MDD-based propagation of among constraints

Samid Hoda     Willem-Jan van Hoeve     John N. Hooker
Carnegie Mellon University

INFORMS Annual Meeting
San Diego – October 14, 2009
Outline

• Introduction
  ▪ Domain stores and MDD stores
  ▪ Propagation

• Experiments
  ▪ Generating all solutions
  ▪ Finding the first feasible

• Conclusion and research issues
INTRODUCTION
Local vs global

• This tension pervades science and mathematics
  ✓ Pros of local structure: **simplicity**
  ✗ Cons of local structure: **limited**, lacks global pt of view

• Strategies for extending local reasoning
  ▪ Expand the notion of locality
  ▪ Combine summaries of local structure
Constraint programming

• In CP local structure = processing individual (basic) constraints
  ✓ Pros: able to exploit structure of (basic) constraints
  ✗ Cons: overlooks implications of combined constraints

• Strategies
  ▪ Expanding notion of locality: global constraints
  ▪ Combining summaries of local structure: domain store
Domain store

• Stores the current variable domains
  ▪ Values that occur in some feasible solution

• The domain store relaxation
  ▪ Provides a (weak) summary of global structure
  ▪ Combines local structure (can be very lossy)
  ▪ Basis for constraint propagation
Domain store: advantages

✓ Simple structure
  • Provides natural input to filtering algorithms
  • Minimal overhead when embedded in search

✓ Guides branching (on variables) in a natural way
  • Just split variables
Domain store: disadvantages

✗ Transmits relatively **little information** between constraints

✗ A **weak relaxation** of the problem
  - Ignores variable interaction
  - Relaxation is a Cartesian product of domains

✗ Result
  - Search trees **too large**
  - **Too little** processing at each node
A stronger relaxation

• Enrich the constraint store
  ▪ Use a relaxed *multivalued decision diagram* (MDD)
  ▪ With binary domains MDD = BDD (binary decision diagram)

• An MDD is a **compact representation** of the search tree
  ▪ Isomorphic subtrees are merged
  ▪ An MDD is relaxed by limiting width
Advantages of a BDD store

✓ Transmits more information than a domain store
  - Strength is adjustable: depends on width

✓ Guides branching in a natural way
  - Representation is closer to branching tree

✓ Results
  - Smaller search trees
  - Justifies more processing per node
  - Better integration of CP/IP
Global constraints and MDD stores

• Global constraints
  ▪ Static
  ▪ Modeler **imposes** structure

• MDD store
  ▪ Dynamic
  ▪ **Identifies** structure as the solution process evolves

• Best of both
  ▪ Propagating global constraints through MDD
EXAMPLE

SEARCH TREE

among(\{x_1, x_3, x_4\}, \{1\}, 2, 2)
REMOVE INFEASIBLE SOLUTIONS

among({x_1, x_3, x_4}, {1}, 2, 2)
among({x_1, x_3, x_4}, {1}, 2, 2)
EXAMPLE

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

REMOVE REDUNDANT EDGES
$x_1$  $x_2$  $x_3$  $x_4$

REDUCED BDD

among($\{x_1, x_3, x_4\}, \{1\}, 2, 2$)
among(\{x_1, x_3, x_4\}, \{1\}, 2, 2)
Each path corresponds to a Cartesian product of solutions

\[
\{1\} \times \{0,1\} \times \{0\} \times \{0\}
\]

among (\{x_1, x_3, x_4\}, \{1\}, 2, 2)
$\text{EXACT BDD}$

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

$\text{NEW CONSTRAINT}$

$\text{6 SOLUTIONS}$
Let's use a BDD of maximum width 2.

RELAXED BDD

14 SOLUTIONS

among(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2)
A BDD with maximum width 1 is just the domain store

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

\[ x_1 : \{0,1\} \]

\[ x_2 : \{0,1\} \]
BRANCHING SEARCH

\[ x_1 : \{0,1\} \]
\[ x_2 : \{0,1\} \]
\[ x_3 : \{1\} \]
And so forth.

