Relaxation Based on Multivalued Decision Diagrams

Samid Hoda
Exxon-Mobil

Willem-Jan van Hoeve
J.N. Hooker
Carnegie Mellon University

INFORMS 2010
Relaxation in CP and MIP

- Relaxation in CP
  - Domain store (set of variable domains)
  - Fast and versatile
  - Suitable for propagation
  - But very weak
  - Imbalance of search and inference
  - Must process many nodes quickly

- Relaxation in MIP
  - Generally LP
  - Strong, rich structure
  - Can be strengthened with cutting planes
  - But requires linearization
  - May be weak for combinatorial or nonconvex constraints
  - Unsuitable for propagation
A New Relaxation

- **Multivalued decision diagram (MDD).**
  - Discrete structure
  - A compact representation of branching tree
  - Optimization = shortest path computation
  - Domain store is a special case

- MDD relaxation.
  - An approximate representation of constraints
  - Improve the relaxation with node splitting, edge domain filtering, and shortest path reasoning.

- Advantages.
  - Like domain store, suitable for propagation.
  - Like LP, provides strong relaxation
  - Rebalances search and inference
  - Natural for discrete problems
Objectives

• Show that MDD-based propagation can follow a standard pattern.
  ▪ And yet can be specialized to each type of constraint to exploit structure.
  ▪ Implement a general MDD-based constraint solver based on this pattern.

• Test the MDD-based solver on a class of constraints.
  ▪ Multiple among constraints.
MDD representing
\[ x_1 + x_2 + x_3 \leq 5 \]
\[ x_j \in \{1,2,3\} \]

Assigns \( x_1 = 3 \)
MDD basics

MDD representing
\[ x_1 + x_2 + x_3 \leq 5 \]
\[ x_j \in \{1, 2, 3\} \]

Assigns \( x_1 = 3 \)

Represents 2 edges
MDD basics

MDD representing
\[ x_1 + x_2 + x_3 \leq 5 \]
\[ x_j \in \{1,2,3\} \]

Assigns \( x_1 = 3 \)

Edge domain
MDD basics

MDD representing

\[ x_1 + x_2 + x_3 \leq 5 \]
\[ x_j \in \{1,2,3\} \]

Each path from top to bottom represents a set of solutions

\[
\{1\} \times \{1\} \times \{1,2,3\} = \{(1,1,1), (1,1,2), (1,1,3)\}
\]

10 solutions total
**MDD relaxation**

MDD representing
\[ x_1 + x_2 + x_3 \leq 5 \]
\[ x_j \in \{1,2,3\} \]

A **relaxed** MDD (width 2) representing a superset of the solutions

10 solutions

15 solutions
MDD relaxation

Conventional domain store
(width 1 relaxation)

MDD representing

\[ x_1 + x_2 + x_3 \leq 5 \]

\[ x_i \in \{1, 2, 3\} \]

10 solutions
We will use among constraint as a running example.

At least \( L \) and at most \( U \) variables in set \( X \) take a value in set \( S \):

\[
\text{among}(X, S, L, U)
\]

Applications in sequencing and scheduling, etc.
Among constraint

- Example: Employee scheduling

\[ x_t = \text{shift assigned on day } t. \]
\[ S = \{M,A,N\} = \{\text{shifts}\} \]

Between 2 and 4 afternoon/night shifts in a week:

\[ \text{Among}(\{x_1, \ldots, x_7\}, \{A,N\}, 2, 4) \]

<table>
<thead>
<tr>
<th>( t )</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>( x_t )</td>
<td>M</td>
<td>A</td>
<td>N</td>
<td>M</td>
<td>M</td>
<td>A</td>
<td>M</td>
</tr>
</tbody>
</table>

WOLOG, suppose \( x_t \in \{0,1\}, S = \{1\} \)

<table>
<thead>
<tr>
<th>( t )</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>( x_t )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Among constraint

Exact MDD for $\text{among}\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2$
MDD-based constraint programming

• Maintain limited-width MDD
  ▪ Serves as relaxation
  ▪ Typically start with width 1 (domain store)
  ▪ Update MDD by propagating constraints sequentially

• Constraint Propagation
  ▪ Edge filtering: Remove provably inconsistent edges
  ▪ Node refinement: Split nodes to separate edge information

• Search
  ▪ As in classical CP, but may now be guided by MDD

• Easily extended to optimization
  ▪ Shortest path in MDD
Example

- We will propagate

\[ \text{among}(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2) \]

through a BDD of maximum width 3

- All path lengths must be = 2.
Example

Initially, the MDD has width 1 (traditional domain store)

among(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2)
Example

Try to filter edge domain

No filtering possible.

