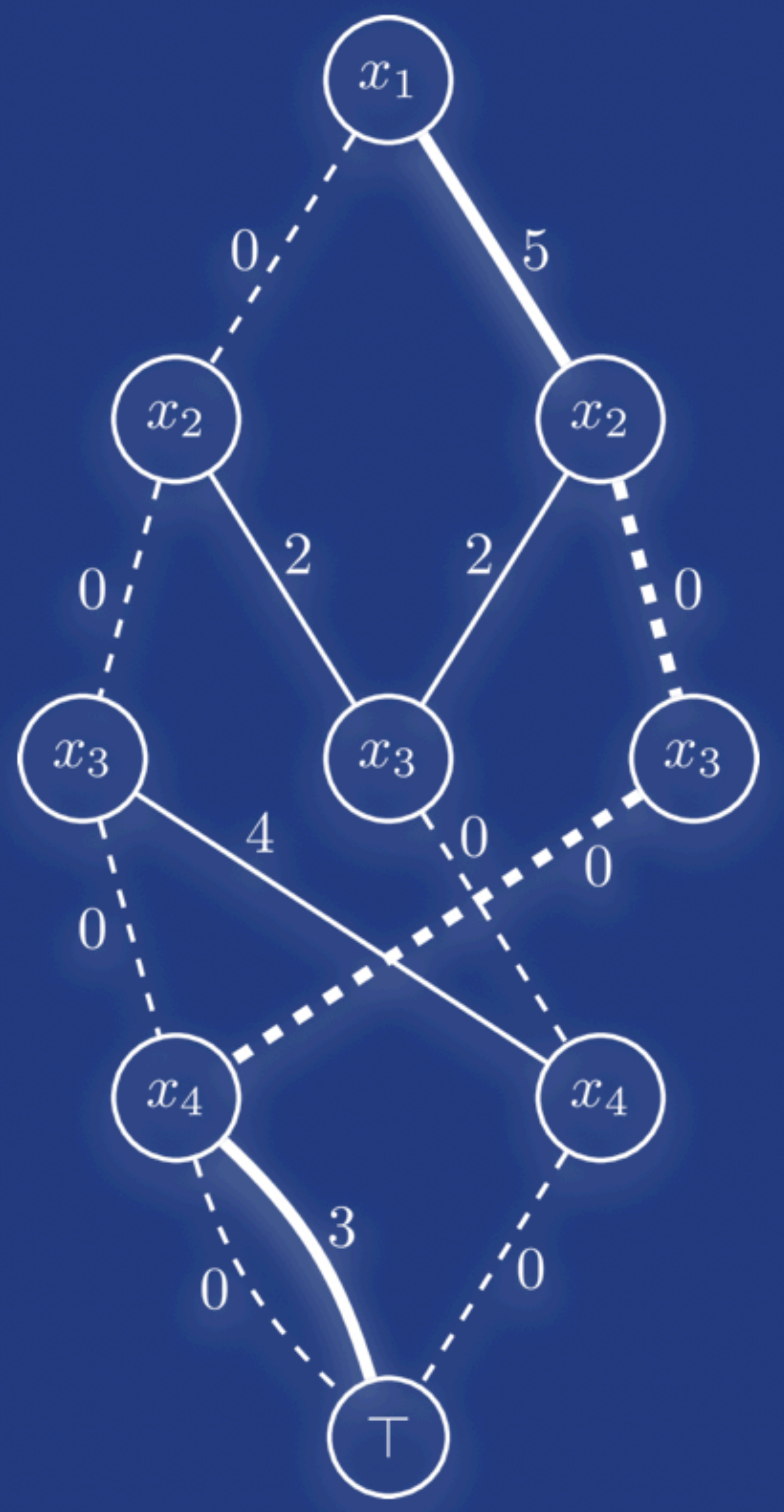
An Introduction to Decision Diagrams for Optimization

Willem-Jan van Hoeve

October 22, 2024 | Seattle, WA



informs ANNUAL MEETING

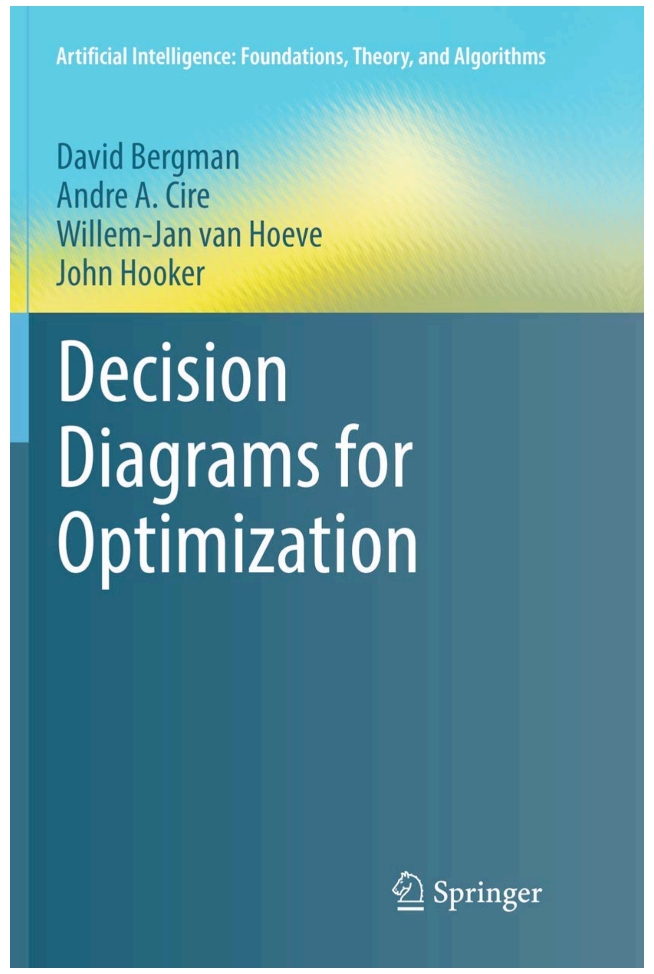
Carnegie Mellon University

Willem-Jan van Hoeve

Carnegie Bosch Professor of Operations Research

Tepper School of Business

Carnegie Mellon University



David Bergman
Andre Cire
Willem-Jan van Hoeve
John Hooker

Springer, 2016

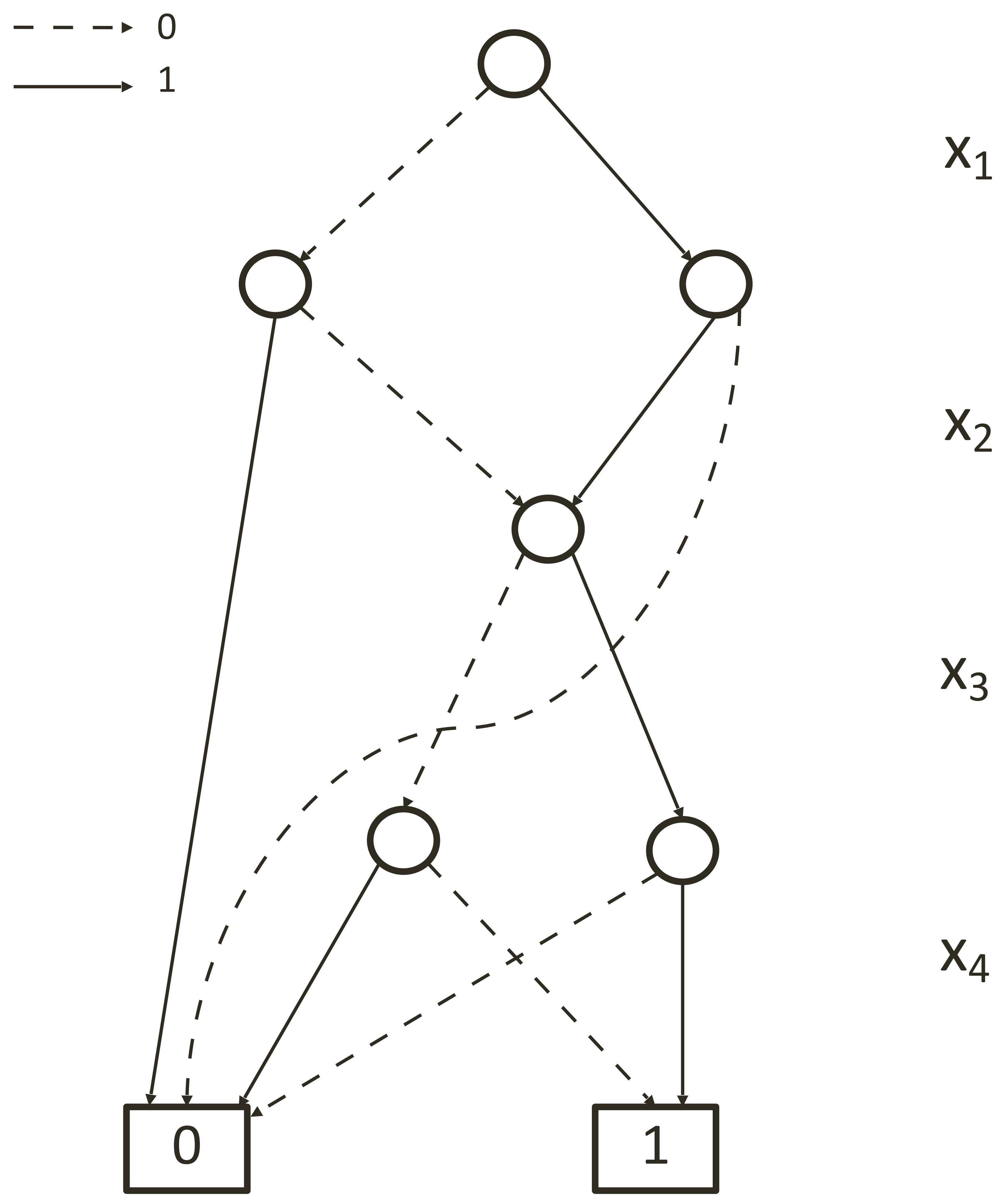
website url

Decision Diagrams?

Graphical representation of
Boolean functions

$$f(x) = (x_1 \Leftrightarrow x_2) \wedge (x_3 \Leftrightarrow x_4)$$

x_1	x_2	x_3	x_4	**f(x)**
0	0	0	0	**1**
0	0	0	1	**0**
0	1	1	0	**0**
0	0	1	1	**1**
…	…	…	…	**…**

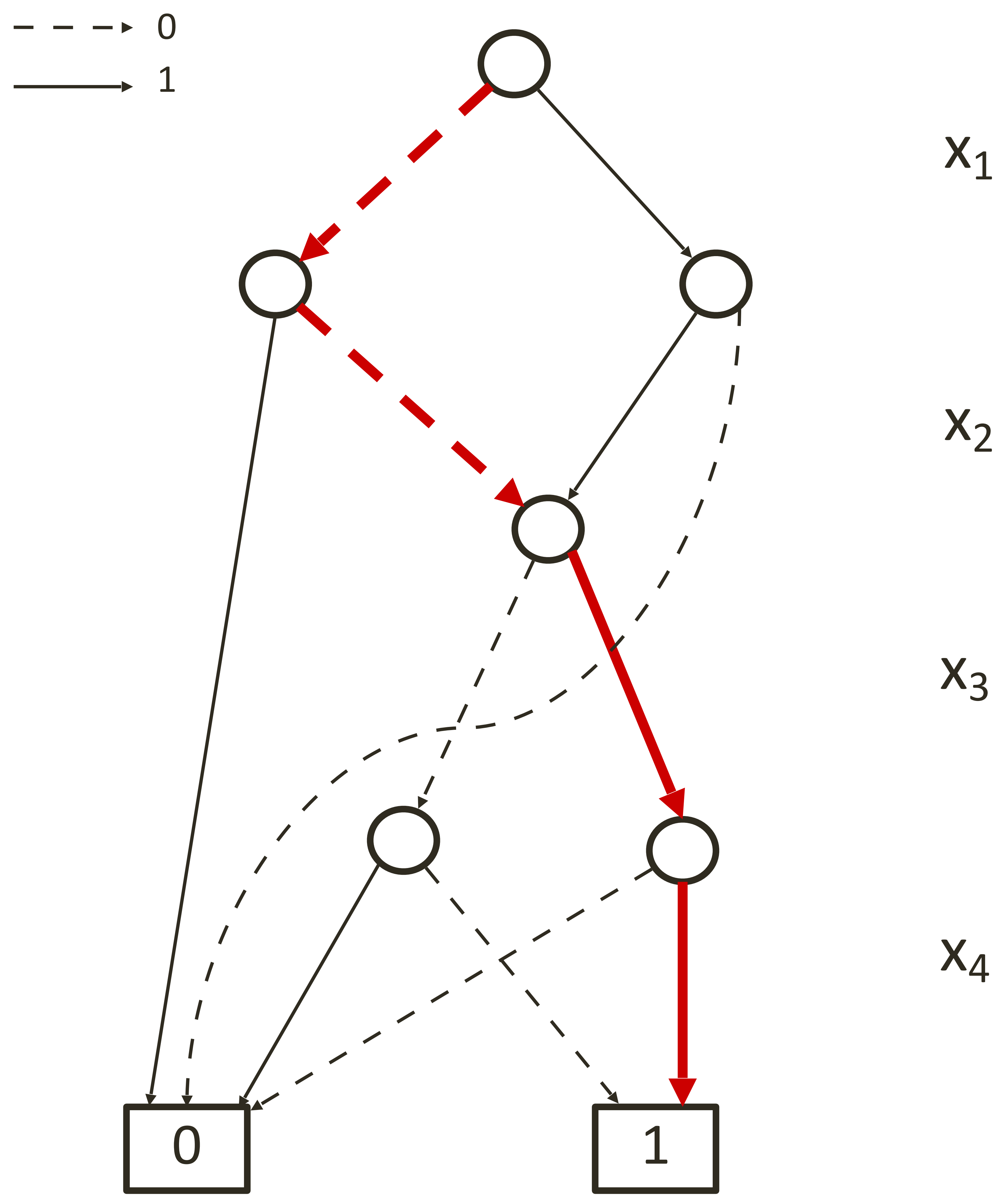


Decision Diagrams?

Graphical representation of
Boolean functions

$$f(x) = (x_1 \Leftrightarrow x_2) \wedge (x_3 \Leftrightarrow x_4)$$

x_1	x_2	x_3	x_4	**f(x)**
0	0	0	0	**1**
0	0	0	1	**0**
0	1	1	0	**0**
0	0	1	1	**1**
…	…	…	…	**…**



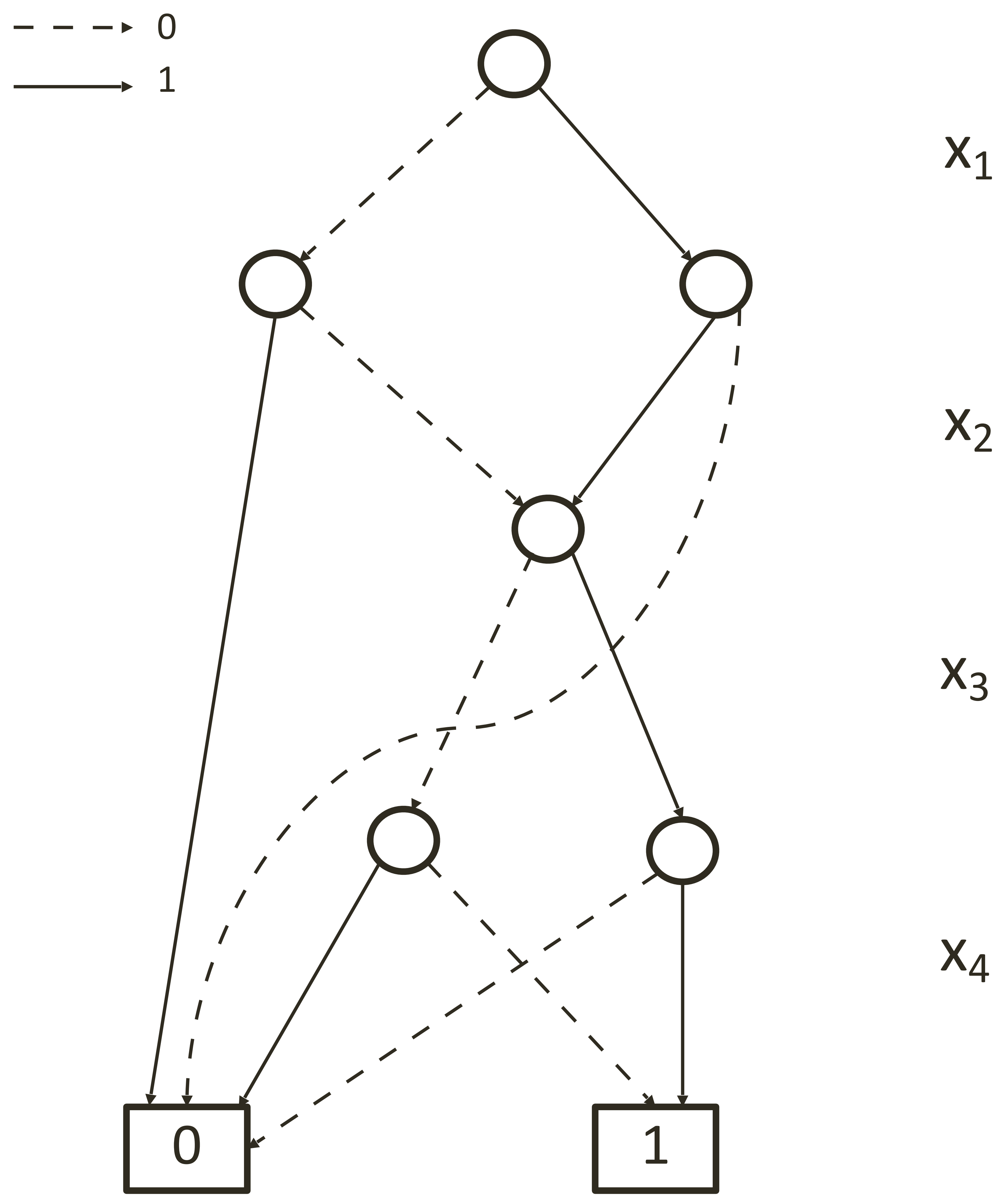
Decision Diagrams?

Graphical representation of
Boolean functions

$$f(x) = (x_1 \Leftrightarrow x_2) \wedge (x_3 \Leftrightarrow x_4)$$

- BDD: binary decision diagram
- MDD: multi-valued decision diagram

Applications: Formal verification, configuration problems, …



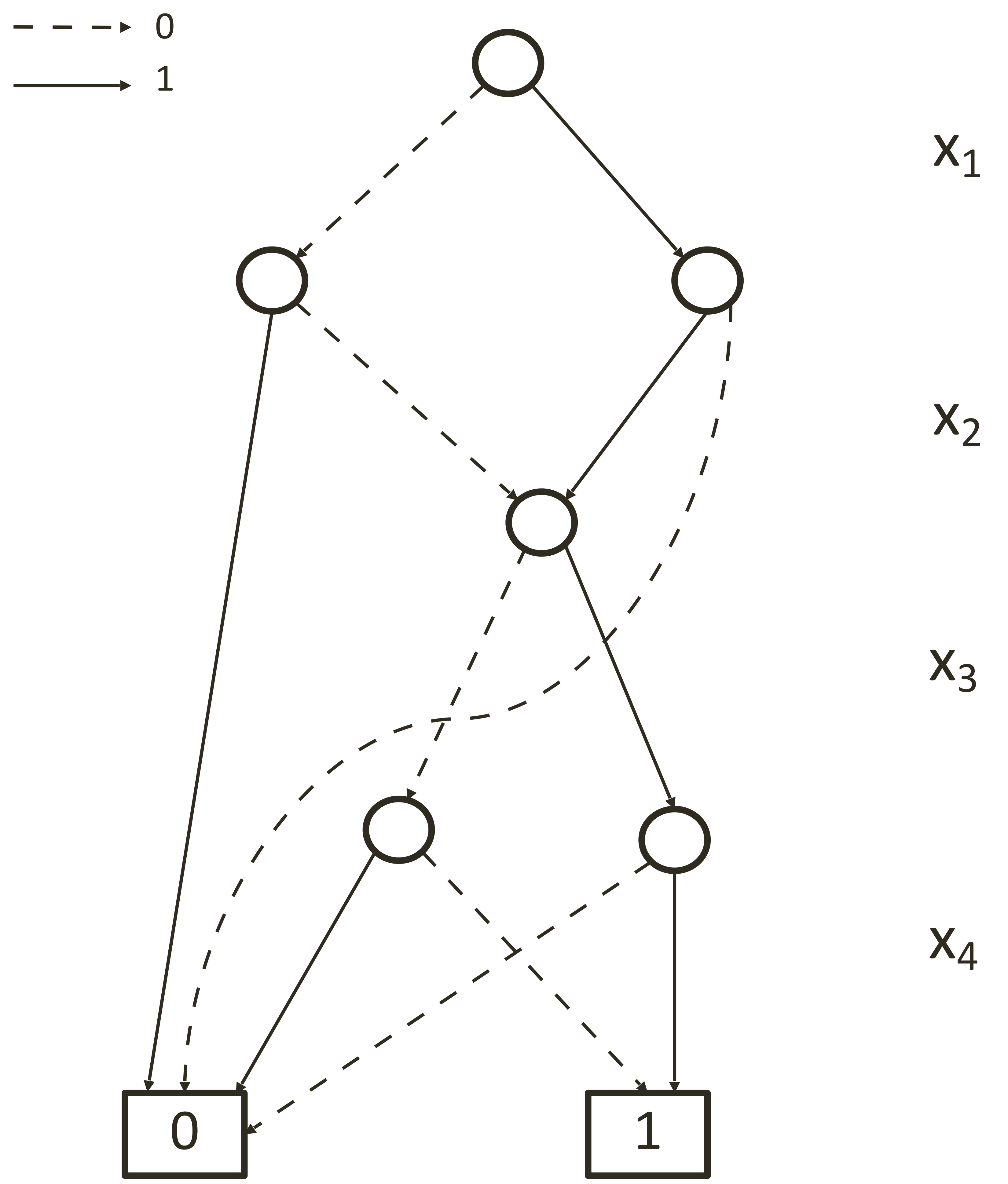
Decision Diagrams: Optimization View

Graphical representation of
Boolean functions

$$f(x) = (x_1 \Leftrightarrow x_2) \wedge (x_3 \Leftrightarrow x_4)$$

Optimization perspective:

- literals → variables
- arcs → assignments
- paths → solutions



Decision Diagrams: Optimization View

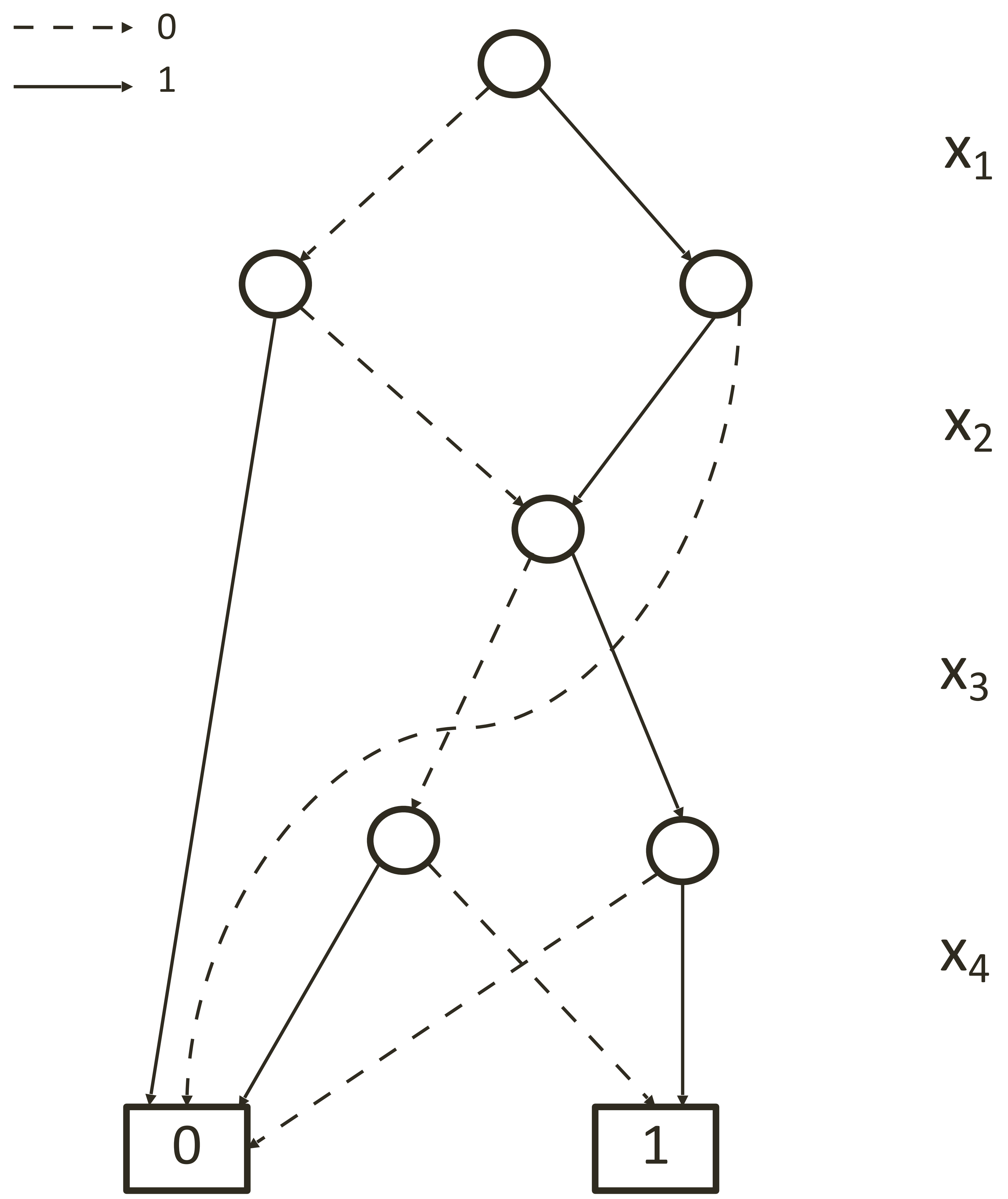
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

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



Decision Diagrams: Optimization View

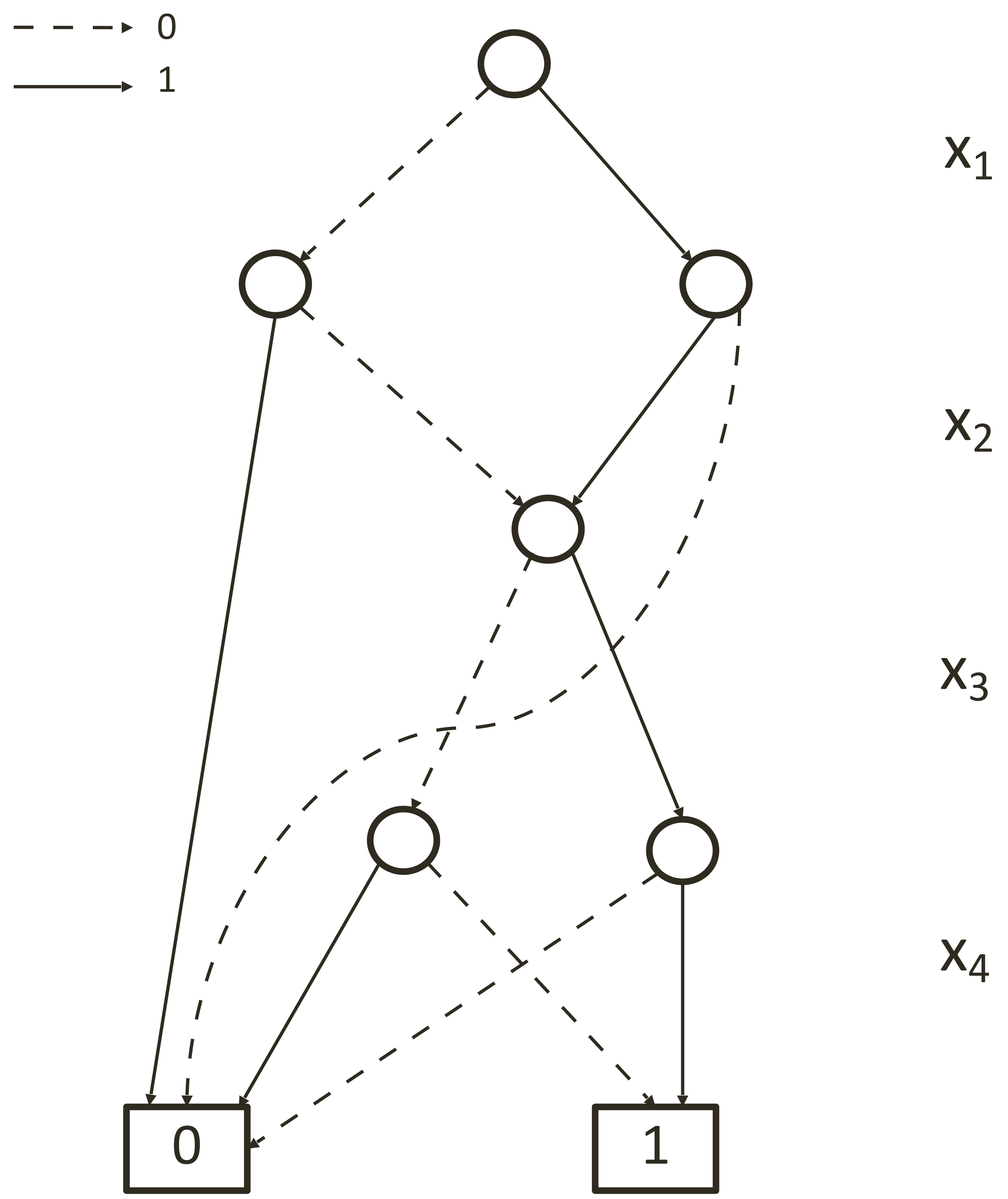
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

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



Decision Diagrams: Optimization View

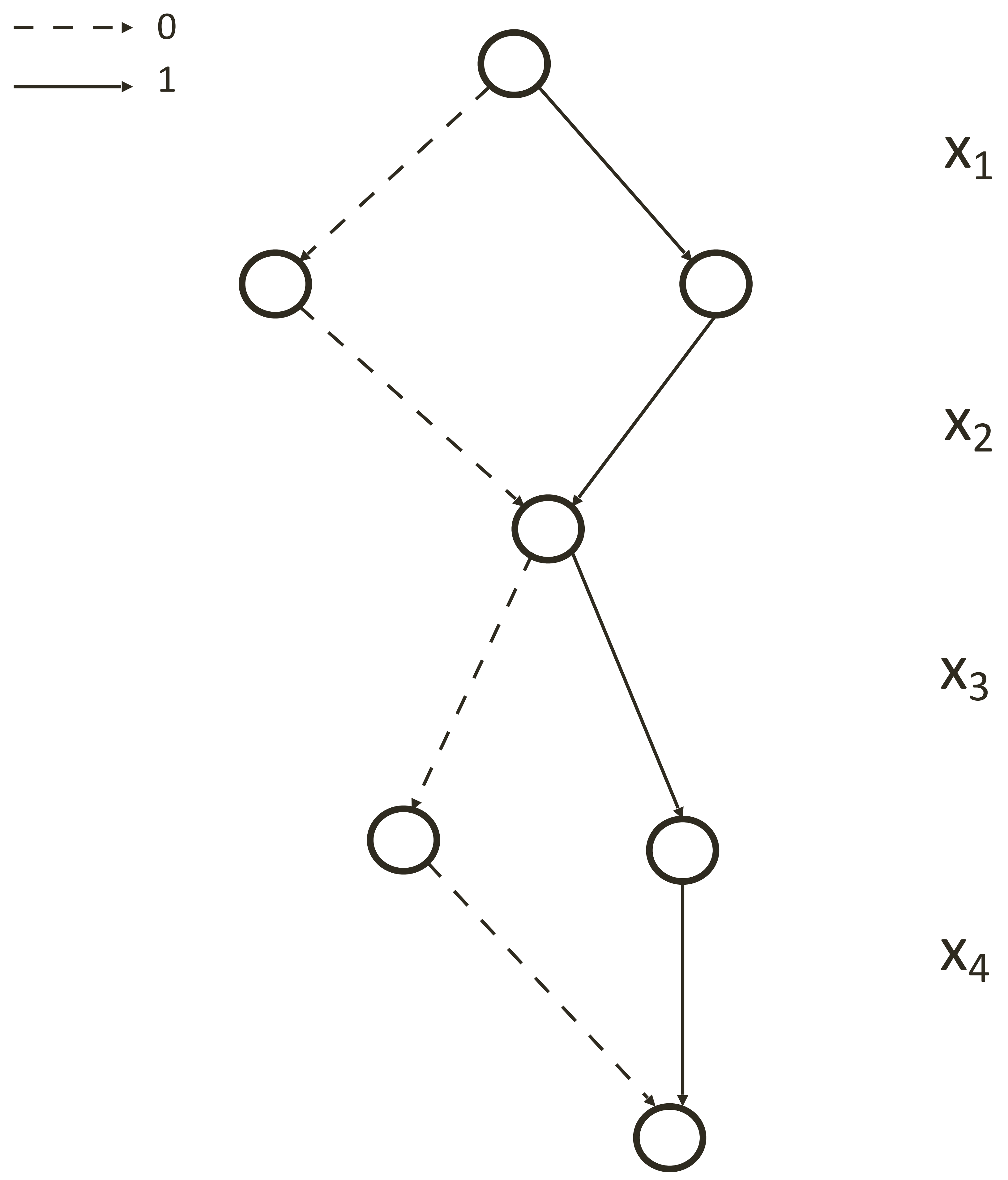
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

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



Decision Diagrams: Optimization View

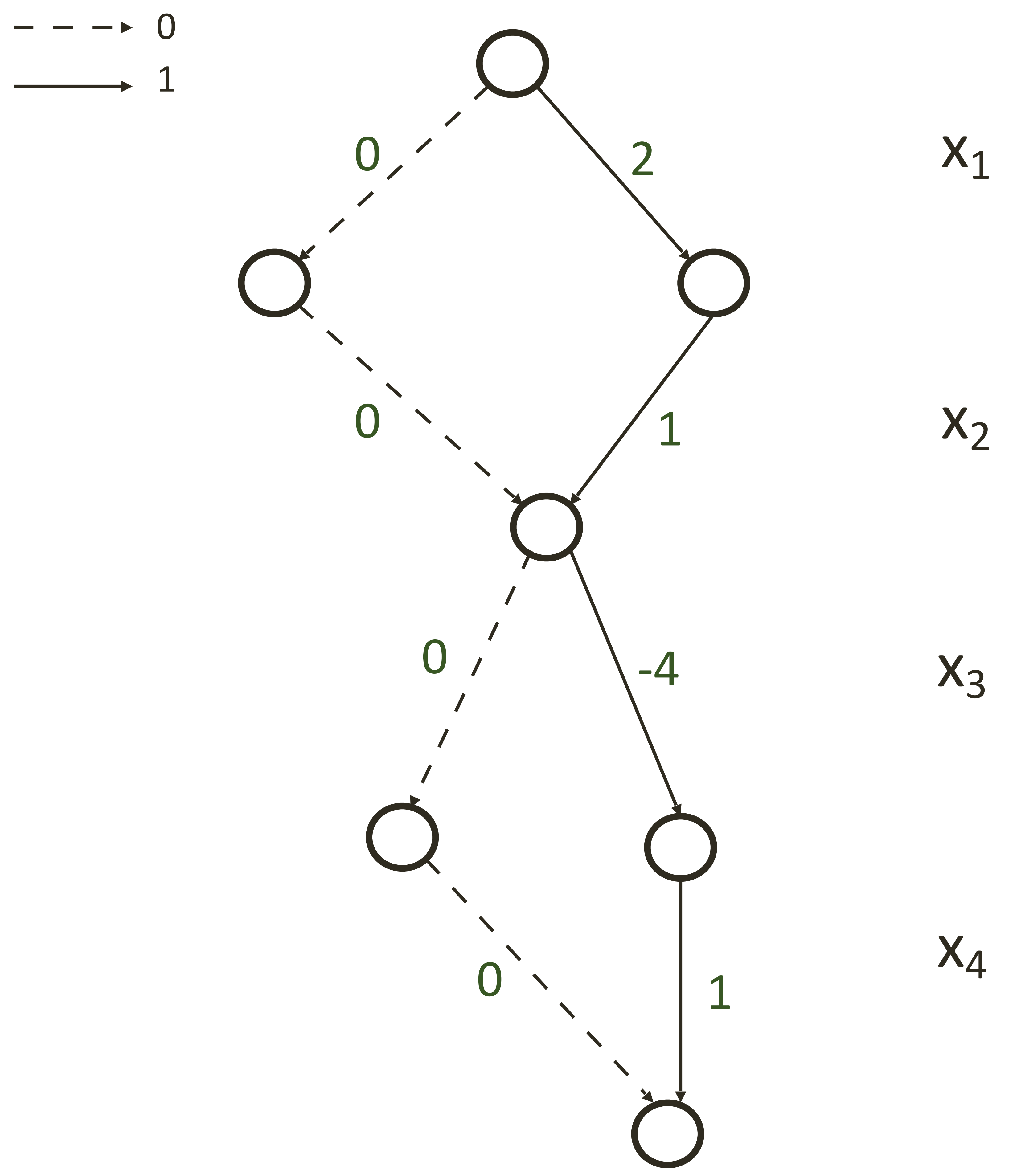
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

