Decision Diagrams for Discrete Optimization

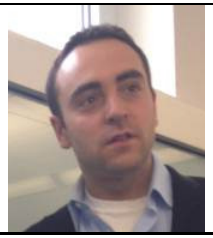
Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

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

Acknowledgments:






David Bergman André Ciré Samid Hoda John Hooker Brian Kell Tallys Yunes

Summary

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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 and Scheduling

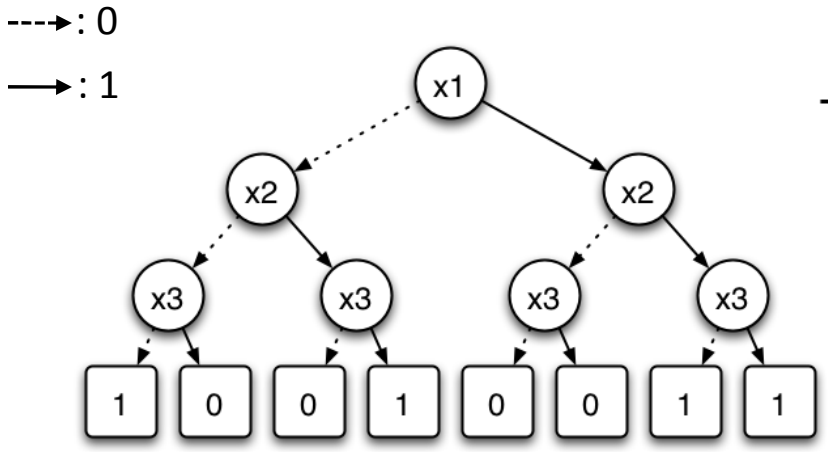
- MDD propagation natural generalization of domain propagation
- Orders of magnitude improvement possible

MDDs for optimization (CP/ILP/MINLP)

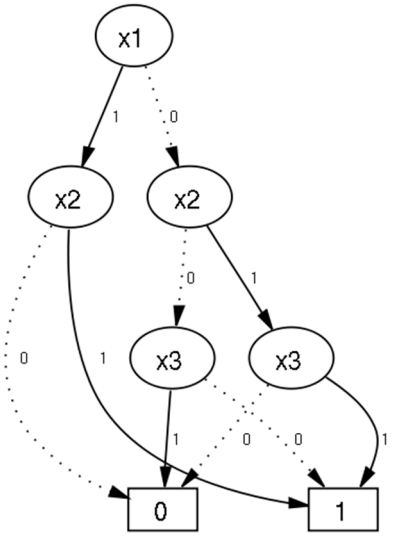
- MDDs provide *discrete relaxations*
- Much stronger bounds can be obtained in much less time

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

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)

Brief background

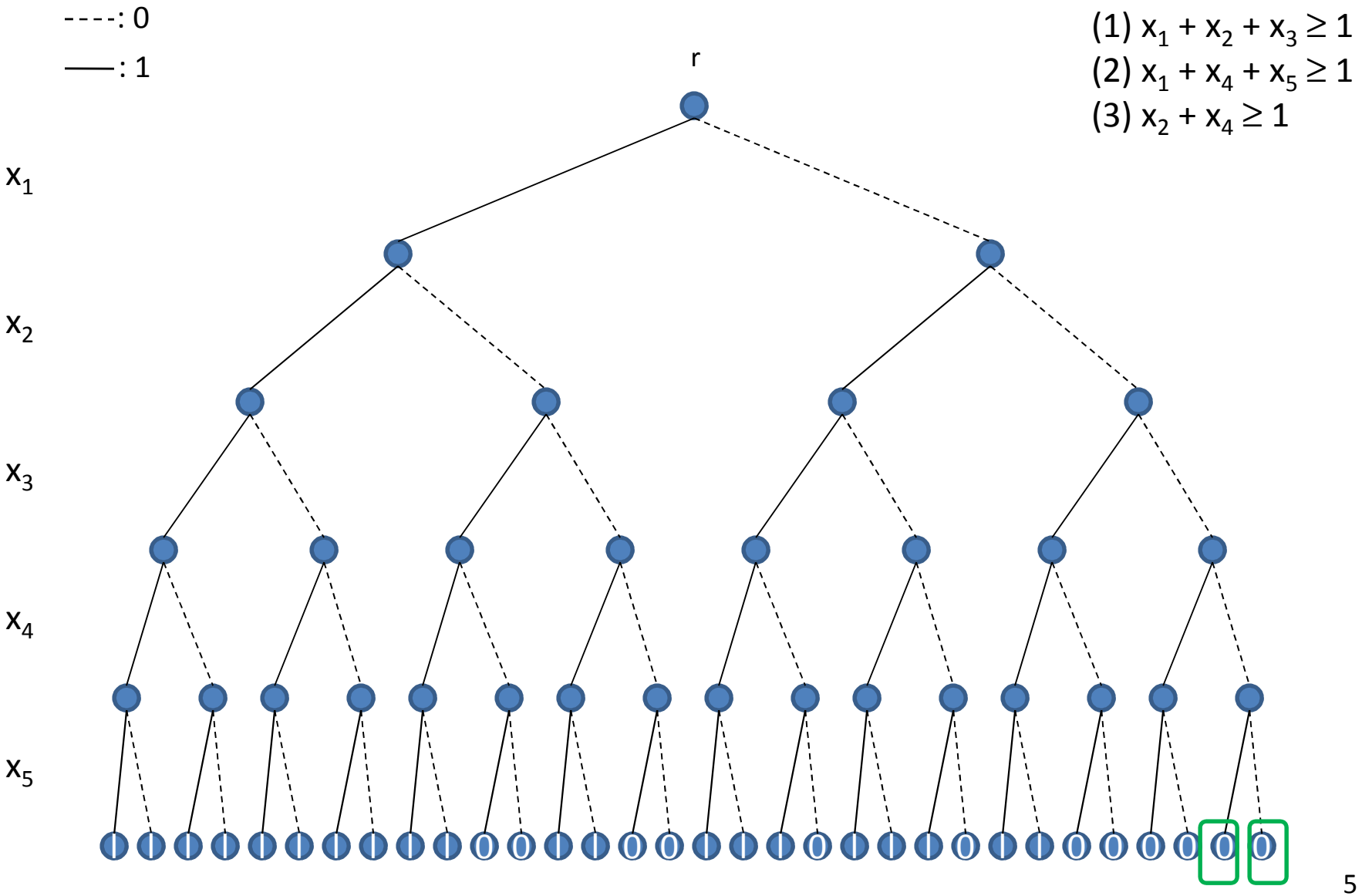
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

    [Andersen, Hadzic, Hooker & Tiedemann, CP 2007]

Exact MDDs for discrete optimization

Carnegie Mellon Tepper SCHOOL OF BUSINESS



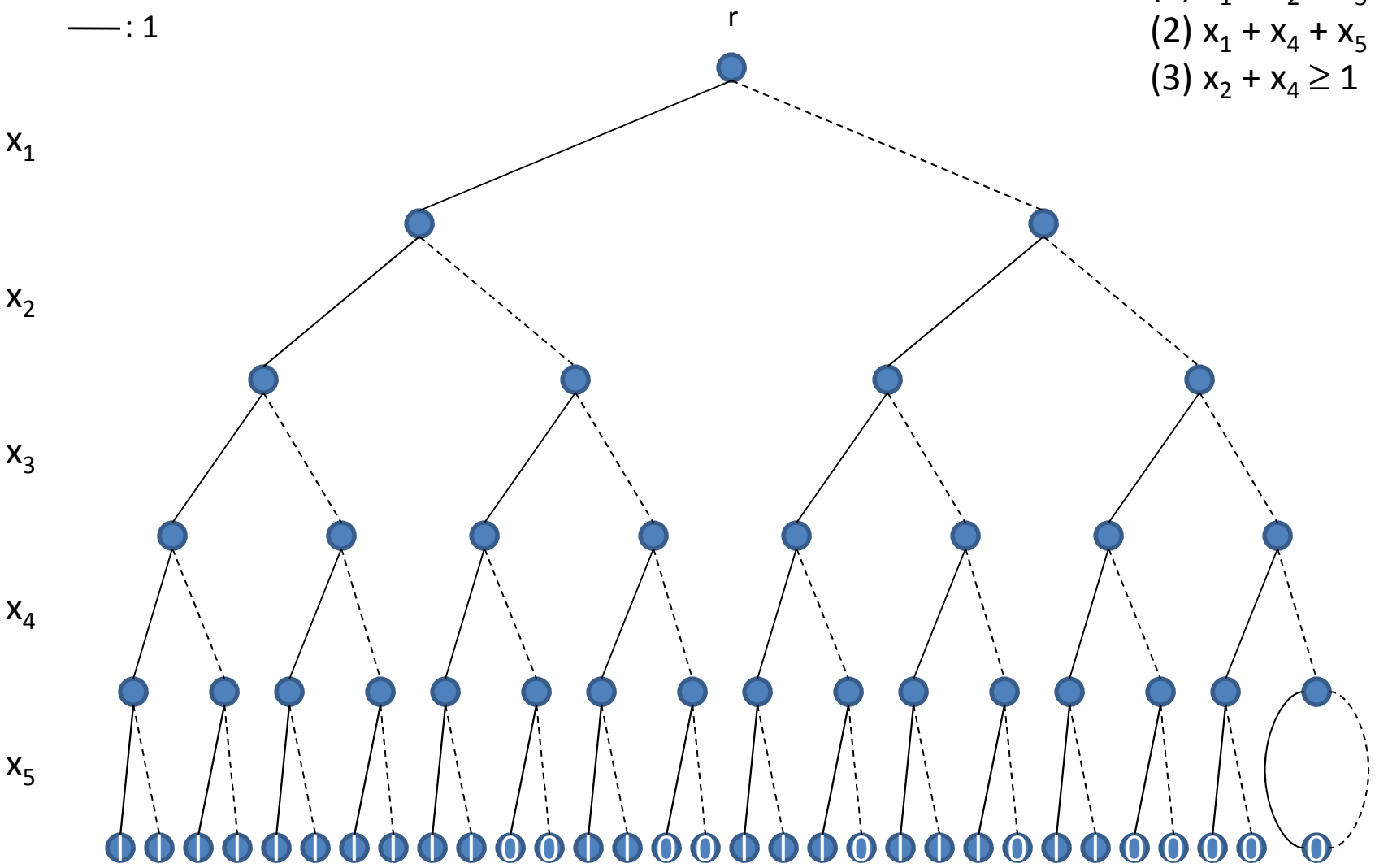
Exact MDDs for discrete optimization

Carnegie Mellon Tepper SCHOOL OF BUSINESS

----: 0

——: 1

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



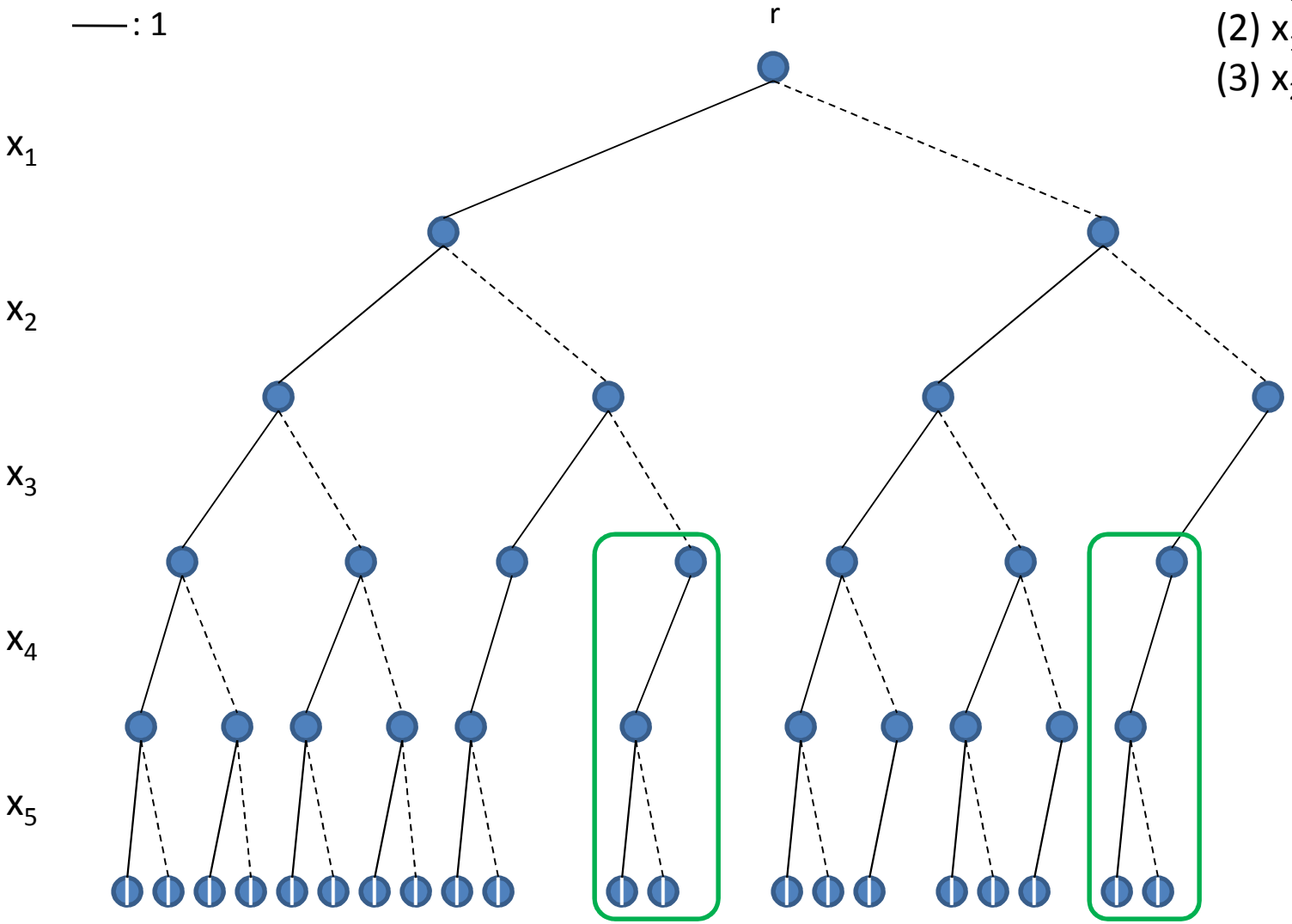
Exact MDDs for discrete optimization

Carnegie Mellon Tepper SCHOOL OF BUSINESS

----: 0

——: 1

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



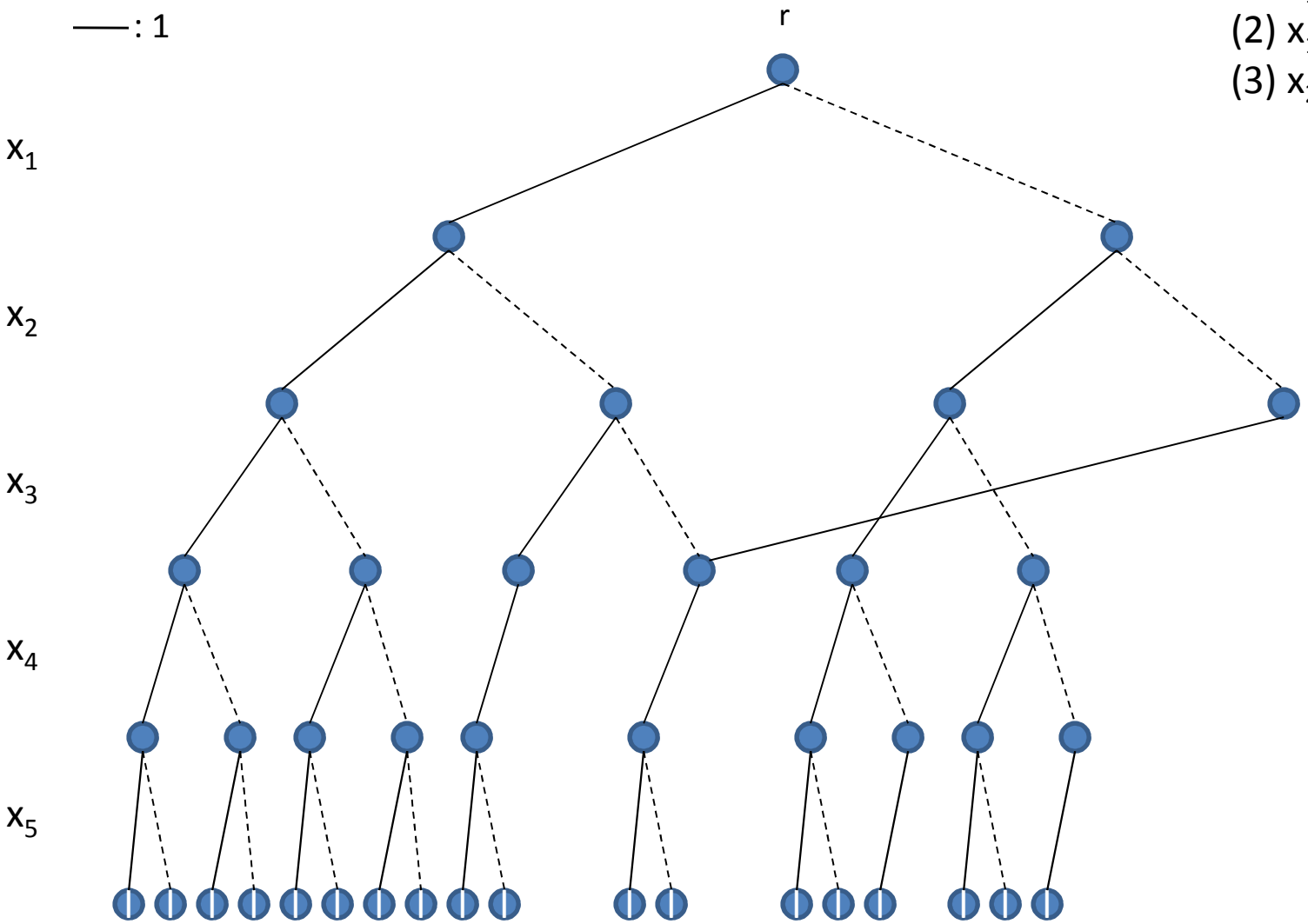
Exact MDDs for discrete optimization

Carnegie Mellon Tepper SCHOOL OF BUSINESS

----: 0

——: 1

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



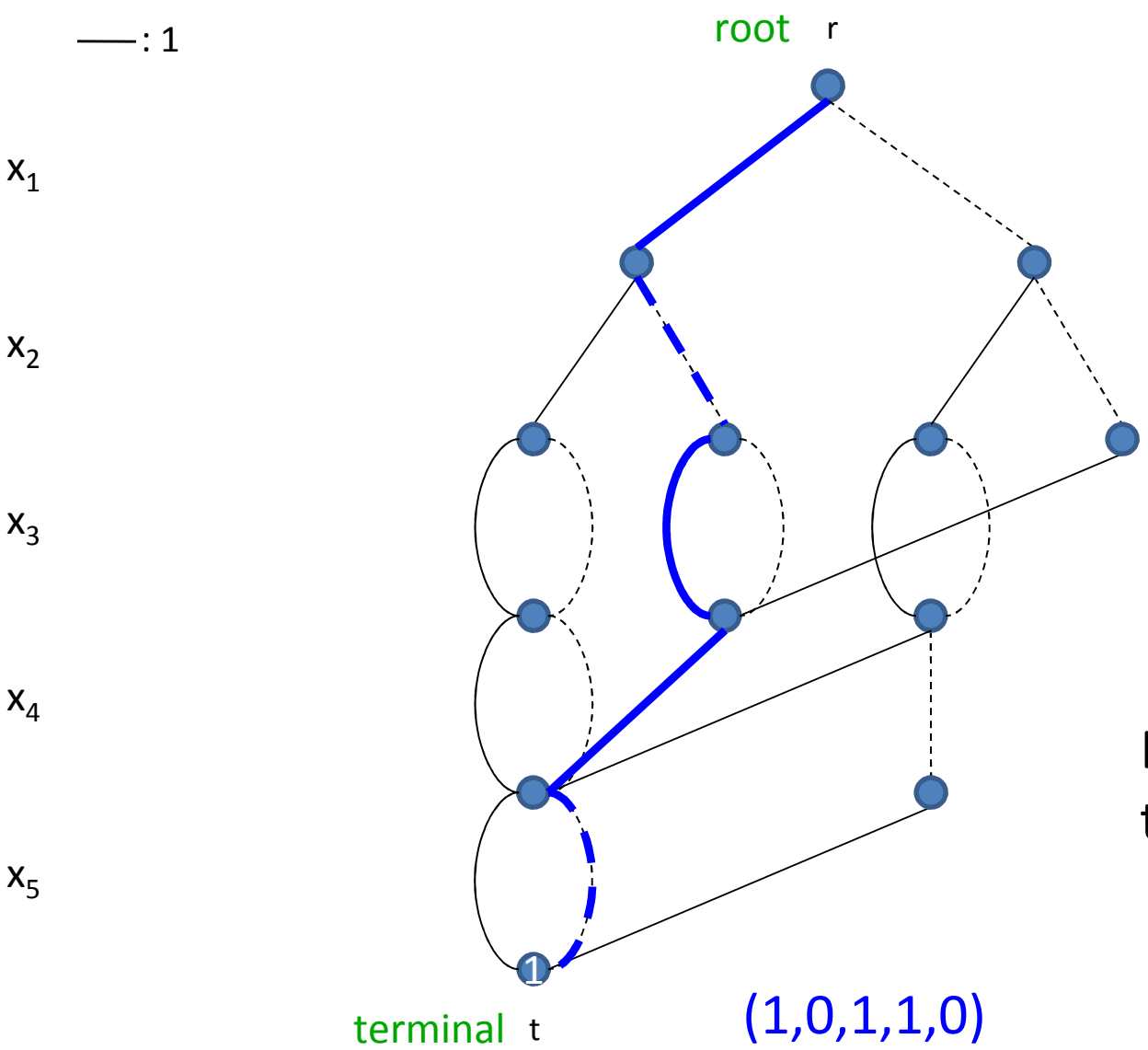
Exact MDDs for discrete optimization

Carnegie Mellon Tepper SCHOOL OF BUSINESS

----: 0

——: 1

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



(1,0,1,1,0)

Each path corresponds to a solution

Approximate MDDs

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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)

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Related Work: Exact MDDs for Constraint Programming

- Set bounds propagation [Hawkins, Lagoon, Stuckey, 2005], [Gange, Lagoon, Stuckey, 2008]
- Ad-hoc Table constraints [Cheng and Yap, 2008]
- Market Split Problem [Hadzic et al., 2009]
- Explanations for MDD propagators [Gange, Stuckey, Szymanek, 2011]
- Regular constraint [Cheng, Xia, Yap, 2012]
- Explaining Propagators for Edge-valued Decision Diagrams [Gange, Stuckey, Van Hentenryck, 2013]

Related Work: Exact MDDs for Integer Programming

- BDDs for IP cut generation: Becker et al. [2005]
- BDDs for 0/1 vertex and facet enumeration: Behle, Eisenbrand [2007]
- Post-optimality analysis: Hadzic, Hooker [2006, 2007]

References (Approximate MDDs)

Carnegie Mellon Tepper SCHOOL OF BUSINESS

MDD-Based Constraint Programming

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

Specific MDD Propagation Algorithms

- Hadzic, Hooker, Tiedemann: Propagating Separable Equalities in an MDD Store. CPAIOR 2008: 318-322
- Ciré, v.H.: MDD Propagation for Disjunctive Scheduling. ICAPS 2012: 11-19 (Extended version 'MDDs for Sequencing Problems' under review)
- Ciré, v.H.: MDD Propagation for Sequence Constraints. Tepper School of Business Working Paper 2011-E12, Carnegie Mellon University, 2011

References (cont'd)

Carnegie Mellon Tepper SCHOOL OF BUSINESS

MDD-Based Optimization

- Bergman, v.H., Hooker: Manipulating MDD Relaxations for Combinatorial Optimization. CPAIOR 2011: 20-35
- Bergman, Ciré, v.H., Hooker: Variable Ordering for the Application of BDDs to the Maximum Independent Set Problem. CPAIOR 2012: 34-49
- Bergman, Ciré, v.H., Hooker: Optimization Bounds from Binary Decision Diagrams. INFORMS J. Computing, to appear.
- Bergman, Ciré, v.H., Yunes: BDD-Based Heuristics for Binary Optimization. Under review, 2013.
- Kell, v.H.: An MDD Approach to Multidimensional Bin Packing. CPAIOR 2013: 128-143

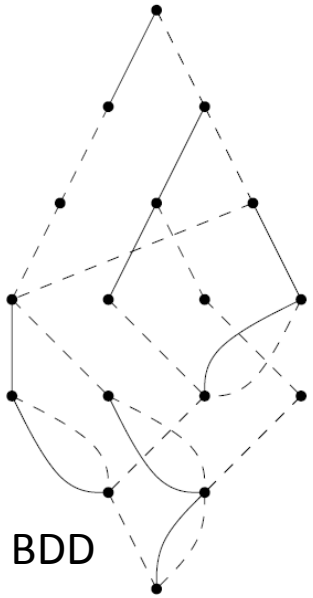
MDDs and Dynamic Programming

- Hooker: Decision Diagrams and Dynamic Programming. CPAIOR 2013: 94-110

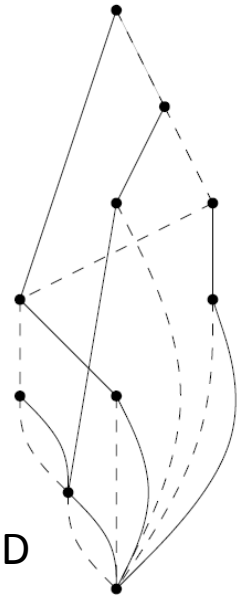
Suppressed Decision Diagrams

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Zero-suppressed BDD (0-BDD or ZDD)
  - arc skips layers for which variables will take value 0
- One-suppressed BDD (1-BDD)
  - arc skips layers for which variables will take value 1
- Zero/one-suppressed BDD (0/1-BDD)
  - arc skips layers for which variables will take value 0/1



standard BDD



0-BDD

- Similarly compressed MDDs can be defined
- Will not be discussed in detail, but methodology can be extended

MDDs for Constraint Programming

Motivation

Constraint Programming applies

- systematic search and
- inference techniques

to solve discrete optimization problems

Inference mainly takes place through:

- Filtering provably inconsistent values from variable domains
- Propagating the updated domains to other constraints

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

$x_1 < x_2$ → $x_2 \in \{2,3\}$

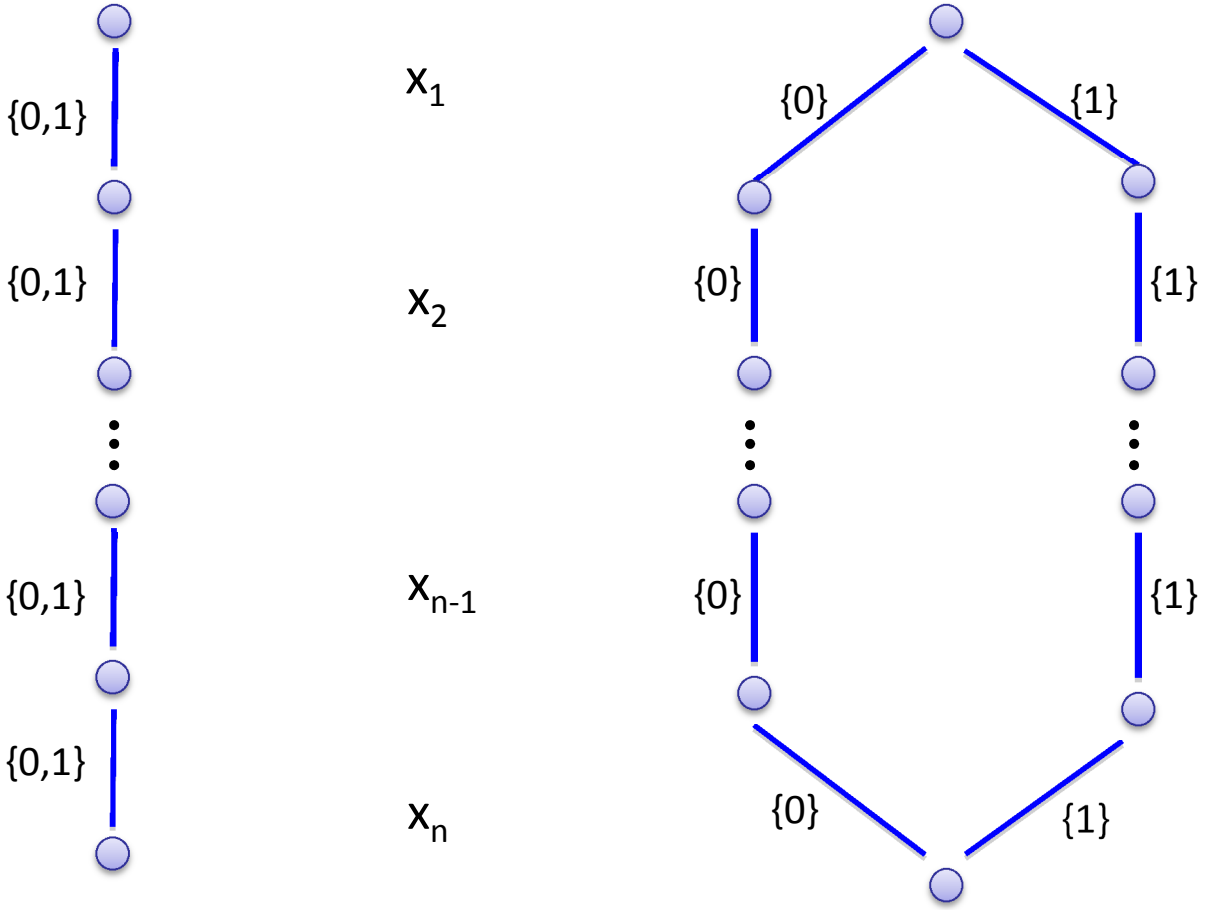
alldifferent(x_1,x_2,x_3) → $x_1 \in \{1\}$

Illustrative Example

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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 *relaxed* MDD
- Strength is controlled by the imposed width

MDD-based Constraint Programming

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

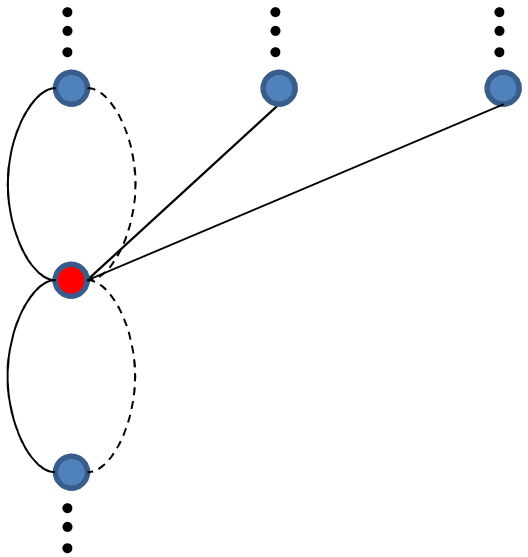
Specific MDD propagation algorithms

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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, 2013]
- *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