Less branching than with domain store.
Complete (domain store) filter for among

- **Special case:** \( \text{among}(\{x_1, \ldots, x_n\}, \{1\}, L, U) \) where \( D(x_i) \subseteq \{0,1\} \)
  - Let \( SP = \left| \{x_i : 0 \in D(x_i)\} \right| \) and \( LP = \left| \{x_i : 1 \in D(x_i)\} \right| \)

  \( \times \) If \( LP < L \) or \( SP > U \) then inconsistent
  \( \checkmark \) If \( LP = L \) then filter 0 from non-singleton domains
  \( \checkmark \) If \( SP = U \) then filter 1 from non-singleton domains

- **General case** can be reduced to special case
  - Use element constraint
Propagation in MDDs

- Propagate in a MDD using
  - edge domain filtering, and
  - refinement (node splitting)
  - without exceeding maximum width

- Example:
  - We will propagate among \( \{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2 \) through a BDD of maximum width 3
Try to filter edge domain \((u_1, u_2)\)

SP using \((u_1, u_2, \{0, 1\})\) has length < \(U\)

LP using \((u_1, u_2, \{0, 1\})\) has length > \(L\)

Can't filter

Path lengths are from the root \(u_1\) to the sink \(1\)

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

Split $u_2$?

SP using $(u_1,0) = 0$
SP using $(u_1,1) = 1$

Incoming edge-value pairs are not equivalent: so split $u_2$

$$\text{among} \left( \{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2 \right)$$
SPLIT $u_2$ into two classes (less than maximum width)

$\text{among} (\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2)$
among\(\{x_1, x_2, x_3, x_4\}, \{1\}, 2, 2\)
Filter edge domains $(u_2', u_3)$ and $(u_2'', u_3)$

(No filtering possible)

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

Split $u_3$?

$SP(u_2',u_3,0) = 0$

$SP(u_2',u_3,1) = 1$

$SP(u_2'',u_3,0) = 1$

$SP(u_2'',u_3,1) = 2$

Split $u_3$ into 3 equivalence classes

$among([x_1,x_2,x_3,x_4],\{1\},2,2)$
Split $u_3$ into 3 equivalence classes

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

Duplicate outgoing edges
EXAMPLE

Filter edge domains

LP using \((u_3', u_4, 0) = 1 < 2 = L\)

SP using \((u_3'', u_4, 1) = 3 > 2 = U\)

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

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

• Example: edge-value equivalence was exact
  ▪ Problem: a few nodes “consume” BDD when processing a constraint
  ▪ Want intra-constraint diversification
  ▪ Want inter-constraint diversification

• One solution: approximate equivalence
  ▪ Edge-value pairs are equivalent if SPs/LPs differ by at most some threshold value
EXPERIMENTS
Problem instances

- Nurse rostering instances (horizon $n$ days)
  - Work 4-5 days per week
  - Max $A$ days every $B$ days (Max $A/B$)
  - Min $C$ days every $D$ days (Min $A/B$)

- Class 1 Max 6/8 Min 22/30
- Class 2 Max 6/9 Min 20/30
- Class 3 Max 7/9 Min 22/30

<table>
<thead>
<tr>
<th>Horizon</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>2284</td>
<td>4575</td>
<td>6567</td>
<td>2810</td>
<td>730</td>
</tr>
<tr>
<td>Class 2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Class 3</td>
<td>137593</td>
<td>388726</td>
<td>718564</td>
<td>105618</td>
<td>22650</td>
</tr>
</tbody>
</table>
FINDING ALL FEASIBLE SOLUTIONS
Computation time

Class 1 (n=80)

Time (sec)

Maximum Width

Threshold 1
Threshold 2
Threshold 3
Threshold 4
Threshold 5
Computation time

Class 2 (n=40)

