MDD Propagation for Disjunctive Scheduling

André A. Ciré and Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

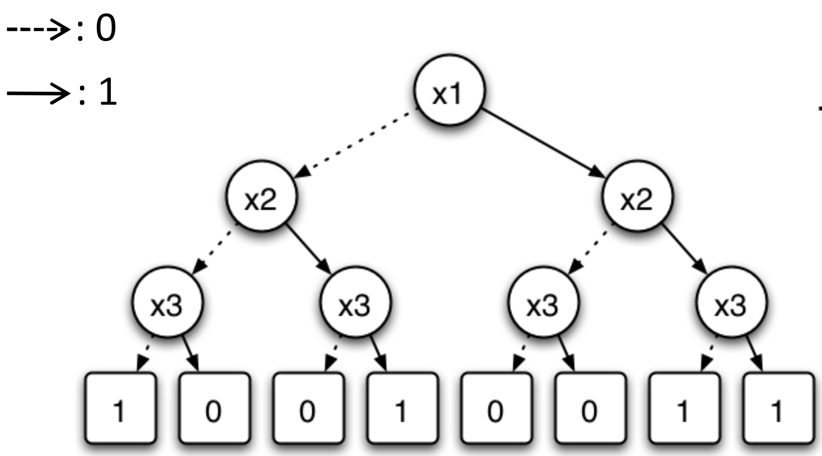
(paper at ICAPS 2012)

Acknowledgments: NSF, Google

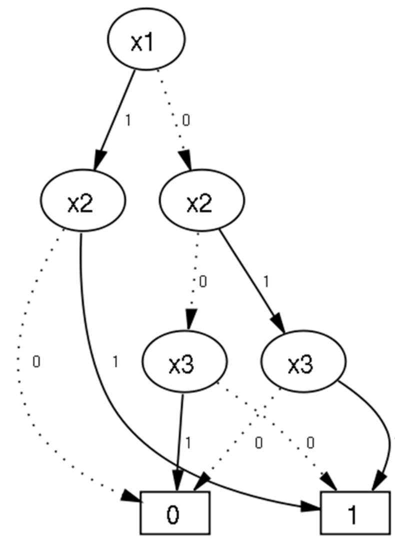
Outline

- Motivation
- Disjunctive Scheduling
- MDD representation
- Filtering and precedence relations
- Experimental results
- Conclusion

Decision Diagrams



x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



$$f(x1, x2, x3) = (\neg x1 \wedge \neg x2 \wedge \neg x3) \vee (x1 \wedge x2) \vee (x2 \wedge x3)$$

- Binary Decision Diagrams were introduced to compactly represent Boolean functions [Lee, 1959], [Akers, 1978], [Bryant, 1986]
- BDD: merge isomorphic subtrees of a given binary decision tree
- MDDs are multi-valued decision diagrams (i.e., for discrete variables)

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

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

$x_1 < x_2$

$x_2 \in \{2,3\}$

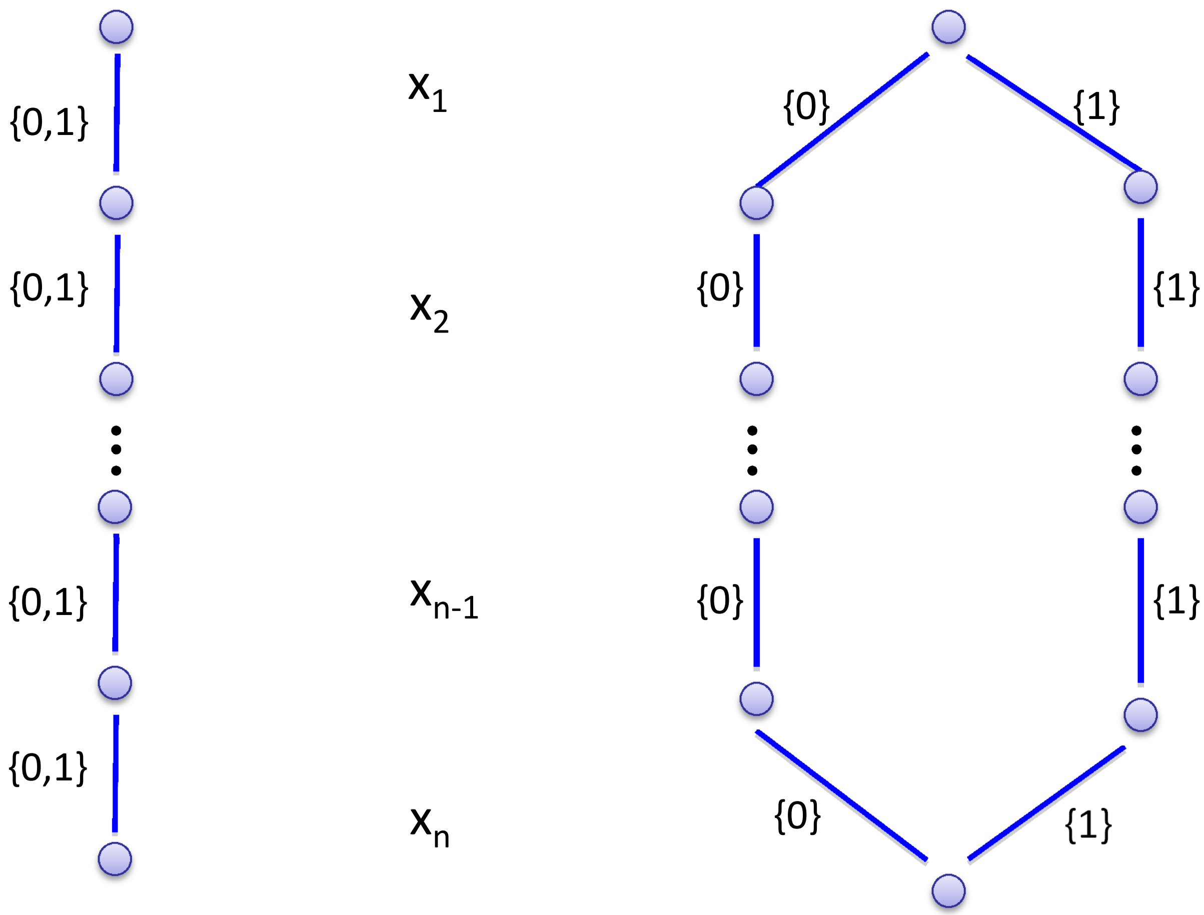
alldifferent(x_1,x_2,x_3)

$x_1 \in \{1\}$

Illustrative Example

AllEqual$(x_1, x_2,\ldots, x_n)$, all x_i binary

$x_1 + x_2 + \ldots + x_n \geq n/2$



domain representation, size 2^n

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 (can still be exponentially large)
- Limited width defines *relaxation* MDD
- Strength is controlled by the imposed width