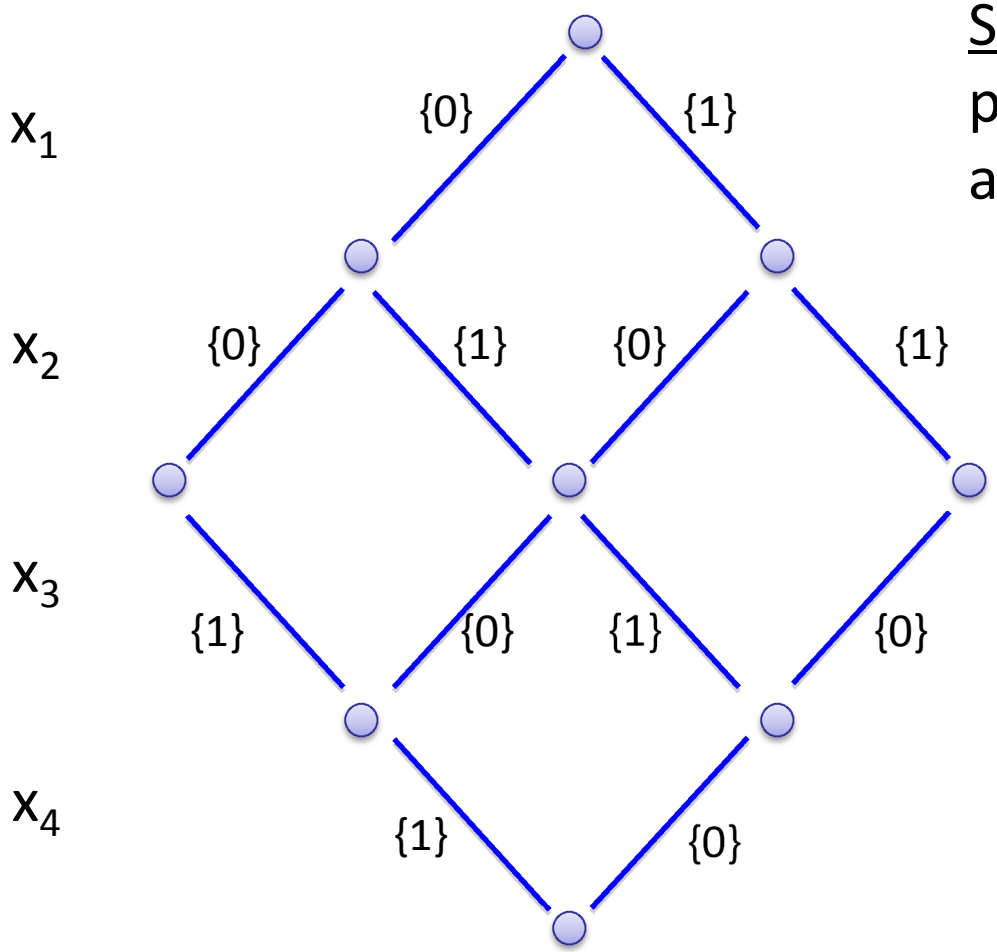
- Given a set of variables X, and a set of values S, a lower bound l and upper bound u,

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

"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



State information:
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

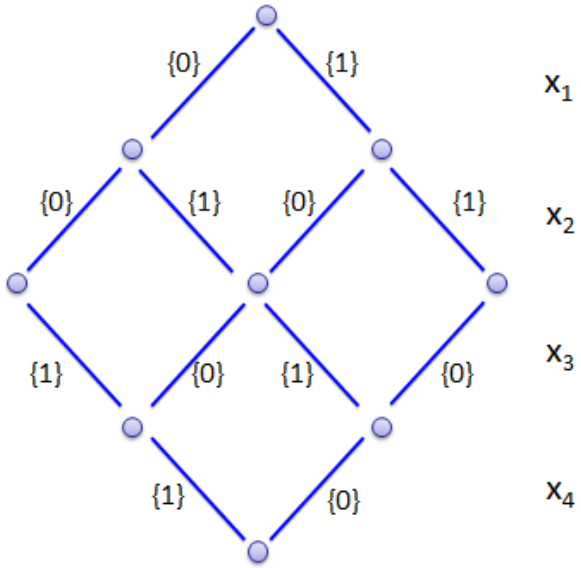
Carnegie Mellon Tepper SCHOOL OF BUSINESS

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 inconsist
  - Maintain and check bounds (longest and shortest



Node refinement for Among

Carnegie Mellon Tepper SCHOOL OF BUSINESS

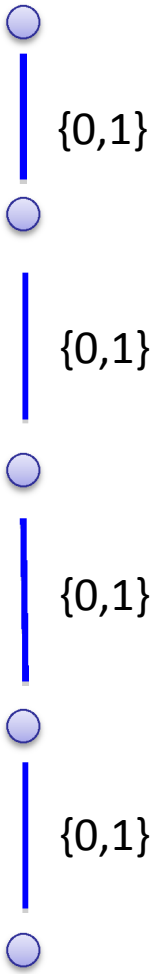
For each layer in MDD, we first apply edge filter, and then try to refine

- 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:

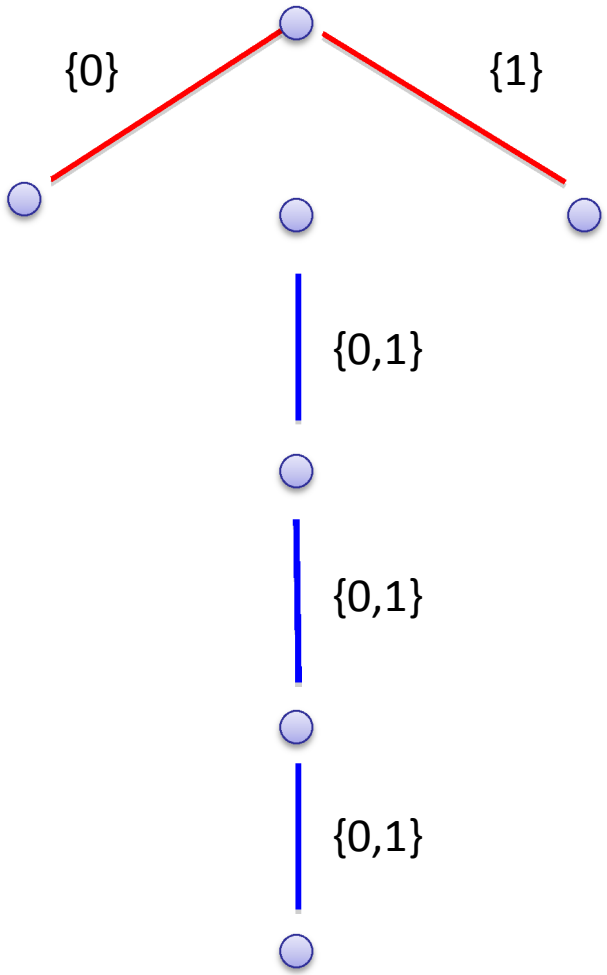
- We will propagate *Among*($\{x_1,x_2,x_3,x_4\}$,{1},2,2) through a BDD of maximum width 3

Example



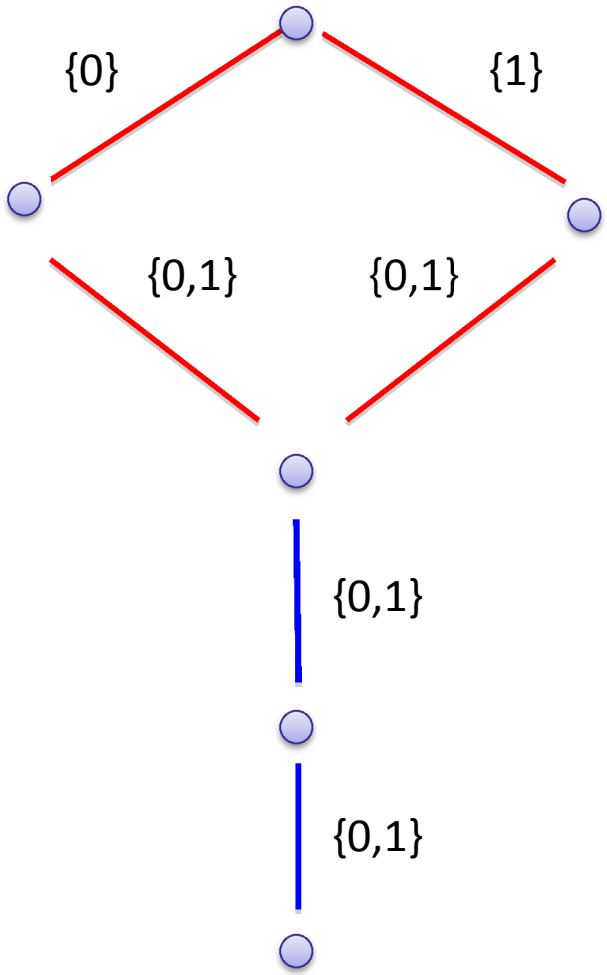
Among($\{x_1,x_2,x_3,x_4\}$,{1},2,2)

Example



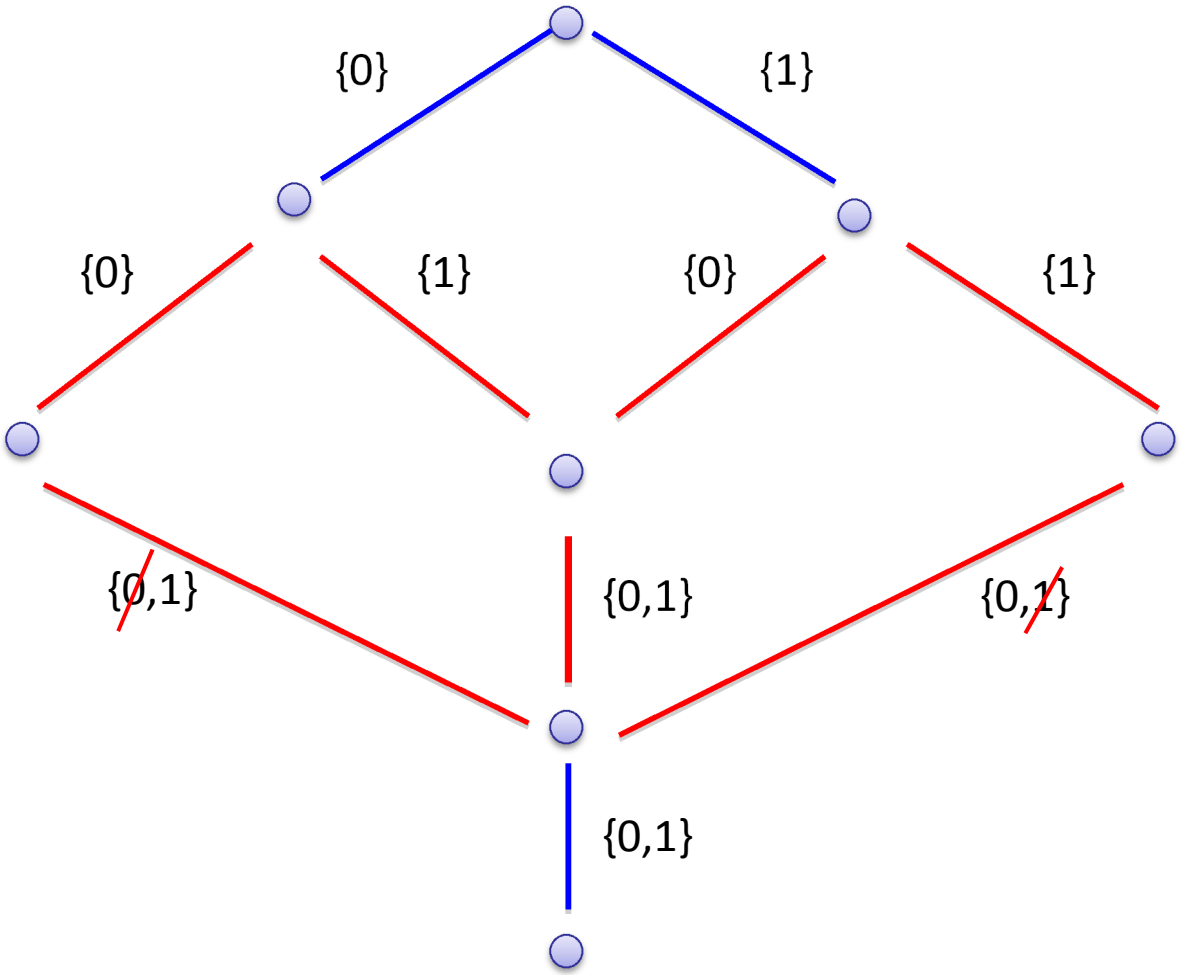
Among($\{x_1,x_2,x_3,x_4\}$,{1},2,2)

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

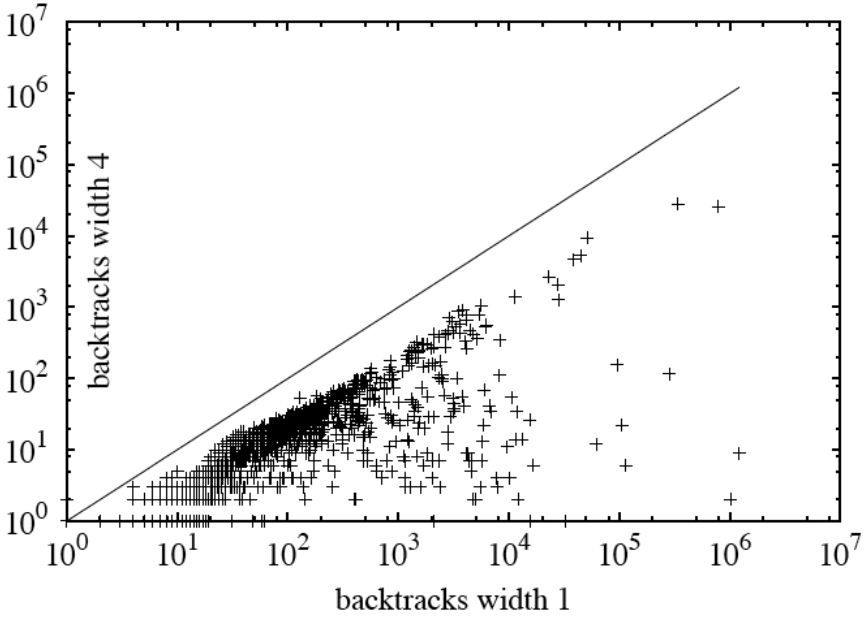
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

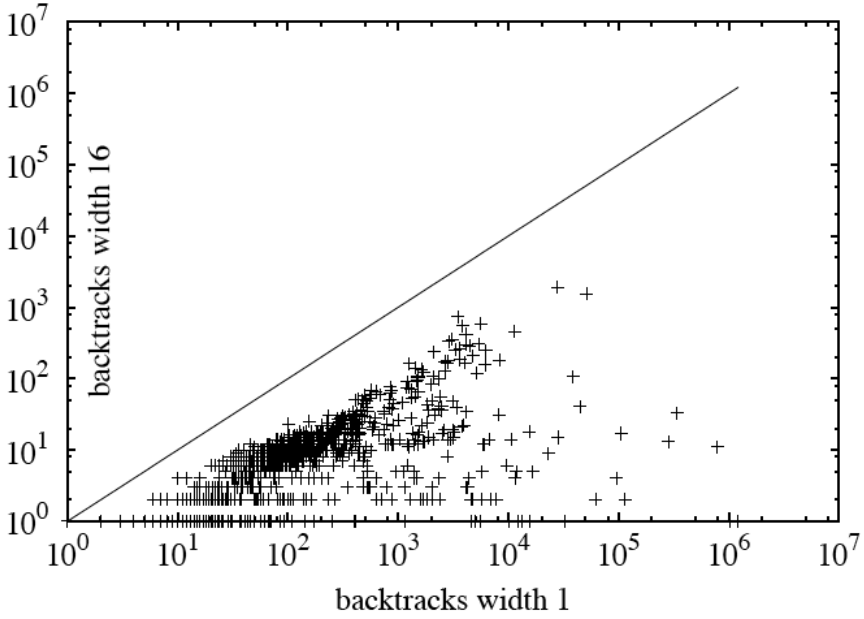
- Nurse rostering instances (horizon *n* days)
  - Work 4-5 days per week
  - Max A days every B days
  - Min C days every D days
  - Three problem classes

- Compare width 1 (traditional domains) with increasing widths

Multiple Amongs: Backtracks

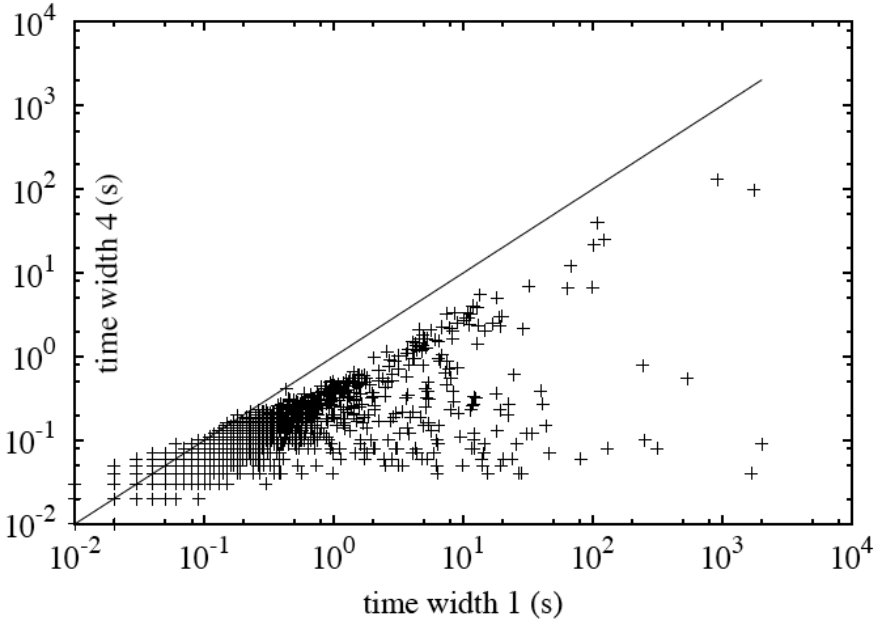


width 1 vs 4

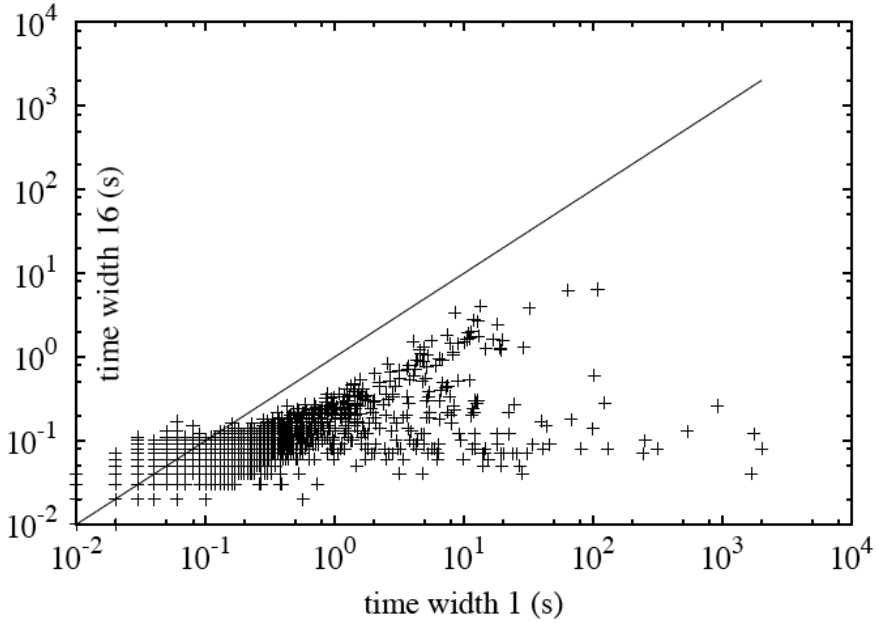


width 1 vs 16

Multiple Amongs: Running Time



width 1 vs 4



width 1 vs 16

Nurse rostering problems

Carnegie Mellon Tepper SCHOOL OF BUSINESS

		Width 1		Width 4		Width 32	
	Size	BT	CPU	BT	CPU	BT	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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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}

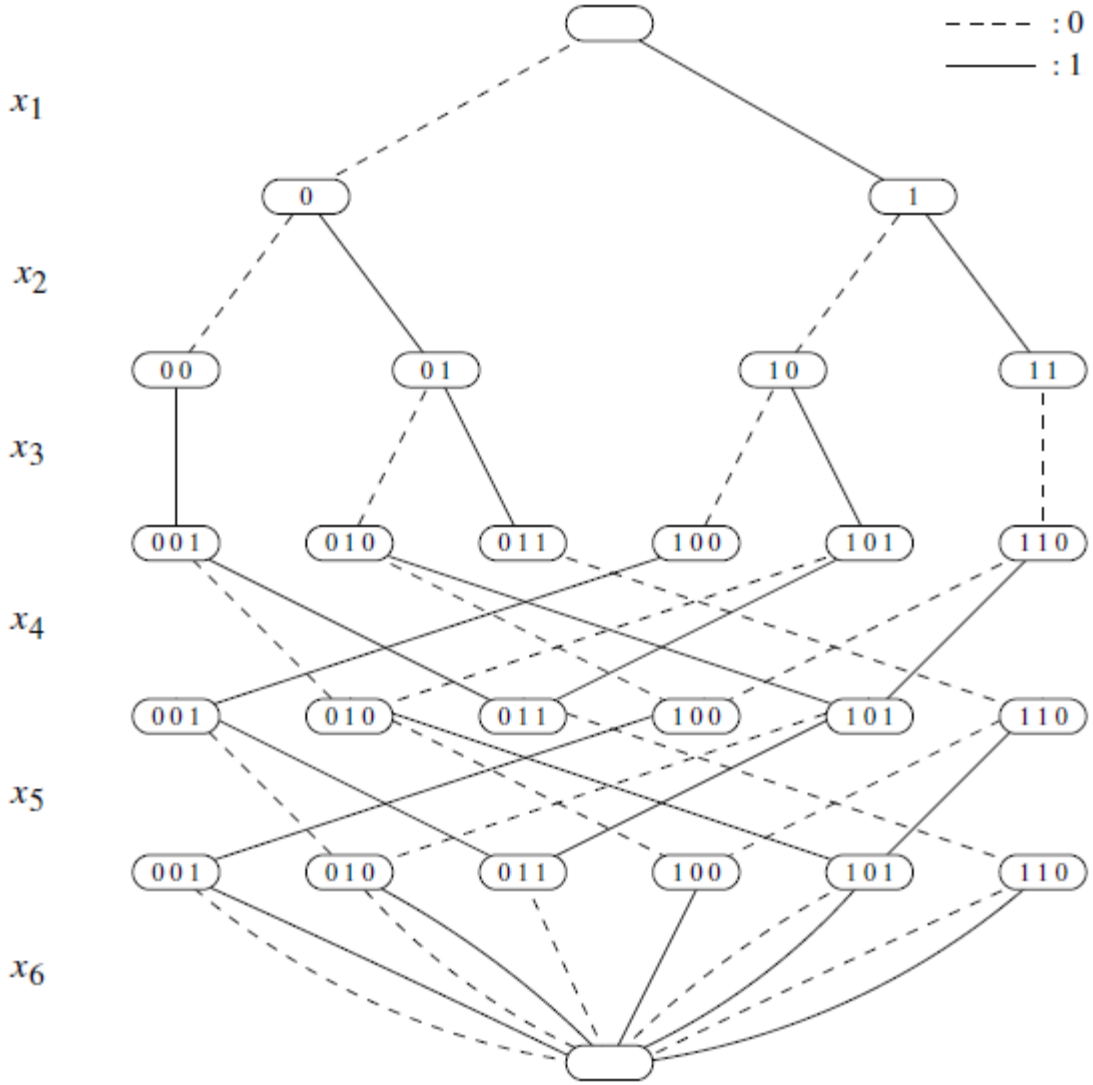
$$\left.\begin{array}{l} 0 \le x_1+x_2+ \ldots +x_9 \le 7 \\ 0 \le x_2+x_3+ \ldots +x_{10} \le 7 \\ 0 \le x_3+x_4+ \ldots +x_{11} \le 7 \\ 0 \le x_4+x_5+ \ldots +x_{12} \le 7 \end{array}\right\} =: \mathit{Sequence}([x_1,x_2,\ldots,x_{12}], q=9, S=\{1\}, l=0, u=7)$$

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

$$\downarrow$$

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

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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

Hardness Proof

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

Proof structure:

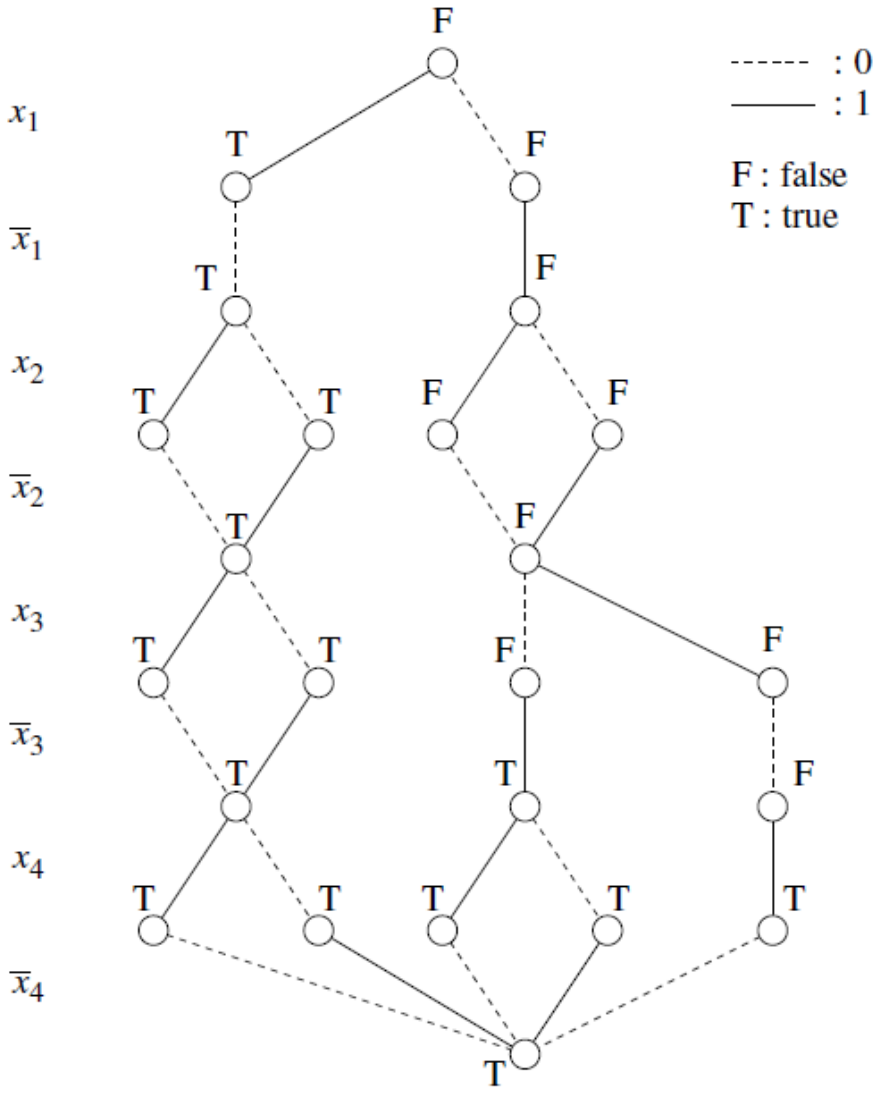
- Given 3-SAT problem (NP-complete)
- We will construct a polynomial-size MDD such that a particular *Sequence* constraint will have a solution in the MDD if and only if the 3-SAT instance is satisfiable

- Example 3-SAT problem

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

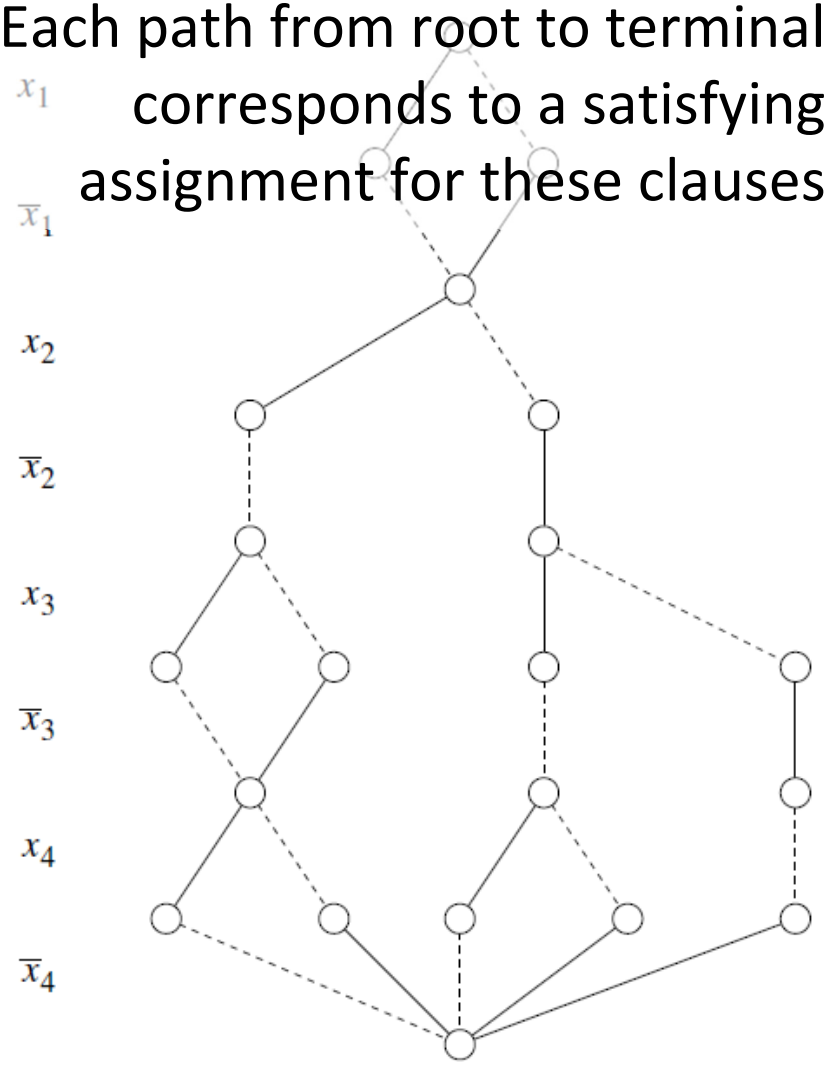
$$c_2 = (x_2 \vee x_3 \vee \overline{x}_4)$$

Single clause representation



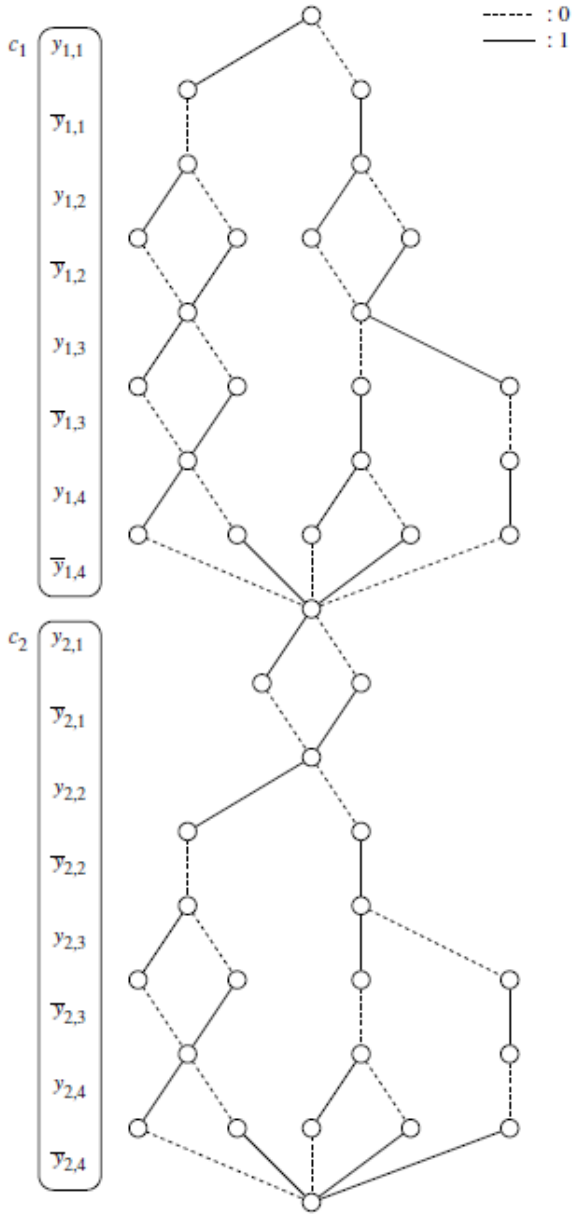
$$c_1 = (x_1 \vee \overline{x}_3 \vee x_4)$$

Each path from root to terminal corresponds to a satisfying assignment for these clauses



