Decision Diagrams for Discrete Optimization

Willem-Jan van Hoeve

Tepper School of Business

Carnegie Mellon University

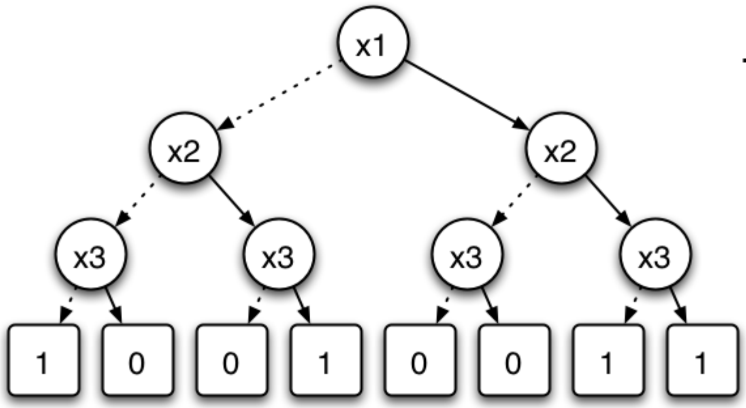
based on joint work with

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

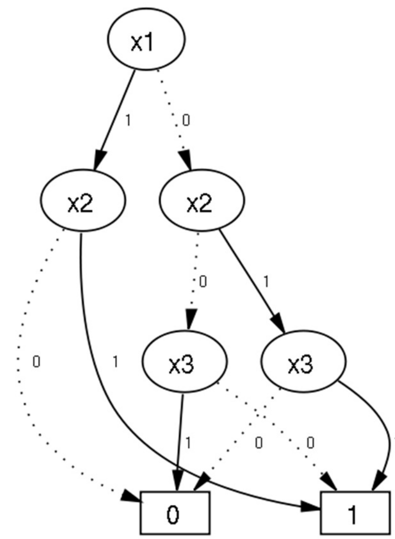
Outline

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

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) = -x1 * -x2 * -x3 + x1 * x1 * x2 + x2 * 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

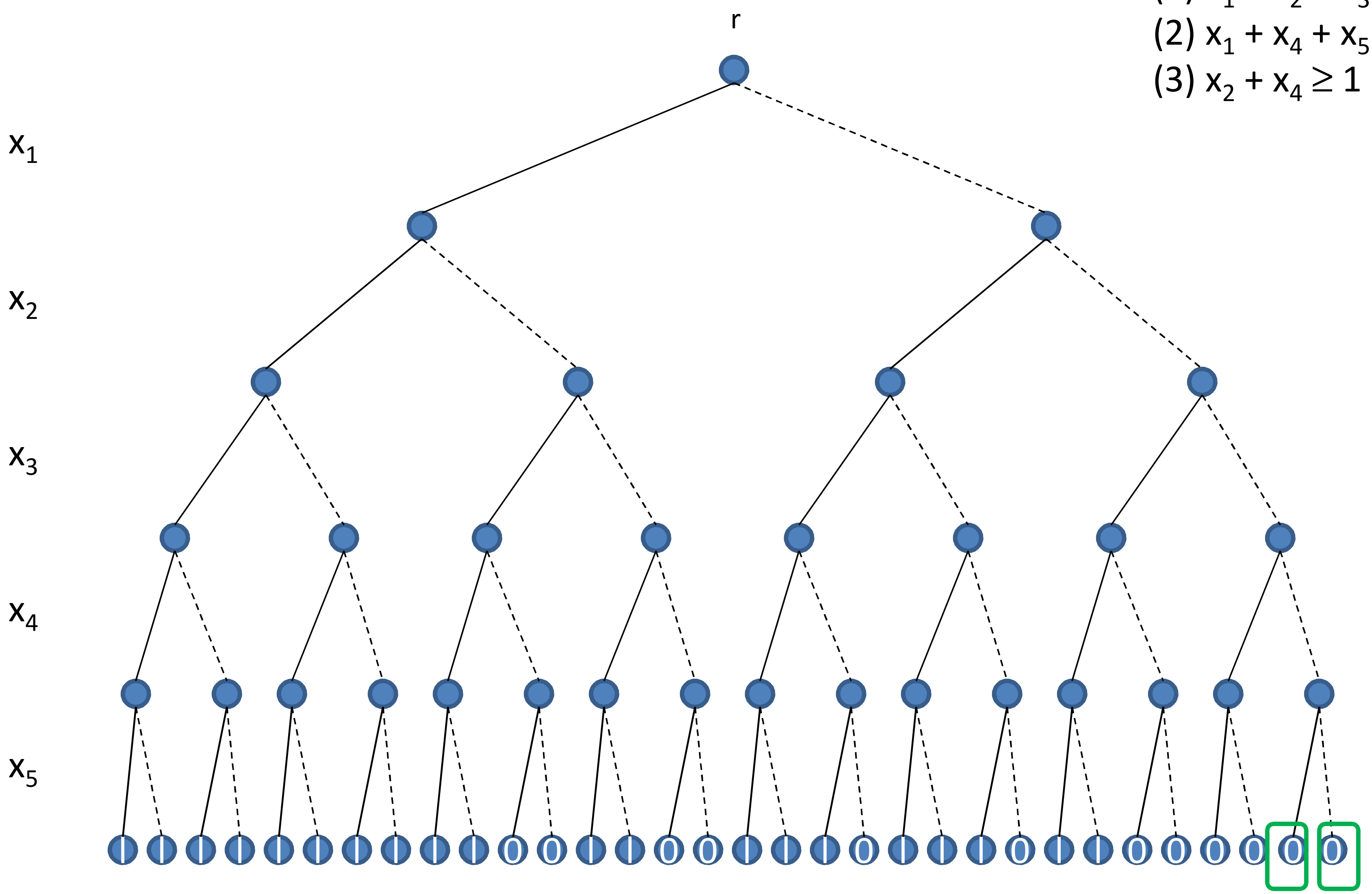
- Original application areas: circuit design, verification
- Usually Reduced Ordered BDDs/MDDs are applied
  - fixed variable ordering
  - minimal exact representation
- Recent interest from optimization community
  - cut generation [Becker et al., 2005]
  - 0/1 vertex and facet enumeration [Behle & 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

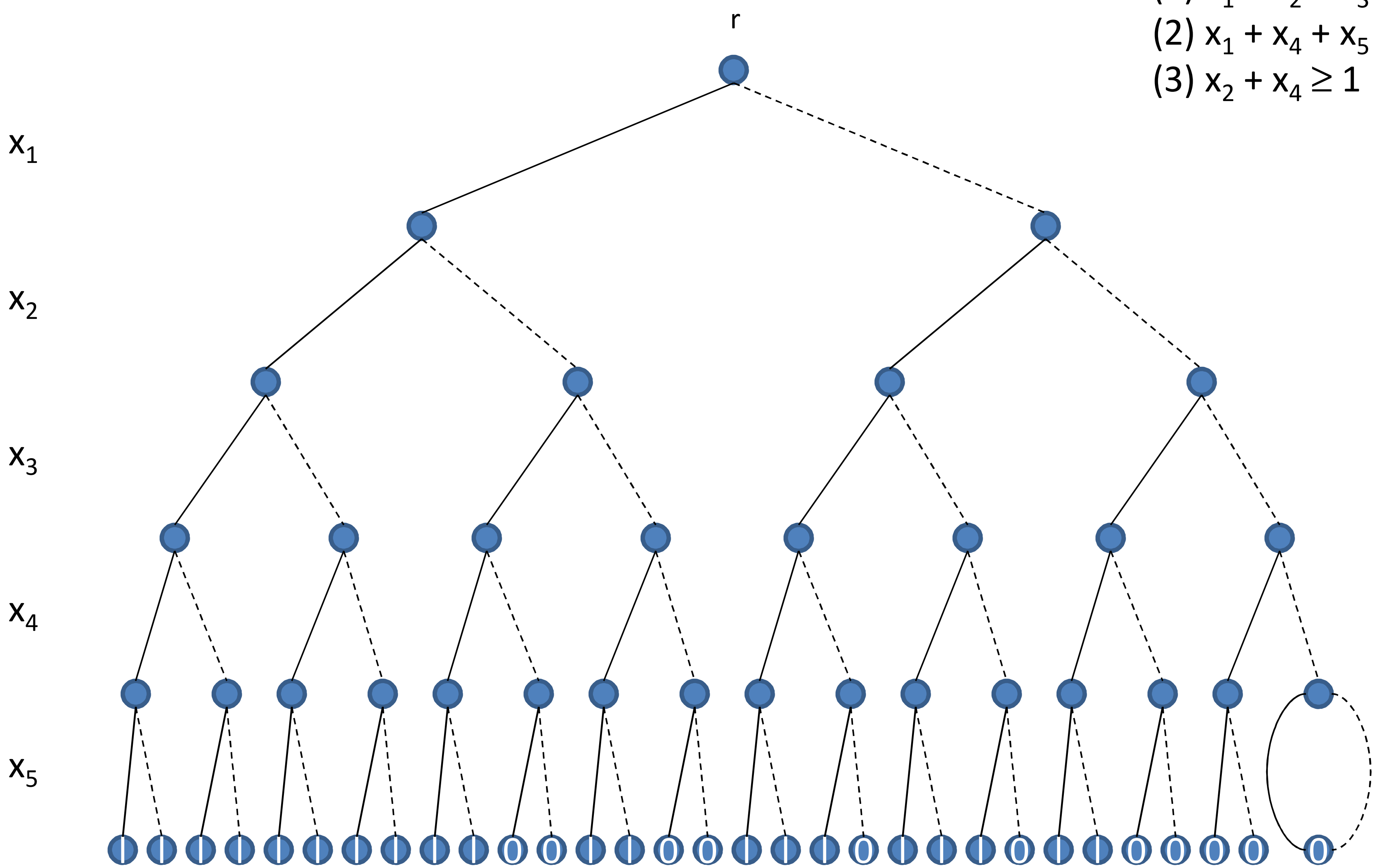
(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$



Exact MDDs for discrete optimization

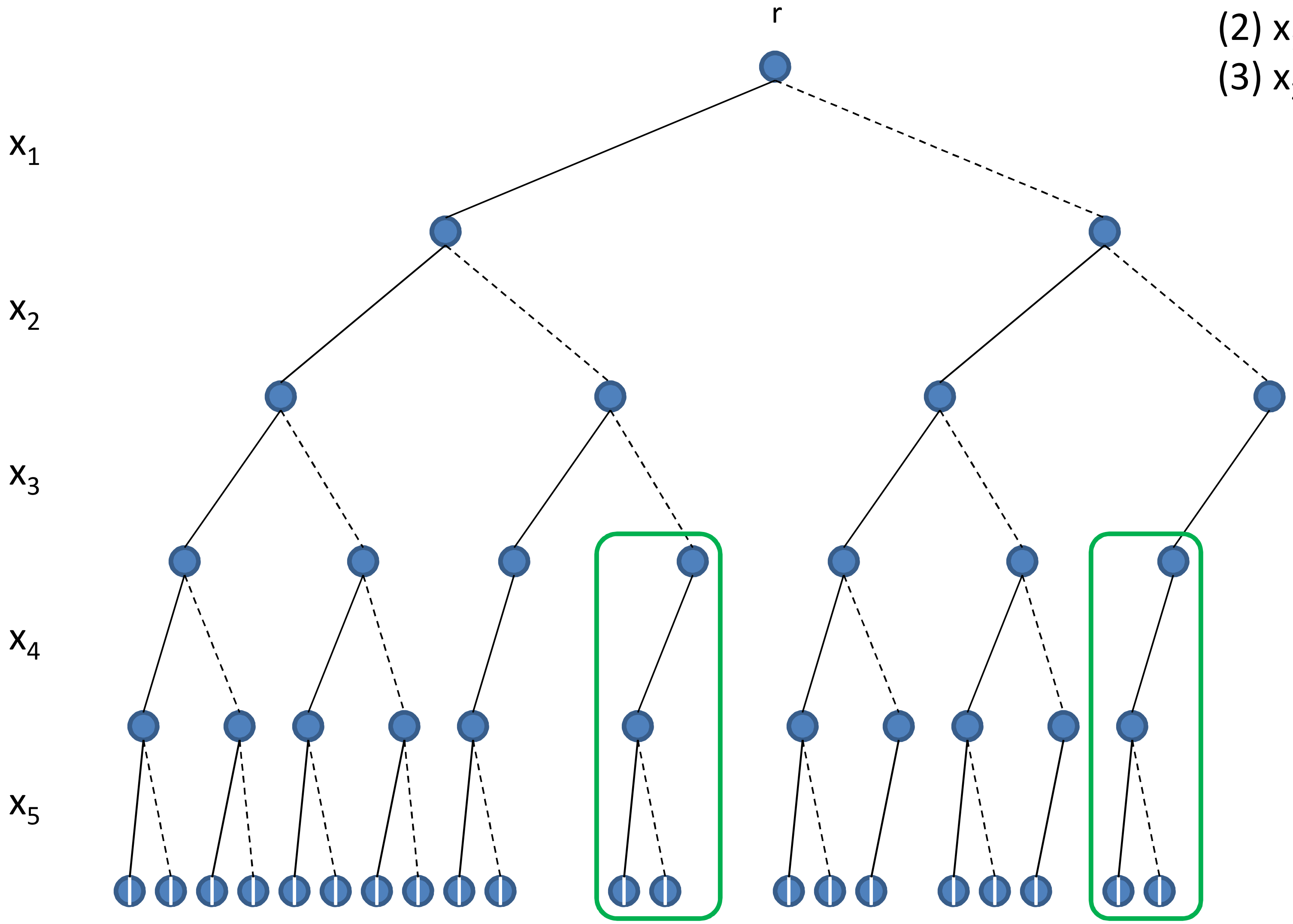
Carnegie Mellon Tepper SCHOOL OF BUSINESS