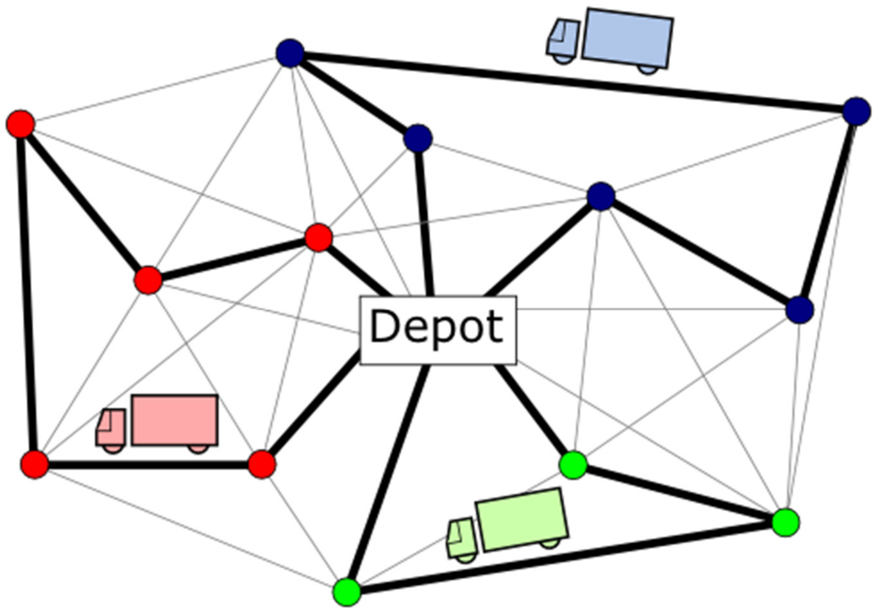
MDD-based Constraint Programming

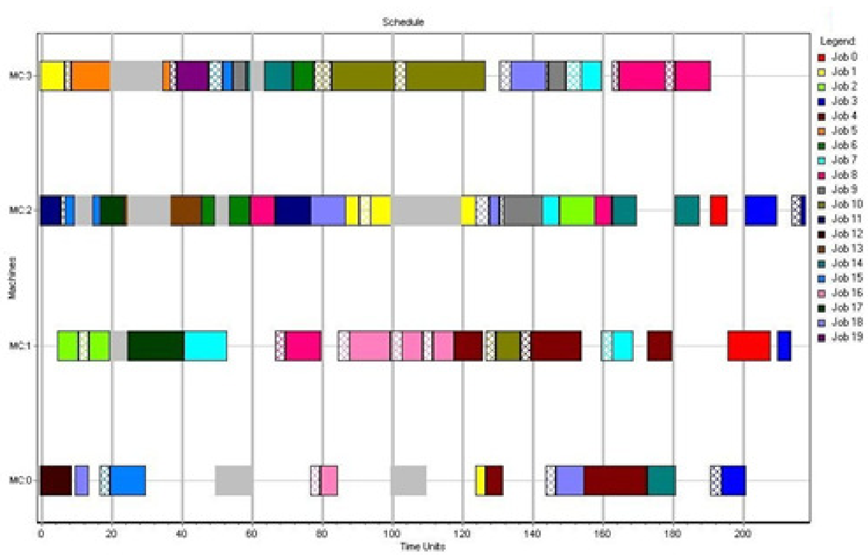
- Maintain limited-width MDD
  - Serves as relaxation
  - Typically start with width 1 (initial variable domains)
  - Dynamically adjust MDD, based on constraints
- Constraint Propagation
  - Edge filtering: Remove provably inconsistent edges (those that do not participate in any solution)
  - Node refinement: 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]
- *Sequence* constraints (combination of *Amongs*) [v.H., 2011]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]

Disjunctive Scheduling





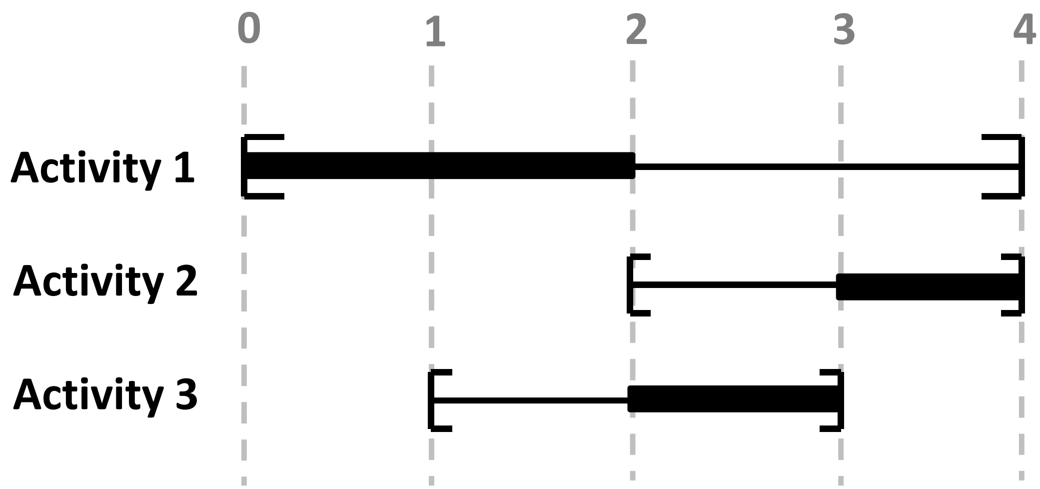
Constraint-Based Scheduling

- 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.)
  - 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)
  - setup times are present
  - …
- What can MDDs bring here?

Disjunctive Scheduling

- 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

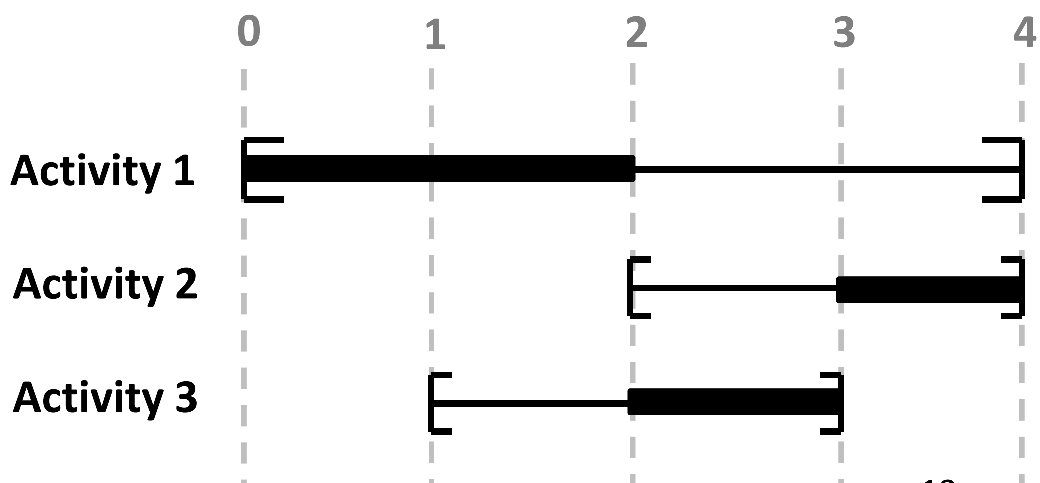
Common Side Constraints

- Precedence relations between activities
- Sequence-dependent setup times
- Induced by objective function
  - Makespan
  - Sum of setup times
  - Sum of completion times
  - 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: $1 \ll 3$

  $s_3 \geq 3$



MDD Representation

- Natural representation as 'permutation MDD'

