Decision Diagrams for Optimization

Andre Augusto Cire

Dept. of Management, University of Toronto Scarborough

Rotman School of Management

MIP 2015, Chicago, June 2015

Collaborators

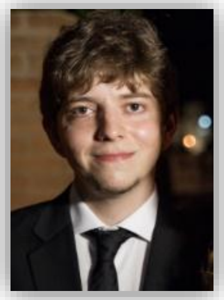
David Bergman
University of Connecticut

Joris Kinable
Carnegie Mellon University

Willem-Jan van Hoeve
Carnegie Mellon University

Christian Tjandraatmadja
Carnegie Mellon University

John Hooker
Carnegie Mellon University



Thiago Serra
Carnegie Mellon University

Our Main Research Goal

Investigate the use of Decision Diagrams for solving discrete optimization problems

Contributions so far

- **New relaxation/bounding technique**
  - Bounds can be superior to state-of-the-art methods in certain problems
- **Generic primal heuristic**
  - Scales to large-scale problems
- **Inference techniques**
  - New types of cuts for MIPs and other optimization technologies
- **Novel complete solution technique**
  - Solved open instances from classical benchmarks
  - **Parallel method** that scales almost linearly with number of processors

Decision Diagrams

- Graphical representation of **Boolean functions**

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

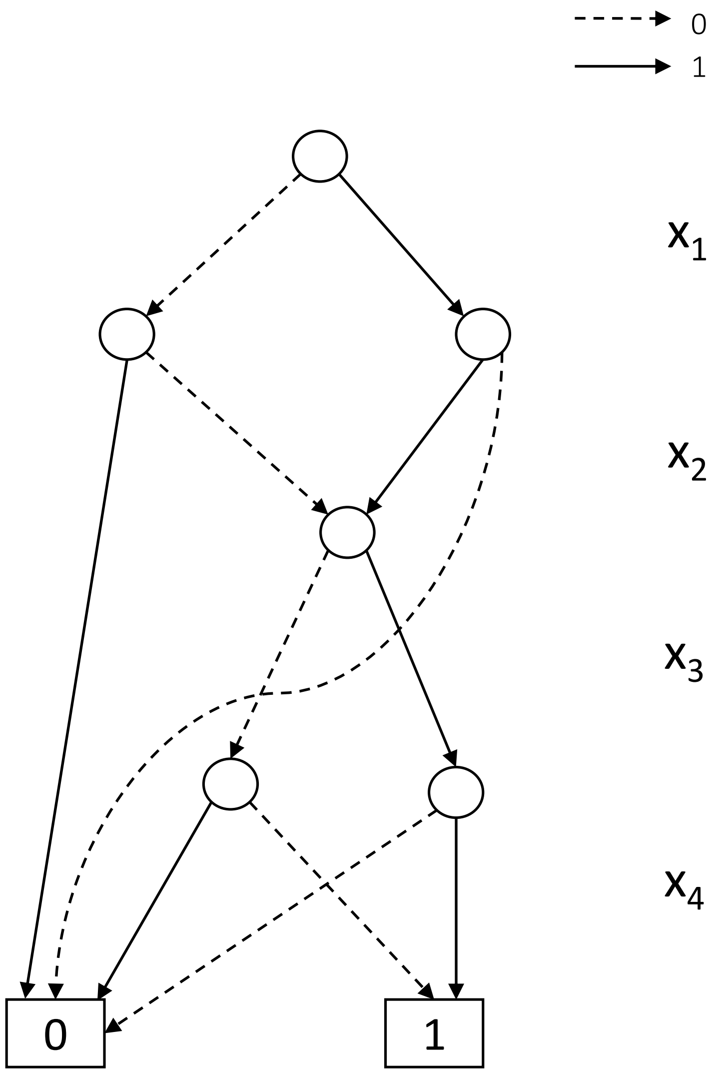
x_1	x_2	x_3	x_4	**f(x)**
0	0	0	0	**1**
0	0	0	1	**0**
0	1	1	0	**0**
0	0	1	1	**1**
…	…	…	…	**…**

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**
…	…	…	…	**…**

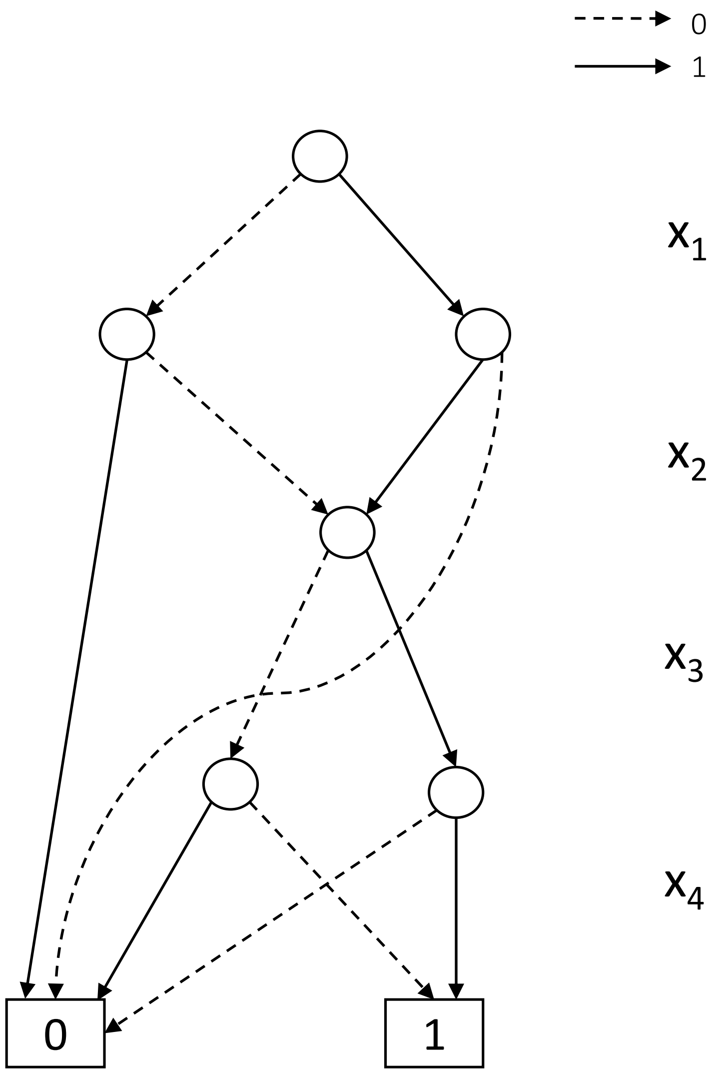


Decision Diagrams

- Graphical representation of **Boolean functions**

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

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**
…	…	…	…	**…**

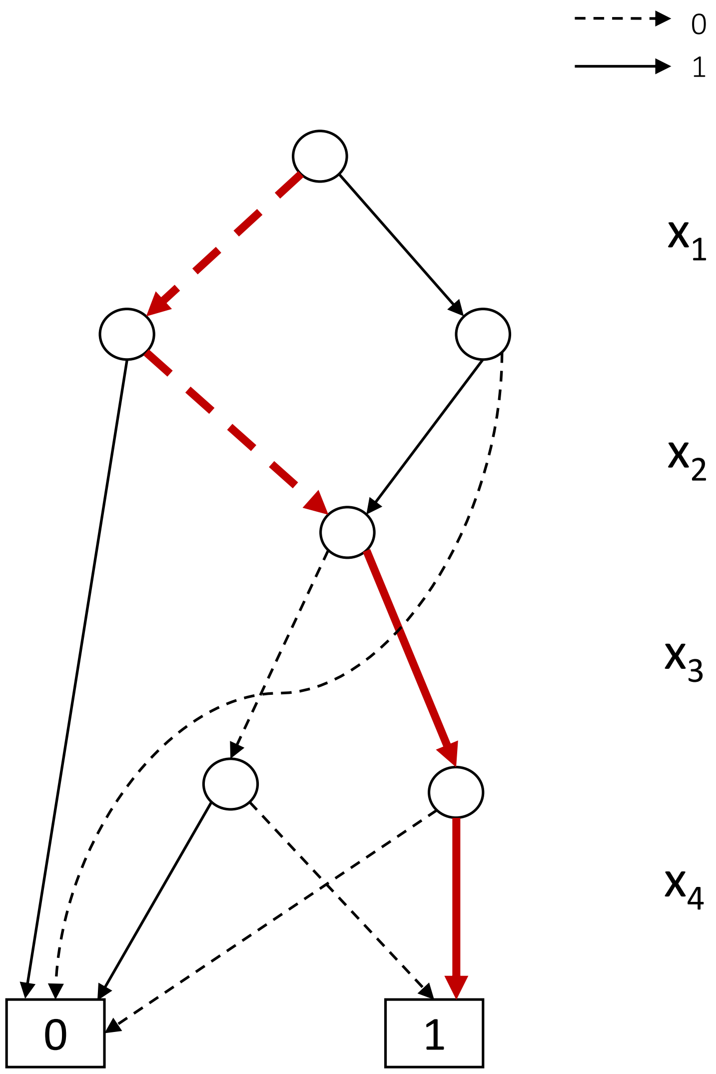


Decision Diagrams

- 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**
…	…	…	…	**…**

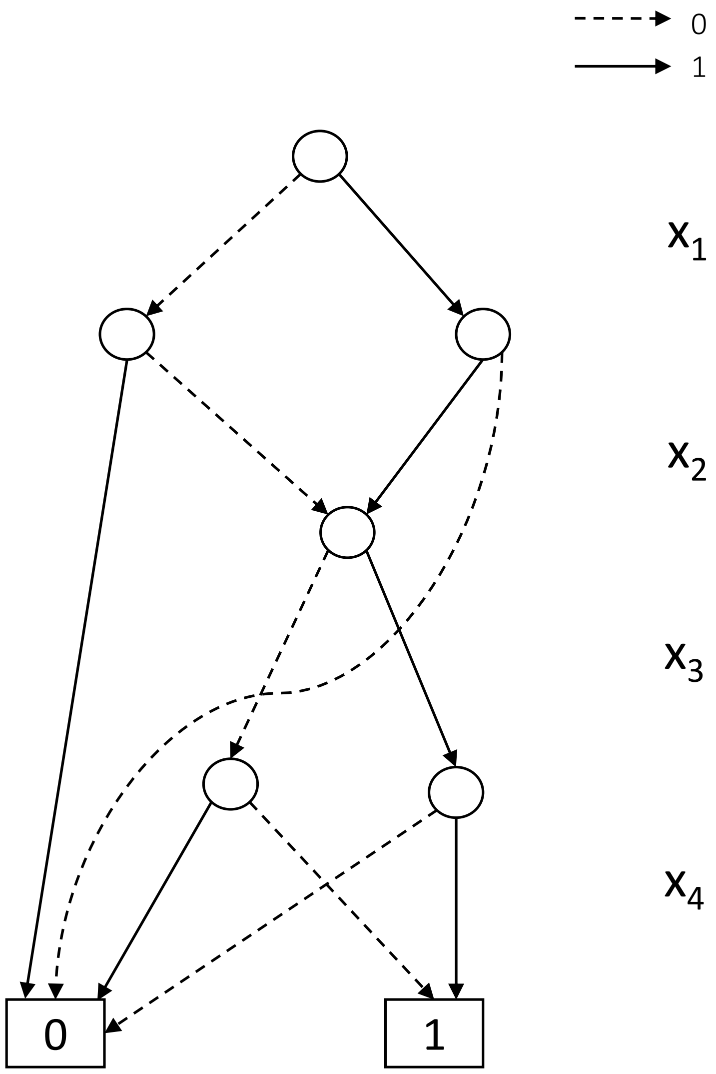


Decision Diagrams

- Graphical representation of **Boolean functions**

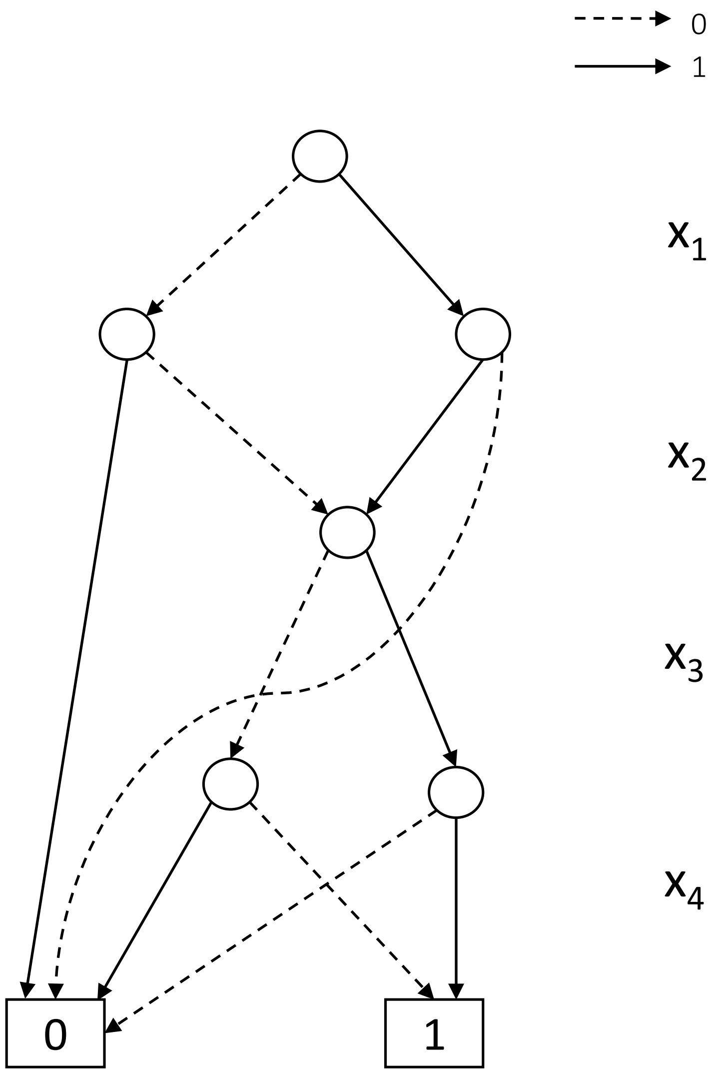
$$f(x) = \left(x_1 \Leftrightarrow x_2\right) \wedge \left(x_3 \Leftrightarrow x_4\right)$$

- **Dual role**
  - Computational model
  - Graphical encoding
- [Lee'59, Akers'78, Bryant'86]



Decision Diagrams

- **Application** in several areas
  - Circuit design
  - Formal verification
  - Symbolic model checking
  - …
- Our focus: **Optimization**
  - Literals → variables
  - Arcs → value assignments
  - Paths encode **solutions**

