MDD Filtering for Sequence Constraints

Willem-Jan van Hoeve
Tepper School of Business
Carnegie Mellon University
Outline

• Motivation and background
• Partial MDD filtering
• Hardness of complete MDD filtering
• Experimental results
• Conclusions
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

\[ x_1 \in \{1,2\}, \; x_2 \in \{1,2,3\}, \; x_3 \in \{2,3\} \]

\[ x_1 < x_2 \quad x_2 \in \{2,3\} \]

\[ \text{alldifferent}(x_1,x_2,x_3) \quad x_1 \in \{1\} \]
Observations:

- Communication between constraints only via variable domains.
- Information can only be expressed as a domain change.
- Other (structural) information that may be learned from a constraint is lost: it must be projected onto variable domains.
- Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation).

This drawback can be addressed by communicating more expressive information, using MDDs [Andersen et al. 2007].

- Explicit representation of more refined potential solution space.
- Limited-width defines relaxation MDD.
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]
- Sequential scheduling constraints [Hoda et al., 2010] [Cire & v.H., 2011]
- `Sequence constraints` (combination of `Amongs`) [v.H., 2011]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]
Sequence Constraint

Employee must work at most 7 days every 9 consecutive days

<table>
<thead>
<tr>
<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>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
<td>$x_4$</td>
<td>$x_5$</td>
<td>$x_6$</td>
<td>$x_7$</td>
<td>$x_8$</td>
<td>$x_9$</td>
<td>$x_{10}$</td>
<td>$x_{11}$</td>
<td>$x_{12}$</td>
</tr>
</tbody>
</table>

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

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

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

$\downarrow$

$Among(X, S, l, u)$
MDD Representation for Sequence

**Equivalent to the DFA representation of Sequence for domain propagation**
[v.H. et al., 2006, 2009]

- Size $O(n2^q)$

Exact MDD for $Sequence(X, q=3, S=\{1\}, l=1, u=2)$
**Goal:** Given an arbitrary MDD and a *Sequence* constraint, remove all inconsistent edges from the MDD (i.e., MDD-consistency)

(Assumption: MDD order follows the sequence of variables $X$)

Can this be done in polynomial time?

- The sub-sequence constraints impose a strong structure (i.e., consecutive-ones LP formulation)
- Exact MDD representation is polynomial in $n$ (i.e., just fix $q$)
- There are several efficient *domain* filtering algorithms for *Sequence*, some of which have a dynamic programming flavor
- Several existing domain filtering algorithms only reason on the bounds of the variables, suggesting that intervals may suffice
Sequence Decomposition

- **Sequence**($X, q, S, l, u$) with $X = x_1, x_2, ..., x_n$

- Introduce a ‘cumulative’ variable $y_i$ representing the sum of the first $i$ variables in $X$
  
  $\begin{align*}
  y_0 &= 0 \\
  y_i &= y_{i-1} + (x_i \in S) \quad \text{for } i=1..n
  \end{align*}$

- Then the sub-constraint on $[x_{i+1}, ..., x_{i+q}]$ is equivalent to
  
  $\begin{align*}
  l &\leq y_{i+q} - y_i \\
  y_{i+q} - y_i &\leq u \quad \text{for } i = 0..n-q
  \end{align*}$

- [Brand et al., 2007] show that bounds reasoning on this decomposition suffices to reach domain consistency for **Sequence** (in poly-time)
MDD filtering from decomposition

Approach
• The auxiliary variables $y_i$ can be naturally represented at the nodes of the MDD
• We can now actively filter this node information (not only the edges)

Sequence($X, q=3, S=\{1\}, l=1, u=2$)
MDD filtering from decomposition

Sequence($X$, $q=3$, $S=\{1\}$, $l=1$, $u=2$)

$$y_i = y_{i-1} + x_i$$

$$1 \leq y_3 - y_0 \leq 2$$

$$1 \leq y_4 - y_1 \leq 2$$

$$1 \leq y_5 - y_2 \leq 2$$
MDD filtering from decomposition

Sequence($X$, $q=3$, $S=\{1\}$, $l=1$, $u=2$)

\[ y_i = y_{i-1} + x_i \]
\[ 1 \leq y_3 - y_0 \leq 2 \]
\[ 1 \leq y_4 - y_1 \leq 2 \]
\[ 1 \leq y_5 - y_2 \leq 2 \]
This procedure does not guarantee MDD consistency.