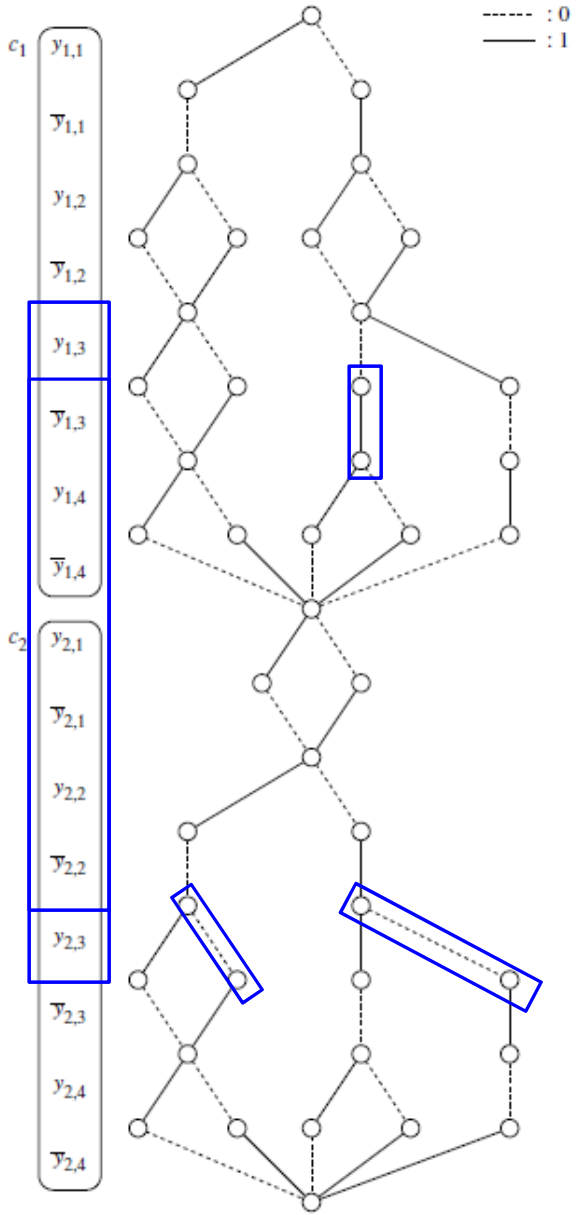
$$c_2 = (x_2 \vee x_3 \vee \overline{x}_4)$$

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

Partial filter from decomposition

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

$$y_0 = 0$$

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

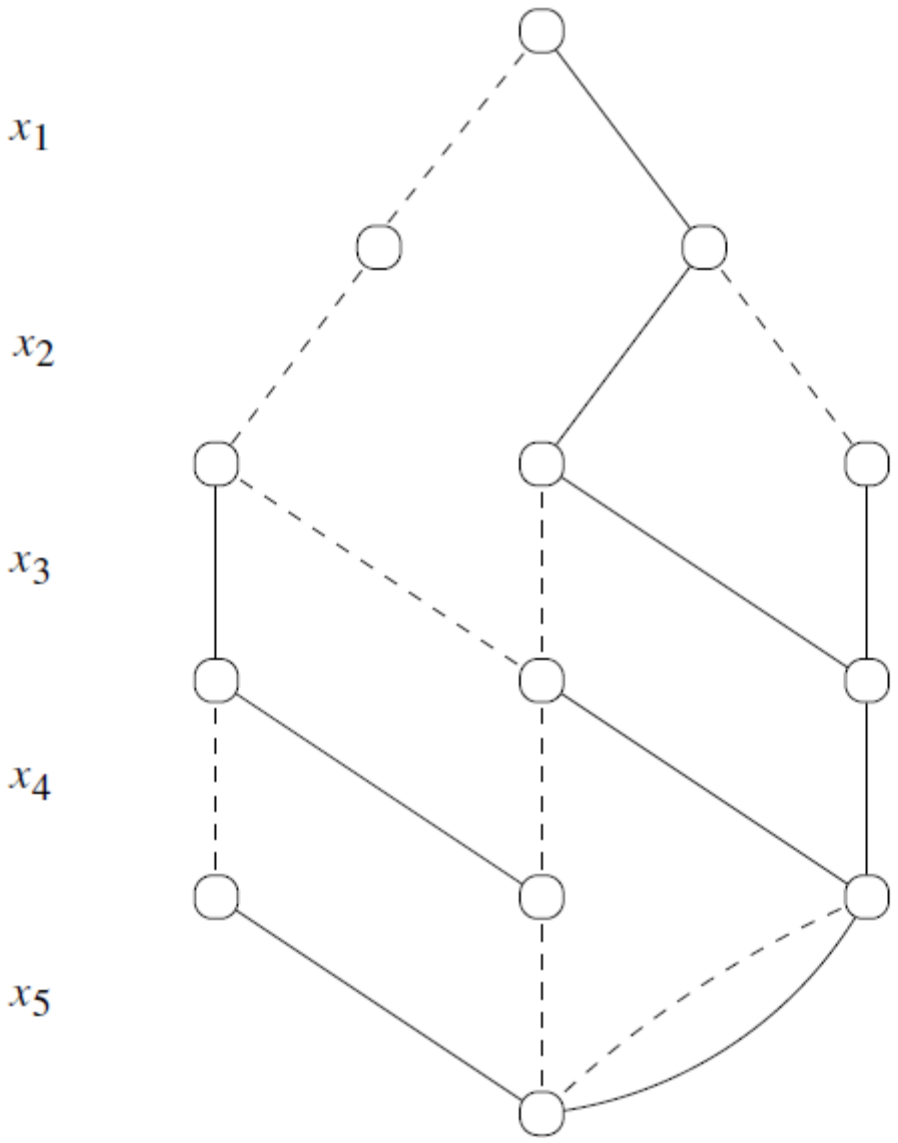
- Then the *Among* constraint on $[x_{i+1},\ldots, x_{i+q}]$ is equivalent to

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

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

- [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

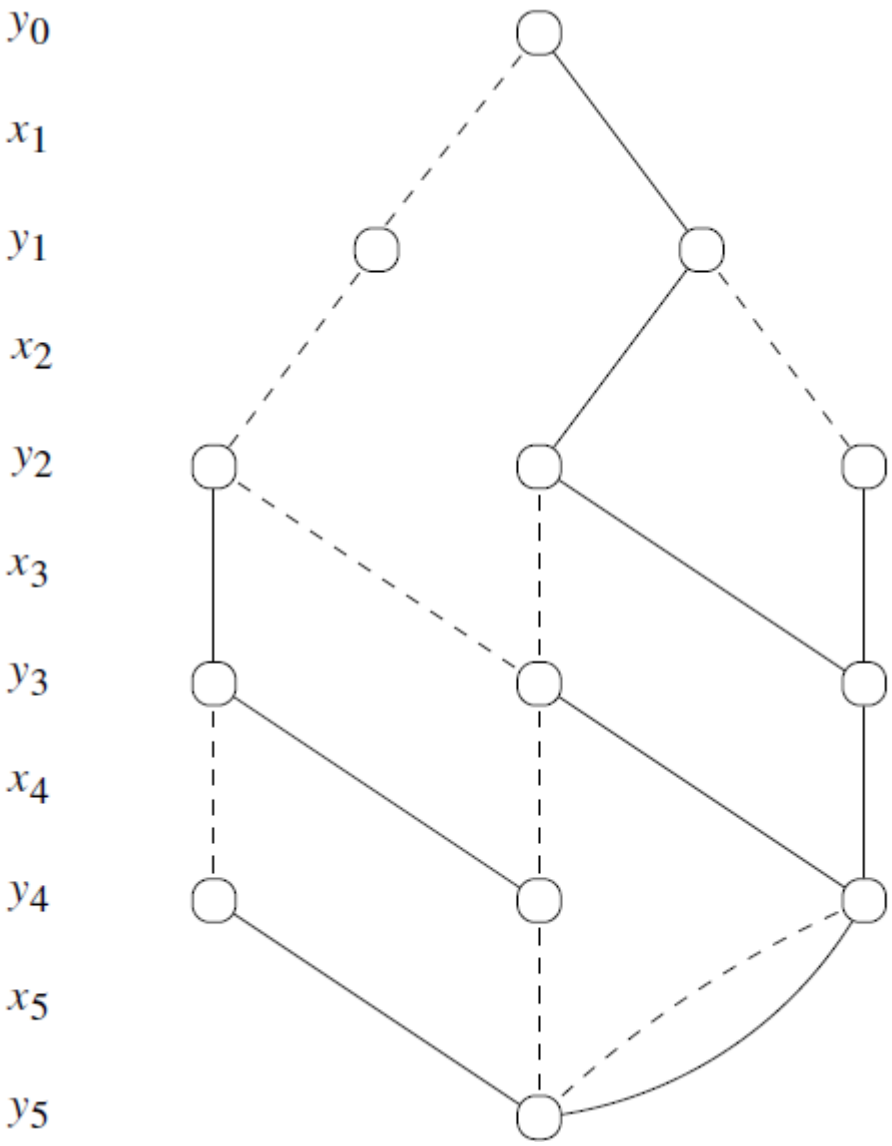


Sequence(X, q=3, S={1}, l=1, u=2)

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)

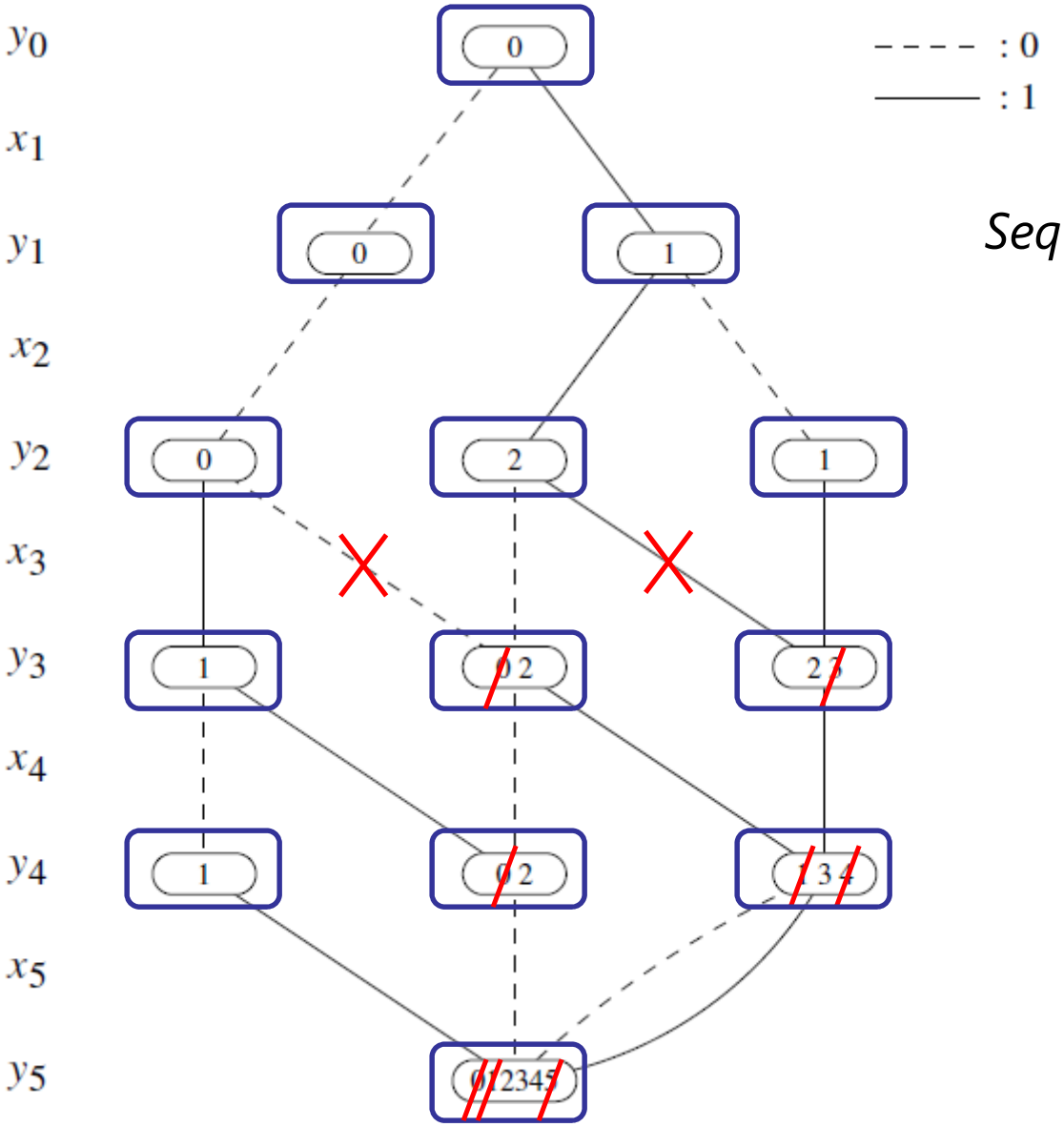
$$y_i = y_{i-1} + x_i$$

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

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

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

MDD filtering from decomposition



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

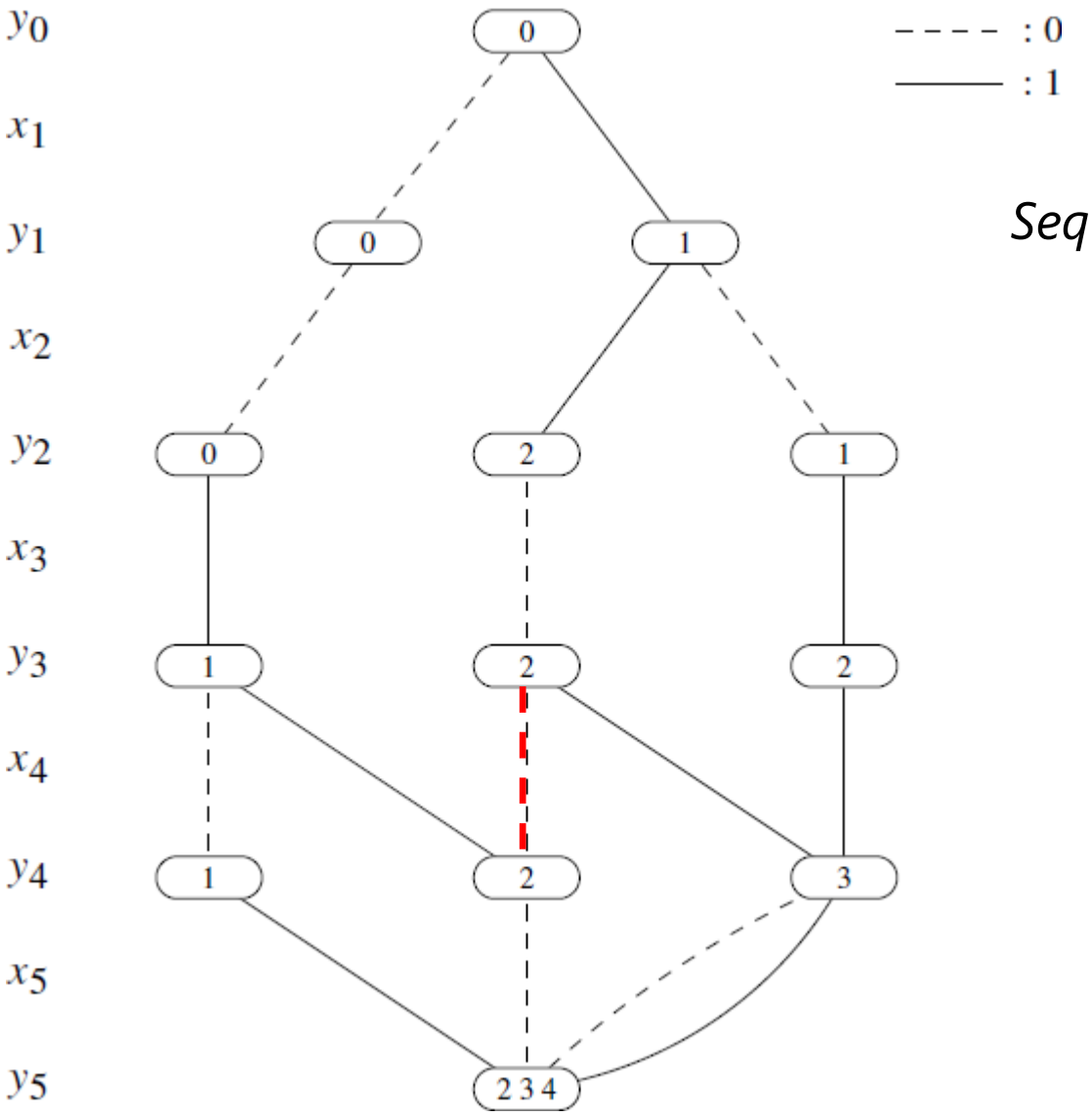
$y_i = y_{i-1} + x_i$

$1 \le y_3 - y_0 \le 2$

$1 \le y_4 - y_1 \le 2$

$1 \le y_5 - y_2 \le 2$

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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

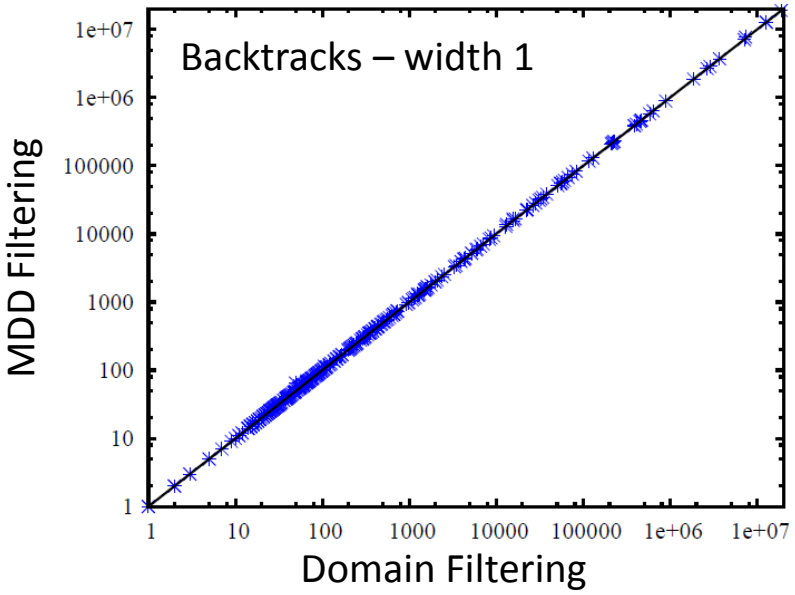
Random instances

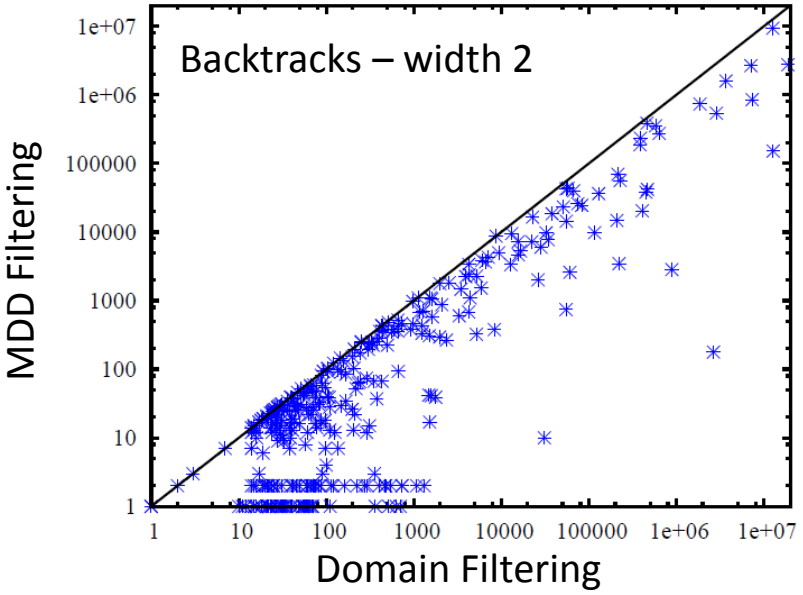
Carnegie Mellon Tepper SCHOOL OF BUSINESS

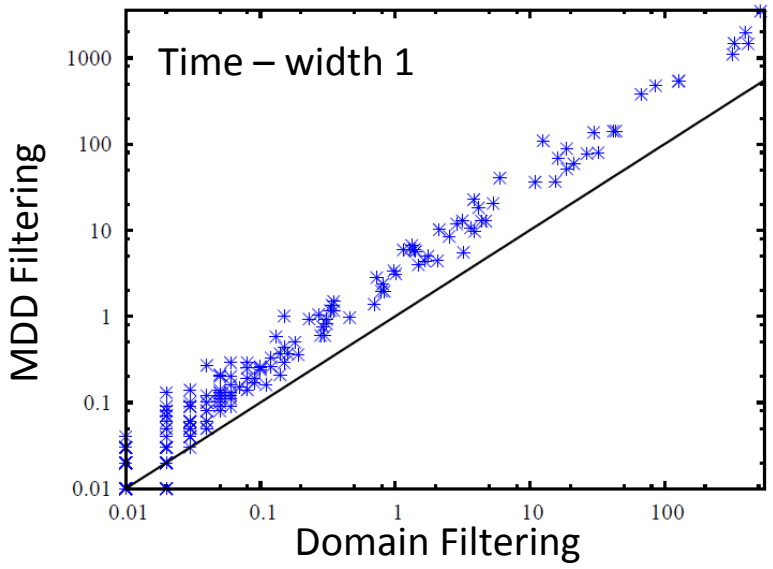
- Randomly generated instances
  - n=20-48 variables
  - domain size between 10 and 30
  - 1, 2, 5, 7, or 10 *Sequence* constraints
  - q random from [2..n/2]
  - $u - l$ random from 0 to q-1
  - 360 instances

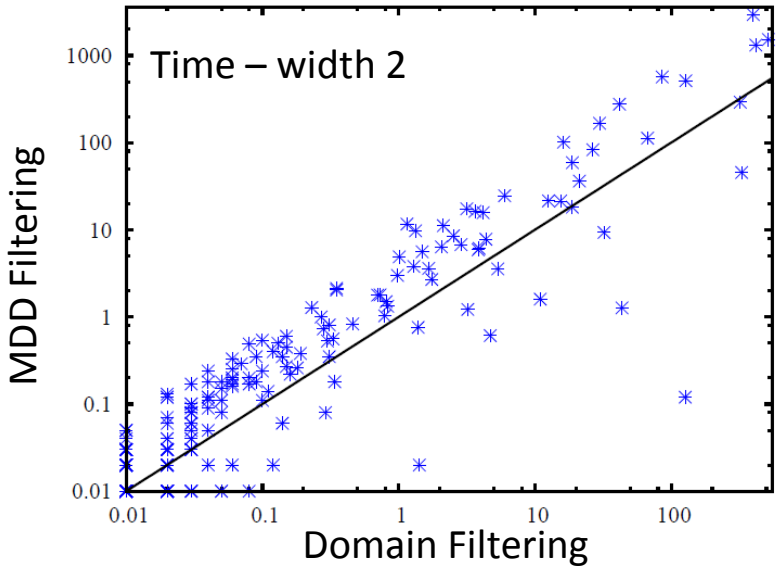
- Vary maximum width of MDD
  - widths 1 up to 32

Random instances results

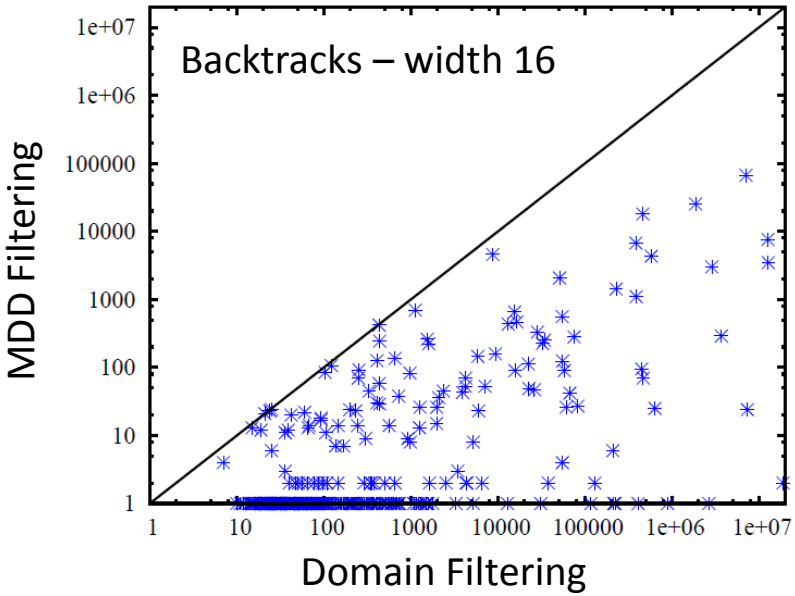


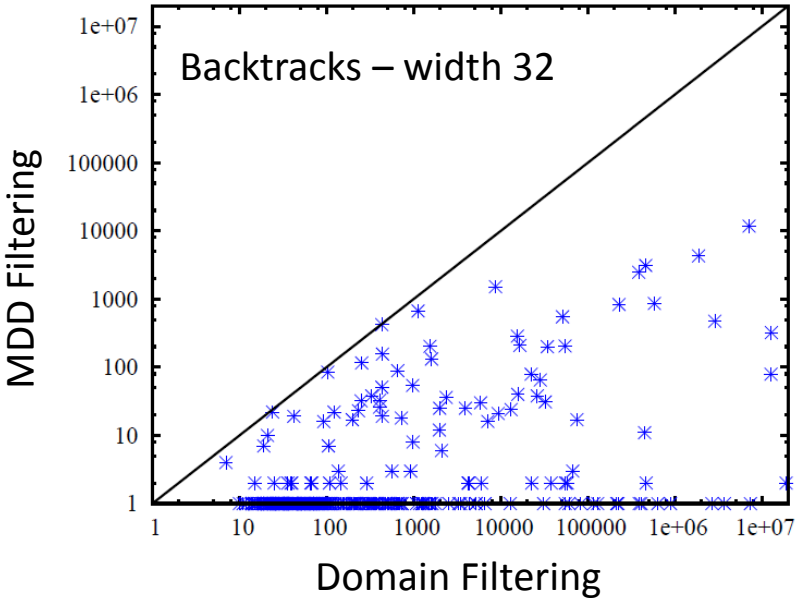


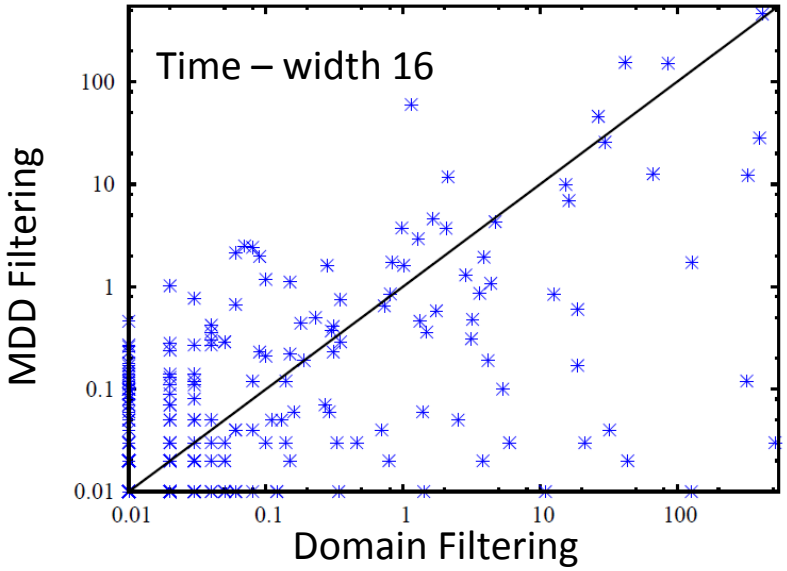


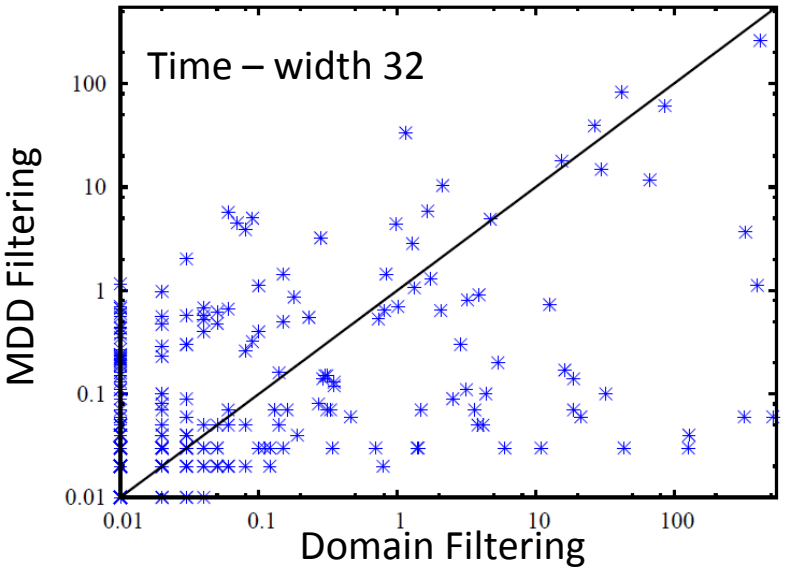


Random instances results (cont'd)









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

Summary for MDD-based CP

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Key idea: Propagate approximate MDDs instead of domains
- Strength of MDD can be controlled by its width
- Constraint-specific propagation algorithms
  - very similar to domain propagators
  - define state information for each constraint
  - central operations: edge filtering and node refinement
- Detailed examples: *Among* and *Sequence*
- Huge reduction in the amount of backtracking and solution time is possible

Exercises

Carnegie Mellon Tepper SCHOOL OF BUSINESS

1. Consider the constraint $x \neq y$ for two finite-domain variables x and y. Assume that x and y belong to a set X of variables for which we are given a relaxed MDD. Design an MDD propagator for $x \neq y$.

2. Consider the following CSP:

   $x_1 \in \{0,1\}$, $x_2 \in \{0,1,2\}$, $x_3 \in \{1,2\}$

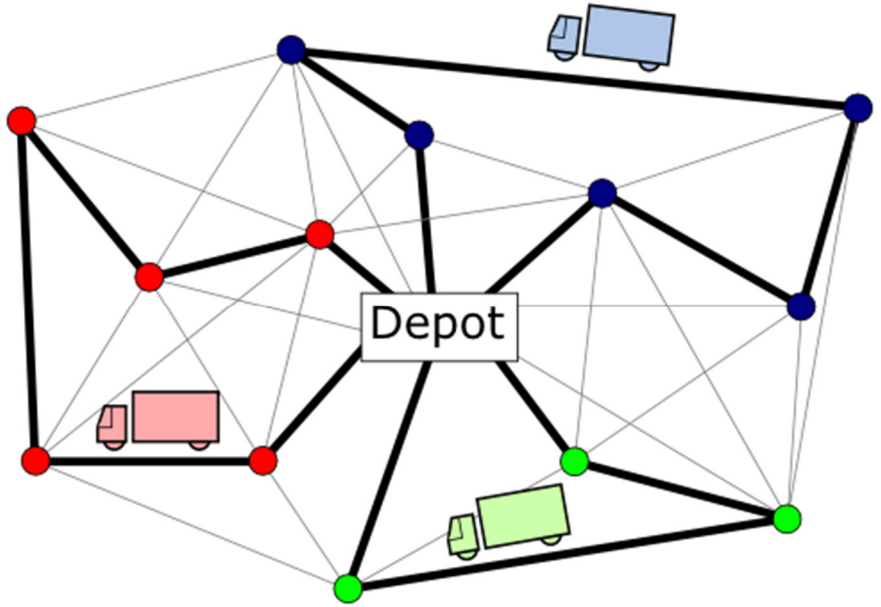
   $x_1 \neq x_2$, $x_2 \neq x_3$, $x_1 \neq x_3$

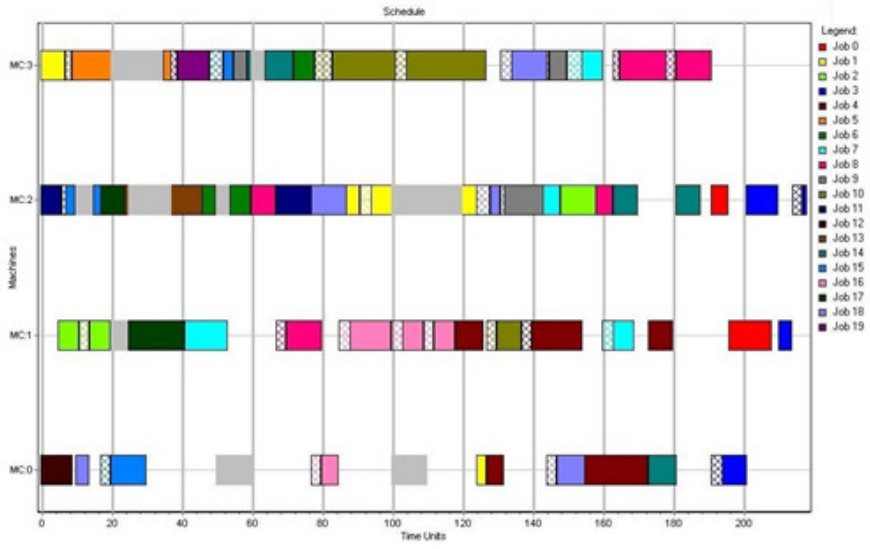
   Apply the propagators from Exercise 2, starting from a width-1 MDD, until the MDD represents all solutions to the CSP.