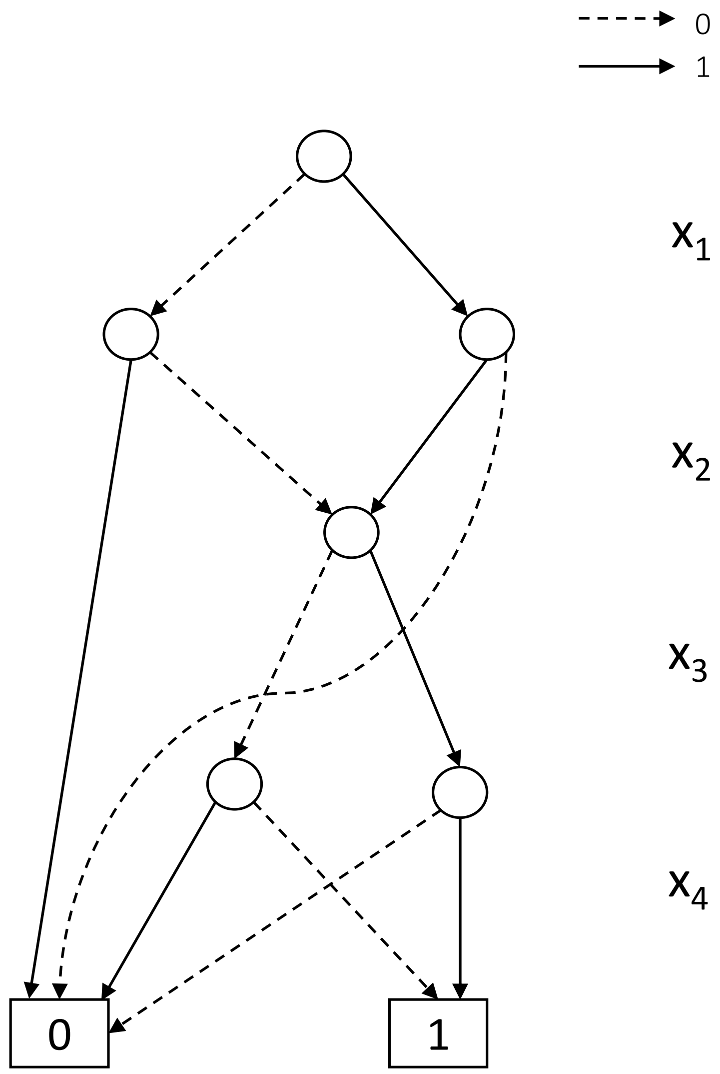


Decision Diagrams

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

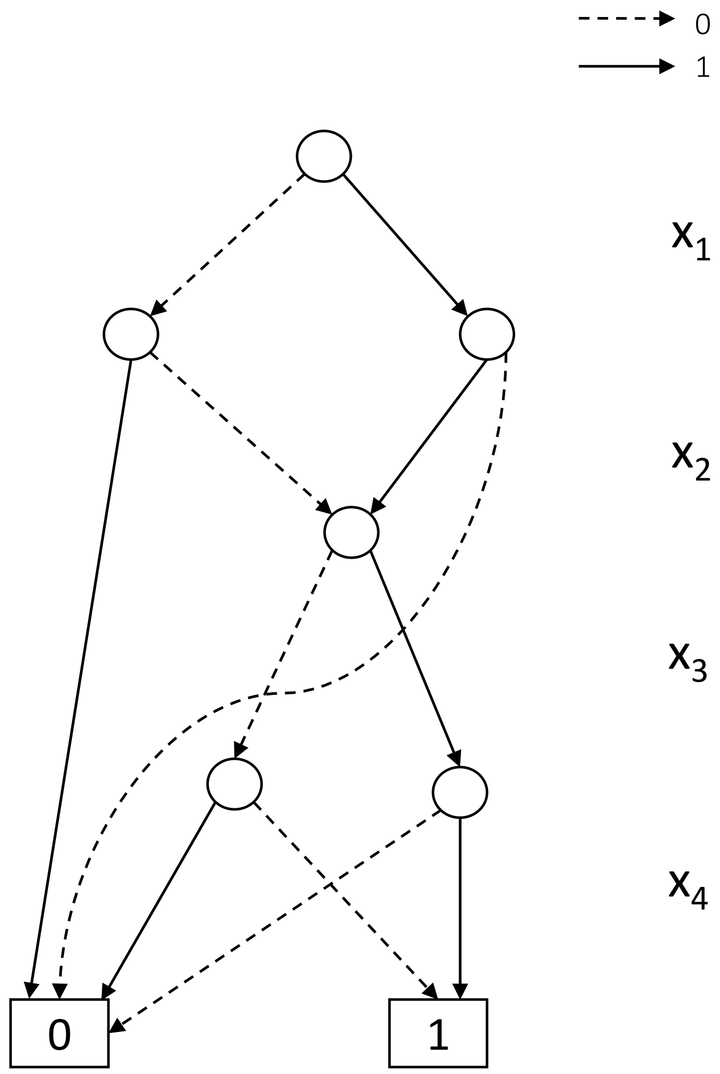
max $2x_1 + x_2 - 4x_3 + x_4$

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

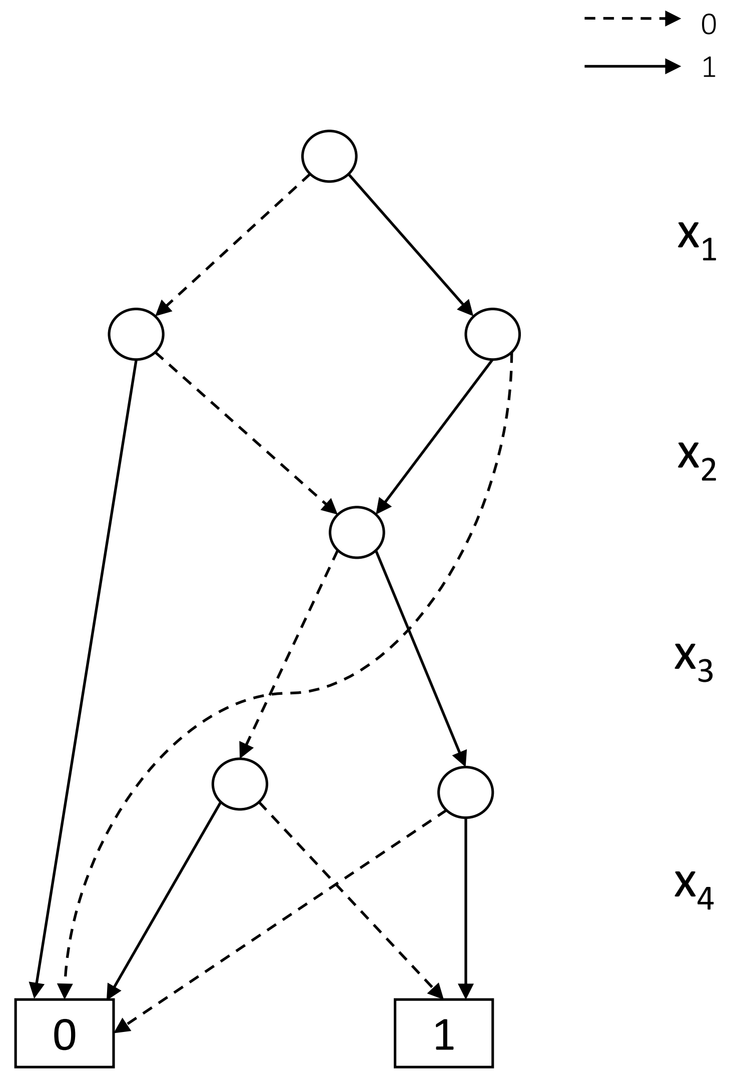
$\max \ 2x_1 + x_2 - 4x_3 + x_4$

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

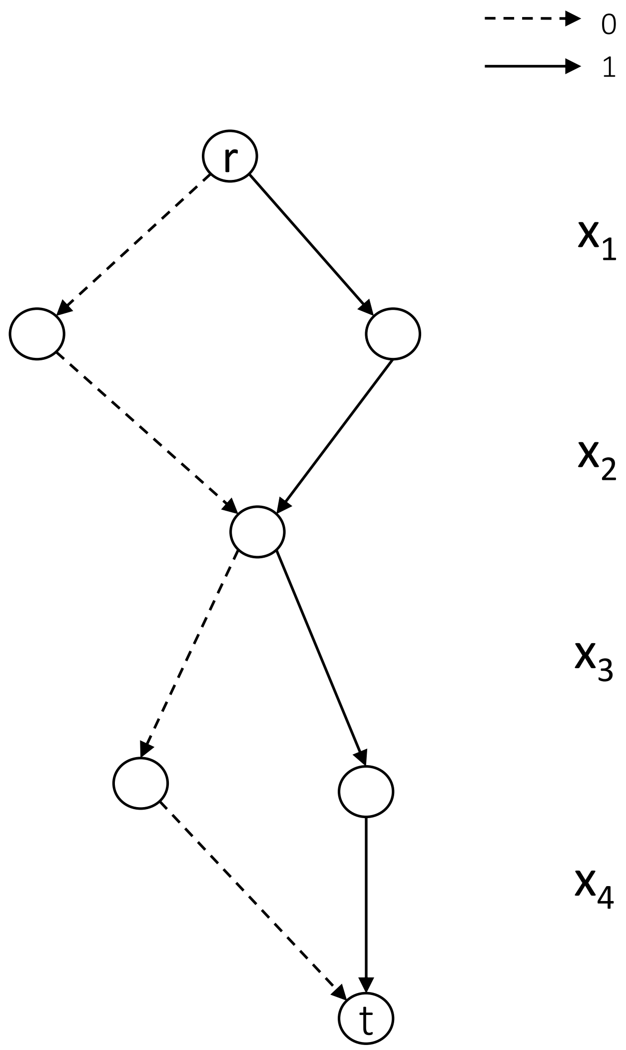
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$$x_1 - x_2 = 0$$

$$x_3 - x_4 = 0$$

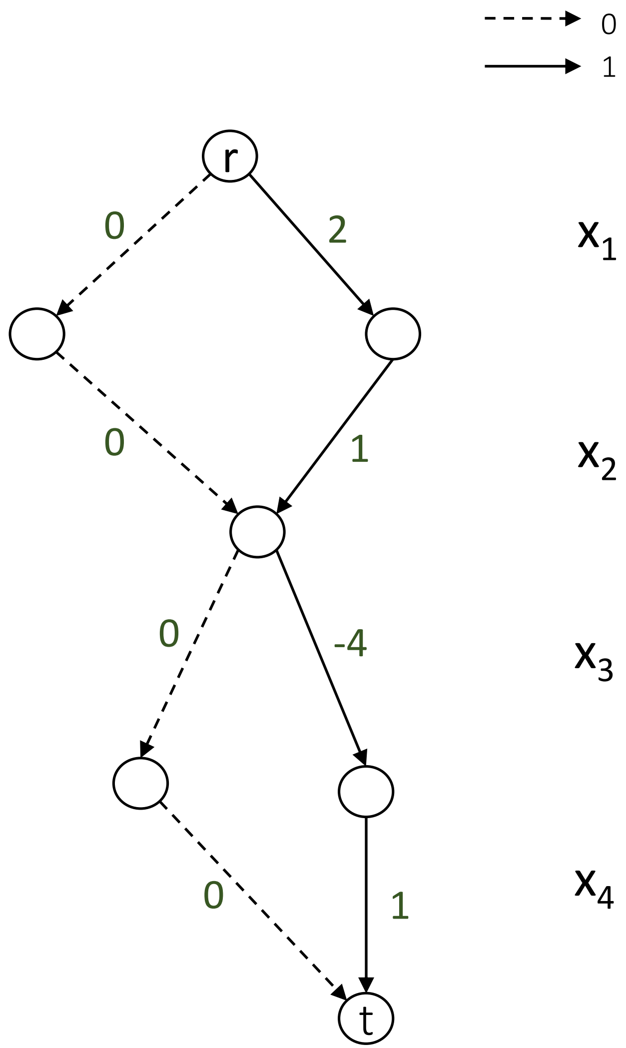
$$x_1, x_2, x_3, x_4 \in \{0,1\}$$



Decision Diagrams

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

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

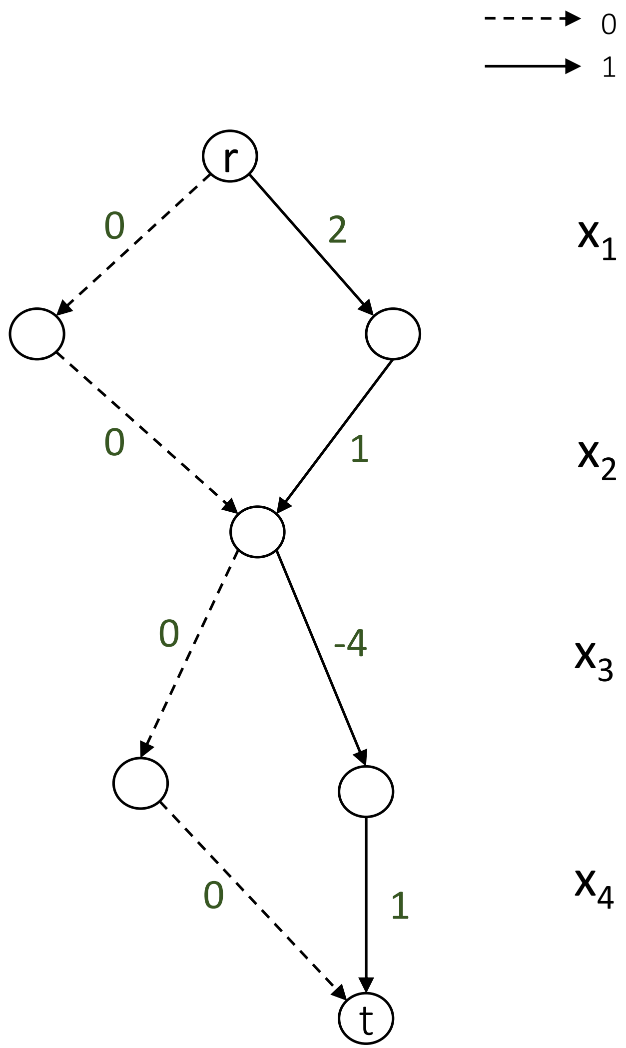


- Maximizing a linear (or separable) function:
  - Arc lengths: contribution to the objective
  - Longest path: optimal solution

Decision Diagrams

- Uses of this framework:
  - Solution counting (Lobbing'96)
  - Large-scale network flows **(Hachtel et al'97)**
  - Postoptimality analysis (Hadzic & Hooker'08)
  - Few others, typically **domain-specific**.

