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

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

$x_1 < x_2$ $\quad$ $x_2 \in \{2,3\}$

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

Drawback of domain propagation

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

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}

$0 \leq x_1 + x_2 + \ldots + x_9 \leq 7$

$0 \leq x_2 + x_3 + \ldots + x_{10} \leq 7$

$0 \leq x_3 + x_4 + \ldots + x_{11} \leq 7$

$0 \leq x_4 + x_5 + \ldots + x_{12} \leq 7$

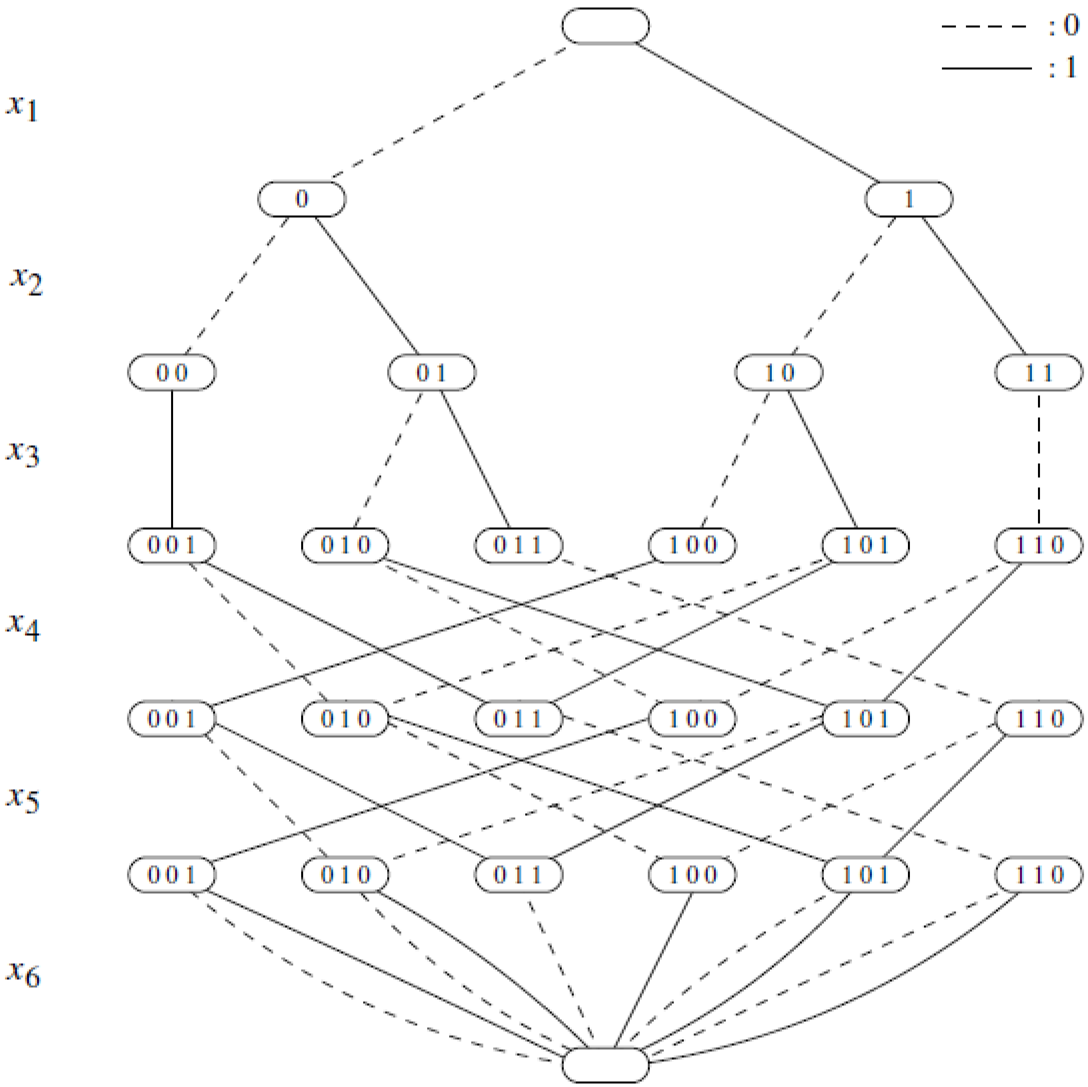
=: *Sequence*($[x_1, x_2, \ldots, x_{12}]$, q=9, S={1}, l=0, u=7)

