Decision Diagrams for Discrete Optimization

Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

based on joint work with

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

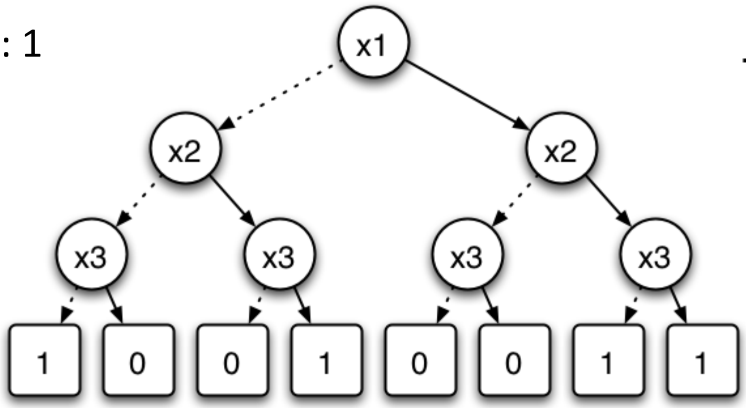
Outline

- Motivation and background
  - multi-valued decision diagrams (MDDs)
- Constraint Programming with MDDs
- MDDs as bounding mechanism
  - Relaxations
  - Restrictions
- Conclusions

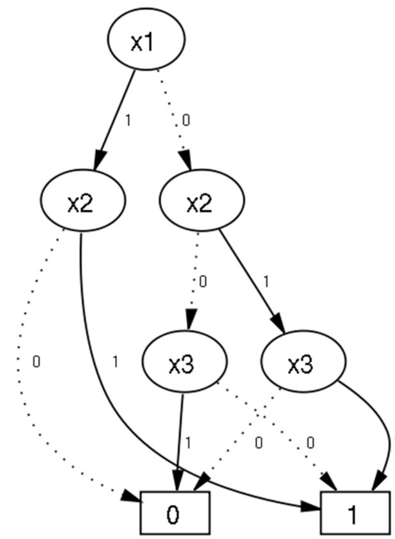
Decision Diagrams

---> : 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]
- Main operation: 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]
- Interesting variant
  - approximate MDDs