- Our goal: exploit the use of decision diagrams in **generic optimization methods**



Relaxation Methods

E.g., Linear Programming Relaxation

Modeling Framework

E.g., Linear Inequalities

Primal Heuristics

E.g., Feasibility Pump

Generic Optimization Techniques

E.g., Mixed-integer Programming

Inference

E.g., valid cuts

Search

E.g., Branch and bound

Relaxation Methods

E.g., Linear Programming Relaxation

Modeling Framework

E.g., Linear Inequalities

Primal Heuristics

E.g., Feasibility Pump

Generic Optimization Techniques

E.g., Mixed-integer Programming

Inference

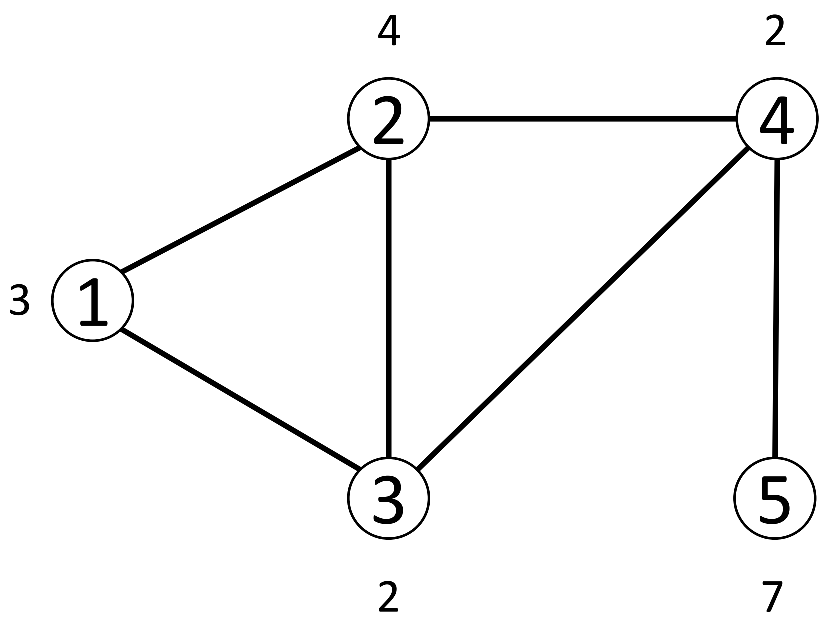
E.g., valid cuts

Search

E.g., Branch and bound

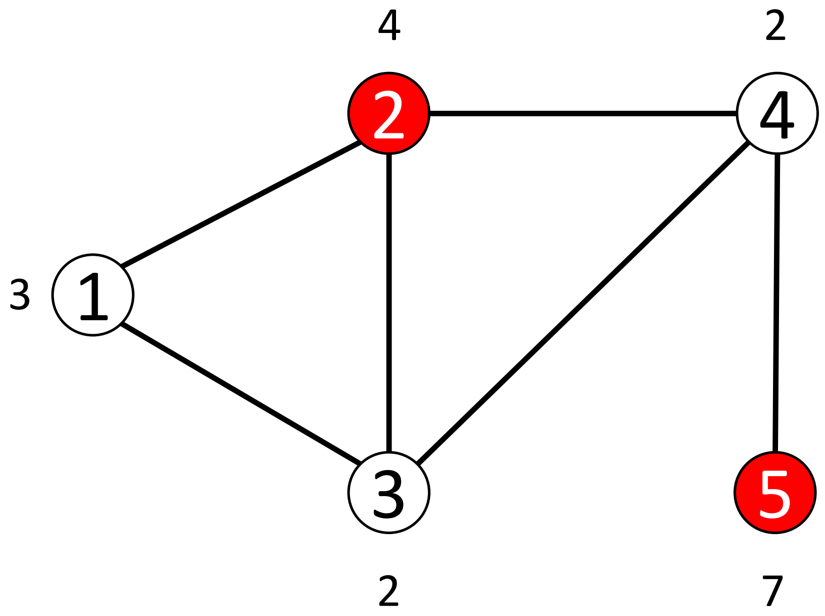
Modeling Framework

Ex.: Maximum independent set problem



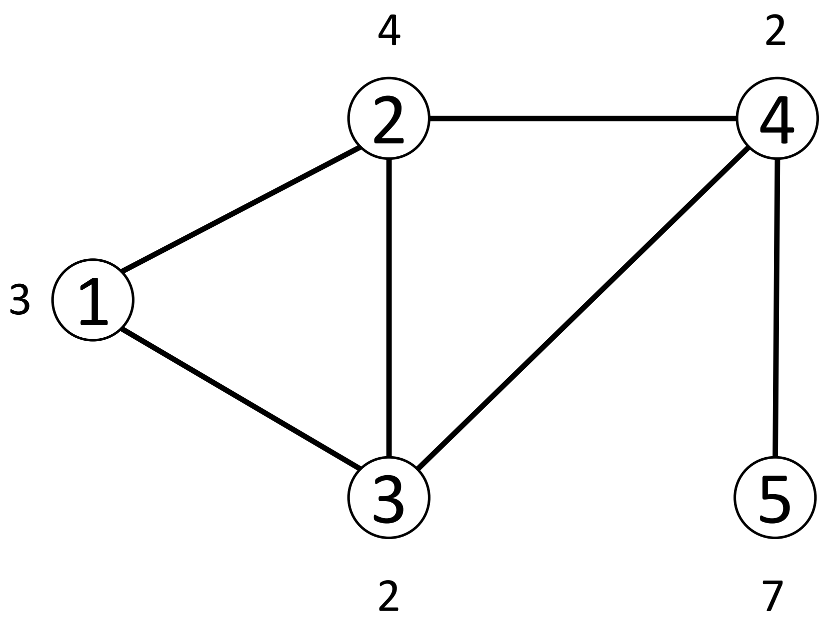
Modeling Framework

Ex.: Maximum independent set problem



Modeling Framework

Ex.: Maximum independent set problem



- Integer Programming Formulation:

max $3x_1 + 4x_2 + 2x_3 + 2x_4 + 7x_5$

subject to

$x_1 + x_2 \leq 1$

$x_1 + x_3 \leq 1$

$x_2 + x_3 \leq 1$

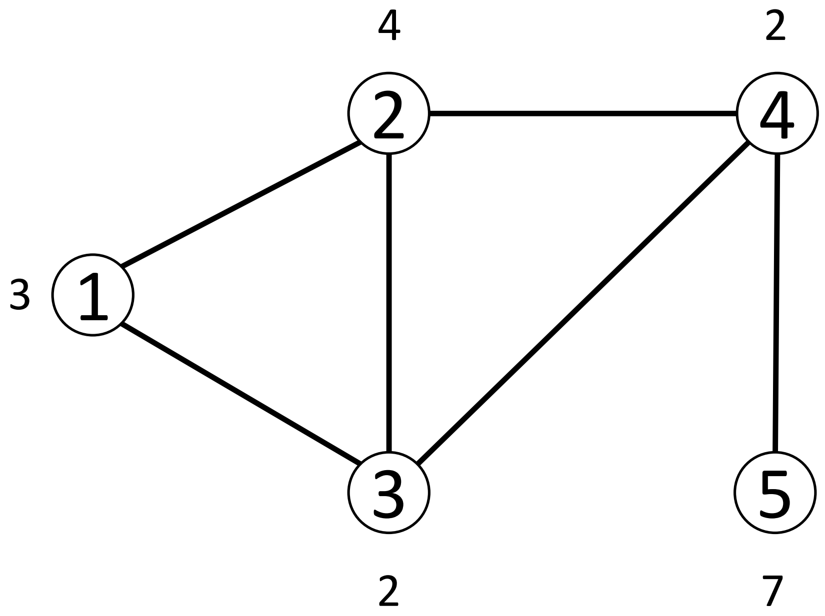
$x_3 + x_4 \leq 1$

$x_4 + x_5 \leq 1$

$x_1, x_2, x_3, x_4, x_5 \in \{0,1\}$

Modeling Framework

Ex.: Maximum independent set problem



- Our model: **Dynamic Programming**
  - Exploit *recursiveness*
  - Model is formulated through **states**
  - **Decisions** (or *controls):* define state transitions

- Decision diagram: **State-Transition Graph**
  - **Nodes** corresponds to **states**
  - **Arcs** are **state transitions**
  - **Arc weights** are **transition costs**

Modeling Framework

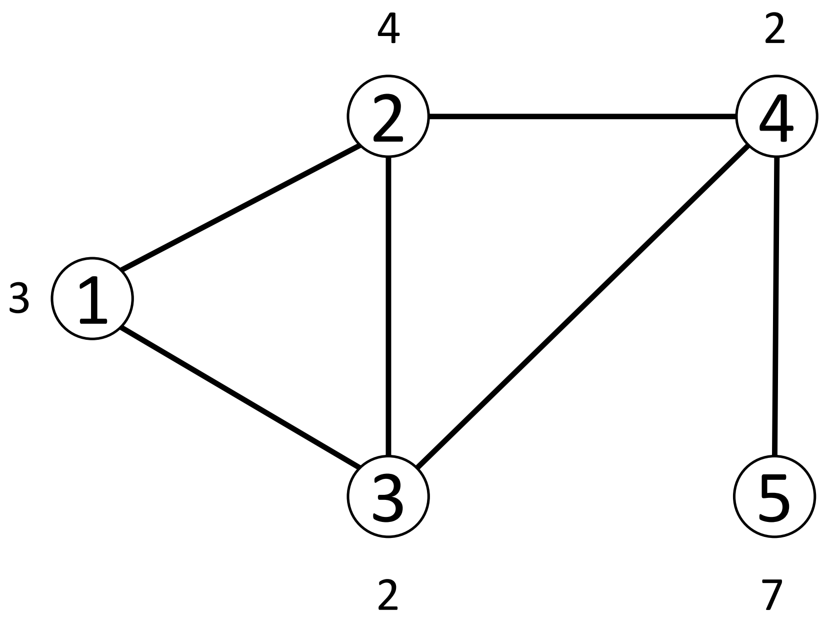
- DP model for the maximum independent set:
  - **State:** vertices that can be added to an independent set (**eligible vertices**)
  - **Decision:** select or not a vertex i from the eligibility set

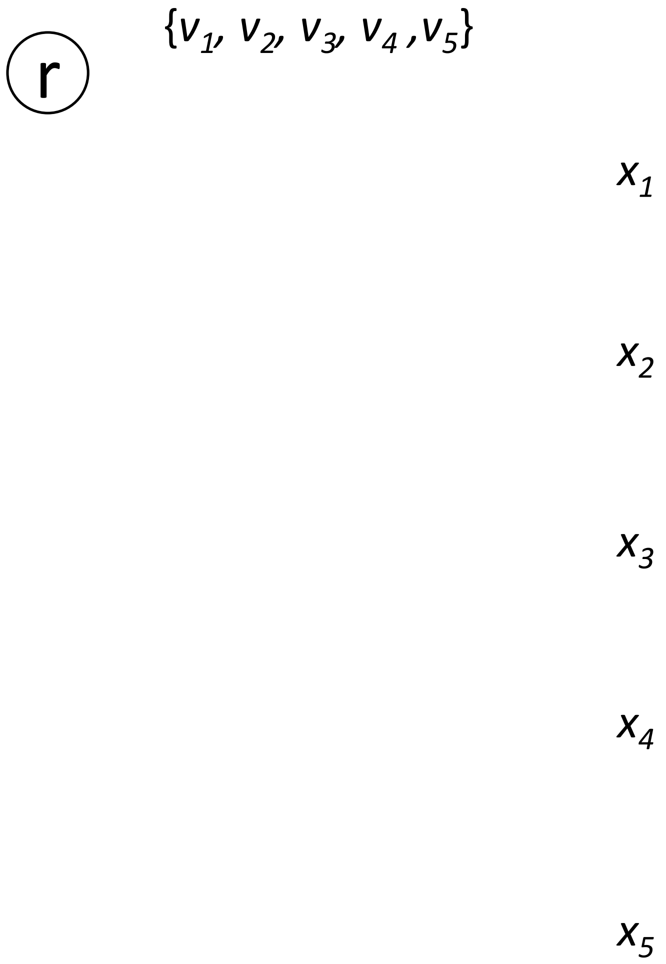
- Formal model*:*

$$V_i(S) = \begin{cases} max\left\{V_{i-1}(S \setminus \{i\}), V_{i-1}\big(S \setminus N(i)\big) + 1\right\}, & i \in S \\ V_{i-1}\big(S \setminus N(i)\big), & o.w. \end{cases}$$

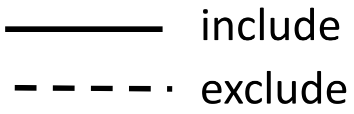
$$V_i(\emptyset) = 0, \qquad i = 1, \ldots, 5$$

