Constraint Programming with Decision Diagrams

Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

Acknowledgments:

David Bergman, Andre A. Cire, Sam Hoda, and John N. Hooker

NSF, Google

Summary

What can MDDs do for discrete optimization?

- *Compact representation* of all solutions to a problem
- Limit on size gives *approximation*
- Control strength of approximation by size limit

MDDs for Constraint Programming

- MDD propagation natural generalization of domain propagation
- Orders of magnitude improvement possible

MDDs for optimization (CP/ILP/MINLP)

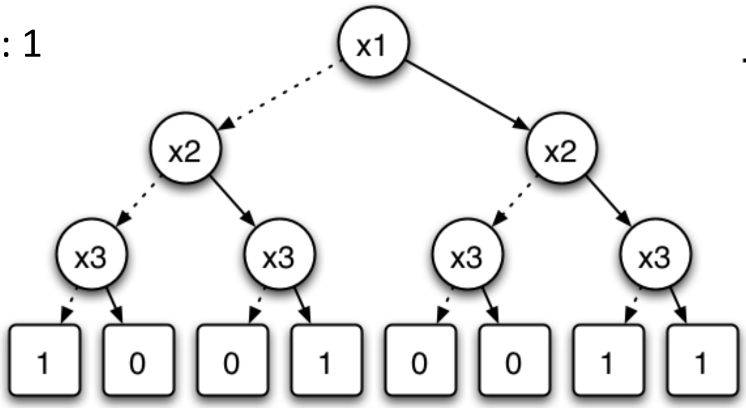
- MDDs provide *discrete relaxations*
- Much stronger bounds can be obtained in much less time

Many opportunities: search, stochastic programming, integrated methods, theory, …

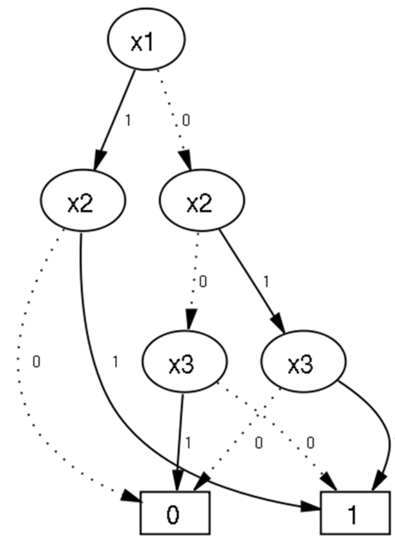
Decision Diagrams

---->: 0

⟶: 1



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)

Brief background

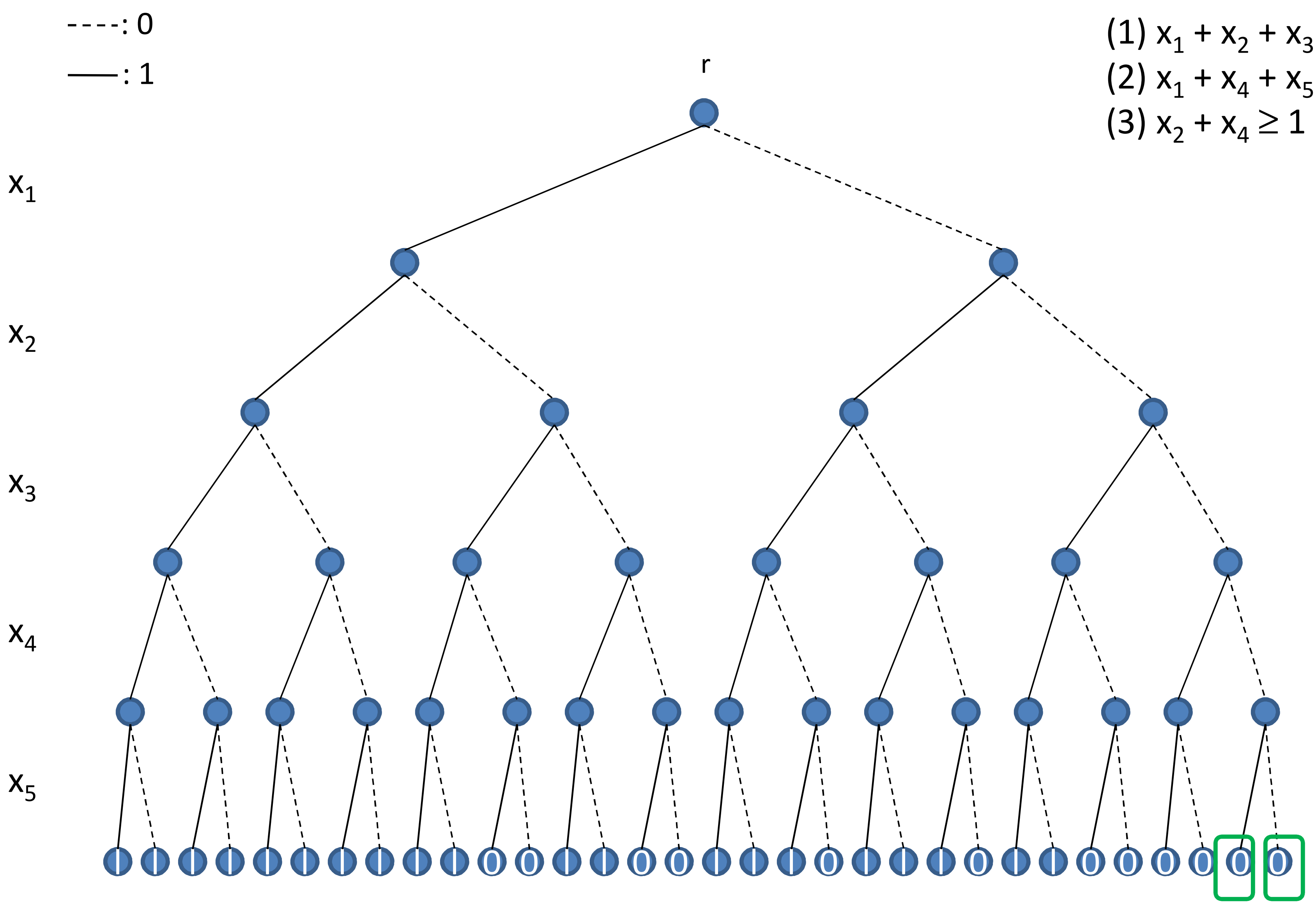
- Original application areas: circuit design, verification
- Usually *reduced ordered* BDDs/MDDs are applied
  - fixed variable ordering
  - minimal exact representation
- Recent interest from optimization community
  - cut generation [Becker et al., 2005]
  - 0/1 vertex and facet enumeration [Behle & Eisenbrand, 2007]
  - post-optimality analysis [Hadzic & Hooker, 2006, 2007]
  - set bounds propagation [Hawkins, Lagoon, Stuckey, 2005]
- Interesting variant
  - approximate MDDs

    [H.R. Andersen, T. Hadzic, J.N. Hooker, & P. Tiedemann, CP 2007]

Exact MDDs for discrete optimization

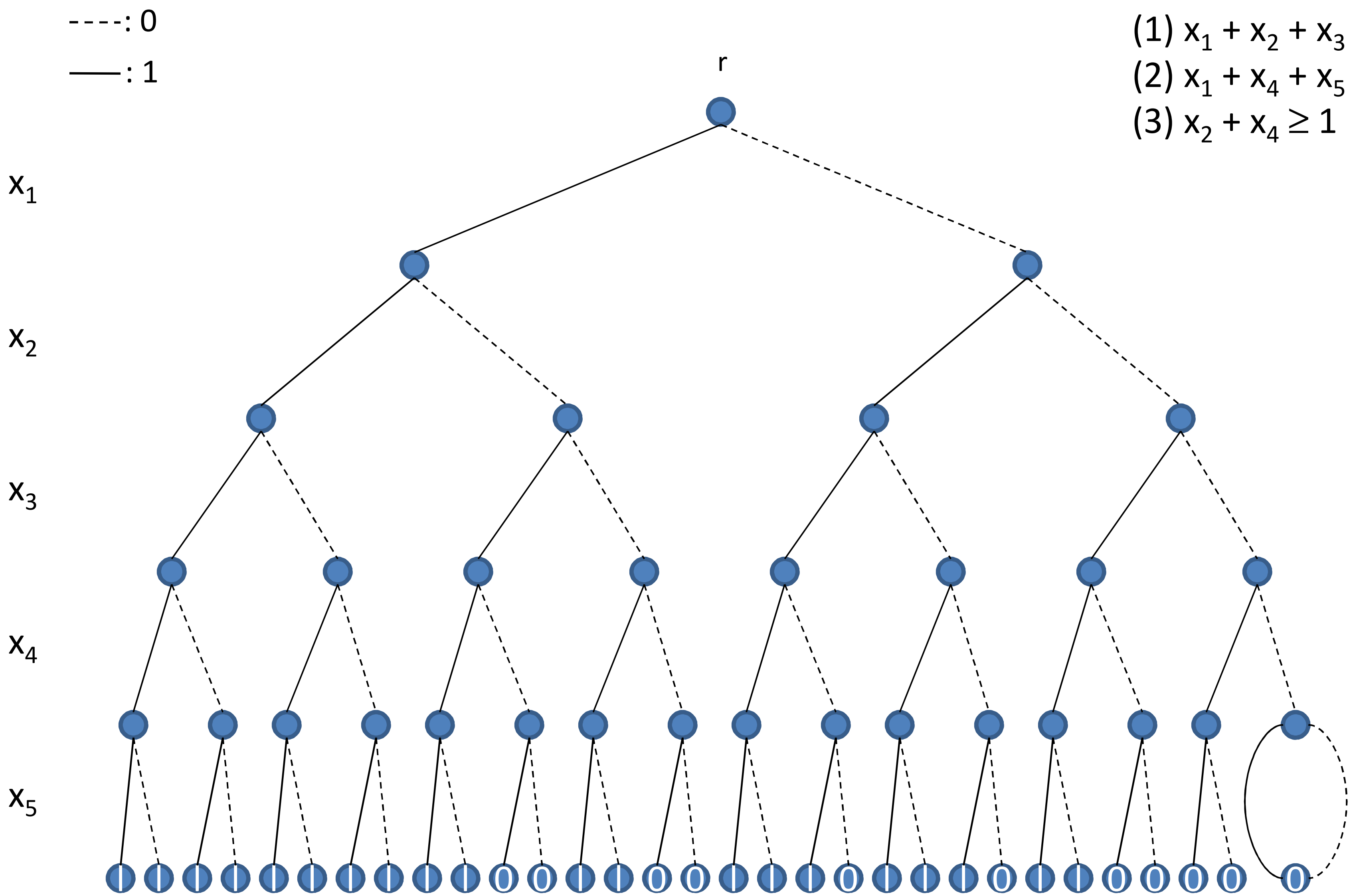
Carnegie Mellon Tepper SCHOOL OF BUSINESS

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$



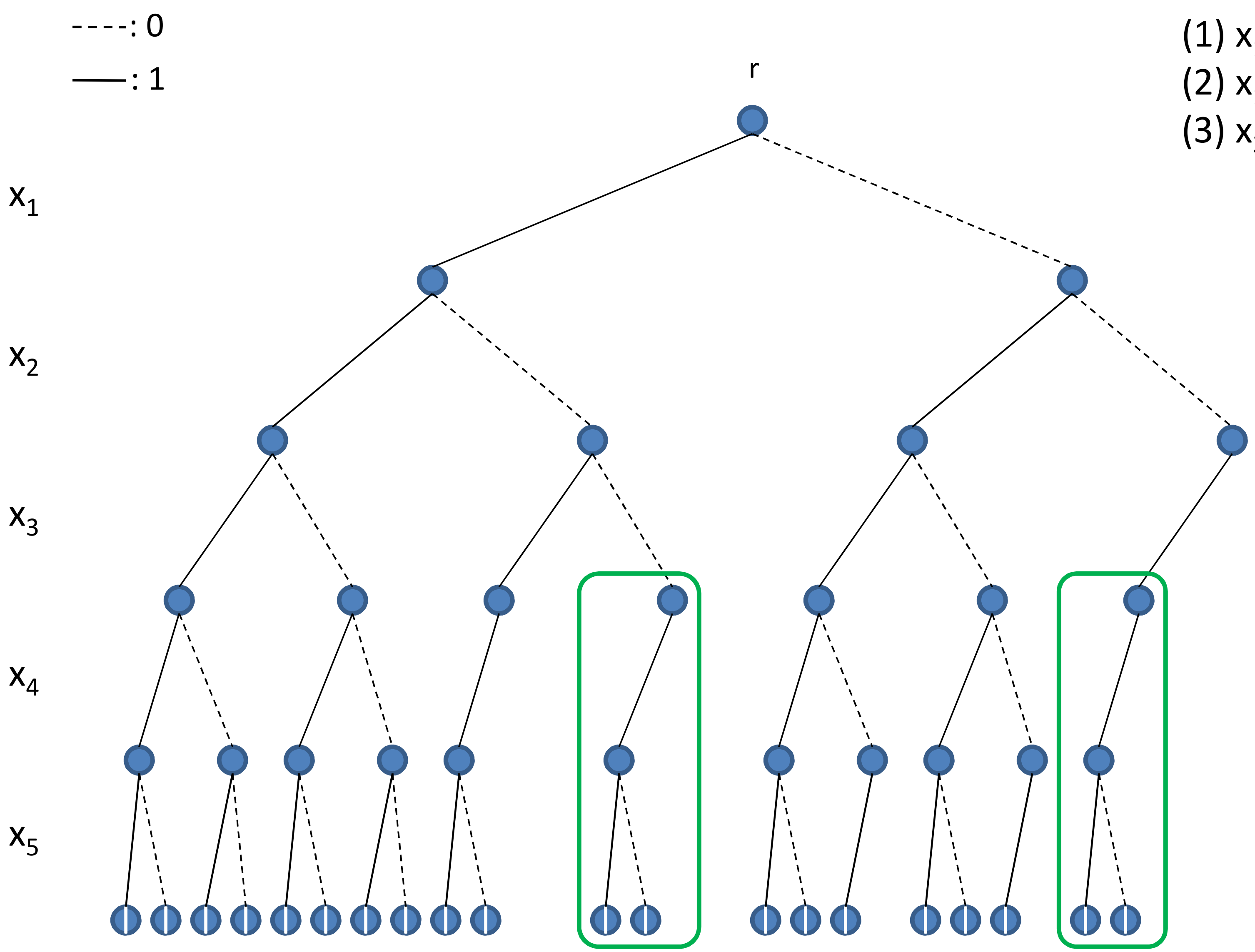
Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$



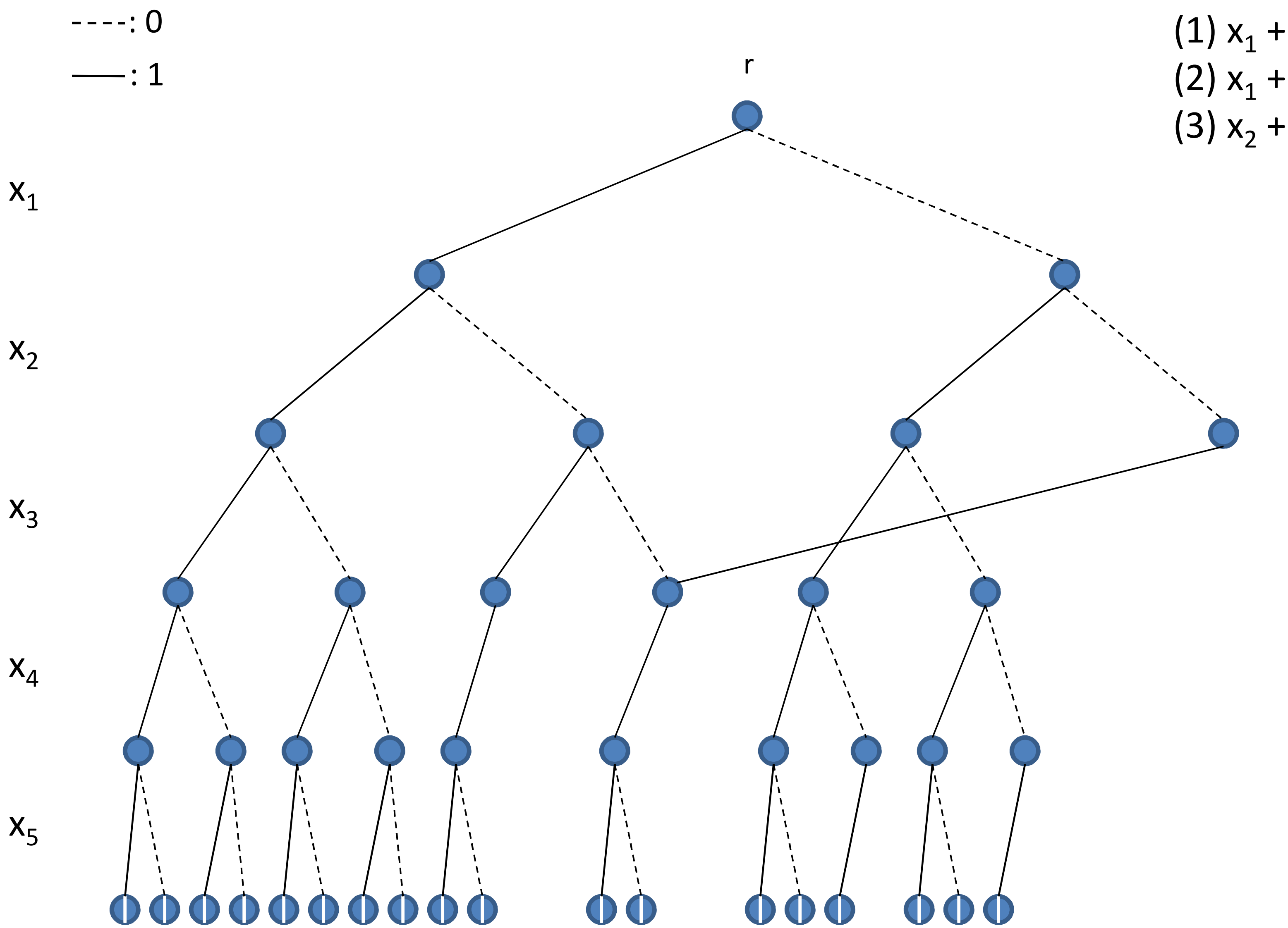
Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$

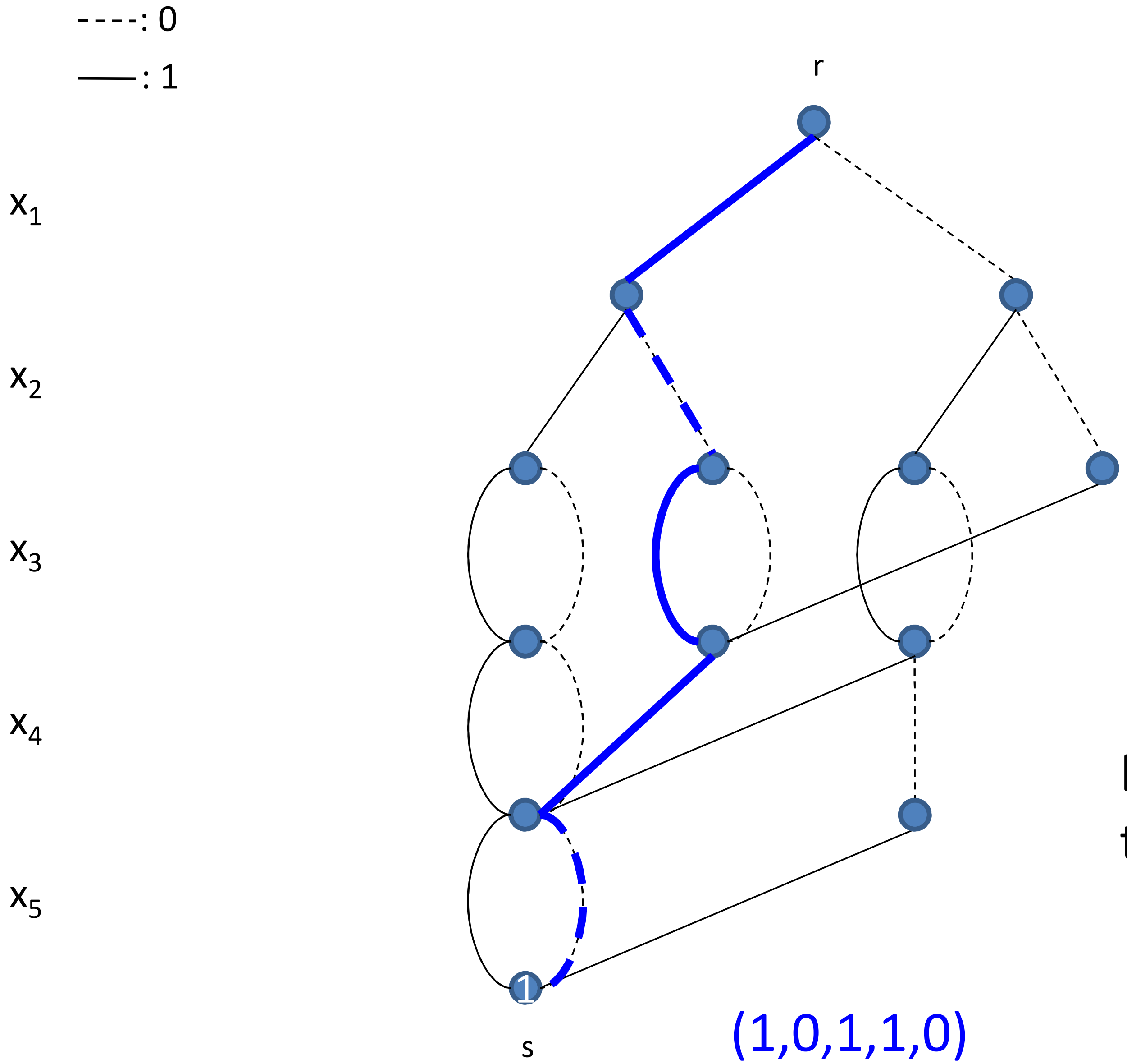


Exact MDDs for discrete optimization

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$



Exact MDDs for discrete optimization



(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$

Each path corresponds to a solution

Approximate MDDs

- Exact MDDs can be of exponential size in general
- Can we limit the size of the MDD and still have a meaningful representation?
  - Yes, first proposed by Andersen et al. [2007] :

    Limit the *width* of the MDD (the maximum number of nodes on any layer)

- Approximate MDDs: main focus of this talk

References

Related Work: Exact MDDs for constraint propagation

1. Set bounds propagation [Hawkins, Lagoon, Stuckey, 2005], [Gange, Lagoon, Stuckey, 2008]
2. Ad-hoc Table constraints [Cheng and Yap, 2008]
3. Regular constraint [Cheng, Xia, Yap, 2012]
4. Market Split Problem [Hadzic et al., 2009]

MDD-Based Constraint Programming

5. Andersen, Hadzic, Hooker, Tiedemann: A Constraint Store Based on Multivalued Decision Diagrams. CP 2007: 118-132
6. Hadzic, Hooker, O'Sullivan, Tiedemann: Approximate Compilation of Constraints into Multivalued Decision Diagrams. CP 2008: 448-462
7. Hoda, v.H., Hooker: A Systematic Approach to MDD-Based Constraint Programming. CP 2010: 266-280

References (cont'd)

Specific MDD Propagation Algorithms

8. Hadzic, Hooker, Tiedemann: Propagating Separable Equalities in an MDD Store. CPAIOR 2008: 318-322
9. Ciré, v.H.: MDD Propagation for Disjunctive Scheduling. ICAPS 2012
10. Ciré, v.H.: MDD Propagation for Sequence Constraints. Tepper School of Business Working Paper 2011-E12, Carnegie Mellon University, 2011

