Decision Diagrams for Constraint Programming

Part 2

Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

www.andrew.cmu.edu/user/vanhoeve/mdd/

Plan

- Yesterday: MDD-based constraint propagation
  - Propagate relaxed MDDs instead of domains
  - Strength of MDD can be controlled by its maximum width
- Constraint-specific propagation algorithms
  - very similar to domain propagators
  - define state information for each constraint
  - central operations: edge filtering and node refinement
- Detailed example: *Among*
- Today:
  - MDD propagation for *Sequence* constraint
  - MDD propagation for disjunctive scheduling

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}

$$\left.\begin{array}{l} 0 \le x_1+x_2+ \ldots +x_9 \le 7 \\ 0 \le x_2+x_3+ \ldots +x_{10} \le 7 \\ 0 \le x_3+x_4+ \ldots +x_{11} \le 7 \\ 0 \le x_4+x_5+ \ldots +x_{12} \le 7 \end{array}\right\} =: \mathit{Sequence}([x_1,x_2,\ldots,x_{12}], q=9, S=\{1\}, l=0, u=7)$$

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

$$\downarrow$$

$$\mathit{Among}(X, S, l, u)$$

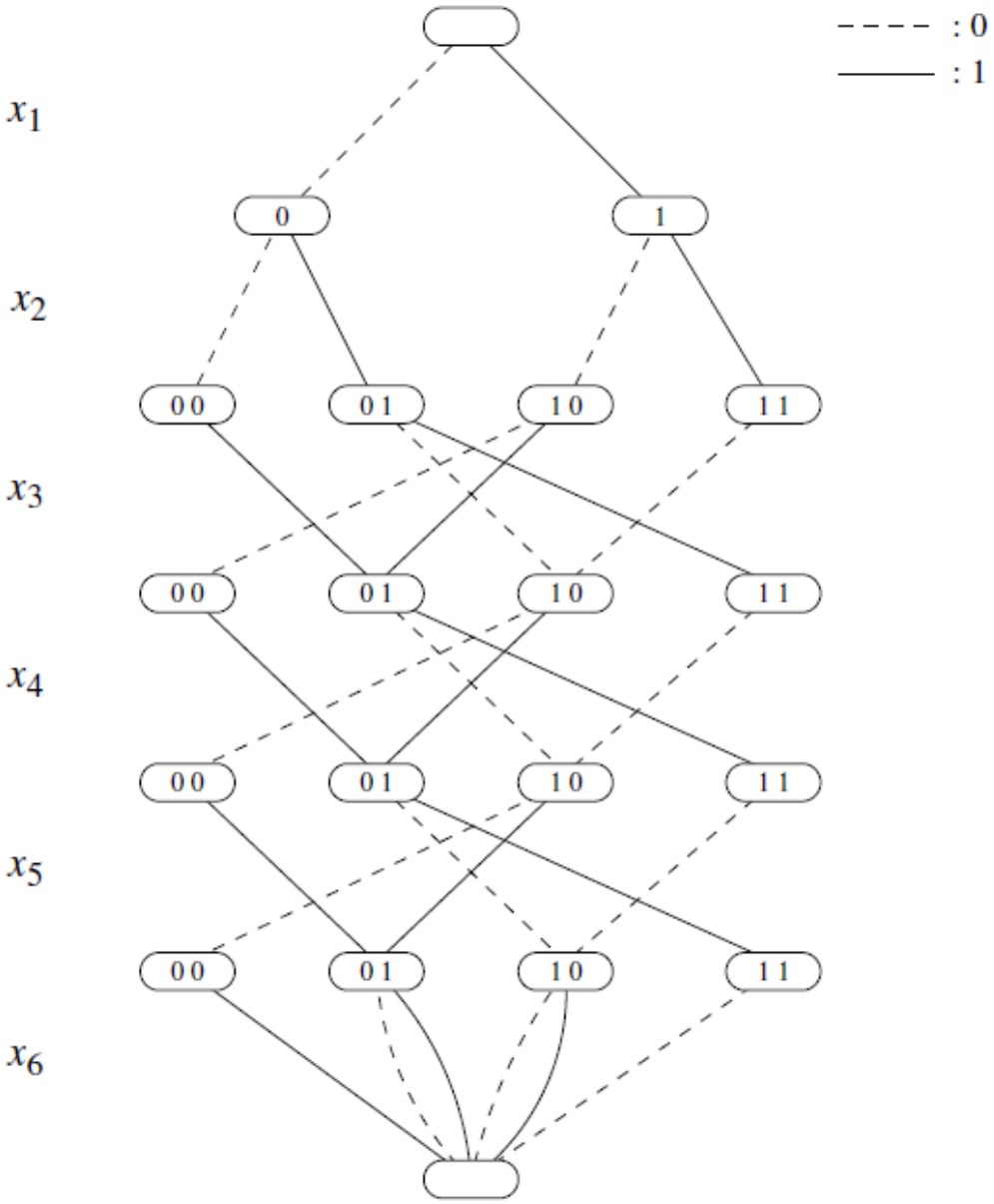
Motivation

- Domain propagation well understood for *Sequence*
  - several papers with improved propagation or time complexity
  - domain consistency can be established in $O(n^2)$ time
- Combining the *Among* constraints into a single *Sequence* global constraint can yield huge speedups

- We have developed MDD propagation for Among
  - Would it be useful to do the same for *Sequence*?
  - Perhaps MDD consistency in polynomial time?

D. Bergman, A. A. Cire, and W.-J. van Hoeve. MDD Propagation for Sequence Constraints. *JAIR*, Volume 50, pages 697-722, 2014.

MDD Representation for Sequence



- Equivalent to the DFA representation of *Sequence* for domain propagation

  [v.H. et al., 2006, 2009]

- Size $O(n2^{q-1})$

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 ordering follows the sequence of variables X)

Proof: Reduction from 3-SAT [JAIR, 2014]

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

Partial filter from 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

$$y_0 = 0$$

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

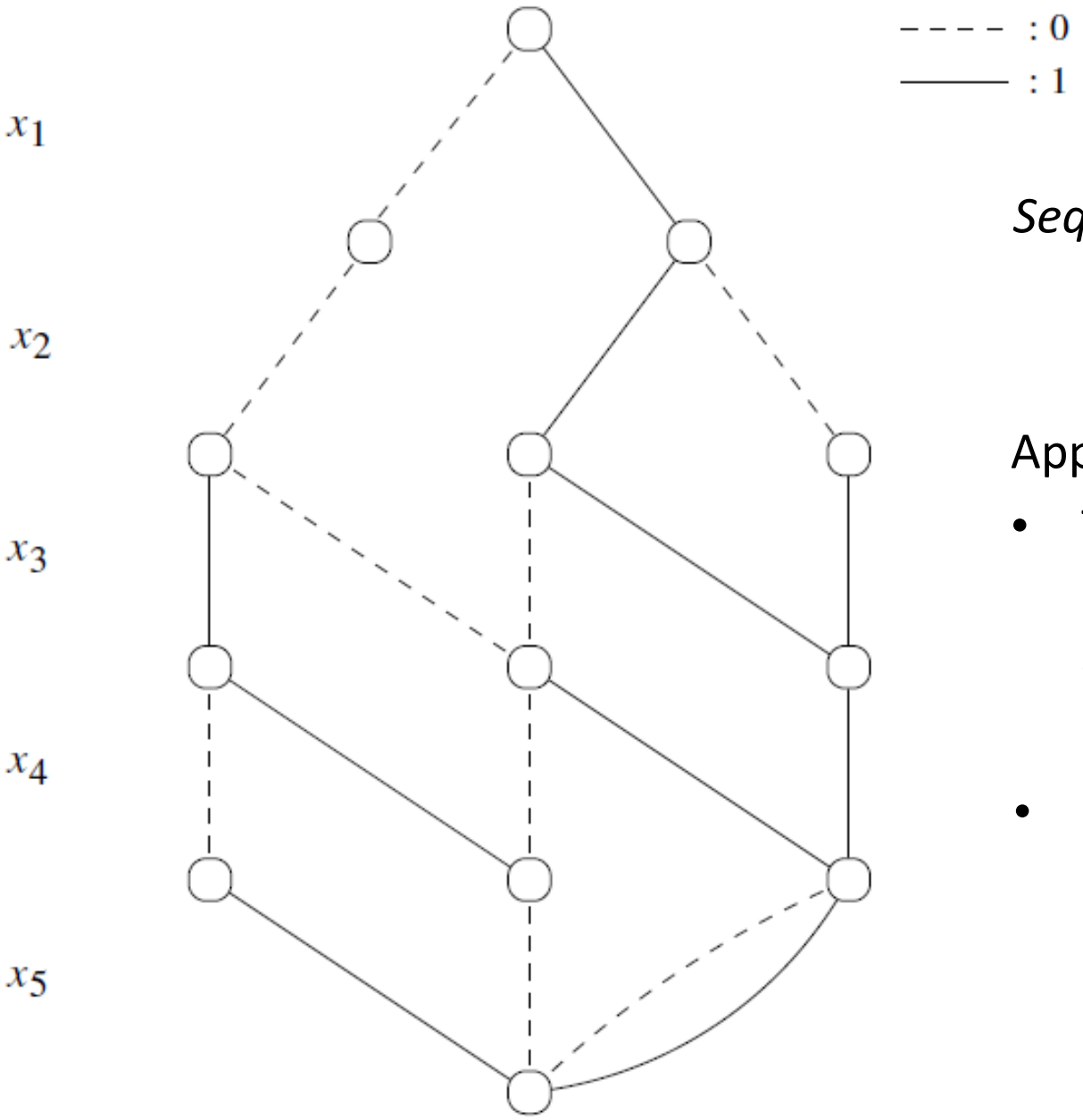
- Then the *Among* constraint on $[x_{i+1},\ldots, x_{i+q}]$ is equivalent to

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

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

- [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

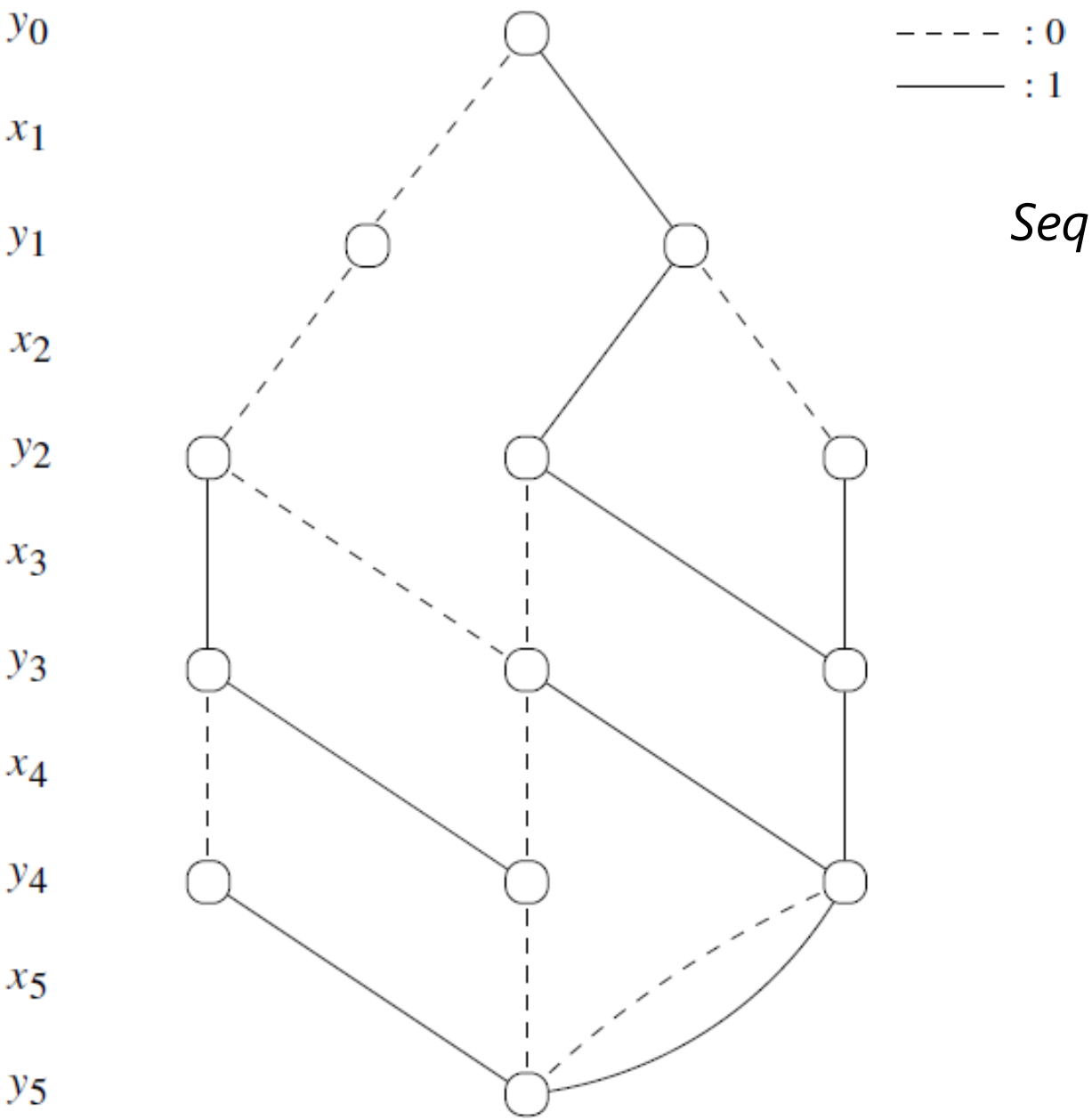


Sequence(X, q=3, S={1}, l=1, u=2)