Maximum Independent Set Problem

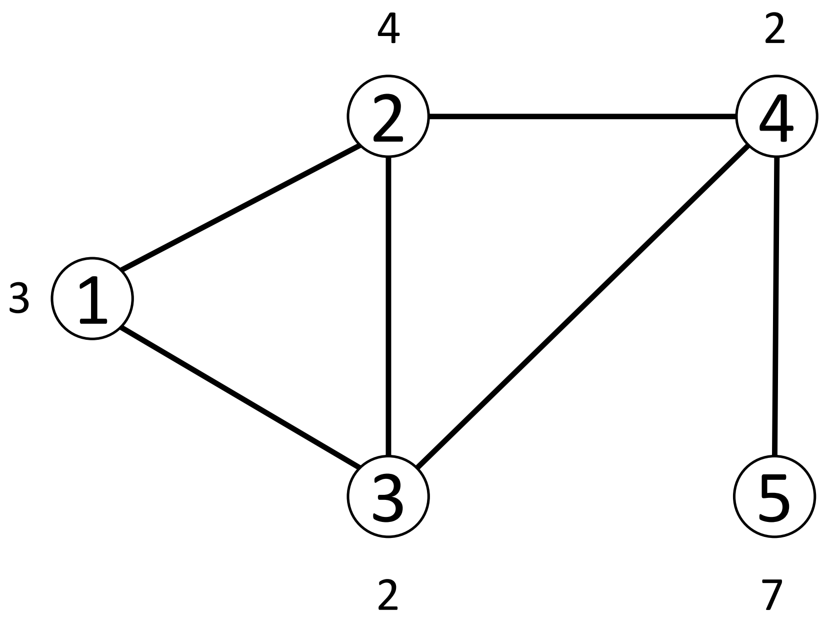


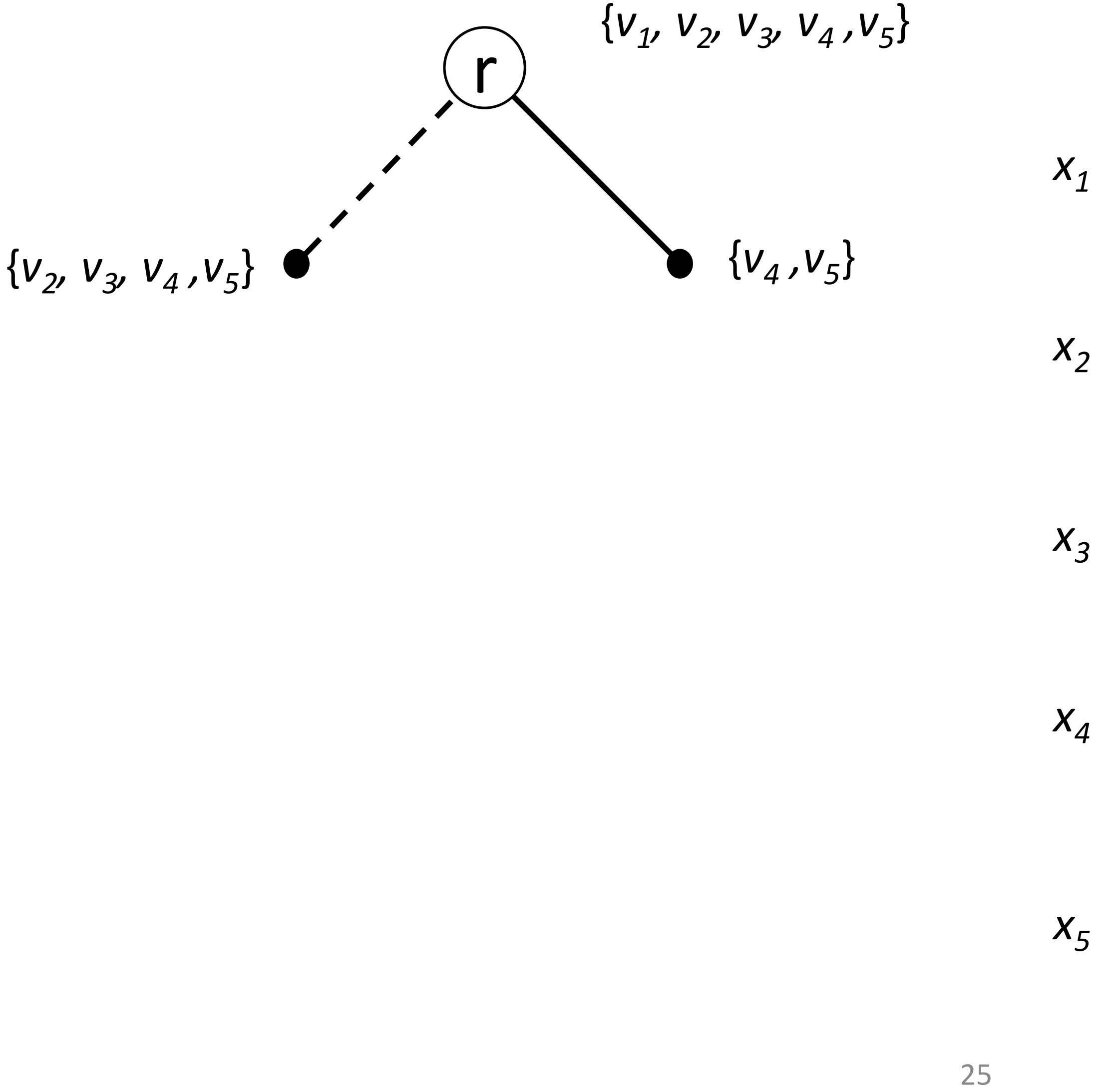


State: set of eligible vertices



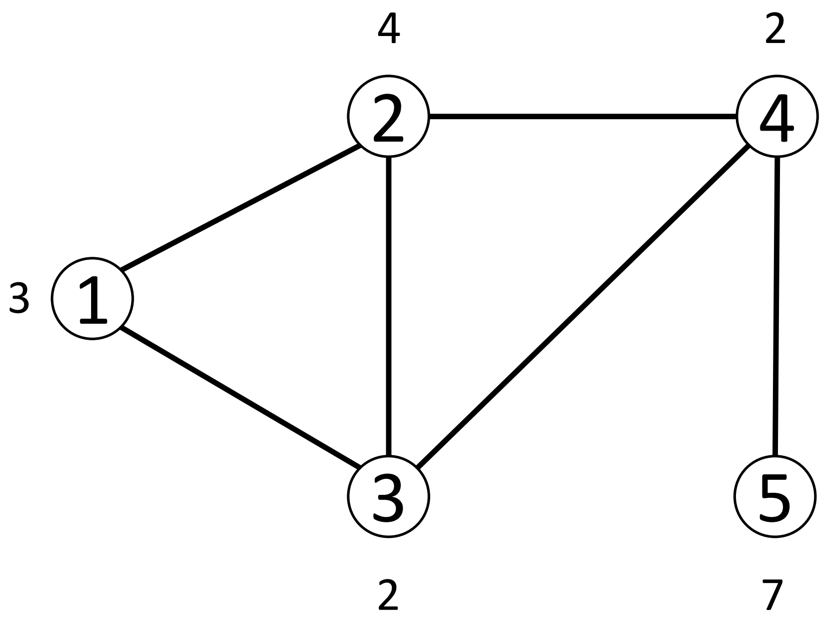
Maximum Independent Set Problem



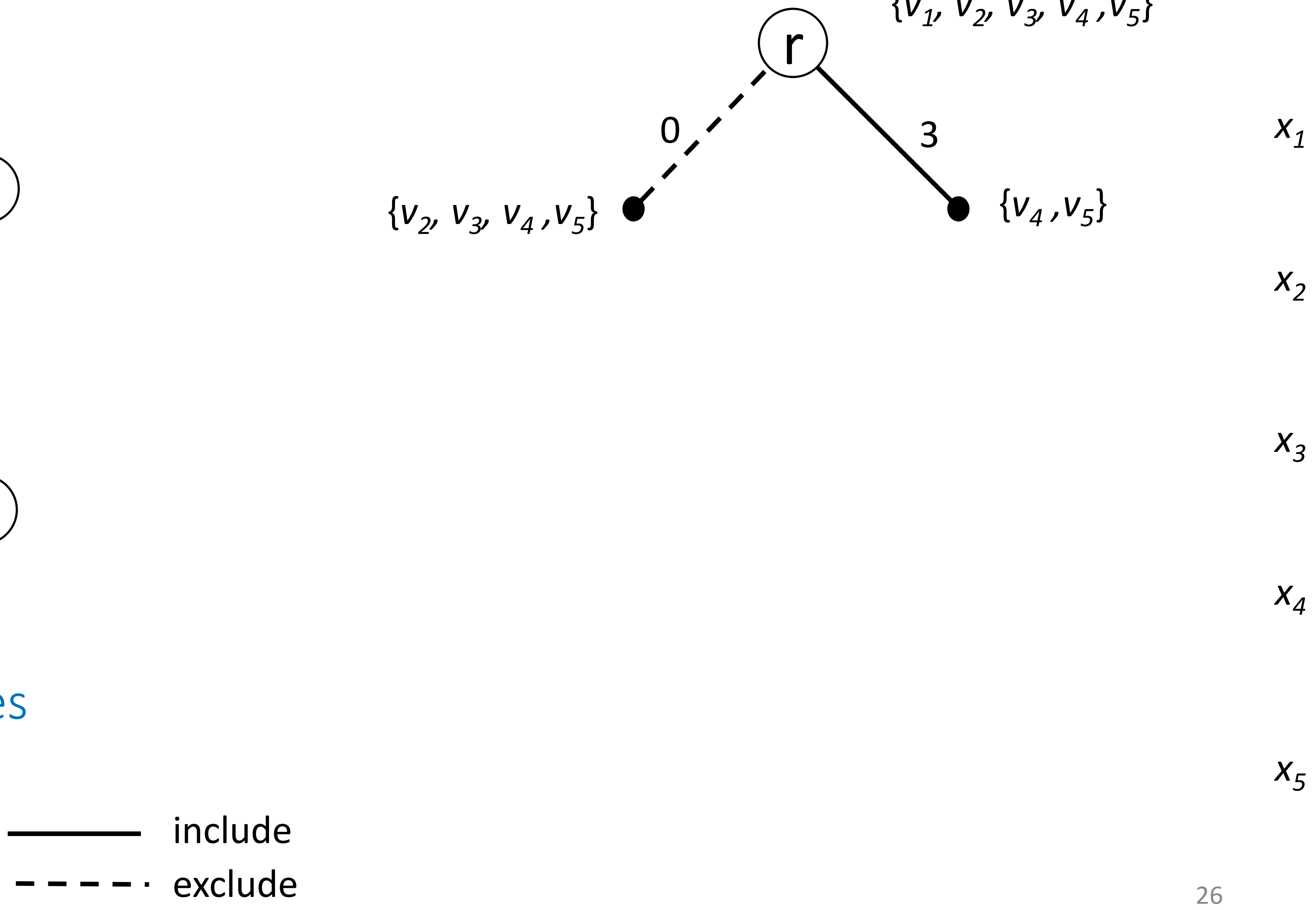


State: set of eligible vertices

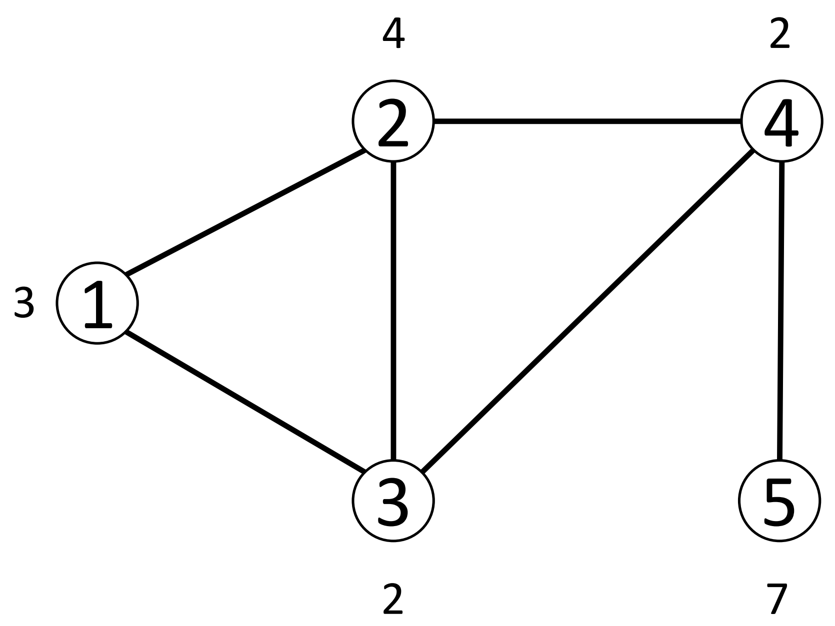
Maximum Independent Set Problem

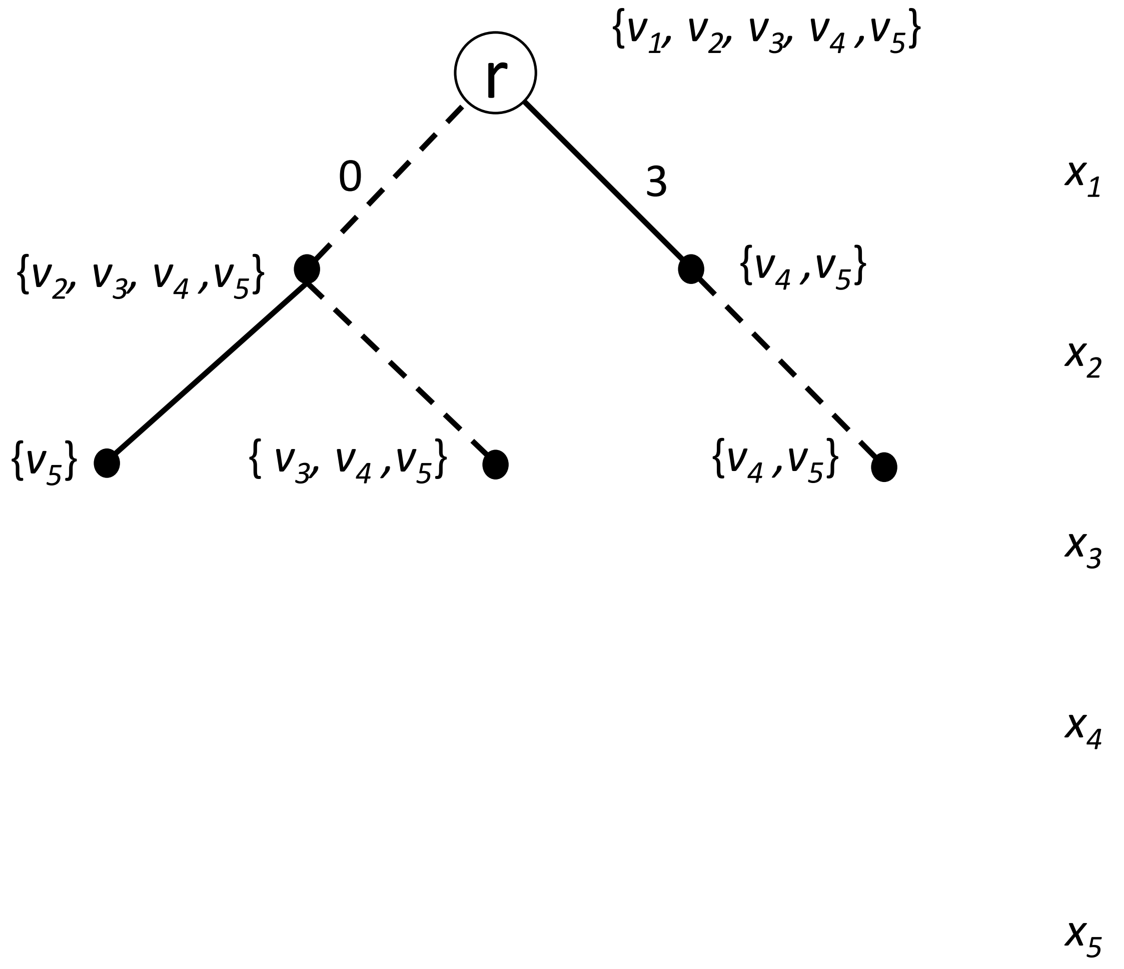


State: set of eligible vertices

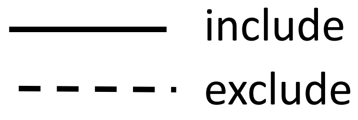


Maximum Independent Set Problem

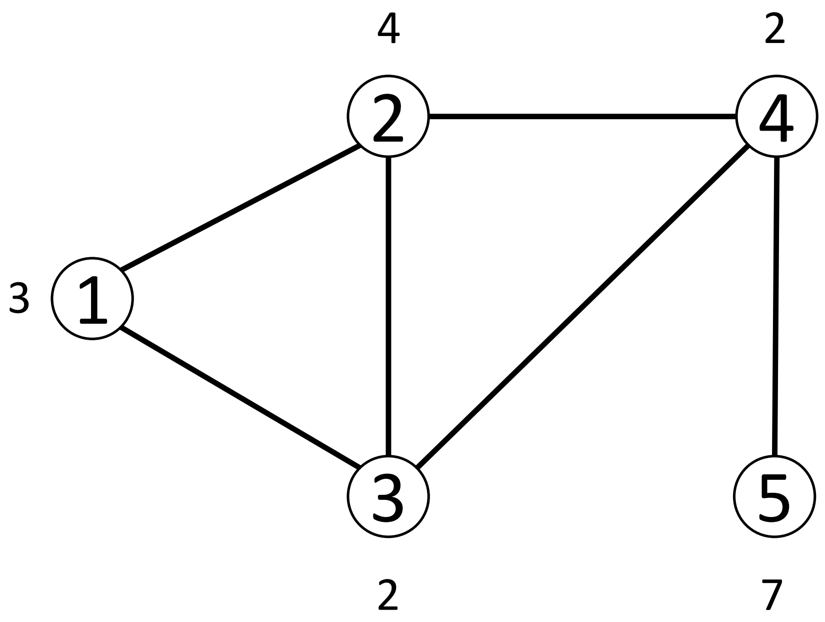


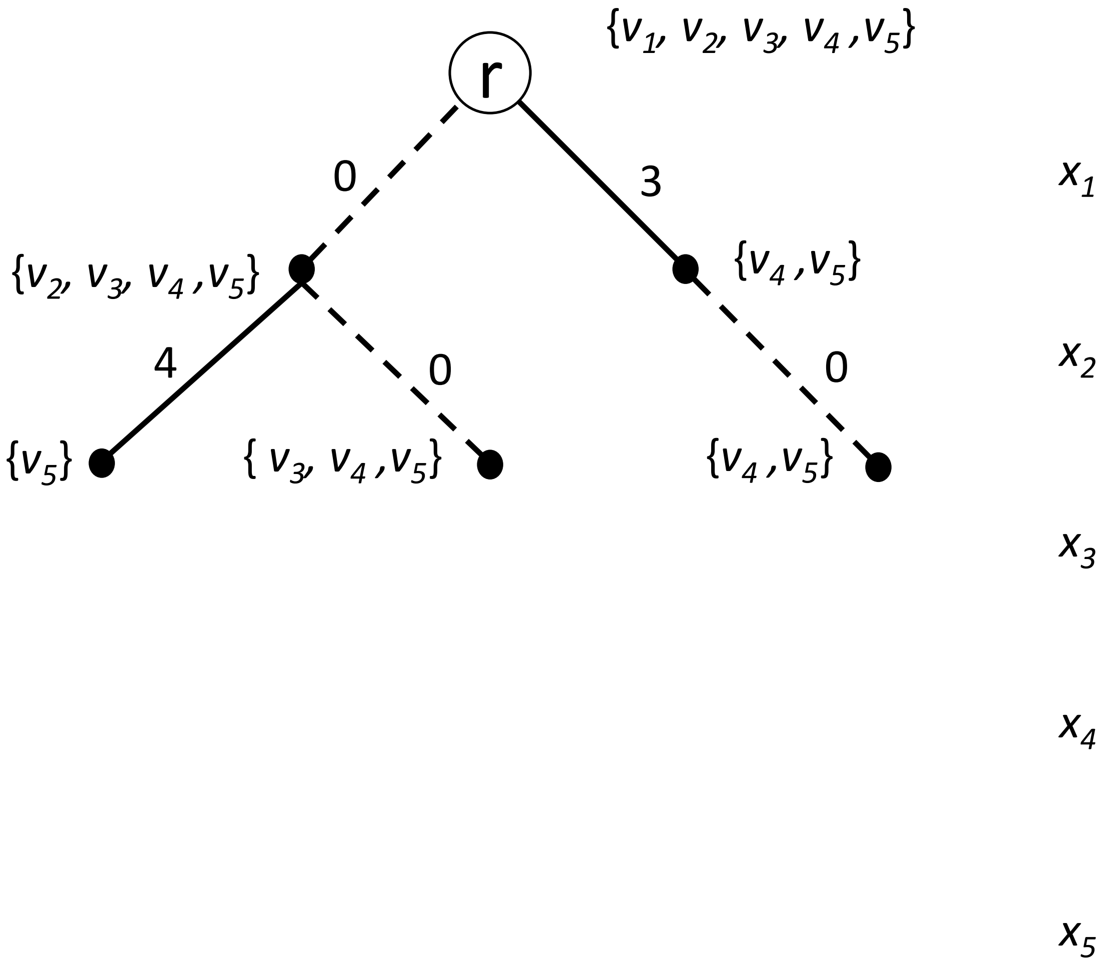