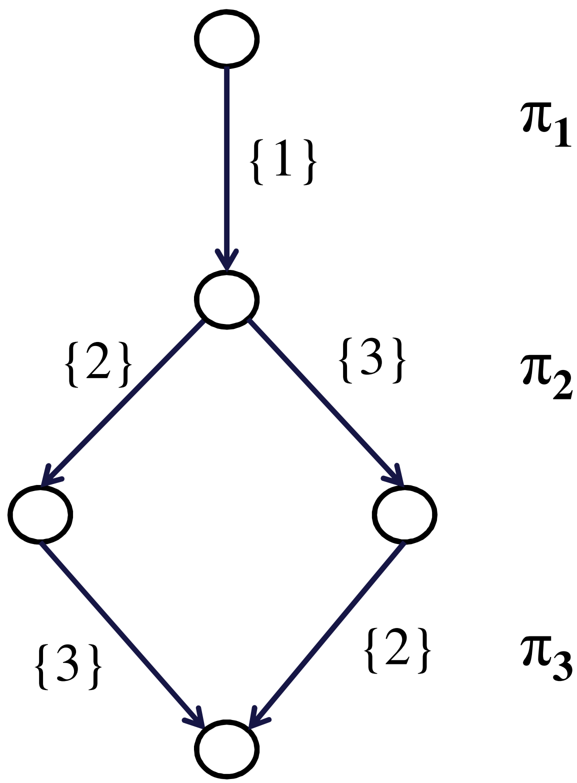
- Every solution can be written as a permutation $\boldsymbol{\pi}$

$\pi_1, \pi_2, \pi_3, \ldots, \pi_n$: 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



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

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*

Nevertheless, there are interesting restrictions, e.g. (Balas [99]):

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

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

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

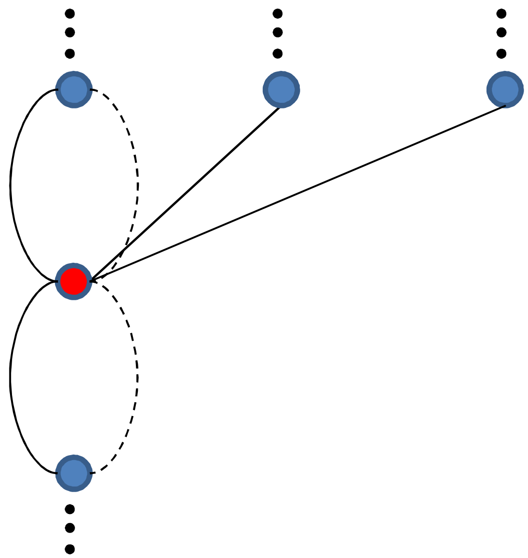
Relaxed MDD Propagation

We can apply several propagation algorithms to the relaxed MDD

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

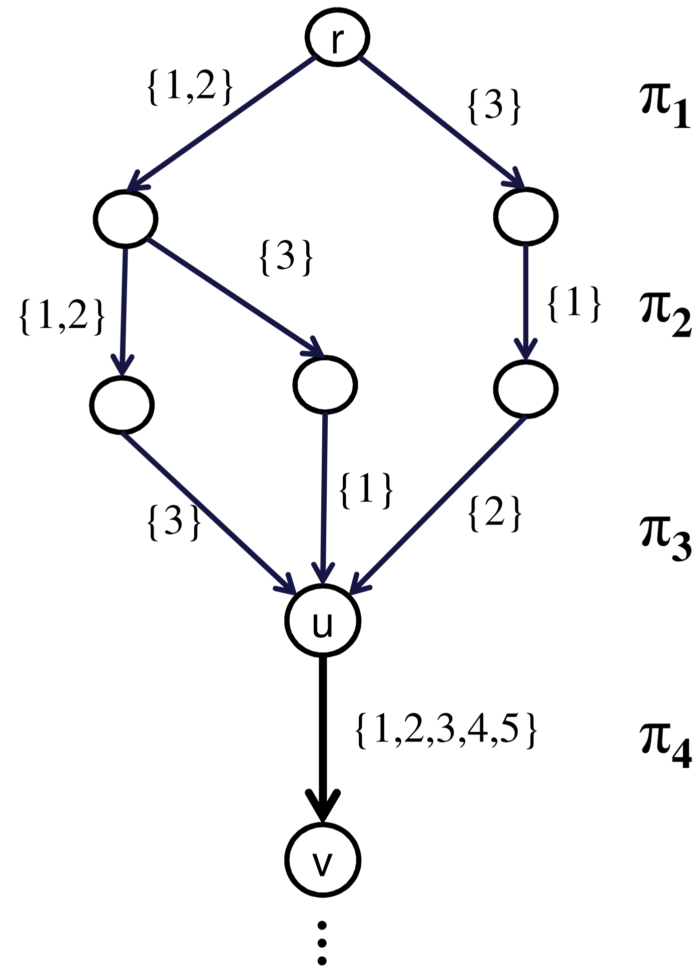
Constraint Representation in MDDs

- For a given constraint type we maintain specific 'state information' at each node in the MDD
- Computed from incoming arcs (both from top and from bottom)
- State information is basis for MDD *filtering* and for MDD *refinement*



Propagation for disjunctive

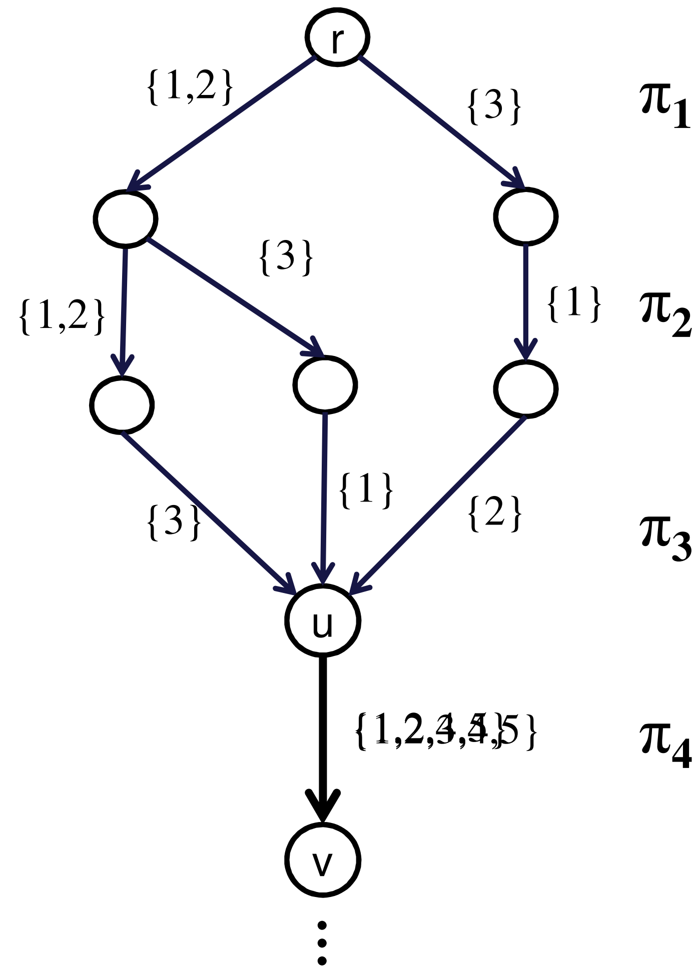
- State information at each node *i*
  - labels on *all* paths: A_i
  - labels on *some* paths: S_i
  - earliest starting time: E_i
  - latest completion time: L_i
- Top down example for arc (u,v)



