MDD Filtering for Sequence Constraints

Andre Cire and Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

Outline

- Motivation and background
- MDD filtering for *Sequence*
- 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

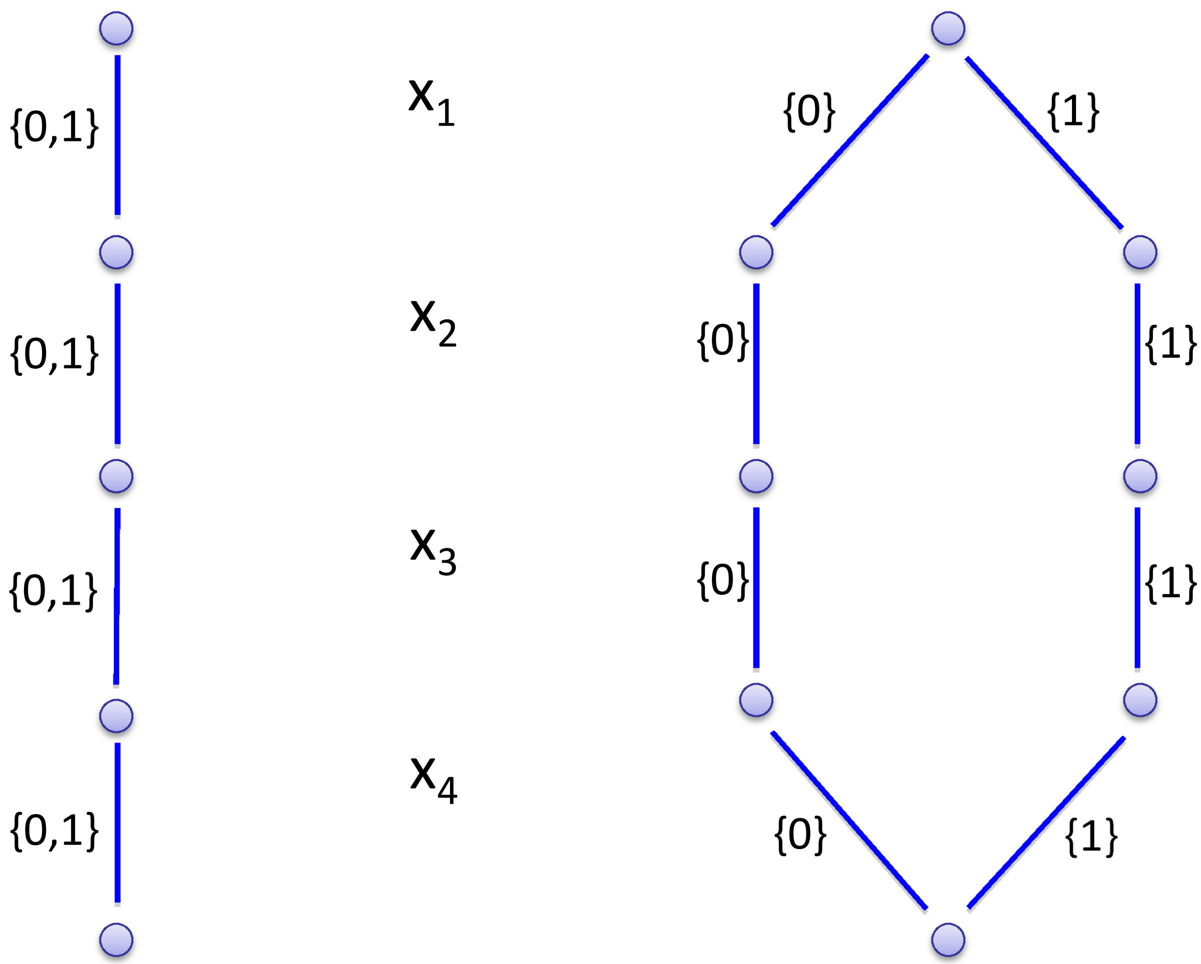
$x_1 \in \{1,2\}$, $x_2 \in \{1,2,3\}$, $x_3 \in \{2,3\}$

$x_1 < x_2$ → $x_2 \in \{2,3\}$

alldifferent(x_1,x_2,x_3) → $x_1 \in \{1\}$

Illustrative Example

AllEqual(x_1, x_2, x_3, x_4), all x_i binary



domain representation, size 2^4

MDD representation, size 2

Drawback of domain propagation

- All structural relationships among variables are projected 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 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]
- Disjunctive scheduling constraints [Hoda et al., 2010] [Cire & v.H., 2012]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]
- *Sequence* constraints (combination of *Amongs*)

Sequence Constraint

Employee must work at most 7 days every 9 consecutive days

sun	mon	tue	wed	thu	fri	sat	sun	mon	tue	wed	thu
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}

