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, Andre Cire, Samid Hoda, John Hooker, Brian Kell, Joris Kinable, Ashish Sabharwal, Horst Samulowitz, Vijay Saraswat, Marla Slusky, 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

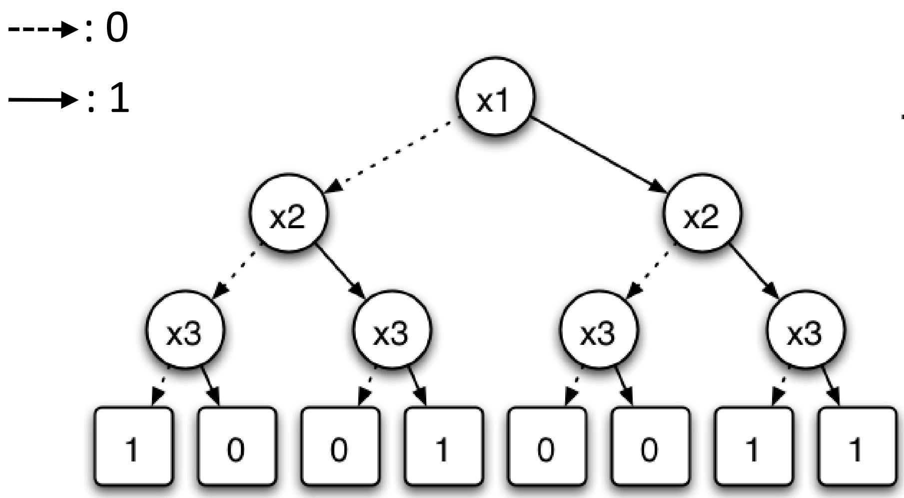
MDDs for integer optimization

- MDD *relaxations* provide upper bounds
- MDD *restrictions* provide lower bounds
- Incorporation in branch-and-bound can be very effective

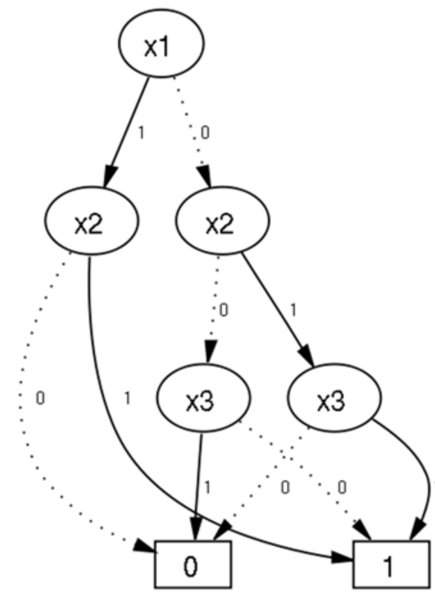
MDDs for constraint programming and scheduling

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

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



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

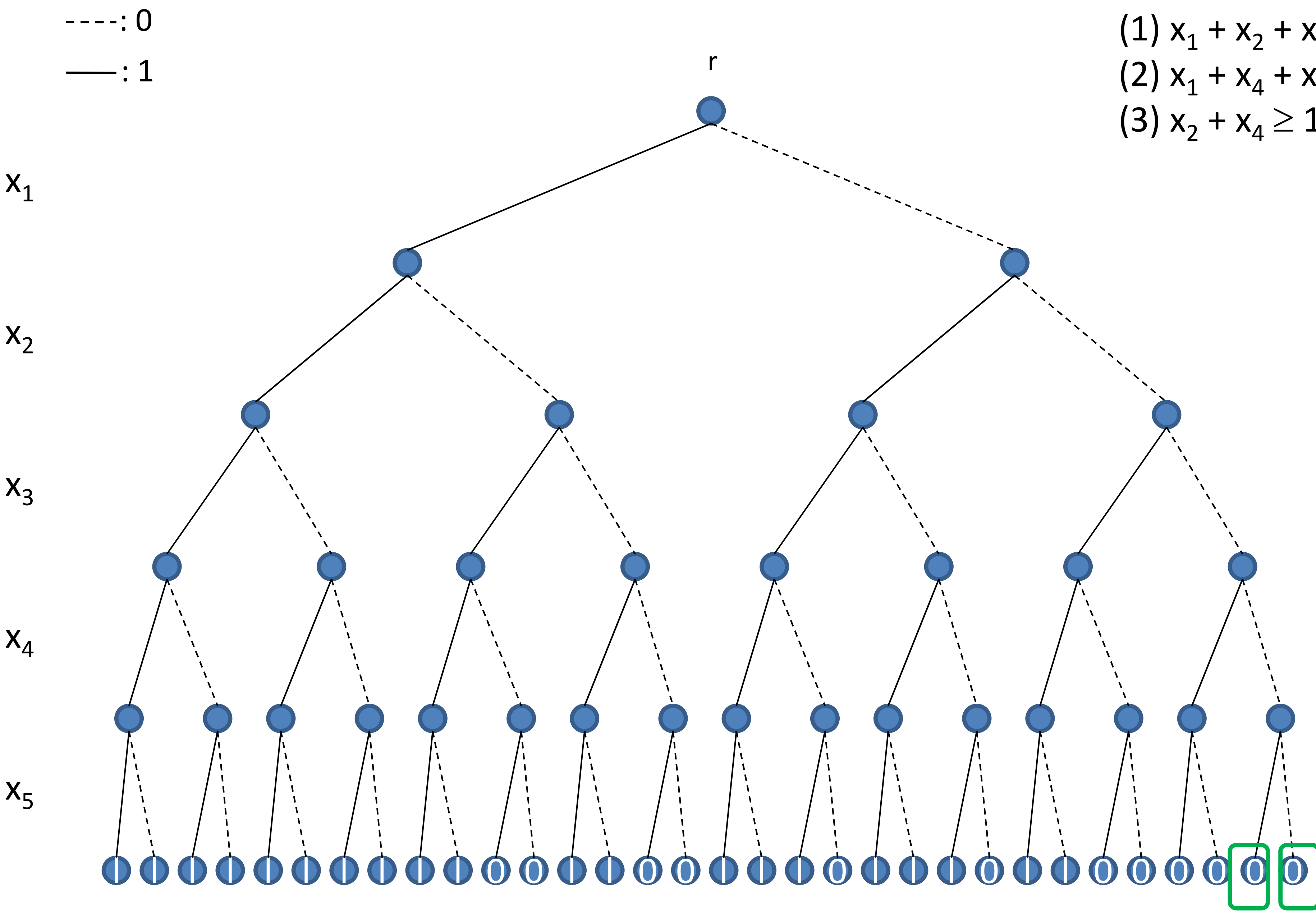
- Binary Decision Diagrams were introduced to compactly represent Boolean functions [Lee, 1959], [Akers, 1978], [Bryant, 1986]
- BDD: merge isomorphic subtrees of a given binary decision tree
- MDDs are multi-valued decision diagrams (i.e., for discrete variables)

Brief background

- 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

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

Exact MDDs for discrete optimization



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

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

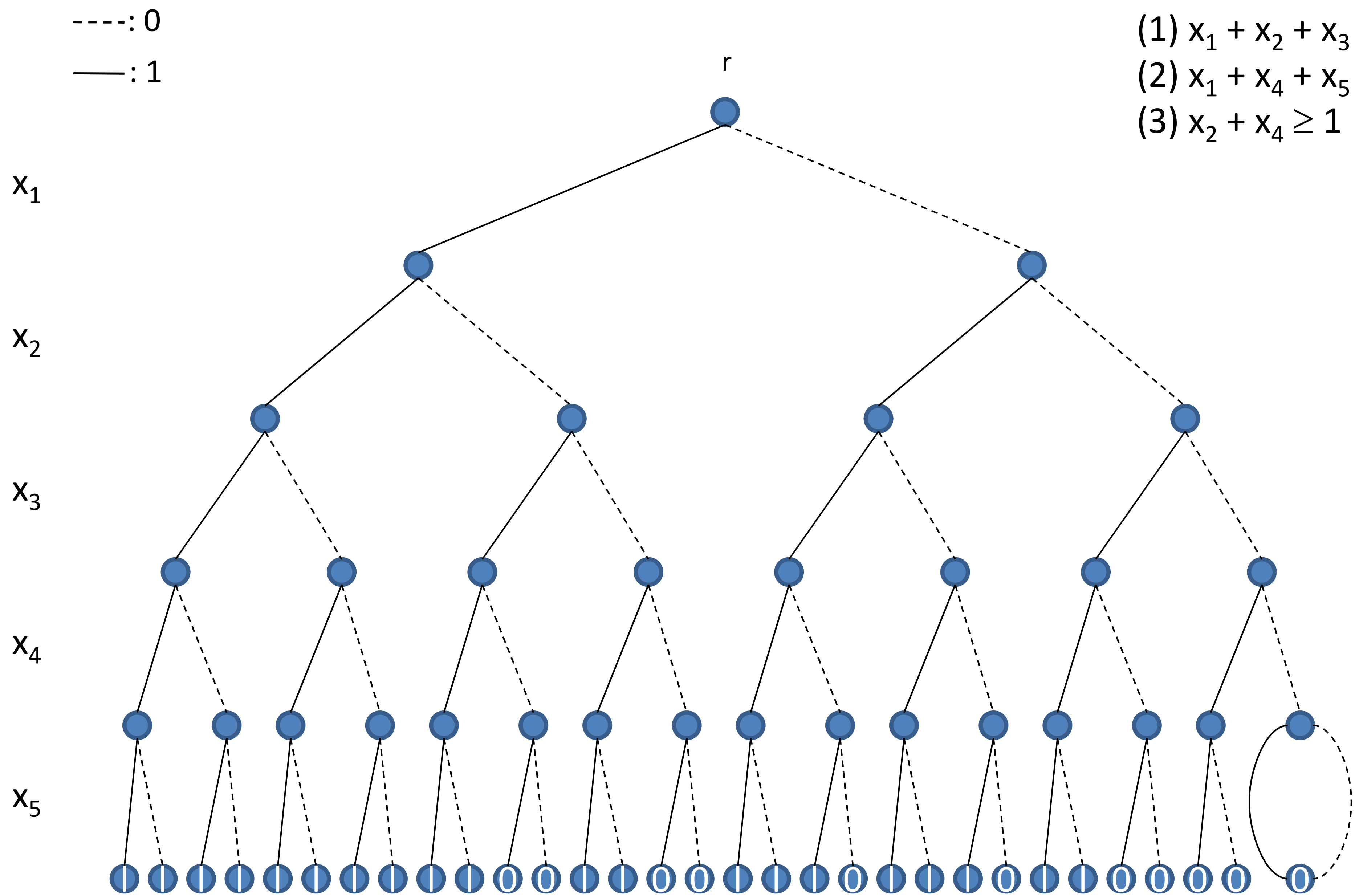
(3) $x_2 + x_4 \geq 1$

Exact MDDs for discrete optimization

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

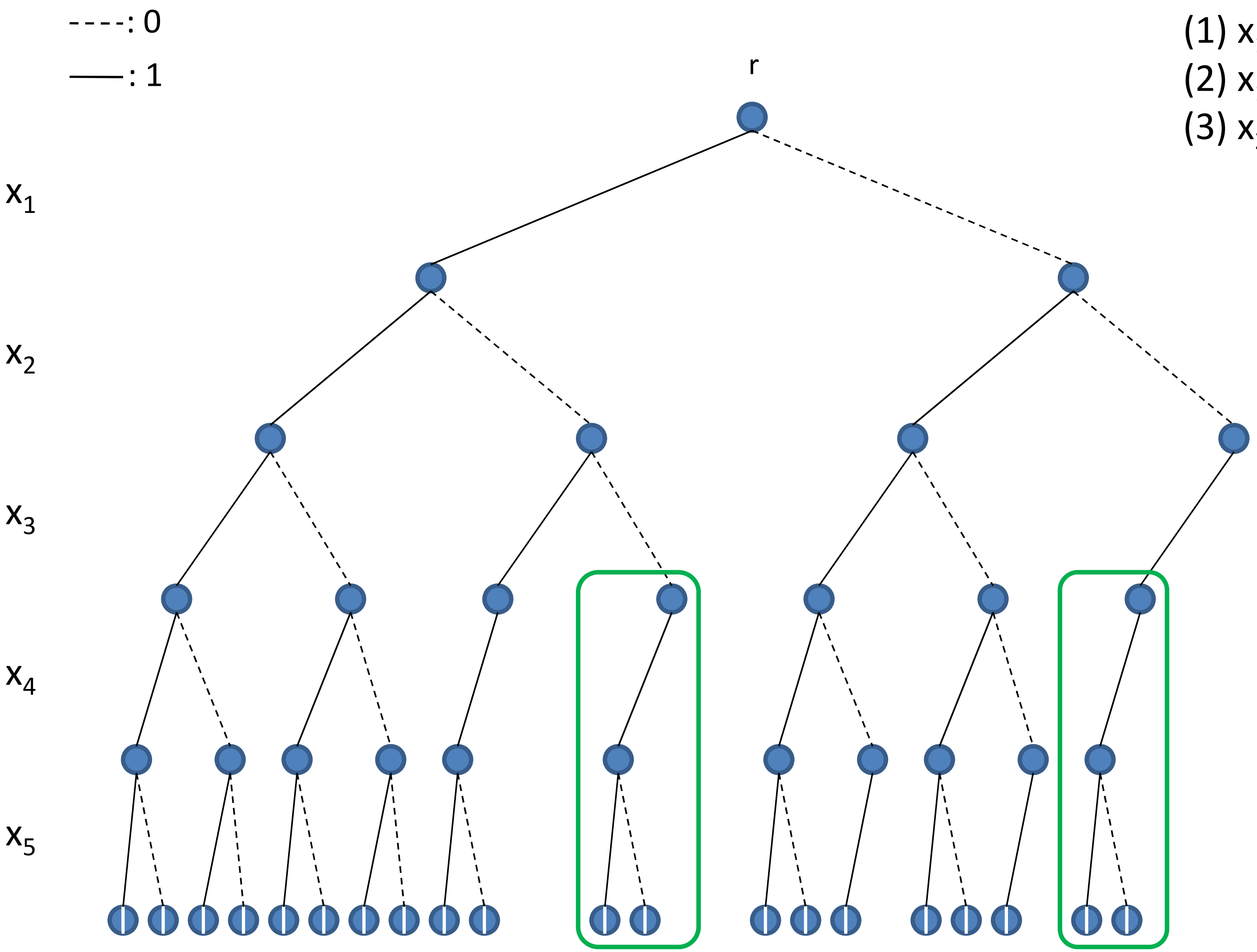
$(2)\ x_1 + x_4 + x_5 \geq 1$

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

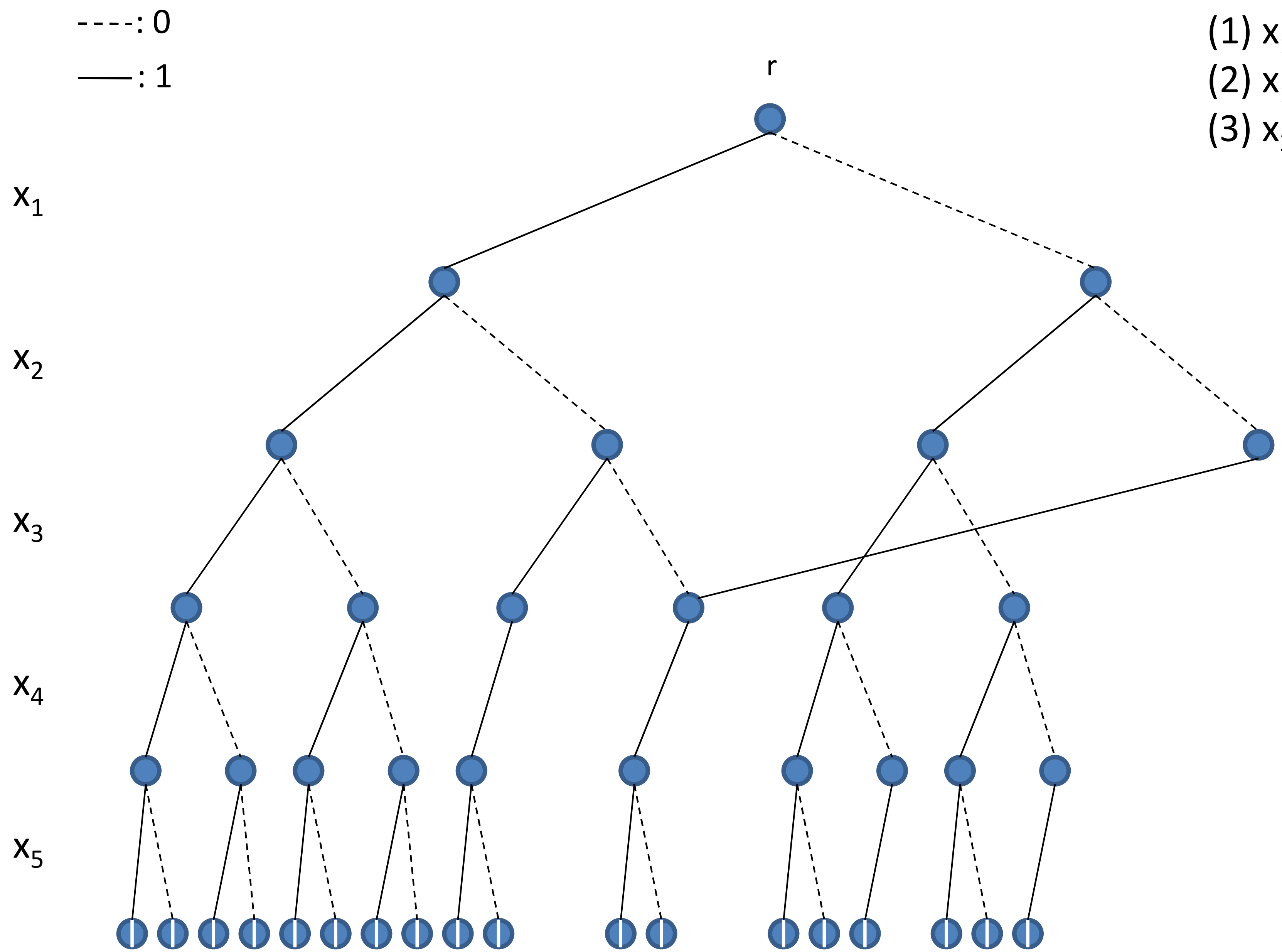


Exact MDDs for discrete optimization

(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



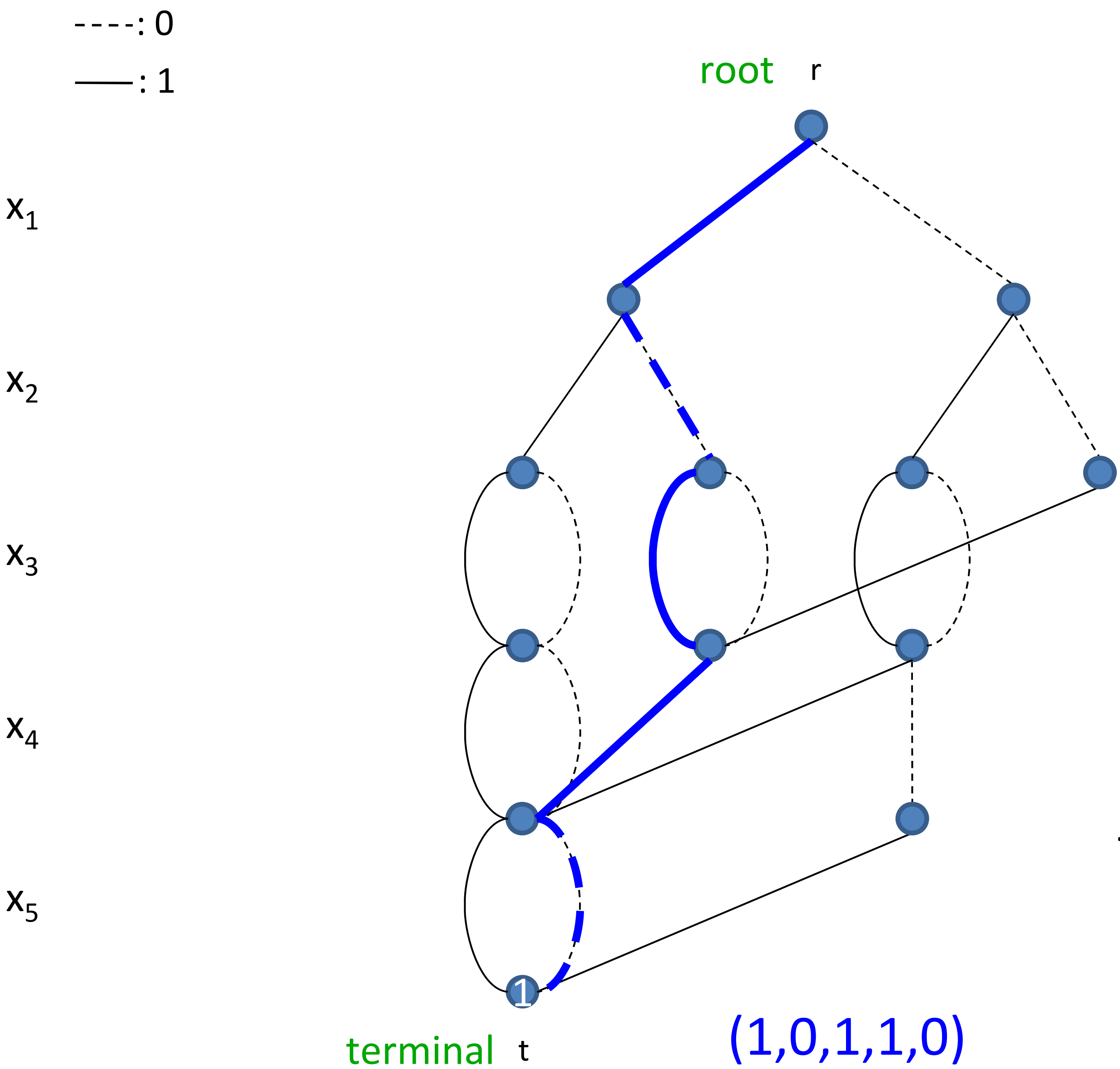
(1) $x_1 + x_2 + x_3 \geq 1$

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

(3) $x_2 + x_4 \geq 1$

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
- We can limit the size (width) of the MDD to obtain a relaxation [Andersen et al., 2007]
  - strength is controlled by the width
- Can provide bounds on objective function
- Can also be used for cut generation, constraint propagation, guiding search, …

MDDs for Integer Optimization

- Bergman, Cire, v.H., Hooker: Optimization Bounds from Binary Decision Diagrams. *INFORMS J. Computing* 26(2): 253-268, 2014.
- Bergman, Cire, v.H., Yunes: BDD-Based Heuristics for Binary Optimization. *Journal of Heuristics* 20: 211-234, 2014.
- Bergman, Cire, v.H., Hooker. Discrete Optimization with Decision Diagrams. *Under review*, 2013.
- Bergman, Cire, Sabharwal, Samulowitz, Saraswat, and v.H. Parallel Combinatorial Optimization with Decision Diagrams. In *Proceedings of CPAIOR*, Springer LNCS, 2014.