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

$\downarrow$

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)

MDD Filtering for Sequence

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 $X = x_1, x_2, \ldots, x_n$
- Introduce a ‘cumulative’ variable y_i representing the sum of the first i variables in X

$$y_0 = 0$$

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

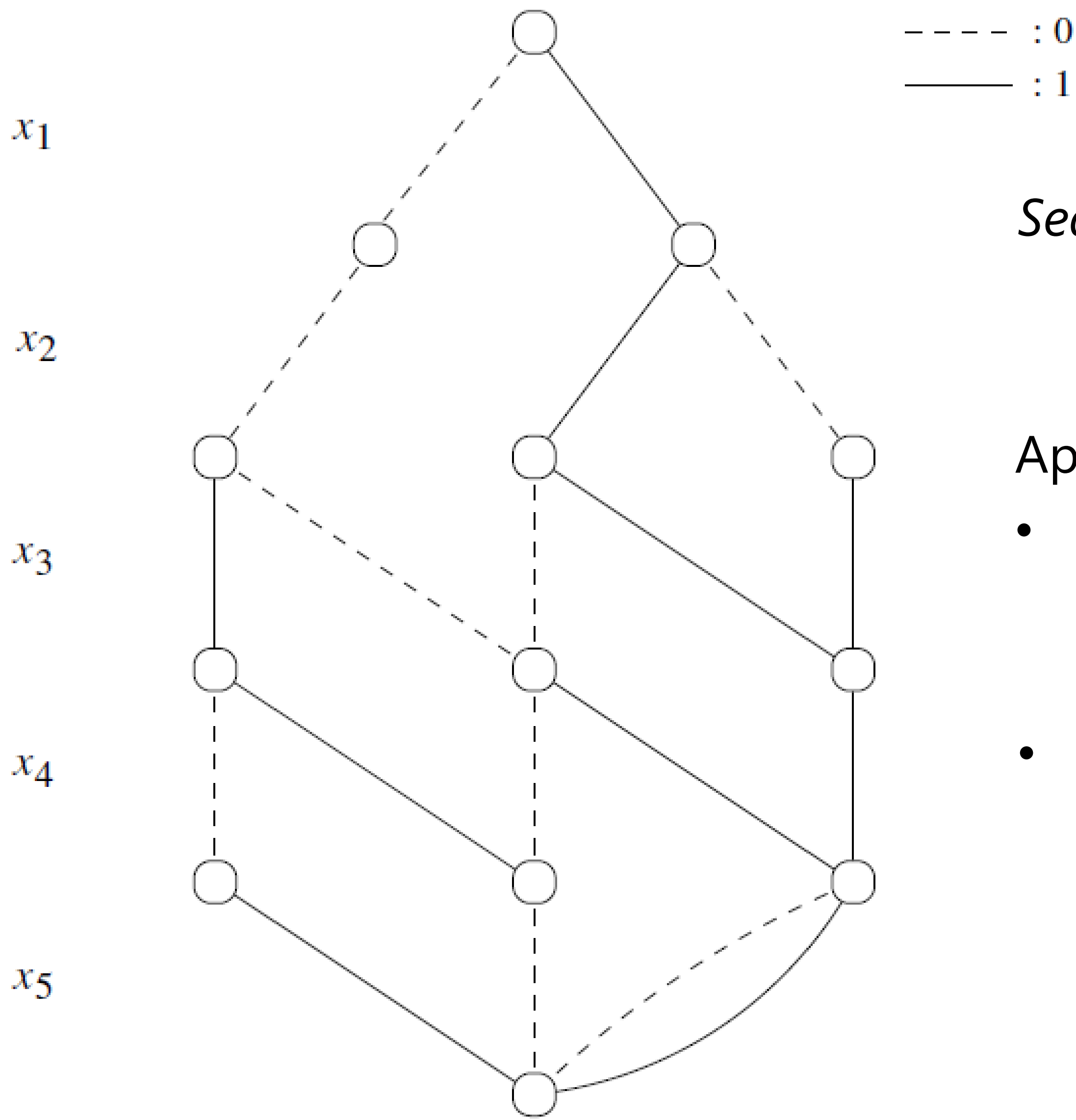
- Then the sub-constraint on $[x_{i+1},\ldots, x_{i+q}]$ is equivalent to