MDD-Based Optimization

11. Bergman, v.H., Hooker: Manipulating MDD Relaxations for Combinatorial Optimization. CPAIOR 2011: 20-35
12. Bergman, Ciré, v.H., Hooker: Variable Ordering for the Application of BDDs to the Maximum Independent Set Problem. CPAIOR 2012: 34-49
13. Bergman, Ciré, v.H., Hooker: Optimization Bounds from Binary Decision Diagrams. Tepper School of Business Working Paper 2012-E15, Carnegie Mellon University, 2012

MDDs for Constraint Programming

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

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

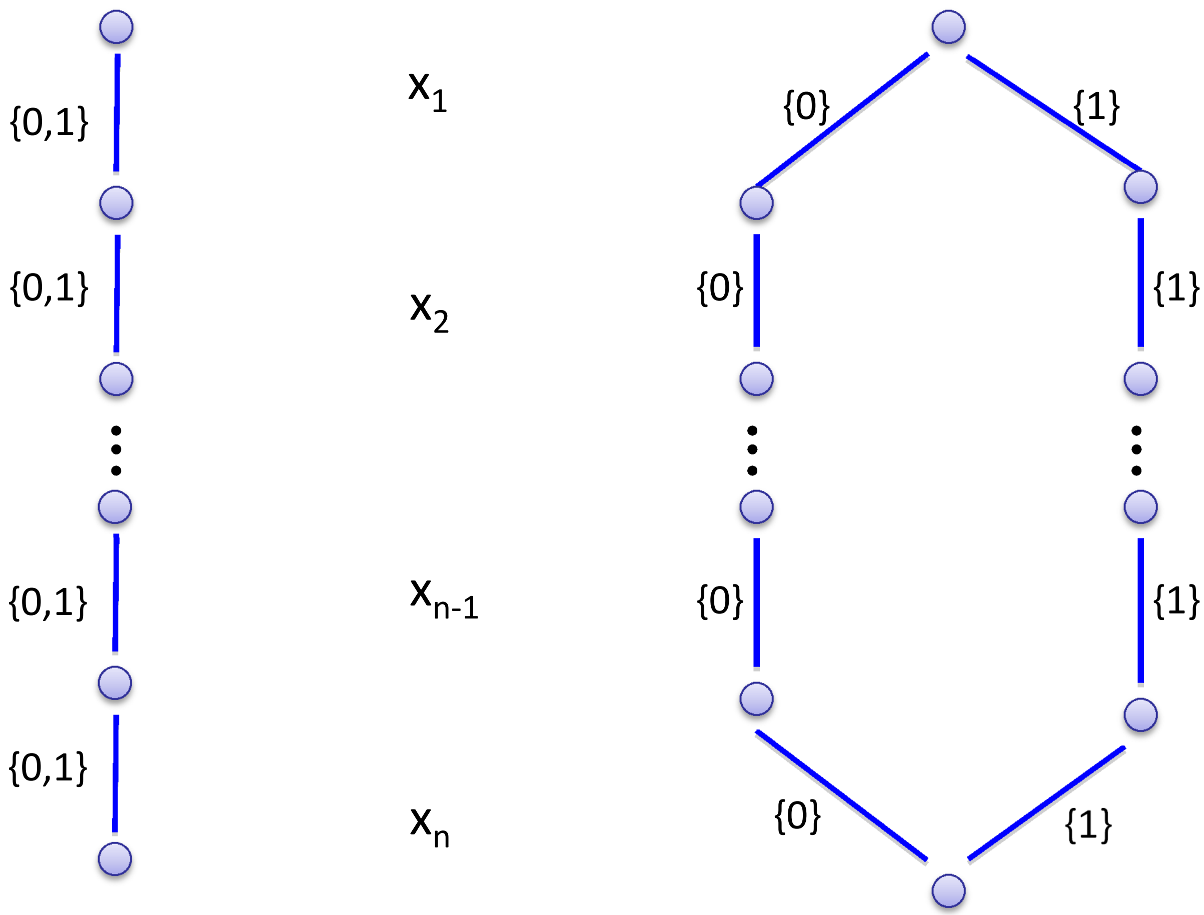
$x_1 < x_2$ $x_2 \in \{2,3\}$

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
- 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

Characterization of Propagation

Domain consistency generalizes naturally to MDDs:

- Let C(X) be a constraint on variables X and let M be an MDD on X
- Constraint C is MDD consistent if for each arc in M, there is at least one path in M that represents a solution to C

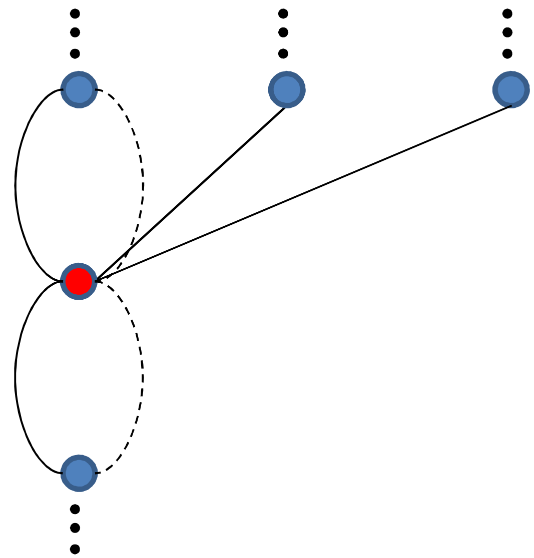
Equivalent to domain consistency for MDD of width 1

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., 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]

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*



First example: Among constraints

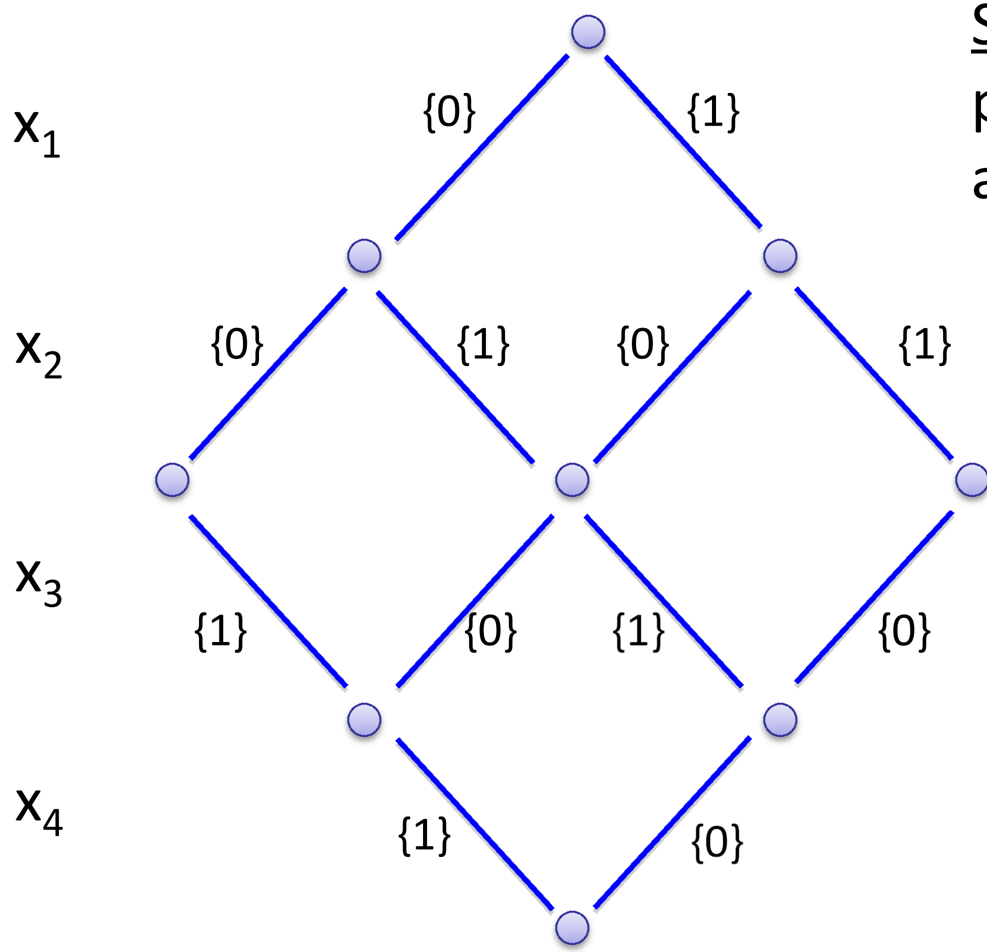
- Given a set of variables X, and a set of values S, a lower bound l and upper bound u,

$$\mathit{Among}(X, S, l, u) := \quad l \leq \sum_{x \in X} (\, x \in S \,) \leq u$$

"among the variables in X, at least l and at most u take a value from the set S"

- Applications in, e.g., sequencing and scheduling
- WLOG assume here that X are binary and $S = \{1\}$

Example MDD for Among



<u>State information:</u>
path length from top and from bottom

Exact MDD for *Among*($\{x_1,x_2,x_3,x_4\},\{1\},2,2$)

MDD Filtering for Among

Goal: Given an MDD and an *Among* constraint, remove *all* inconsistent edges from the MDD

(establish MDD-consistency) [Hoda et al., CP 2010]

Approach:

- Compute path lengths from the root and from the sink to each node in the MDD
- Remove edges that are not on a path with length between lower and upper bound
- Complete (MDD-consistent) version
  - Maintain all path lengths; quadratic time
- Partial version (does not remove all inconsistent edges)
  - Maintain and check bounds (longest and shortest paths); linear time

Node refinement for Among

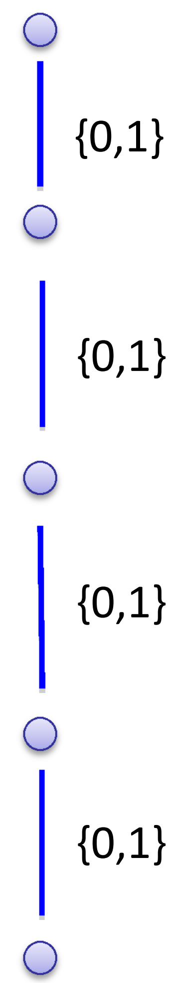
For each layer in MDD, we first apply edge filter, and then try to refine

- consider incoming edges for each node
- split the node if there exist incoming edges that are not equivalent (w.r.t. path length)
- in other words, need to identify *equivalence classes*

Example:

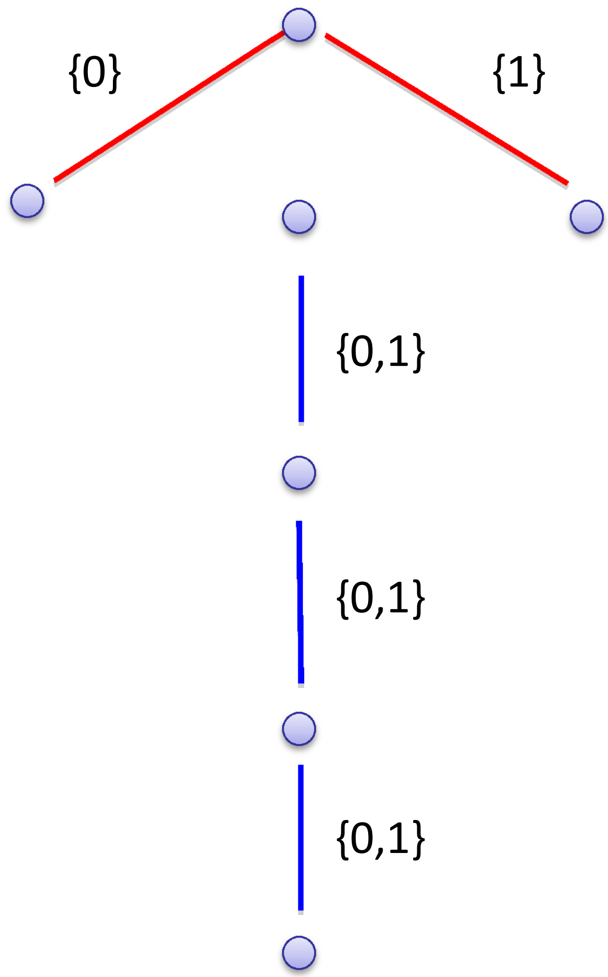
- We will propagate *Among*($\{x_1,x_2,x_3,x_4\},\{1\},2,2$) through a BDD of maximum width 3

Example



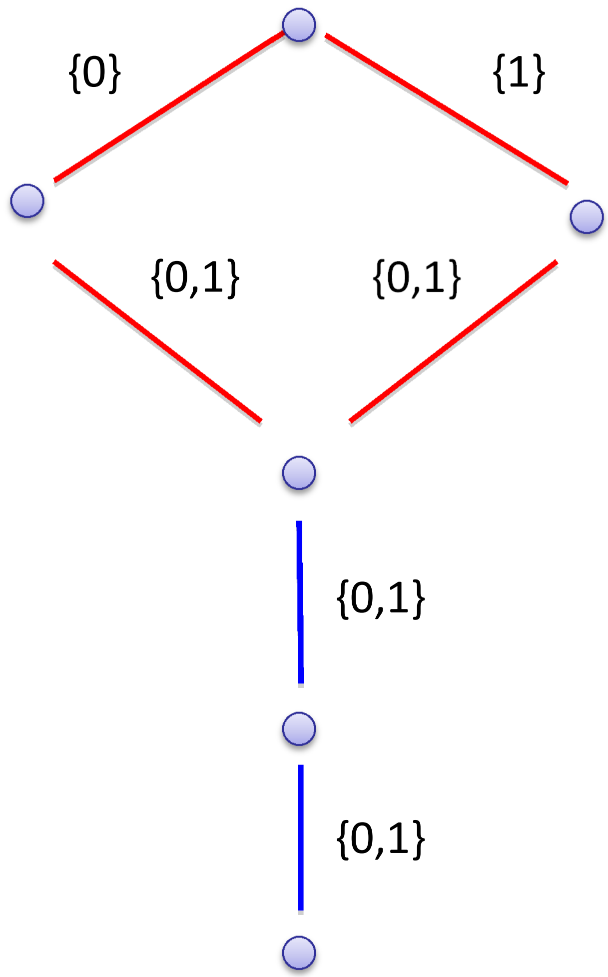
$Among(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example



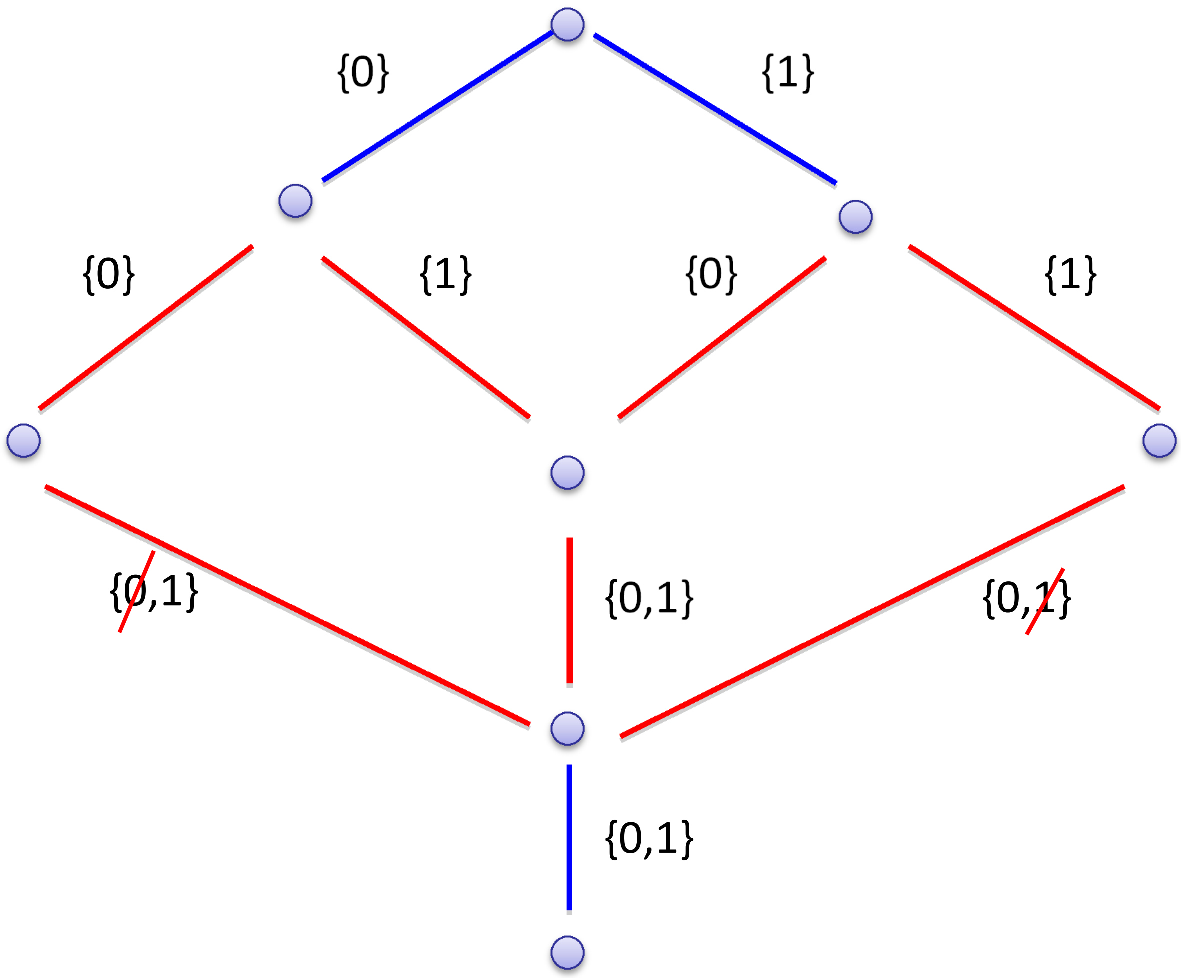
$Among(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example



$Among(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example



Among$(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

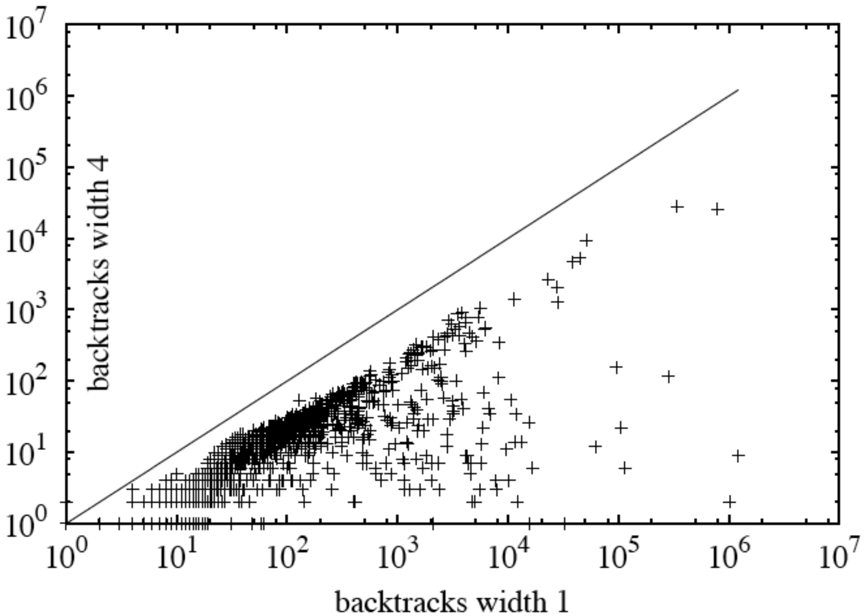
Experiments

- Multiple among constraints
  - 50 binary variables total
  - 5 variables per among constraint, indices chosen from normal distribution with uniform-random mean in [1..50] and stdev 2.5, modulo 50 (i.e., somewhat consecutive)
  - Classes: 5 to 200 among constraints (step 5), 100 instances per class

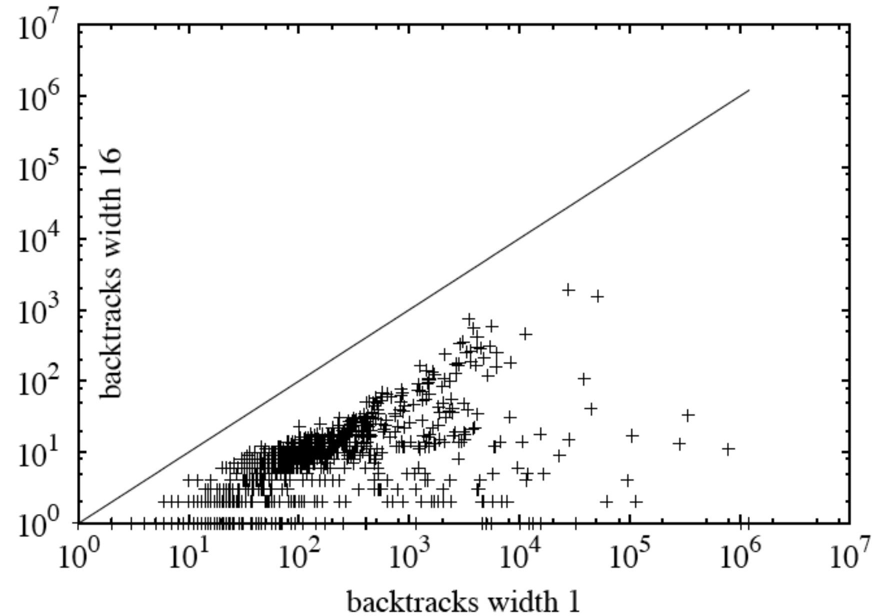
- Nurse rostering instances (horizon *n* days)
  - Work 4-5 days per week
  - Max A days every B days
  - Min C days every D days
  - Three problem classes