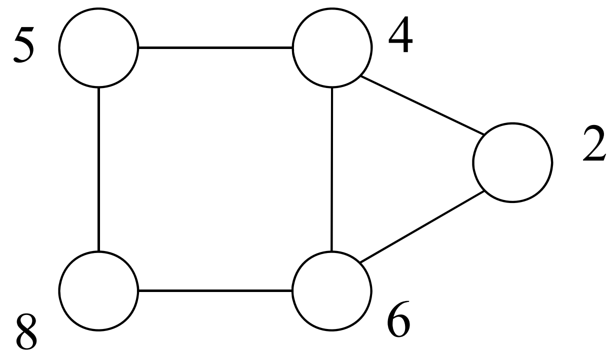
Motivation

- Conventional integer programming relies on branch-and-bound based on continuous LP relaxations
  - Relaxation bounds
  - Feasible solutions
  - Branching
- We propose a novel branch-and-bound algorithm for discrete optimization based on decision diagrams
  - Relaxation bounds – **Relaxed BDDs**
  - Feasible solutions – **Restricted BDDs**
  - Branching – **Nodes** of relaxed BDDs
- Potential benefits: stronger bounds, efficiency, memory requirements, models need not be linear

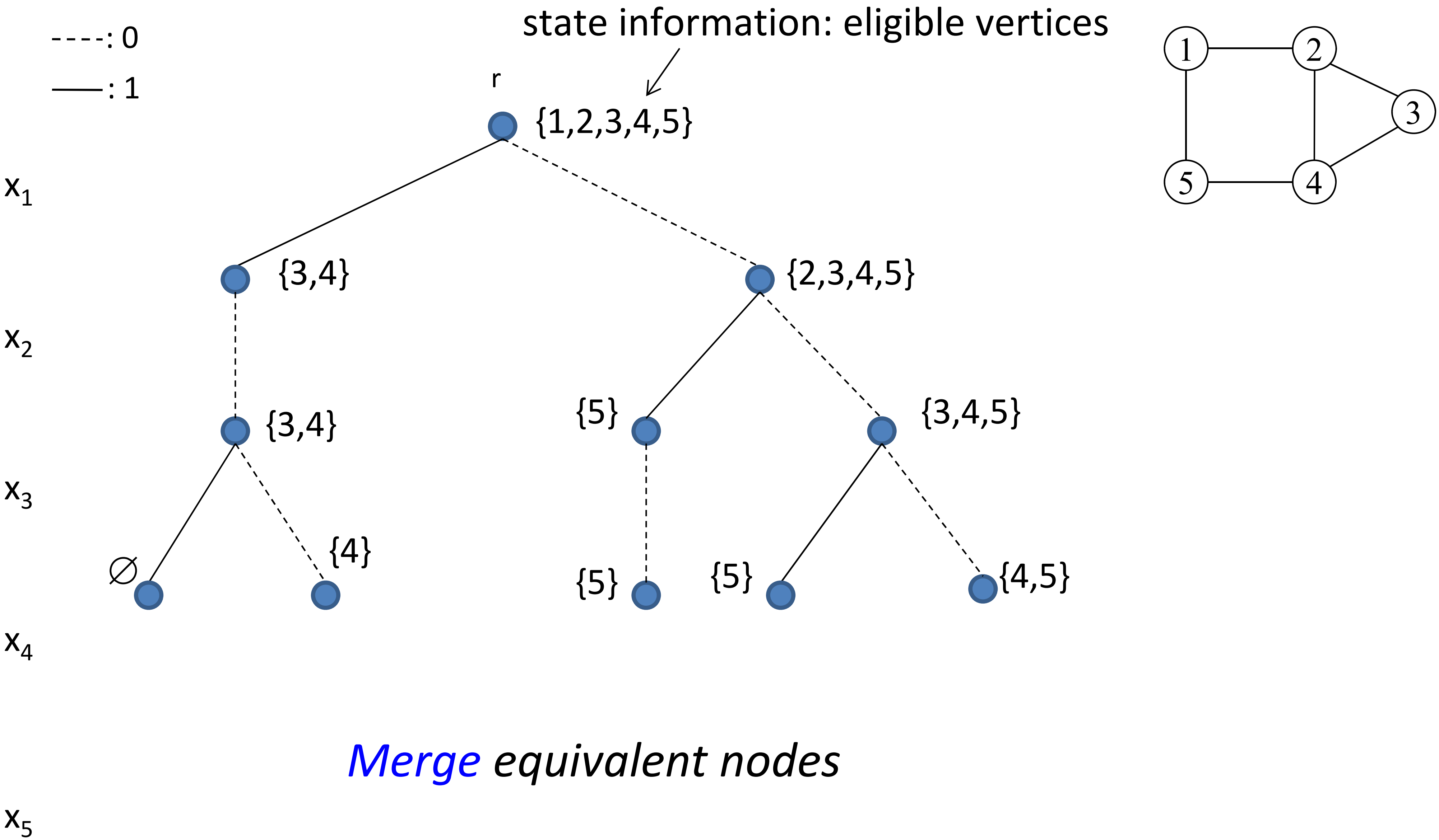
Case Study: Independent Set Problem

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

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

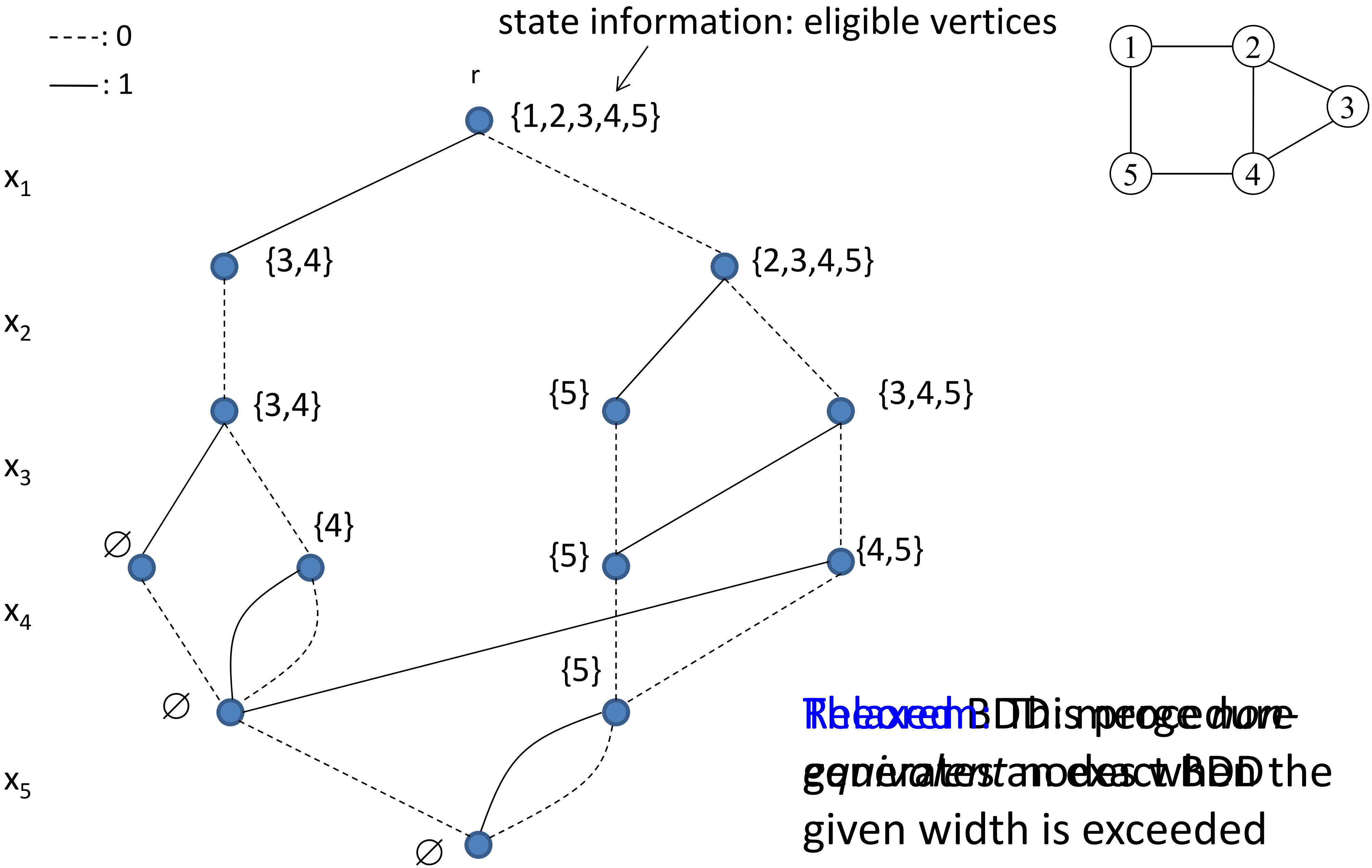


Exact top-down compilation



Merge equivalent nodes

Node Merging

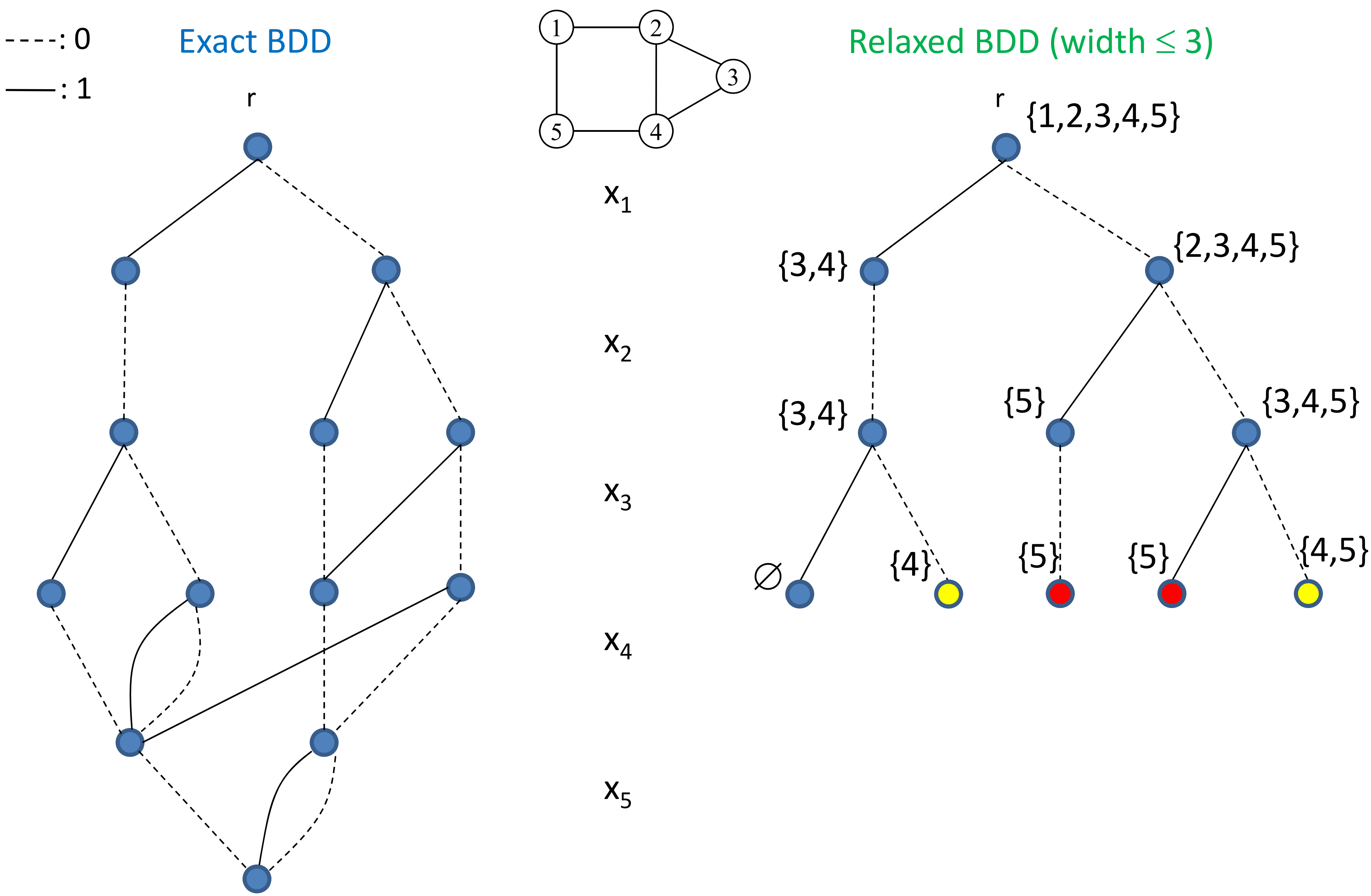


Relaxed BDD: merge non-*equivalent* nodes when the given width is exceeded

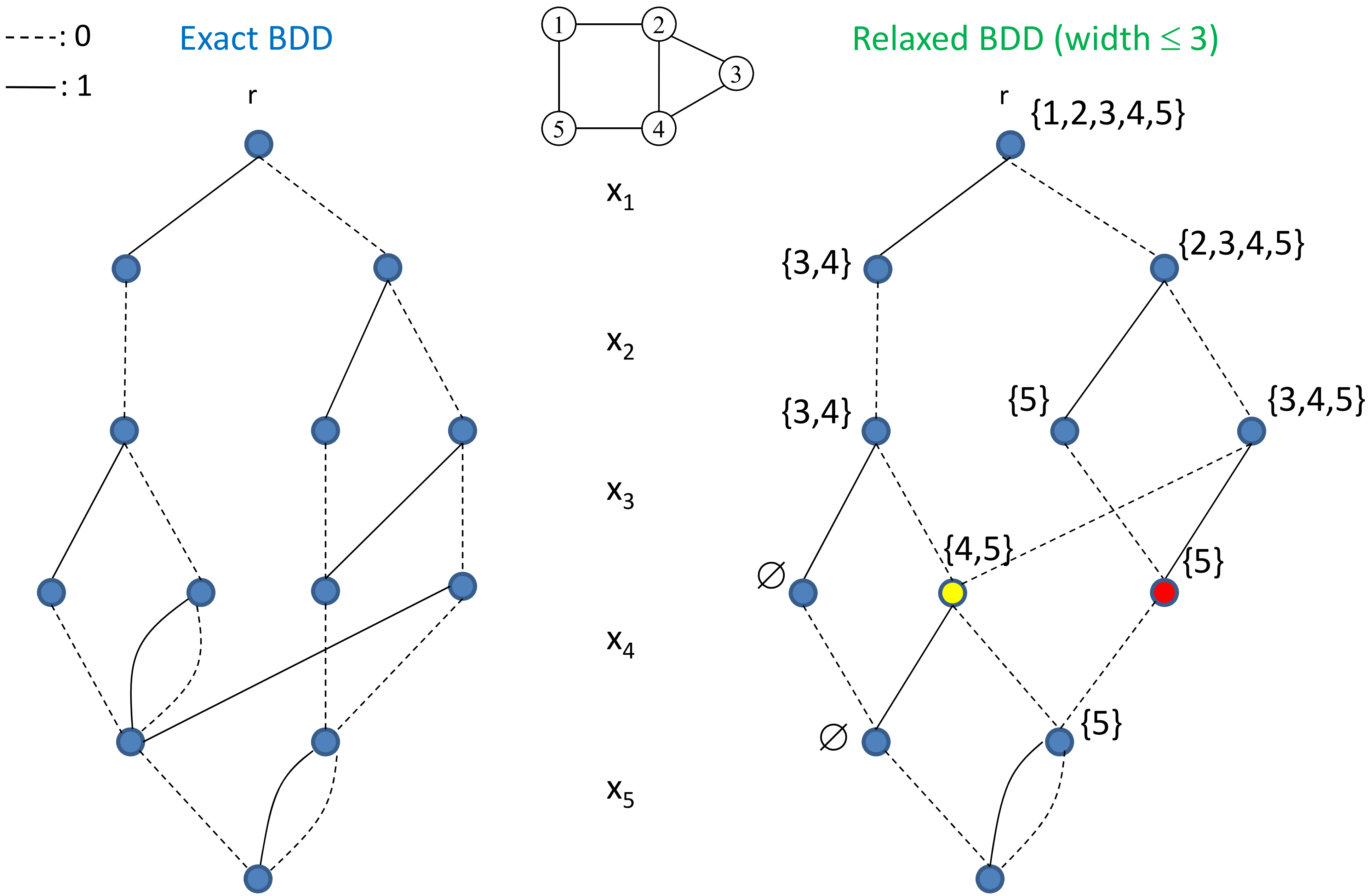
Theorem: This procedure generates an exact BDD

[Bergman et al., 2012]

Relaxed BDD

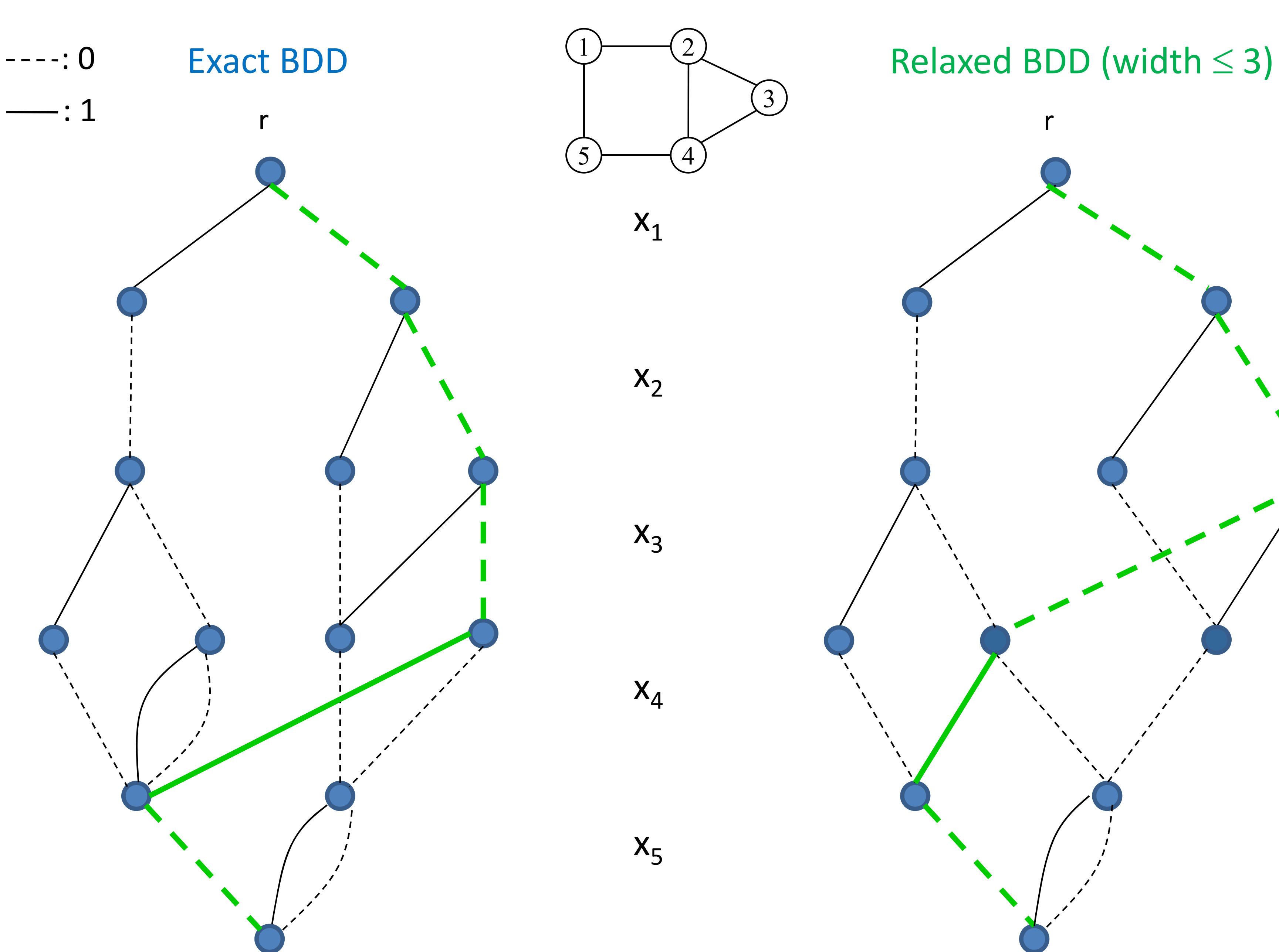

----: 0
——: 1
Exact BDD
r
1
2
3
4
5
x_1
x_2
x_3
x_4
x_5
Relaxed BDD (width ≤ 3)
r
{1,2,3,4,5}
{3,4}
{2,3,4,5}
{3,4}
{5}
{3,4,5}
∅
{4}
{5}
{5}
{4,5}

Relaxed BDD



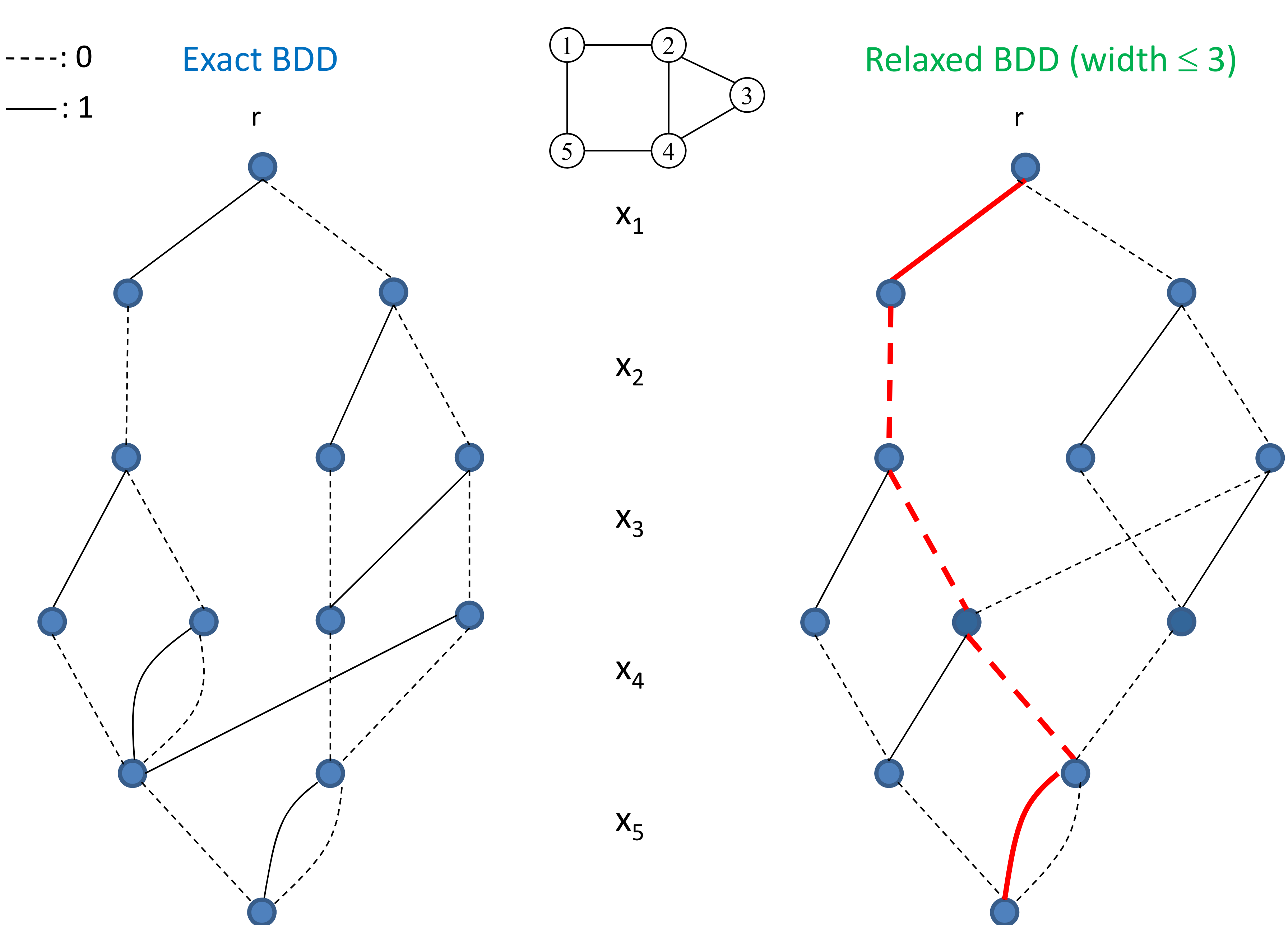
- - - -: 0
——: 1
Exact BDD
Relaxed BDD (width ≤ 3)
r
1
2
3
4
5
x1
x2
x3
x4
x5
{1,2,3,4,5}
{3,4}
{2,3,4,5}
{3,4}
{5}
{3,4,5}
∅
{4,5}
{5}
∅
{5}