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

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

- [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

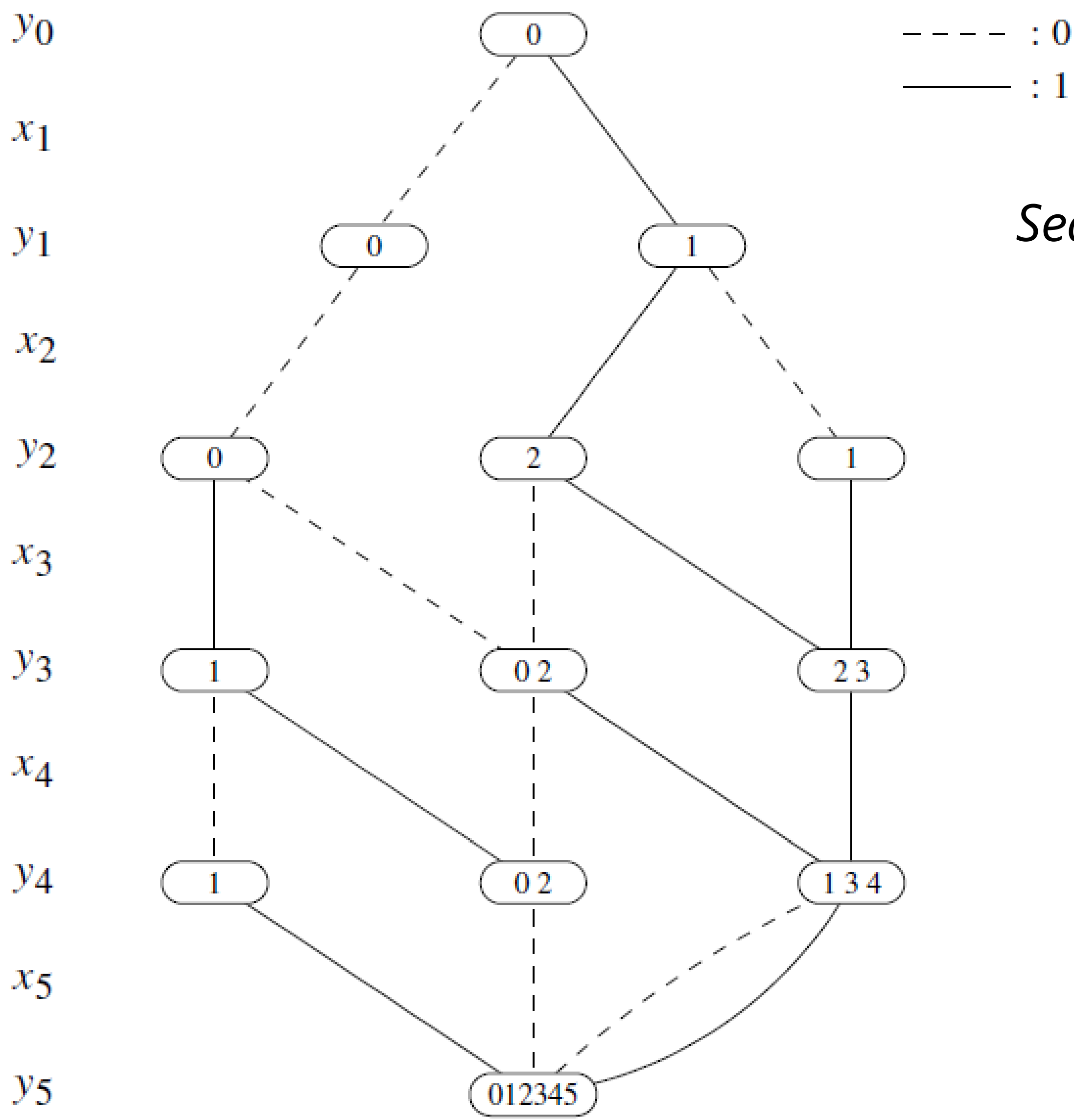


Sequence(X, q=3, S={1}, l=1, u=2)

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)