Approach

- The auxiliary variables y_i can be naturally represented at the *nodes* of the MDD – this will be our state information
- 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)

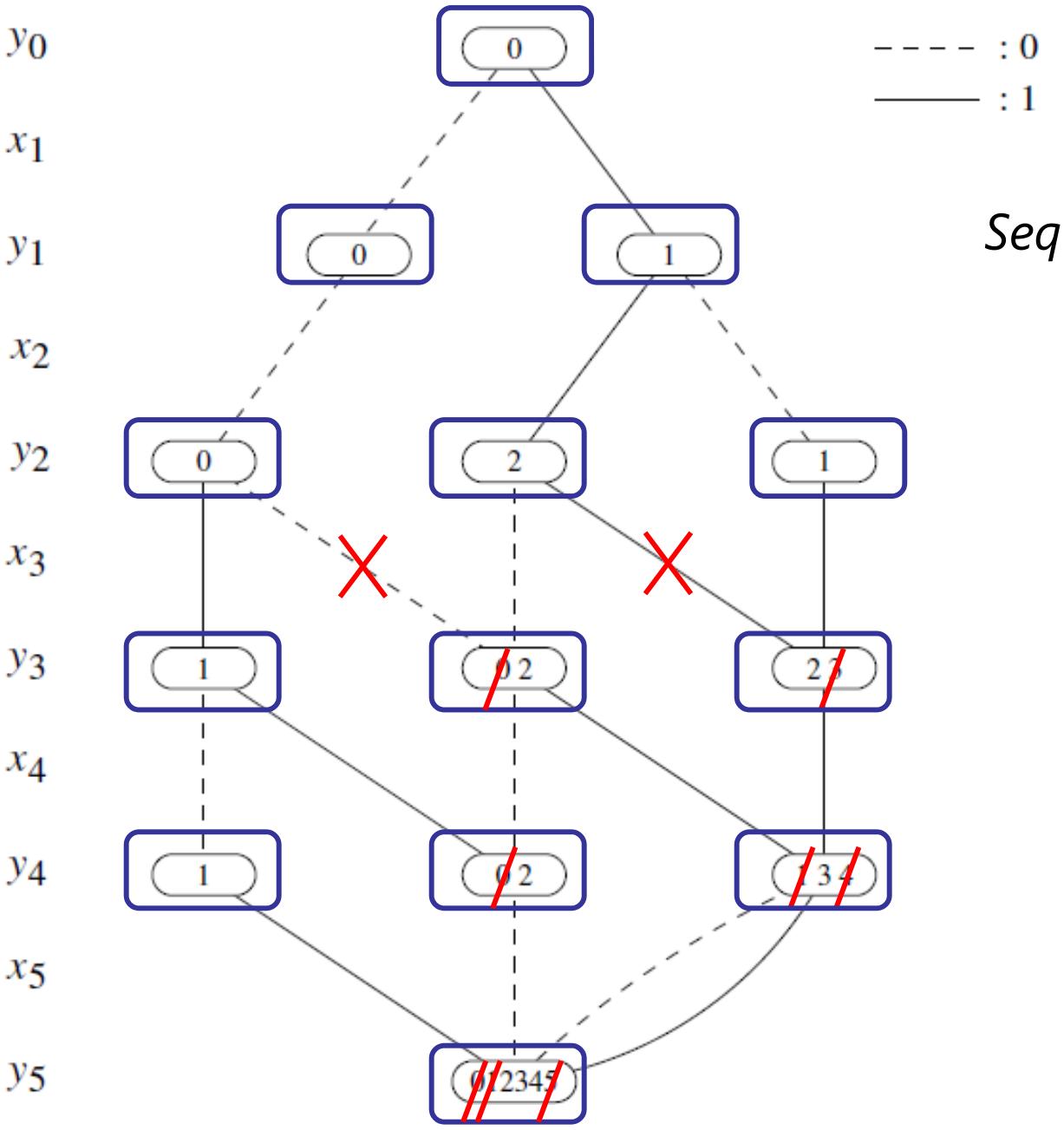
$$y_i = y_{i-1} + x_i$$

$$1 \le y_3 - y_0 \le 2$$

$$1 \le y_4 - y_1 \le 2$$

$$1 \le y_5 - y_2 \le 2$$

MDD filtering from decomposition



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

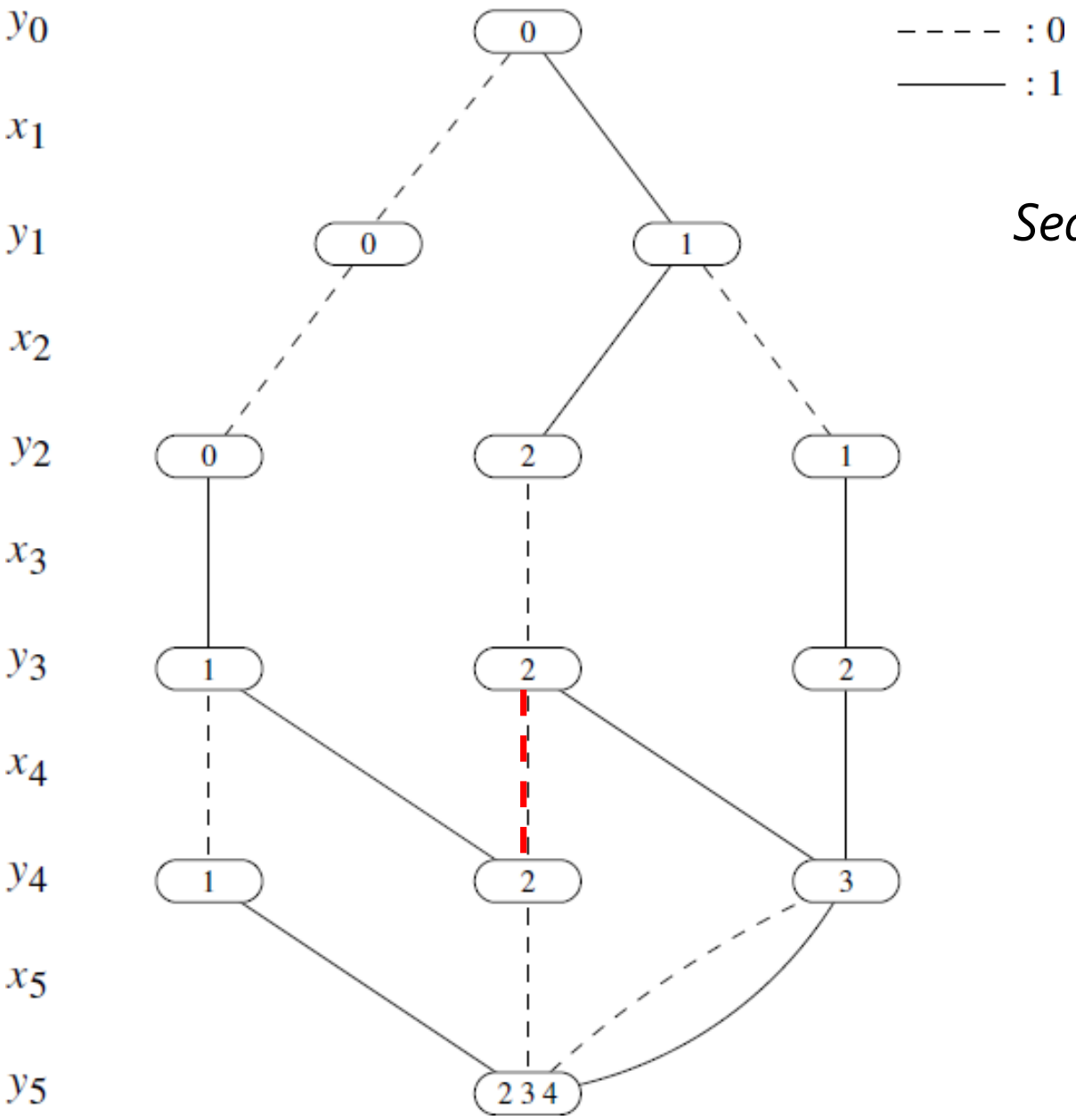
$y_i = y_{i-1} + x_i$

$1 \le y_3 - y_0 \le 2$

$1 \le y_4 - y_1 \le 2$

$1 \le y_5 - y_2 \le 2$

MDD filtering from decomposition



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

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

$$1 \le y_3 - y_0 \le 2$$

$$1 \le y_4 - y_1 \le 2$$

$$1 \le y_5 - y_2 \le 2$$

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)
  - but since we propagate inequalities, we only need to maintain min and max value over previous *q* layers: $O(qW)$
- One top-down and one bottom-up pass

Experimental Setup

- Decomposition-based MDD filtering algorithm
  - Implemented as global constraint in IBM ILOG CPLEX CP Optimizer 12.4
- Evaluation
  - Compare MDD filtering with Domain filtering for Sequence and for the same ‘cumulative sums’ decomposition (achieves domain consistency for 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

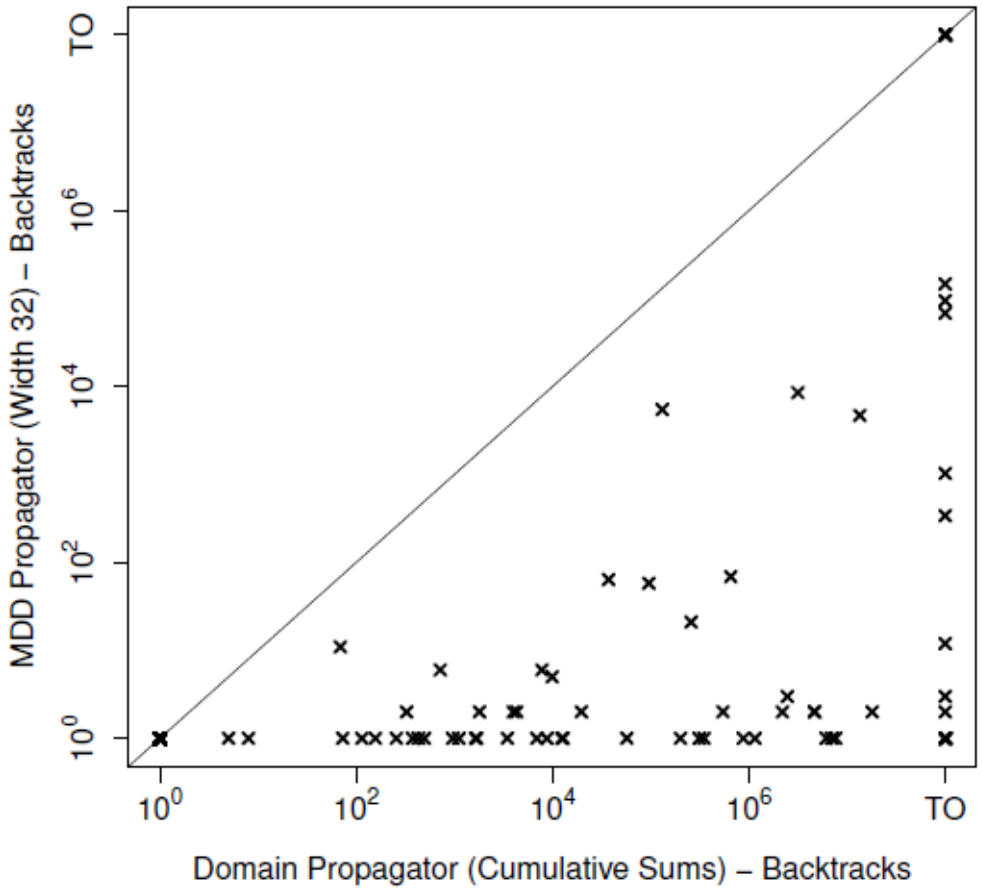
- Randomly generated instances
  - n=50 variables
  - domain {0,1,…,10}
  - 5 random *Sequence* constraints
  - 250 instances total

(see paper for more details)