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



Maximizing a linear (or separable) function:

- Arc lengths: contribution to the objective
- Longest path: optimal solution

  (can also handle nonlinear functions)

Decision Diagrams: Optimization View

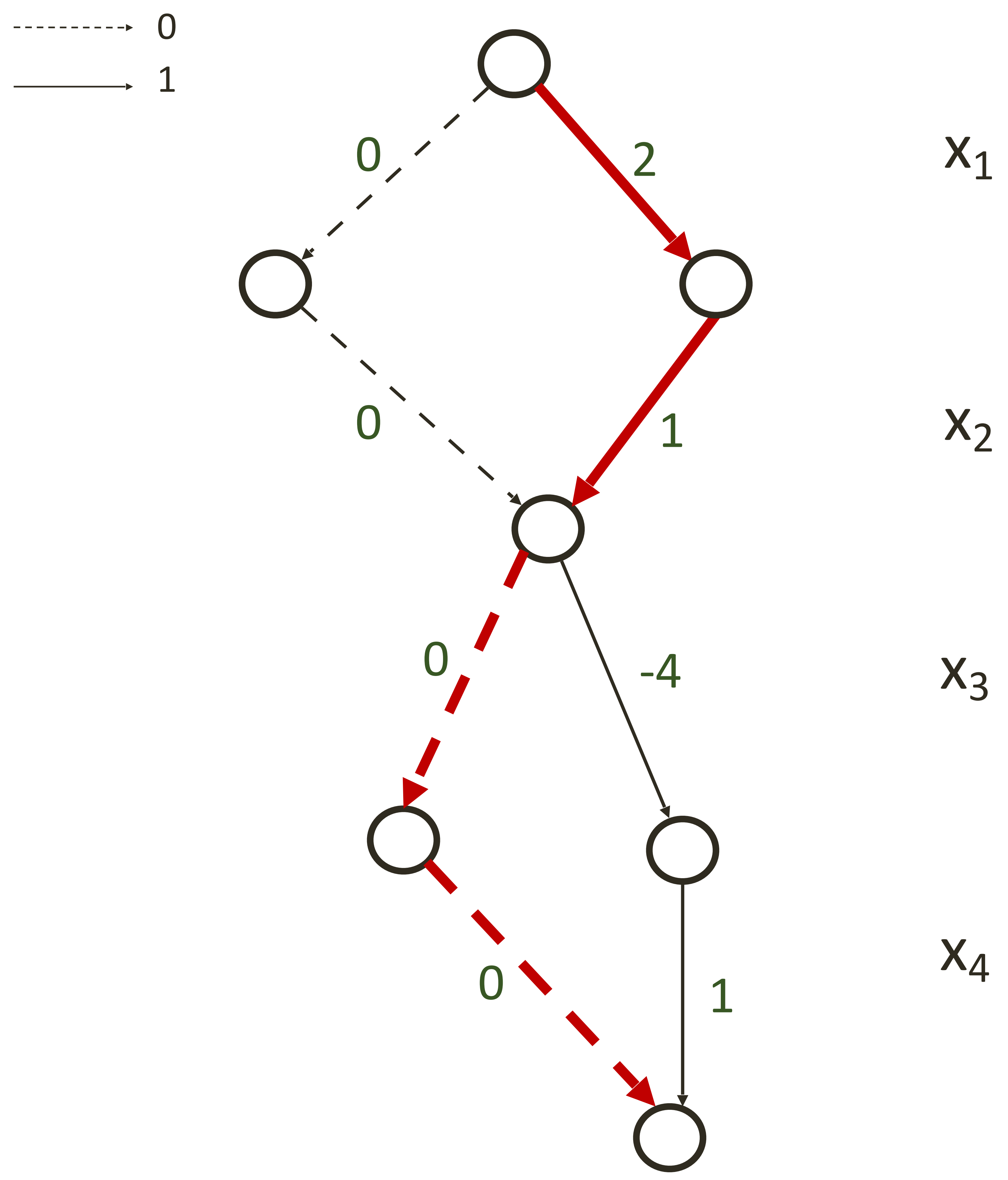
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

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



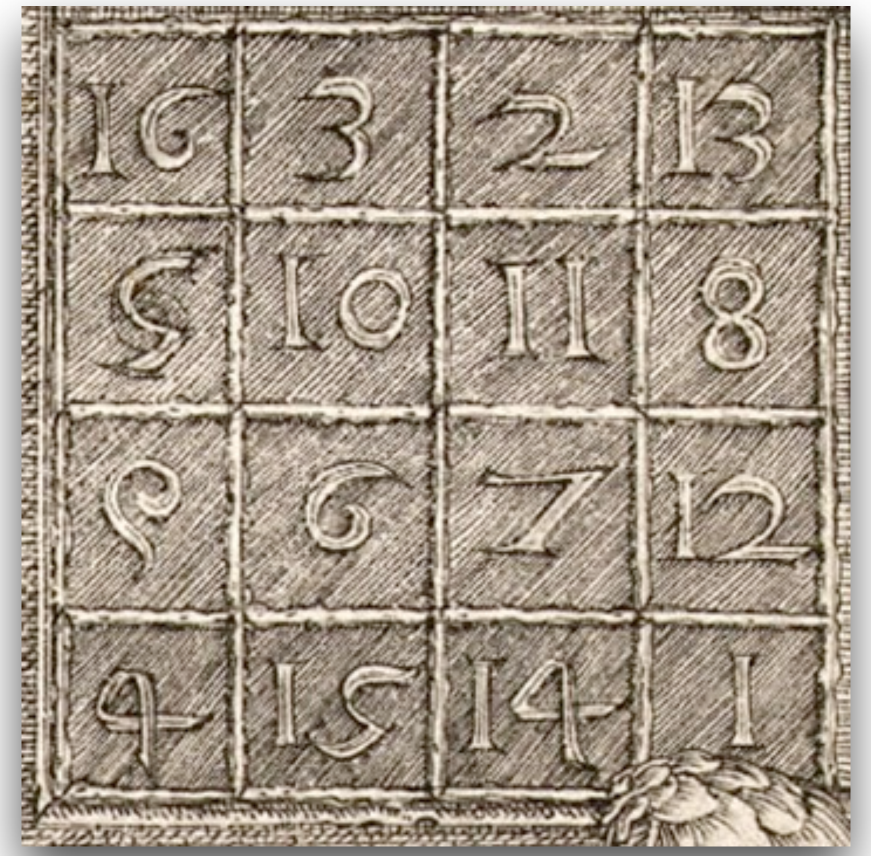
Maximizing a linear (or separable) function:

- Arc lengths: contribution to the objective
- Longest path: optimal solution

(can also handle nonlinear functions)

(DALL-E, 2024)

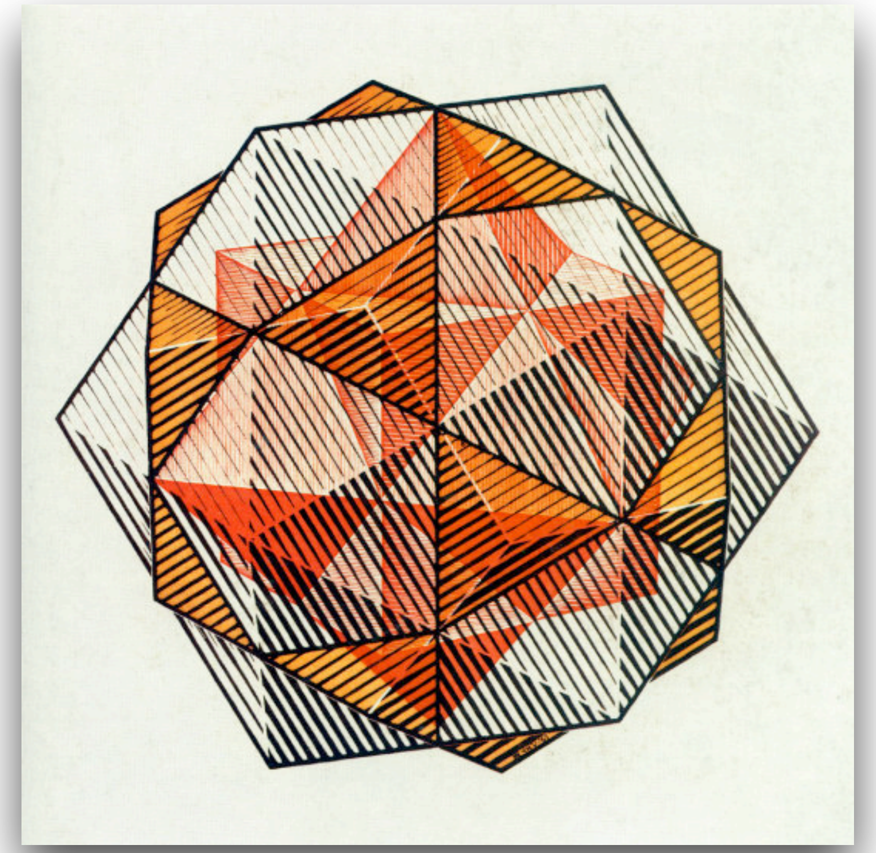
Branch-and-Bound Solver



(Durer, 1514)

Constraint Programming

(Escher, 1961)



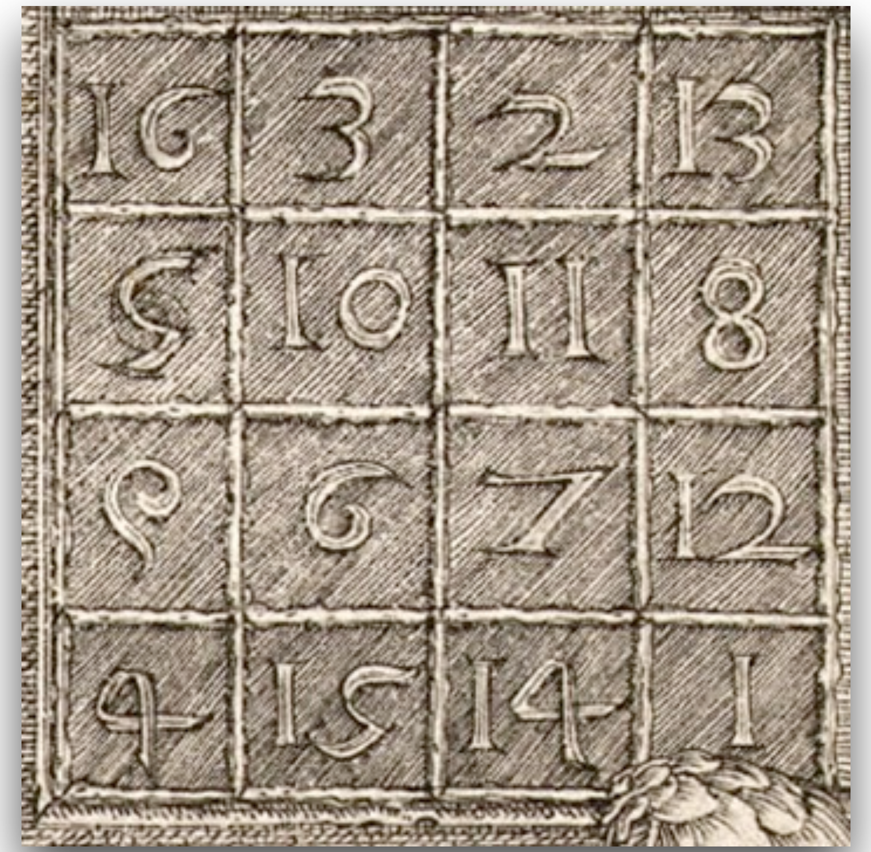
Integer Programming

(DALL-E, 2024)

Column Elimination

(DALL-E, 2024)

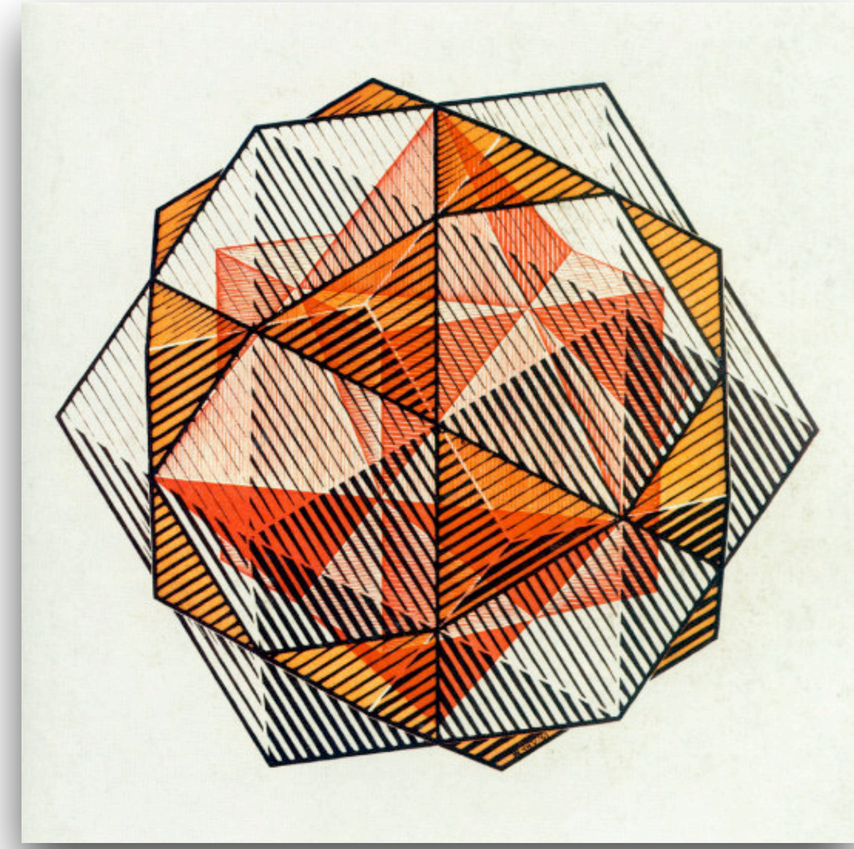
Branch-and-Bound Solver



(Durer, 1514)

Constraint Programming

(Escher, 1961)

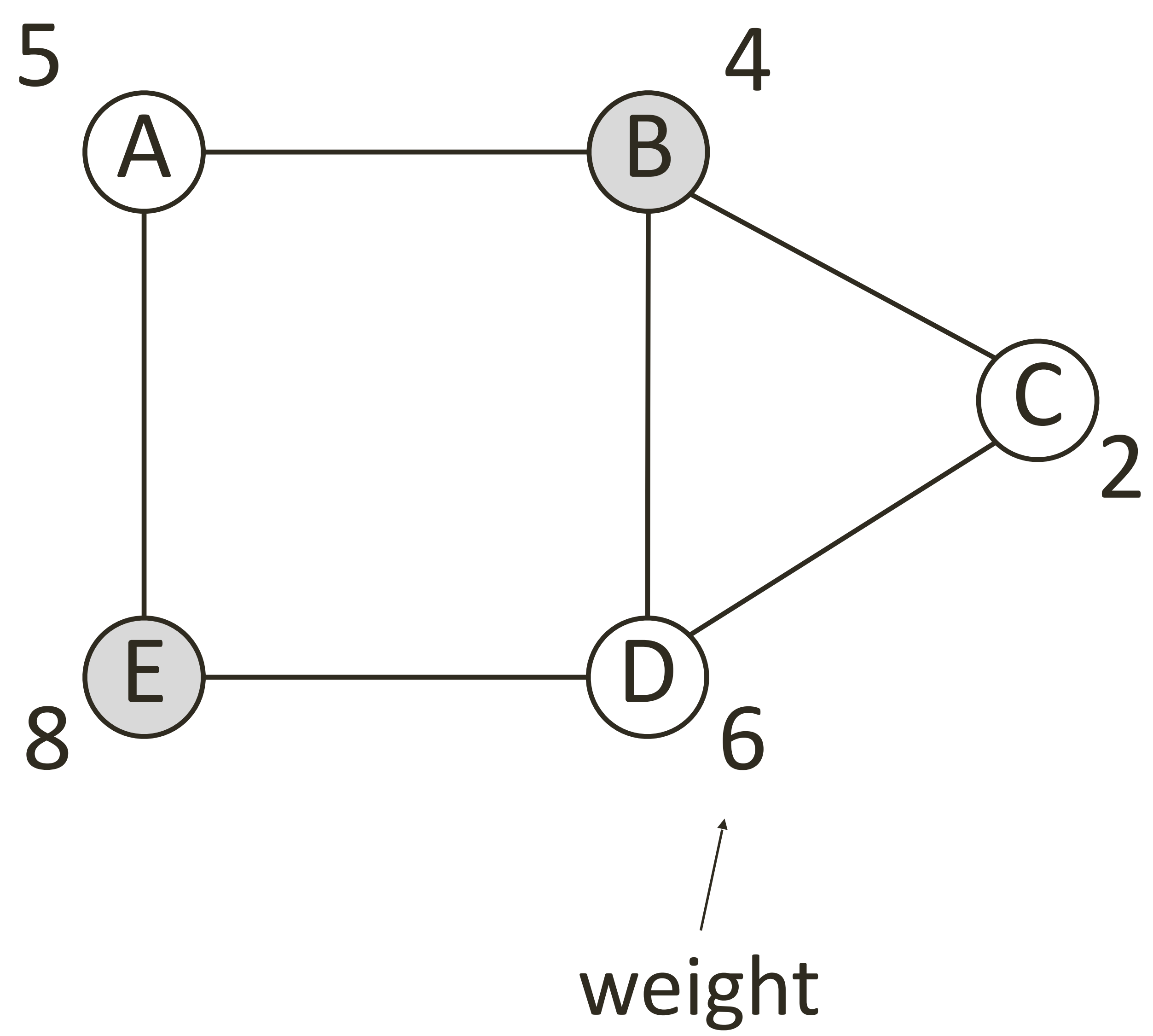


Integer Programming

(DALL-E, 2024)

Column Elimination

Example Application: Independent Set Problem



Independent set in a graph:

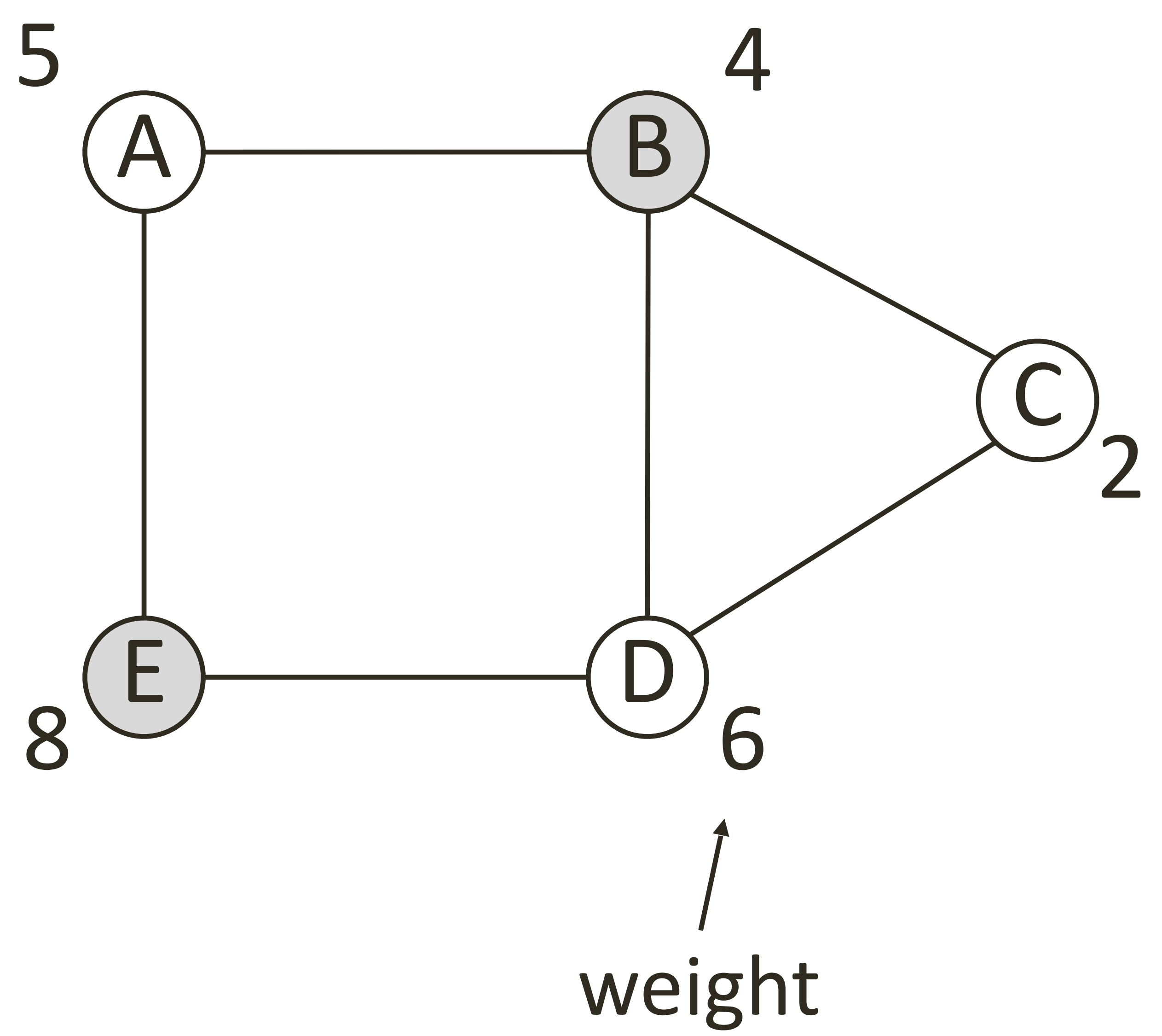
- Subset of non-adjacent vertices

Maximum Independent Set Problem:

- Find independent set with maximum weight

- Classical combinatorial optimization problem (equivalent to maximum clique in complement graph)
- Wide applications, ranging from scheduling to social network analysis

Integer Programming Formulation



Independent set in a graph:

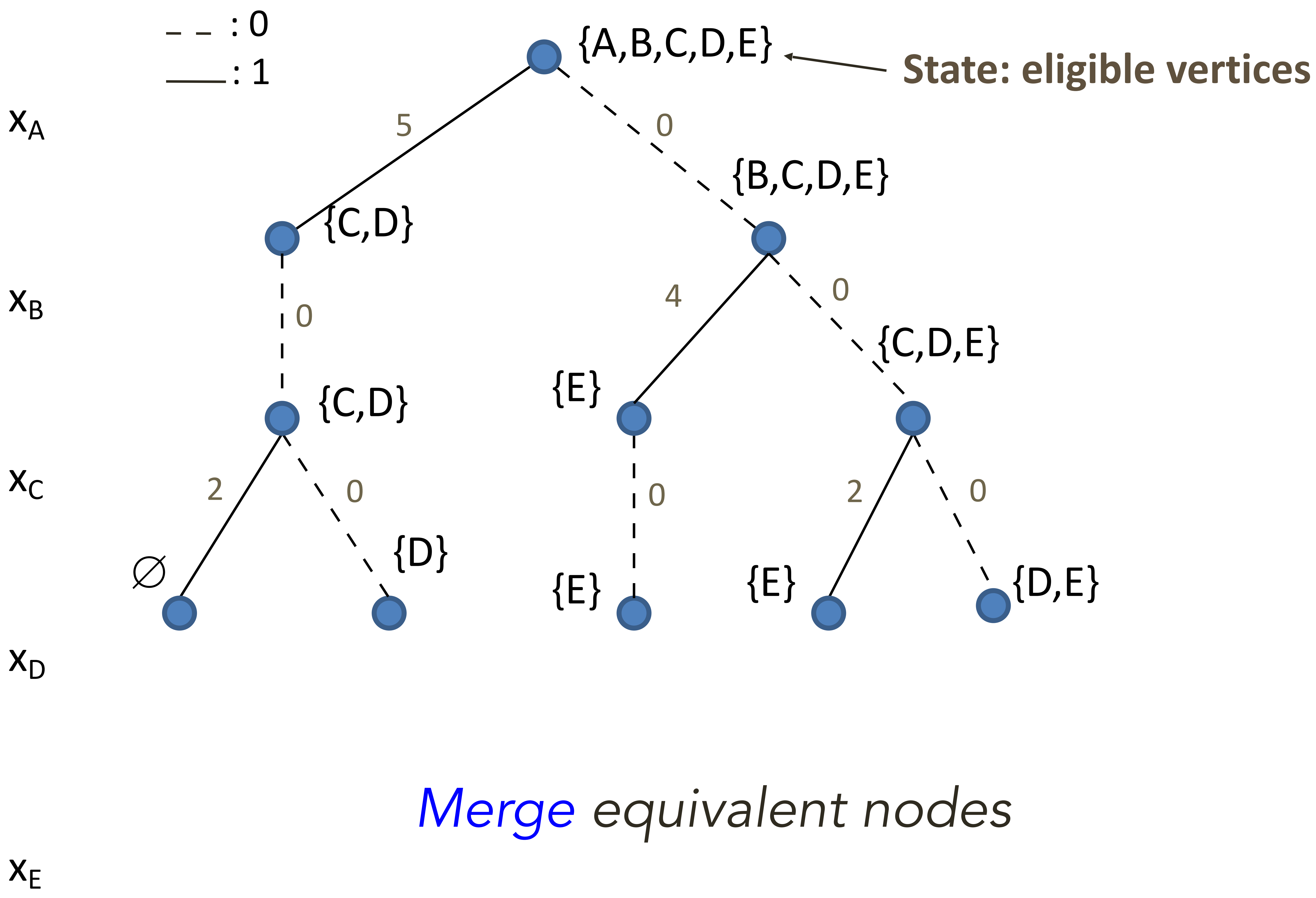
- Subset of non-adjacent vertices

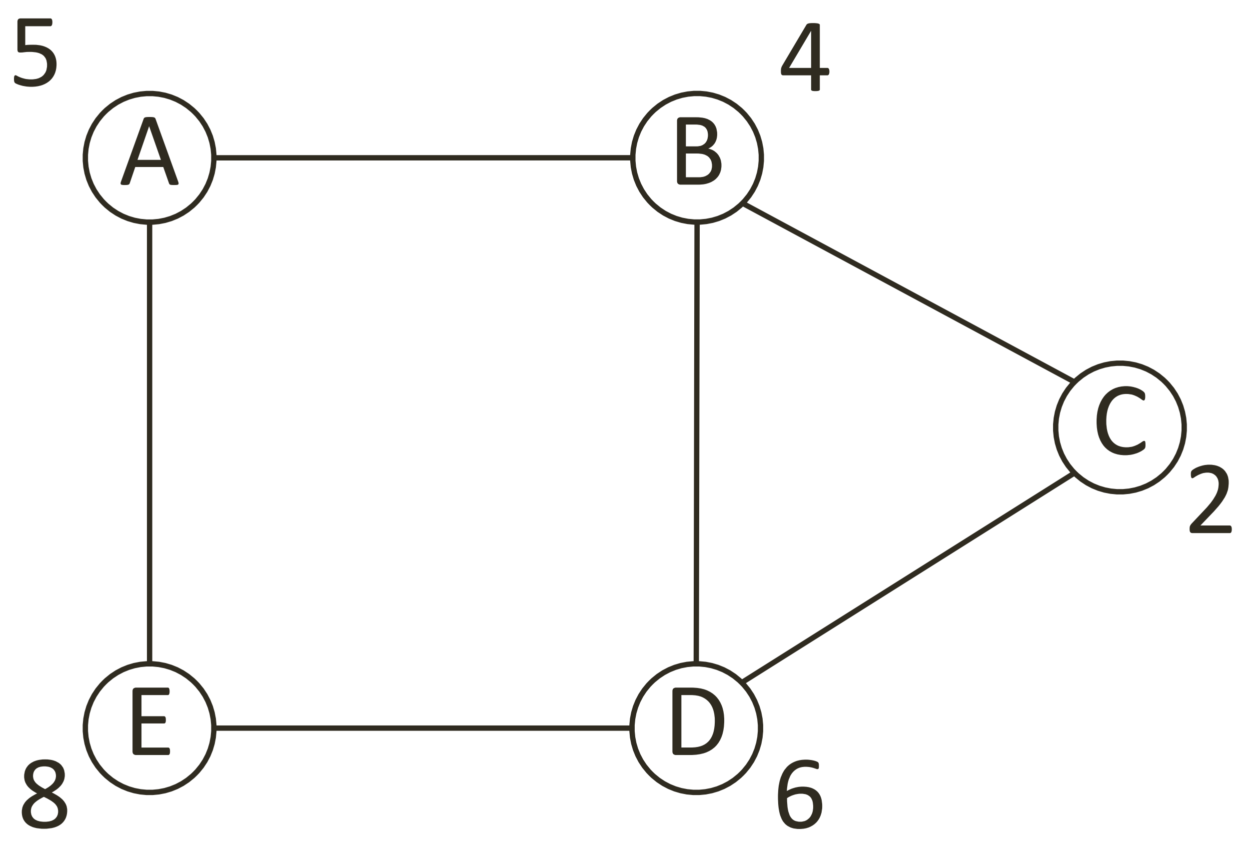
Maximum Independent Set Problem:

- Find independent set with maximum weight

$$
\begin{aligned}
\text{max} \quad & 5x_A + 4x_B + 2x_C + 6x_D + 8x_E \\
\text{subject to} \quad & x_A + x_B \leq 1 \\
& x_A + x_E \leq 1 \\
& x_B + x_C \leq 1 \\
& x_B + x_D \leq 1 \\
& x_C + x_D \leq 1 \\
& x_D + x_E \leq 1 \\
& x_A, x_B, x_C, x_D, x_E \in \{0,1\}
\end{aligned}
$$

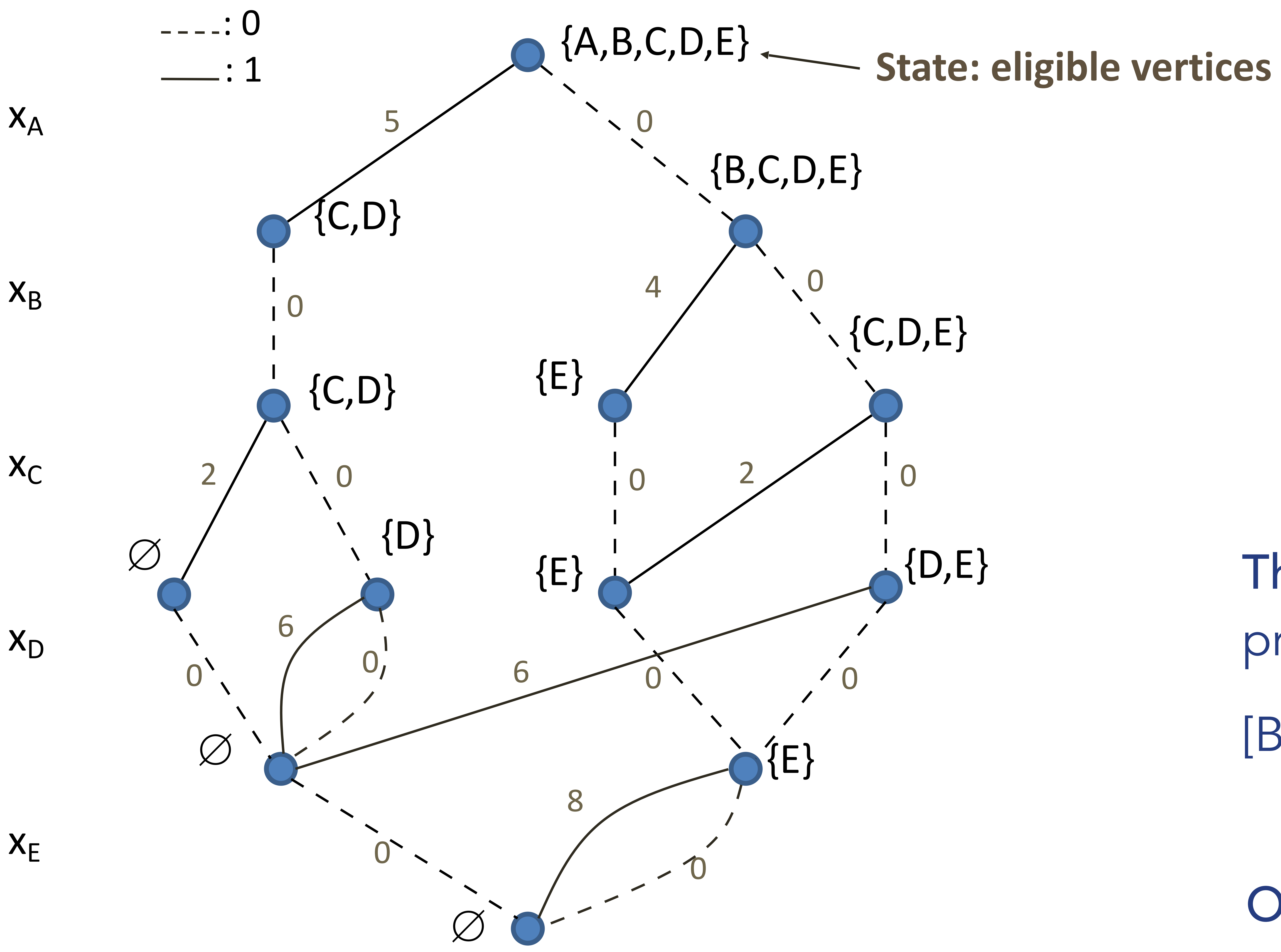
BDD Compilation for Maximum Independent Set