State: set of eligible vertices

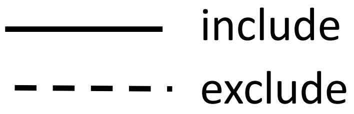


Maximum Independent Set Problem

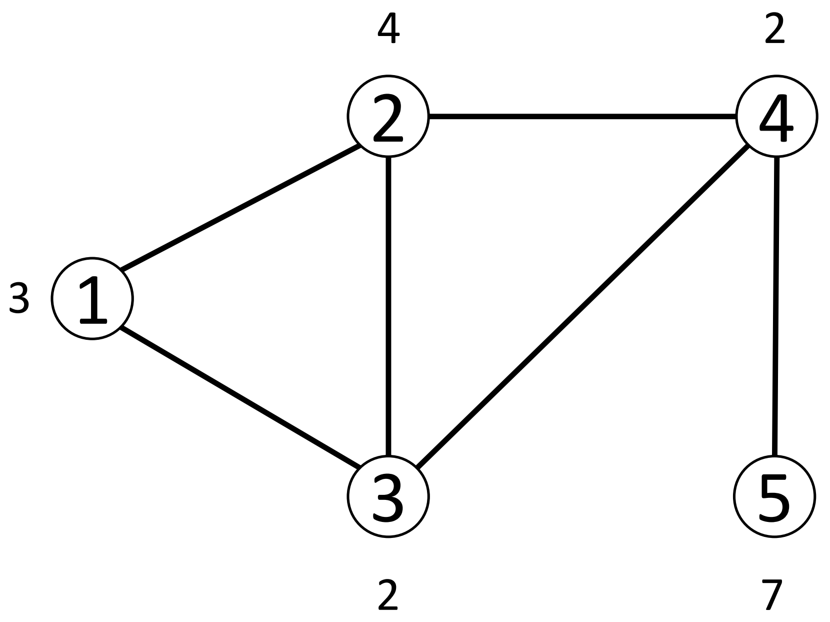




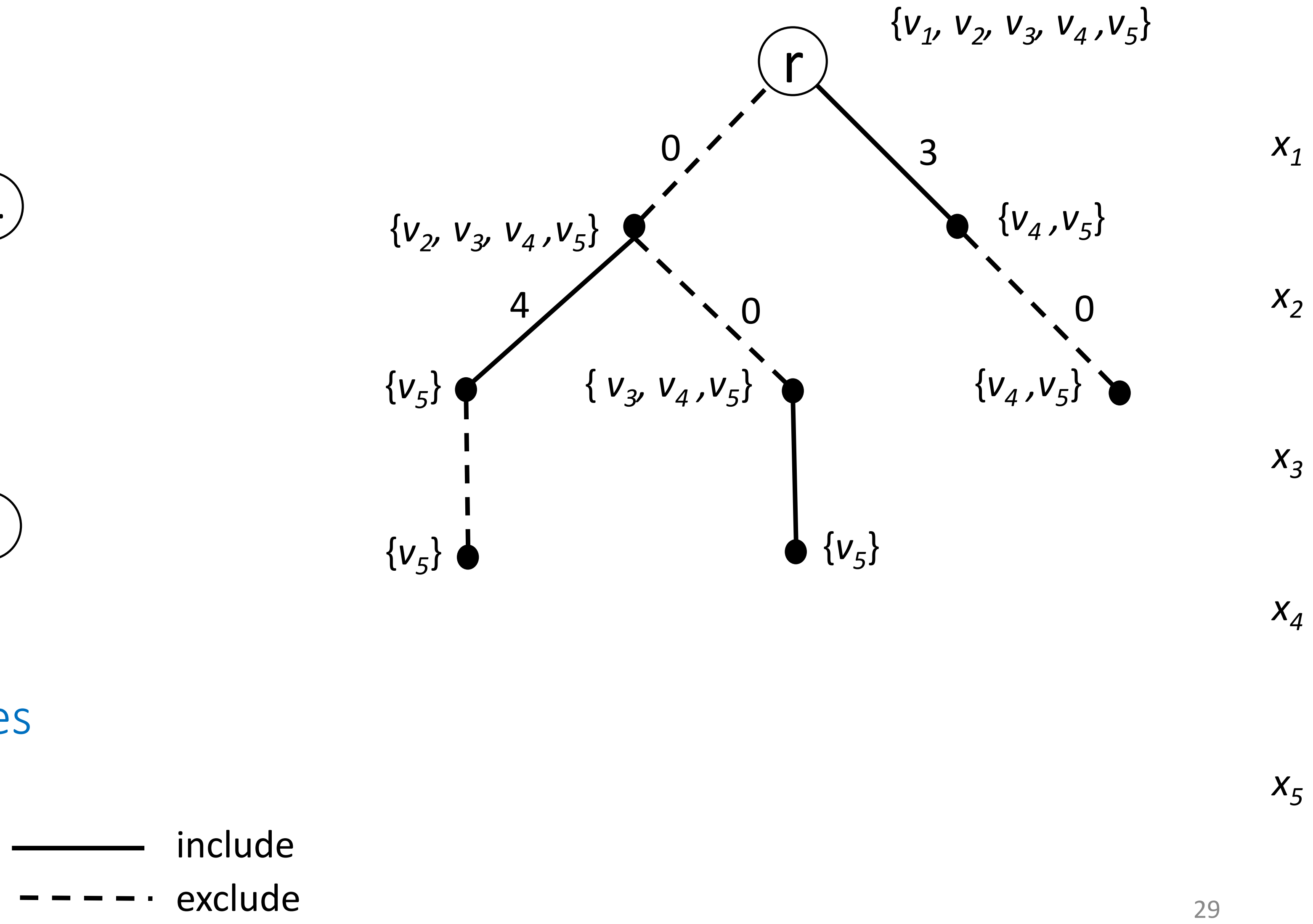
State: set of eligible vertices



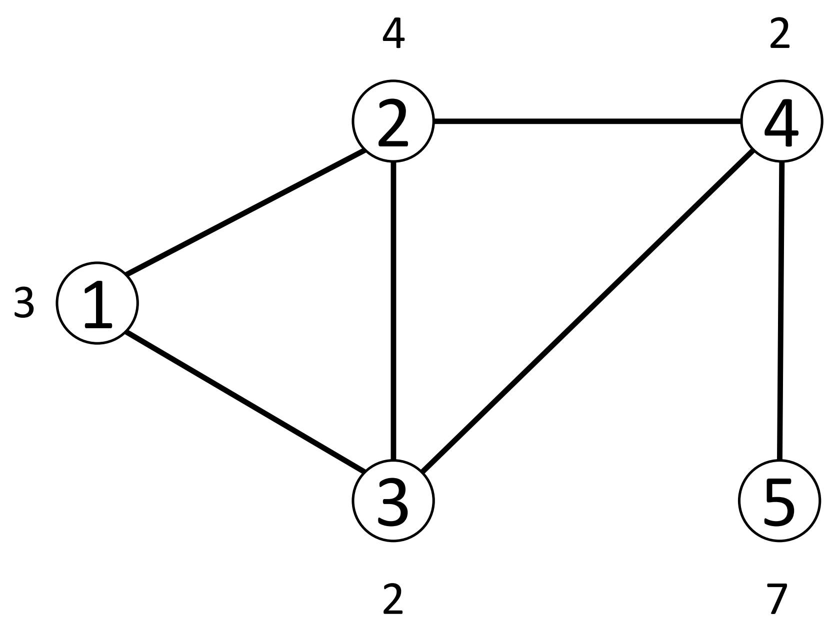
Maximum Independent Set Problem

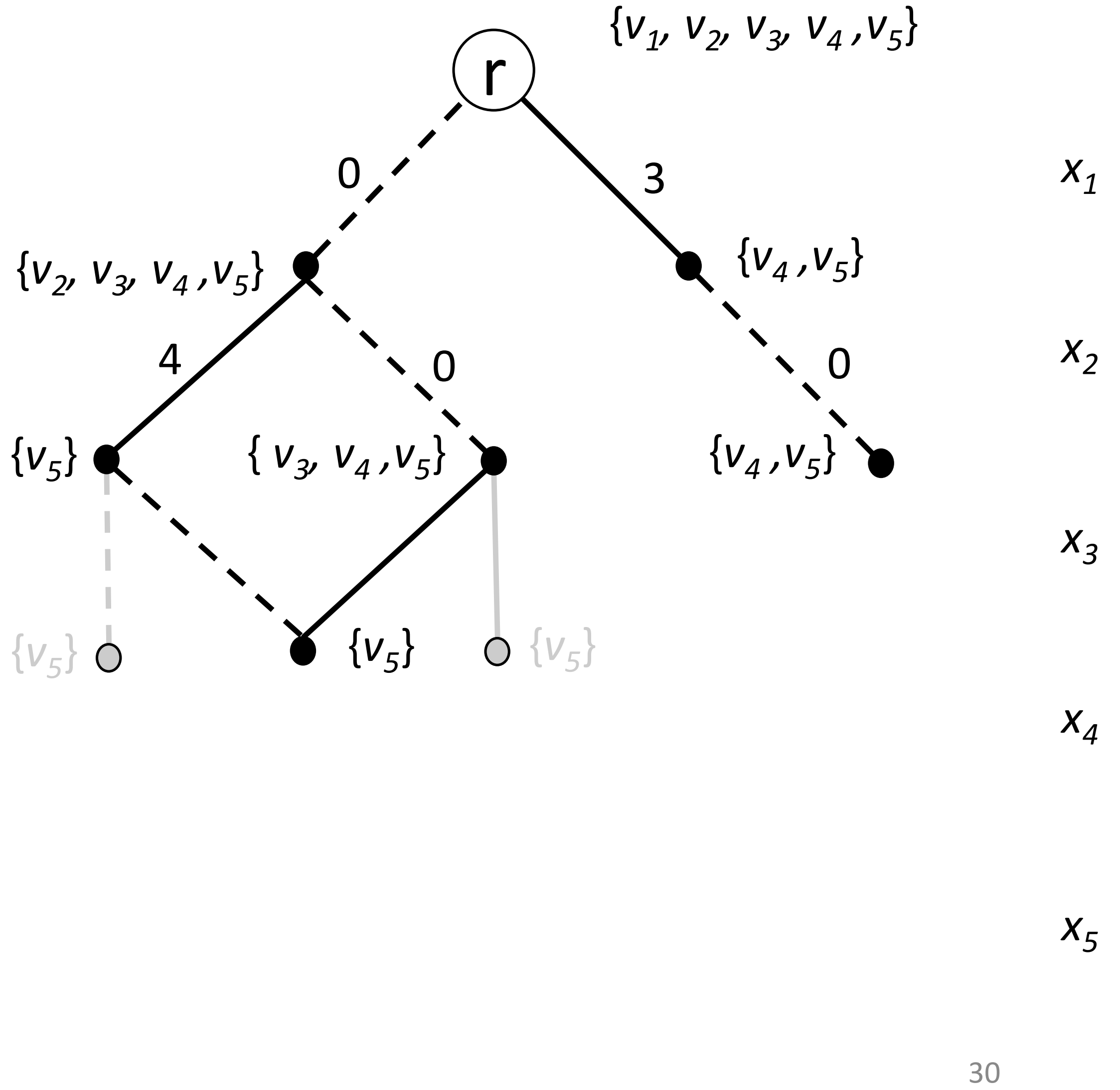


State: set of eligible vertices



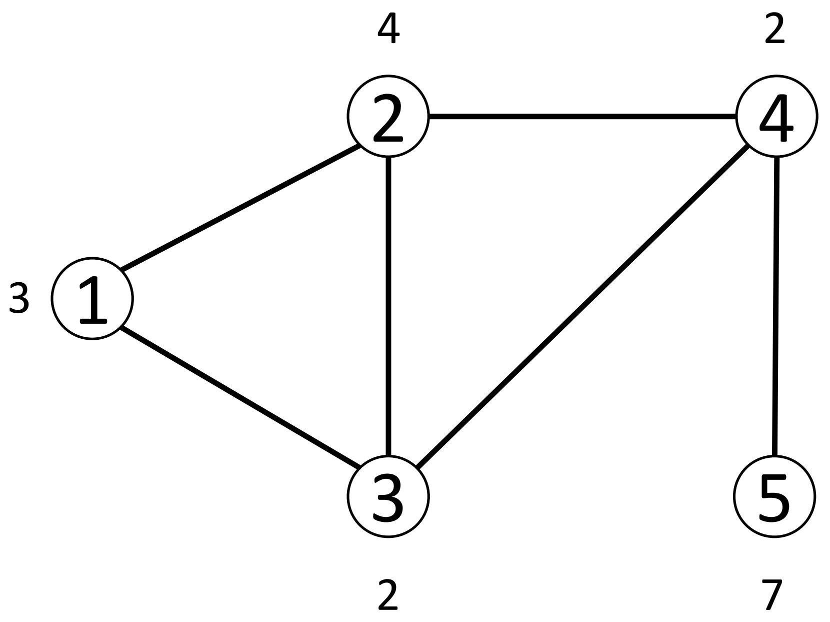
Maximum Independent Set Problem



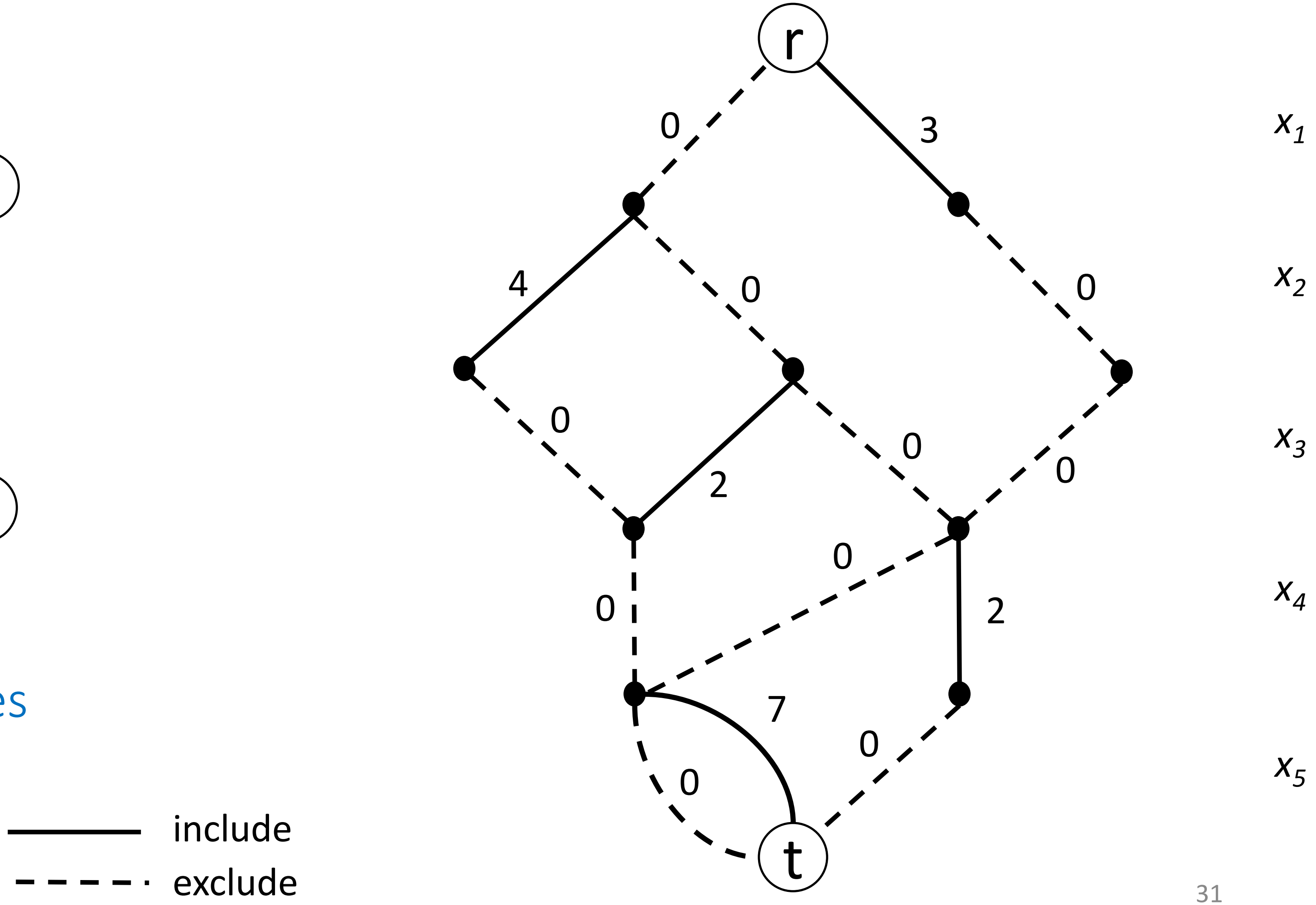


State: set of eligible vertices

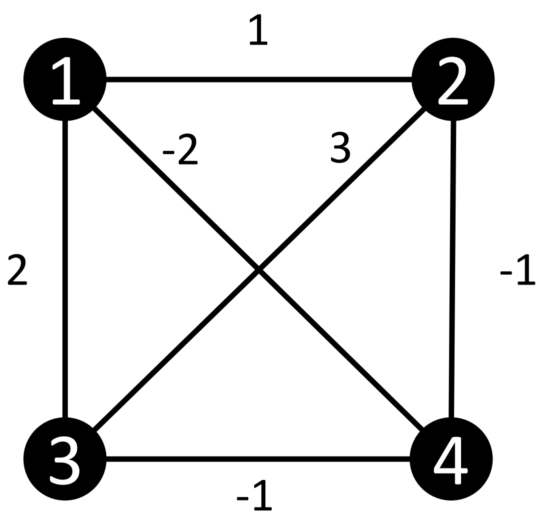
Maximum Independent Set Problem