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

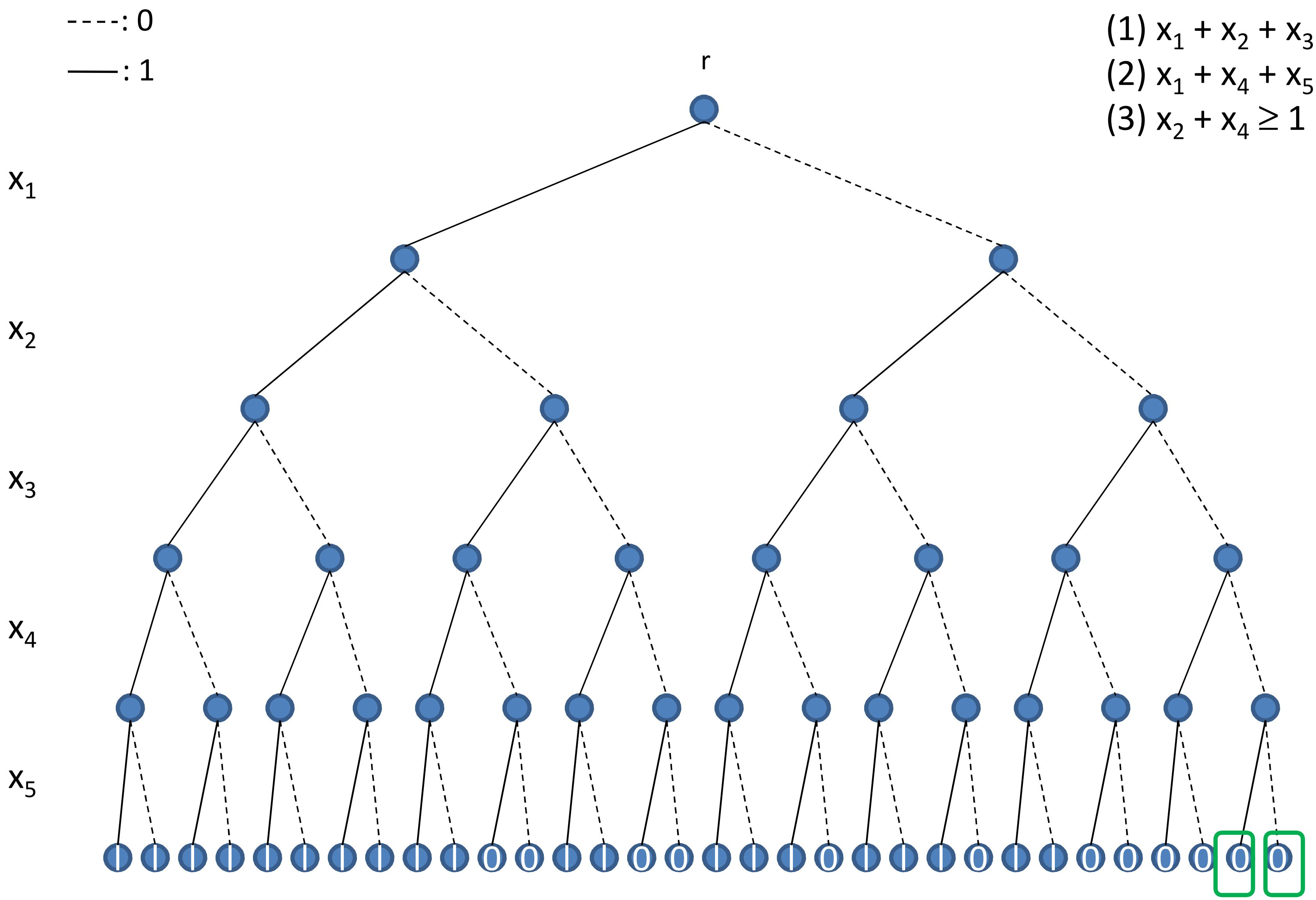
Exact MDDs for discrete optimization

Carnegie Mellon Tepper School of Business

(1) $x_1 + x_2 + x_3 \geq 1$

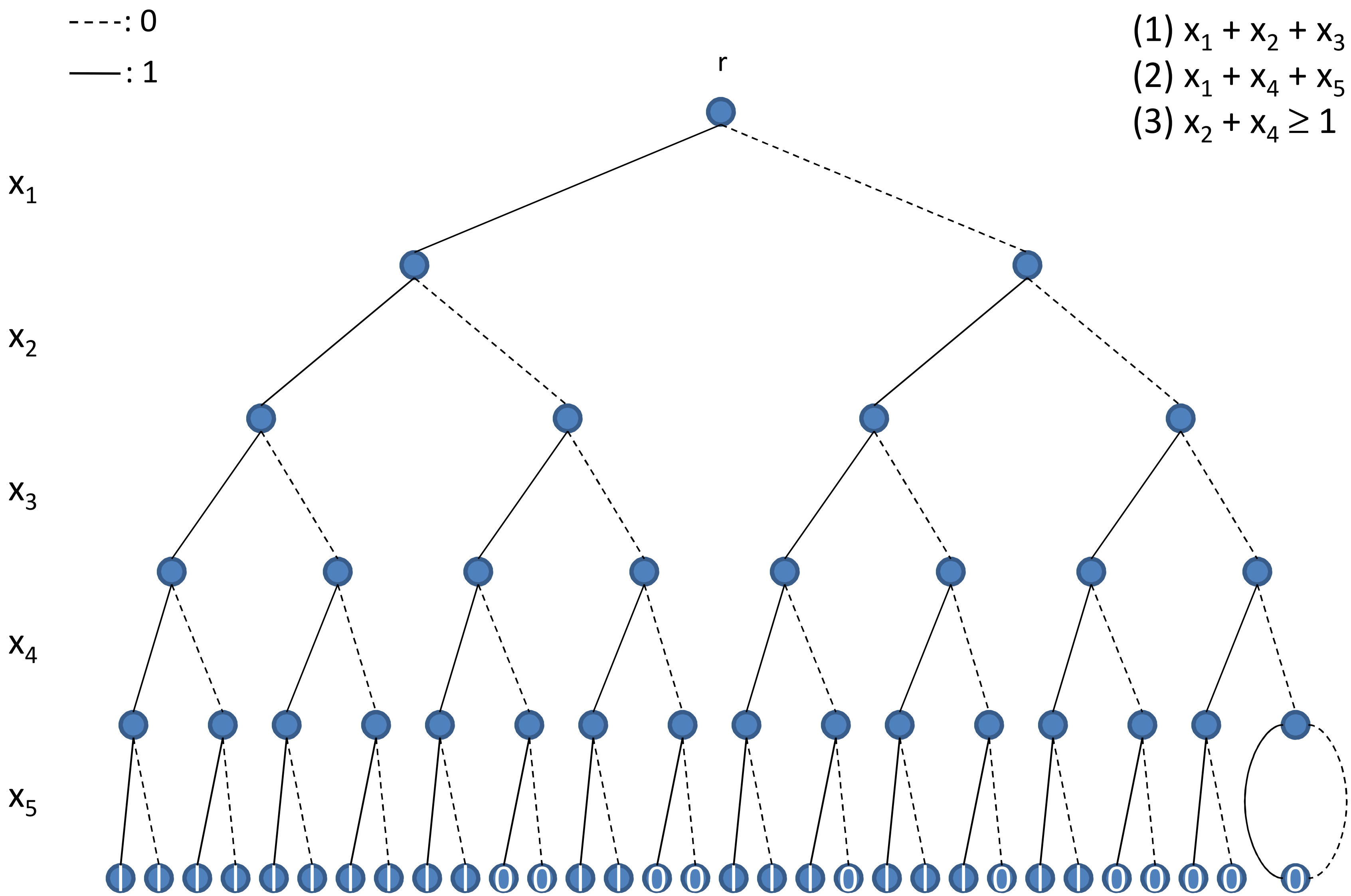
(2) $x_1 + x_4 + x_5 \geq 1$

(3) $x_2 + x_4 \geq 1$



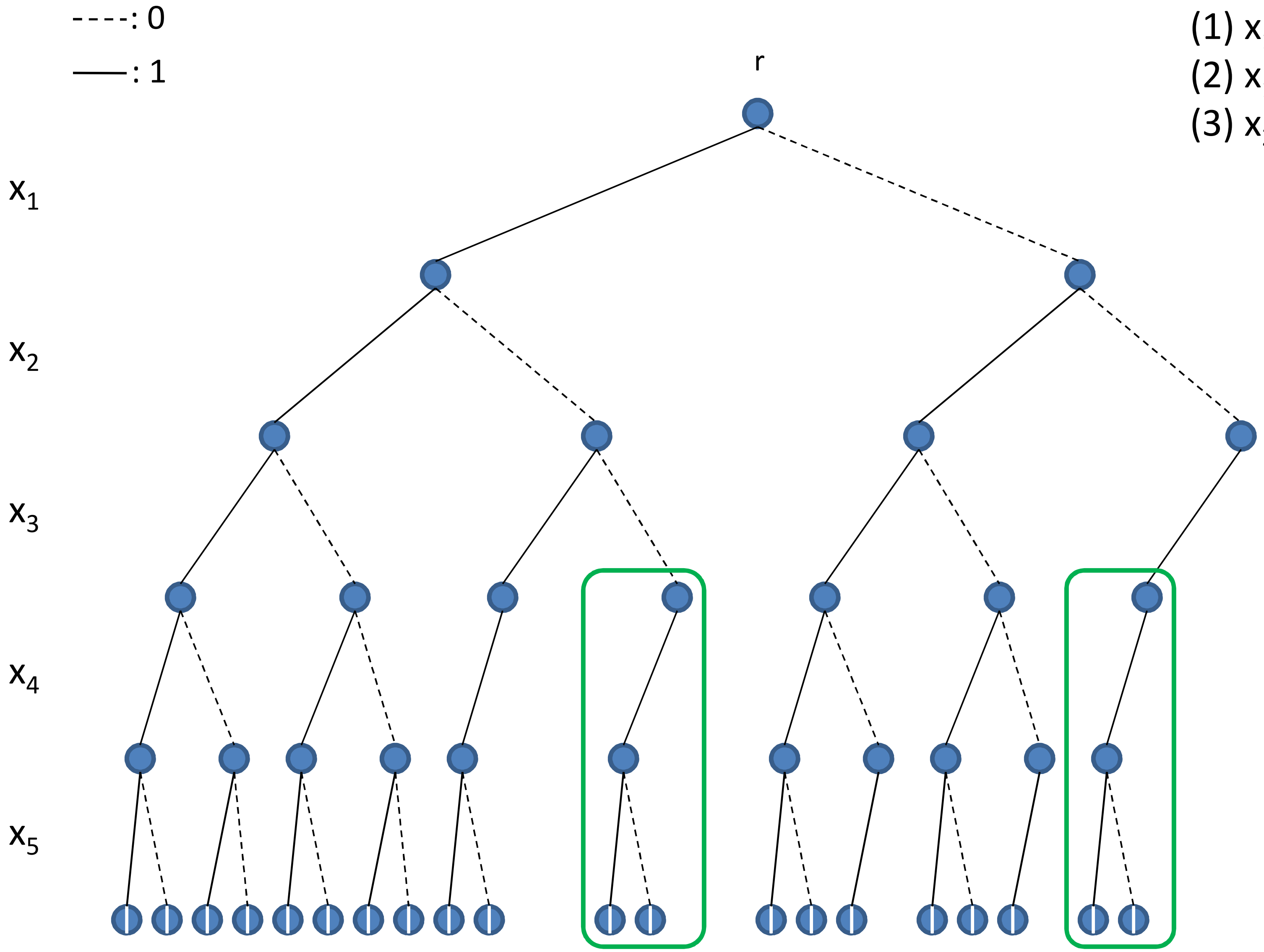
Exact MDDs for discrete optimization

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

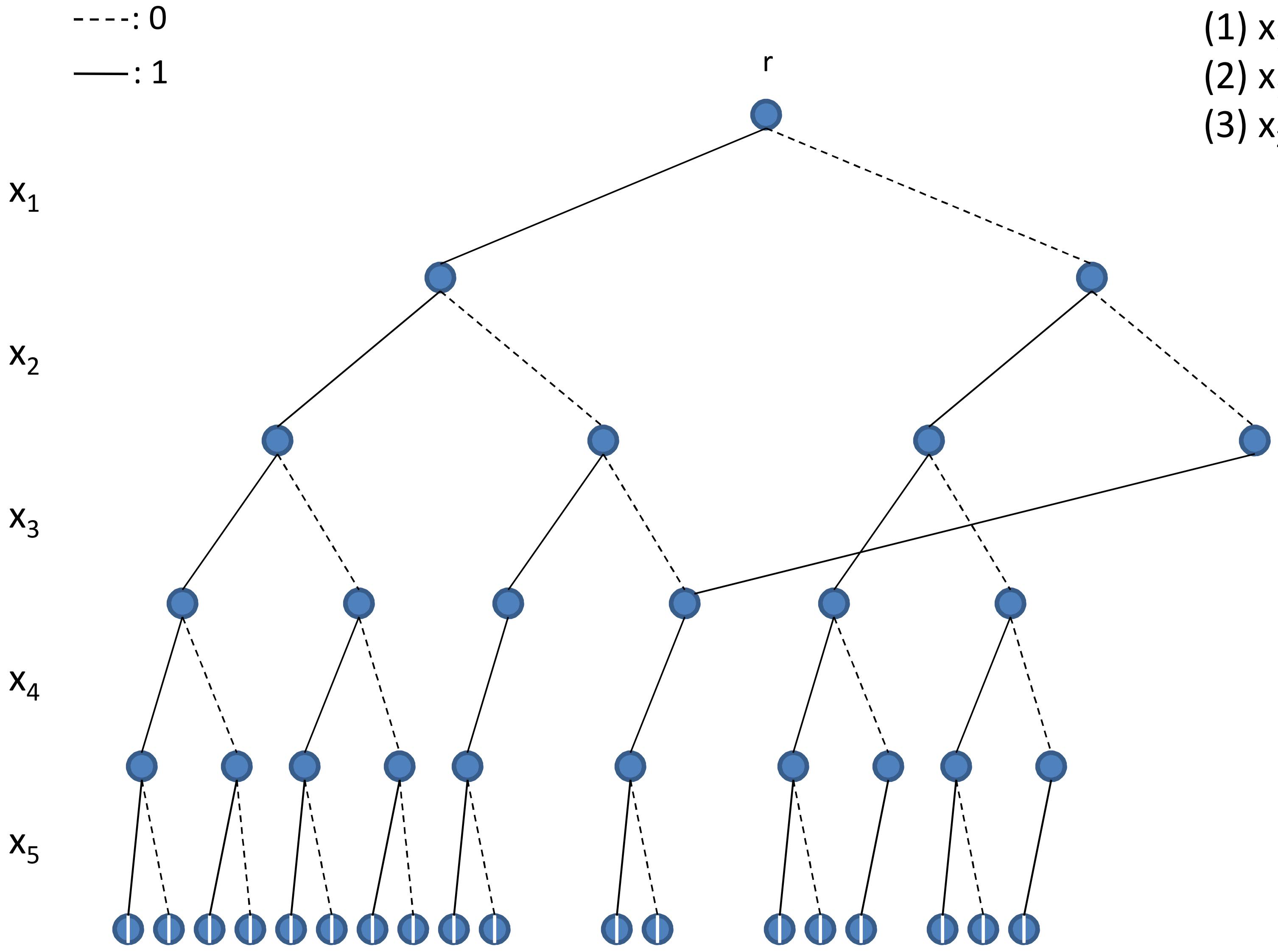


Exact MDDs for discrete optimization

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



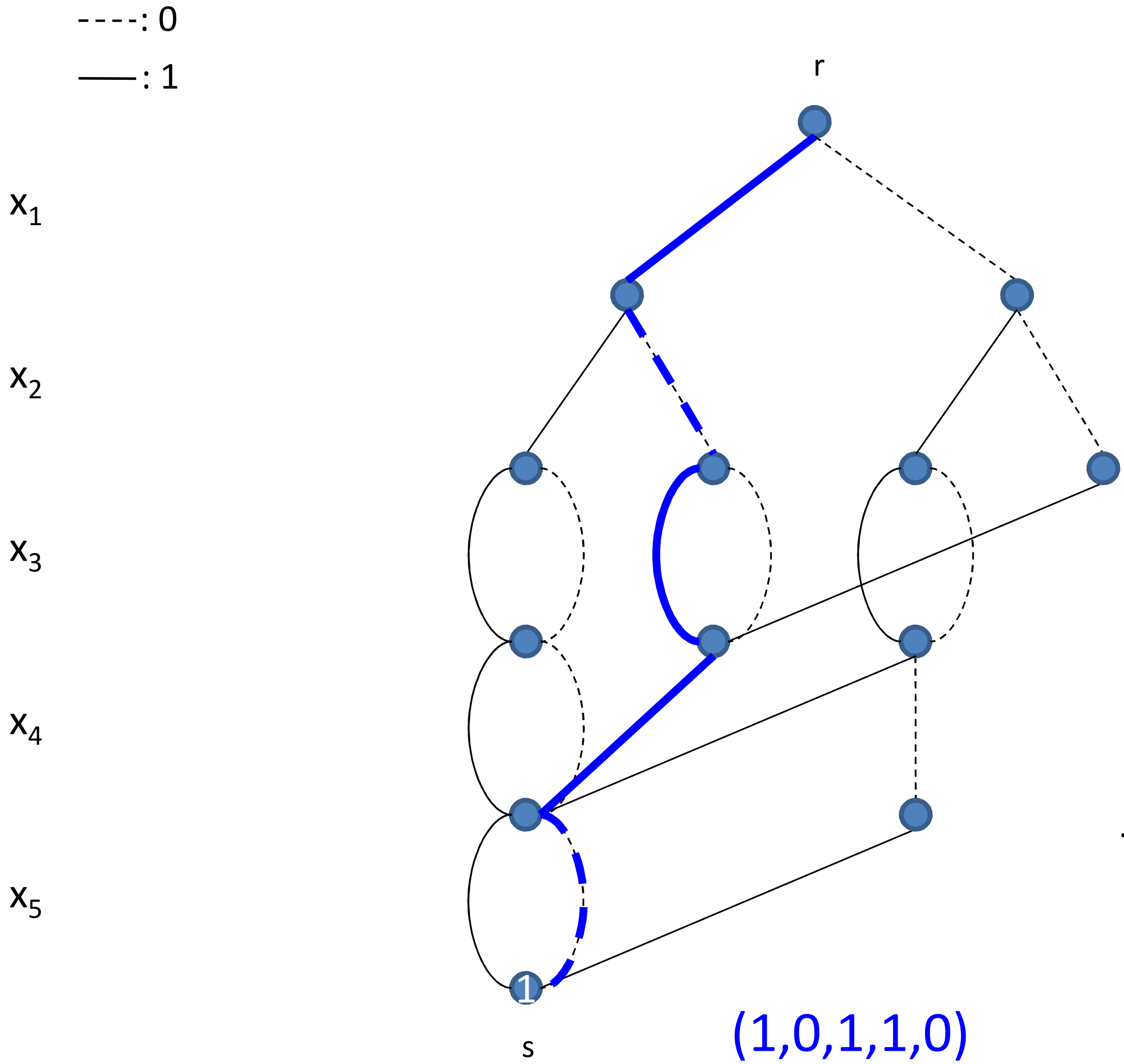
Exact MDDs for discrete optimization



(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) $x_2 + x_4 \geq 1$



(1,0,1,1,0)

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)

- This talk: applications to CP and IP

MDDs for Constraint Programming

Hoda, v.H., and Hooker. A Systematic Approach to MDD-Based Constraint Programming. In *Proceedings of CP*. LNCS 6308, pp. 266-280. Springer, 2010.

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

$x_1 > x_2$

$x_1 + x_2 = x_3$

alldifferent(x_1, x_2, x_3, x_4)

$x_1 \in \{2\}$, $x_2 \in \{1\}$, $x_3 \in \{3\}$, $x_4 \in \{0\}$

Drawback of domain propagation

Observations:

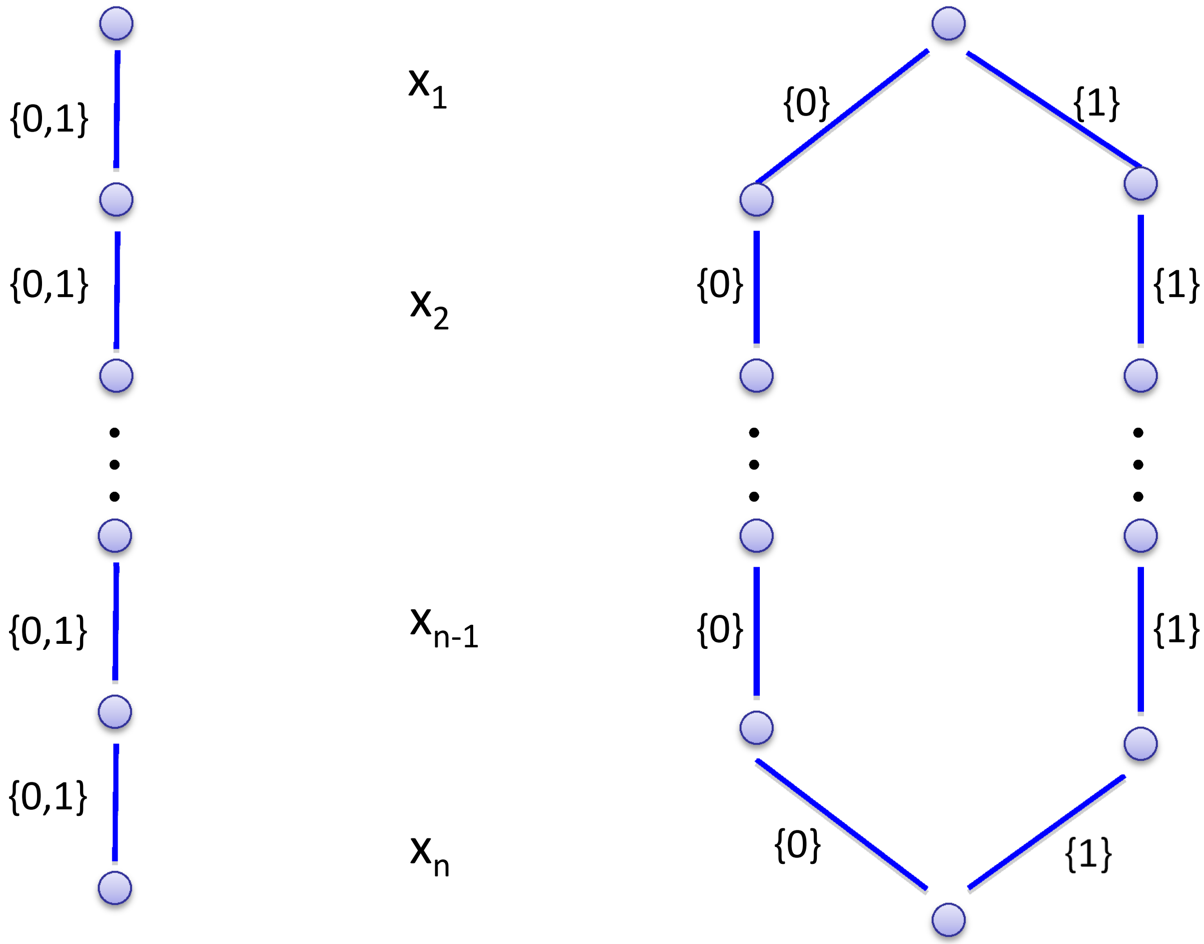
- Communication between constraints only via variable domains
- Information can only be expressed as a domain change
- Other (structural) information that may be learned from a constraint is lost: it must be projected onto variable domains
- Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation)

This drawback can be addressed by communicating more expressive information, using MDDs [Andersen et al. 2007]