- Compare width 1 (traditional domains) with increasing widths

Multiple Amongs: Backtracks

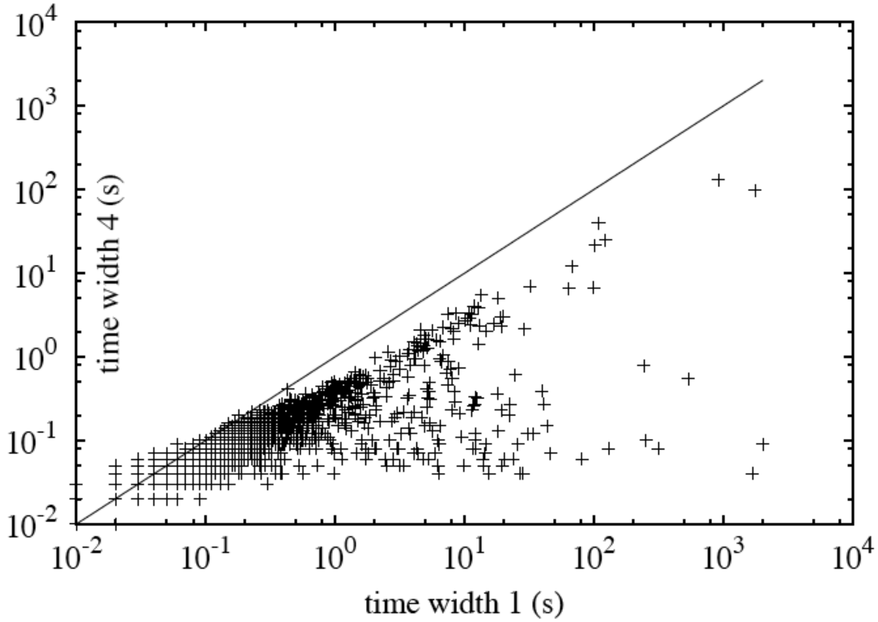


width 1 vs 4

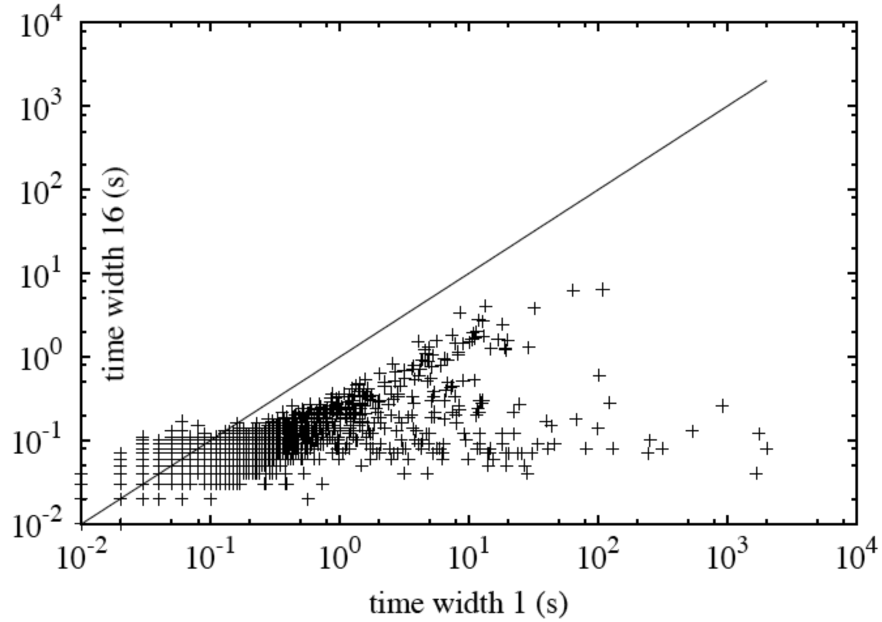


width 1 vs 16

Multiple Amongs: Running Time



width 1 vs 4



width 1 vs 16

Nurse rostering problems

	Size	Width 1 BT	Width 1 CPU	Width 4 BT	Width 4 CPU	Width 32 BT	Width 32 CPU
Class 1	40	61,225	55.63	8,138	12.64	3	0.09
	80	175,175	442.29	5,025	44.63	11	0.72
Class 2	40	179,743	173.45	17,923	32.59	4	0.07
	80	179,743	459.01	8,747	80.62	2	0.32
Class 3	40	91,141	84.43	5,148	9.11	7	0.18
	80	882,640	2,391.01	33,379	235.17	55	3.27

Sequence Constraint

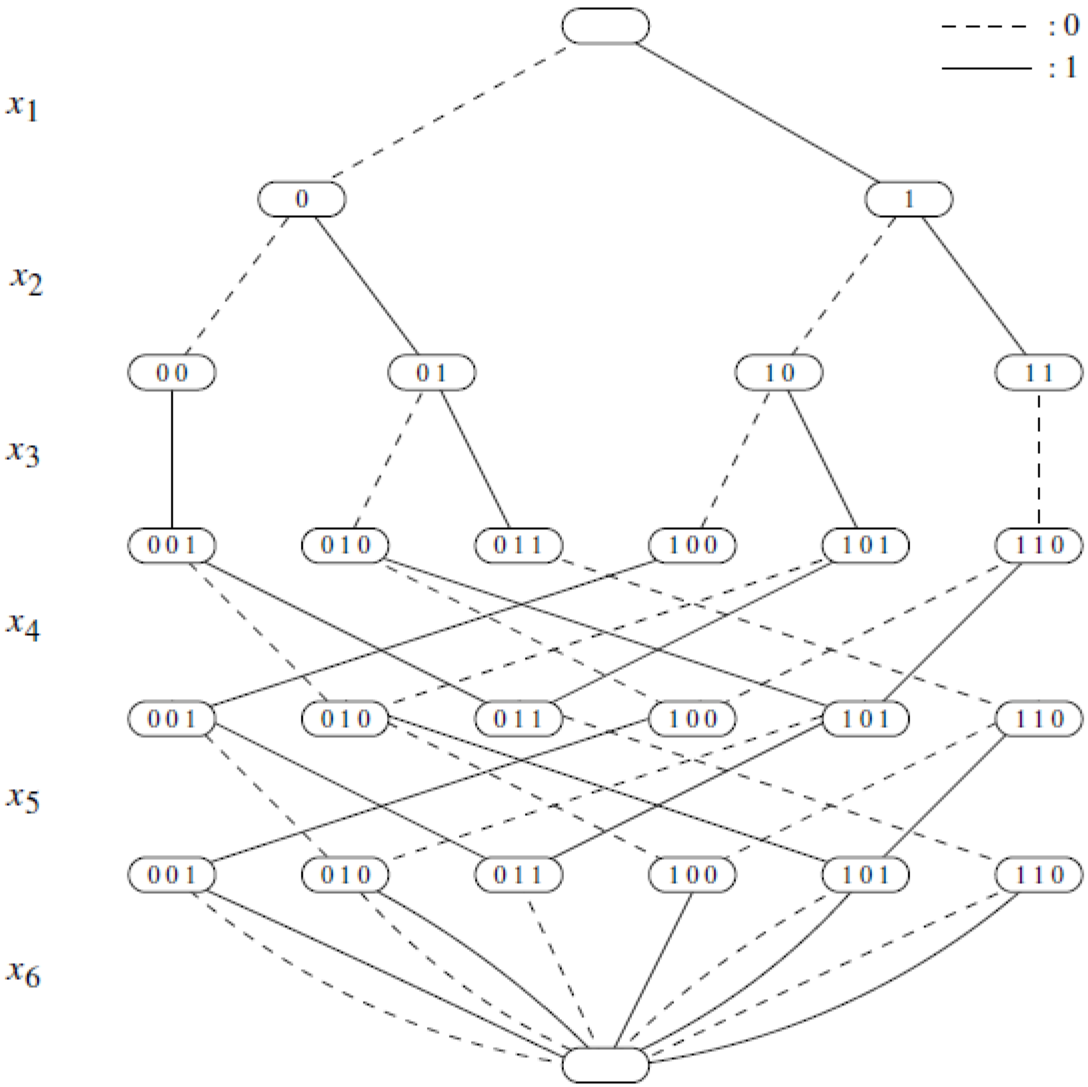
Employee must work at most 7 days every 9 consecutive days

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}

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

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

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

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

$$y_0 = 0$$

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

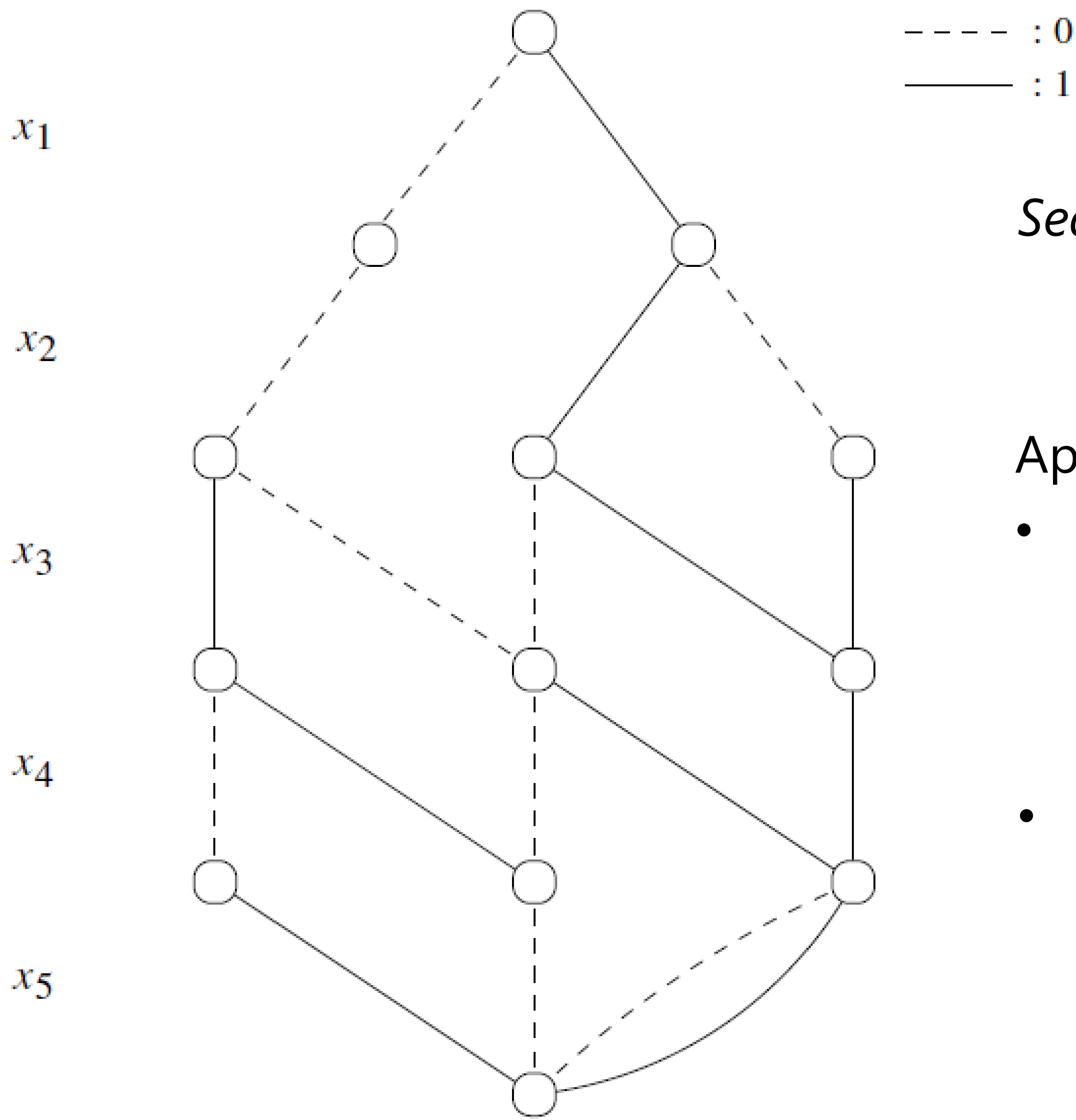
- Then the among constraint on $[x_{i+1}, \ldots, x_{i+q}]$ is equivalent to

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

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

- [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

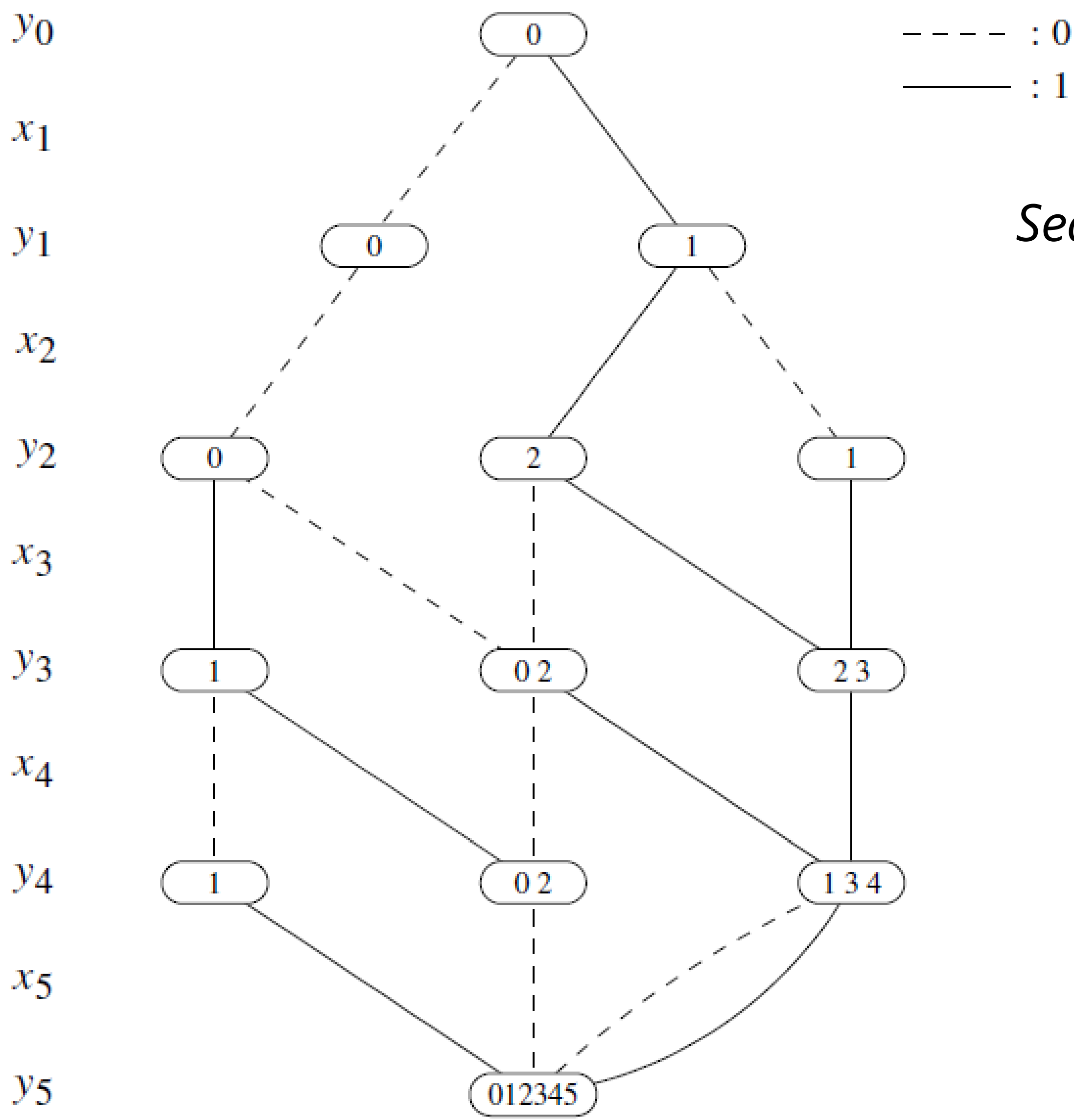


Sequence(X, q=3, S={1}, l=1, u=2)

Approach

- The auxiliary variables y_i can be naturally represented at the *nodes* of the MDD – 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)