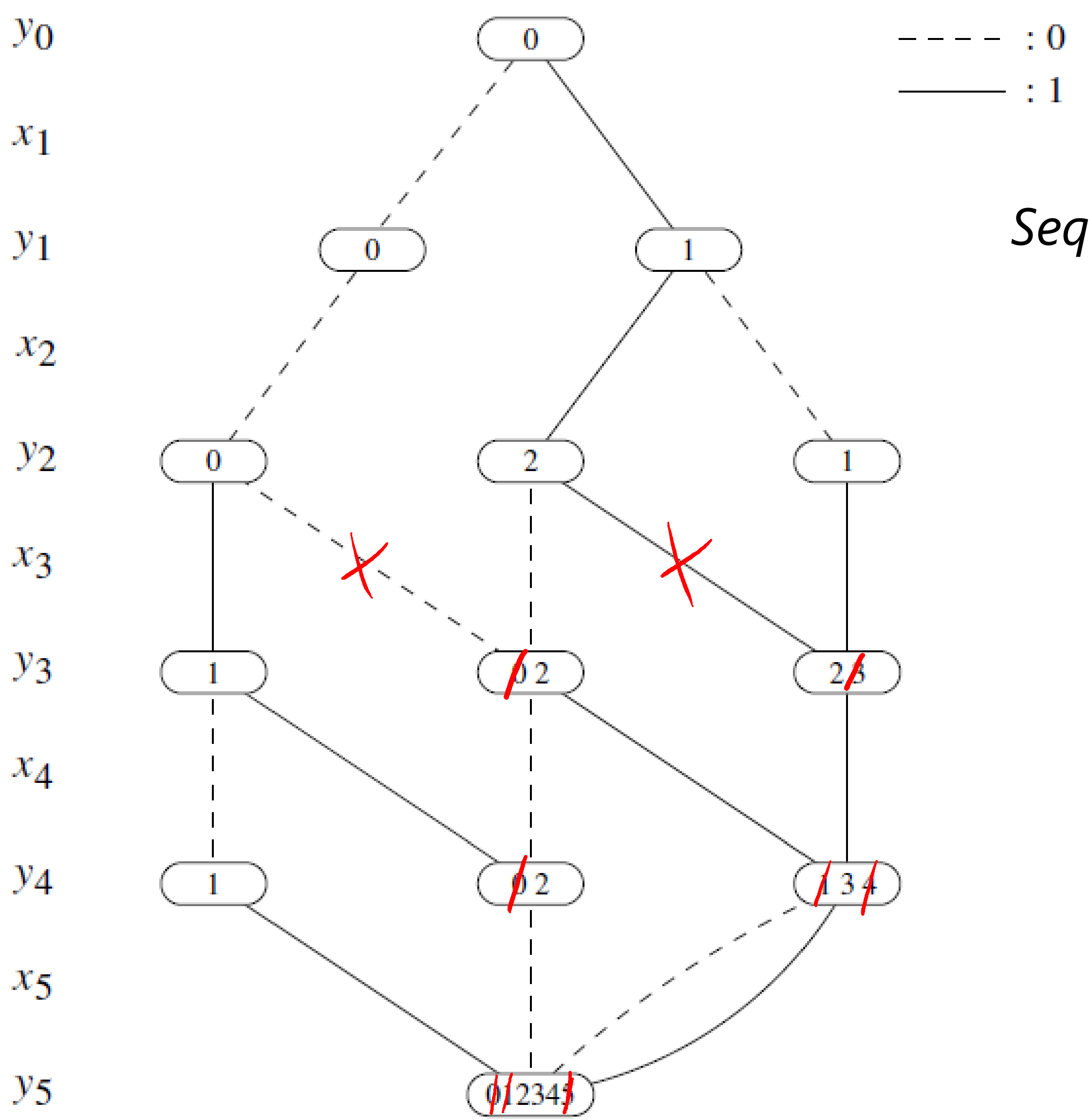
$y_i = y_{i-1} + x_i$

$1 \leq y_3 - y_0 \leq 2$

$1 \leq y_4 - y_1 \leq 2$

$1 \leq y_5 - y_2 \leq 2$

MDD filtering from decomposition

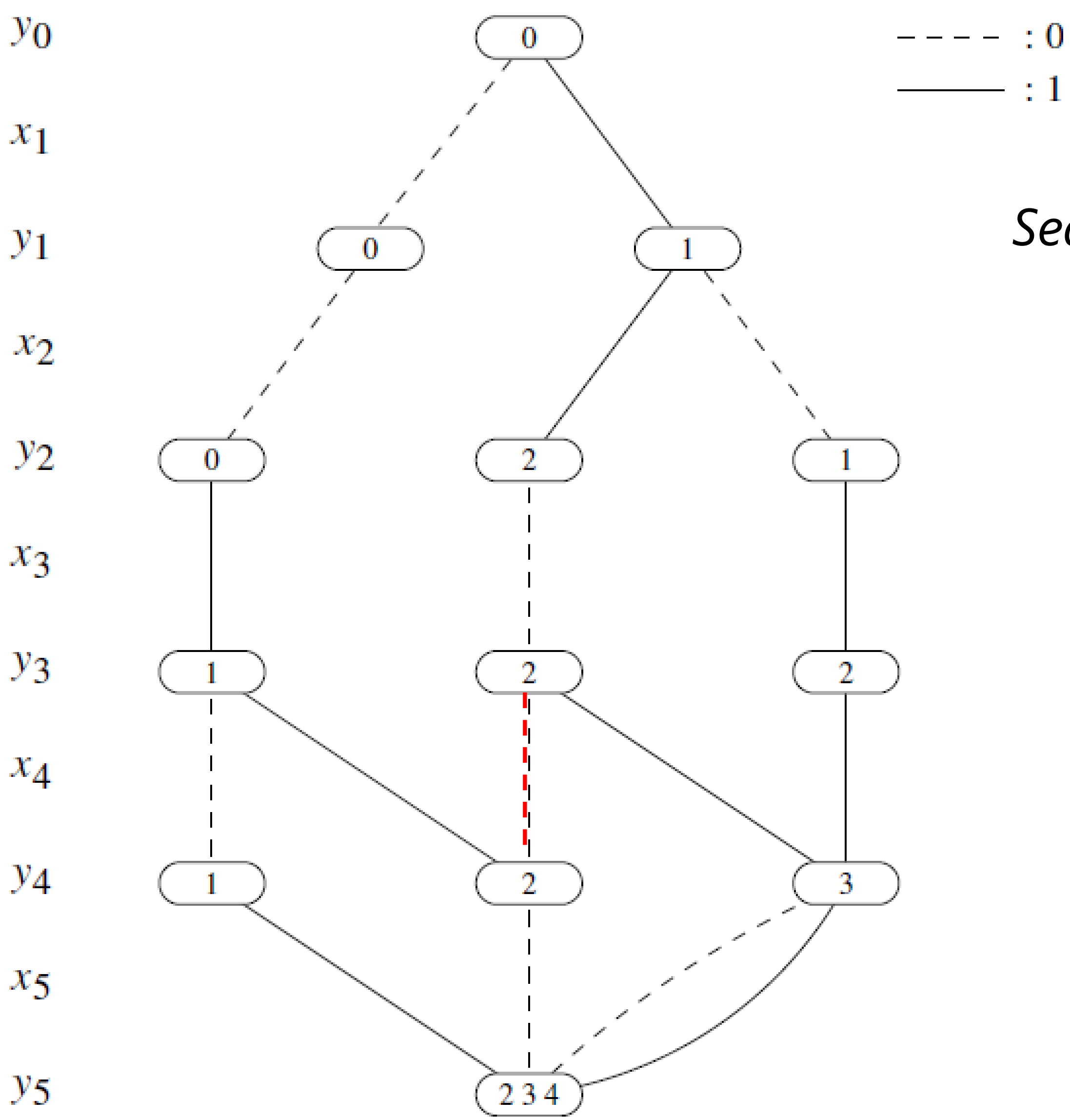


Sequence(X, q=3, S={1}, l=1, u=2)

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

$$1 \le y_3 - y_0 \le 2$$
$$1 \le y_4 - y_1 \le 2$$
$$1 \le y_5 - y_2 \le 2$$

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

Hardness of MDD Consistency

Result

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

Proof structure:

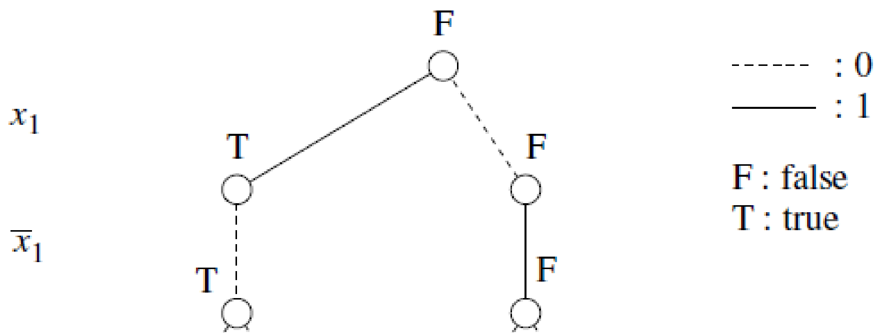
- Given 3-SAT problem (NP-complete)
- We will construct a polynomial-size MDD such that a particular *Sequence* constraint will have a solution in the MDD if and only if the 3-SAT instance is satisfiable