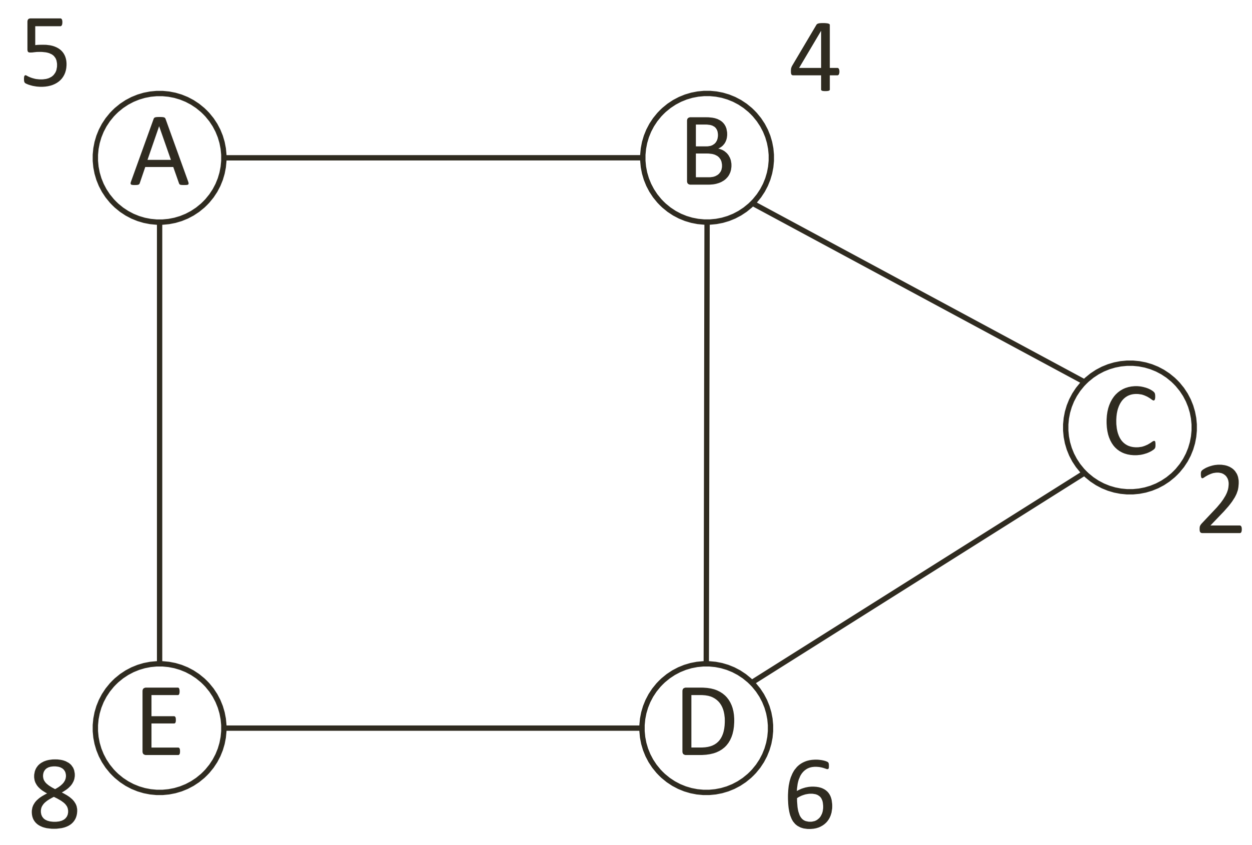




Merge equivalent nodes

BDD Compilation for Maximum Independent Set



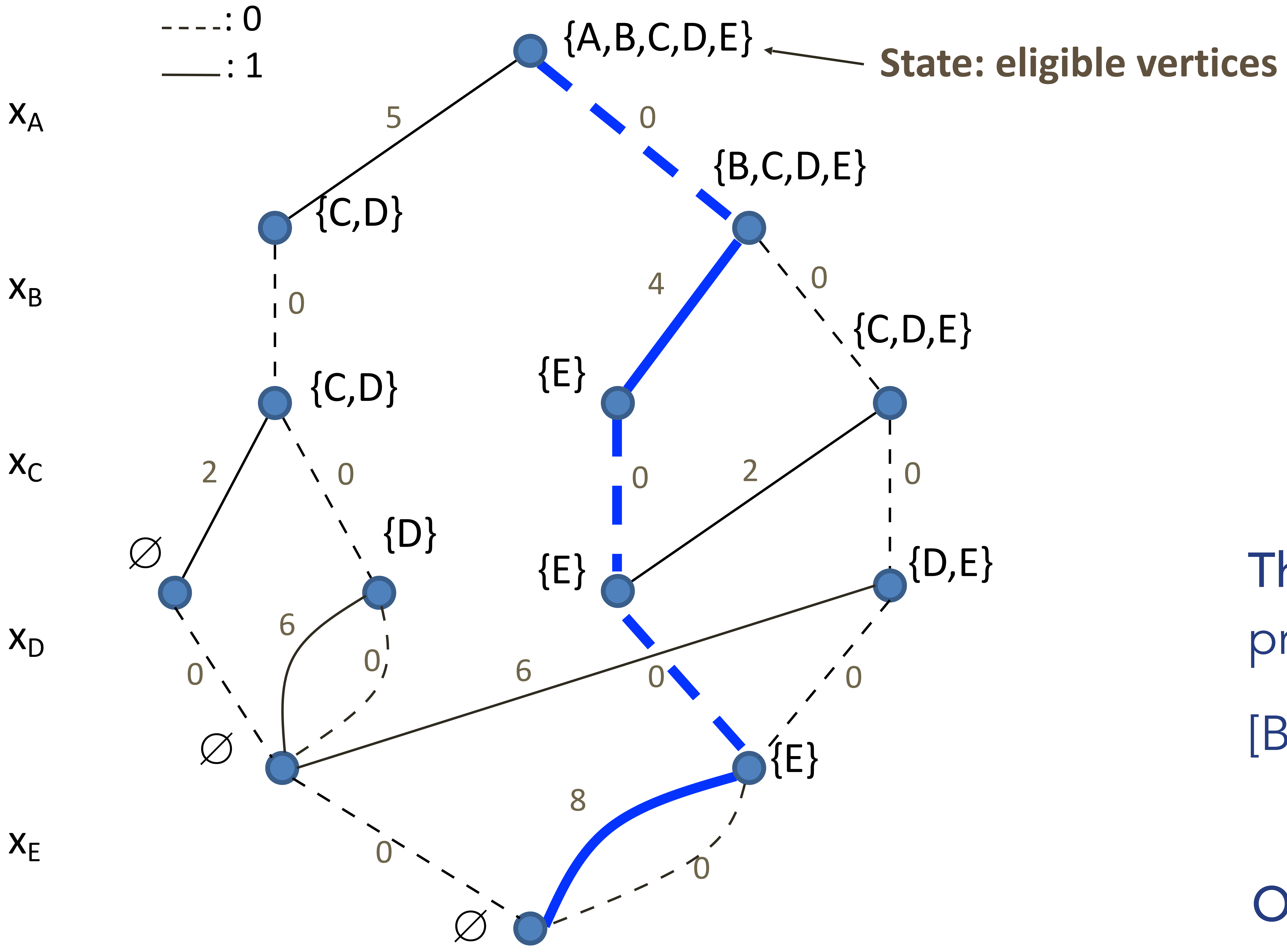


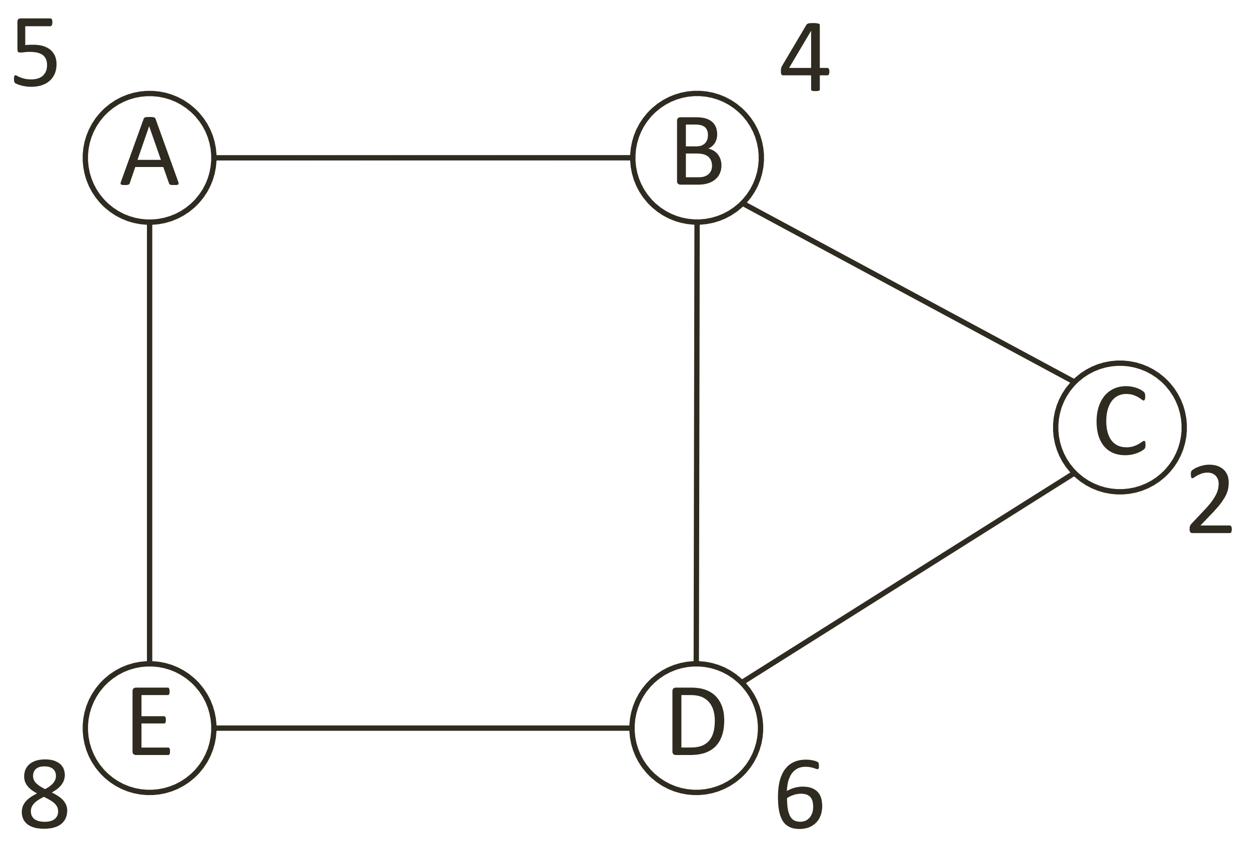
Theorem: This top-down compilation procedure generates a reduced exact BDD

[Bergman, Cire, vH, Hooker, IJOC 2014]

Optimal solution: Longest path

BDD Compilation for Maximum Independent Set





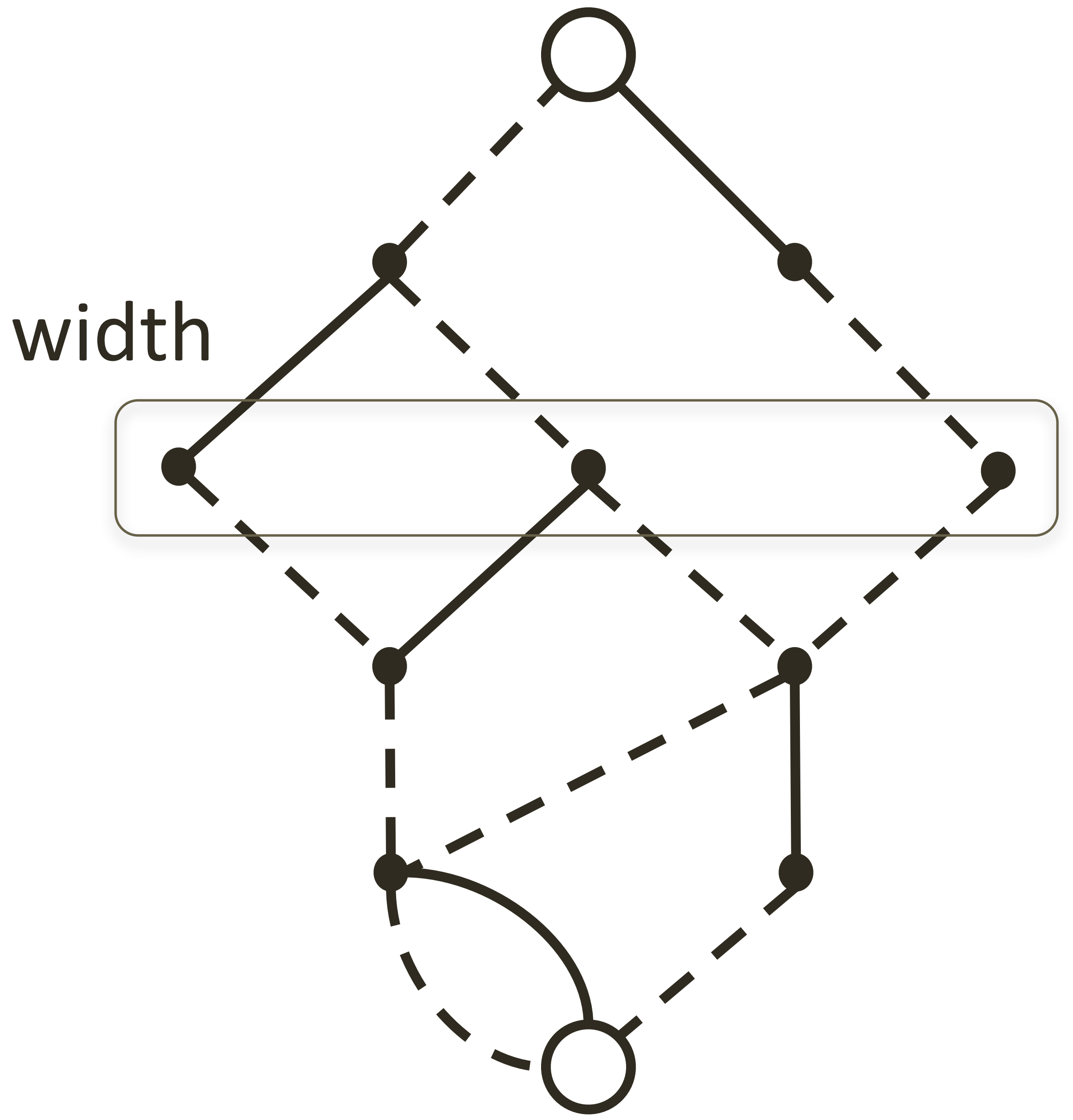
Theorem: This top-down compilation procedure generates a reduced exact BDD

[Bergman, Cire, vH, Hooker, IJOC 2014]

Optimal solution: Longest path

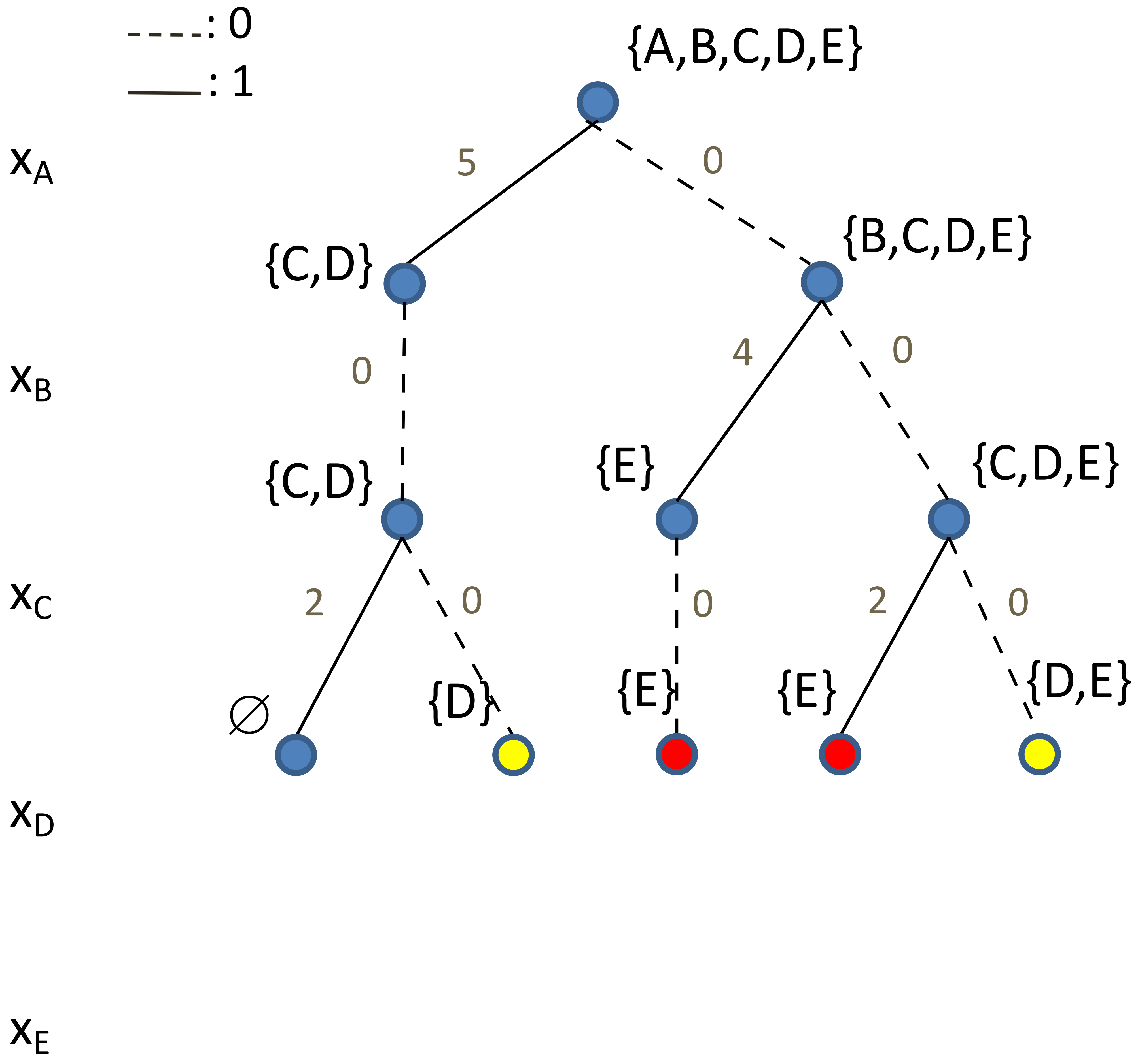
Relaxed Decision Diagrams: Polynomial Size

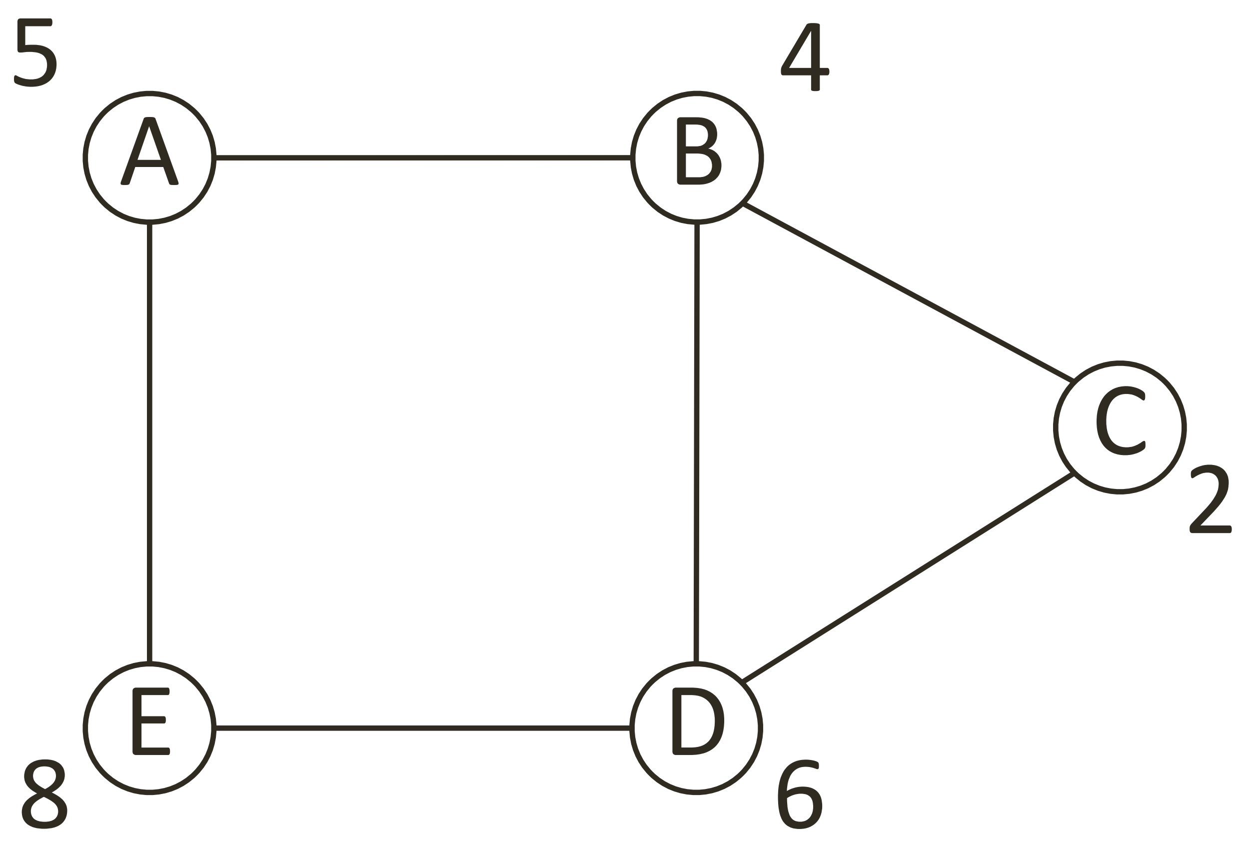
- Limit the size of the diagram to a maximum width
- Merge non-equivalent nodes
  - Define *node merging rule* to safely aggregate states
- Requirements for relaxation
  1. Must represent a superset of exact solutions
  2. The path costs are valid (w.r.t. exact solutions)
- Provides discrete relaxation
  - Strength is controlled by the maximum width



[Andersen, Hadzic, Hooker, Tiedemann, CP 2007]
[Bergman, Cire, vH, Hooker, CPAIOR 2011, IJOC 2016]

