(\max\{0, c_i - \delta_i\}\)

• Objective: minimize total (weighted) tardiness

• 120 test instances
  – 15 activities per instance
  – varying \(r_i\), \(p_i\), and \(\delta_i\), and tardiness weights
  – no side constraints, setup times (measure only impact of objective)
  – lexicographic search, time limit of 1,800s
Total Tardiness Results

**Total Tardiness**

- **Total Tardiness**
- **Total Weighted Tardiness**

### Graphs

**Graph 1:**
- **X-axis:** Time (s)
- **Y-axis:** Number of Instances Solved
- **Legend:**
  - CPO
  - CPO + MDD Width 16
  - CPO + MDD Width 32
  - CPO + MDD Width 64
  - CPO + MDD Width 128
- **Lines:**
  - **MDD-128**
  - **MDD-64**
  - **MDD-32**
  - **MDD-16**
  - **CPO**

**Graph 2:**
- **X-axis:** Time (s)
- **Y-axis:** Number of Instances Solved
- **Legend:**
  - CPO
  - MDD-128
  - MDD-64
  - MDD-32
  - CPO
  - MDD-16
- **Lines:**
  - **MDD-128**
  - **MDD-64**
  - **MDD-32**
  - **CPO**
  - **MDD-16**

**Total Tardiness**

**Total Weighted Tardiness**
## Sequential Ordering Problem (TSPLIB)

<table>
<thead>
<tr>
<th>instance</th>
<th>vertices</th>
<th>bounds</th>
<th>CPO</th>
<th>best</th>
<th>time (s)</th>
<th>CPO+MDD, width 2048</th>
<th>best</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>br17.10</td>
<td>17</td>
<td>55</td>
<td>55</td>
<td>0.01</td>
<td></td>
<td>55</td>
<td>4.98</td>
<td></td>
</tr>
<tr>
<td>br17.12</td>
<td>17</td>
<td>55</td>
<td>55</td>
<td>0.01</td>
<td></td>
<td>55</td>
<td>4.56</td>
<td></td>
</tr>
<tr>
<td>ESC07</td>
<td>7</td>
<td>2125</td>
<td>2125</td>
<td>0.01</td>
<td></td>
<td>2125</td>
<td>0.07</td>
<td></td>
</tr>
<tr>
<td>ESC25</td>
<td>25</td>
<td>1681</td>
<td>1681</td>
<td>TL</td>
<td></td>
<td>1681</td>
<td>48.42</td>
<td></td>
</tr>
<tr>
<td>p43.1</td>
<td>43</td>
<td>28140</td>
<td>28205</td>
<td>TL</td>
<td></td>
<td>28140</td>
<td>287.57</td>
<td></td>
</tr>
<tr>
<td>p43.2</td>
<td>43</td>
<td>[28175, 28480]</td>
<td>28545</td>
<td>TL</td>
<td></td>
<td><strong>28480</strong></td>
<td><strong>279.18</strong></td>
<td>*</td>
</tr>
<tr>
<td>p43.3</td>
<td>43</td>
<td>[28366, 28835]</td>
<td>28930</td>
<td>TL</td>
<td></td>
<td><strong>28835</strong></td>
<td><strong>177.29</strong></td>
<td>*</td>
</tr>
<tr>
<td>p43.4</td>
<td>43</td>
<td>83005</td>
<td>83615</td>
<td>TL</td>
<td></td>
<td>83005</td>
<td>88.45</td>
<td></td>
</tr>
<tr>
<td>ry48p.1</td>
<td>48</td>
<td>[15220, 15805]</td>
<td>18209</td>
<td>TL</td>
<td></td>
<td>16561</td>
<td>TL</td>
<td></td>
</tr>
<tr>
<td>ry48p.2</td>
<td>48</td>
<td>[15524, 16666]</td>
<td>18649</td>
<td>TL</td>
<td></td>
<td>17680</td>
<td>TL</td>
<td></td>
</tr>
<tr>
<td>ry48p.3</td>
<td>48</td>
<td>[18156, 19894]</td>
<td>23268</td>
<td>TL</td>
<td></td>
<td>22311</td>
<td>TL</td>
<td></td>
</tr>
<tr>
<td>ry48p.4</td>
<td>48</td>
<td>[29967, 31446]</td>
<td>34502</td>
<td>TL</td>
<td></td>
<td><strong>31446</strong></td>
<td><strong>96.91</strong></td>
<td>*</td>
</tr>
<tr>
<td>ft53.1</td>
<td>53</td>
<td>[7438, 7531]</td>
<td>9716</td>
<td>TL</td>
<td></td>
<td>9216</td>
<td>TL</td>
<td></td>
</tr>
<tr>
<td>ft53.2</td>
<td>53</td>
<td>[7630, 8026]</td>
<td>11669</td>
<td>TL</td>
<td></td>
<td>11484</td>
<td>TL</td>
<td></td>
</tr>
<tr>
<td>ft53.3</td>
<td>53</td>
<td>[9473, 10262]</td>
<td>12343</td>
<td>TL</td>
<td></td>
<td>11937</td>
<td>TL</td>
<td></td>
</tr>
<tr>
<td>ft53.4</td>
<td>53</td>
<td>14425</td>
<td>16018</td>
<td>TL</td>
<td></td>
<td>14425</td>
<td>120.79</td>
<td></td>
</tr>
</tbody>
</table>