MDDs for Disjunctive Scheduling

Disjunctive Scheduling

Carnegie Mellon Tepper SCHOOL OF BUSINESS





Constraint-Based Scheduling

Carnegie Mellon Tepper SCHOOL OF BUSINESS

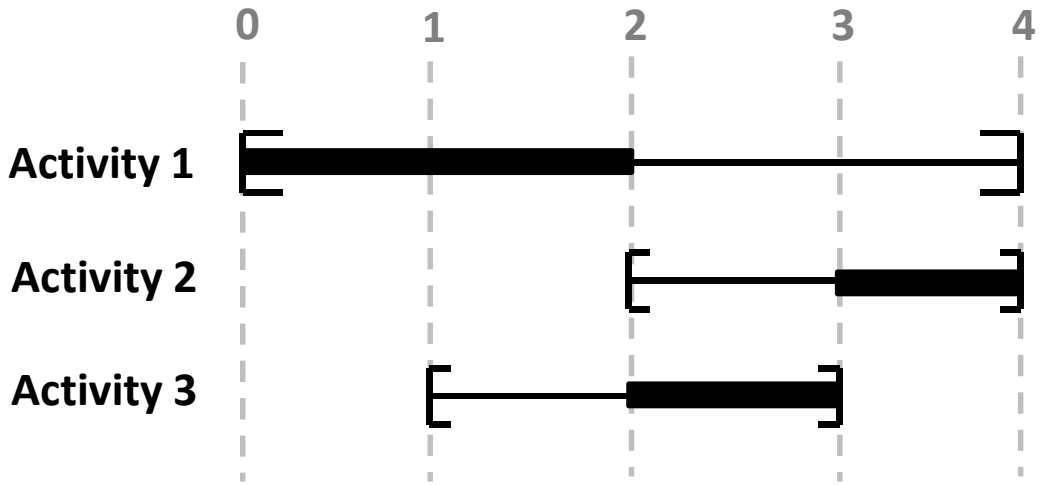
- Disjunctive scheduling may be viewed as the 'killer application' for CP
  - Natural modeling (activities and resources)
  - Allows many side constraints (precedence relations, time windows, setup times, etc.)
  - State of the art while being generic methodology
- However, CP has some problems when
  - 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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

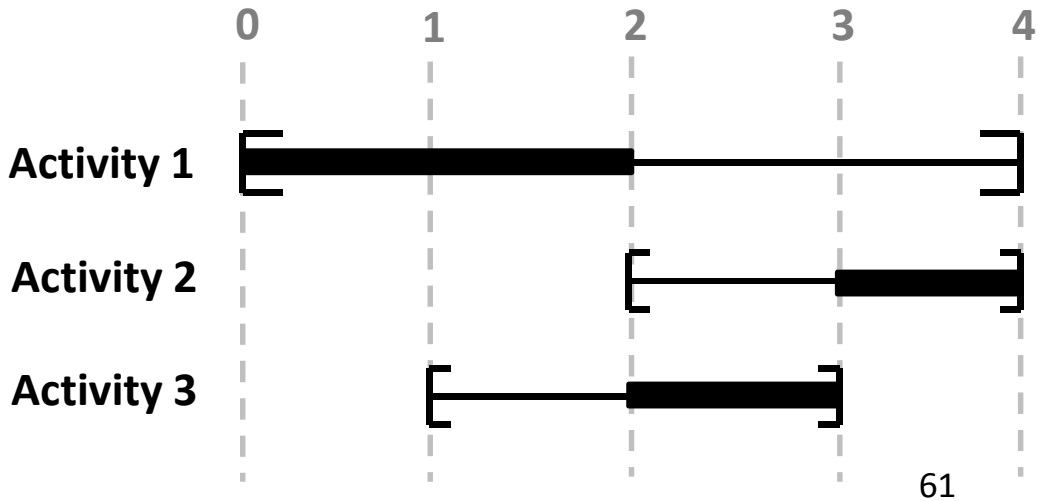
- Precedence relations between activities
- Sequence-dependent setup times
- Induced by objective function
  - Makespan
  - Sum of setup times
  - Sum of completion times
  - Tardiness / number of late jobs
  - …

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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Three main considerations: Cire & v.H. [2012, 2013]

- 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

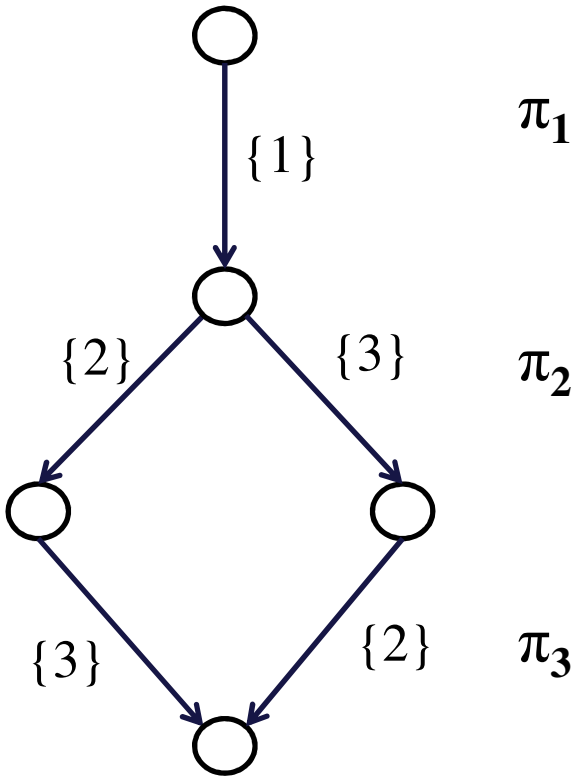
- Natural representation as 'permutation MDD'
- Every solution can be written as a permutation $\boldsymbol{\pi}$

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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

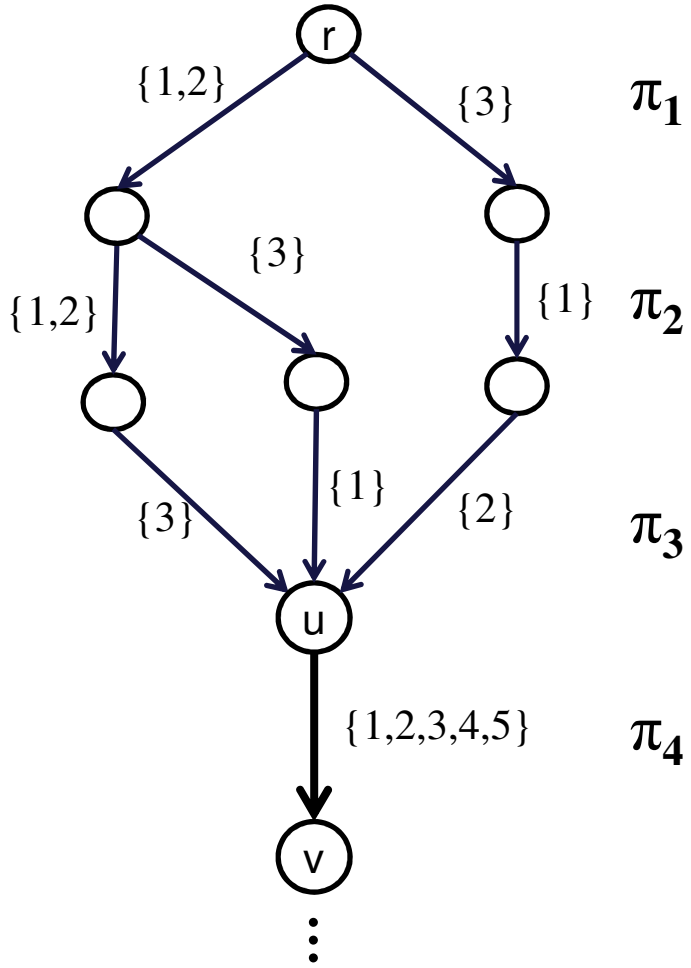
Carnegie Mellon Tepper SCHOOL OF BUSINESS

We can apply several propagation algorithms:

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

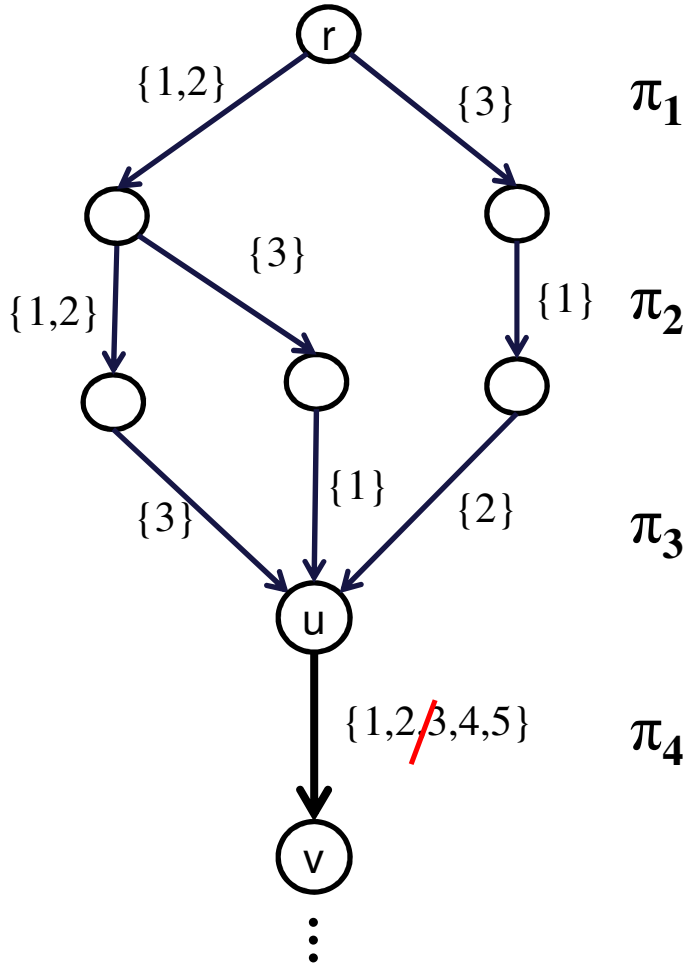
Propagation (cont'd)

- 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

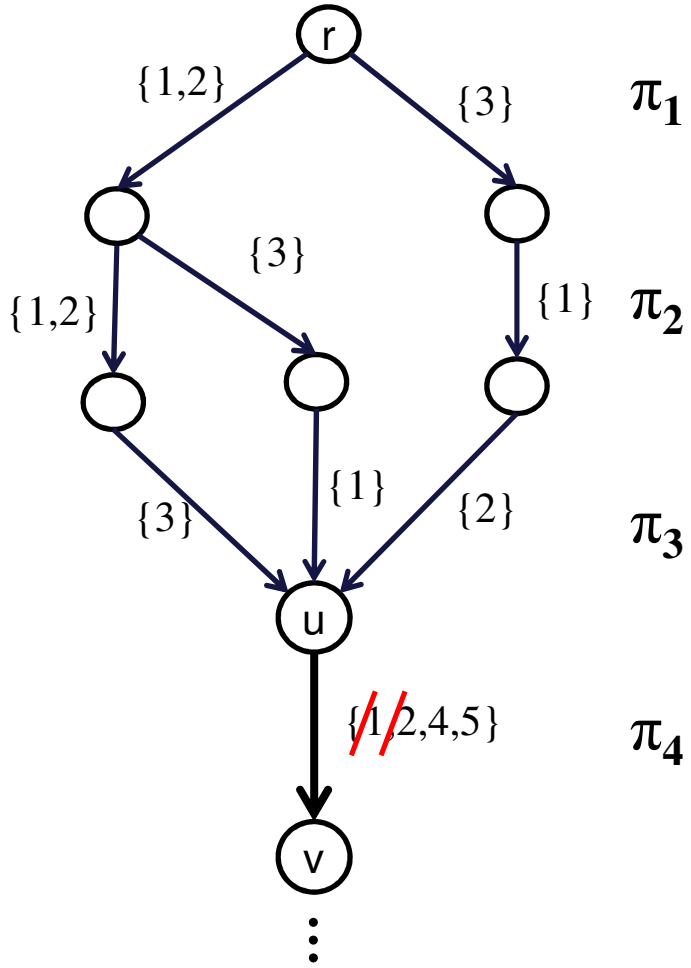
- All-paths state: A_u
  - Labels belonging to all paths from node r to node u
  - $A_u = \{3\}$
  - Thus eliminate {3} from (u,v)



[Andersen et al., 2007]

Alldifferent Propagation

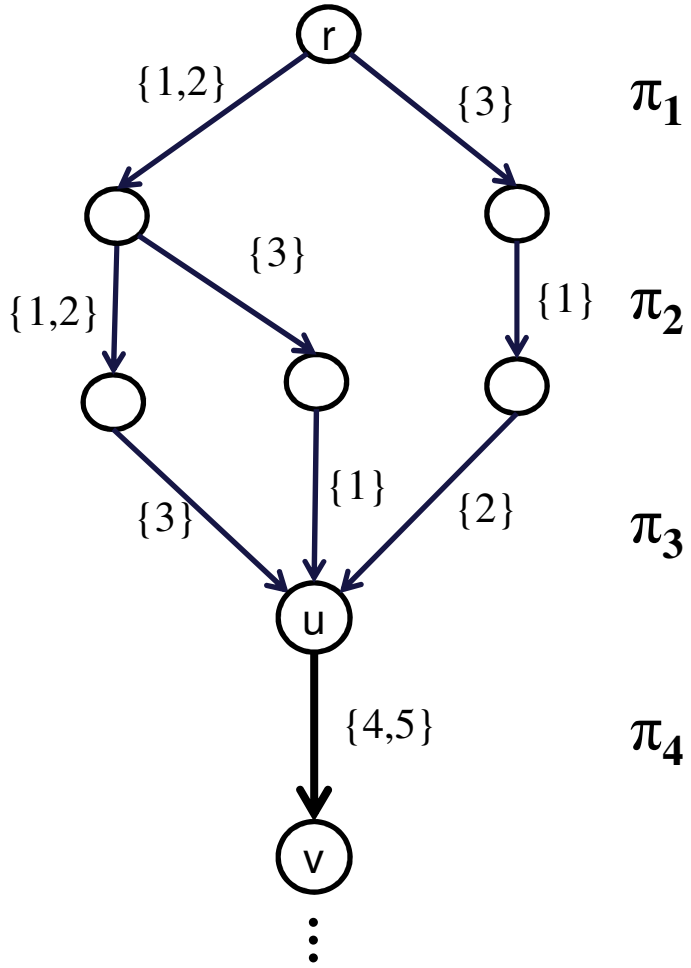
- Some-paths state: S_u
  - Labels belonging to some path from node r to node u
  - $S_u = \{1,2,3\}$
  - Identification of Hall sets
  - Thus eliminate {1,2,3} from (u,v)



Propagate Earliest Completion Time

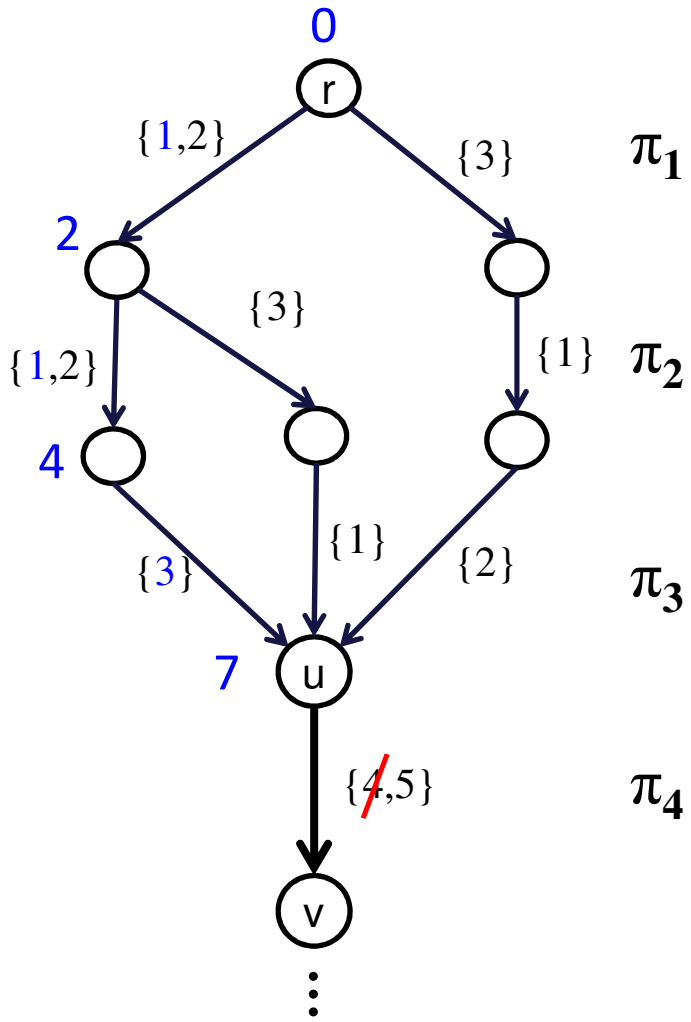
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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)

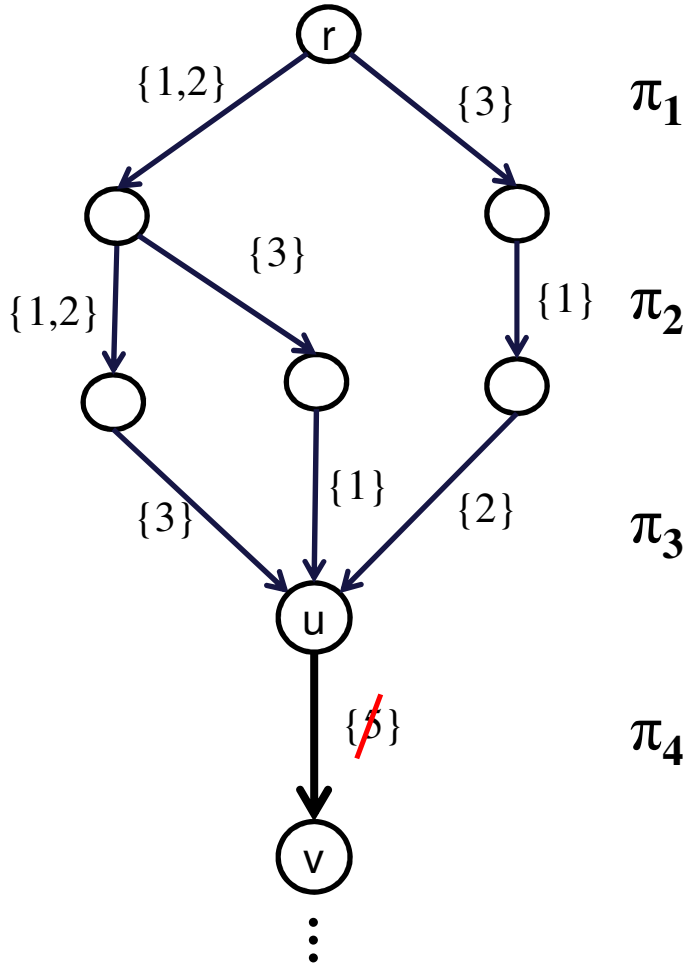
Propagate Precedence Relations

- 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)

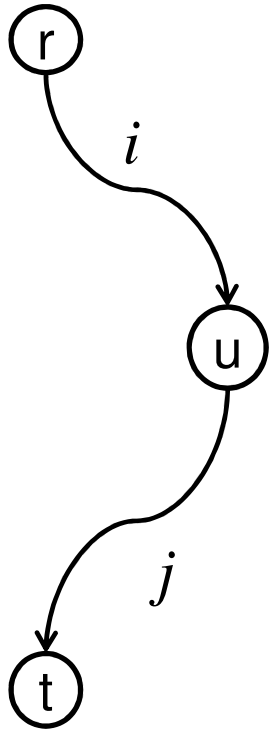
- Similarly: Bottom-up for $j \ll i$



More MDD Inference

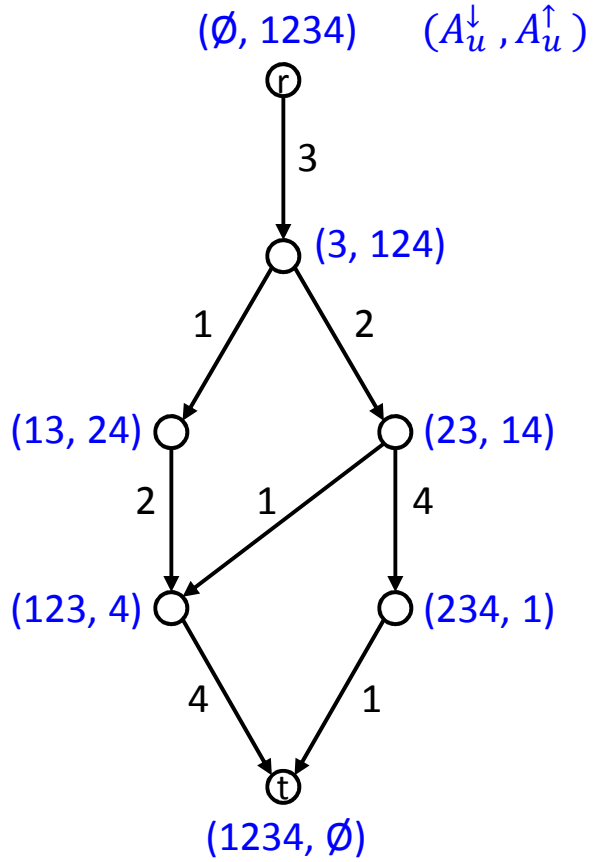
Theorem: *Given the exact MDD M, we can deduce all implied activity precedences in polynomial time in the size of M*

- 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

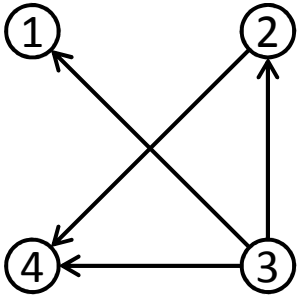


Extracting precedence relations

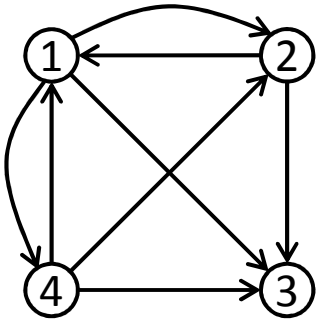
- Build a digraph G=(V, E) where V is the set of activities
- For each node u in M
  - if $j \in A_u^{\downarrow}$ and $i \in A_u^{\uparrow}$ add edge (*i*,*j*) to E
  - represents that $i \ll j$ cannot hold
- Take complement graph $\overline{G}$
  - complement edge exists iff $i \ll j$ holds



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



$\overline{G}$



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
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 is based on *alldifferent*
  - Order activities by some criterion (e.g., decreasing r_i+p_i)
  - For given layer, expand all nodes into next layer
  - Separate nodes that are exact relative to as many of the ordered activities, i.e., $A_u = S_u$
  - Nodes beyond maximum width are merged

See [Cire, v.H., 2013] for more details

Experiments

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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
  - MDD is added as user-defined propagator

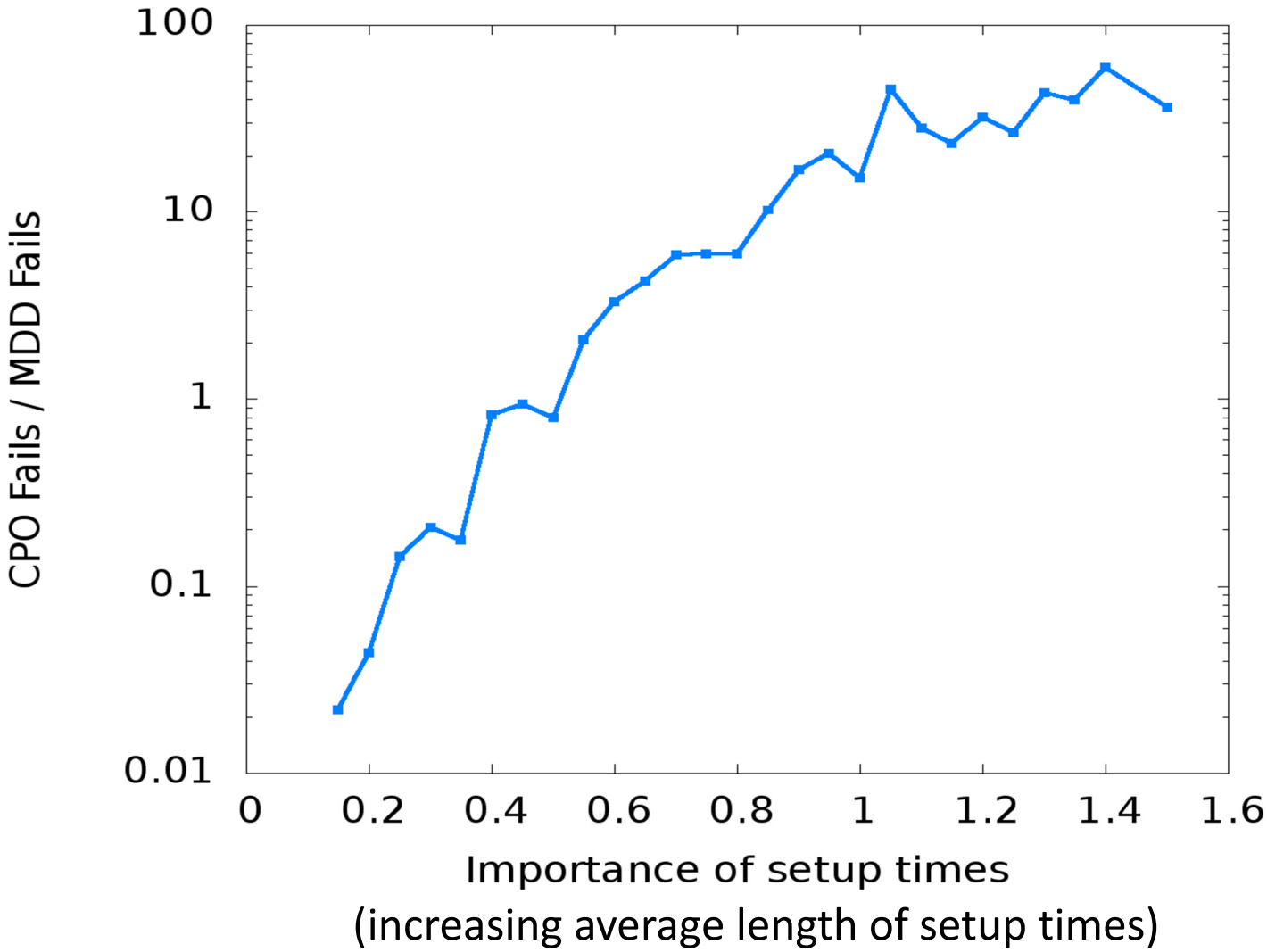
- Main purpose of experiments
  - where can MDDs bring strength to CP
  - compare stand-alone MDD versus CP
  - compare CP versus CP+MDD (most useful in practice)