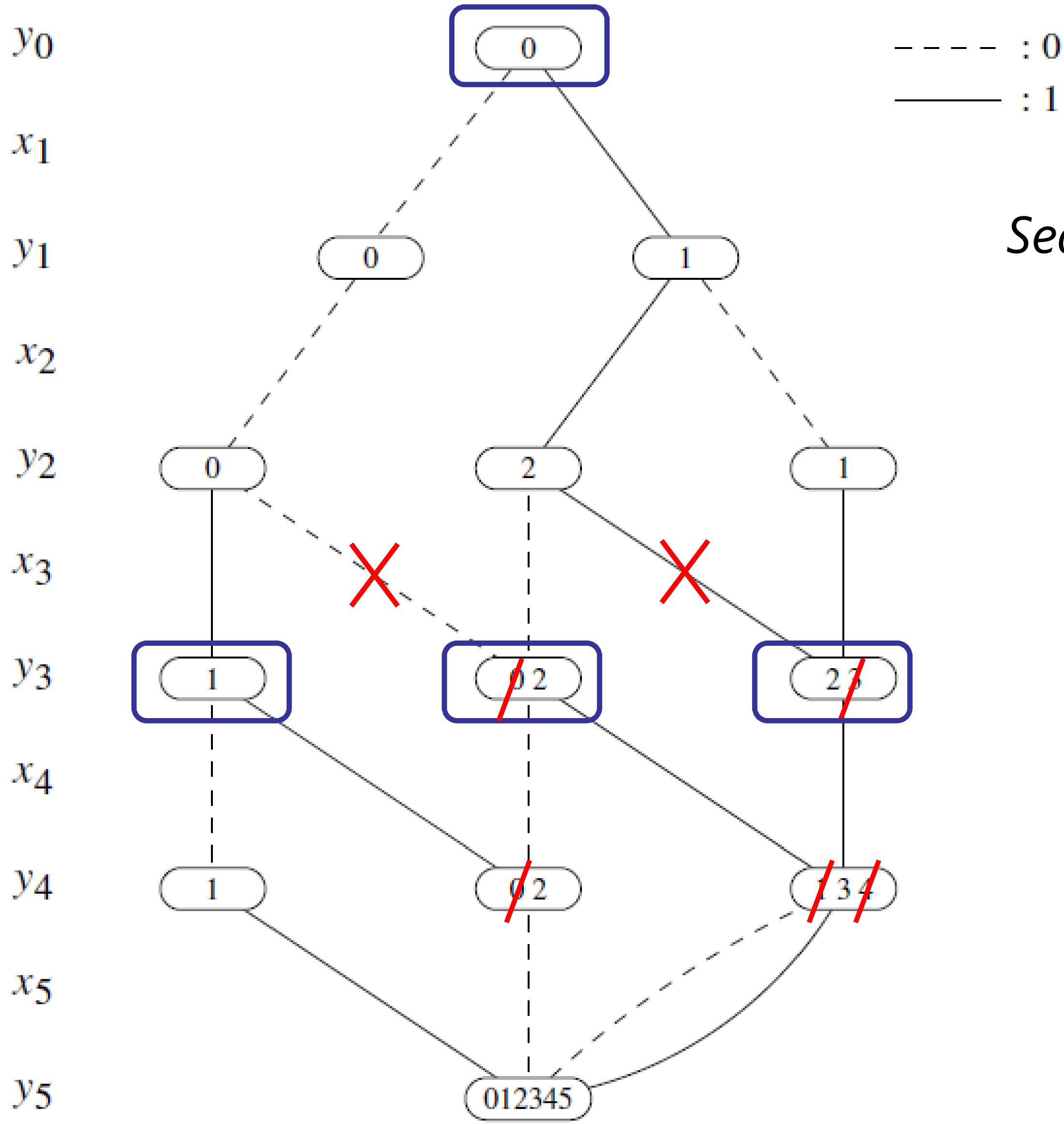
$\boxed{y_i = y_{i-1} + x_i}$

$1 \le y_3 - y_0 \le 2$

$1 \le y_4 - y_1 \le 2$

$1 \le y_5 - y_2 \le 2$

MDD filtering from decomposition



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

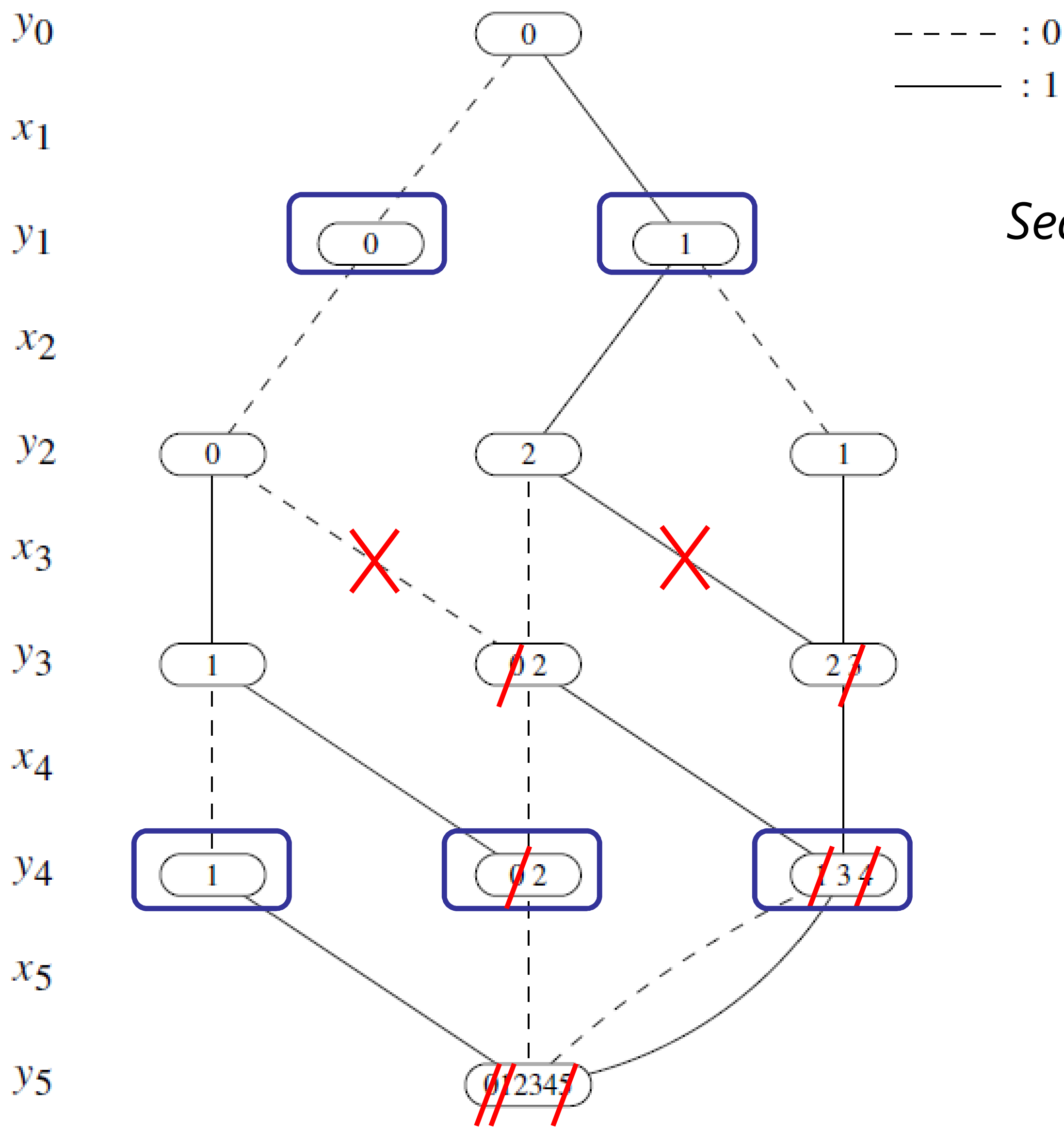
$$y_i = y_{i-1} + x_i$$

$$\boxed{1 \le y_3 - y_0 \le 2}$$

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

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

MDD filtering from decomposition



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

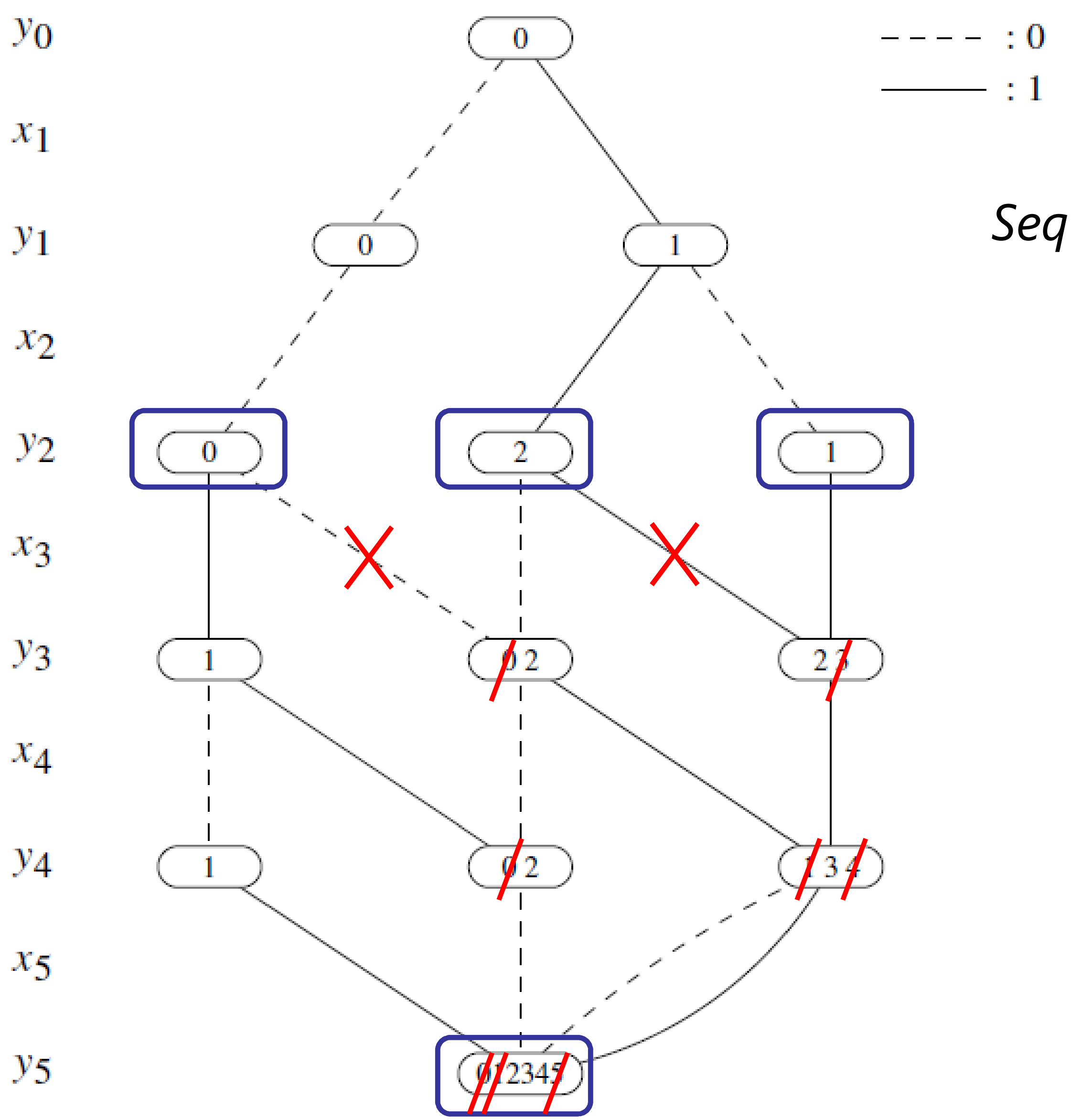
$y_i = y_{i-1} + x_i$

$1 \le y_3 - y_0 \le 2$

$\boxed{1 \le y_4 - y_1 \le 2}$

$1 \le y_5 - y_2 \le 2$

MDD filtering from decomposition



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

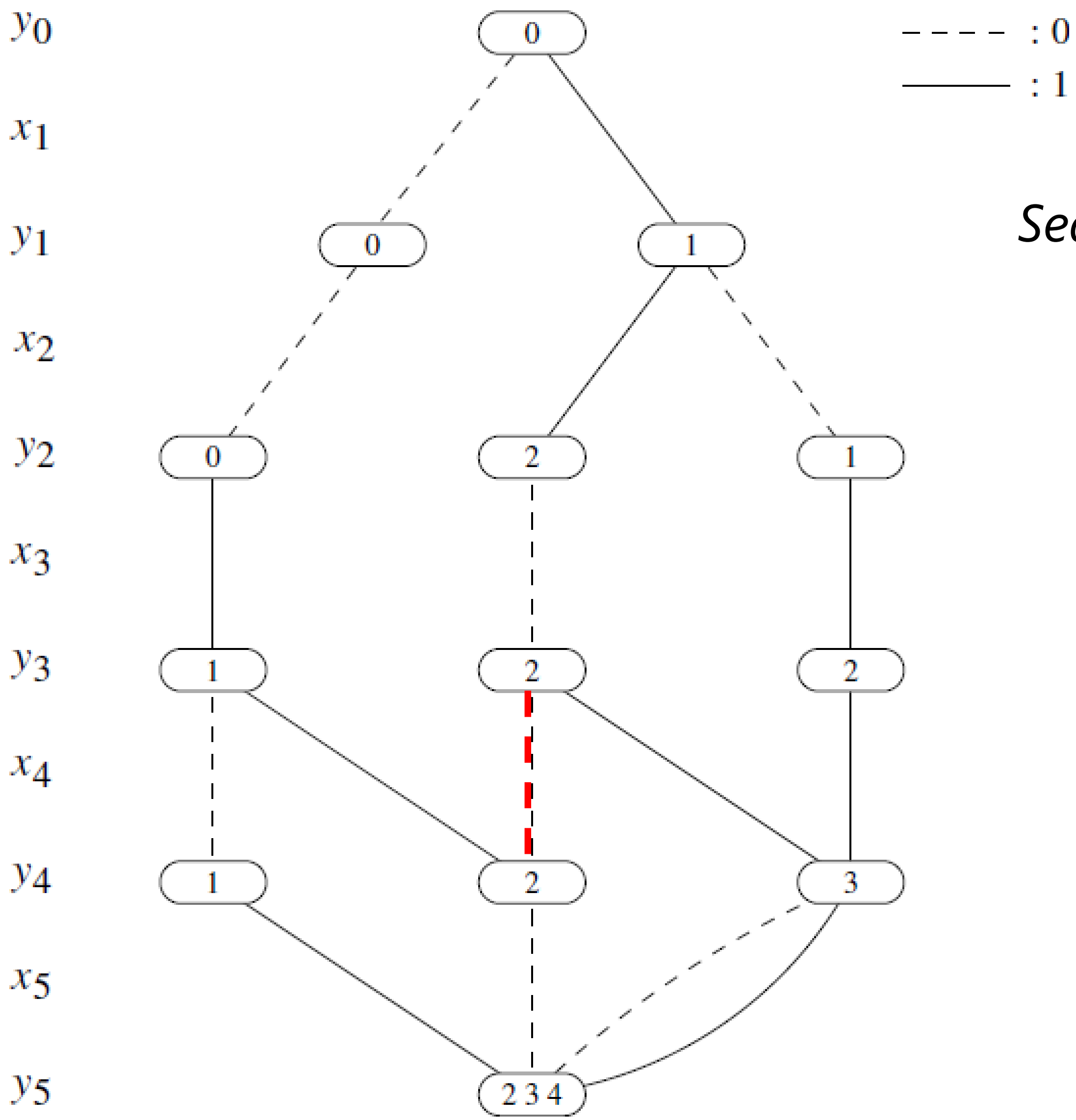
$y_i = y_{i-1} + x_i$

$1 \le y_3 - y_0 \le 2$

$1 \le y_4 - y_1 \le 2$

$\boxed{1 \le y_5 - y_2 \le 2}$

MDD filtering from decomposition



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

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

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

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

$$1 \leq y_5 - y_2 \leq 2$$

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 Setup

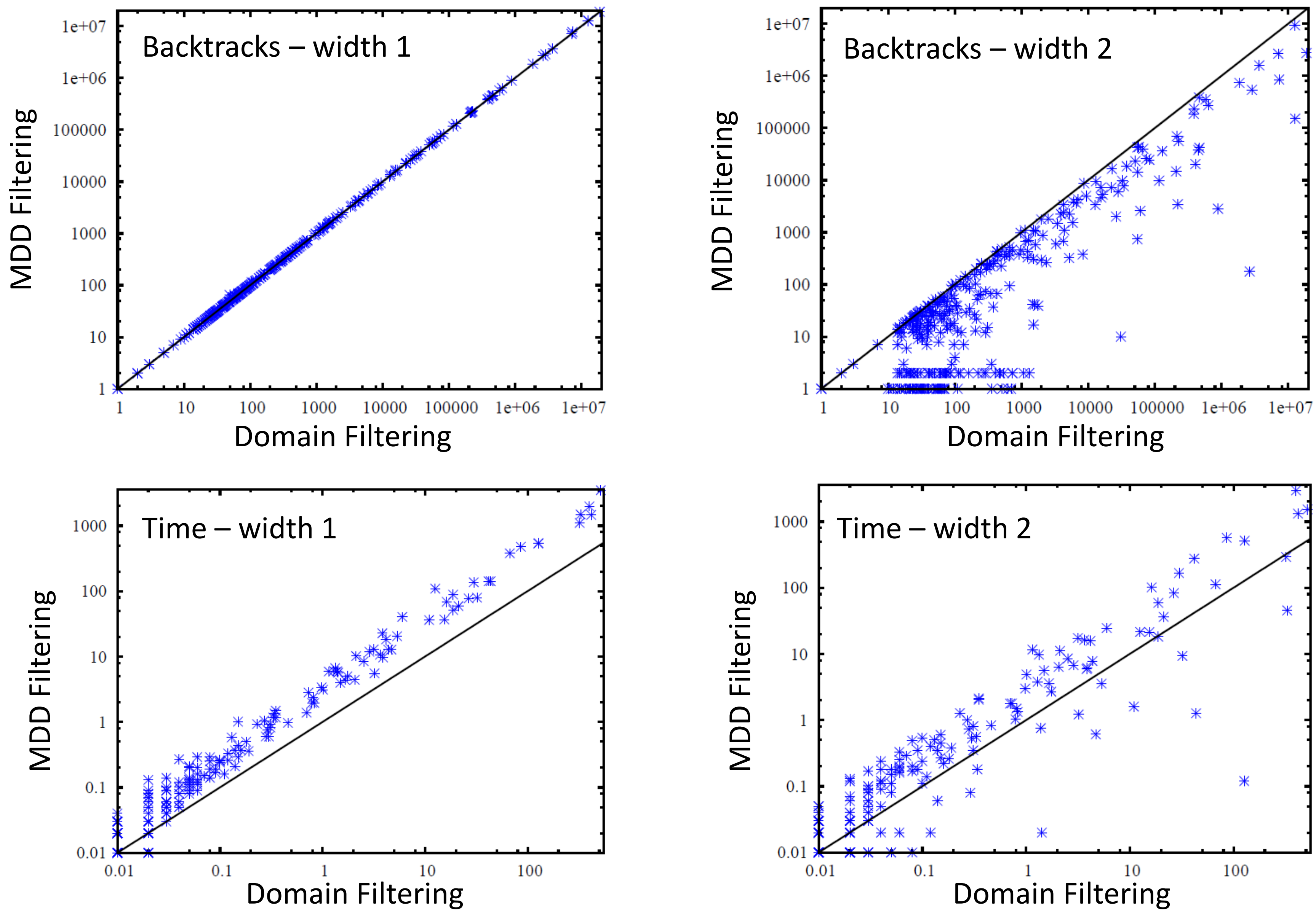
- Decomposition-based MDD filtering algorithm
  - Implemented as global constraint in IBM ILOG CPLEX CP Optimizer 12.3
- Evaluation
  - Compare MDD filtering with Domain filtering
  - Domain filter based on the same decomposition (achieves domain consistency for all our instances)
  - Random instances 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

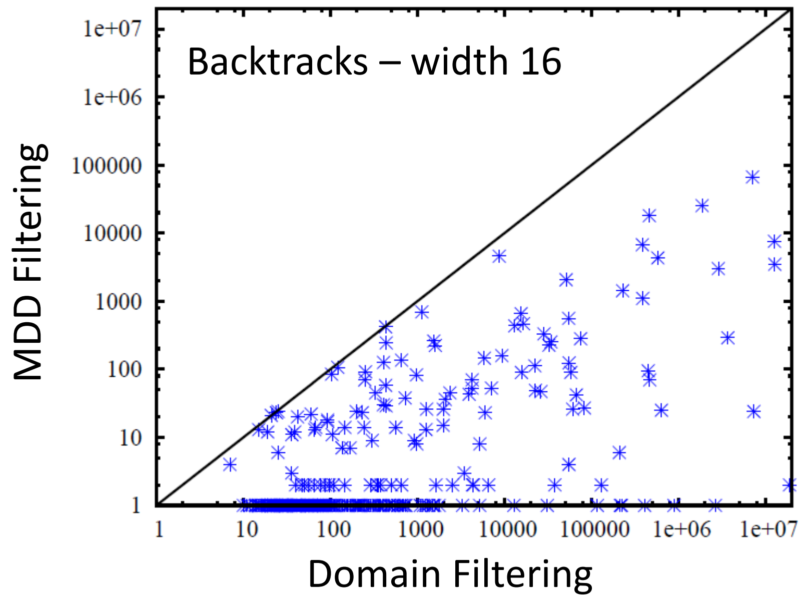
- Randomly generated instances
  - n=20-48 variables
  - domain size between 10 and 30
  - 1, 2, 5, 7, or 10 *Sequence* constraints
  - q random from [2..n/2]
  - $u - l$ random from 0 to q-1
  - 360 instances

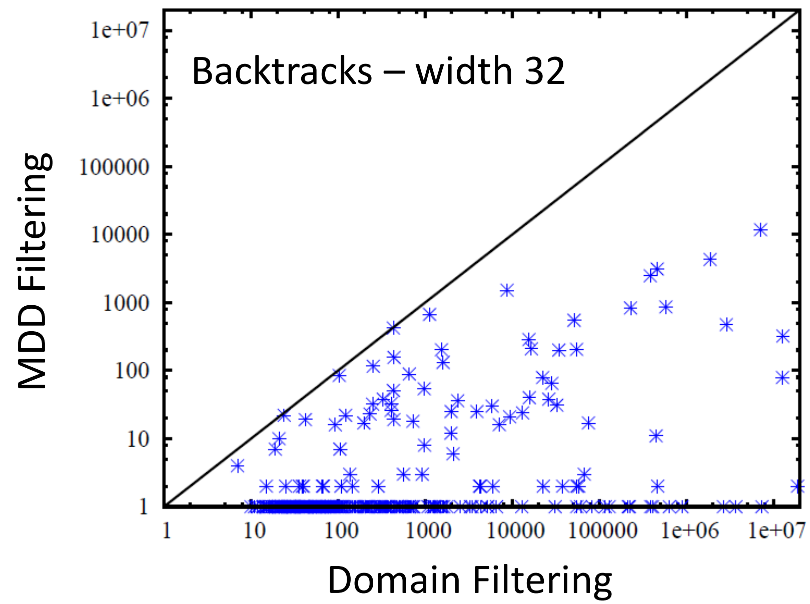
- Vary maximum width of MDD
  - widths 1 up to 32

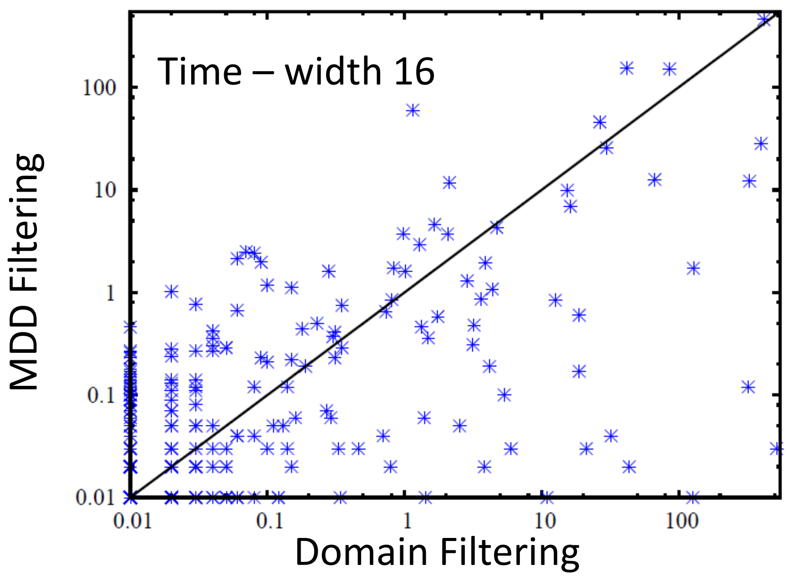
Random instances results

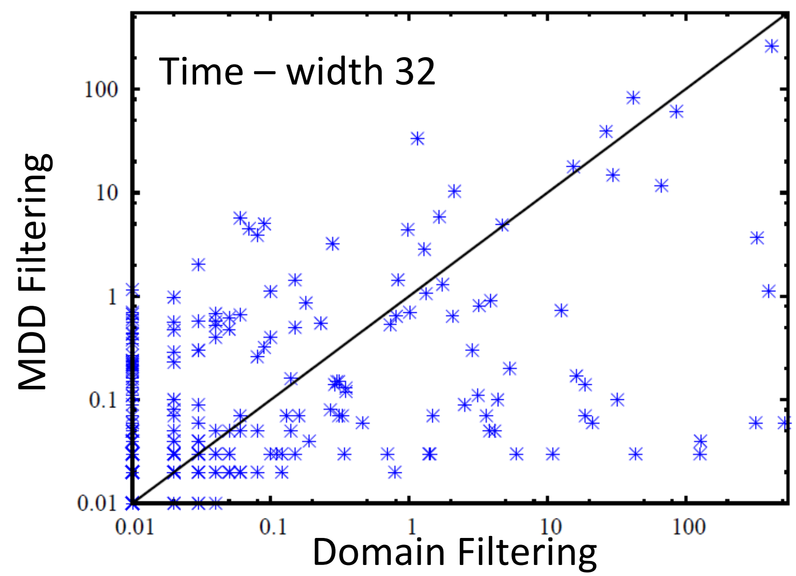


Random instances results (cont'd)









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

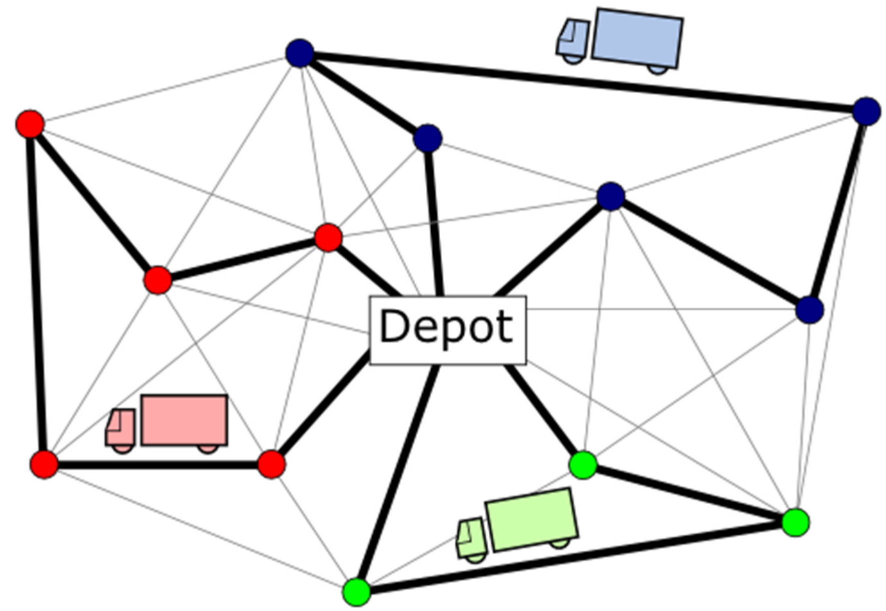
    Sequence(X, q=7, S={O}, l=1, u=4)

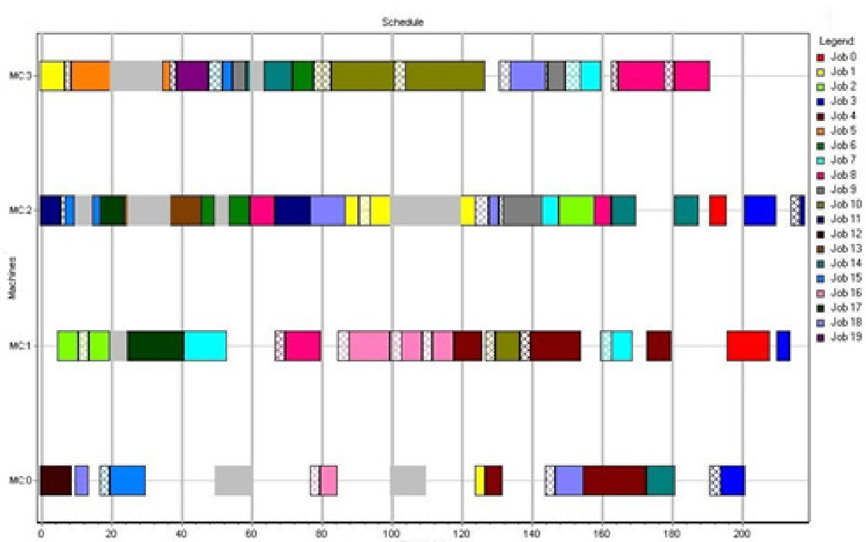
- Problem type P-II
  - *Sequence*(X, q=30, S={D, E}, l=23, u=30)
  - *Sequence*(X, q=5, S={N}, l=1, u=2)

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

MDDs for Disjunctive Scheduling





Constraint-Based Scheduling

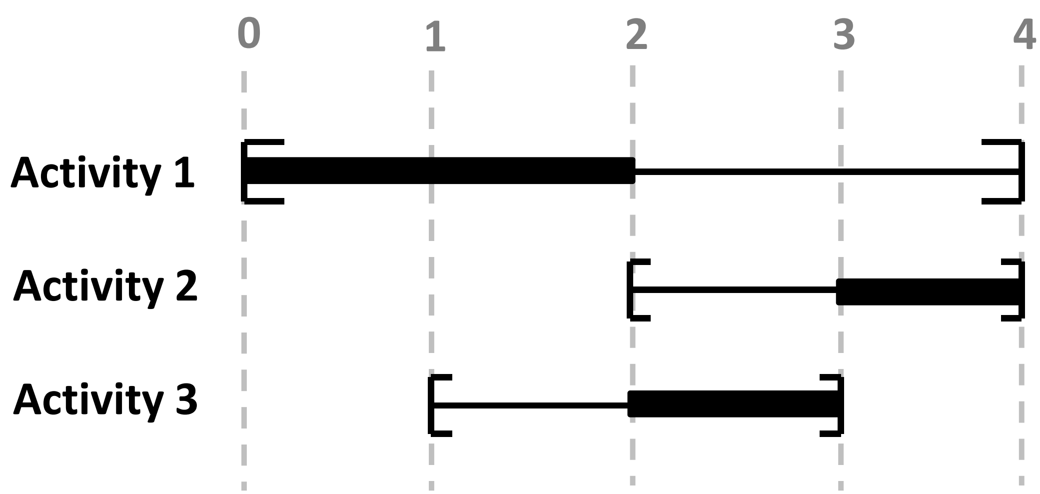
- Disjunctive scheduling may be viewed as the ‘killer application’ for CP
  - Natural modeling (activities and resources)
  - Allows many side constraints (precedence relations, time windows, setup times, etc.)
  - State of the art while being generic methodology
- However, CP has some problems when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

Heinz & Beck [CPAIOR 2012] compare CP and MIP

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

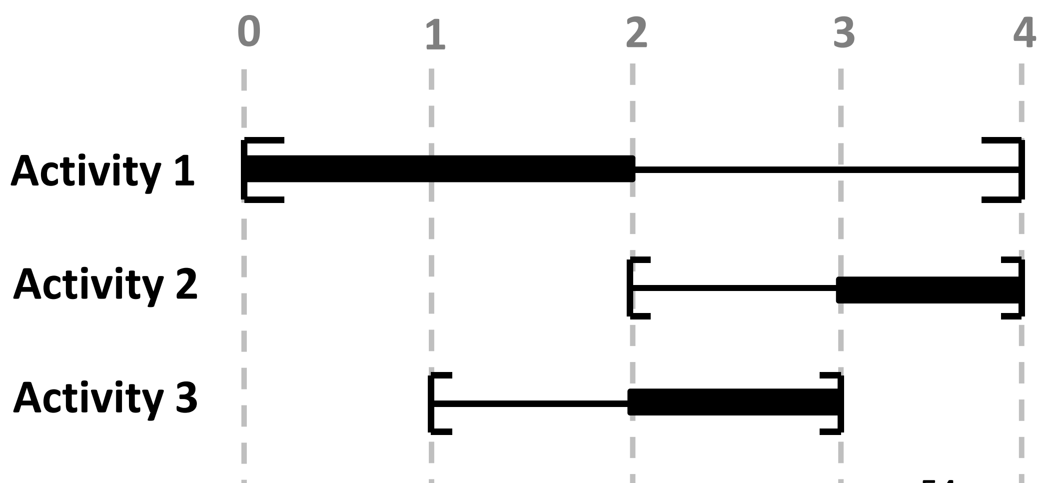
Common Side Constraints

- 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: $1 \ll 3$

  $s_3 \geq 3$



MDDs for Disjunctive Scheduling

Our three main considerations:

Cire & v.H. [2012]

- 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'

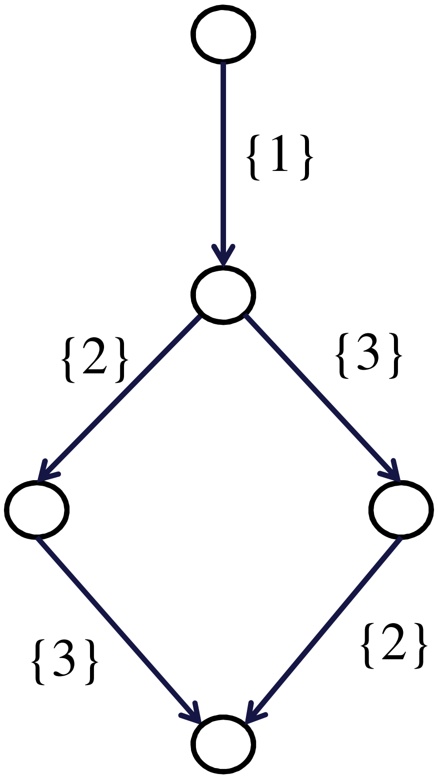
- Every solution can be written as a permutation $\boldsymbol{\pi}$

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

MDD Propagation

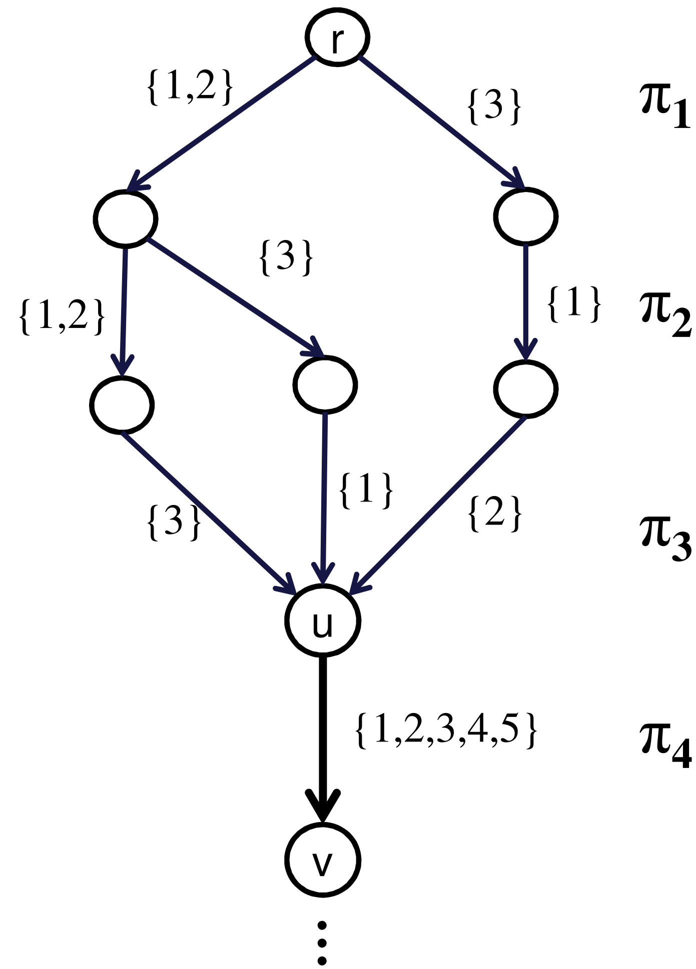
We can apply several propagation algorithms:

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

Propagation (cont'd)

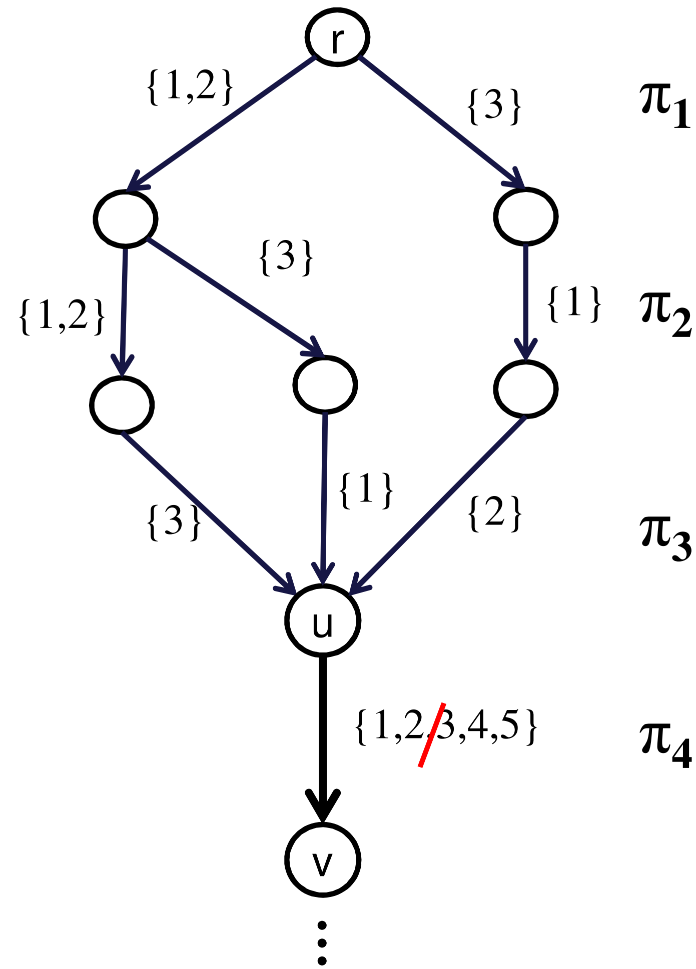
- State information at each node *i*
  - labels on *all* paths: A_i
  - labels on *some* paths: S_i
  - earliest starting time: E_i
  - latest completion time: L_i

- Top down example for arc (u,v)



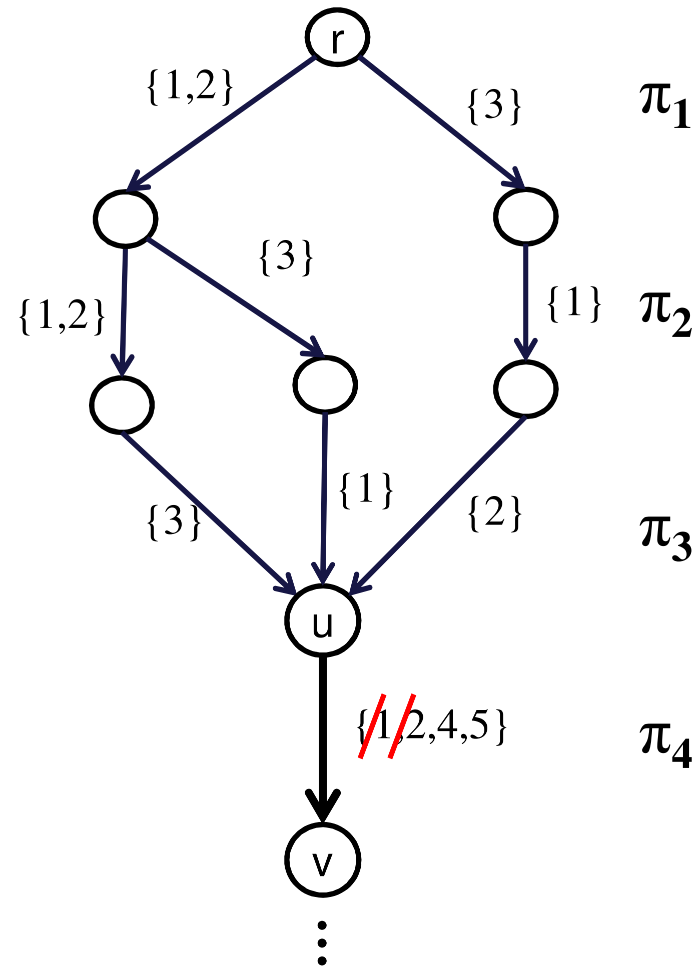
Alldifferent Propagation

- 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)



[Andersen et al., 2007]

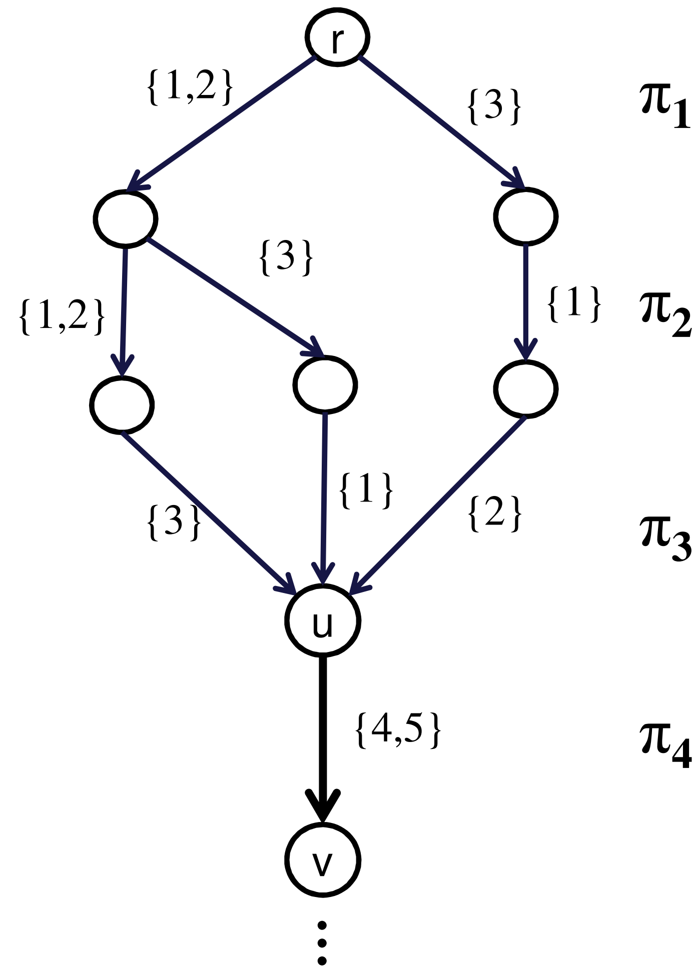
- Some-paths state: S_u
  - Labels belonging to some path from node r to node u
  - $S_u = \{1,2,3\}$
  - Identification of Hall sets
  - Thus eliminate {1,2,3} from (u,v)



Propagate Earliest Completion Time

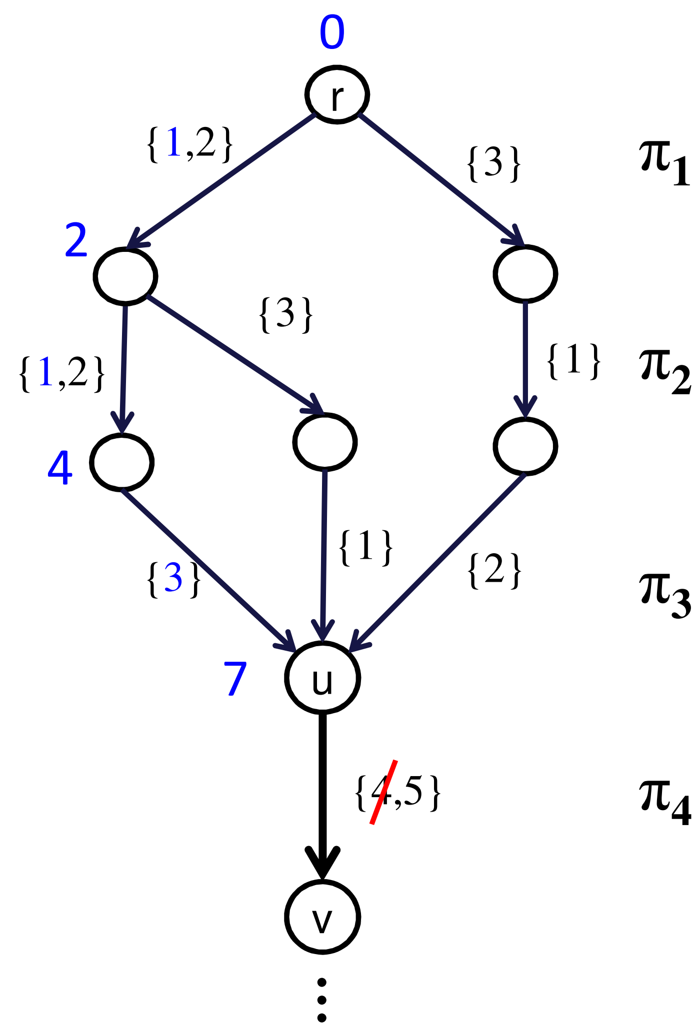
- Earliest Completion Time: E_u
  - Minimum completion time of all paths from root to node u

- 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)

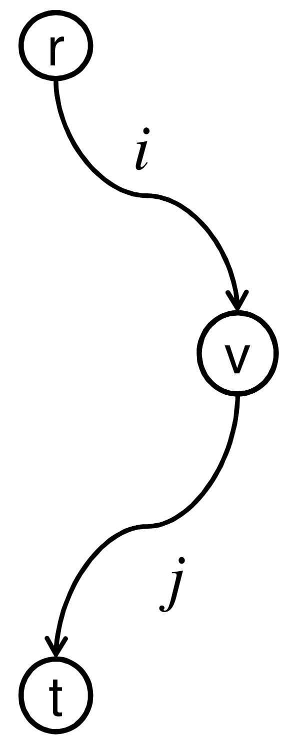
More MDD Inference

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

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



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

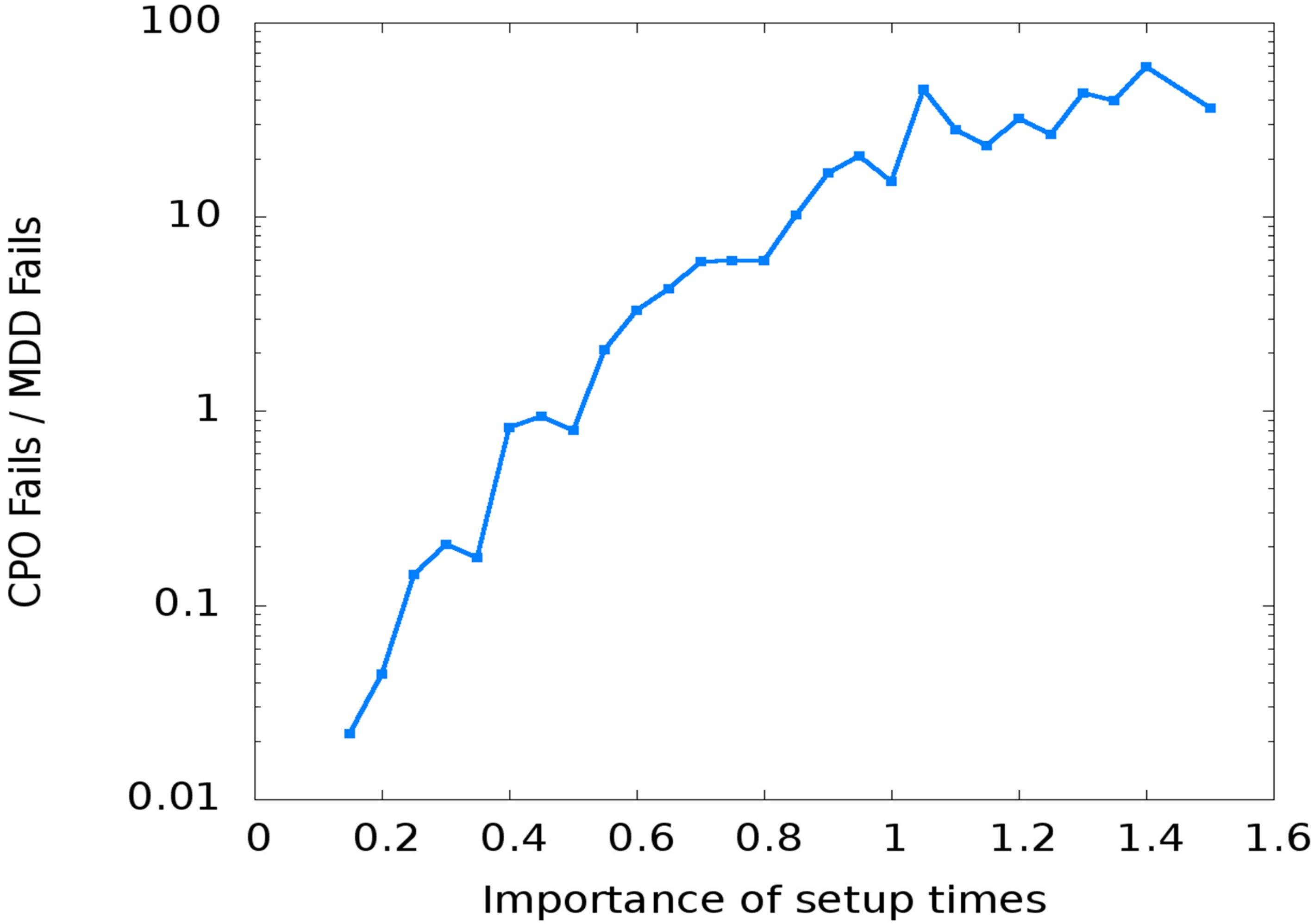
- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation

- Main purpose of experiments
  - where can MDDs bring strength to 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

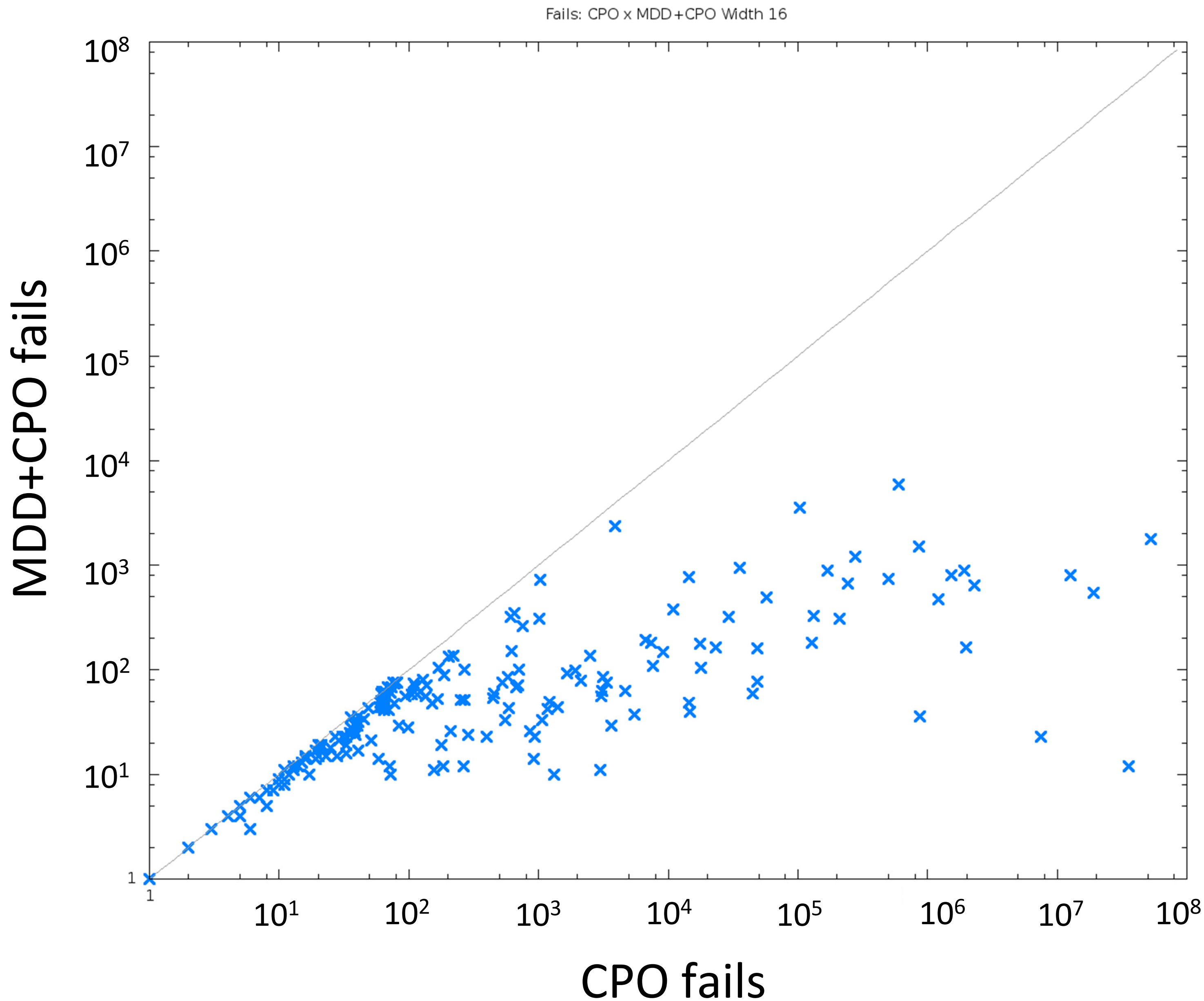
- 229 TSPTW instances with up to 100 jobs
- Minimize makespan
- Time limit 7,200s
- Max MDD width is 16