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



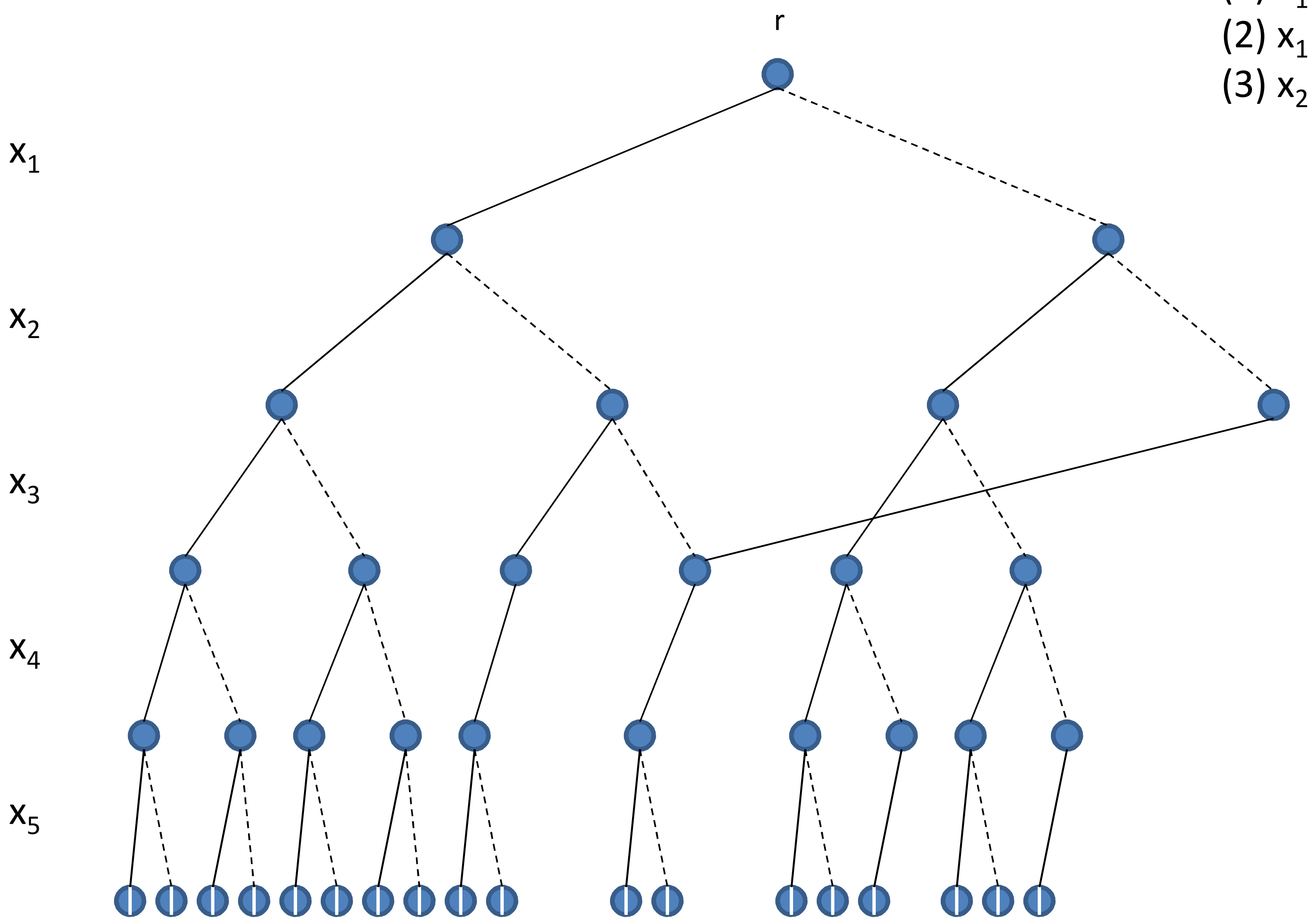
Exact MDDs for discrete optimization

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



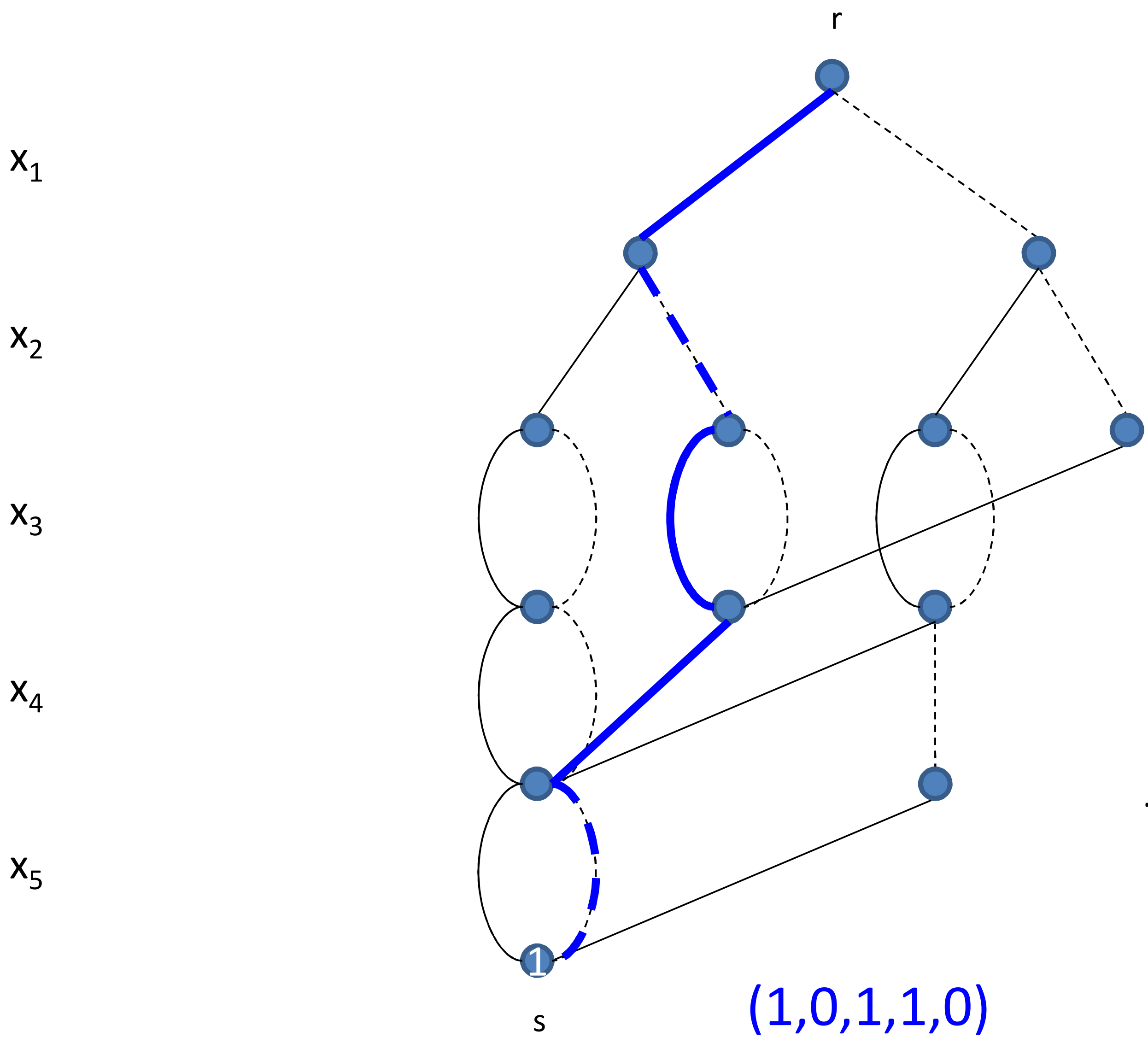
Exact MDDs for discrete optimization

(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) $x_2 + x_4 \geq 1$



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

Carnegie Mellon Tepper SCHOOL OF BUSINESS

Observations:

- Communication between constraints only via variable domains
- Information can only be expressed as a domain change
- Other (structural) information that may be learned by 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)

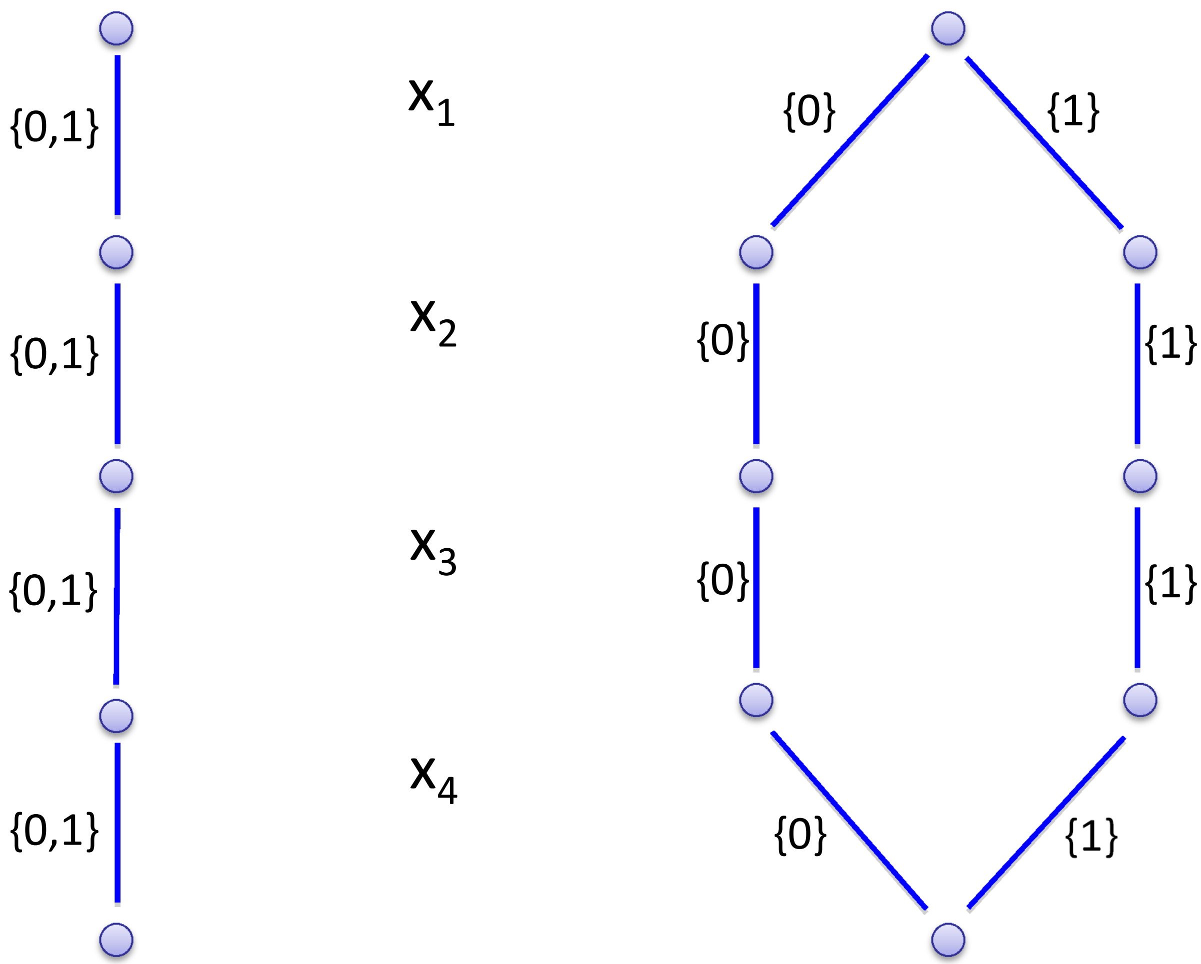
This drawback can be addressed by communicating more expressive information, using MDDs [Andersen et al. 2007]

- Explicit representation of more refined potential solution space

Illustrative Example

Carnegie Mellon Tepper SCHOOL OF BUSINESS

AllEqual(x_1, x_2, x_3, x_4), all x_i binary



domain representation, size 2^4

MDD representation, size 2

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

Specific MDD propagation algorithms

- Linear equalities and inequalities [Hadzic et al., 2008] [Hoda et al., 2010]
- *Alldifferent* constraints [Andersen et al., 2007]
- *Element* constraints [Hoda et al., 2010]
- *Among* constraints [Hoda et al., 2010]
- 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]

Case study: Among constraints

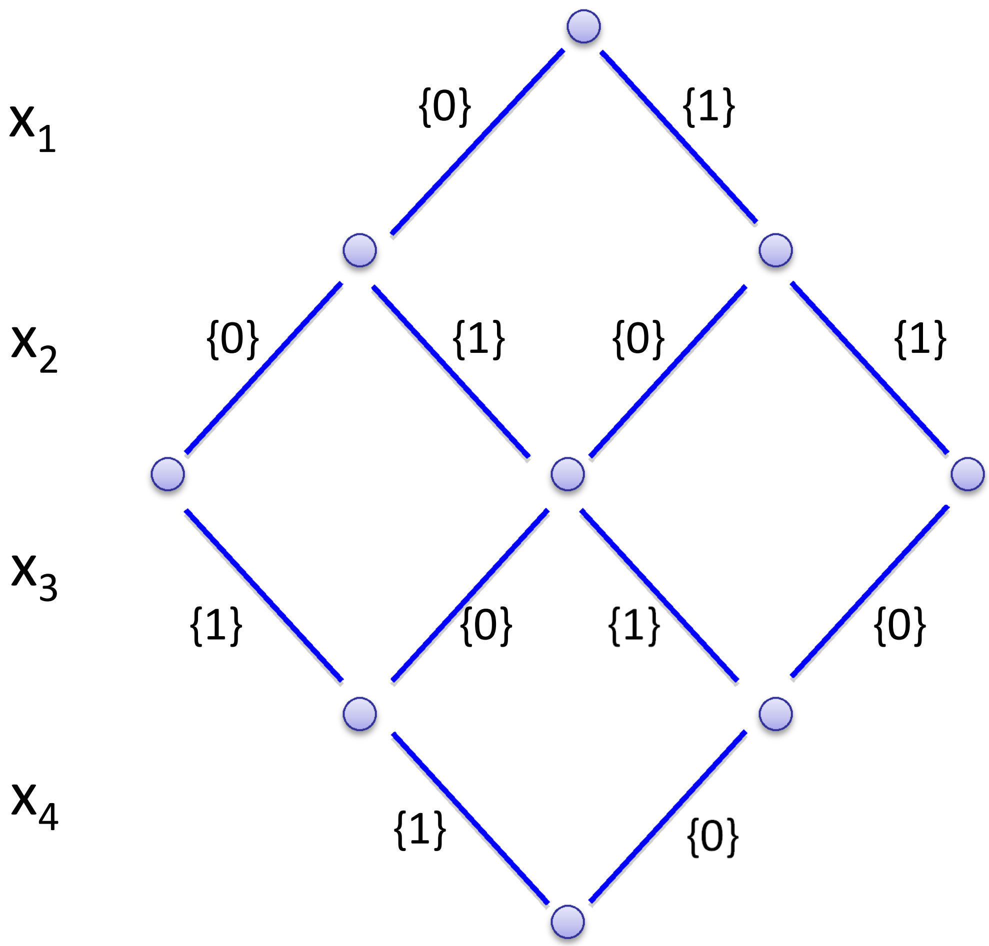
- Given a set of variables X, and a set of values *S*, a lower bound *l* and upper bound *u*,

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

"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



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

MDD Filtering for Among

Goal: Given an MDD and an *Among* constraint, remove *all* inconsistent edges from the MDD
(establish "MDD-consistency")

Approach:

- Compute path lengths from the top node and from the bottom node
- Remove edges that are not on a path with lengths 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

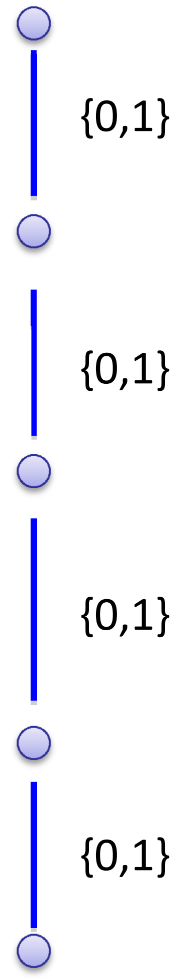
For each layer in MDD, we first apply edge filter, and then try to refine

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

Example:

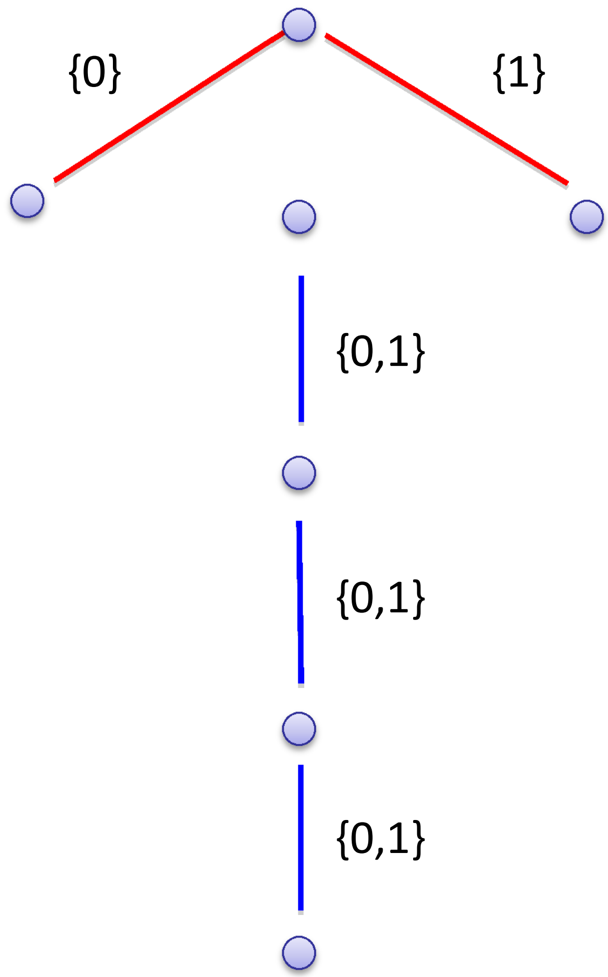
- We will propagate *Among*($\{x_1,x_2,x_3,x_4\}$,{1},2,2) through a BDD of maximum width 3