- Explicit representation of more refined potential solution space

Illustrative Example

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



domain representation, size 2^n

MDD representation, size 2

MDD-based constraint programming

- Maintain limited-width MDD
  - Serves as relaxation
  - Typically start with width 1 (initial variable domains)
  - Dynamically adjust MDD, based on constraints
- Constraint Propagation
  - Edge filtering: Remove provably inconsistent edges (those that do not participate in any solution)
  - Node refinement: Split nodes to separate edge information
- Search
  - As in classical CP, but may now be guided by MDD

Specific MDD propagation algorithms

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]
- Sequential scheduling constraints [Hoda et al., 2010] [Cire & v.H., 2011]
- *Sequence* constraints (combination of *Amongs*) [v.H., 2011]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]

Among constraints

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

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

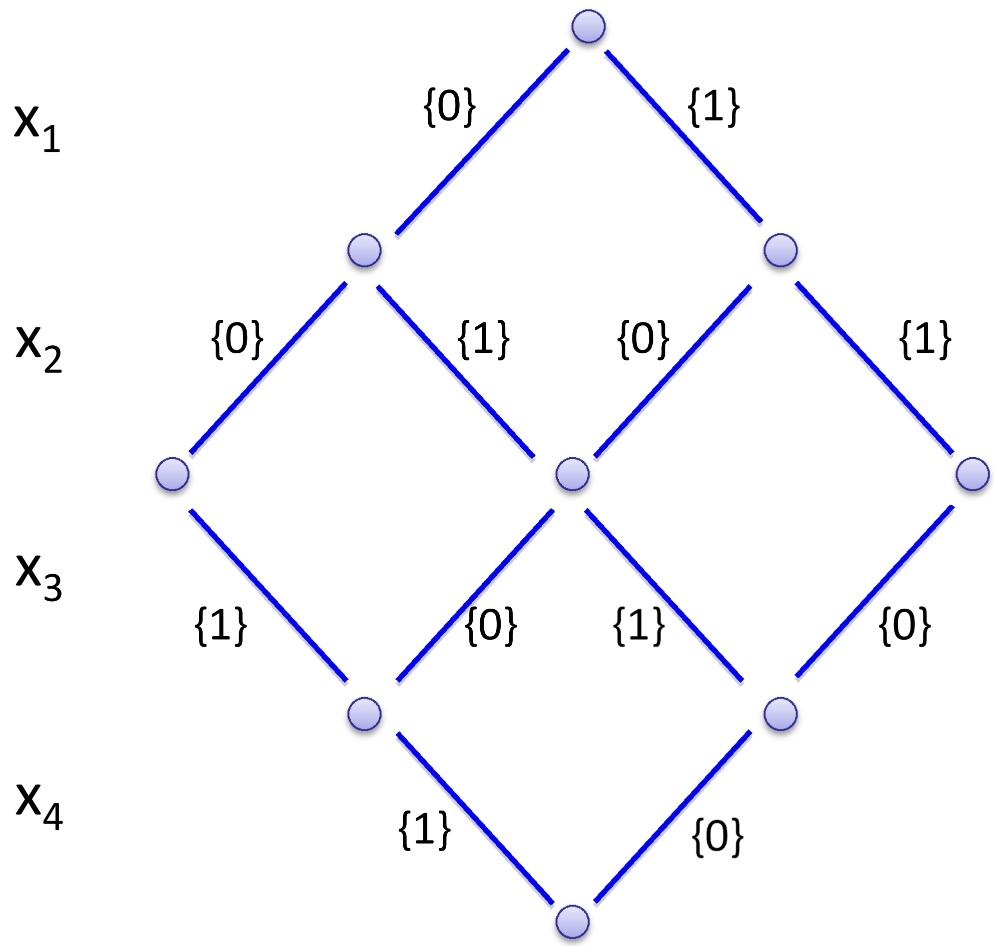
  "among the variables in X, at least l and at most u take a value from the set S"

- <u>Example:</u> X represents 7-day shift schedule for an employee that must work either 1 or 2 night shifts:

  Among(X, {night}, 1, 2)

- (WLOG assume that X are binary and S = {1})

Example: MDD for Among



Size $O(n(u - l))$

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

MDD Filtering for Among

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

(establish "MDD-consistency")

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

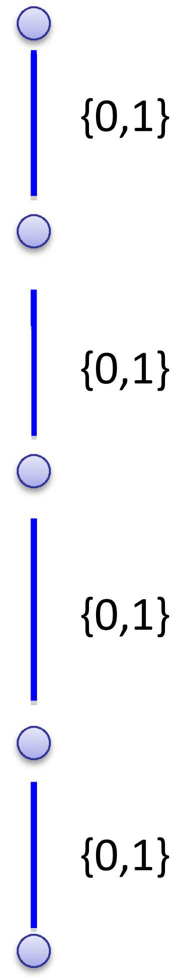
For each layer in MDD, we first apply edge filter, and then try to refine

- consider incoming edges for each node
- split the node if there exist incoming edges that are not equivalent (w.r.t. path length)

Example:

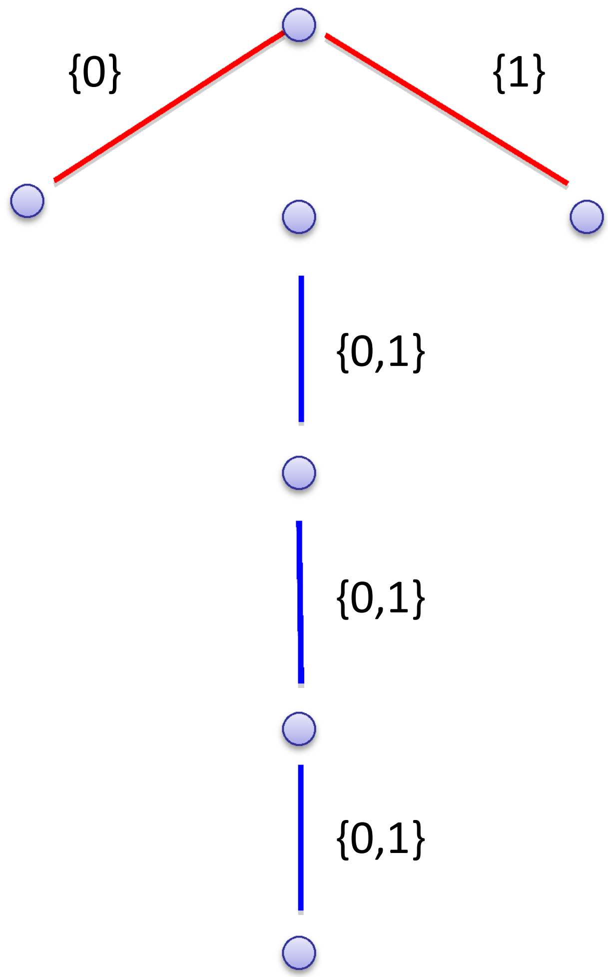
- We will propagate *Among*($\{x_1,x_2,x_3,x_4\}$,{1},2,2) through a BDD of maximum width 3

Example



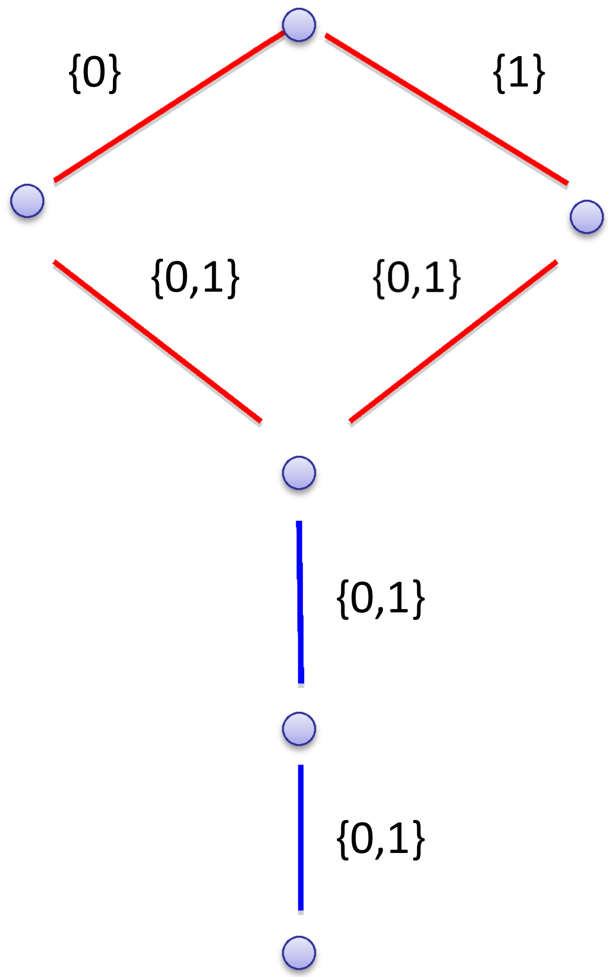
Among$(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example



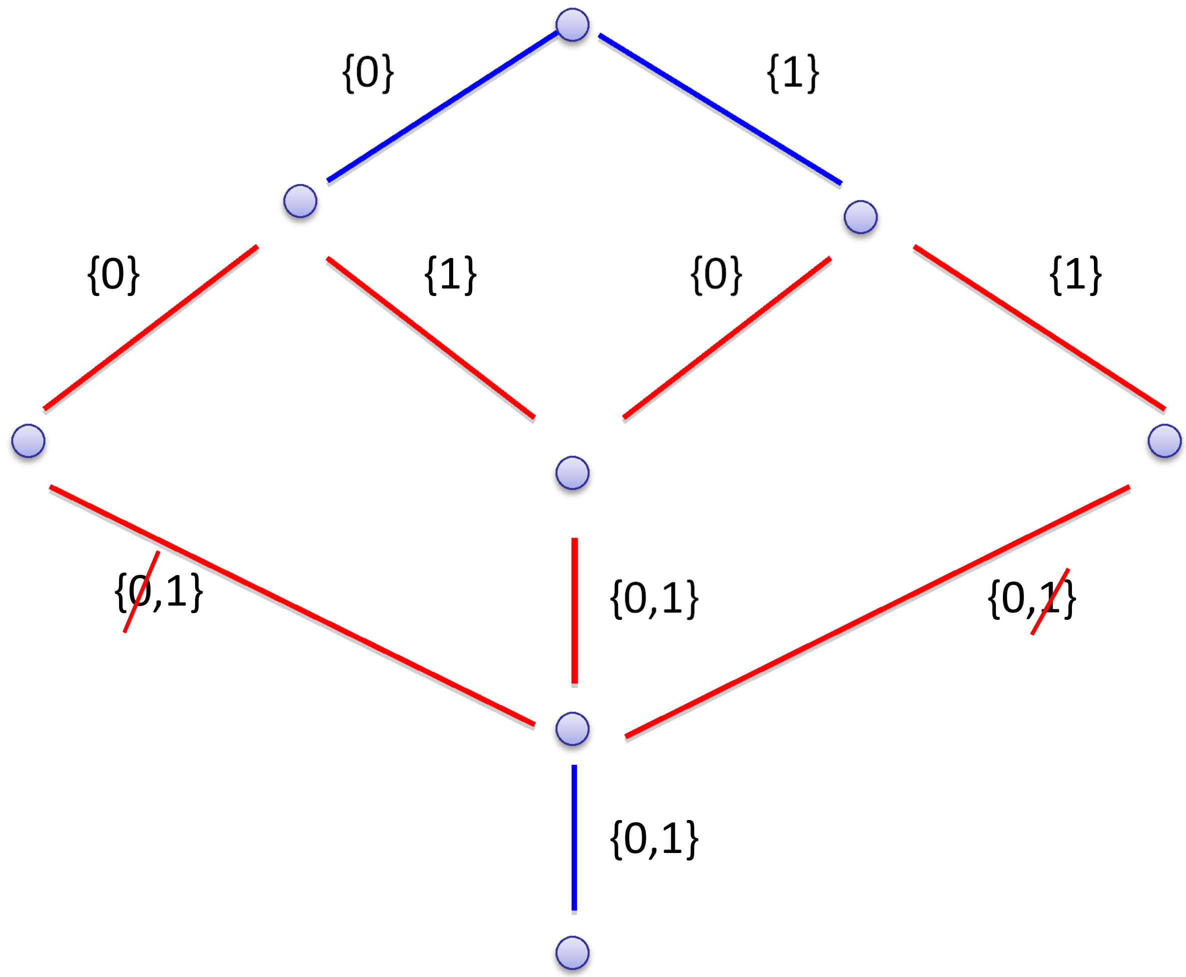
$Among(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example