* solved for the first time
Extension: Lagrangian bounds

• Observation: MDD bounds can be very loose

Main cause: repetition of activities

Proposed remedy:
• add Lagrangian relaxation
• penalize repeated activities

\[
\min z + \sum_{j=1}^{n} \lambda_j \left( \sum_{i=1}^{n} \pi_i = j \right) - 1
\]

\[
= z + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_j \pi_i = j - \sum_{j=1}^{n} \lambda_j
\]

• Shortest path with updated weights
Example: Relaxed Decision Diagram

- First task: r = 2
- Second task: r = 1
- Third task: r = 1

Tasks:
- 2
- 4
- 3

Release Date:
- r = 2
- r = 1
- r = 1
Shortest Path: Lower Bound

• Shortest path
  • Length: Lower bound on the optimal solution value
Shortest Path: Lower Bound

First task
- Arc 2
- Completion Time: 8
- Release Date: r = 2

Second task
- Arc 4
- Release Date: r = 1

Third task
- Arc 3
- Release Date: r = 1

Tasks
- 2
- 4
- 3

Tasks Layout:

- First task
- Second task
- Third task

Completion Time: 8
• Solutions of a relaxed DD may violate several constraints of the problem

• **Violation**: “All tasks performed once”

\[
\sum_{e \mid v(e) = i} x_e = 1 \quad \text{for all tasks } i
\]
Remedy: Lagrangian Relaxation

\[
\begin{align*}
\text{min} & \quad z = \text{shortest path} \\
\text{s.t.} & \quad \sum_{e \mid v(e) = i} x_e = 1, \text{ for all tasks } i \\
& \quad (+\text{other problem constraints}) \\
& \quad \text{Lagrangian multipliers } \lambda_i
\end{align*}
\]

\[
\begin{align*}
\text{min} & \quad z = \text{shortest path} + \sum_{i} \lambda_i (1 - \sum_{e \mid v(e) = i} x_e ) \\
\text{s.t.} & \quad (\text{other problem constraints}) \\
& \quad \text{This is done by updating shortest path weights!}
\end{align*}
\]

[Bergman et al., 2015]
We penalize infeasible solutions in a relaxed DD: any separable constraint of the form

\[ f_1(x_1) + f_2(x_2) + ... + f_n(x_n) \leq c \]

that **must** be satisfied by solutions of an MDD can be dualized.

We need only to focus on the **shortest path solution**

- Identify a violated constraint and penalize
- Systematic way directly adapted from LP
- Shortest paths are **very fast** to compute
Improving Relaxed Decision Diagram

Arcs and Tasks

First task: r = 2
Second task: r = 1
Third task: r = 1

Tasks

- 2
- 4
- 3

Release Date

- r = 2
- r = 1
- r = 1
Improving Relaxed Decision Diagram

Penalization:
- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight
Improving Relaxed Decision Diagram

Penalization:
- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight

Completion Time: 8
Improving Relaxed Decision Diagram

Penalization:
- If a task is repeated, increase its arc weight
- If a task is unused, decrease its arc weight
Improving Relaxed Decision Diagram

- New shortest path: 10
  - Guaranteed to be a valid lower bound for any penalties
Cost-Based Filtering

• If minimum solution value through an arc exceeds $\max(D(z))$ then arc can be deleted

• Suppose a solution of value 10 is known

• MDD filtering extends to Lagrangian weights: More filtering possible
Impact on TSP with Time Windows

TSPTW instances

(Constraints, 2015)
State-Dependent Costs

Context and Motivation

• Time-dependent sequencing
  – machine scheduling, routing

• Challenging problem
  – best results so far use dedicated methods
  – not easy to extend with side constraints

• Utilize constraint programming framework?
  – strengthened constraint propagation with MDDs
  – improved bounds via additive bounding with LP
  – evaluate on TD-TSP and TD-SOP
Time-Dependent Sequencing

• Activities
  – processing time $p_i$
  – released date $r_i$
  – deadline $d_i$

• Resource
  – non-preemptive
  – process one activity at a time
  – sequence-dependent setup times: also depend on position!