![Graph showing computation time vs. maximum width for different thresholds.](image)
Computation time

Class 3 (n=80)

Time (sec)

Maximum Width

Threshold 1
Threshold 2
Threshold 3
Threshold 4
Threshold 5
Search tree nodes

Class 1 (n=80)

Maximum Width

Search Tree Nodes

Threshold 1
Threshold 2
Threshold 3
Threshold 4
Threshold 5
Search tree nodes

Class 2 (n=80)

Search Tree Nodes

Maximum Width

Threshold 1
Threshold 2
Threshold 3
Threshold 4
Threshold 5
Search tree failures

Class 1 (n=80)

Tree Failures

Maximum Width

Threshold 1
Threshold 2
Threshold 3
Threshold 4
Threshold 5
Search tree failures

Class 2 (n=40)

Tree Failures vs. Maximum Width

- Threshold 1
- Threshold 2
- Threshold 3
- Threshold 4
- Threshold 5
Search tree failures

Class 3 (n=80)
FINDING THE FIRST FEASIBLE
Computation time

Class 2 (n=80)
Class 2 (n=80)

Search tree nodes

- Threshold 1
- Threshold 2
- Threshold 3
- Threshold 4
- Threshold 5

*Maximum Width vs. Search Tree Nodes*
Search tree failures

Class 1 (n=40)

Tree Failures

Threshold 1
Threshold 2
Threshold 3
Threshold 4
Threshold 5

Maximum Width
Search tree failures

Class 1 (n=80)

Tree Failures

Maximum Width

Threshold 1
Threshold 2
Threshold 3
Threshold 4
Threshold 5
Search tree failures

Class 3 (n=40)

Tree Failures vs. Maximum Width

- Threshold 1
- Threshold 2
- Threshold 3
- Threshold 4
- Threshold 5
Compared to the state of the art

<table>
<thead>
<tr>
<th>Class</th>
<th>Size</th>
<th>gcc+seq BT</th>
<th>gen-seq BT</th>
<th>among BDD store BT</th>
<th>gcc+seq CPU</th>
<th>gen-seq CPU</th>
<th>among BDD store CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class 1</td>
<td>40</td>
<td>185287</td>
<td>0</td>
<td>253</td>
<td>216.49</td>
<td>0.77</td>
<td>3.76</td>
</tr>
<tr>
<td>Class 1</td>
<td>80</td>
<td>198091</td>
<td>0</td>
<td>97</td>
<td>1061.62</td>
<td>0.61</td>
<td>7.50</td>
</tr>
<tr>
<td>Class 2</td>
<td>40</td>
<td>393748</td>
<td>0</td>
<td>204</td>
<td>390.93</td>
<td>0.01</td>
<td>1.38</td>
</tr>
<tr>
<td>Class 2</td>
<td>80</td>
<td>393748</td>
<td>0</td>
<td>51</td>
<td>1786.62</td>
<td>0.05</td>
<td>5.38</td>
</tr>
<tr>
<td>Class 3</td>
<td>40</td>
<td>328376</td>
<td>0</td>
<td>510</td>
<td>417.63</td>
<td>34.43</td>
<td>177.37</td>
</tr>
<tr>
<td>Class 3</td>
<td>80</td>
<td>1847335</td>
<td>0</td>
<td>295</td>
<td>7457.36</td>
<td>15.41</td>
<td>160.85</td>
</tr>
</tbody>
</table>
CONCLUSION AND RESEARCH ISSUES
Conclusion

- MDD store provides **substantial advantage** over domain store for filtering **multiple among constraints**
  - **Wider** MDDs yield greater speedups
  - **Huge reduction** in the amount of backtracking

- **Intensive processing** at search nodes can pay off when the constraint store is richer
Some research issues

• Adjusting the **width** and **threshold**
  - **Dynamic** adjustment

• Interaction with branching schemes

• How to propagate other constraints?
  - **Regular** constraint
  - **Sequence** constraint