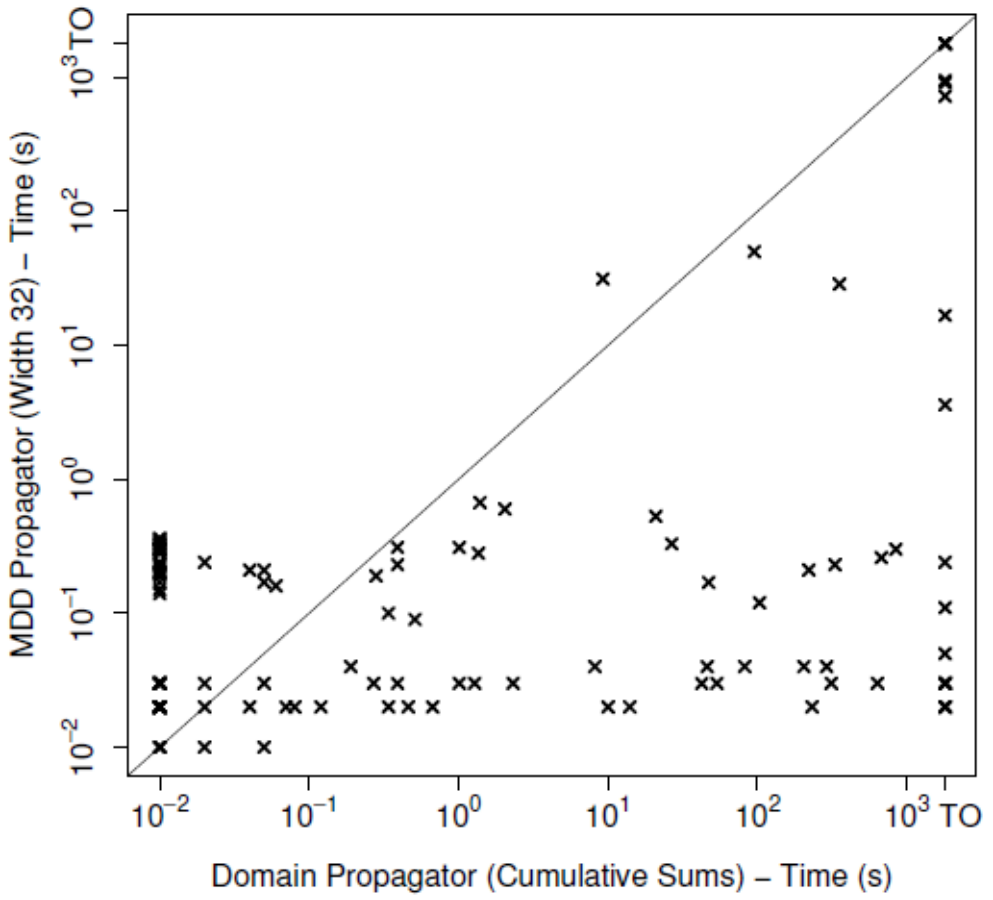
- Vary maximum width of MDD
  - widths 2 up to 128

MDD vs Domain Propagation

backtracks

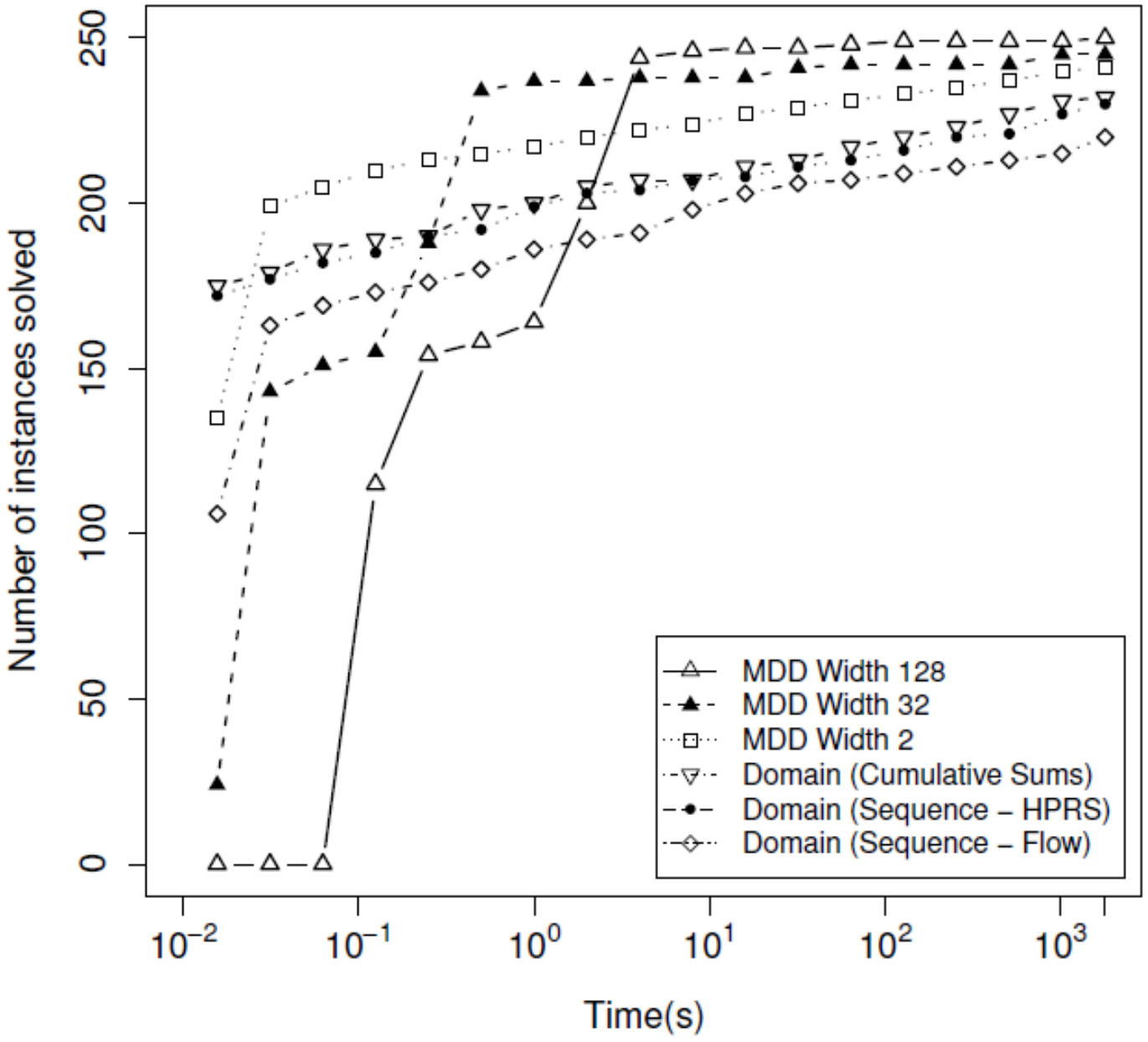


time

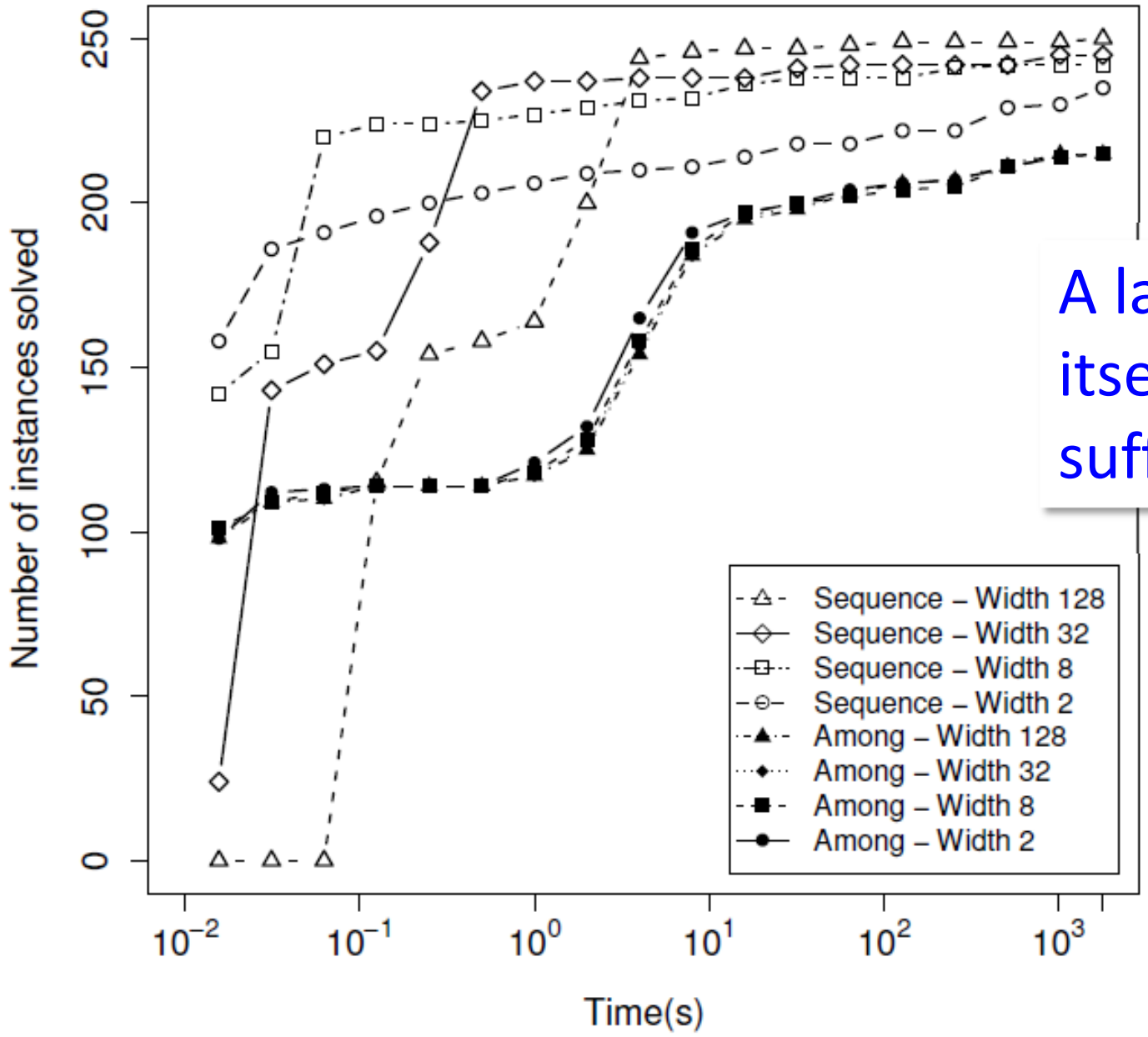


Performance of MDD (width 32) and Domain propagation on systems of Sequence constraints

Performance Comparison for Sequence



Sequence vs. Among

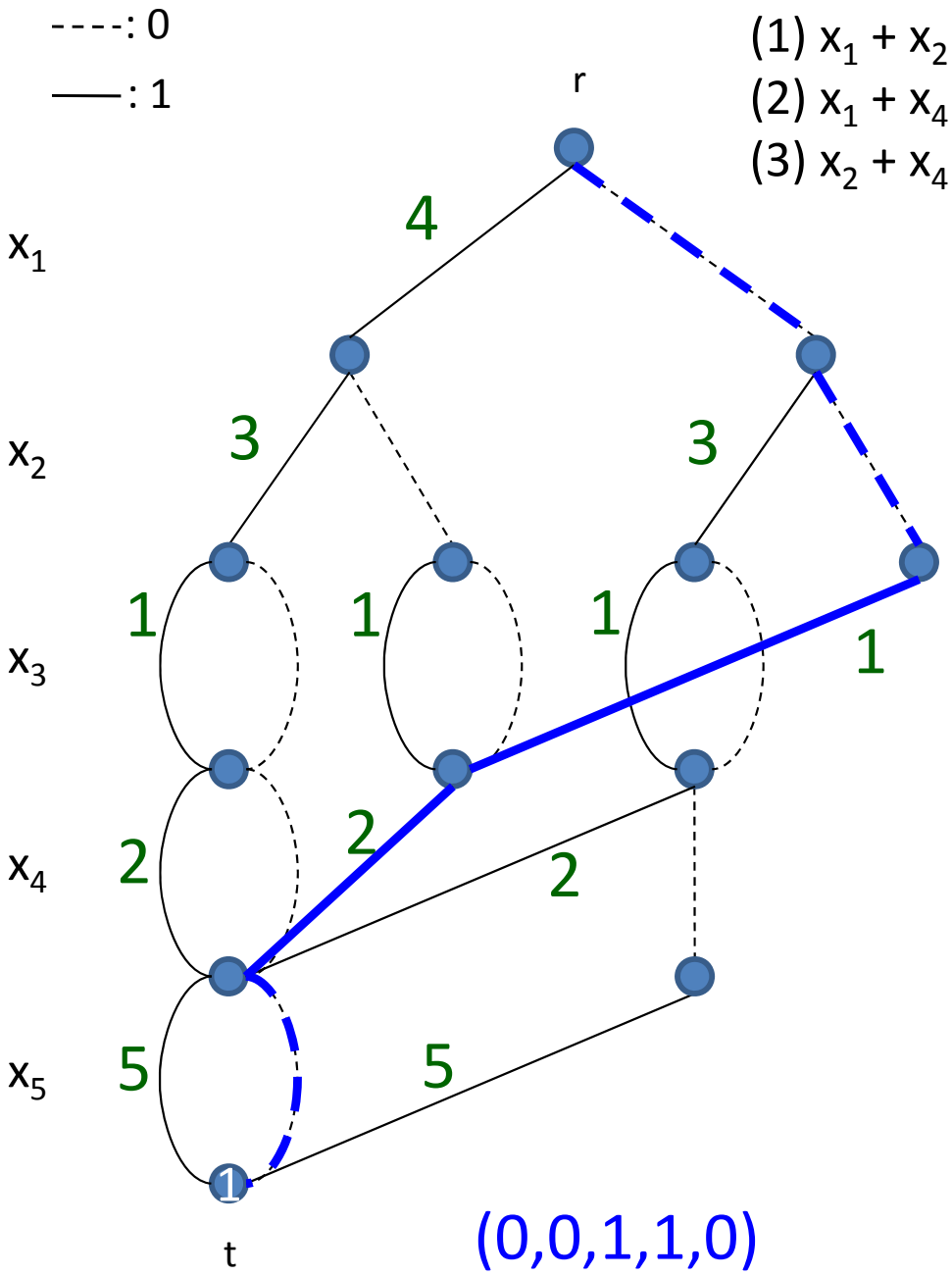


A large MDD by itself may not be sufficient!