Relaxed BDD



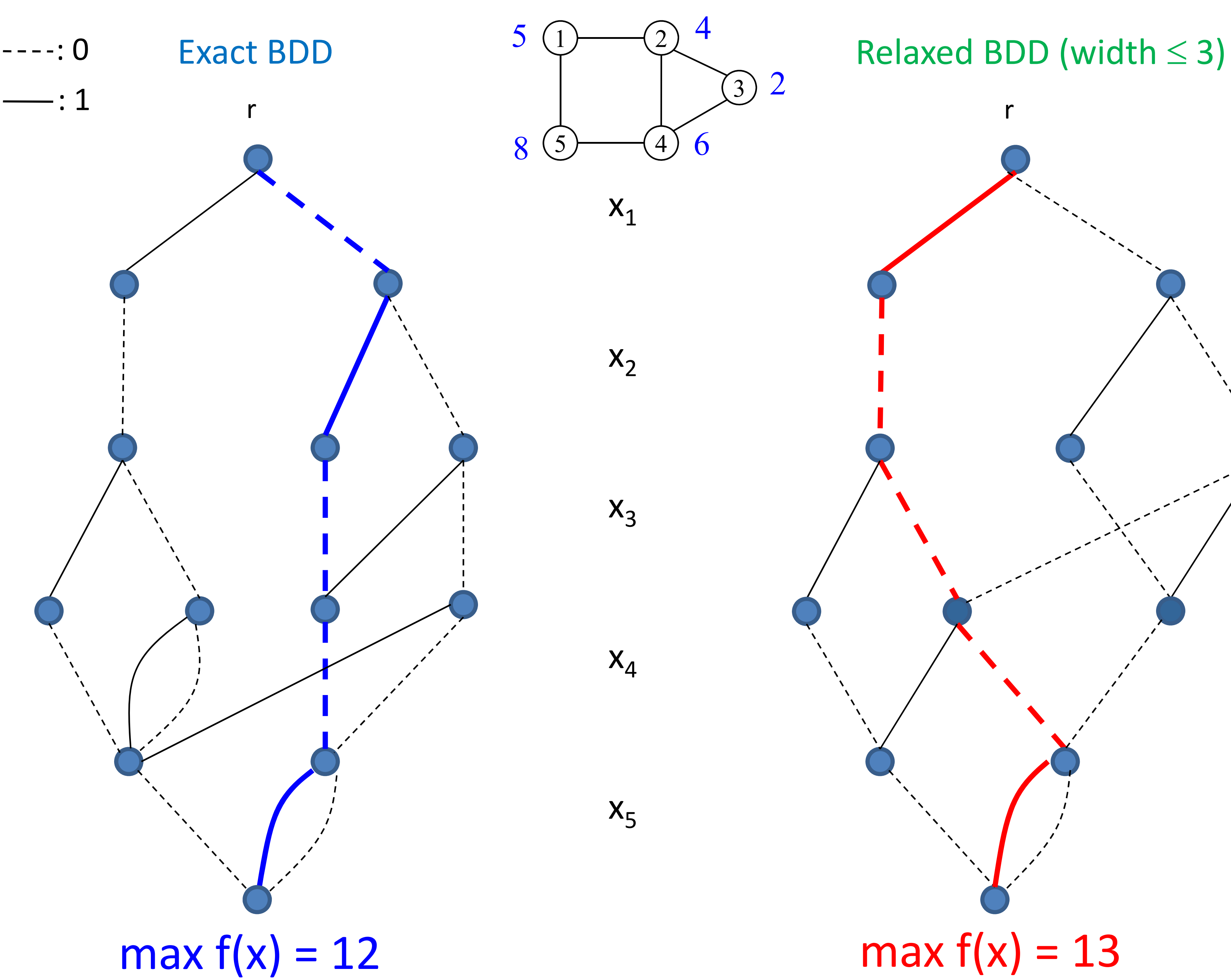
(0,0,0,1,0)

Relaxed BDD



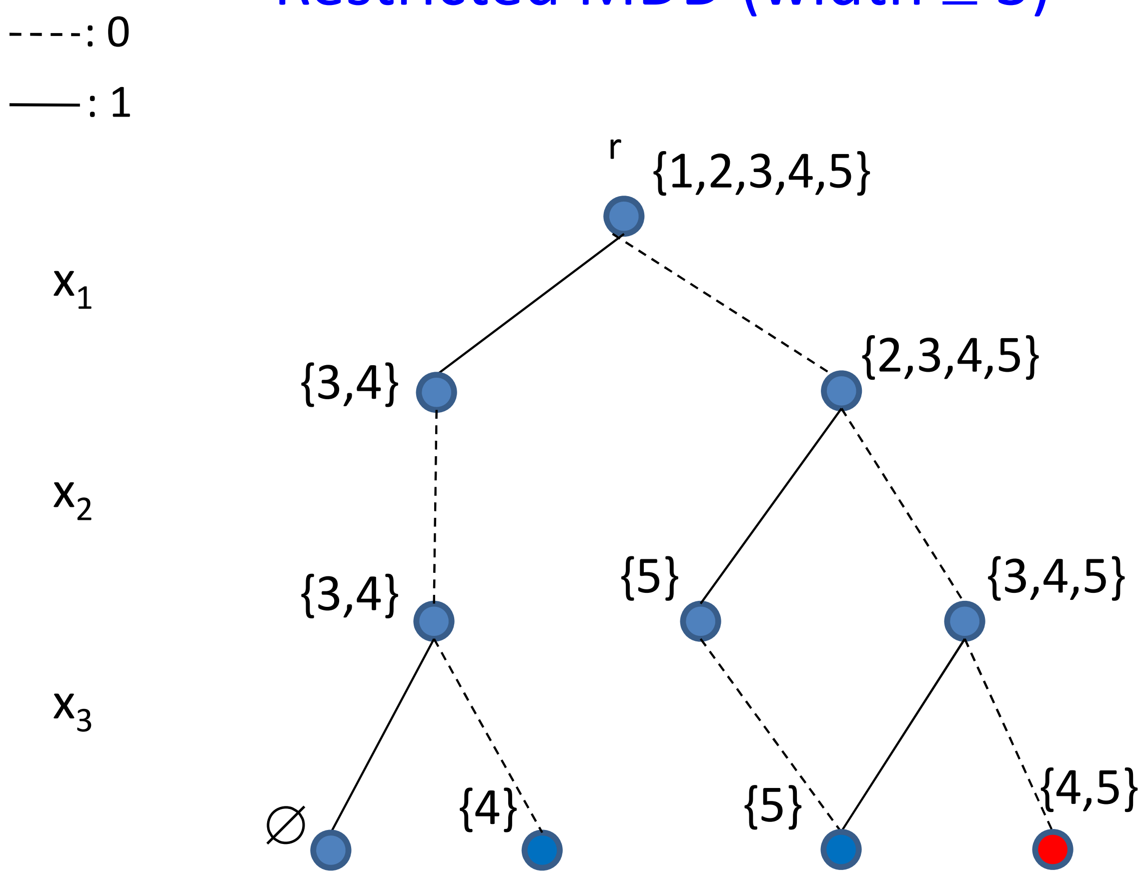
(1,0,0,0,1)

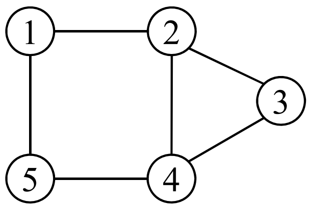
Evaluate Objective Function



Restricted BDD

Restricted MDD (width $\leq$ 3)



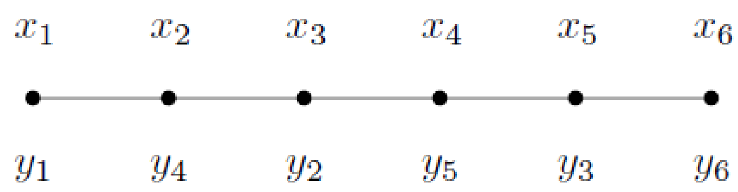


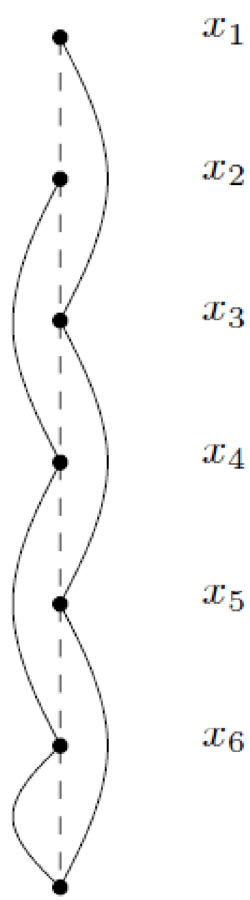
Variable Ordering

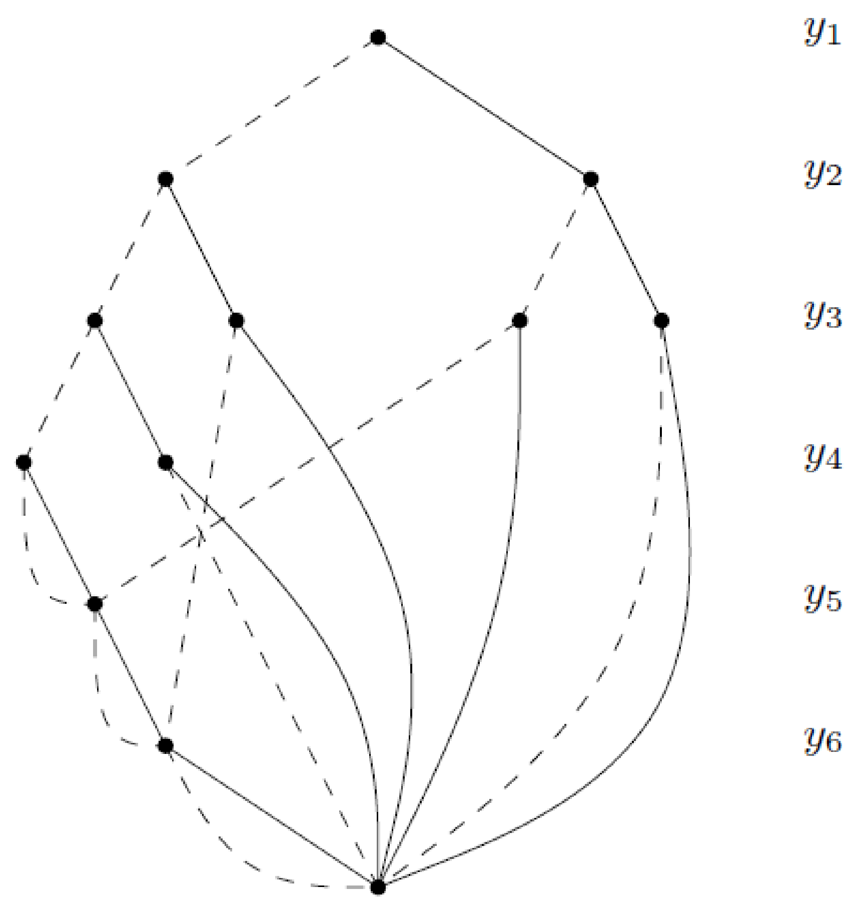
- Order of variables greatly impacts BDD size
  - also influences bound from relaxed BDD (see next)
- Finding ‘optimal ordering’ is NP-hard

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

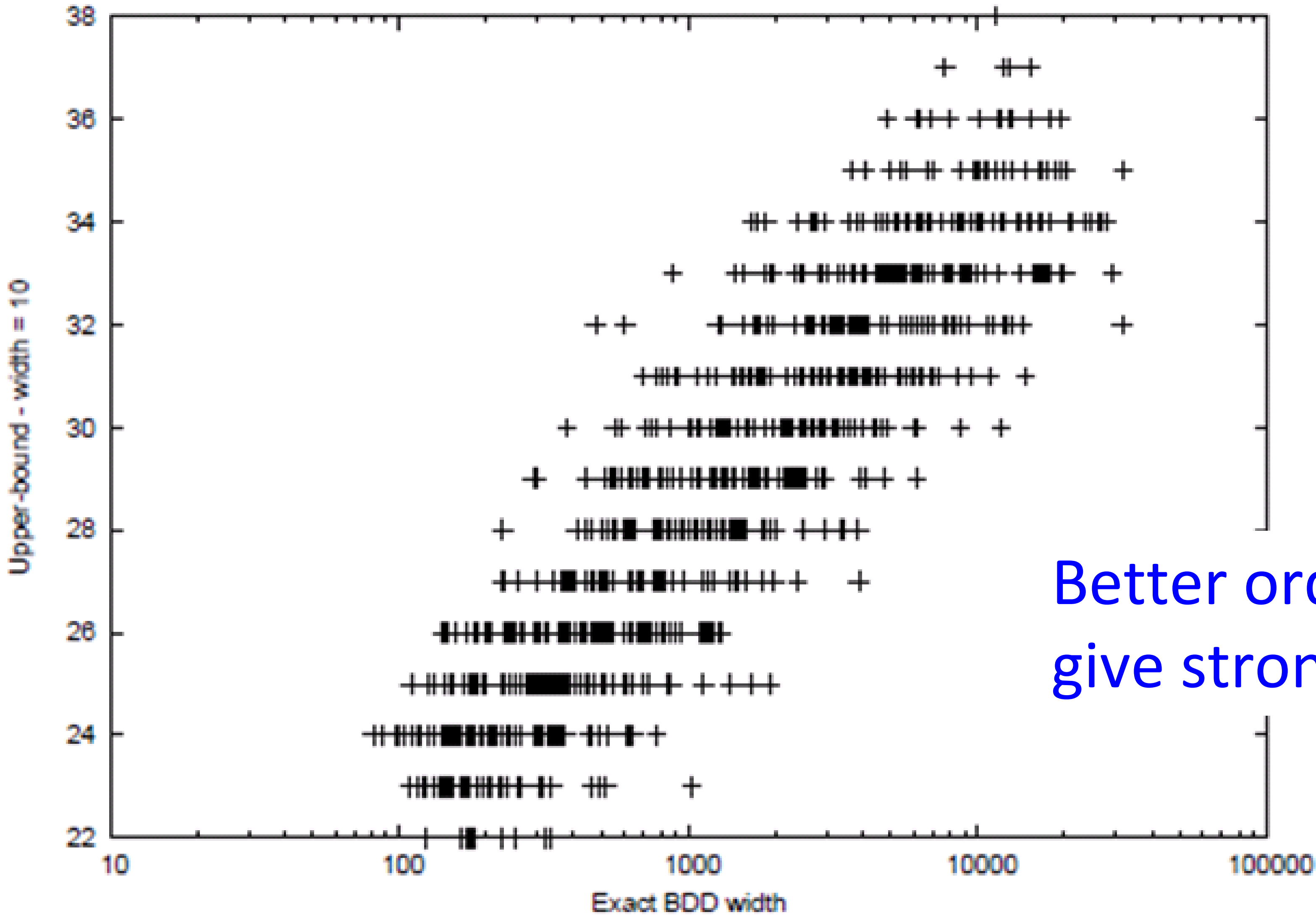
Exact BDD orderings for Paths







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

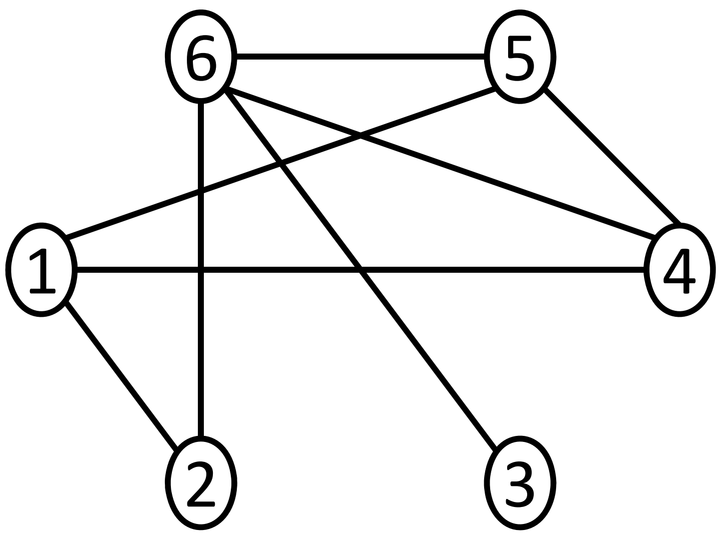
Formal Results for Independent Set

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

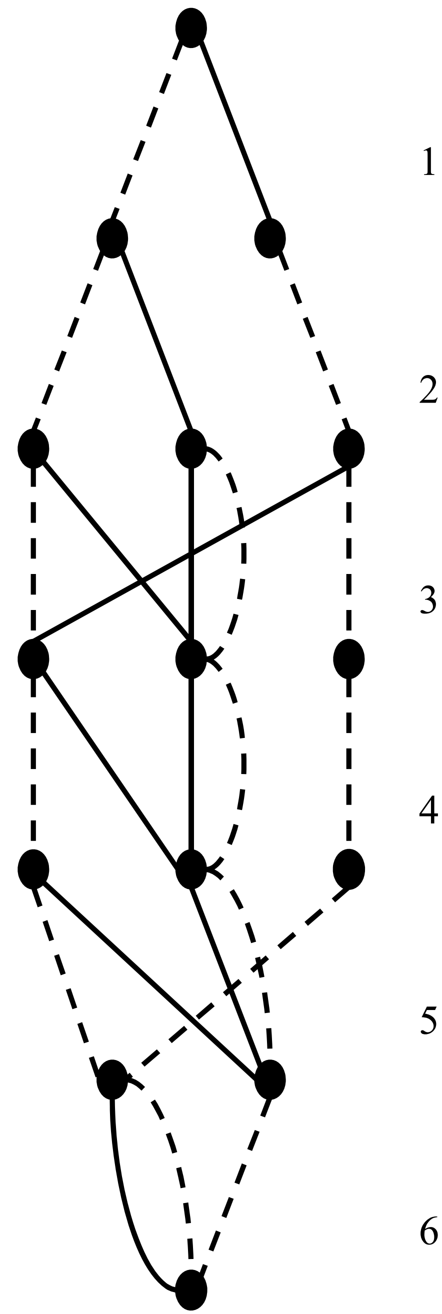
(The proof for general graphs is based on a maximal path decomposition of the graph)