Alldifferent Propagation

Carnegie Mellon Tepper School of Business

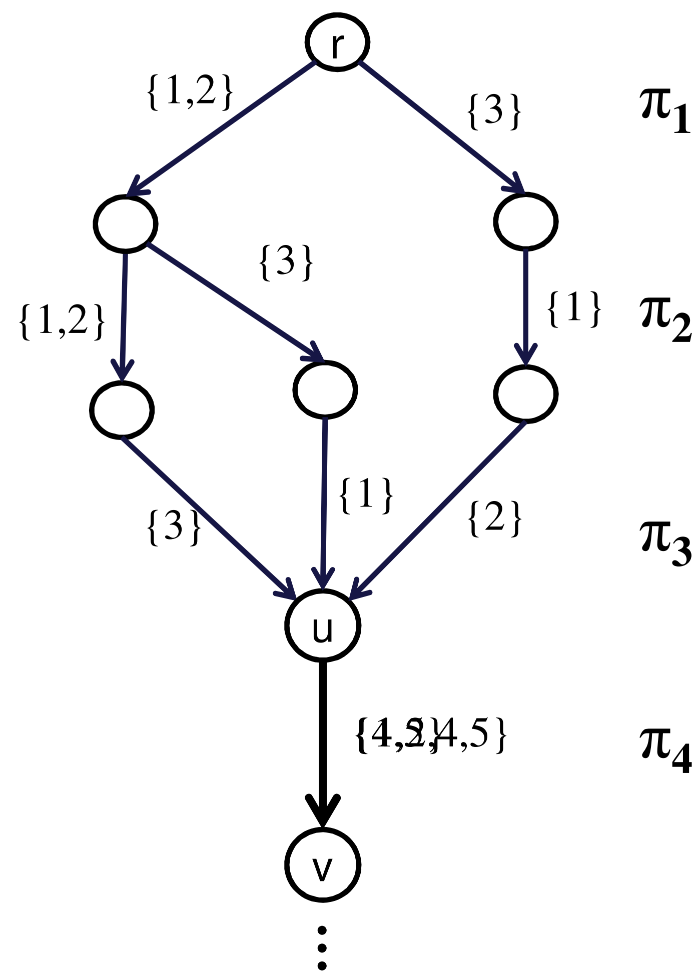
- All-paths state: A_u
  - Labels belonging to all paths from node r to node u
  - $A_u = \{3\}$
  - Thus eliminate {3} from (u,v)



[Andersen et al., 2007]

Alldifferent Propagation

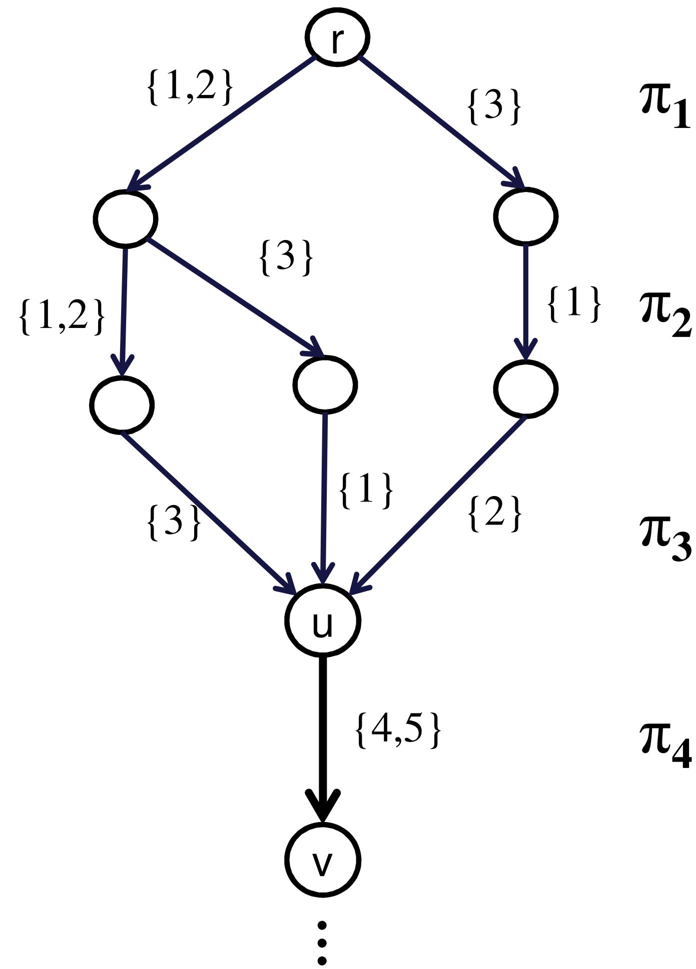
- Some-paths state: S_u
  - Labels belonging to some path from node r to node u
  - $S_u = \{1,2,3\}$
  - Identification of Hall sets
  - Thus eliminate {1,2,3} from (u,v)



Propagate Earliest Completion Time

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

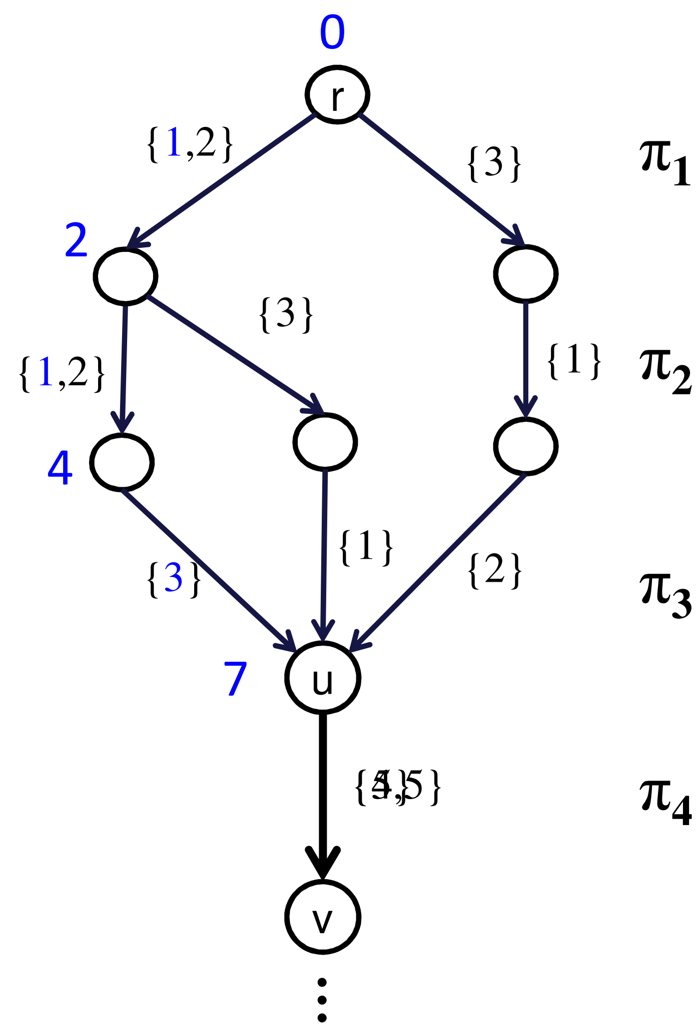
- Similarly: Latest Completion Time



Propagate Earliest Completion Time

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

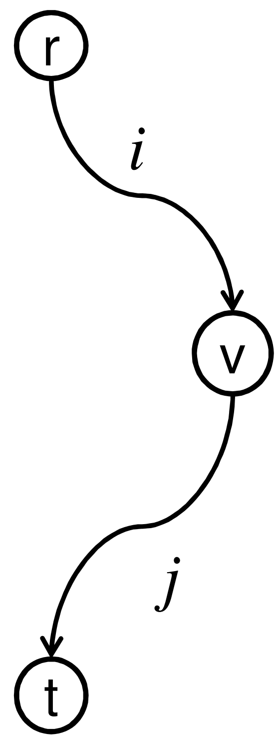
- $E_u = 7$
- Eliminate 4 from (u,v)