$0 \le x_1 + x_2 + \dots + x_9 \le 7$

$0 \le x_2 + x_3 + \dots + x_{10} \le 7$

$0 \le x_3 + x_4 + \dots + x_{11} \le 7$

$0 \le x_4 + x_5 + \dots + x_{12} \le 7$

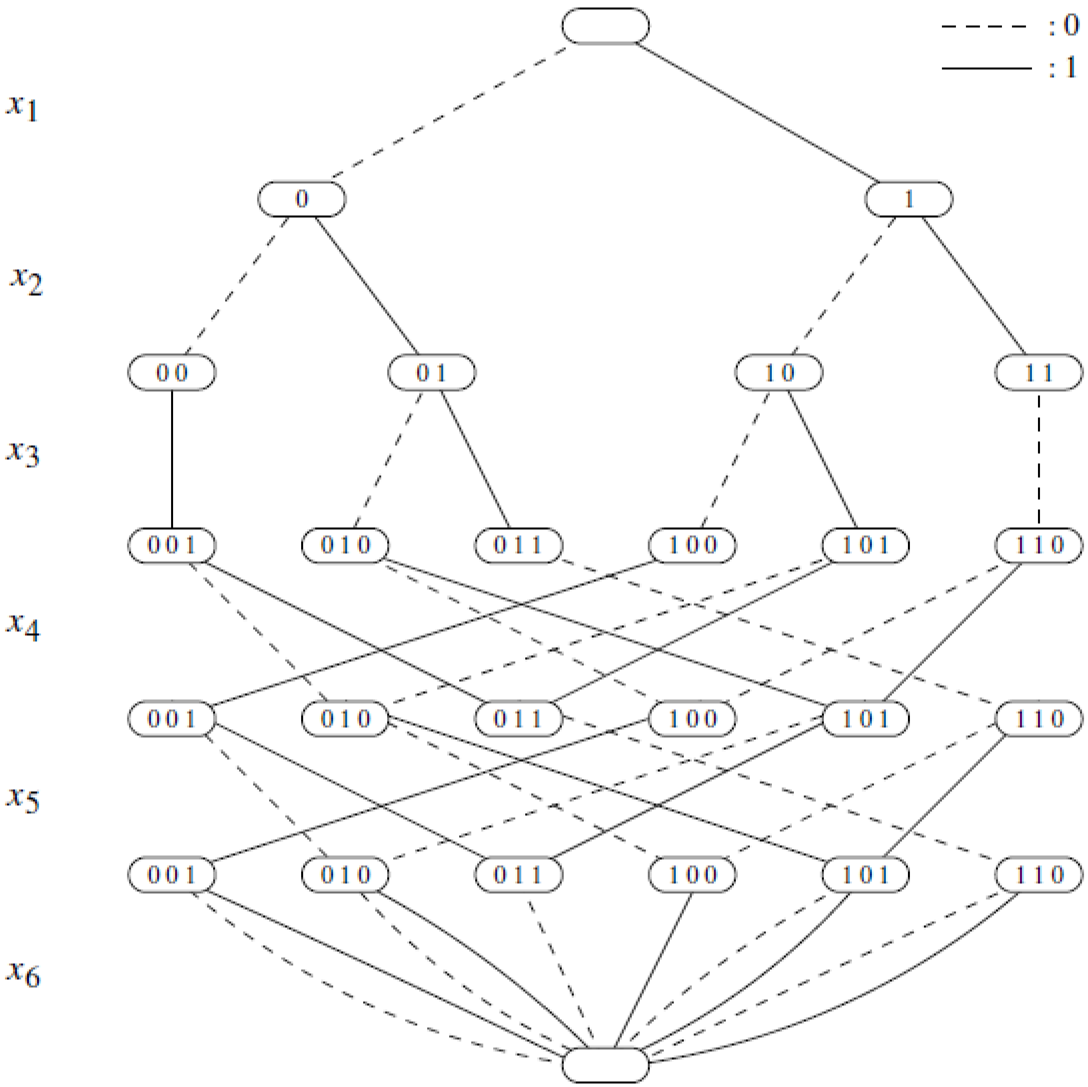
=: *Sequence*($[x_1, x_2, \dots, x_{12}]$, q=9, S={1}, l=0, u=7)

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

↓

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)

MDD Filtering for Sequence

Goal: Given an arbitrary MDD and a *Sequence* constraint, remove *all* inconsistent edges from the MDD (i.e., MDD-consistency)

Can this be done in polynomial time?

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

(even if the MDD order follows the sequence of variables *X*)

Proof: Reduction from 3-SAT

Next goal: Develop a *partial* filtering algorithm, that does not necessarily achieve MDD consistency

Sequence Decomposition

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- *Sequence*(X, q, S, l, u) with $X = x_1, x_2, \ldots, x_n$
- Introduce a 'cumulative' variable y_i representing the sum of the first i variables in X

  $y_0 = 0$

  $y_i = y_{i-1} + (x_i \in S)$ for i=1..n

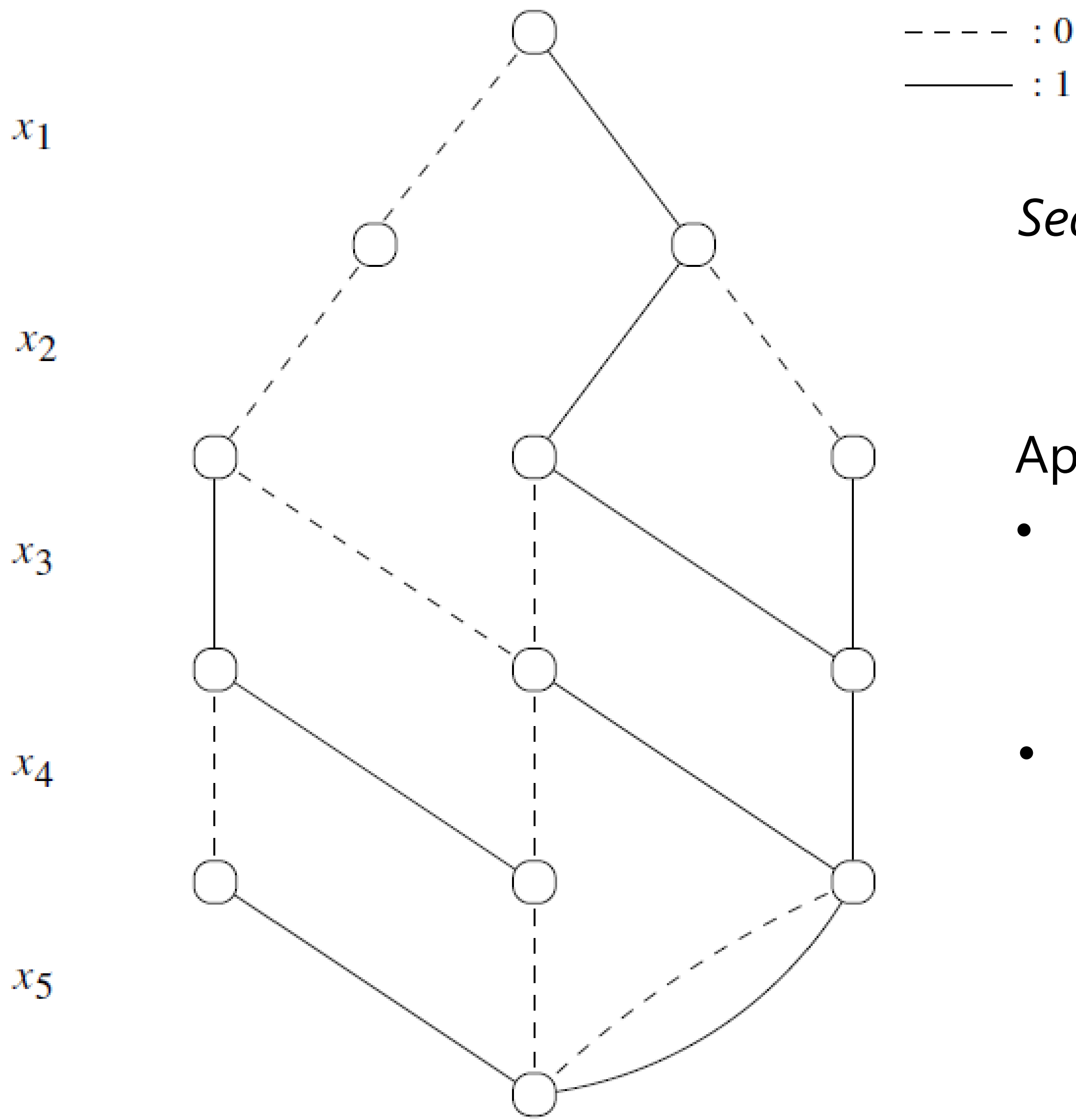
- Then the sub-constraint on $[x_{i+1}, \ldots, x_{i+q}]$ is equivalent to