Example



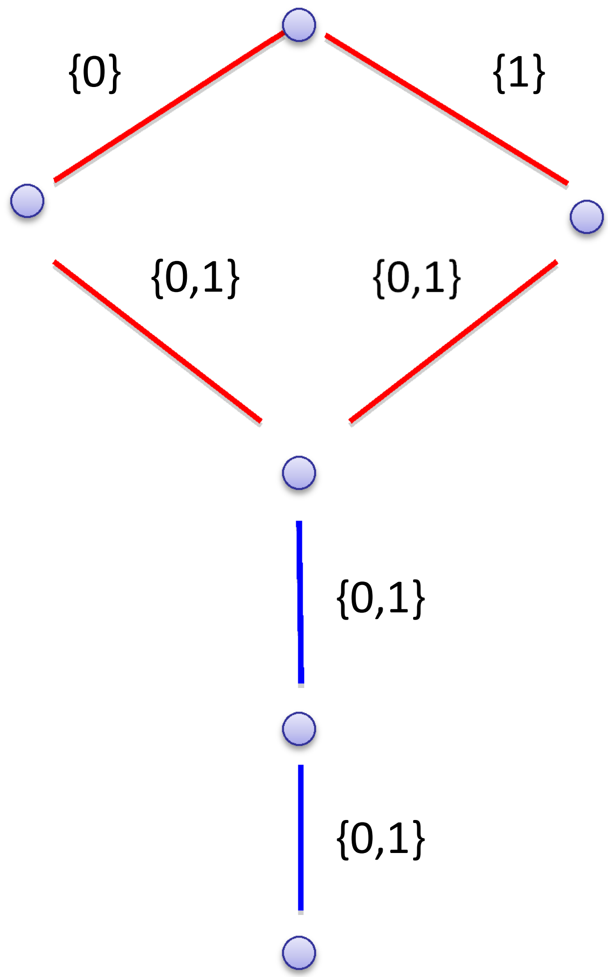
$Among(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example



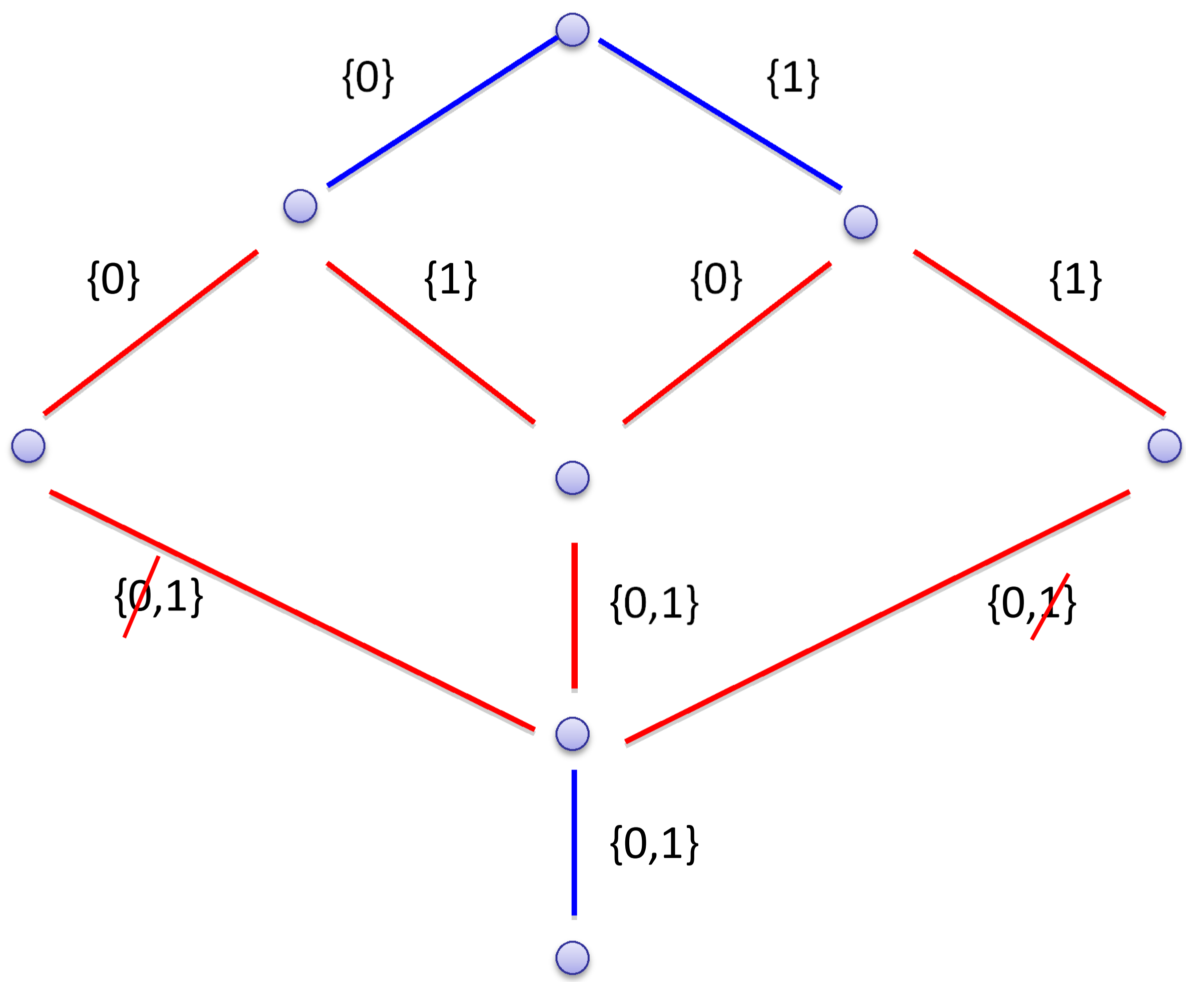
$Among(\{x_1,x_2,x_3,x_4\},\{1\},2,2)$

Example



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

Example



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

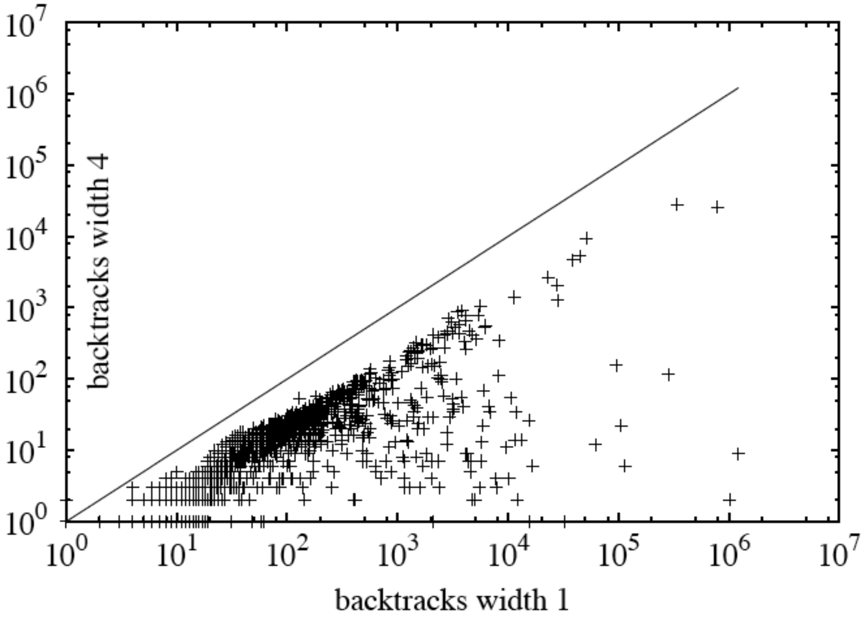
Experiments

- 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

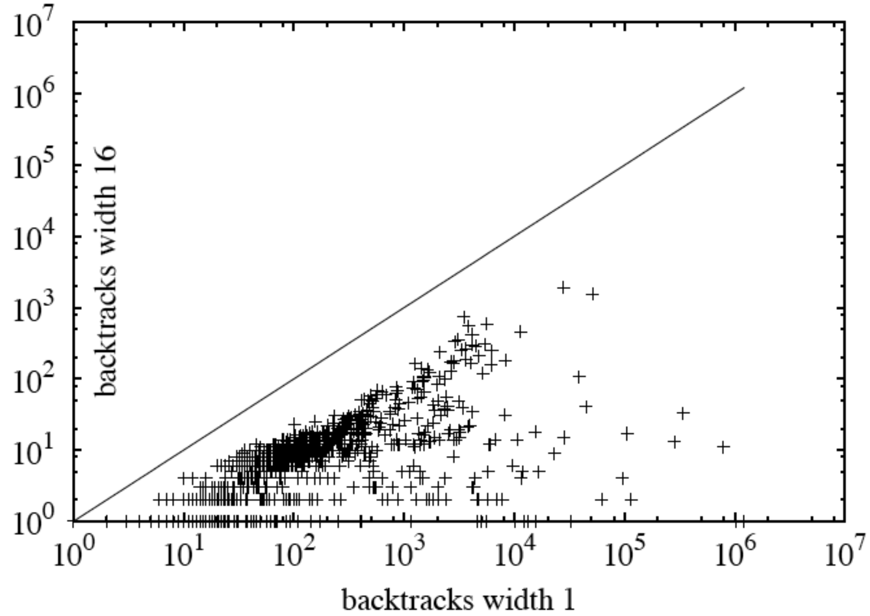
- Nurse rostering instances (horizon *n* days)
  - Work 4-5 days per week
  - Max A days every B days
  - Min C days every D days
  - Three problem classes

▪ Compare width 1 (traditional domains) with increasing widths

Multiple Amongs: Backtracks

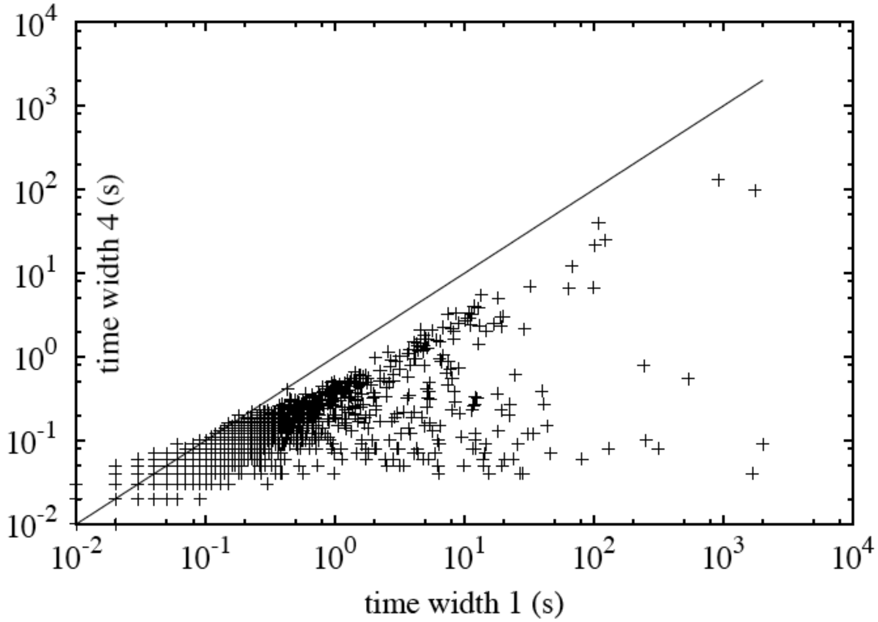


width 1 vs 4

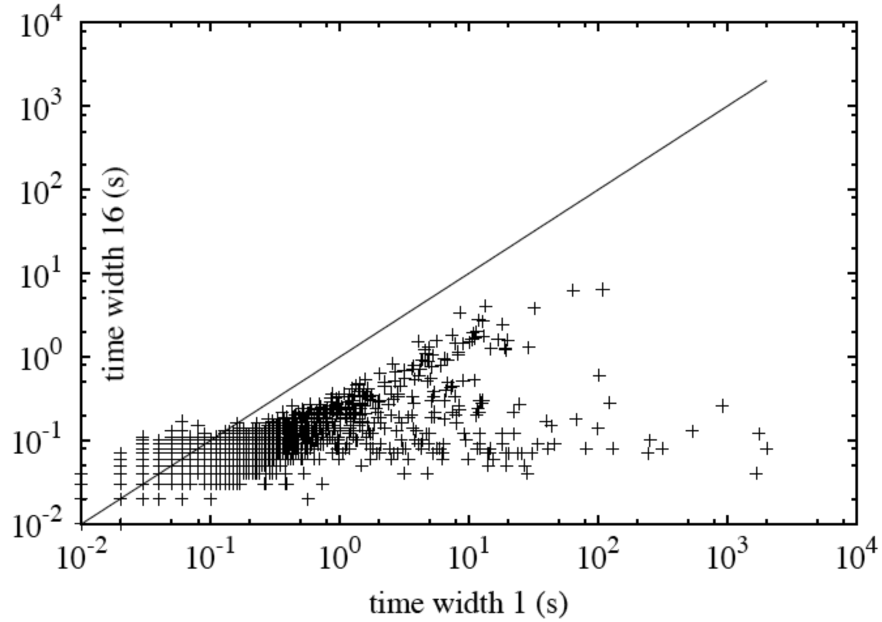


width 1 vs 16

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

- MDD provides 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

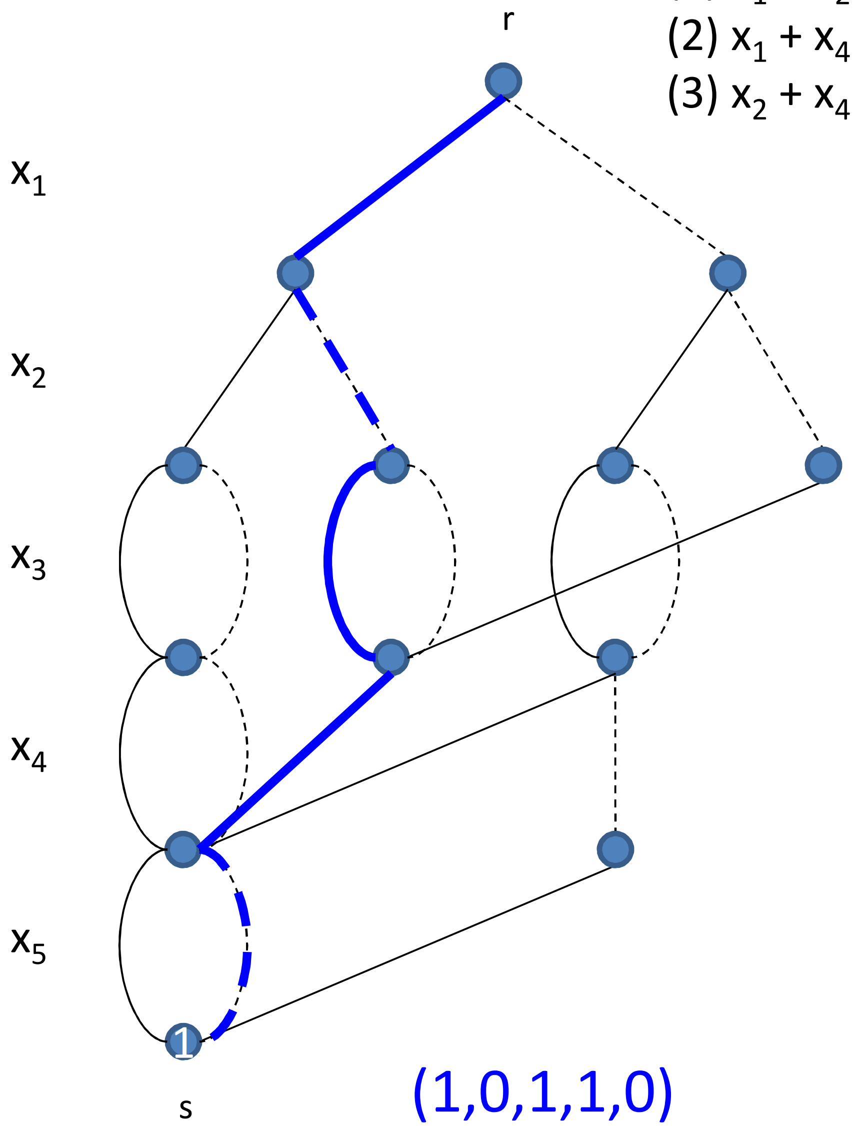
Bergman, v.H., and Hooker. Manipulating MDD Relaxations for Combinatorial Optimization. In *Proceedings of CPAIOR*, LNCS 6697, pp. 20-35. Springer, 2011.