More MDD Inference

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

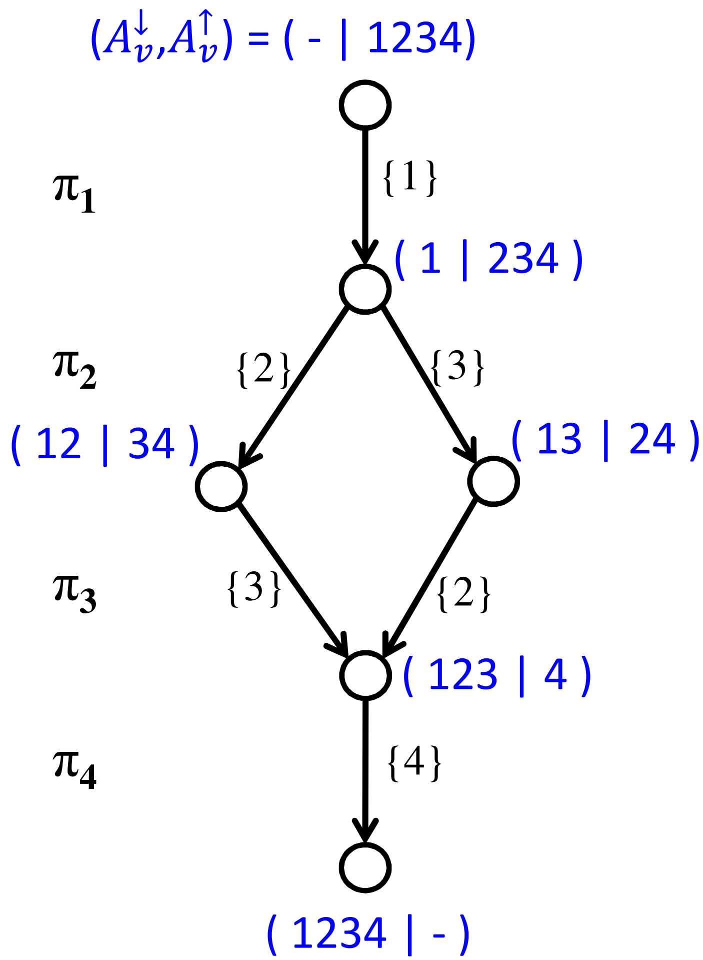
- For a node *u*,
  - $A_u^{\downarrow}$: values in all paths from root to *u*
  - $A_u^{\uparrow}$: values in all paths from node u to terminal

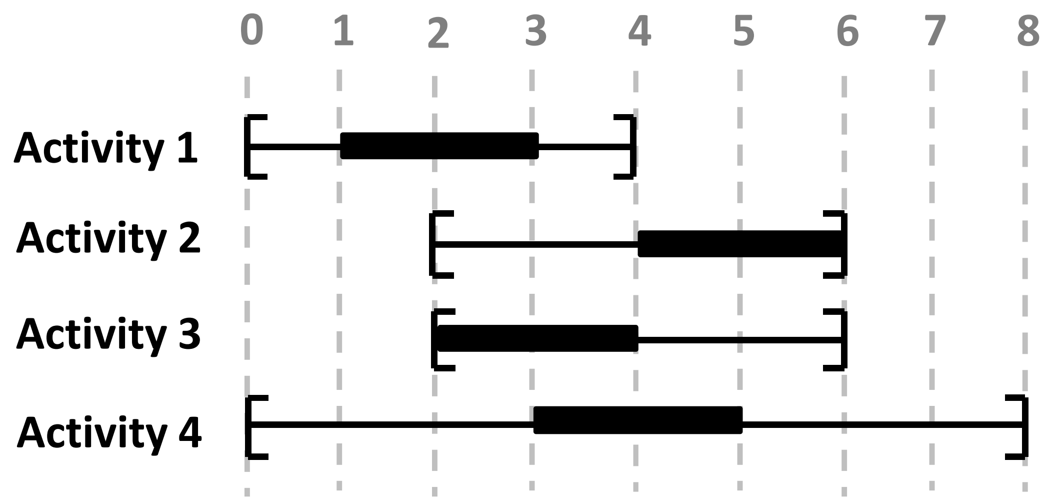


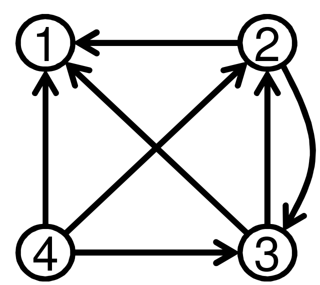
- *Precedence relation i ≪ j holds if and only if* $(j \notin A_u^{\downarrow})$ or $(i \notin A_u^{\uparrow})$ *for all nodes u in M*

- Same technique applies to relaxed MDD: use $S_u^{\downarrow}$ and $S_u^{\uparrow}$

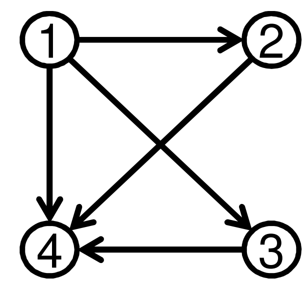
Precedence relations: example







$\bar{G}$



G

Arc (i,j) in $\bar{G}$ if $j \in A_u^{\downarrow}$ and $i \in A_u^{\uparrow}$
for *some* node *u* in *M*

$O(n^2|M|)$ time

Communicate Precedence Relations

1. Provide precedence relations from MDD to CP
   - update start/end time variables
   - other inference techniques may utilize them

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

MDD Refinement

- For refinement, we generally want to identify equivalence classes among nodes in a layer
- Theorem:

  Let M represent a Disjunctive Instance. Deciding if two nodes u and v in M are equivalent 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)