  $l \leq y_{i+q} - y_i$

  $y_{i+q} - y_i \leq u$ for i = 0..n-q

- [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

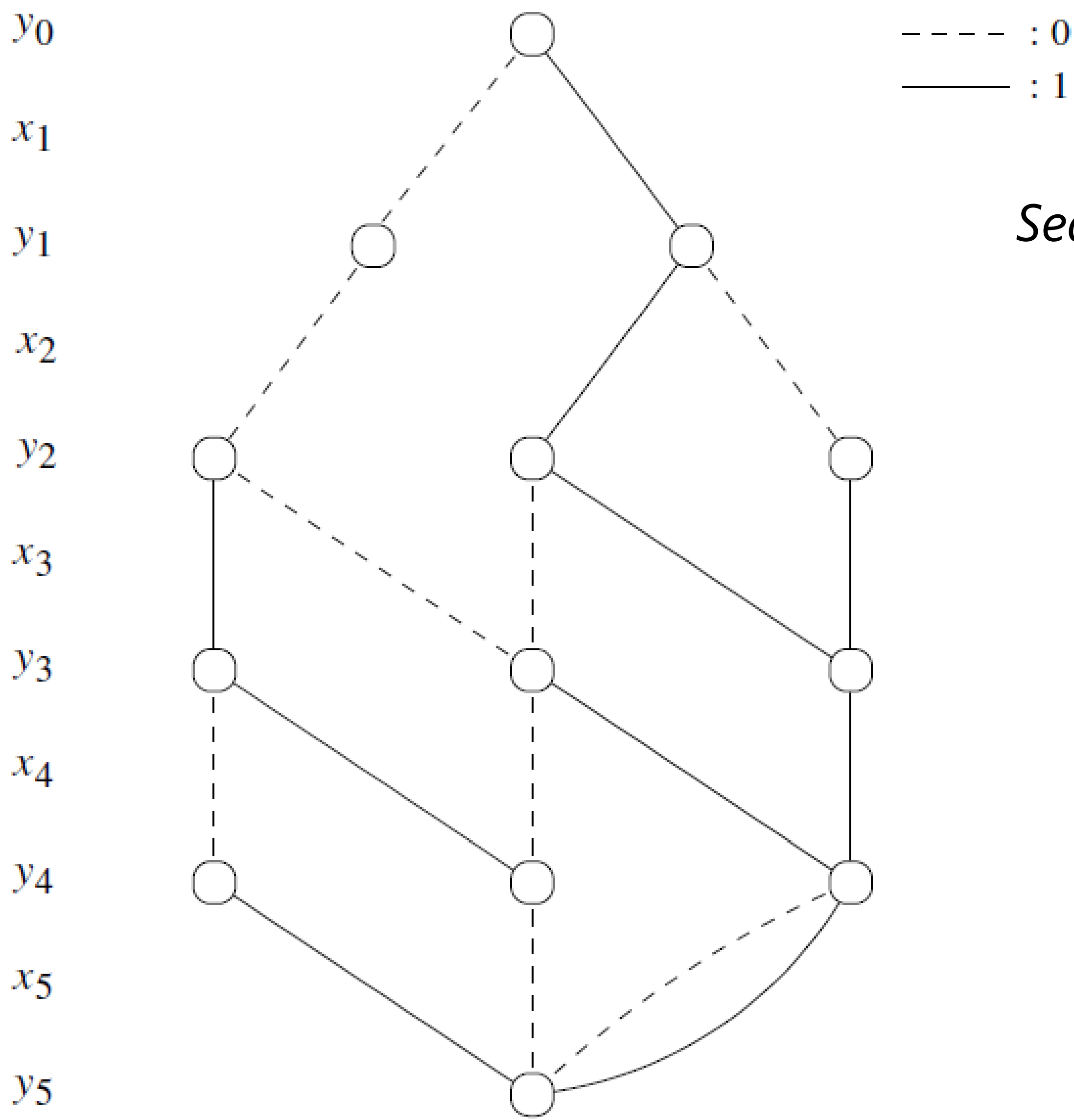


Sequence(*X*, *q*=3, *S*={1}, *l*=1, *u*=2)

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)

