Decision Diagrams for Optimization and Scheduling

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

What can MDDs do for combinatorial optimization?

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

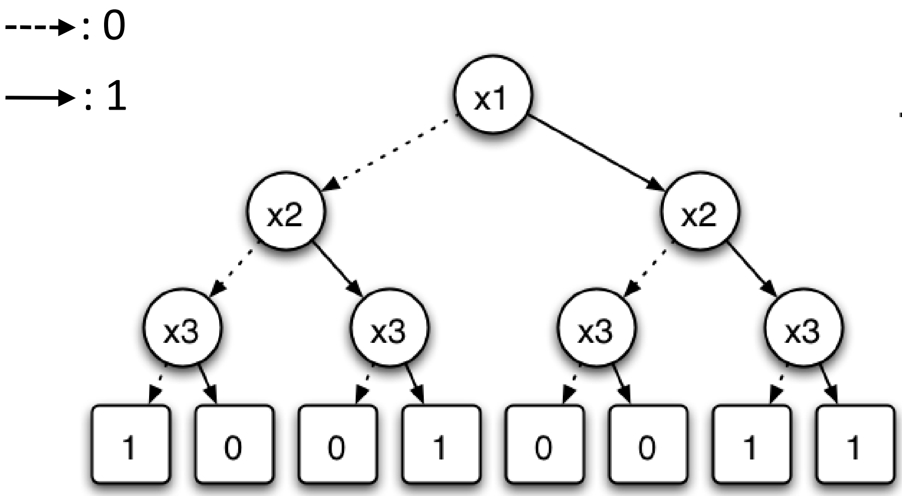
MDDs for integer optimization

- MDD *relaxations* provide upper bounds
- MDD *restrictions* provide lower bounds
- New branch-and-bound scheme

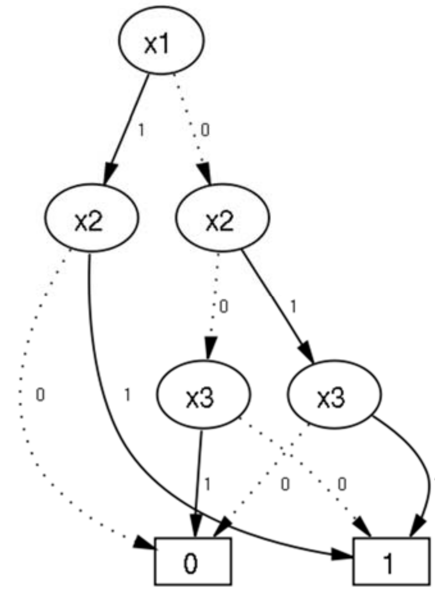
MDDs for constraint-based scheduling

- Constraint propagation with MDDs
- 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



$$f(x_1, x_2, x_3) = (\neg x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_2) \vee (x_2 \wedge x_3)$$

- 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
- Mid-2000s: 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
  - relaxed MDDs (polynomial size)

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

MDDs for combinatorial optimization

- Discrete Optimization
  - MISP, MAX-CUT, set covering, set packing, MAX-2SAT, …
- Constraint Programming
  - MDD propagation (*alldifferent*, *sequence*, …)
- Scheduling and Sequencing
  - Machine scheduling, routing, …
- Integer Programming
  - Cut generation
- Boolean Satisfiability
  - Clause learning