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

Example



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

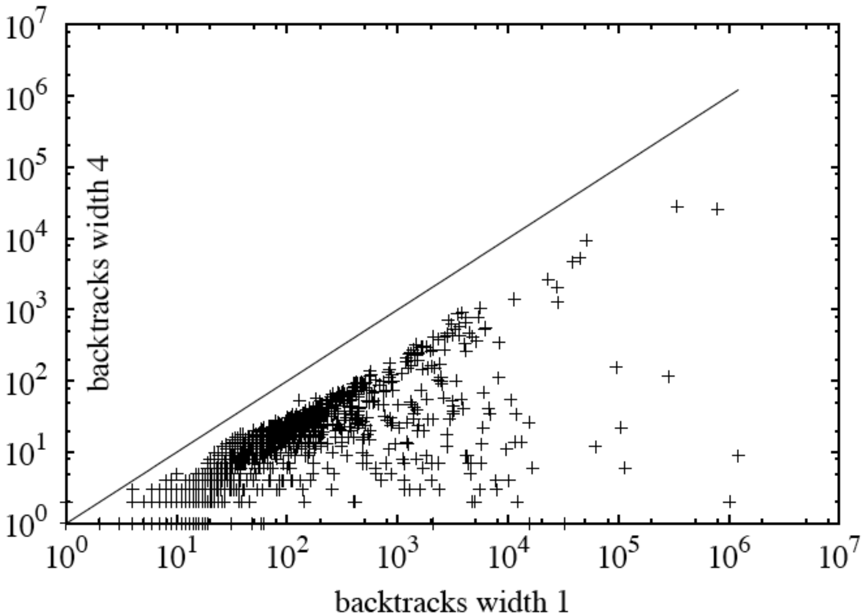
Experiments

- 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
  - Classes: 5 to 200 among constraints (step 5), 100 instances per class

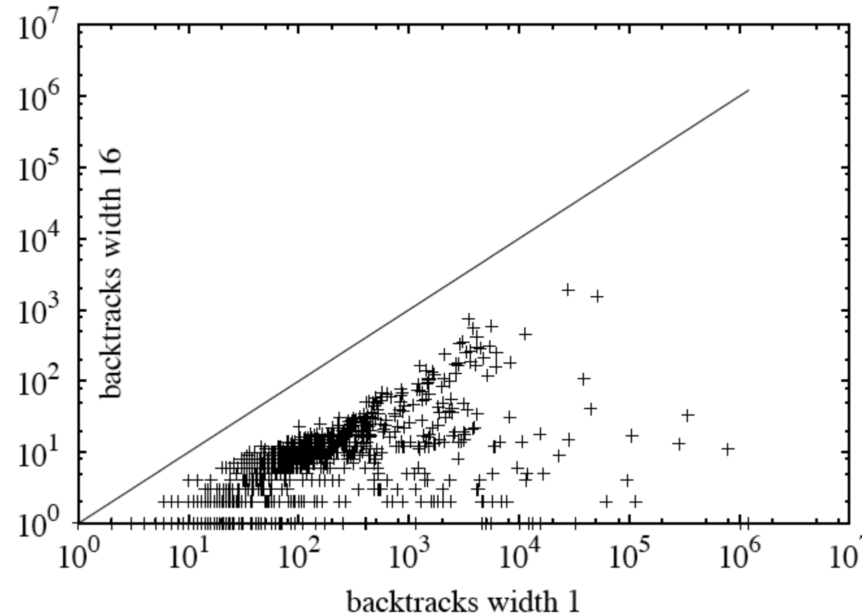
- Nurse rostering instances (horizon *n* days)
  - Work 4-5 days per week
  - Max A days every B days
  - Min C days every D days
  - Three problem classes

- Compare width 1 (traditional domains) with increasing widths

Multiple Amongs: Search tree size

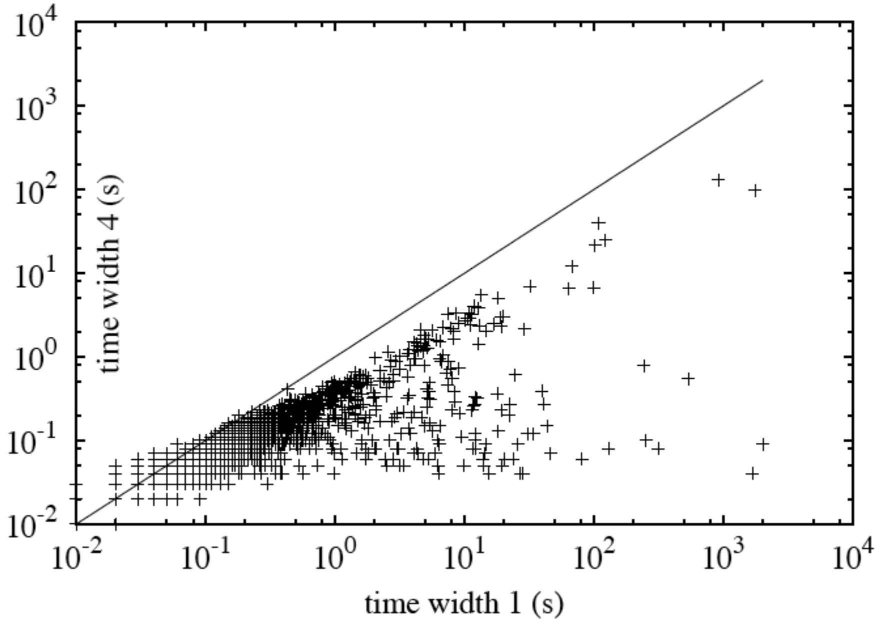


width 1 vs 4

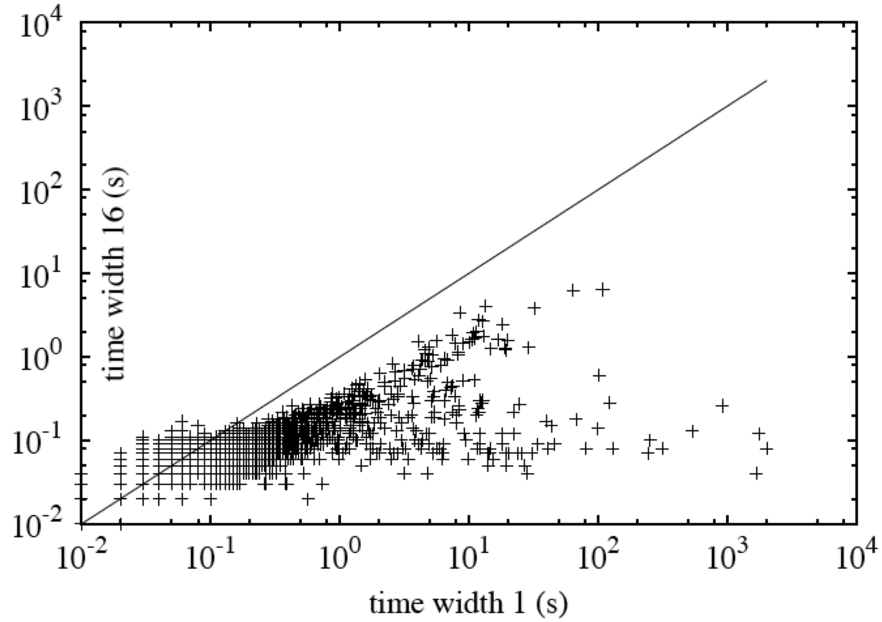


width 1 vs 16

Multiple Amongs: Running Time



width 1 vs 4



width 1 vs 16

Nurse rostering problems

		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

Summary for MDD-based CP

Carnegie Mellon Tepper SCHOOL OF BUSINESS

- MDDs provide substantial advantage over traditional domains for filtering multiple *Among* constraints
  - Strength of MDD can be controlled by the width
  - Wider MDDs yield greater speedups
  - Huge reduction in the amount of backtracking and solution time

- Intensive processing at search nodes can pay off when more structural information is communicated between constraints

Relaxation MDDs

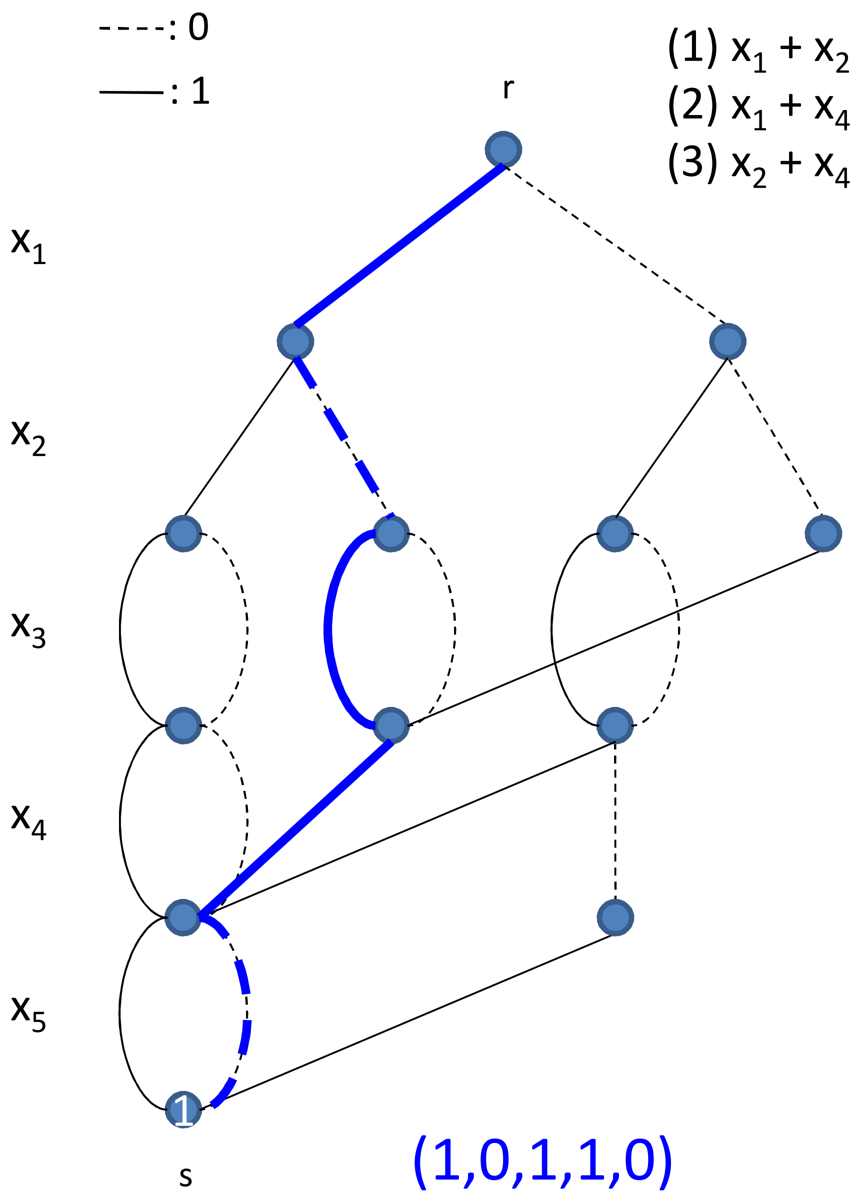
Bergman, v.H., and Hooker. Manipulating MDD Relaxations for Combinatorial Optimization. In *Proceedings of CPAIOR*, LNCS 6697, pp. 20-35. Springer, 2011.

Bergman, Cire, v.H., and Hooker. Bounds for Discrete Optimization Problems via Multivalued Decision Diagrams. Working paper, 2011.

Motivation and outline

- 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

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

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

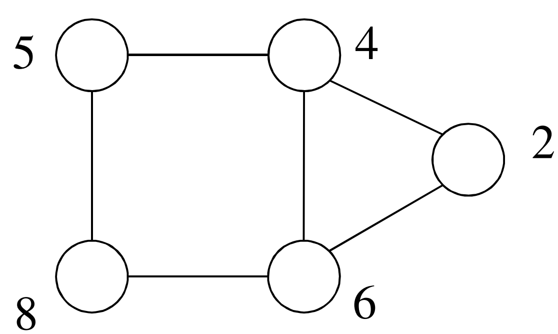
- Construct the relaxation MDD using a top-down compilation method
- Find shortest path $\rightarrow$ provides bound B
- Extension to an exact method
  1. Isolate all paths of length B, and verify if any of these paths is feasible*
  2. if not feasible, set B := B + 1 and go to 1
  3. otherwise, we found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