Coming up

- MDDs can handle objective functions as well
- Important for many CP problems
  - e.g., disjunctive scheduling
  - minimize makespan, weighted completion times, etc.
- We will develop an MDD approach to disjunctive scheduling
  - combines MDD propagation and optimization reasoning

Handling objective functions



Suppose we have an objective:

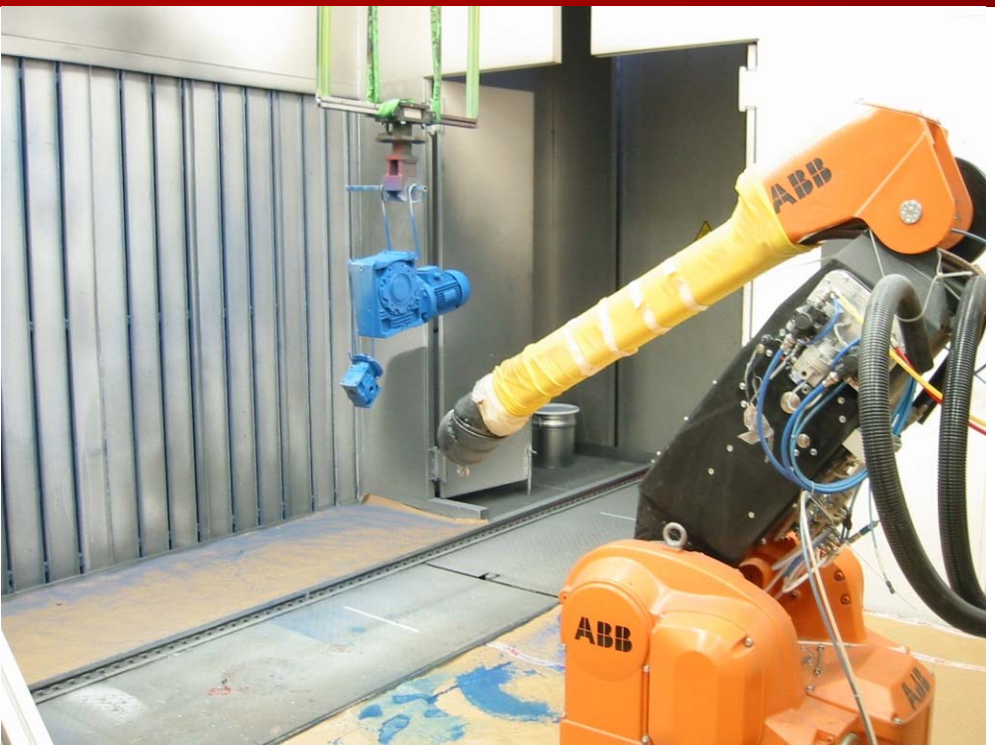
$\min 4x_1+3x_2+x_3+2x_4+5x_5$

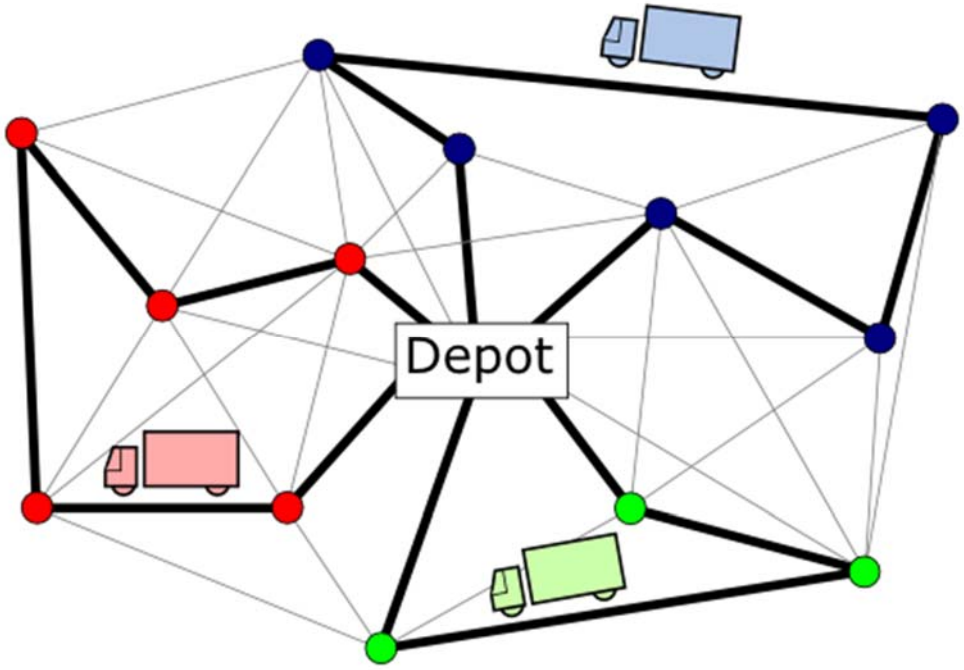
shortest path computation

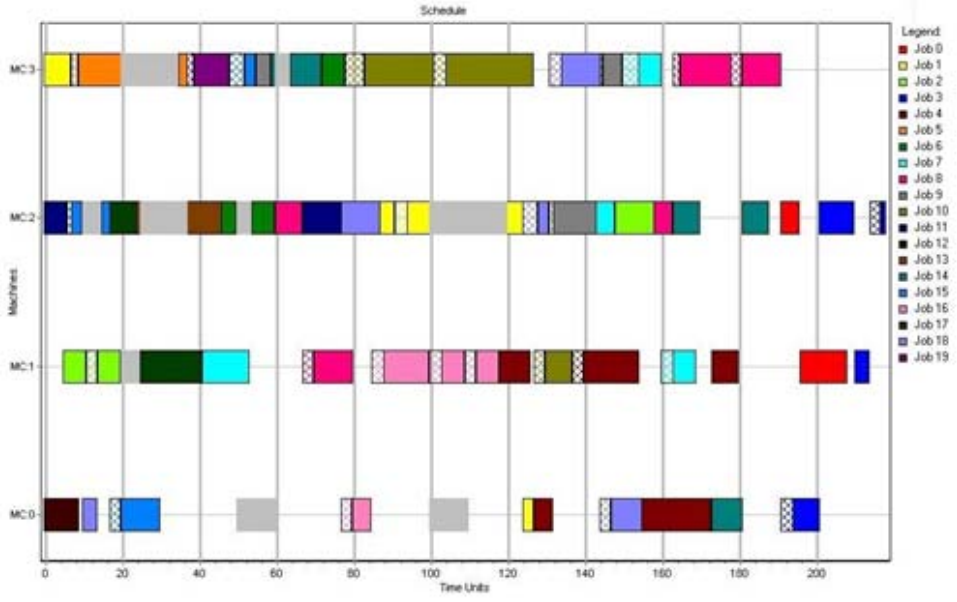
MDDs for Disjunctive Scheduling

- Cire and v.H. Multivalued Decision Diagrams for Sequencing Problems. *Operations Research* 61(6): 1411-1428, 2013.

Disjunctive Scheduling



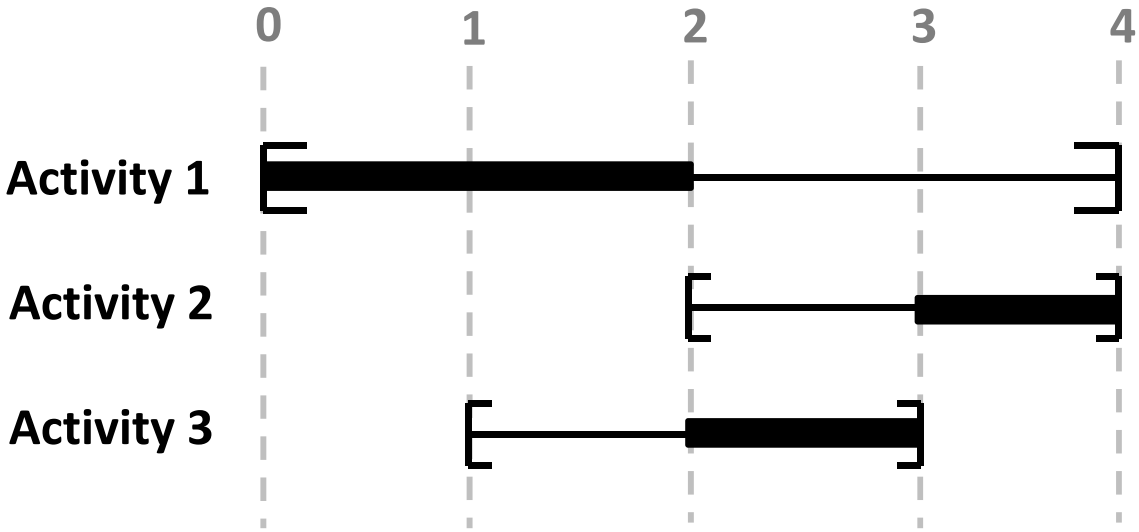




Disjunctive Scheduling in CP

- Sequencing and scheduling of activities on a resource

- *Activities*
  - Processing time: p_i
  - Release time: r_i
  - Deadline: d_i
  - Start time variable: s_i



- *Resource*
  - Nonpreemptive
  - Process one activity at a time

Extensions

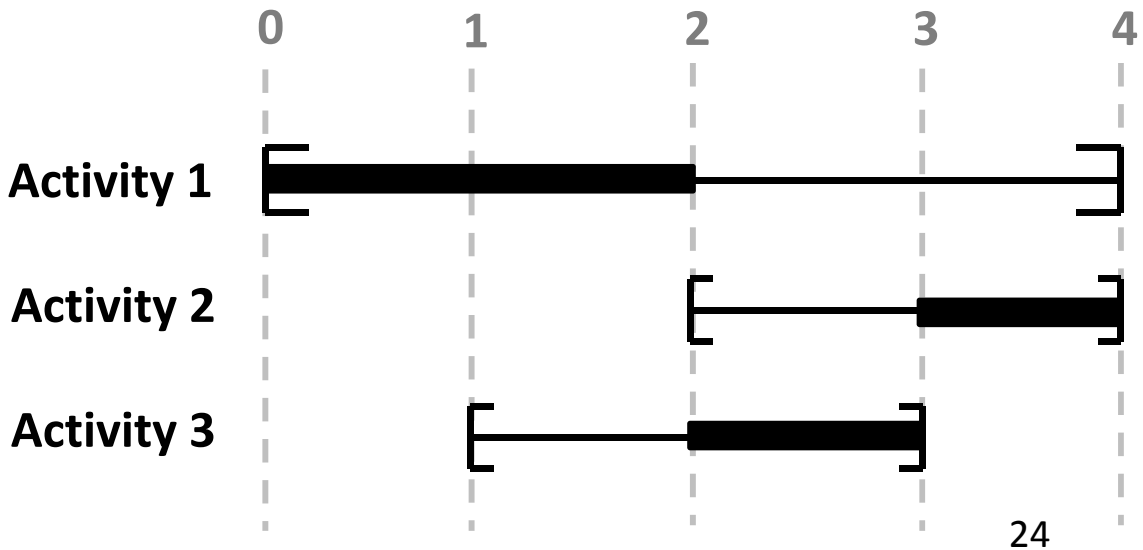
- Precedence relations between activities
- Sequence-dependent setup times
- Various objective functions
  - Makespan
  - Sum of setup times
  - (Weighted) sum of completion times
  - (Weighted) tardiness
  - number of late jobs
  - …

Inference

- Inference for disjunctive scheduling
  - Precedence relations
  - Time intervals that an activity can be processed
- Sophisticated techniques include:
  - *Edge-Finding*
  - *Not-first / not-last rules*

- Examples: $1 \ll 3$

  $s_3 \geq 3$



Assessment of CP Scheduling

- Disjunctive scheduling may be viewed as the 'killer application' for CP
  - Natural modeling (activities and resources)
  - Allows many side constraints (precedence relations, time windows, setup times, etc.)
  - Among state of the art while being generic methodology
- However, CP has some problems when
  - objective is not minimize makespan (but instead, e.g., weighted sum of lateness)
  - setup times are present
  - …
- What can MDDs bring here?

optimization

MDDs for Disjunctive Scheduling

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Three main considerations:

- Representation
  - How to represent solutions of disjunctive scheduling in an MDD?
- Construction
  - How to construct this relaxed MDD?
- Inference techniques
  - What can we infer using the relaxed MDD?