Exact MDDs for discrete optimization

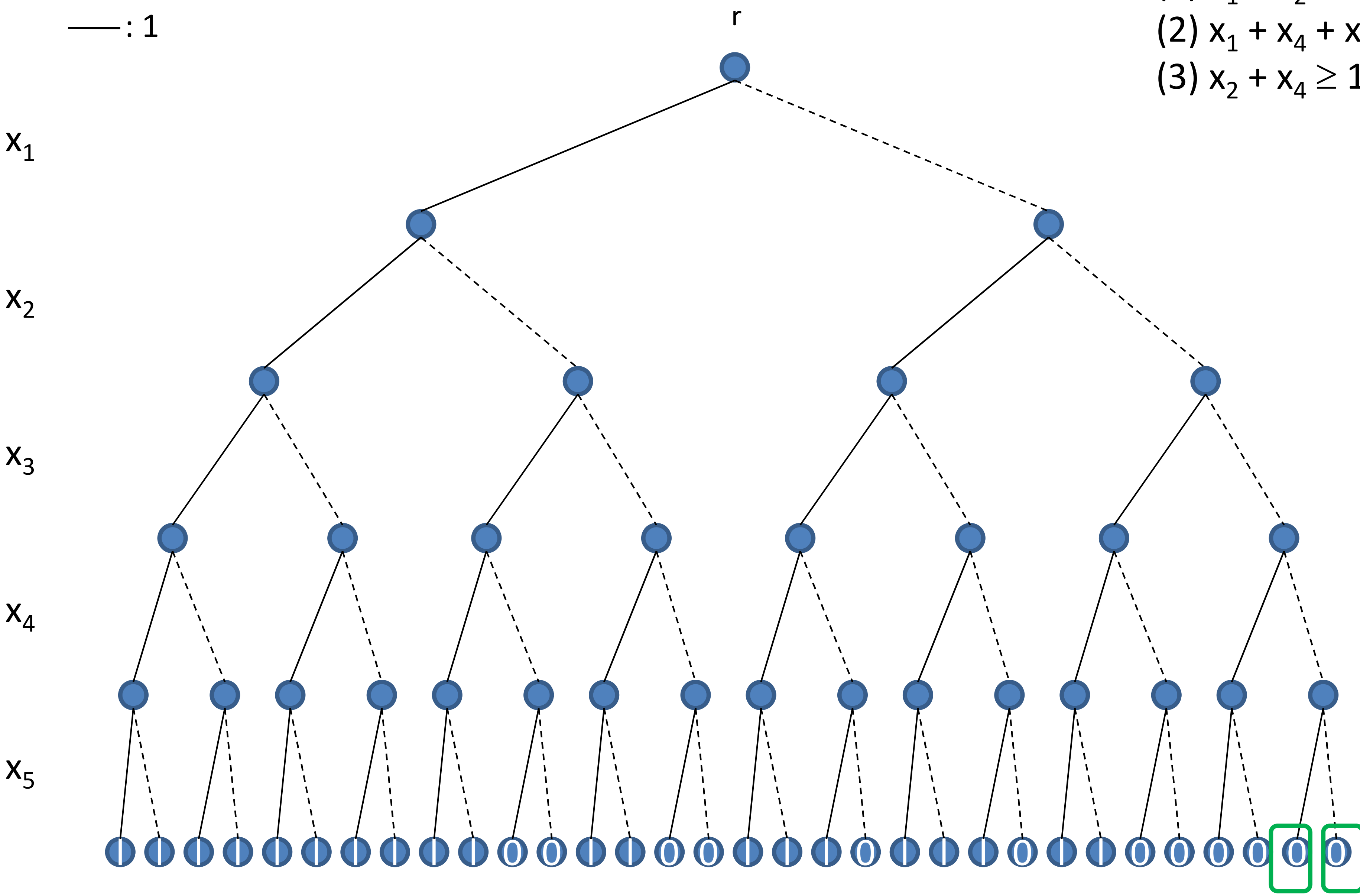
- - - - : 0

——: 1

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

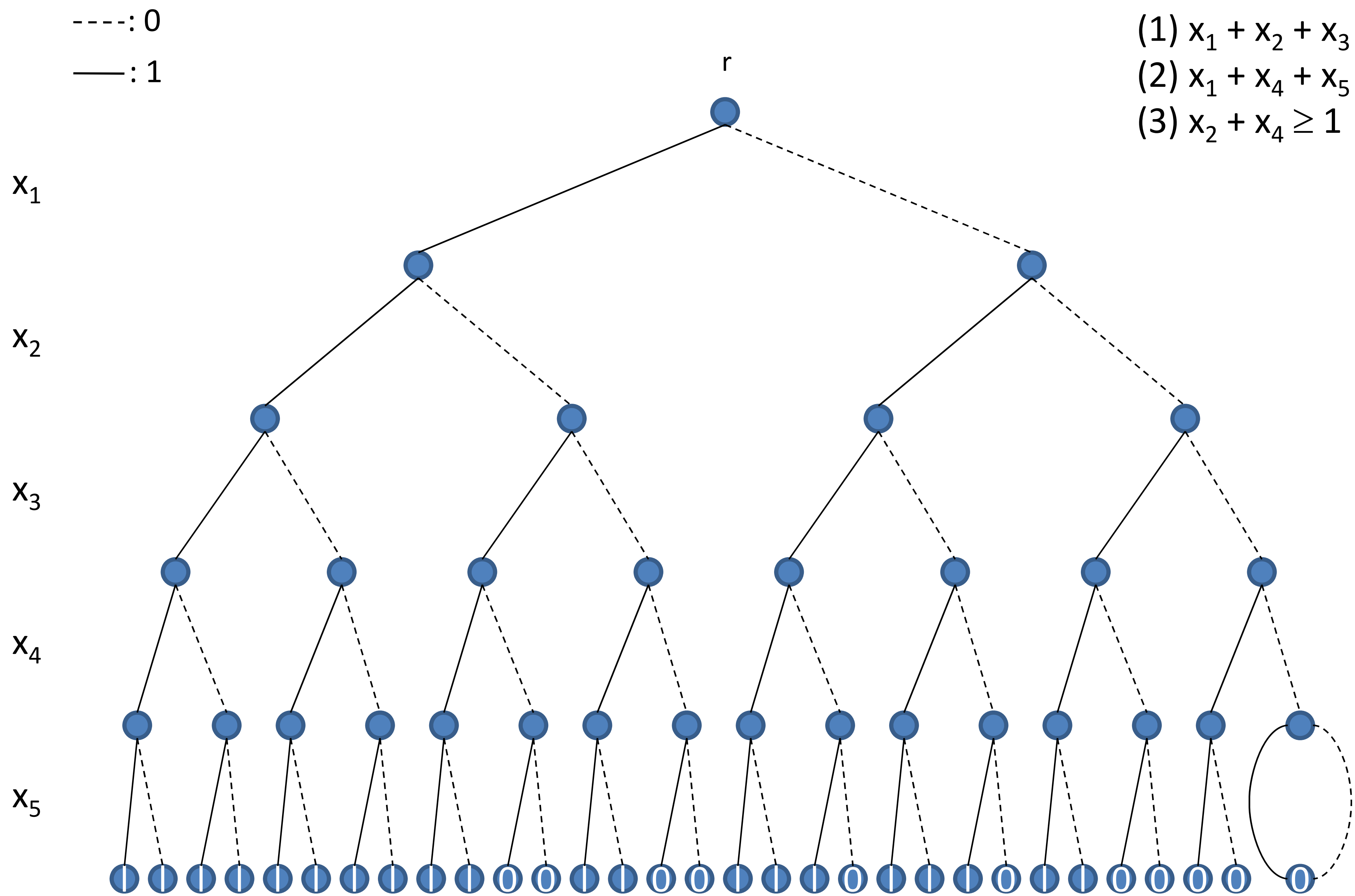
(2) $x_1 + x_4 + x_5 \geq 1$

(3) $x_2 + x_4 \geq 1$



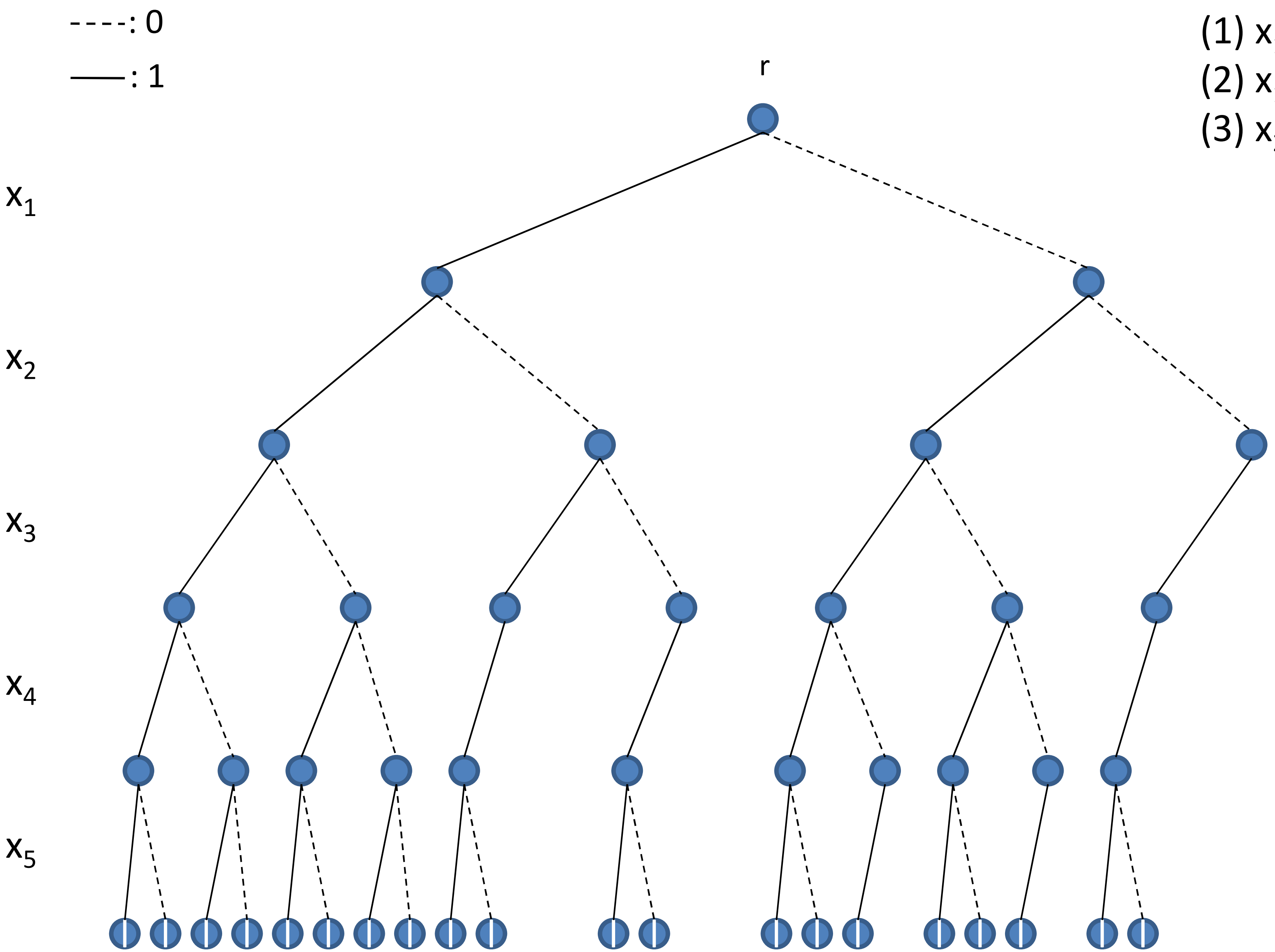
Exact MDDs for discrete optimization

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



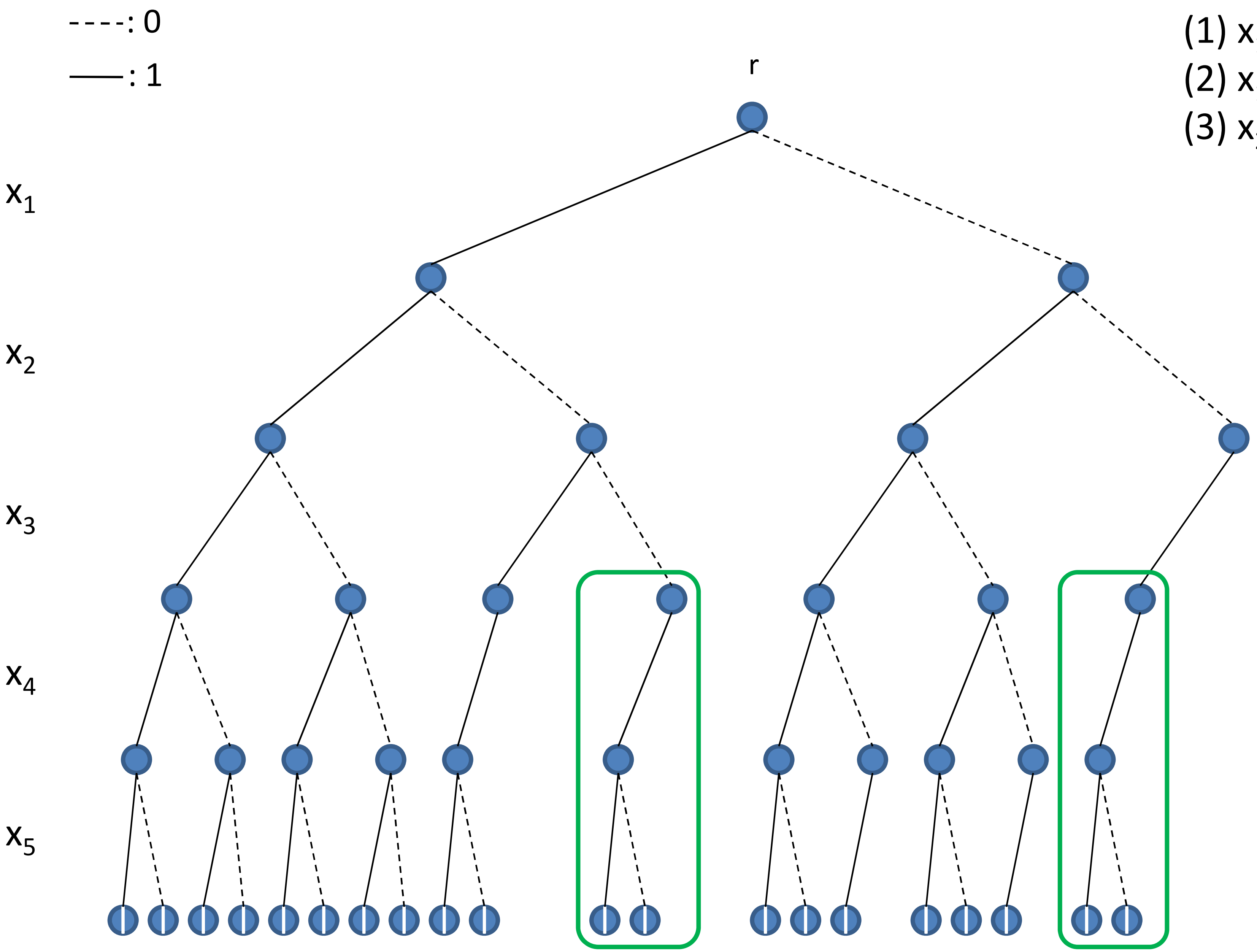
Exact MDDs for discrete optimization

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

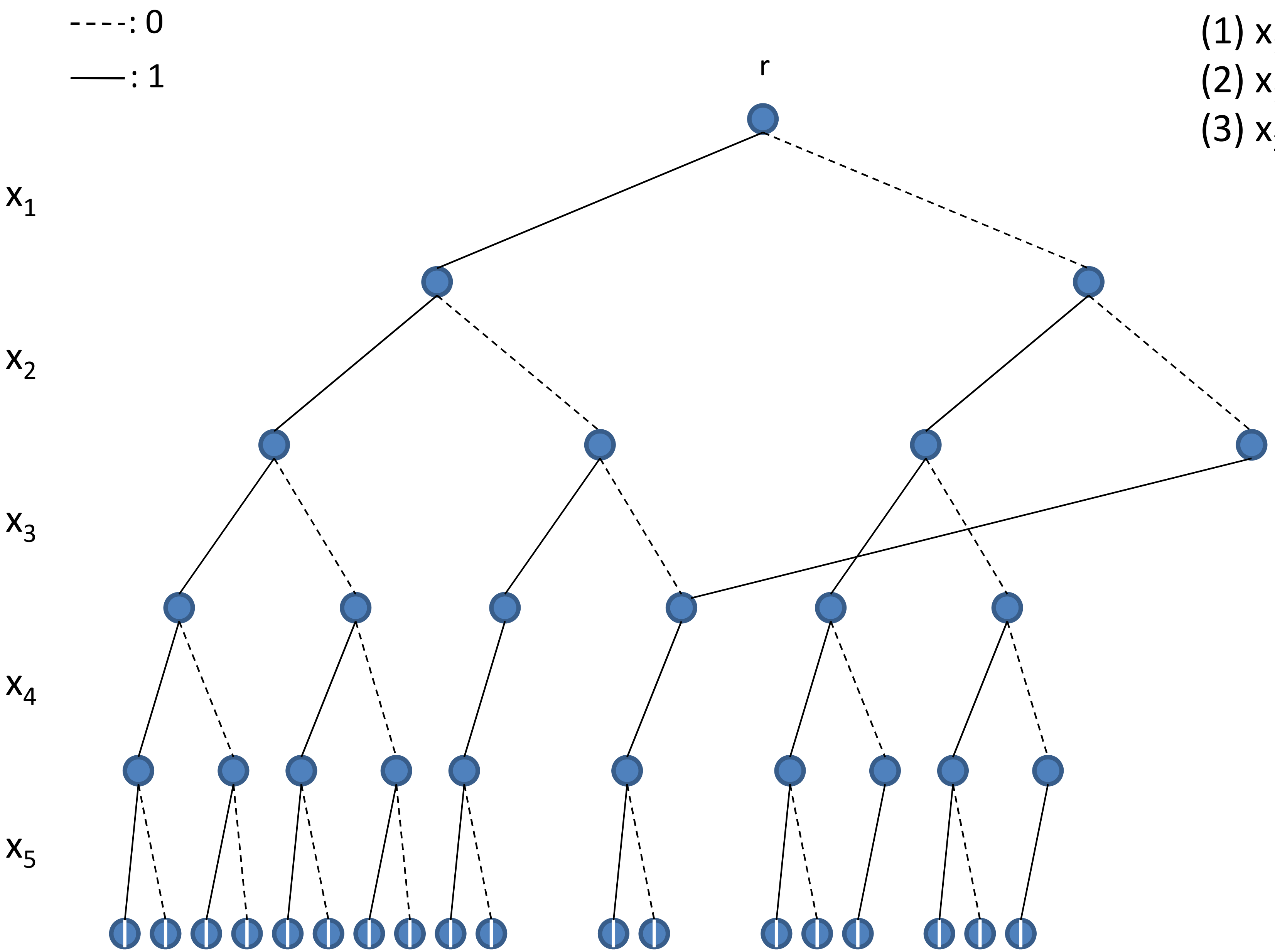


Exact MDDs for discrete optimization

(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

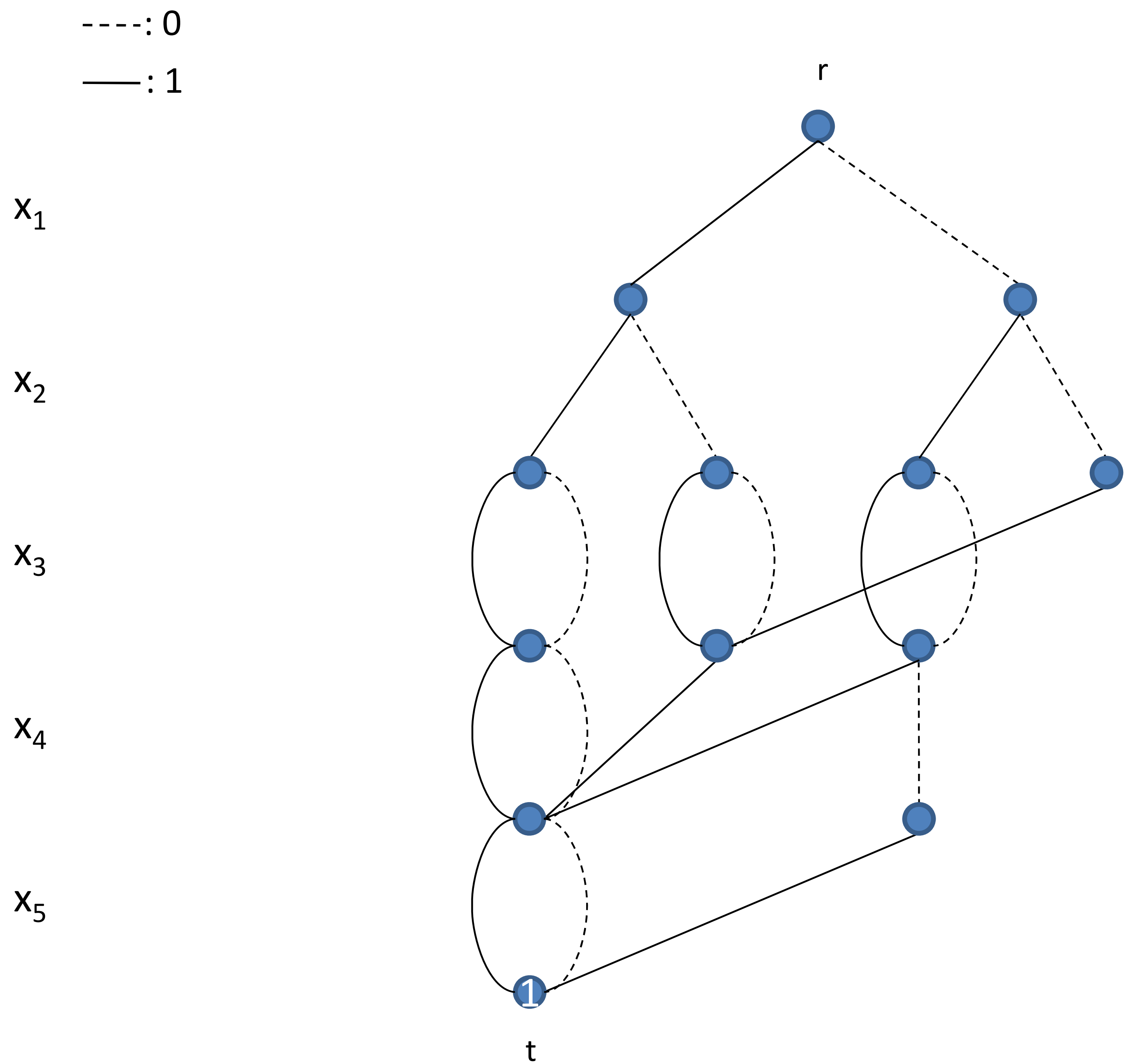


(1) $x_1 + x_2 + x_3 \geq 1$

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

(3) $x_2 + x_4 \geq 1$

Exact MDDs for discrete optimization

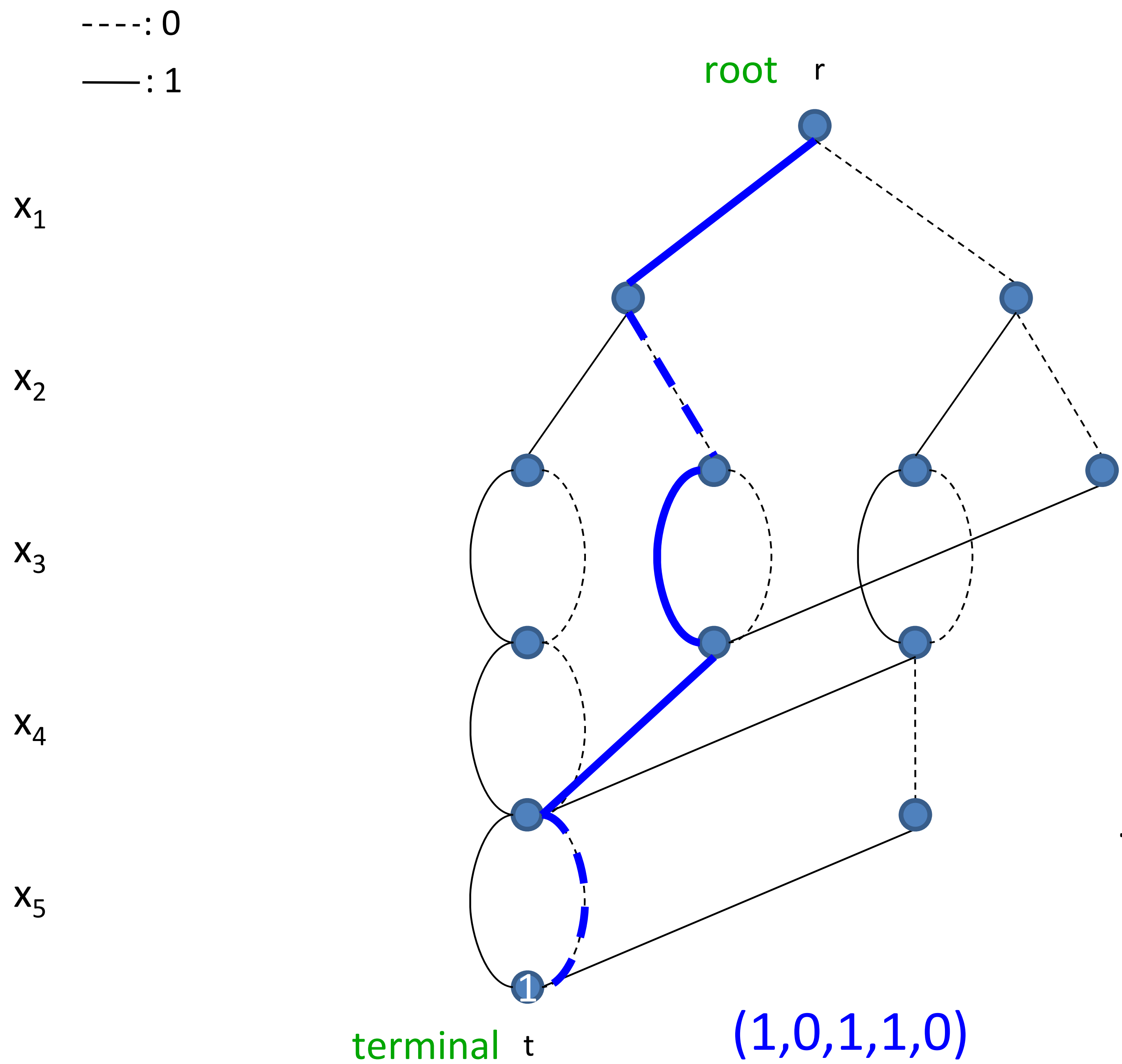


(1) $x_1 + x_2 + x_3 \geq 1$

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

(3) $x_2 + x_4 \geq 1$

Exact MDDs for discrete optimization



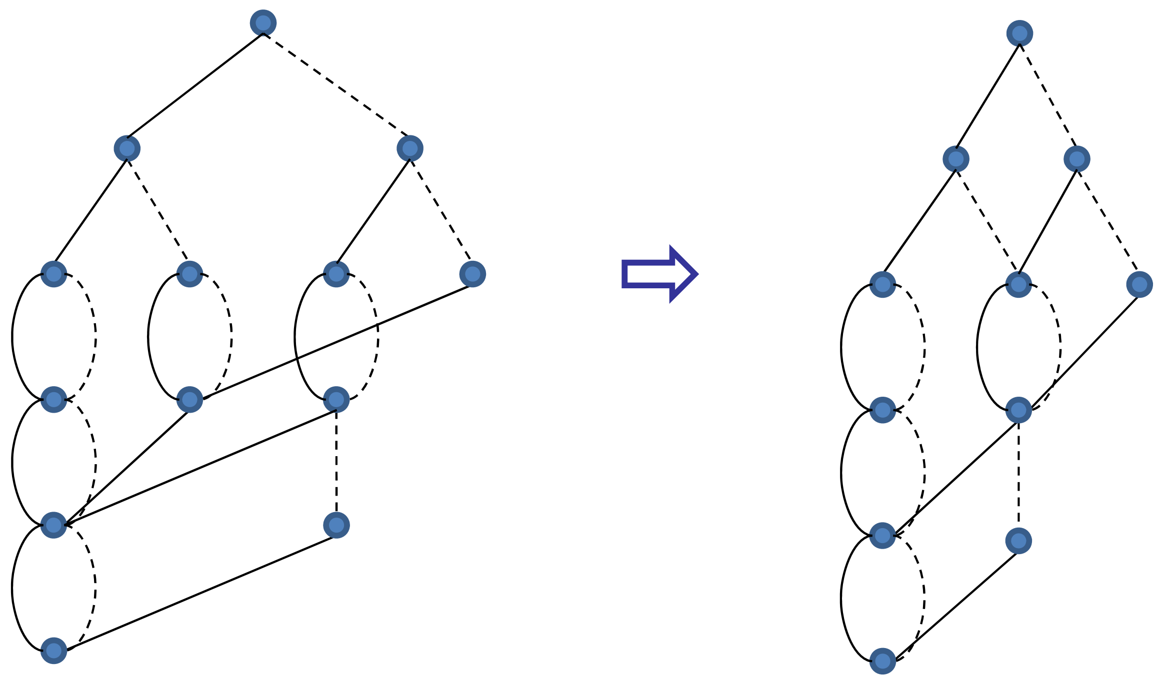
(1) $x_1 + x_2 + x_3 \geq 1$
(2) $x_1 + x_4 + x_5 \geq 1$
(3) $x_2 + x_4 \geq 1$

Each path corresponds to a solution

(1,0,1,1,0)

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 maximum width



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. *INFORMS J. Computing*, to appear.
- 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

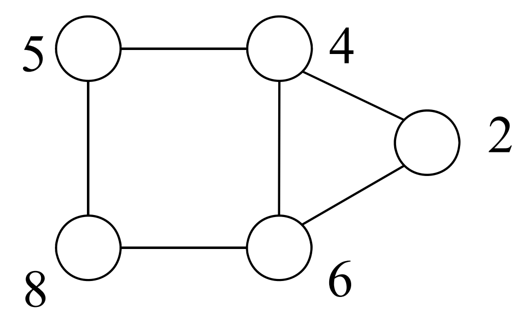
Case Study: Independent Set Problem

- 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

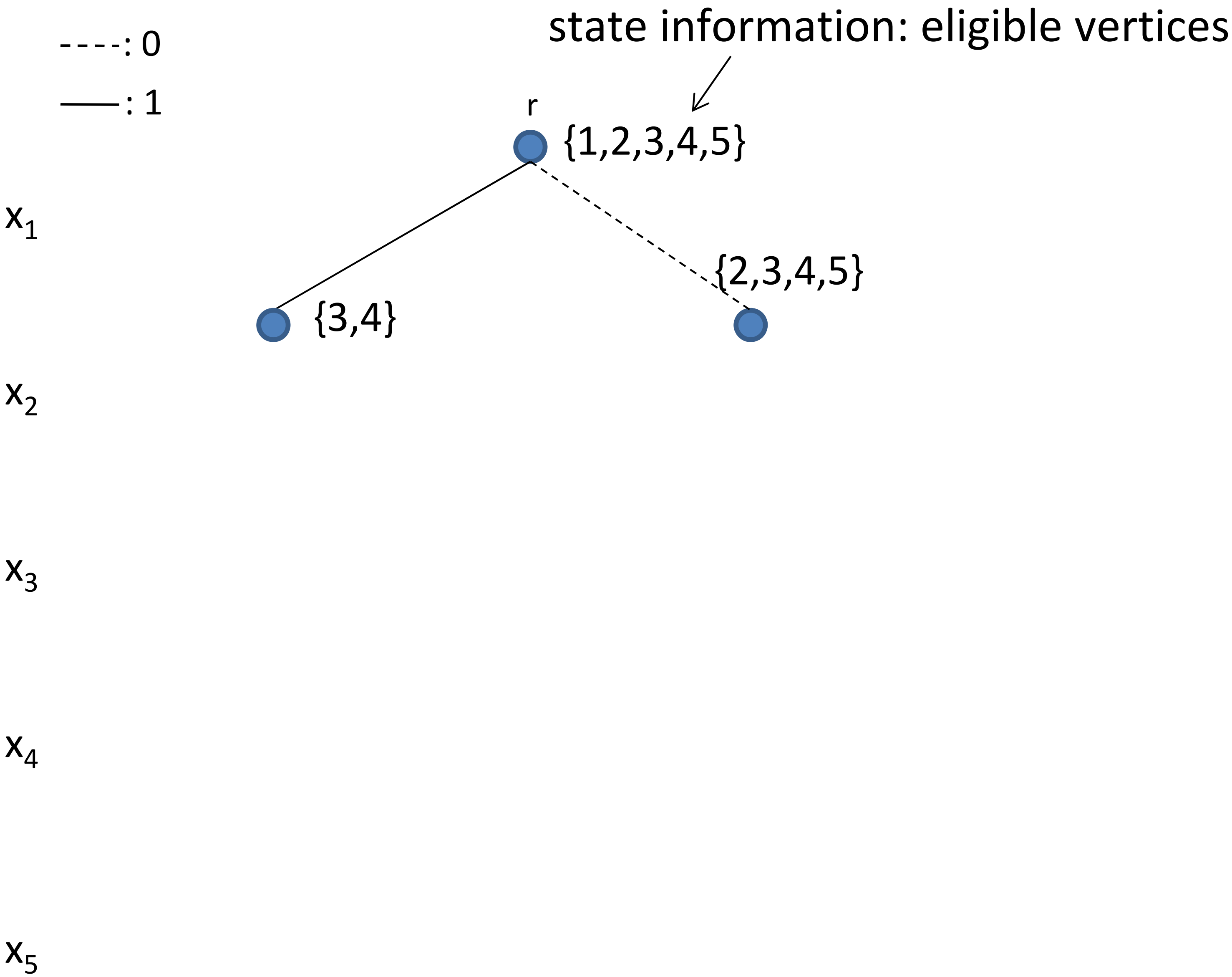
max $\sum_i w_i x_i$

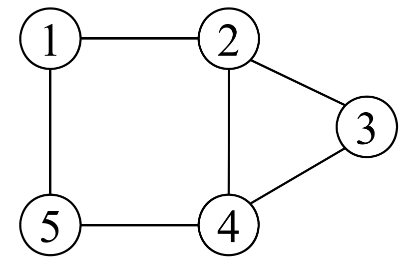
s.t. $x_i + x_j \leq 1$ for all (i,j) in E