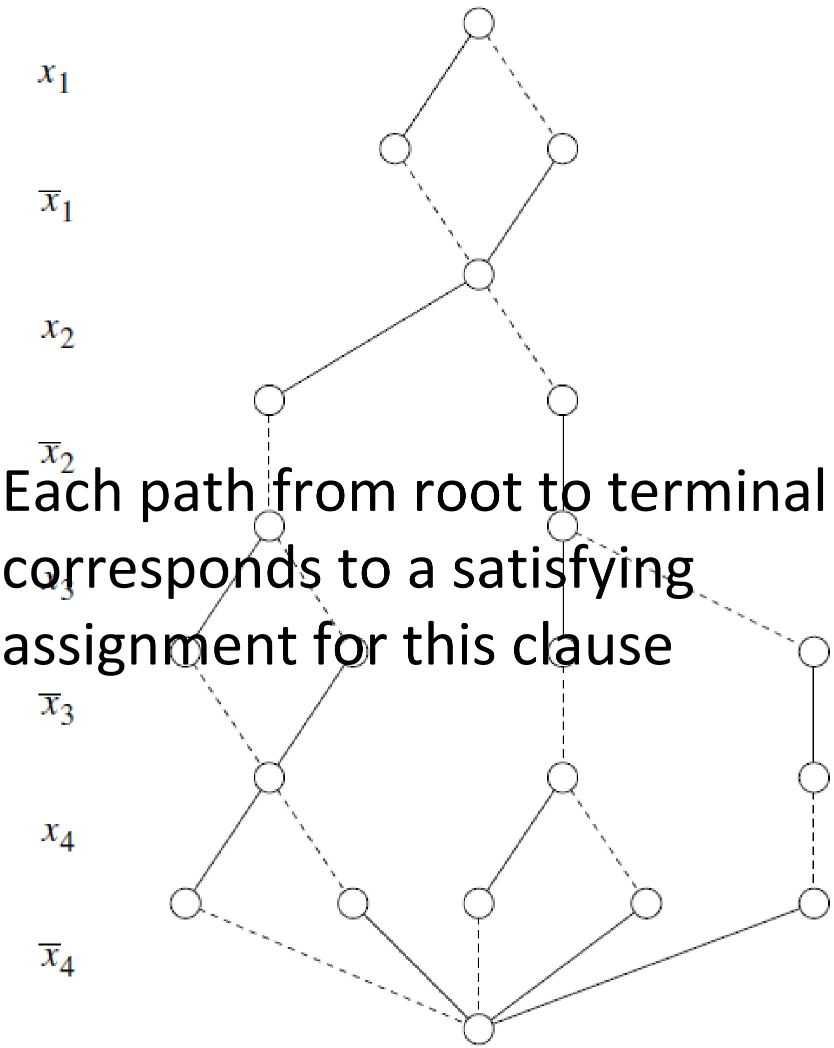
- Example 3-SAT problem

$$c_1 = (x_1 \vee \overline{x}_3 \vee x_4)$$

$$c_2 = (x_2 \vee x_3 \vee \overline{x}_4)$$

Single clause representation



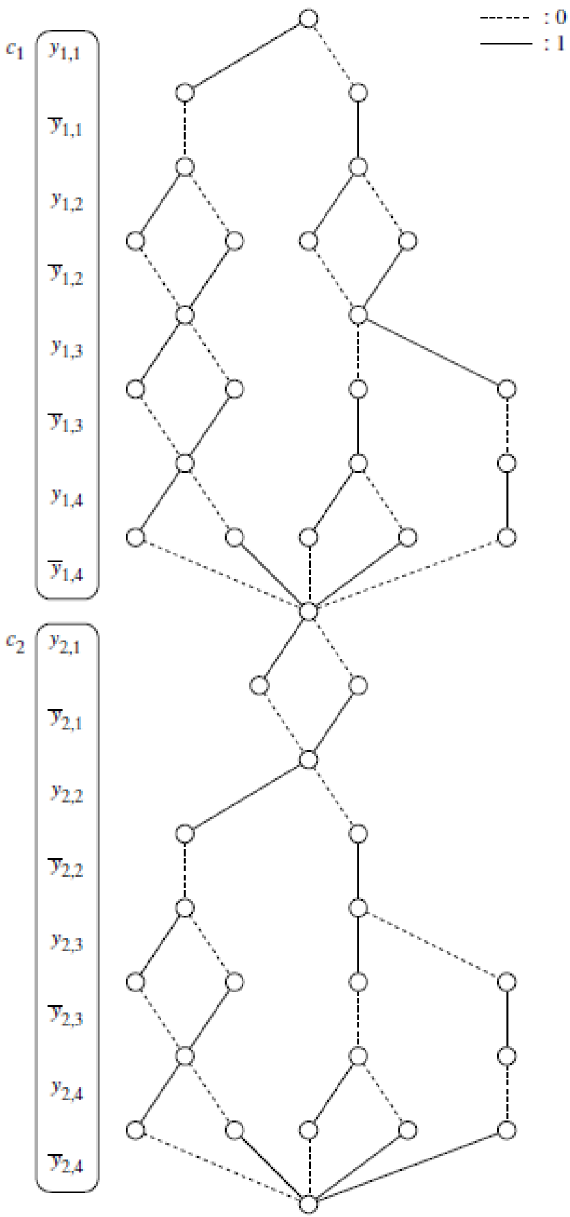