MDD Representation

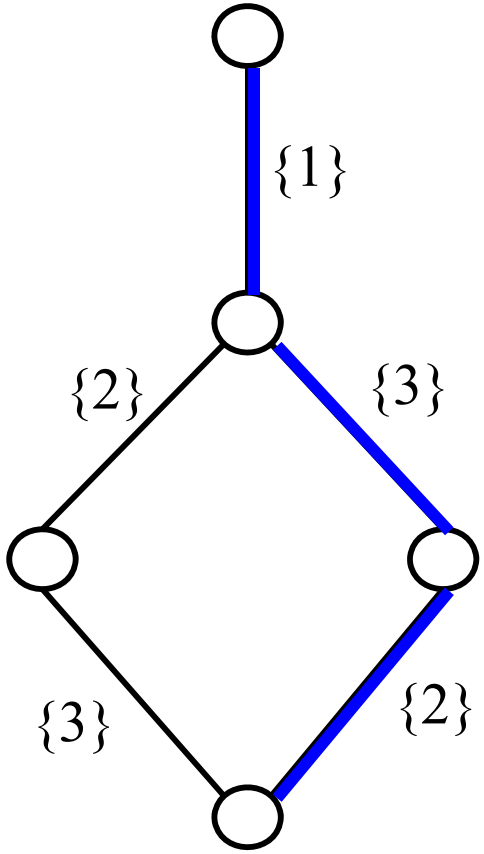
- Natural representation as 'permutation MDD'
- Every solution can be written as a permutation $\boldsymbol{\pi}$

  $\pi_1, \pi_2, \pi_3, \ldots, \pi_n$: activity sequencing in the resource

- Schedule is *implied* by a sequence, e.g.:

$$start_{\pi_i} \geq start_{\pi_{i-1}} + p_{\pi_{i-1}} \qquad i = 2, \ldots, n$$

MDD Representation



π_1

π_2

π_3

Act	r_i	p_i	d_i
1	0	2	3
2	4	2	9
3	3	3	8

Path {1} – {3} – {2} :

$0 \leq start_1 \leq 1$

$6 \leq start_2 \leq 7$

$3 \leq start_3 \leq 5$

Exact MDD Compilation

Theorem: *Constructing the exact MDD for a Disjunctive Instance is an NP-Hard problem*

- We work with MDD relaxations instead
- Bounded size in specific cases, e.g. (Balas [99]):

- TSP defined on a complete graph
- Given a fixed parameter **k**, we must satisfy

$$i \ll j \quad \text{if} \quad j - i \geq k \quad \text{for cities i, j}$$

Theorem: *The exact MDD for the TSP above has $O(n2^k)$ nodes*

MDD-based propagation

Propagation: remove infeasible arcs from the MDD

We can utilize several structures/constraints:

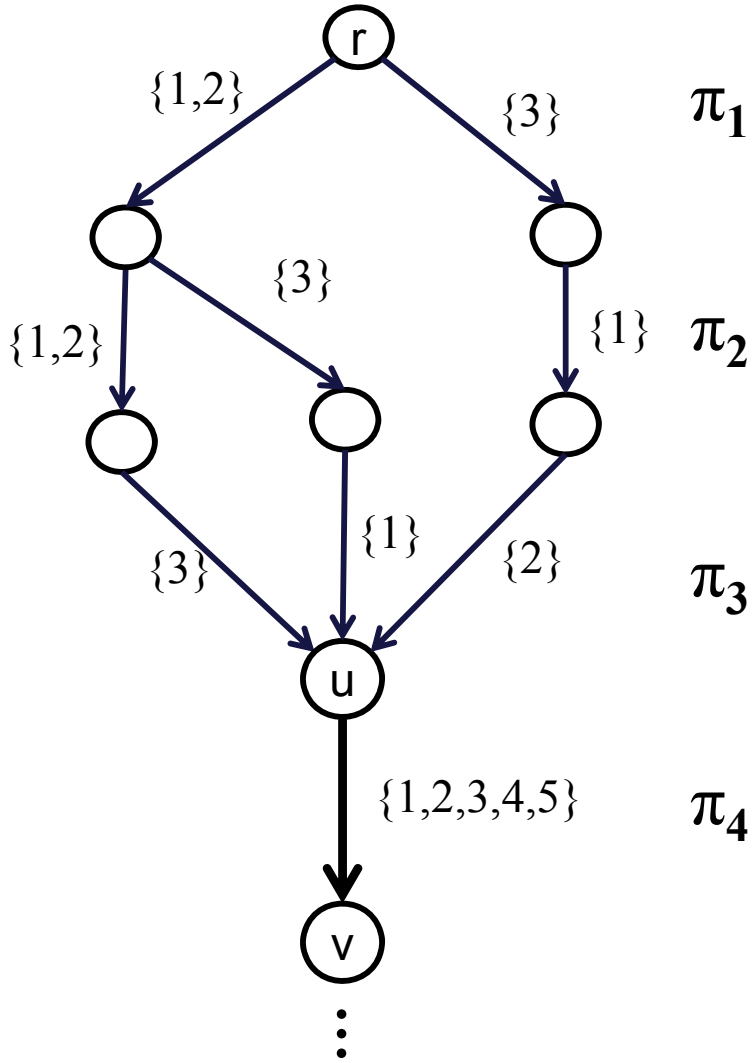
- *Alldifferent* for the permutation structure
- Earliest start time and latest end time
- Precedence relations

For a given constraint type we maintain specific 'state information' at each node in the MDD

  - both top-down and bottom-up

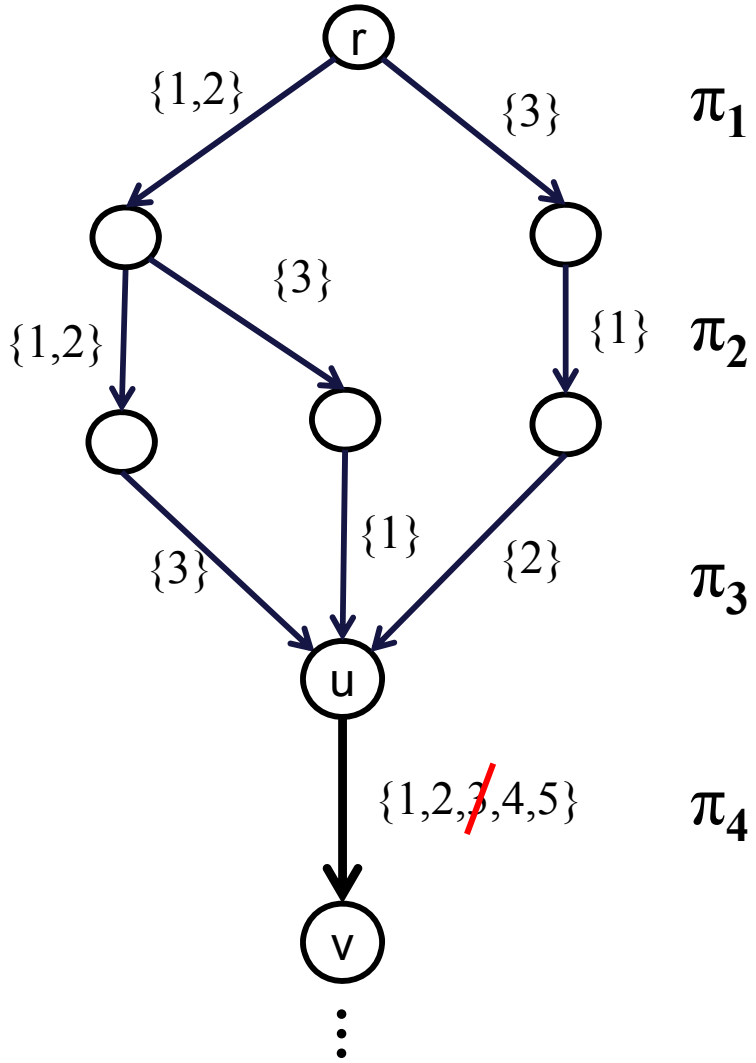
Propagation (cont'd)

- State information at each node *i*
  - labels on *all* paths: A_i
  - labels on *some* paths: S_i
  - earliest starting time: E_i
  - latest completion time: L_i
- Top down example for arc (u,v)



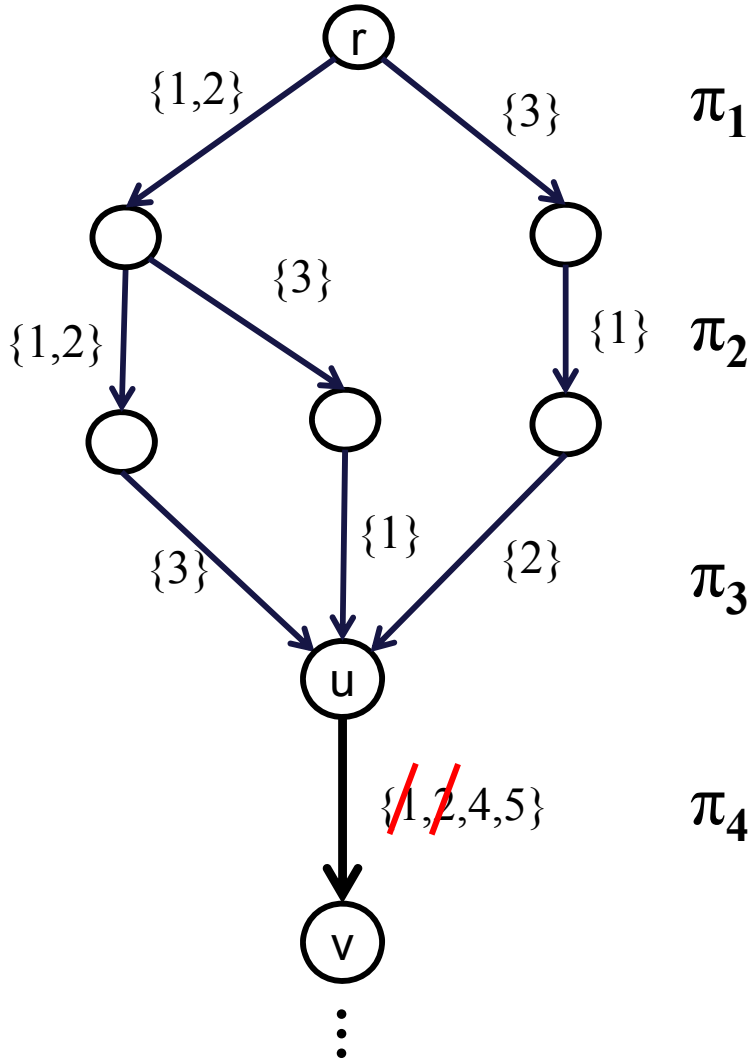
Alldifferent Propagation

- All-paths state: A_u
  - Labels belonging to all paths from node r to node u
  - $A_u = \{3\}$
  - Thus eliminate {3} from (u,v)



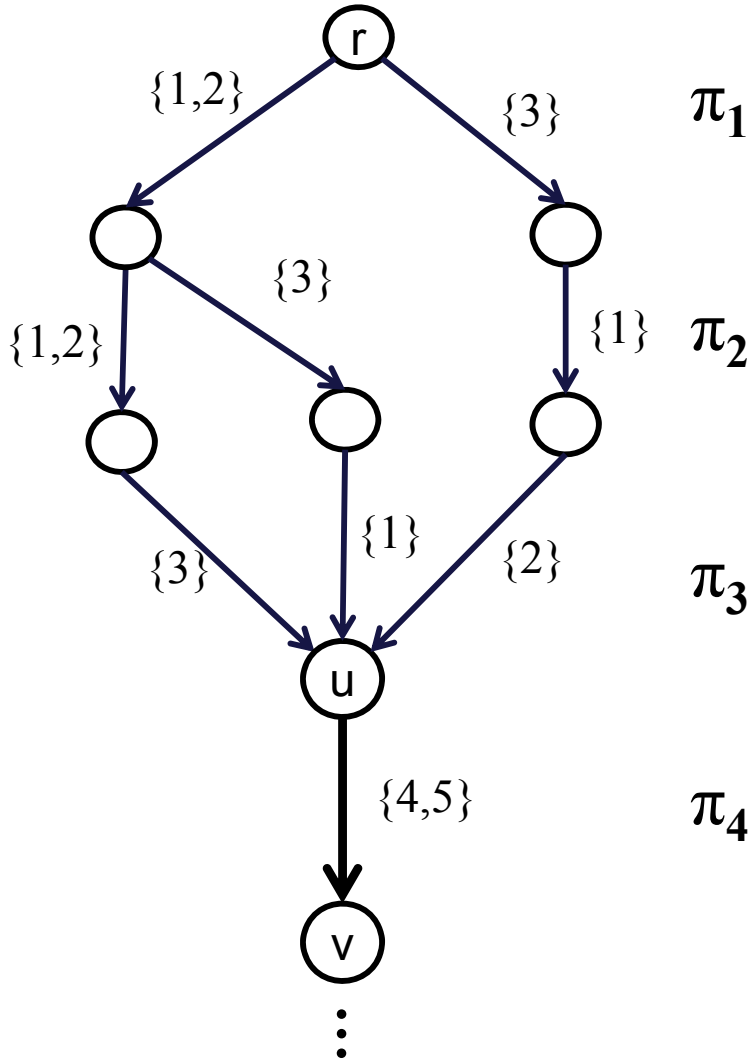
Alldifferent Propagation

- Some-paths state: S_u
  - Labels belonging to some path from node r to node u
  - $S_u = \{1,2,3\}$
  - Identification of Hall sets
  - Thus eliminate {1,2,3} from (u,v)