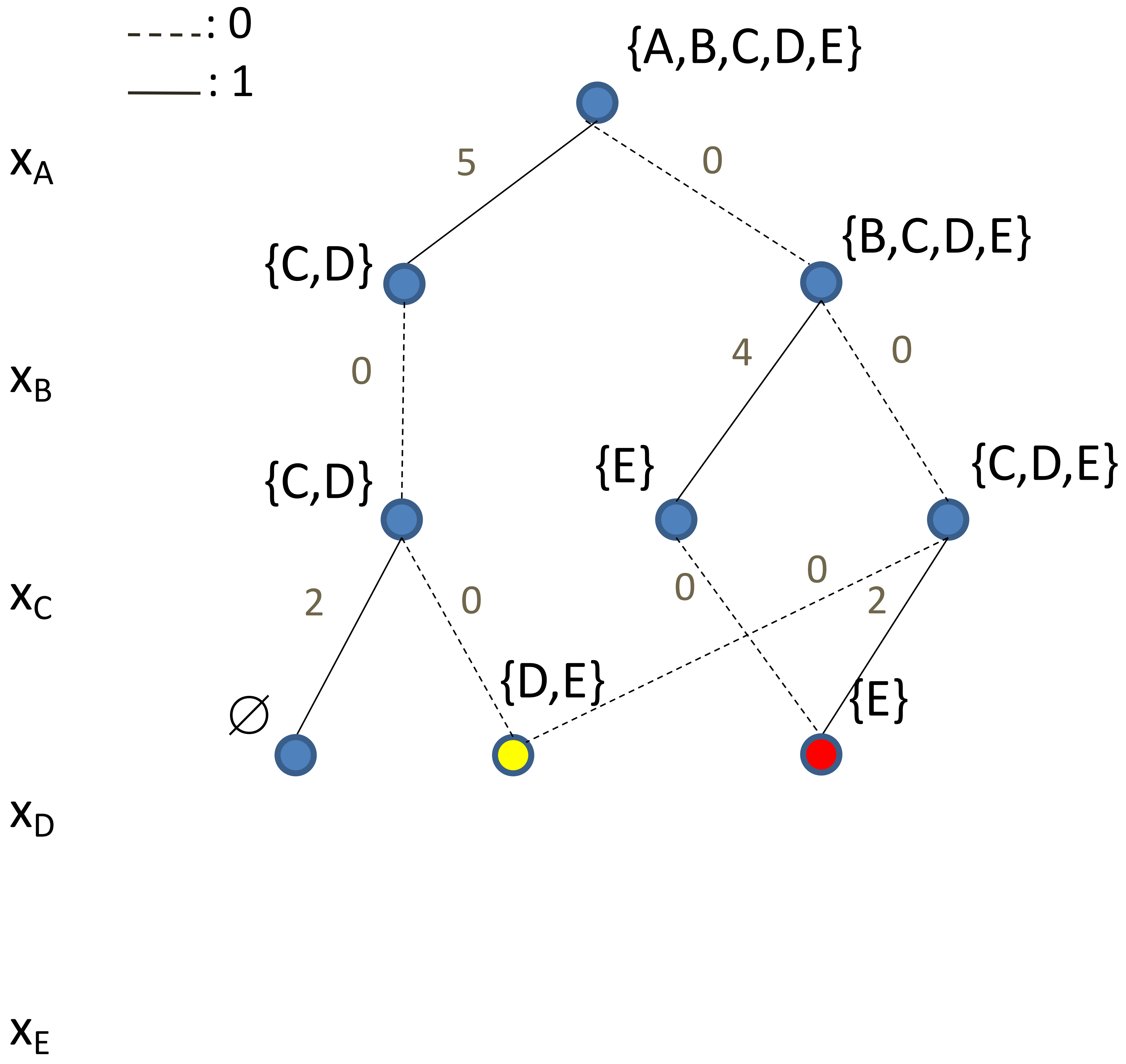
Independent Set Problem: Relaxed DD

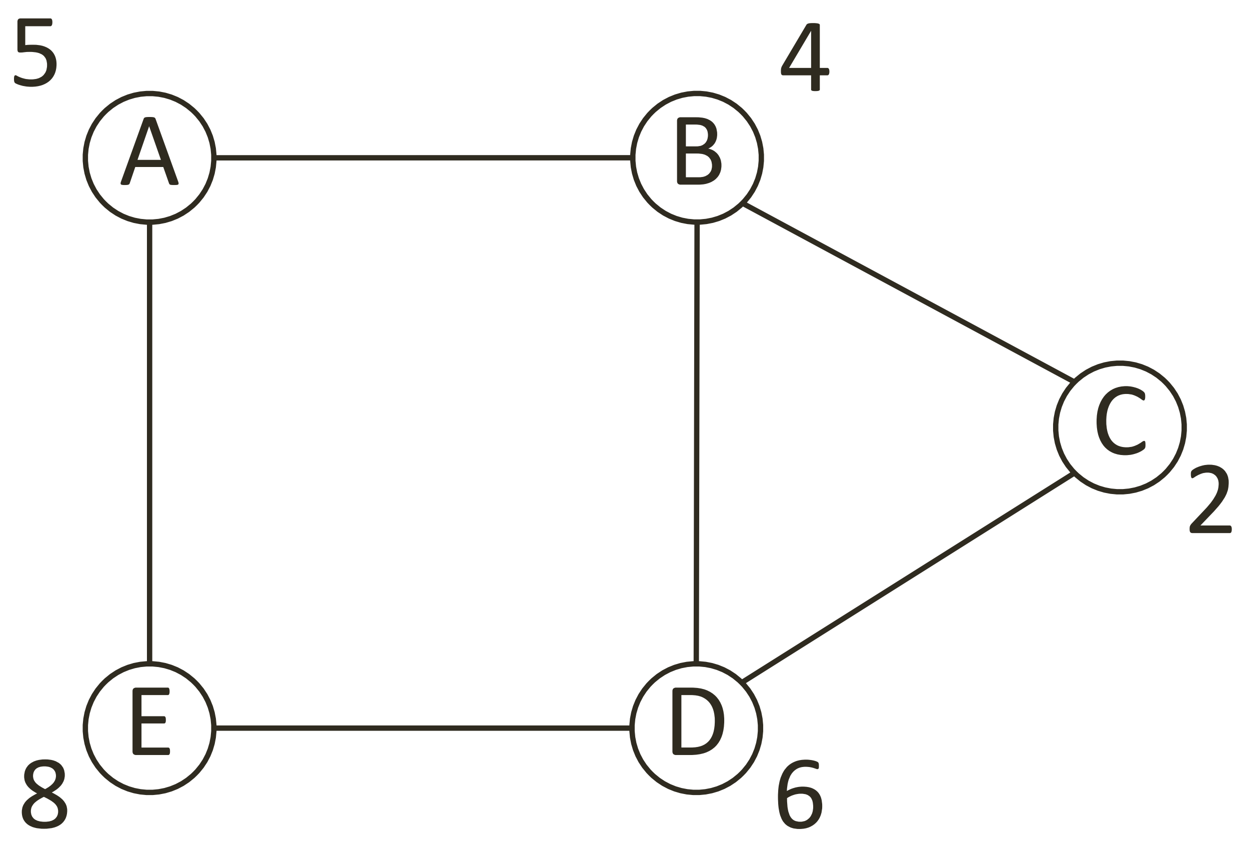




Maximum width = 3

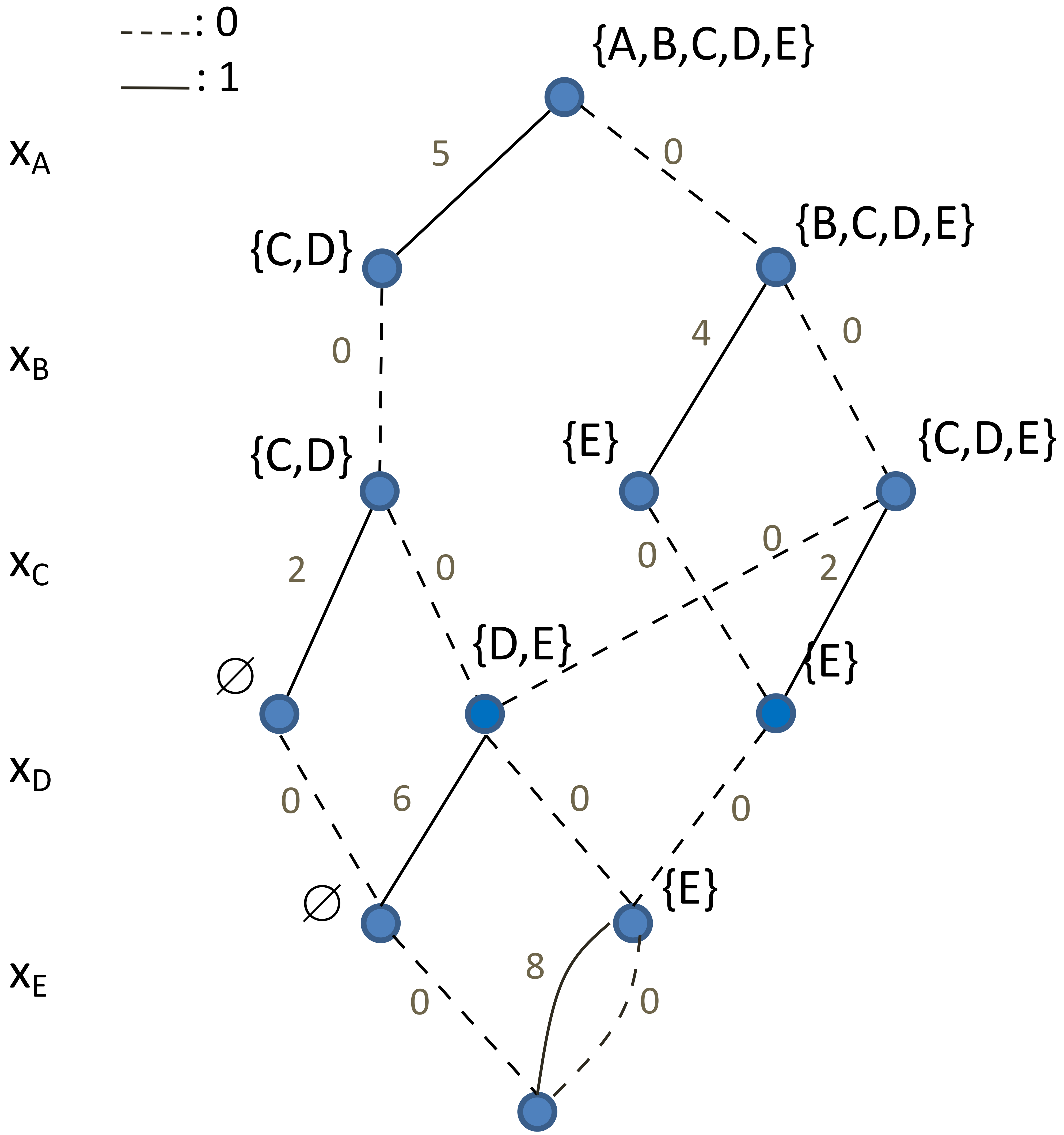
Independent Set Problem: Relaxed DD

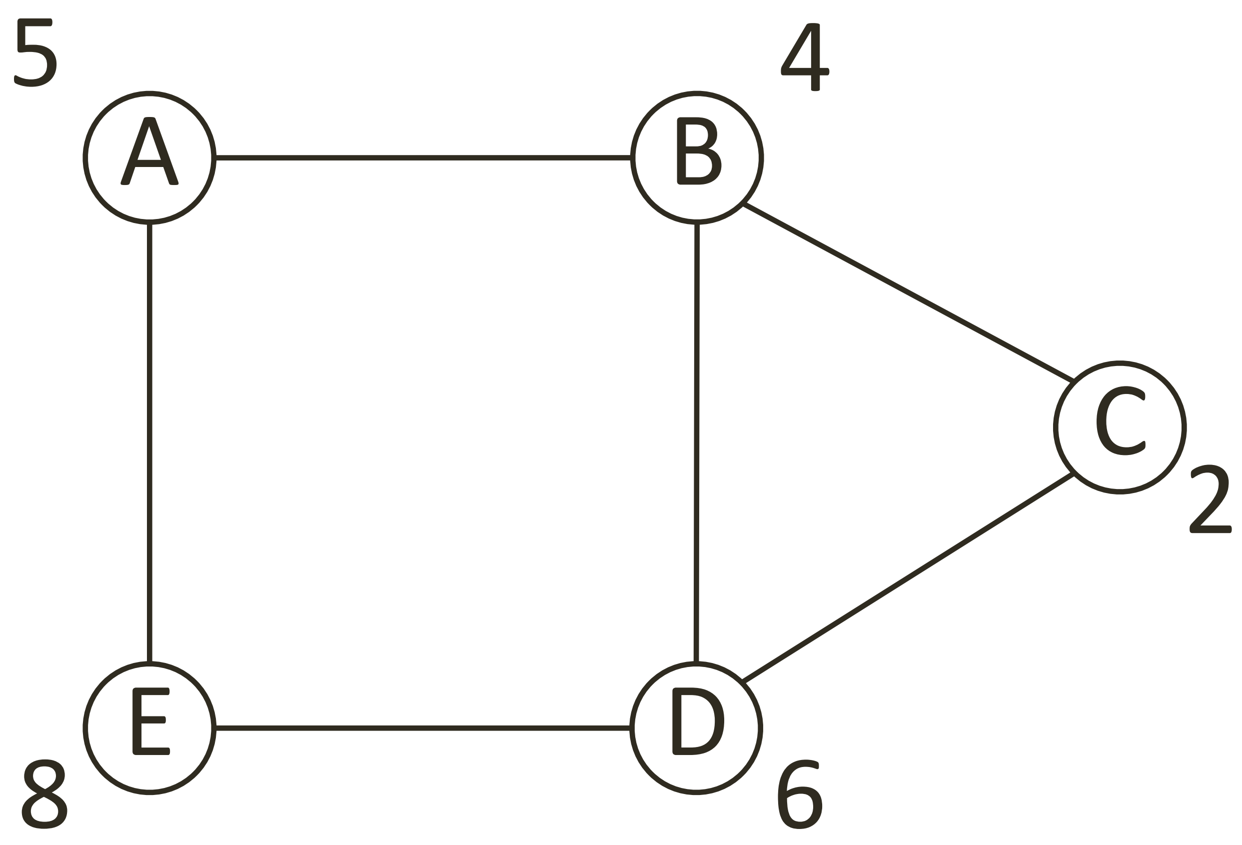




Maximum width = 3

Independent Set Problem: Relaxed DD

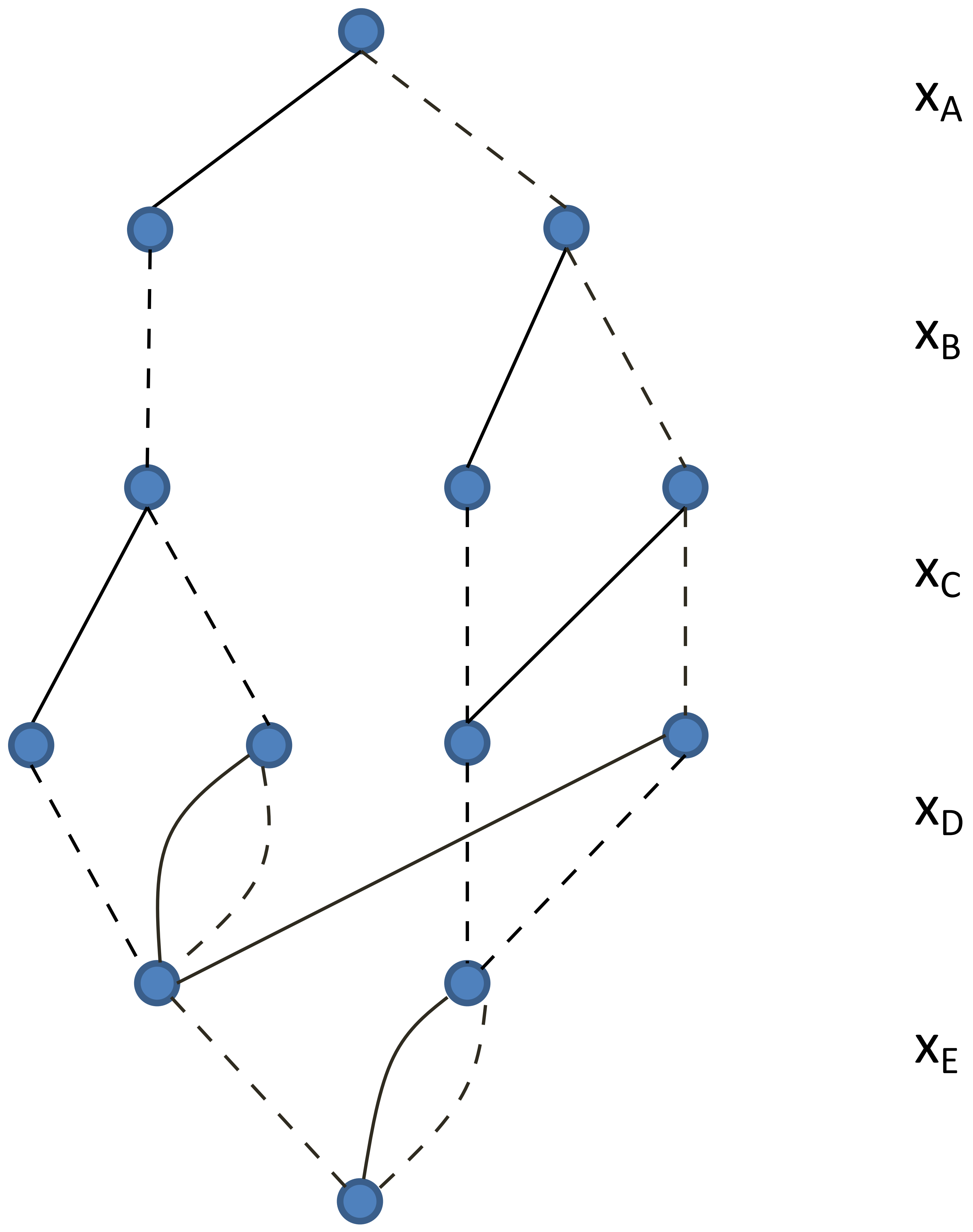




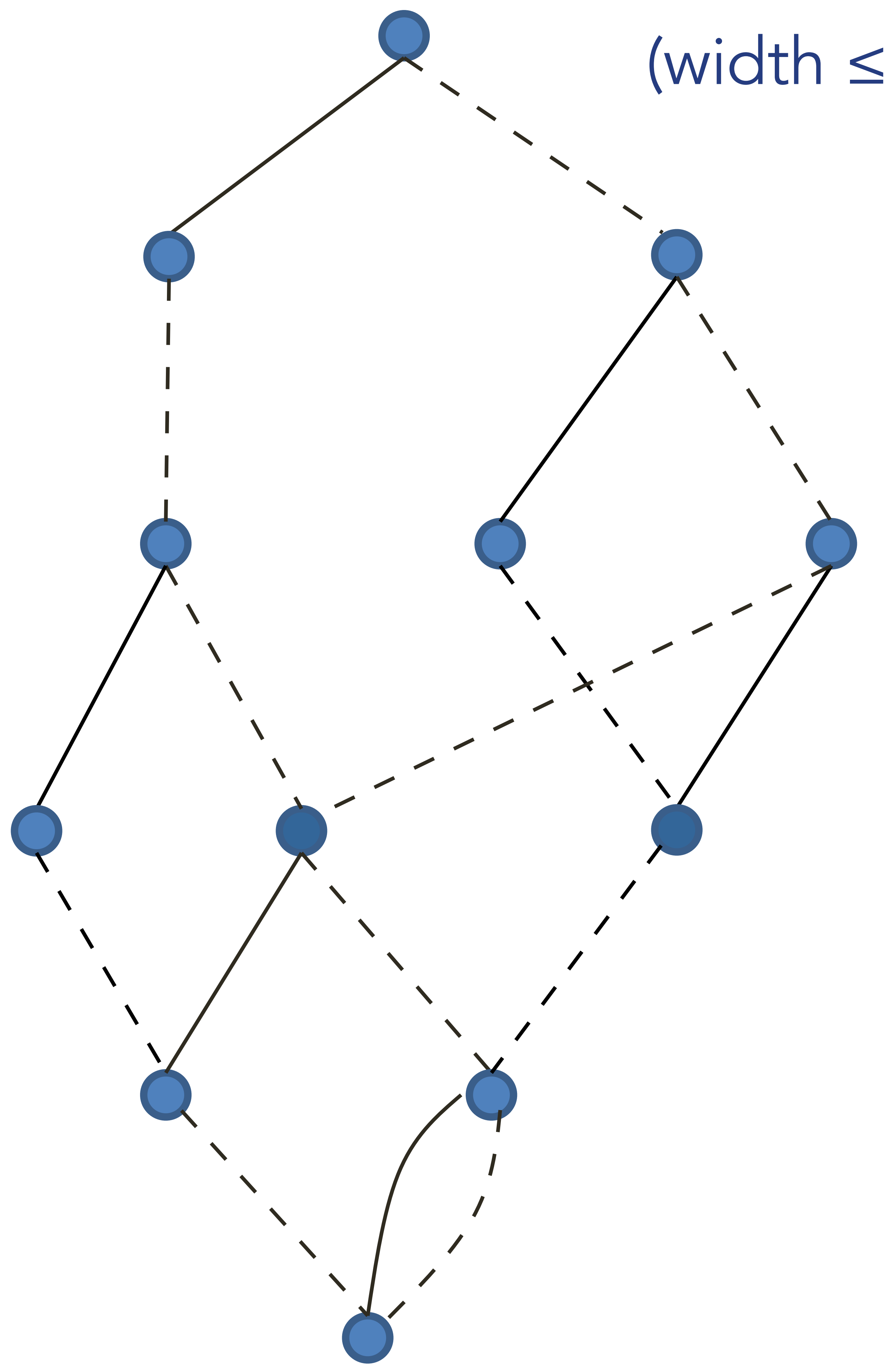
Maximum width = 3

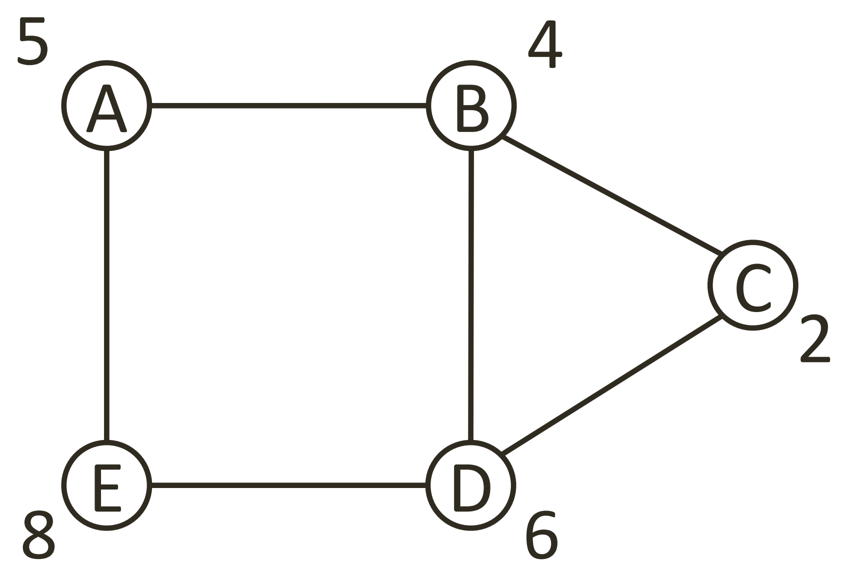
Exact vs. Relaxed Decision Diagrams

Exact



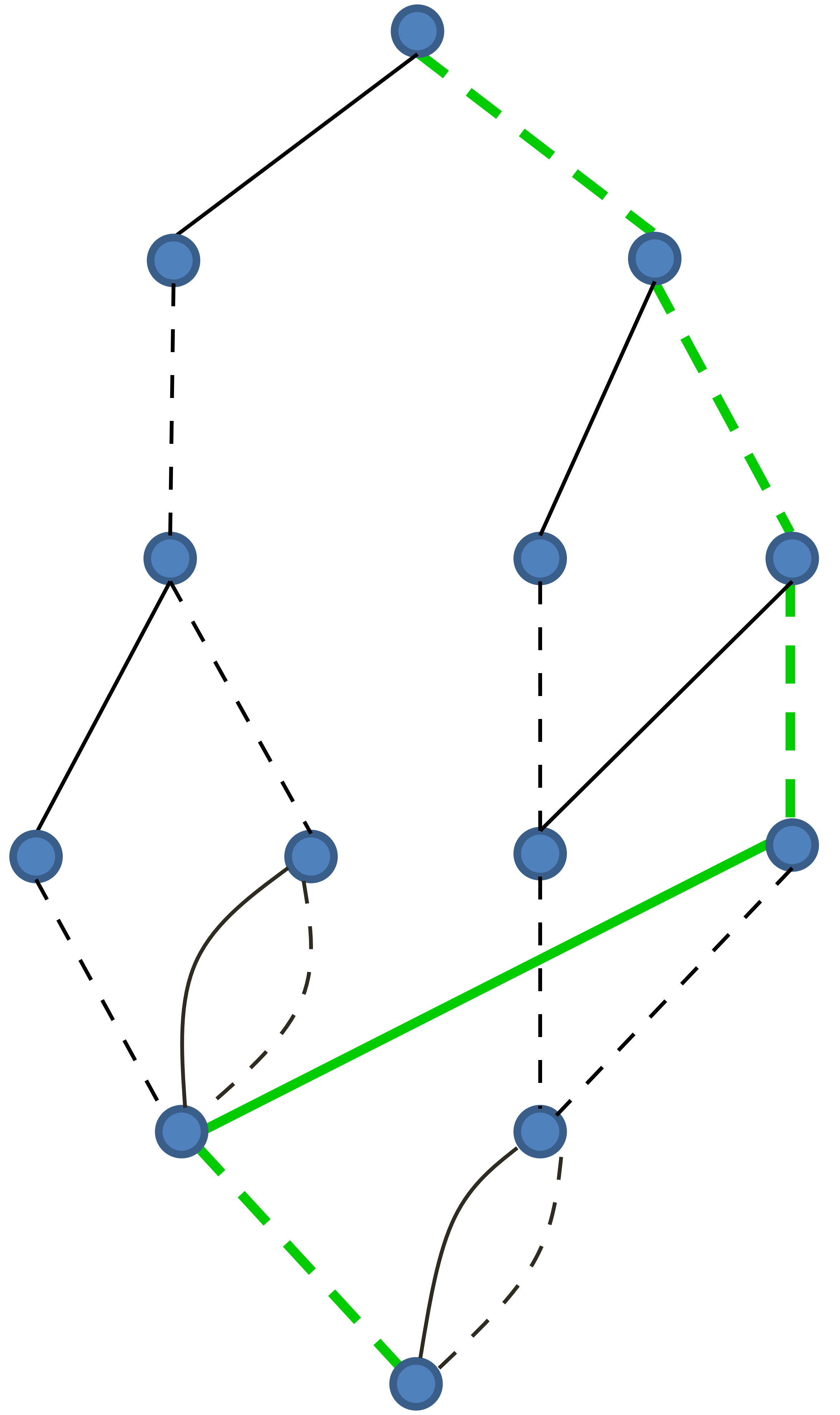
Relaxed
(width ≤ 3)





Exact vs. Relaxed Decision Diagrams

Exact



x_A

x_B

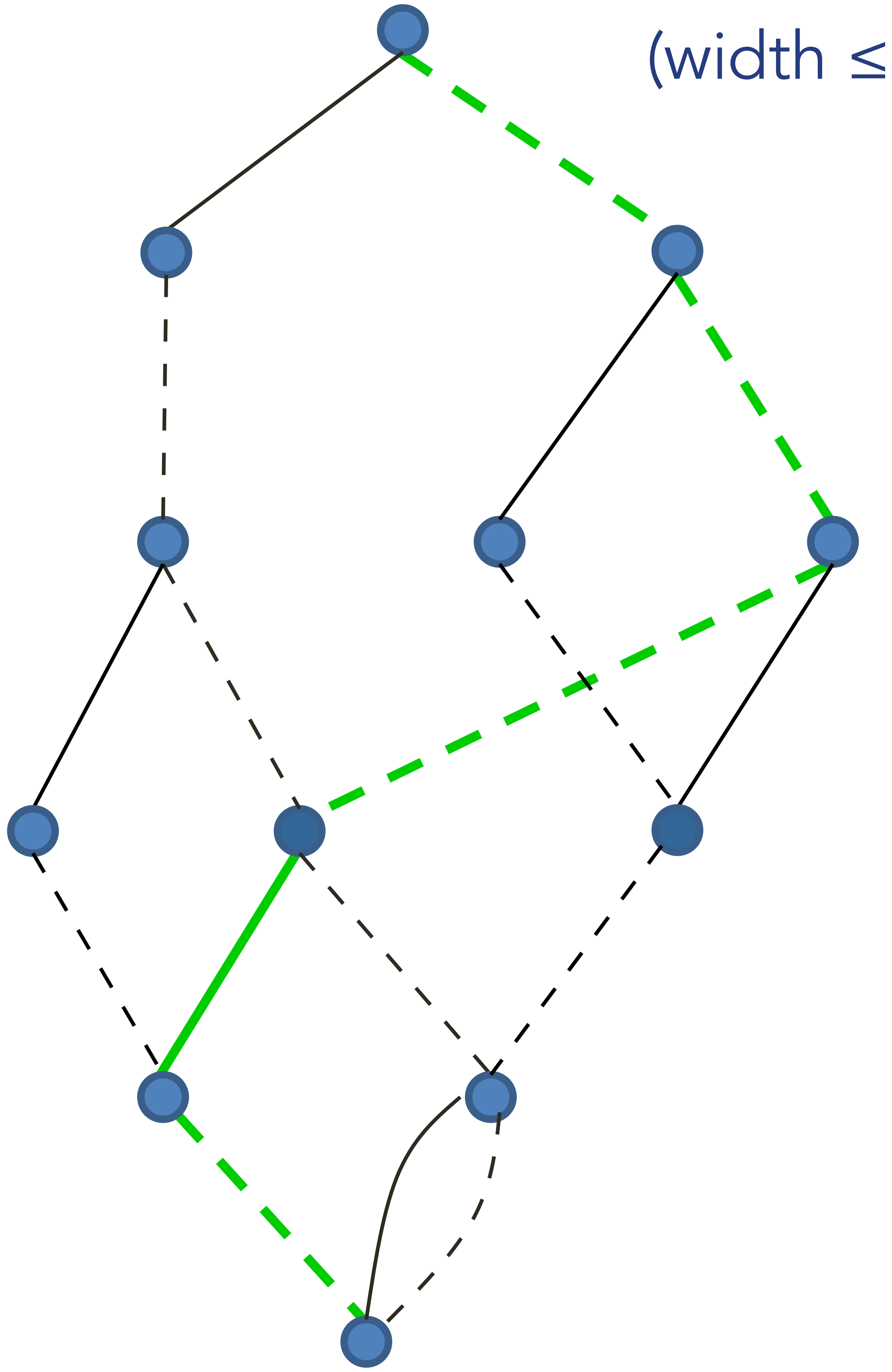
x_C

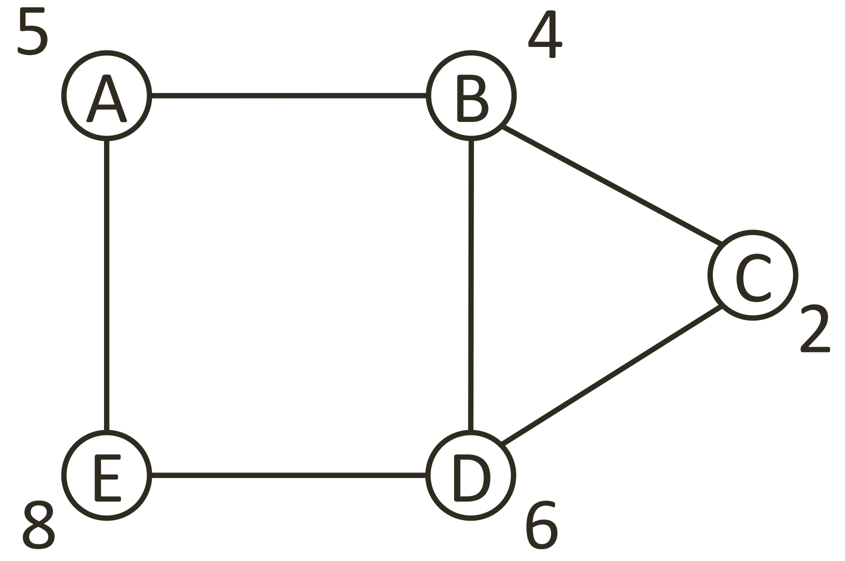
x_D

x_E

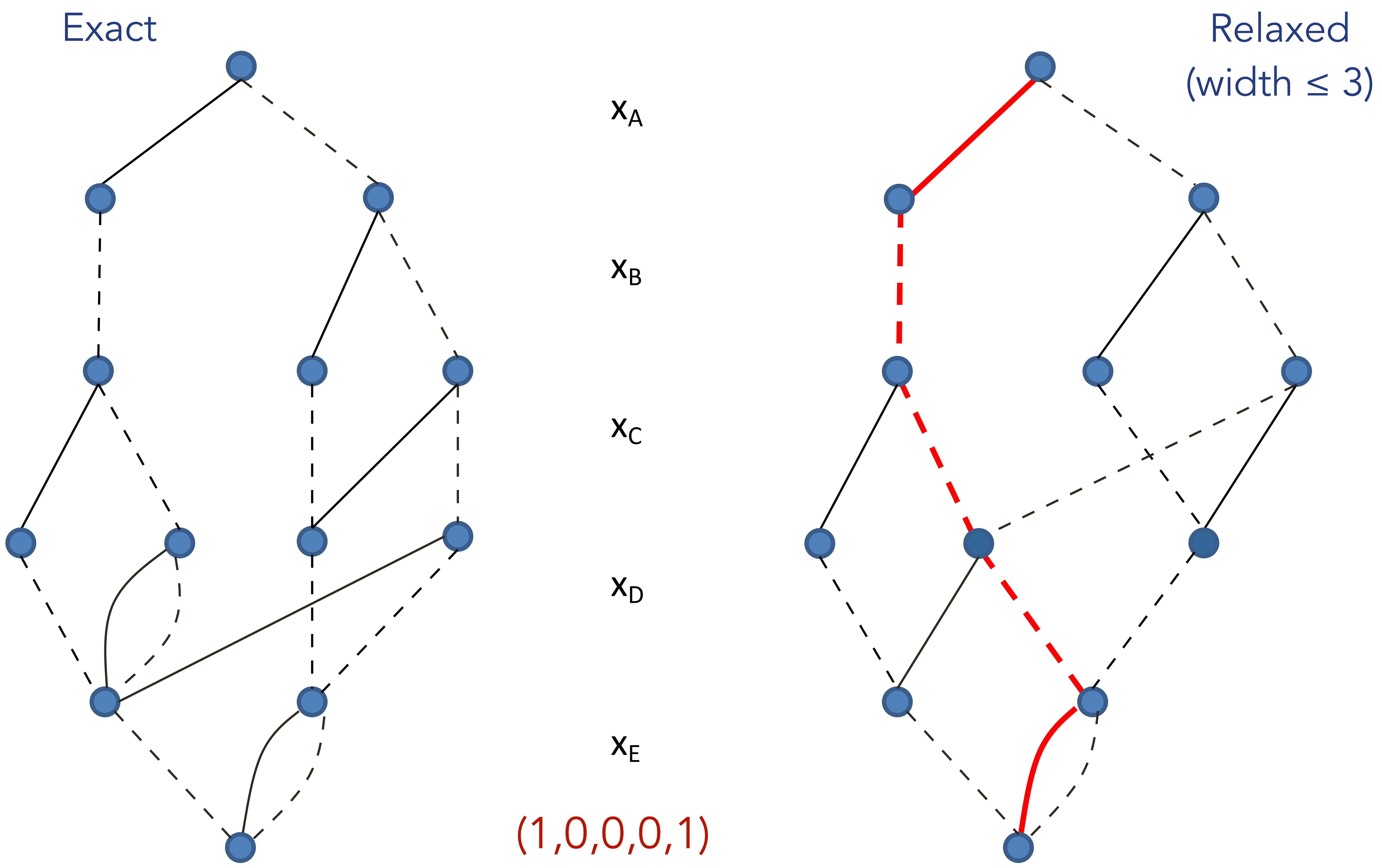
(0,0,0,1,0)

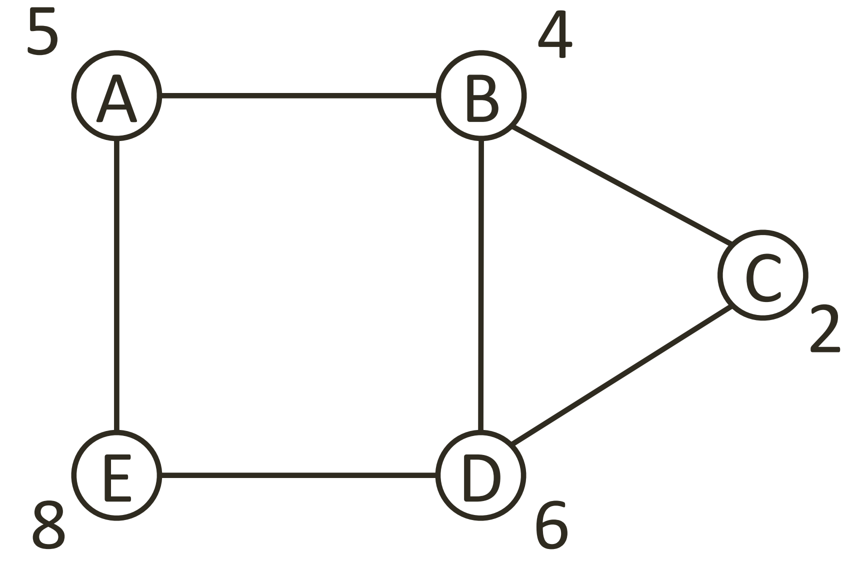
Relaxed
(width ≤ 3)





Exact vs. Relaxed Decision Diagrams

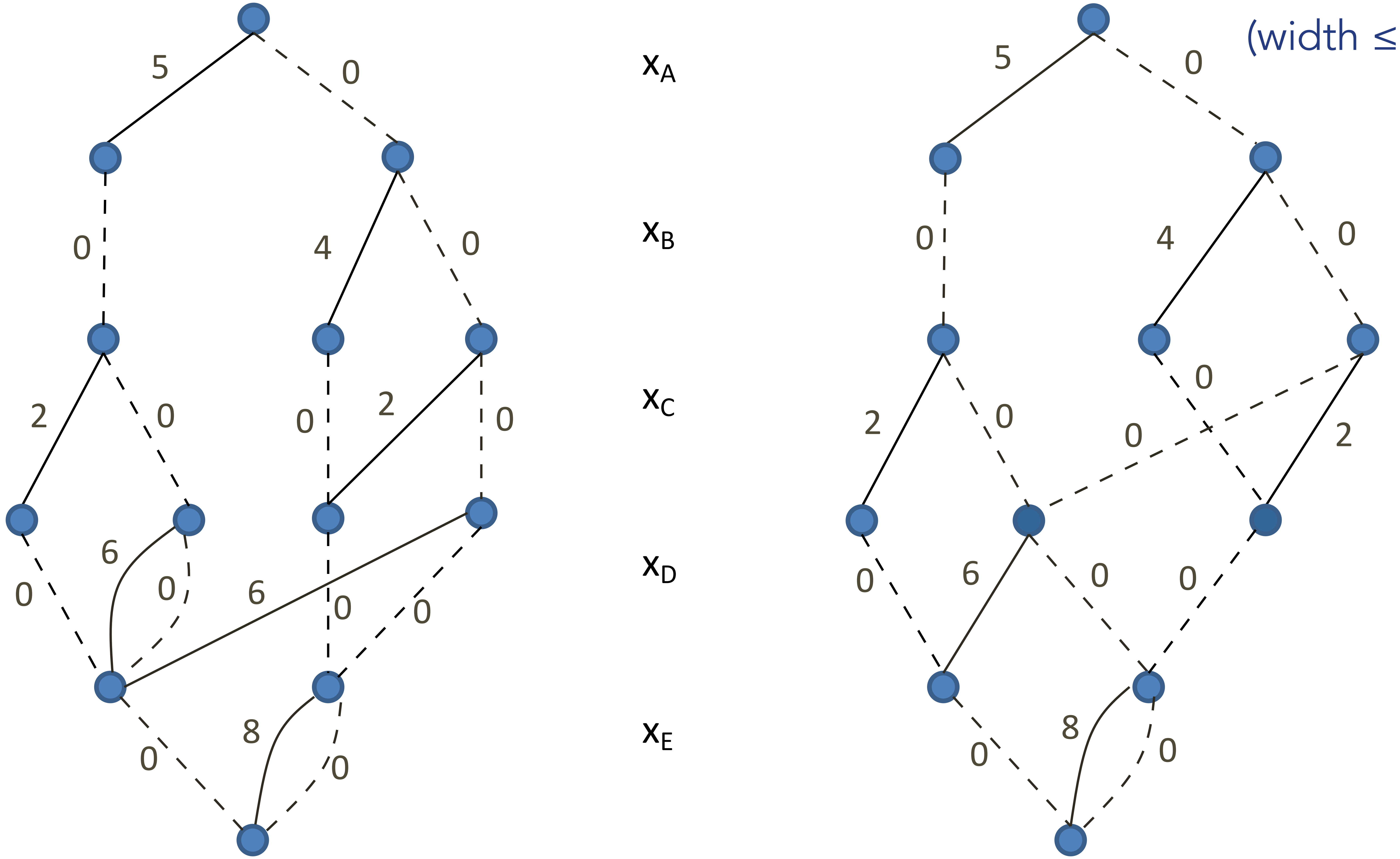


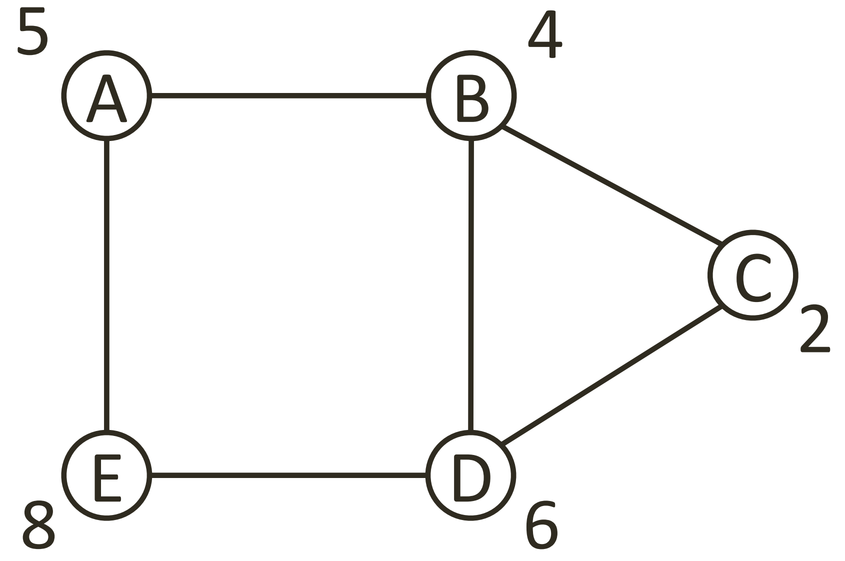


Exact vs. Relaxed Decision Diagrams

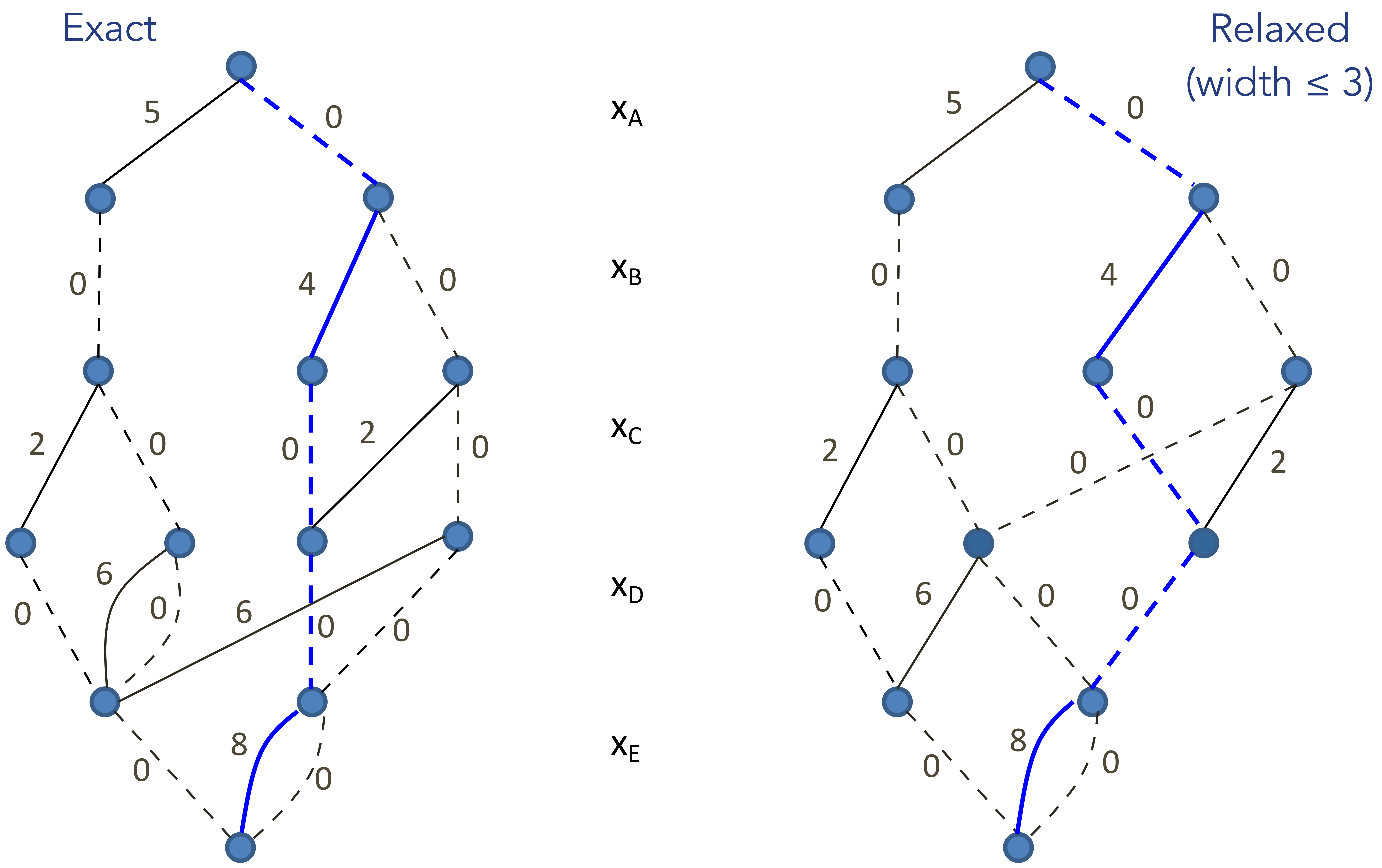
Exact

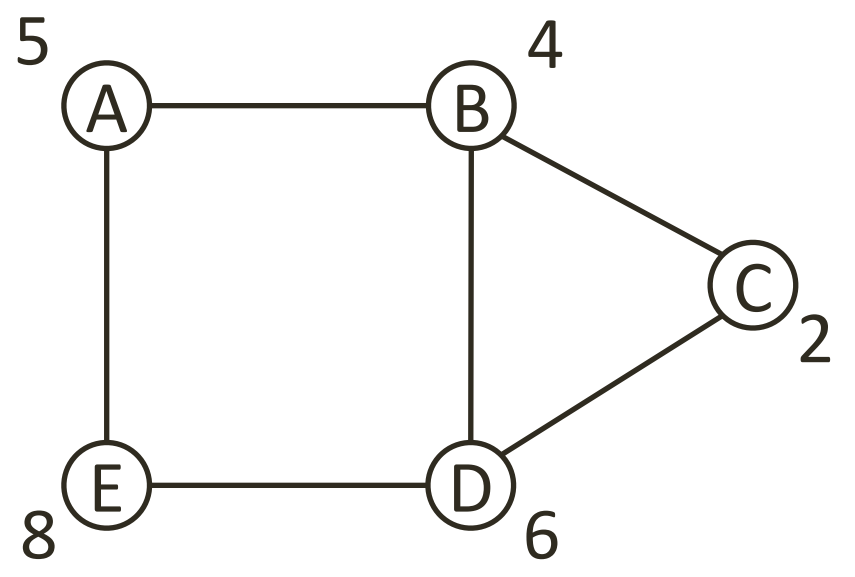
Relaxed
(width ≤ 3)





Exact vs. Relaxed Decision Diagrams

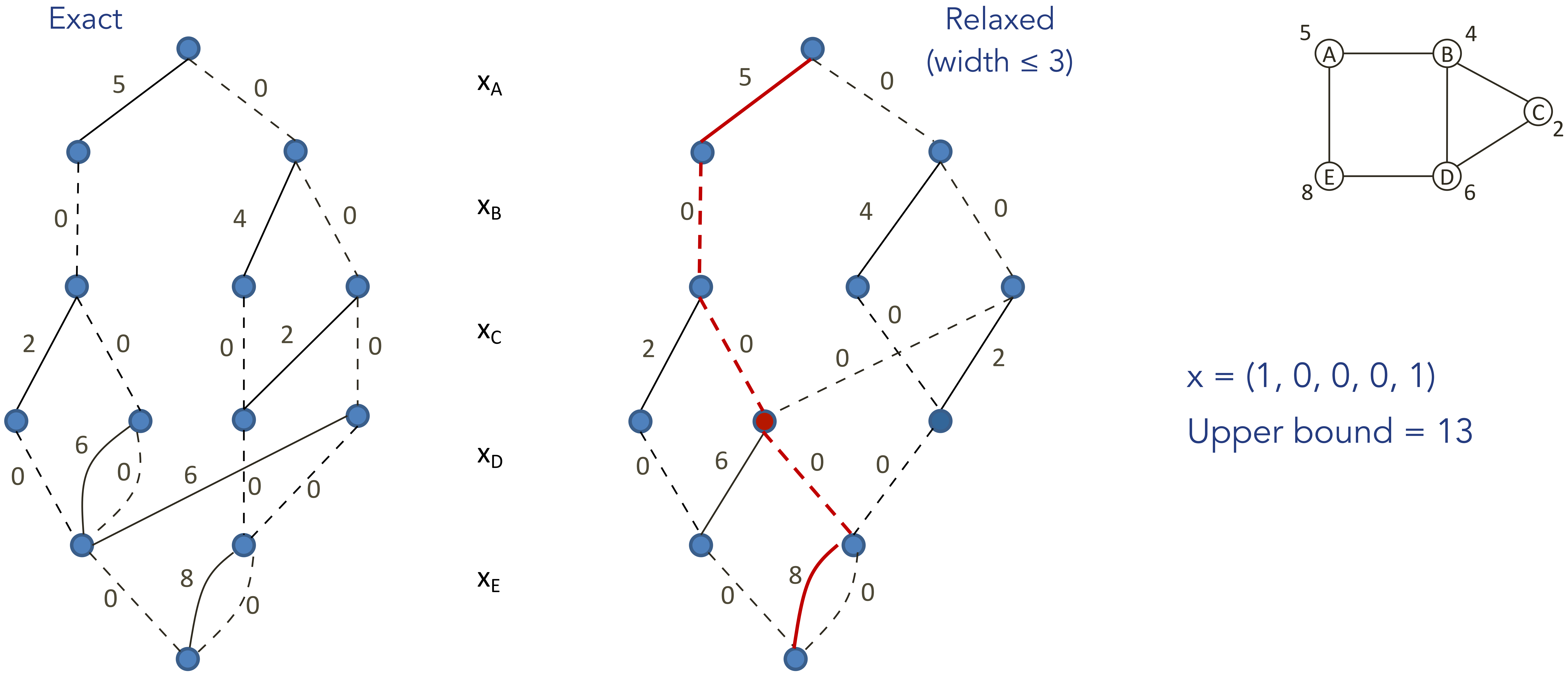




x = (0, 1, 0, 0, 1)