Experiments

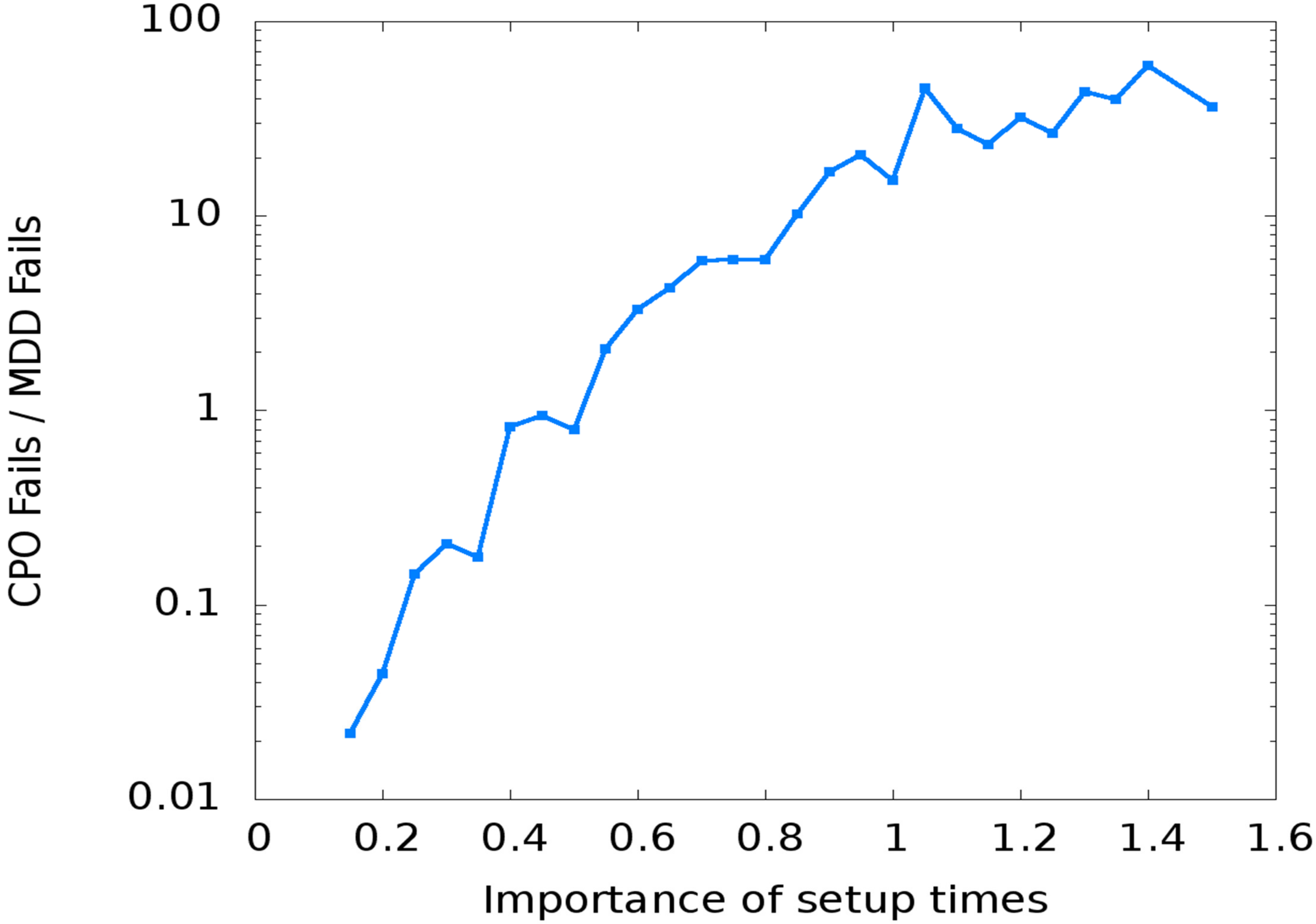
- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation

- Main purpose of experiments
  - When can MDDs strengthen CP
  - Compare stand-alone MDD versus CP
  - Compare CP versus CP+MDD (most practical)

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
- Benchmarks
  - Random instances with varying setup times
  - TSP-TW instances (Dumas, Ascheuer, Gendreau)
  - Sequential Ordering Problem

Test 1: Importance of setup times



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

Test 2: Minimize Makespan

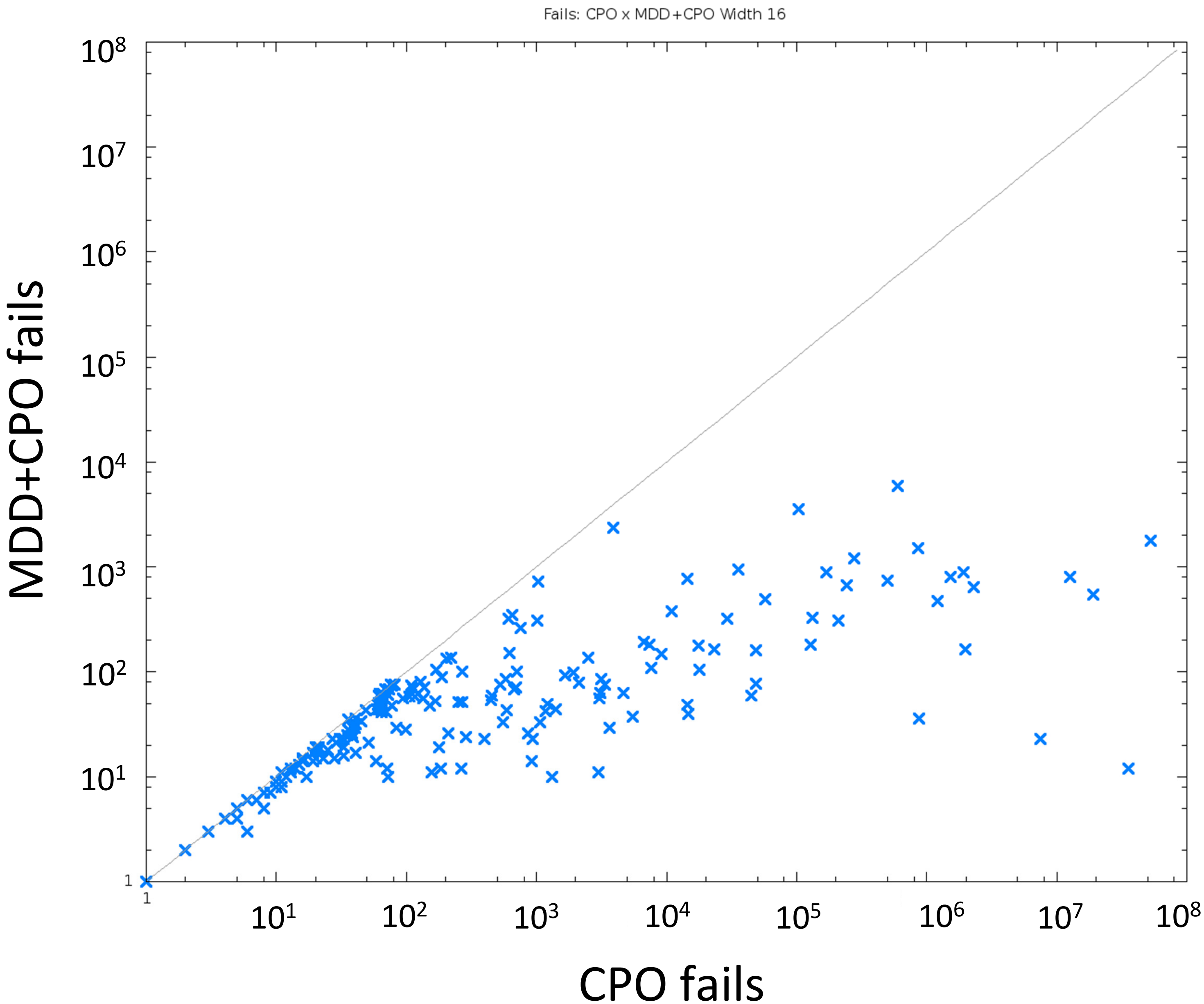
- 229 TSPTW instances with up to 100 jobs
- Minimize makespan
- Time limit 7,200s
- Max MDD width is 16