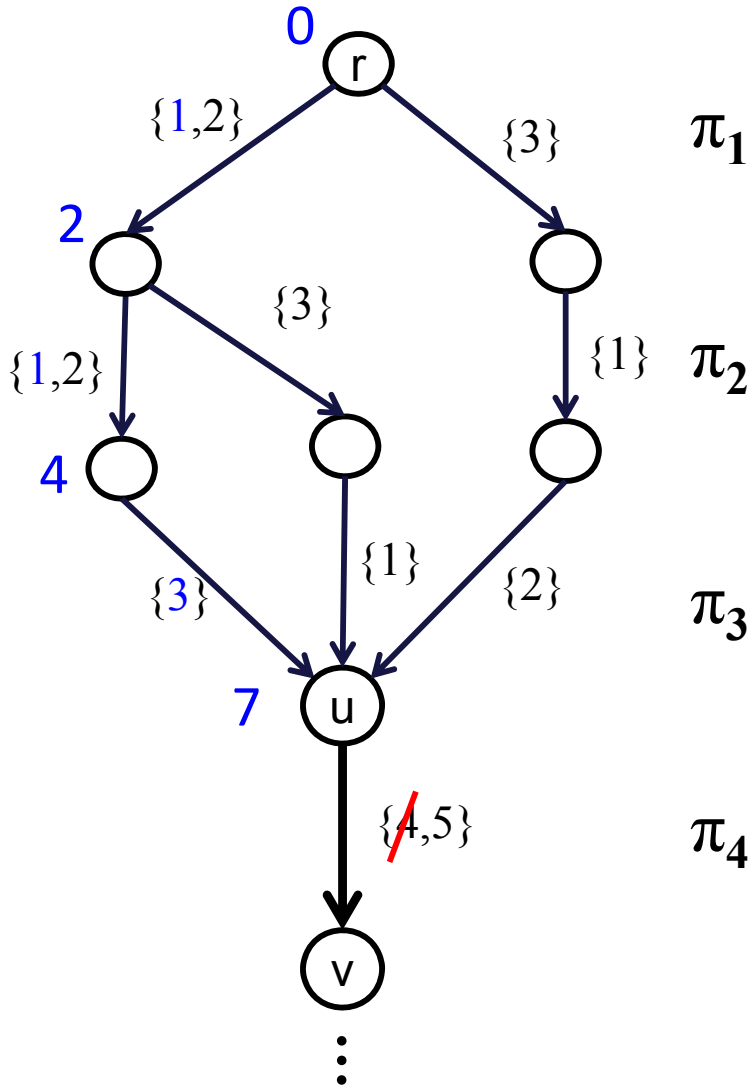
Propagate Earliest Completion Time

- Earliest Completion Time: E_u
  - Minimum completion time of all paths from root to node u
- Similarly: Latest Completion Time



Propagate Earliest Completion Time

Act	r_i	d_i	p_i
1	0	4	2
2	3	7	3
3	1	8	3
4	**5**	**6**	**1**
5	2	10	3



- $E_u = 7$
- Eliminate 4 from (u,v)

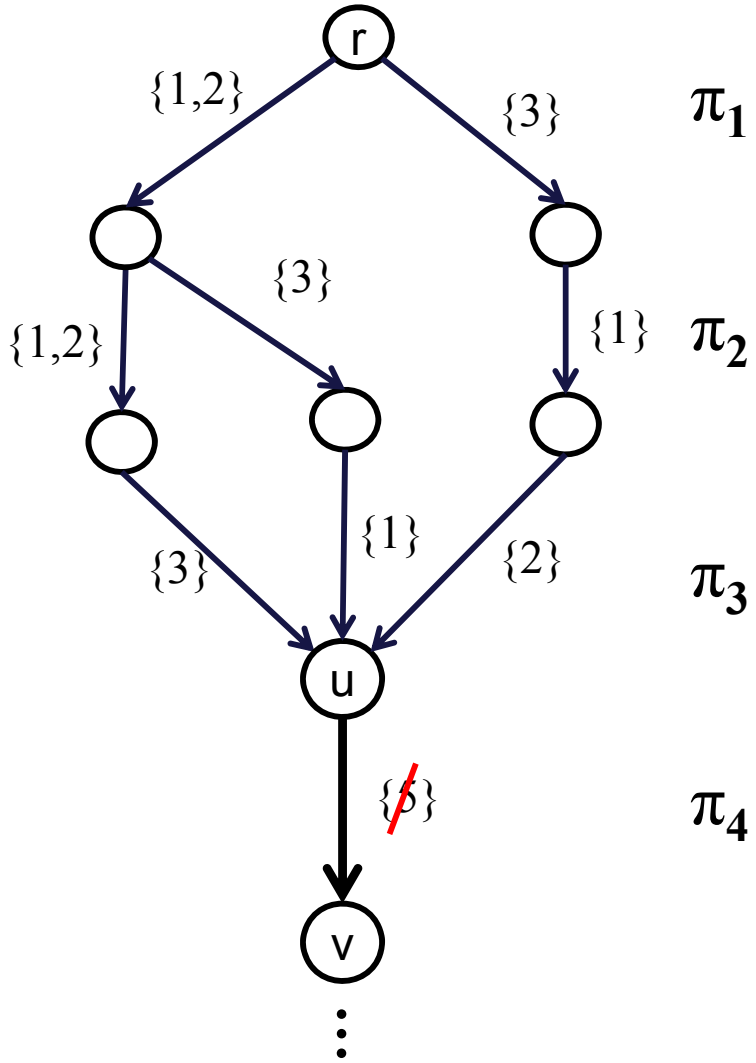
Propagate Precedence Relations

- Arc with label *j* infeasible if

  $i \ll j$ and *i* not on some path from r

- Suppose $4 \ll 5$
  - $S_u = \{1,2,3\}$
  - Since 4 not in S_u, eliminate 5 from (u,v)

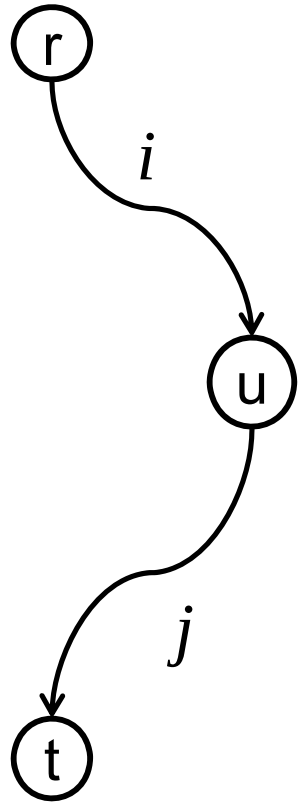
- Similarly: Bottom-up for $j \ll i$



More MDD Inference

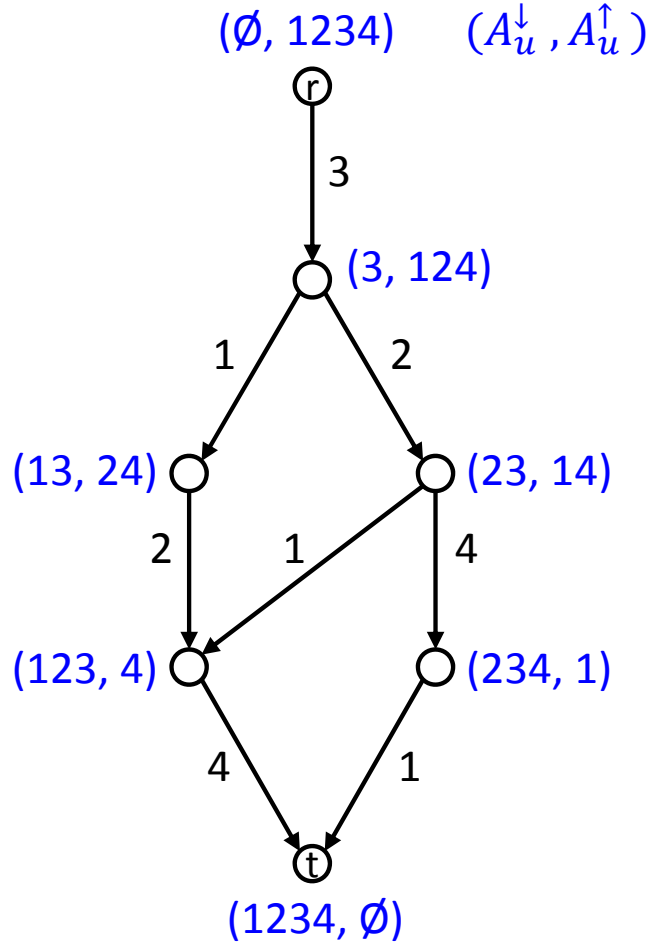
Theorem: *Given the exact MDD M, we can deduce all implied activity precedences in polynomial time in the size of M*

- For a node *v*,
  - $A_u^{\downarrow}$: values in all paths from root to *u*
  - $A_u^{\uparrow}$: values in all paths from node *u* to terminal
- *Precedence relation* $i \ll j$ *holds if and only if* $(j \notin A_u^{\downarrow})$ or $(i \notin A_u^{\uparrow})$ *for all nodes u in M*
- Same technique applies to relaxed MDD

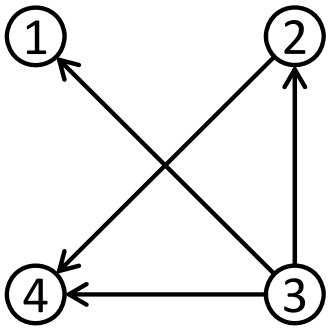


Extracting precedence relations

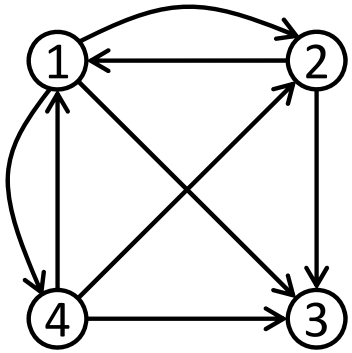
- Build a digraph G=(V, E) where V is the set of activities
- For each node u in M
  - if $j \in A_u^{\downarrow}$ and $i \in A_u^{\uparrow}$ add edge (*i*,*j*) to E
  - represents that $i \ll j$ cannot hold
- Take complement graph $\overline{G}$
  - complement edge exists iff $i \ll j$ holds



$3 \ll 1$
$3 \ll 2$
$3 \ll 4$
$2 \ll 4$



$\overline{G}$



G

Extracting precedence relations

- Build a digraph G=(V, E) where V is the set of activities
- For each node u in M
  - if $j \in A_u^{\downarrow}$ and $i \in A_u^{\uparrow}$ add edge (*i*,*j*) to E
  - represents that $i \ll j$ cannot hold
- Take complement graph $\overline{G}$
  - complement edge exists iff $i \ll j$ holds
- Time complexity: O($|M|n^2$)

- Same technique applies to *relaxed* MDD
  - add an edge if $j \in S_u^{\downarrow}$ and $i \in S_u^{\uparrow}$
  - complement graph represents subset of precedence relations