MDD filtering from decomposition



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

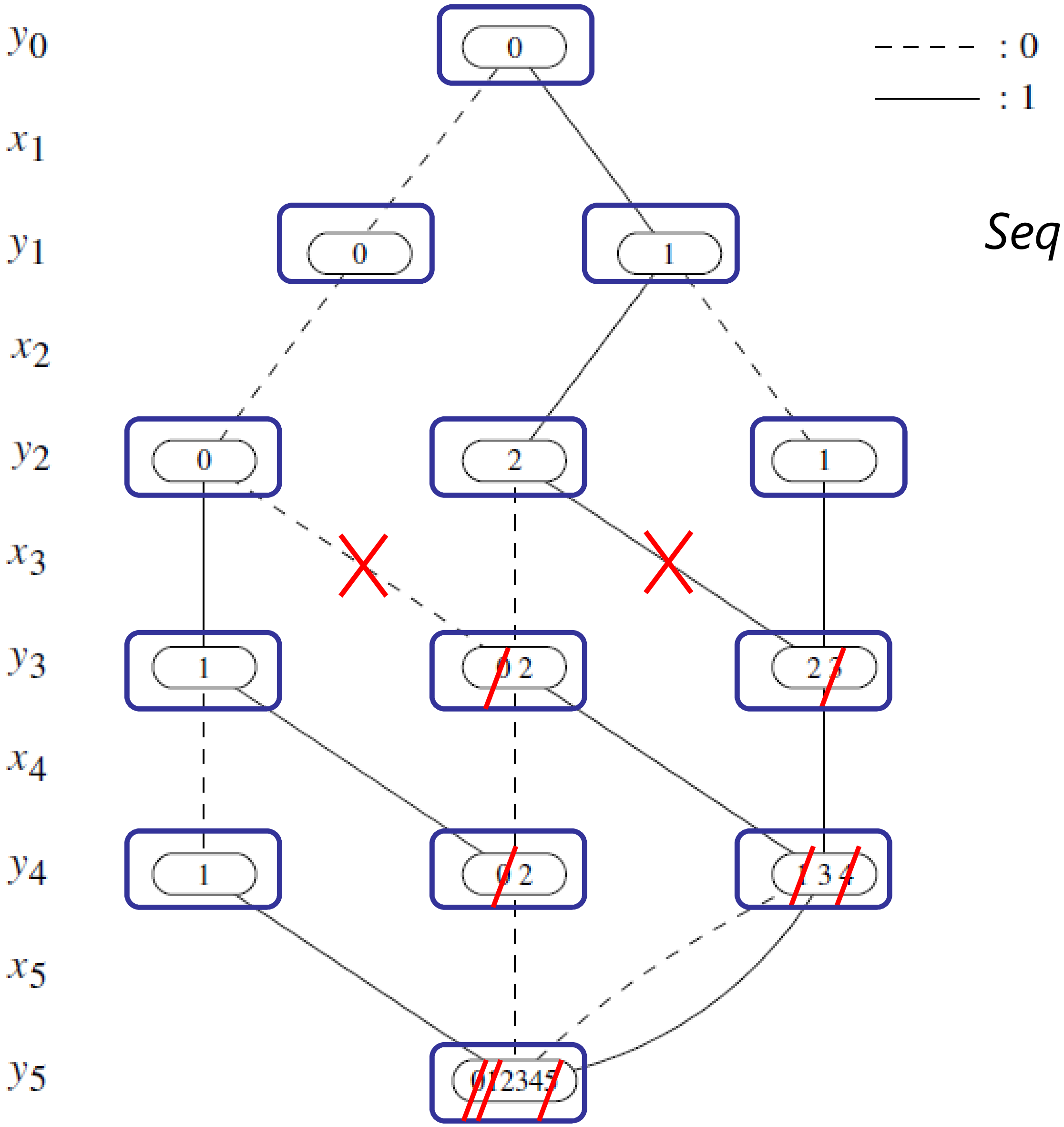
$y_i = y_{i-1} + x_i$

$1 \leq y_3 - y_0 \leq 2$

$1 \leq y_4 - y_1 \leq 2$

$1 \leq y_5 - y_2 \leq 2$

MDD filtering from decomposition



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

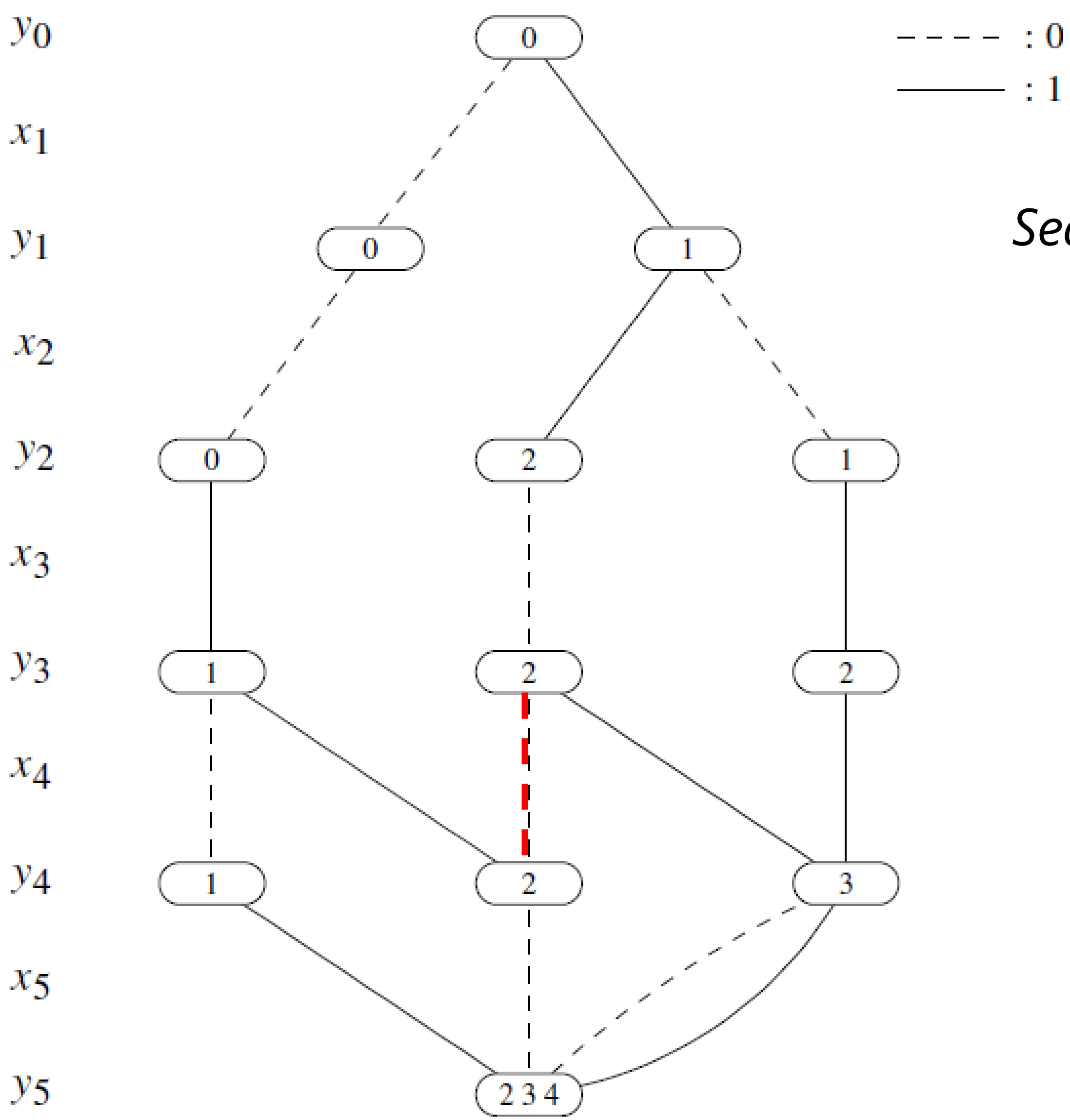
$y_i = y_{i\text{-}1} + x_i$

$1 \leq y_3 - y_0 \leq 2$

$1 \leq y_4 - y_1 \leq 2$

$1 \leq y_5 - y_2 \leq 2$

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

Analysis of Algorithm

- Initial population of node domains (y variables)
  - linear in MDD size
- Analysis of each state in layer k
  - maintain list of ancestors from layer k-q