* Feasibility can be checked using MDD-based CP

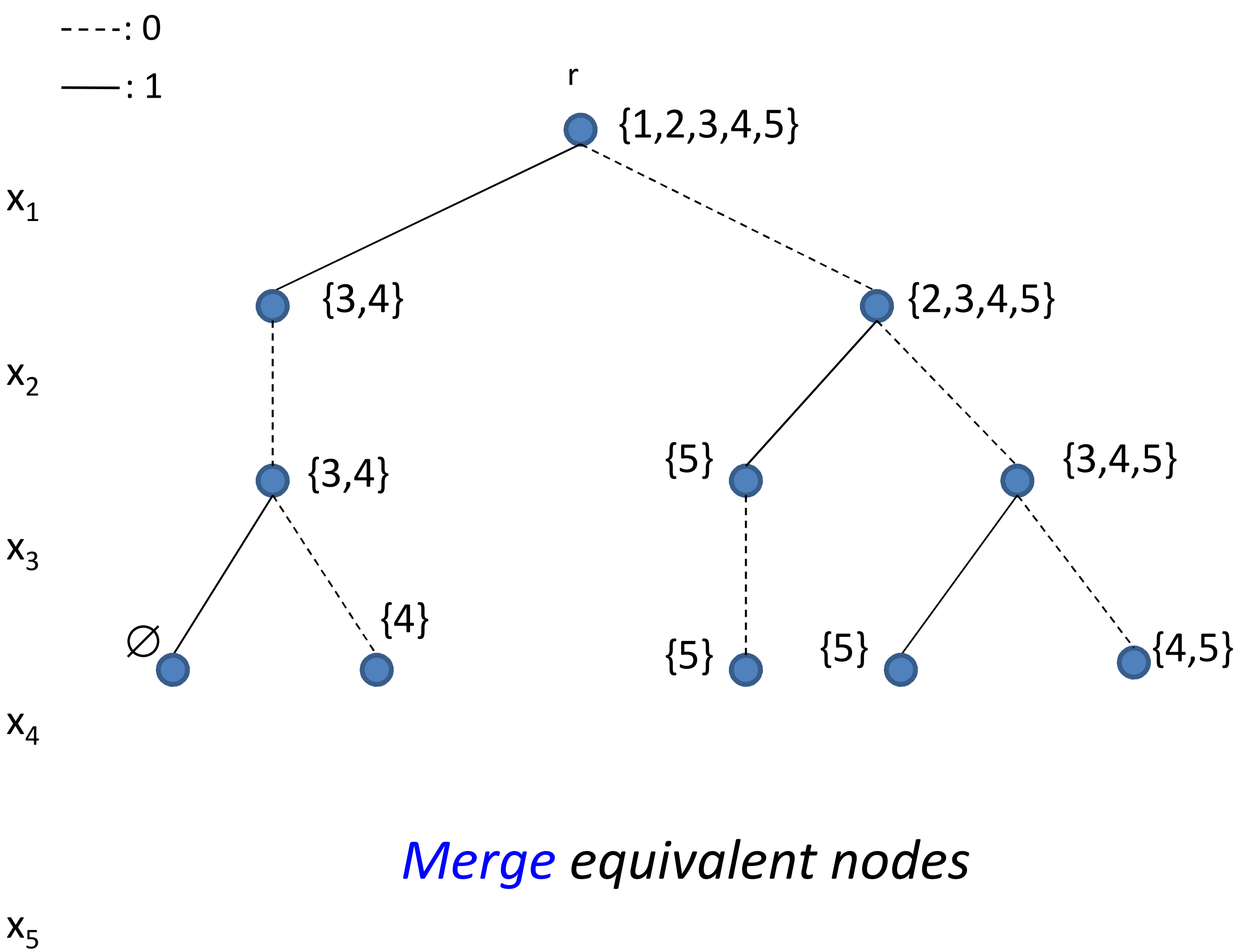
Case Study: Independent Set Problem

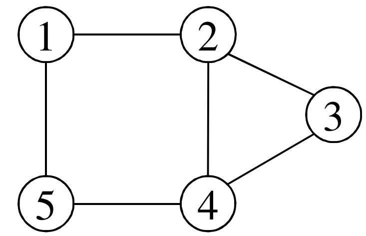
- Given graph G = (V, E) with vertex weights w_i
- Find a subset of vertices S with maximum total weight such that no edge exists between any two vertices in S