Branch and Bound

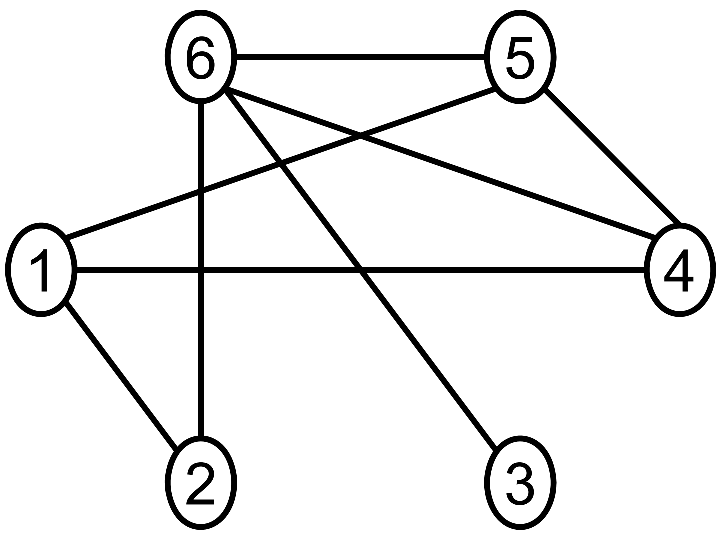
Width 3 relaxed decision diagram

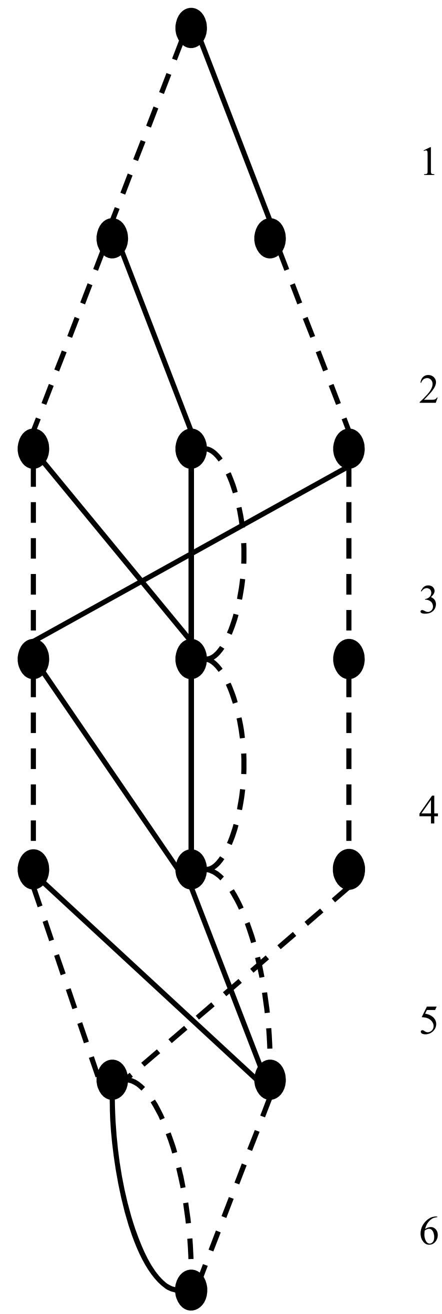


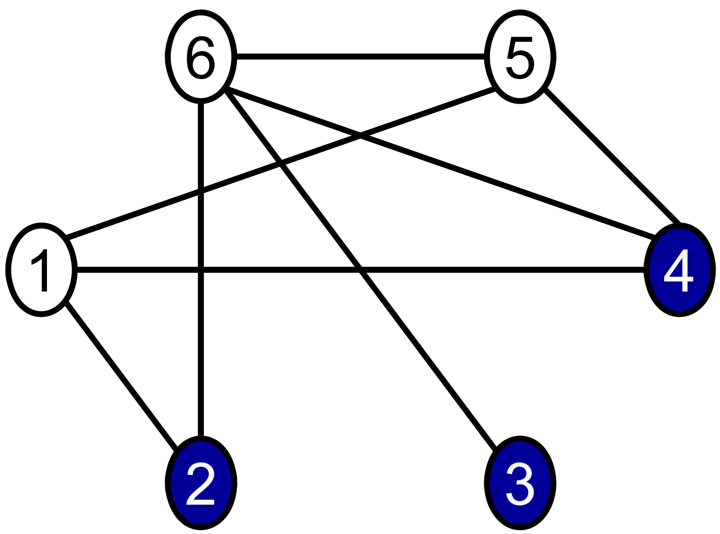
Upper Bound = 4

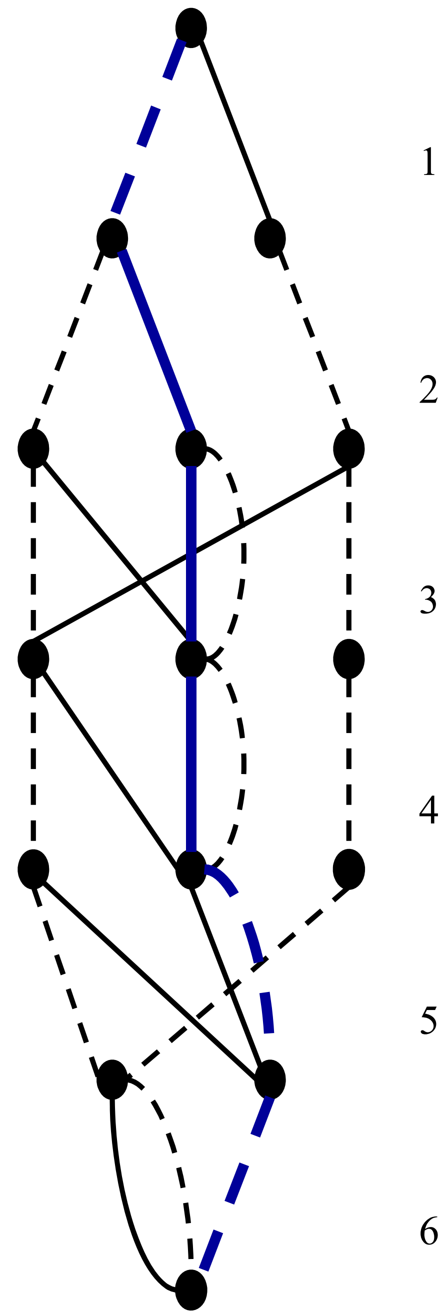


Branch and Bound

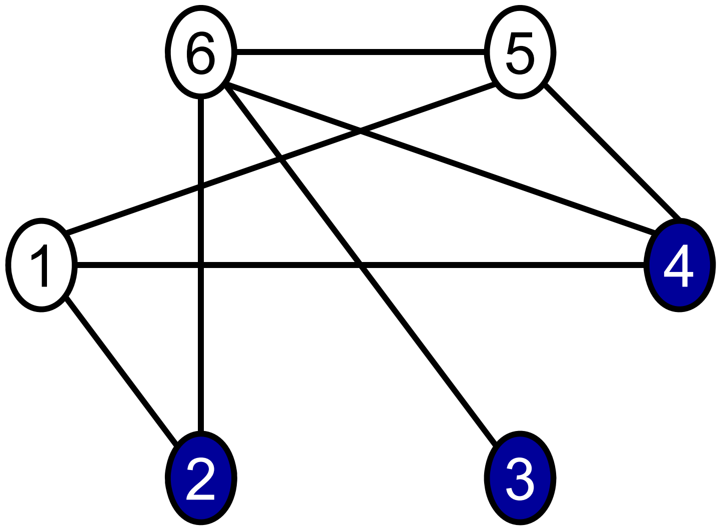


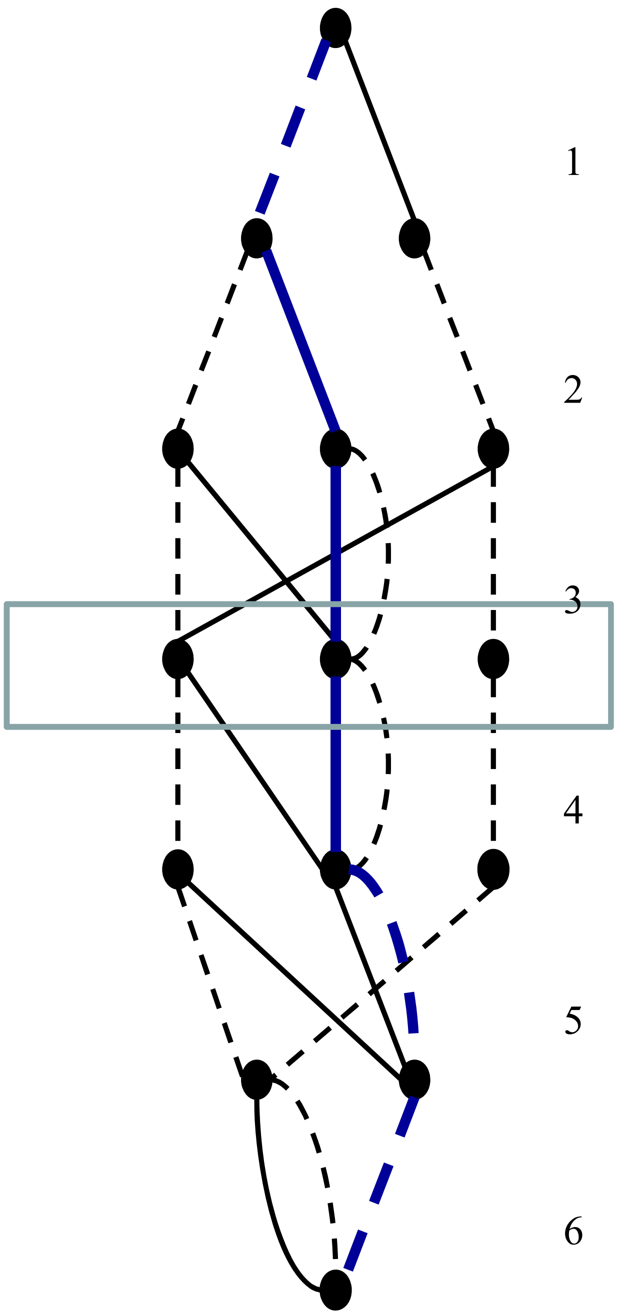



6
5
1
4
2
3

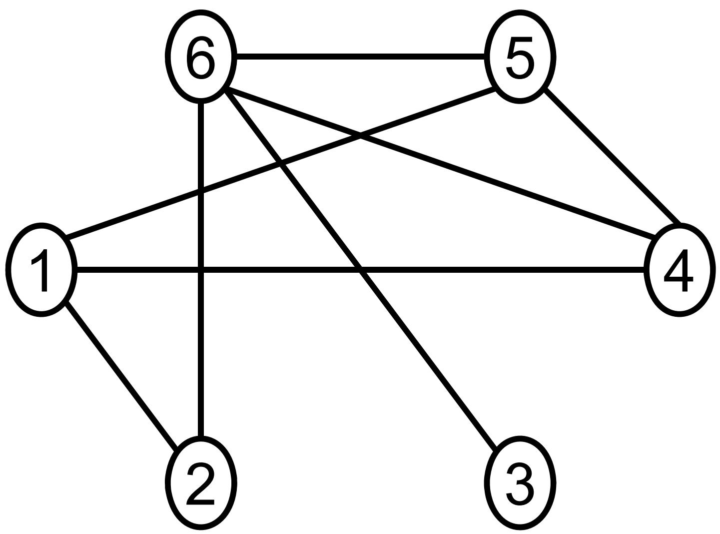

1
2
3
4
5
6

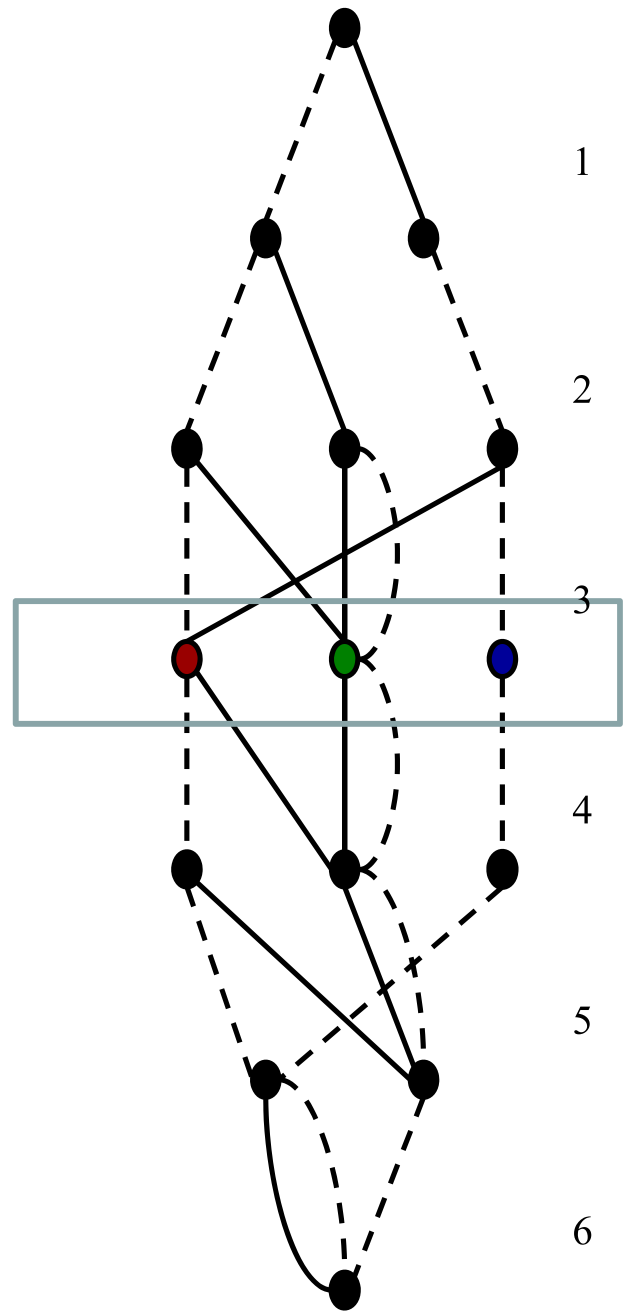
Branch and Bound

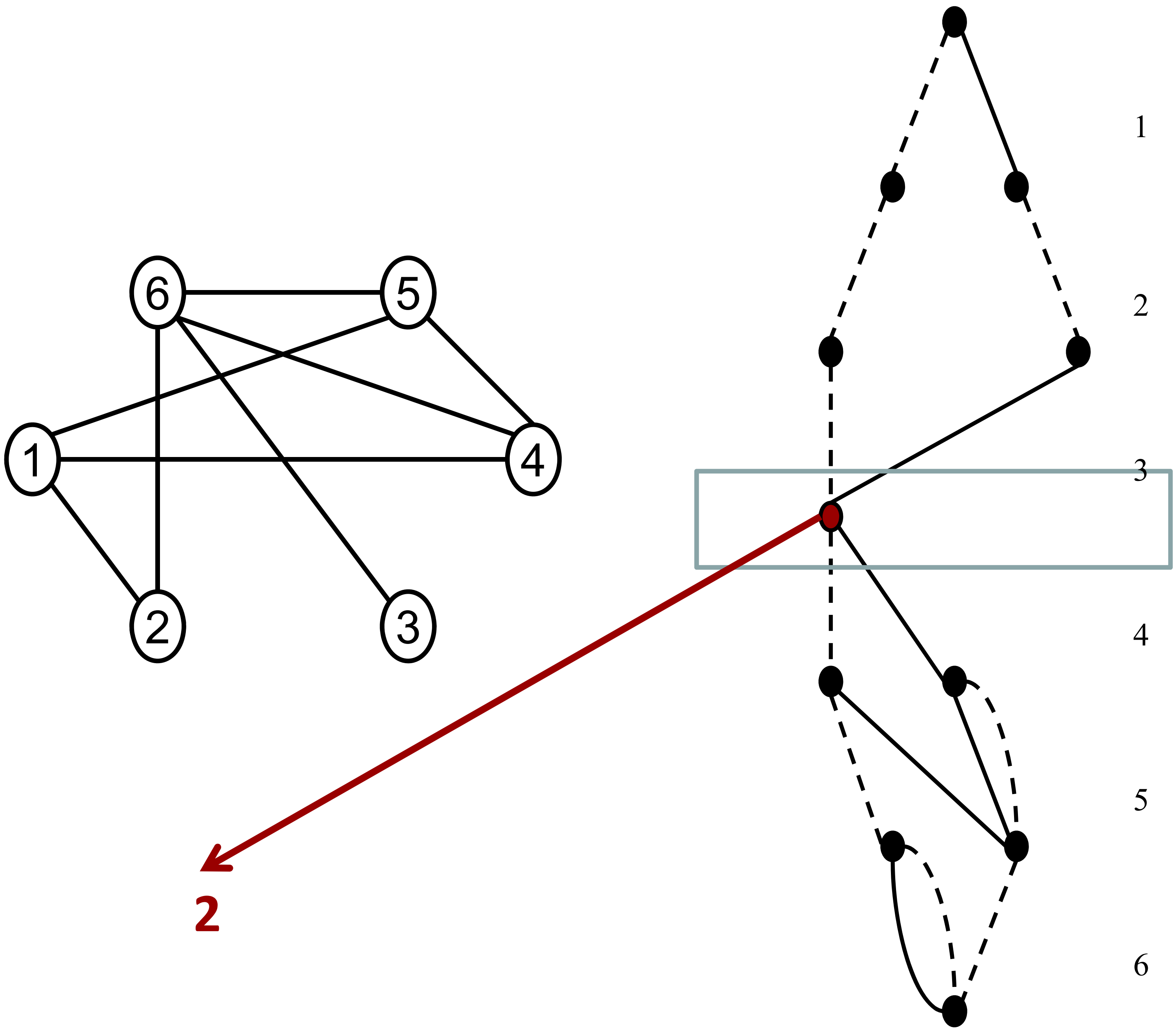




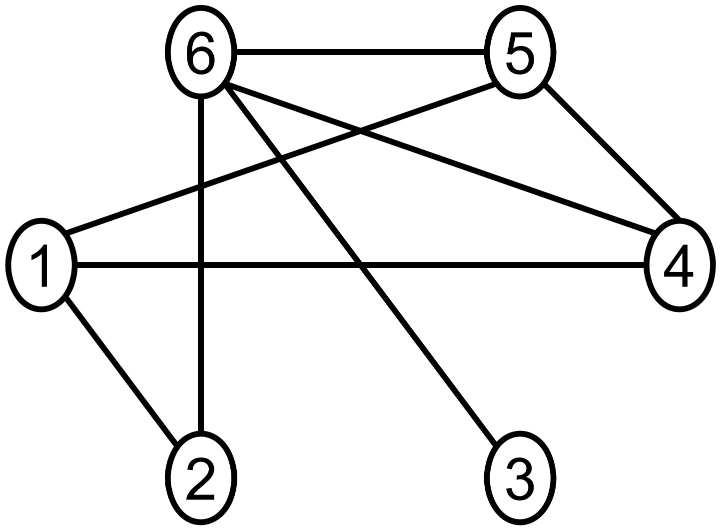
Branch and Bound

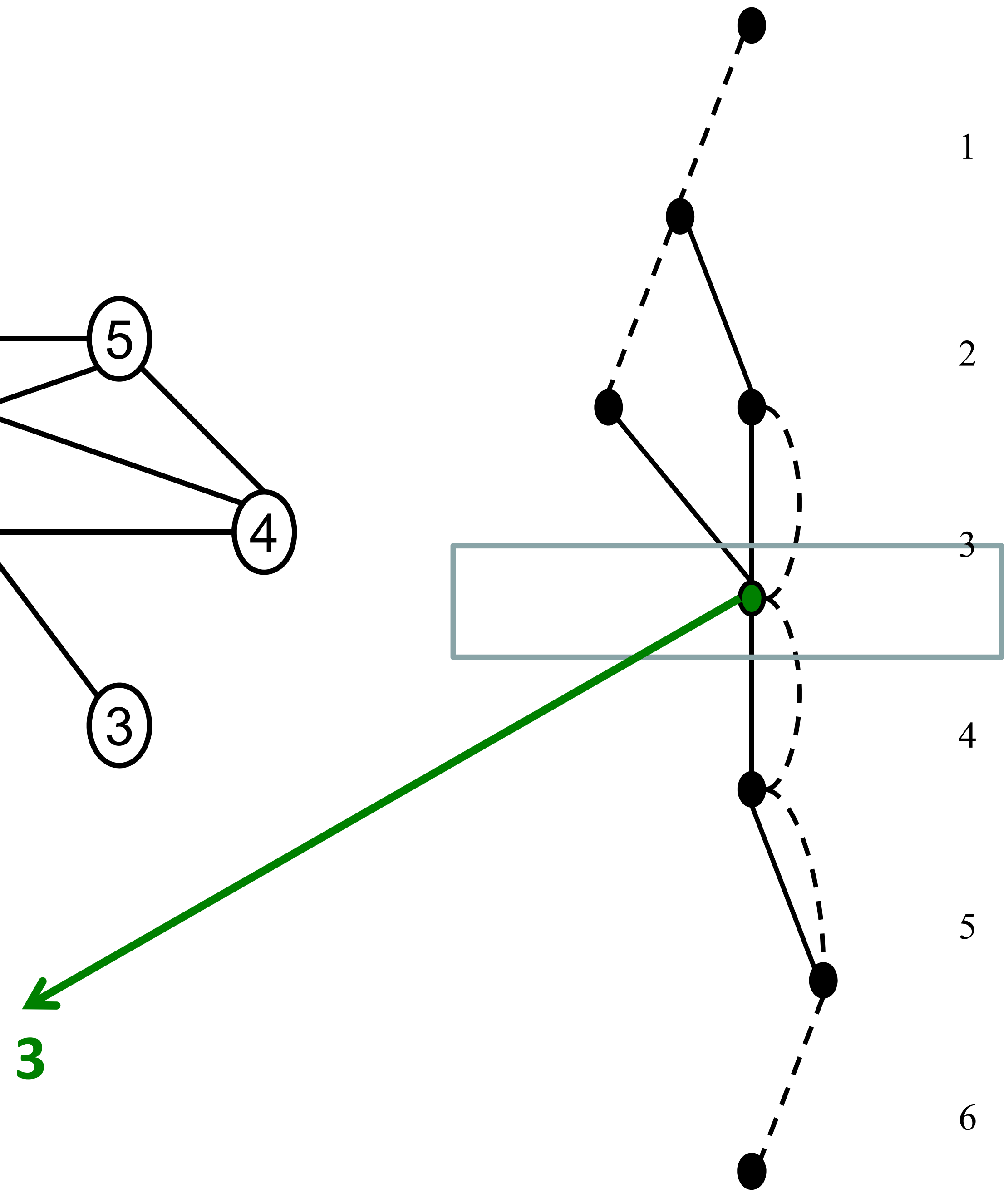




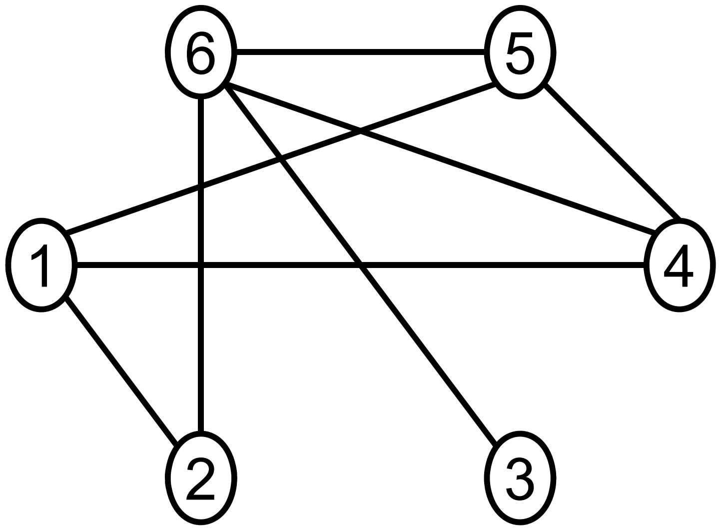

6
5
1
4
2
3
1
2
3
4
5
6
2

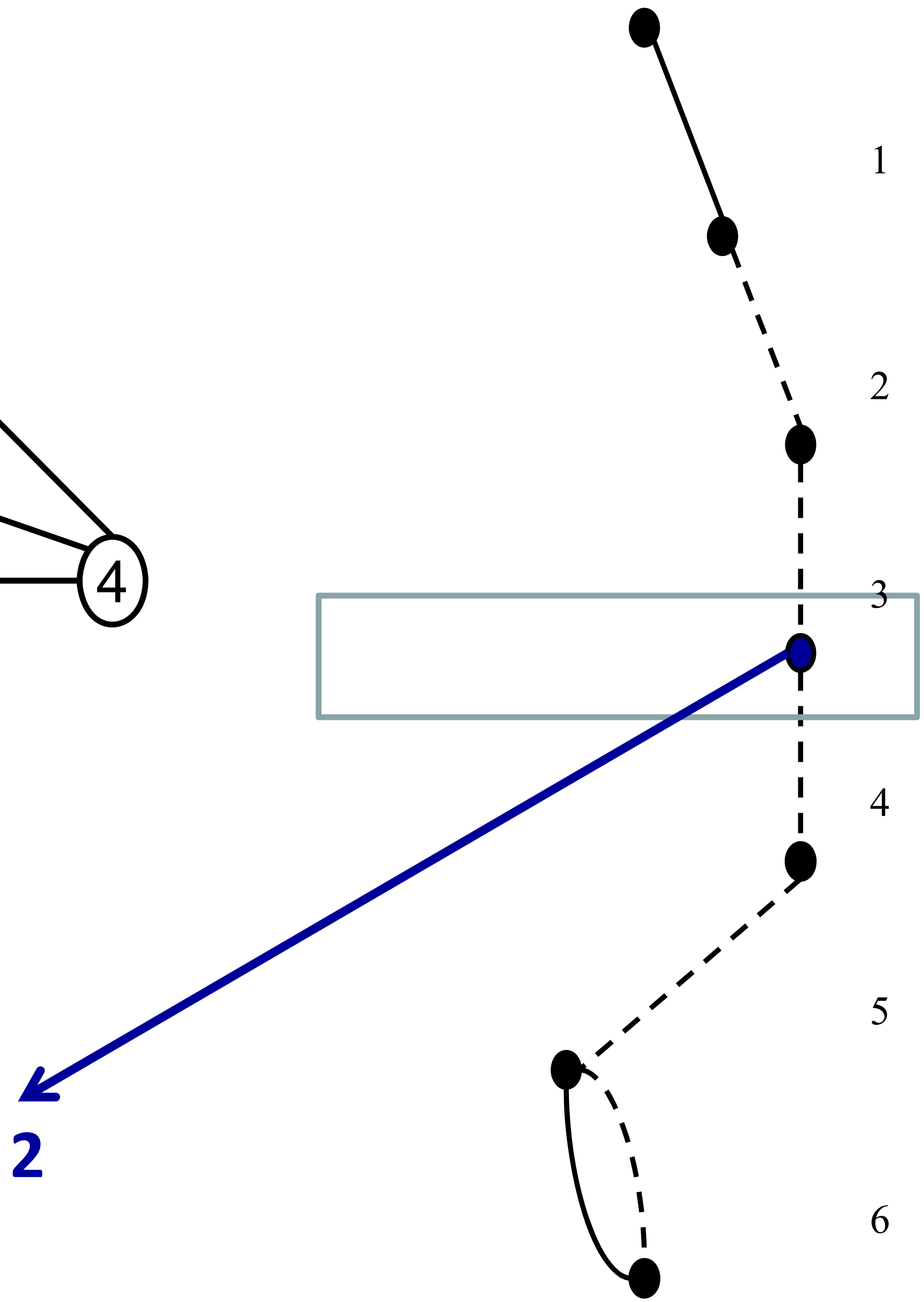
Branch and Bound



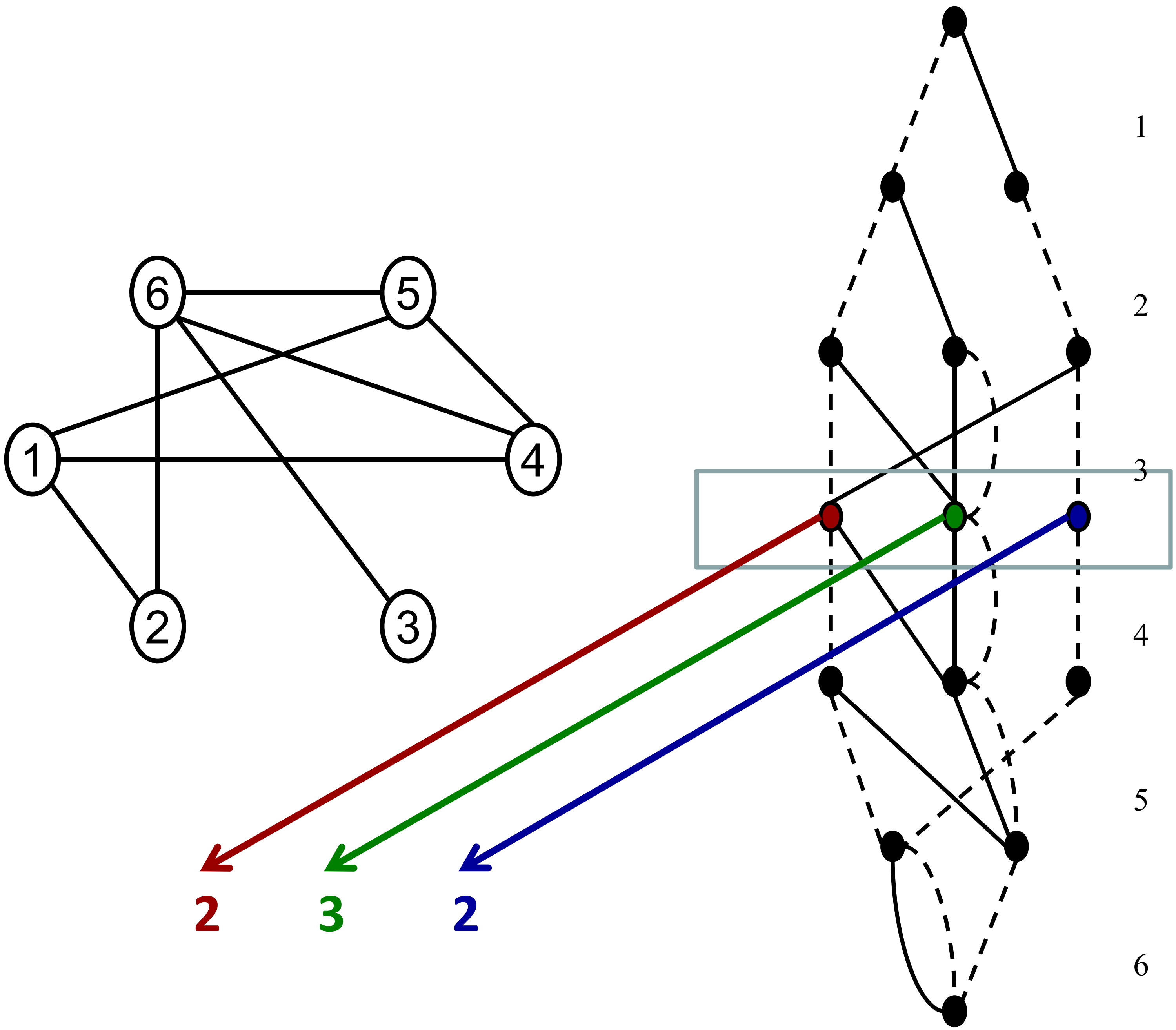


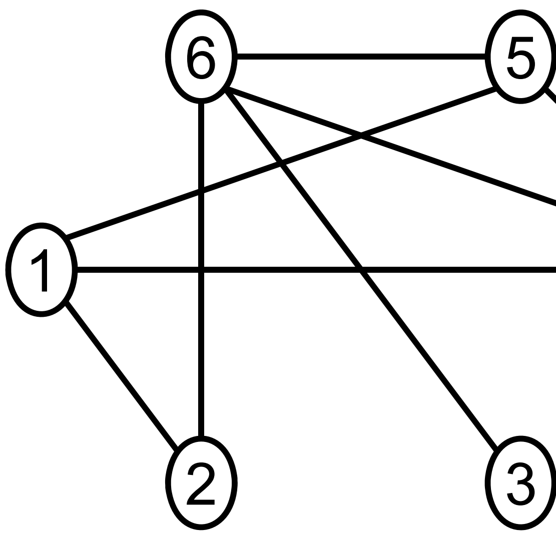
Branch and Bound

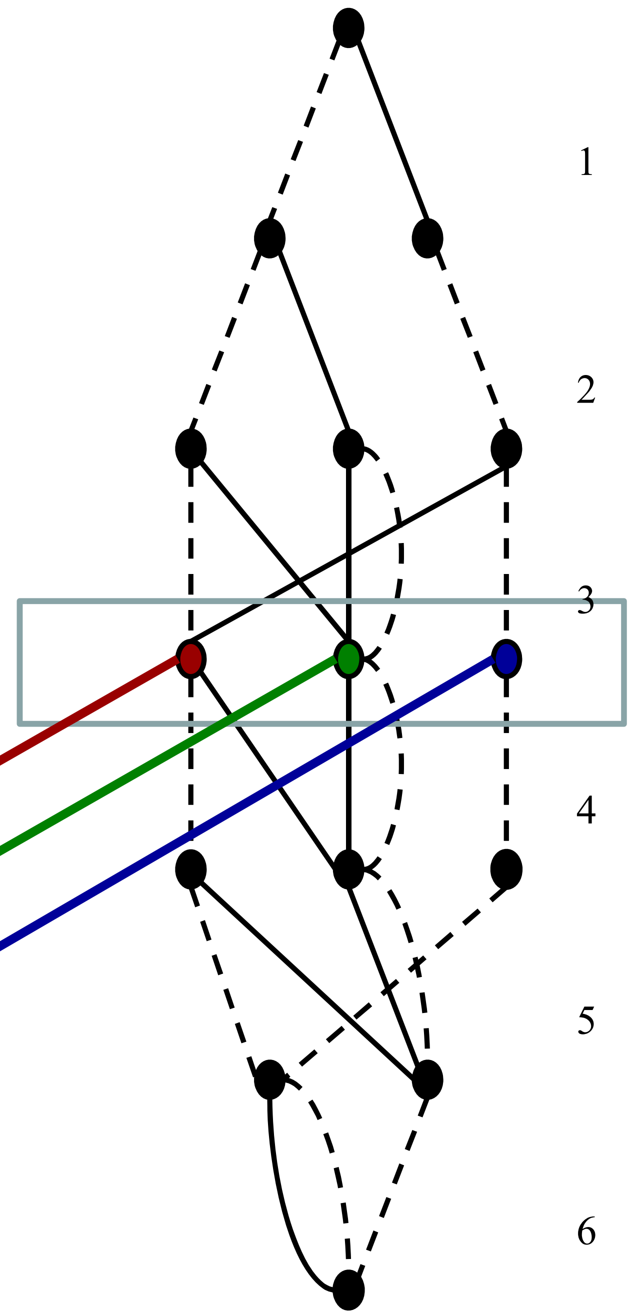