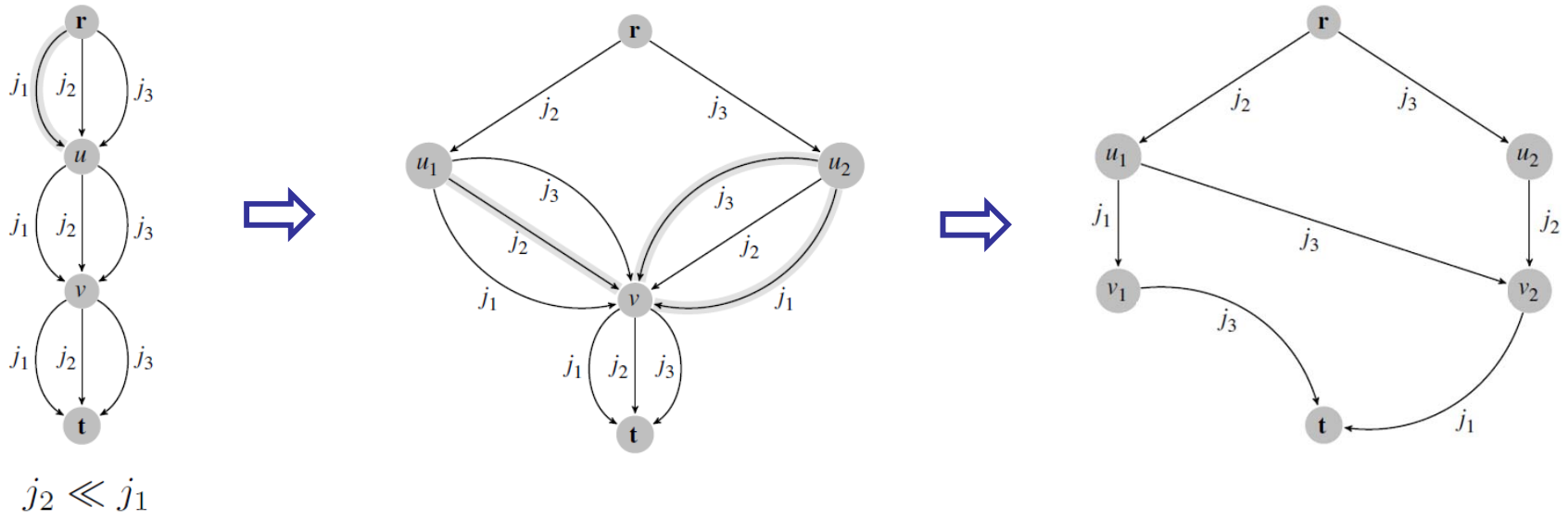
Communicate Precedence Relations

Carnegie Mellon Tepper SCHOOL OF BUSINESS

1. Provide precedence relations from MDD to CP
   - update start/end time variables
   - other inference techniques may utilize them
   - (some of the precedence relations found by the MDD may not be detected by existing CP methods)

2. Filter the MDD using precedence relations from other (CP) techniques

MDD Construction and Refinement



- To refine the MDD, we generally want to identify equivalence classes among nodes in a layer
  - For sequencing, deciding equivalence is NP-hard
- In practice, refinement can be based on
  - earliest starting time
  - latest earliest completion time r_i+p_i
  - *alldifferent* constraint (A_i and S_i states)

Computational Evaluation

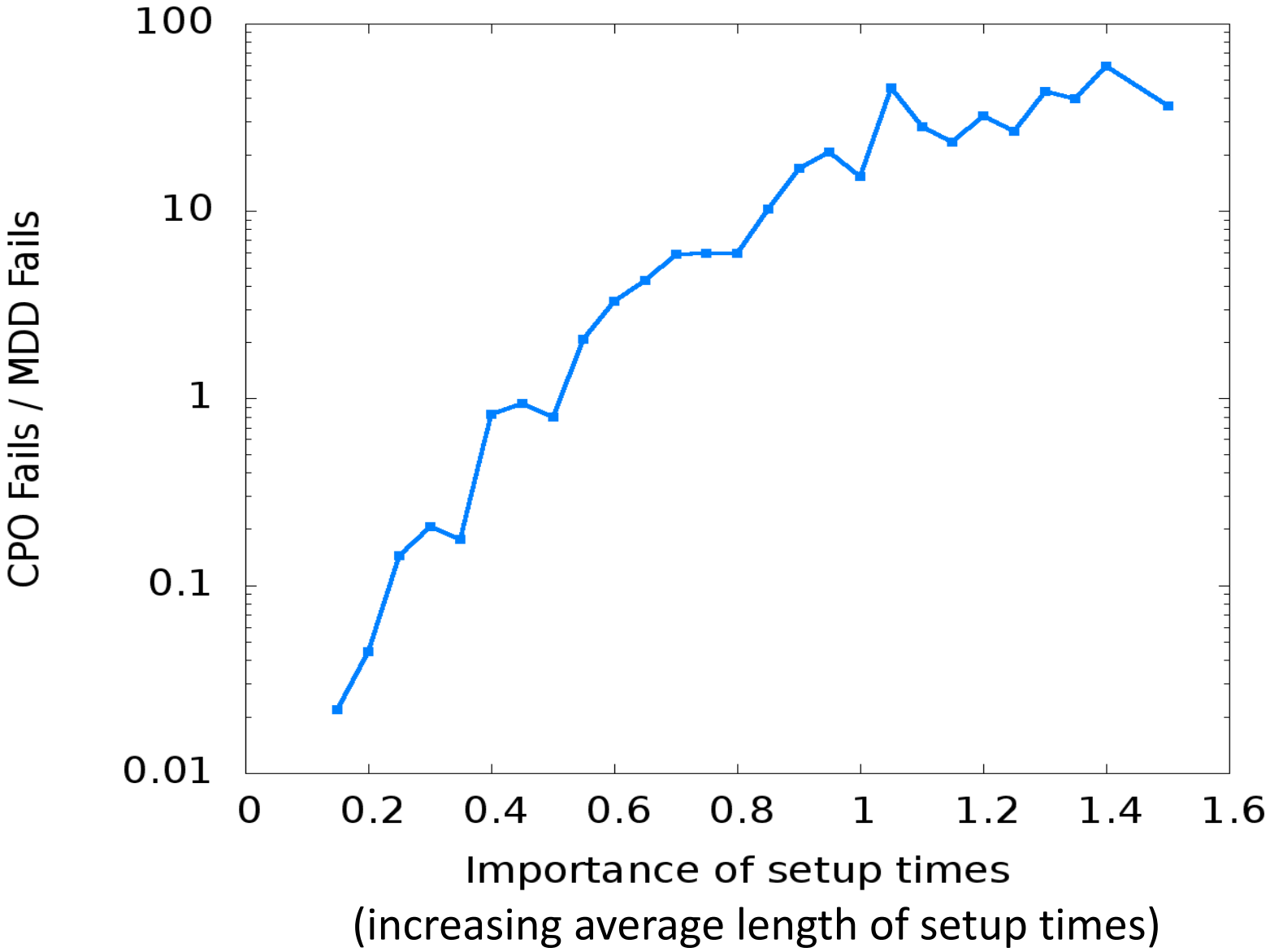
- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation

- Three different variants
  - CPO (only use CPO propagation)
  - MDD (only use MDD propagation)
  - CPO+MDD (use both)

Problem classes

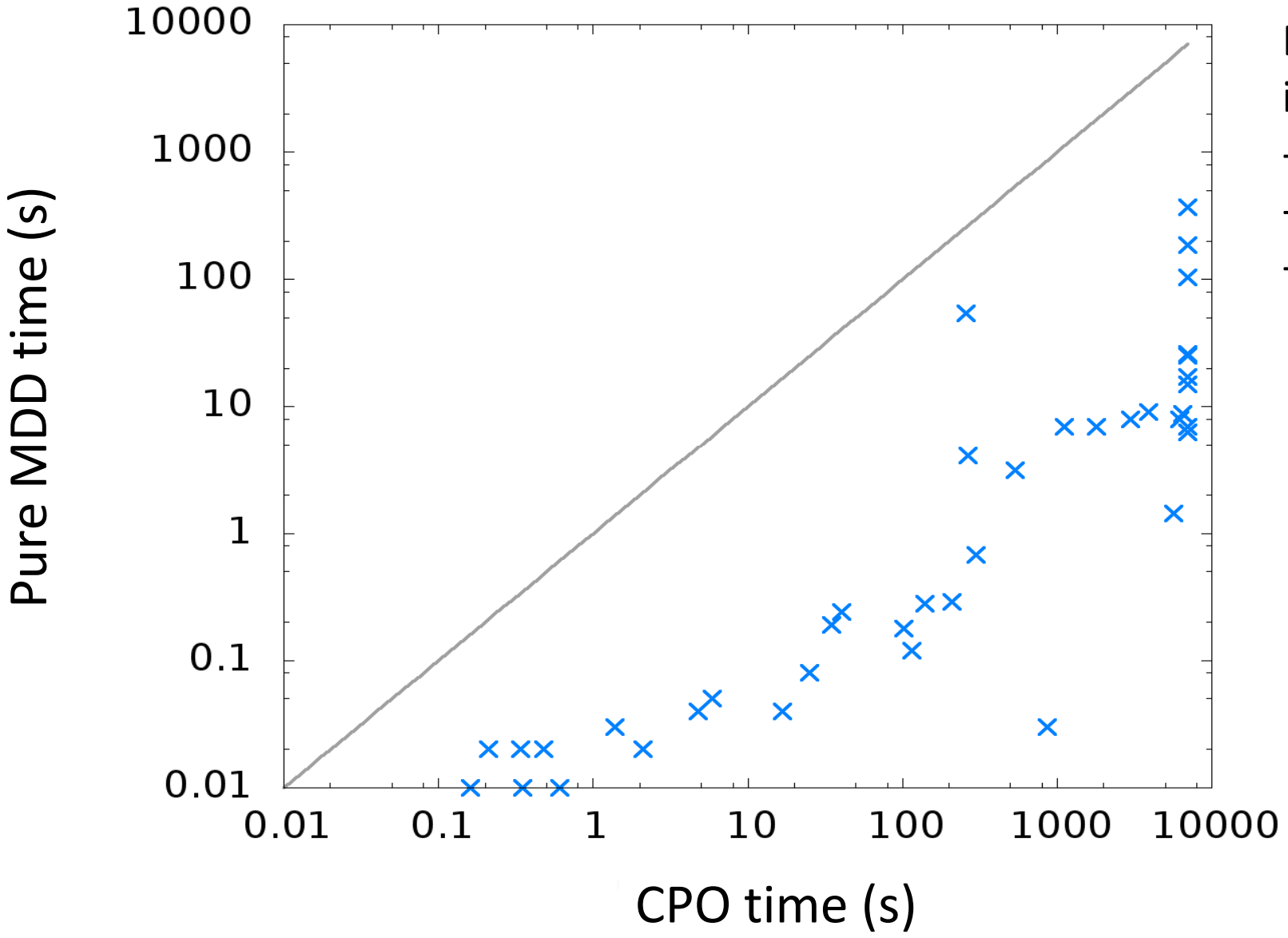
- Disjunctive instances with
  - sequence-dependent setup times
  - release dates and deadlines
  - precedence relations
- Objectives (that are presented here)
  - minimize makespan
  - minimize sum of setup times
  - minimize total tardiness
- Benchmarks
  - Random instances with varying setup times
  - TSP-TW instances (Dumas, Ascheuer, Gendreau)
  - Sequential Ordering Problem

Importance of setup times



Random instances
- 15 jobs
- lex search
- MDD width 16
- min makespan

TSP with Time Windows



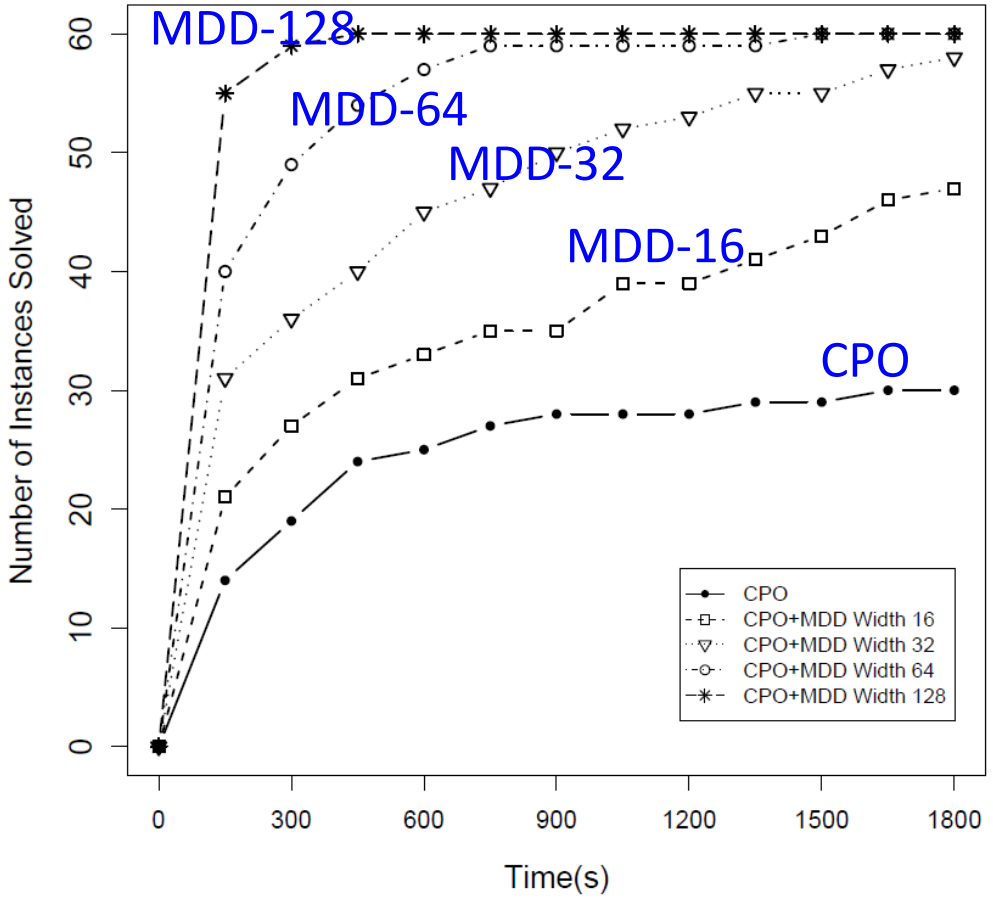
Dumas/Ascheuer instances
- 20-60 jobs
- lex search
- MDD width: 16