instances solved by CP: 211

instances solved by pure MDD: 216

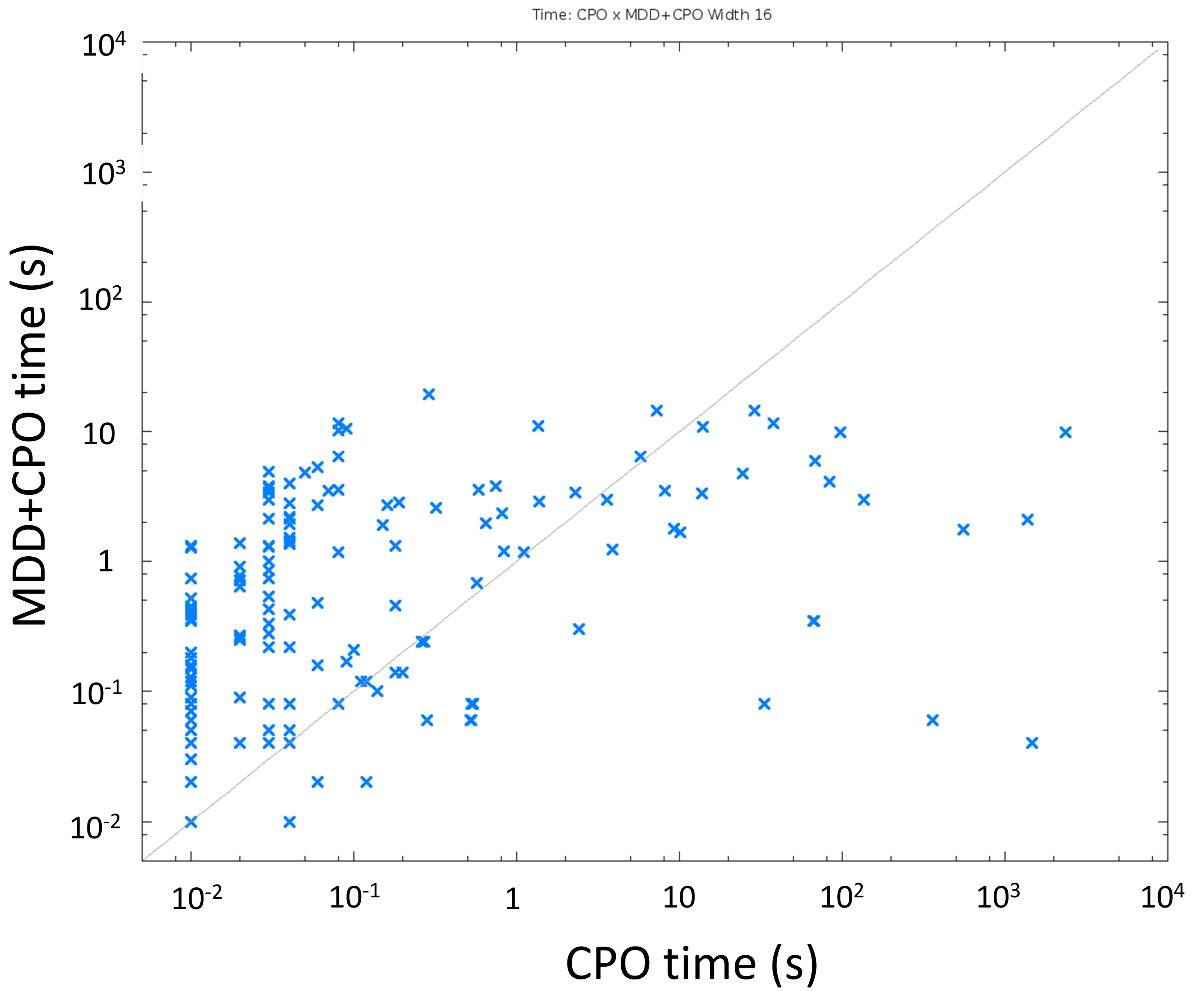
instances solved by CP+MDD: 225

Minimize Makespan: Fails

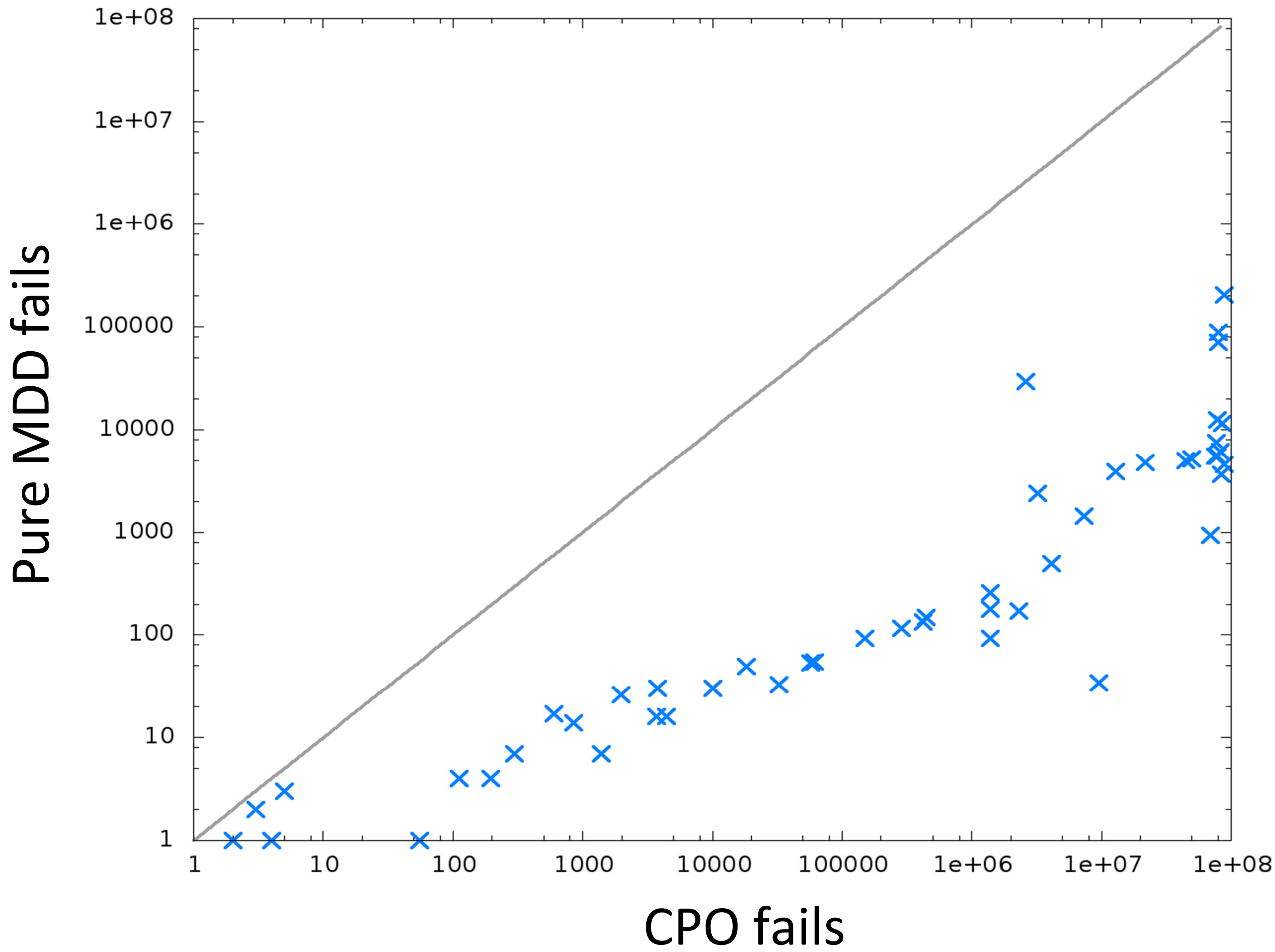


plot only on instances that were solved by all methods

Minimize Makespan: Time

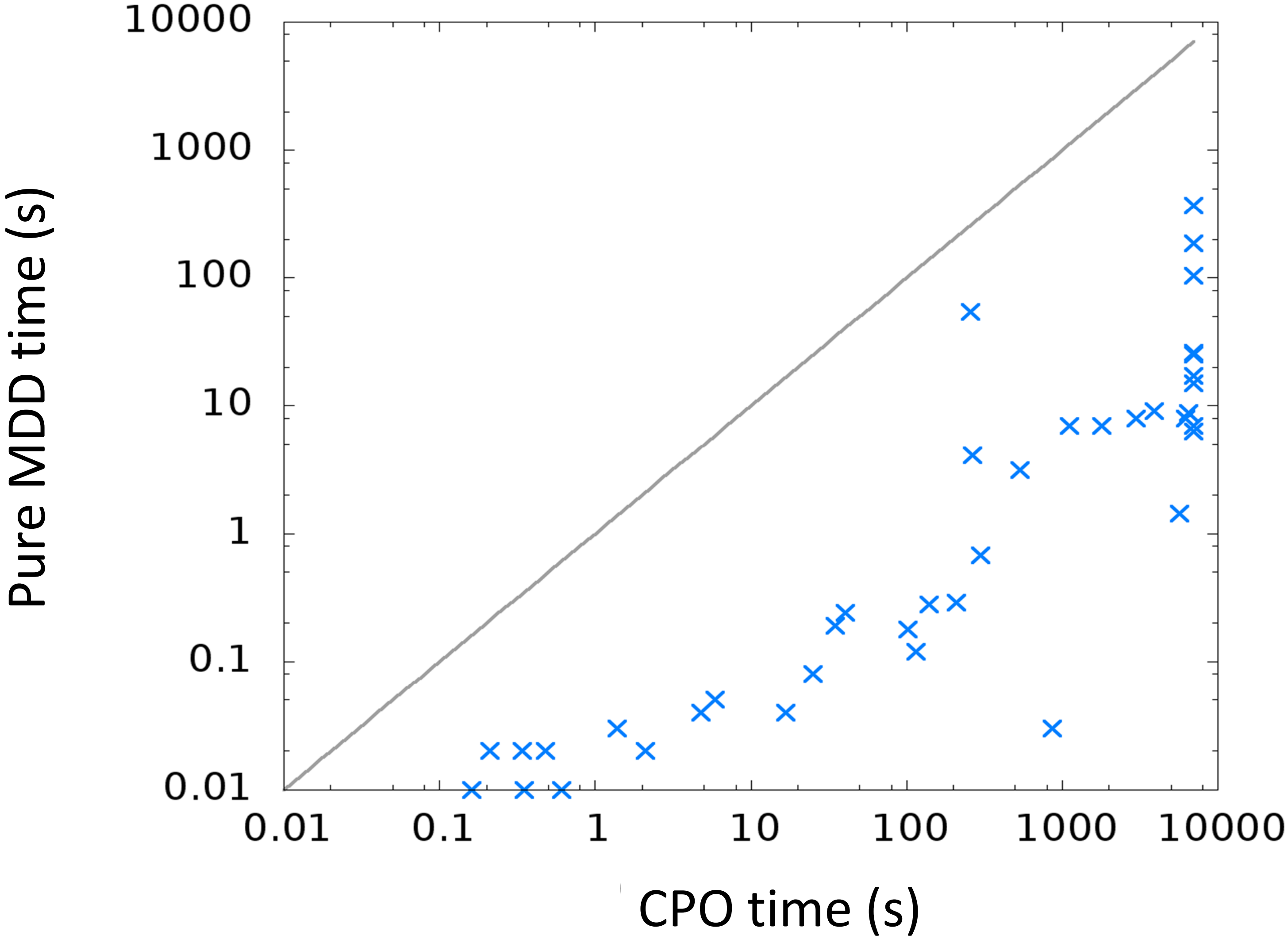


Min sum of setup times: Fails



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

Min sum of setup times: Time



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

Instances Dumas (TSPTW)

		CPO		CPO+MDD	
Instance	**Cities**	**Backtracks**	**Time (s)**	**Backtracks**	**Time (s)**
n40w40.004	40	480,970	50.81	18	0.06
n60w20.001	60	908,606	199.26	50	0.22
n60w20.002	60	84,074	14.13	46	0.16
n60w20.003	60	> 22,296,012	> 3600	99	0.32
n60w20.004	60	2,685,255	408.34	97	0.24

minimize sum of setup times

MDDs have maximum width 16

Sequential Ordering Problem

- TSP with precedence constraints (no time windows)

- Instances up to 53 jobs
- Time limit 1,800s
- CPO: default search
- MDD+CPO: search guided by MDD (shortest path)
- Max MDD width 2,048

Sequential Ordering Problem Results

		CPO		MDD+CPO	
Instance	Known Bounds	Best Solution	Time	Best Solution	Time
br17.10.sop	55	55	TL	**55**	4.64