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

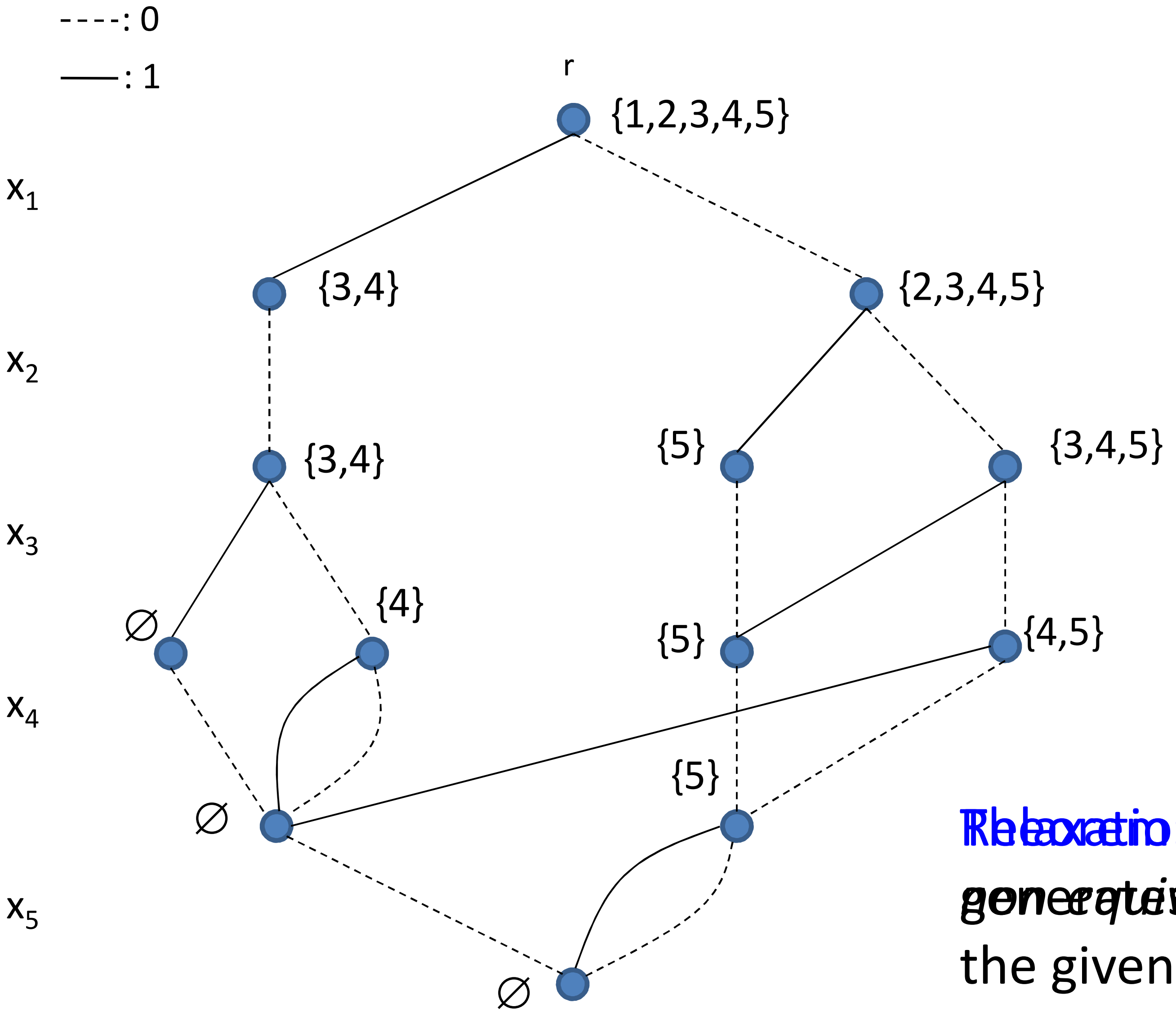


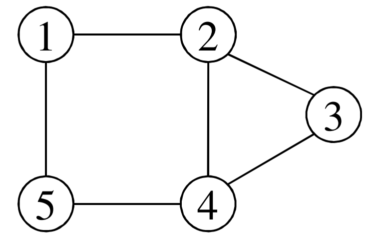


Merge equivalent nodes

{vertices that can still be included}

Node Merging

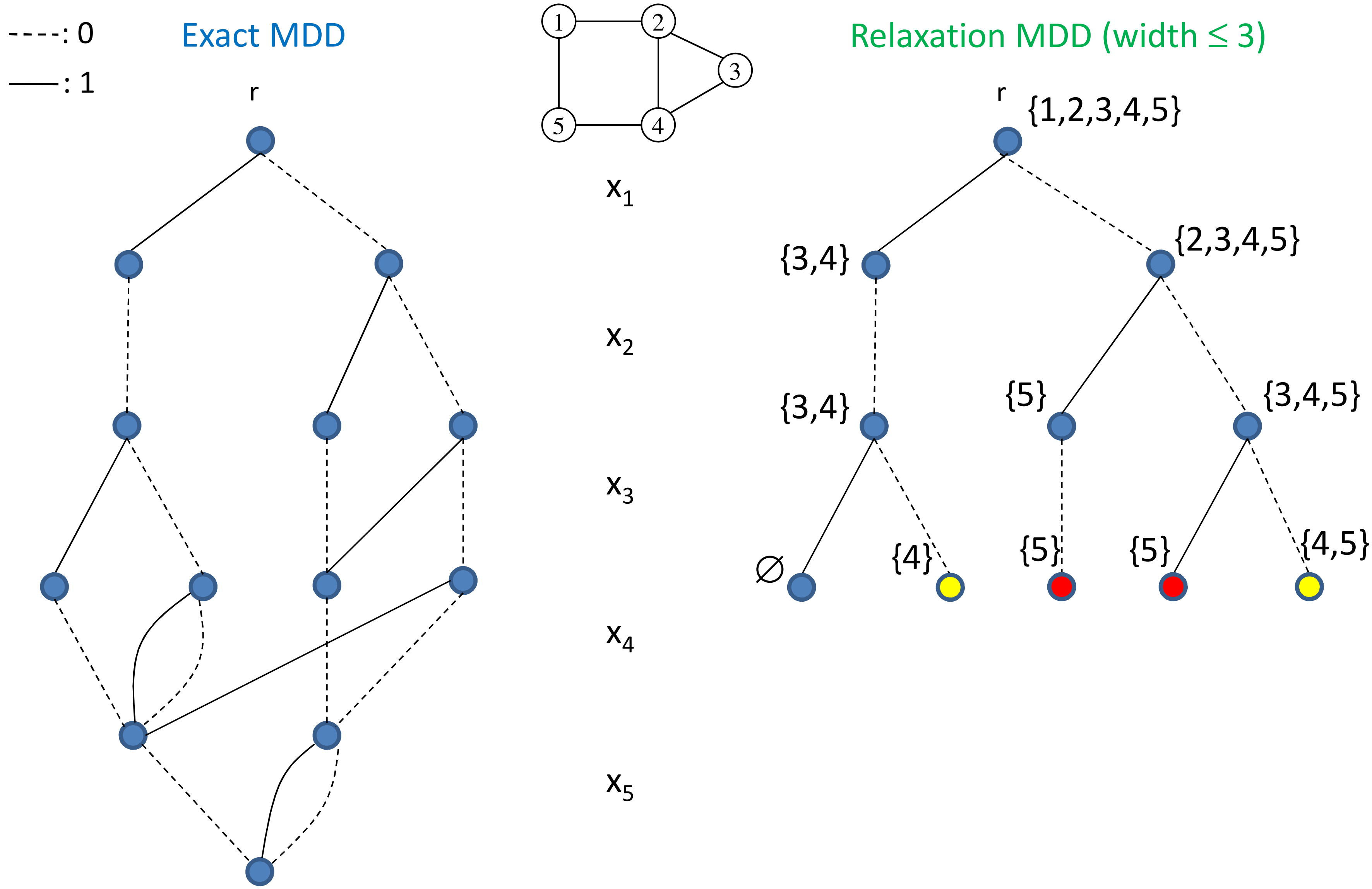




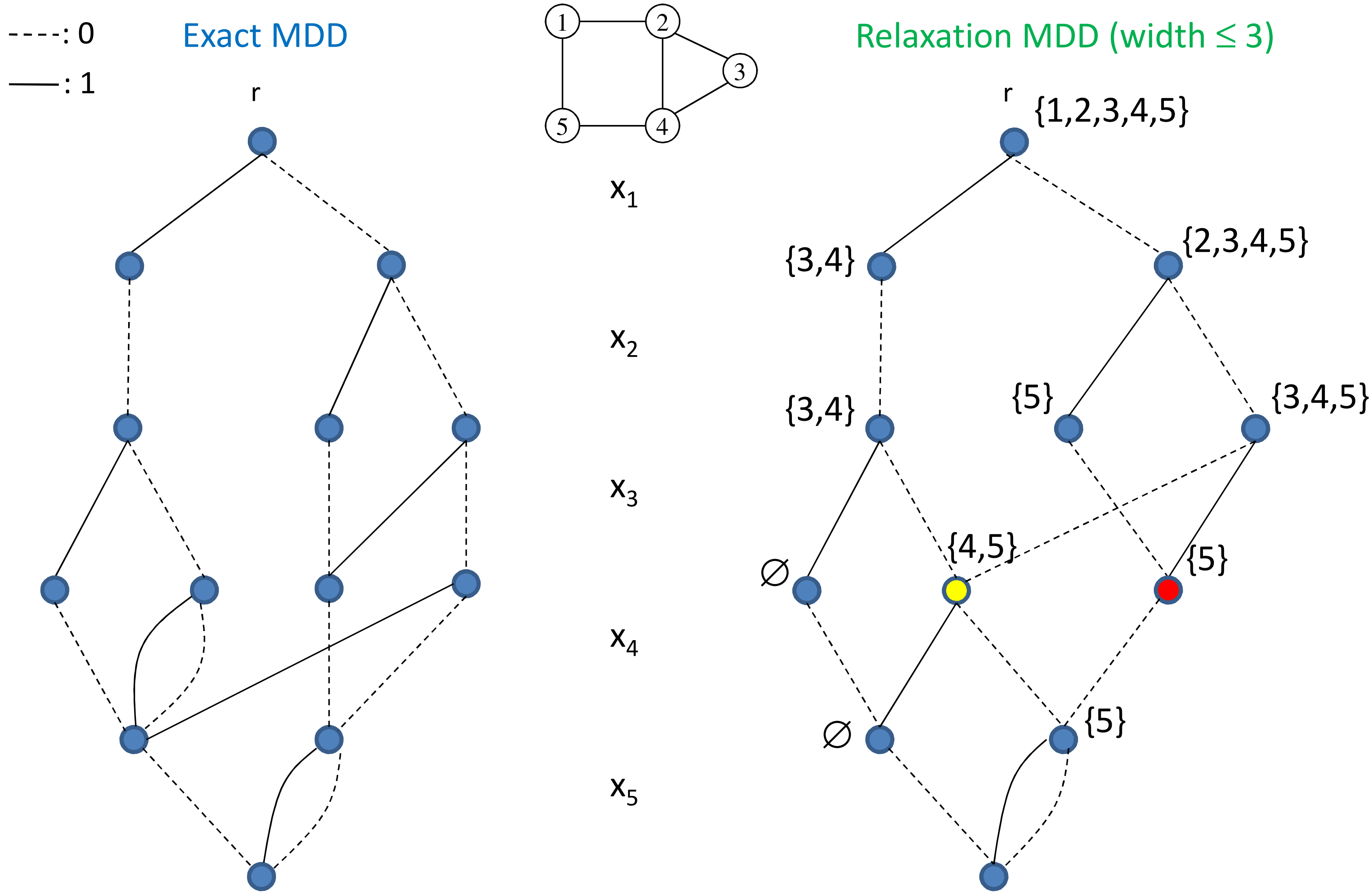
This procedure generates an exact MDD

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

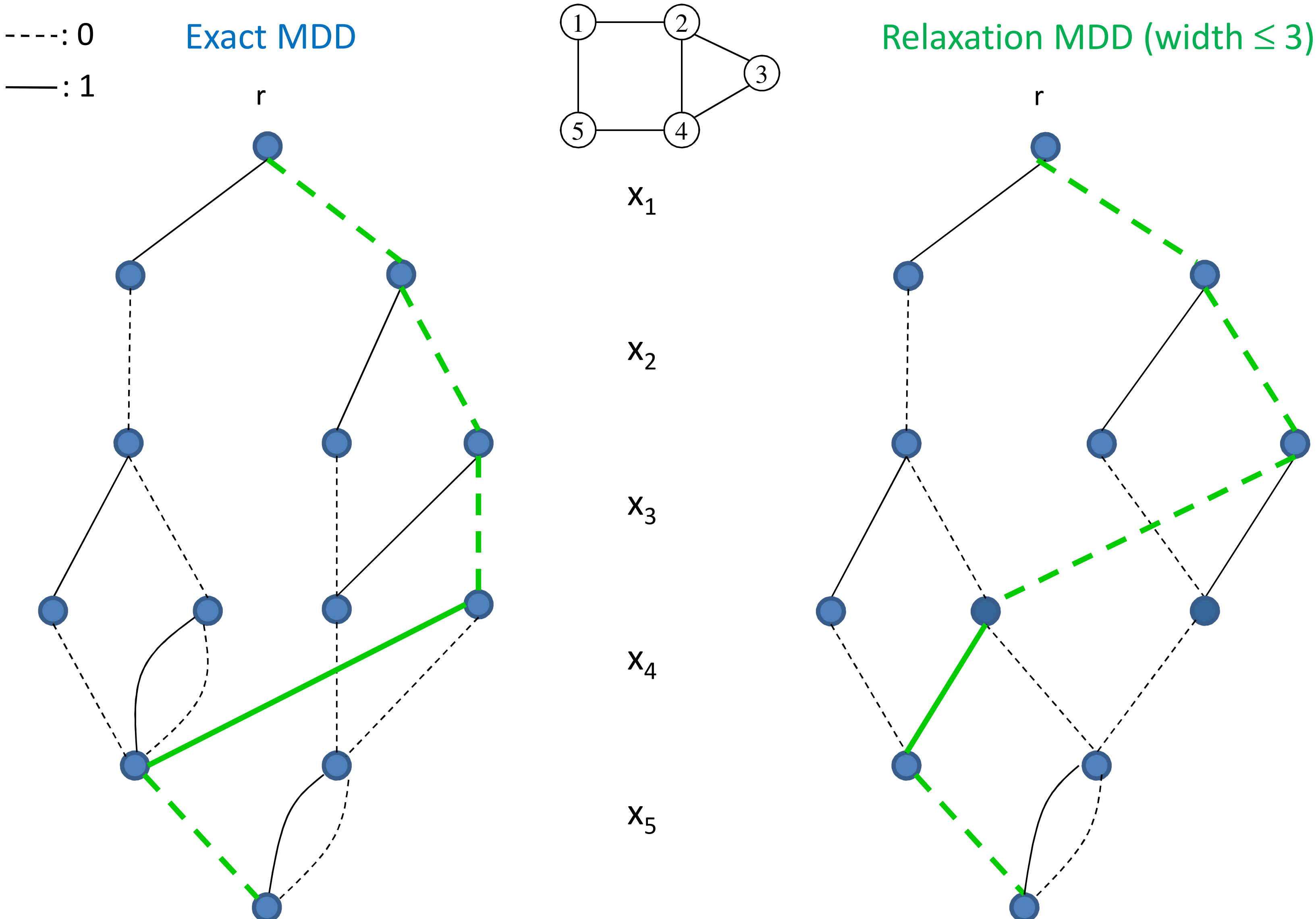
Relaxation MDD



Relaxation MDD

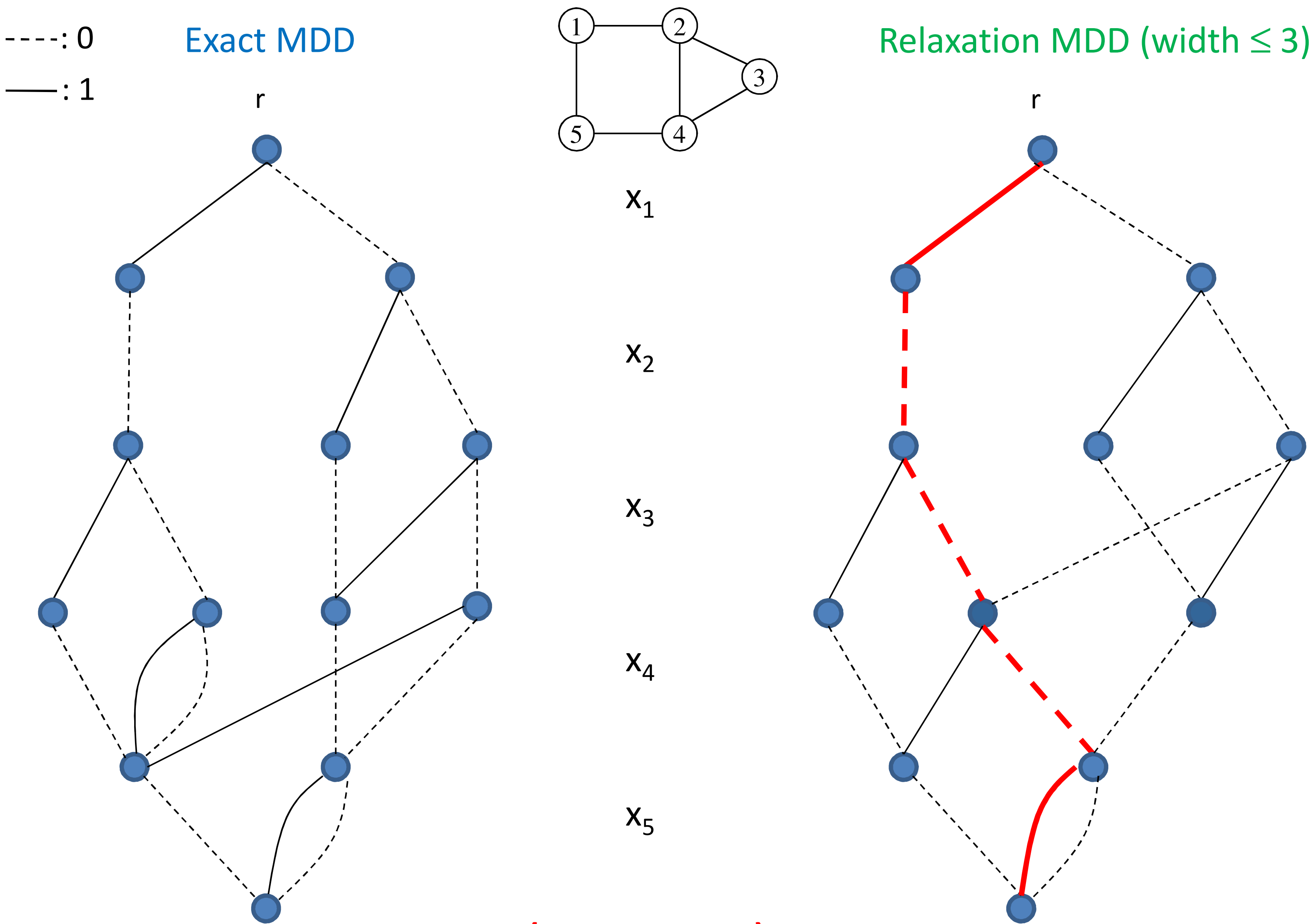


Relaxation MDD



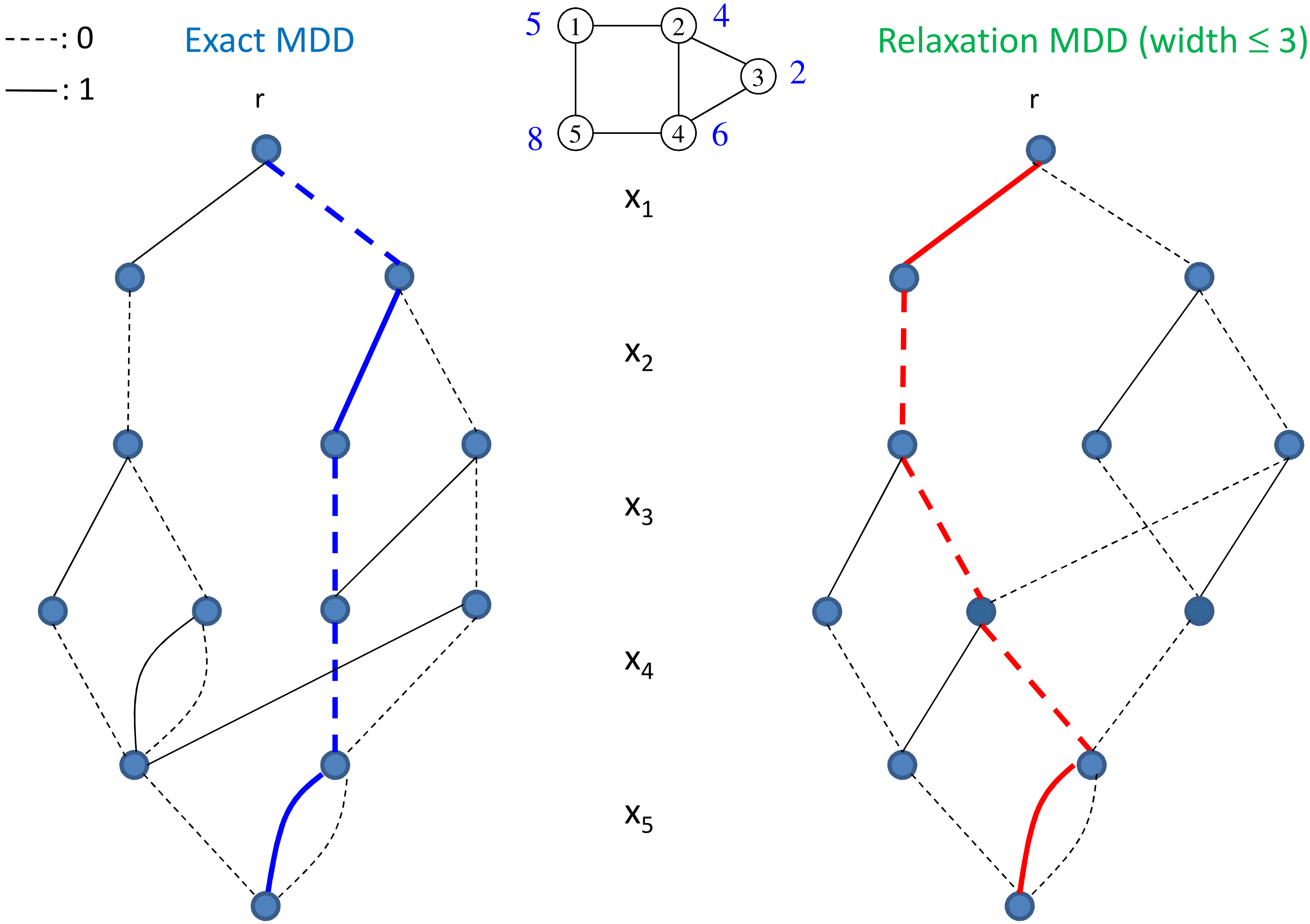
(0,0,0,1,0)