Solution value = 12

Exact vs. Relaxed Decision Diagrams

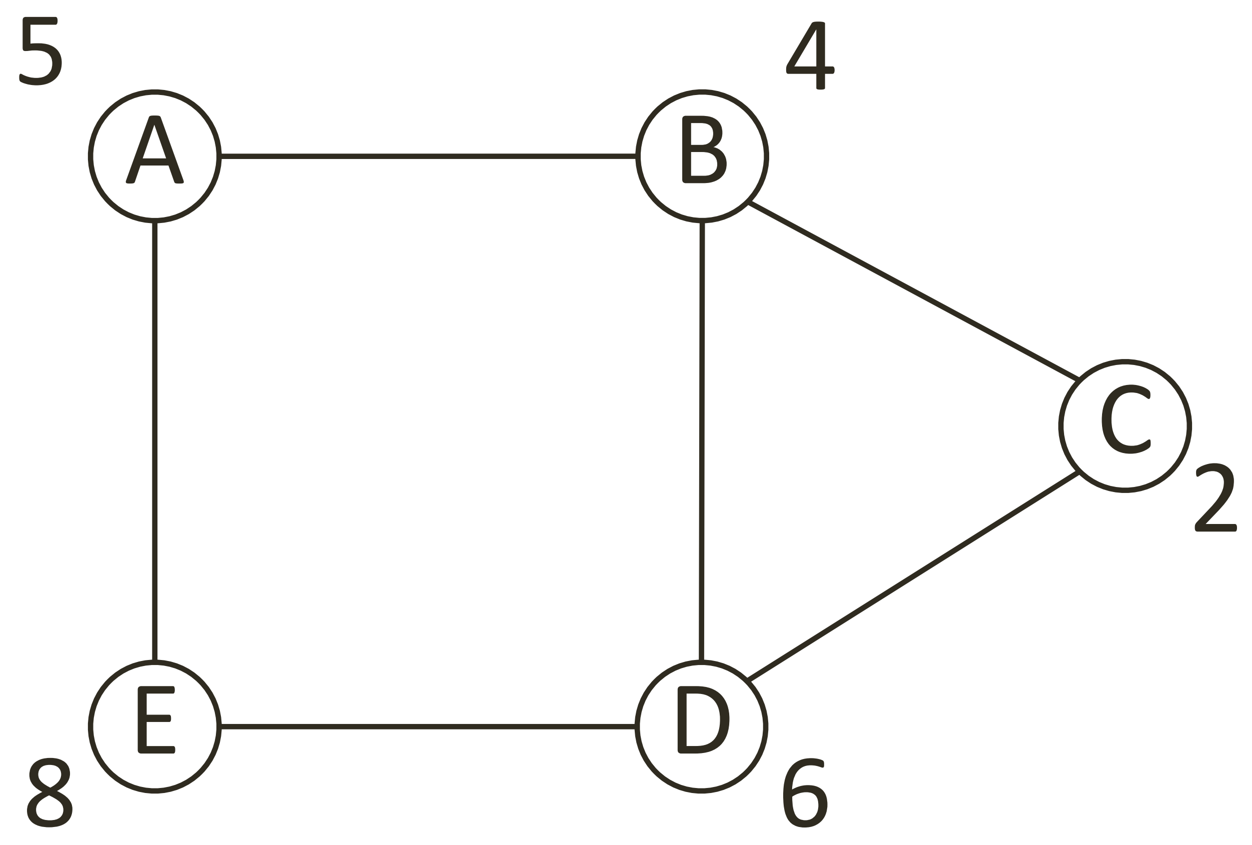


$x = (1, 0, 0, 0, 1)$

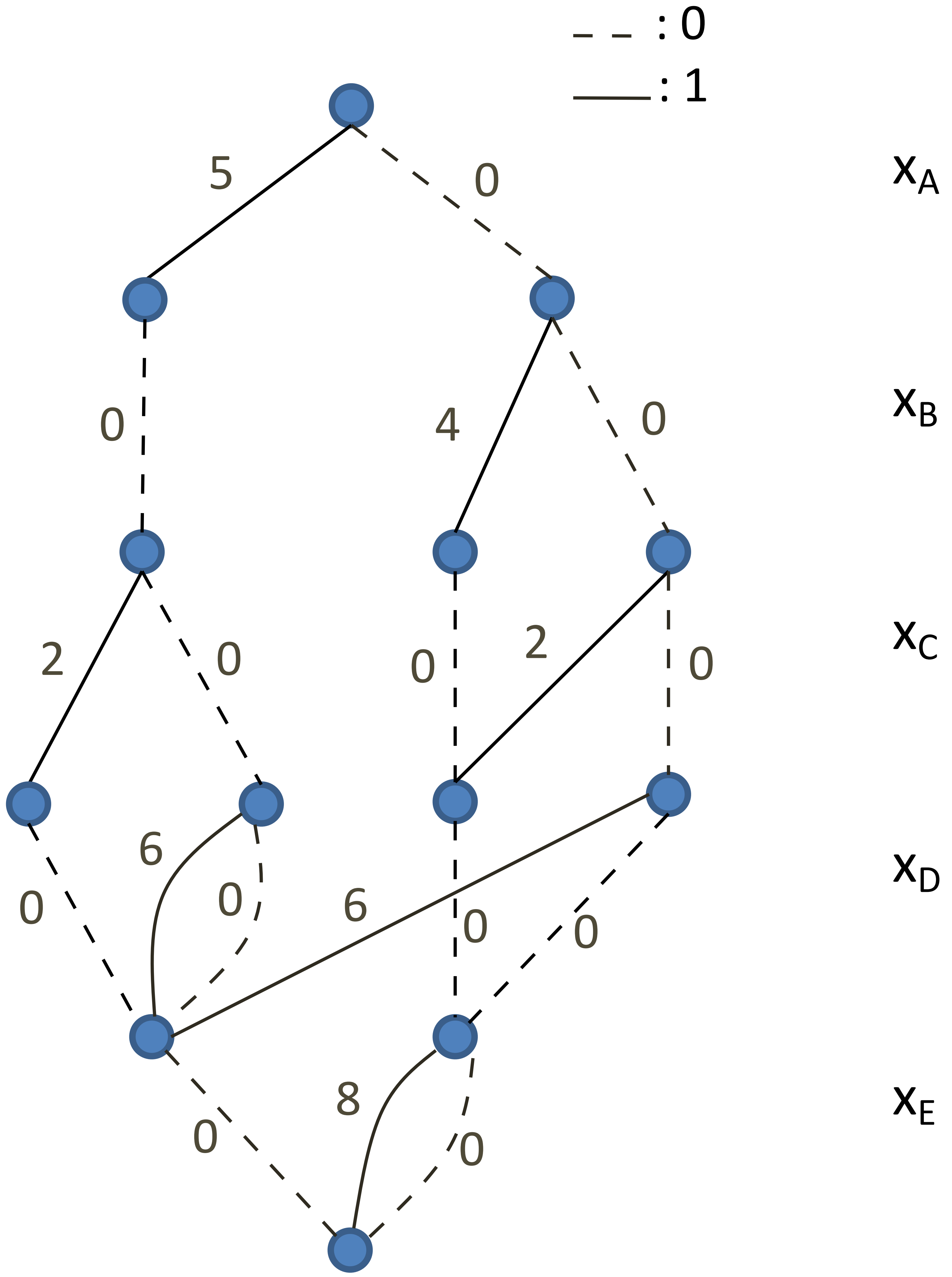
Upper bound = 13

Restricted Decision Diagrams

- Under-approximation of the feasible set



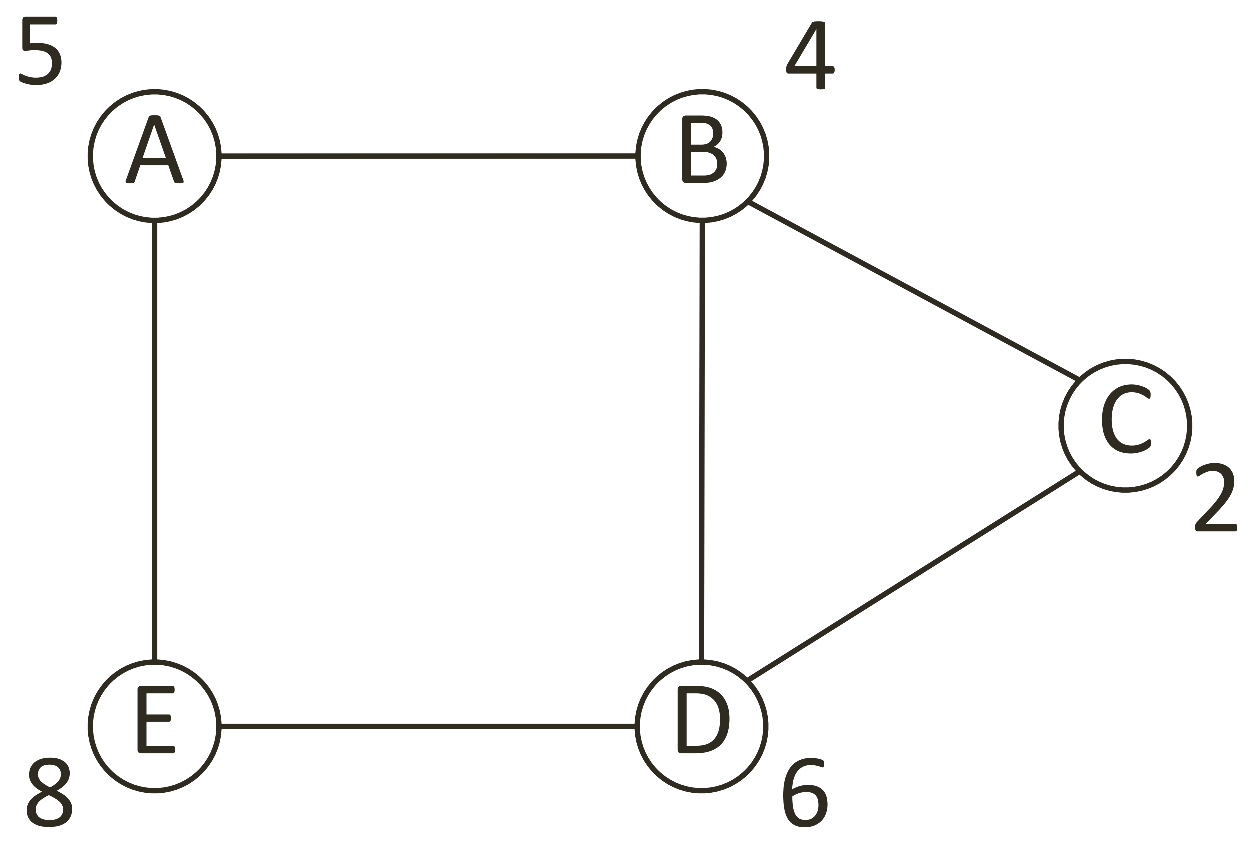
Maximum width = 3



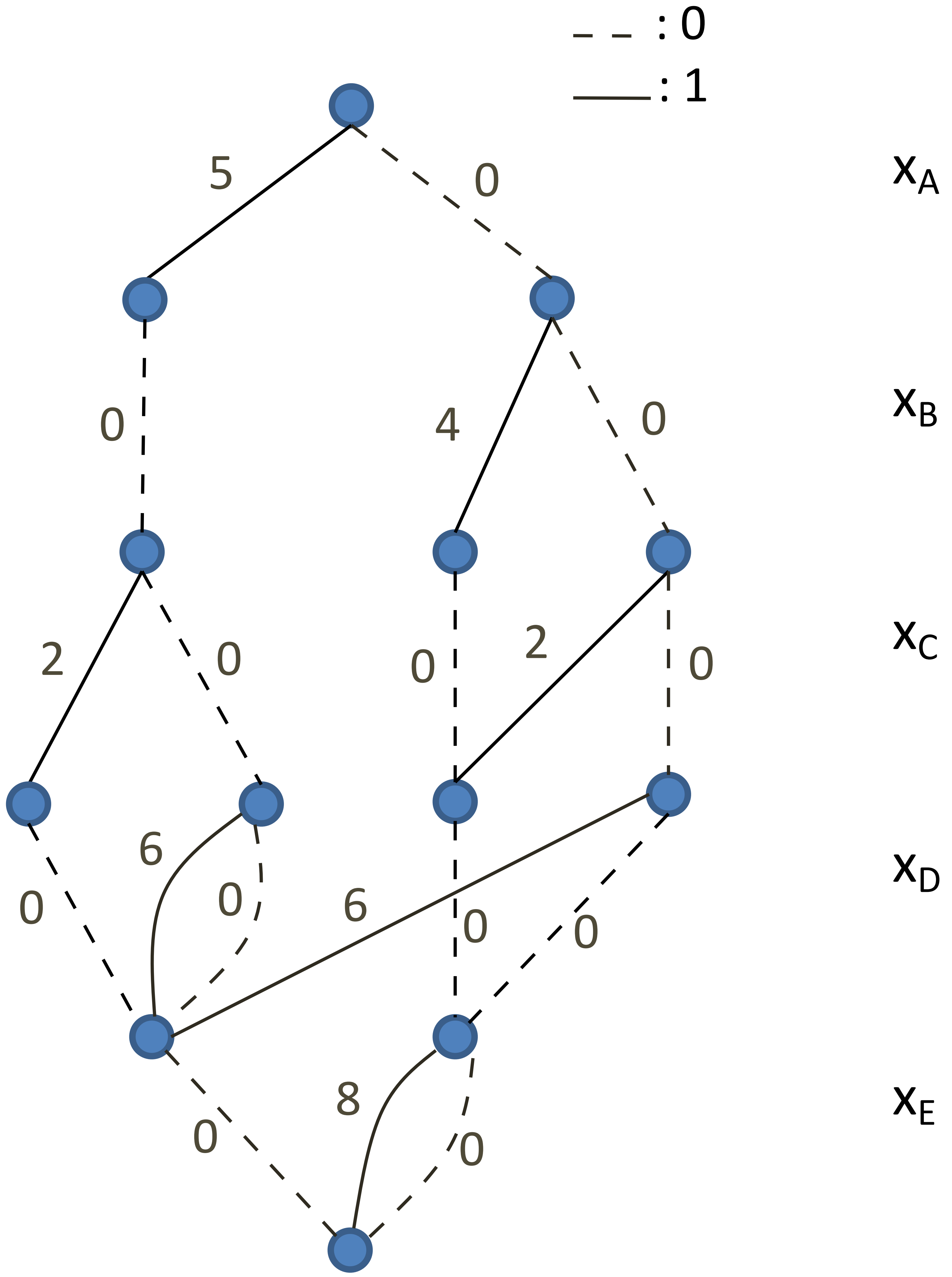
[Bergman, Cire, vH, Yunes, J Heur. 2014]

Restricted Decision Diagrams

- Under-approximation of the feasible set



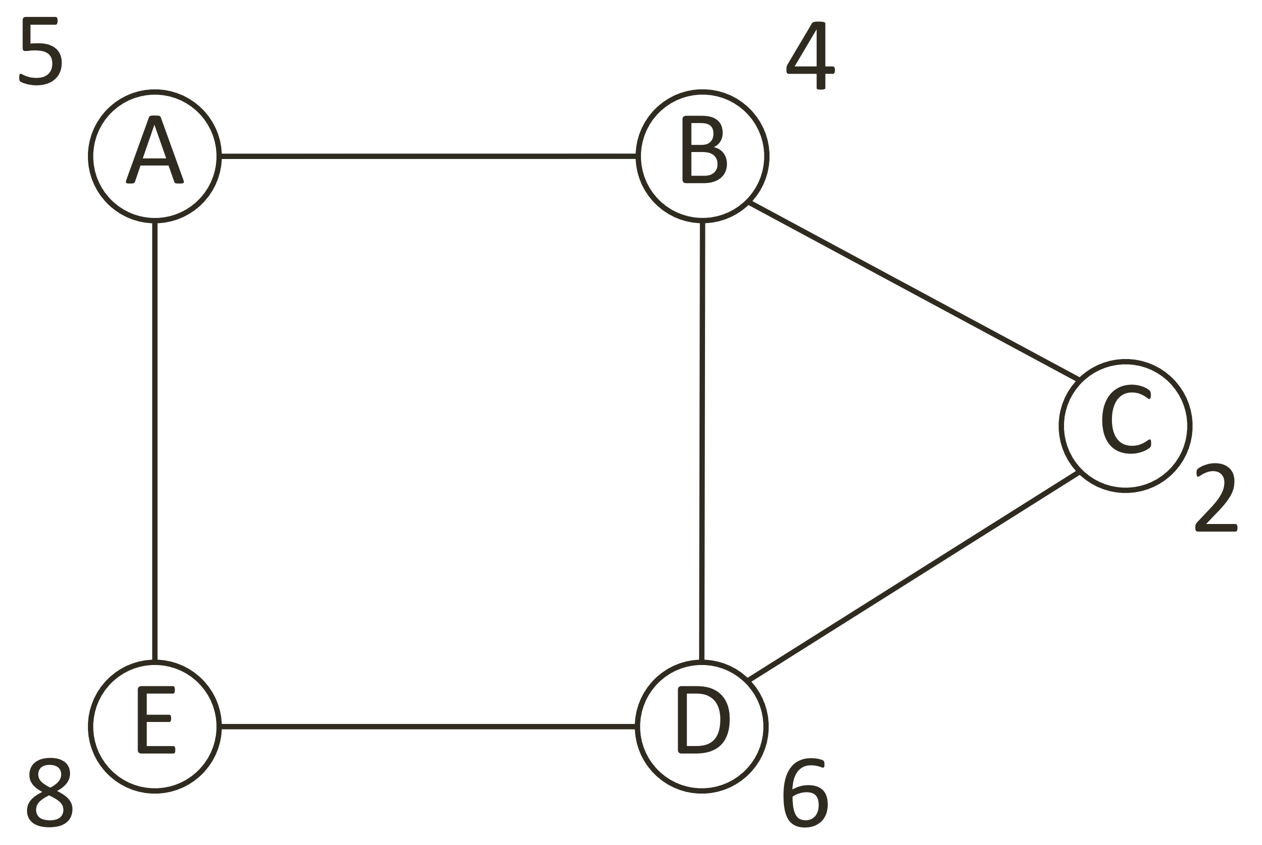
Maximum width = 3



[Bergman, Cire, vH, Yunes, J Heur. 2014]

Restricted Decision Diagrams

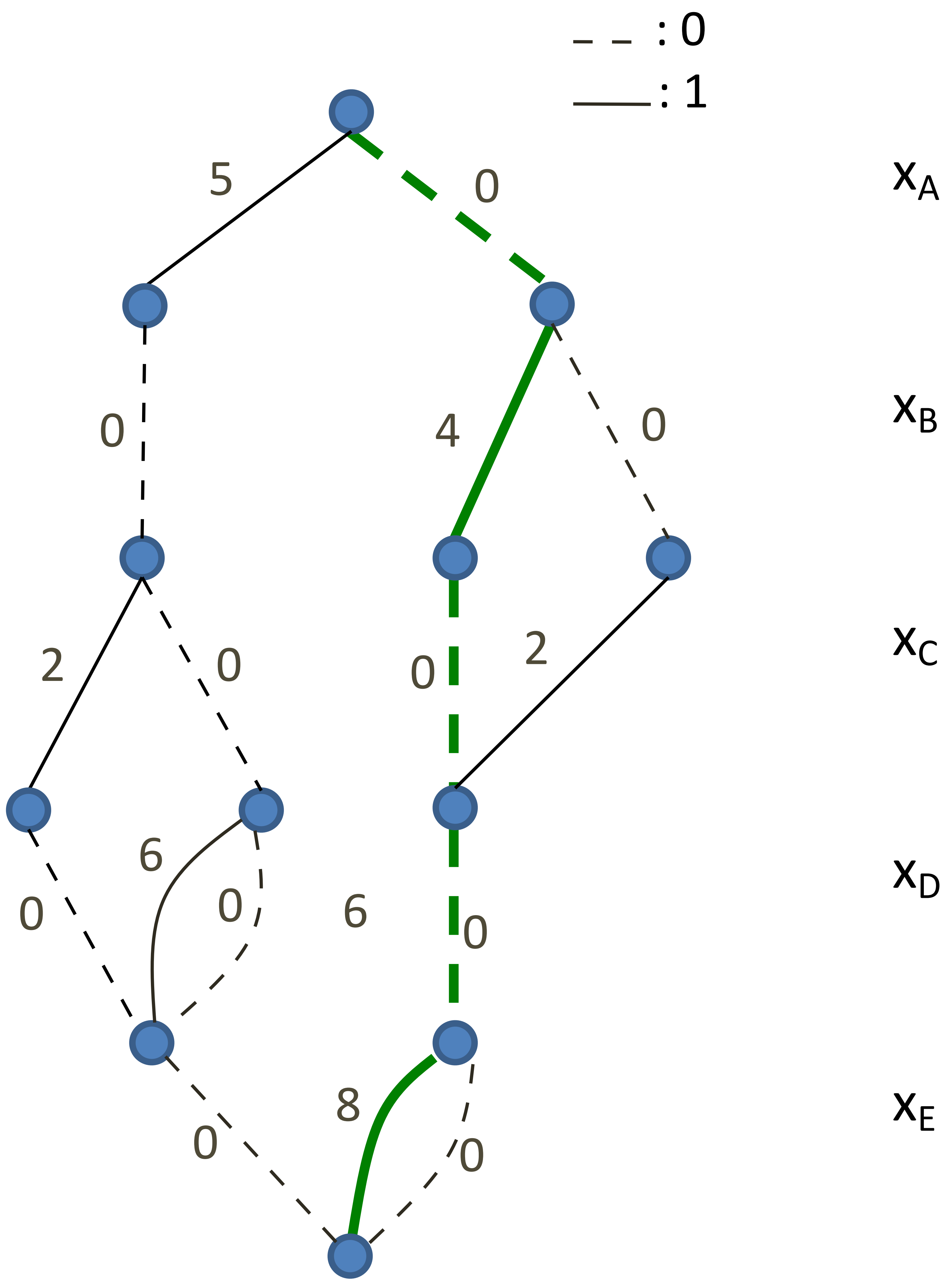
- Under-approximation of the feasible set



Maximum width = 3

$x = (0, 1, 0, 0, 1)$

Lower bound = 12



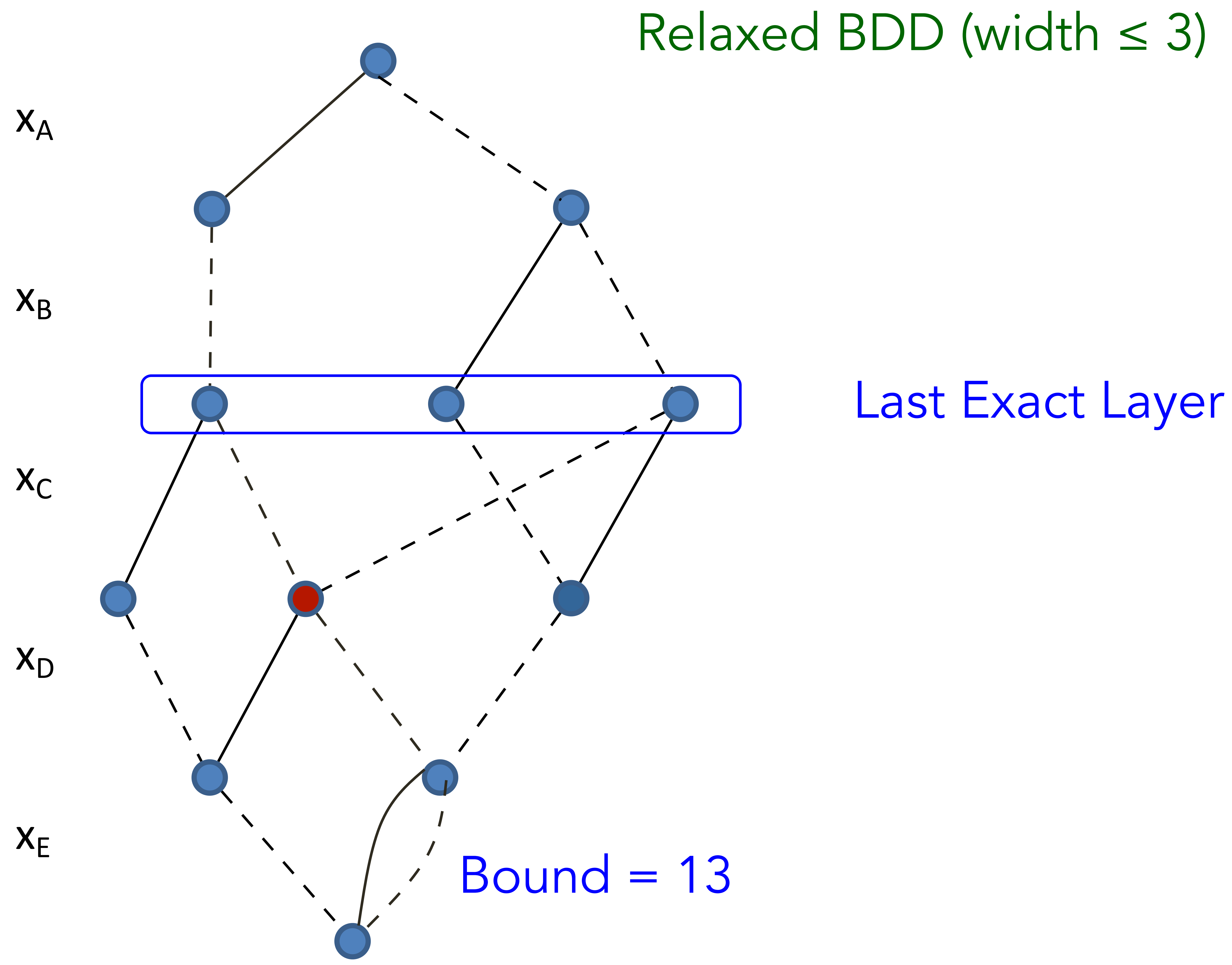
[Bergman, Cire, vH, Yunes, J Heur. 2014]

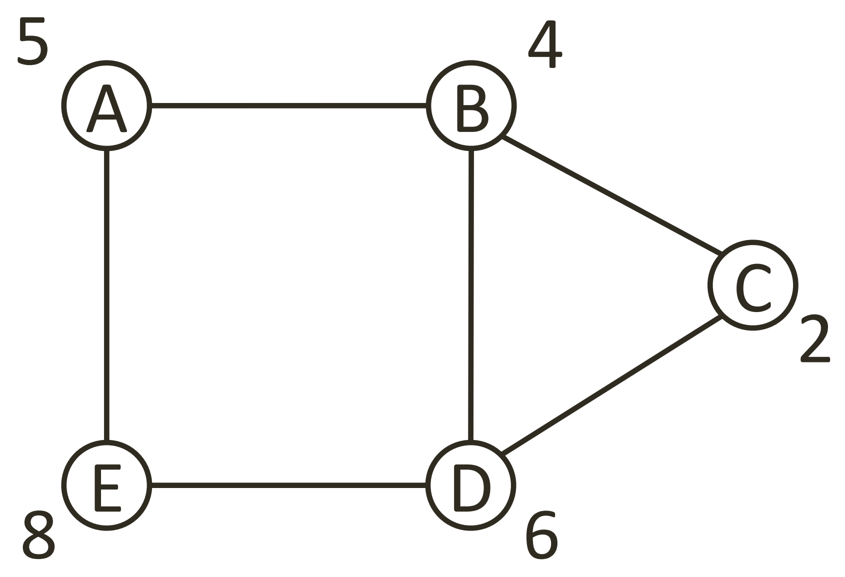
Exact Search Method

- Branch-and-bound scheme based on decision diagrams
  - Dual bounds: Relaxed decision diagrams
  - Primal bounds: Restricted decision diagrams
  - Branching is done on the *nodes* of the diagram

[Bergman, Cire, vH, Hooker, IJOC 2016]

Branch and Bound



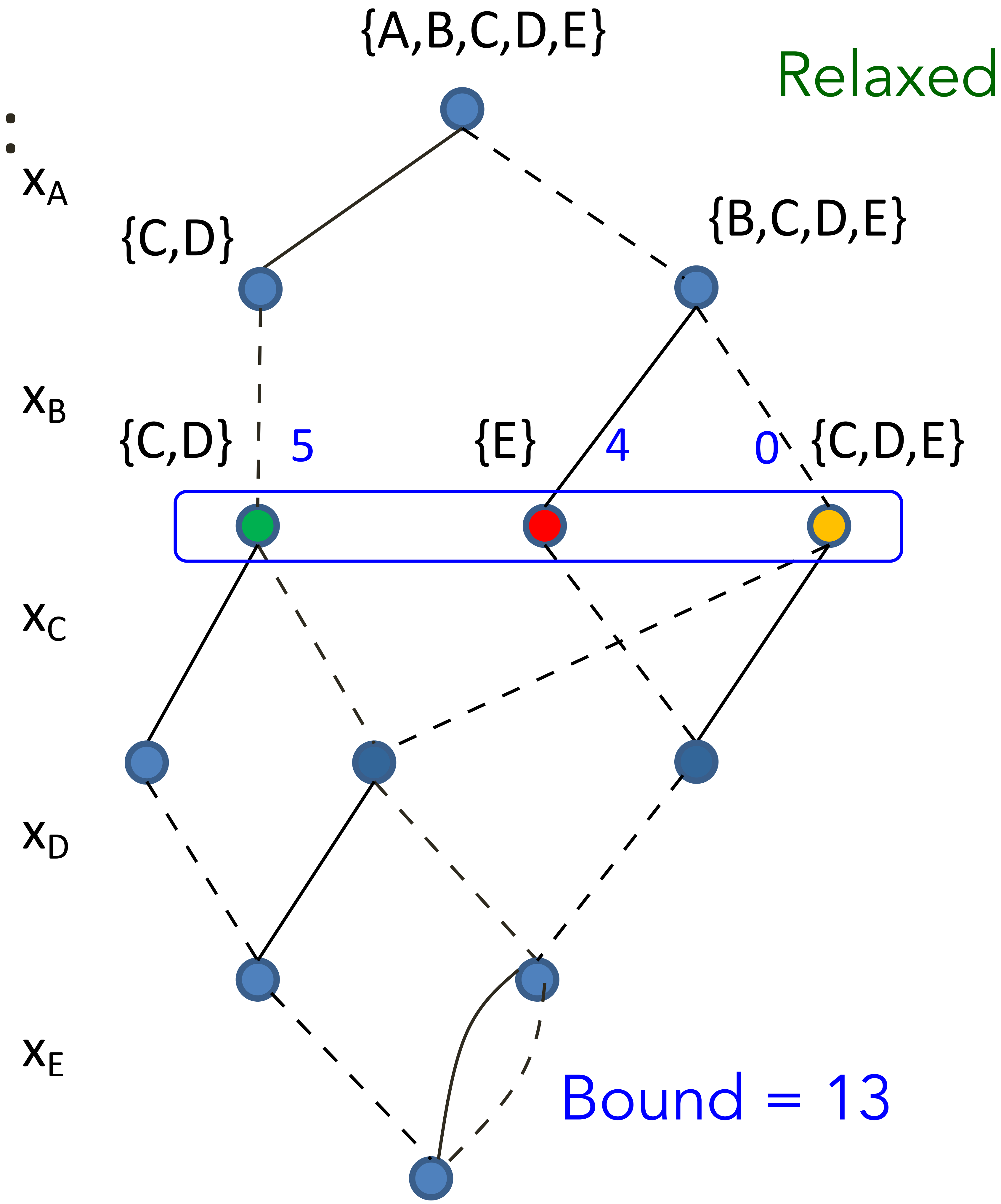


Node Queue

Relaxed BDD (width ≤ 3)

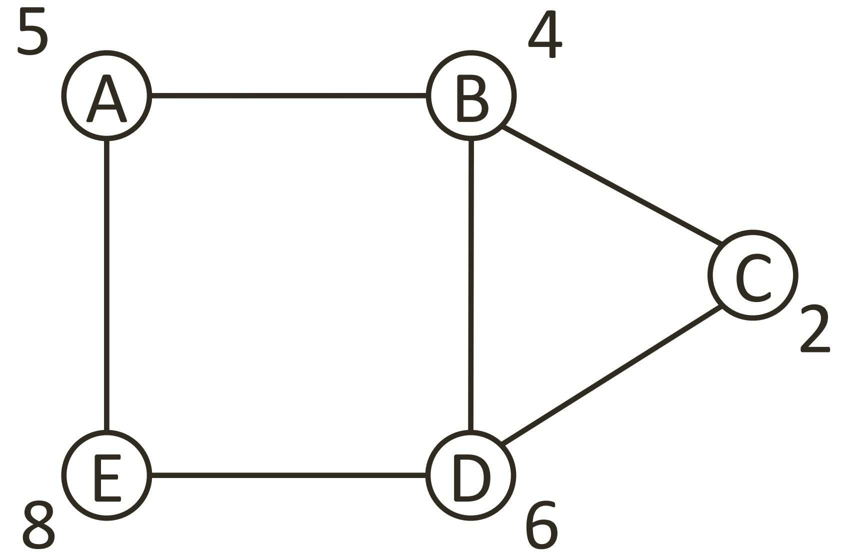
Upper bound = 13

Q:

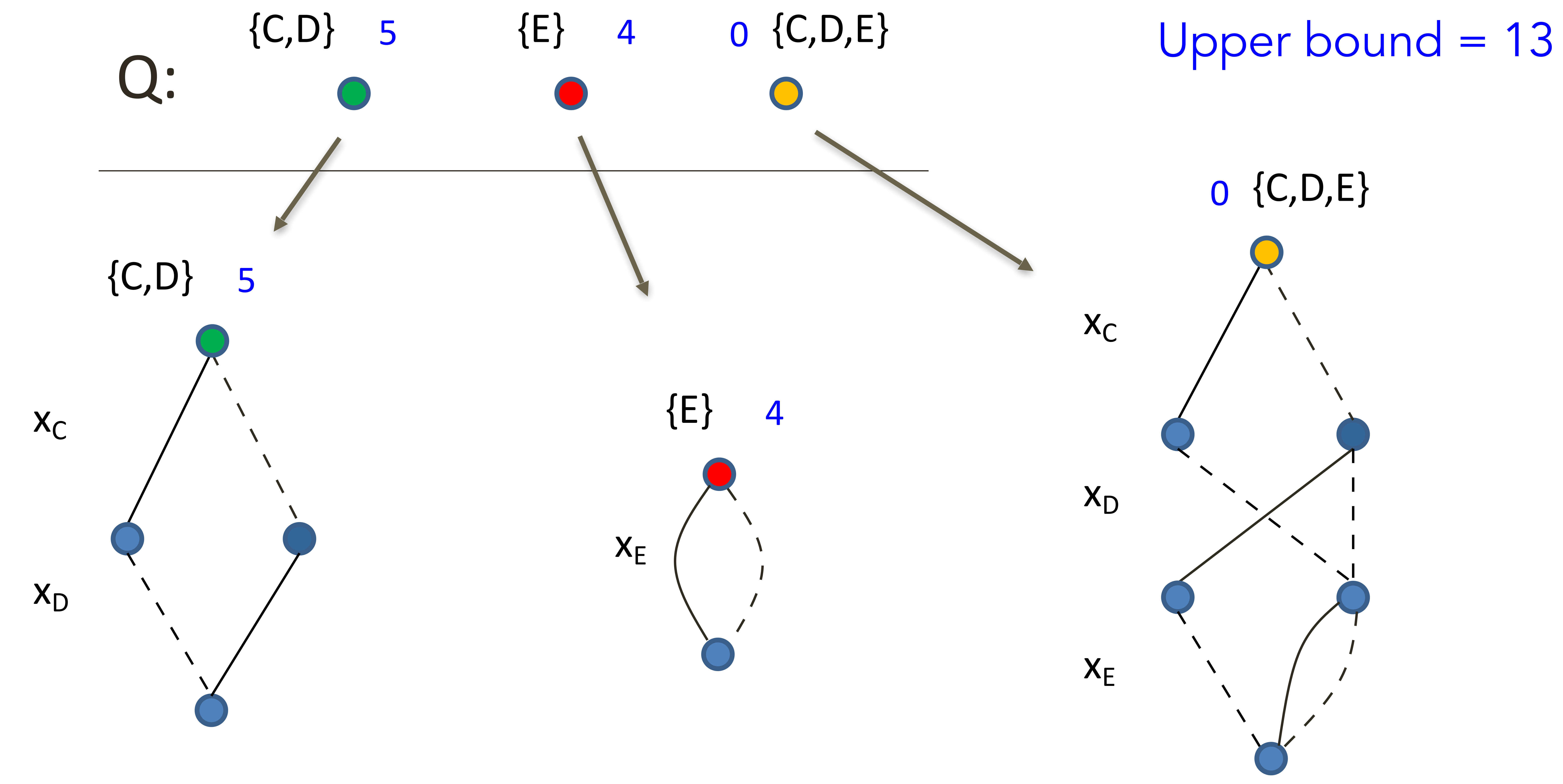


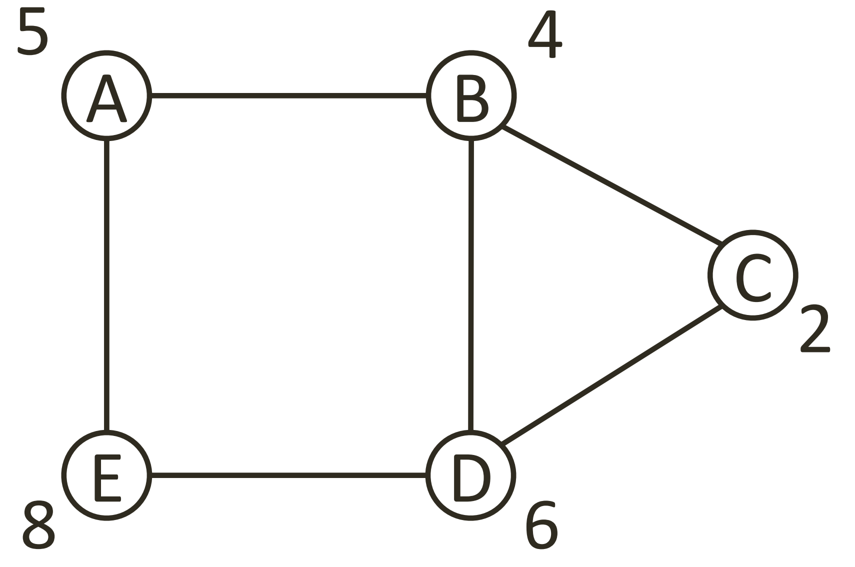
Last Exact Layer

Bound = 13



Node Queue





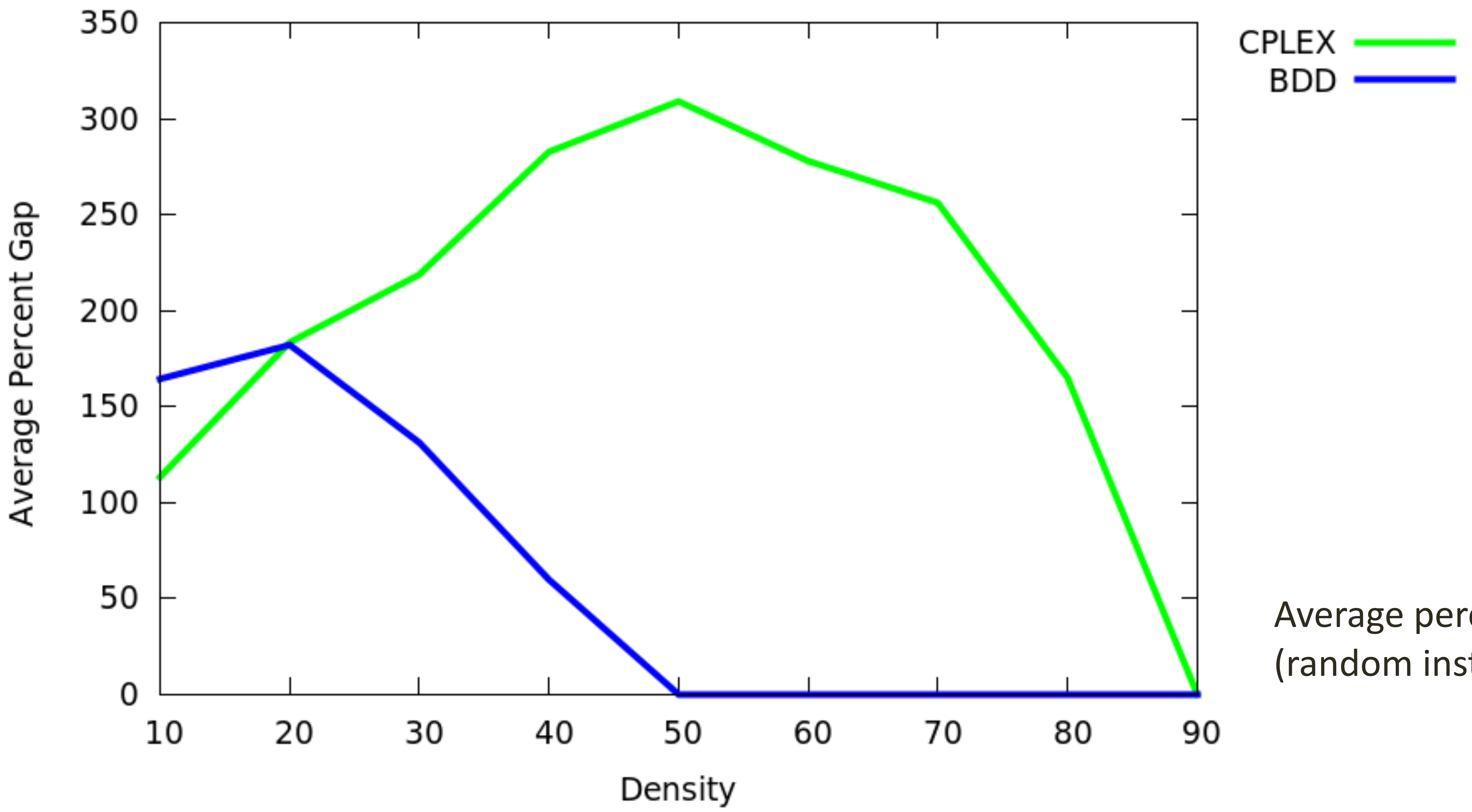
Exact solution: 11

Exact solution: 12

Exact solution: 10

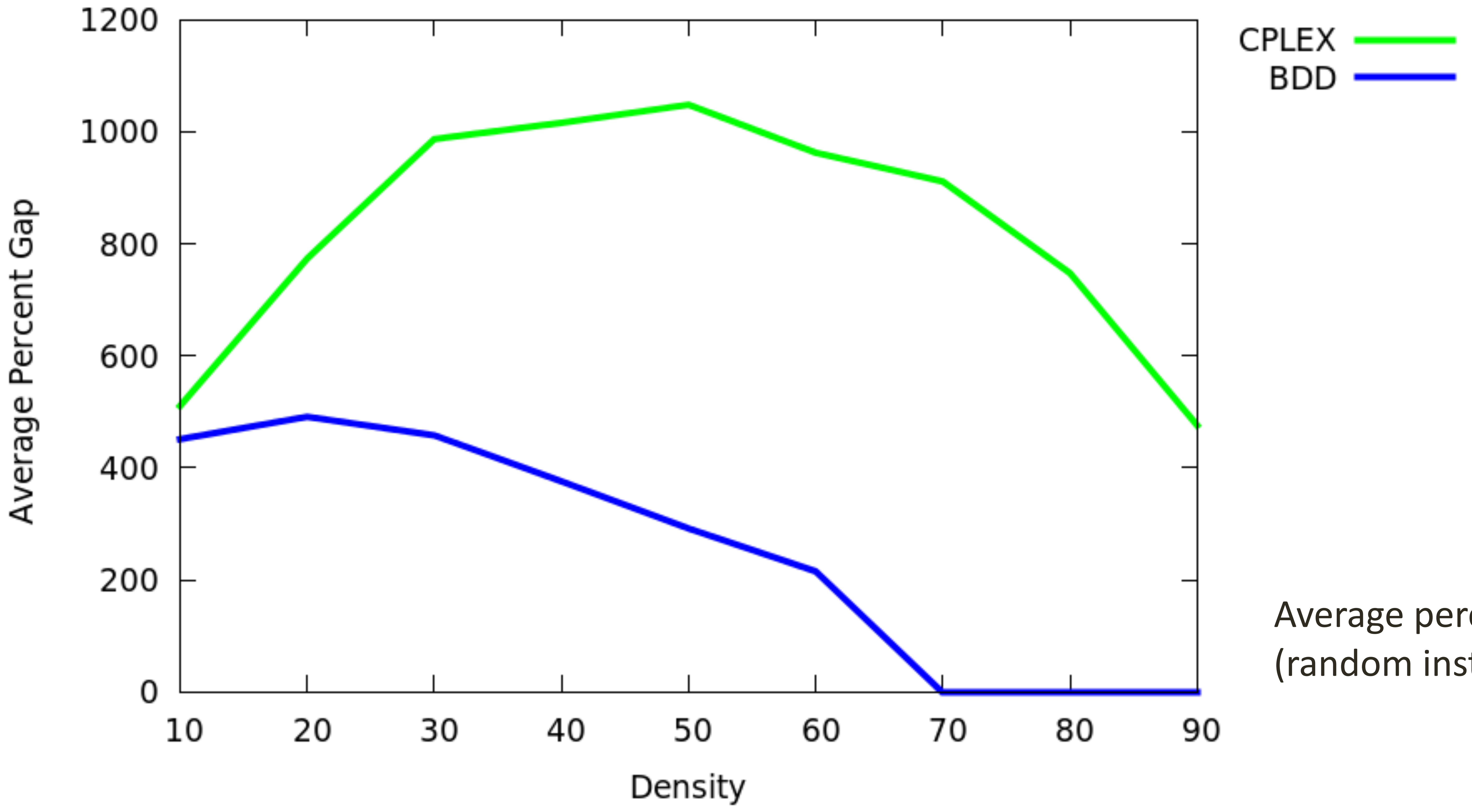
Optimal solution: 12

Maximum Independent Set: 500 variables



Average percent optimality gap (random instances) with 1h time limit

Maximum Independent Set: 1500 variables



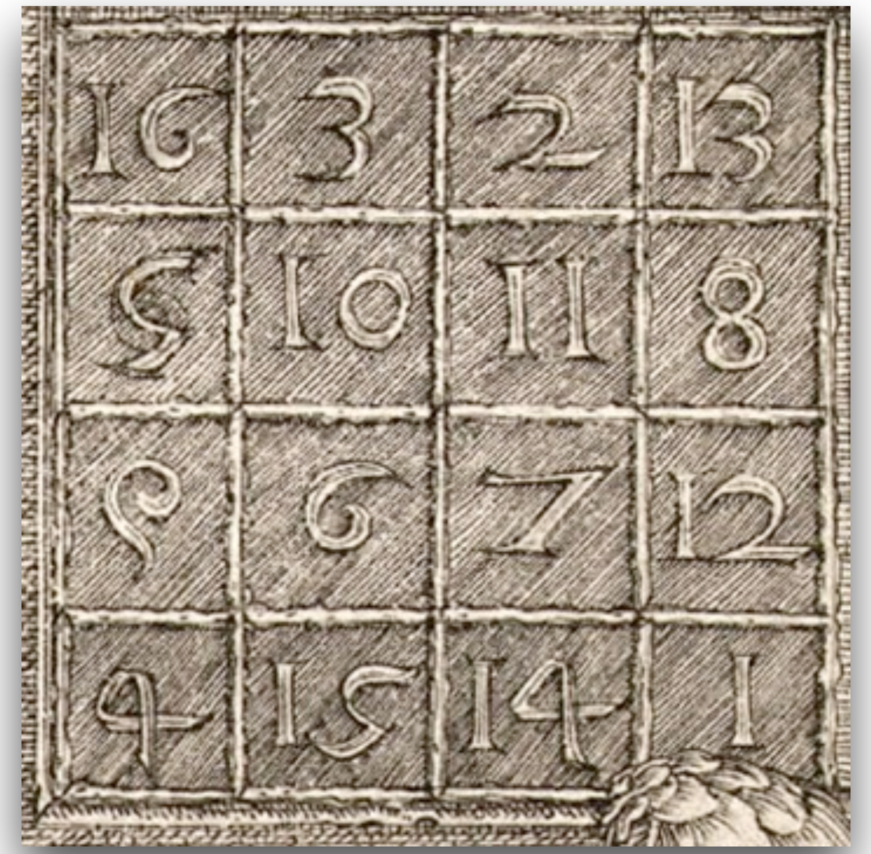
Average percent optimality gap (random instances) with 1h time limit

Maximum Cut Problem: BiqMac vs BDD

	BiqMac		BDD		Best known (2015)	
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

(DALL-E, 2024)

Branch-and-Bound Solver



(Durer, 1514)

Constraint Programming

(Escher, 1961)

Integer Programming

(DALL-E, 2024)

Column Elimination

Constraint Programming = Propagate (+ Search)

- Constraint propagation algorithm for individual constraints
  - remove inconsistent values from variable domains
  - propagate updated domains to other constraints

$$\boxed{x_1 > x_2}$$

$$\boxed{x_1 + x_2 = x_3}$$

$$\boxed{\textit{alldifferent}(x_1, x_2, x_3, x_4)}$$

$$x_1 \in \{\cancel{1},2,\cancel{3}\}, x_2 \in \{1,\cancel{2},\cancel{3},\cancel{4}\}, x_3 \in \{\cancel{2},3\}, x_4 \in \{\cancel{2},\cancel{3},4\}$$

Constraint Programming = Propagate (+ Search)

- Constraint propagation algorithm for individual constraints
  - remove inconsistent values from variable domains
  - propagate updated domains to other constraints

$$x_1 > x_2$$

$$x_1 + x_2 = x_3$$

$$alldifferent(x_1, x_2, x_3, x_4)$$

$$x_1 \in \{2\}, x_2 \in \{1\}, x_3 \in \{3\}, x_4 \in \{4\}$$

Domain propagation has its limitations however

Propagate Decision Diagrams!

$$alldifferent(x_1, x_2, x_3, x_4)$$

$$x_1 + x_2 + x_3 \geq 9$$

$$x_i \in \{1,2,3,4\} \quad (i = 1,2,3,4)$$

Propagate decision diagrams

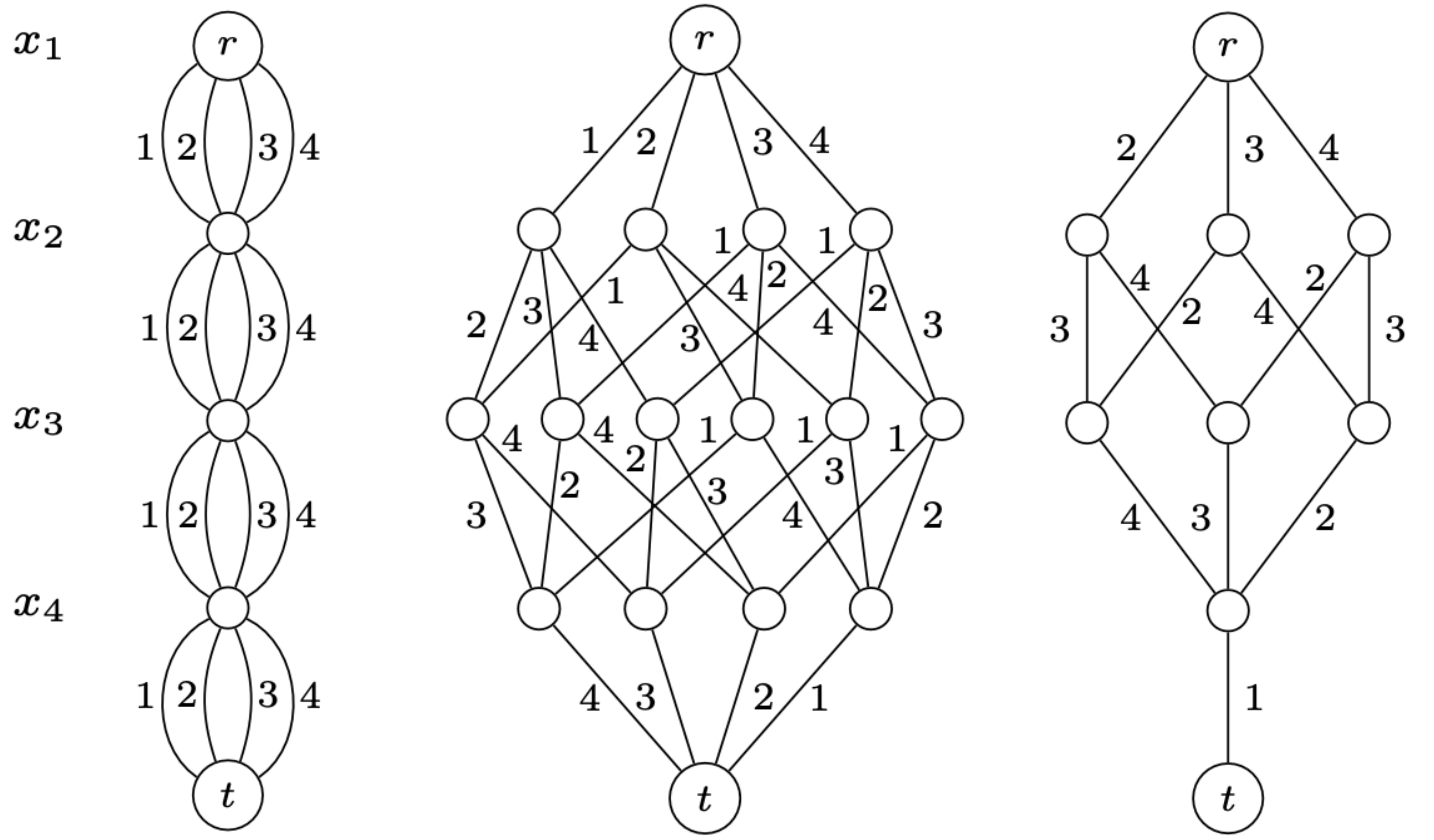
- Remove inconsistent arcs from diagram
- Use relaxed diagrams of polynomial size

[Andersen, Hadzic, Hooker, Tiedemann, 2007]

Optimization

- Evaluate objective to retrieve dual bound

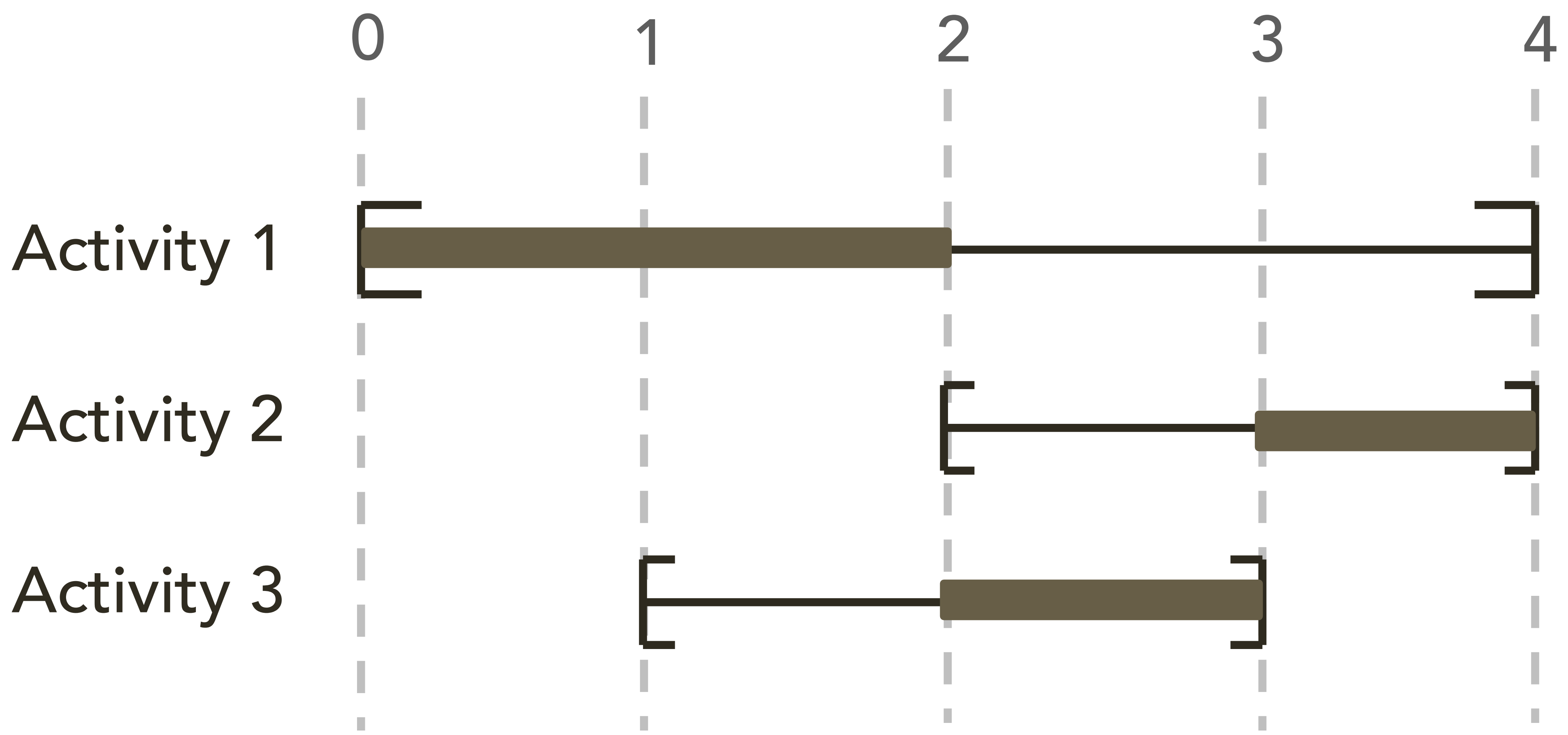
domain relaxation | propagate alldifferent | propagate $x_1 + x_2 + x_3 \geq 9$



256 solutions | 24 solutions | 6 solutions

Example Application: Disjunctive Scheduling

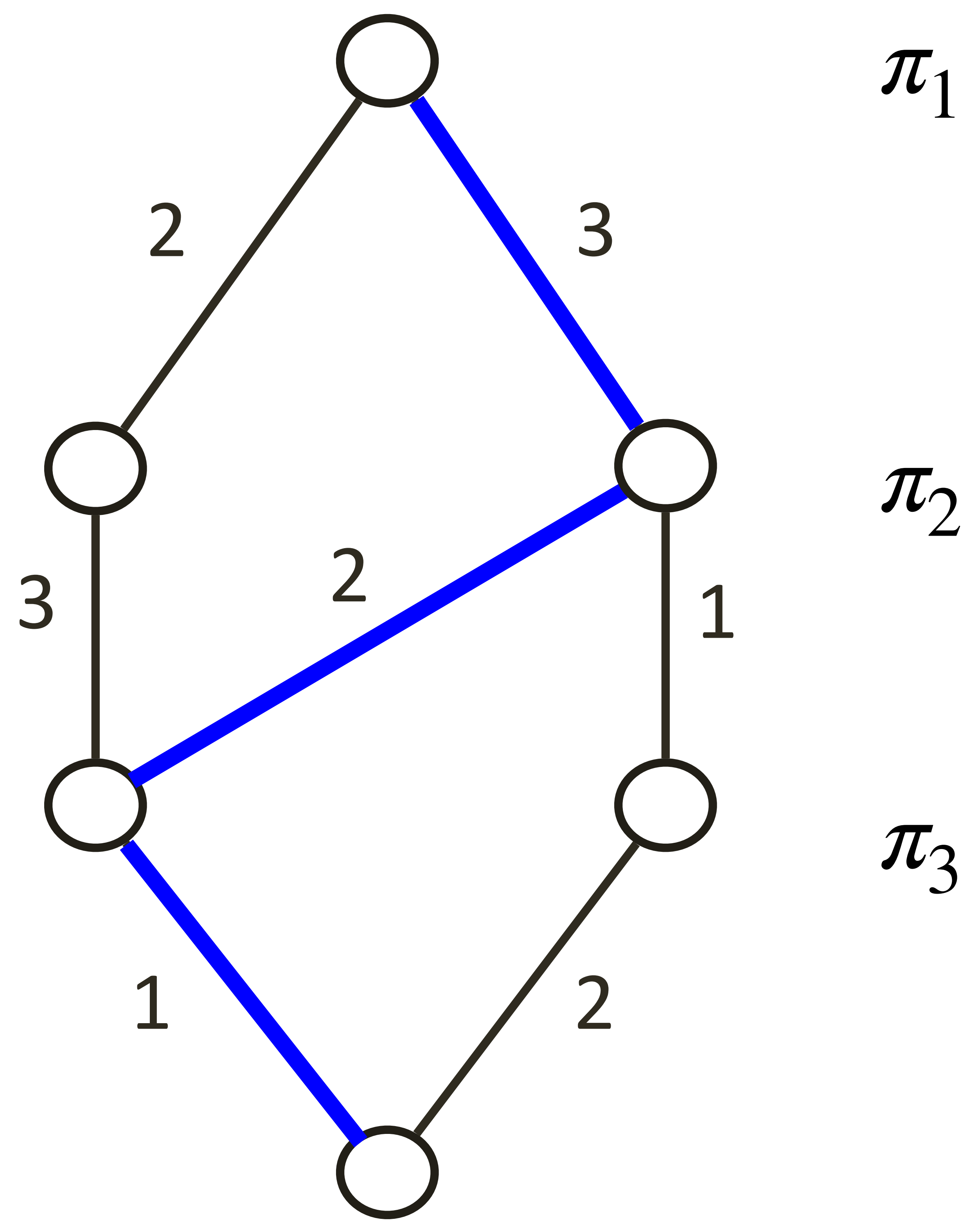
- Activities
  - Processing time: p_i
  - Release time: r_i
  - Deadline: d_i
- Resource
  - Nonpreemptive
  - Process one activity at a time
- Objective
  - Minimize makespan, sum of completion times, tardiness, …
- Optional side constraints
  - Precedence constraints, sequence-dependent setup times, …



Decision Diagram: Solution = Permutation

Act	r_i	p_i	d_i
1	3	4	12
2	0	3	11
3	1	2	10

precedence: $3 \ll 1$



Path 3 – 2 – 1 :

$6 \leq \text{start}_1 \leq 8$

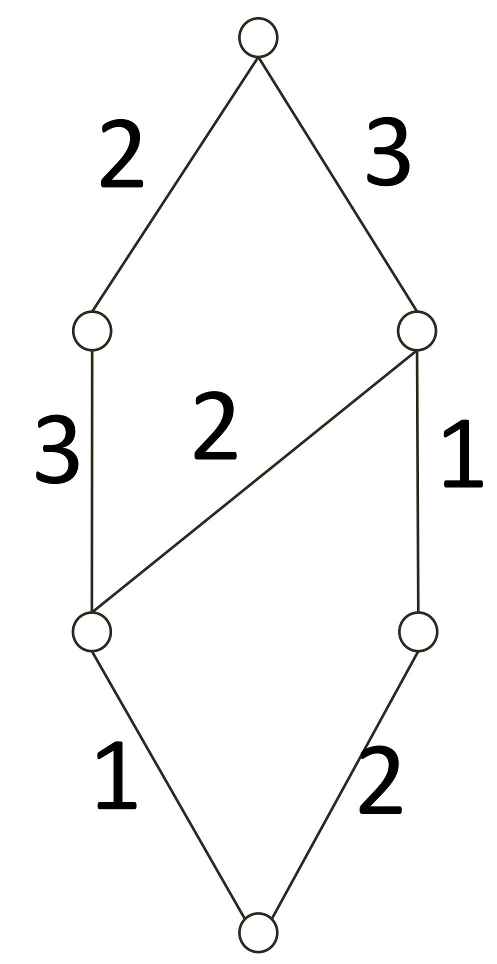
$3 \leq \text{start}_2 \leq 5$

$1 \leq \text{start}_3 \leq 3$

Solution: sequence of activities $\pi_1, \pi_2, \ldots, \pi_n$

MDD-Based Propagation

Propagation: remove infeasible arcs from the MDD



We can utilize several structures/constraints:

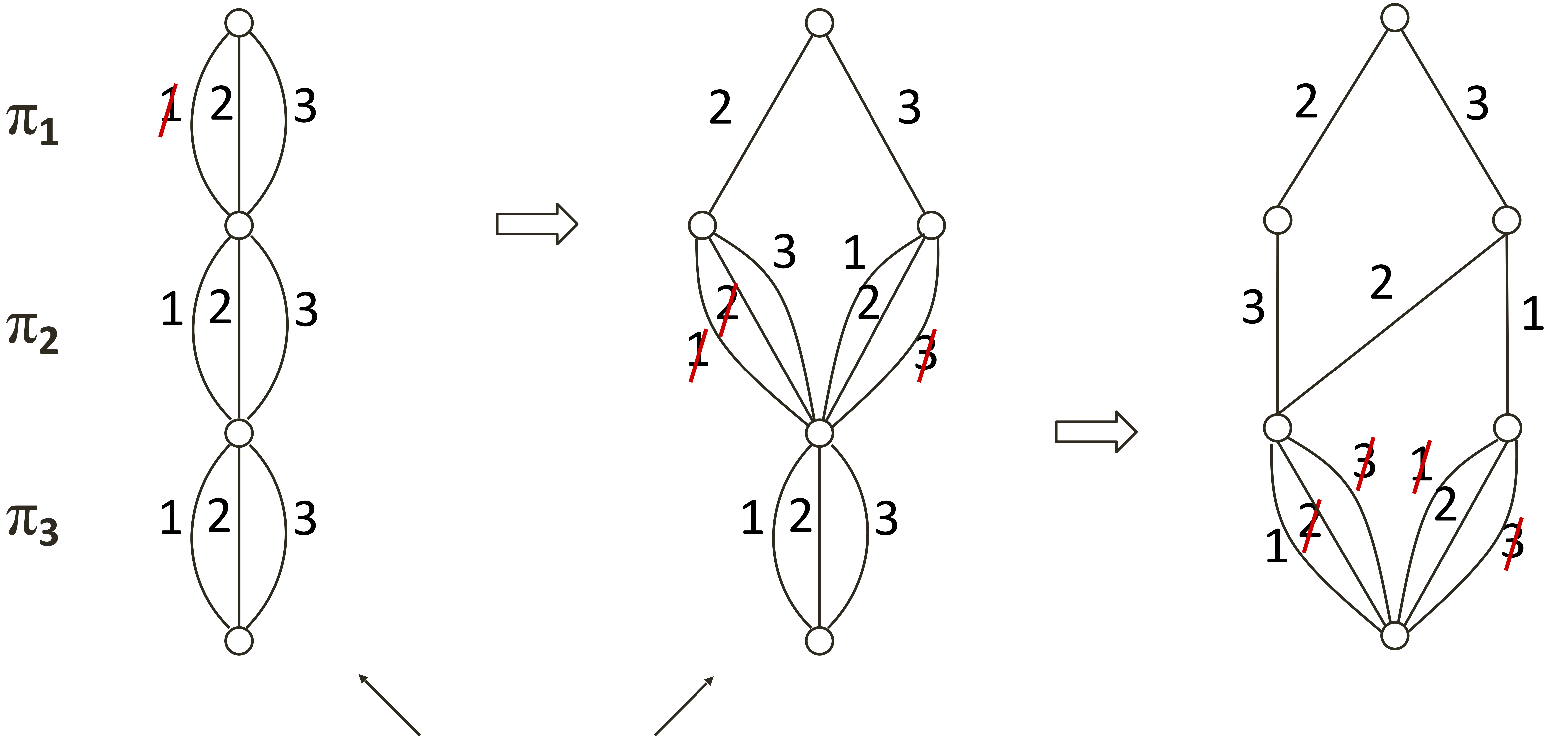
- *Alldifferent* for the permutation structure
- Earliest start time and latest end time
- Precedence relations

For each constraint type we maintain specific state information at each node in the MDD (both from top down and bottom up)

Bounding: Evaluate the objective function (longest/shortest path)

Top-down MDD compilation: Example

precedence:
$3 \ll 1$



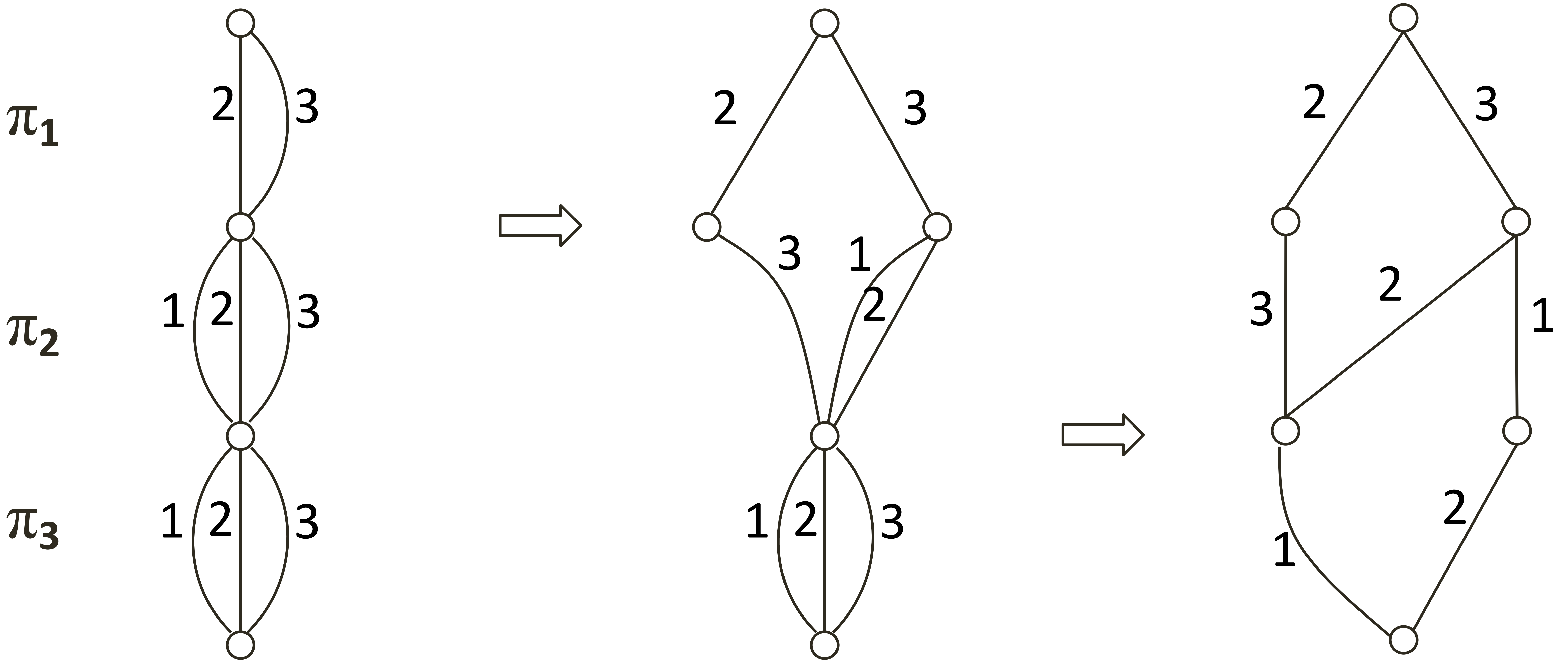
relaxed MDDs
(strength is controlled by maximum width)

exact MDD

Top-down MDD compilation: Example

precedence:

$3 \ll 1$

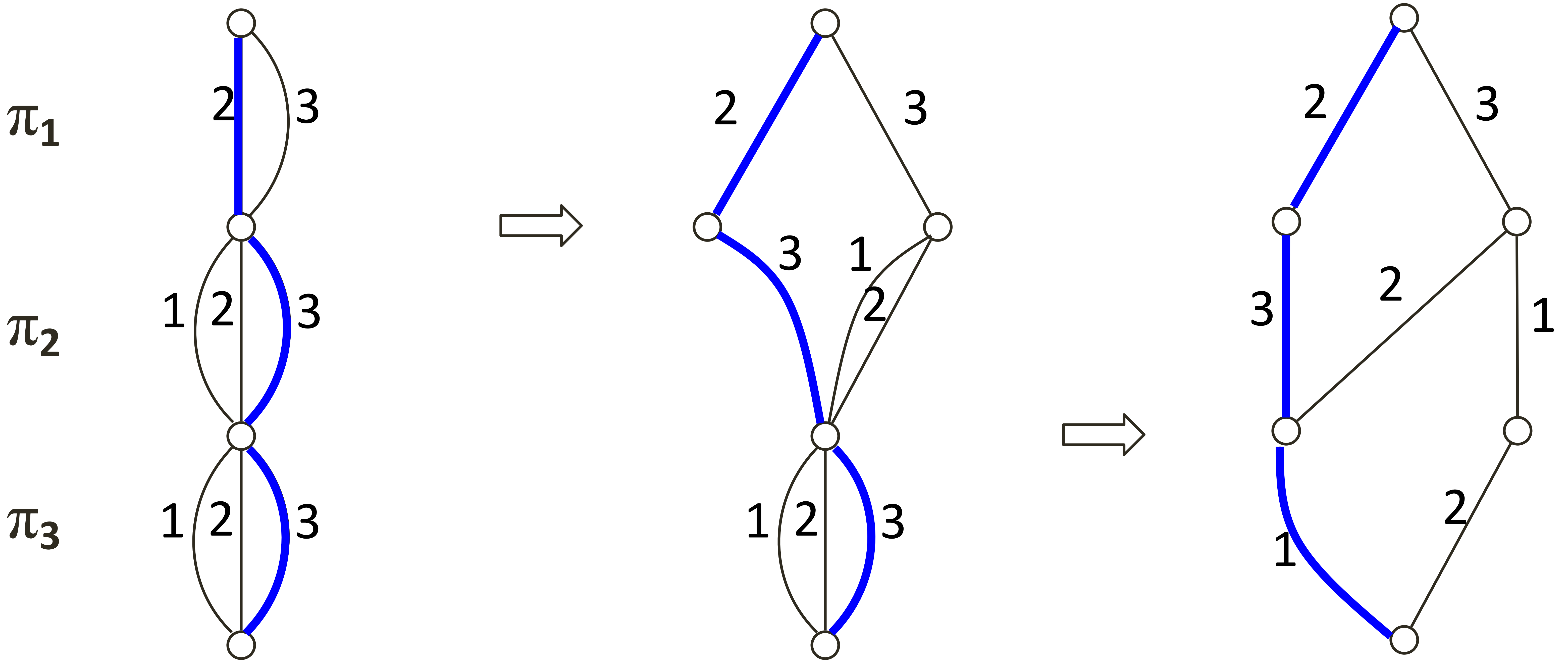


Top-down MDD compilation: Example

precedence:

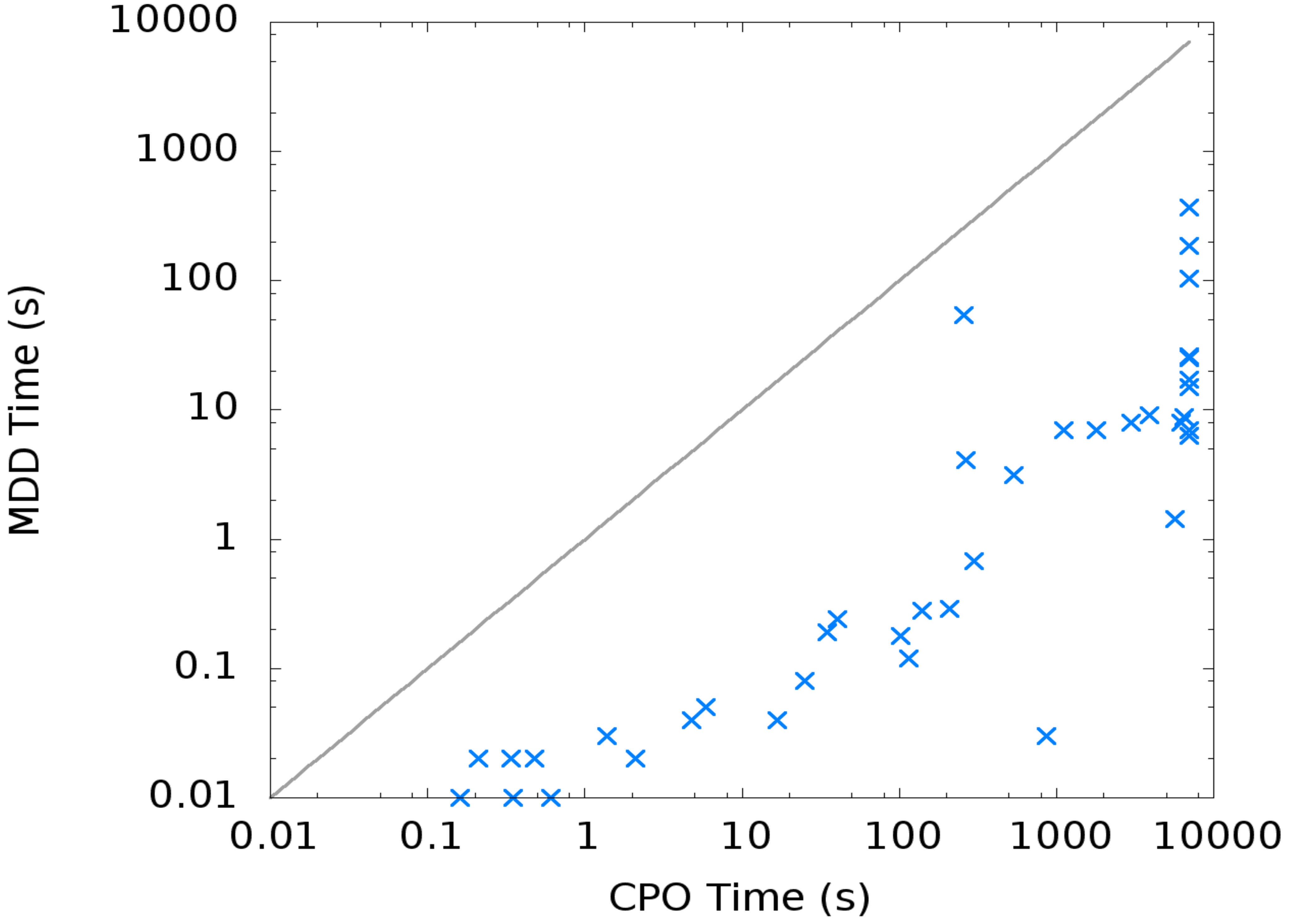
$3 \ll 1$

Act	r_i	p_i	d_i
1	3	4	12
2	0	3	11
3	1	2	10



minimize makespan: lower bound = 7 lower bound = 7 optimum = 9

Sequencing and Scheduling Applications



instance	vertices	bounds	CPO best	CPO time (s)	CPO+MDD, width 2048 best	CPO+MDD, width 2048 time (s)
br17.10	17	55	55	0.01	55	4.98
br17.12	17	55	55	0.01	55	4.56
ESC07	7	2125	2125	0.01	2125	0.07
ESC25	25	1681	1681	TL	1681	48.42
p43.1	43	28140	28205	TL	28140	287.57
p43.2	43	[28175, 28480]	28545	TL	**28480**	**279.18**
p43.3	43	[28366, 28835]	28930	TL	**28835**	**177.29**
p43.4	43	83005	83615	TL	83005	88.45
ry48p.1	48	[15220, 15805]	18209	TL	16561	TL
ry48p.2	48	[15524, 16666]	18649	TL	17680	TL
ry48p.3	48	[18156, 19894]	23268	TL	22311	TL
ry48p.4	48	[29967, 31446]	34502	TL	**31446**	**96.91**
ft53.1	53	[7438, 7531]	9716	TL	9216	TL
ft53.2	53	[7630, 8026]	11669	TL	11484	TL
ft53.3	53	[9473, 10262]	12343	TL	11937	TL
ft53.4	53	14425	16018	TL	14425	120.79

TSP with Time Windows

- 20-60 cities (Dumas/Ascheuer instances)
- max MDD width: 16

Compare MDD with CP Optimizer

Sequential Ordering Problem (TSPLib)

- TSP + precedence constraints
- max MDD width: 2048

Closed 3 instances for the first time

[Cire&vH 2013]

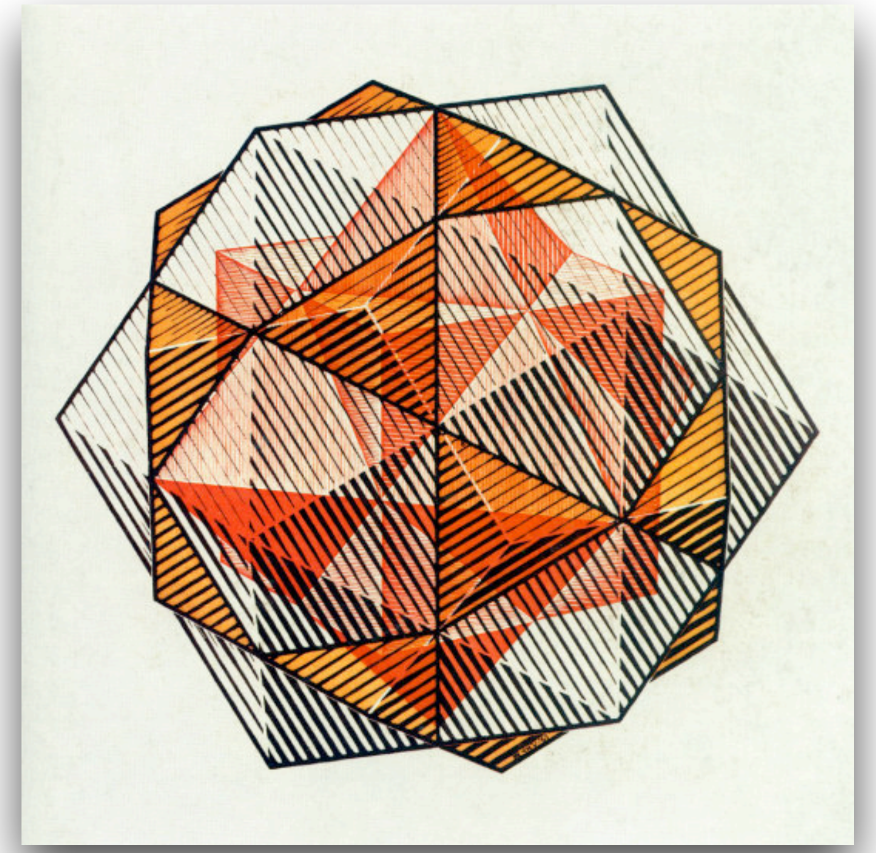
(DALL-E, 2024)

Branch-and-Bound Solver

(Durer, 1514)

Constraint Programming

(Escher, 1961)

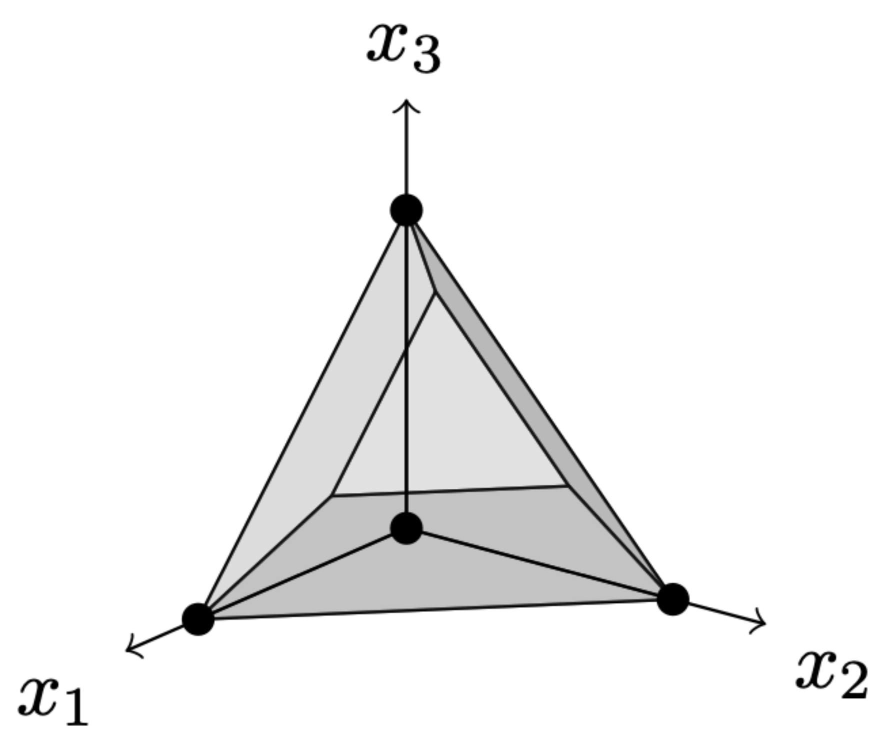


Integer Programming

(DALL-E, 2024)

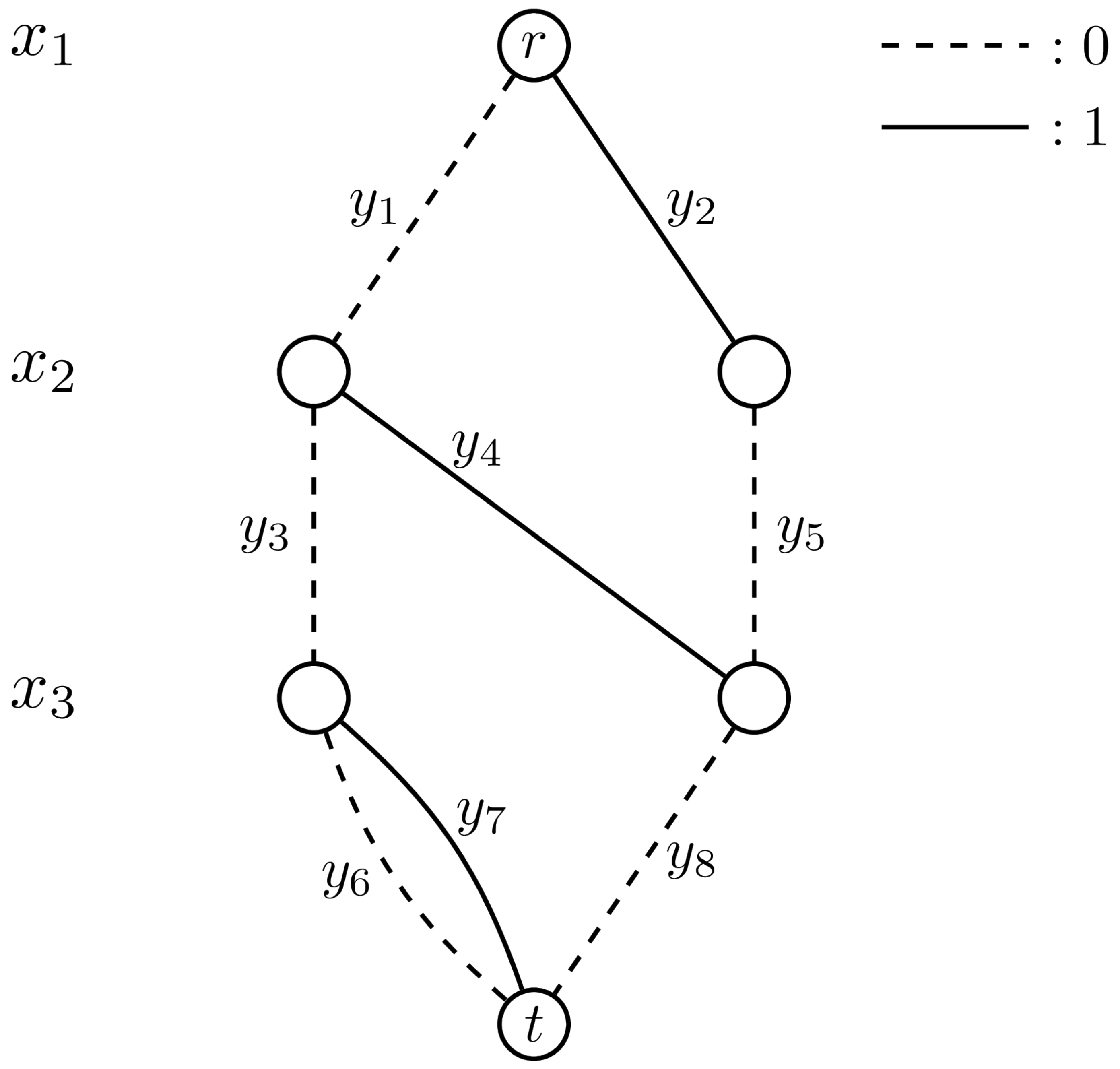
Column Elimination

Representing Integer Feasible Sets



$$\{x \in \{0,1\}^3 : \quad \begin{aligned} x_1 + x_2 - x_3 &\leq 1, \\ x_1 - x_2 + x_3 &\leq 1, \\ -x_1 + x_2 + x_3 &\leq 1, \\ x_1 + x_2 + x_3 &\leq 2 \,\} \end{aligned}$$

Integer Program Representation



Feasible Set as Decision Diagram

$$y_1 + y_2 = 1$$

$$\begin{aligned} y_1 &= y_3 + y_4 \\ y_2 &= y_5 \\ y_3 &= y_6 + y_7 \\ y_4 + y_5 &= y_8 \end{aligned}$$

$$y_6 + y_7 + y_8 = 1$$

$$0 \leq y_i \leq 1, \;\; i \in \{1, \ldots, 8\}$$

$$\begin{aligned} x_1 &= 0y_1 + 1y_2 \\ x_2 &= 0y_3 + 1y_4 + 0y_5 \\ x_3 &= 0y_6 + 1y_7 + 0y_8 \end{aligned}$$

Network Flow + Mapping

Theorem [Behle07]: Projecting the decision diagram network flow reformulation back to the original space yields the convex hull.

Applications to Integer Programming

1. Reformulate (non-linear) components as decision diagram, and add the network flow model to the integer program
   - Quadratic objectives [BergmanCire18], quadratic constraints [BergmanLozano21]

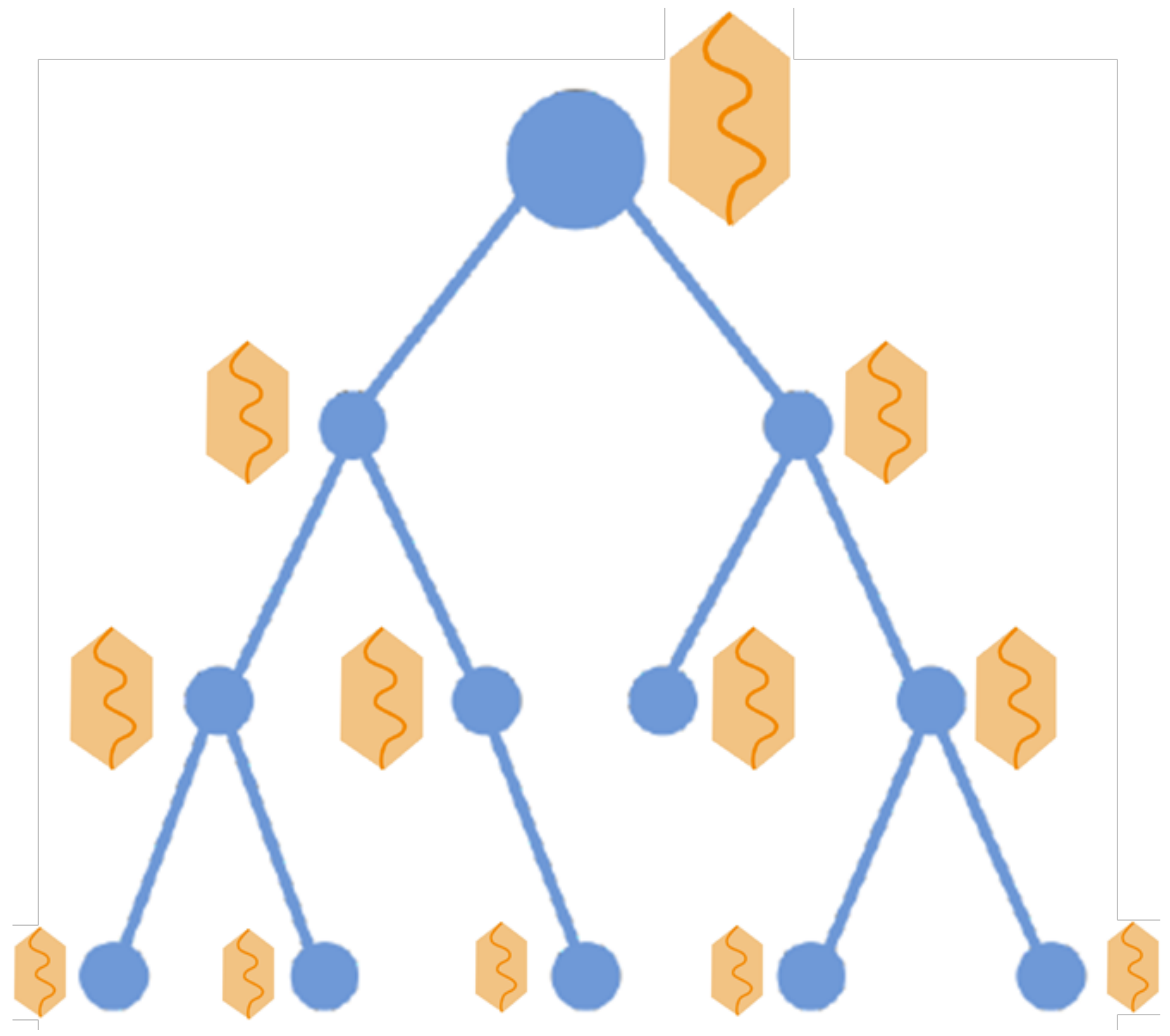
2. Use (relaxed) decision diagrams to represent (part of) the model to generate cutting planes
   - Integer linear programs: [Behle+05] [Behle07] [TjandraatmadjaVH19]
   - Integer nonlinear programs: [DavarniaVH21] [Castro+22]

3. Use (relaxed) decision diagrams to compute dual bounds within the branch-and-bound search tree [TjandraatmadjaVH21]

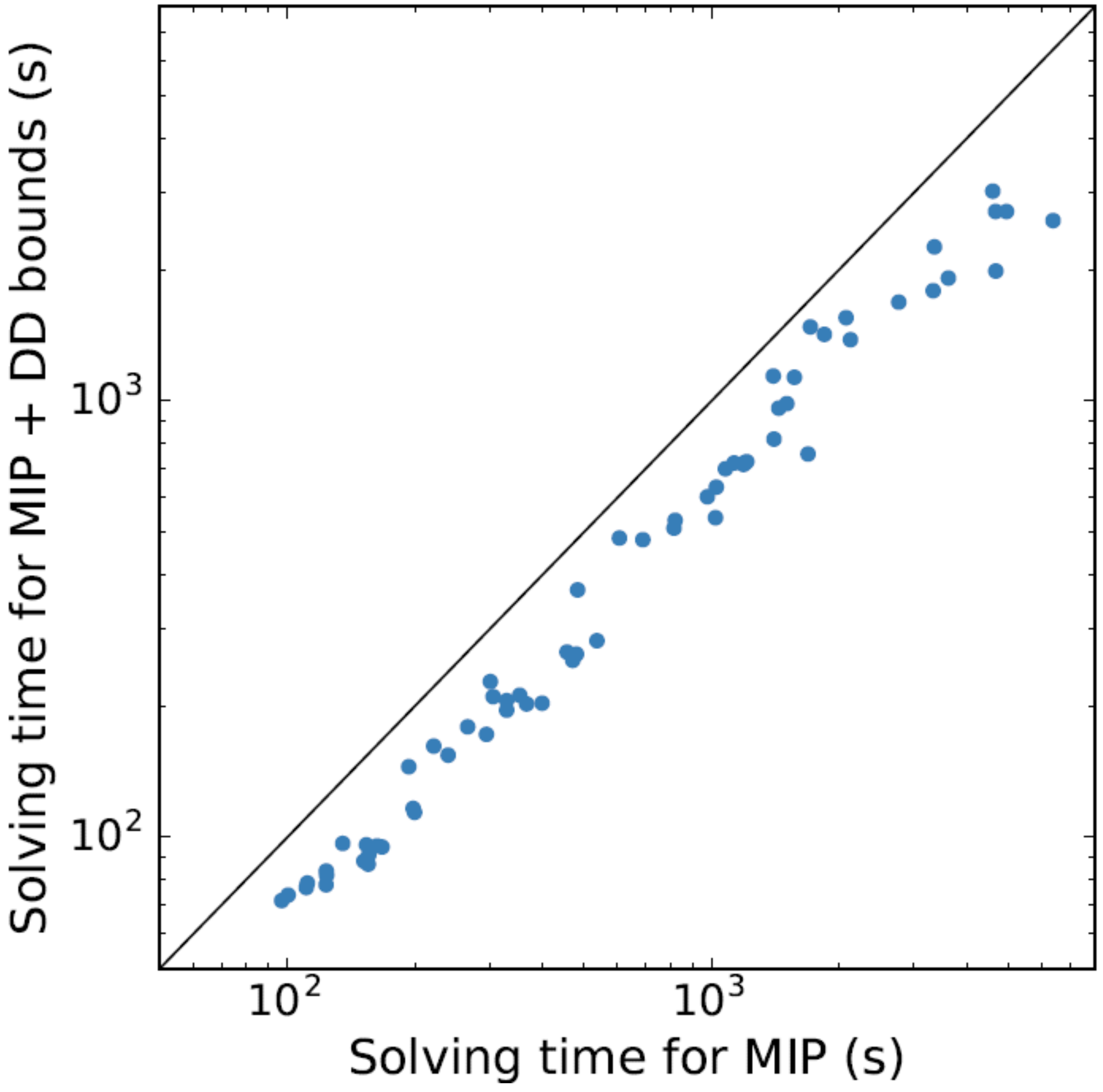
Decision Diagram Bounds in Integer Programming

[TjandraatmadjaVH21]

- Integration Design
  - Compile relaxed decision diagram for *conflict graph* in IP solver
  - Strengthen diagram by constraint propagation and Lagrangian bound
- Implemented in SCIP 5.0.1
  - Only IP model is given to solver
  - DD compiled automatically at each search node
- Experimental Setup
  - Independent set problem on random graphs
  - Add set of random knapsack constraints



Decision Diagrams Can Speed Up IP Solvers



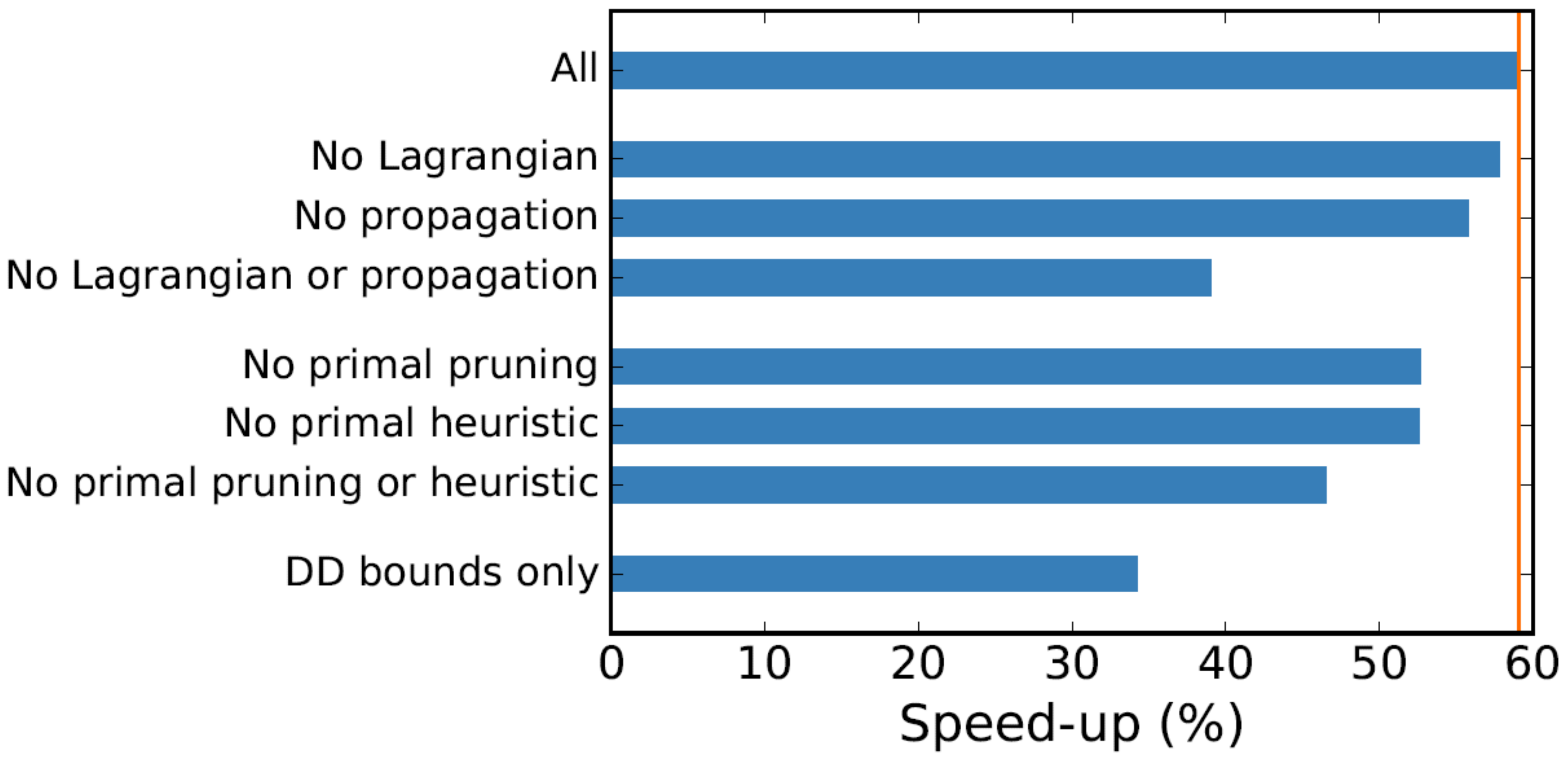
Instances: Watts-Strogatz random graphs

n = 300, 350, 400, 450 vertices

m = 0.1n knapsack constraints

On average: 65.5% node reduction

1.59x speedup



(DALL-E, 2024)

Branch-and-Bound Solver

(Durer, 1514)

Constraint Programming

(Escher, 1961)

Integer Programming

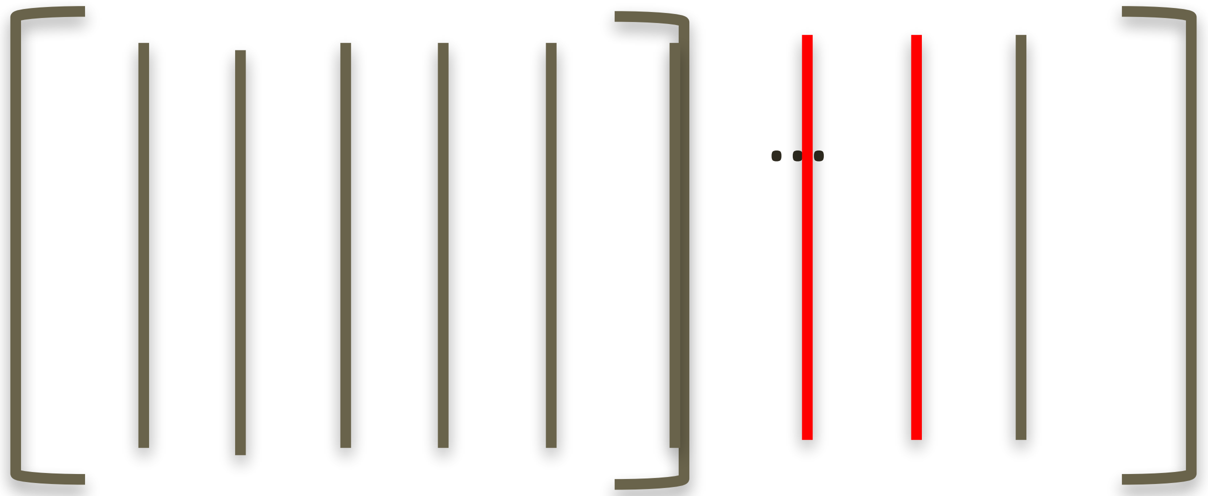
(DALL-E, 2024)

Column Elimination

From Column Generation to Column Elimination

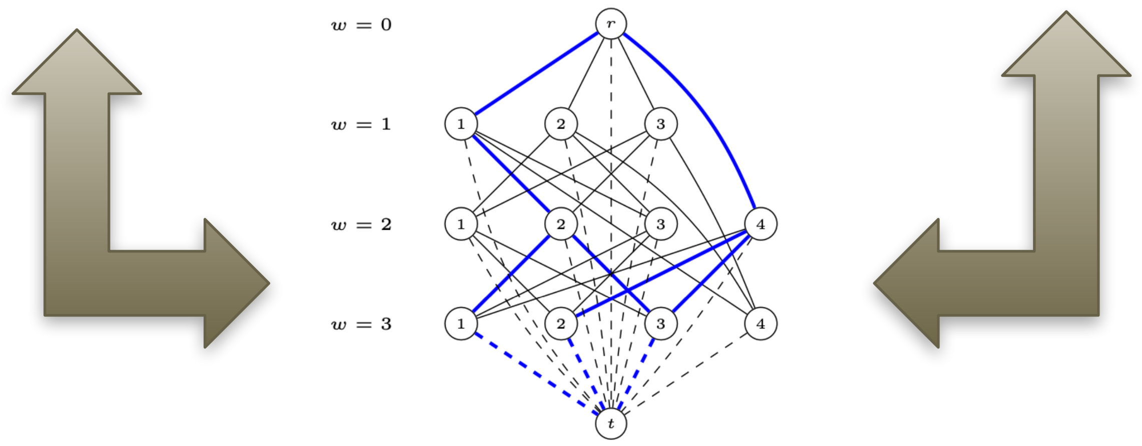
Column Generation Model

Min $c^T x$ s.t.



$x \geq b$

$x \in \{0,1\}^n$

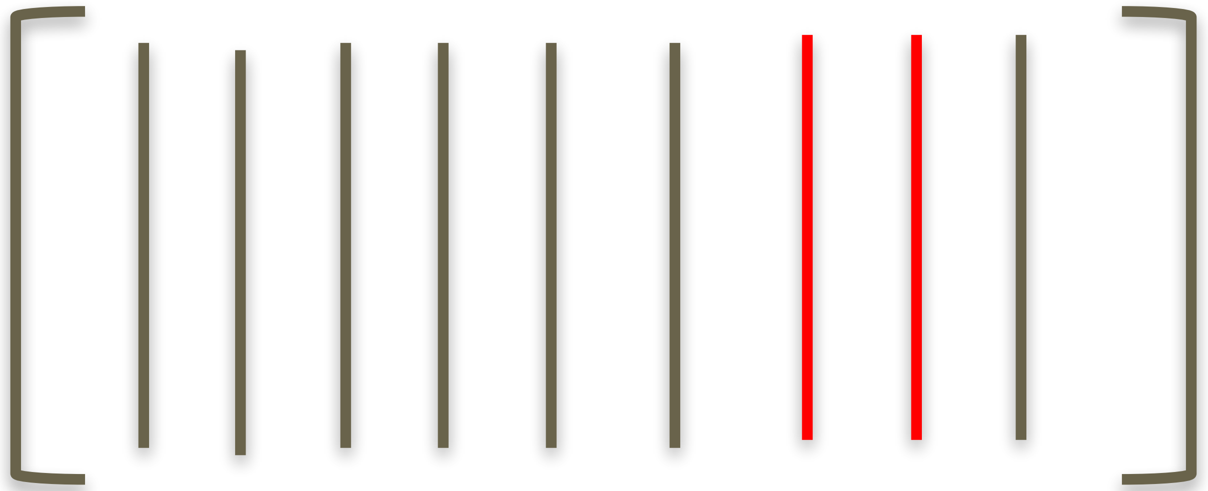


The pricing problem is often relaxed and solved with a smaller dynamic program.

From Column Generation to Column Elimination

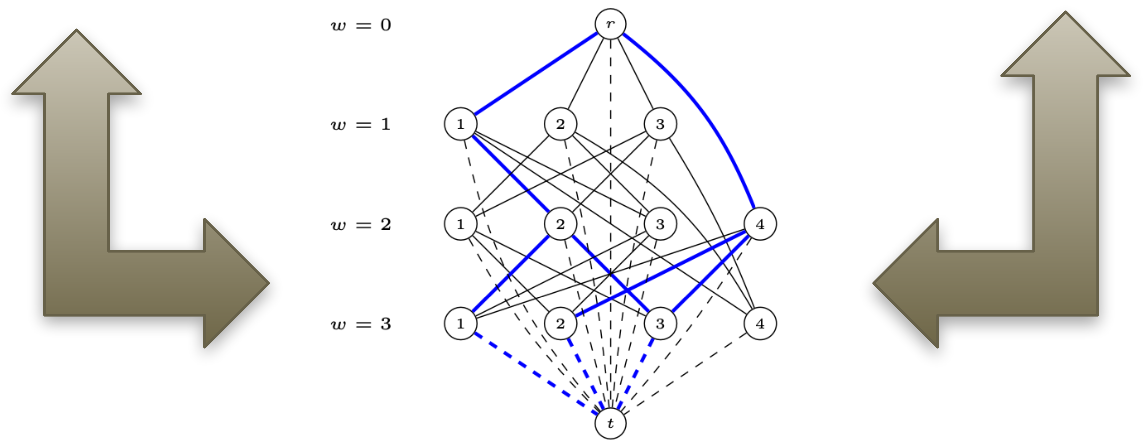
Column Generation Model

Min $c^T x$ s.t.