Each path from root to terminal corresponds to a satisfying assignment for this clause

$$c_1 = (x_1 \vee \overline{x}_3 \vee x_4)$$

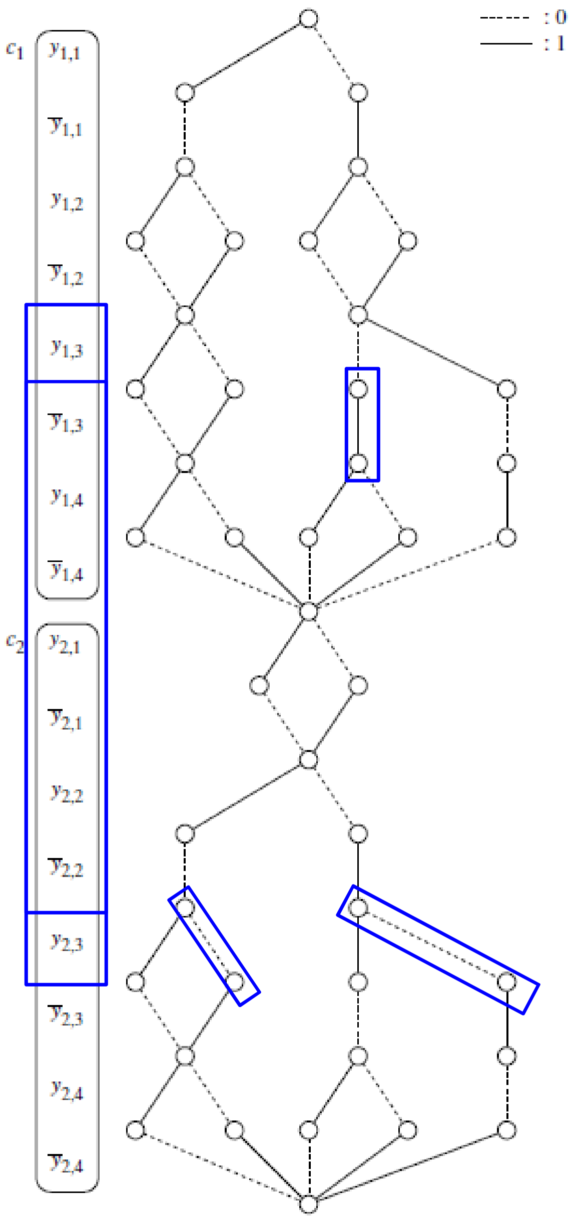
$$c_2 = (x_2 \vee x_3 \vee \overline{x}_4)$$