Minimize Total Tardiness

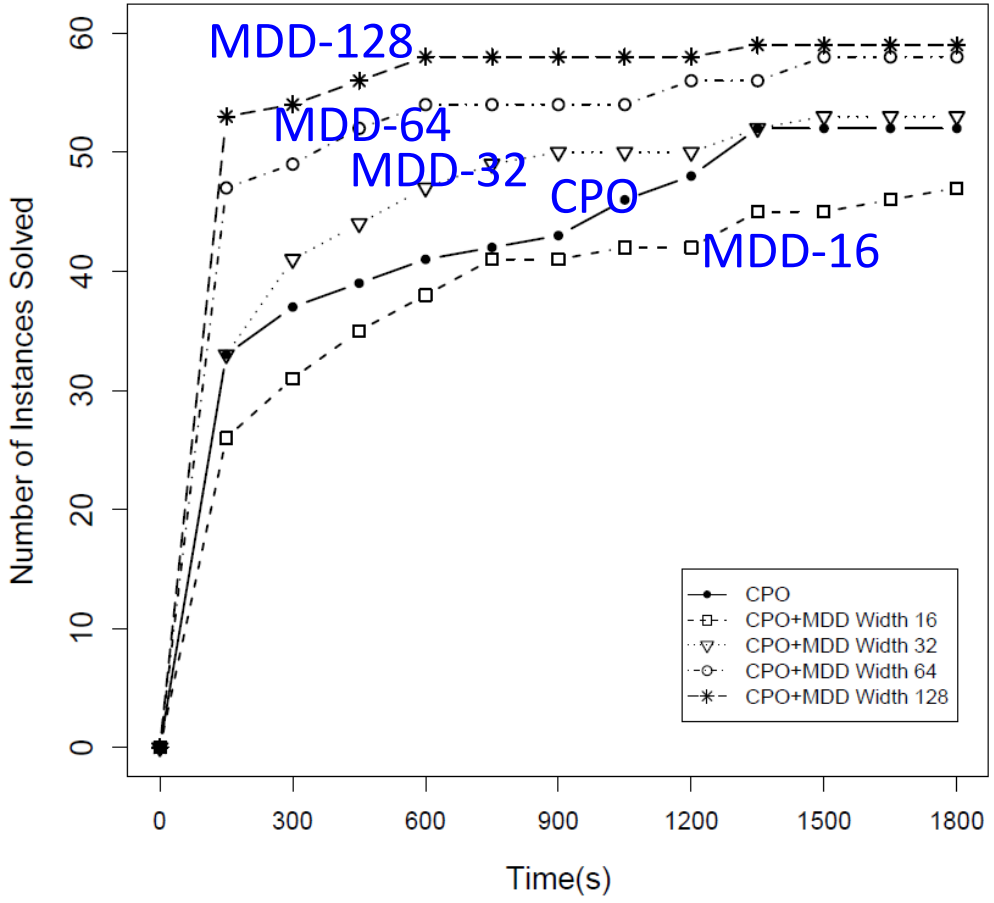
- Consider activity i with due date δ_i
  - Completion time of i: $c_i = s_i + p_i$
  - Tardiness of i: $\max\{0, c_i - \delta_i\}$
- Objective: minimize total (weighted) tardiness

- 120 test instances
  - 15 activities per instance
  - varying r_i, p_i, and δ_i, and tardiness weights
  - no side constraints, setup times (measure only impact of objective)
  - lexicographic search, time limit of 1,800s

Total Tardiness Results



total tardiness



total weighted tardiness

Sequential Ordering Problem (TSPLIB)

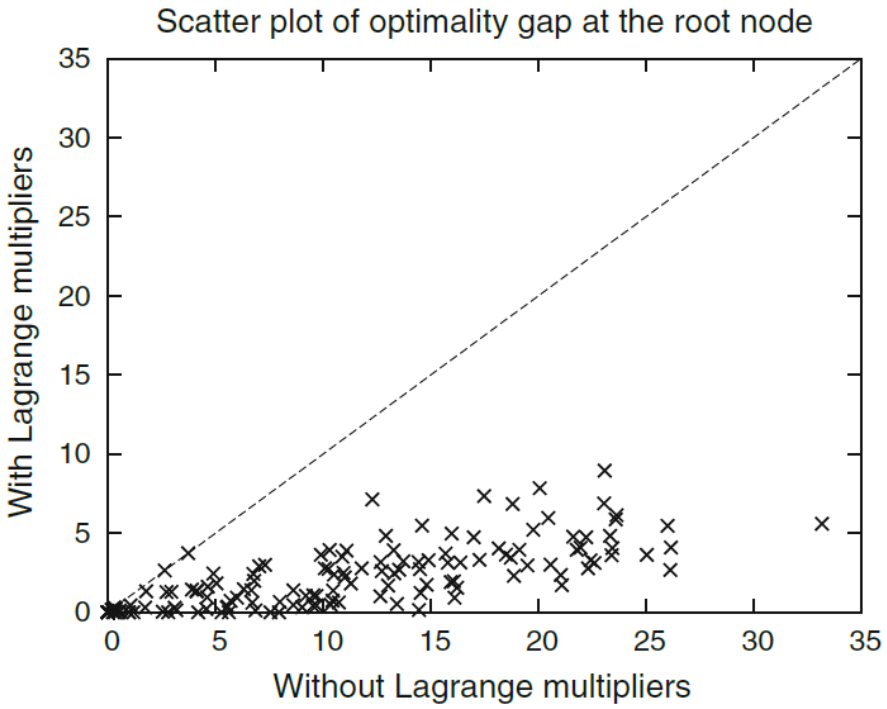
Carnegie Mellon Tepper SCHOOL OF BUSINESS

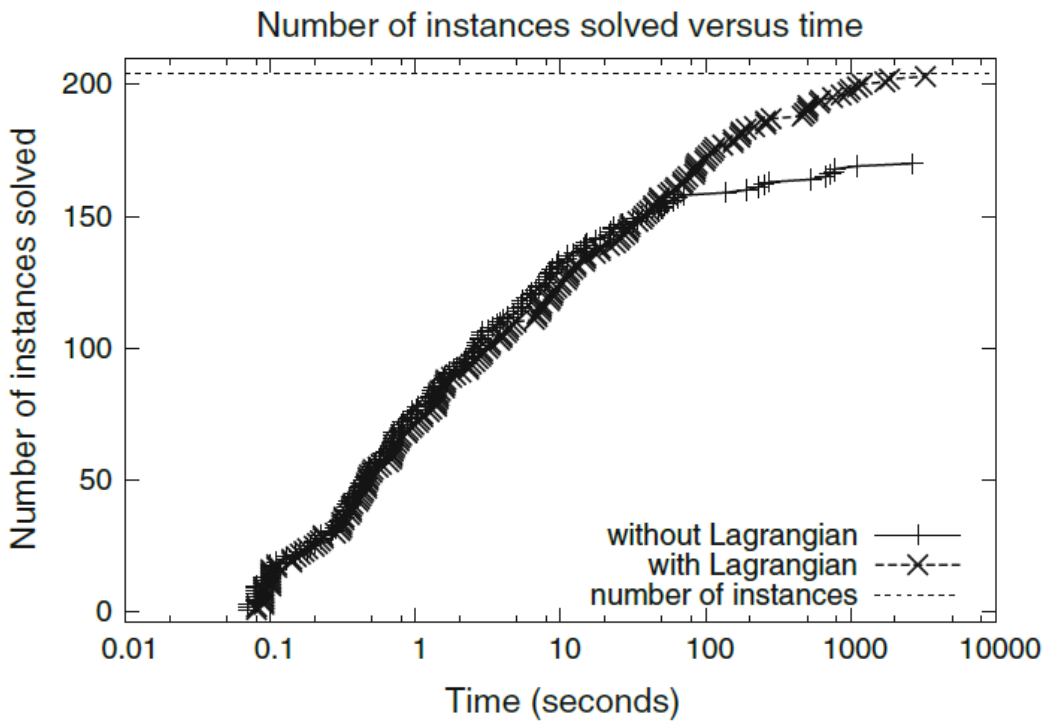
instance	vertices	bounds	CPO best	CPO time (s)	CPO+MDD, width 2048 best	CPO+MDD, width 2048 time (s)
br17.10	17	55	55	0.01	55	4.98
br17.12	17	55	55	0.01	55	4.56
ESC07	7	2125	2125	0.01	2125	0.07
ESC25	25	1681	1681	TL	1681	48.42
p43.1	43	28140	28205	TL	28140	287.57
p43.2	43	[28175, 28480]	28545	TL	**28480**	**279.18** *
p43.3	43	[28366, 28835]	28930	TL	**28835**	**177.29** *
p43.4	43	83005	83615	TL	83005	88.45
ry48p.1	48	[15220, 15805]	18209	TL	16561	TL
ry48p.2	48	[15524, 16666]	18649	TL	17680	TL
ry48p.3	48	[18156, 19894]	23268	TL	22311	TL
ry48p.4	48	[29967, 31446]	34502	TL	**31446**	**96.91** *
ft53.1	53	[7438, 7531]	9716	TL	9216	TL
ft53.2	53	[7630, 8026]	11669	TL	11484	TL
ft53.3	53	[9473, 10262]	12343	TL	11937	TL
ft53.4	53	14425	16018	TL	14425	120.79

* solved for the first time

Extensions

- Improved bounds
  - Lagrangian relaxation for violated constraints





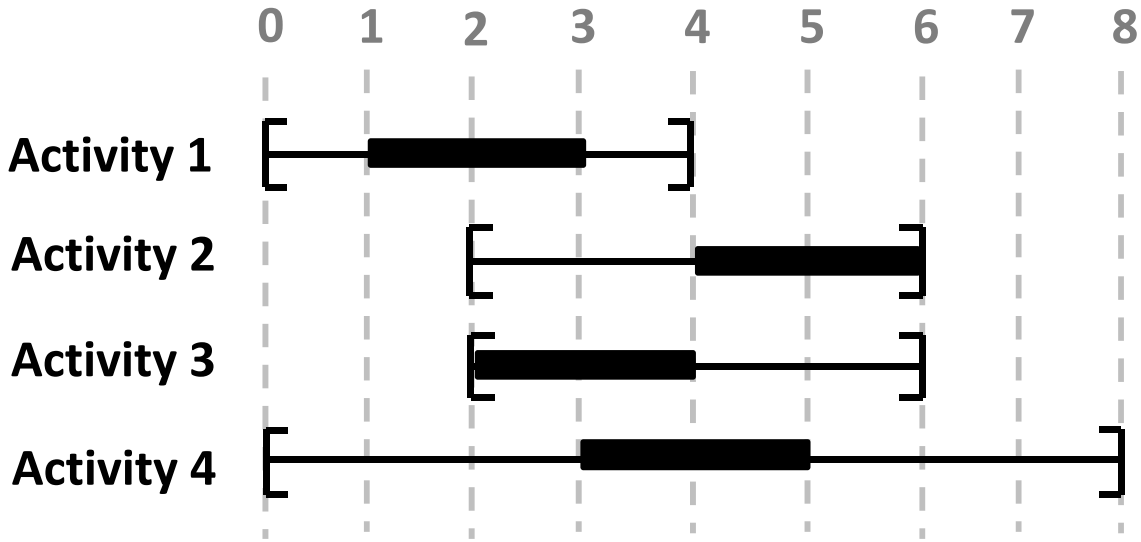
TSPTW instances (Constraints, 2015)

Conclusion

- MDD propagation natural generalization of domain propagation
  - Strength of MDD relaxation can be controlled by the width
  - Huge reduction in the amount of backtracking and solution time is possible
- For sequencing/disjunctive scheduling problems
  - MDD can handle all side constraints and objectives from existing CP systems
  - Polynomial cases (e.g., Balas variant)
  - MDD propagation algorithms (alldifferent, time windows, …)
  - Extraction of precedence constraints from MDD
  - Great addition to CP systems

5. Consider the following scheduling problem

Act	r_i	d_i	p_i
1	0	**4**	2
2	**2**	6	2
3	**2**	6	2
4	**0**	8	2



a) Create an exact MDD M representation for this problem.

b) Use the state information $A_u^{\downarrow}$ and $A_u^{\uparrow}$ to derive all precedence relations from M.

Exercises

6. Consider an arbitrary disjunctive scheduling problem, and assume we are given an exact MDD M representing all its solutions.
   a) Verify that the optimal solution to objectives ‘minimize makespan’ and ‘minimize sum of setup times’ can be derived by recursion/computing a shortest path in M.
   b) Give an example that shows that for objective ‘minimize total tardiness’, a shortest path in M provides a lower bound.