Problem classes

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

Importance of setup times



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

Minimize Makespan

Carnegie Mellon Tepper SCHOOL OF BUSINESS

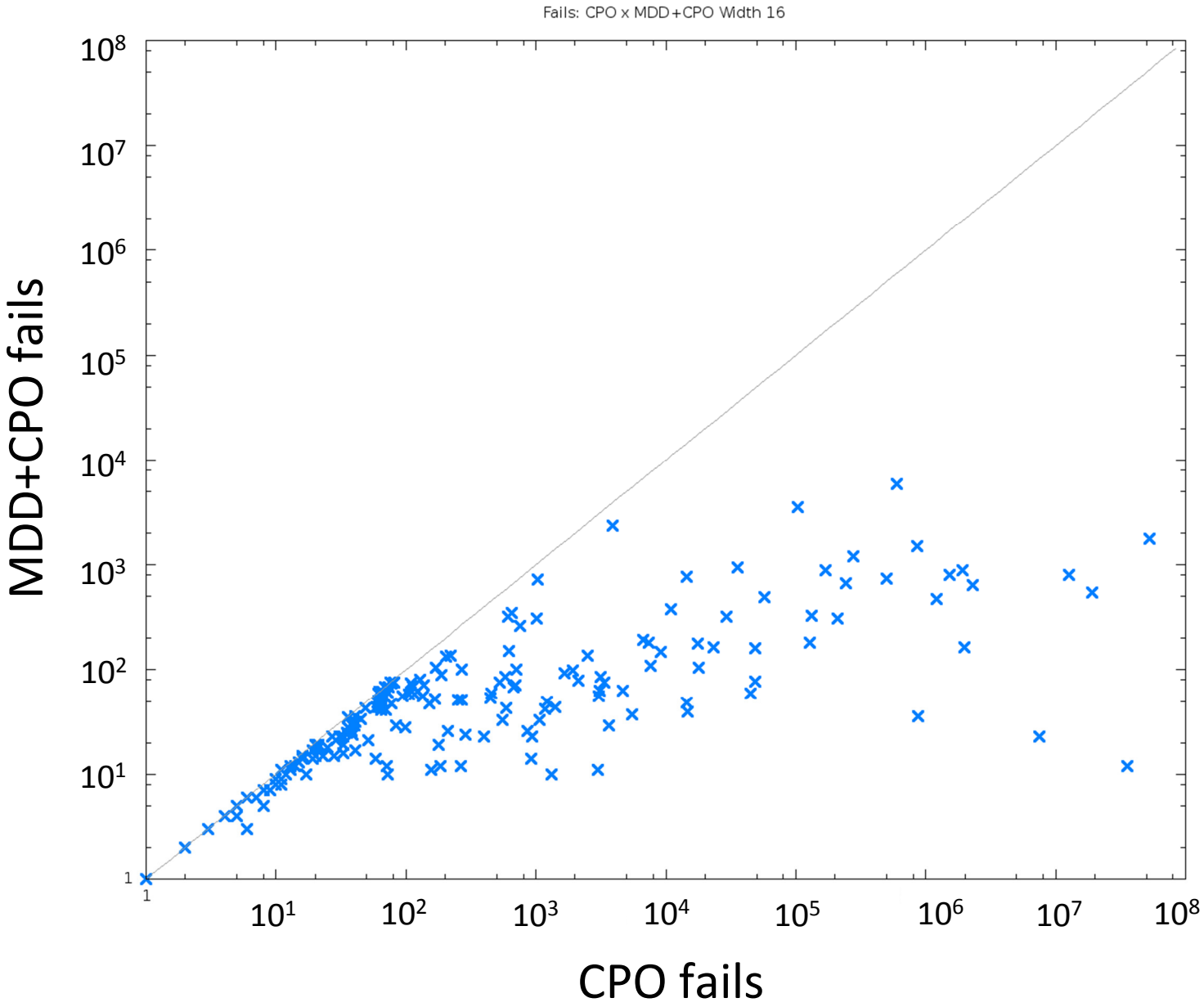
- 229 TSPTW benchmark instances with up to 100 jobs
- Minimize makespan
- Time limit 7,200s
- Max MDD width is 16

instances solved by CP: 211

instances solved by pure MDD: 216

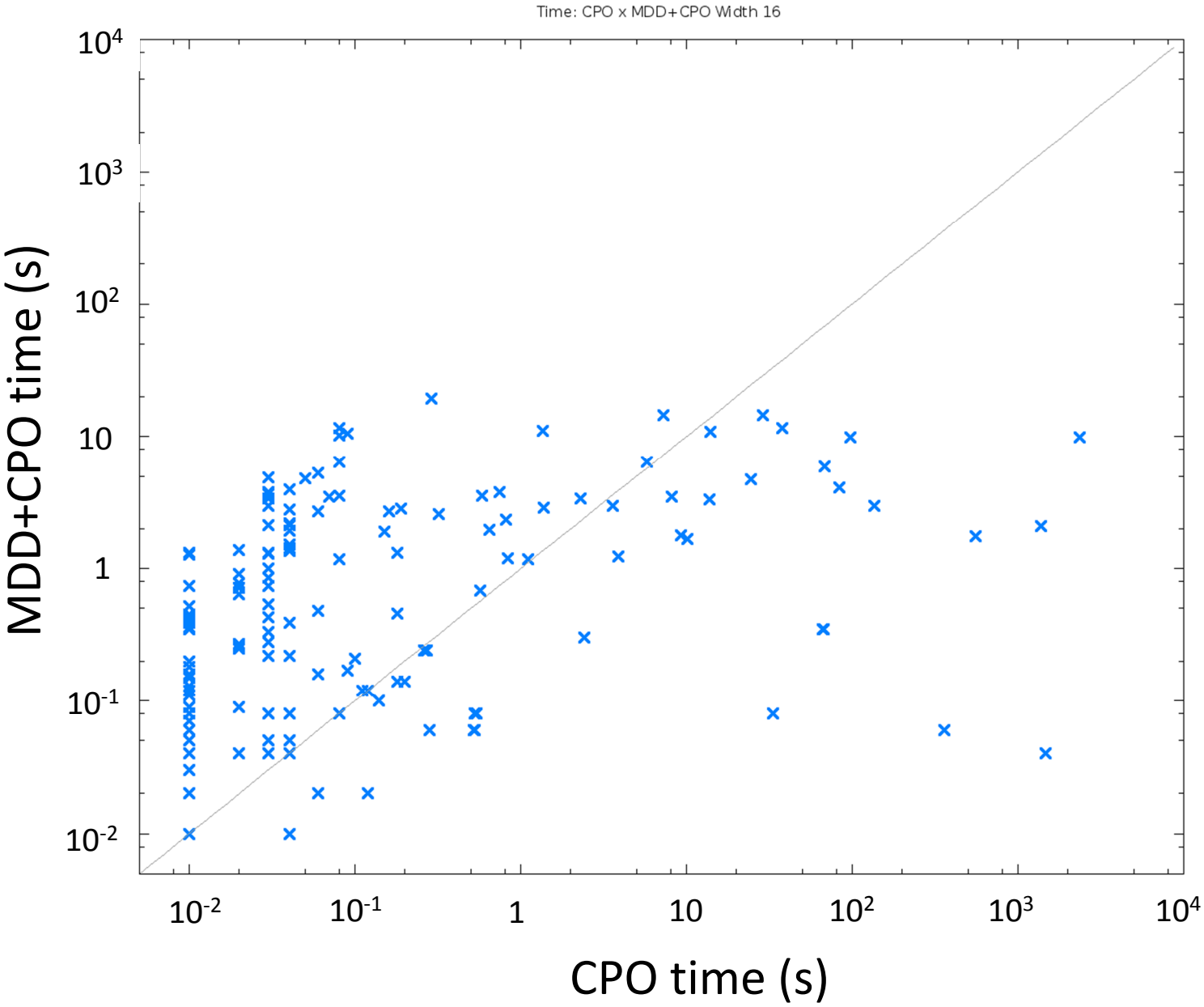
instances solved by CP+MDD: 227

Minimize Makespan: Search tree size

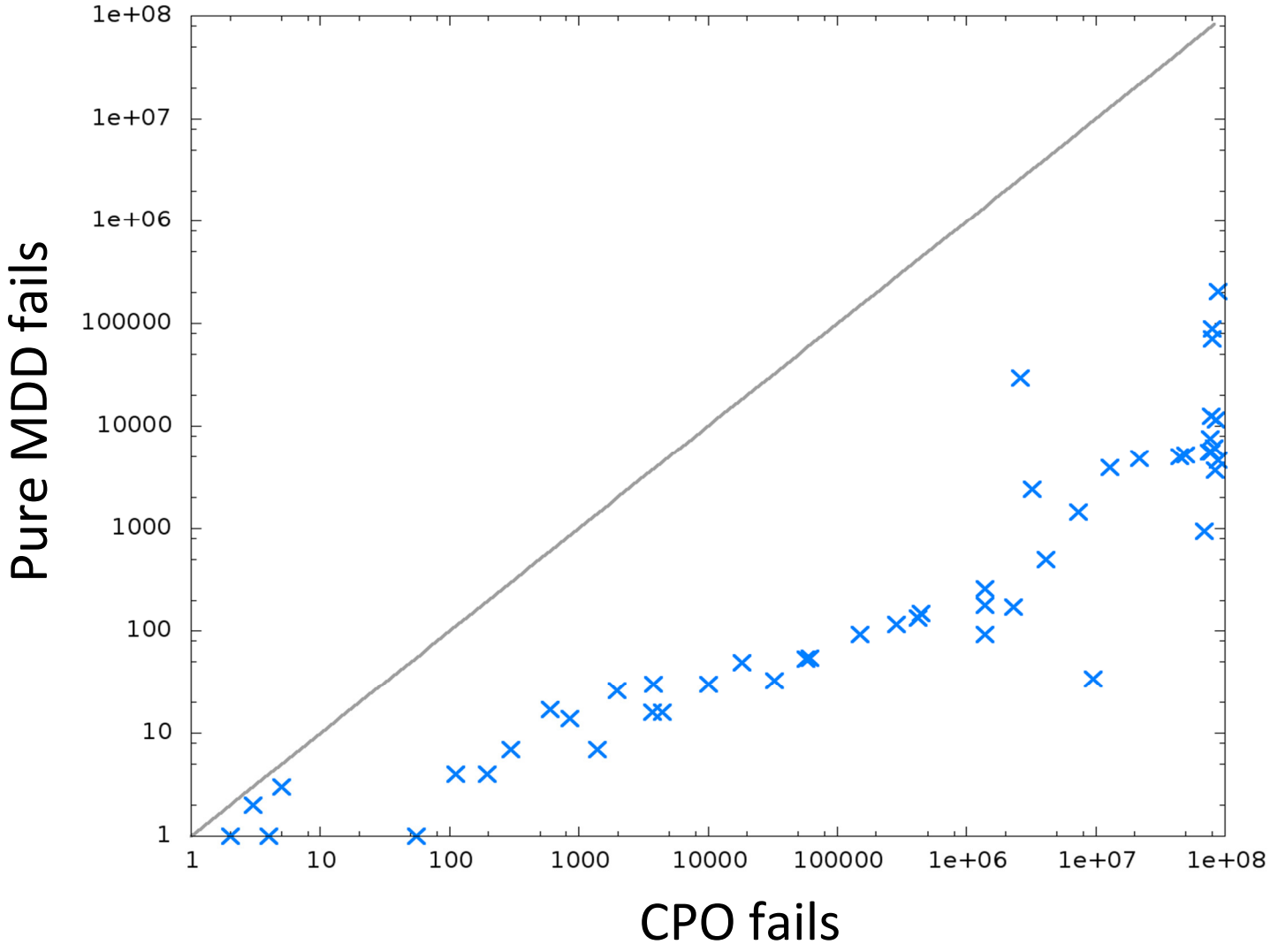


plot only shows instances that were solved by all methods

Minimize Makespan: Time

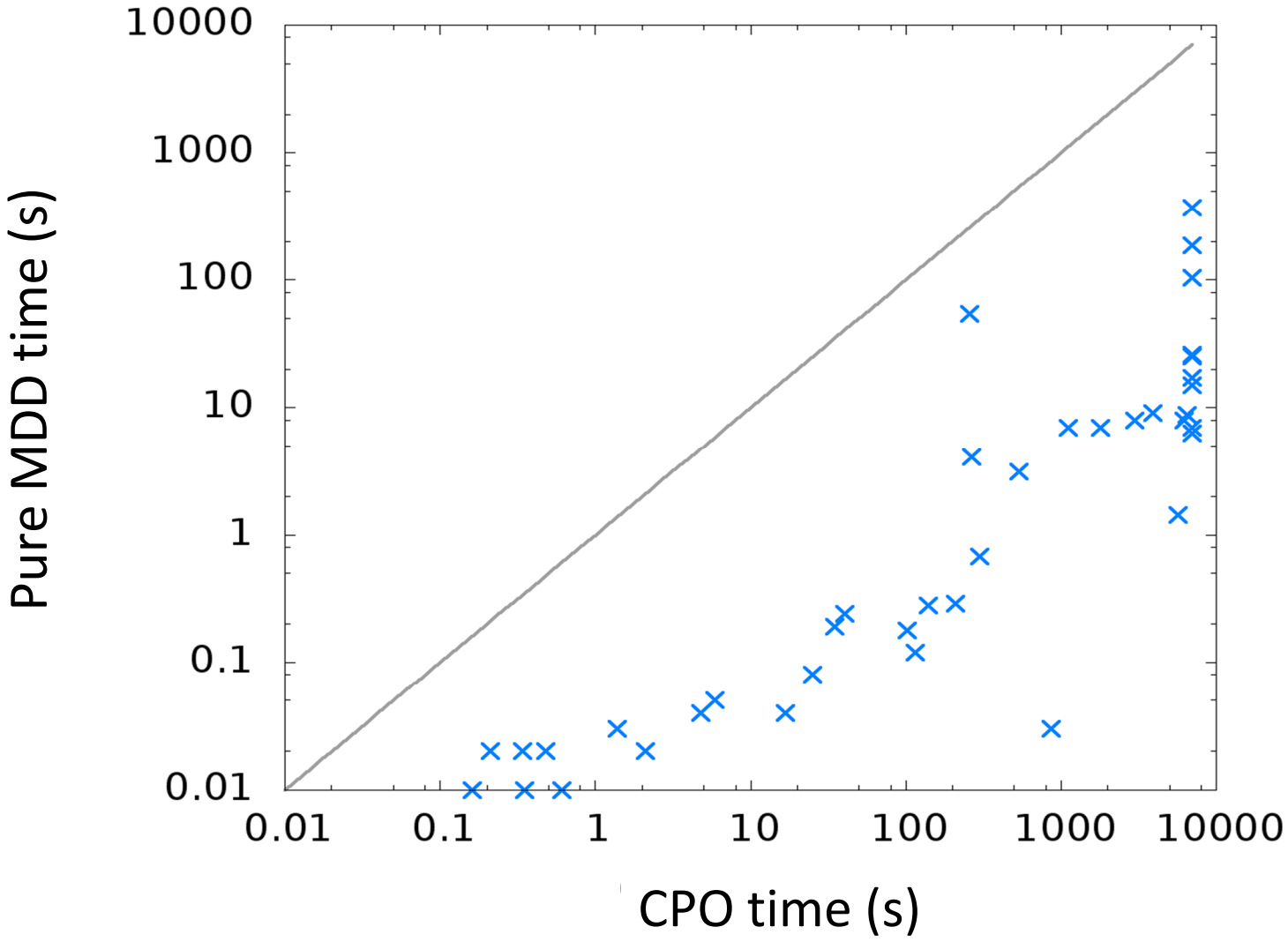


Min sum of setup times: Fails



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)

Carnegie Mellon Tepper SCHOOL OF BUSINESS

		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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- ATSP with precedence constraints (no time windows)

- Instances up to 53 jobs
- Time limit 1,800s
- Default CPO search
- Max MDD width 2,048

Sequential Ordering Problem Results

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

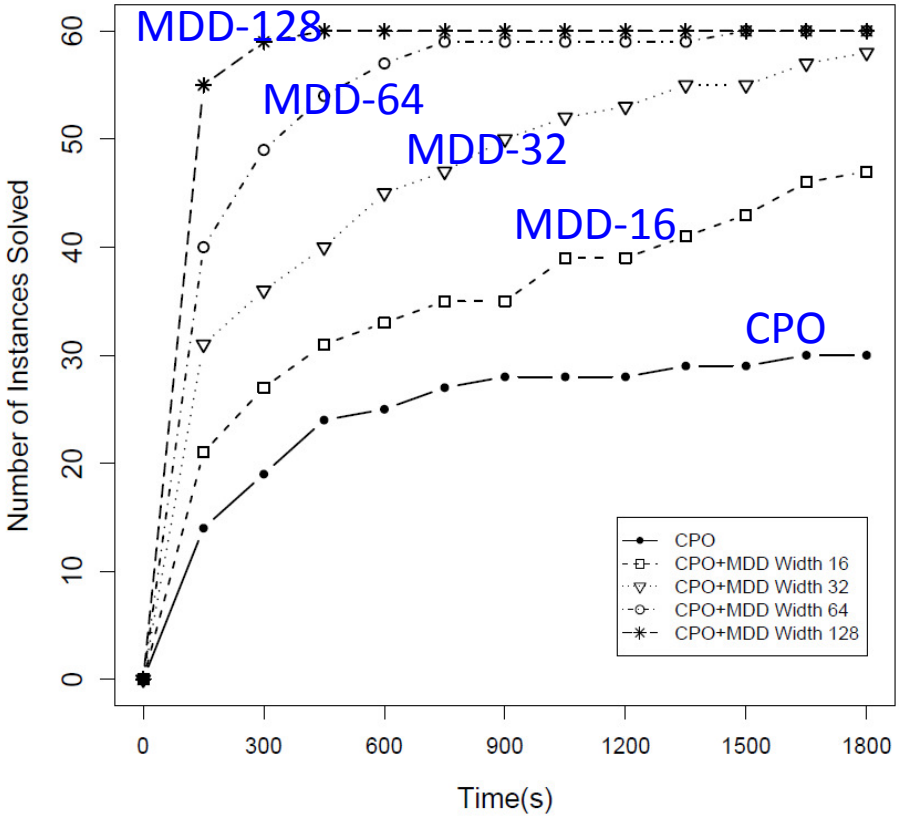
Minimize Total Tardiness

Carnegie Mellon Tepper SCHOOL OF BUSINESS

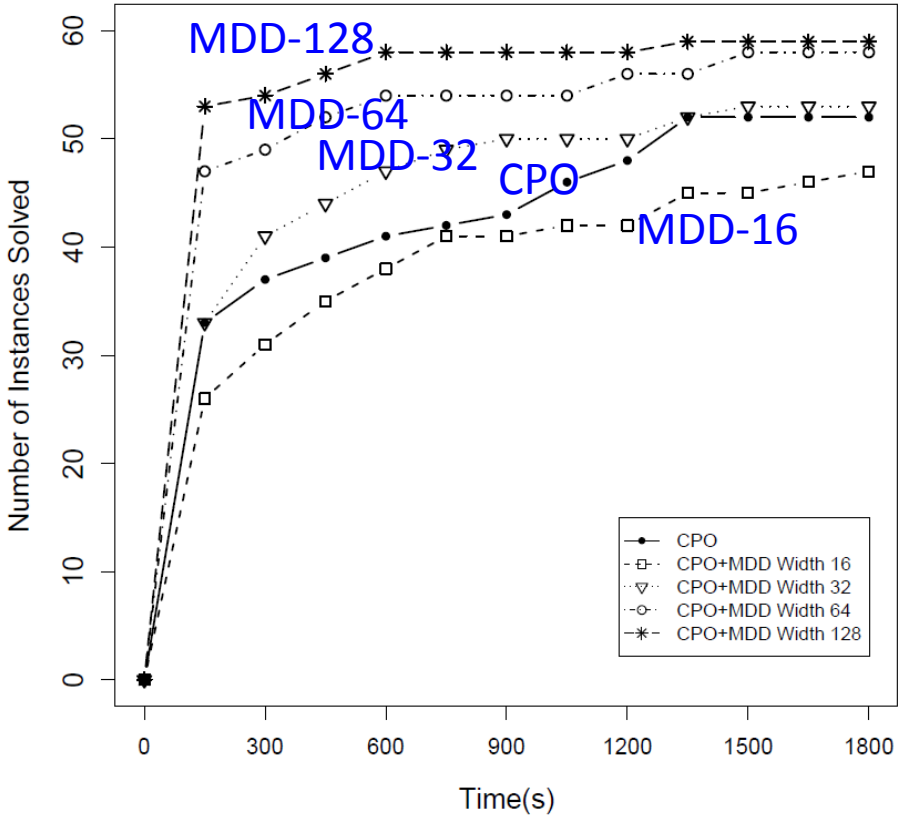
- Consider activity i with due date δ_i
  - Completion time of i: $c_i = s_i + p_i$
  - Tardiness of i: $\max\{0, c_i - \delta_i\}$
- Objective: minimize total (weighted) tardiness

- 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

Summary for Disjunctive Scheduling

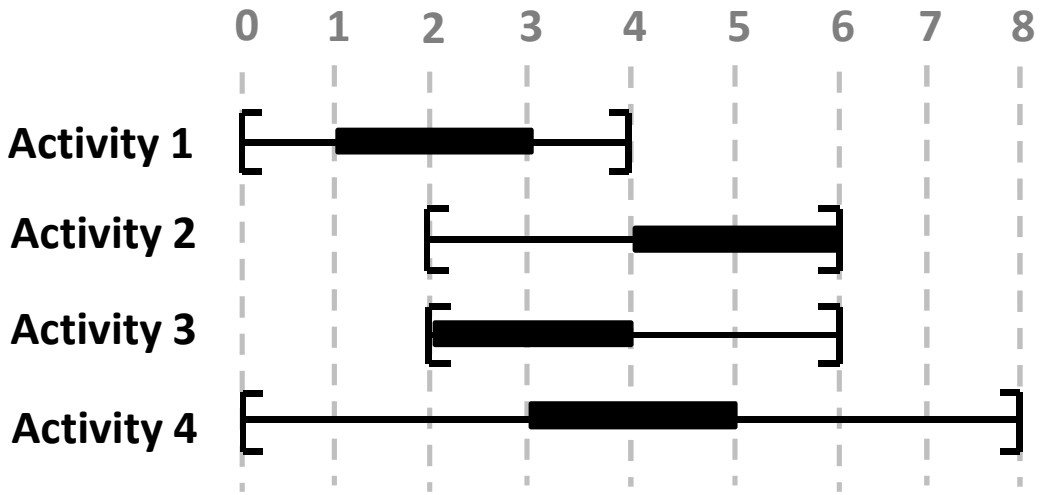
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Application of MDD-based CP to generic disjunctive scheduling
- MDD propagation for
  - *Alldifferent* constraint (permutation of activities)
  - Precedence constraints
  - Earliest start time/Latest end time
  - Various objective functions (makespan, setup times, tardiness, …)
- Communication of precedence constraints between MDD and domain propagators
- Orders of magnitude improvement

Exercises

3. 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

4. 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 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.

MDDs for Discrete Optimization

Motivation

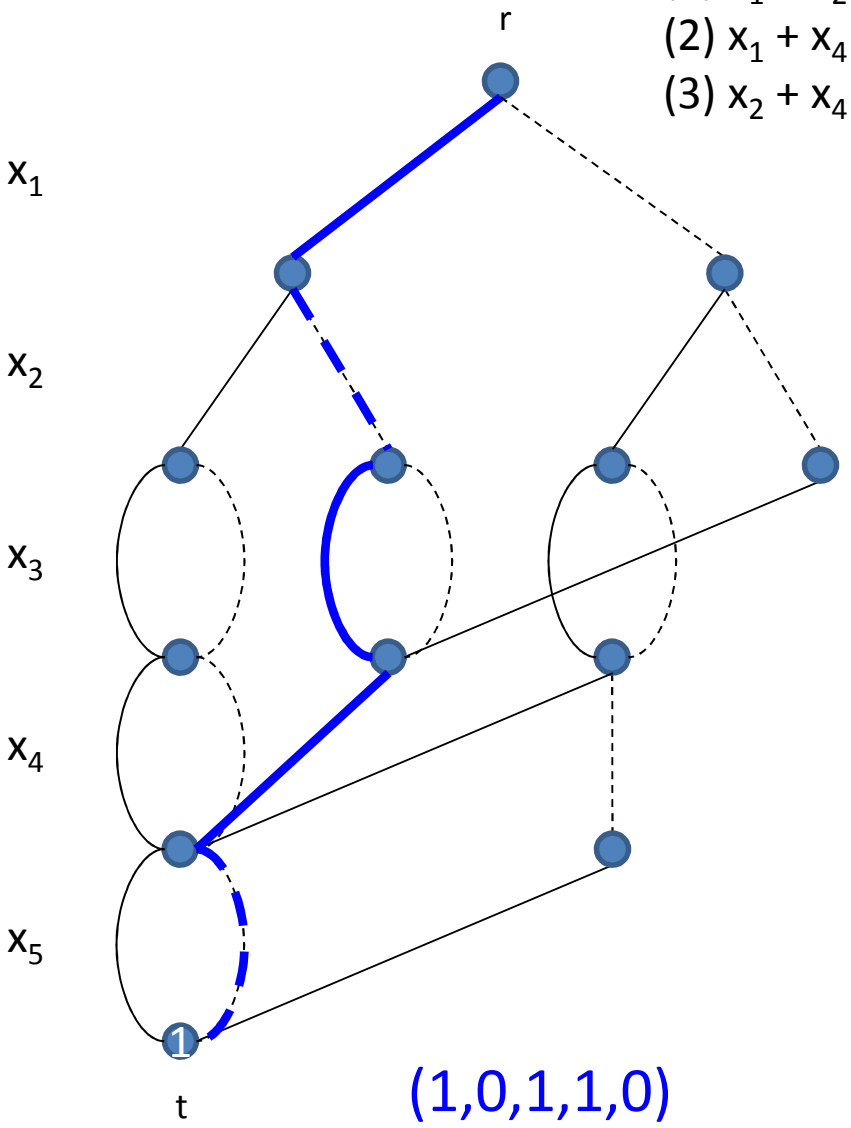
Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

- Relaxation: [Bergman et al., CPAIOR 2011; IJOC to appear]
- Restriction (heuristic solutions): [Bergman et al., 2013]

Handling objective functions

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



(1,0,1,1,0)

Suppose we have an objective function of the form

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

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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

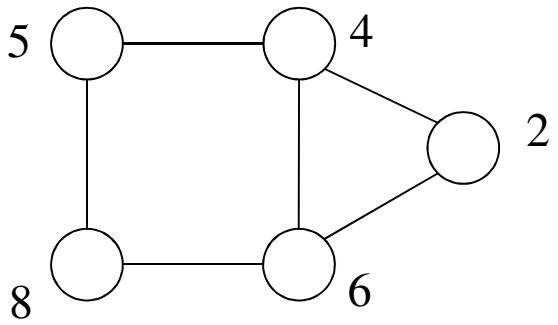
* Feasibility can be checked using MDD-based CP

Case Study: Independent Set Problem

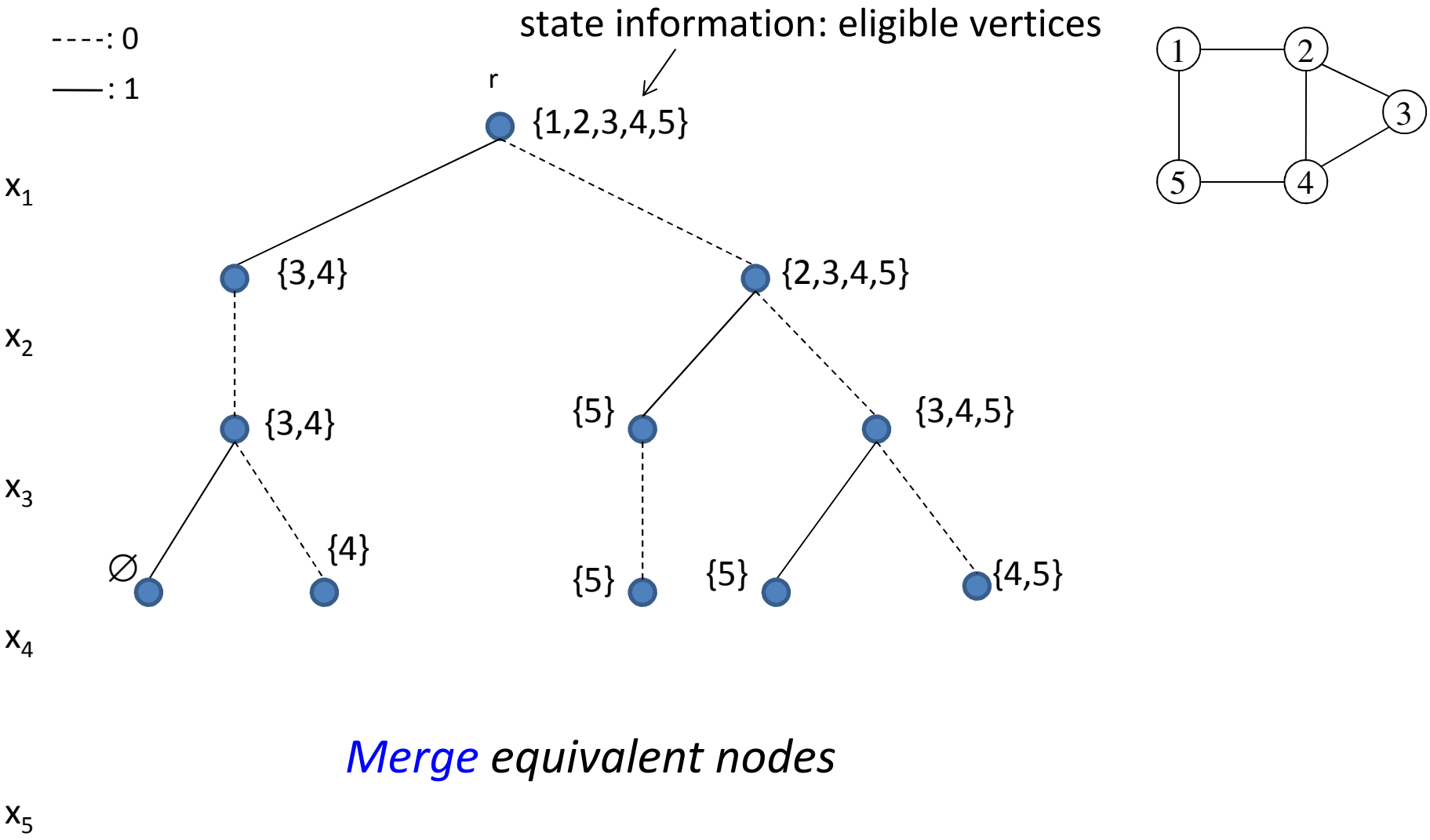
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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

$$\begin{aligned} \max \quad & \sum_i w_i x_i \\ \text{s.t.} \quad & x_i + x_j \le 1 \qquad \text{for all } (i,j) \text{ in } E \\ & x_i \text{ binary} \qquad \text{for all } i \text{ in } V \end{aligned}$$

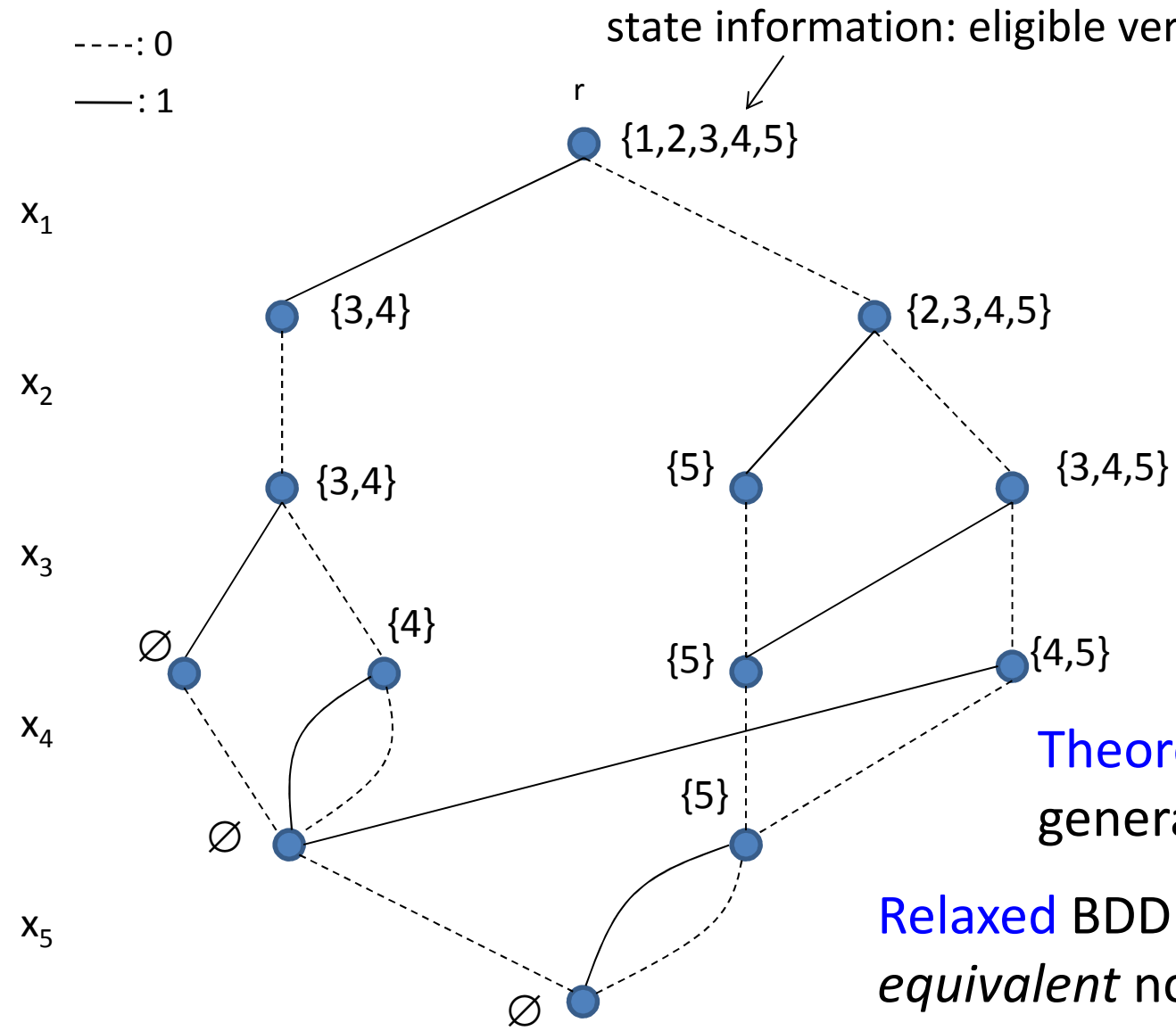


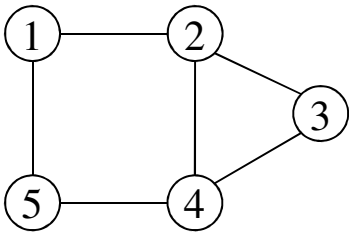
Exact top-down compilation



Merge equivalent nodes

Node Merging





[Bergman et al., 2012]

Theorem: This procedure generates an exact BDD

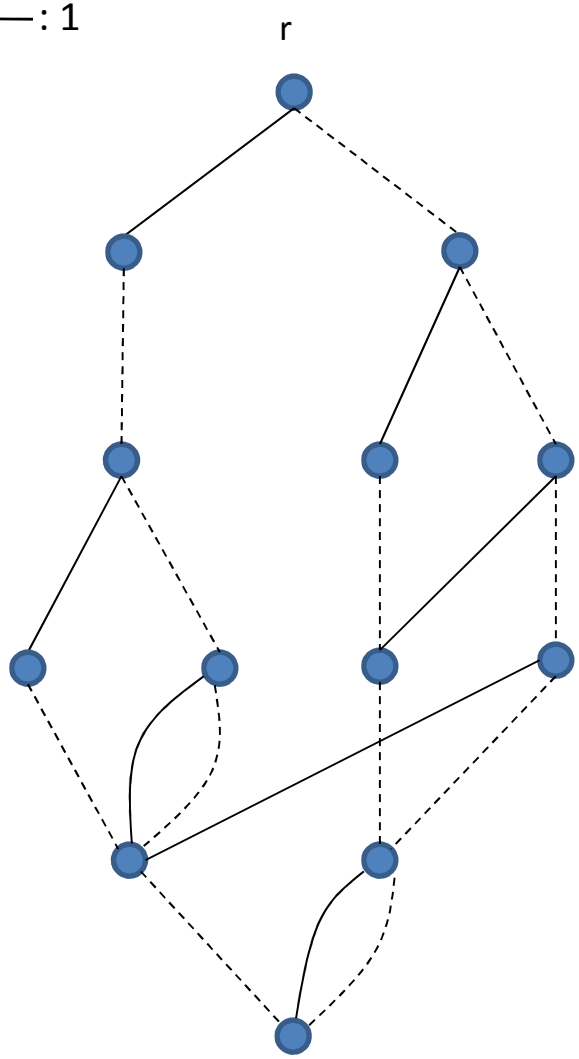
Relaxed BDD: merge *non-equivalent* nodes when the given width is exceeded