Initially, the MDD has width 1
(traditional domain store)

among\(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2\)
Example

Split? → among({x₁,x₂,x₃,x₄}, {1}, 2,2)
Example

Split? The 2 incoming edges (length 0 and 1) are not equivalent.

The partial assignments \( x_1 = 0, x_1 = 1 \) don’t have the same set of possible completions.

\[
\text{among}\{(x_1, x_2, x_3, x_4), \{1\}, 2, 2\}
\]
So we split. This doesn’t exceed max width of 3.

\[
\text{among}\left(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2\right)
\]
Example

So we split. This doesn’t exceed max width of 3.

Duplicate outgoing Edges.

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Example

Try to filter edge domain.

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Try to filter edge domain.

Compute all path lengths from top...

\[
\text{among} \{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2
\]
Example

Try to filter edge domain.

Compute all path lengths from top... and from bottom.

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Example

Try to filter edge domain.

Compute all path lengths from top...
and from bottom.

Both domain values are consistent.

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
No filtering possible for $x_2$.

$$\text{among}({x_1, x_2, x_3, x_4}, \{1\}, 2, 2)$$
Example

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Example

Incoming edges partition into 3 equivalence classes, corresponding to path lengths 0, 1, 2

$$\text{among}\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2$$
So we split into 3 nodes

among(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2)
Duplicate outgoing edges

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
To speed filtering, we identify approximate equivalence classes of incoming edges...

\[ \text{among}\left(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2\right) \]
To speed filtering, we identify **approximate** equivalence classes of incoming edges, based on **shortest** and **longest** paths.

\[
\text{among}([x_1, x_2, x_3, x_4], \{1\}, 2, 2)
\]
Example

Shortest and longest path from top:

among(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2)
Example

Shortest and longest path from bottom

SP, LP = 0,0

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Example

\[
\text{SP, LP} = 0, 0
\]

0 is inconsistent

\[
\text{among} \{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2
\]
Two edge domains can be filtered.

\[
\text{among}(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2)
\]
Continue this process...

$\text{among}\{x_1,x_2,x_3,x_4\}, \{1\}, 2, 2$
...and obtain a relaxed MDD of width 3

among({x_1,x_2,x_3,x_4}, {1}, 2, 2)
In this case, the MDD is exact.

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
Example

Now, propagate the next constraint through this MDD...

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
General Scheme for Node Splitting

- Identify (approximate) equivalence classes of incoming edges.
- Create a copy of the node for each equivalence class.
  - But without exceeding the max width.
- Duplicate outgoing arcs.

- Method for identifying equivalence classes is *constraint-specific*.
  - This exploits *special structure*.
  - Can also be *dynamic*, e.g. depending on current width.
• Filtering on an edge is based on information at either end of the edge.

• Information is accumulated during top-down and bottom-up passes.
  ▪ Based on extend ($\otimes$) and merge ($\oplus$) operations.

• For example...
Suppose we don’t split $u_4$ and want to filter this edge domain among ($\{x_1, x_2, x_3, x_4\}$, $\{1\}$, 2, 2)
We want to compute **top down** information—shortest & longest path length among\(\{(x_1, x_2, x_3, x_4), \{1\}, 2, 2\}\)
First **extend** node information across edges by applying operator \( \otimes = + \)

\[
\begin{align*}
(0,0) \otimes \{0,1\} &= (0,0), (1,1) \\
(1,1) \otimes \{0,1\} &= (1,1), (2,2)
\end{align*}
\]

among(\(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2\))
General Scheme for Filtering

Merge incoming edge information by applying operator $\oplus = (\text{min}, \text{max})$

among($\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2$)
General Scheme for Filtering