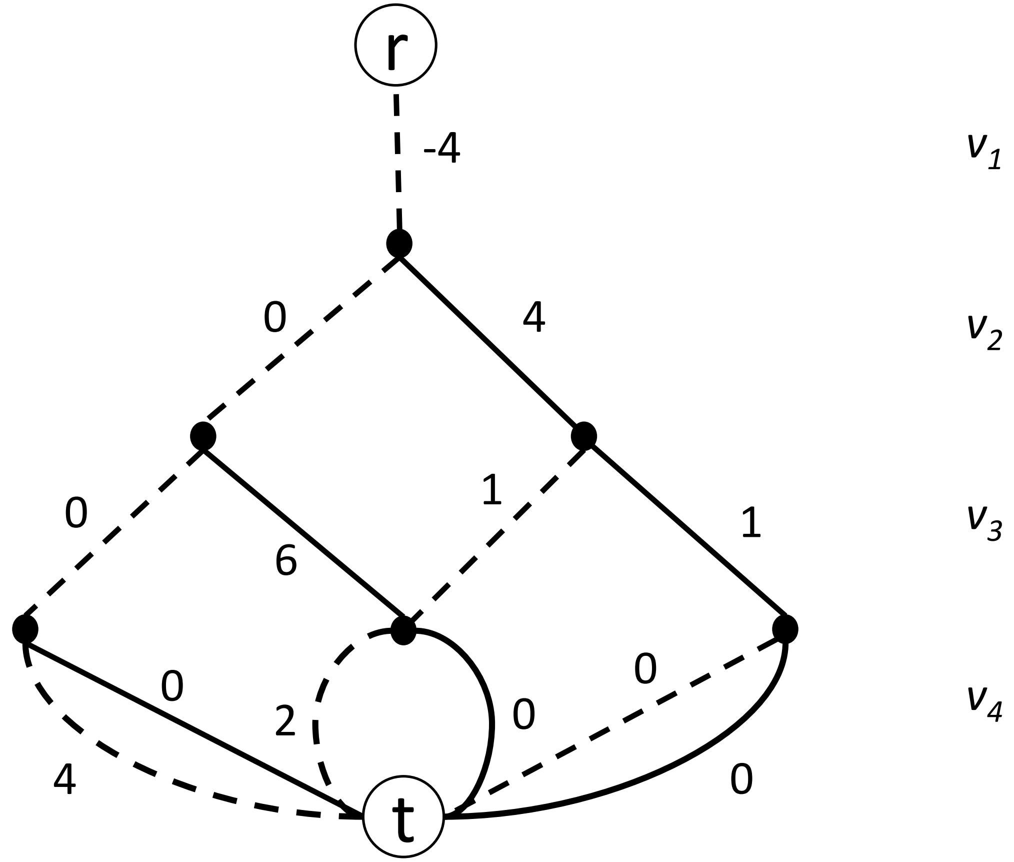


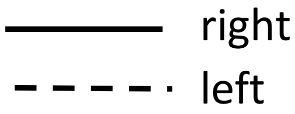
State: set of eligible vertices



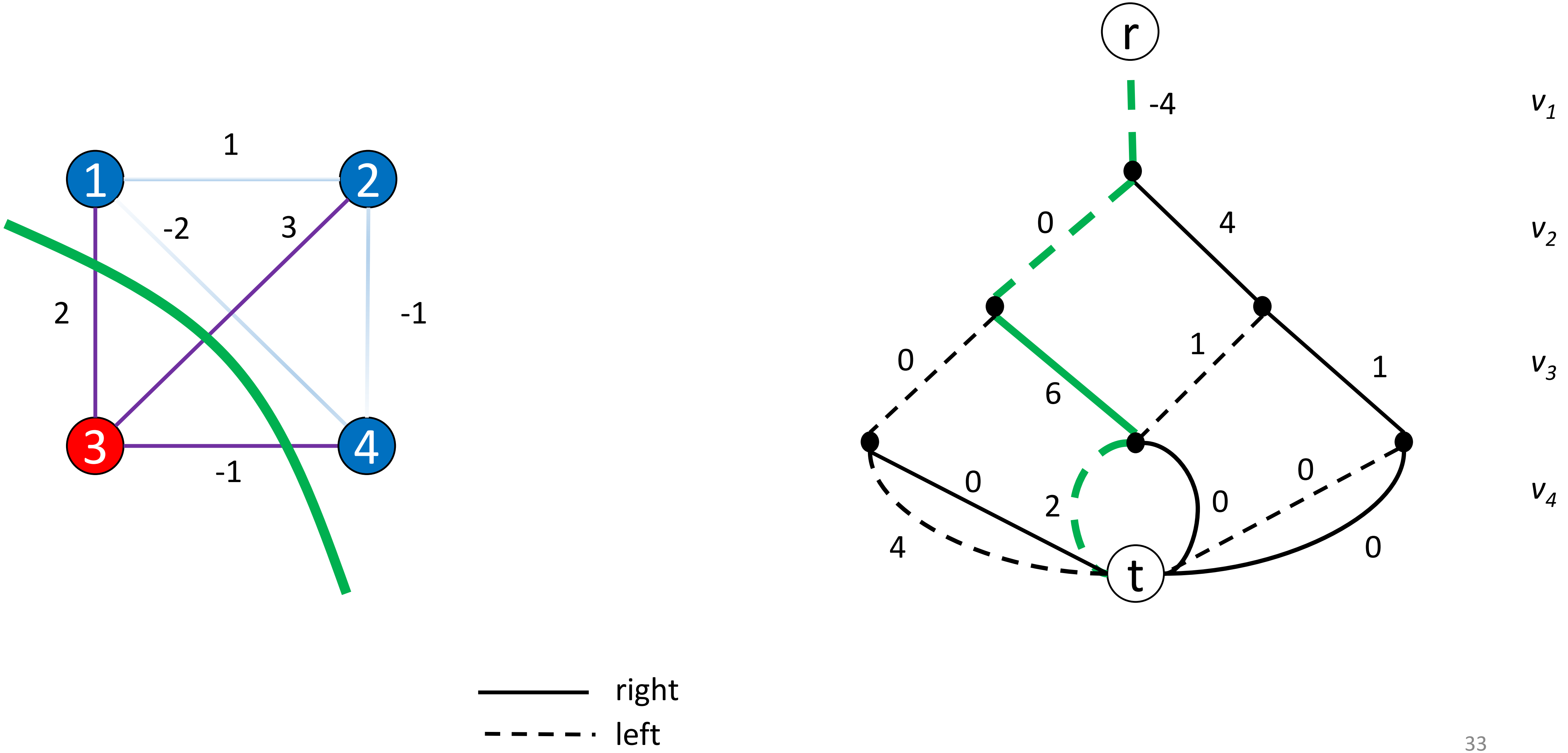
Other Example: Maximum Cut Problem







Other Example: Maximum Cut Problem



Some quick observations

- Variable ordering plays a big role on **size**
  - Closely connected to **treewidth** and **bandwidth**
  - Independent Set: polynomial for certain classes of graphs
  - TSP: parameterized-size depending on precendence relations

- In general, decision diagrams grow **exponentially** large
  - Proof: Extended Formulations for the Independent Set Problem