Relaxation MDD



(1,0,0,0,1)

Evaluate Objective Function



max f(x) = 12

max f(x) = 13

Tightening the Upper Bound

- Value extraction method
  - Given: an MDD relaxation, M
  - Given: a valid upper 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
  - Decrease v to v-1 (if no solution exists)
  - Find a feasible (and optimal) solution

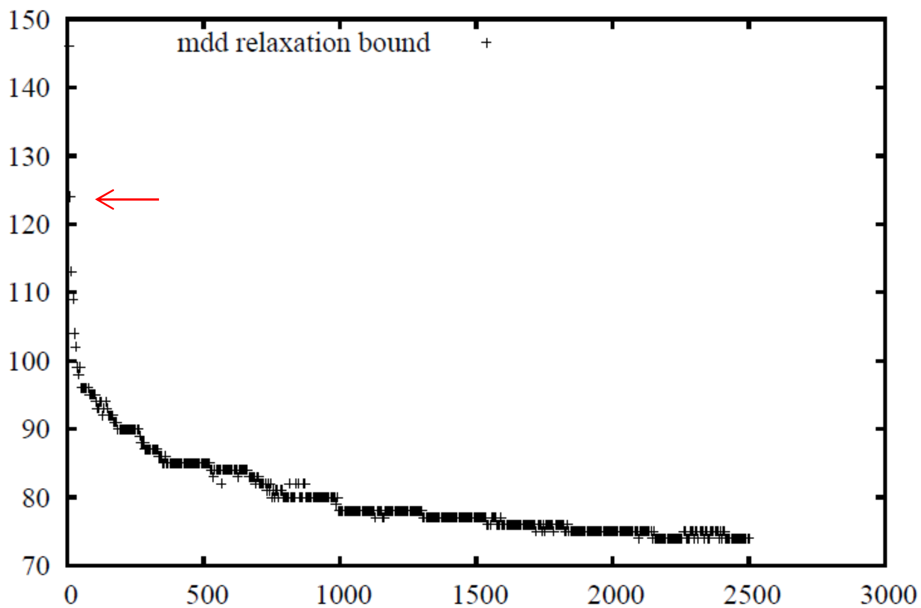
Experimental Results

- Impact of maximum width on strength of bound (and running time)
- Evaluate value extraction method
- Compare MDD bounds to LP bounds

- DIMACS clique instances (unweighted graphs)

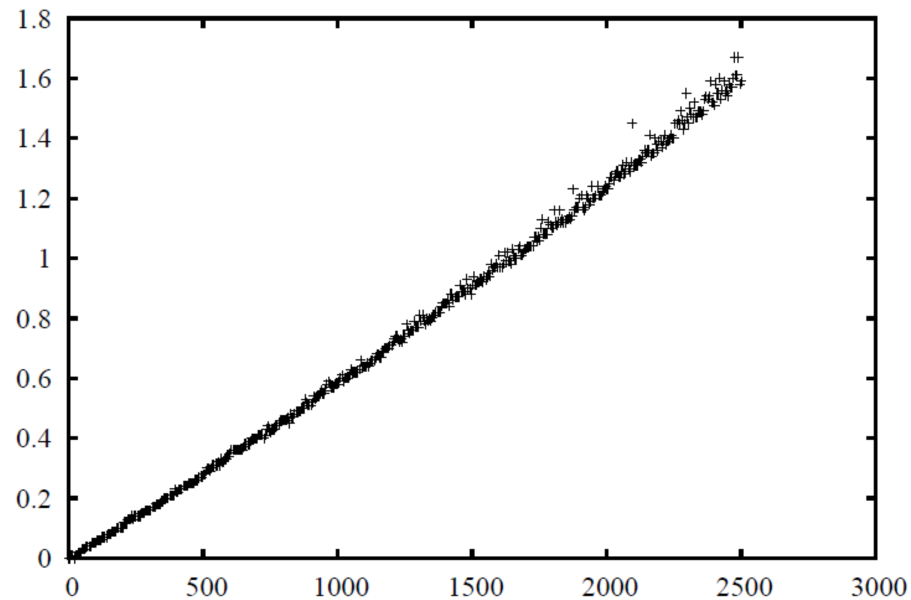
Impact of width on relaxation

upper bound



maximum width

time (s)



maximum width

brock_200-2 instance

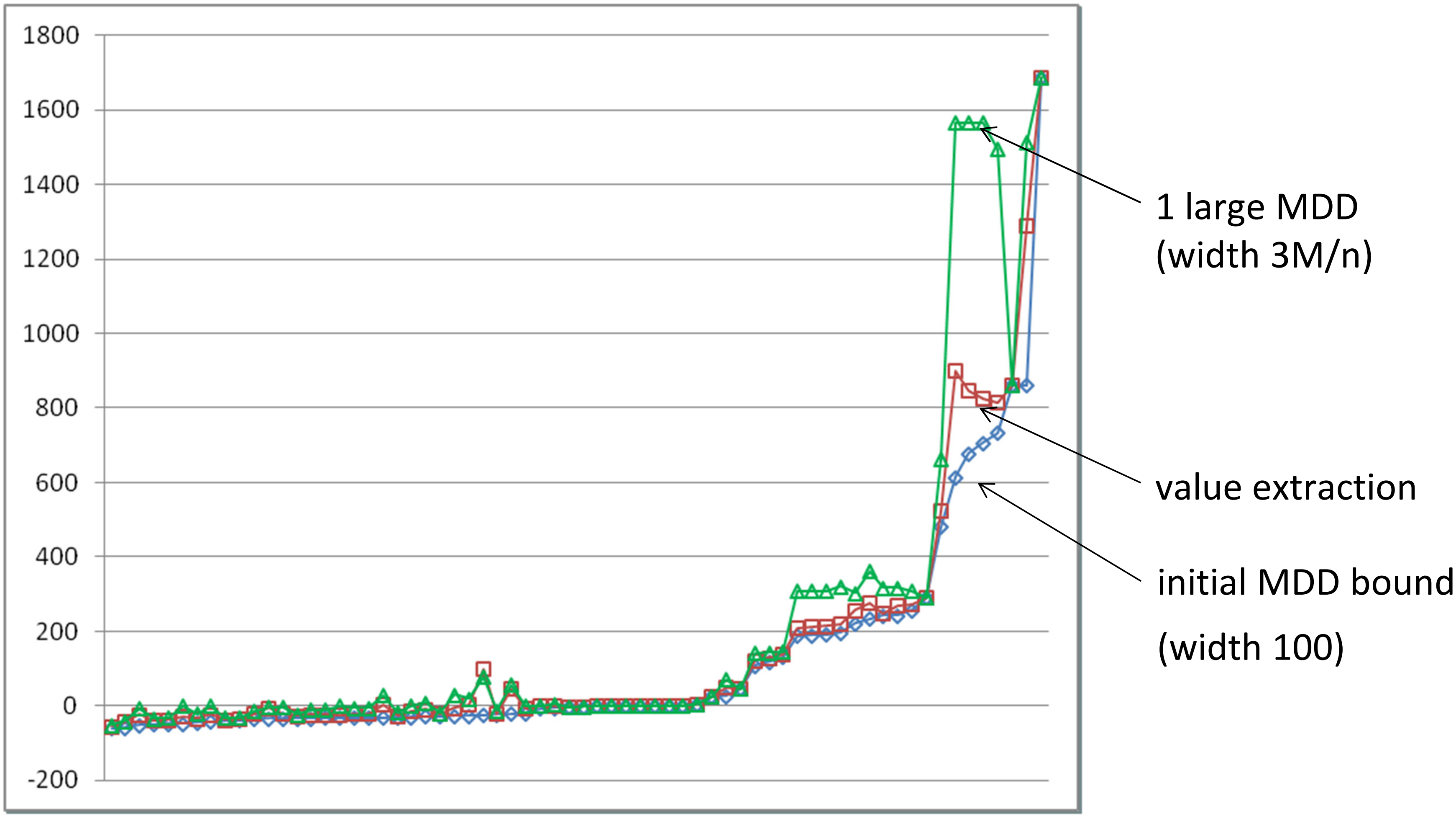
Compare MDD and LP bounds

- CPLEX root node relaxation
  - no primal heuristics, no presolve
  - maximum 5 minutes
- MDD bound – version 1
  - maximum width 100
  - apply value extraction for the same time as CPLEX
- MDD bound – version 2
  - maximum width 3,000,000/*n* (fill memory)
  - no value extraction

MDD versus LP (CPLEX)

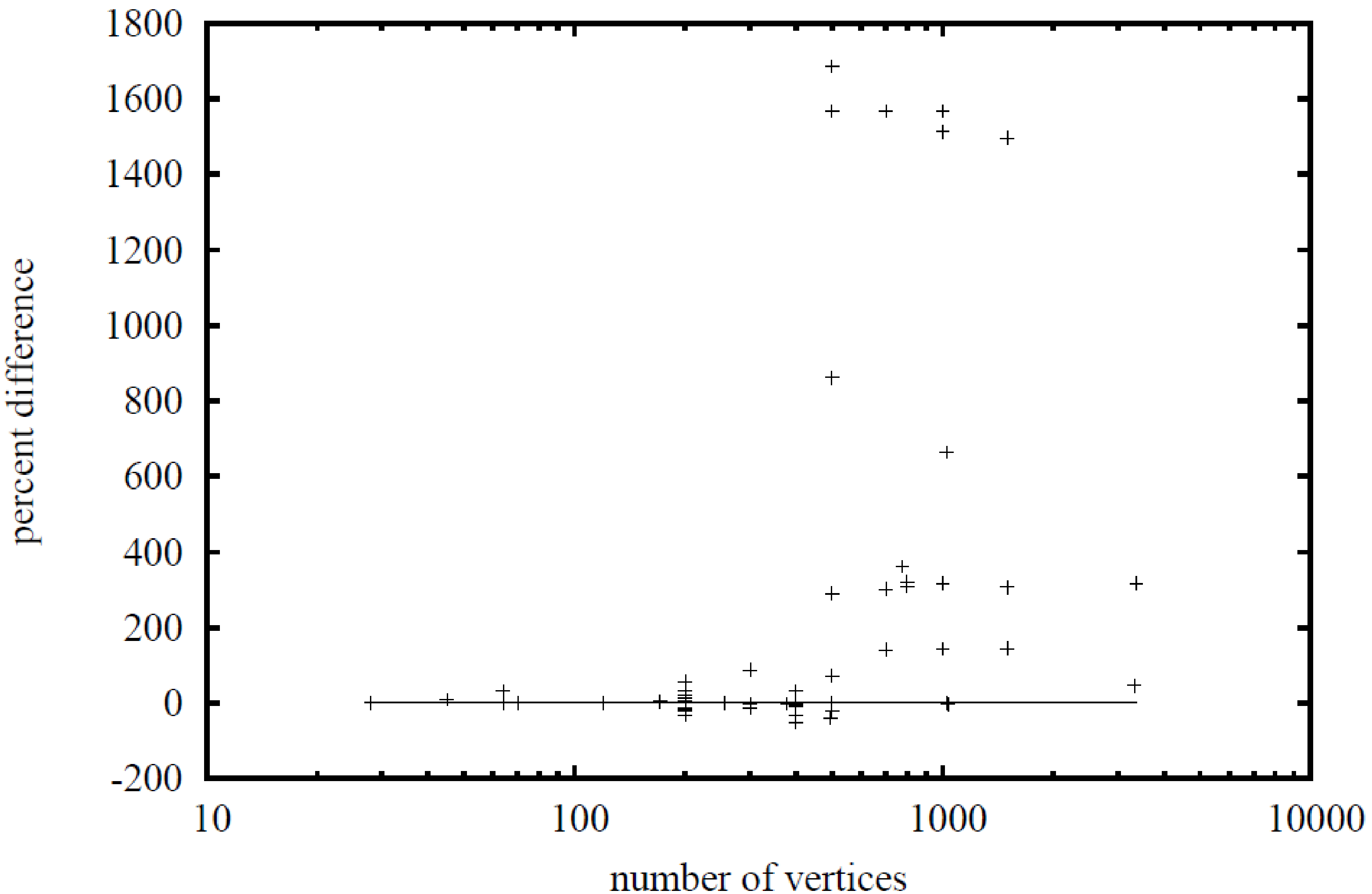
Carnegie Mellon Tepper School of Business

%difference, i.e., $100 \cdot (z_{LP} - z_{MDD})/z_{MDD}$



instances

Large MDD versus LP (CPLEX)