Branch and Bound







max { **2** , **3** , **2** } = 3

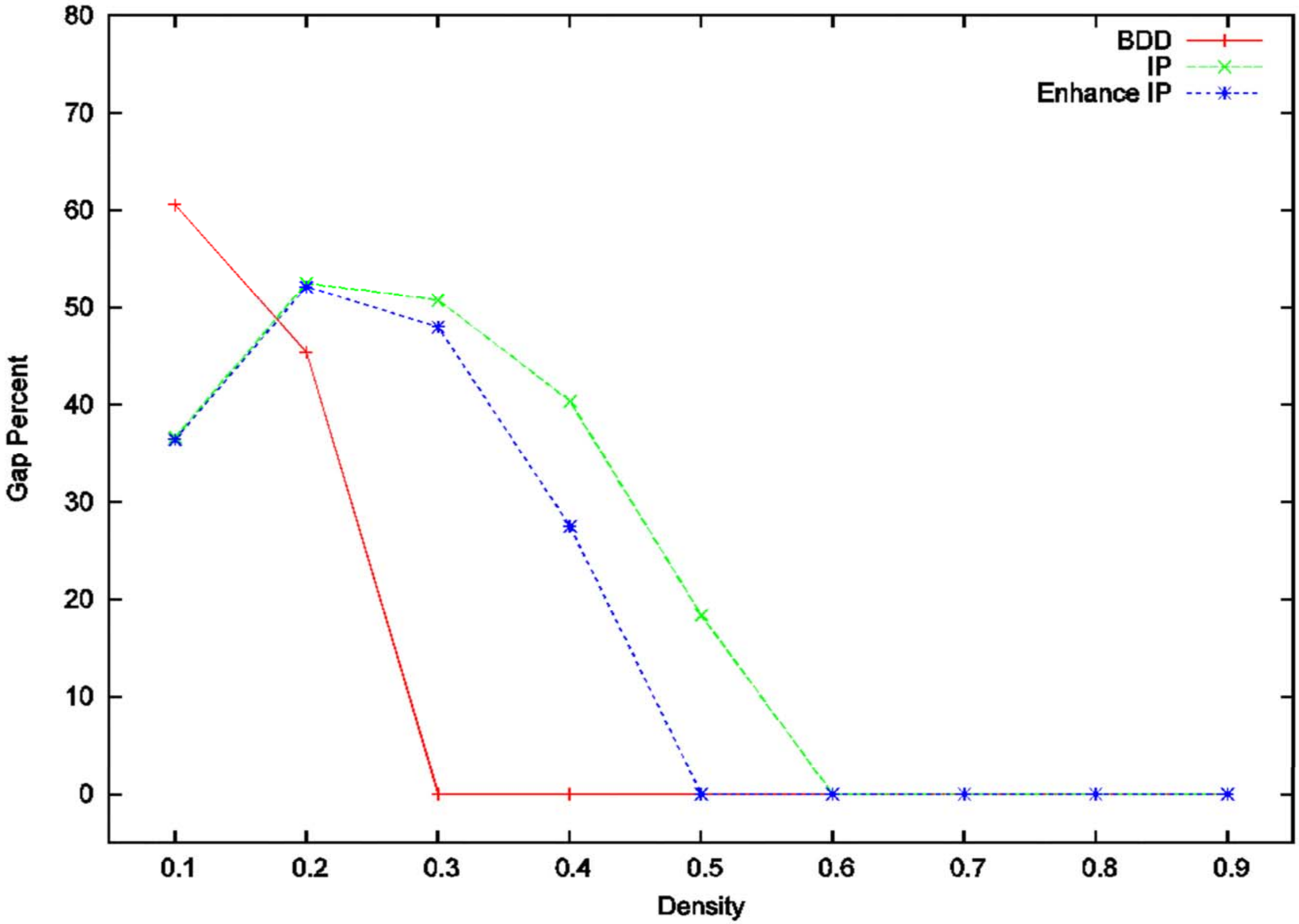
New Branching Scheme

- Novel branching **scheme**
  - Branch on **pools** of partial solutions
  - Remove **symmetry** from search
    - Symmetry with respect to feasible completions
  - Can be combined with other techniques
    - Use decision diagrams for branching, and LP for bounds
  - Immediate **parallelization**
    - Send nodes to different workers, recursive application
    - DDX10 (CPAIOR 2014)

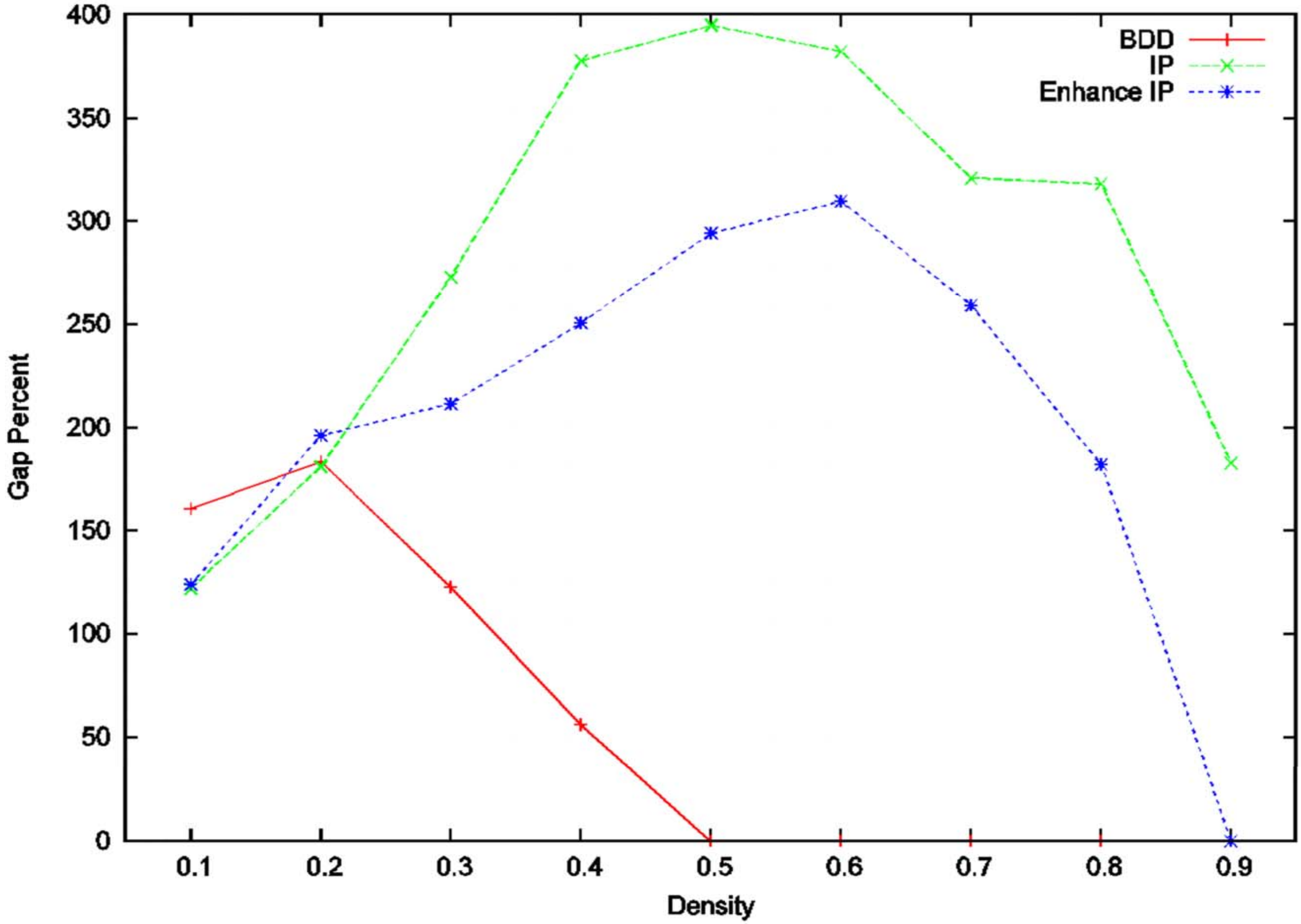
Computational Results

- Compare with IBM ILOG CPLEX
  - State-of-the-art integer programming technology
- Use typical, strong formulations
  - Edge formulation and clique formulation for maximum independent set problem
    - $O(n)$ variables, $O(n^2)$ constraints
- Random Erdös-Rényi G(n,p) graphs and DIMACS Clique graphs
  - Compare end gaps after 1,800 seconds