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

- 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

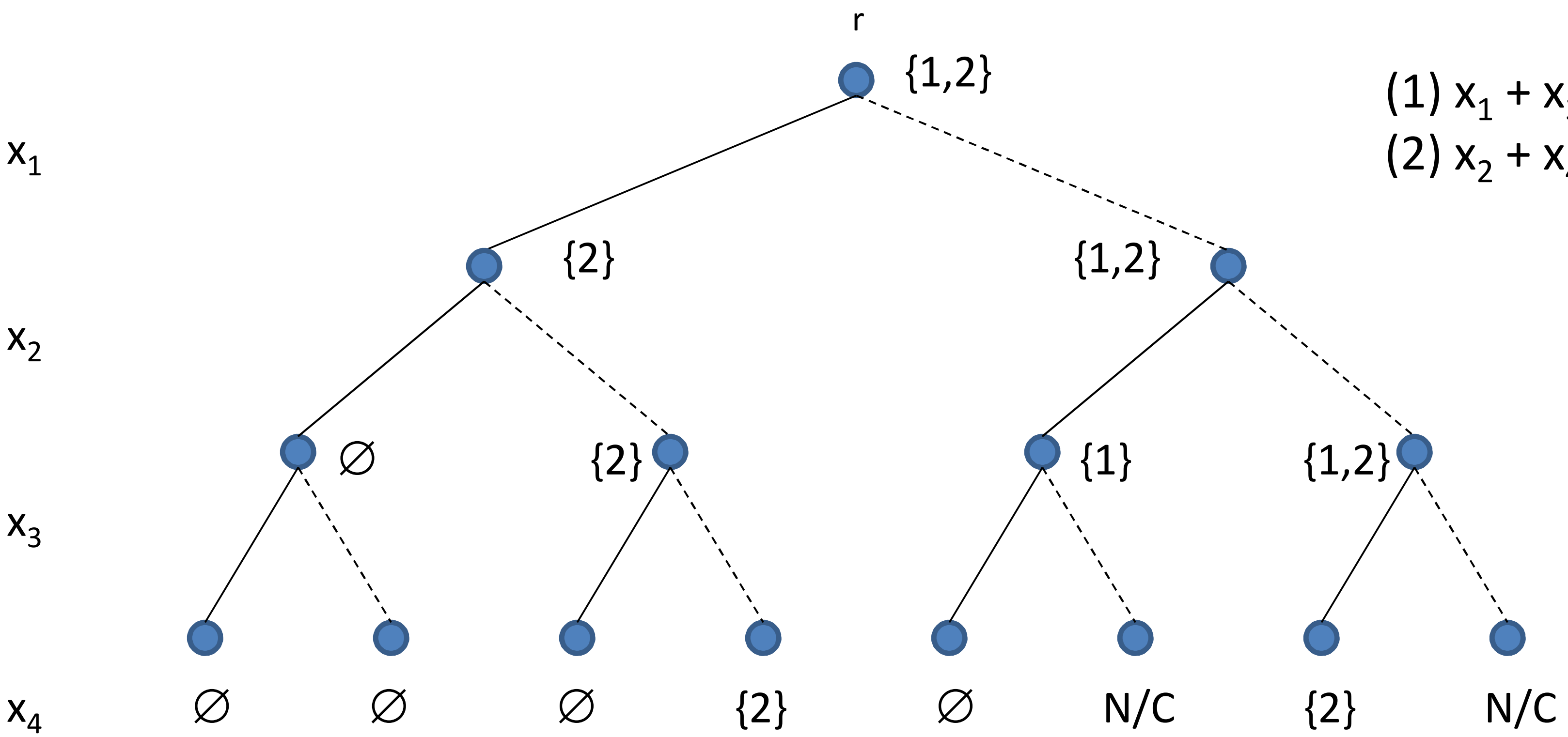
* Feasibility can be checked using MDD-based CP

Case Study: Set covering problem

- Given set $S=\{1,\ldots,n\}$ 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 $i=1,\ldots,m$

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

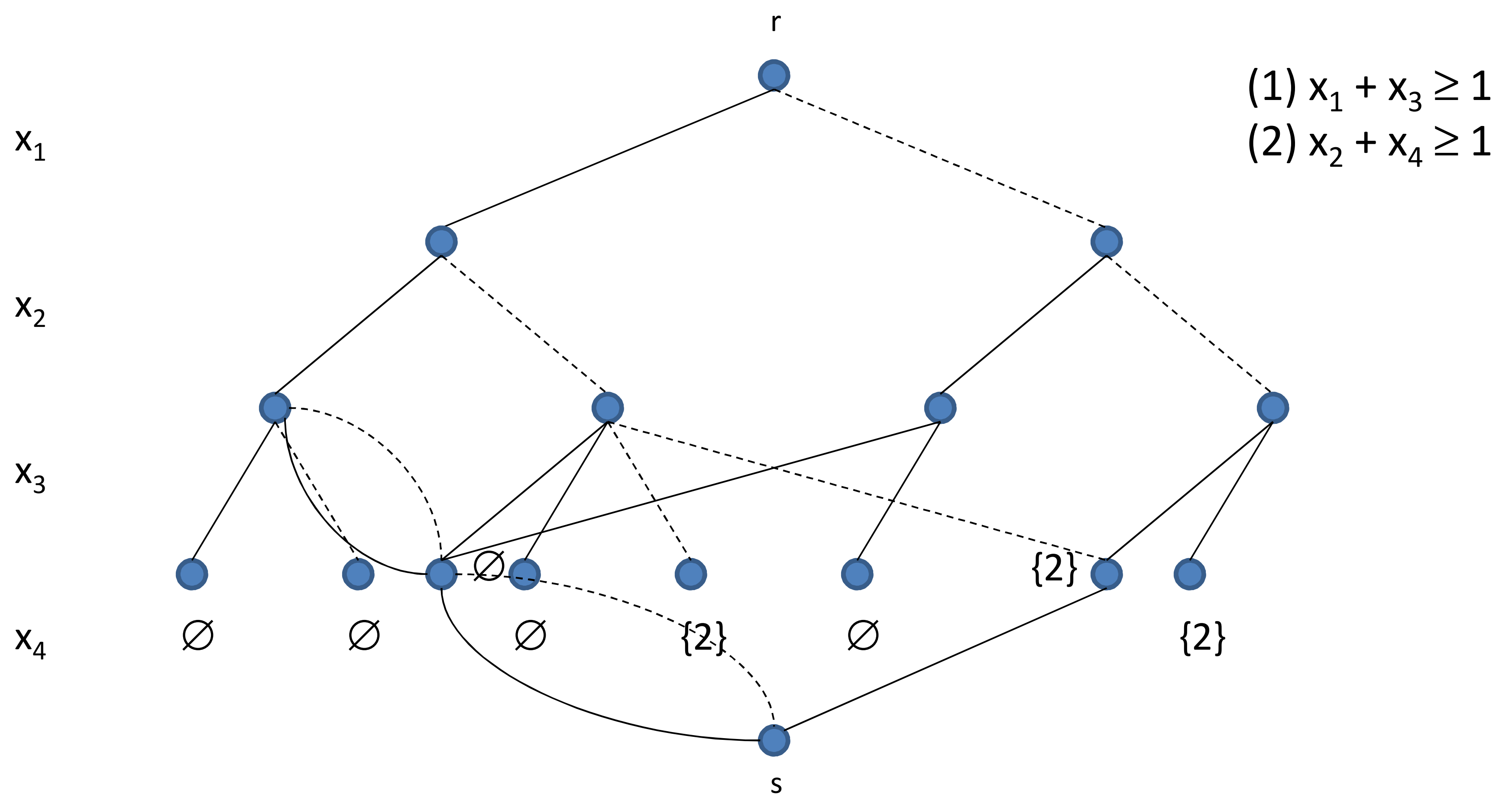
Exact top-down compilation



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

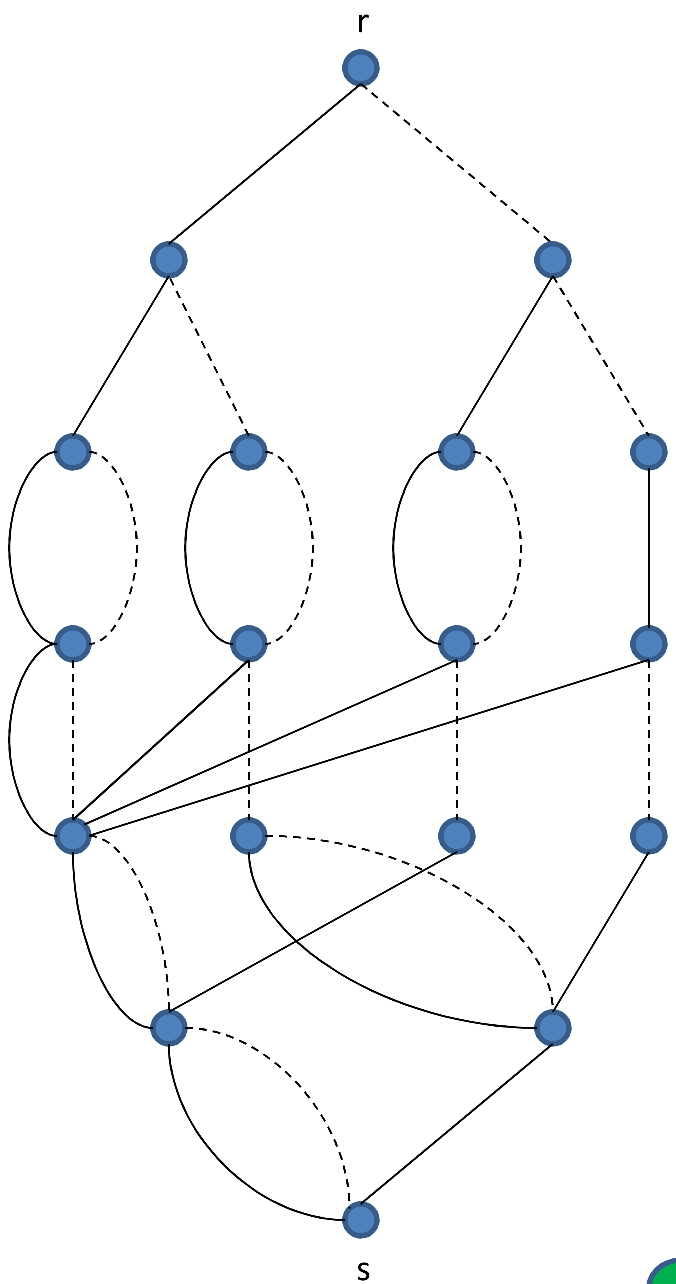
{Indices of the constraints that still need a 1}

Equivalence test for set covering



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

Exact MDD



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

x_1

x_2

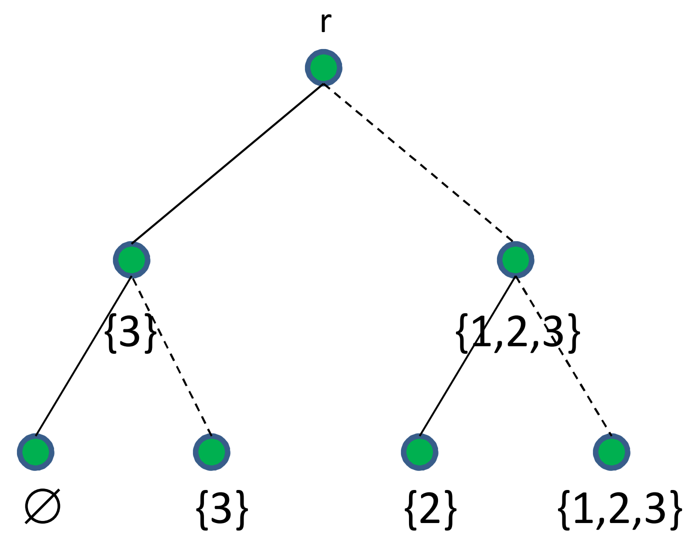
x_3

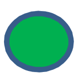
x_4

x_5

x_6

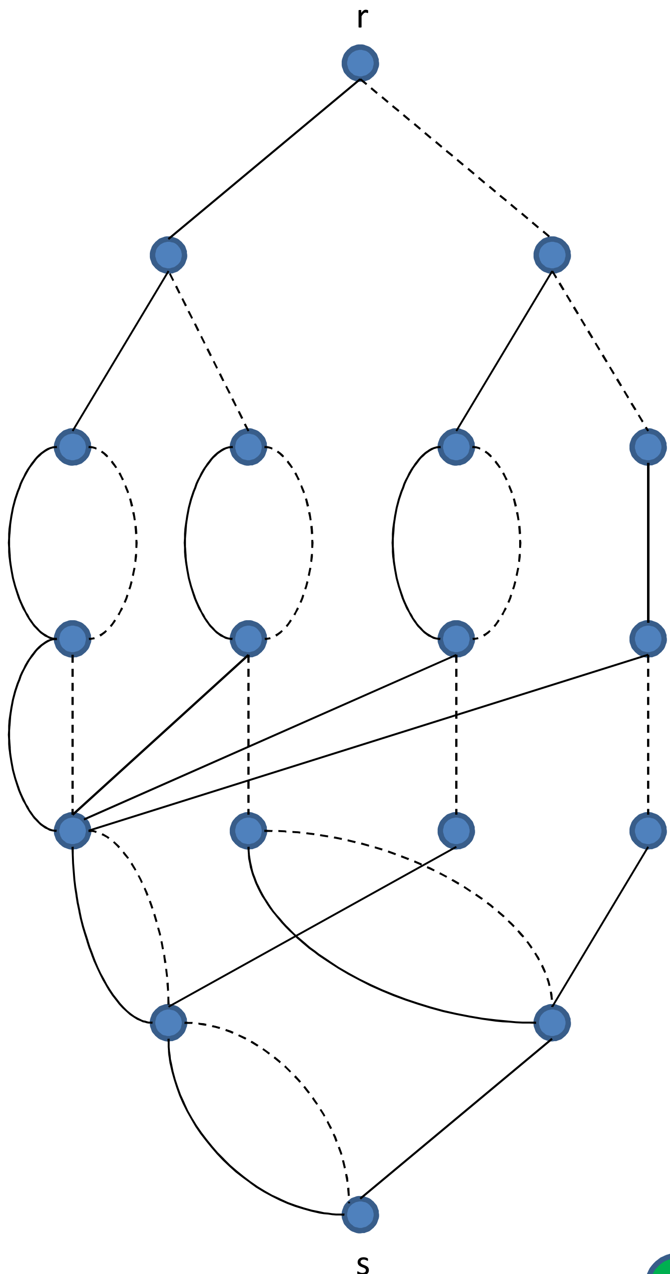
Relaxation MDD (width ≤ 3)