CPLEX time: total 10,424s, average 158s
MDD time: total 381s, average 6s

Restriction MDDs

Definition

- 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

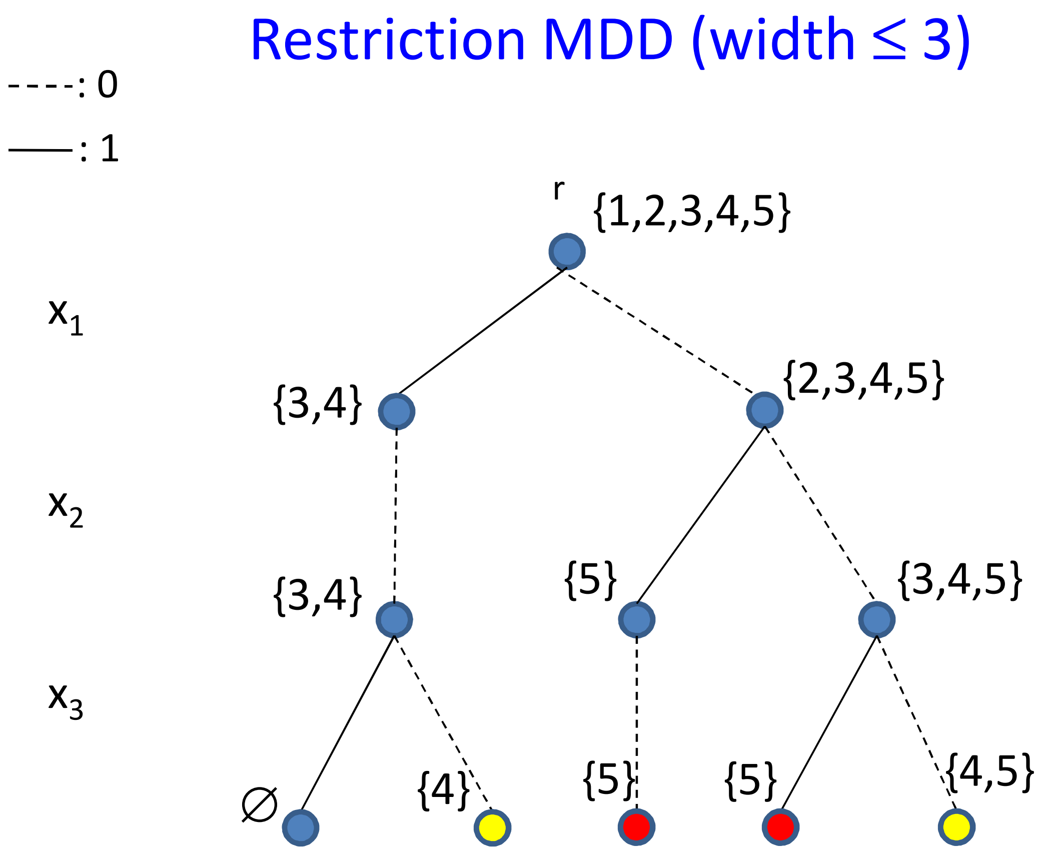
Creating Restriction MDDs

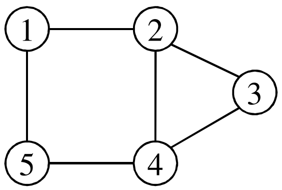
Using an exact top-down compilation method, we can create a limited-width restriction MDD by

1. merging nodes, or
2. deleting nodes

while ensuring that no solution is lost

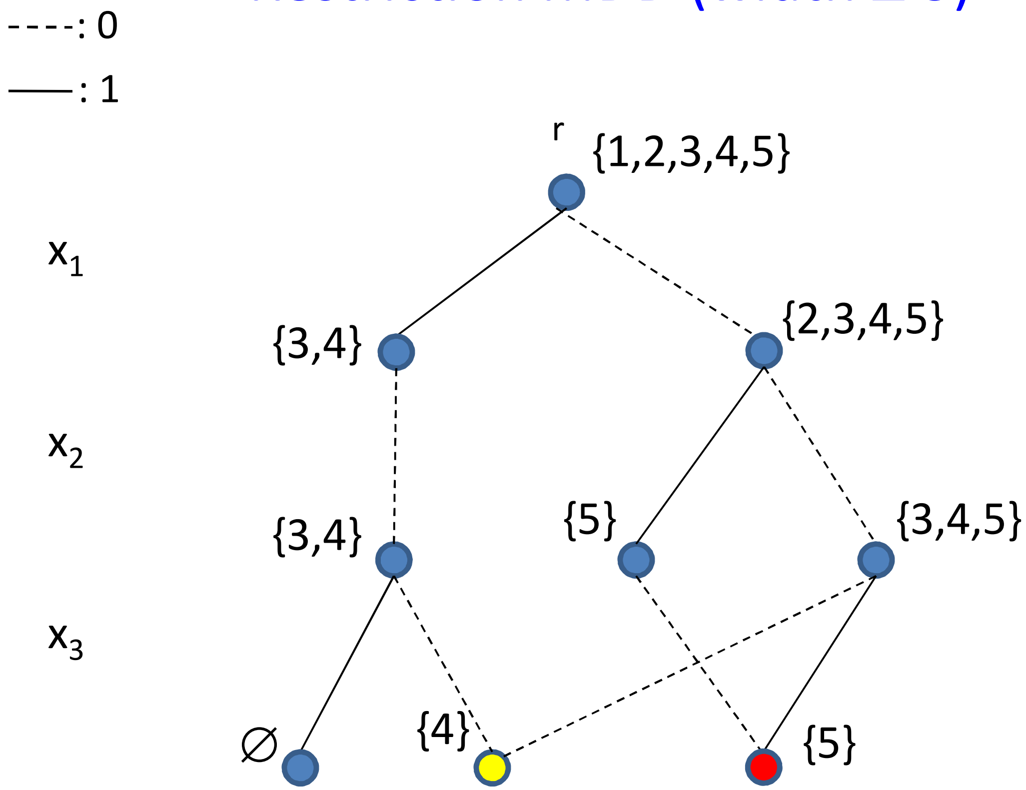
Node merging by example

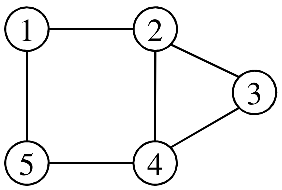




Node merging by example

Restriction MDD (width ≤ 3)



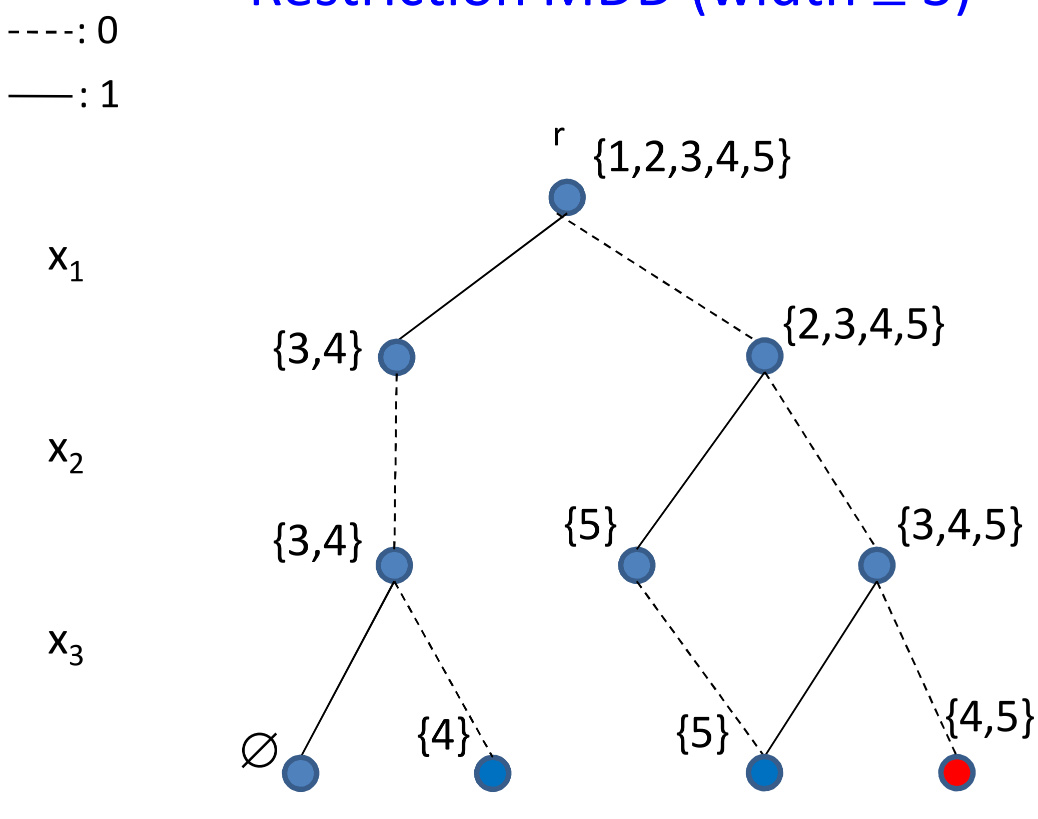


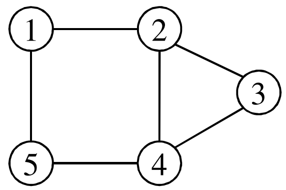
Node merging heuristics

- 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 $\leq$ 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

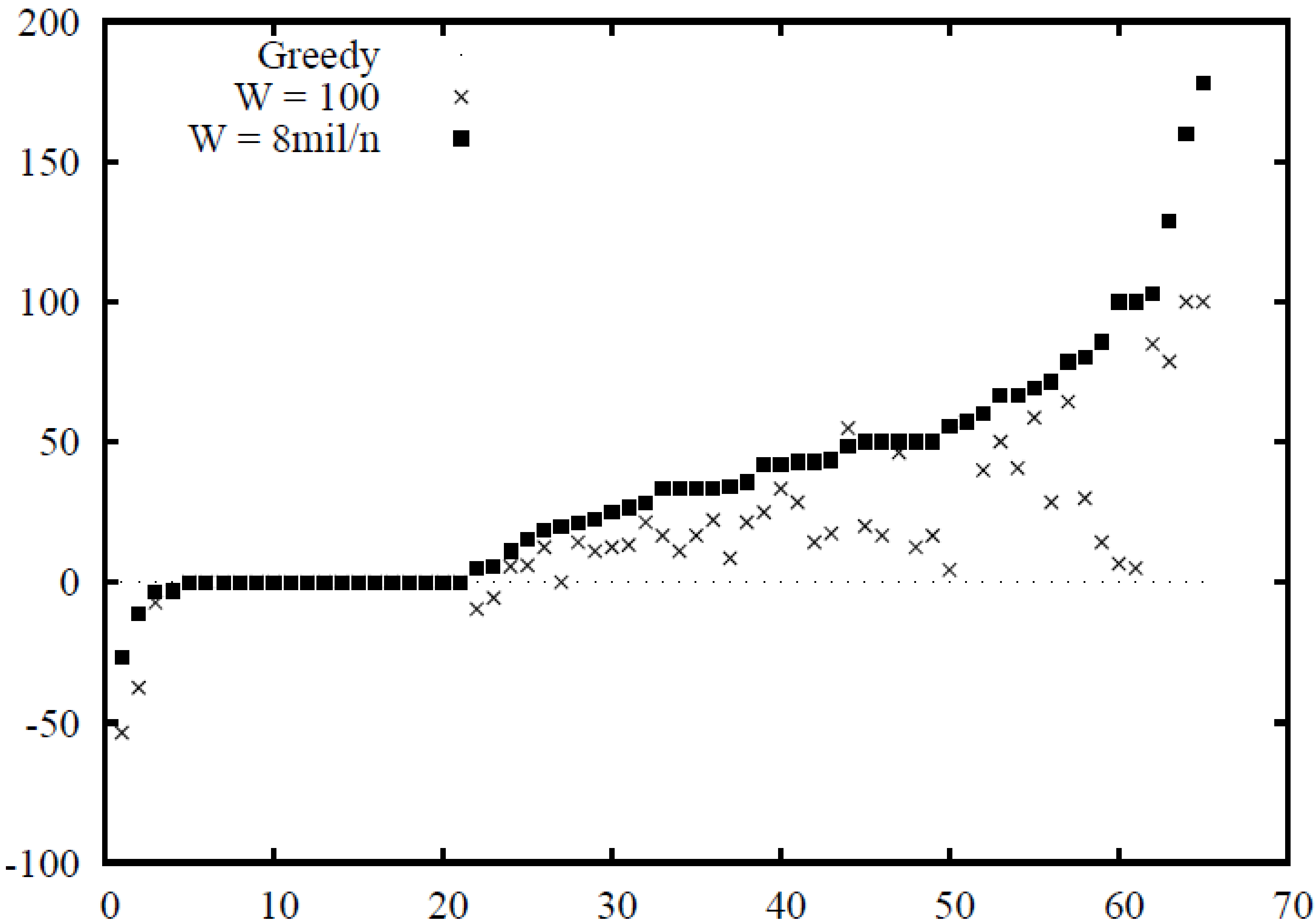
Experimental Results

- Comparison to greedy heuristic
  - select vertex v with smallest degree and add it to independent set
  - remove v and its neighbors and repeat

- MDD version 1: maximum width 100
- MDD version 2: maximum width 8,000,000/*n*

Greedy versus MDD

%difference, i.e., $100 \cdot (z_{MDD} - z_{Gr})/z_{Gr}$



CPU times for Greedy and MDD are similar

Conclusions

- Limited-width MDDs can be a very useful tool for discrete optimization
  - The maximum width provides a natural trade-off between computational efficiency and strength
  - Powerful inference mechanism for constraint propagation
  - Generic discrete relaxation and restriction method for MIP-style problems
- Many open questions
  - MDD variable ordering, interaction with search, formal characterizations, …