x_i binary for all i in V

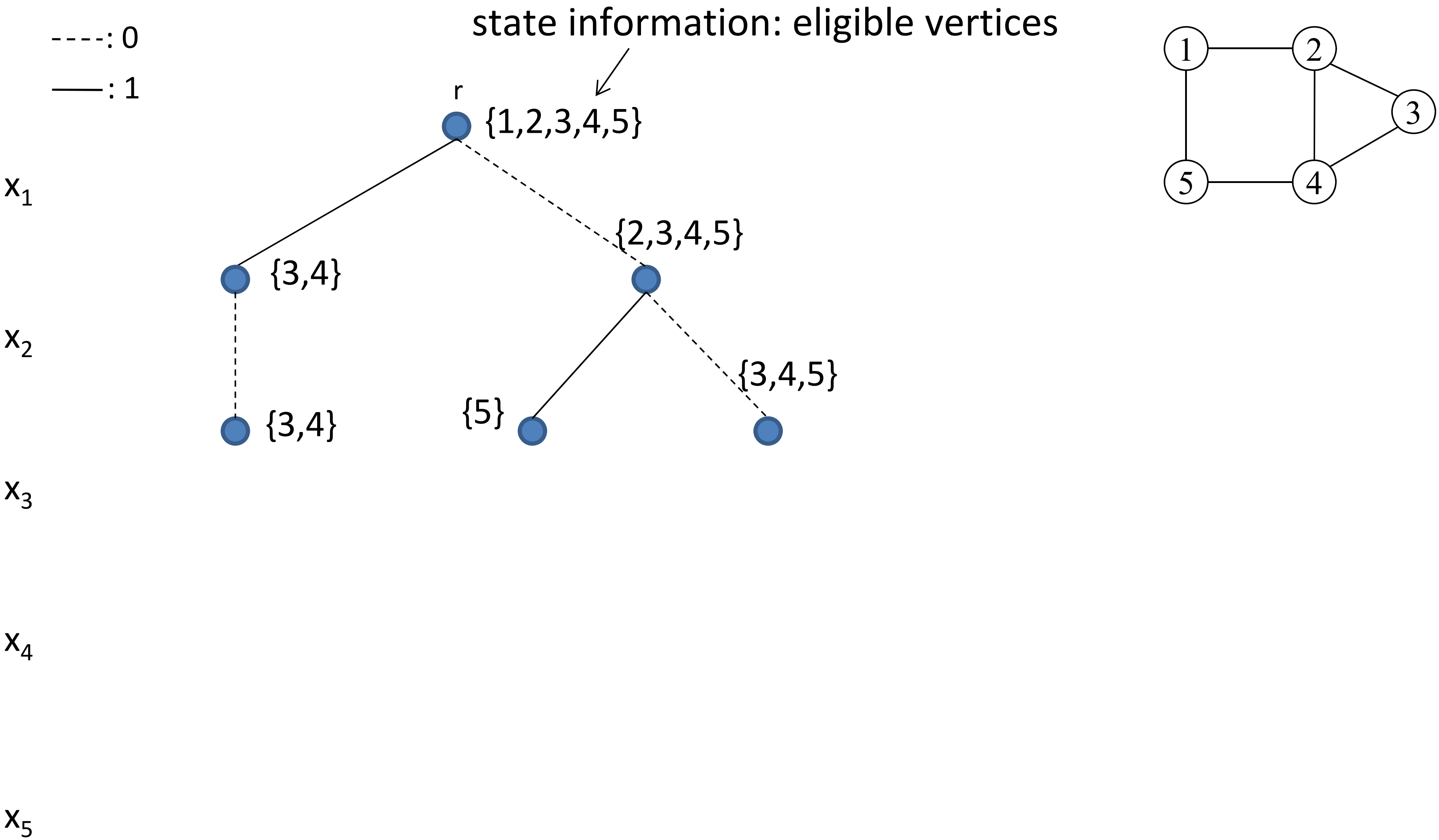


Exact top-down compilation

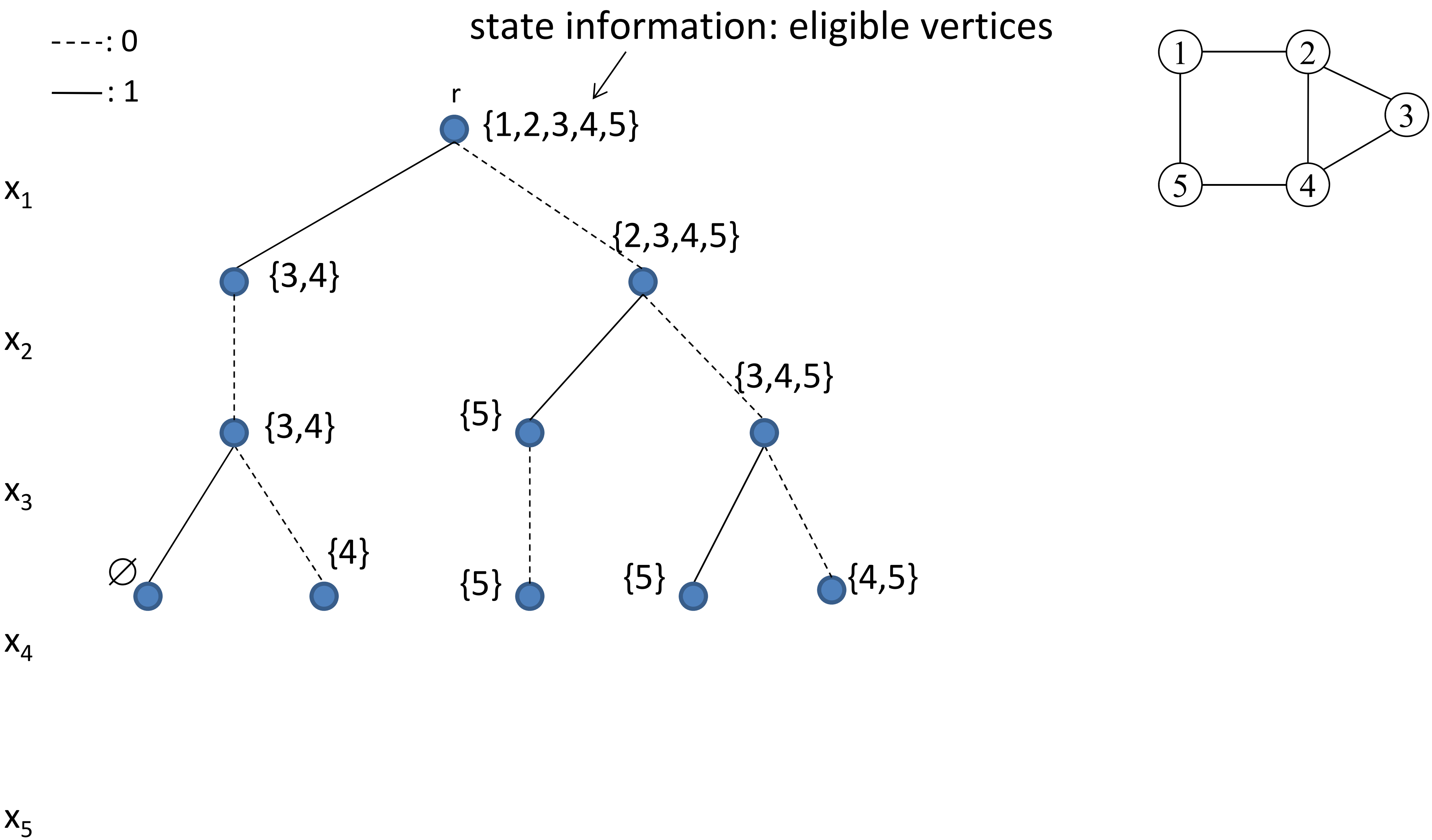




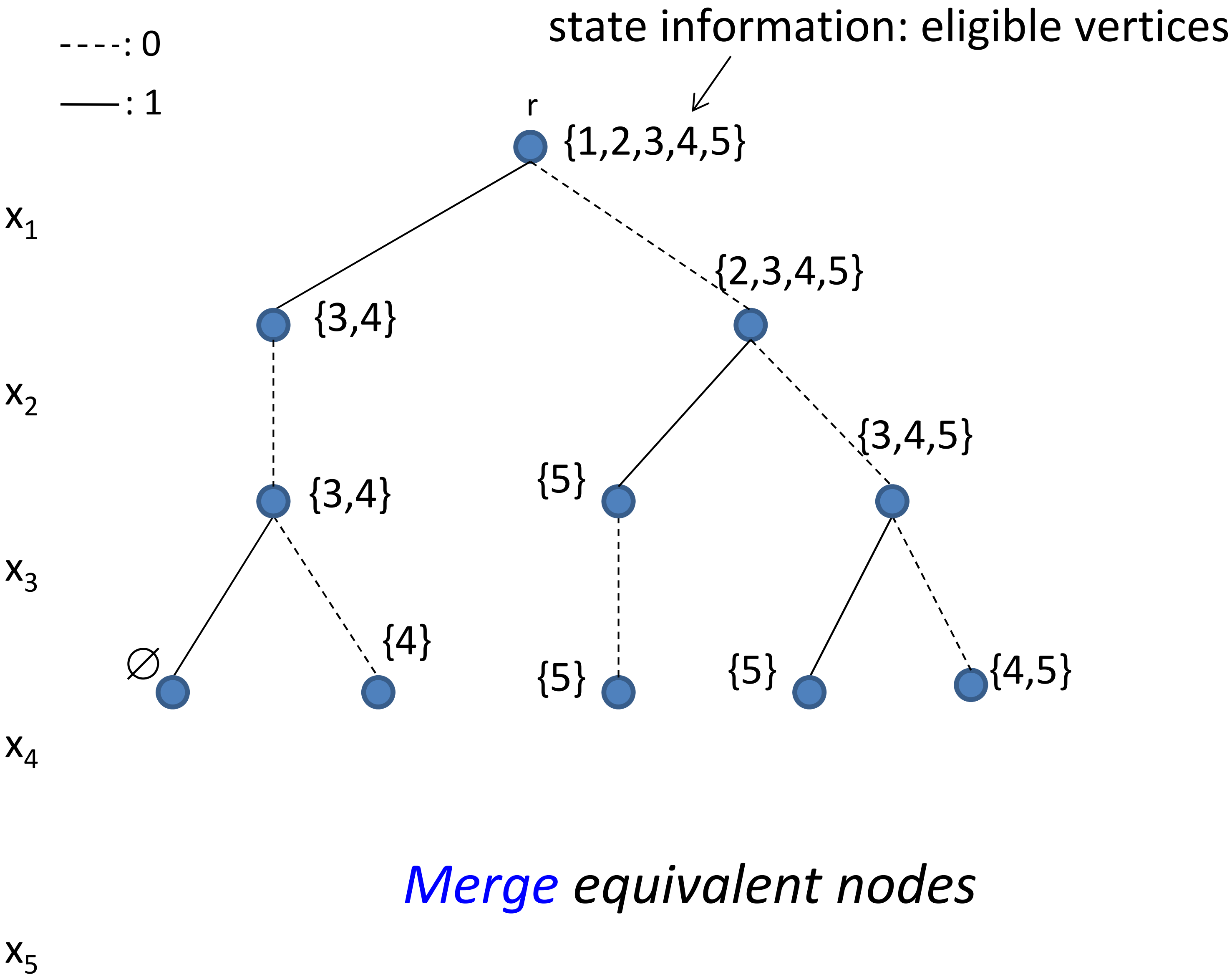
Exact top-down compilation

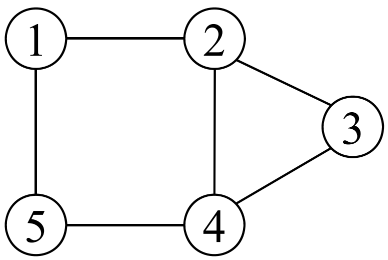


Exact top-down compilation



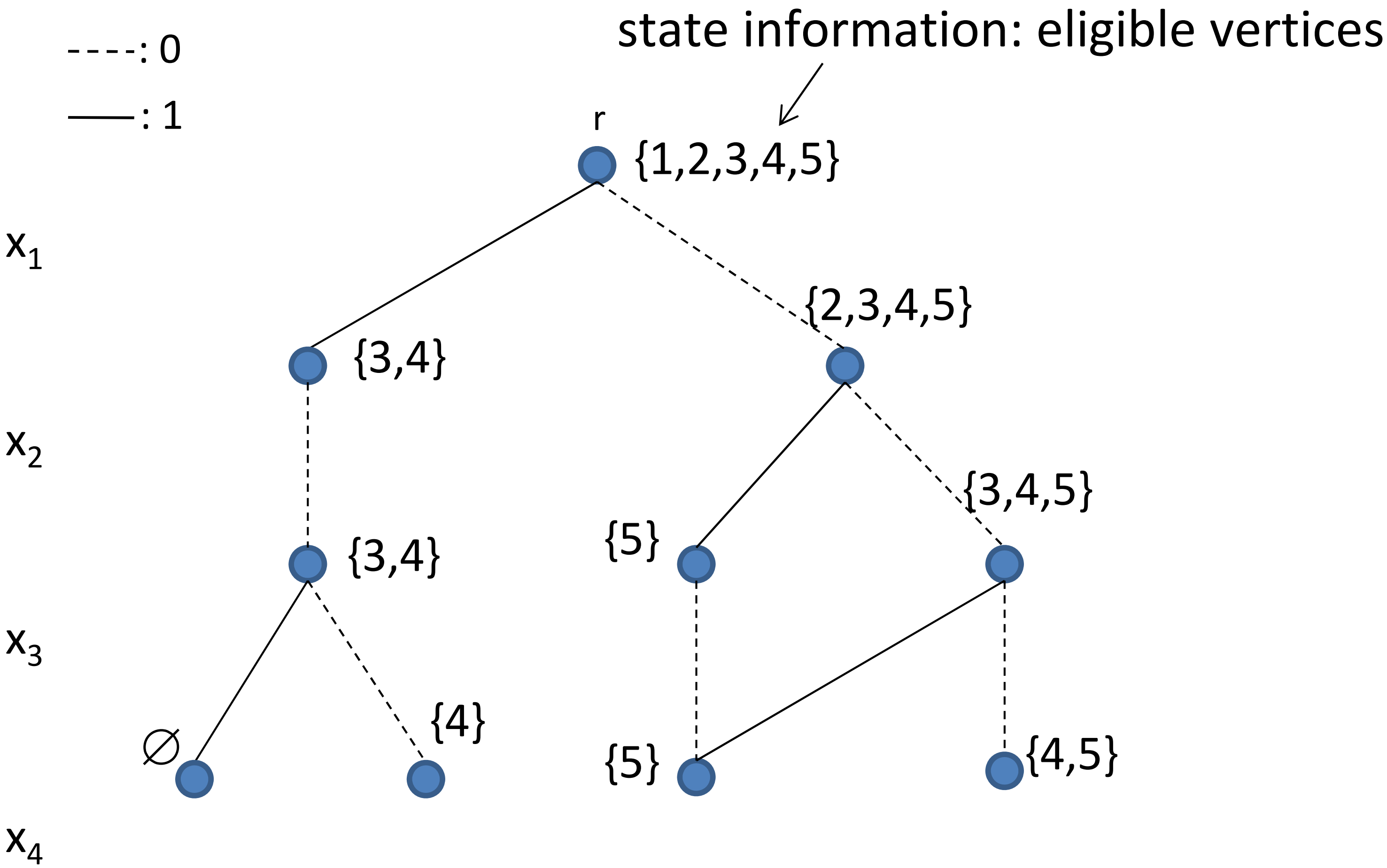
Exact top-down compilation

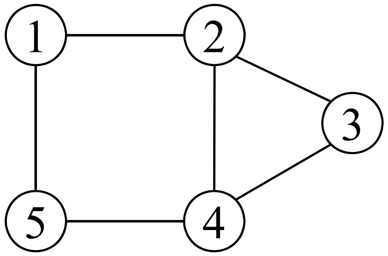




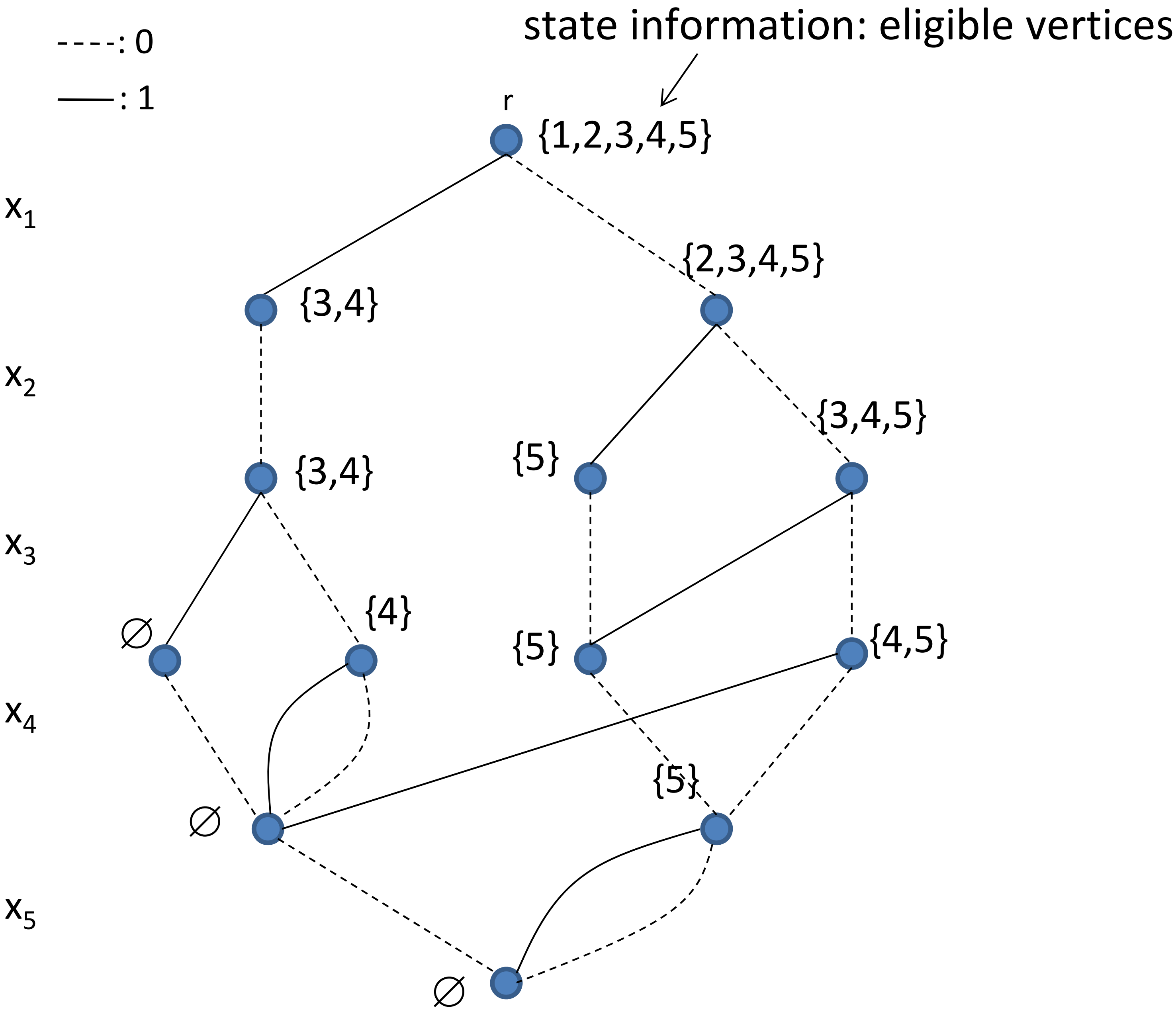
Merge equivalent nodes

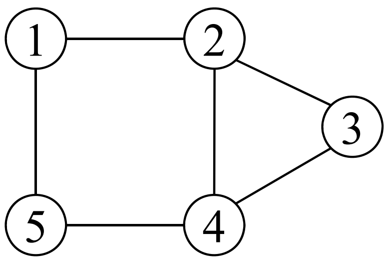
Node Merging



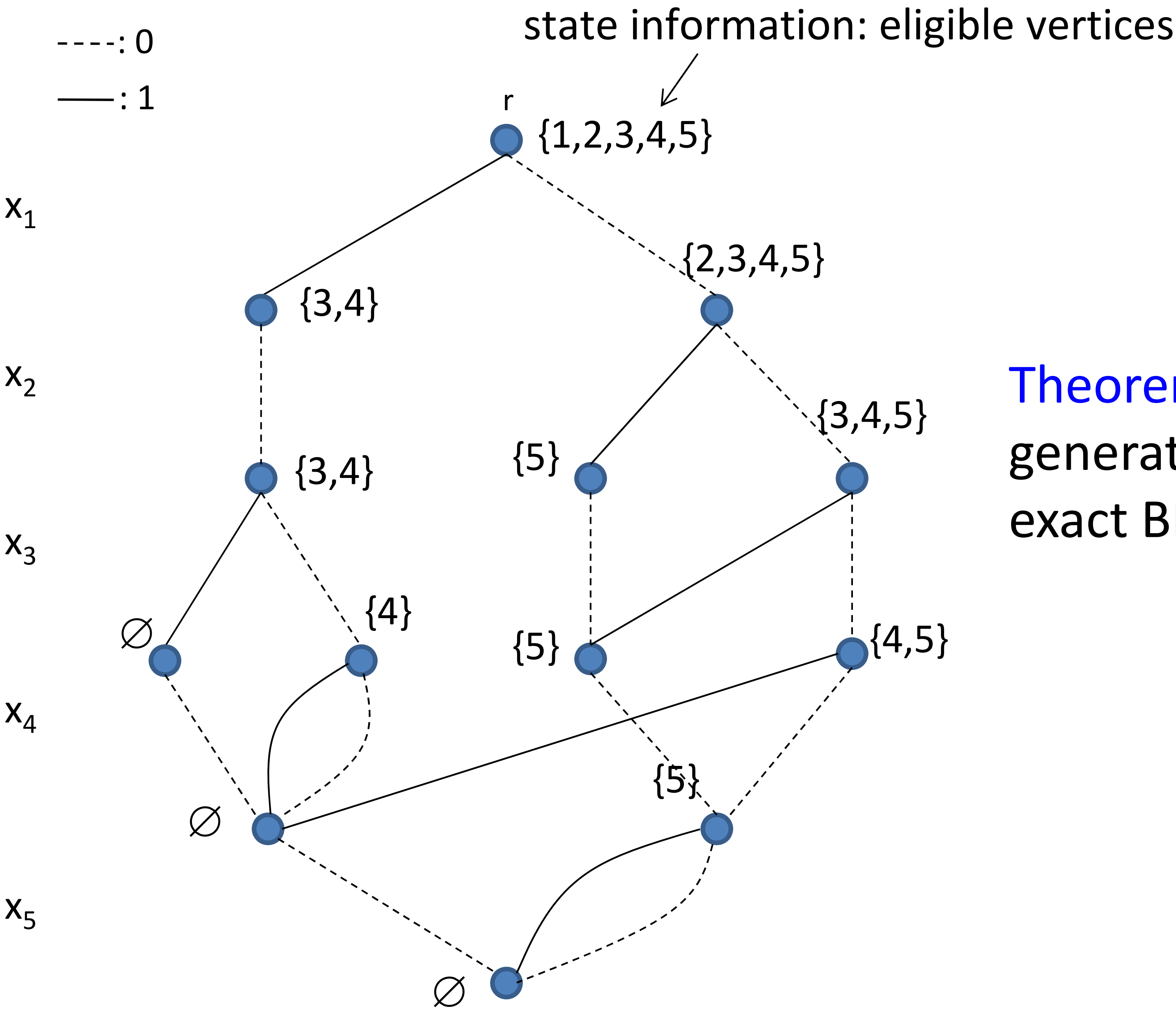


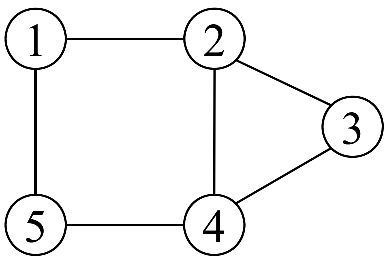
Node Merging





Node Merging

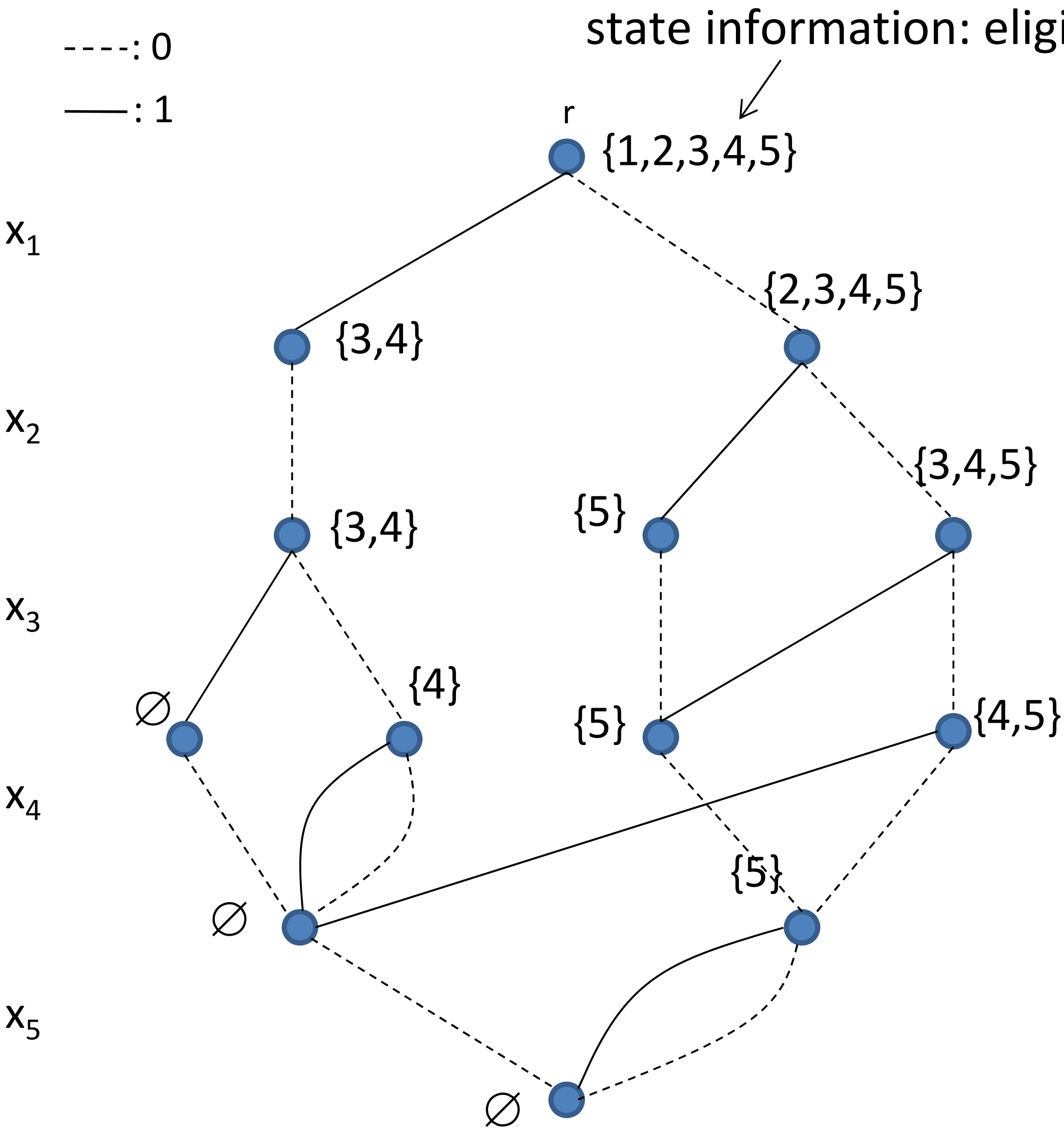


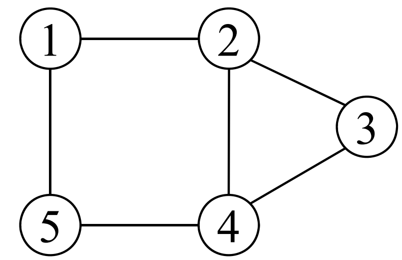


Theorem: This procedure generates a reduced exact BDD

[Bergman et al., 2012]

Node Merging





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

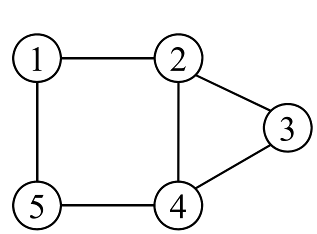
[Bergman et al., 2012]

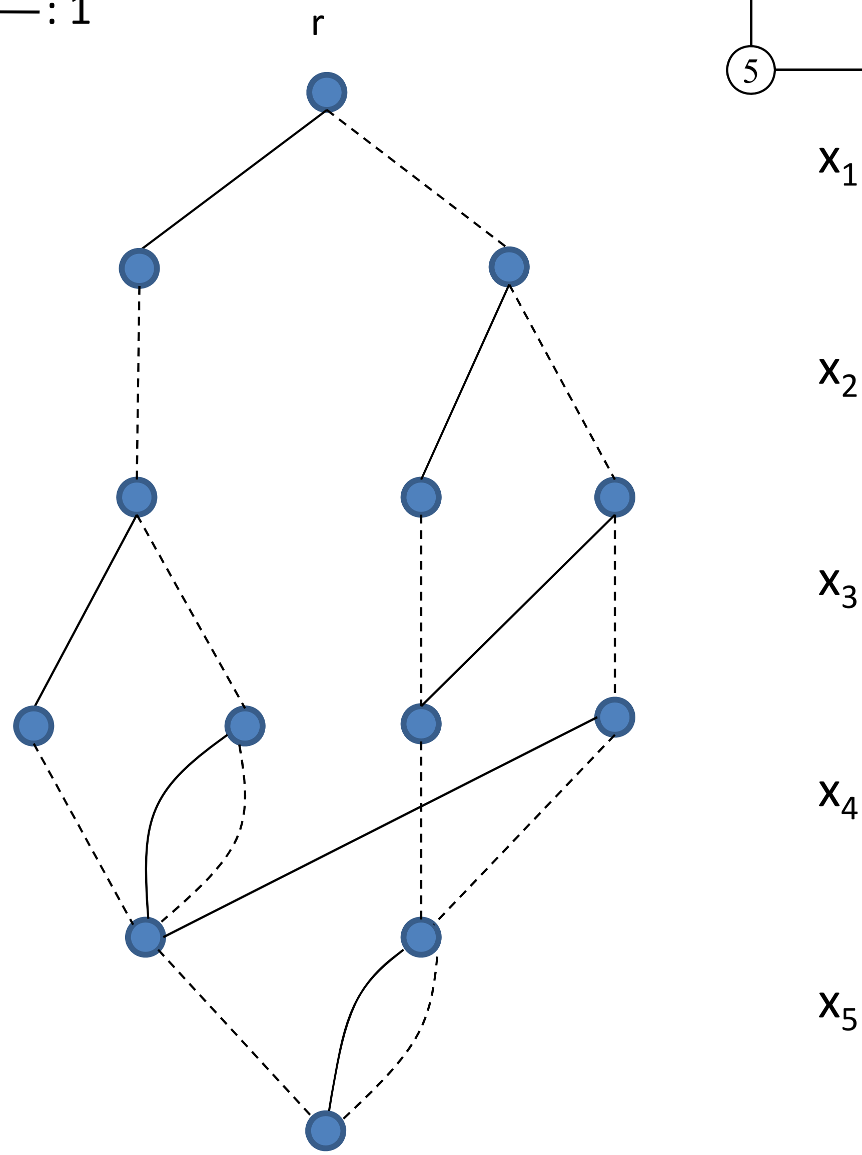
Relaxed BDD

- - - - : 0

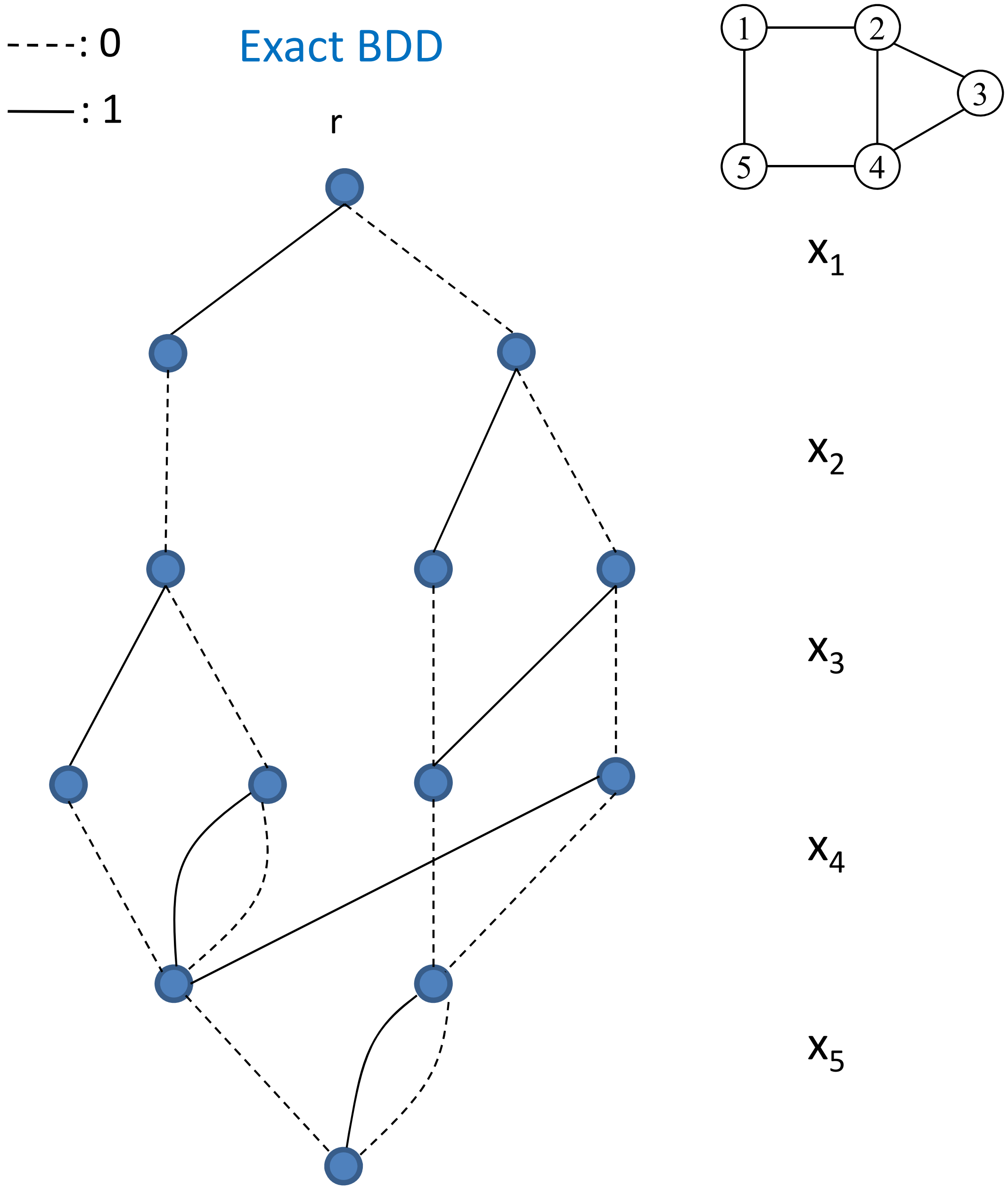
——— : 1

Exact BDD



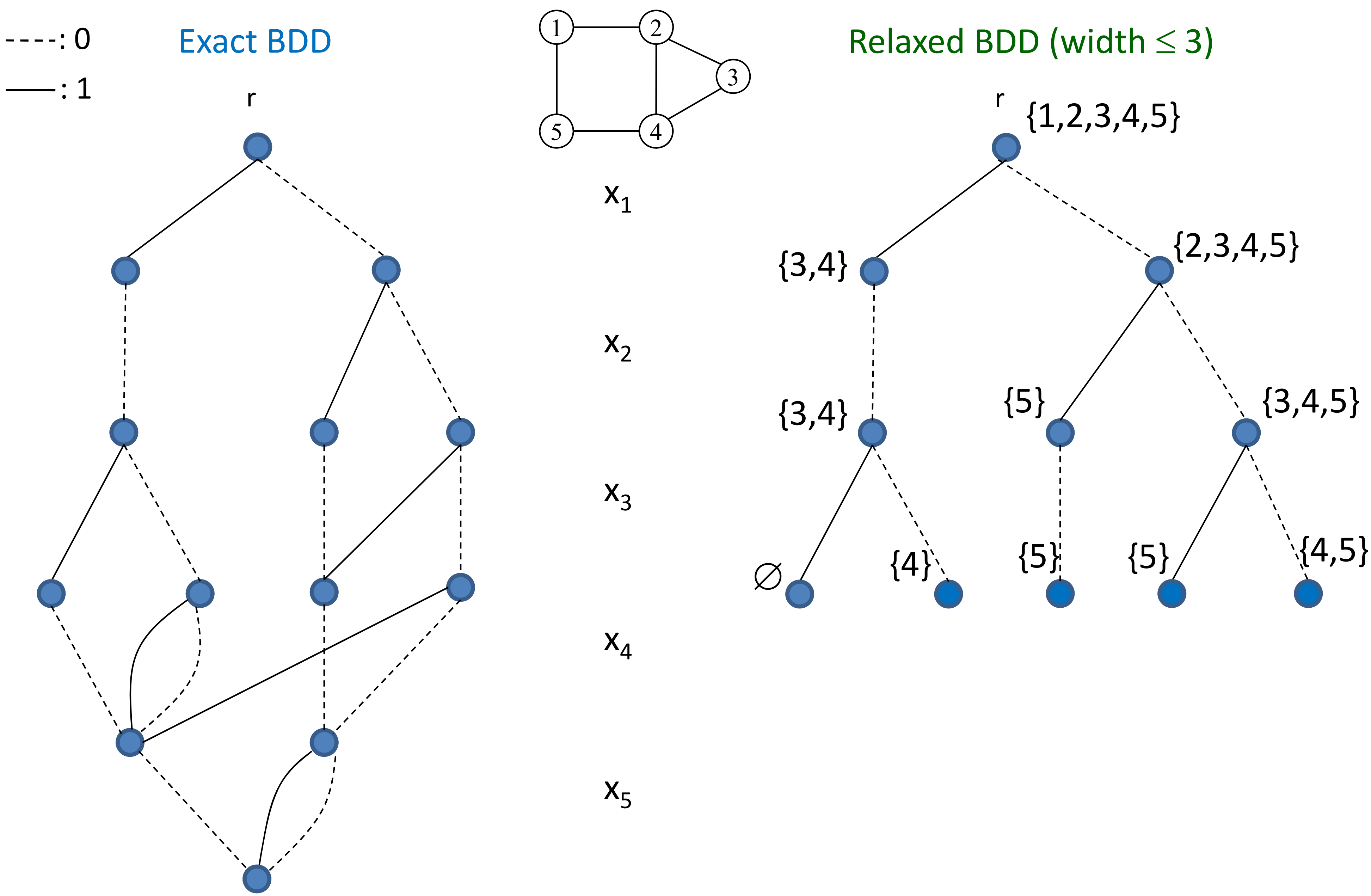


Relaxed BDD



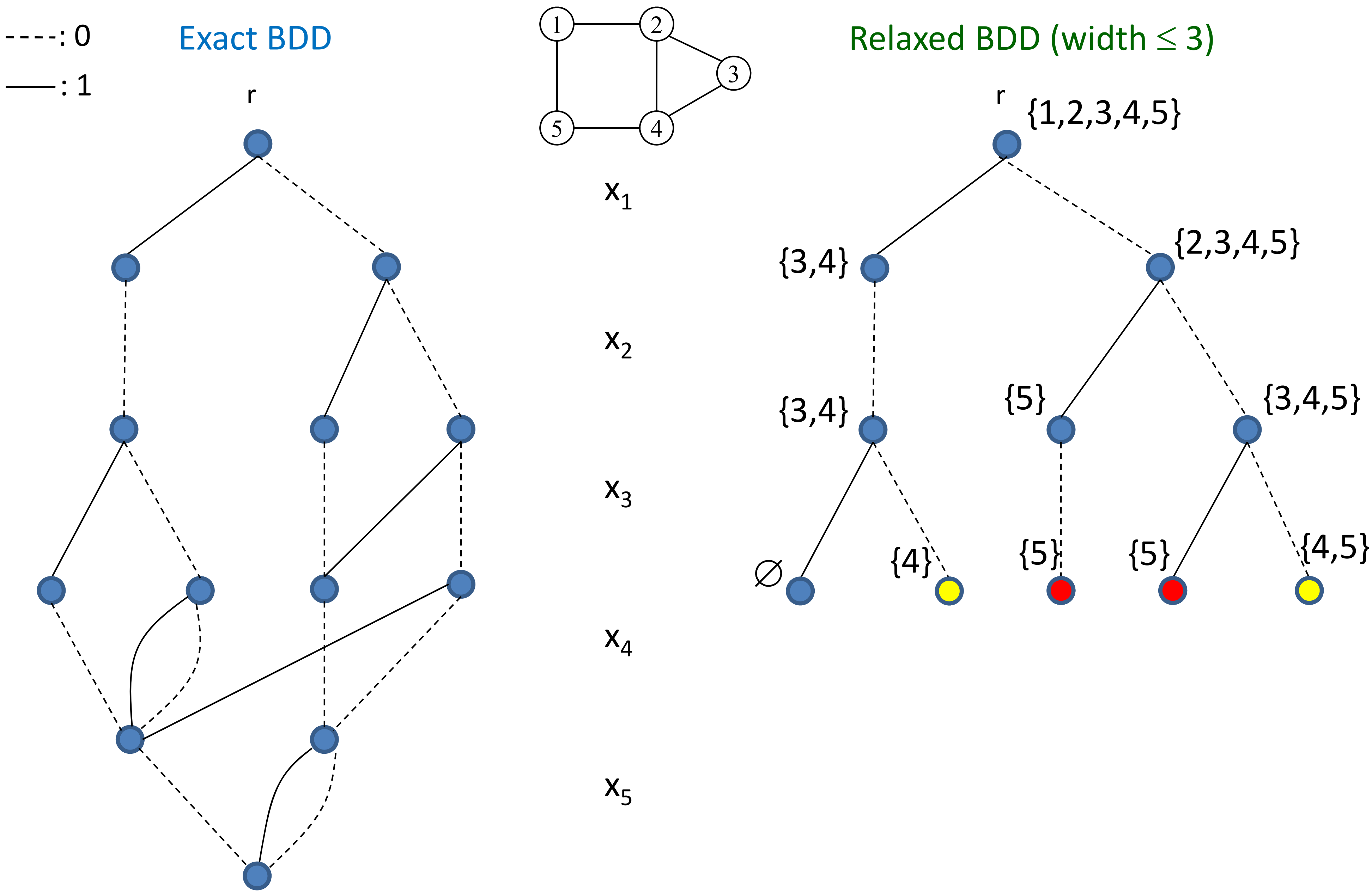
Relaxed BDD (width ≤ 3)