{Indices of the constraints that still need a 1}

Exact MDD



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

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

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

x_1

x_2

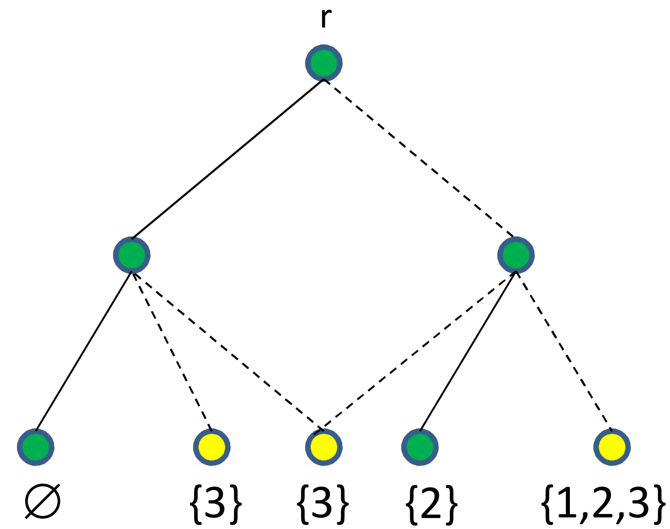
x_3

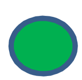
x_4

x_5

x_6

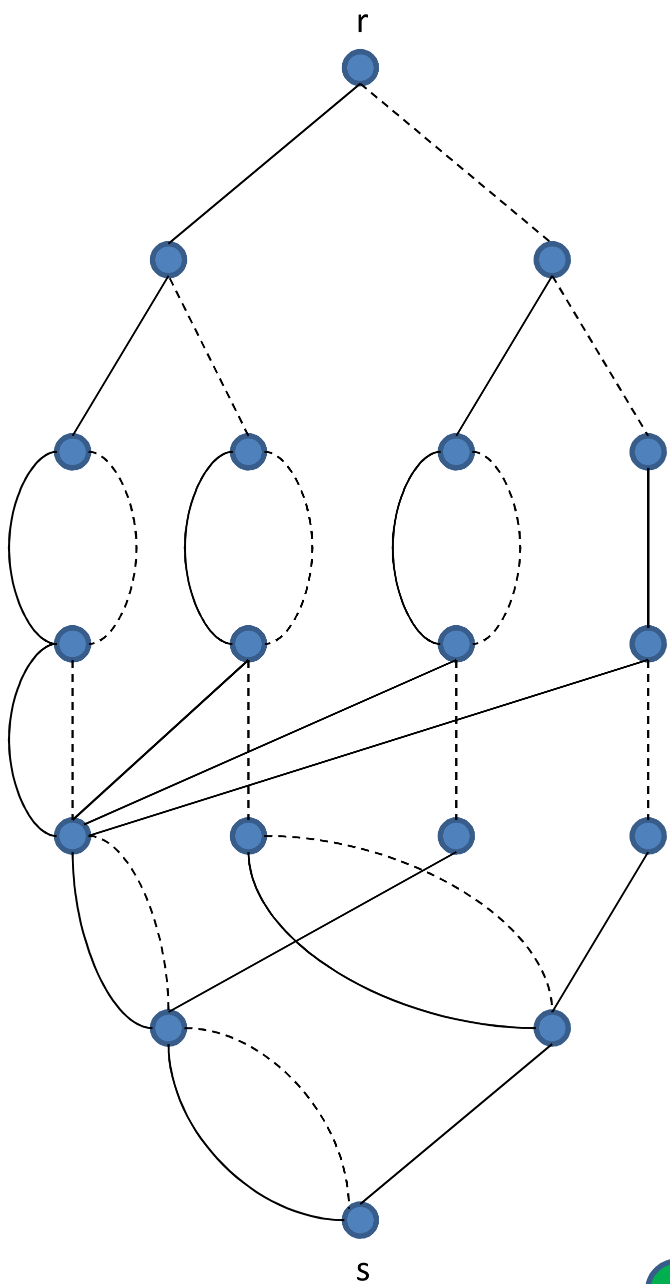
Relaxation MDD (width $\leq$ 3)





{Indices of the constraints that still need a 1}

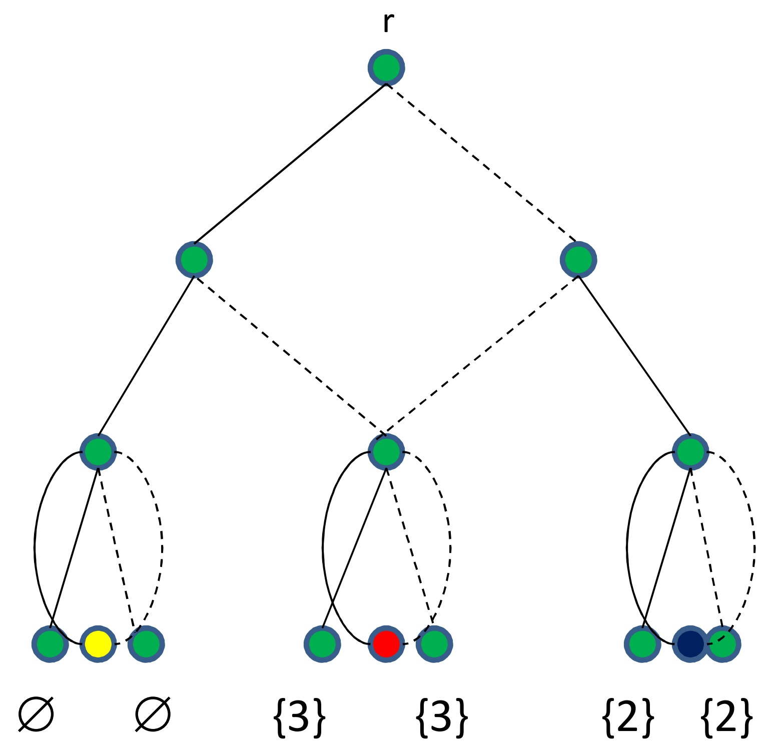
Exact MDD



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

x_1
x_2
x_3
x_4
x_5
x_6

Relaxation MDD (width ≤ 3)



{Indices of the constraints that still need a 1}

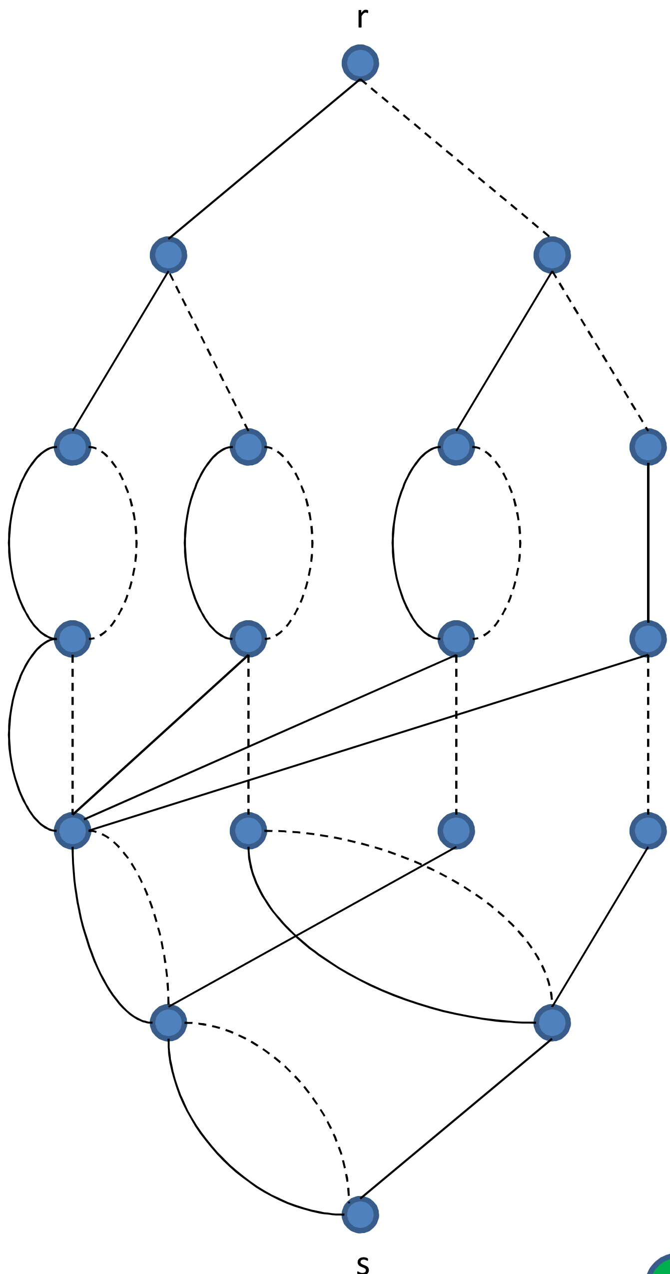
Exact MDD

Relaxation MDD (width ≤ 3)

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

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

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



x_1

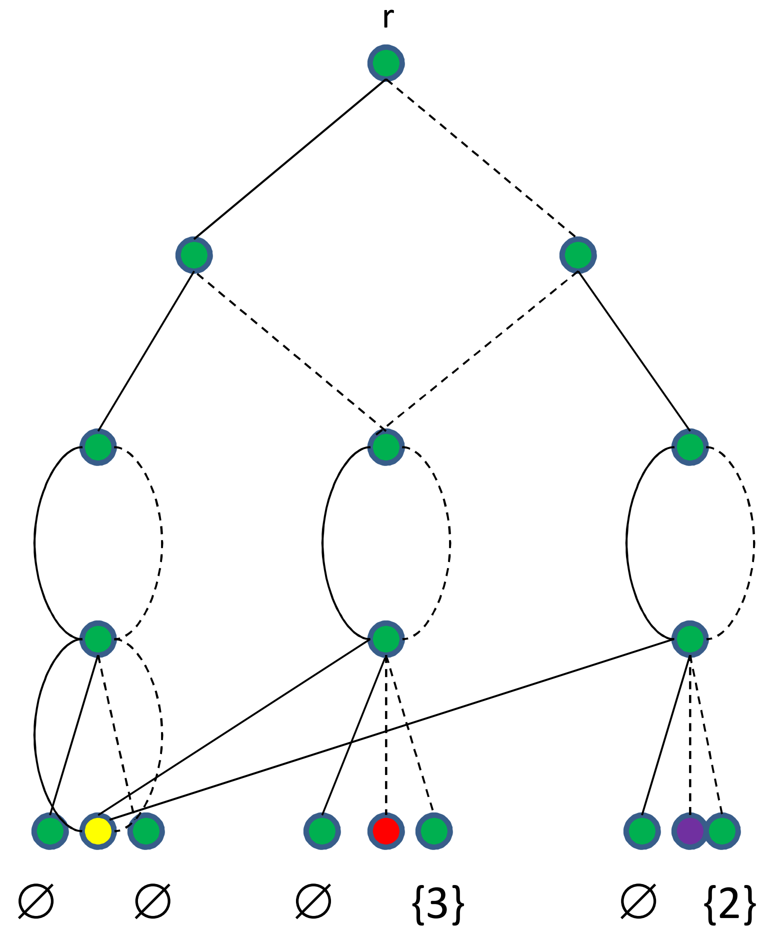
x_2

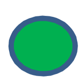
x_3

x_4

x_5

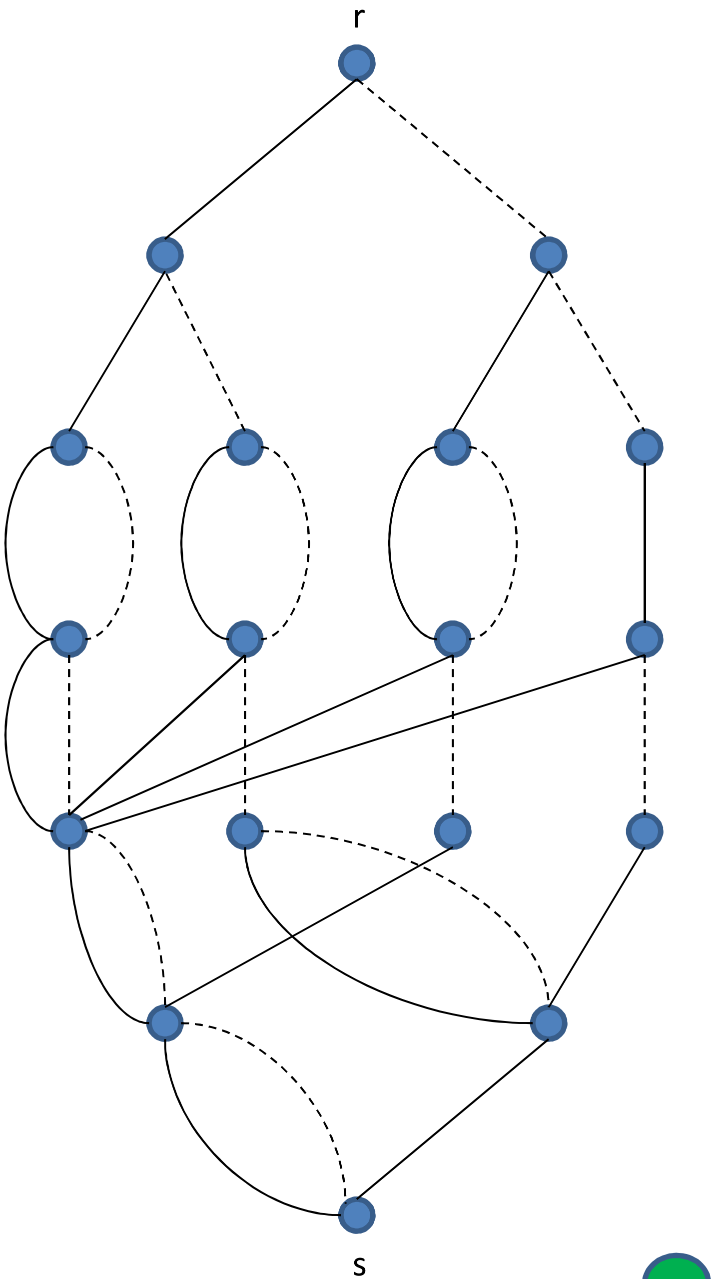
x_6





{Indices of the constraints that still need a 1}

Exact MDD



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

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

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

x_1

x_2

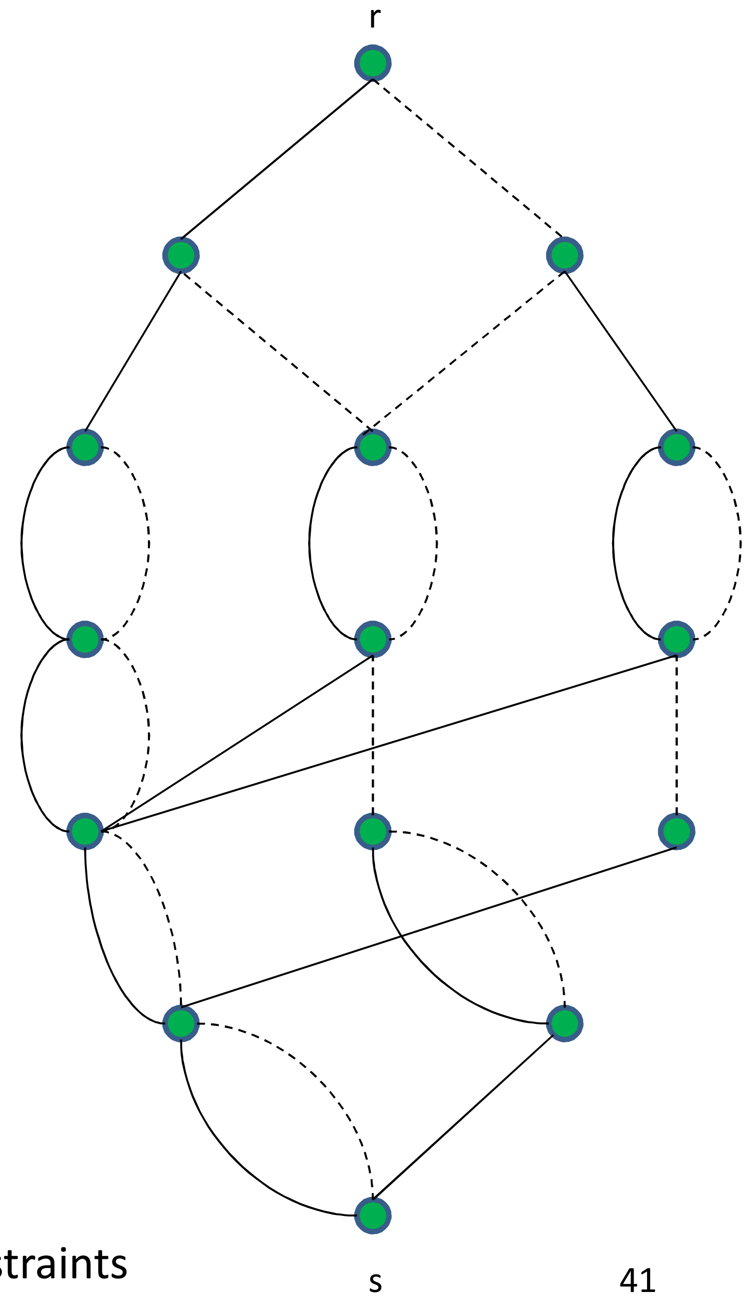
x_3

x_4

x_5

x_6

Relaxation MDD (width $\leq$ 3)