instances solved by CP: 211

instances solved by pure MDD: 216

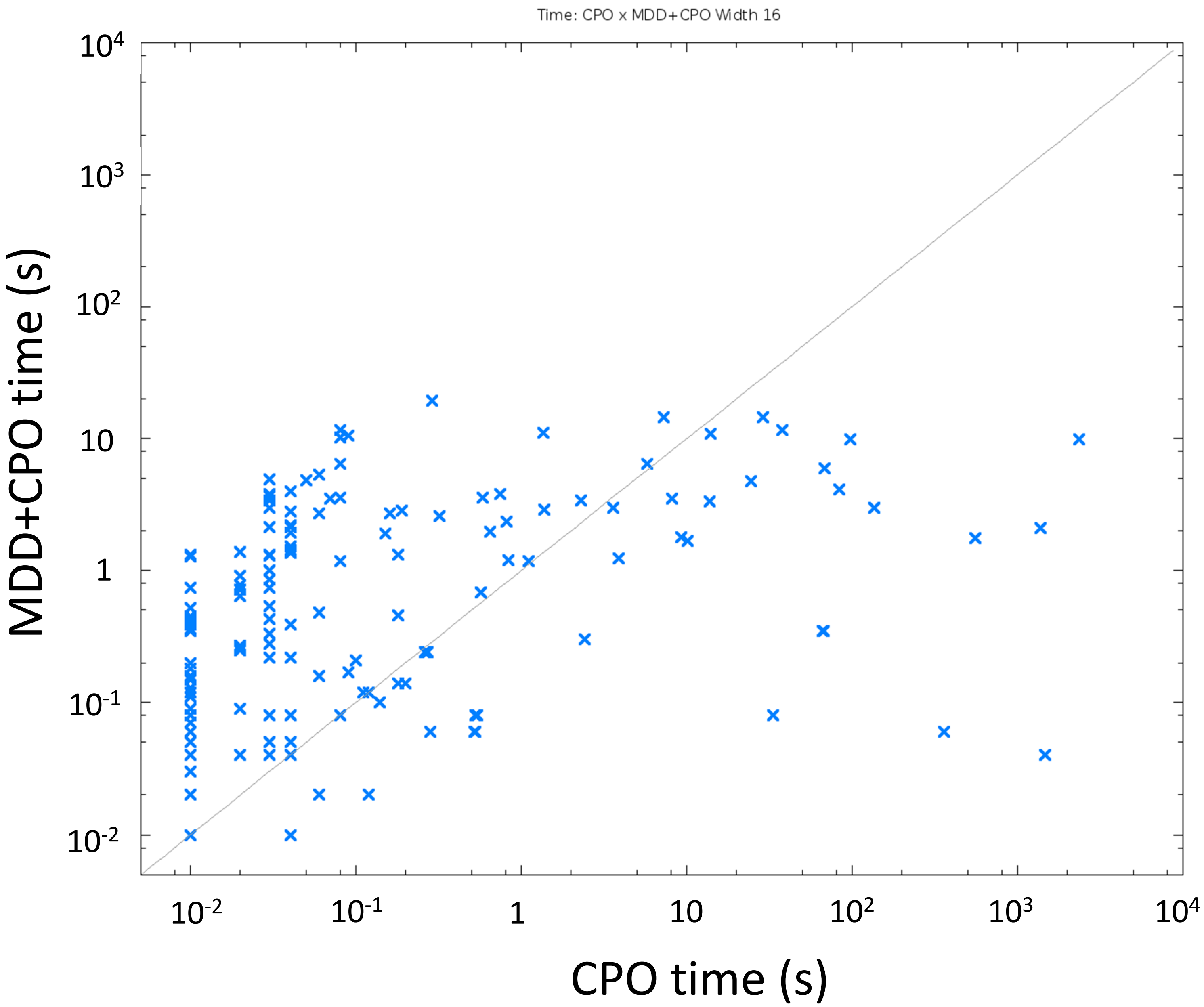
instances solved by CP+MDD: 225

Minimize Makespan: Fails

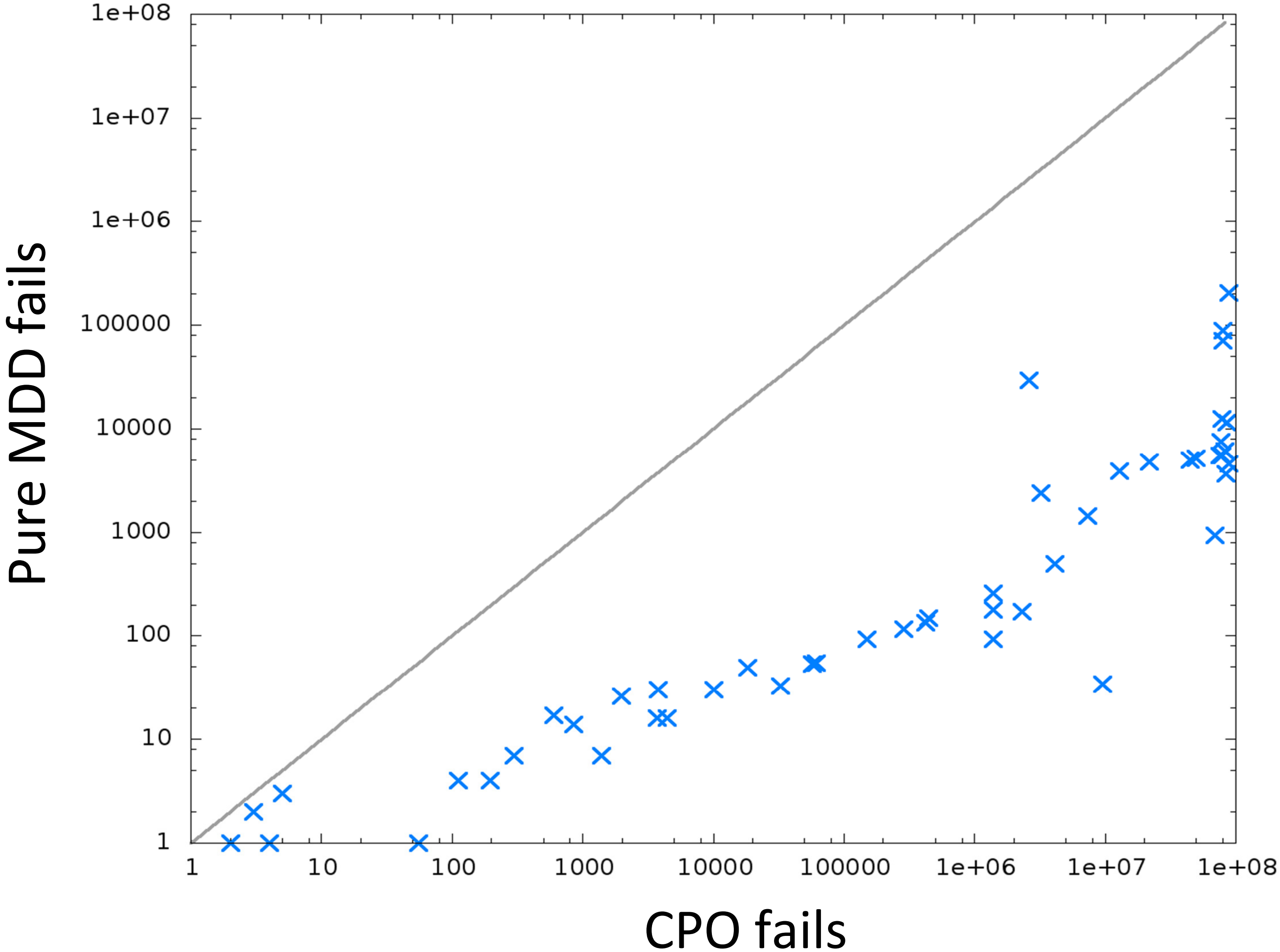


plot only on instances that were solved by all methods

Minimize Makespan: Time

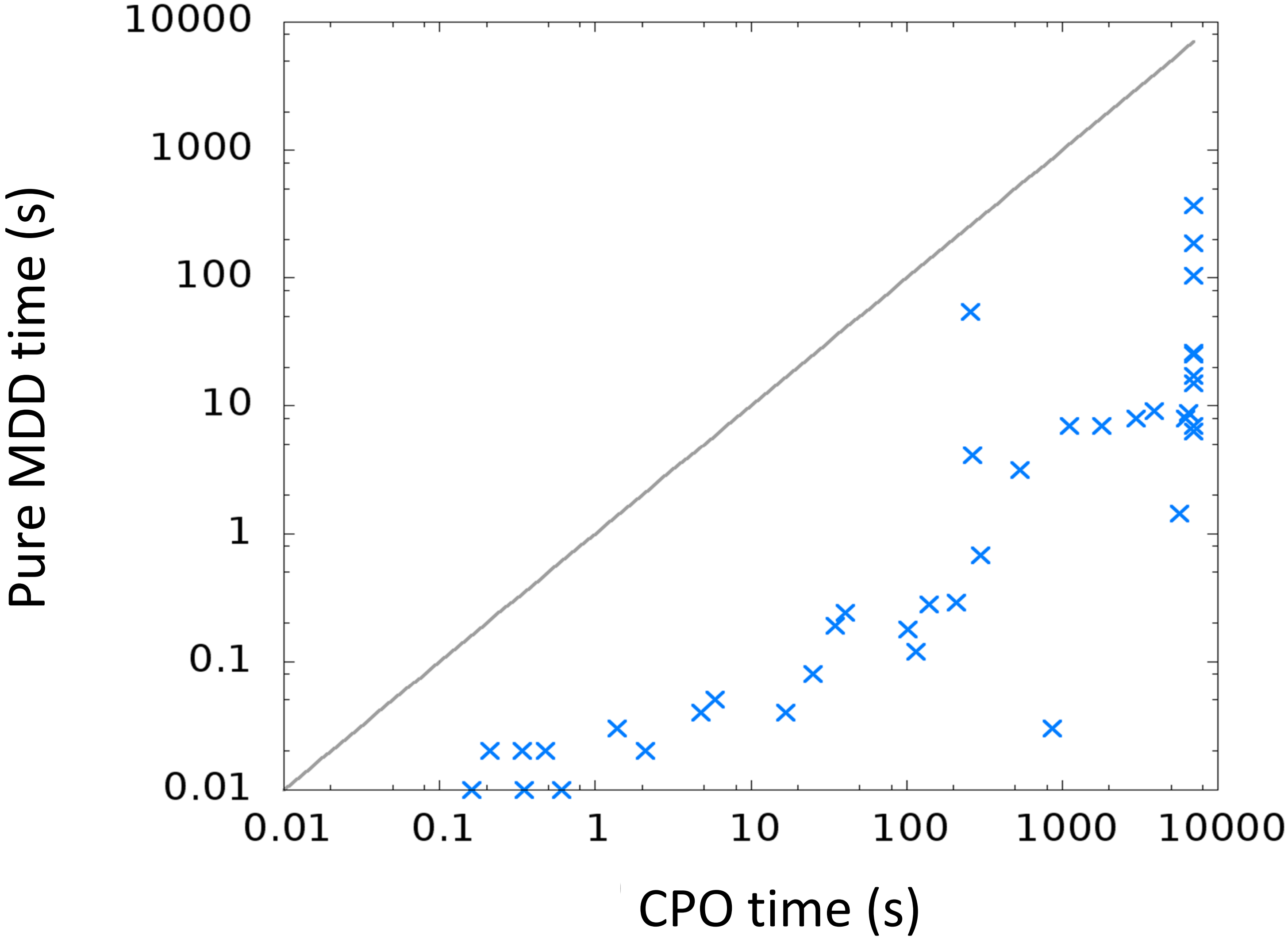


Min sum of setup times: Fails



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	*** CP improved bound**
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	*** closed by MDD**
p43.3.sop	[28366,28680]	28855	TL	**28835**	251.4	*** closed by MDD**
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	*** closed by MDD**

Summary for MDD-based CP

- MDDs 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
  - Particular examples: among, sequence, and disjunctive scheduling constraints

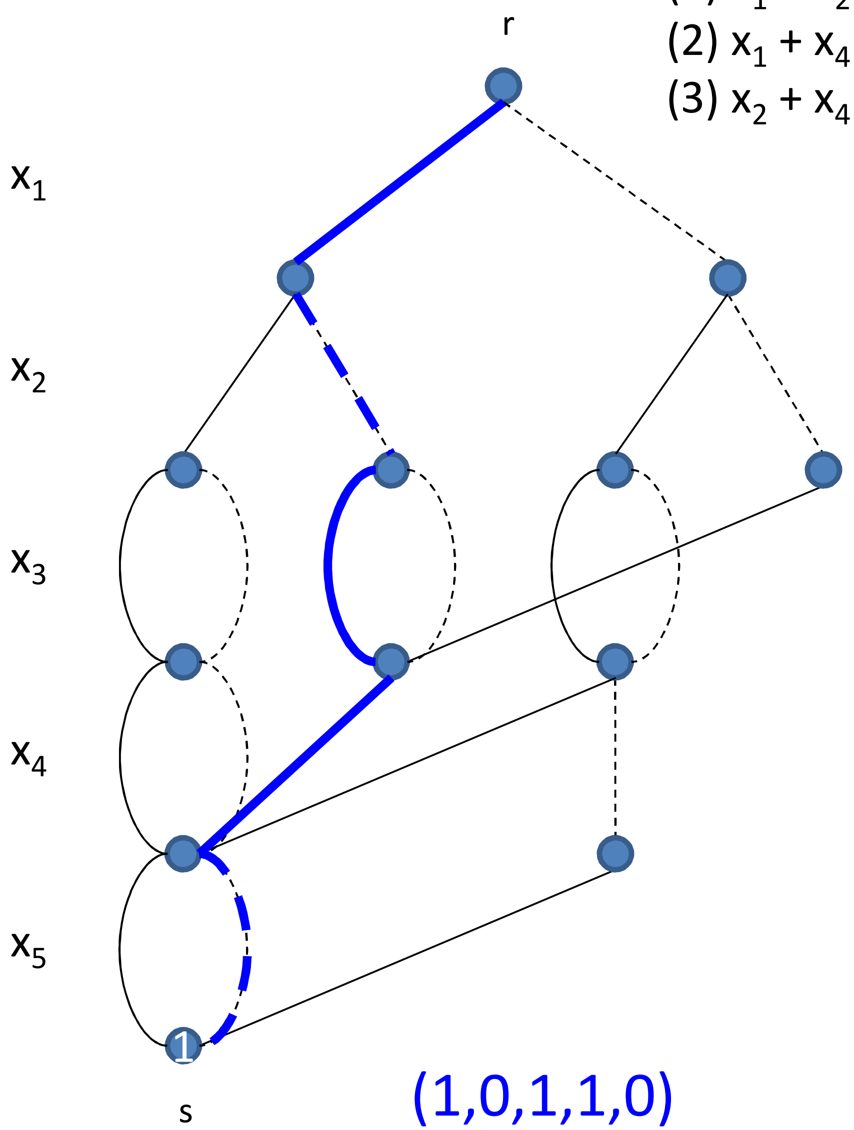
MDDs for Discrete Optimization

Motivation

- Limited width MDDs provide a (discrete) relaxation to the solution space
- Can we exploit MDDs to obtain bounds for discrete optimization problems?

Handling objective functions

(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$



Suppose we have an objective function of the form

$$\min \textstyle\sum_i f_i(x_i)$$

for arbitrary functions f_i

In an exact MDD, the optimum can be found by a shortest r-s path computation
(edge weights are $f_i(x_i)$)

Approach

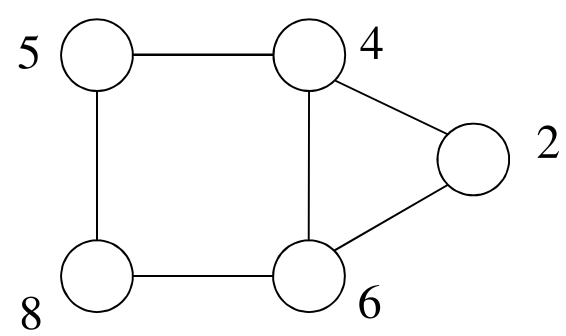
- Construct the relaxation MDD using a *top-down* compilation method
- Find shortest path $\rightarrow$ provides bound B
- Extension to an exact method
  1. Isolate all paths of length B, and verify if any of these paths is feasible*
  2. if not feasible, set B := B + 1 and go to 1
  3. otherwise, we found the optimal solution

* Feasibility can be checked using MDD-based CP

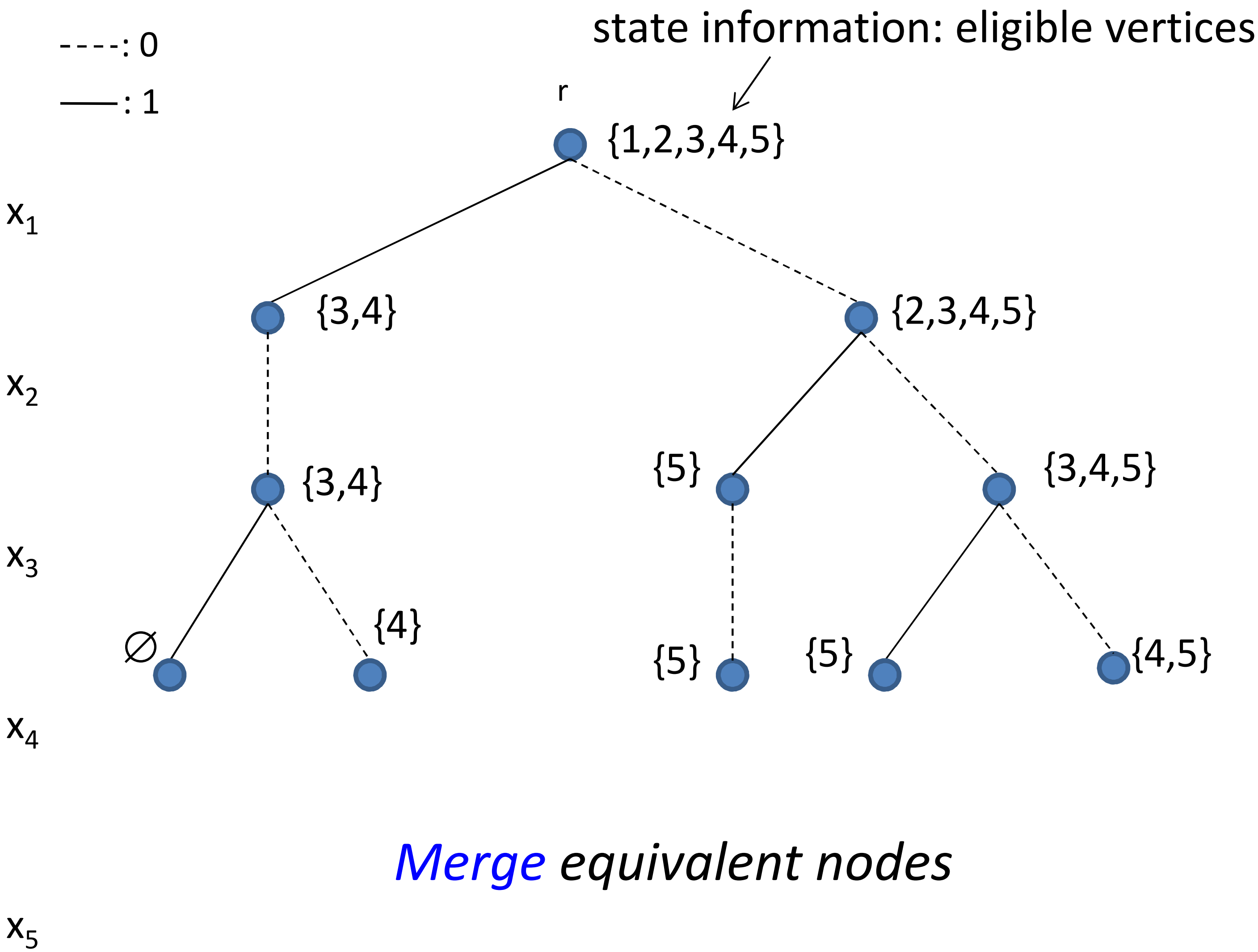
Case Study: Independent Set Problem

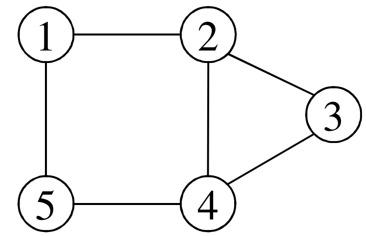
- Given graph $G = (V, E)$ with vertex weights w_i
- Find a subset of vertices S with maximum total weight such that no edge exists between any two vertices in S

max	$\sum_i w_i x_i$	
s.t.	$x_i + x_j \leq 1$	for all (i,j) in E
	x_i binary	for all i in V