Sequence($X$, $q=3$, $S=\{1\}$, $l=1$, $u=2$)

\[ y_i = y_{i-1} + x_i \]

\[ 1 \leq y_3 - y_0 \leq 2 \]
\[ 1 \leq y_4 - y_1 \leq 2 \]
\[ 1 \leq y_5 - y_2 \leq 2 \]
Hardness of MDD Consistency
**Result**

**Theorem:** Establishing MDD consistency for *Sequence* on an arbitrary MDD is NP-hard

**Proof structure:**

- Given 3-SAT problem (NP-complete)
- We will construct a polynomial-size MDD such that a particular *Sequence* constraint will have a solution in the MDD if and only if the 3-SAT instance is satisfiable

- Example 3-SAT problem
  
  \[ c_1 = (x_1 \lor \overline{x}_3 \lor x_4) \]
  
  \[ c_2 = (x_2 \lor x_3 \lor \overline{x}_4) \]
Single clause representation

Each path from root to terminal corresponds to a satisfying assignment for this clause.

\[ c_1 = (x_1 \lor \overline{x}_3 \lor x_4) \]

\[ c_2 = (x_2 \lor x_3 \lor \overline{x}_4) \]
Group clauses together

- Literal $x_j$ in clause $c_i$ represented by variable $y_{ij}$
- MDD size $O(6(2mn+1))$
- How to ensure that a variable takes the same value in each clause?
Impose Sequence Constraint

Sequence($Y, q=2n, S=\{1\}, l=n, u=n$)

- Start from a positive literal: sub-sequence always contains $n$ times the value 1 (namely, for each variable it contains both literals)
- Start from a negative literal: the corresponding positive literal in the next clause must take the opposite value (all other variables sum up to $n-1$)
- Therefore, variables take the same value in each clause
- Solution to Sequence in this MDD is equivalent to 3-SAT solution
Preliminary Experimental Results
Experimental Setup

• Decomposition-based filtering algorithm
  – implemented in MDD solver of [Hoda, PhD 2010]

• Evaluation
  – compare Sequence MDD filter with Among MDD filter
    (the Among MDD filter is also implemented in [Hoda, PhD 2010])
  – compare Sequence MDD filter with Sequence domain filter
    (the domain filter is based on the same decomposition)

• All methods use the same search strategy
  – variable selection: smallest domain first
  – value selection: lexicographic ordering
MDD Sequence versus Among

• Randomly generated instances
  – 50 variables
  – two Sequence constraints
  – $q = 14^*$
  – $u - l = 1$ (select l uniform-randomly from $[1,n-1]$)
  – 100 instances

• Vary maximum width of MDD
  – widths 1, 4, 8

* For $q \leq 7$ Among and Sequence performed similarly
MDD Sequence versus Among
MDD Filter versus Domain Filter

• Shift scheduling problem for \( n = 40, 50, 60, 70, 80 \) days
• Shifts: day (D), evening (E), night (N), off (O)
• Problem type P-I
  – work at least 22 day or evening shifts every 30 days
    \( \text{Sequence}(X, q=30, S=\{D, E\}, l=22, u=30) \)
  – have between 1 and 4 days off every 7 consecutive days
    \( \text{Sequence}(X, q=7, S=\{O\}, l=1, u=4) \)
• Problem type P-II
  – \( \text{Sequence}(X, q=30, S=\{D, E\}, l=23, u=30) \)
  – \( \text{Sequence}(X, q=5, S=\{N\}, l=1, u=2) \)
## MDD Filter versus Domain Filter

<table>
<thead>
<tr>
<th>instance</th>
<th>domain propagator</th>
<th>MDD width 1</th>
<th>MDD width 4</th>
<th>MDD width 8</th>
<th>MDD width 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BT</td>
<td>CPU</td>
<td>BT</td>
<td>CPU</td>
<td>BT</td>
</tr>
<tr>
<td>P-I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>121,767</td>
<td>4.63</td>
<td>34,108</td>
<td>251.06</td>
<td>75</td>
</tr>
<tr>
<td>50</td>
<td>121,777</td>
<td>5.67</td>
<td>34,108</td>
<td>487.25</td>
<td>75</td>
</tr>
<tr>
<td>60</td>
<td>121,782</td>
<td>6.51</td>
<td>34,108</td>
<td>796.21</td>
<td>75</td>
</tr>
<tr>
<td>70</td>
<td>121,787</td>
<td>6.99</td>
<td>34,108</td>
<td>1,110.88</td>
<td>75</td>
</tr>
<tr>
<td>80</td>
<td>121,792</td>
<td>7.48</td>
<td>34,108</td>
<td>1,492.33</td>
<td>75</td>
</tr>
<tr>
<td>P-II</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>116,548</td>
<td>3.52</td>
<td>233,096</td>
<td>1,492.93</td>
<td>71</td>
</tr>
<tr>
<td>50</td>
<td>116,548</td>
<td>3.98</td>
<td>-</td>
<td>&gt;1,600</td>
<td>69</td>
</tr>
<tr>
<td>60</td>
<td>116,548</td>
<td>4.56</td>
<td>-</td>
<td>&gt;1,600</td>
<td>65</td>
</tr>
<tr>
<td>70</td>
<td>116,548</td>
<td>4.95</td>
<td>-</td>
<td>&gt;1,600</td>
<td>67</td>
</tr>
<tr>
<td>80</td>
<td>116,548</td>
<td>5.08</td>
<td>-</td>
<td>&gt;1,600</td>
<td>73</td>
</tr>
</tbody>
</table>
Conclusions

• Complete MDD filtering for Sequence is NP-hard

• Partial MDD filtering based on cumulative decomposition can be quite effective
  – represent auxiliary variables at nodes
  – actively filter node information

• Preliminary experimental results are promising

• Future/current work: better implementation, in ILOG CP Optimizer