Compute bottom up information

among({x_1,x_2,x_3,x_4}, {1}, 2 ,2)
General Scheme for Filtering

Filter based on information at either end of edge

among({x_1, x_2, x_3, x_4}, {1}, 2, 2)
General Scheme for Filtering

Filter based on information at either end of edge

Delete 1 from edge domain

among(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2)
General Scheme for Filtering

\[ \text{among}(X, S, L, U) \]

\[
\begin{align*}
&x_k & u & \text{Top down information} = (SP, LP) \\
&\{e\} & & \text{Shortest, longest path to } u \\
&x_{k+1} & u' & \text{Bottom up information} = (SP, LP) \\
& & & \text{Shortest, longest path to } u'
\end{align*}
\]
General Scheme for Filtering

\[ x_k \quad u \quad \text{Top down information} = (SP, LP) \]

among(\(X, S, L, U\))

\[ \{e\} \quad \text{Delete } e \text{ if } SP + e + SP' > U \]
\[ \quad \text{or } LP + e + LP' < L \]

\[ x_{k+1} \quad u' \quad \text{Bottom up information} = (SP', LP') \]
General Scheme for Filtering

$\mathbf{x}_k \quad u \quad \text{Top down information} = (SP, LP)$

among($X, S, L, U$)

$\{e\}$

Delete $e$ if $SP + e + SP' > U$

or $LP + e + LP' < L$

$\mathbf{x}_{k+1} \quad u' \quad \text{Bottom up information} = (SP', LP')$

$(SP, LP) \otimes \{e\} = (SP + e, LP + e)$

$(SP_1, LP_1)$ \quad (SP_2, LP_2)$

$\oplus$

$(\min\{SP_1, SP_2\}, \max\{LP_1, LP_2\})$
Any binary constraint $B$ with variables $x_i, x_j$

- $S = \{\text{all values assigned to } x_i \text{ on some path to } u\}$
- $T = \{\text{all values assigned to } x_j \text{ on some path to } u'\}$
General Scheme for Filtering

Any binary constraint $B$ with variables $x_i, x_j$

- $X_k \rightarrow u \rightarrow S = \{\text{all values assigned to } x_i \text{ on some path to } u\}$
- $\{e\}$
- $x_{k+1} \rightarrow u' \rightarrow T = \{\text{all values assigned to } x_j \text{ on some path to } u'\}$

Delete $e$ if $k = j$ and $(x_i, x_j) = (v, e)$ satisfies $B$

for no $v \in S$

or $k = i$ and $(x_i, x_j) = (v, e)$ satisfies $B$

for no $v \in T$
Any binary constraint $B$ with variables $x_i$, $x_j$

- $S = \{\text{all values assigned to } x_i \text{ on some path to } u\}$
- Delete $e$ if $k = j$ and $(x_i, x_j) = (v, e)$ satisfies $B$ for no $v \in S$
- or $k = i$ and $(x_i, x_j) = (v, e)$ satisfies $B$ for no $v \in T$
- $T = \{\text{all values assigned to } x_j \text{ on some path to } u'\}$

$S \otimes \{e\} = \{e\}$ if $k = i$, $S$ otherwise
Any binary constraint $B$ with variables $x_i, x_j$.

**Theorem.** This scheme achieves MDD consistency for any binary constraint $B$ in time that is polynomial in size of $B$ and the MDD.
**Theorem.** This scheme achieves MDD consistency for any binary constraint $B$ in time that is polynomial in size of $B$ and the MDD.

**Corollary.** Polytime MDD consistency for: equality, not-equal constraints, element constraint, etc.
General Scheme for Filtering

\[ \text{alldiff}(X) \]

\[ X_k \quad u \quad (A,S) \]
\[ \{e\} \]
\[ \text{all values that appear on some path to } u \}
\[ \text{all values that appear on all paths to } u \}

\[ X_{k+1} \quad u' \quad (A',S') \]
General Scheme for Filtering

\[
x_k \quad u \quad (A,S)
\]

\[
\text{alldiff}(X)
\]

\[
\{e\}
\]

\[
x_{k+1} \quad u' \quad (A',S')
\]

Delete \( e \) if \( e \in A \cup A' \), or:

\[|S| = k - 1 \text{ or } |S'| = n - k\]

(variables above or below form a Hall set)
General Scheme for Filtering

\[
\begin{align*}
    x_k & \quad u \quad (A,S) \\
    \{e\} & \quad \text{Delete } e \text{ if } e \in A \cup A', \text{ or:} \\
    \quad & \quad |S| = k - 1 \text{ or } |S'| = n - k \\
    \quad & \quad \text{(variables above or below form a Hall set)} \\
    x_{k+1} & \quad u' \quad (A',S') \\
    (A,S) \otimes \{e\} & = (A \cup \{e\}, S \cup \{e\}) \\
\end{align*}
\]

\[
\begin{align*}
    (A_1,S_1) & \quad \oplus \quad (A_2,S_2) \\
    & \quad (A_1 \cap A_2, S_1 \cup S_2)
\end{align*}
\]
General Scheme for Filtering