Relaxation Methods

E.g., Linear Programming Relaxation

Modeling Framework

E.g., Linear Inequalities

Generic Optimization Techniques

E.g., Mixed-integer Programming

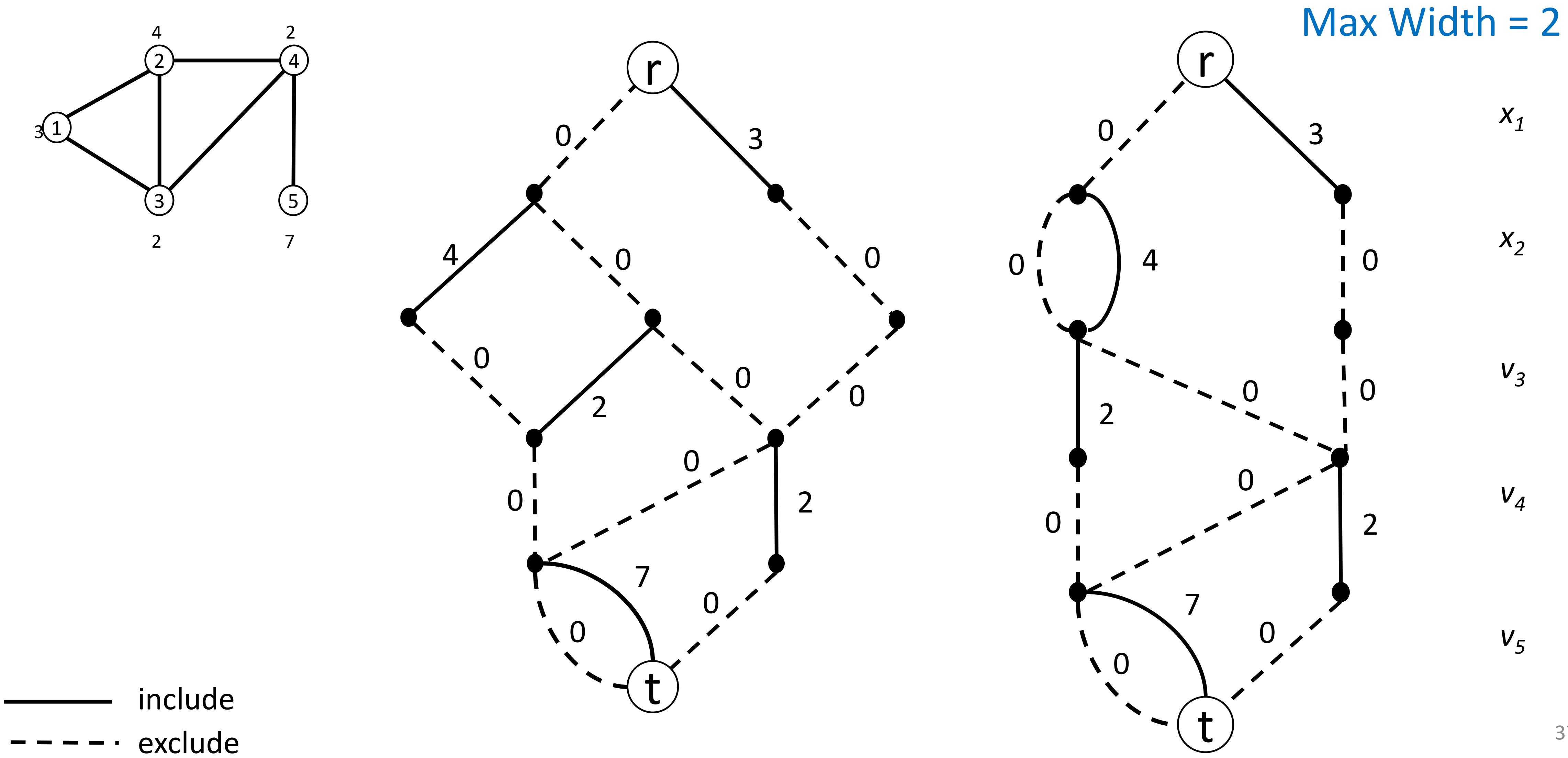
Relaxed Decision Diagrams

- In practice, we cannot work with exact diagrams

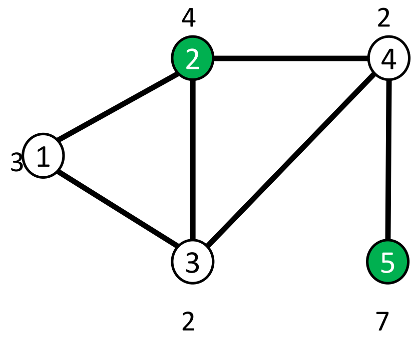
- Alternative: limit the size to **approximate** the feasible space
  - Parameter on the **width** of the diagram
  - *Relaxed Decision Diagrams:* Over-approximation

- Introduced by [Andersen et al'07]

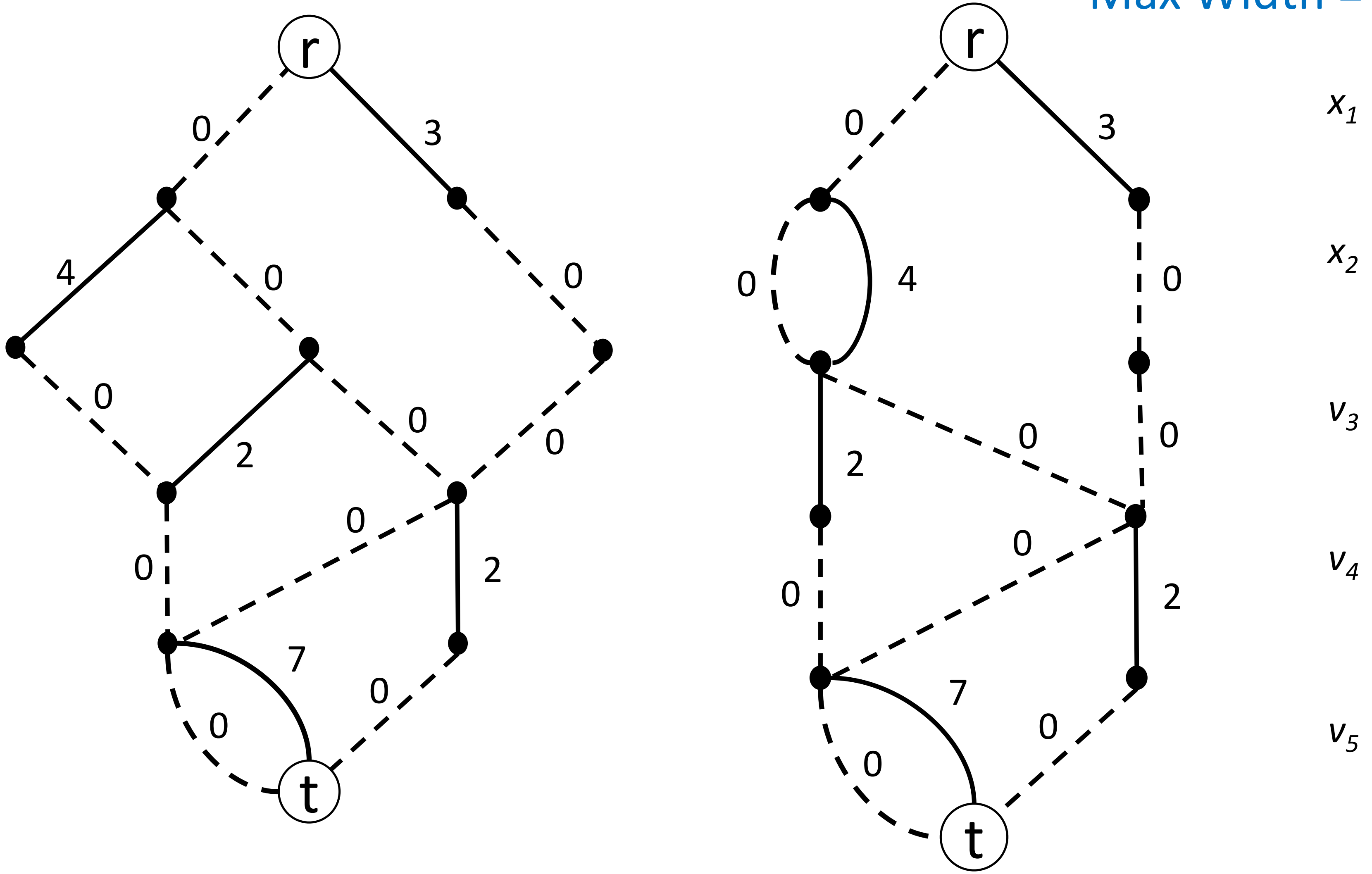
Relaxed Decision Diagrams



Relaxed Decision Diagrams



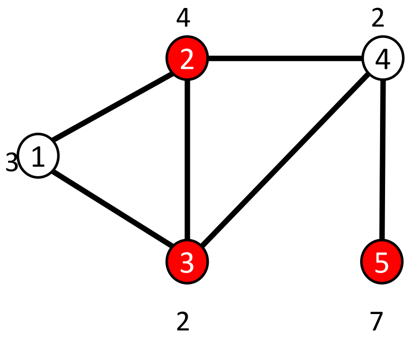
Max Width = 2

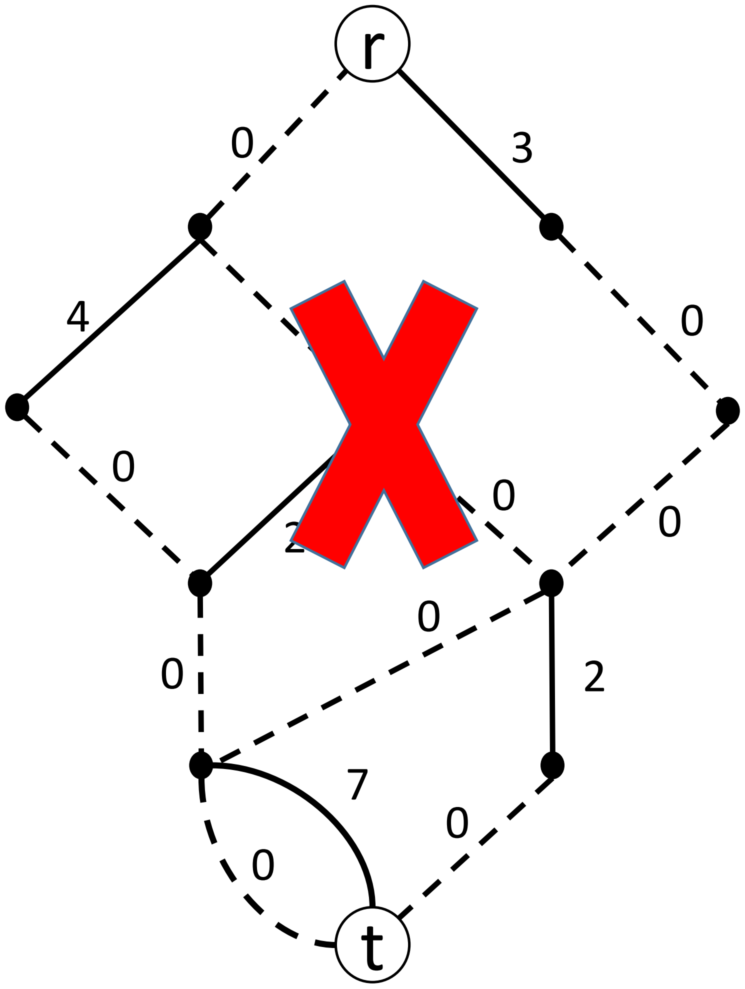


x = (0, 1, 0, 0, 1)
Solution value = 11

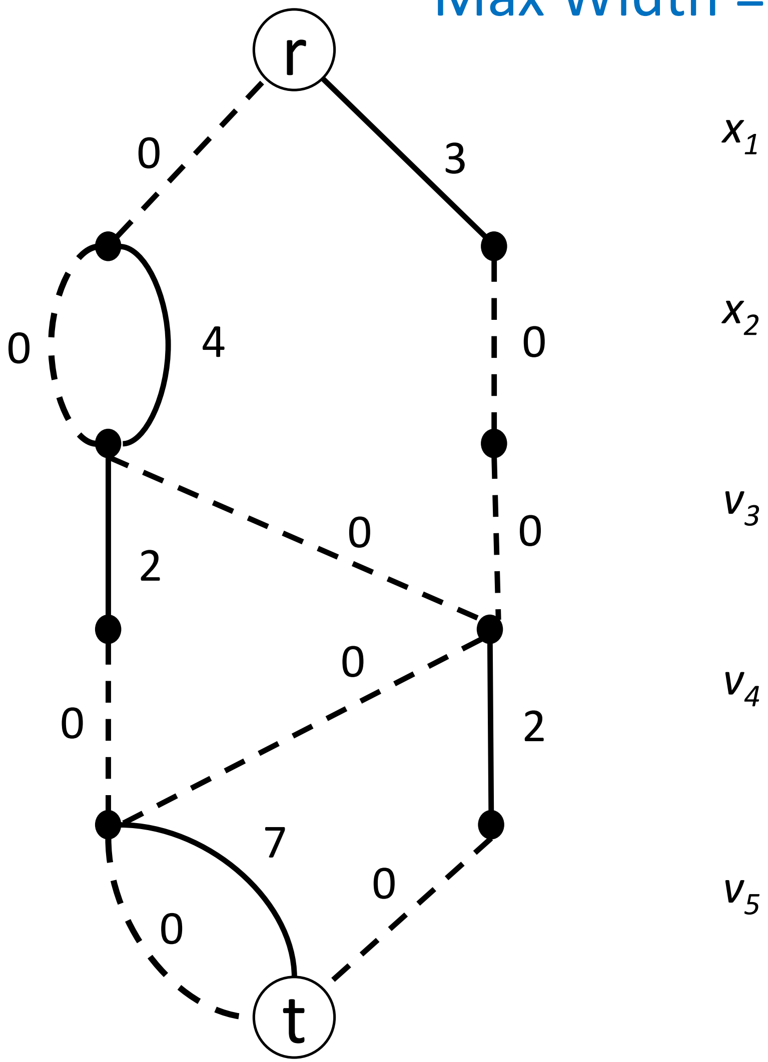
include
exclude

Relaxed Decision Diagrams





Max Width = 2



x = (0, 1, 1, 0, 1)
Upper bound = 13

include
exclude

Compiling Relaxed Decision Diagrams

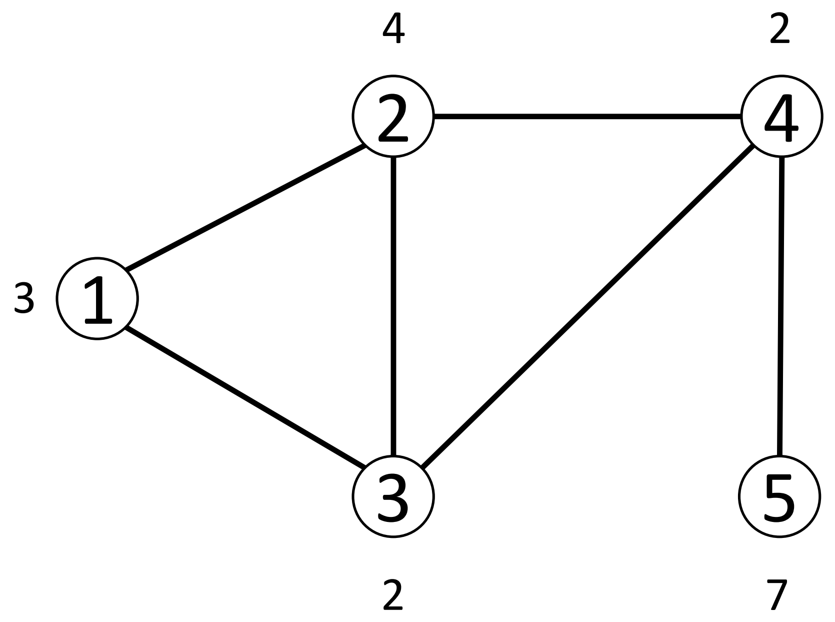
- Model is **augmented** with a **state agregation** operator
  - Recipe on how to merge nodes so that no feasible solution is lost