  - direct implementation gives $O(qW^2)$ operations per state (W is maximum width)
  - need only maintain min and max value over previous q layers: $O(Wq)$
- One top-down and one bottom-up pass

Experimental Results

Experimental Setup

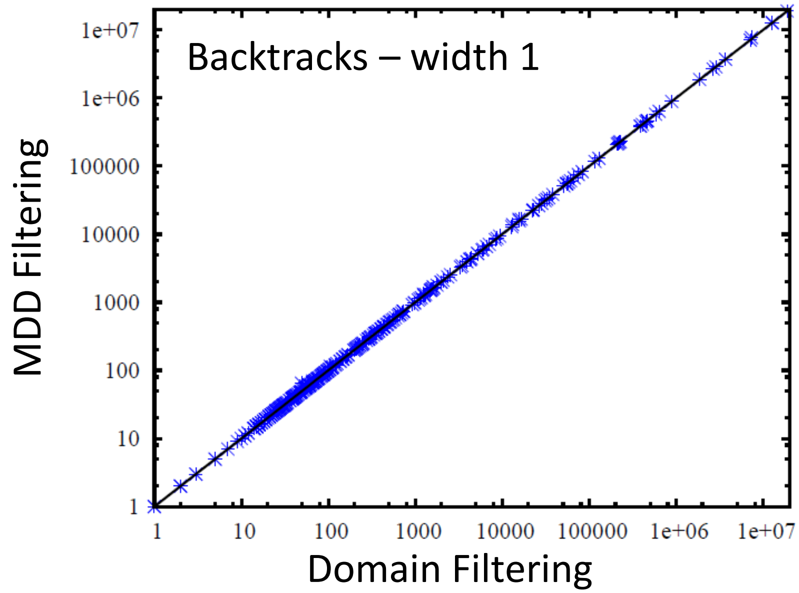
- Decomposition-based filtering algorithm
  - Implemented as global constraint in IBM ILOG CPLEX/CP Optimizer 12.3
- Evaluation
  - Compare MDD filtering with Domain filtering
  - Domain filter based on the same decomposition (achieved domain consistency for almost all our instances)
  - Random instances and structured shift scheduling instances
- All methods apply the same fixed search strategy
  - lexicographic variable and value ordering
  - find first solution or prove that none exists