{Indices of the constraints that still need a 1}

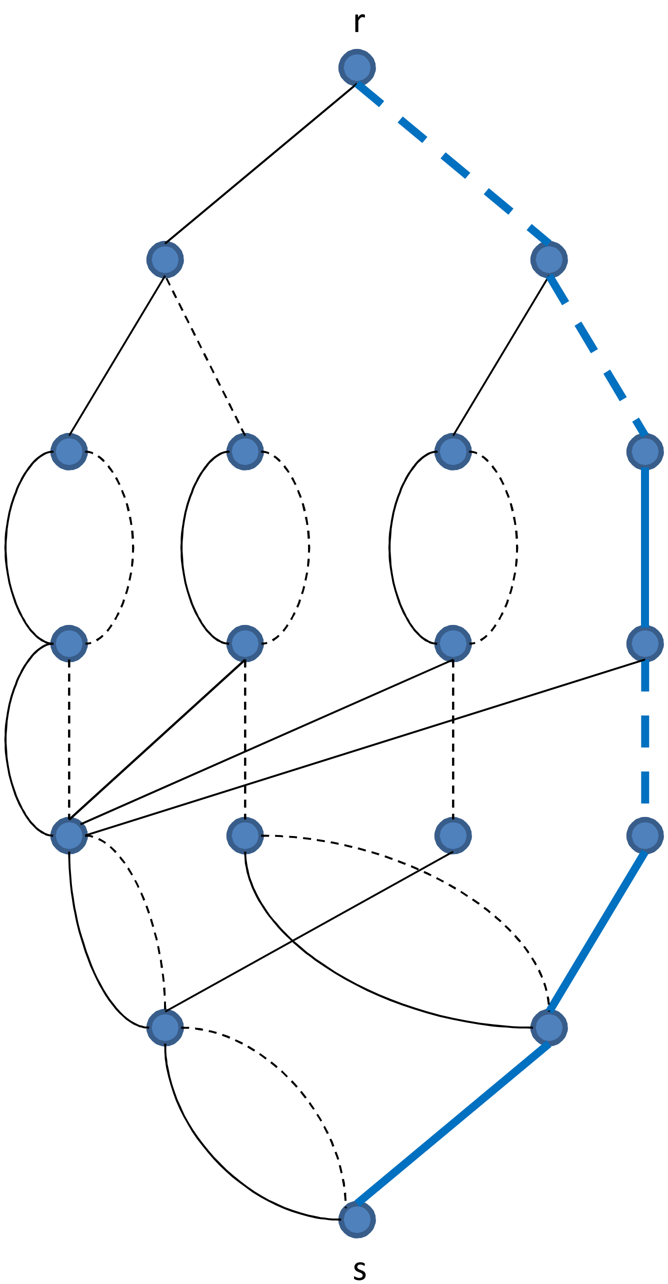
Exact MDD

Relaxation MDD (width $\leq$ 3)

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

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

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



x_1

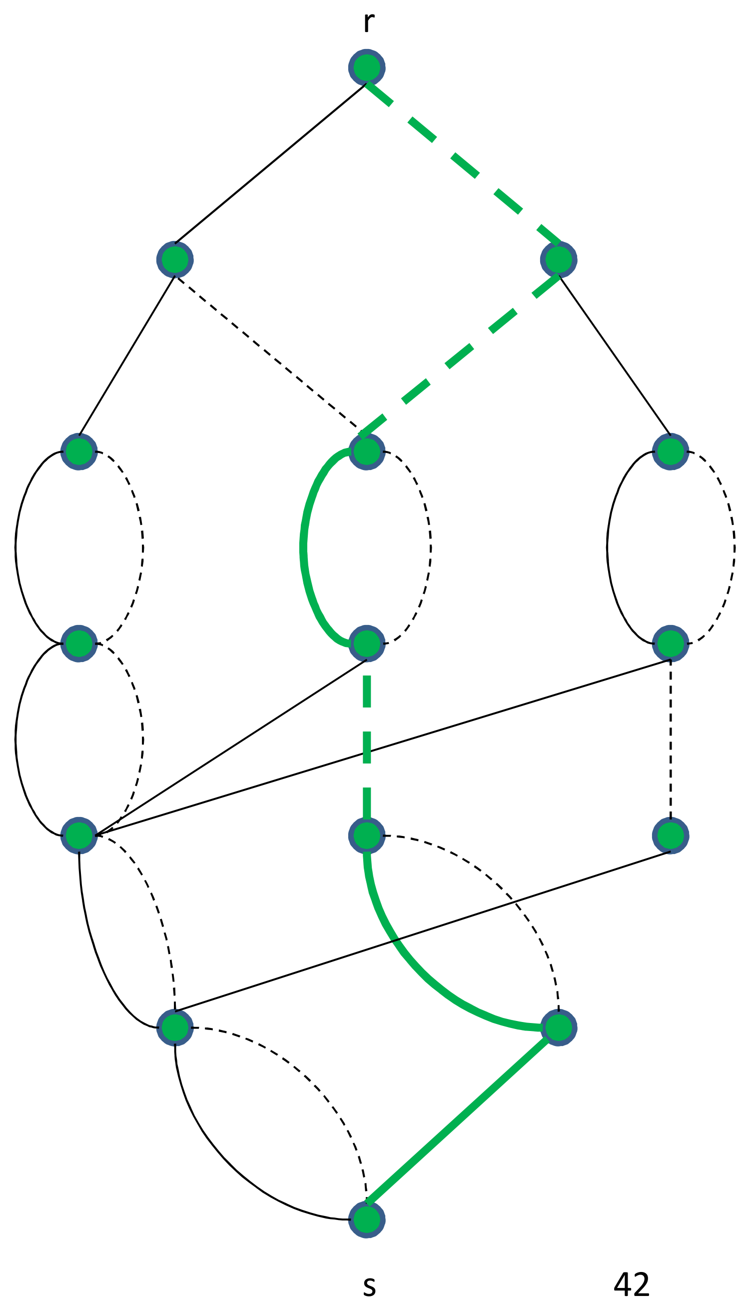
x_2

x_3

x_4

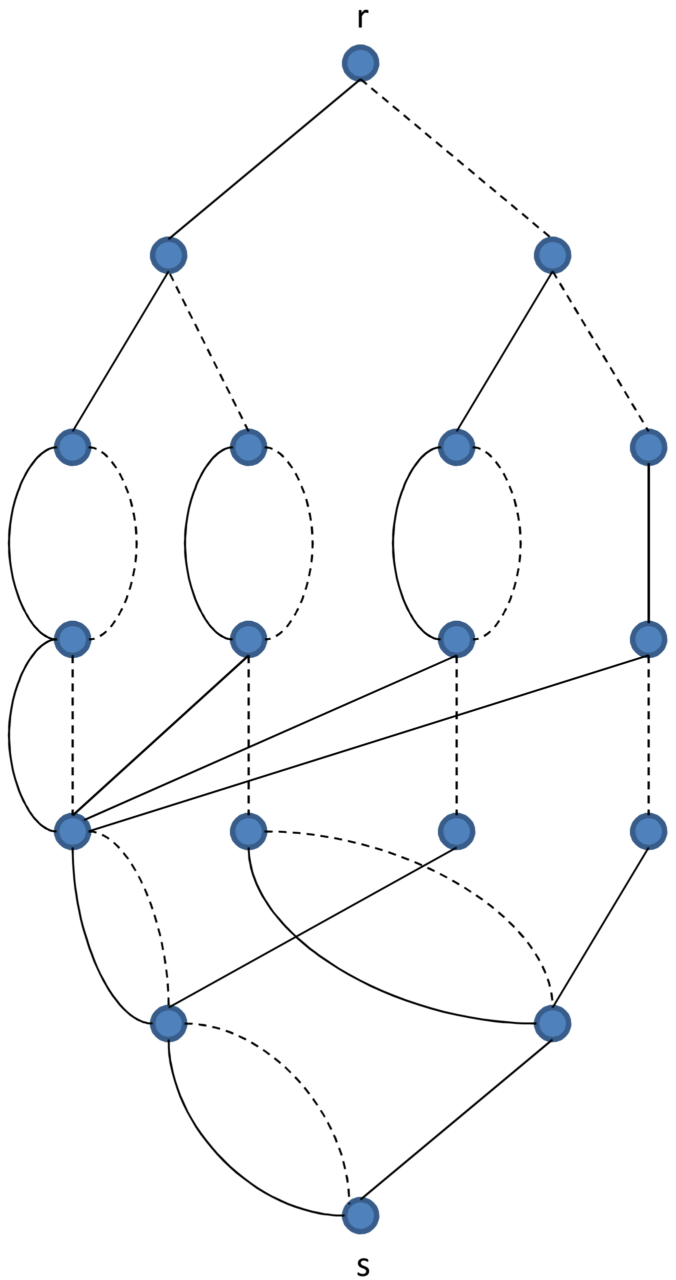
x_5

x_6



$(0,0,1,0,1,1)$

Exact MDD



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

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

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

x_1

x_2

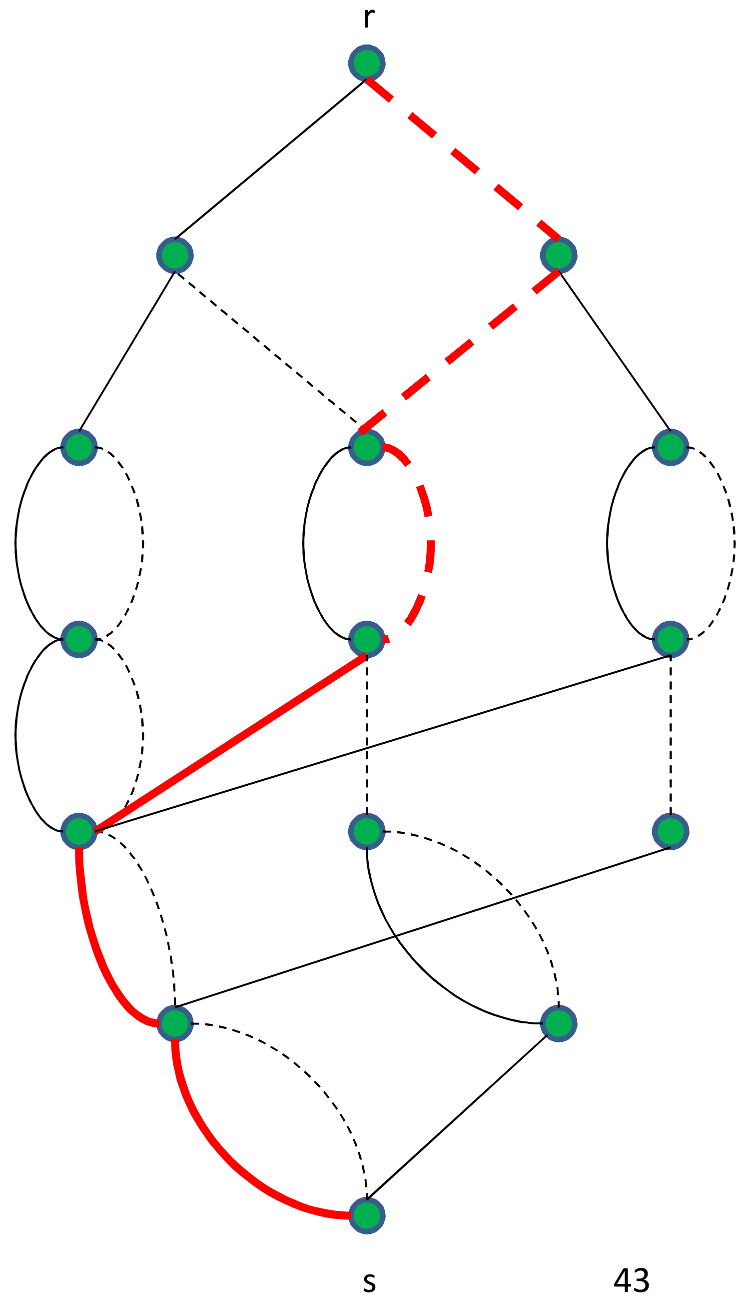
x_3

x_4

x_5

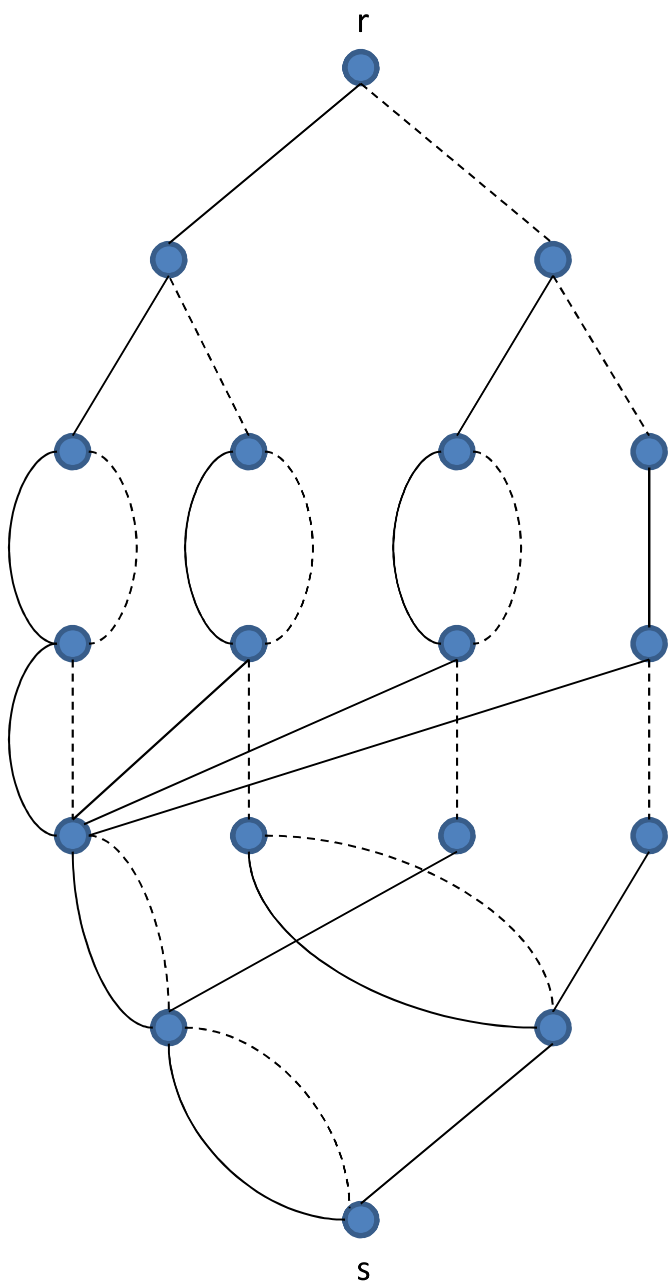
x_6

Relaxation MDD (width $\leq$ 3)



(0,0,0,1,1,1)