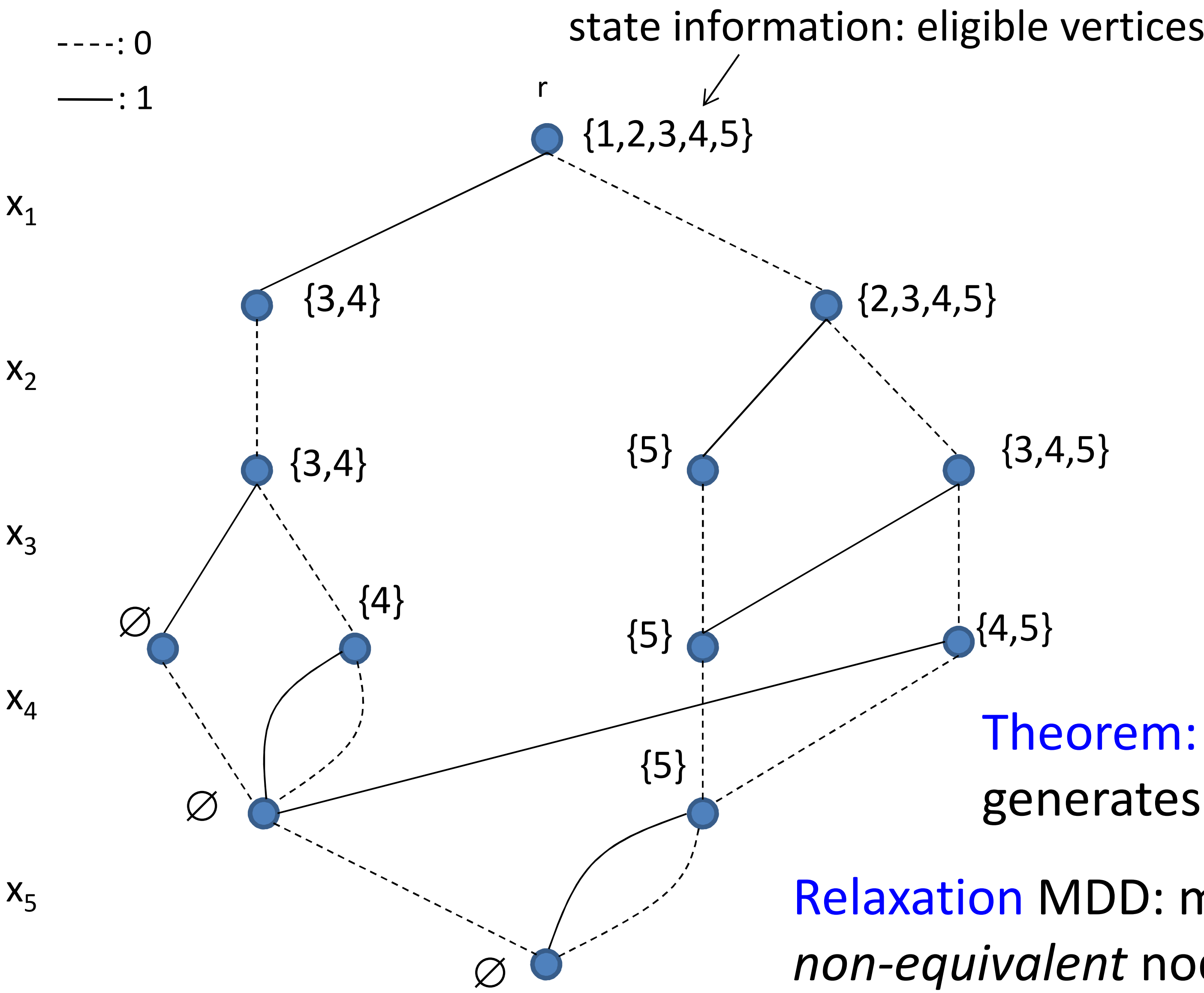
Exact top-down compilation

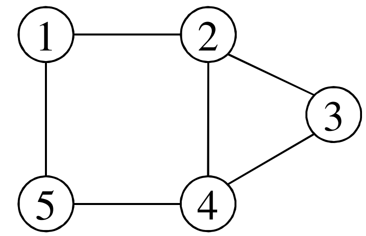




Merge equivalent nodes

Node Merging



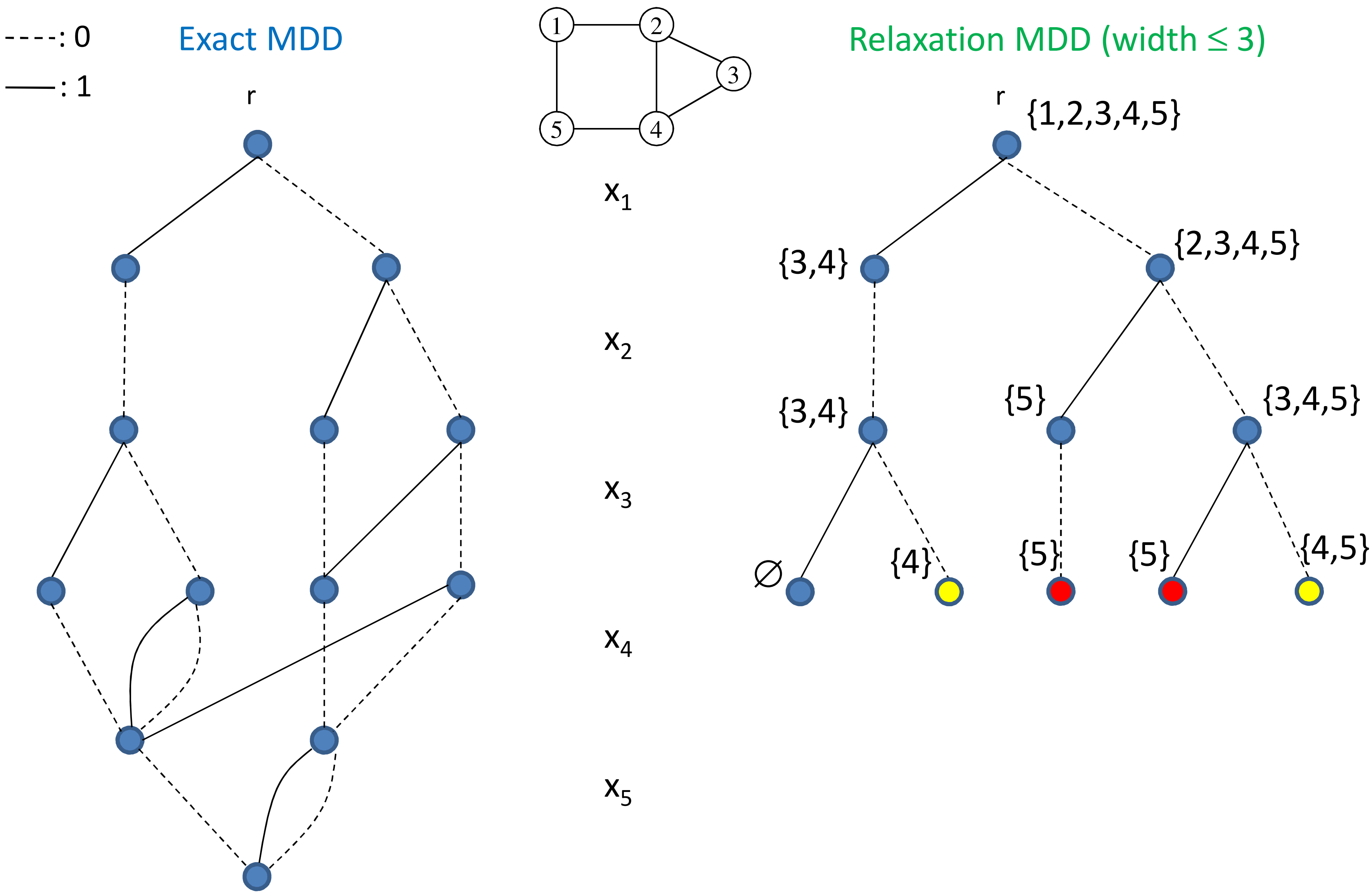


Theorem: This procedure generates an exact MDD

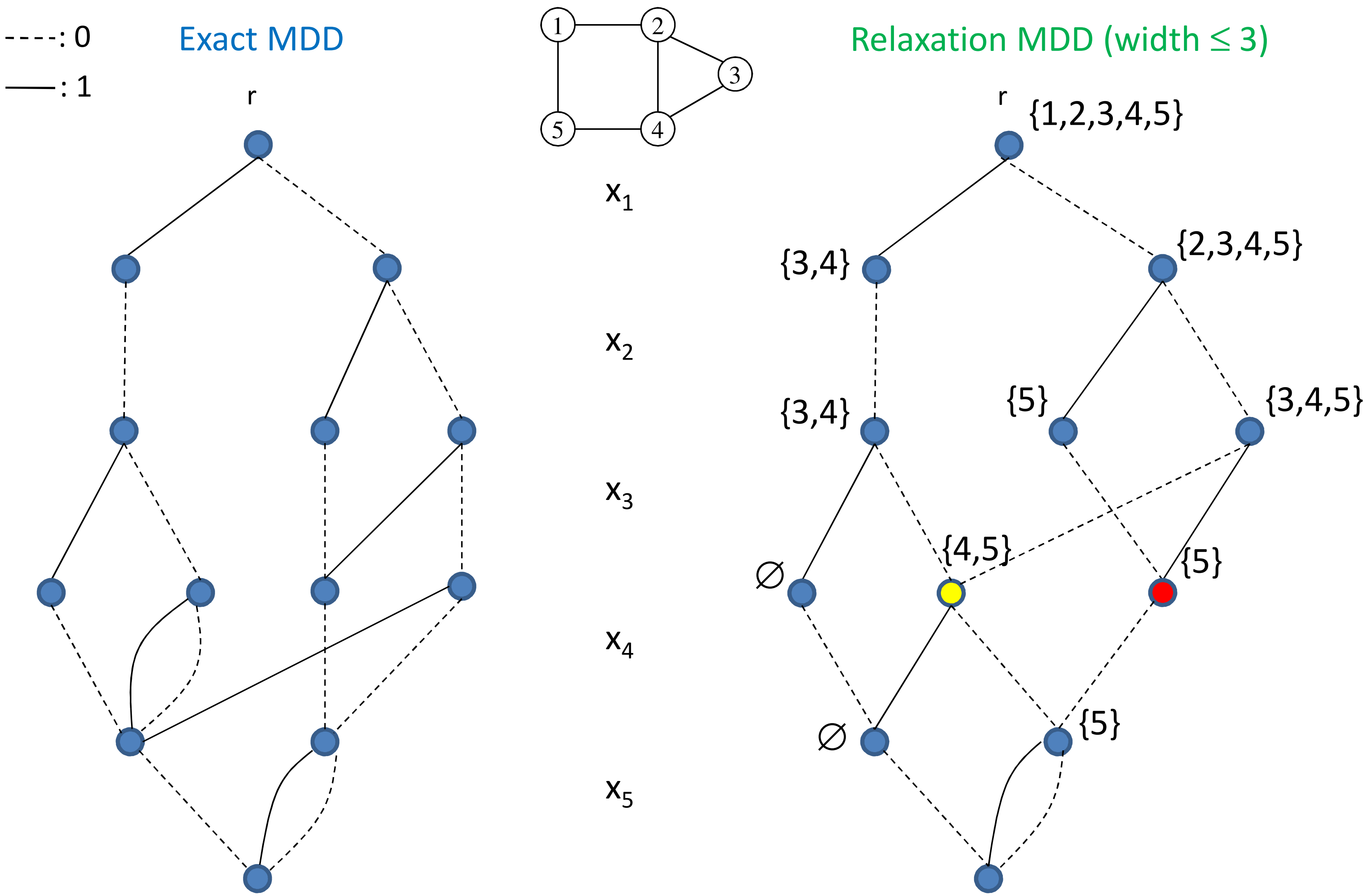
Relaxation MDD: merge *non-equivalent* nodes when the given width is exceeded

[Bergman et al., 2012]

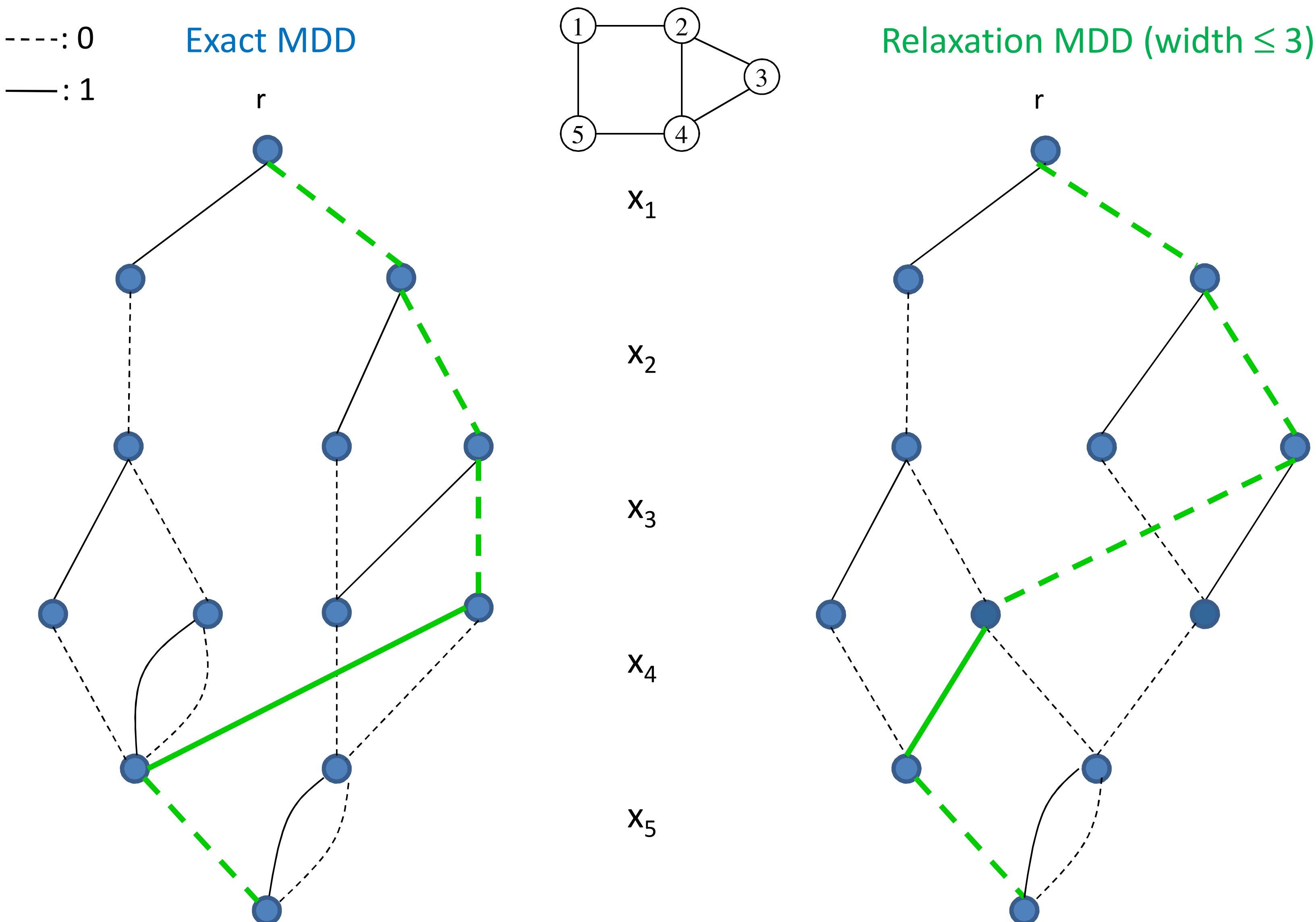
Relaxation MDD



Relaxation MDD

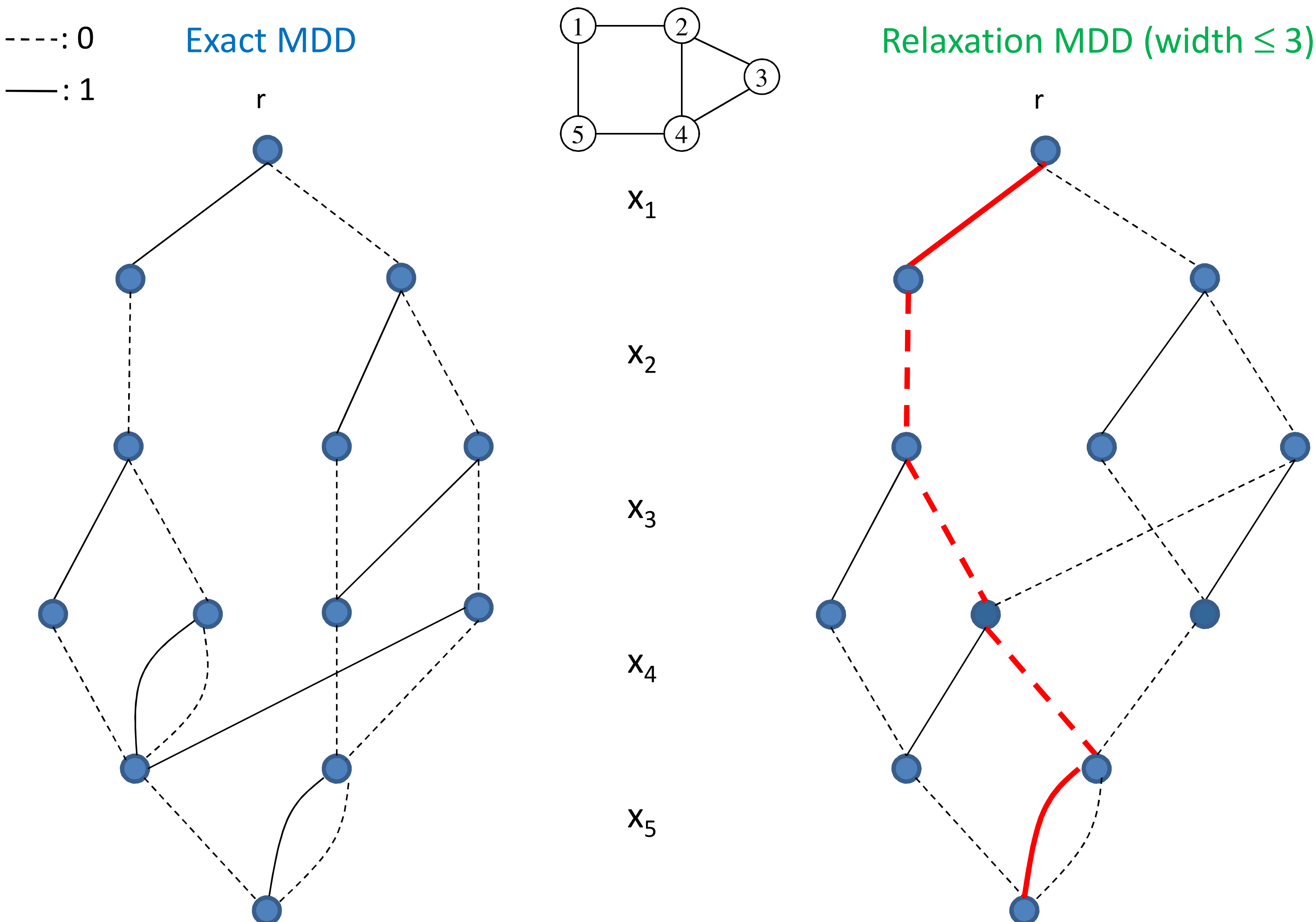


Relaxation MDD



(0,0,0,1,0)

Relaxation MDD

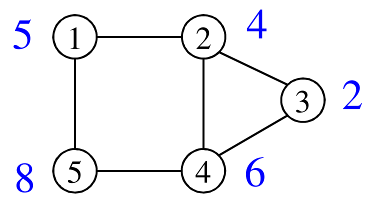


(1,0,0,0,1)

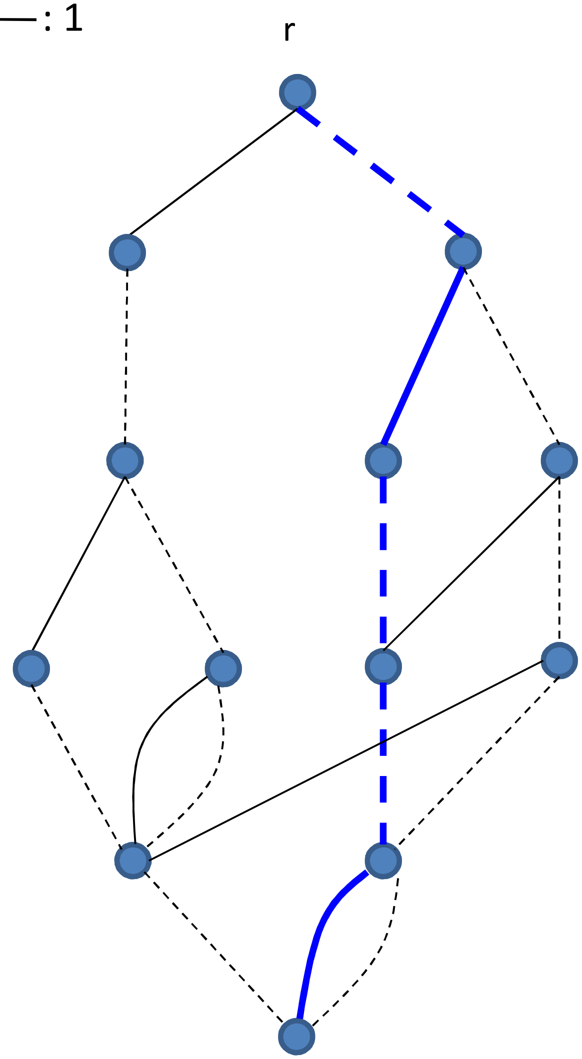
Evaluate Objective Function

- - - -: 0
——: 1

Exact MDD



Relaxation MDD (width ≤ 3)



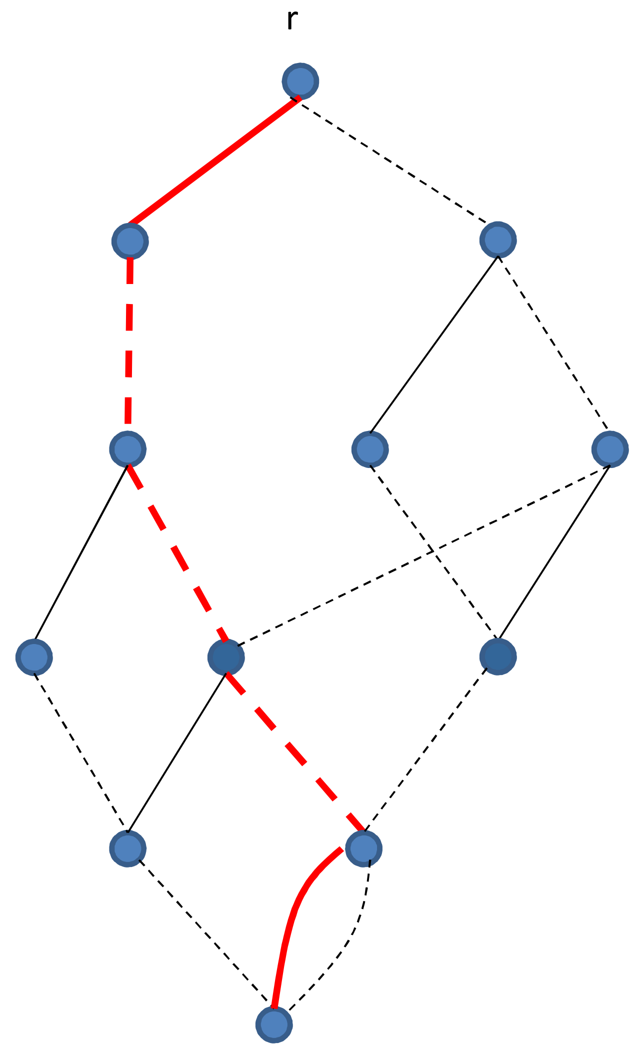
x_1

x_2

x_3

x_4

x_5



max f(x) = 12

max f(x) = 13

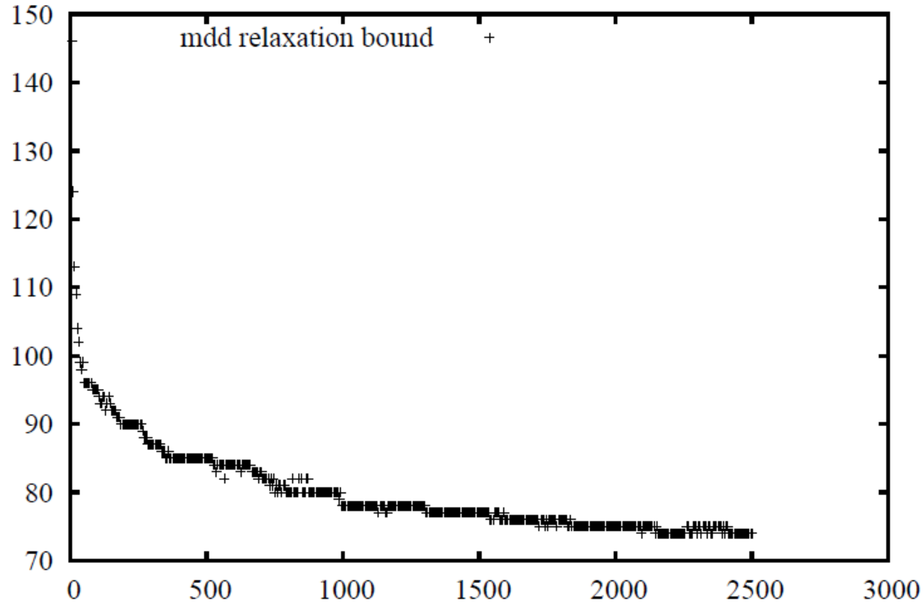
Experimental Results

- Impact of maximum width on strength of bound (and running time)
- Compare MDD bounds to LP bounds
  - IBM ILOG CPLEX 12.4
  - root node relaxation, no presolve, aggressive clique cuts, MIPemphasis
- Time Limit 3,600s

- DIMACS clique instances (unweighted graphs)