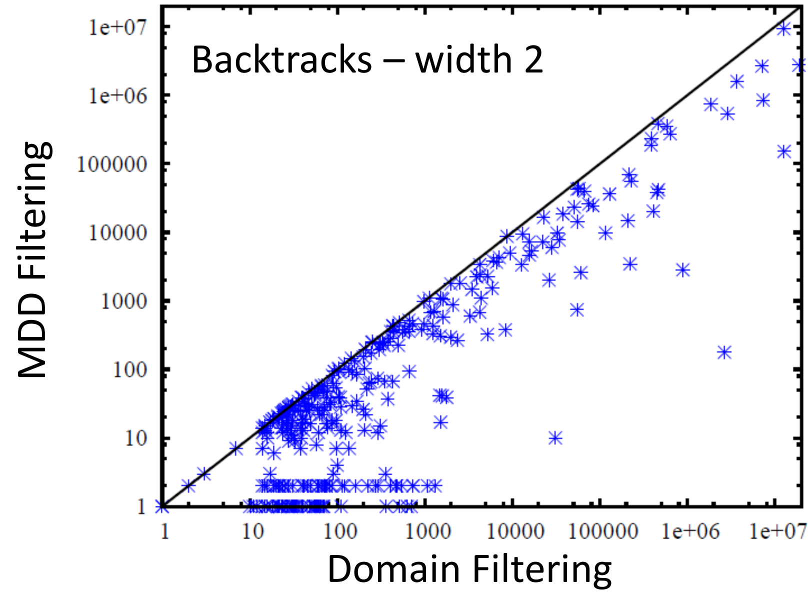
Random instances

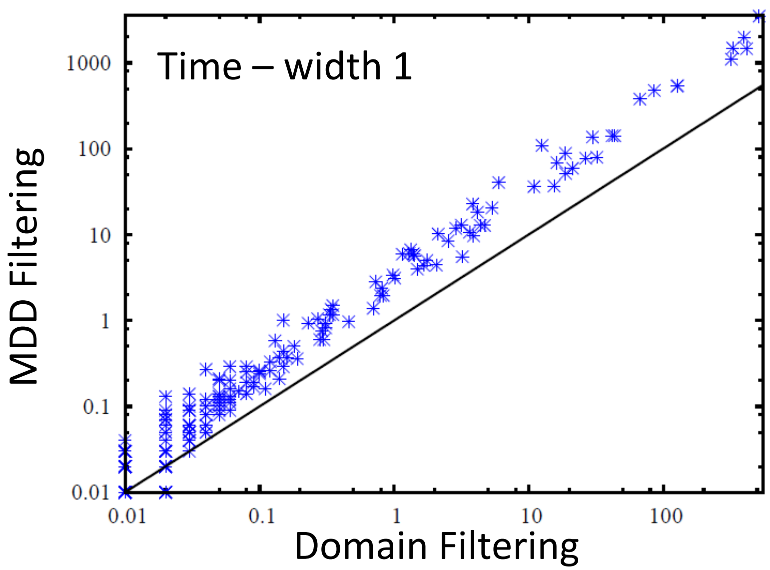
- Randomly generated instances
  - n=20-48 variables
  - domain size between 10 and 30
  - 1, 2, 5, 7, or 10 *Sequence* constraints
  - q random from [2..n/2]
  - $u - l$ random from 0 to q-1
  - 360 instances

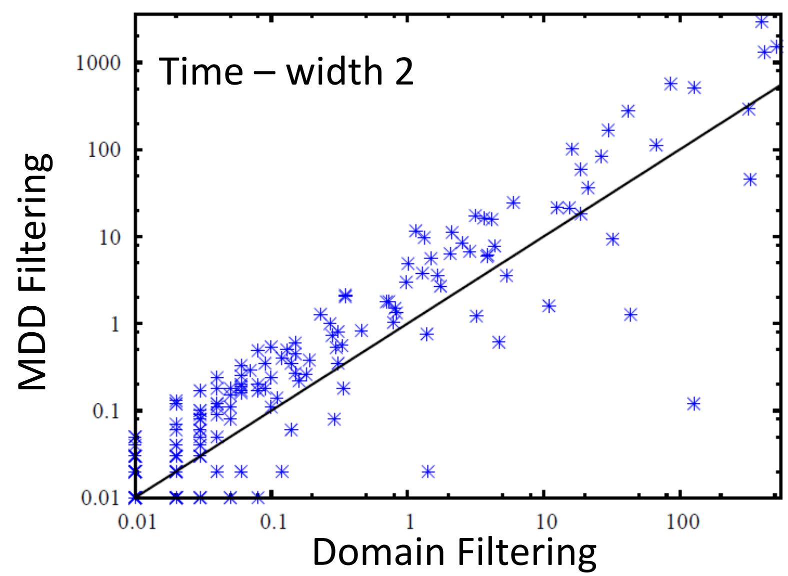
- Vary maximum width of MDD
  - widths 1 up to 32