Relaxed BDD



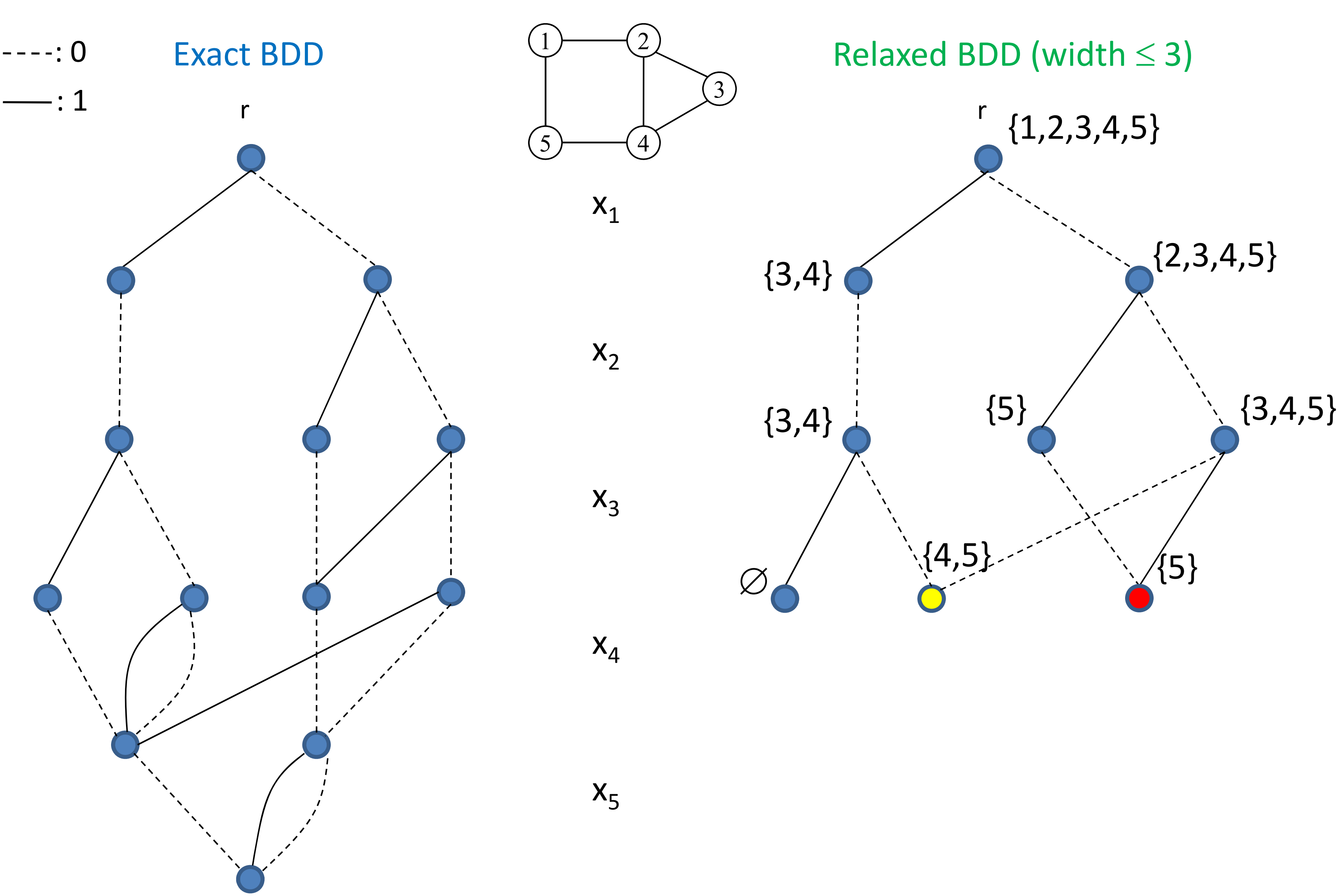
- - - -: 0
——: 1
Exact BDD
r
1
2
3
4
5
x_1
x_2
x_3
x_4
x_5
Relaxed BDD (width ≤ 3)
r
{1,2,3,4,5}
{3,4}
{2,3,4,5}
{3,4}
{5}
{3,4,5}
∅
{4}
{5}
{5}
{4,5}

Relaxed BDD



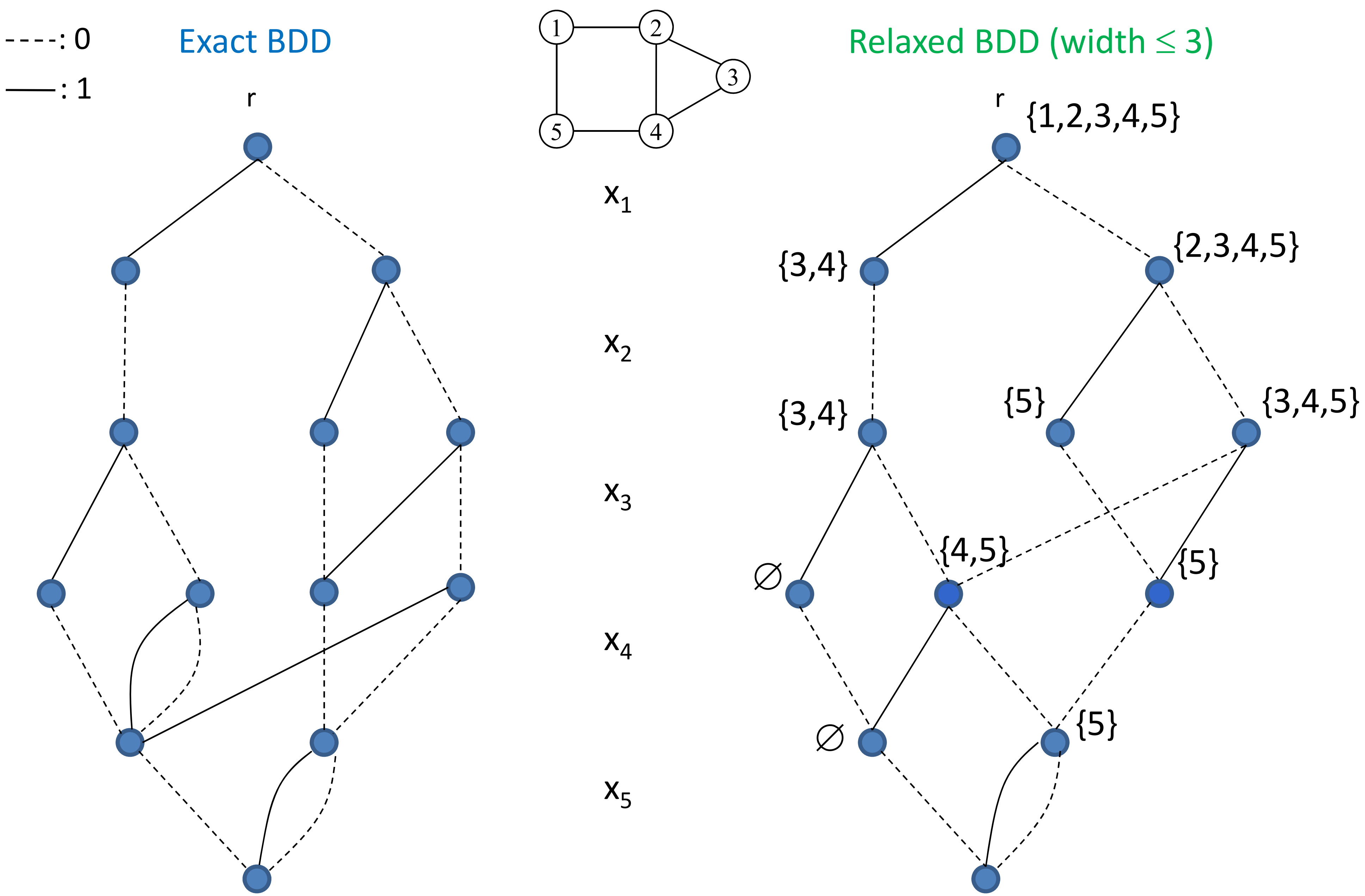
- - - -: 0
——: 1
Exact BDD
r
1
2
3
4
5
x_1
x_2
x_3
x_4
x_5
Relaxed BDD (width ≤ 3)
r
{1,2,3,4,5}
{3,4}
{2,3,4,5}
{3,4}
{5}
{3,4,5}
∅
{4}
{5}
{5}
{4,5}

Relaxed BDD



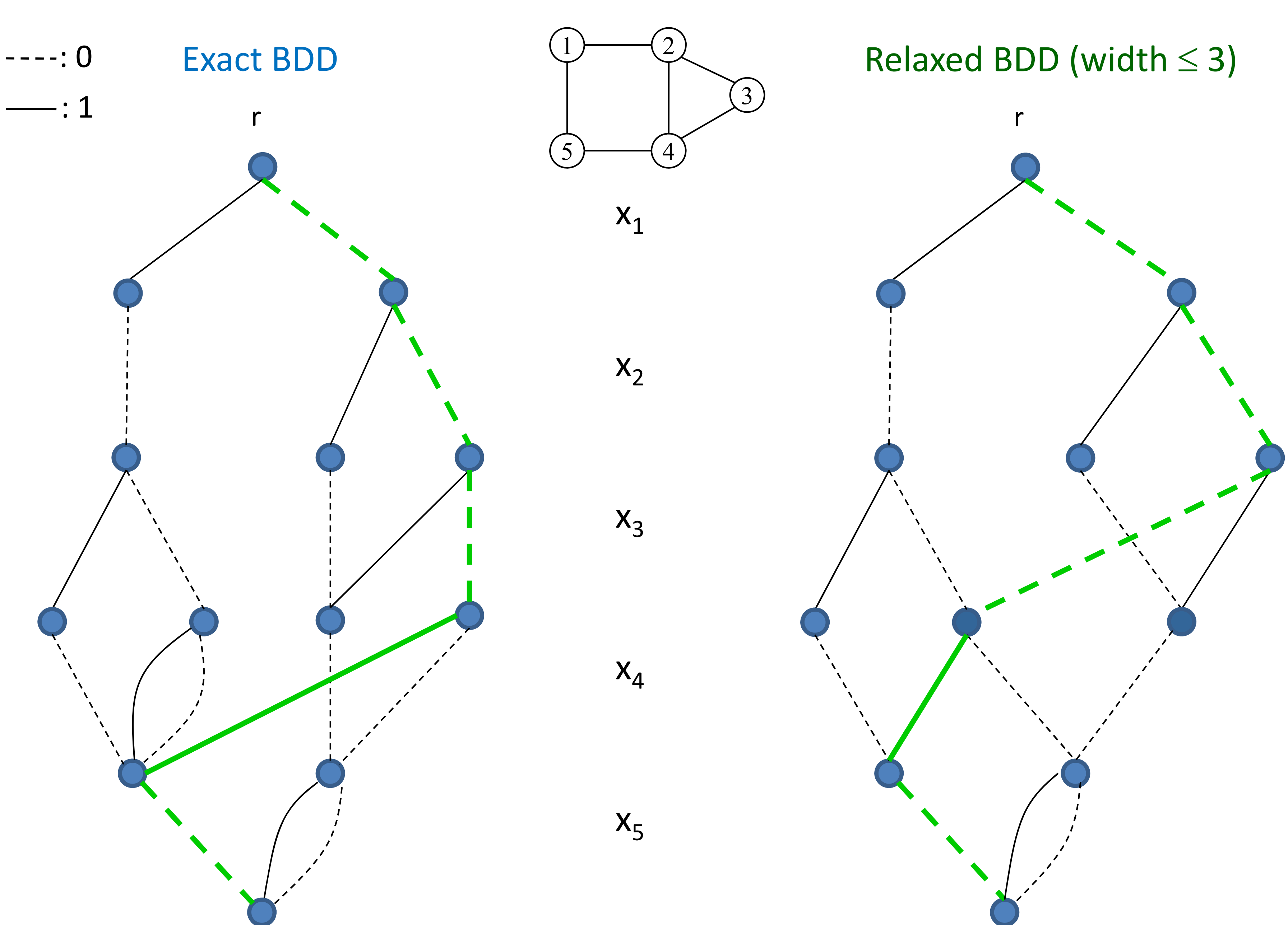
- - - -: 0
——: 1
Exact BDD
r
Relaxed BDD (width ≤ 3)
r
{1,2,3,4,5}
{3,4}
{2,3,4,5}
{3,4}
{5}
{3,4,5}
∅
{4,5}
{5}
x1
x2
x3
x4
x5
1
2
3
4
5

Relaxed BDD



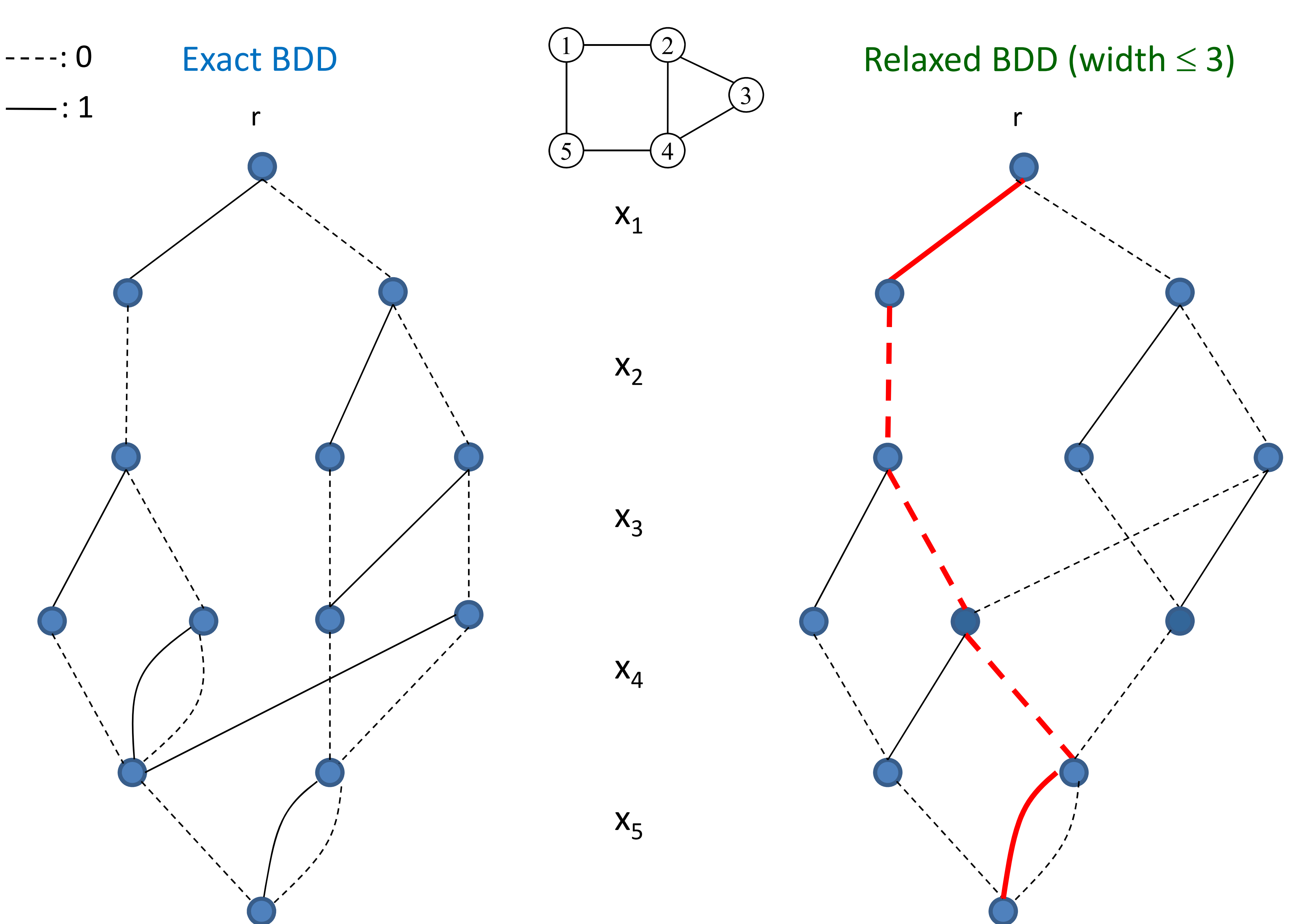
- - - -: 0
——: 1
Exact BDD
r
1
2
3
4
5
x_1
x_2
x_3
x_4
x_5
Relaxed BDD (width ≤ 3)
r
{1,2,3,4,5}
{3,4}
{2,3,4,5}
{3,4}
{5}
{3,4,5}
∅
{4,5}
{5}
∅
{5}

Relaxed BDD



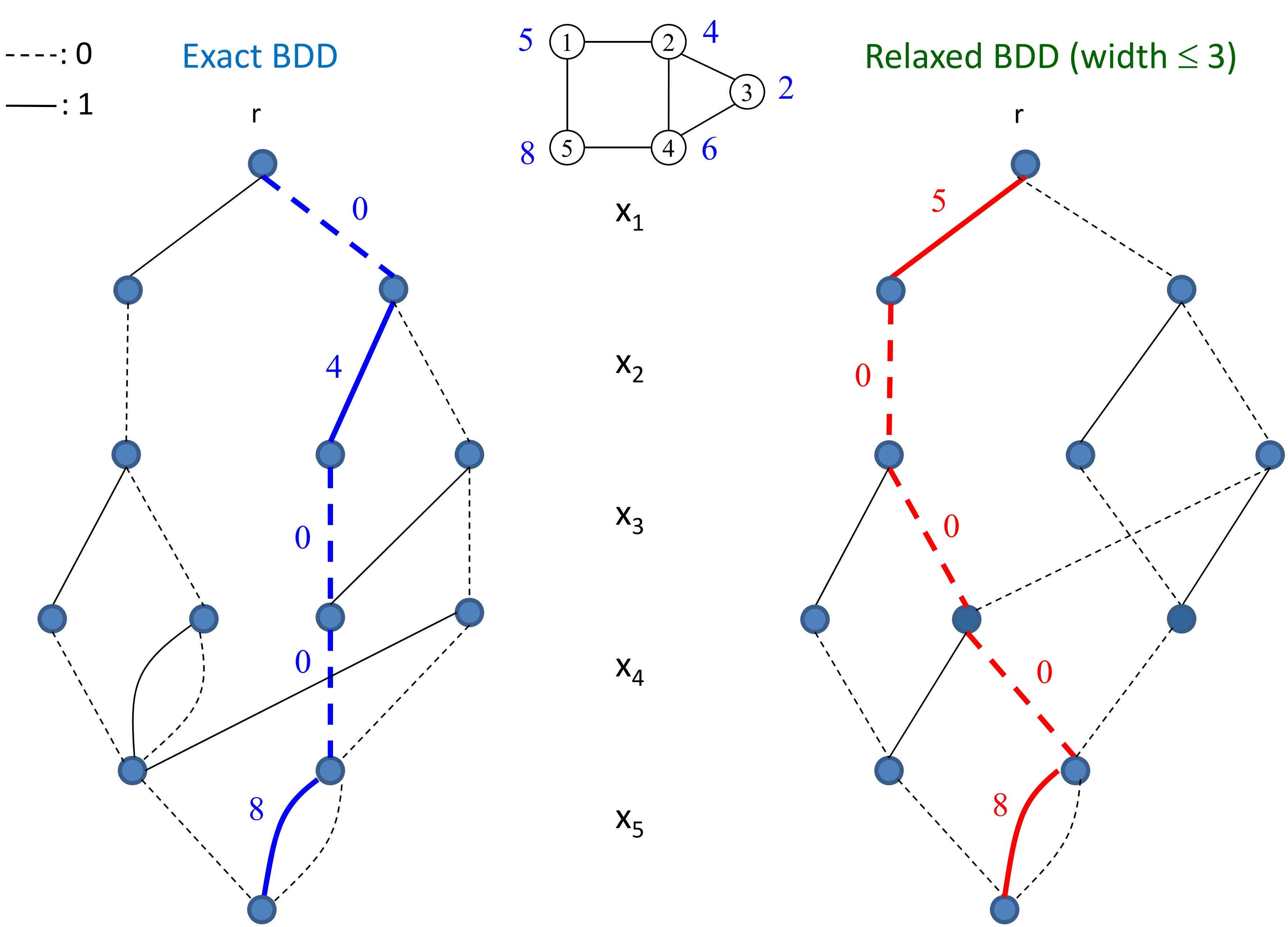
(0,0,0,1,0)

Relaxed BDD

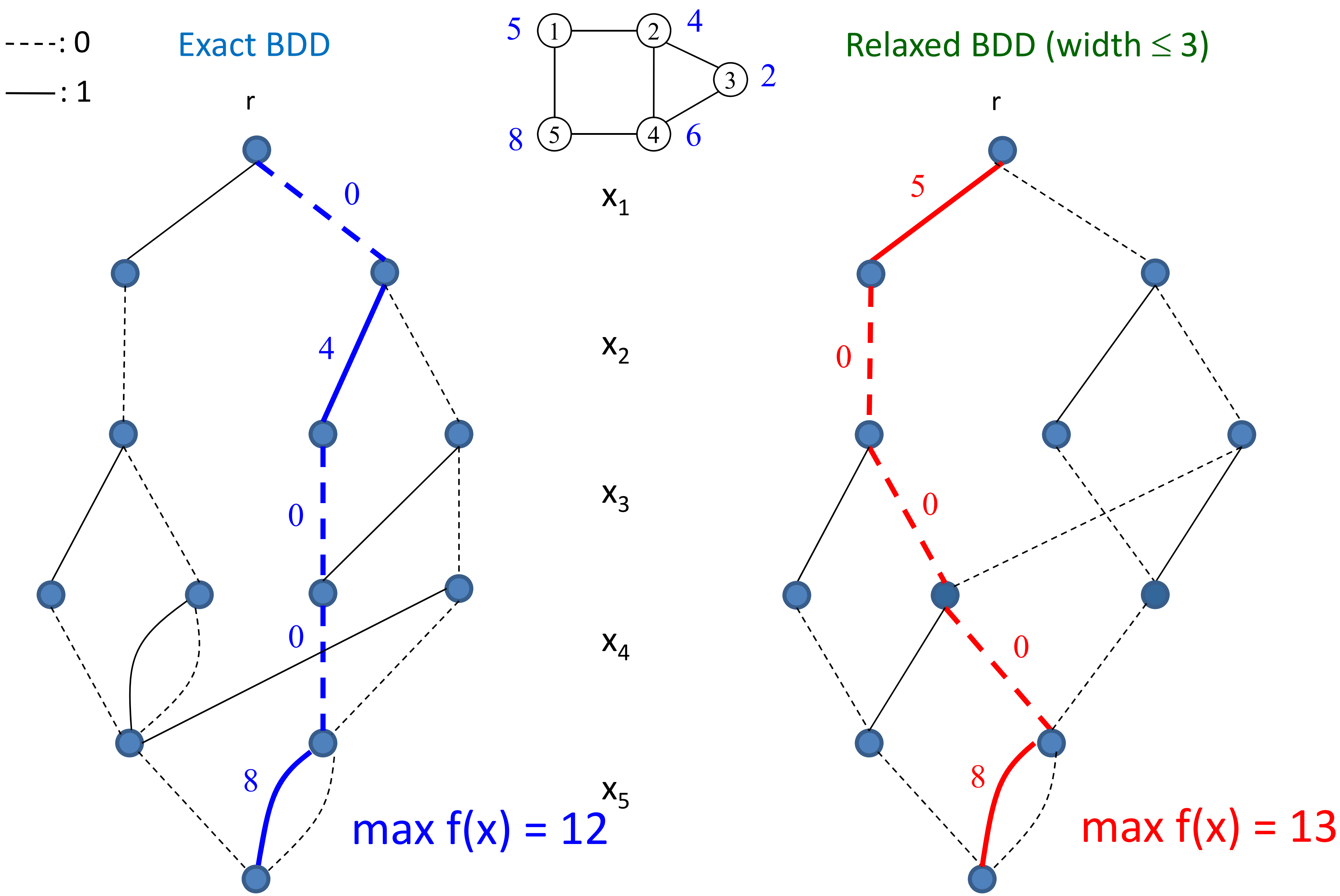


(1,0,0,0,1)