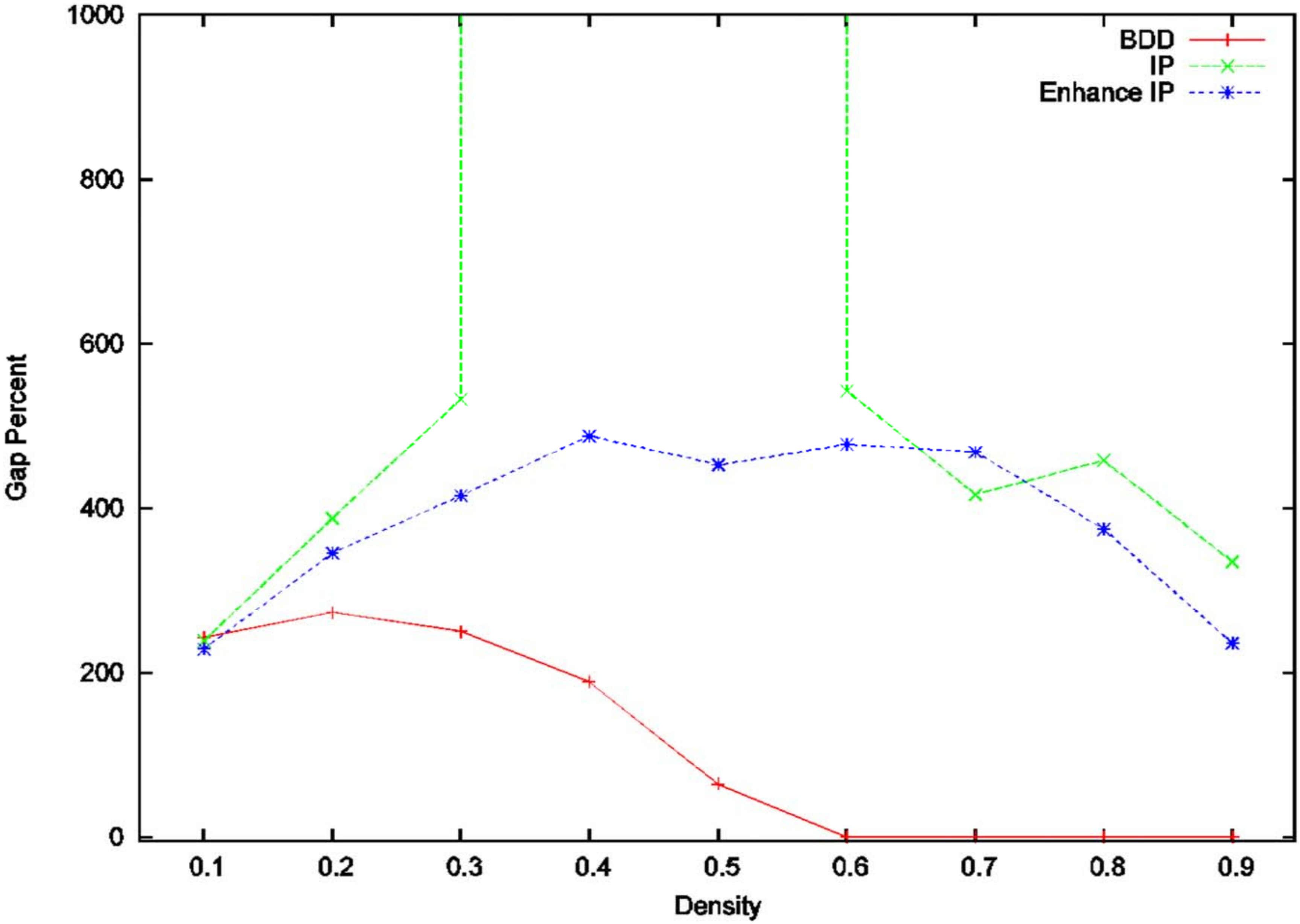
Random graphs: n=250



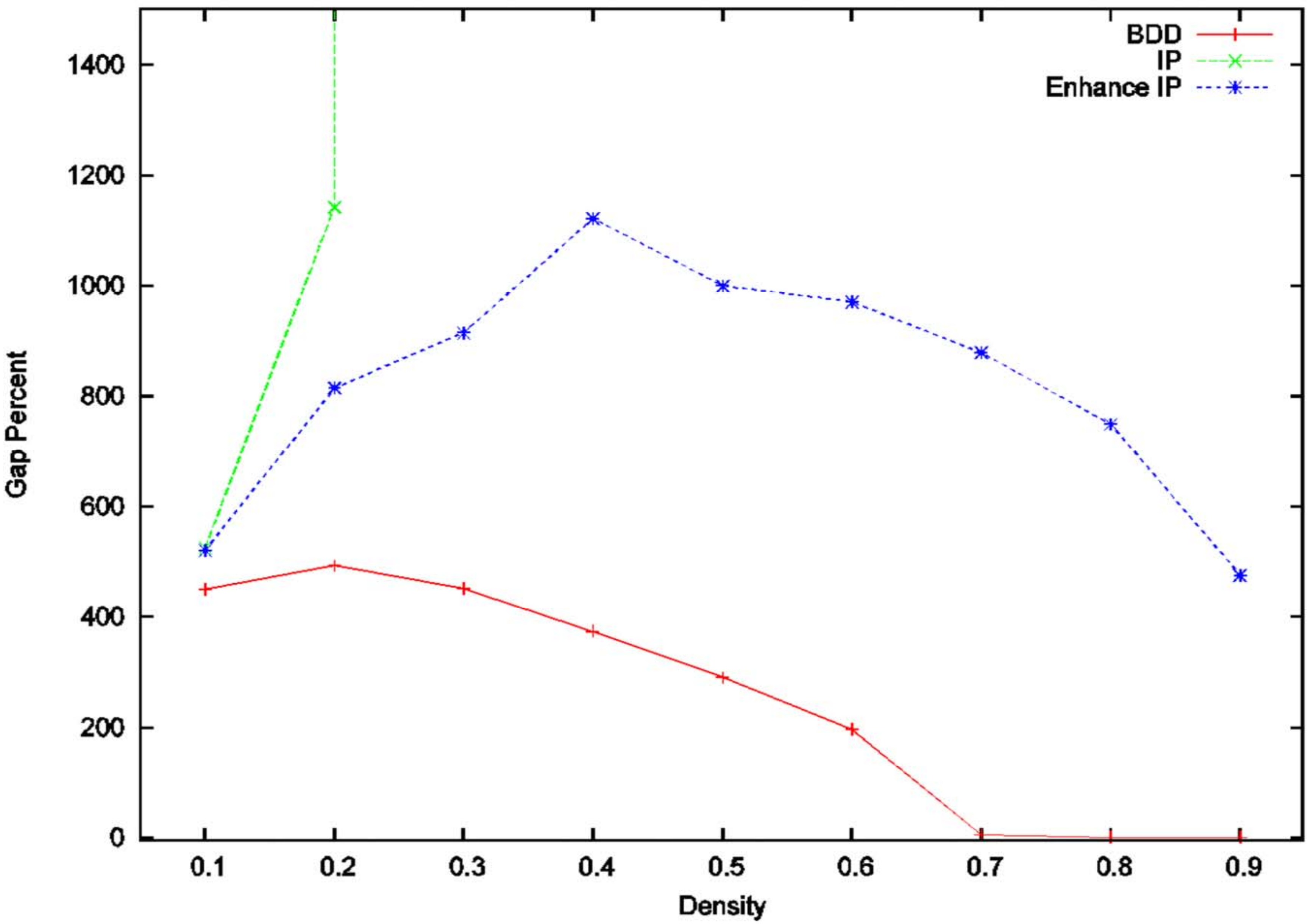
Random graphs: n=500



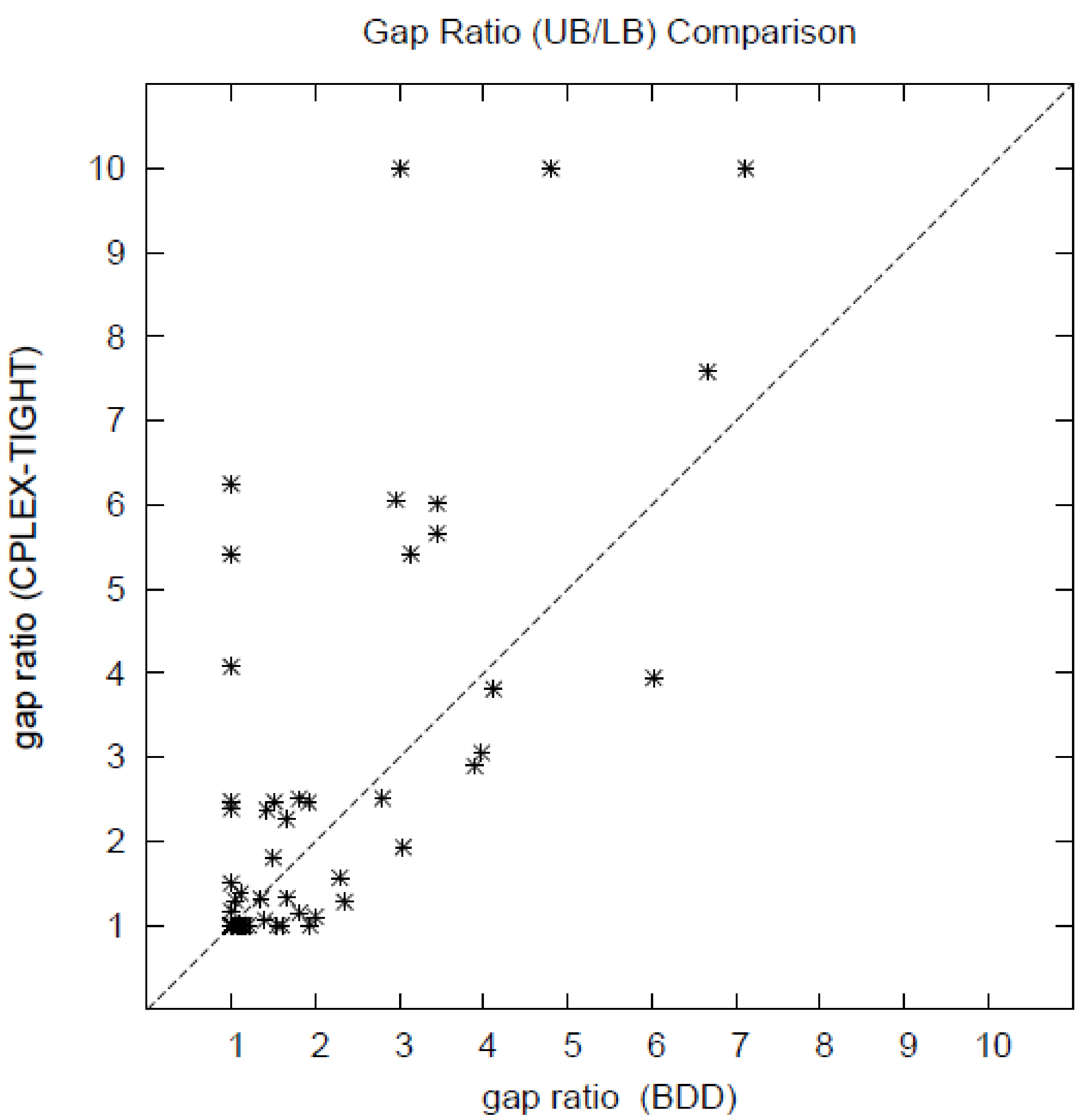
Random graphs: n=750



Random graphs: n=1500

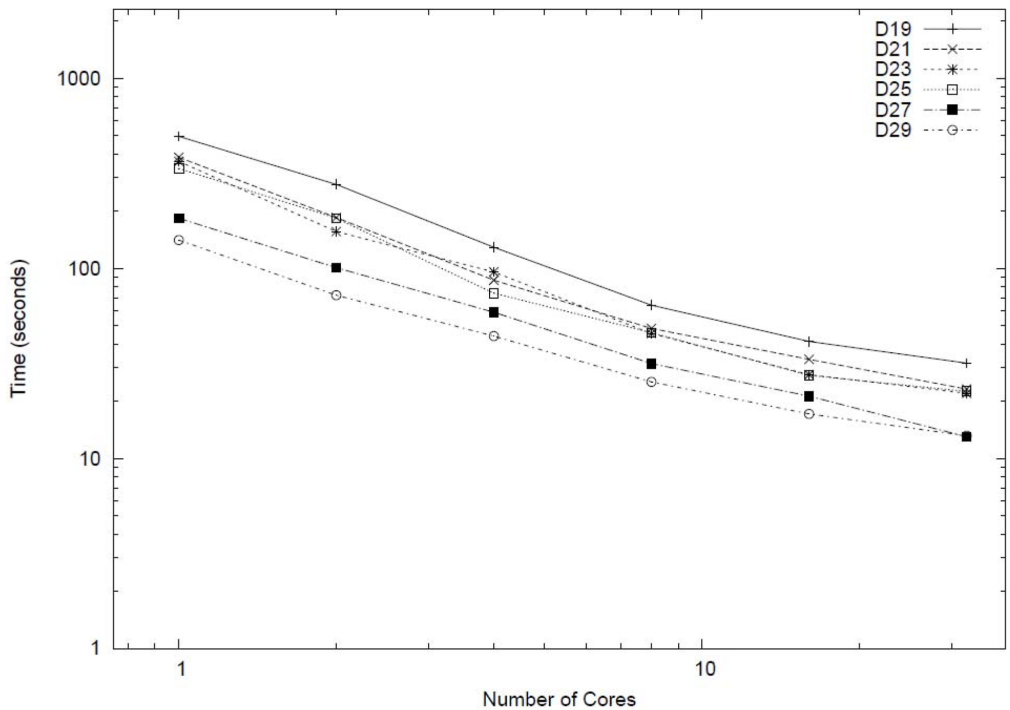


DIMACS Graphs: End Gap (1,800s)

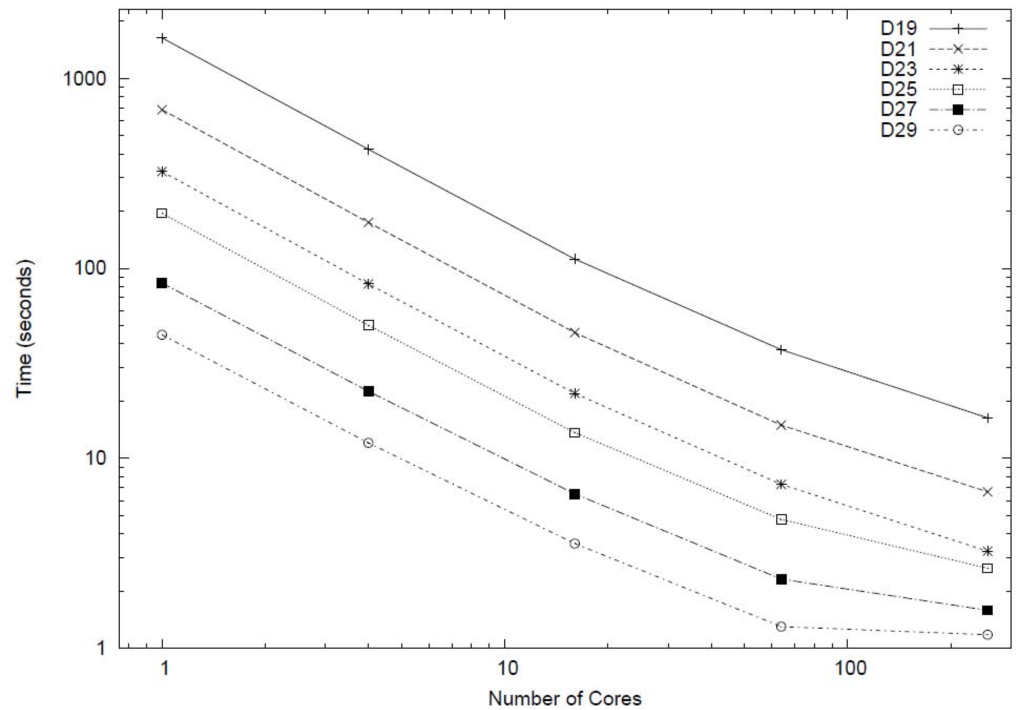


Parallelization: BDD vs CPLEX

CPLEX



BDD



- n = 170, each data point avg over 30 instances
- 1 worker: BDD 1.25 times faster than CPLEX (density 0.29)
- 32 workers: BDD 5.5 times faster than CPLEX (density 0.29)
- BDDs scale to well to (at least) 256 workers

General Approach

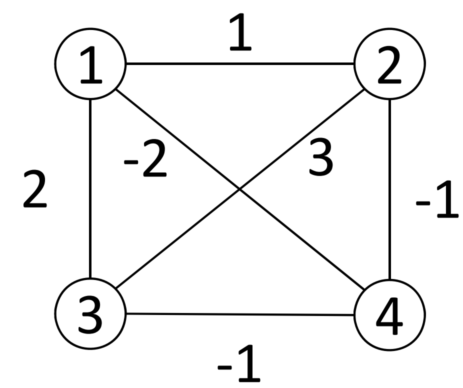
- In general, our approach can be applied when problem is formulated as a dynamic programming model
  – We can build exact BDD from DP model using top-down compilation scheme (exponential size in general)
  – Note that we do not use DP to solve the problem, only to represent it

- Other problem classes considered
  – MAX-CUT, set covering, set packing, MAX 2-SAT, SAT, …

MAX-CUT representation

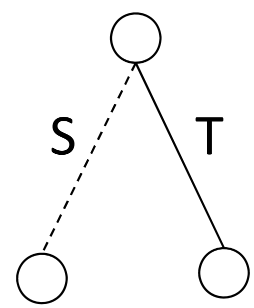
- Value of a cut (S,T) is

$$\sum_{s,t \mid s \in S,\, t \in T} w(s,t)$$



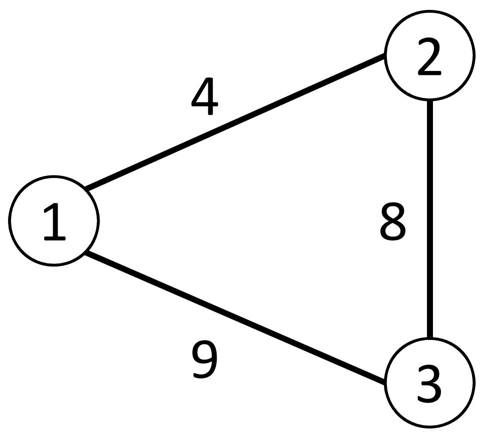
- Example: cut ({1,2}, {3,4)) has value 2
- MAX-CUT: Find a cut with maximum value

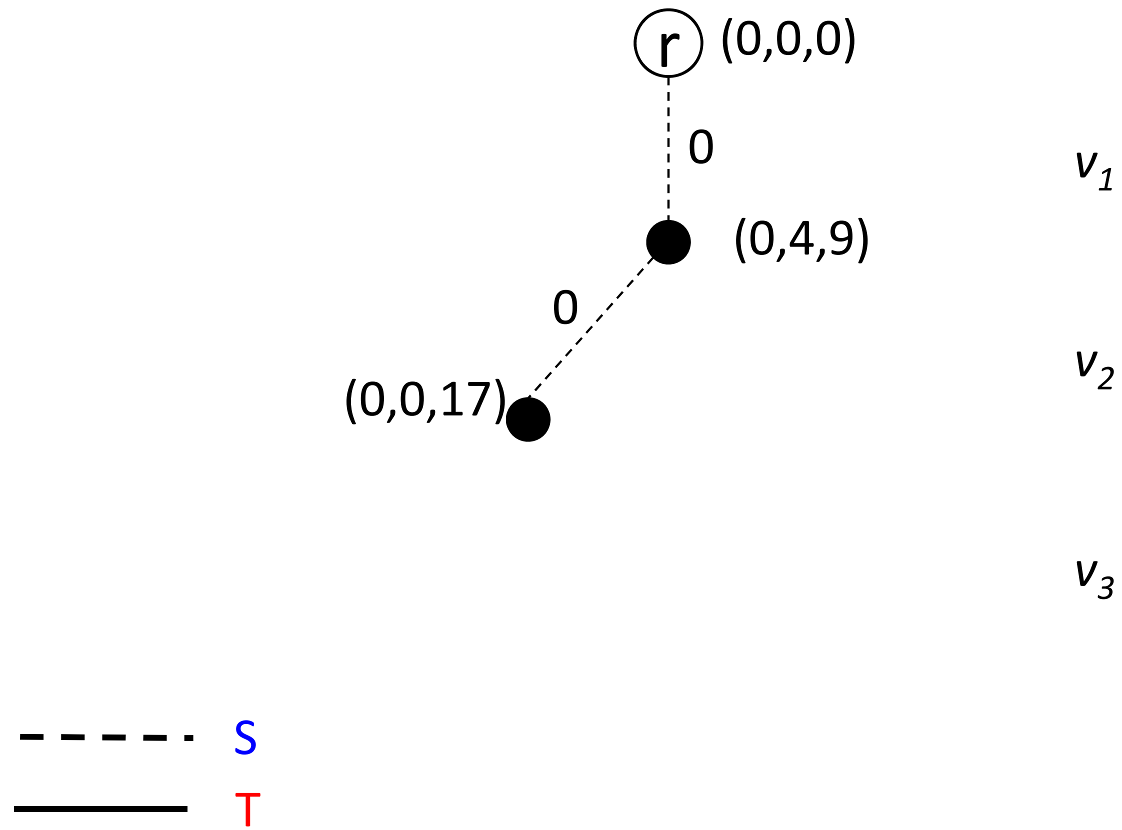
- How can we represent this in a BDD?
  - state represents vertices included in S?
  - we propose a state to represent the marginal cost of including vertex in S



MAX-CUT example BDD

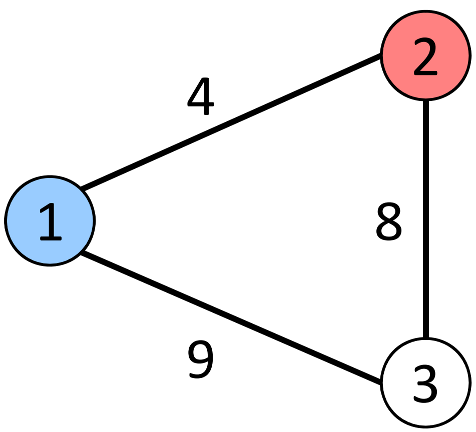
- State: j^{th} element is additional value of adding vertex j to S (if positive)

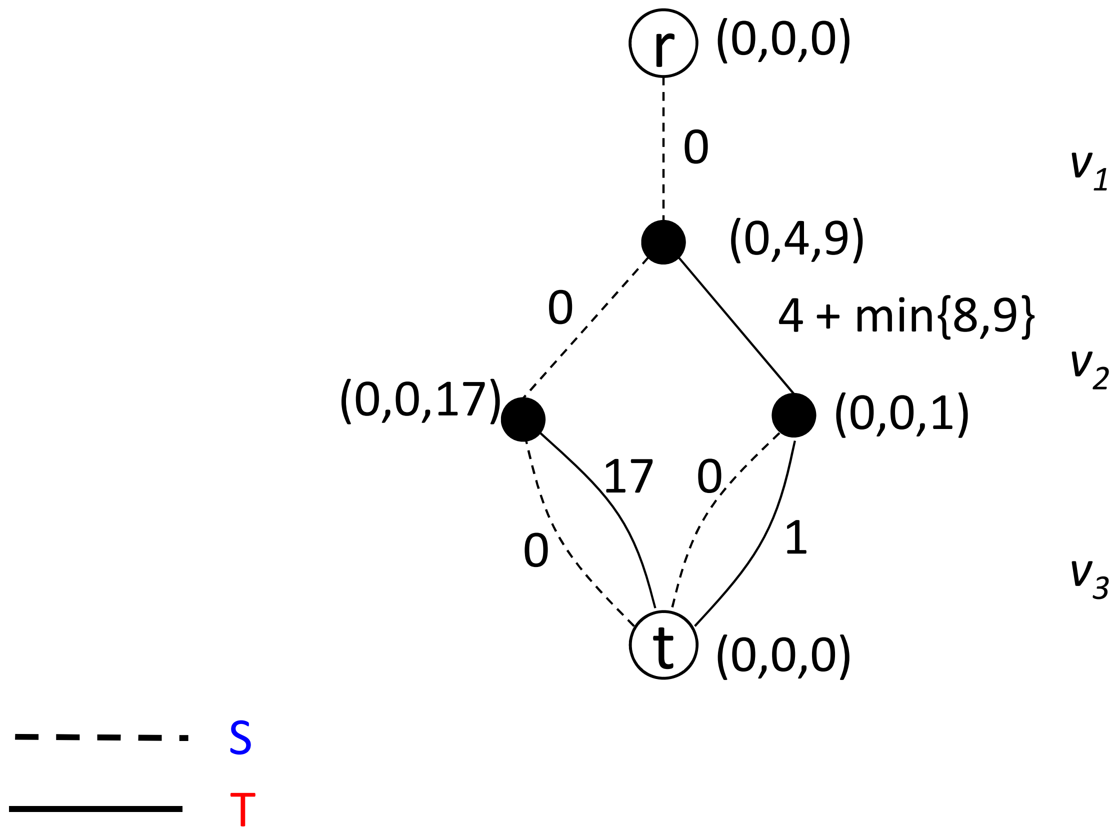




MAX-CUT example BDD

- State: j^{th} element is additional value of adding vertex j to S (if positive)

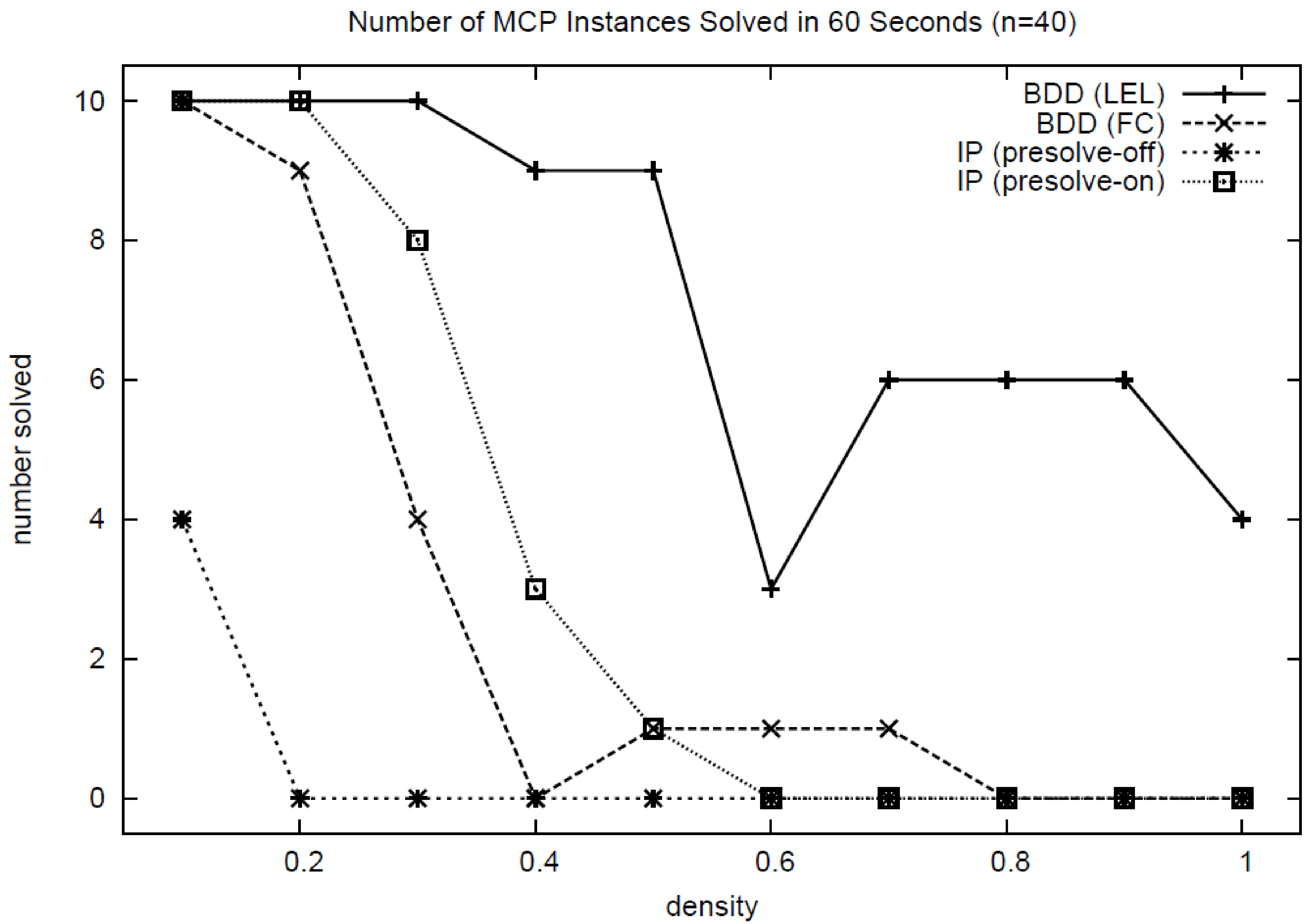




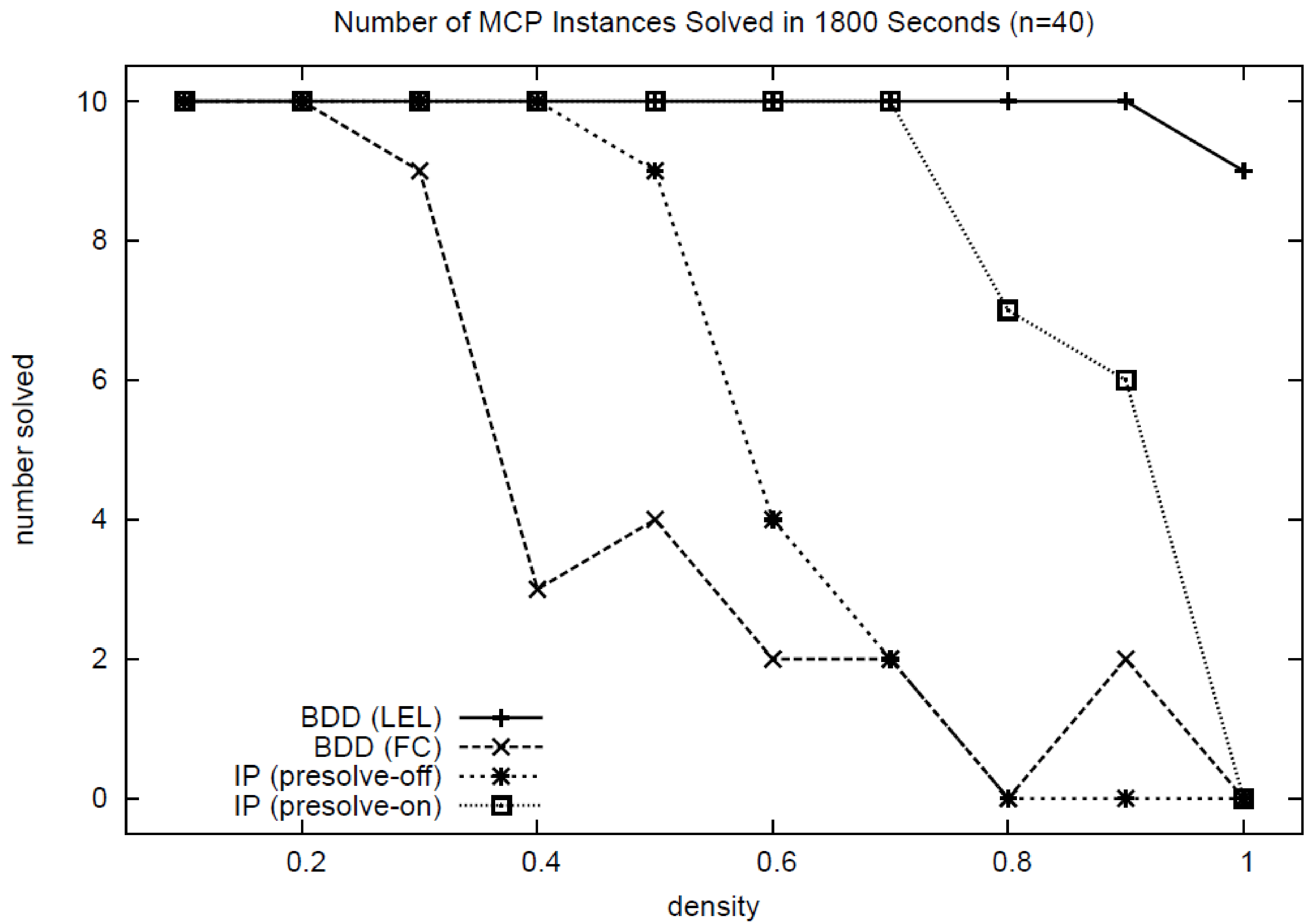
Computational Results

- Compare with IBM ILOG CPLEX
- Typical MIP formulation + triangle inequalities
  - $O(n^2)$ variables, $O(n^3)$ constraints
- Benchmark problems
  - *g* instances
  - Helmberg and Rendl instances, which were taken from Rinaldi's random graph generator
  - n ranges from 800 to 3000 – very large/difficult problems, mostly open
  - Also compared performance with BiqMac