Random instances results

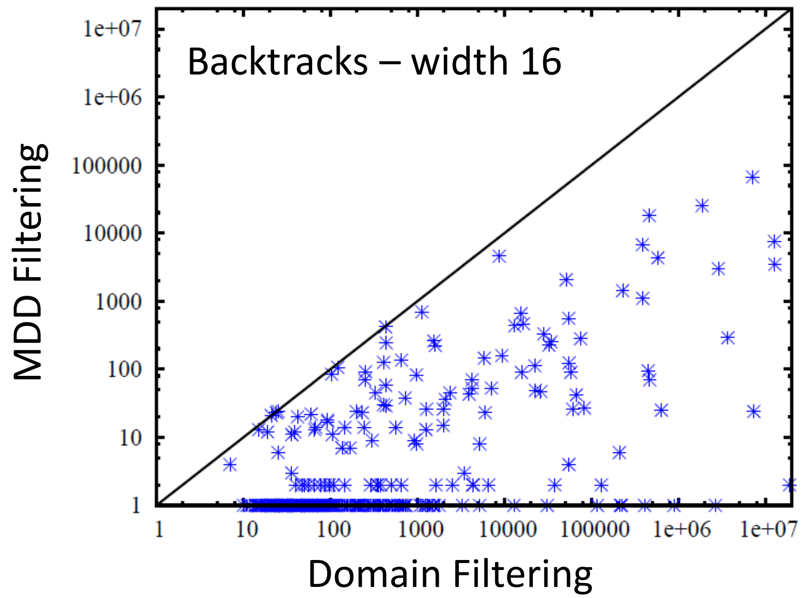


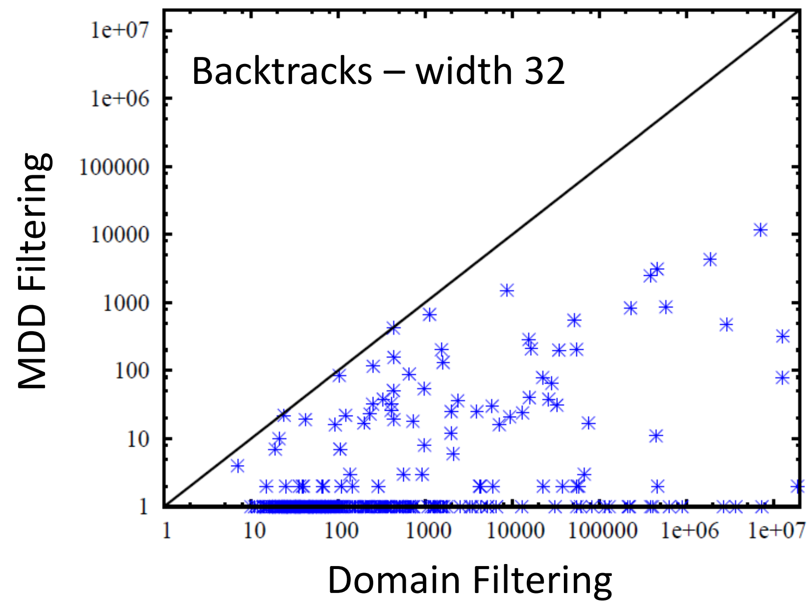


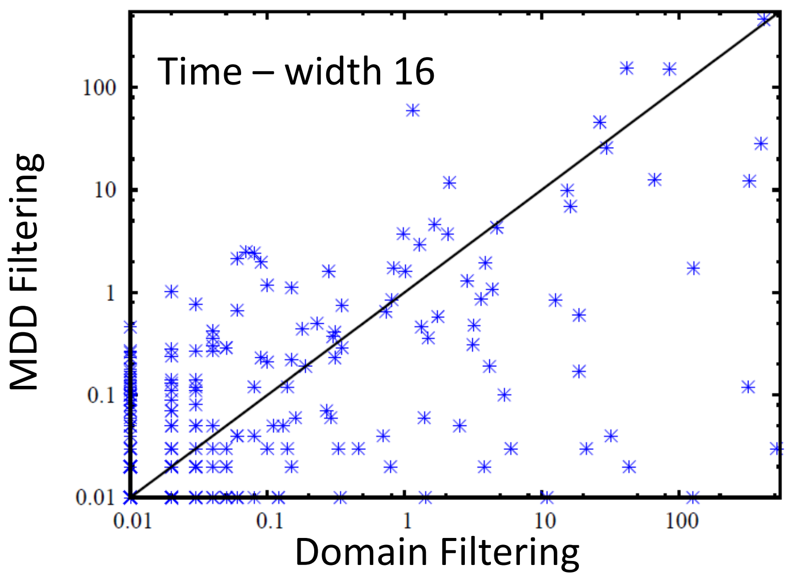


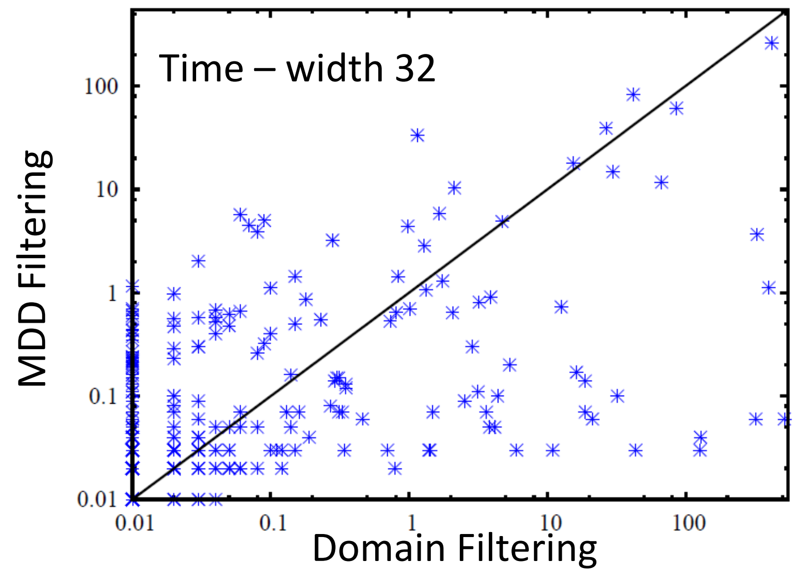


Random instances results (cont'd)