Evaluate Objective Function

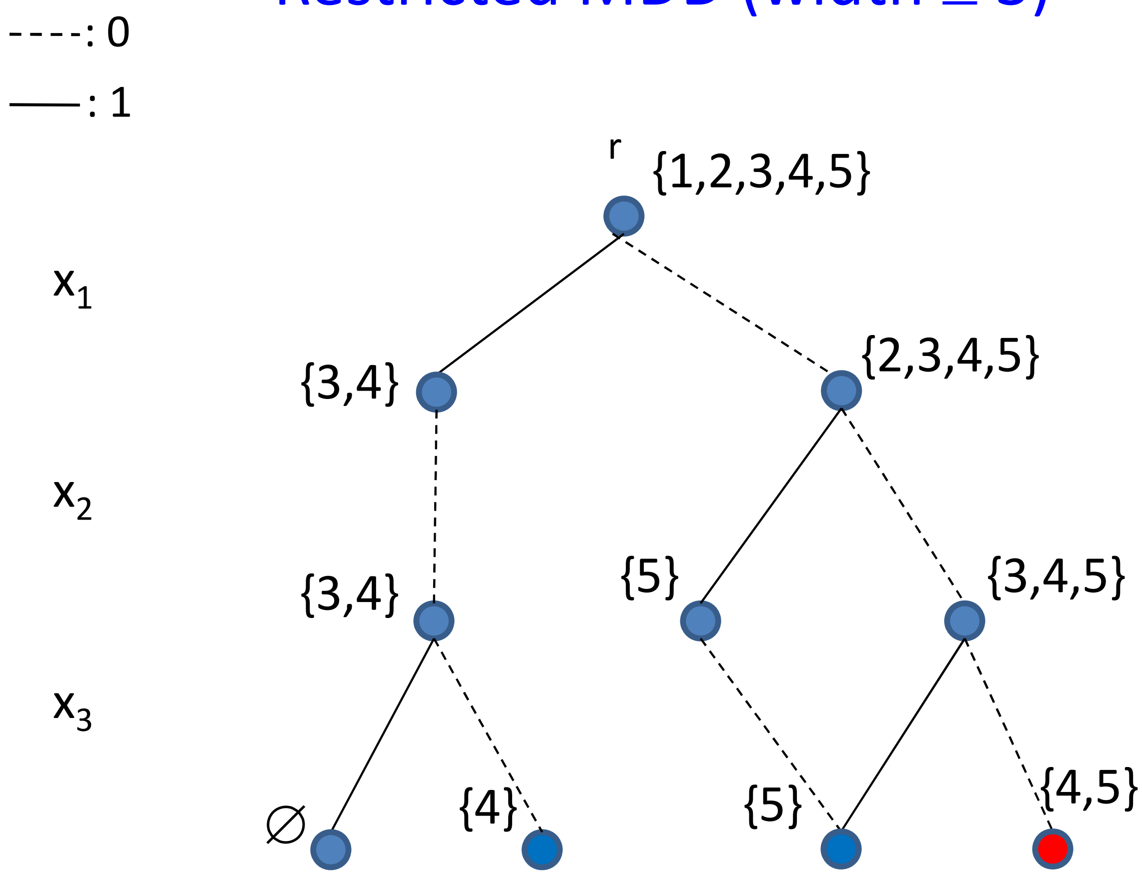


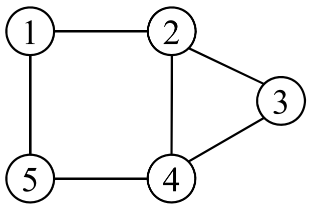
Evaluate Objective Function



Restricted BDD

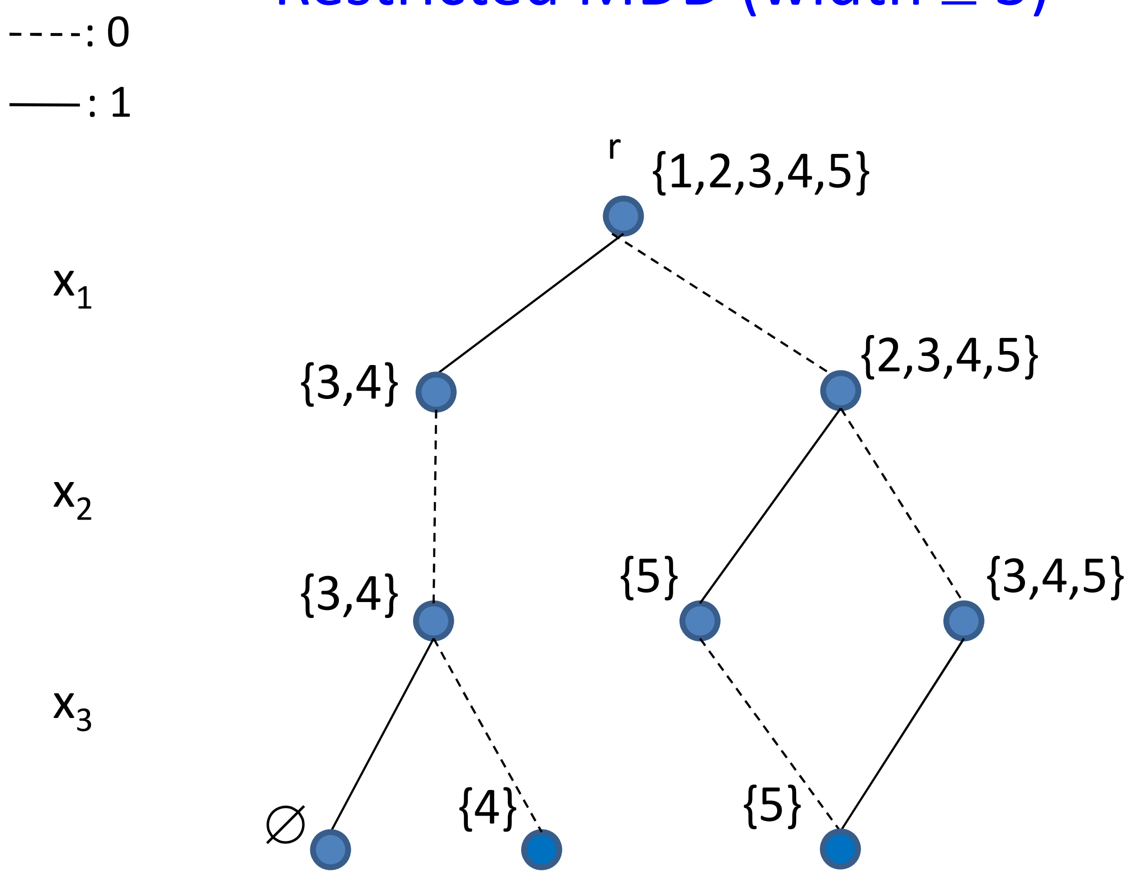
Restricted MDD (width ≤ 3)

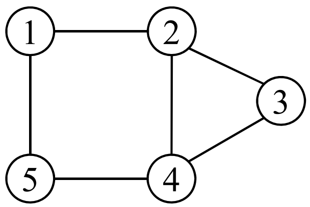




Restricted BDD

Restricted MDD (width $\leq$ 3)



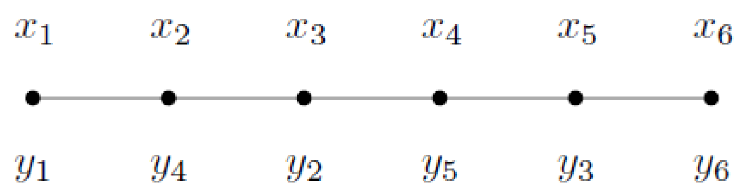


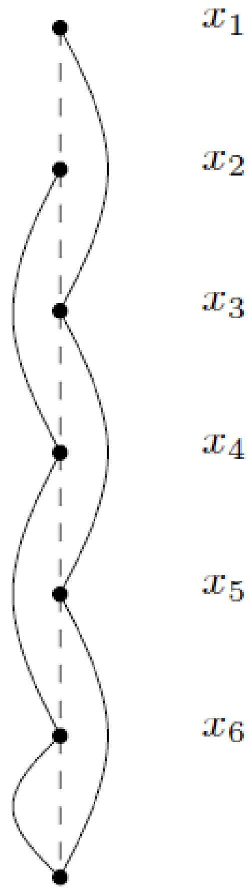
Variable Ordering

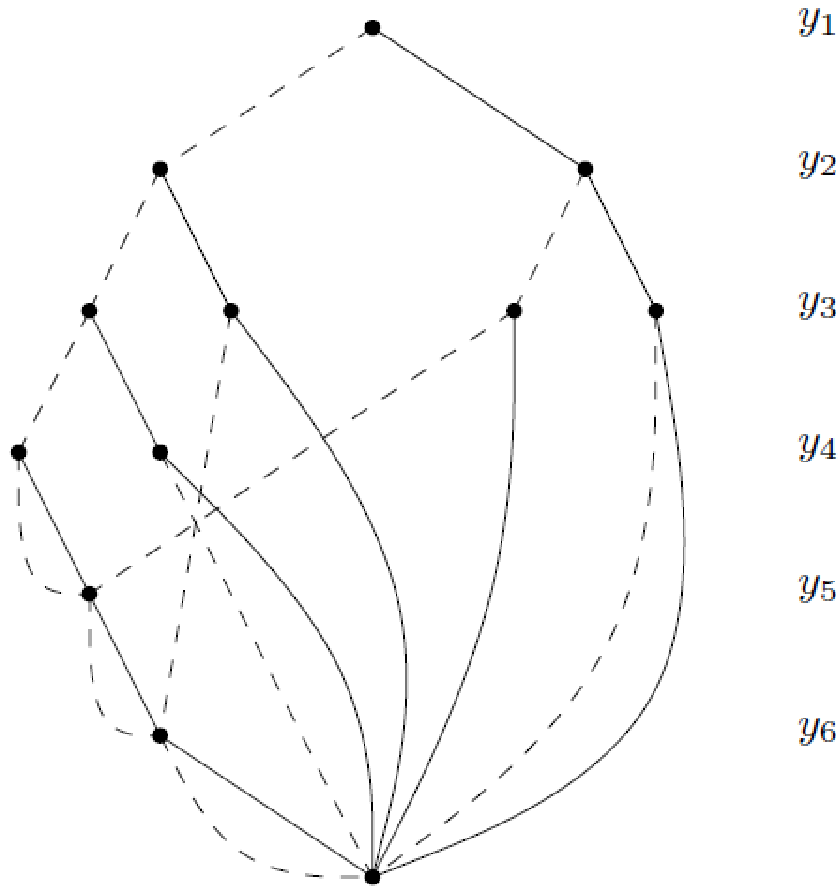
- Order of variables greatly impacts BDD size
  - also influences bound from relaxed BDD (see next)
- Finding 'optimal ordering' is NP-hard

- Insights from independent set as case study
  - formal bounds on BDD size

Exact BDD orderings for Paths







Formal Results for Independent Set

Graph Class	Bound on Width
Paths	1
Cliques	1
Interval Graphs	1
Trees	n/2
General Graphs	Fibonacci Numbers: $\|Layer\ j\| \leq F_{j+1}$

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

INFORMS J. Computing (2014)

Variable ordering heuristics

- Several possibilities
  - choose vertex at random
  - choose vertex that appears in fewest states in current layer
  - choose vertex according to maximal path decomposition

Variable ordering heuristics

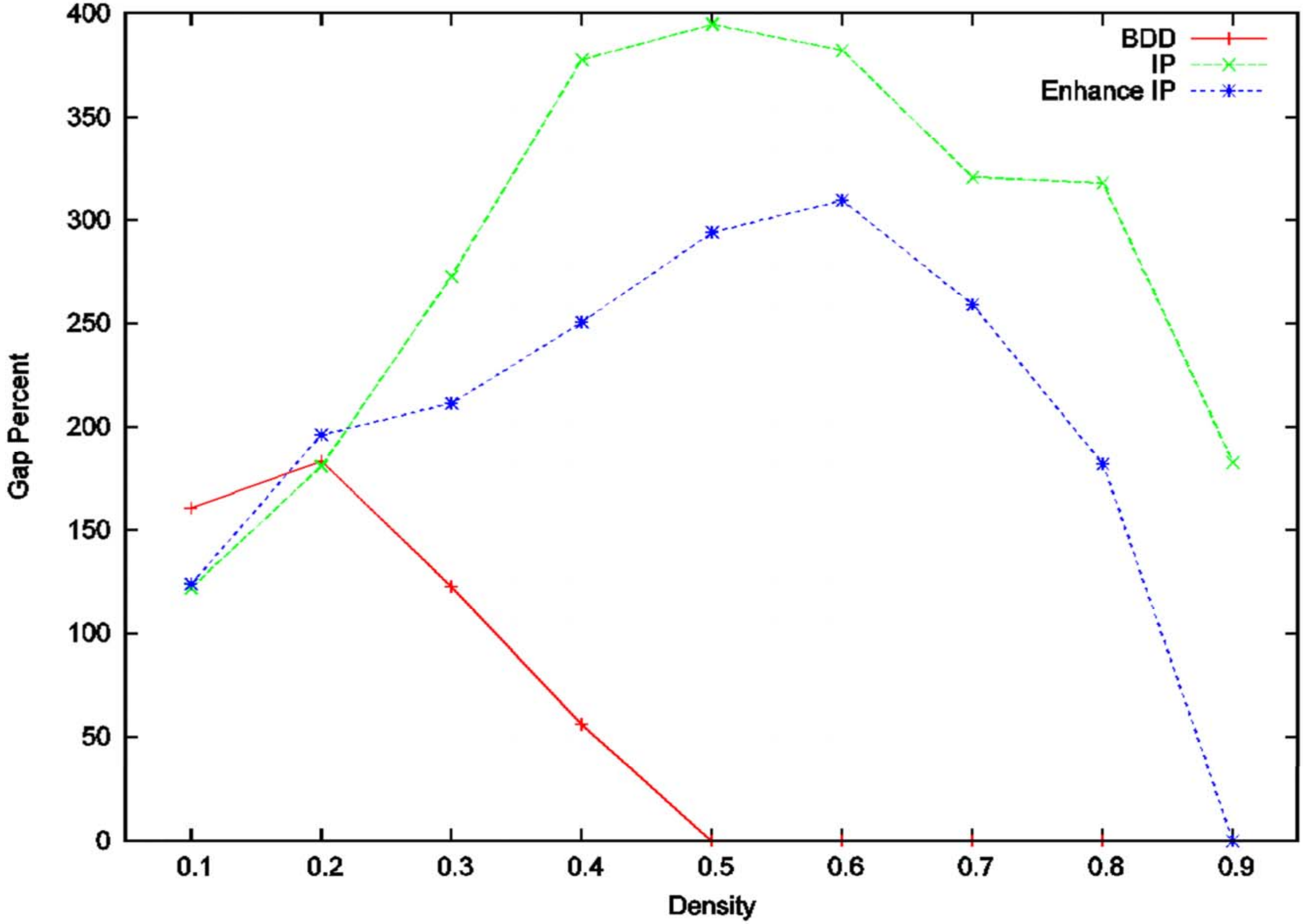
- Several possibilities
  - choose vertex at random
  - choose vertex that appears in fewest states in current layer
  - choose vertex according to maximal path decomposition

- Evaluate quality of the bounds in practice
  - Random Erdös-Rényi G(n,p) graphs
  - DIMACS clique graphs (87 instances)
  - Compare with CPLEX 12.5

    (standard MIP model and clique cover model)

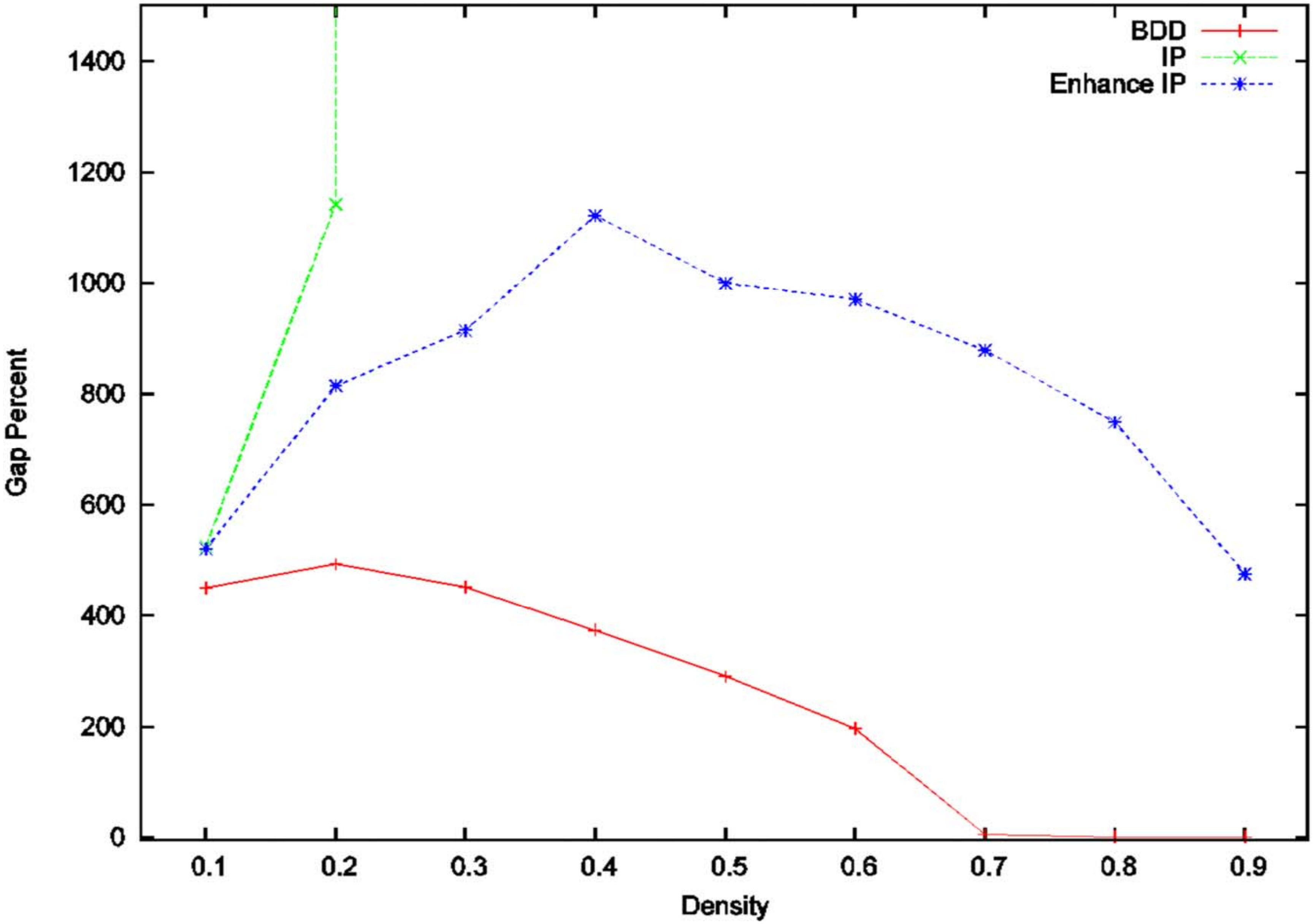
Bounds in practice

random graphs (n=500)



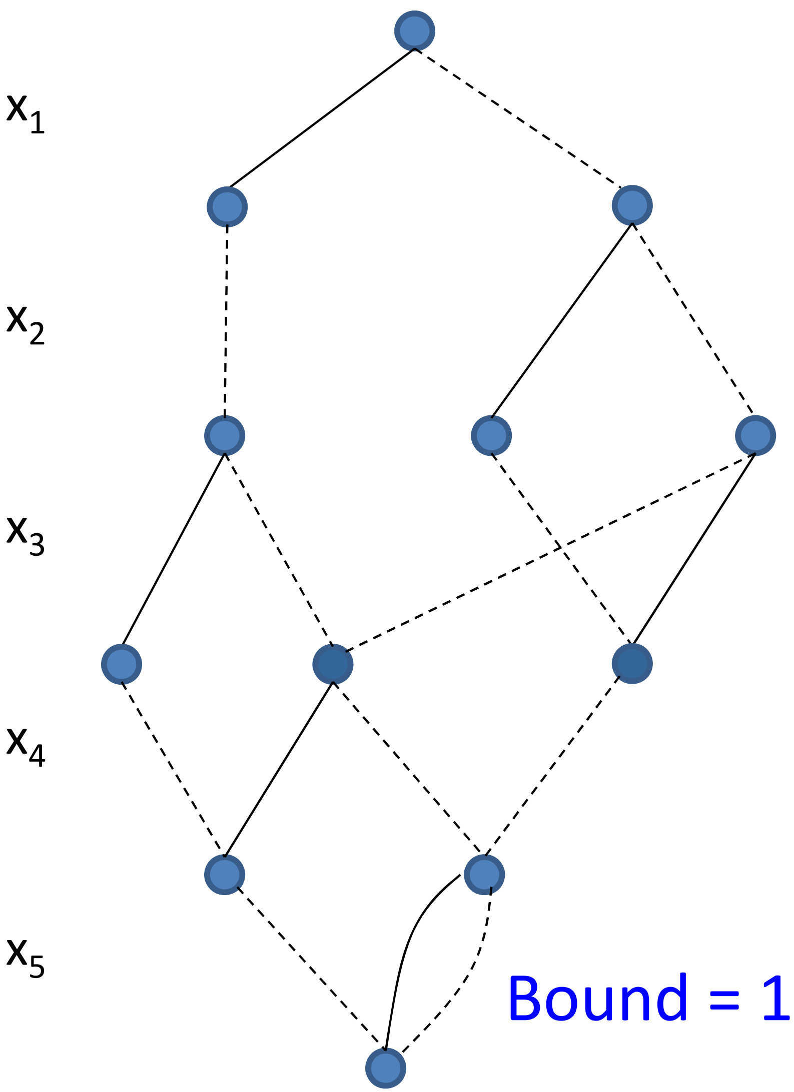
Bounds in practice

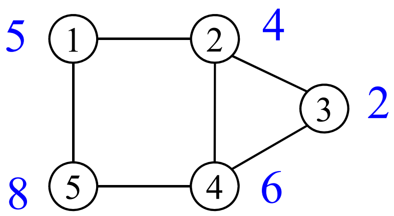
random graphs (n=1500)



Branch and Bound

Relaxed BDD (width ≤ 3)