MIP vs BDD: 60 seconds (n=40)



MIP vs BDD: 1,800 seconds (n=40)



BiqMac vs BDD

	BiqMac		BDD		Best known	
instance	LB	UB	LB	UB	LB	UB
g50	5880	5988.18	5880	5899*	5880	5988.18
g32	1390	1567.65	1410*	1645	1398	1560
g33	1352	1544.32	1380*	1536*	1376	1537
g34	1366	1546.70	1376*	1688	1372	1541
g11	558	629.17	564	567*	564	627
g12	548	623.88	556	616*	556	621
g13	578	647.14	580	652	580	645

MDDs for Constraint Programming

- Andersen, Hadzic, Hooker, Tiedemann: A Constraint Store Based on Multivalued Decision Diagrams. *CP* 2007: 118-132
- Hoda, v.H., Hooker: A Systematic Approach to MDD-Based Constraint Programming. *CP* 2010: 266-280
- Cire, v.H.: MDDs for Sequencing Problems. *Operations Research*, 61(6): 1411-1428, 2013.

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

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

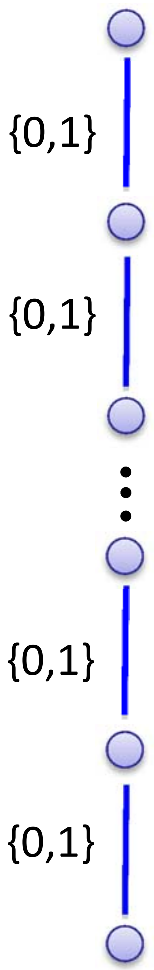
$x_1 < x_2$ $\quad x_2 \in \{2,3\}$

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

Illustrative Example

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

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

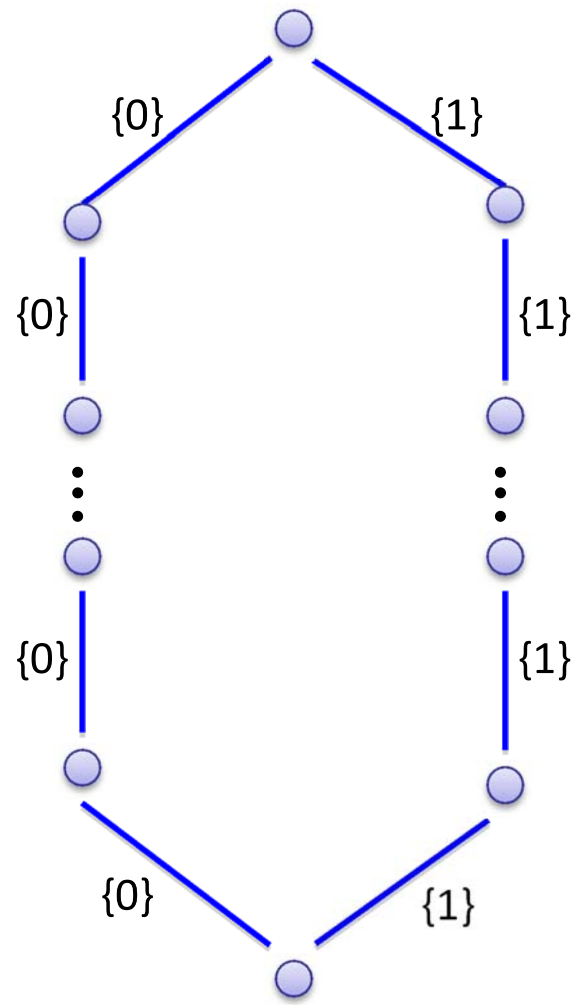


x_1

x_2

x_{n-1}

x_n



domain representation, size 2^n

MDD representation, size 2

Drawback of domain propagation

- All structural relationships among variables are projected onto the domains
- Potential solution space implicitly defined by Cartesian product of variable domains (very coarse relaxation)

We can communicate more information between constraint using MDDs [Andersen et al. 2007]

- Explicit representation of more refined potential solution space
- Limited width defines *relaxed* MDD
- Strength is controlled by the imposed width

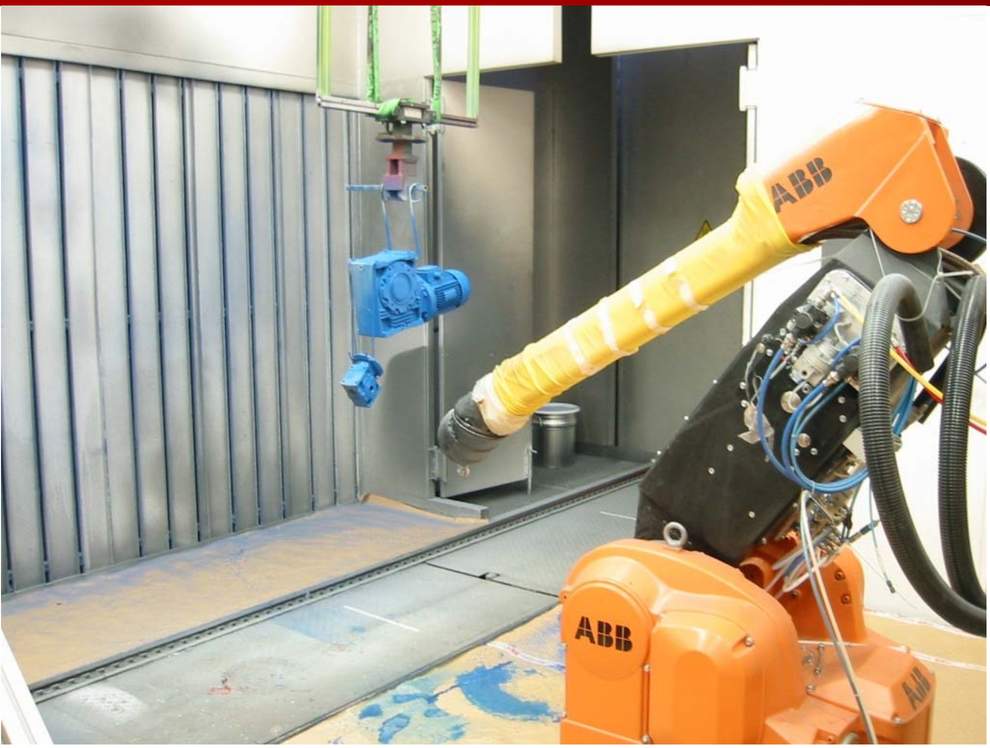
MDD-based Constraint Programming

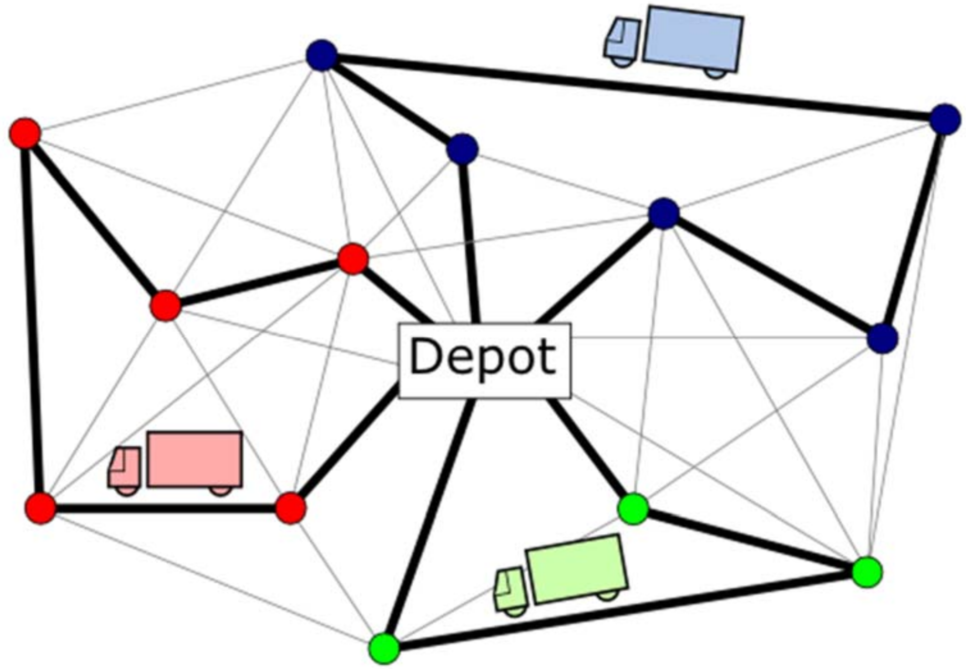
- Maintain limited-width MDD
  - Serves as relaxation
  - Typically start with width 1 (initial variable domains)
  - Dynamically adjust MDD, based on constraints
- Constraint Propagation
  - Edge filtering: Remove provably inconsistent edges (those that do not participate in any solution)
  - Node refinement: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

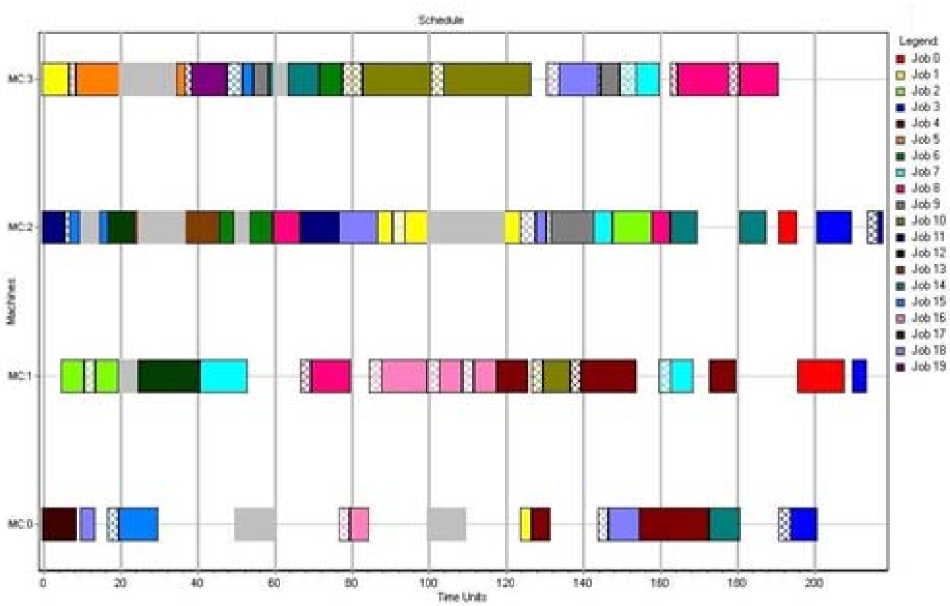
Specific MDD propagation algorithms

- Linear equalities and inequalities [Hadzic et al., 2008] [Hoda et al., 2010]
- *Alldifferent* constraints [Andersen et al., 2007]
- *Element* constraints [Hoda et al., 2010]
- *Among* constraints [Hoda et al., 2010]
- Disjunctive scheduling constraints [Hoda et al., 2010] [Cire & v.H., 2011, 2013]
- *Sequence* constraints (combination of *Amongs*) [Bergman et al., 2013]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]

Case Study: Disjunctive Scheduling



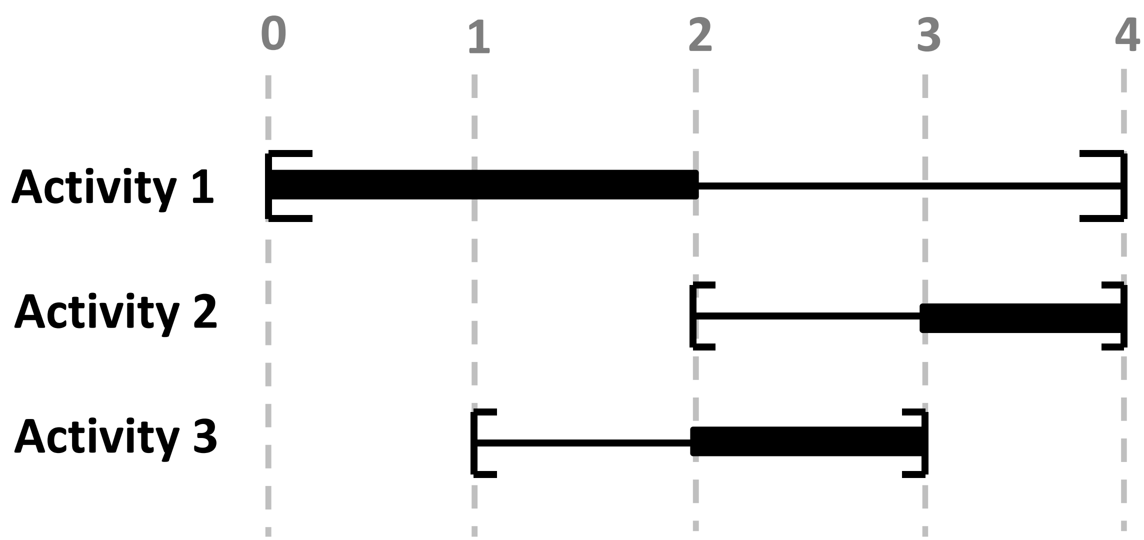




Disjunctive Scheduling

- Sequencing and scheduling of activities on a resource

- *Activities*
  - Processing time: p_i
  - Release time: r_i
  - Deadline: d_i



- *Resource*
  - Nonpreemptive
  - Process one activity at a time

Common Side Constraints

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

MDD Representation

- Natural representation as 'permutation MDD'

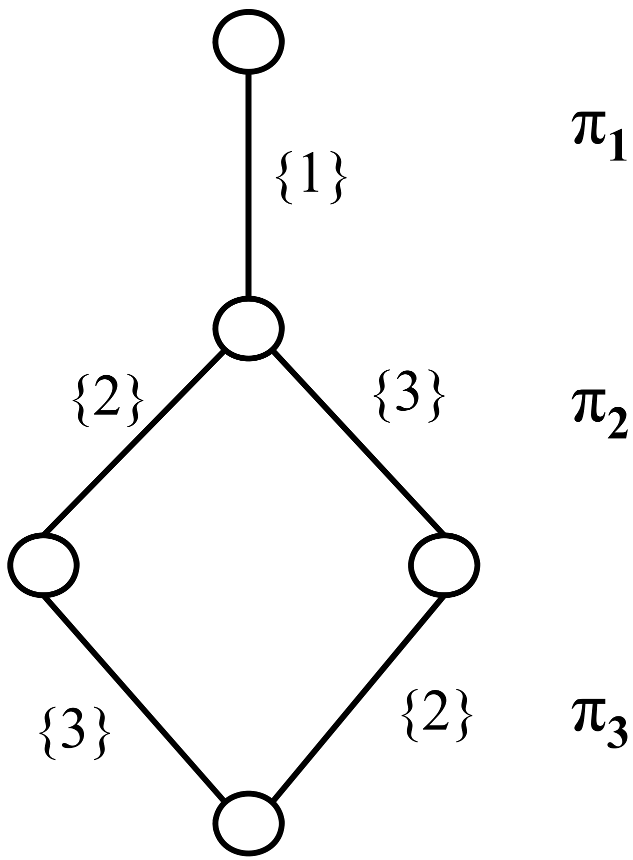
- Every solution can be written as a permutation $\boldsymbol{\pi}$

  $\pi_1, \pi_2, \pi_3, \ldots, \pi_n$: activity sequencing in the resource

- Schedule is *implied* by a sequence, e.g.:

$$start_{\pi_i} \geq start_{\pi_{i-1}} + p_{\pi_{i-1}} \qquad i = 2, \ldots, n$$

MDD Representation



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

Path {1} – {3} – {2} :

$0 \leq start_1 \leq 1$

$6 \leq start_2 \leq 7$

$3 \leq start_3 \leq 5$

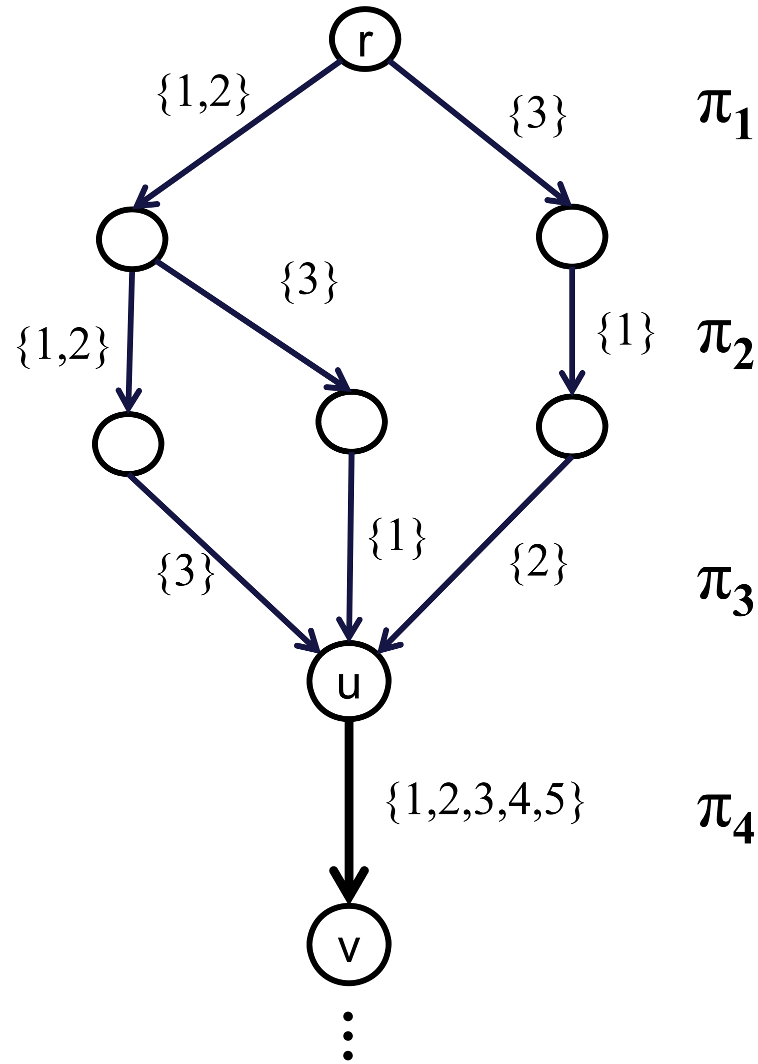
MDD Propagation

We can apply several propagation algorithms:

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

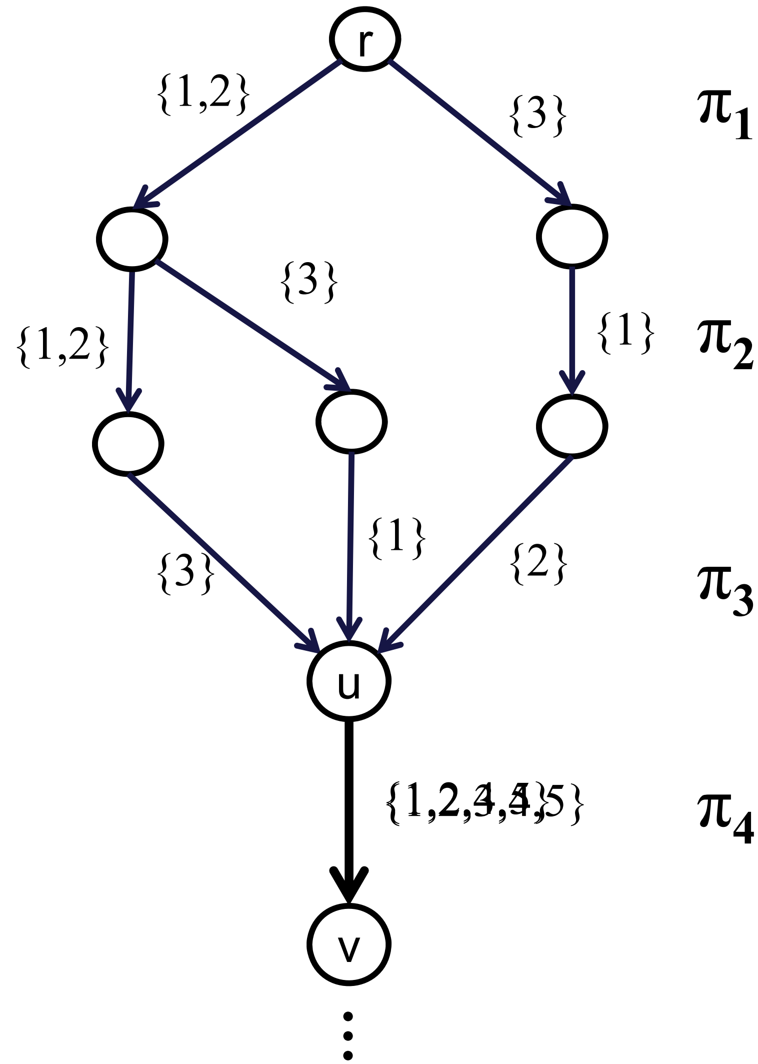
Propagation (cont'd)