br17.12.sop	55	55	TL	**55**	4.29
ESC07.sop	2125	**2125**	0	2125	0.07
ESC11.sop	2075	2075	TL	**2075**	1.25
ESC12.sop	1675	1675	TL	**1675**	1.48
ESC25.sop	1681	1747	TL	**1681**	34.89
ESC47.sop	1288	2044	TL	**1776**	TL
ft53.1.sop	[7438,7531]	**8028**	TL	10376	TL
ft53.2.sop	[7630,8335]	**8774**	TL	11498	TL
ft53.3.sop	[9473,10935]	**10709**	TL	11133	TL
ft53.4.sop	14425	14504	TL	**14425**	154.3
p43.1.sop	27990	28230	TL	**28140**	420.3
p43.2.sop	[28175,28330]	28480	TL	**28480**	776.67
p43.3.sop	[28366,28680]	28855	TL	**28835**	251.4
p43.4.sop	83005	nosol	TL	**83005**	44.73
prob.42.sop	243	302	TL	**256**	TL
rbg048a.sop	351	**351**	TL	386	TL
ry48p.1.sop	[15220,15805]	**16940**	TL	17633	TL
ry48p.2.sop	[15524,16666]	**18153**	TL	**18153**	TL
ry48p.3.sop	[18156,19894]	**21116**	TL	22382	TL
ry48p.4.sop	[29967,31446]	31522	TL	**31446**	112.67

*** CP improved bound** (ft53.3.sop)

*** closed by MDD** (p43.2.sop)

*** closed by MDD** (p43.3.sop)

*** closed by MDD** (ry48p.4.sop)

Summary

- MDDs can provide substantial advantage over traditional domains for constraint propagation
  - Strength of MDD can be controlled by the width
  - Huge reduction in the amount of backtracking and solution time is possible