Unary resource constraint

\[ x_i = \text{activity in position } i \]

of sequence

\[ r_i, d_i = \text{release time, deadline} \]

Enforce\[ \text{alldiff}(X) \]

plus:

\[ x_k \quad u \quad \text{EST} \]

Earliest start time of \( x_k \), given previous activities in sequence

\[ x_{k+1} \quad u' \quad \text{LFT} \]

Latest finish time of \( x_{k+1} \), given subsequent activities in sequence
General Scheme for Filtering

Unary resource constraint

\( x_k u \) EST

\( \{e\} \) Delete \( e \) if \( \max \{\text{EST}, r_e\} + p_e > \min \{\text{LFT}, d_e\} \)

\( x_{k+1} u' \) LFT

\( r_i, d_i = \) release time, deadline

Enforce all \( \text{diff}(X) \) plus:
General Scheme for Filtering

**Unary resource constraint**

\[ x_k \ u \ EST \]

\[ x_i = \text{activity in position } i \text{ of sequence} \]

\[ r_i, d_i = \text{release time, deadline} \]

Enforce \( \text{alldiff}(X) \) plus:

\[ \text{EST} \otimes \{e\} = \max\{\text{EST} + p_e, r_e + p_e\} \]

\[ \text{LFT} \otimes \{e\} = \min\{\text{LFT} - p_e, d_e - p_e\} \]

Delete \( e \) if \( \max\{\text{EST}, r_e\} + p_e > \min\{\text{LFT}, d_e\} \)
• Our MDD-based solver is built around this general scheme for splitting and filtering.

• Plug in constraint-specific (approximate) equivalence test for splitting.

• Plug in constraint-specific $\otimes$, $\oplus$ for filtering.
Experiments – Previous work

• Multiple alldiff constraints
  ▪ Andersen et al., CP 2007
  ▪ MDDs of width 5
  ▪ Reduce search tree of 1 million+ nodes to 1 node.
  ▪ Reduce time by factor of 30
  ▪ Using old software

• Equality constraints $\sum_j g_j(x_j) = b$
  ▪ Hadzic et al., CPAIOR 2008

• Certain configuration problems
  ▪ Hadzic et al, CP 2008
Experiments

- Random multiple among constraints
- Nurse rostering instances
- Compare domain store (conventional filtering) with MDD store
  - Using MDDs of increasing widths.
Experiments

- Random multiple among constraints
  - 50 binary variables total
  - 5 variables per among constraint
  - Indices chosen from normal distribution with mean chosen from $U[1..50] \mod 50$, and standard deviation 2.5
  - So, indices in a given among are near-consecutive, typical of applications.
  - $(L,U) = (2,3)$
  - Number of amongs varies from 5 to 200 (in steps of 5), 100 instances for each number.
  - This range includes phase transition.
Multiple Amongs: Backtracks

Domain store vs width 4 MDD
Multiple Amongs: Backtracks

Domain store vs width 16 MDD
Multiple Amongs: Running Time

Domain store vs width 4 MDD
Multiple Amongs: Running Time

Domain store vs width 16 MDD
Experiments

• Nurse rostering instances
  ▪ Based on van Hoeve et al., *Constraints* 2009
  ▪ Work 4-5 days per week
  ▪ Max A days every B days (Max A/B)
  ▪ Min C days every D days (Min A/B)
  ▪ Time horizon ranges from 40 to 80 days

• Three problem classes
  ▪ I: max 6/8, min 22/30
  ▪ II: max 6/9, min 20/30
  ▪ III: max7/9, min 22/30
Nurse rostering problems

Computation time (sec) vs time horizon, Class I
Nurse rostering problems

Computation time (sec) vs time horizon, Class II
Nurse rostering problems

Computation time (sec) vs time horizon, Class III
Conclusions

• MDD-based propagation can be carried out according to a general scheme.
  ▪ Equivalence testing for node splitting
  ▪ Information operators for filtering
  ▪ Yet the scheme allows exploitation of constraint-specific structure

• MDD-based constraint solving yields substantial speedup over domain store for multiple among constraints.

• Intensive processing at search nodes can pay off when more information is communicated between constraints.