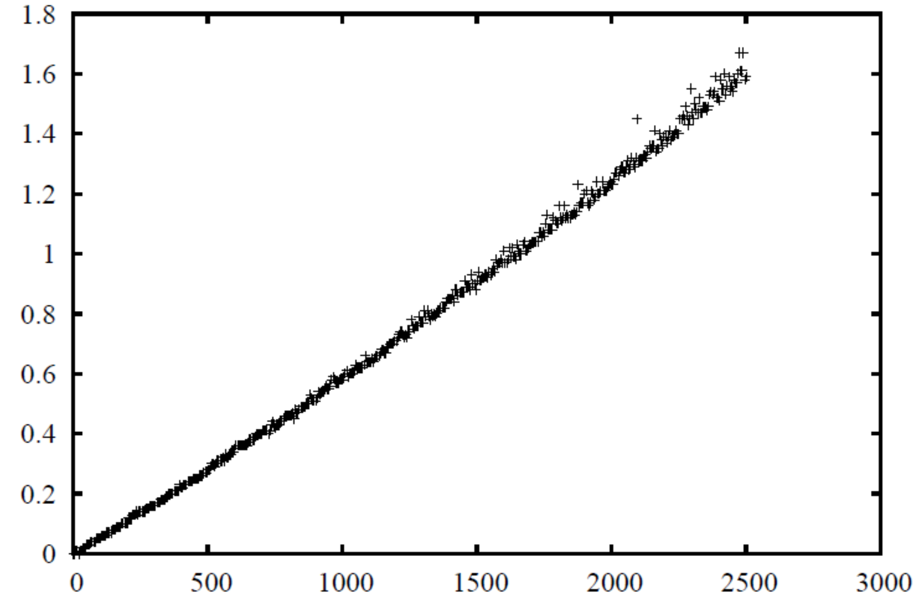
Impact of width on relaxation

upper bound



maximum width

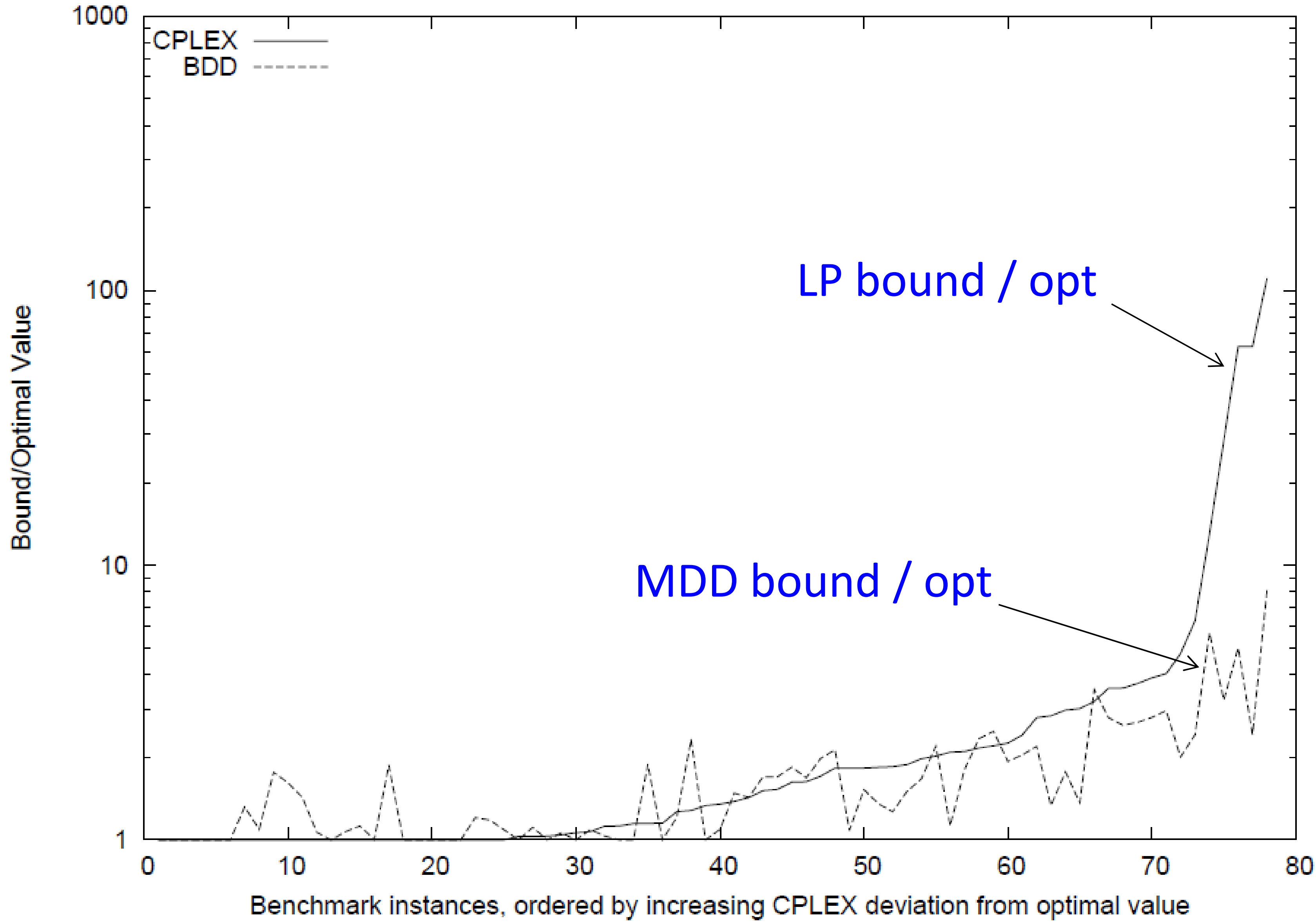
time (s)



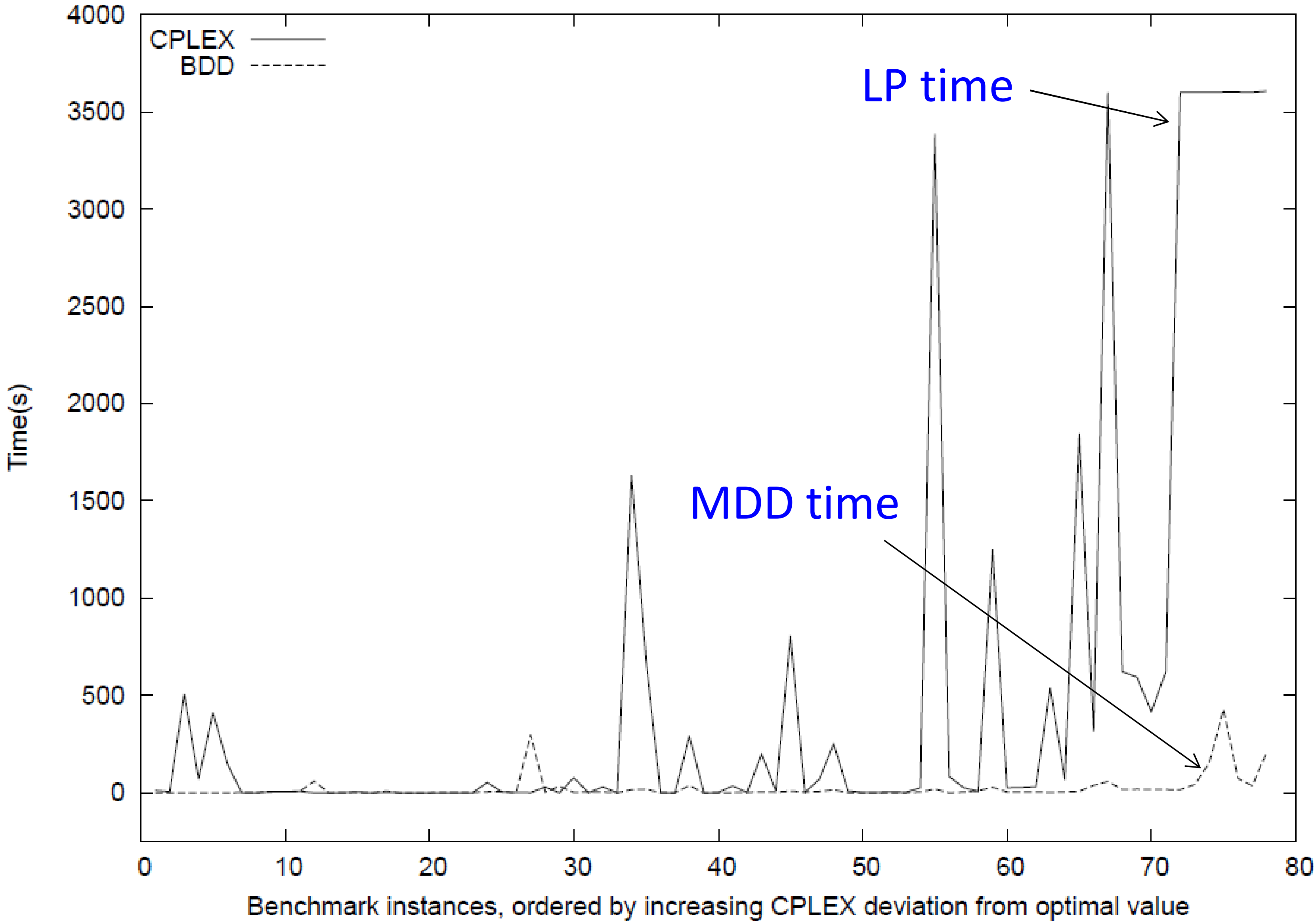
maximum width

brock_200-2 instance

MDD versus LP bounds: Quality



MDD versus LP bounds: Time



Restriction MDDs

- Relaxation MDDs find upper bounds for independent set problem
- Can we use MDDs to find lower bounds as well (i.e., good feasible solutions)?
- Restriction MDDs represent a subset of feasible solutions
  - we require that every r-s path corresponds to a feasible solution
  - but not all solutions need to be represented
- Goal: Use restriction MDDs as a heuristic to find good feasible solutions

Creating Restriction MDDs

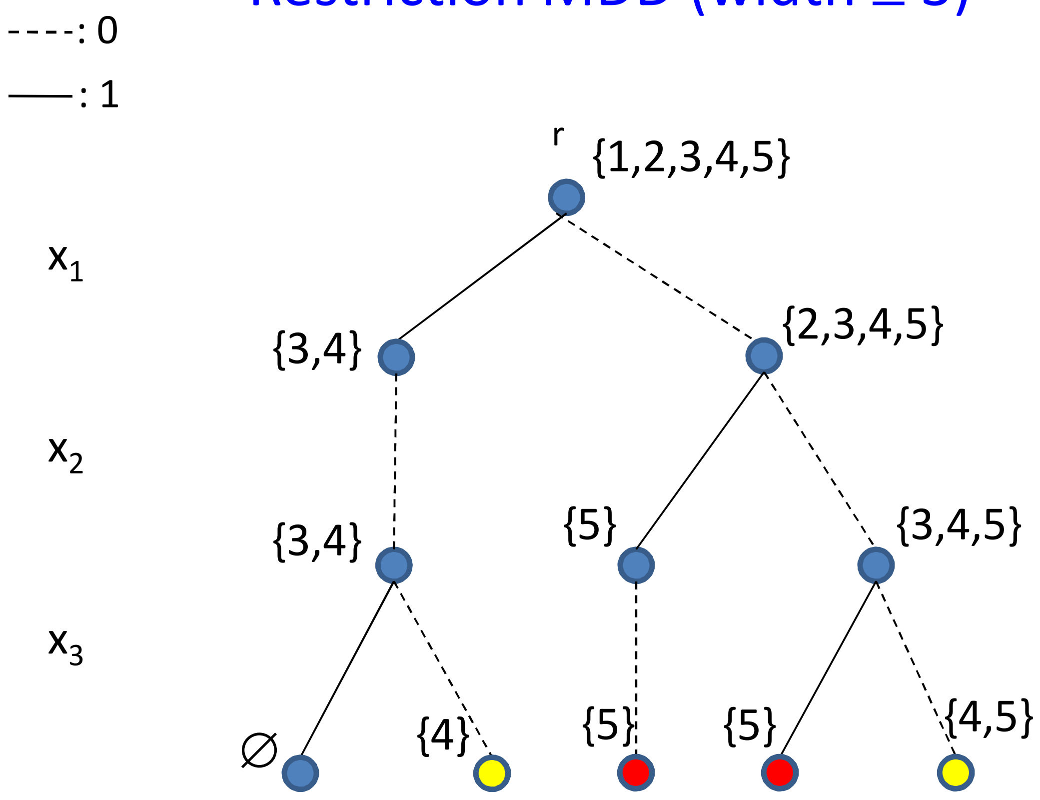
Using an exact top-down compilation method, we can create a limited-width restriction MDD by

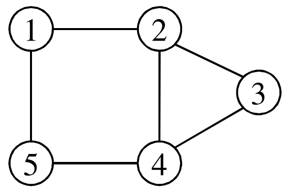
1. merging nodes, or
2. deleting nodes

while ensuring that no solution is lost

Node merging by example

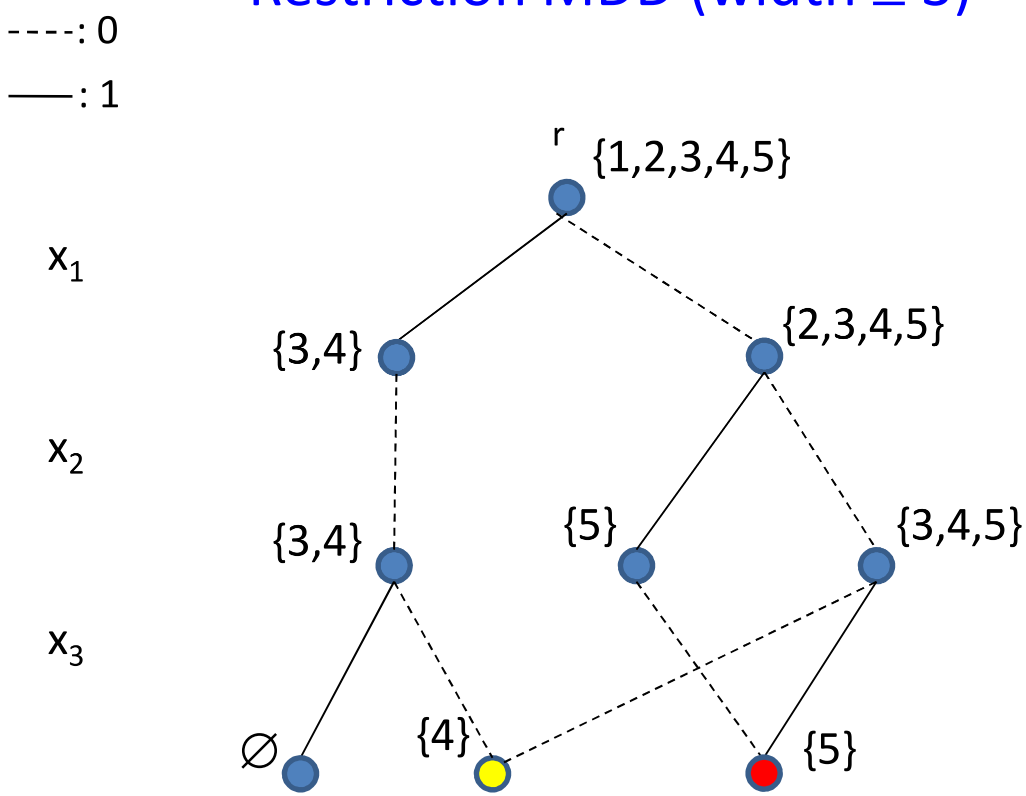
Restriction MDD (width ≤ 3)

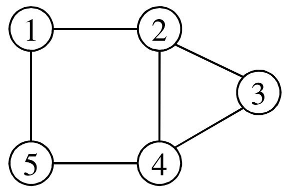




Node merging by example

Restriction MDD (width ≤ 3)



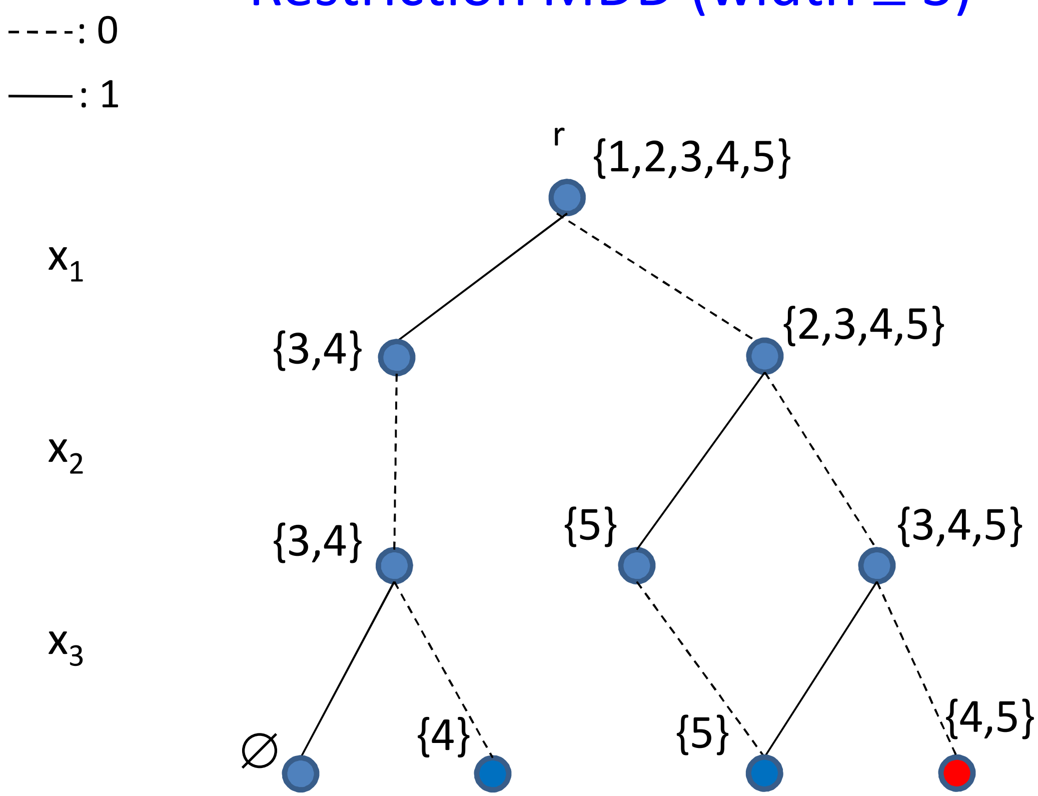


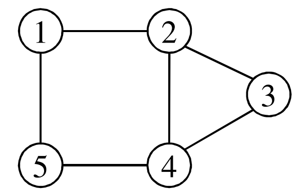
Node merging heuristics

- Random
  - select two nodes $\{u_1, u_2\}$ uniformly at random
- Objective-driven
  - select two nodes $\{u_1, u_2\}$ such that $f(u_1), f(u_2) \leq f(v)$ for all nodes $v \neq u_1, u_2$ in the layer
- Similarity
  - select two nodes $\{u_1, u_2\}$ that are 'closest'
  - problem dependent (or based on semantics)

Node deletion by example

Restriction MDD (width ≤ 3)





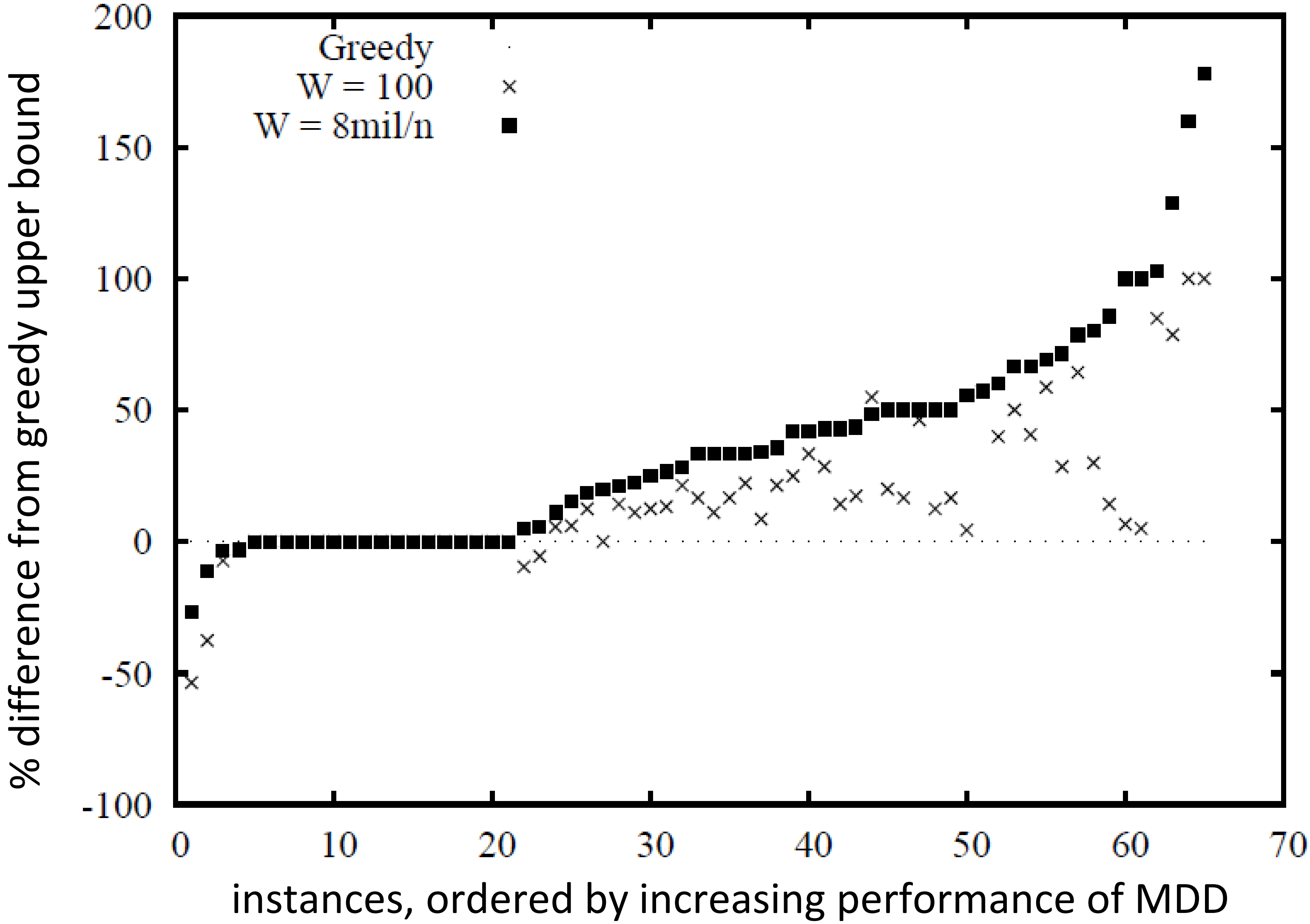
Node deletion heuristics

- Random
  - select node u uniformly at random
- Objective-driven
  - select node u such that
    $f(u) \leq f(v)$ for all nodes $v \neq u$ in the layer
- Information-driven
  - problem specific

Experimental Results

- Comparison to greedy heuristic
  - select vertex v with smallest degree and add it to independent set
  - remove v and its neighbors and repeat
- DIMACS instance set
- MDD version 1: maximum width 100
  - time comparable to greedy heuristic (max 0.25s)
- MDD version 2: maximum width 8,000,000/n
  - maximum time 13s

Greedy versus MDD: Quality



Summary for MDD-Optimization

- Limited-width MDDs can provide useful bounds for discrete optimization
  - The maximum width provides a natural trade-off between computational efficiency and strength
  - Both lower and upper bounds
  - Generic discrete relaxation and restriction method for MIP-style problems
- So far, mainly combinatorial applications
  - Independent Set Problem, Set Covering Problem, Set Packing Problem

Open issues

- Extend application to CP
  - Which other global constraints are suitable? (Cumulative?)
  - Can we develop search heuristics based on the MDD?
  - Can we more efficiently store and manipulate approximate MDDs? (Implementation issues)
  - Can we obtain a tighter integration with CP domains?
- MDD technology
  - Variable ordering is crucial for MDDs. What can we do if the ordering is not clear from the problem statement?
  - How should we handle constraints that partially overlap on the variables? Build one large MDD or have partial MDDs communicate?

Open issues (cont'd)

- Formal characterization
  - Can MDDs be used to identify tractable classes of CSPs?
  - Can we identify classes of global constraints for which establishing MDD consistency is hard/easy?
  - Can MDDs be used to prove approximation guarantees?
  - Can we exploit a connection between MDDs and tight LP representations of the solution space?
- Optimization
  - Approximate MDDs can provide bounds for any nonlinear (separable) objective function. Demonstrate the performance on an actual application.

Open issues (cont'd)

- Beyond classical CP
  - How can MDDs be helpful in presence of uncertainty? E.g., can we use approximate MDDs to represent policy trees for stochastic optimization? [Cire, Coban, v.H., 2012]
  - Can we utilize approximate MDDs for SAT?
  - Can MDDs help generate nogoods, e.g., in lazy clause generation?
  - Can we exploit a tighter integration of MDDs in MIP solvers?
- Applications
  - So far we have looked mostly at generic problems. Are there specific applications for which MDDs work particularly well? (Bioinformatics?)

Summary

What can MDDs do for discrete optimization?

- *Compact representation* of all solutions to a problem
- Limit on size gives *approximation*
- Control strength of approximation by size limit

MDDs for Constraint Programming

- MDD propagation natural generalization of domain propagation
- Orders of magnitude improvement possible

MDDs for optimization (CP/ILP/MINLP)

- MDDs provide *discrete relaxations*
- Much stronger bounds can be obtained in much less time

Many opportunities: search, stochastic programming, integrated methods, theory, …