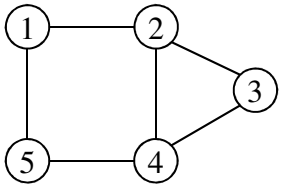
Relaxed BDD

----: 0

——: 1

Exact BDD





x_1

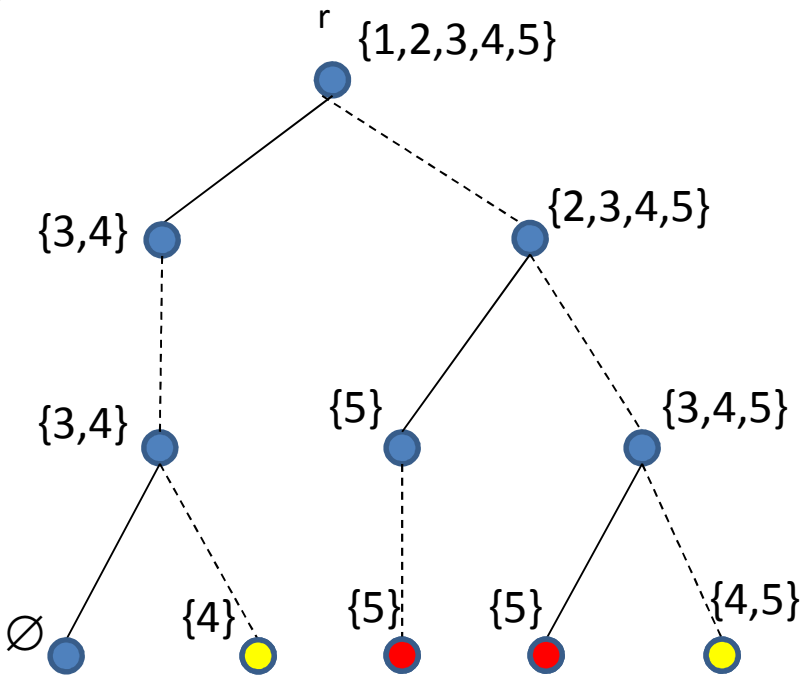
x_2

x_3

x_4

x_5

Relaxed BDD (width ≤ 3)

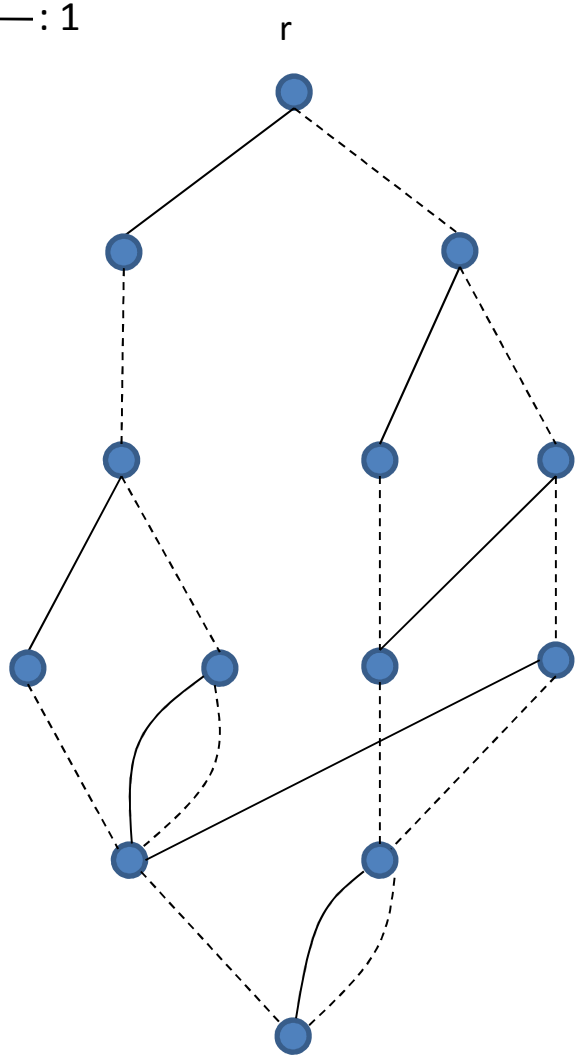


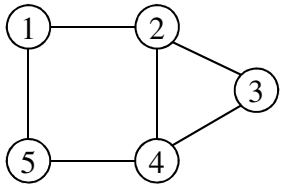
Relaxed BDD

----: 0

——: 1

Exact BDD





x_1

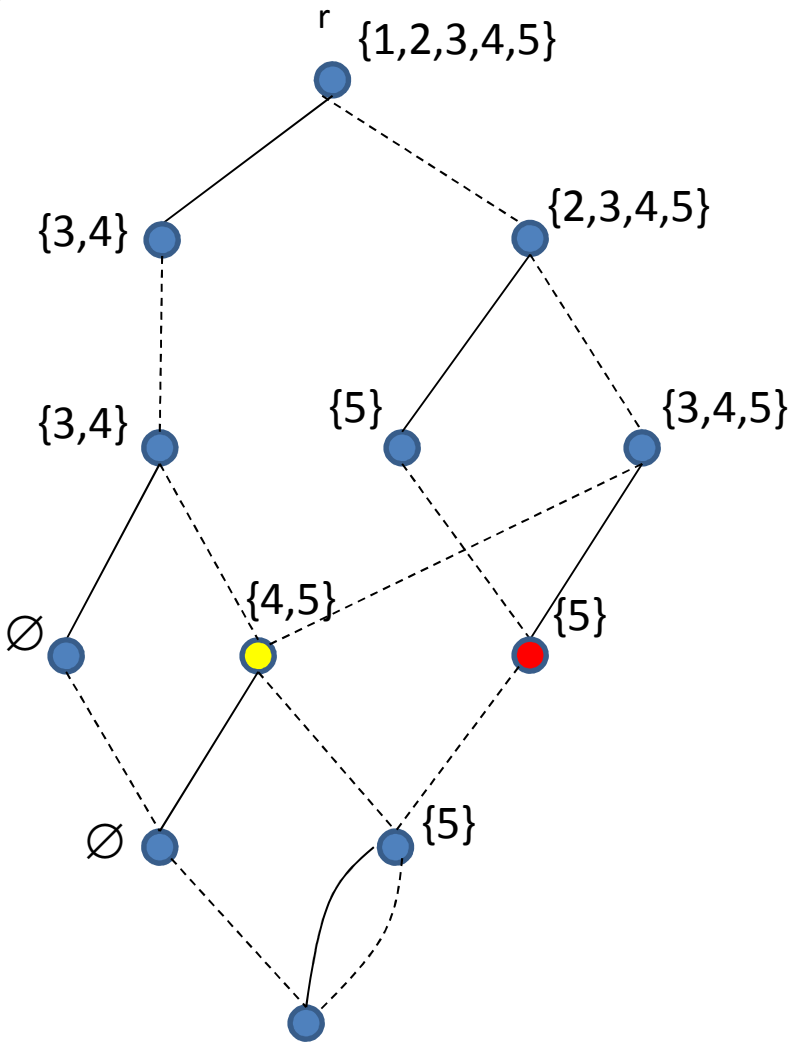
x_2

x_3

x_4

x_5

Relaxed BDD (width ≤ 3)

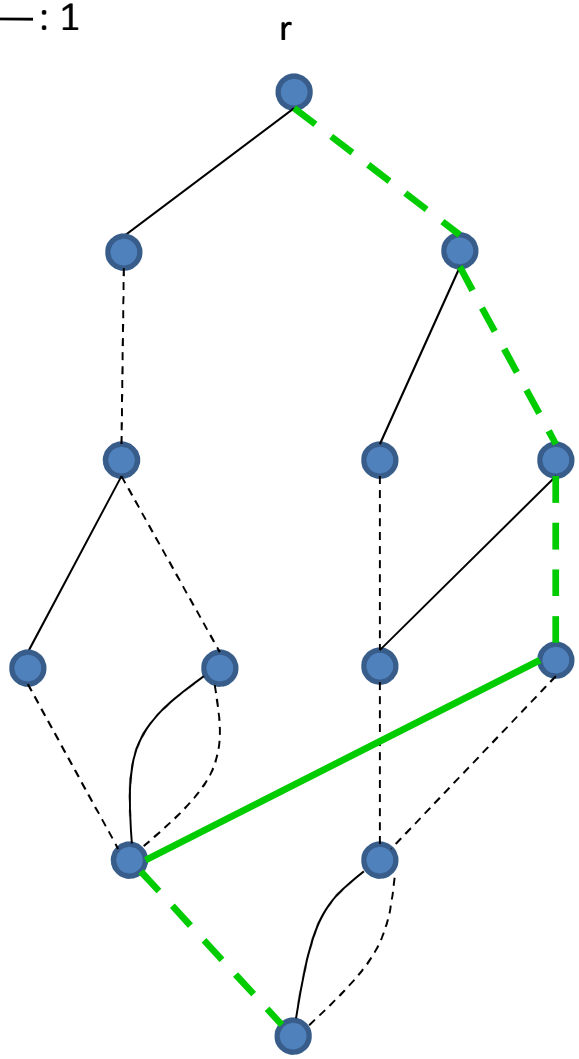


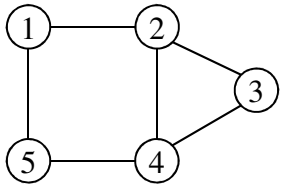
Relaxed BDD

----: 0

——: 1

Exact BDD





x_1

x_2

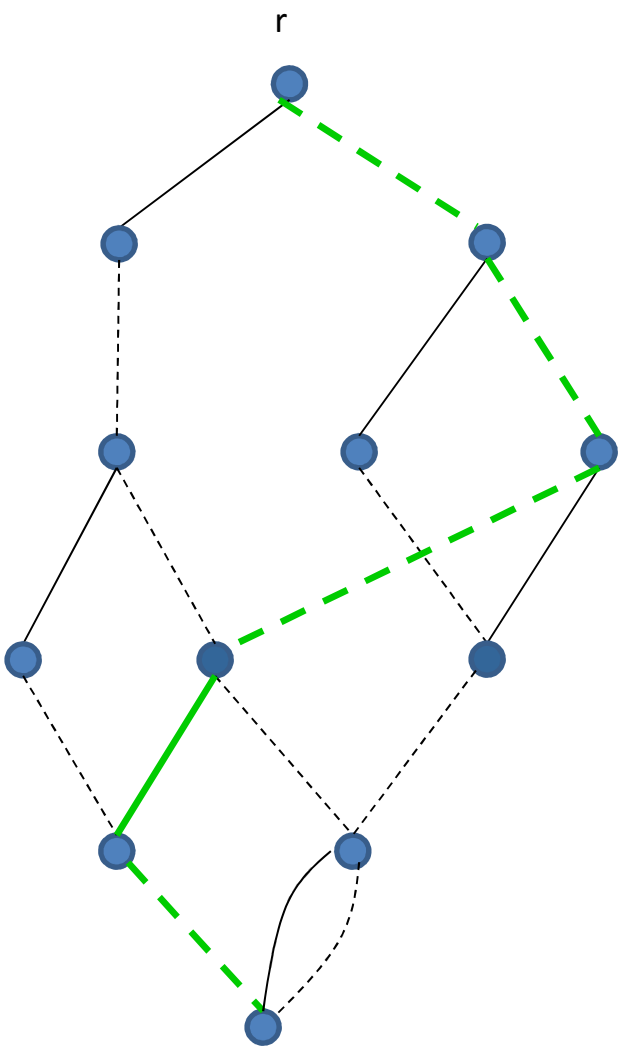
x_3

x_4

x_5

(0,0,0,1,0)

Relaxed BDD (width ≤ 3)



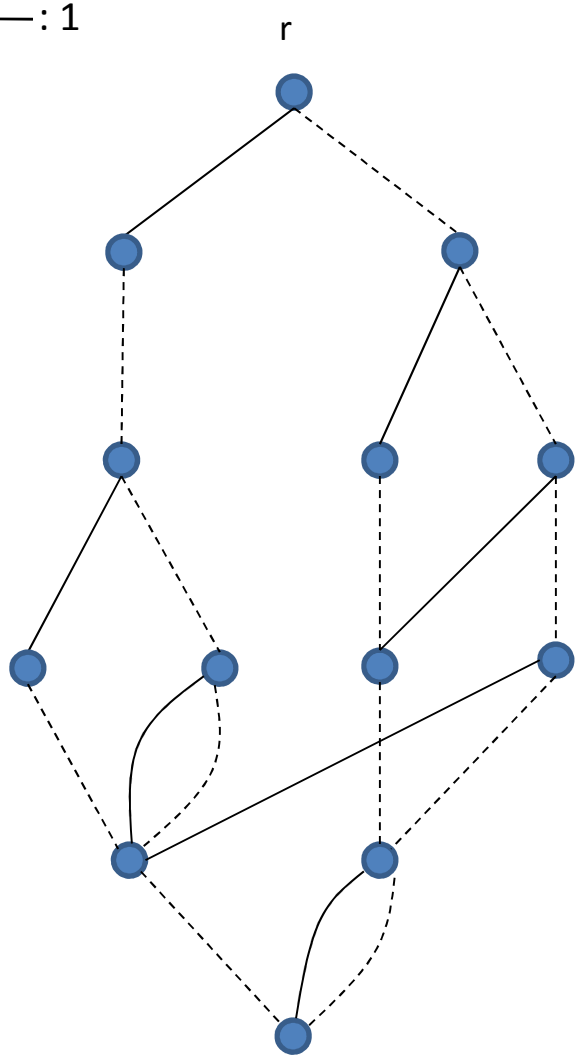
Relaxed BDD

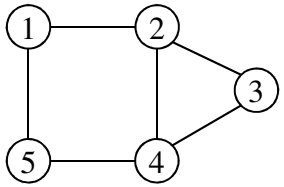
Carnegie Mellon Tepper SCHOOL OF BUSINESS

----: 0

——: 1

Exact BDD





x_1

x_2

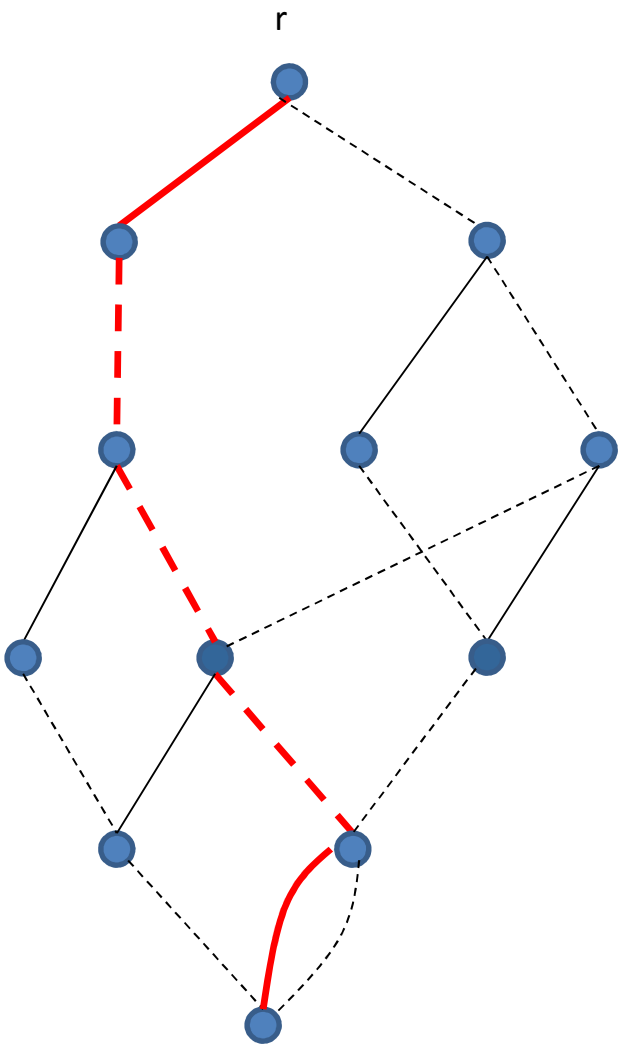
x_3

x_4

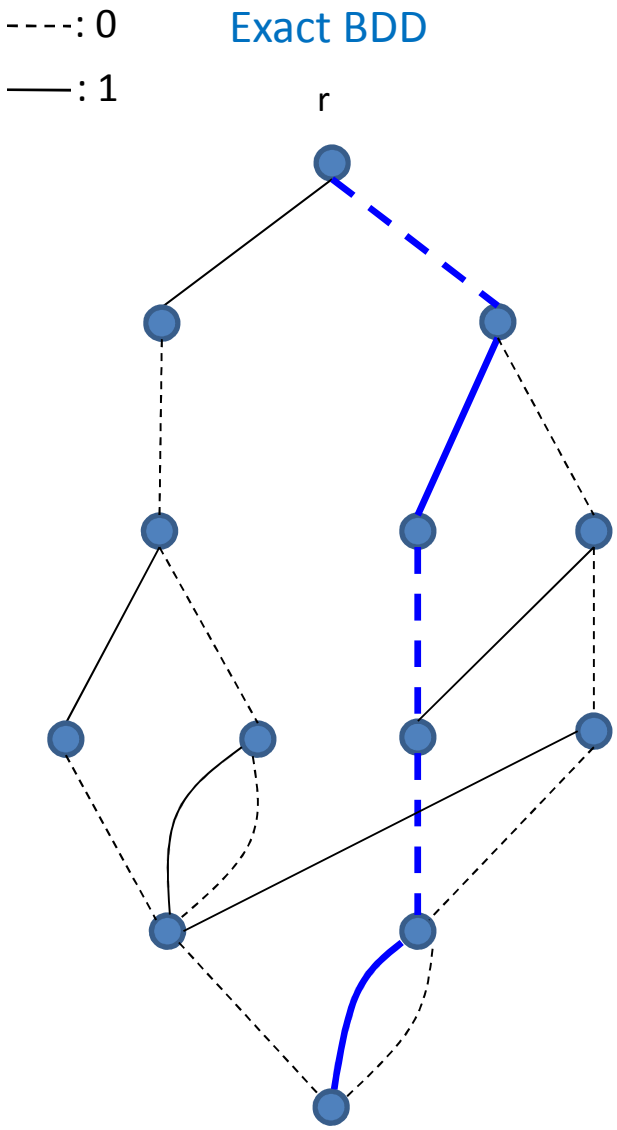
x_5

(1,0,0,0,1)

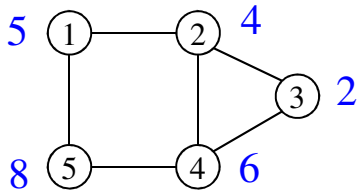
Relaxed BDD (width ≤ 3)



Evaluate Objective Function



max f(x) = 12



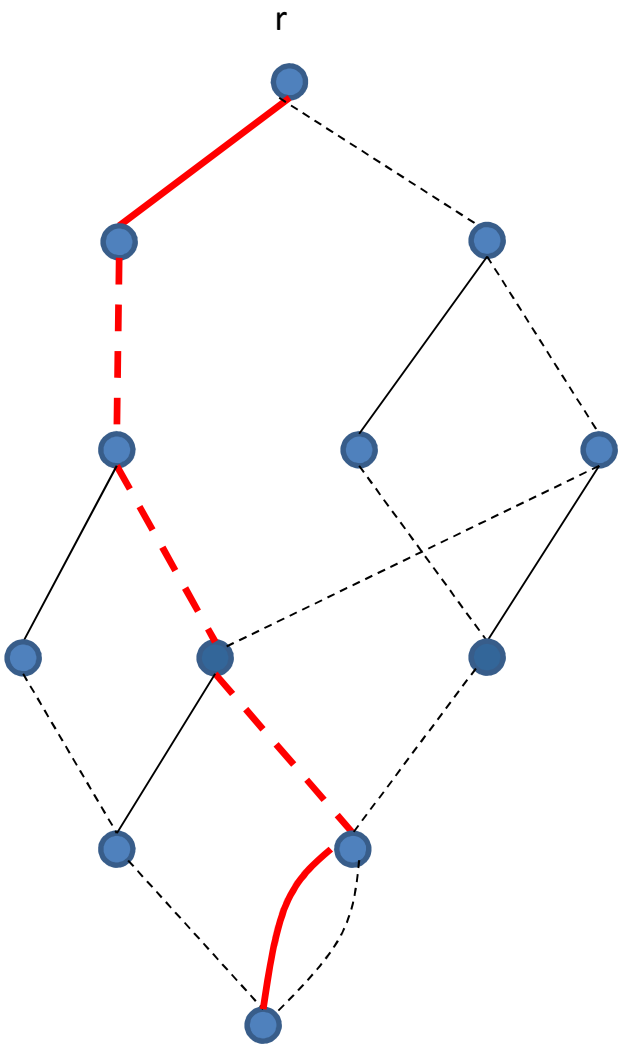
x_1

x_2

x_3

x_4

x_5



max f(x) = 13

Node merging heuristics

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Suppose layer j has nodes L_j with $|L_j| > W$ (max width)

Which subsets of non-equivalent nodes to merge?

- Random
  - select random subset of nodes to merge
- Minimum Longest Path (minLP)
  - sort nodes by increasing longest path value from r
  - merge the first $|L_j| - W + 1$ nodes (i.e., keep best W nodes)
- Minimum State Size (minSize)
  - sort nodes by decreasing state size
  - merge first 2 nodes until $|L_j| \leq W$ (larger states are more likely to have vertices in common and may be more similar)

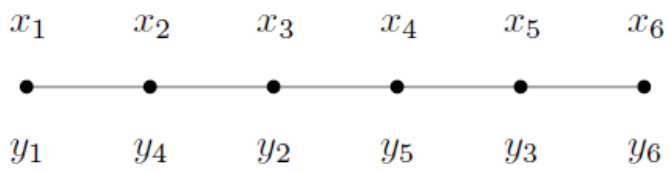
Variable Ordering

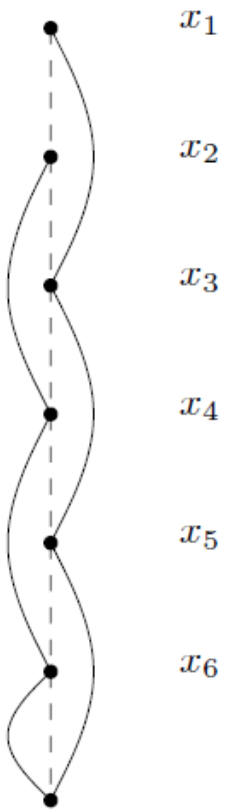
Carnegie Mellon Tepper SCHOOL OF BUSINESS

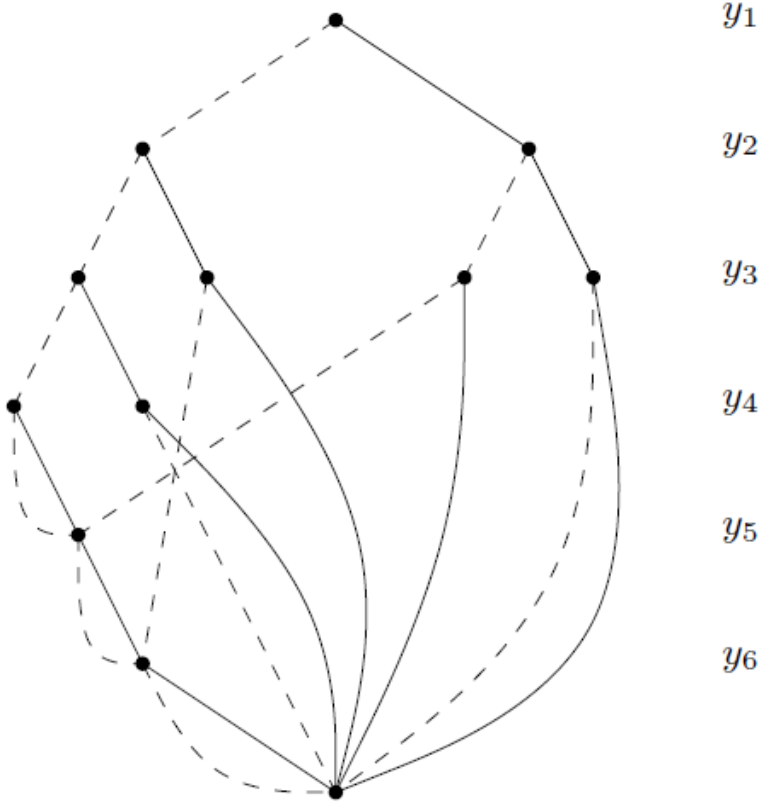
- Order of variables greatly impacts BDD size
  - also influences bound from relaxed BDD (see next)
- Finding 'optimal ordering' is NP-hard

- Insights from independent set as case study
  - formal bounds on BDD size
  - measure strength of relaxation w.r.t. ordering

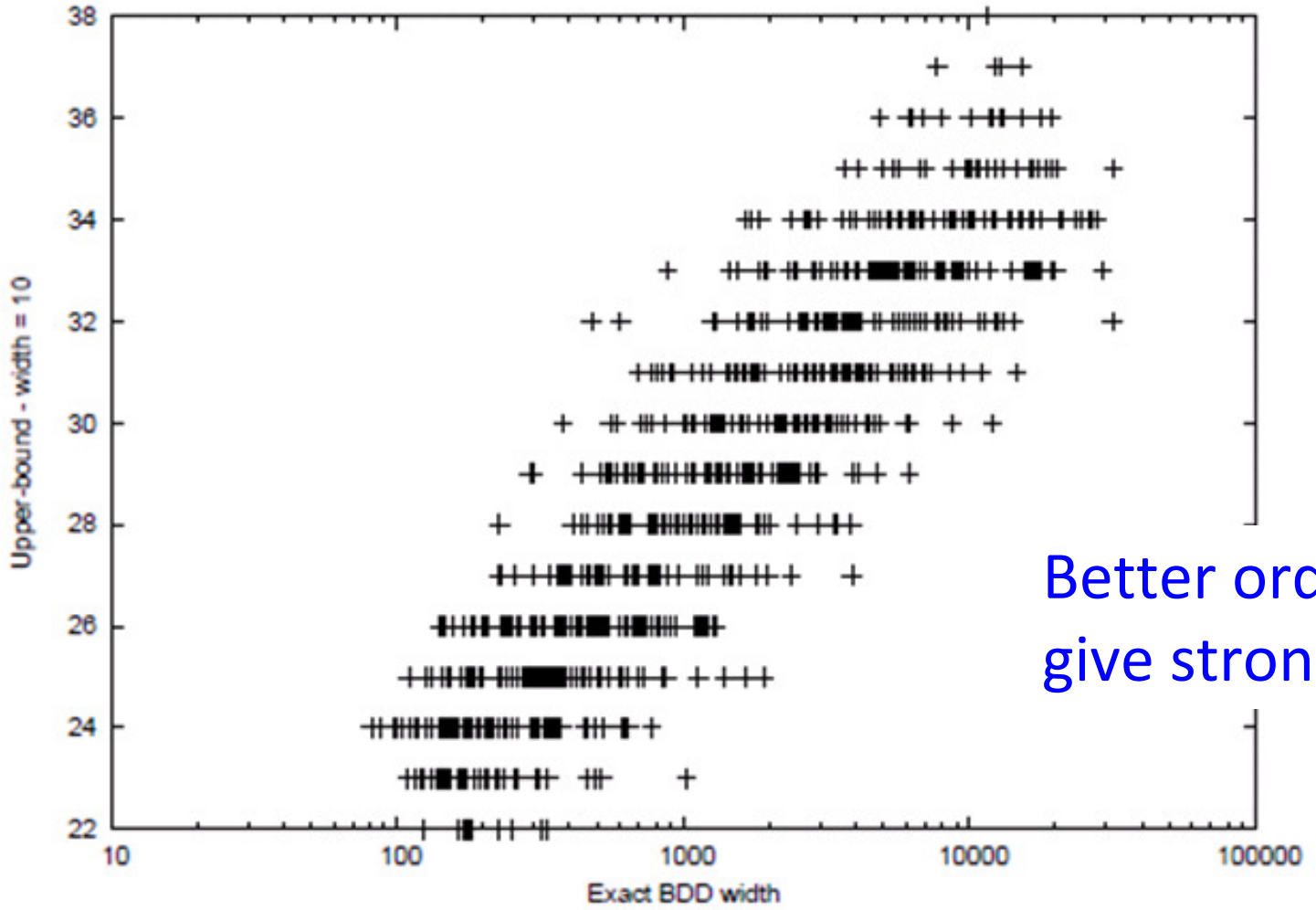
Exact BDD orderings for Paths







Many Random Orderings



Better orderings
give stronger bounds

For each random ordering, plot the exact BDD width and the bound from width-10 BDD relaxation

Formal Results for Independent Set

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Graph Class	Bound on Width
Paths	1
Cliques	1
Interval Graphs	n/2
Trees	n/2
General Graphs	Fibonacci Numbers

(The proof for general graphs is based on a maximal path decomposition of the graph)

Variable Ordering for Relaxed BDDs

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Random
  - select random variable ordering
- Minimum number of states (minState)
  - Given current layer j with node L_j
  - Select vertex that appears in fewest number of states of L_j (dynamic ordering; minimizes the size of the next layer)
- Maximal Path Decomposition (MPD)
  - Greedily compute a maximal path decomposition
  - Order the vertices by the order in which they appear in the paths (static ordering; provides bound on the width)

Impact of node merging and ordering

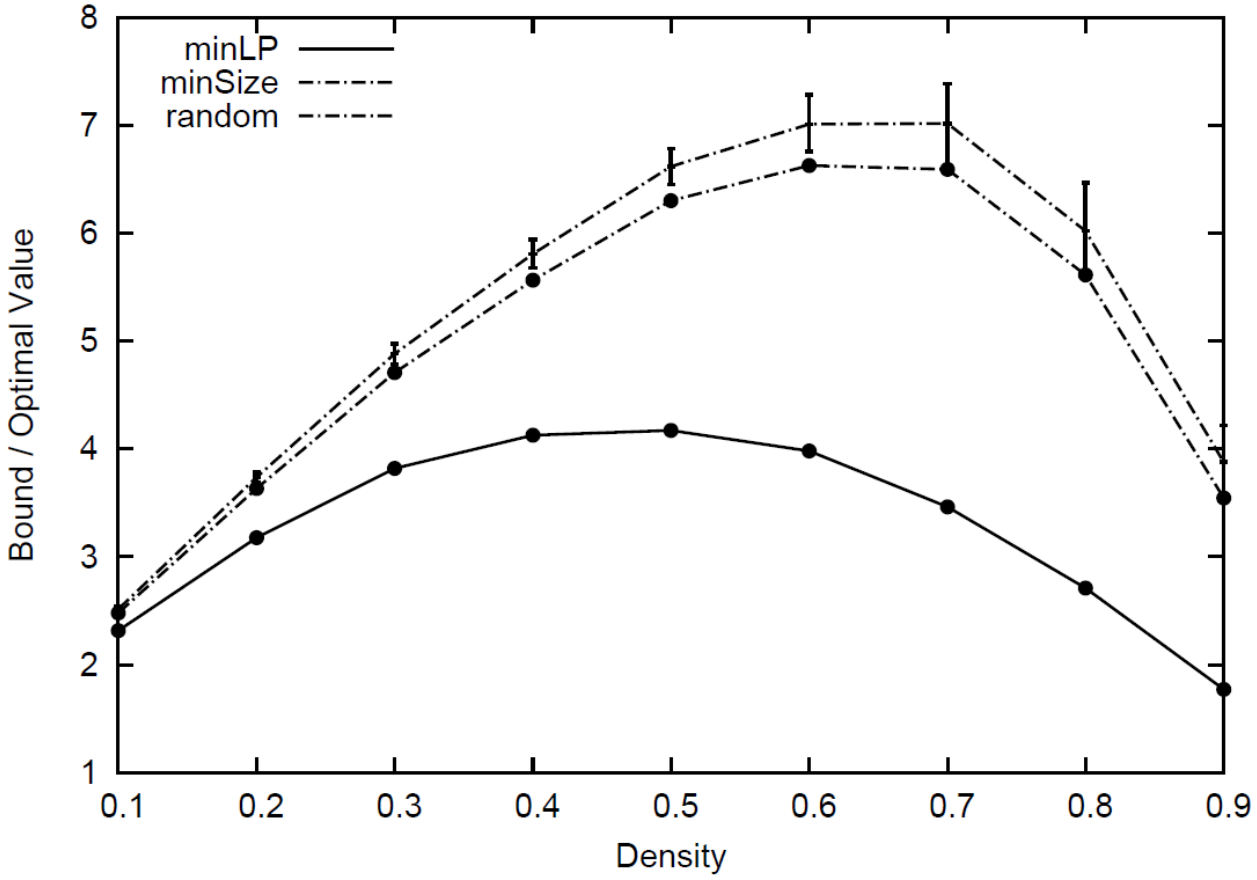
Carnegie Mellon Tepper SCHOOL OF BUSINESS