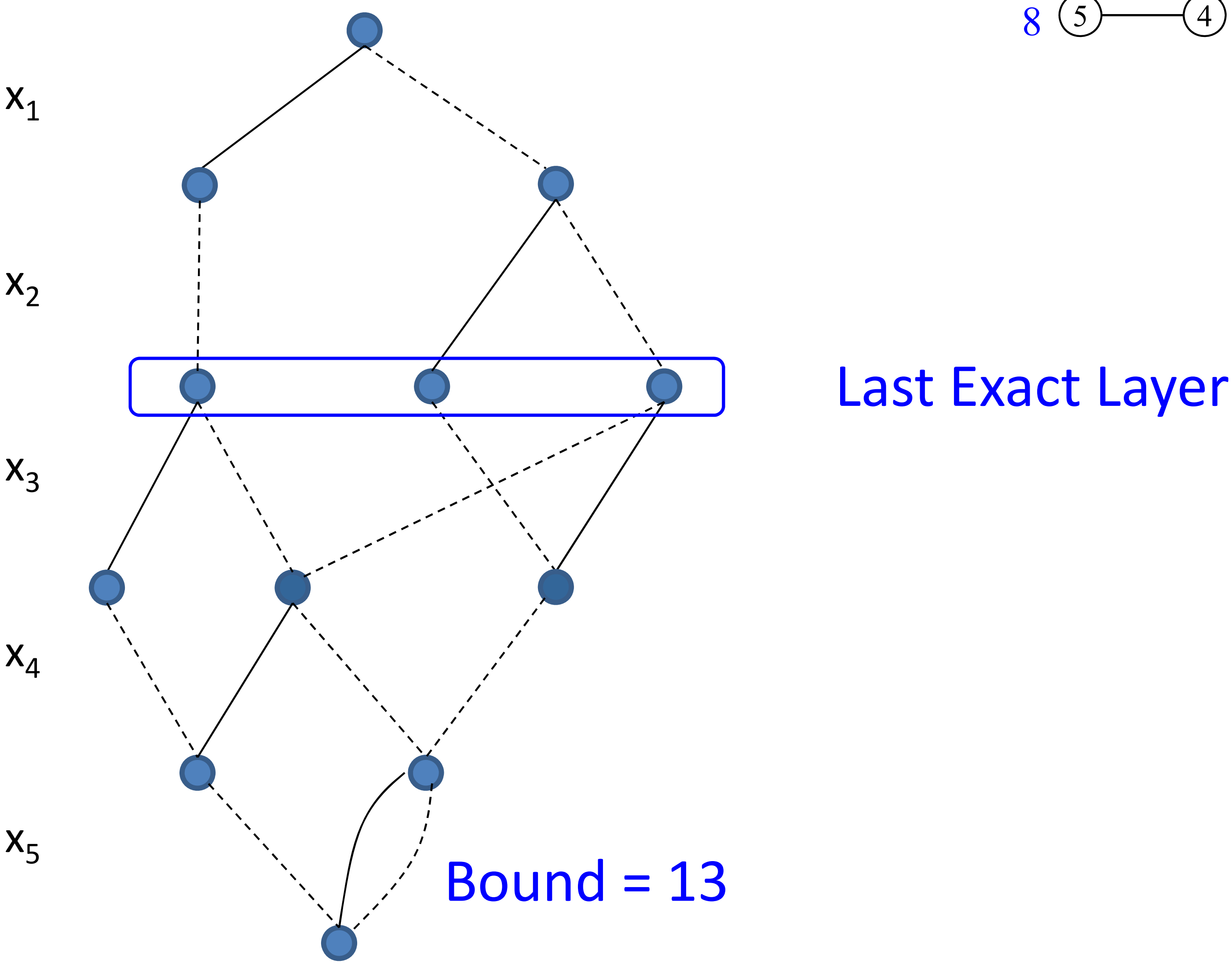


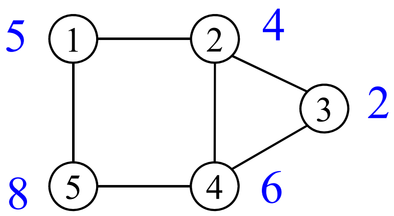


Bound = 13

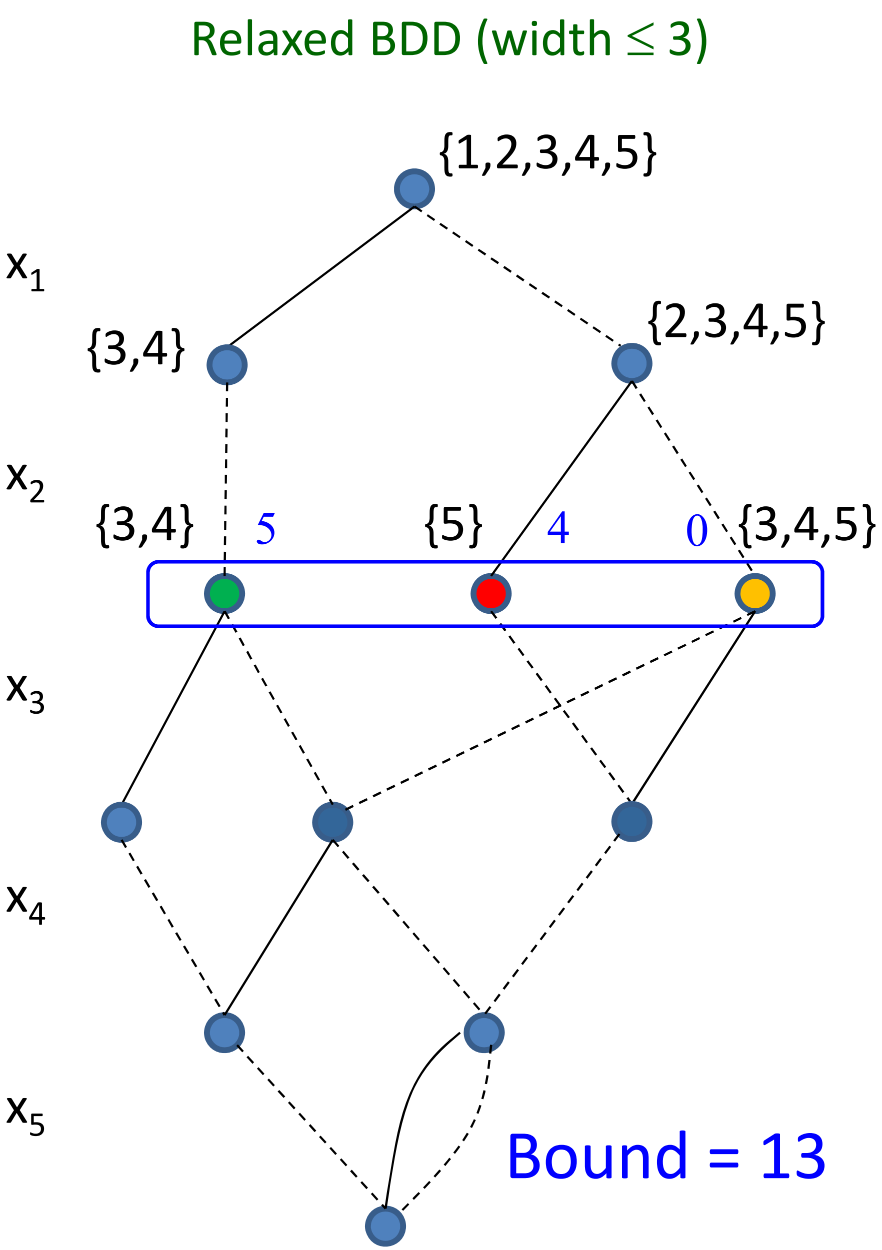
Branch and Bound

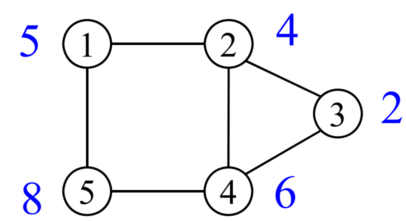
Relaxed BDD (width ≤ 3)





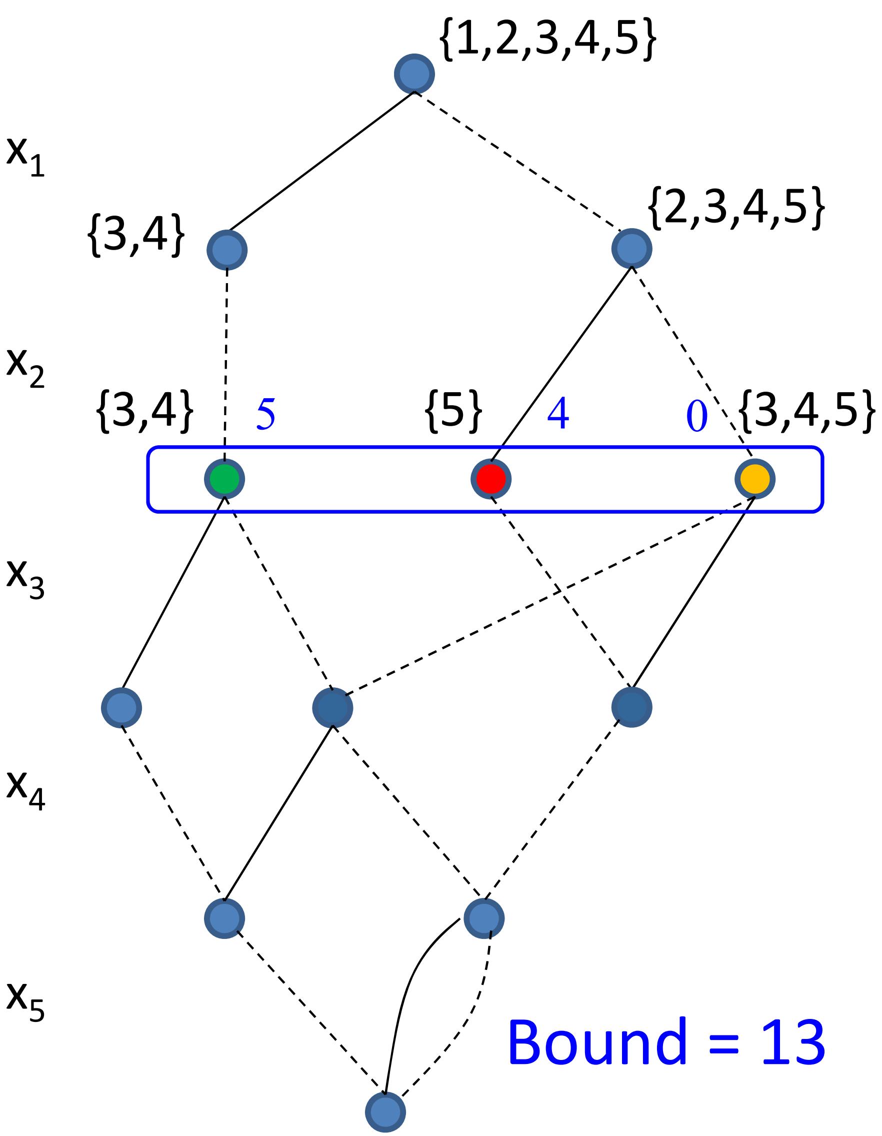
Branch and Bound

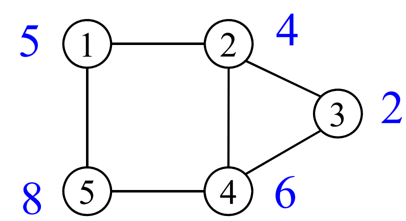




Branch and Bound

Relaxed BDD (width ≤ 3)





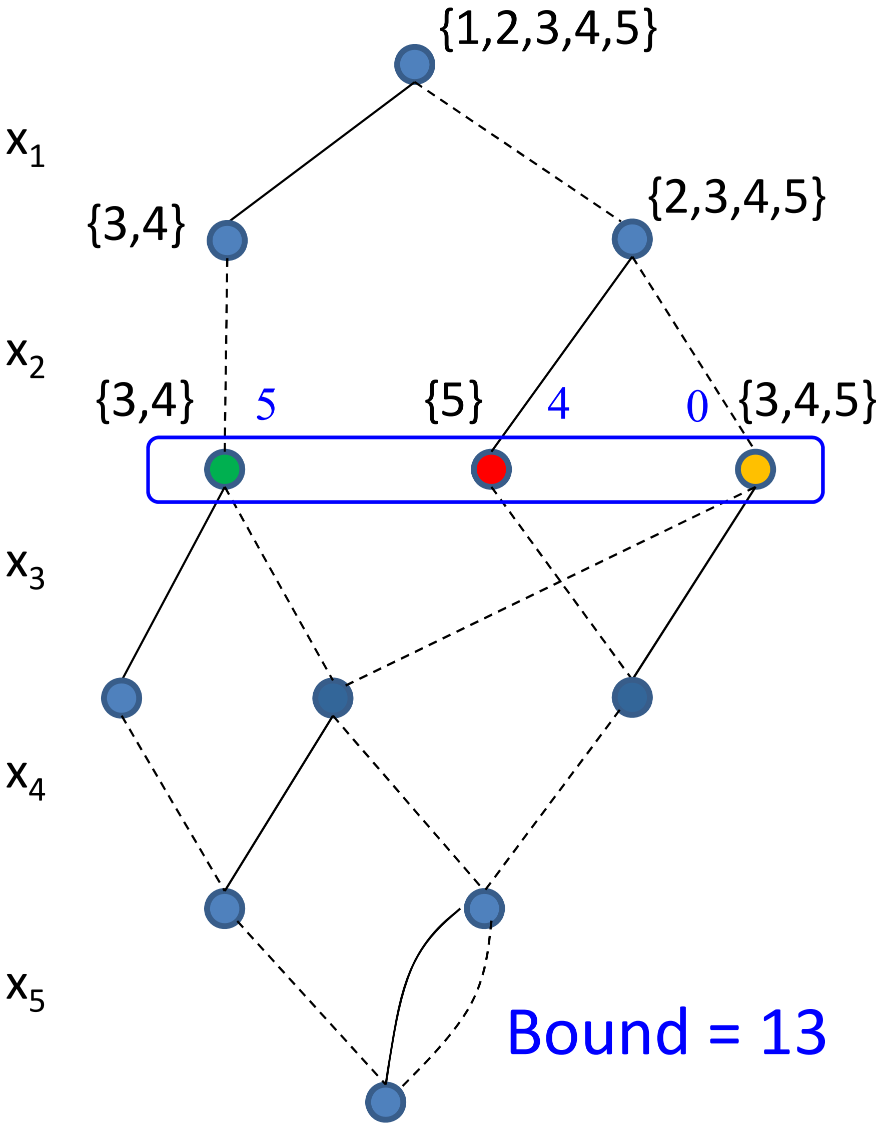
Last Exact Layer

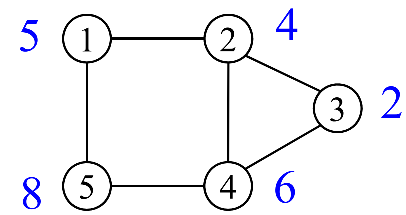
{3,4} : 5+6 = 11

Bound = 13

Branch and Bound

Relaxed BDD (width ≤ 3)



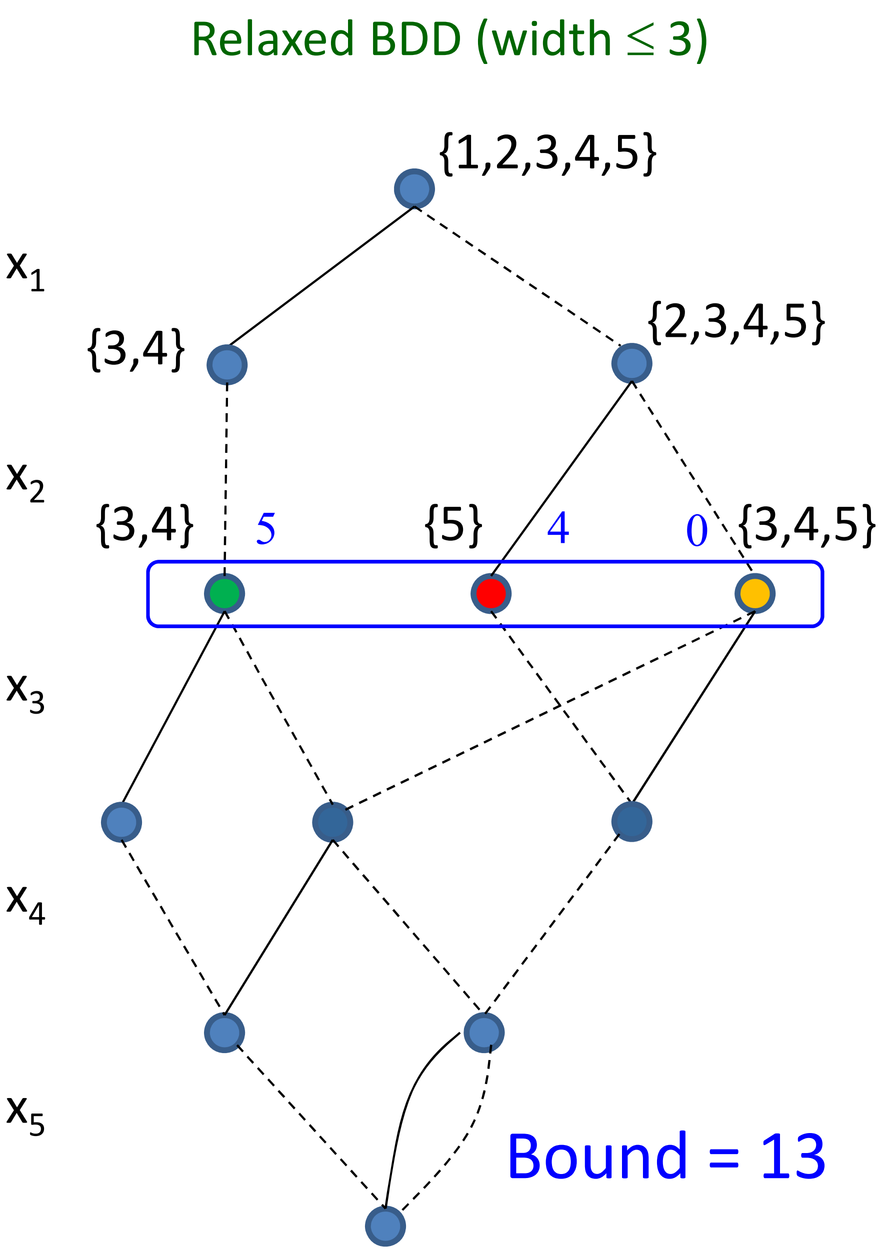


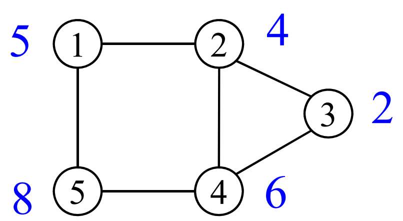
Last Exact Layer

{3,4} : 5+6 = 11

{5} : 4+8 = 12

Branch and Bound

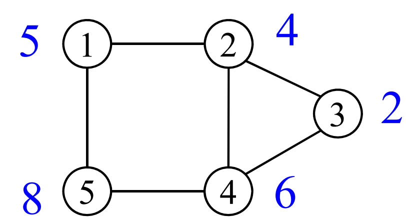


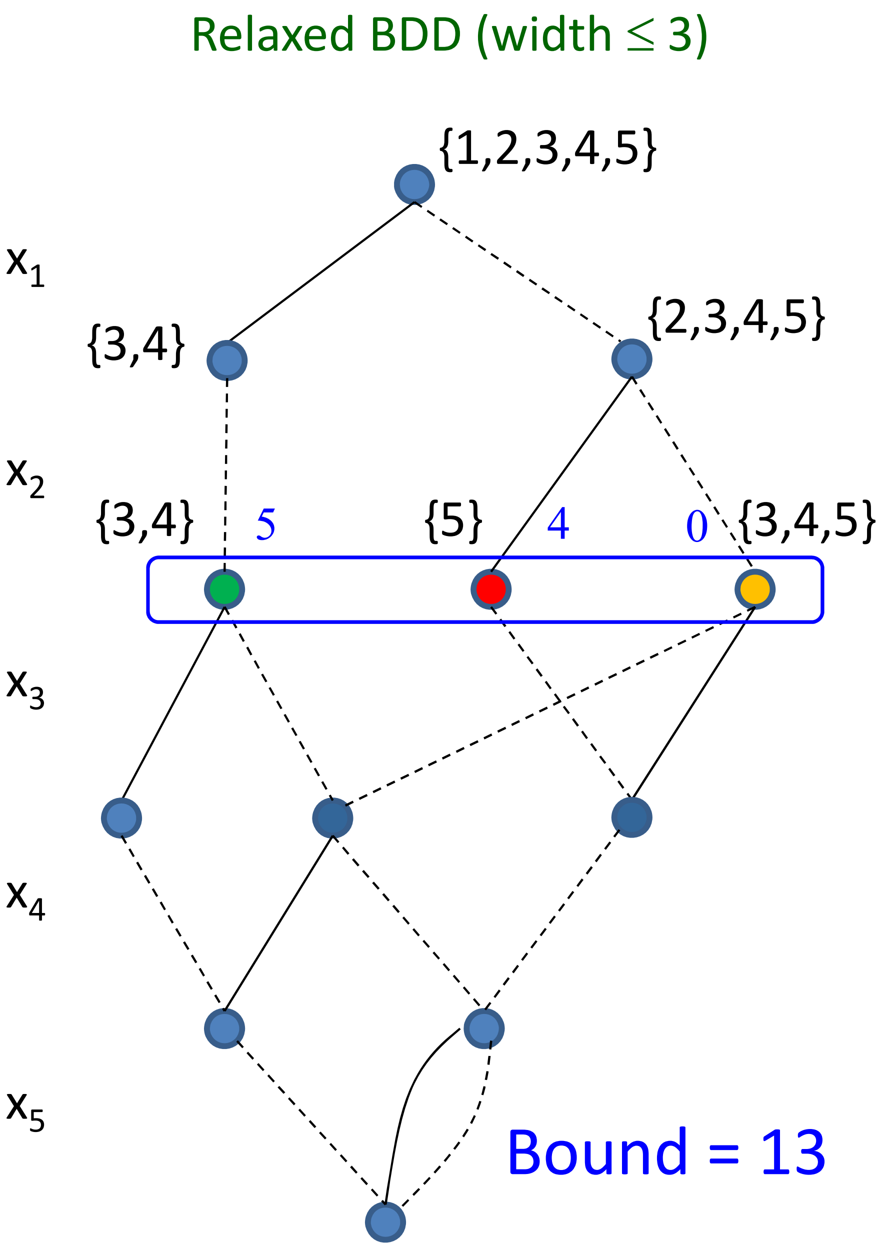


Last Exact Layer

- {3,4} : 5+6 = 11
- {5} : 4+8 = 12
- {3,4,5} : 0+10 = 10

Branch and Bound





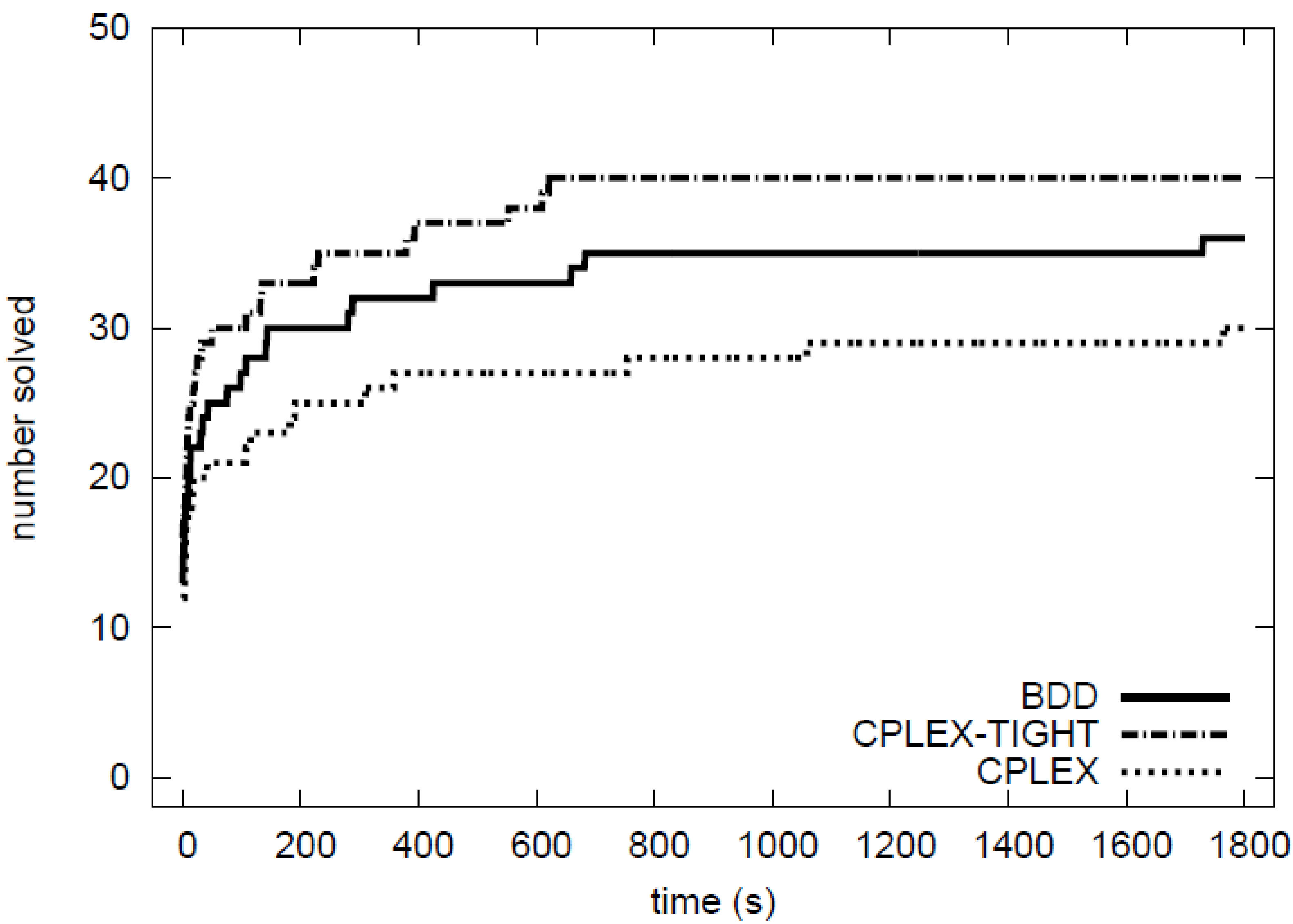
Last Exact Layer

- {3,4} : 5+6 = 11
- {5} : 4+8 = 12 ← maximum
- {3,4,5} : 0+10 = 10

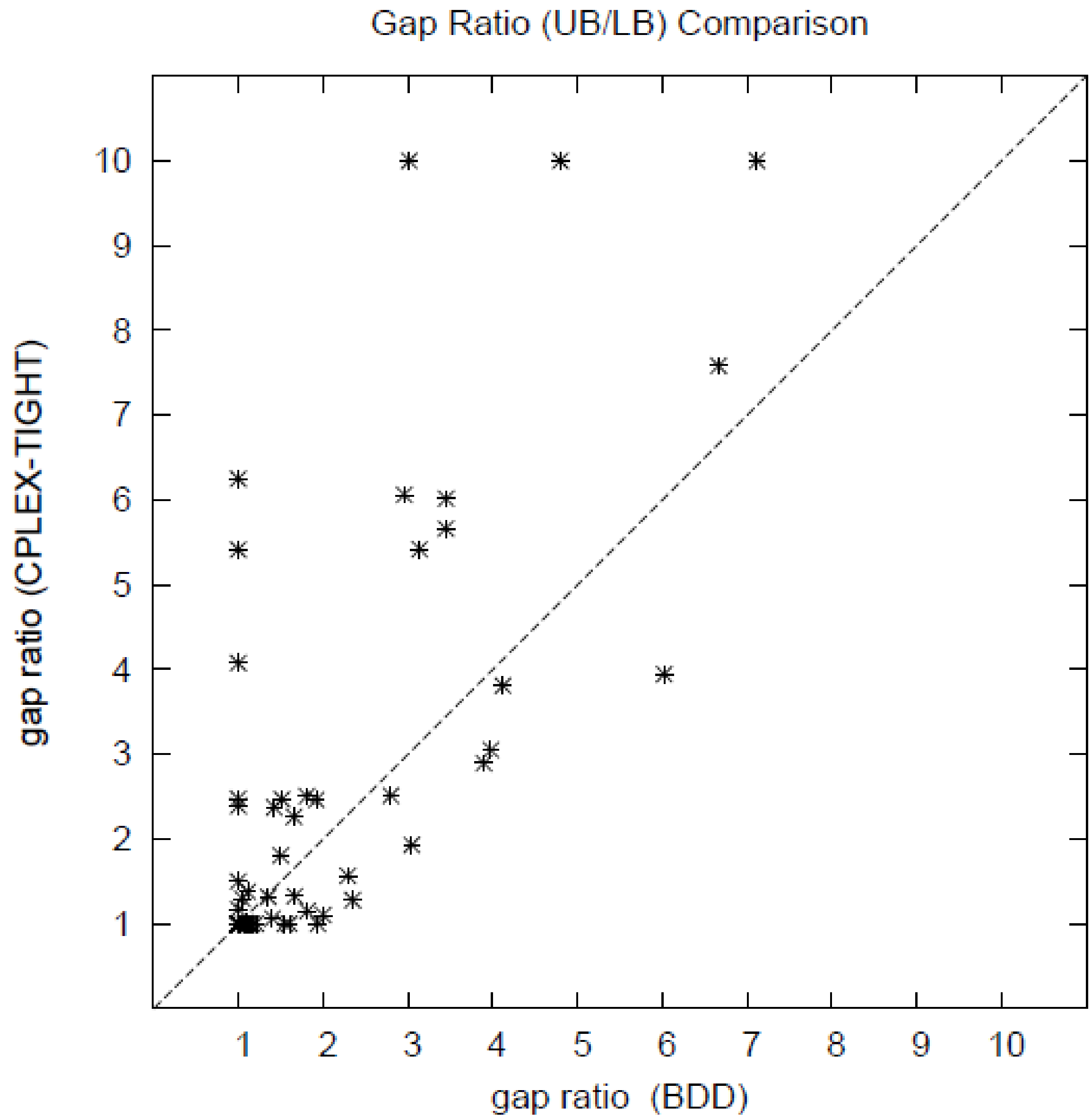
New Branching Scheme

- 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: DIMACS



DIMACS Graphs: End Gap (1,800s)