\[ \delta_{i,j}^t = \text{setup time between } i \text{ and } j \text{ if } i \text{ is at position } t \]
Constraint Programming Model

- Variables \( \pi_i \): label of \( i^{th} \) activity in the sequence
  \( L_i \): position of activity \( i \) in the sequence

\[
\begin{align*}
\text{min} & \quad \sum_{i=0}^{n} \delta_{\pi_i, \pi_{i+1}}^i \\
\text{s.t.} & \quad \text{AllDiff}(\pi_1, \ldots, \pi_n) \\
& \quad L_{\pi_i} = i \quad \forall i = 1, \ldots, n \\
& \quad L_i < L_j \quad \forall (i \ll j) \in P \\
& \quad L_i \in \{1, \ldots, n\} \quad \forall i = 1, \ldots, n \\
& \quad \pi_i \in \{1, \ldots, n\} \quad \forall i = 1, \ldots, n
\end{align*}
\]

- **Weak model**: objective and AllDiff are decoupled
MDD-based propagation

Update MDD propagation algorithms:

- \textit{Alldifferent} for the permutation structure
  - unchanged
- Precedence relations
  - unchanged
- Earliest start time and latest end time
  - adapt rule: $\delta_{i,j}$ becomes $\delta_{i,j}^t$
- \textbf{Objective}
  - minimize sum of setup times
Updated CP Model

\[
\begin{align*}
\text{min} \quad & z \\
\text{s.t.} \quad & \text{AllDiff}(\pi_1, \ldots, \pi_n) \\
& \text{MDDconstr}(\pi_1, \ldots, \pi_n, W, z, \delta^t, P) \\
& L_{\pi_i} = i \quad \forall i = 1, \ldots, n \\
& L_i < L_j \quad \forall (i \ll j) \in P \\
& L_i \in \{1, \ldots, n\} \quad \forall i = 1, \ldots, n \\
& \pi_i \in \{1, \ldots, n\} \quad \forall i = 1, \ldots, n \\
& z \in \{0, \ldots, \infty\}
\end{align*}
\]

Stronger model: objective handled within MDD constraint
Additive Bounding

Add LP reduced costs to MDD relaxation

- Continuous LP relaxation ‘discretized’ through MDD
- Stronger bounds
- Improved cost-based filtering

(Fischetti & Toth, 1989)
MIP and LP relaxation

• Time-space network model (Picard & Queyranne, 1978)

• Variables

\[
x_{i,j}^t = \begin{cases} 
1 & \text{if } i \text{ is performed at } t \text{ and followed by } j \\
0 & \text{otherwise}
\end{cases}
\]

• Constraints: flow conservation; perform each activity

• Valid inequalities: subtour and 4-cycle elimination
Embedding reduced costs in MDD

- State information at each node $i$
  - shortest path from root to $i$ with respect to $\overline{c}_{i,j}^t$
  - root node initialized with LP objective value

- Since MDD is relaxation, shortest path is valid bound
  - filter edges that do not participate in improving shortest path

- MDD maintains both the original objective and this new ‘additive bound’ constraint
Experiments

• Time-dependent TSP and SOP benchmarks
  – 38 instances from TSPLIB (14-107 jobs)
  – $\delta_{i,j}^t = (n-t) \cdot \delta_{i,j}$  
    [Abeledo et al. 2013]

• Time limit: 30 minutes

• MDD added to IBM ILOG CP Optimizer 12.4
  – maximum width 1024

• MIP model (IBM ILOG CPLEX 12.4)
  – state-space integer program
  – subtour and 4-cycle elimination constraints
  – LP relaxation takes several hours for $\geq$90 vertices
Results on Time-dependent TSP

Note: Dedicated branch, price and cut algorithm (Abeledo et al., 2013) solves more TD-TSP instances optimally.
## Results on Time-dependent SOP

<table>
<thead>
<tr>
<th>Method</th>
<th>#Solved</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIP</td>
<td>6/30</td>
</tr>
<tr>
<td>Pure CP</td>
<td>5/30</td>
</tr>
<tr>
<td>CP + MDD + Additive Bounding</td>
<td>10/30</td>
</tr>
</tbody>
</table>

On average, additive MDD+LP bound improves
- LP root node bound by 51.41%
- MDD root node bound by 9.54%
Conclusion

• MDD propagation natural generalization of domain propagation
  – Strength of MDD relaxation can be controlled by the width
  – Huge reduction in solution time is possible
• For sequencing/disjunctive scheduling problems
  – MDD can handle all side constraints and objectives from existing CP scheduling systems
  – Polynomial cases (e.g., Balas variant)
  – MDD propagation algorithms (alldifferent, time windows, ...)
  – Extraction of precedence constraints from MDD
  – Can be enriched with math programming relaxations
  – Great addition to constraint-based systems