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



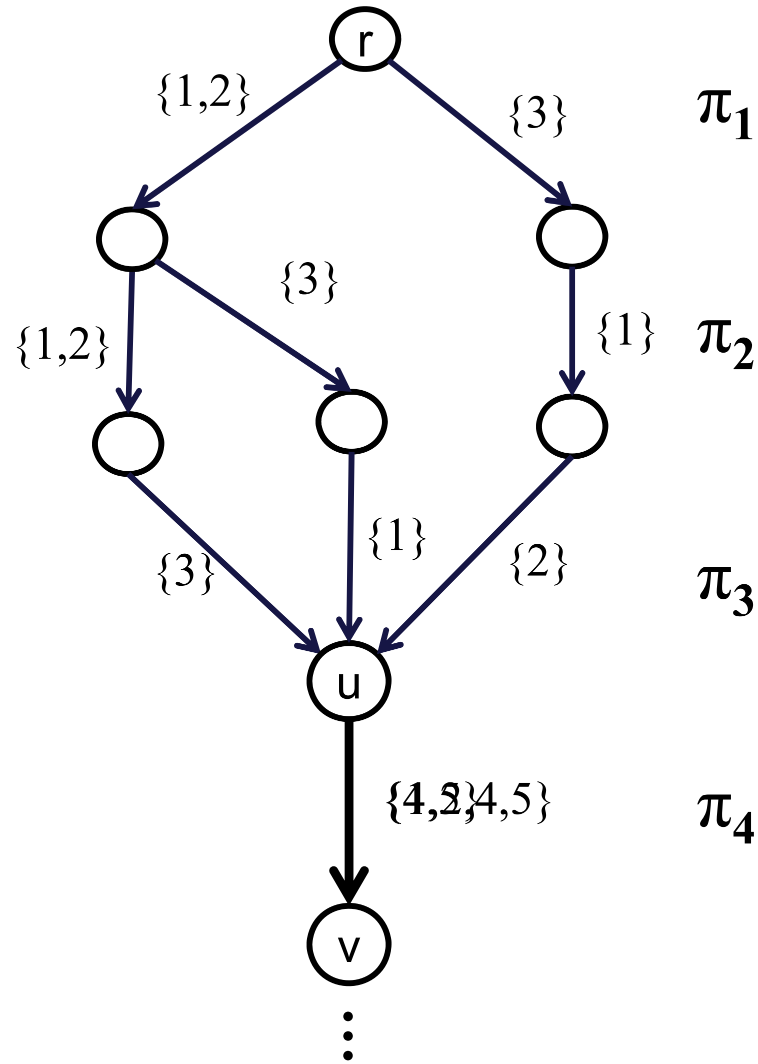
Alldifferent Propagation

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



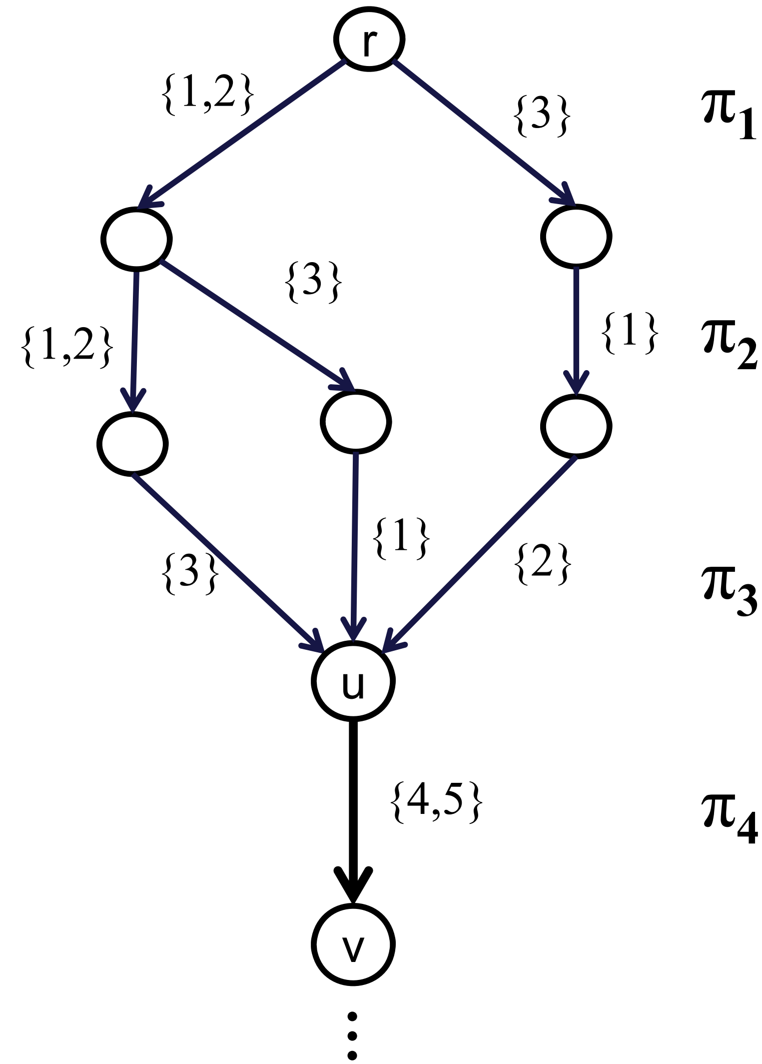
Alldifferent Propagation

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



Propagate Earliest Completion Time

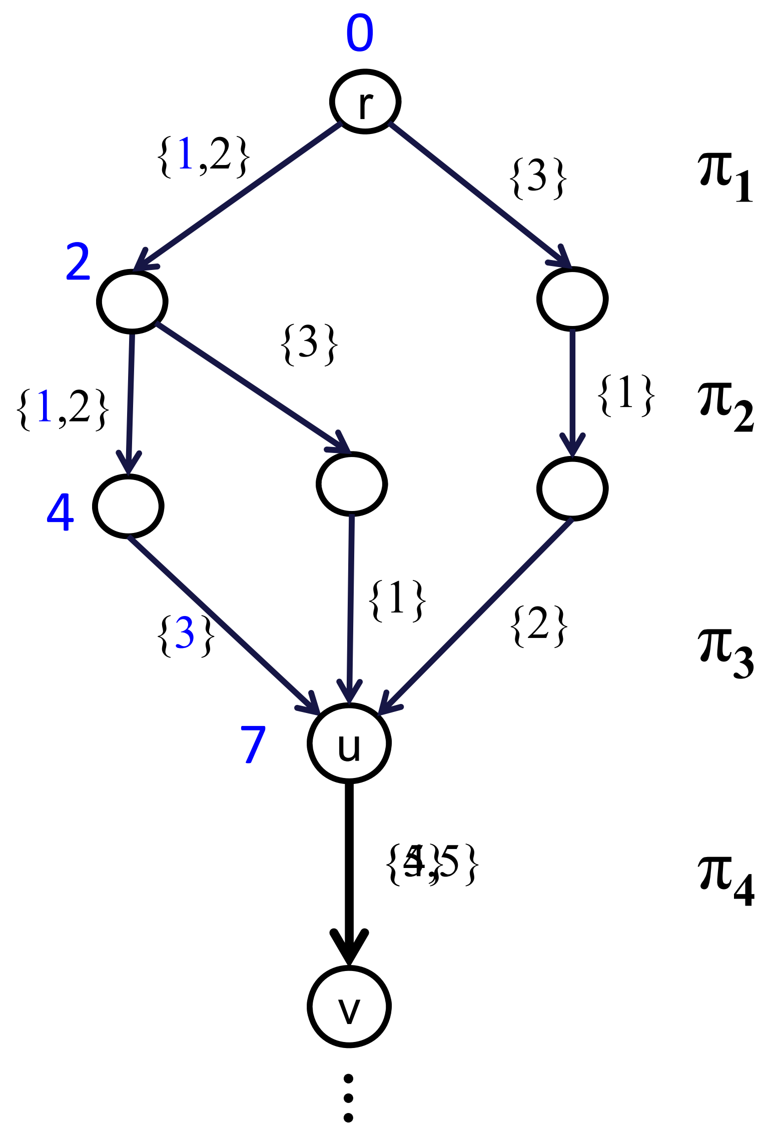
- Earliest Completion Time: E_u
  - Minimum completion time of all paths from root to node u
- Similarly: Latest Completion Time



Propagate Earliest Completion Time

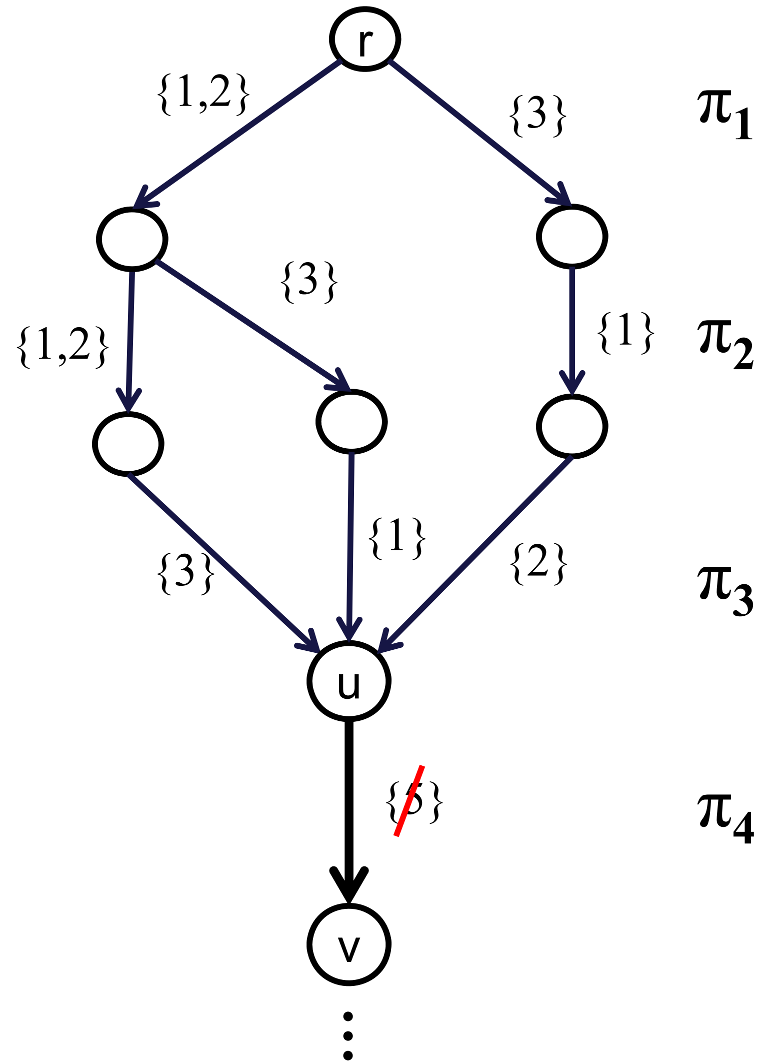
Act	r_i	d_i	p_i
1	0	4	2
2	3	7	3
3	1	8	3
4	**5**	**6**	**1**
5	2	10	3

- $E_u = 7$
- 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 $j \ll i$



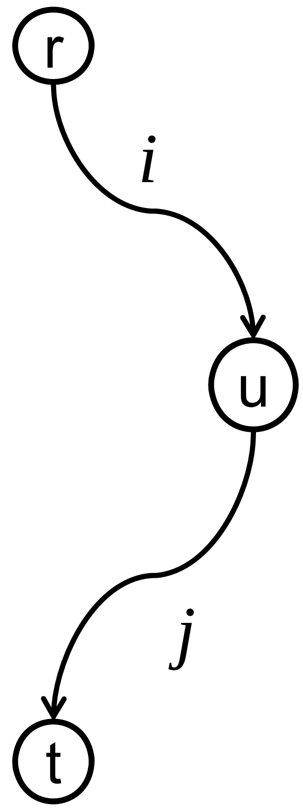
More MDD Inference

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

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

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

- Same technique applies to relaxed MDD



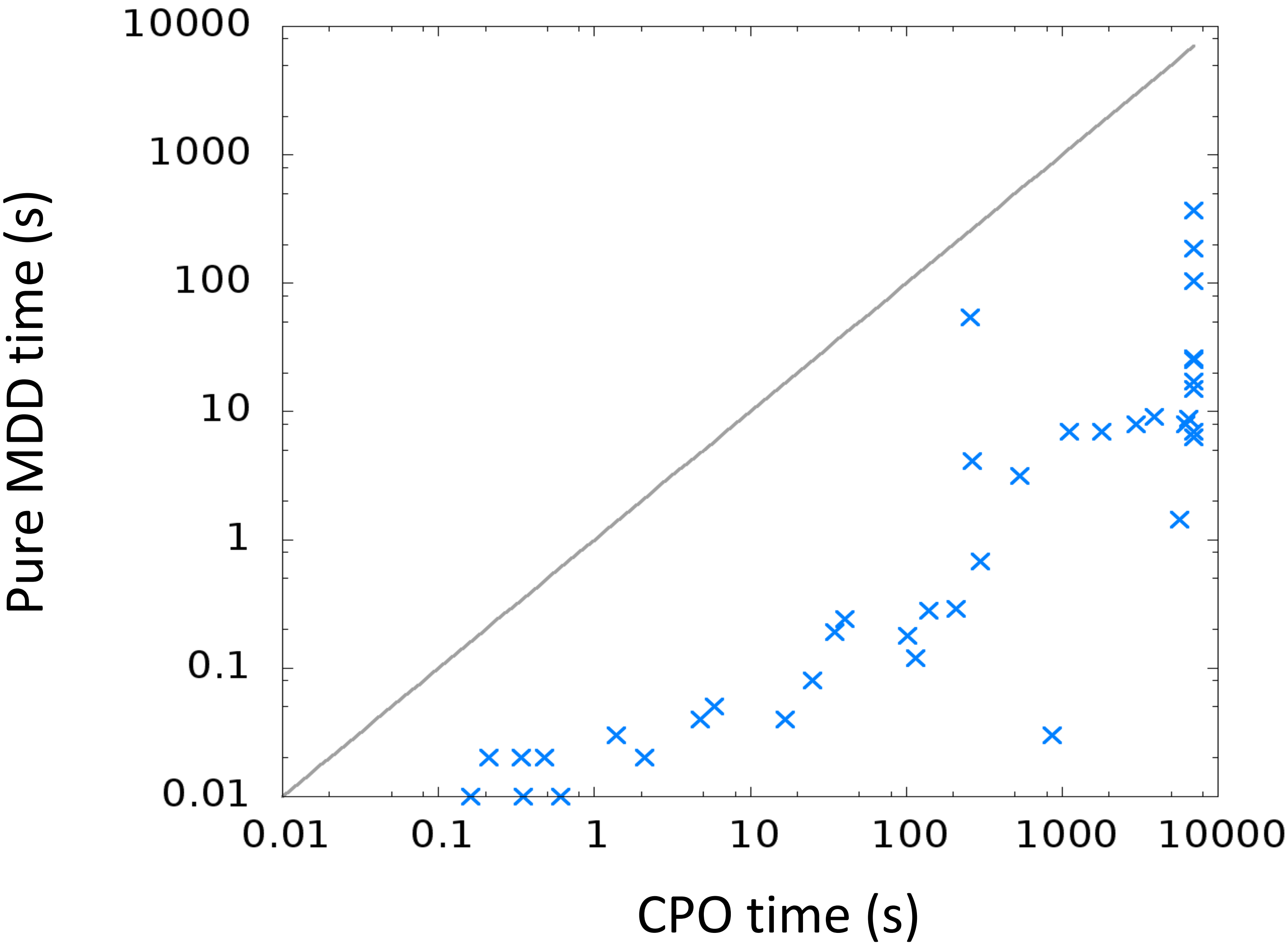
Communicate Precedence Relations

1. Provide precedence relations from MDD to CP
   - update start/end time variables in CP model
   - other inference techniques may utilize them
   - help to guide search
2. Filter the MDD using precedence relations from other (CP) techniques
3. In context of MIP, these can be added as linear inequalities

Experiments

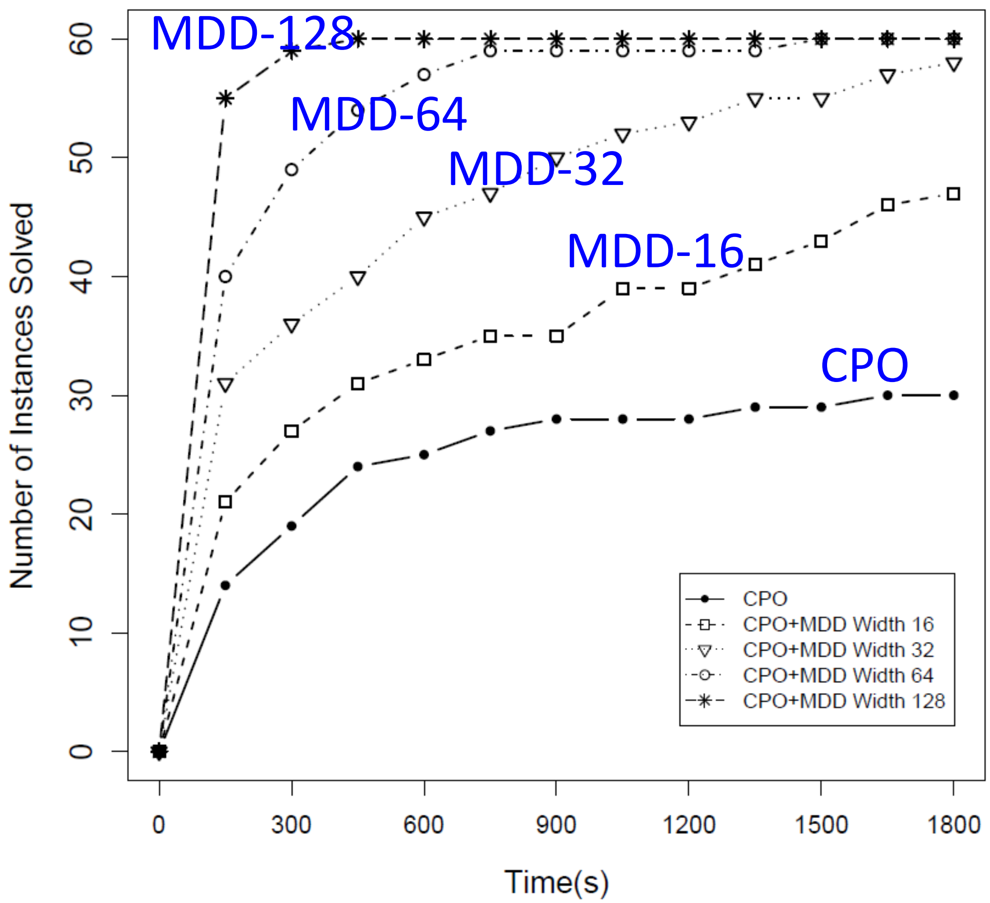
- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation
  - MDD is added as user-defined propagator

TSP with Time Windows

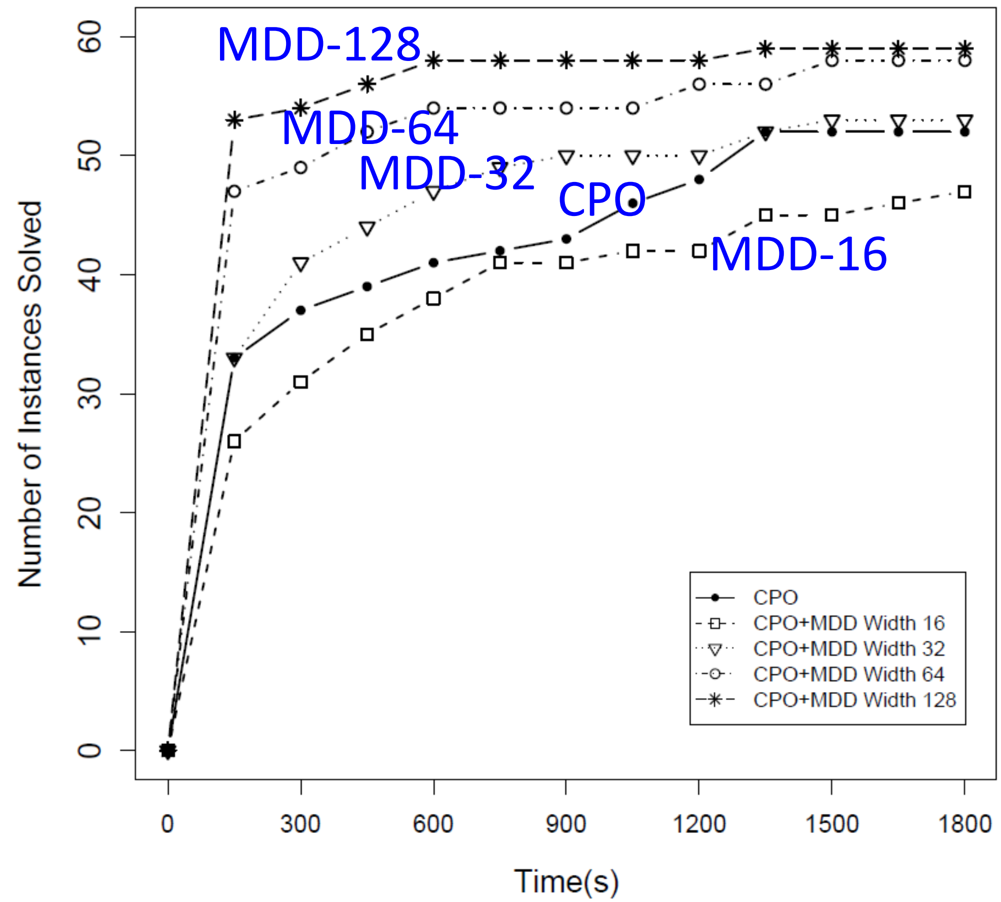


Dumas/Ascheuer instances
- 20-60 jobs
- lex search
- MDD width: 16

Total Tardiness Results



total tardiness



total weighted tardiness

Sequential Ordering Problem (TSPLIB)

			CPO		CPO+MDD, width 2048	
instance	vertices	bounds	best	time (s)	best	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

Summary

What can MDDs do for discrete optimization?

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

MDDs for integer optimization

- MDD *relaxations* provide upper bounds
- MDD *restrictions* provide lower bounds
- Incorporation in branch-and-bound can be very effective

MDDs for constraint programming and scheduling

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