Exact MDD



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

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

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

x_1

x_2

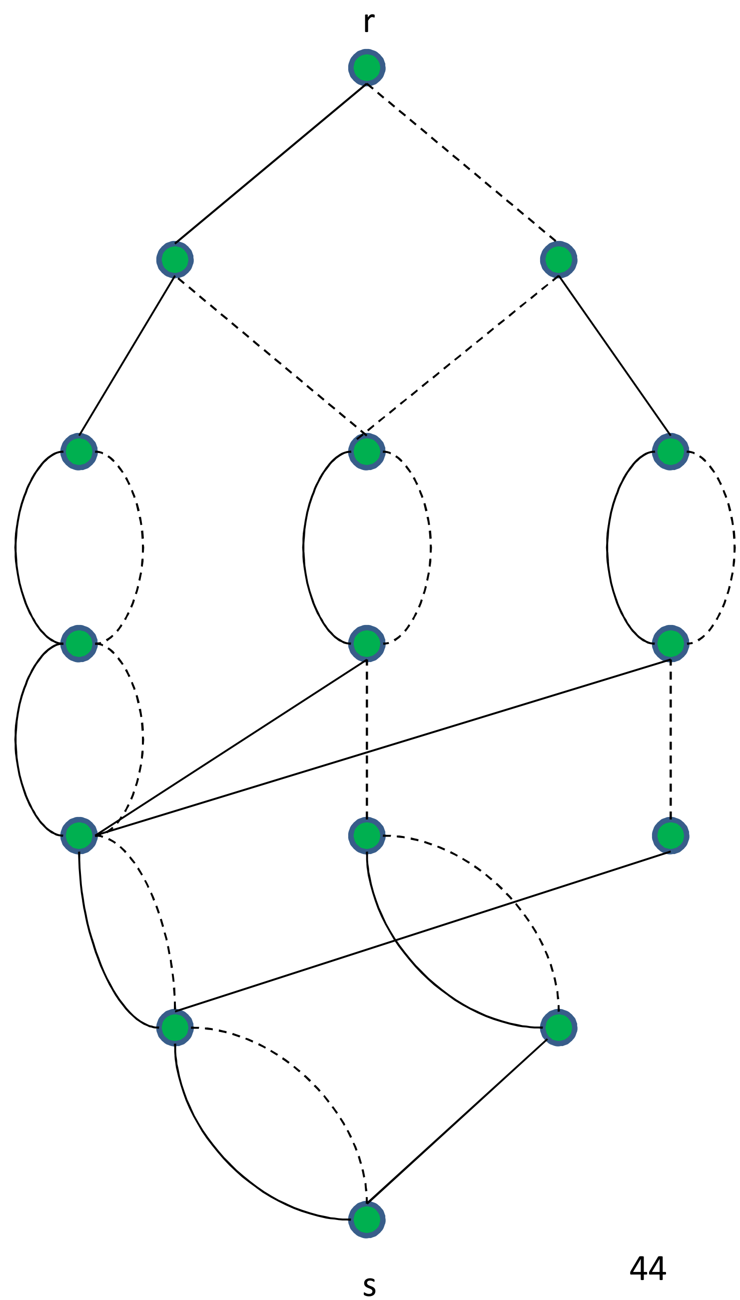
x_3

x_4

x_5

x_6

Relaxation MDD (width $\leq$ 3)

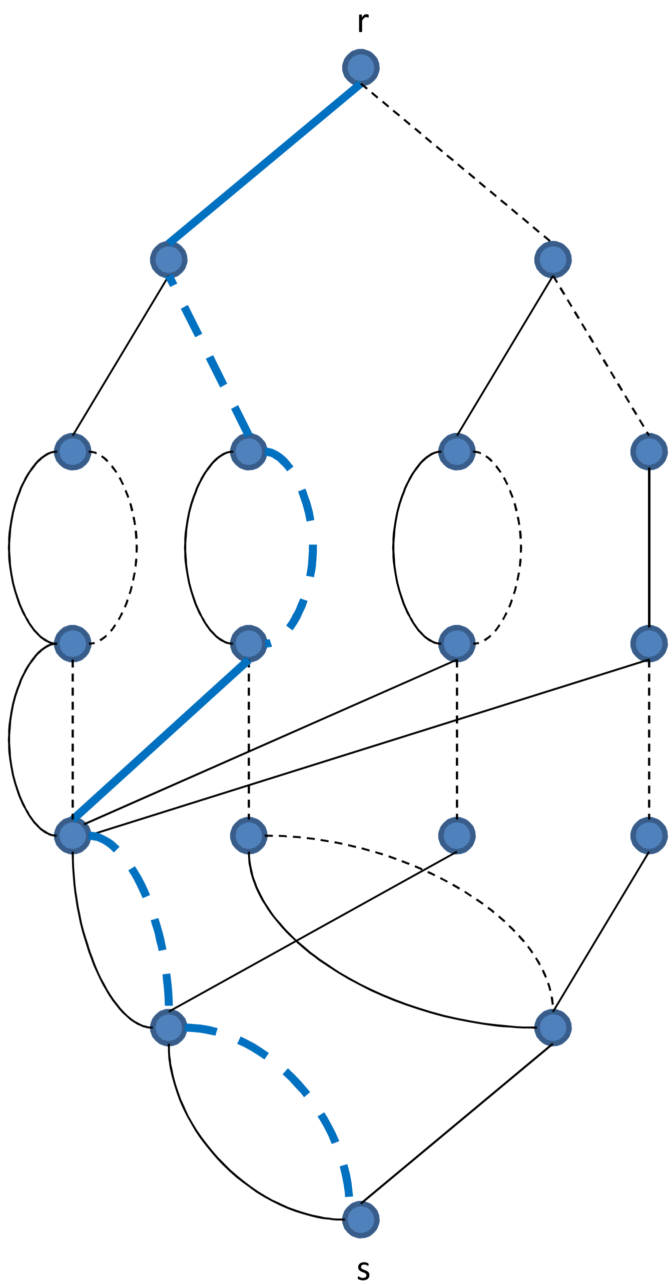


$\min f(x) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$

Exact MDD

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

Relaxation MDD (width $\leq$ 3)



x_1

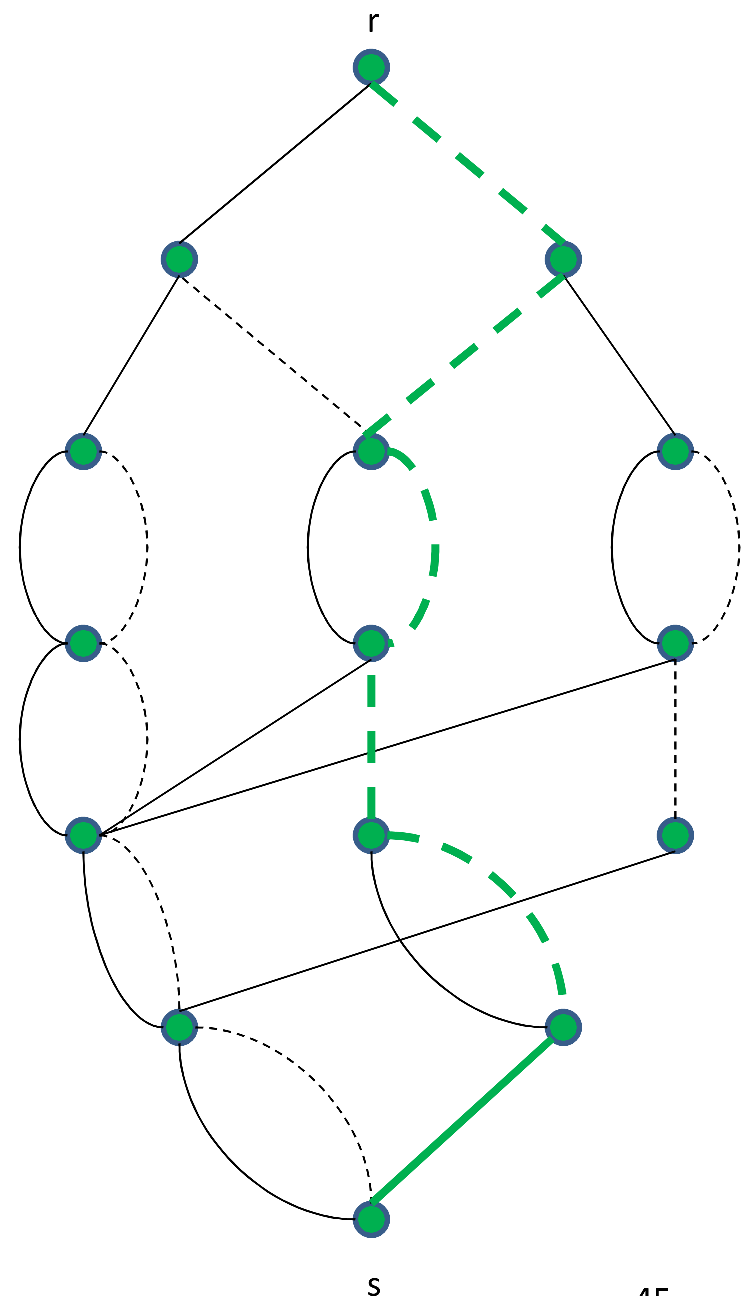
x_2

x_3

x_4

x_5

x_6



$f(x^*) = 2$

$\min f(x) = x_1 + x_2 + x_3 + x_4 + x_5 + x_6$

$f(x^*) = 1$

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

- 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_w
  - Randomly assign a 0 to $b_w - k$ ones in the bandwidth

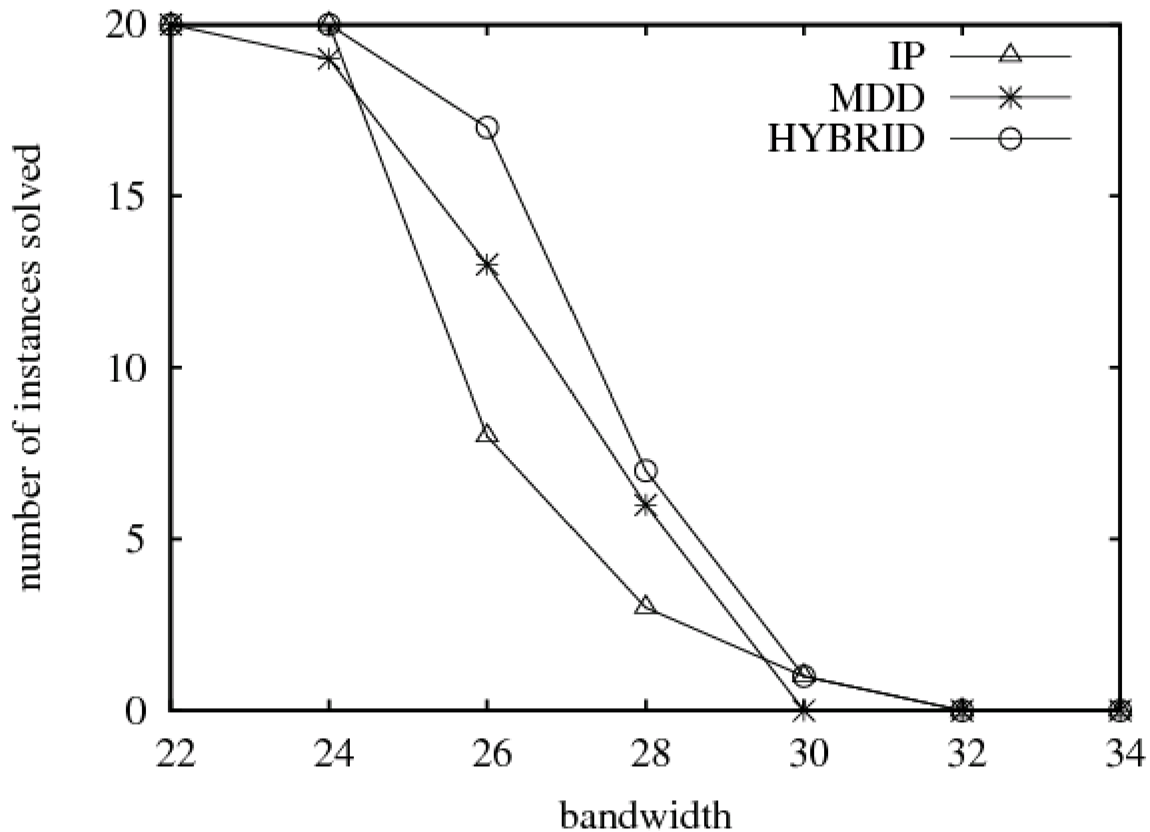
$$
\begin{array}{c} \\ 1\\2\\3\\4 \end{array}
\begin{array}{c} \begin{array}{cccccc} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}
\longrightarrow
\begin{array}{c}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

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

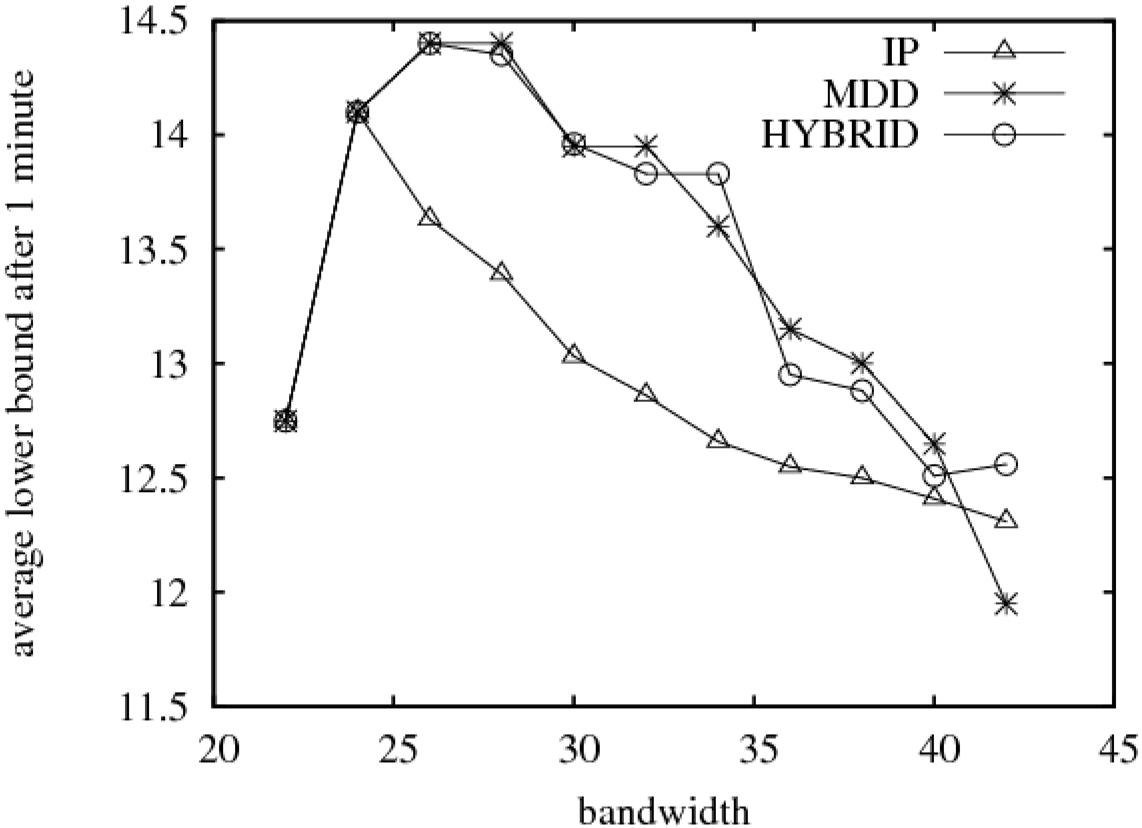
Computational Results

- 250 variables, 20 instances, $k = 20$, $b_w \in \{22,\ldots,44\}$
- 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.)