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

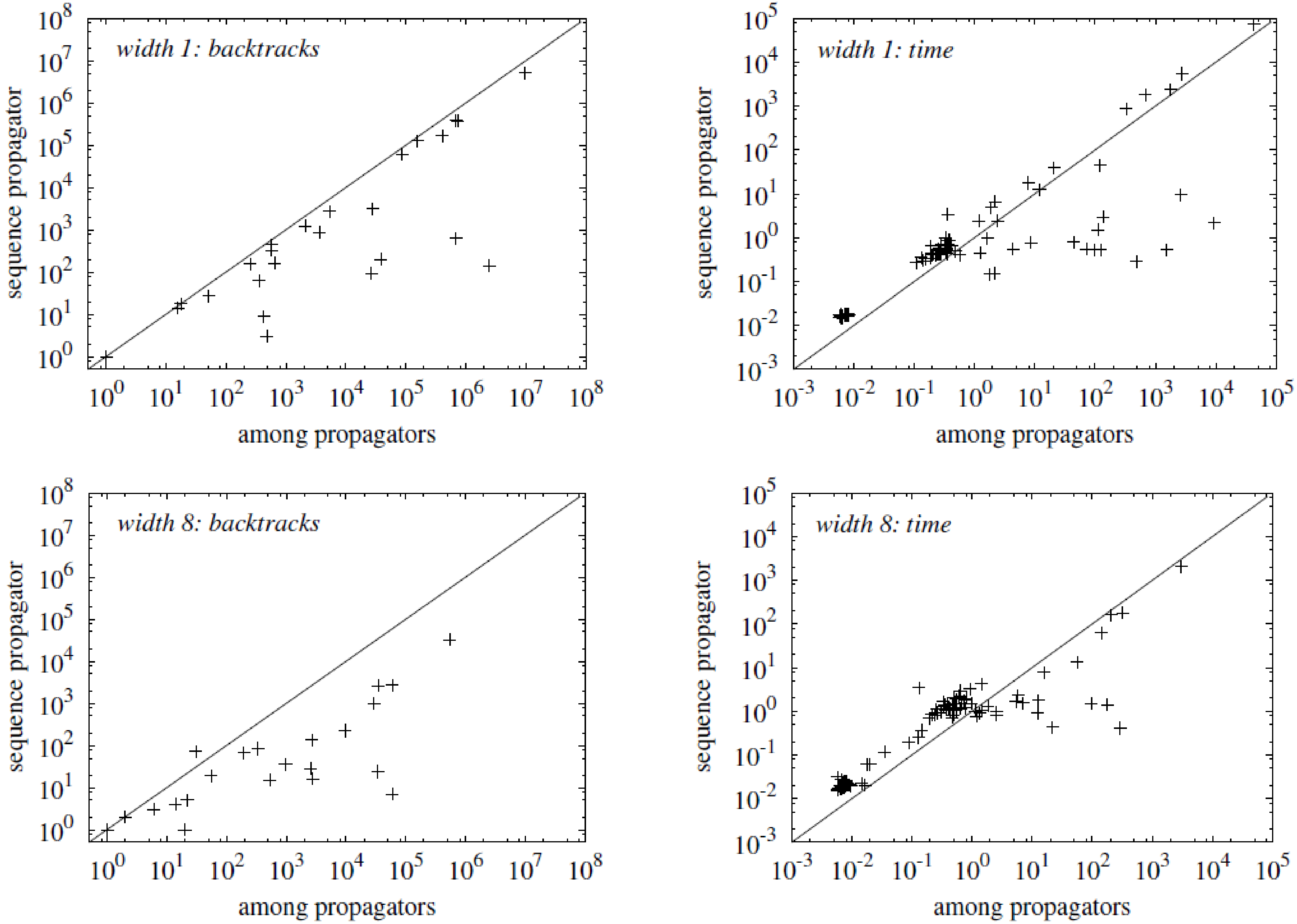
MDD Sequence versus Among

- 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

    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 propagator		MDD width 1		MDD width 4		MDD width 8		MDD width 16	
	n	BT	CPU	BT	CPU	BT	CPU	BT	CPU	BT	CPU
P-I	40	121,767	4.63	34,108	251.06	75	1.67	28	0.99	28	1.00
	50	121,777	5.67	34,108	487.25	75	3.06	29	1.86	29	1.90
	60	121,782	6.51	34,108	796.21	75	5.00	30	3.09	30	3.12
	70	121,787	6.99	34,108	1,110.88	75	6.96	28	4.30	28	4.33
	80	121,792	7.48	34,108	1,492.33	75	9.38	28	5.97	28	5.88
P-II	40	116,548	3.52	233,096	1,492.93	71	1.63	32	1.17	36	1.29
	50	116,548	3.98	-	>1,600	69	2.77	36	2.36	36	2.40
	60	116,548	4.56	-	>1,600	65	4.38	36	3.56	36	3.83
	70	116,548	4.95	-	>1,600	67	6.32	32	4.71	36	5.44
	80	116,548	5.08	-	>1,600	73	8.47	32	6.36	36	7.24

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