Experimental setup:

- 180 randomly generated instances
- Erdos-Renyi model G(n,p)
  - graph on n vertices
  - edge exists between pair of vertices with probability p
  - n = 200, p ∈ { 0.1, 0.2, …, 0.9 } (20 instances per p)

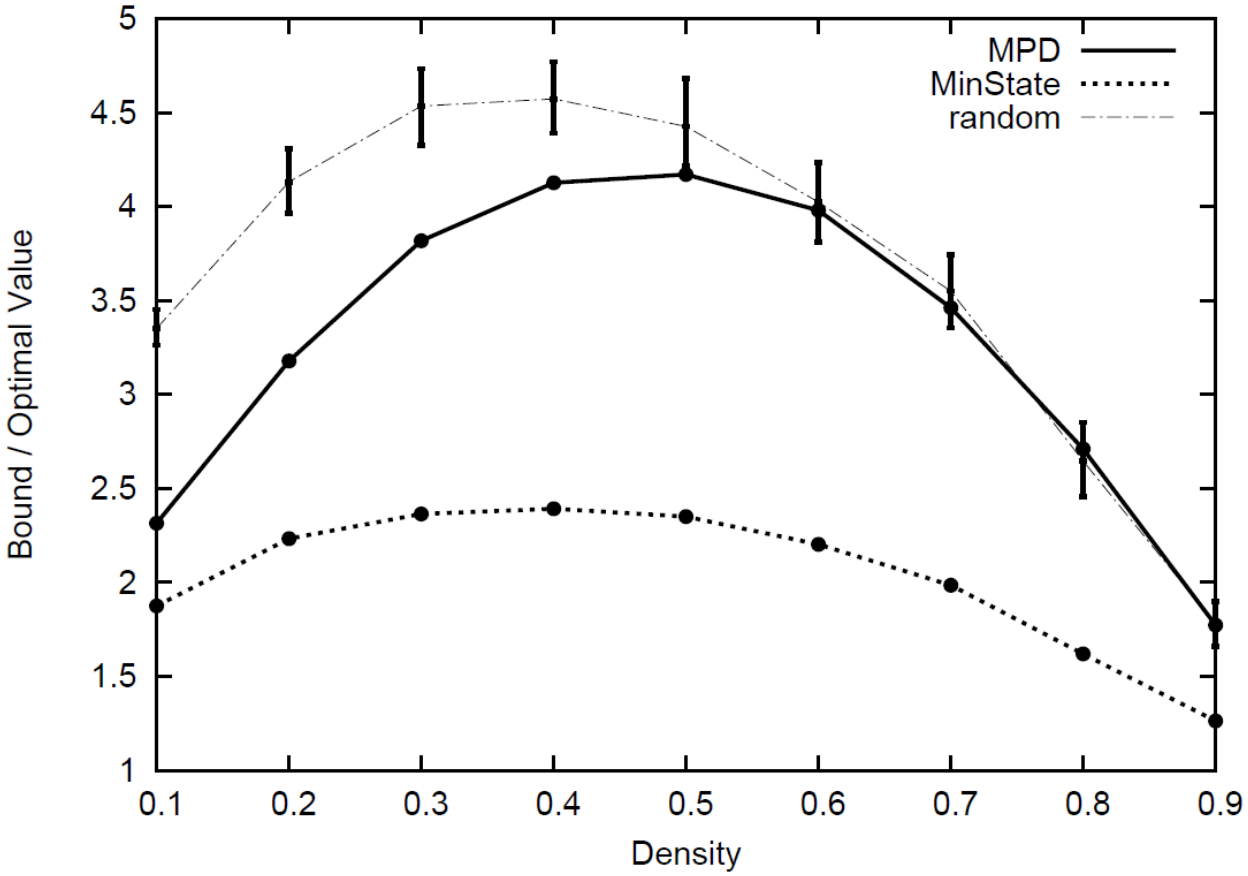
- Maximum BDD width W=10

Impact of node merging heuristics



- Variable ordering: maximal path decomposition (MPD)
- Each data point is average over 20 instances
- For random, line segment indicates range over 5 instances

Impact of variable ordering heuristics



- Node merging heuristic: minLP
- Each data point is average over 20 instances
- For random, line segment indicates range over 5 instances

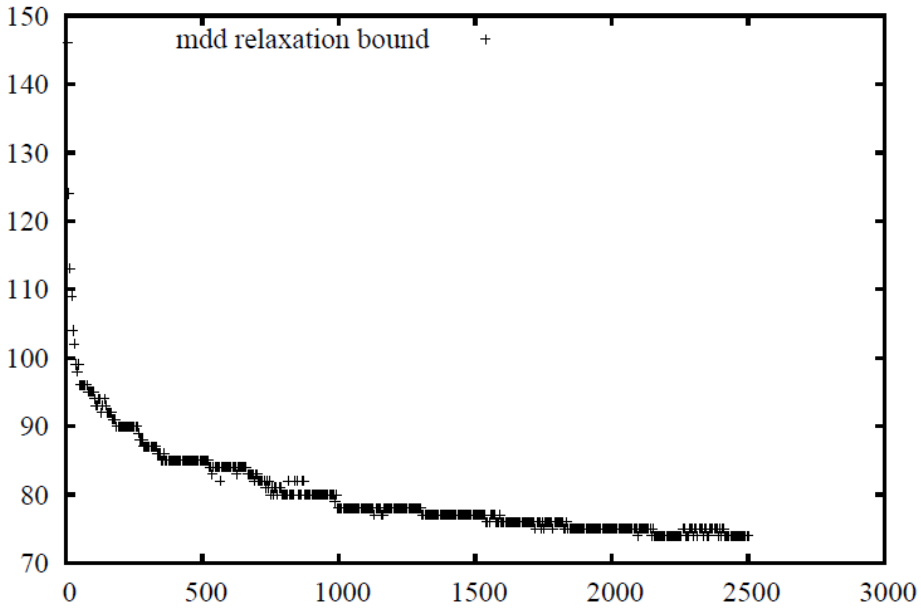
Experimental Evaluation

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Goals
  - Measure impact of maximum width on strength of bound
  - Compare BDD bounds to Linear Programming bounds
- LP Settings
  - LP model uses Clique Cover formulation
  - LP cuts from IBM ILOG CPLEX 12.4
  - root node relaxation, no presolve, barrier method
- BDD settings
  - variable ordering minState, node merging minLP
- Time Limit 3,600s
- Random + DIMACS clique instances

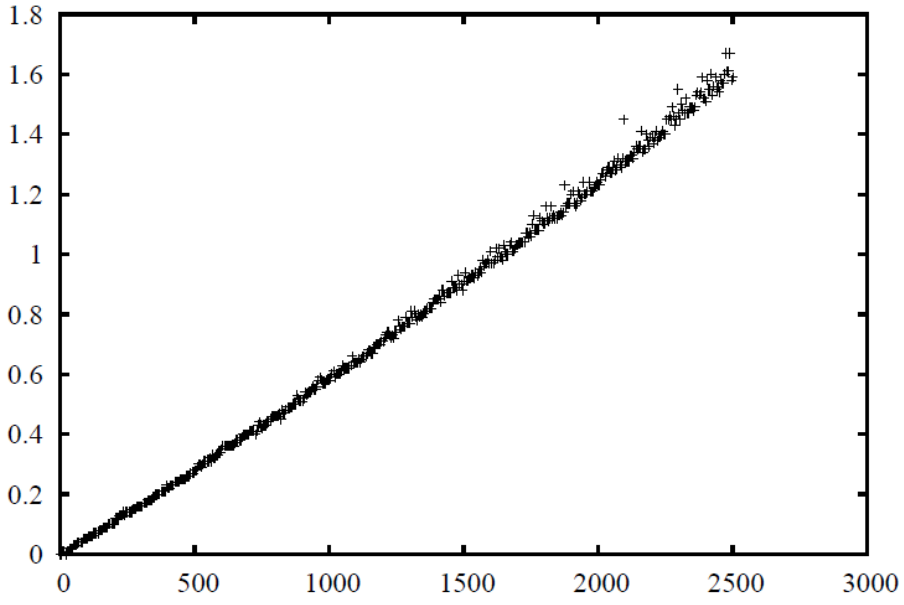
Impact of width on relaxation

upper bound



maximum width

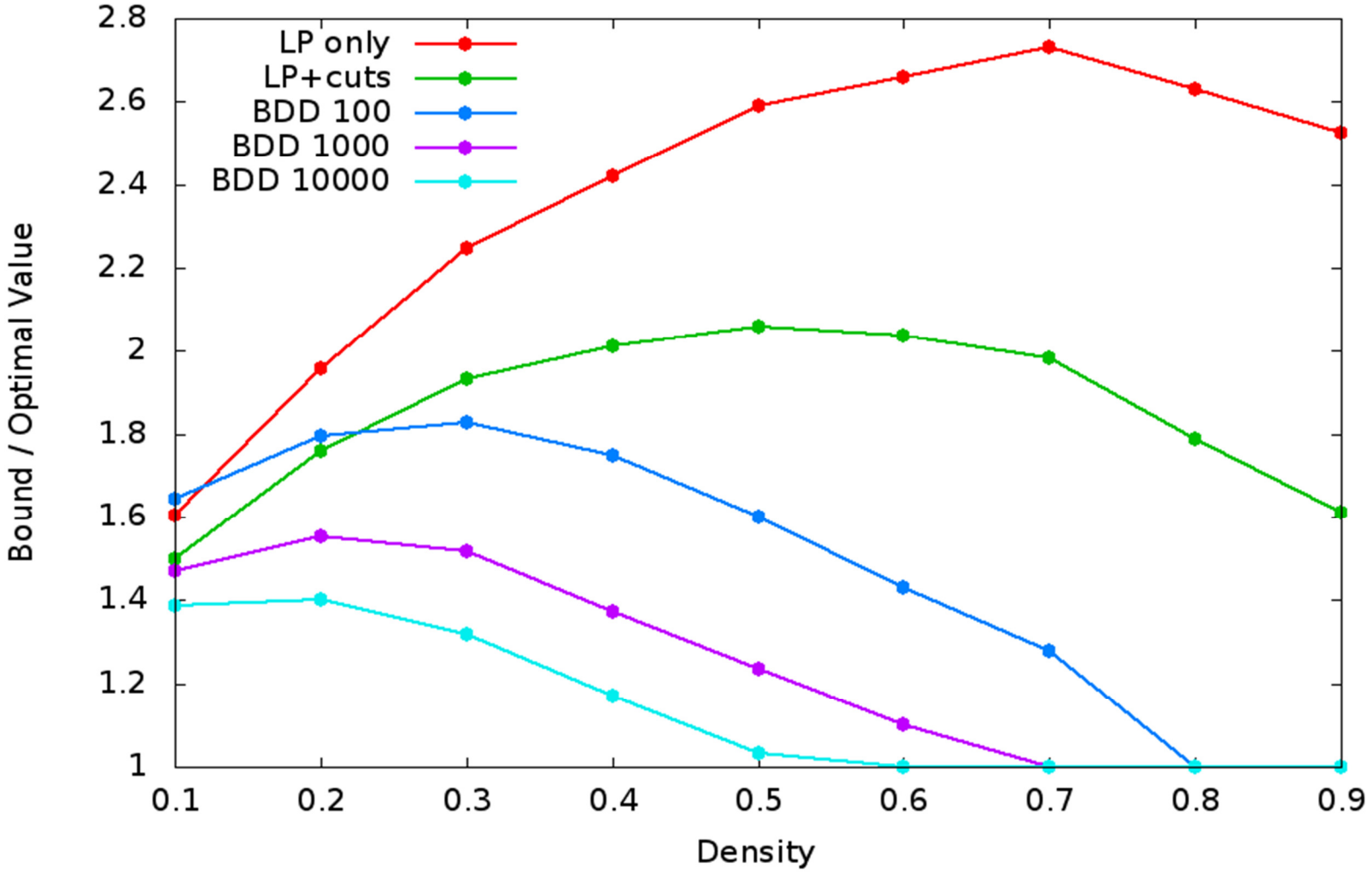
time (s)



maximum width

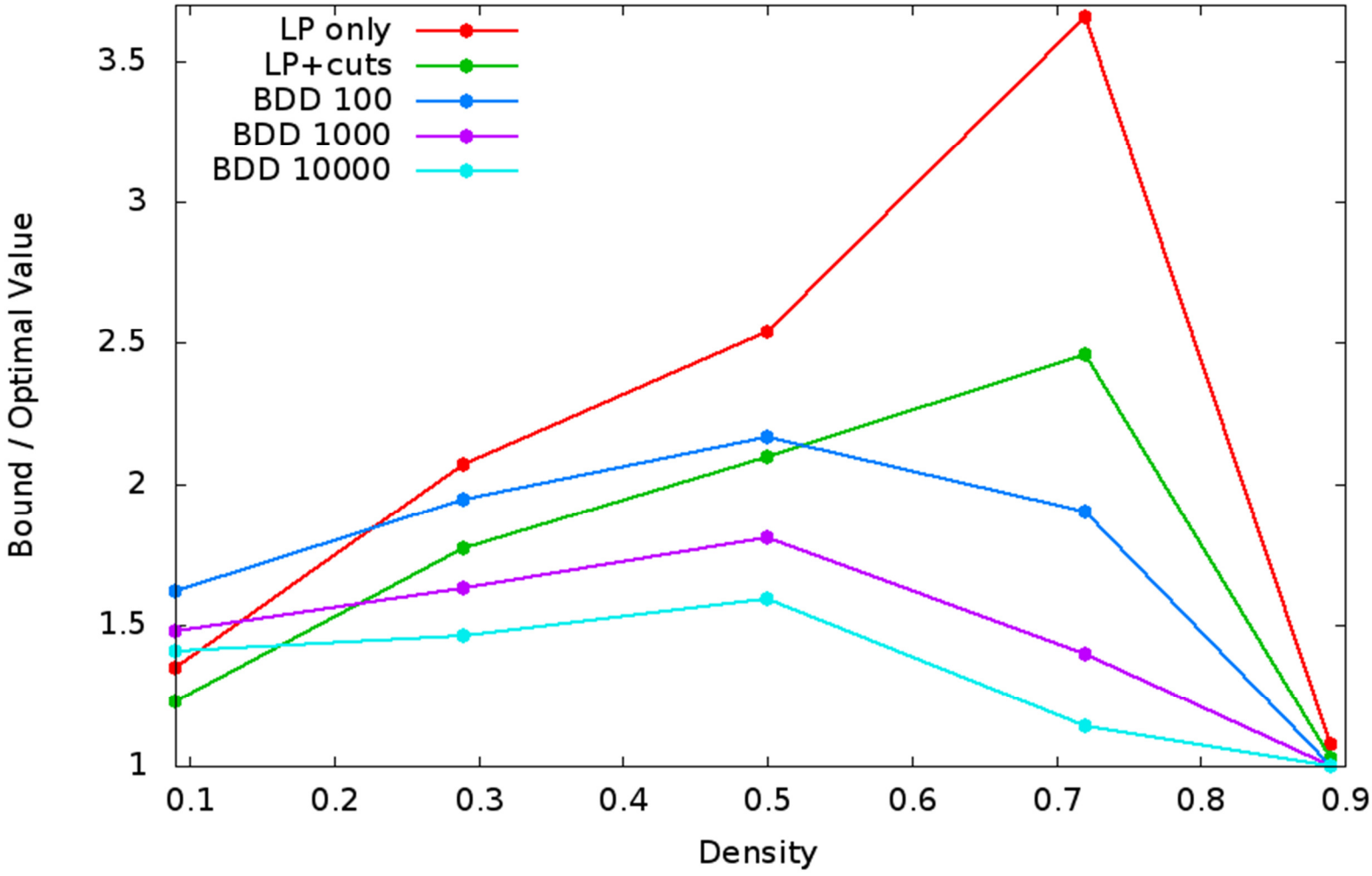
brock_200-2 instance

Bound quality versus density: Random



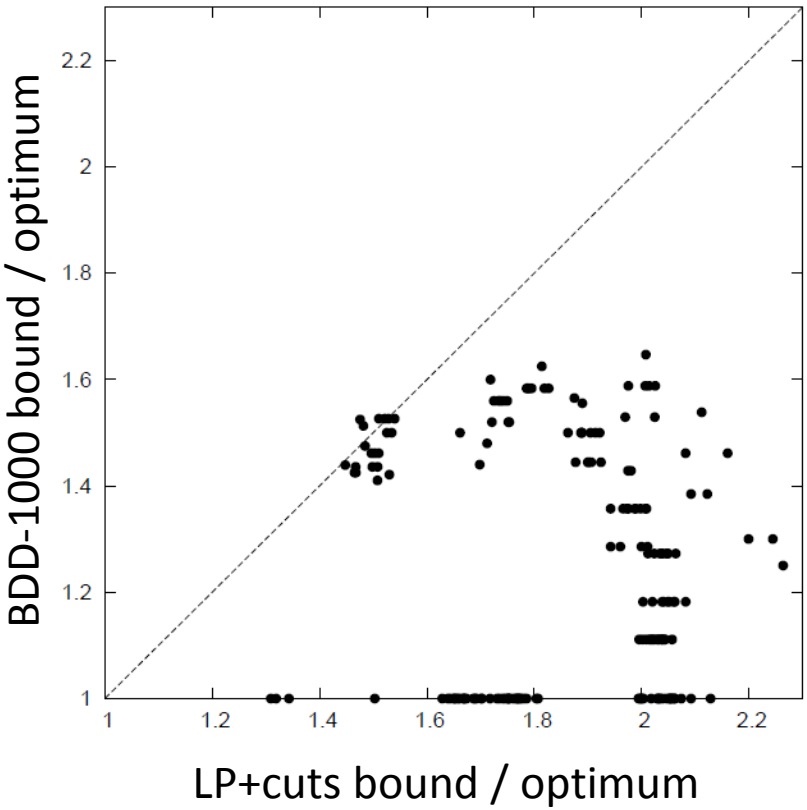
Each data point is geometric mean over 20 instances

Bound quality versus density: DIMACS

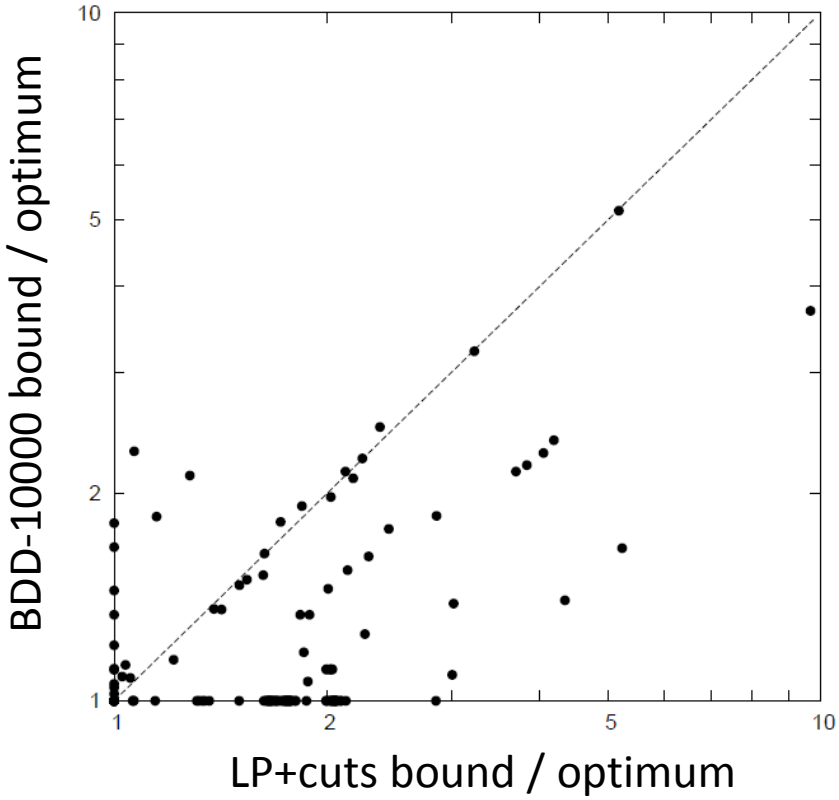


Each data point is geometric mean over 20 instances

Bound quality in more detail



- Random instances
- BDD bounds obtained in 5% of LP time



- DIMACS instances
- BDD bounds obtained in less time than LP except for sparsest

Restricted MDDs

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

Creating Restricted MDDs

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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

1. merging nodes, or
2. deleting nodes

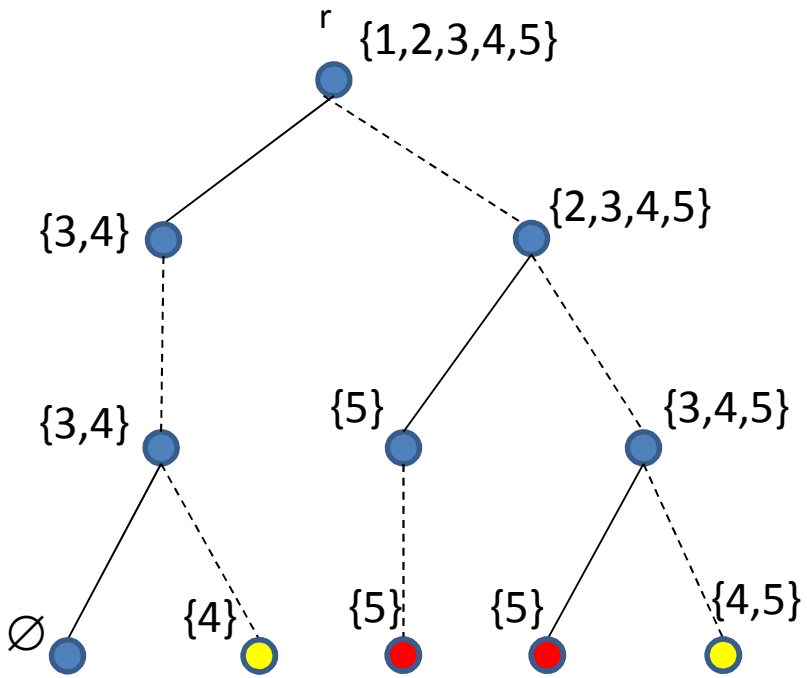
while ensuring that no non-solutions are introduced

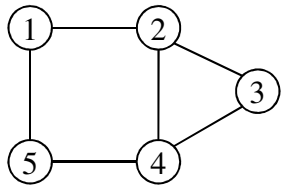
Node merging by example

Restricted BDD (width ≤ 3)

----: 0

——: 1



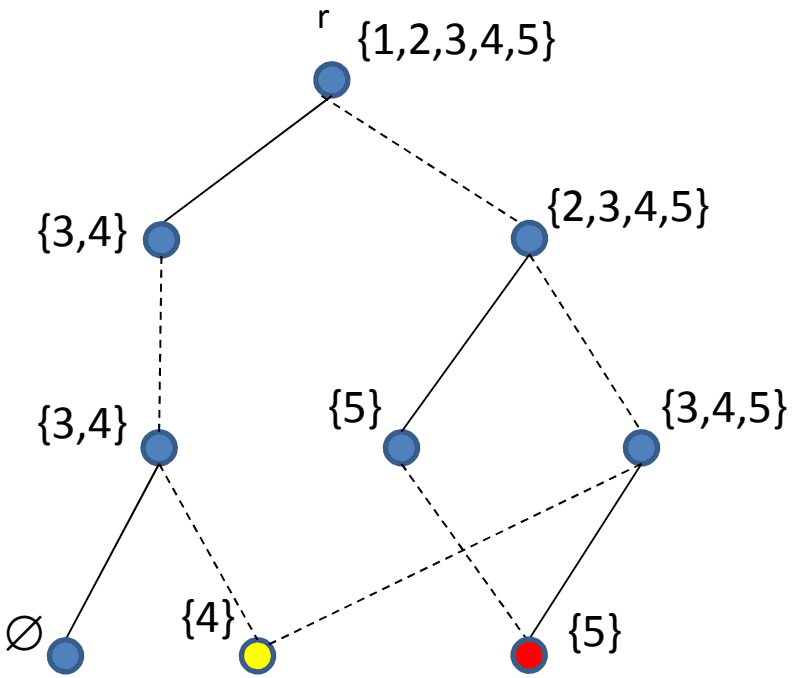


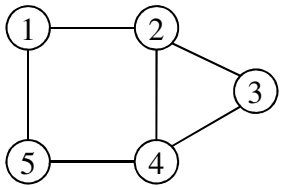
Node merging by example

Restricted BDD (width ≤ 3)

----: 0

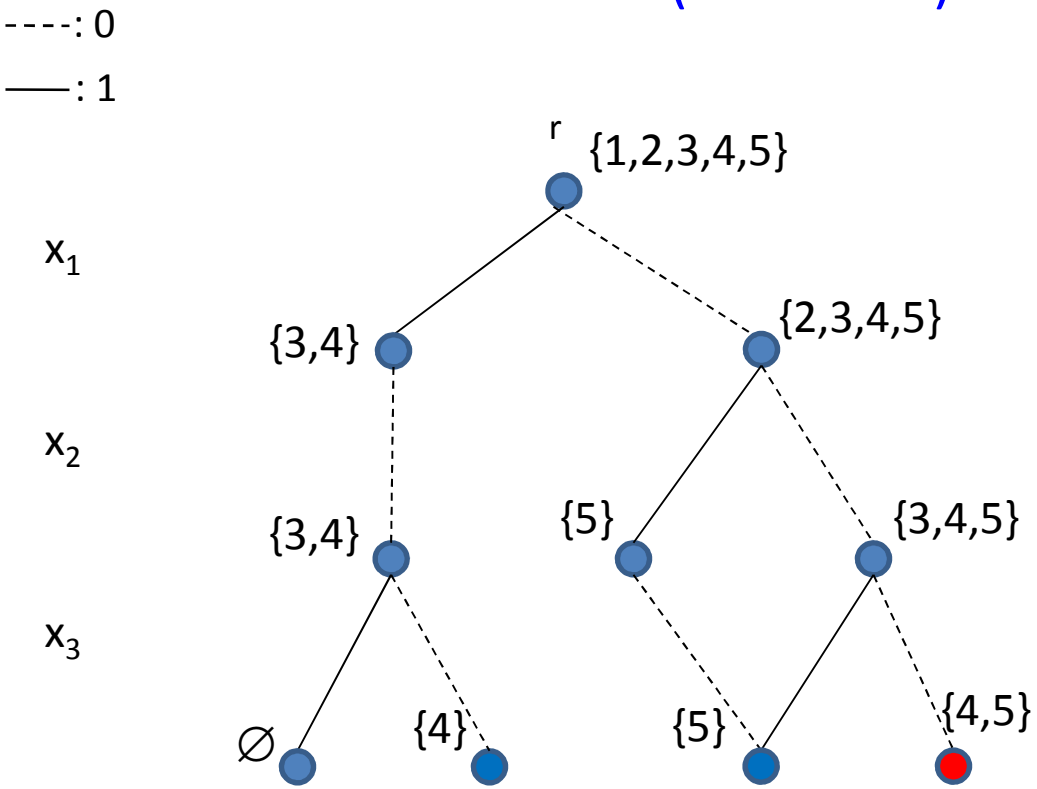
——: 1

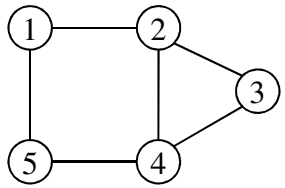




Node deletion by example

Restricted MDD (width ≤ 3)





In practice, node deletion superior to node merging (similar or better bounds, but much faster)

Node deletion heuristics

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Similar to node merging heuristics for MDD relaxations:

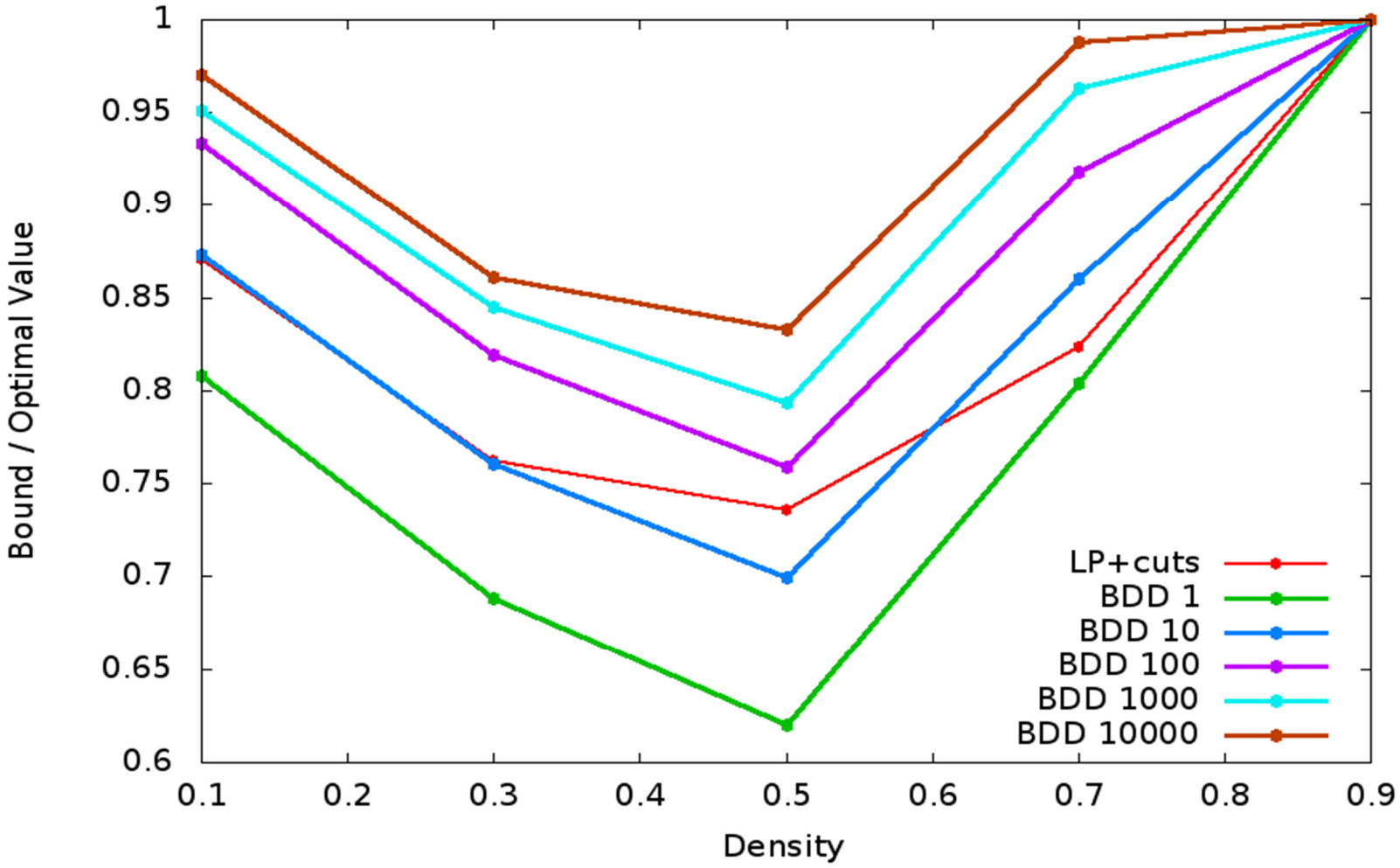
- Random
- Minimum Longest Path (minLP)
  - sort nodes by increasing longest path value from r
  - delete the first $|L_j| - W + 1$ nodes (i.e., keep best W nodes)
- Minimum State Size (minState)