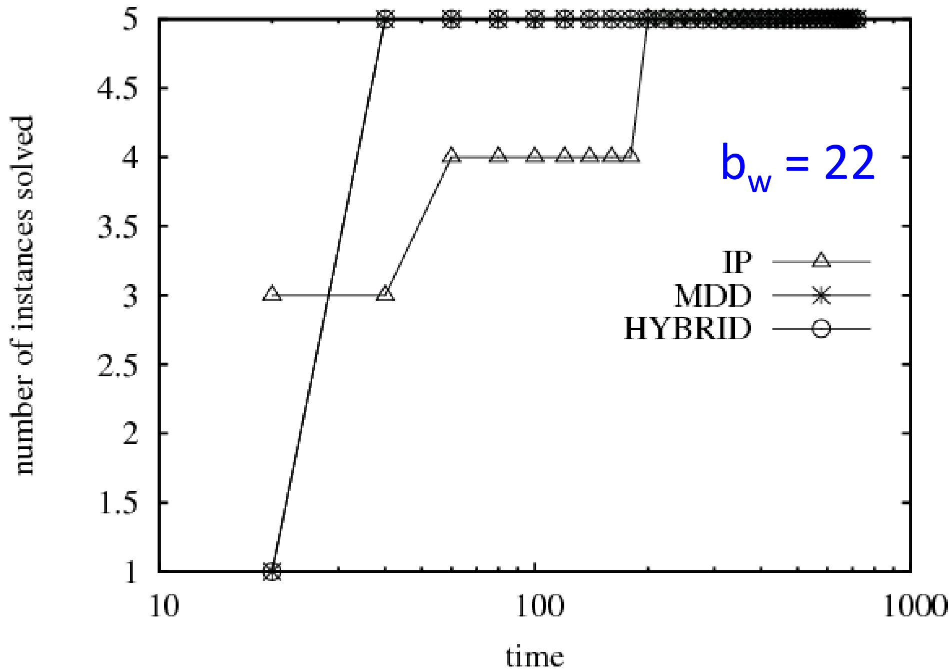


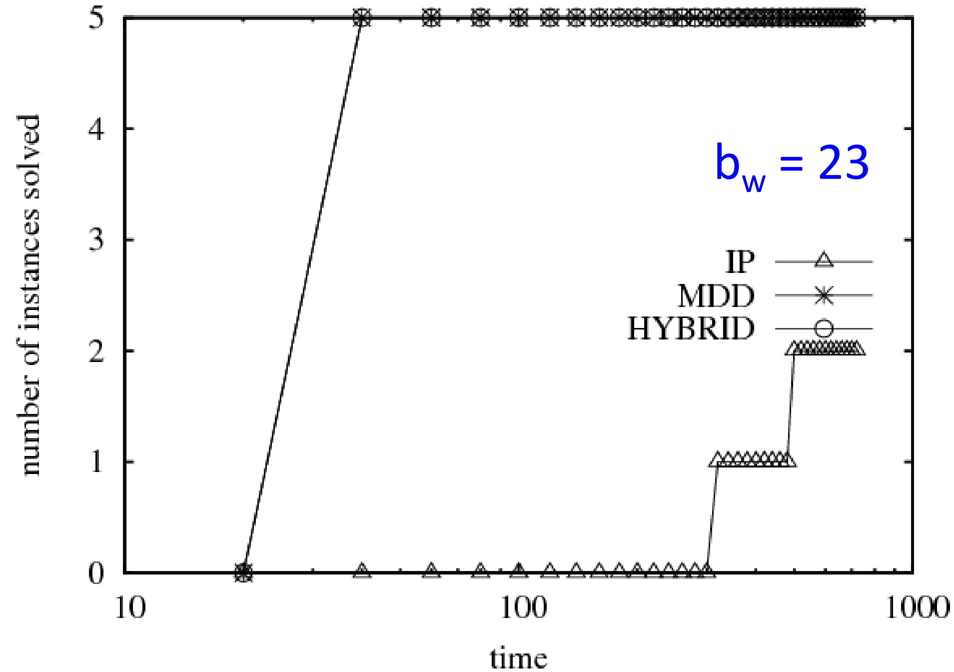
Average Ending Lower Bound (1 min)

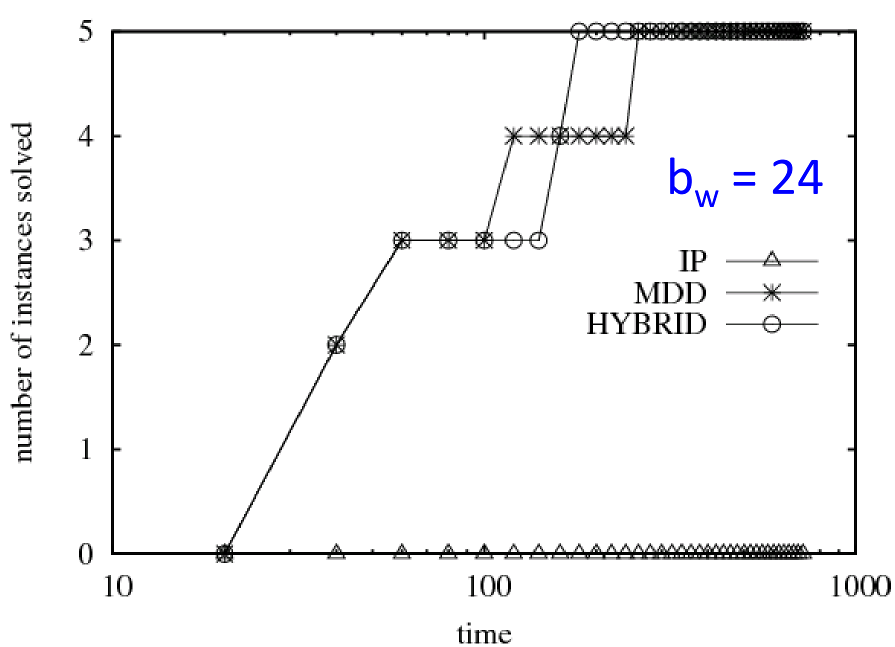


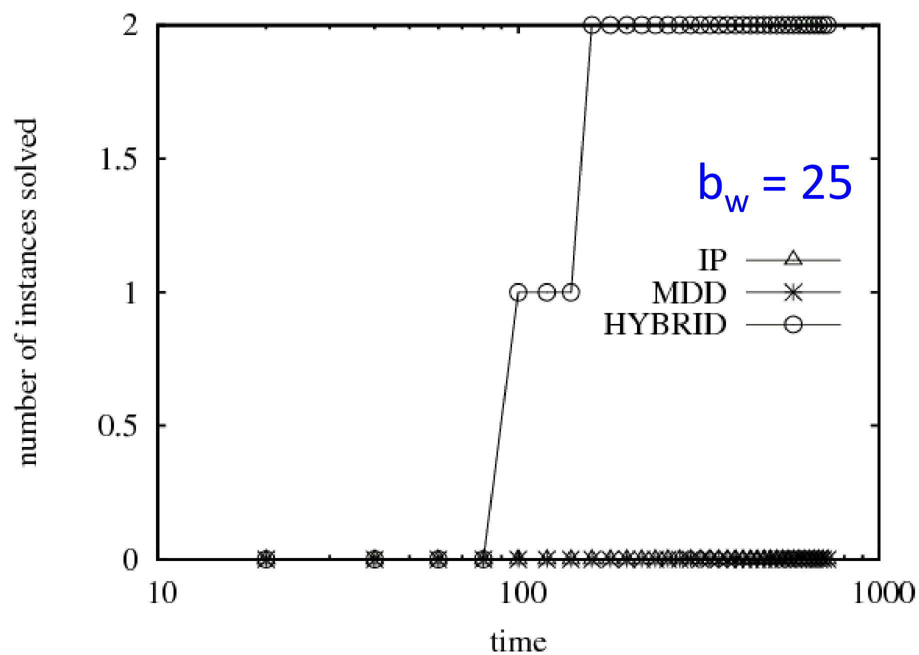
Larger Instances

- 500 variables, 5 instances, k = 20, $b_w \in \{22,\ldots,25\}$


$b_w = 22$
IP
MDD
HYBRID
number of instances solved
time
1
1.5
2
2.5
3
3.5
4
4.5
5
10
100
1000


$b_w = 23$
IP
MDD
HYBRID
number of instances solved
time
0
1
2
3
4
5
10
100
1000


$b_w = 24$
IP
MDD
HYBRID
number of instances solved
time
0
1
2
3
4
5
10
100
1000


$b_w = 25$
IP
MDD
HYBRID
number of instances solved
time
0
0.5
1
1.5
2
10
100
1000

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

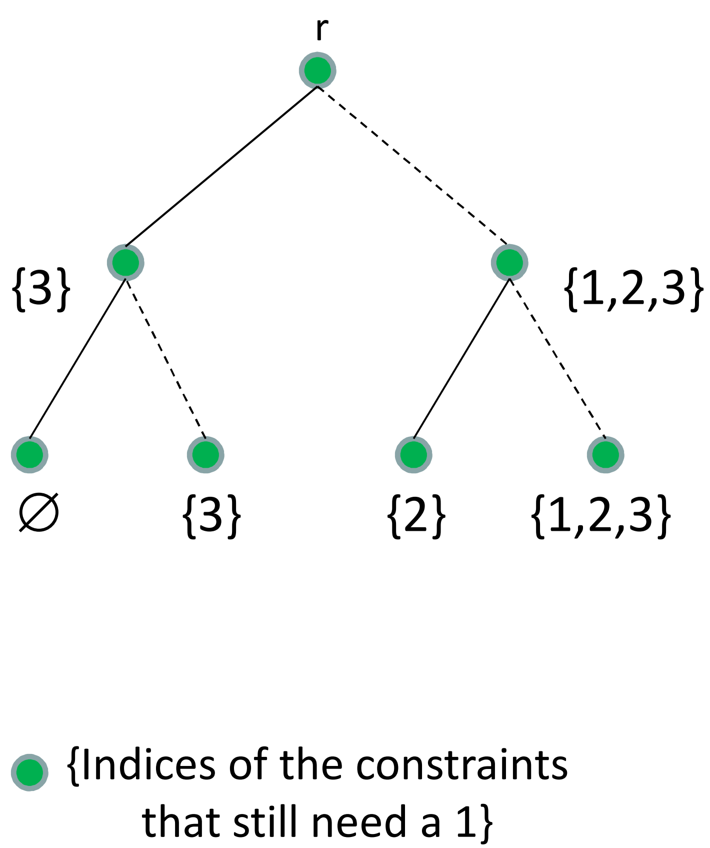
Using an exact top-down compilation method, we can create a limited-width restriction MDD by

1. merging nodes, or
2. deleting nodes

while ensuring that no solution is lost

Node merging by example

Restriction MDD (width ≤ 3)

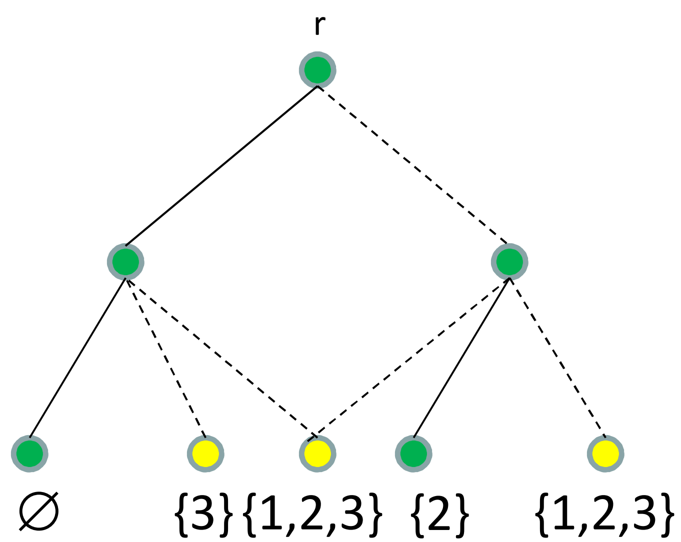


(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 + x_6 \geq 1$

Node merging by example

Restriction MDD (width ≤ 3)

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



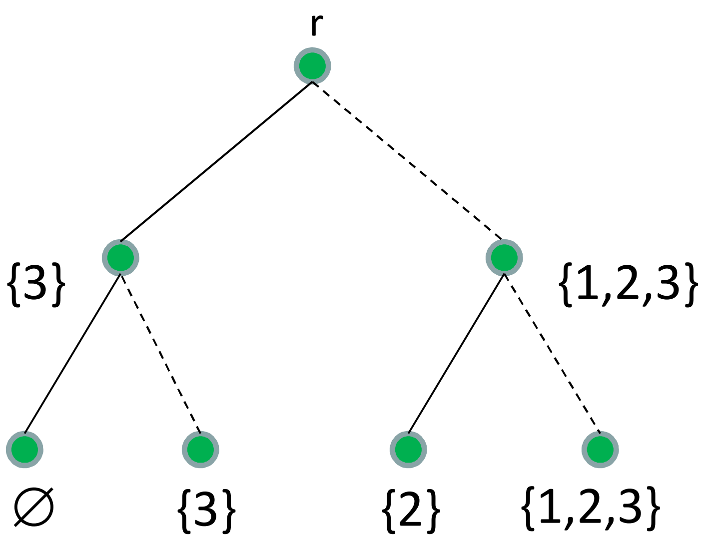
{Indices of the constraints that still need a 1}

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 $f(u_1), f(u_2) \geq f(v)$ 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)
  - for our set covering example: symmetric difference

Node deletion by example

Restriction MDD (width ≤ 3)



● {Indices of the constraints that still need a 1}

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

Node deletion heuristics

- Random
  - select node u uniformly at random
- Objective-driven
  - select node u such that

    $f(u) \geq f(v)$ for all nodes $v \neq u$ in the layer
- Information-driven
  - for set covering: select node u such that

    $I(u) \geq I(v)$ for all nodes $v \neq u$ in the layer

    where $I(u)$ is the set of constraints that still need a 1

Preliminary Experimental Results

- Goals
  - obtain insight in the relative strength of the different restriction heuristics
  - compare to well-known greedy heuristic [Chvátal, 1979]
- Randomly generated set covering instances
  - n variables and m constraints
  - n, m ∈ {25, 50}, with 25 instances per setting
  - unit cost instances and random cost instances (costs are uniform-randomly drawn from {1,...,20}
- MDD widths: 10, 25, 50, 100

Unit costs

n	m	width	Merge rnd	Merge obj	Merge simil	Delete rnd	Delete obj	Delete info
25	25	10	1.47	1.07	1.23	1.36	**1.04**	1.29
		25	1.32	1.03	1.16	1.21	**1.01**	1.17
		50	1.23	**1.00**	1.09	1.17	**1.00**	1.12
		100	1.14	**1.00**	1.08	1.10	**1.00**	1.06
25	50	10	1.41	1.07	1.19	1.34	**1.06**	1.30
		25	1.33	1.05	1.17	1.26	**1.04**	1.26
		50	1.26	1.04	1.15	1.18	**1.03**	1.22
		100	1.23	1.02	1.16	1.15	**1.01**	1.51
50	25	10	1.50	1.10	1.29	1.42	**1.05**	1.29
		25	1.38	1.04	1.19	1.29	**1.03**	1.21
		50	1.29	1.02	1.19	1.24	**1.01**	1.16
		100	1.21	1.01	1.15	1.13	**1.00**	1.10
50	50	10	1.67	1.13	1.27	1.49	**1.10**	1.48
		25	1.60	1.08	1.23	1.43	**1.07**	1.39
		50	1.55	1.05	1.22	1.38	**1.06**	1.33
		100	1.48	**1.04**	1.20	1.33	**1.04**	1.28

averages over
25 instances

compilation time
(obj-based) is
around 0.05s

Random costs

			Merge			Delete		
n	m	width	rnd	obj	simil	rnd	obj	info
25	25	10	1.66	1.12	1.36	1.65	**1.10**	1.53
		25	1.48	**1.04**	1.20	1.37	**1.04**	1.37
		50	1.31	**1.01**	1.14	1.23	**1.01**	1.26
		100	1.17	**1.00**	1.08	1.11	**1.00**	1.16
25	50	10	1.79	1.14	1.41	1.80	**1.13**	1.64
		25	1.60	**1.07**	1.33	1.51	**1.07**	1.52
		50	1.57	**1.04**	1.29	1.41	**1.04**	1.42
		100	1.42	**1.02**	1.17	1.36	**1.02**	1.34
50	25	10	1.76	**1.09**	1.46	1.83	**1.09**	1.51
		25	1.57	**1.02**	1.35	1.63	**1.02**	1.38
		50	1.46	**1.01**	1.28	1.46	**1.01**	1.26
		100	1.30	**1.00**	1.20	1.27	**1.00**	1.15
50	50	10	2.00	1.25	1.55	1.94	**1.21**	1.87
		25	1.87	**1.13**	1.42	1.74	**1.13**	1.72
		50	1.83	**1.09**	1.38	1.68	**1.09**	1.63
		100	1.74	**1.05**	1.30	1.60	**1.05**	1.52

averages over 25 instances

compilation time (obj-based) is around 0.05s

MDDs versus greedy heuristic

			Unit			Random		
n	m	width	MDD+	=	MDD-	MDD+	=	MDD-
25	25	5	20	3	2	8	2	15
		25	21	4	0	13	8	4
		100	21	4	0	15	10	0
25	50	5	23	2	0	6	3	16
		25	24	1	0	11	8	6
		100	24	1	0	19	4	2
50	25	5	15	5	5	8	1	16
		25	18	6	1	20	2	3
		100	21	4	0	23	2	0
50	50	5	16	8	1	4	0	21
		25	21	4	0	9	1	15
		100	24	1	0	18	1	6

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, …