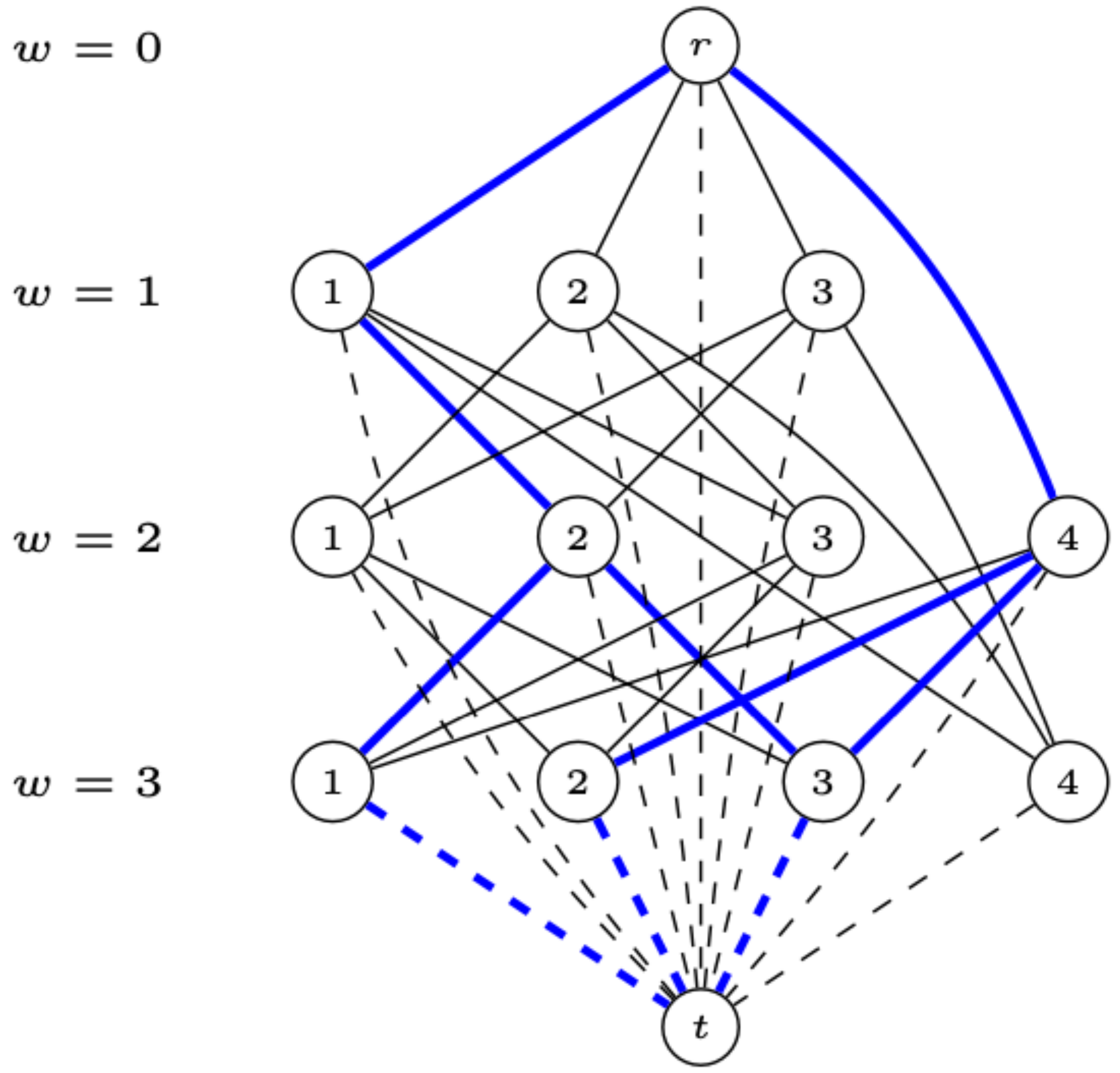
X $\geq$ b

$x \in \{0,1\}^n$



Column Elimination Model

Min $c^T x$ s.t.



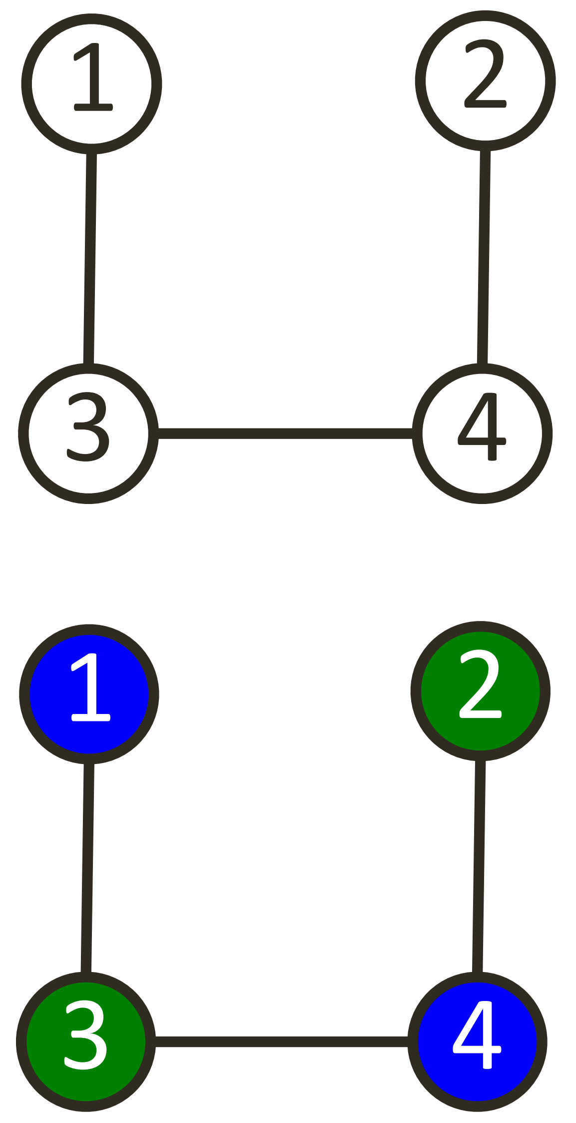
X $\geq$ b

The pricing problem is often relaxed and solved with a smaller dynamic program.

Could we use the smaller dynamic program to directly model a relaxation of the IP?

Example: Graph Coloring

- Assign a color to each vertex such that adjacent vertices have a different color. Minimize the number of colors.
- MIP model: binary variable x_i for each independent set i
- Comparatively strong LP relaxation

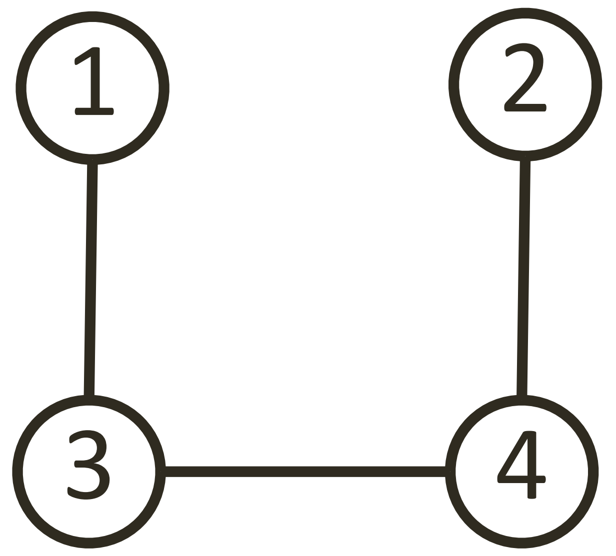


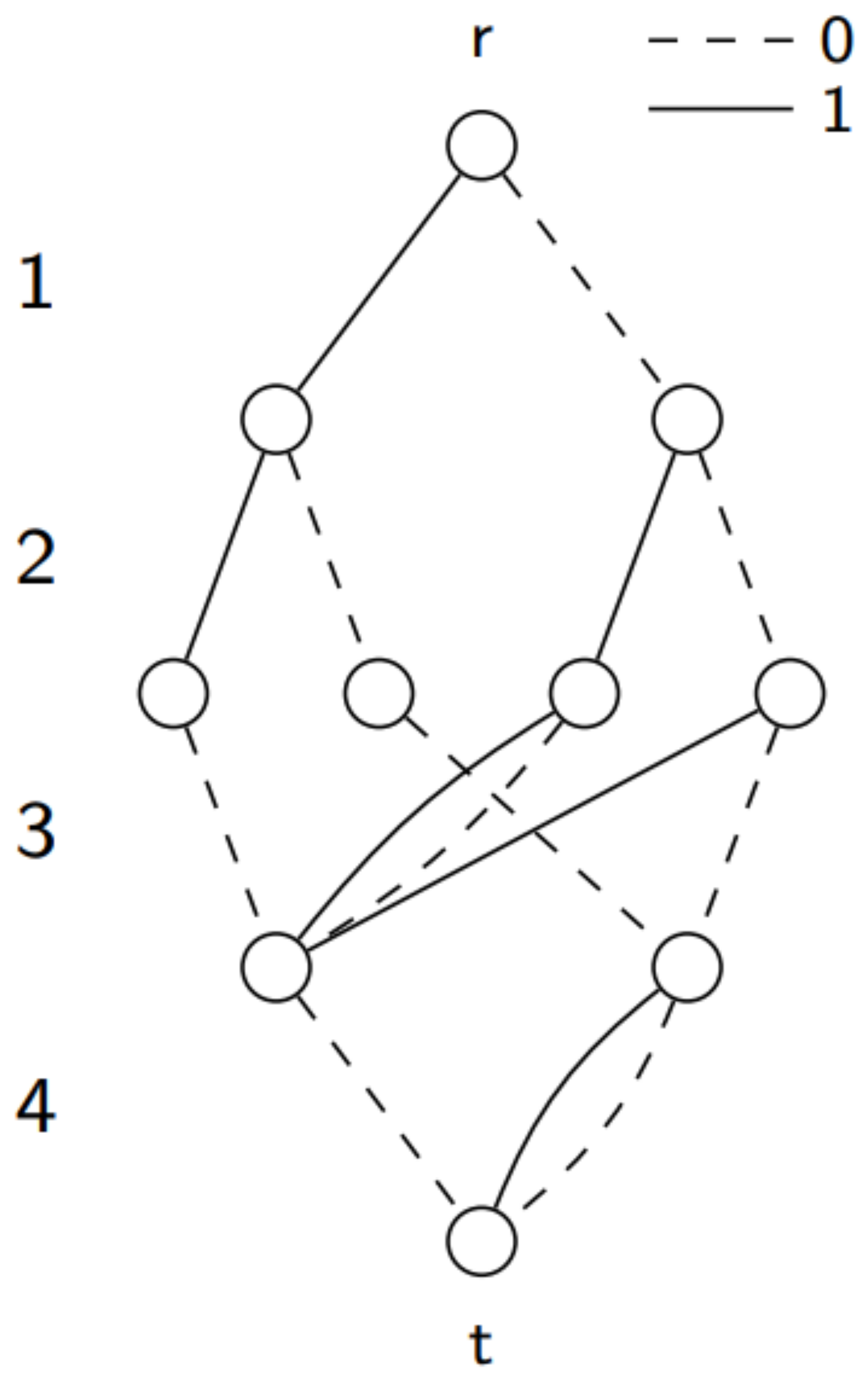
$$
\begin{aligned}
\min \quad & \sum_{i \in I} x_i \\
\text{s.t.} \quad & \sum_{i \in I} a_{ij} x_i = 1 \quad && \forall j \in V \\
& x_i \in \{0, 1\} && \forall i \in I
\end{aligned}
$$

$I = \{ \{1\},\{2\},\{3\},\{4\}, \{1,2\},\{1,4\},\{2,3\} \}$

Drawback: I has exponential size

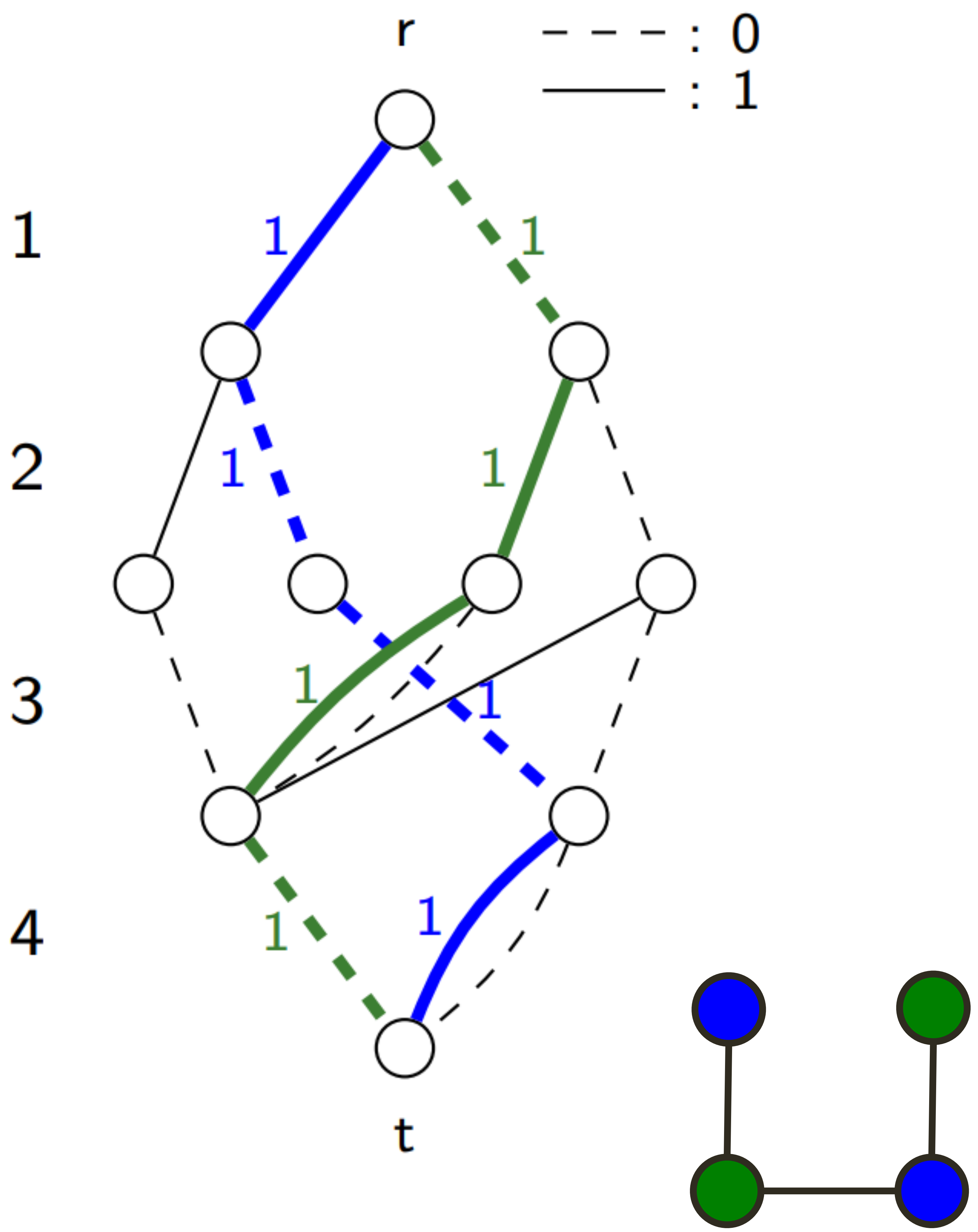
Decision Diagram Represents All Independent Sets





- We know how to compile these!
- Each r-t path corresponds to an independent set
- Compact representation, but still exponential in general

Reformulating the MIP Model as Arc Flow Model



Integer variable y_a : 'flow' through arc *a*

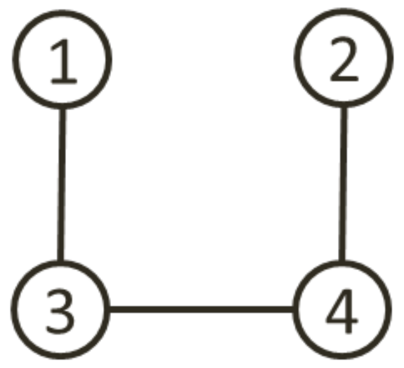
$$(F) = \min \sum_{a \in \delta^+(r)} y_a$$ — minimize number of paths (colors)

$$\text{s.t.} \sum_{a=(u,v) | L(u)=j, \ell(a)=1} y_a = 1 \quad \forall j \in V$$ — one 1-arc per vertex

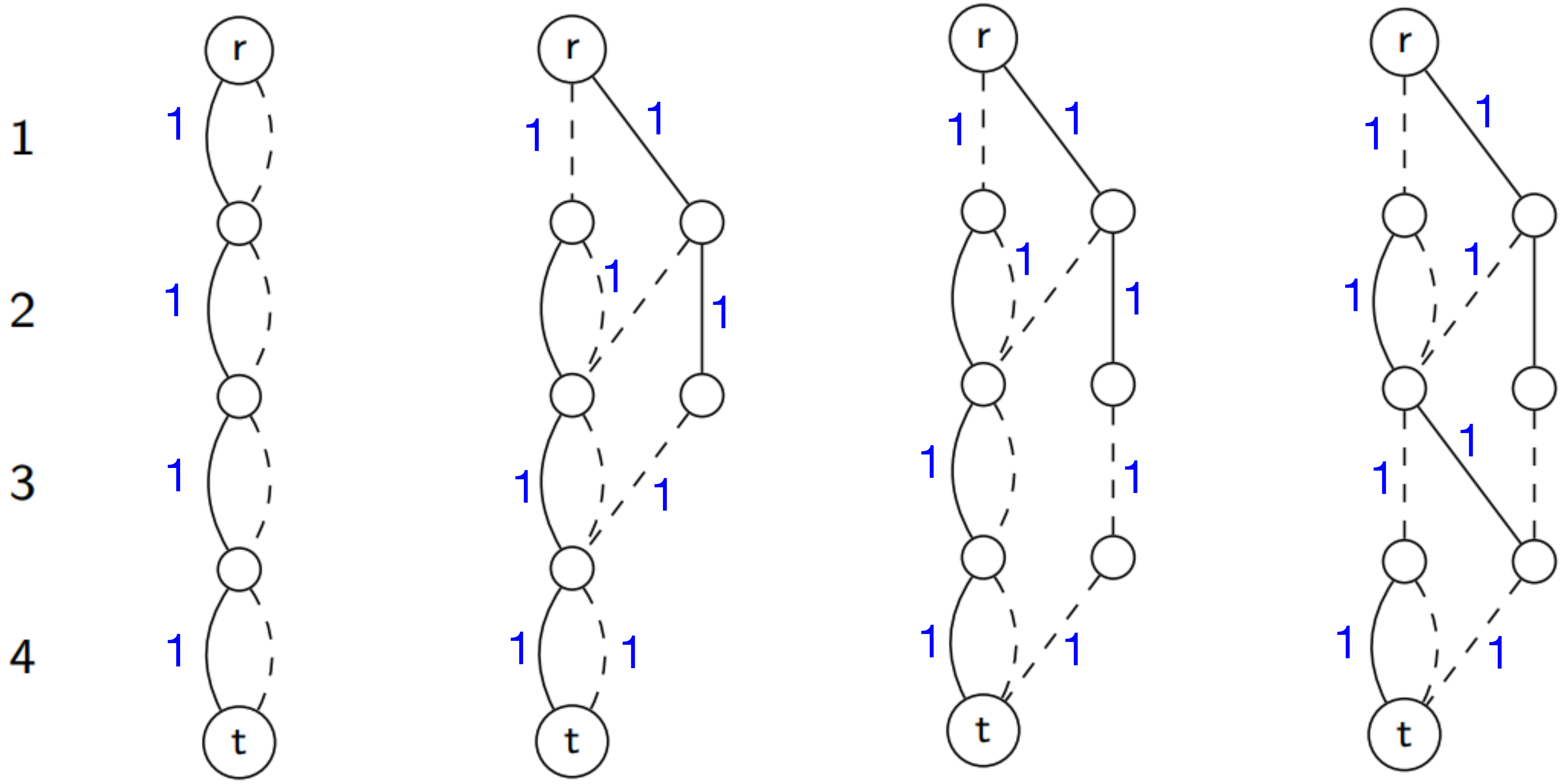
$$\sum_{a \in \delta^-(u)} y_a - \sum_{a \in \delta^+(u)} y_a = 0 \quad \forall u \in N \setminus \{r, t\}$$ — 'flow conservation'

$$y_a \in \{0, 1, \ldots, n\} \quad \forall a \in A$$ — integrality

Column Elimination: Iterative Refinement



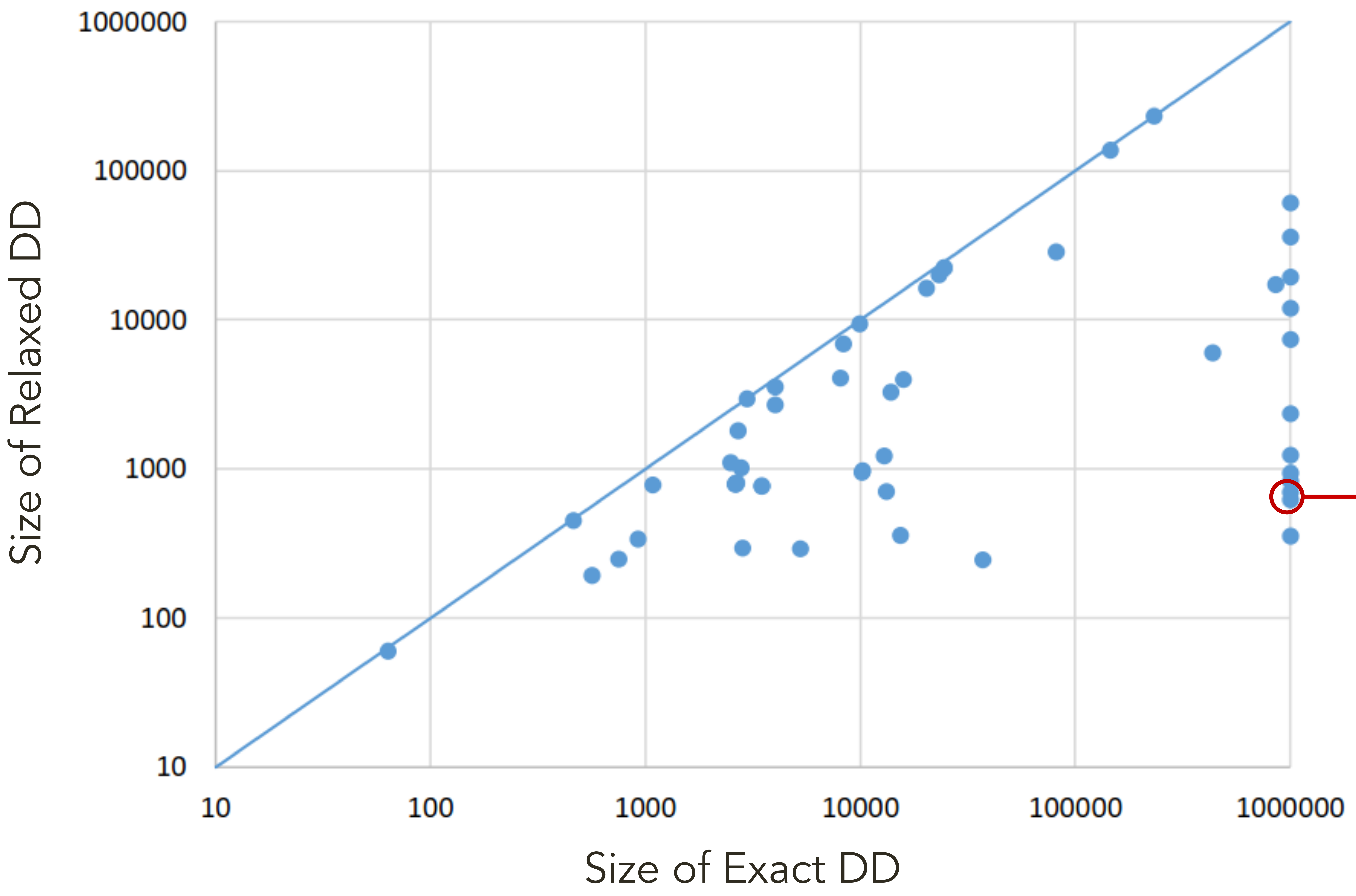
input graph



Optimal!

lower bound:	1	2	2	2
flow paths:	(1,1,1,1)	(1,1,0,1) (0,0,1,0)	(1,1,0,0) (0,0,1,1)	(1,0,0,1) (0,1,1,0)
conflicting edge:	(1,3)	(2,4)	(3,4)	-

Evaluation on DIMACS Benchmark Instances



- Relaxed decision diagram from column elimination can be orders of magnitude smaller than exact decision diagram to prove optimality, but not always
- DSJR500.1 (n=500, m=3,555)
  - Exact DD: ≥1M nodes
  - Relaxed DD: 627 nodes

(Each instance is solved to optimality by at least one of the two methods)

Column Elimination Algorithm

LP

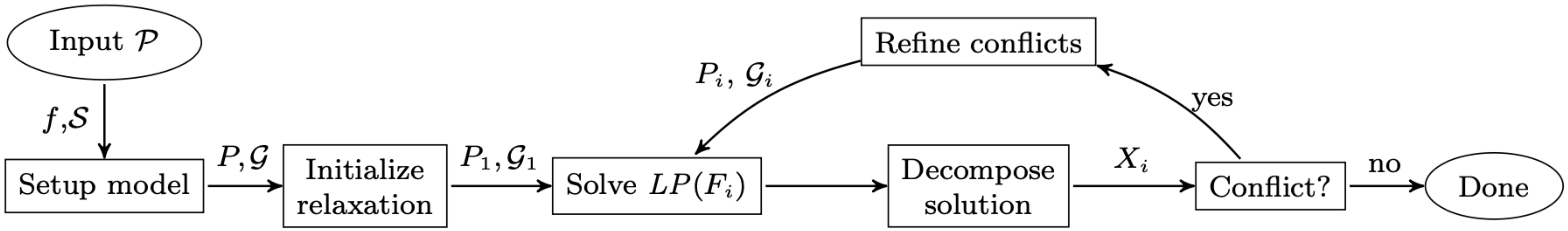


Figure 1 Column elimination for solving $LP(F)$.

IP

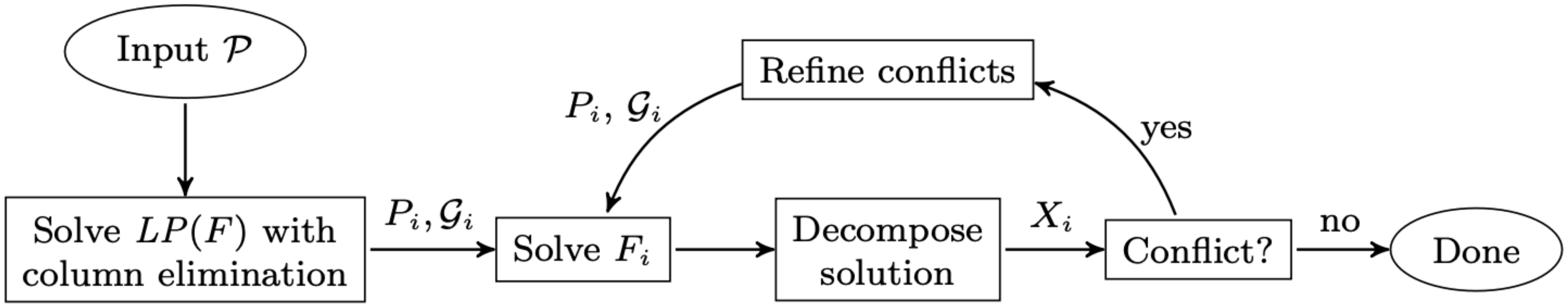
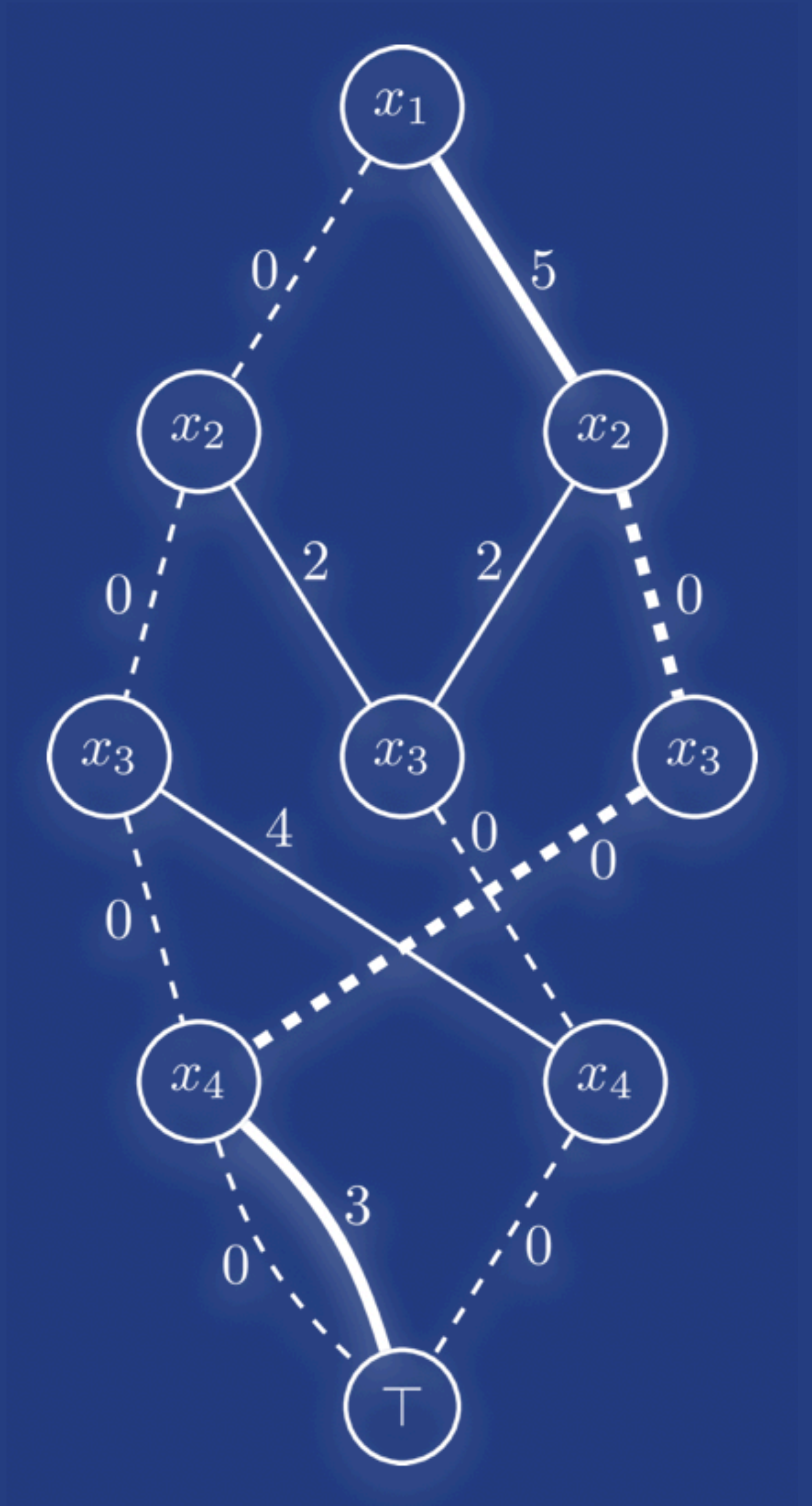


Figure 2 Column elimination for solving F.

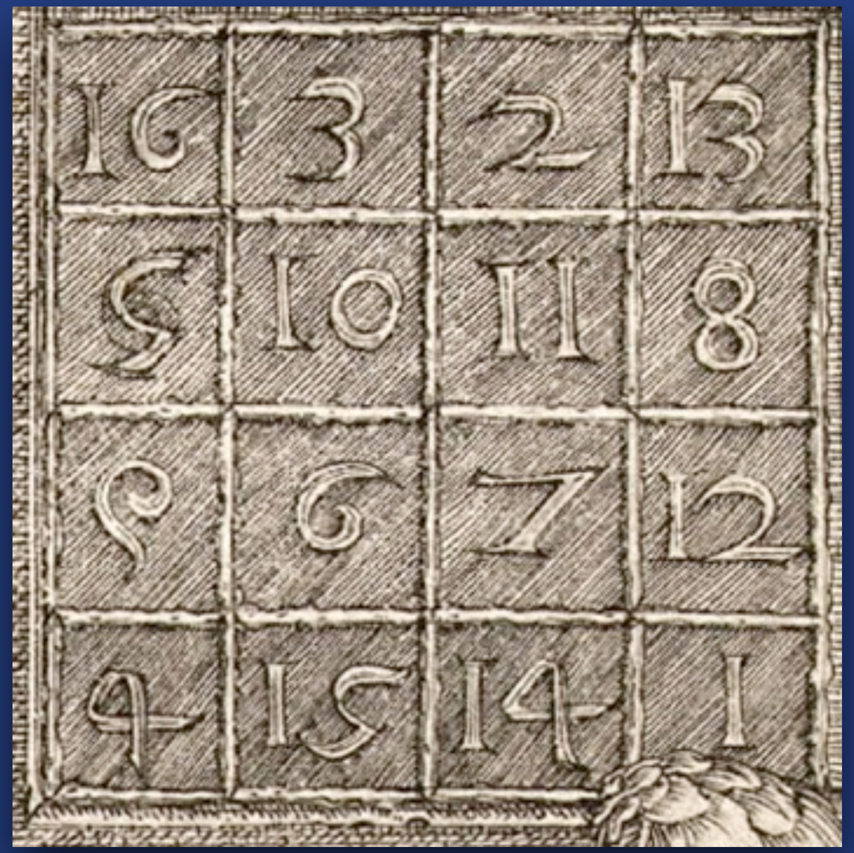
Column Elimination Can Provide State-of-the-Art Results

- Vehicle Routing Problem with Time Windows
  - For some instances column elimination finds better bounds than VRPSolver [Pessoa+20]
  - Column Elimination closes open instance C2_10_1 on 1,000 locations
- Graph Multi-Coloring Problem
  - Column Elimination closes five open benchmark instances
- Pickup-and-Delivery Problem with Time Windows
  - Column Elimination closes six open benchmark instances

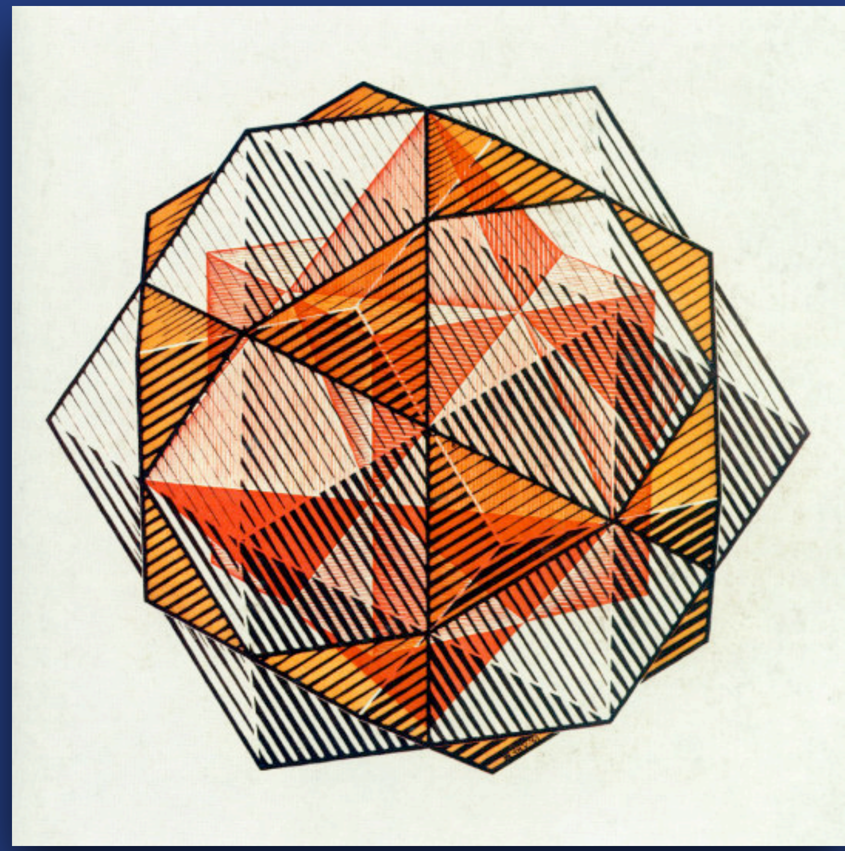
[KarahaliosVH, under review]



Branch-and-Bound Solver



Constraint Programming



Integer Programming

Column Elimination