Experimental Evalution

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Compare with Integer Programming (CPLEX)
  - LP relaxation + cutting planes
  - Root node solution
- DIMACS instance set
- MDDs with varying maximum width

IP versus BDD heuristic



Each data point is geometric mean over 20 instances

More Applications

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Methods for MDD relaxations and restrictions can easily be extended to other problems

- Knapsack problem, Set covering, Set packing, Bin packing (also multi-dimensional),…
- Key is the state representation

One more example: Set covering

Set Covering Problem

- Given set S={1,…,n} and subsets $C_1,\ldots,C_m$ of S
- Find a subset X of S with minimum cardinality such that $|C_i \cap X| \geq 1$ for all i=1,…,m

$$\begin{aligned} \text{min} \quad & \sum_j x_j \\ \text{s.t.} \quad & \sum_{j \text{ in } Ci} x_j \geq 1 \qquad \text{for all } i=1,\ldots,m \\ & x_1,\ldots,x_n \text{ binary} \end{aligned}$$

Set covering example

Carnegie Mellon Tepper SCHOOL OF BUSINESS

minimize $x_1 + x_2 + x_3 + x_4 + x_5 + x_6$

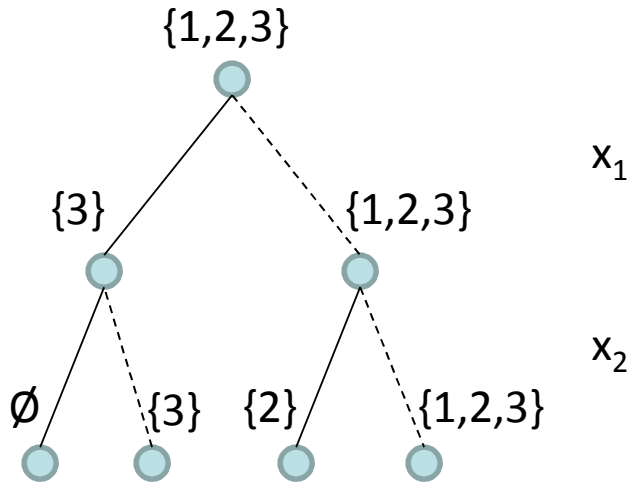
s.t. $x_1 + x_2 + x_3 \geq 1$ (1)

$x_1 + x_4 + x_5 \geq 1$ (2)

$x_2 + x_4 + x_6 \geq 1$ (3)

For BDD we need

- State information
  - set of uncovered constraints
- Merging rule
  - for relaxation: intersection
  - for restriction: union

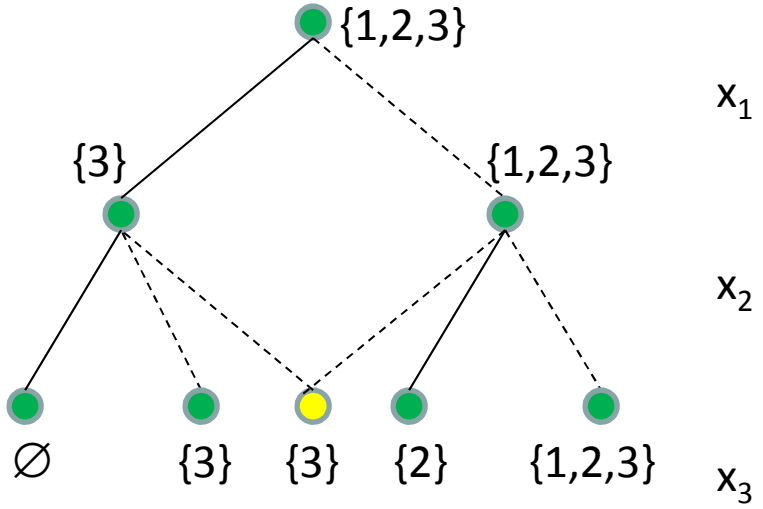


Building relaxed BDD for set covering

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Relaxed BDD (width ≤ 3)

$x_1 + x_2 + x_3 \geq 1$ (1)
$x_1 + x_4 + x_5 \geq 1$ (2)
$x_2 + x_4 + x_6 \geq 1$ (3)



Merging rule

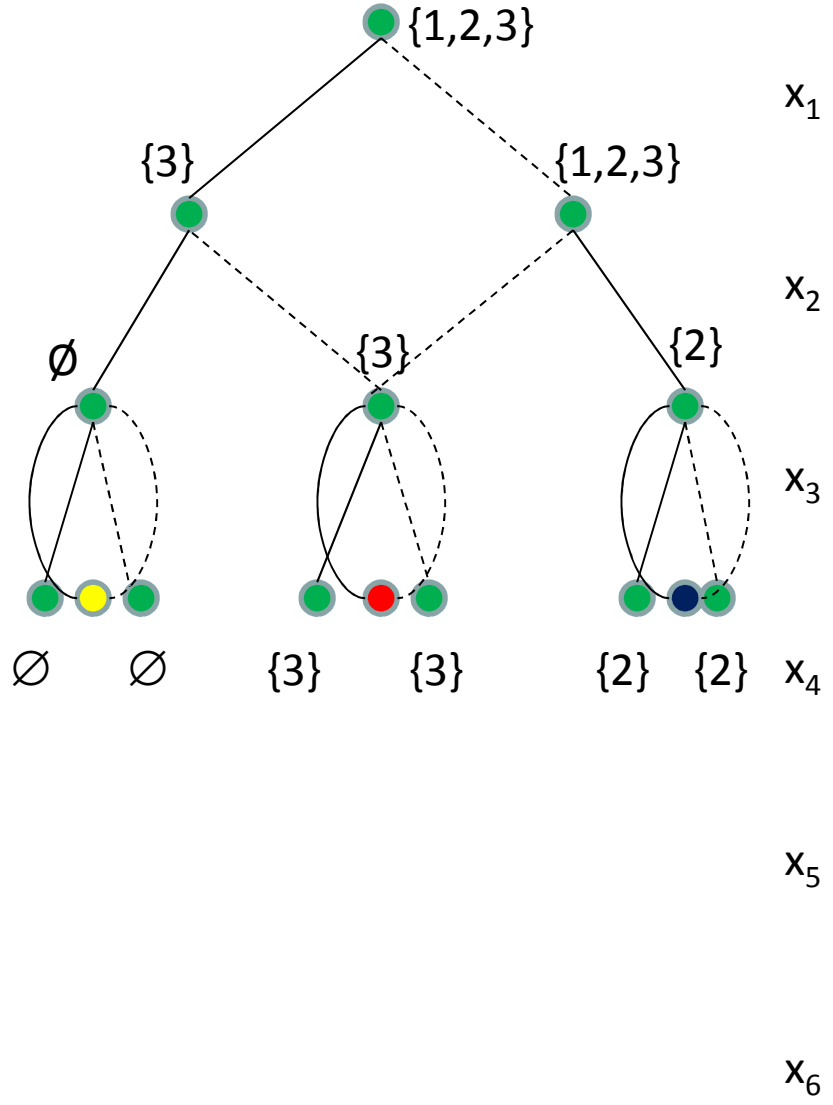
- for relaxation: intersection
- (for restriction: union)

Building relaxed BDD for set covering

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Relaxed BDD (width ≤ 3)

$x_1 + x_2 + x_3 \geq 1 \quad (1)$
$x_1 + x_4 + x_5 \geq 1 \quad (2)$
$x_2 + x_4 + x_6 \geq 1 \quad (3)$



Merging rule

- for relaxation: intersection
- (for restriction: union)

Building relaxed BDD for set covering

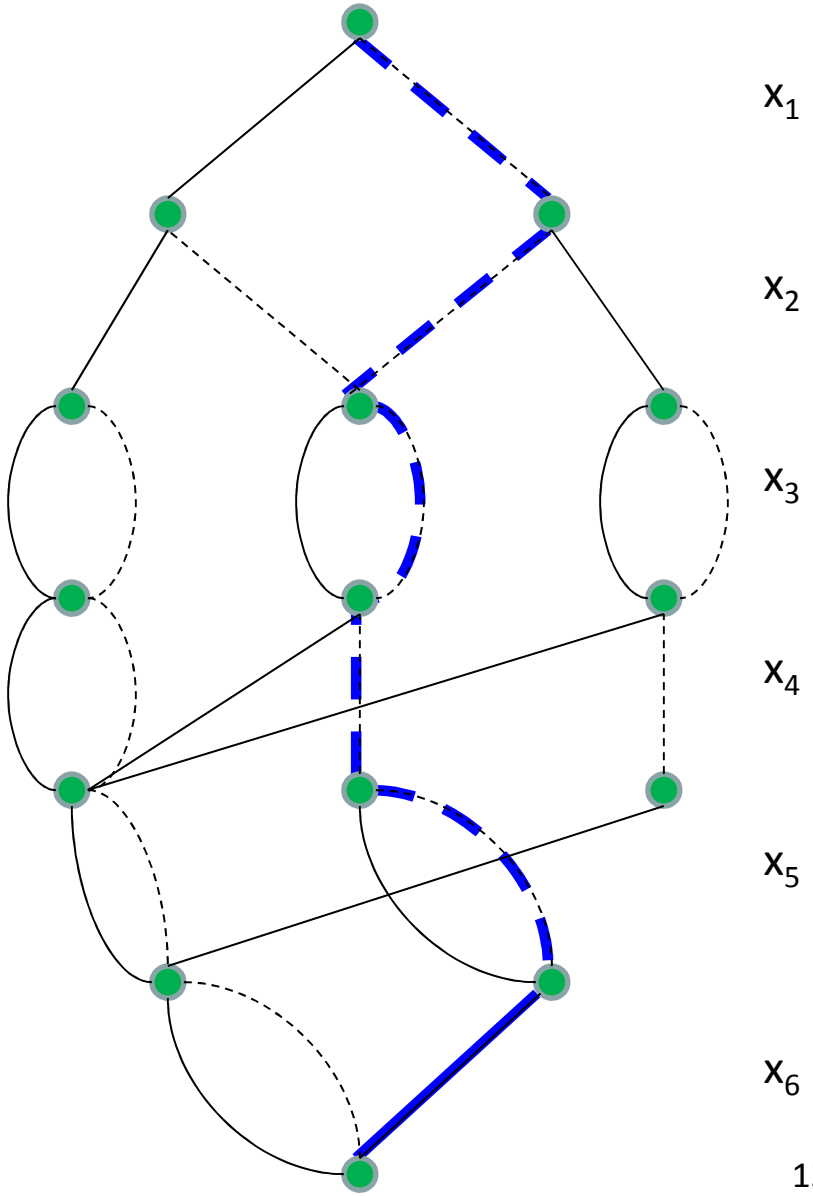
Carnegie Mellon Tepper SCHOOL OF BUSINESS

Relaxed BDD (width ≤ 3)

$x_1 + x_2 + x_3 \geq 1$ (1)
$x_1 + x_4 + x_5 \geq 1$ (2)
$x_2 + x_4 + x_6 \geq 1$ (3)

Merging rule
- for relaxation: intersection
- (for restriction: union)

Edge weights equal to objective coefficients: shortest path gives lower bound



Tightening the Lower Bound

- Value extraction method
  - Given: an MDD relaxation, M
  - Given: a valid lower bound, v
  - Extract all paths in M that correspond to solutions with objective function value equal to v in the form of another MDD $M|_{z=v}$
- Creating $M|_{z=v}$ can be done efficiently
- Apply MDD-based CP to $M|_{z=v}$ in order to either
  - Increase v to v+1 (if no solution exists)
  - Find a feasible (and optimal) solution

Experimental Results

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Investigate whether relaxation MDDs are able to capture and exploit problem structure
  - We consider structured set covering problems
- Purest structure: all constraints are defined on consecutive variables
  - TU constraint matrix; easy for IP
  - Exact MDD has bounded width; also easy for MDD

Instance Generation

- We generated random instances
  - Fix number of variables per constraint, k
  - Vary the bandwidth b
  - Randomly assign a 0 to b – k ones in the bandwidth

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array} \longrightarrow \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

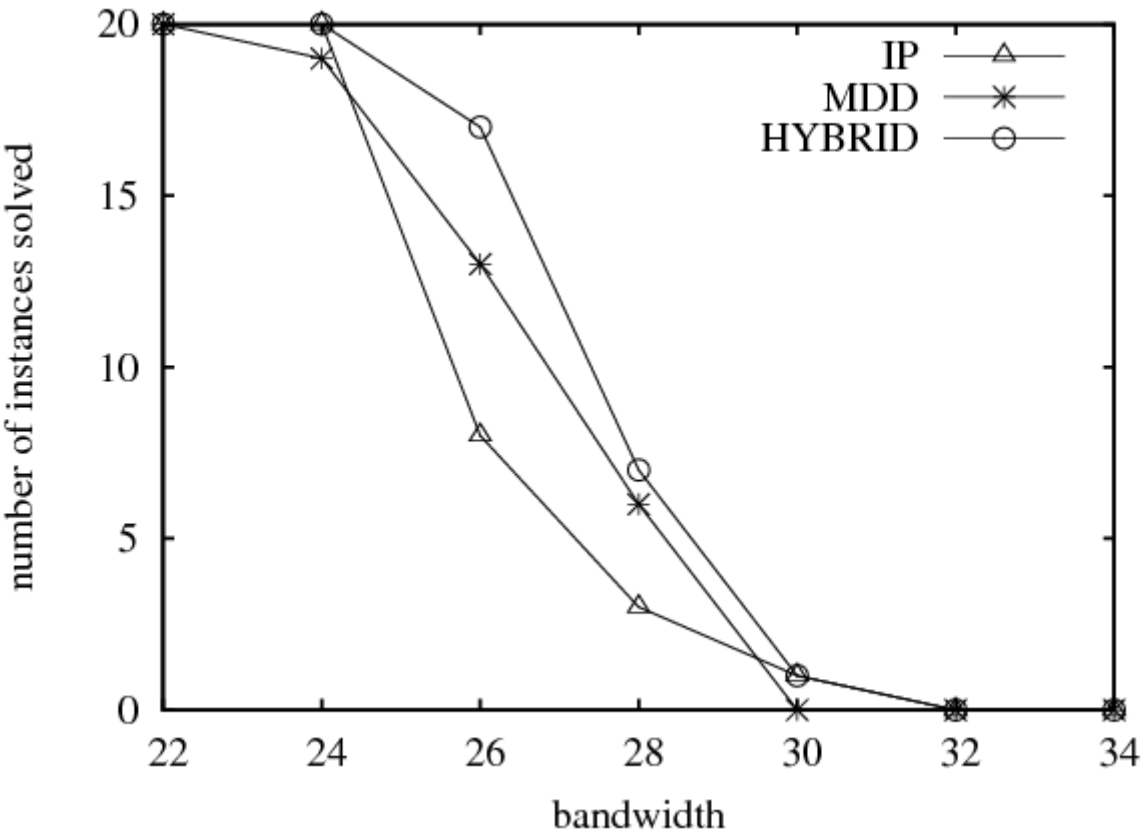
- Destroys both the TU property for IP and the bounded width property for MDD

Computational Results

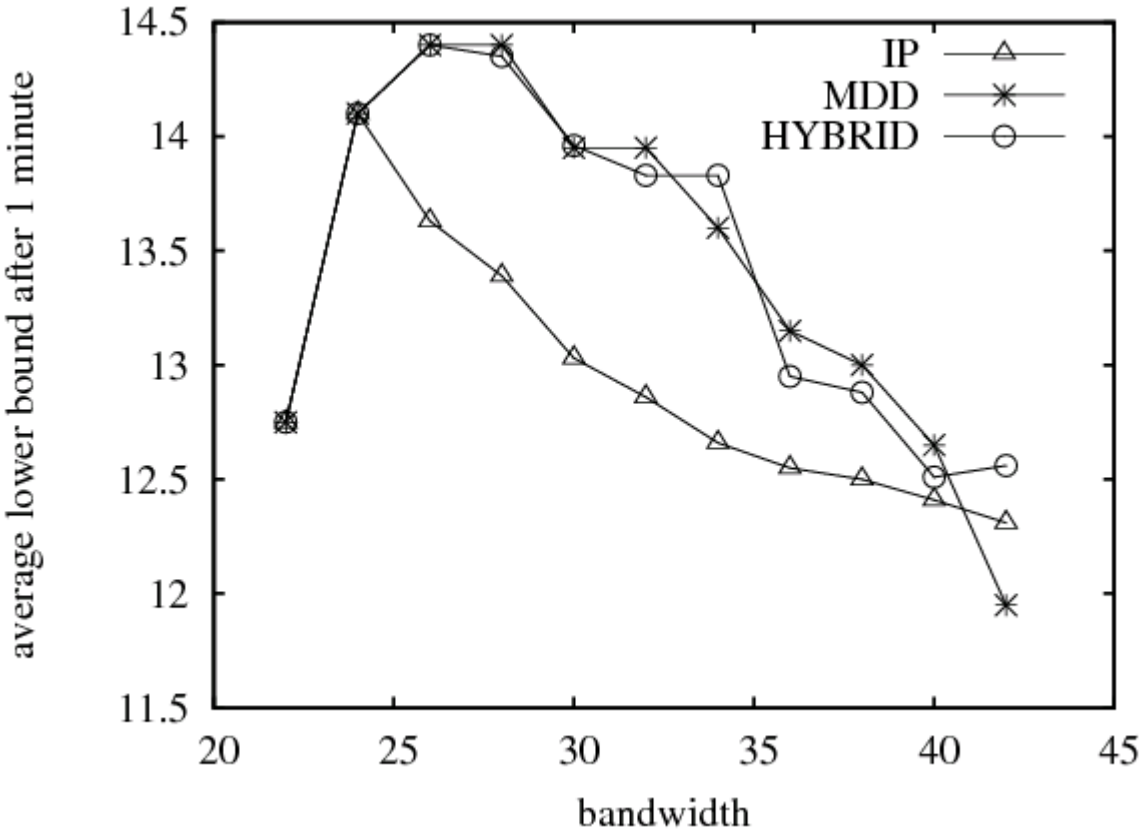
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 250 variables, k = 20, b $\in$ {22,…,44}, 20 instances for each bandwidth
- Compare 3 different solution methods
  - Pure-IP (CPLEX)
  - Pure-MDD (Value Extraction)
  - Hybrid (1/10 solution time given to pure-MDD and then pass bound to CPLEX)

Number of Instances Solved (1 min.)

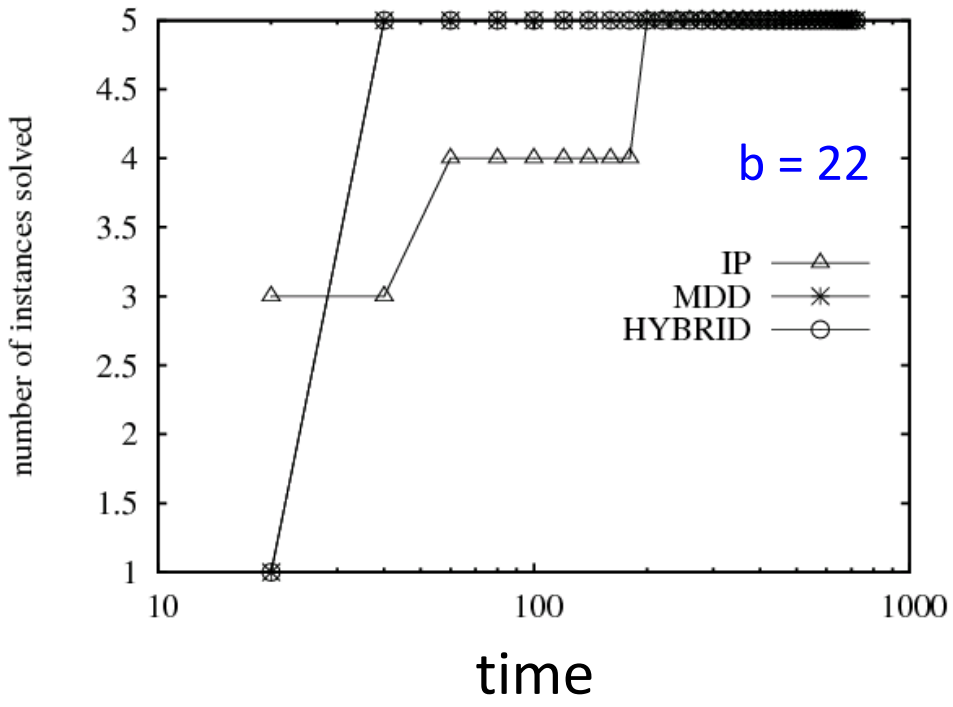


Average Ending Lower Bound (1 min)



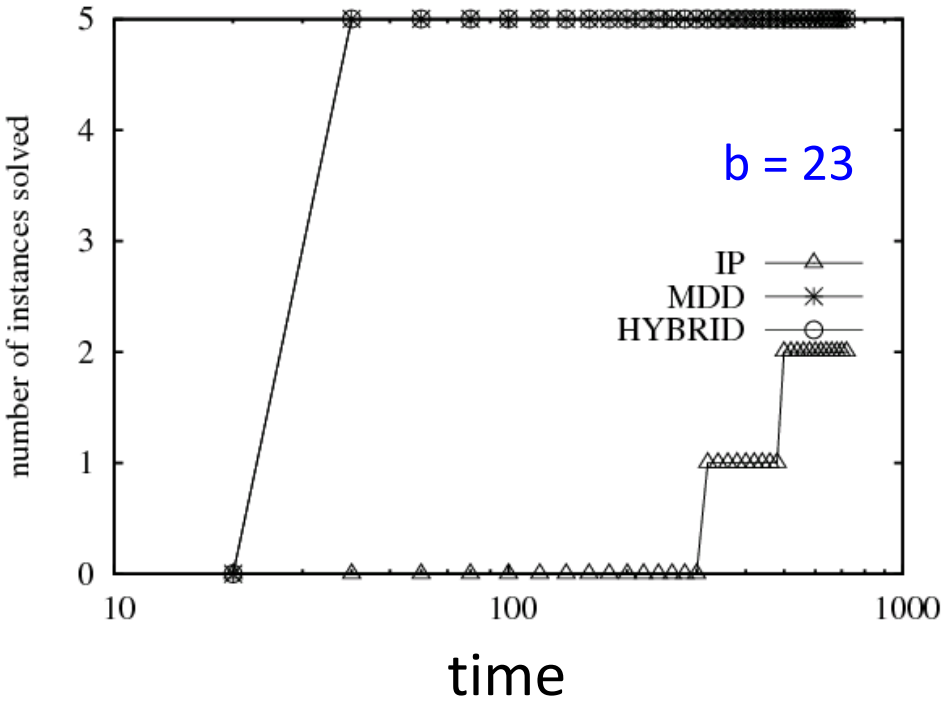
Larger Instances

nr instances solved



time

nr instances solved



time

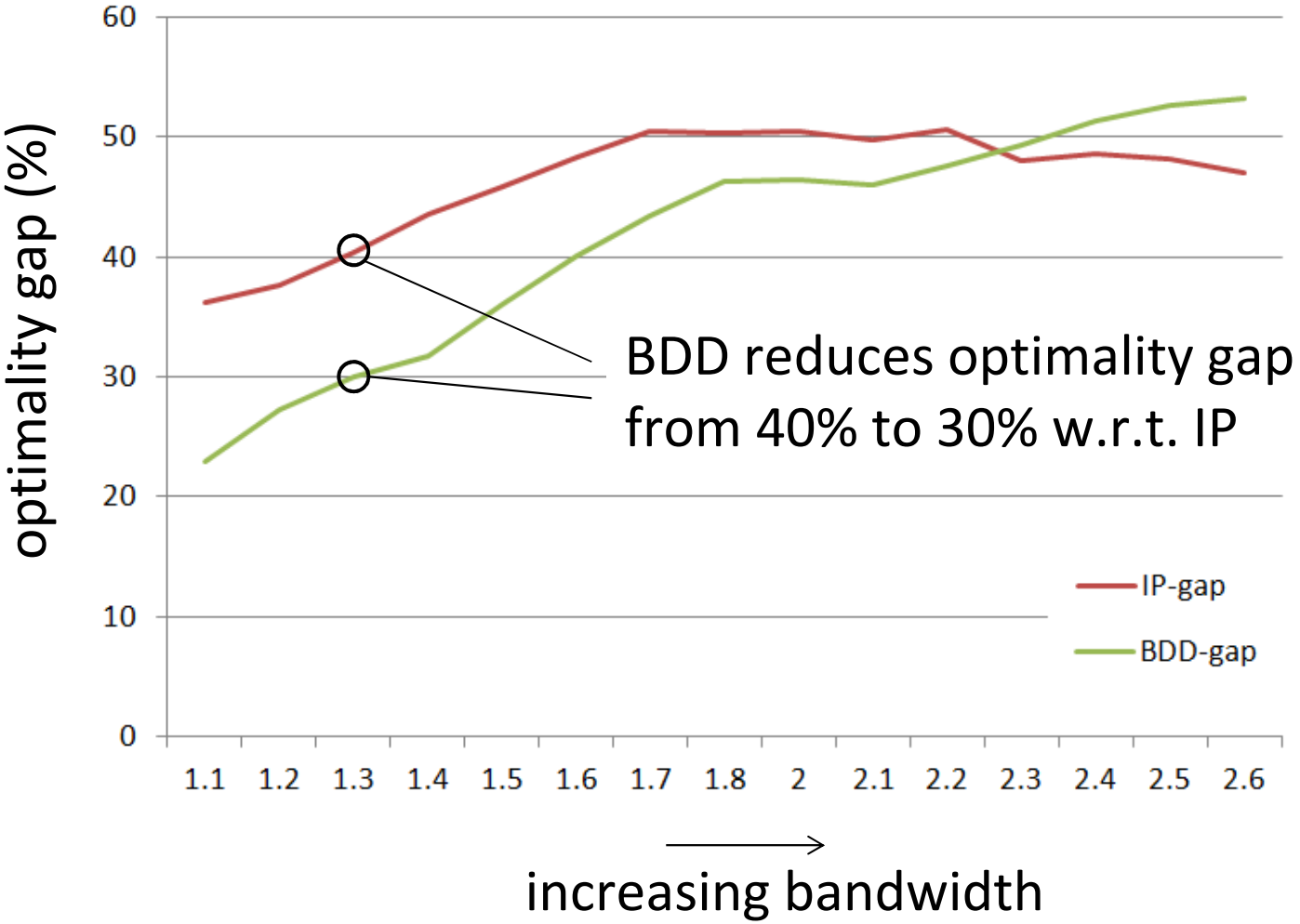
500 variables, 5 instances, k = 20, b∈{22,…,25}

Restricted BDDs

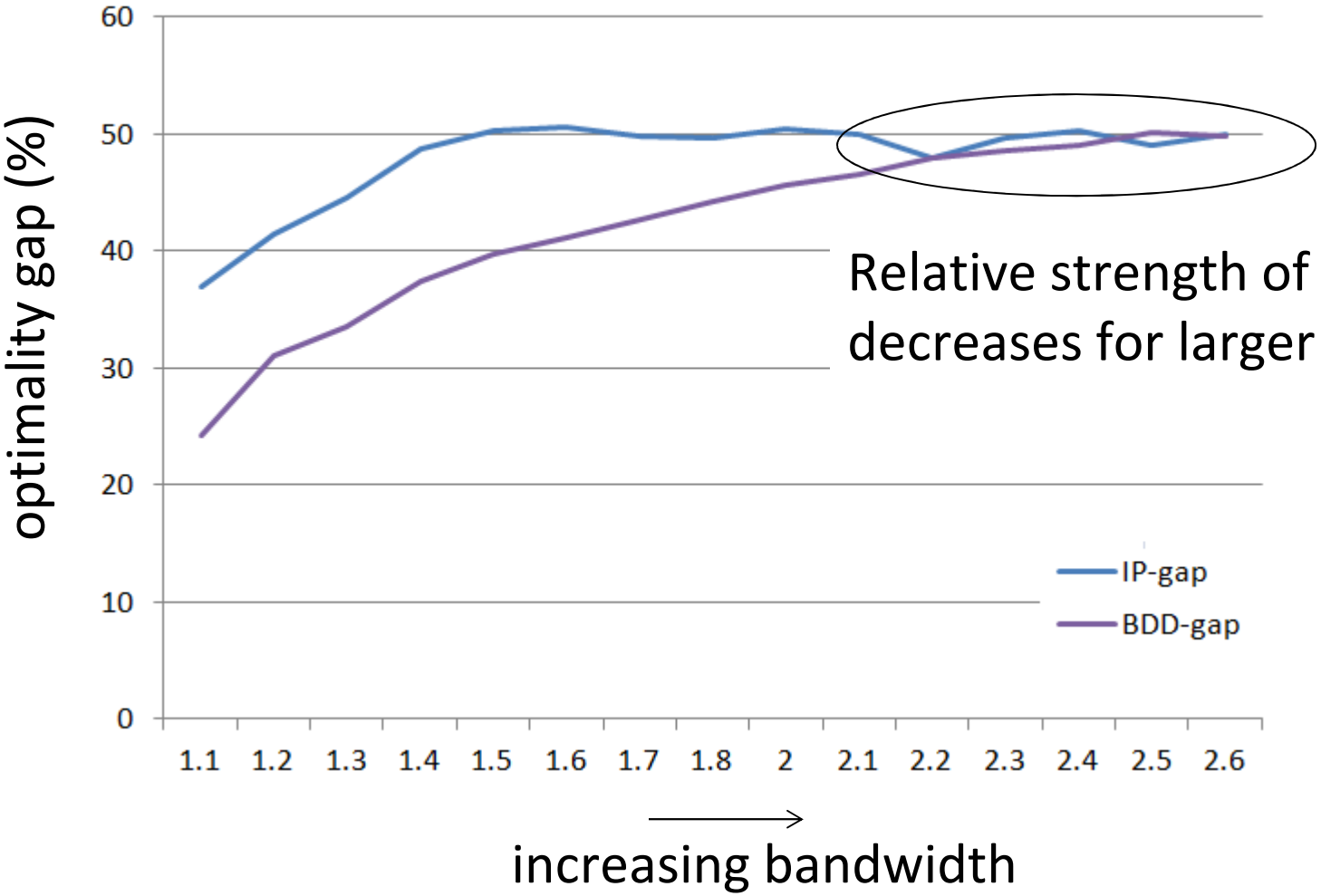
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Compare with Integer Programming (CPLEX)
  - Heuristic solution found at the root node
  - Also compute LP relaxation to measure the gap
- Experimental setup
  - As before, randomly generated with increasing bandwidth
  - n=500 variables, k=75 number of ones per constraints
  - bandwidth is multiple of k: $b \in \{1.1k, 1.2k, \ldots, 2.6k\}$
  - 30 instances per triple (n,k,b)
  - randomly generated weights
- Maximum BDD width 500
- Compare BDD-gap with IP-gap (relative to LP)

Unweighted instances



Weighted instances



Relative strength of BDD decreases for larger bandwidth

Summary for MDD-Optimization

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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
- Successfully applied to number of problems
  - Independent Set Problem, Set Covering Problem, Set Packing Problem, Bin Packing,…

Exercises

5. Consider the following CSP

$$4x_1 + 2x_2 + x_3 + x_4 + 2x_5 + 4x_6 = 7$$

$$x_1, x_2, \ldots, x_6 \in \{0,1\}$$

a) Draw an exact BDD for this problem using the variable ordering $x_1, x_2, x_3, x_4, x_5, x_6$

b) Draw an exact BDD for this problem using the variable ordering $x_1, x_6, x_2, x_5, x_3, x_4$

c) Which of the two orderings yields the smallest width?

Exercises

6. Consider the following set covering instance:

$$\begin{aligned} \text{minimize} \quad & 3x_1 + 2x_2 + x_3 + 4x_4 + 2x_5 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 1 \\ & x_1 + x_4 + x_5 \geq 1 \\ & x_2 + x_4 \geq 1 \end{aligned}$$

Construct a restricted BDD with maximum width 3. Does it yield the optimal solution?

Open issues

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Extend application to CP
  - Which other global constraints are suitable?
  - 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)

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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)

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- 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? (We have done this for disjunctive scheduling)
  - 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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

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 and Scheduling

- 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, applications, …