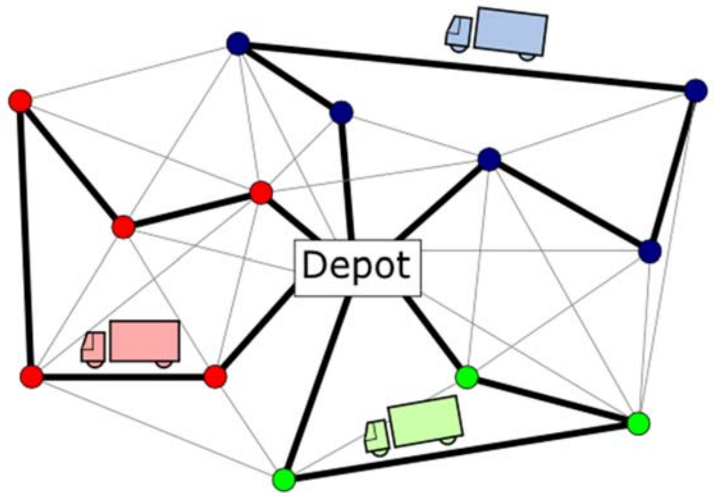
General Approach

- 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, …

INFORMS J. Computing (to appear)

J. Heuristics (2014)



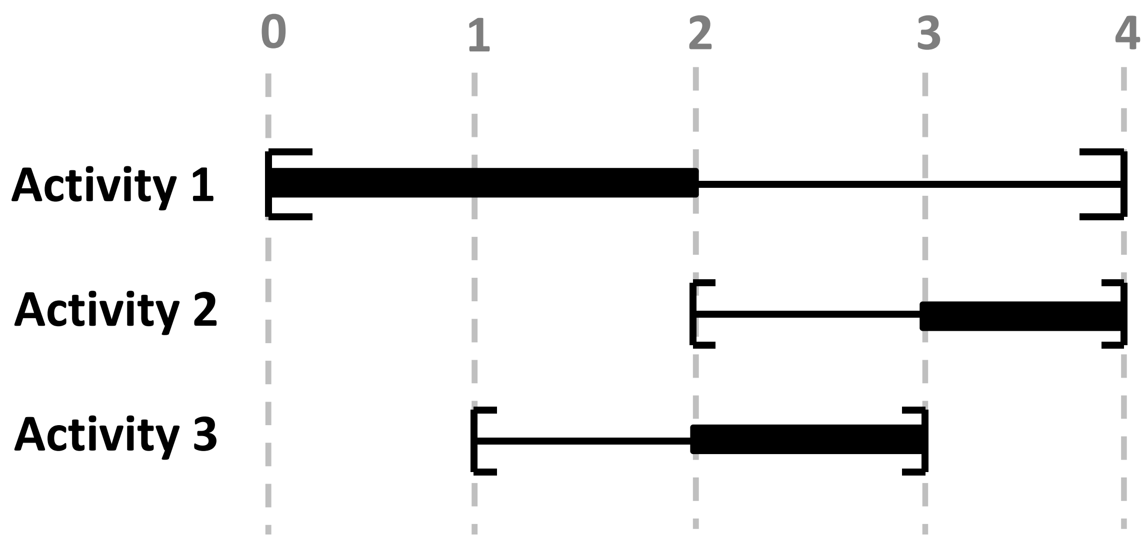
MDDs for Constraint-Based Scheduling

Cire, v.H.: MDDs for Sequencing Problems.

Operations Research, 61(6): 1411-1428, 2013.

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

Extensions

- Precedence relations between activities
- Sequence-dependent setup times
- Various objective functions
  - Makespan
  - Sum of setup times
  - (Weighted) sum of completion times
  - (Weighted) tardiness
  - number of late jobs
  - …

MDD Representation

- Natural representation as ‘permutation MDD’

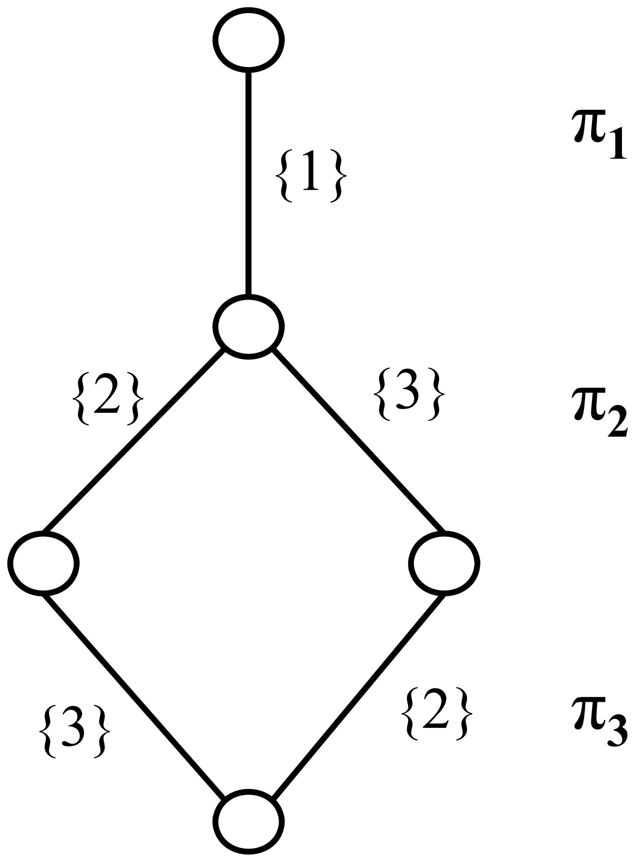
- Every solution can be written as a permutation $\boldsymbol{\pi}$

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

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

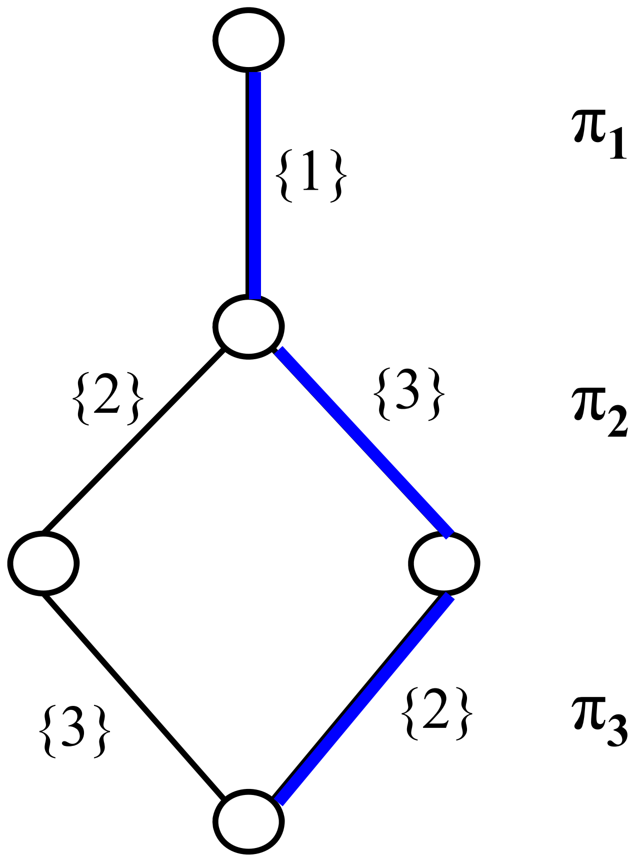
$$start_{\pi_i} \geq start_{\pi_{i-1}} + p_{\pi_{i-1}} \qquad i = 2, \ldots, n$$

MDD Representation



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

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} :

$0 \leq \text{start}_1 \leq 1$

$6 \leq \text{start}_2 \leq 7$

$3 \leq \text{start}_3 \leq 5$

MDD-based propagation

Carnegie Mellon Tepper School of Business

Propagation: remove infeasible arcs from the MDD

We can utilize several structures/constraints:

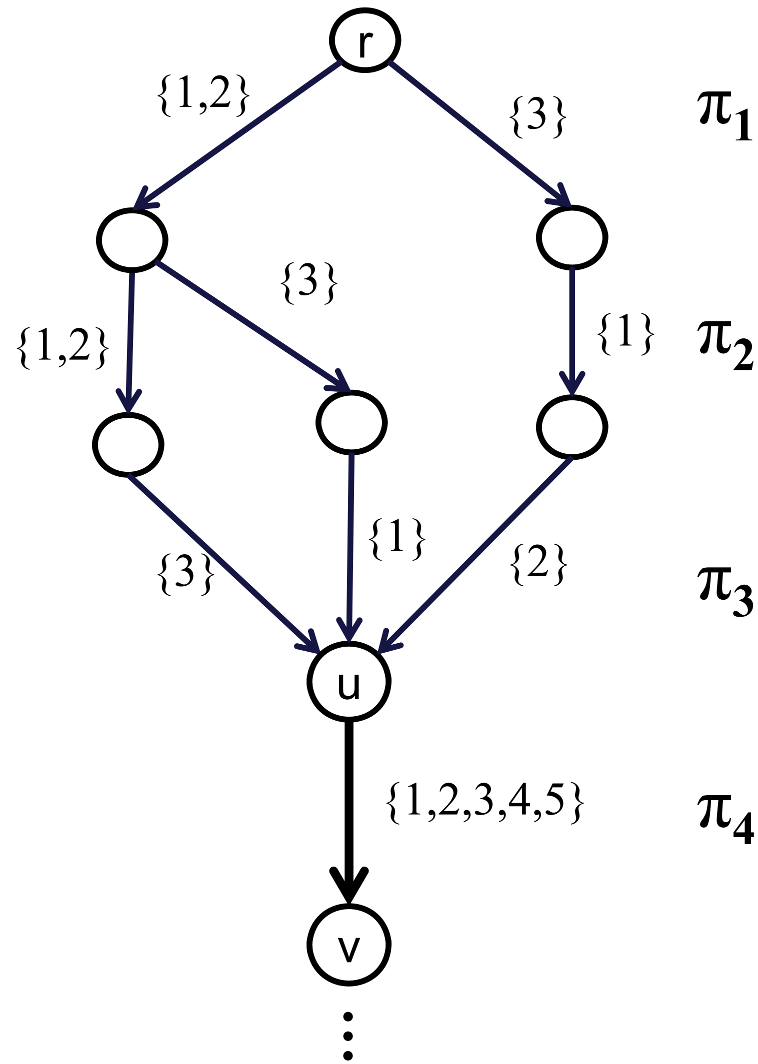
- *Alldifferent* for the permutation structure
- Earliest start time and latest end time
- Precedence relations

For a given constraint type we maintain specific ‘state information’ at each node in the MDD

- both top-down and bottom-up

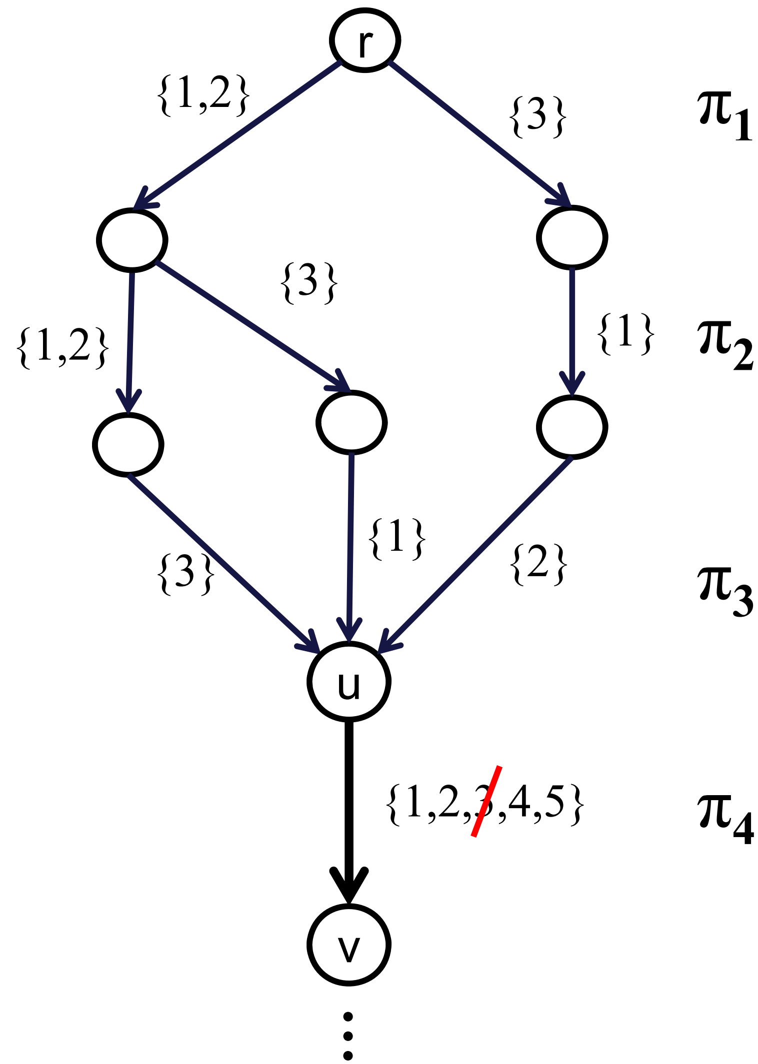
Propagation (cont'd)

- 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)



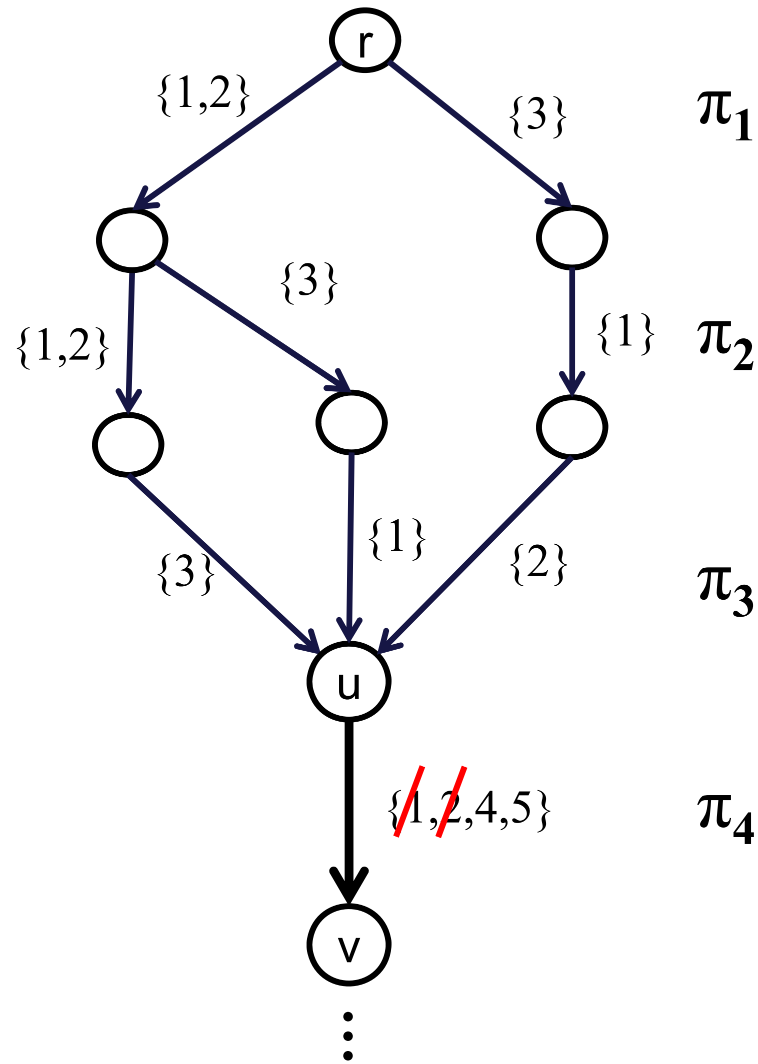
Alldifferent Propagation

- 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

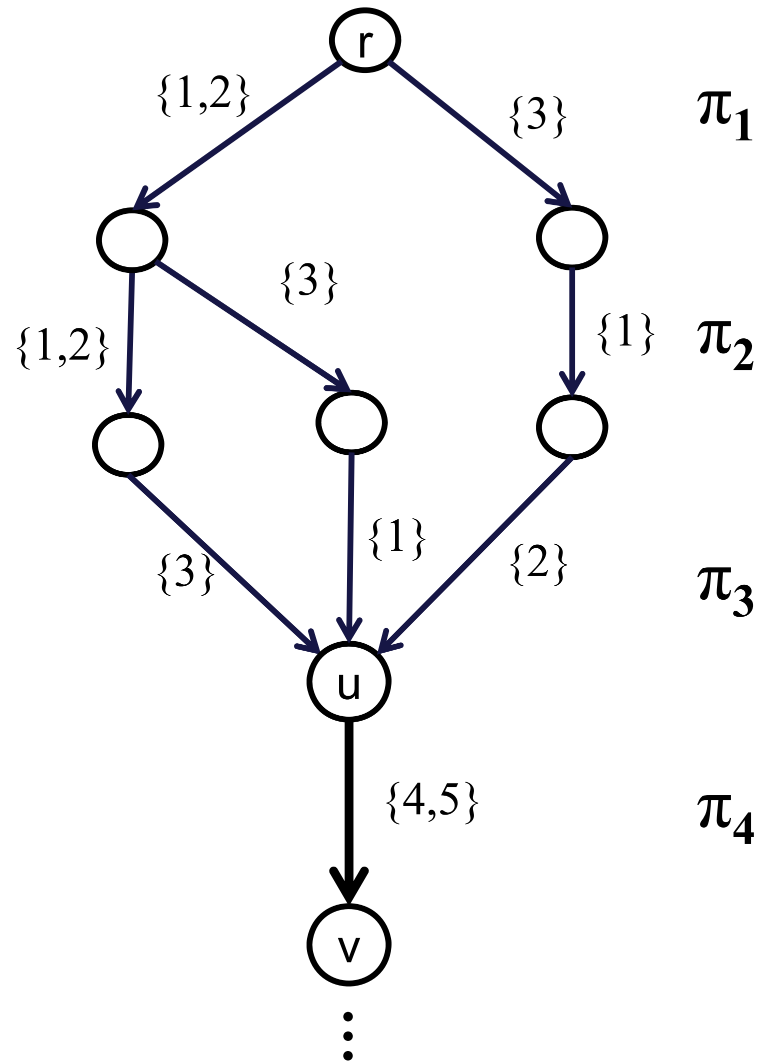
- Some-paths state: S_u
  - Labels belonging to some path from node r to node u
  - $S_u = \{1,2,3\}$
  - Identification of Hall sets
  - Thus eliminate {1,2,3} from (u,v)



Propagate Earliest Completion Time

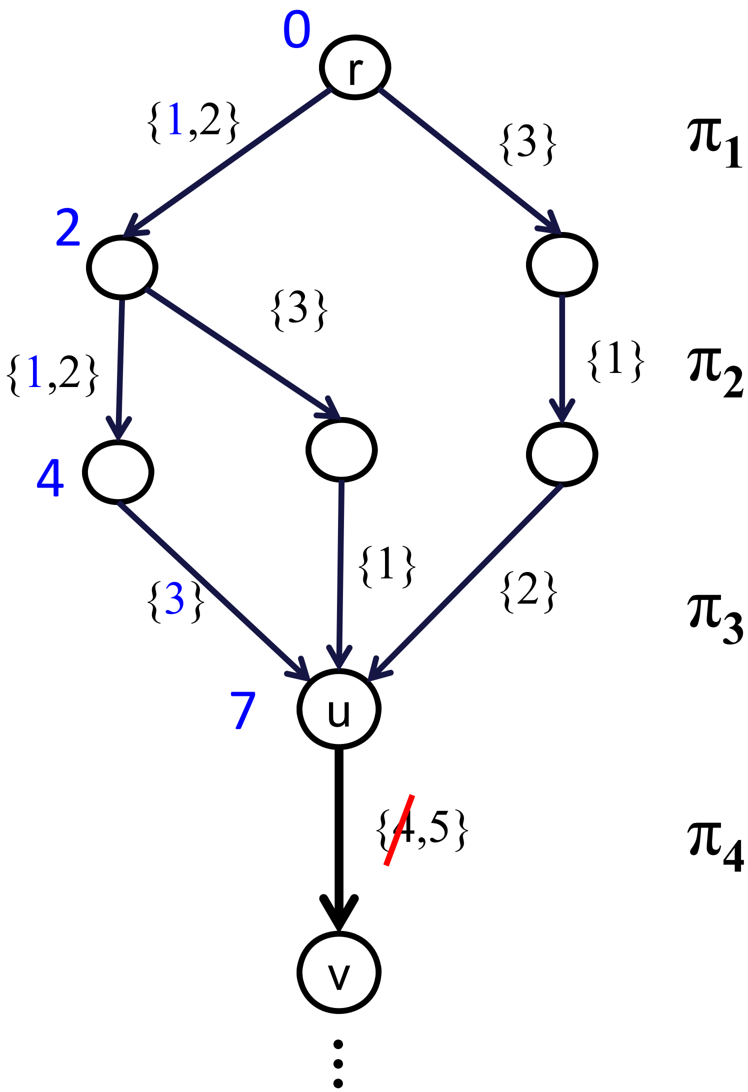
- Earliest Completion Time: E_u
  - Minimum completion time of all paths from root to node u

- 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)

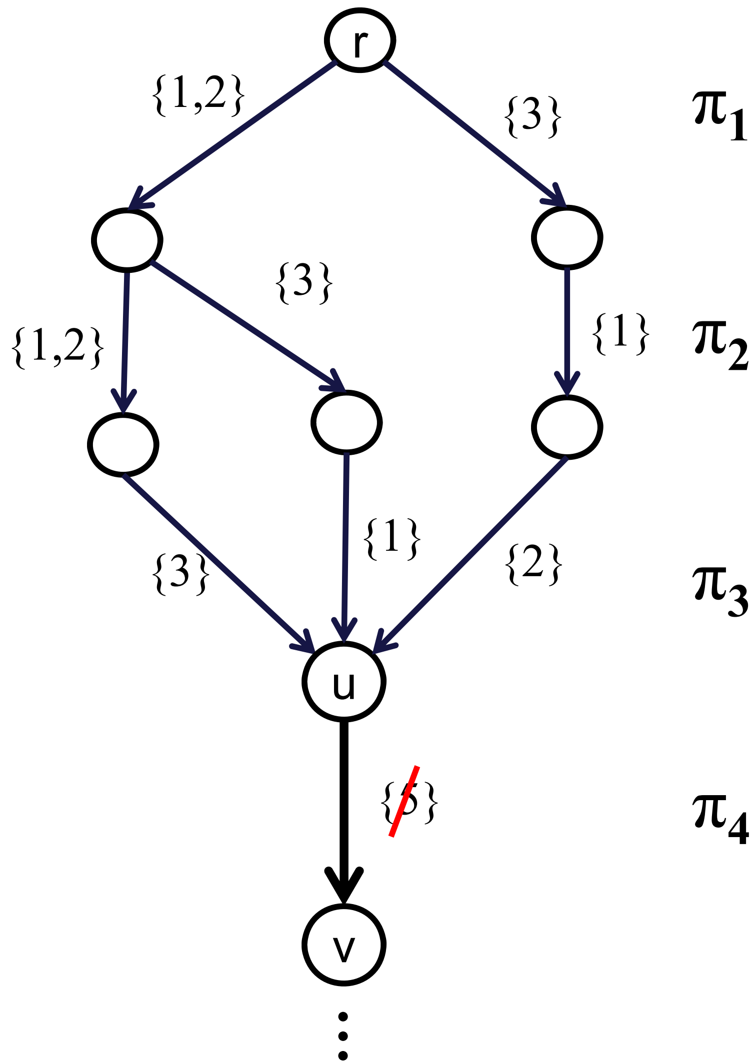
Propagate Precedence Relations

- 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)