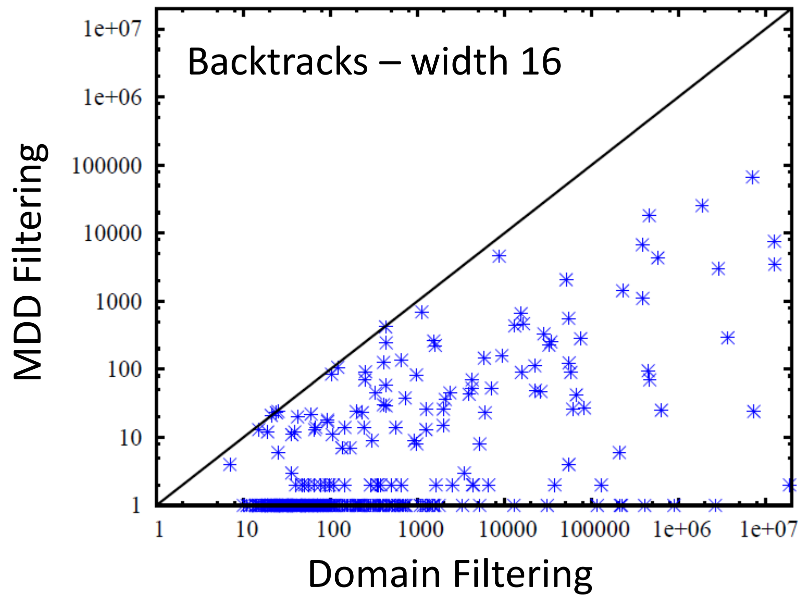


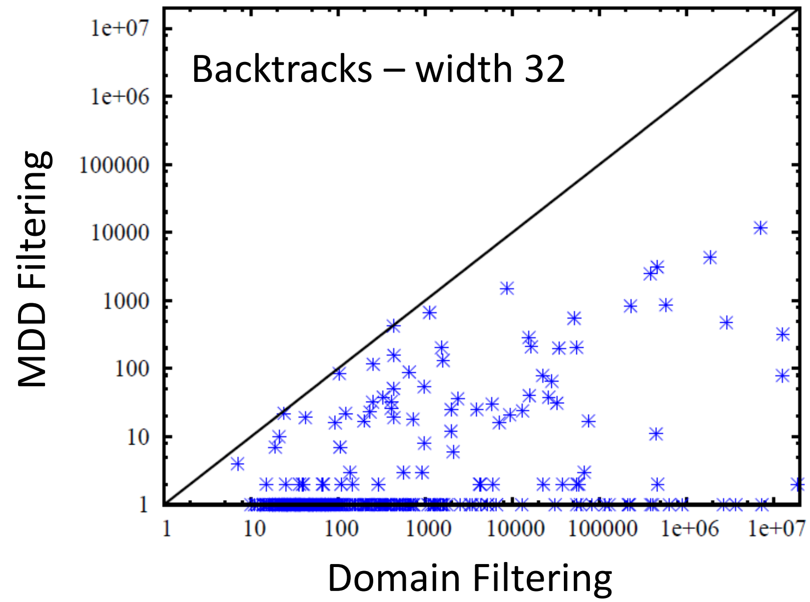


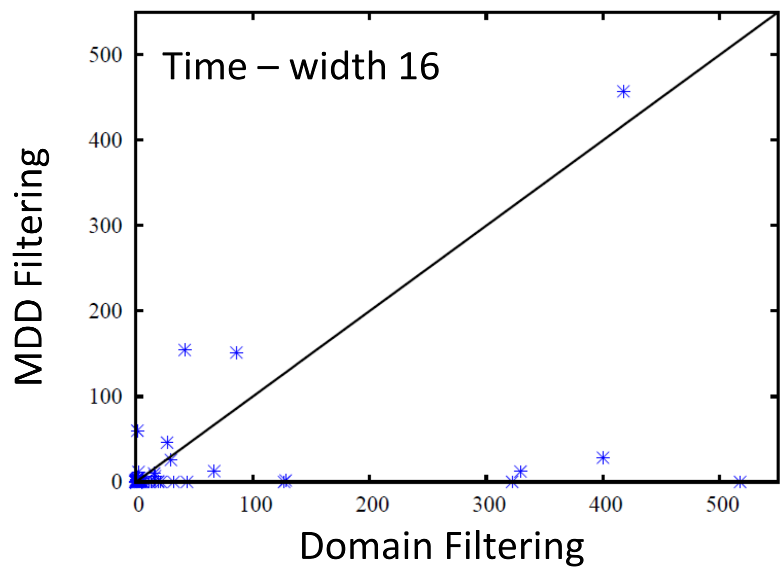


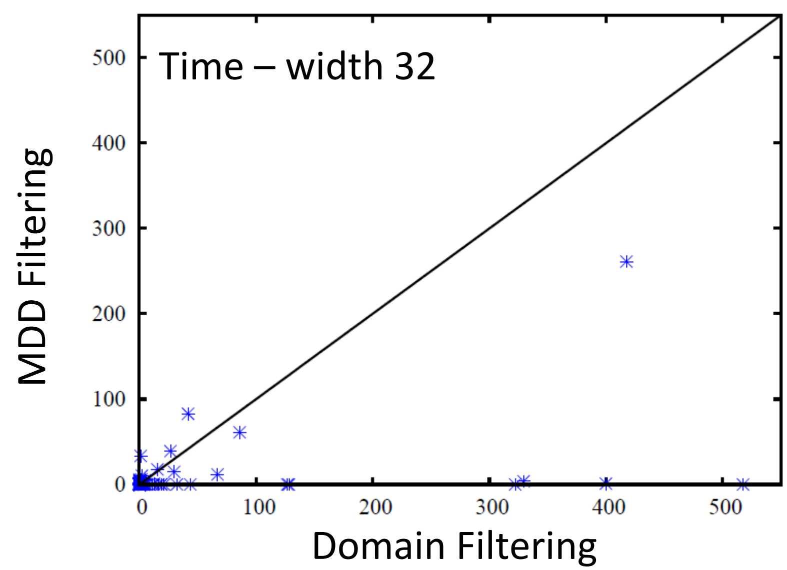


Random instances results (cont'd)









Shift scheduling instances

- 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

    Sequence(X, q=30, S= {D, E}, l=22, u=30)
  - have between 1 and 4 days off every 7 consecutive days

    Sequence(X, q=7, S={O}, l=1, u=4)
- Problem type P-II
  - *Sequence*(X, q=30, S={D, E}, l=23, u=30)
  - *Sequence*(X, q=5, S={N}, l=1, u=2)

MDD Filter versus Domain Filter

Instance		Domain filtering		MDD - width 1		MDD - width 2		MDD - width 8	
	n	*backtracks*	*time*	*backtracks*	*time*	*backtracks*	*time*	*backtracks*	*time*
Type P-I	40	17,054	0.36	17,054	0.61	1,213	0.07	0	0.00
	50	17,054	0.42	17,054	0.75	1,213	0.09	0	0.00
	60	17,054	0.54	17,054	0.90	1,213	0.11	0	0.01
	70	17,054	0.58	17,054	1.04	1,213	0.12	0	0.01
	80	17,054	0.66	17,054	1.26	1,213	0.15	0	0.01
Type P-II	40	126,406	2.00	126,406	4.66	852	0.08	0	0.00
	50	126,406	2.36	126,406	5.90	852	0.09	0	0.00
	60	126,406	2.86	126,406	7.43	852	0.11	0	0.00
	70	126,406	3.04	126,406	8.38	852	0.13	0	0.01
	80	126,406	3.48	126,406	9.46	852	0.15	0	0.01

Summary

- We studied MDD propagation for *Sequence* constraints
- 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
- Large savings possible w.r.t. Domain propagation
- MDD propagation can be a powerful mechanism for solving Constraint Programming problems