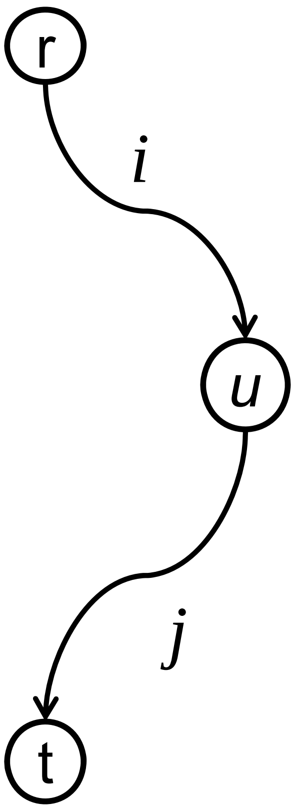
- Similarly: Bottom-up for $j \ll i$



More MDD Inference

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

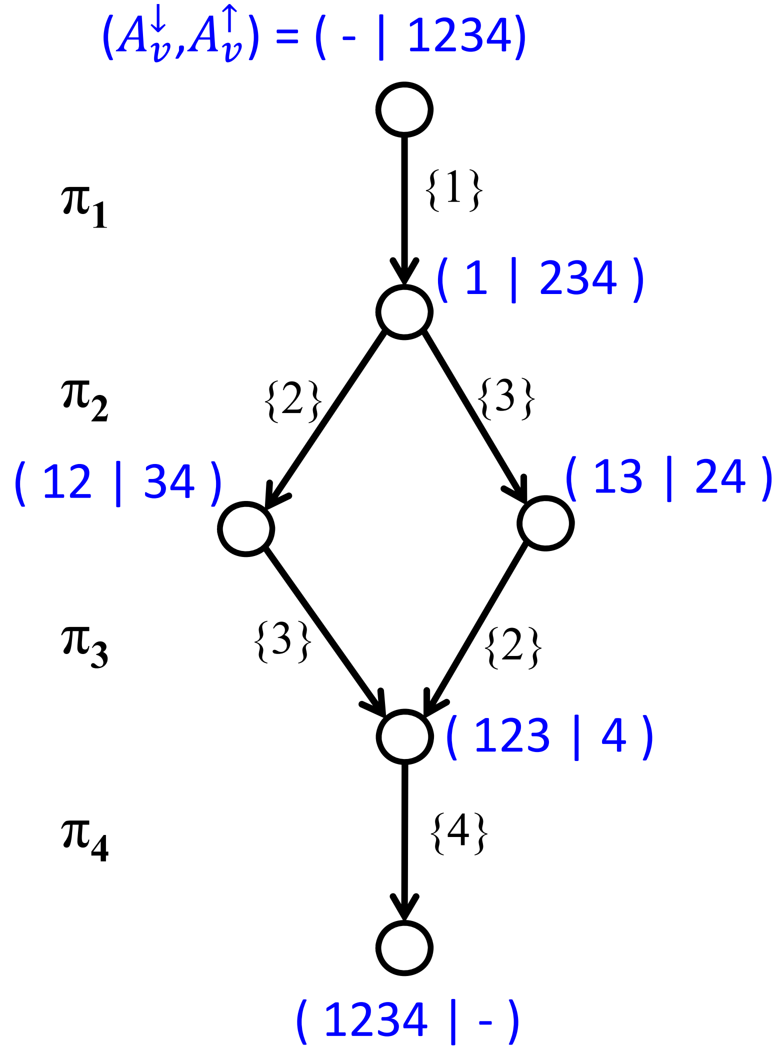
- For a node *u*,
  - $A_u^{\downarrow}$: values in all paths from root to *u*
  - $A_u^{\uparrow}$: values in all paths from node u to terminal

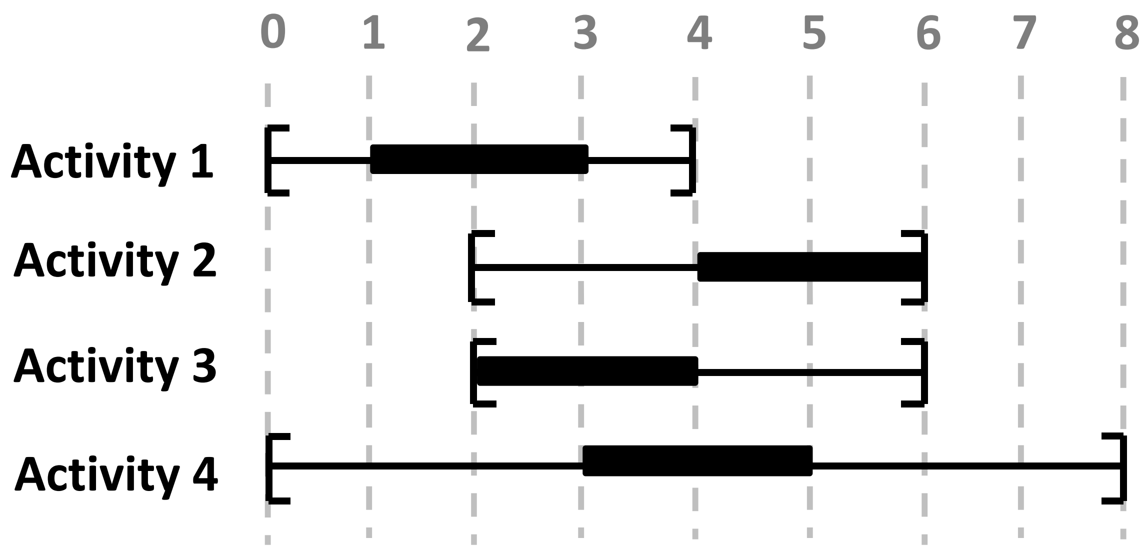


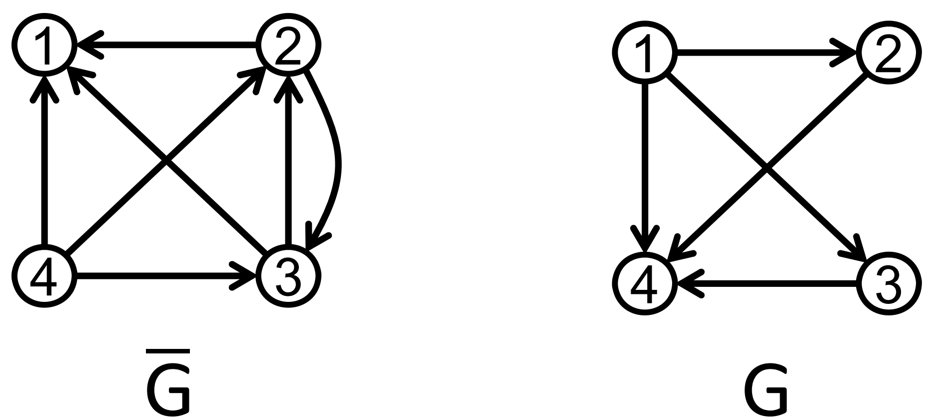
- *Precedence relation i ≪ j holds if and only if* $(j \notin A_u^{\downarrow})$ or $(i \notin A_u^{\uparrow})$ *for all nodes u in M*

relaxed MDD: use $S_u^{\downarrow}$ and $S_u^{\uparrow}$

Precedence relations: example







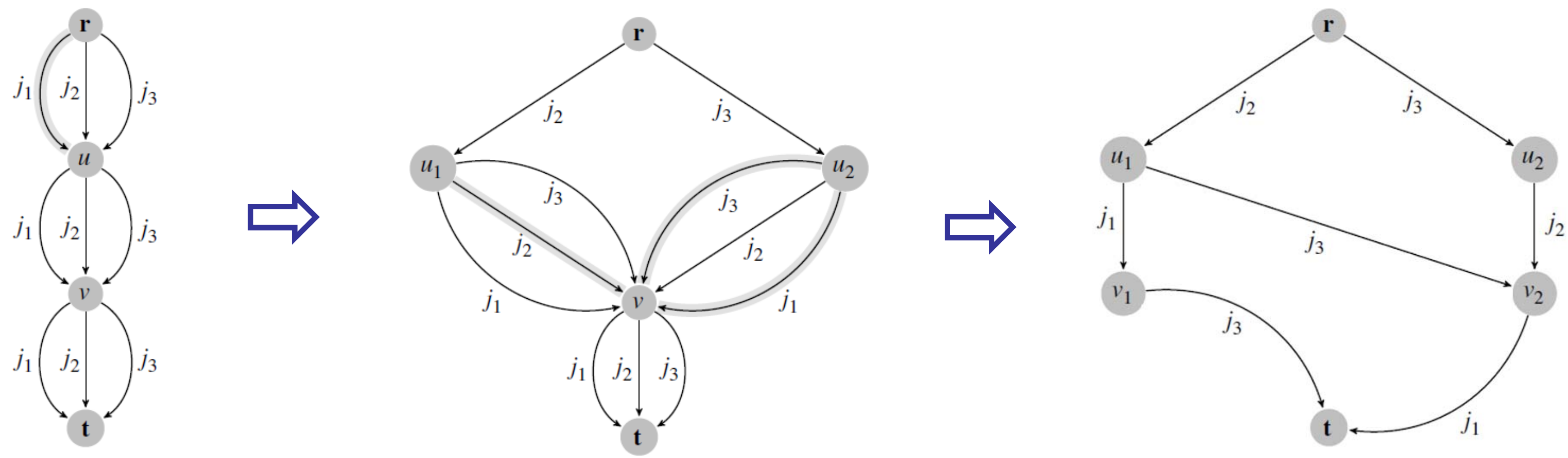
Arc (i,j) in $\overline{G}$ if $j \in A_u^\downarrow$ and $i \in A_u^\uparrow$ for *some* node *u* in *M*

$O(n^2|M|)$ time

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

MDD Construction and Refinement



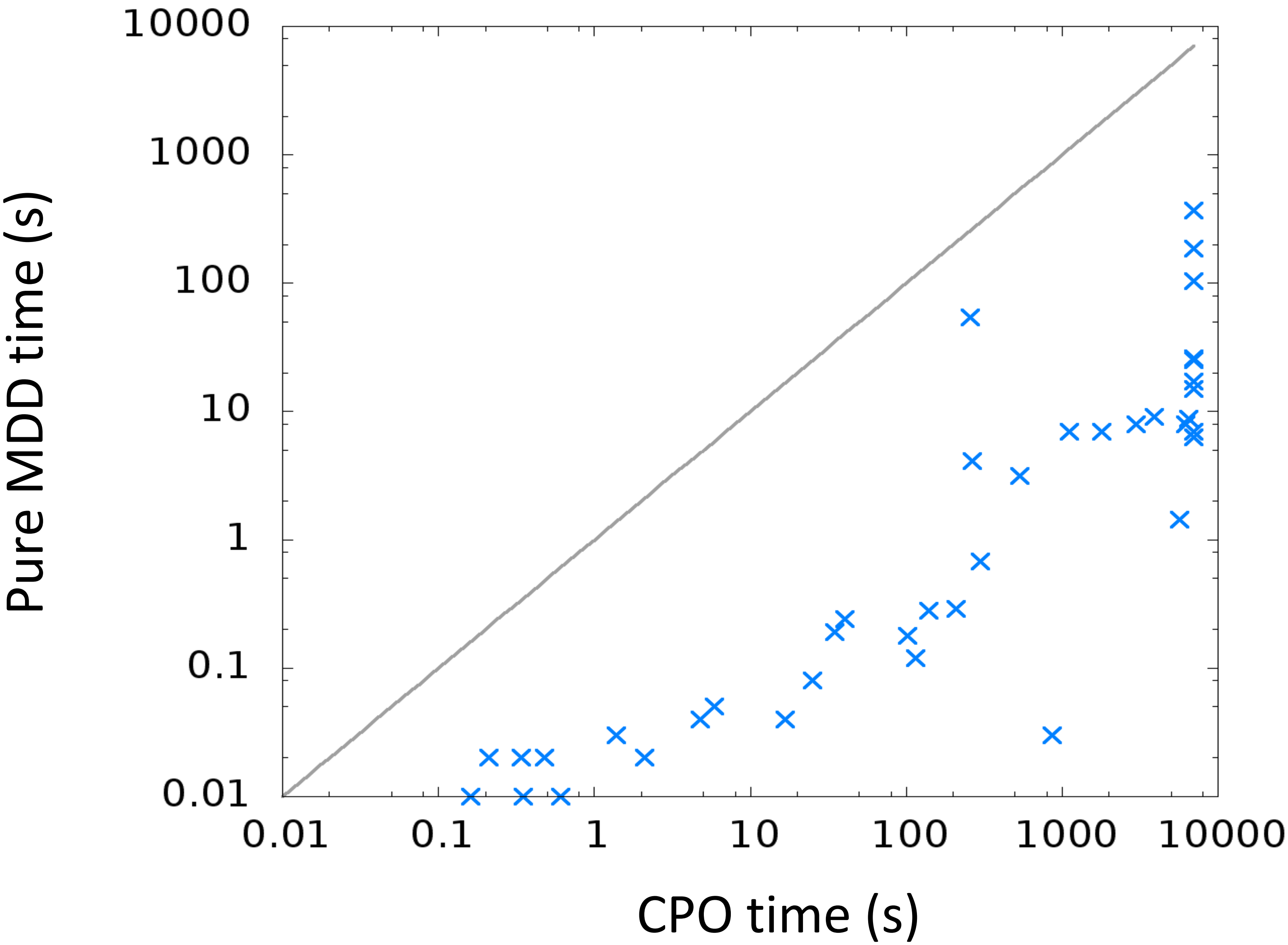
$j_2 \ll j_1$

- To refine the MDD, we generally want to identify equivalence classes among nodes in a layer
  - For sequencing, deciding equivalence is NP-hard
- In practice, refinement can be based on
  - earliest starting time
  - latest earliest completion time $r_i + p_i$
  - *alldifferent* constraint (A_i and S_i states)

Experiments

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

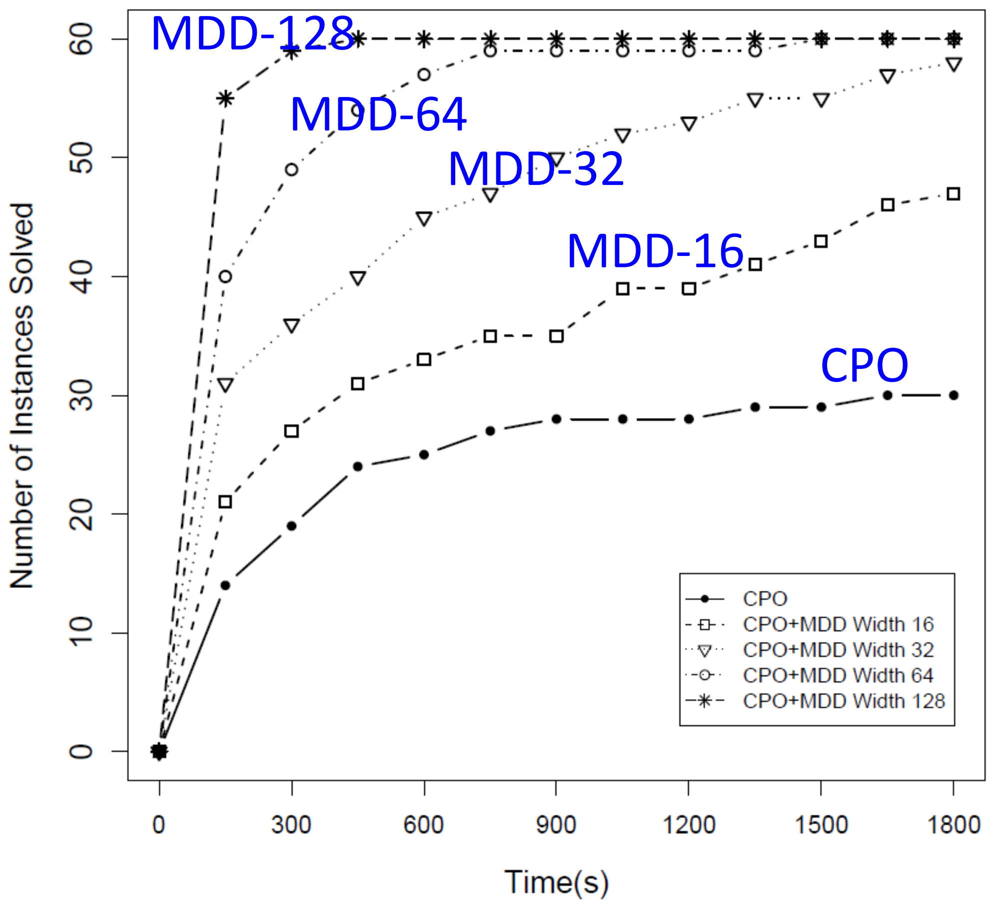
TSP with Time Windows



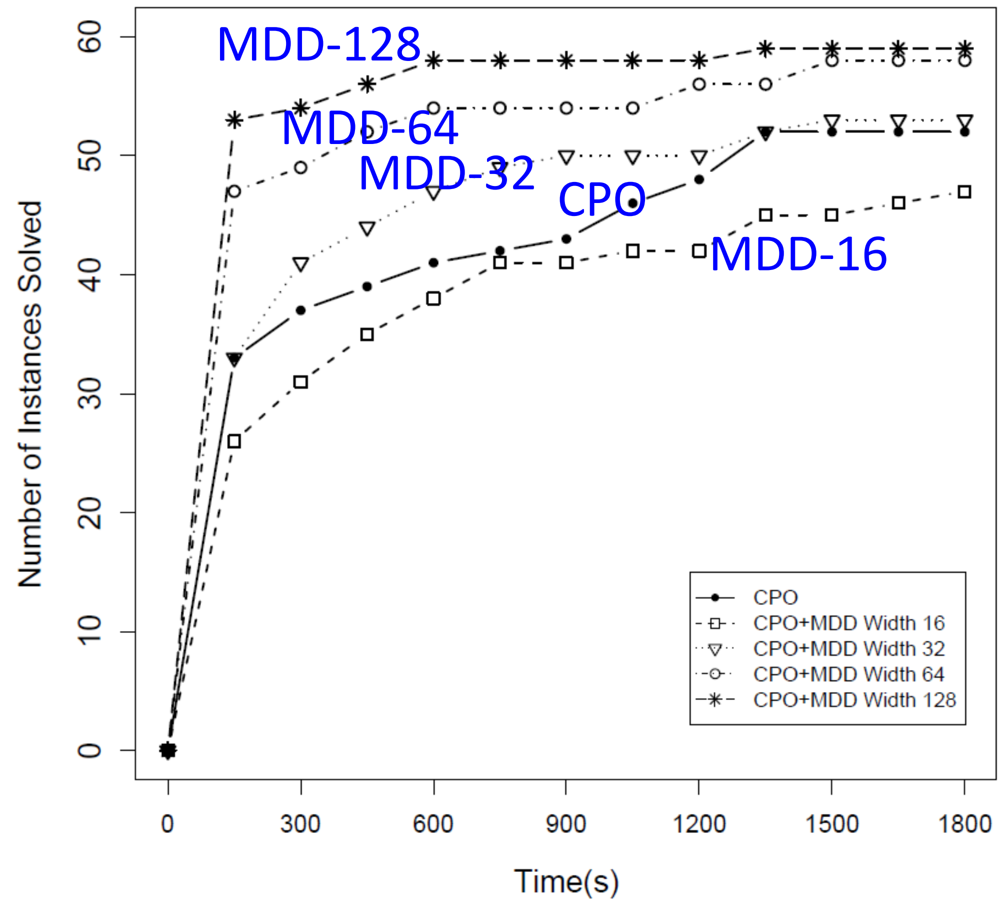
Dumas/Ascheuer instances

- 20-60 jobs
- lex search
- MDD width: 16

Total Tardiness Results



total tardiness



total weighted tardiness

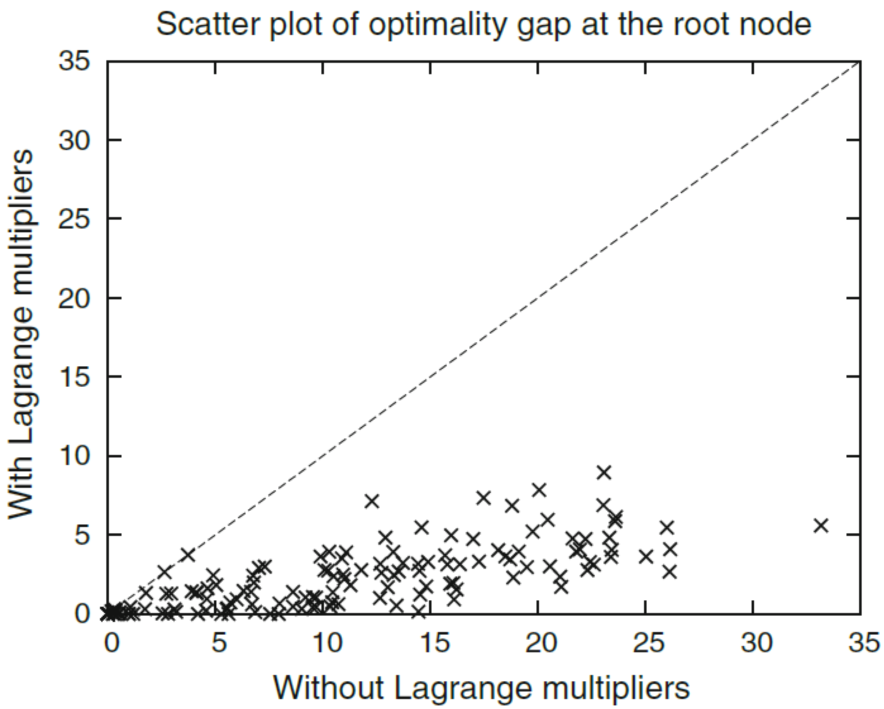
Sequential Ordering Problem (TSPLIB)

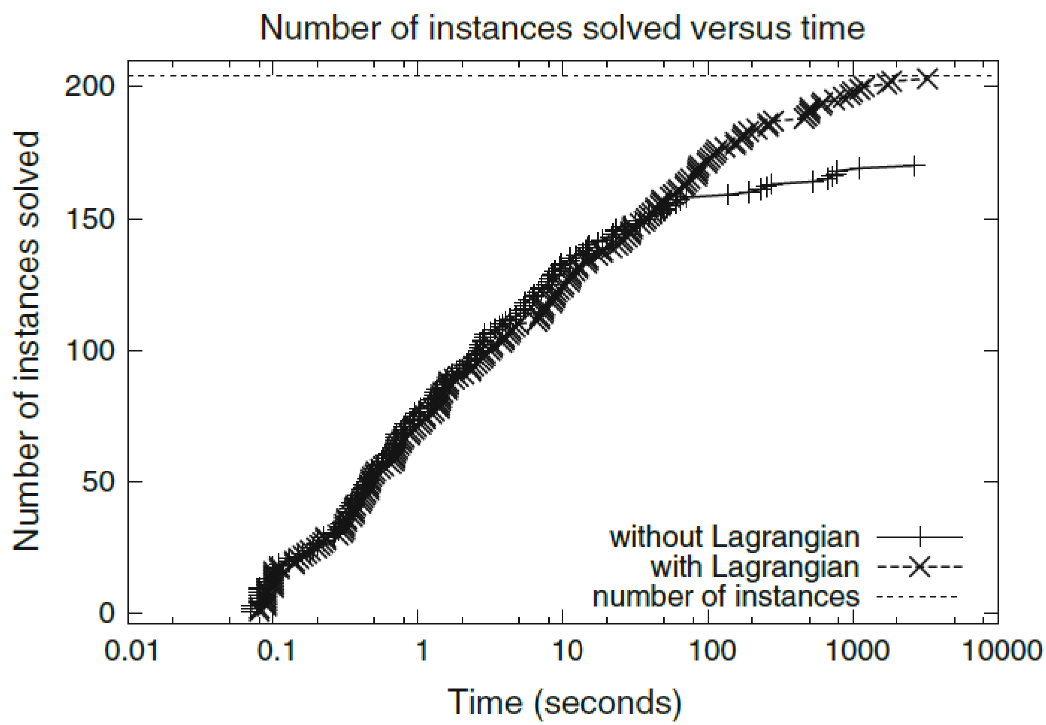
			CPO		CPO+MDD, width 2048	
instance	vertices	bounds	best	time (s)	best	time (s)
br17.10	17	55	55	0.01	55	4.98
br17.12	17	55	55	0.01	55	4.56
ESC07	7	2125	2125	0.01	2125	0.07
ESC25	25	1681	1681	TL	1681	48.42
p43.1	43	28140	28205	TL	28140	287.57
p43.2	43	[28175, 28480]	28545	TL	**28480**	**279.18** *
p43.3	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

Extensions

- Improved bounds
  - Lagrangian relaxation for violated constraints





TSPTW instances (Constraints, 2015)

Extensions

- Improved bounds
  - Lagrangian relaxation for violated constraints
  - Additive bounding to integrate (LP) relaxations

- Sequencing with state-dependent data
  - Position-dependent setup times for single machines
  - TSP with time-dependent travel time

Summary

What can MDDs do for combinatorial 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
- New branch-and-bound scheme

MDDs for constraint-based scheduling

- Constraint propagation with MDDs
- Orders of magnitude improvement possible

Decision Diagrams for Optimization and Scheduling

Preprints, tutorials, presentations, videos, code, benchmark instances:

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