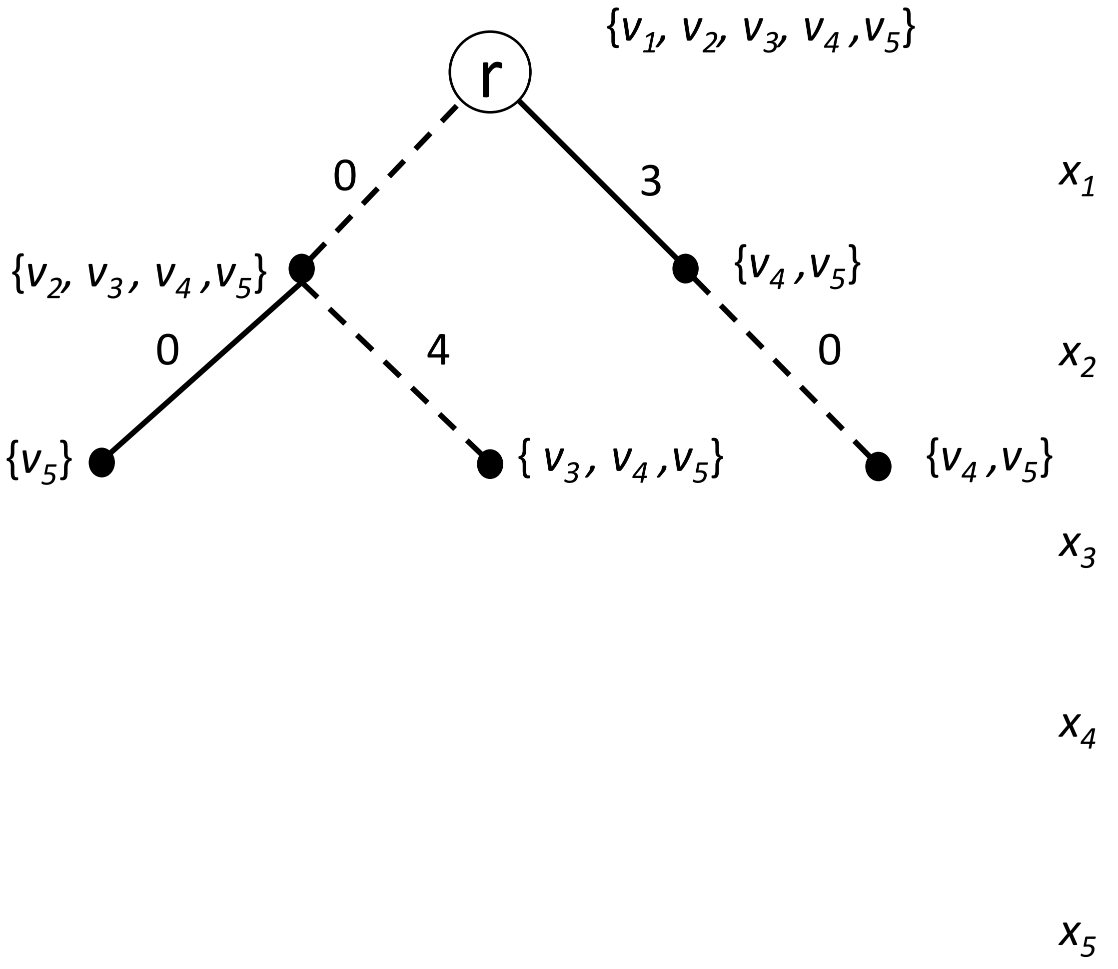
- $$V_i(S) = \begin{cases} max\,\{V_{i-1}(S \setminus \{i\}), V_{i-1}(S \setminus N(i)) + 1\}, & i \in S \\ V_{i-1}(S \setminus N(i)), & o.w. \end{cases}$$

  $$V_i(\emptyset) = 0, \qquad i = 1, \dots, 5$$

- $\Delta(S_1, S_2) = S_1 \cup S_2$

Building Relaxed Decision Diagrams

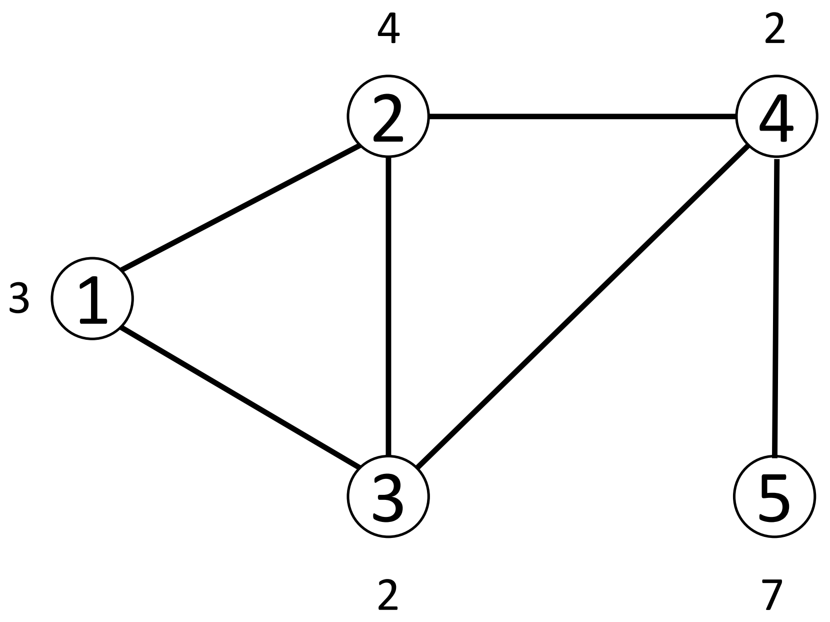




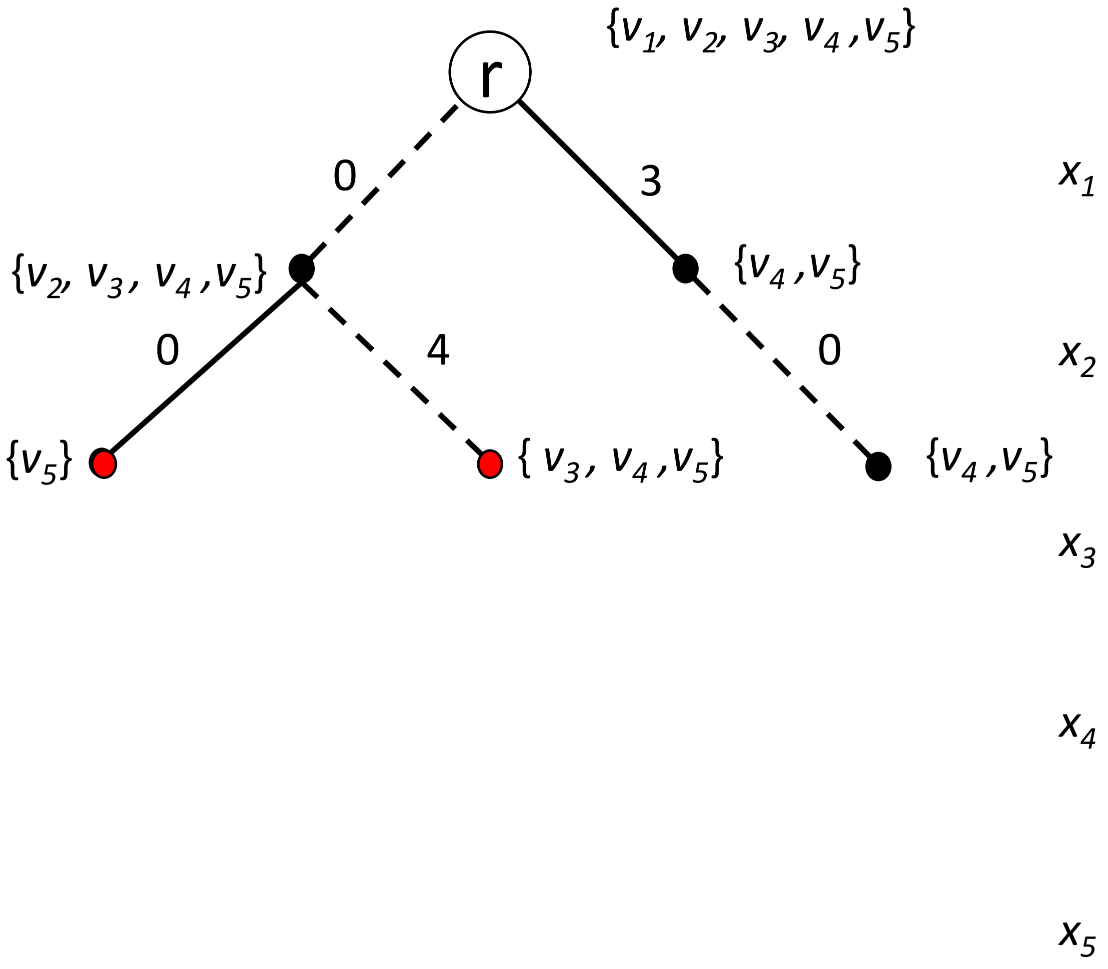
Max Width = 2

include
exclude

Building Relaxed Decision Diagrams



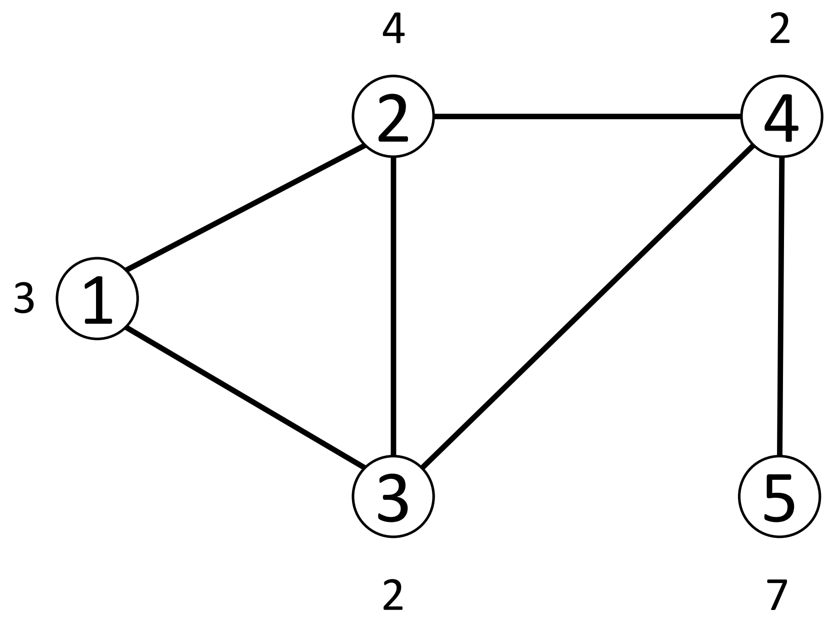
Max Width = 2



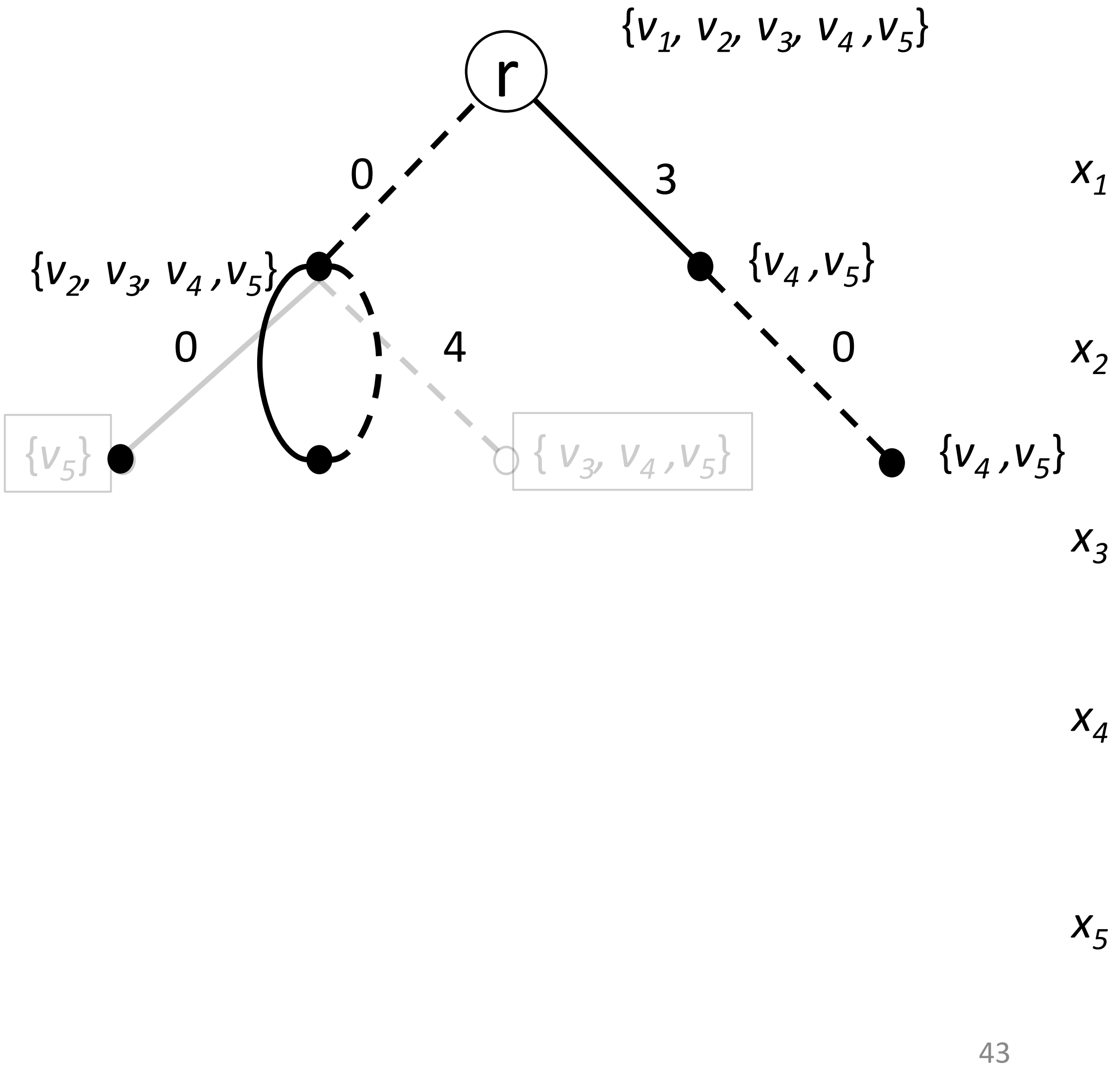
include

exclude

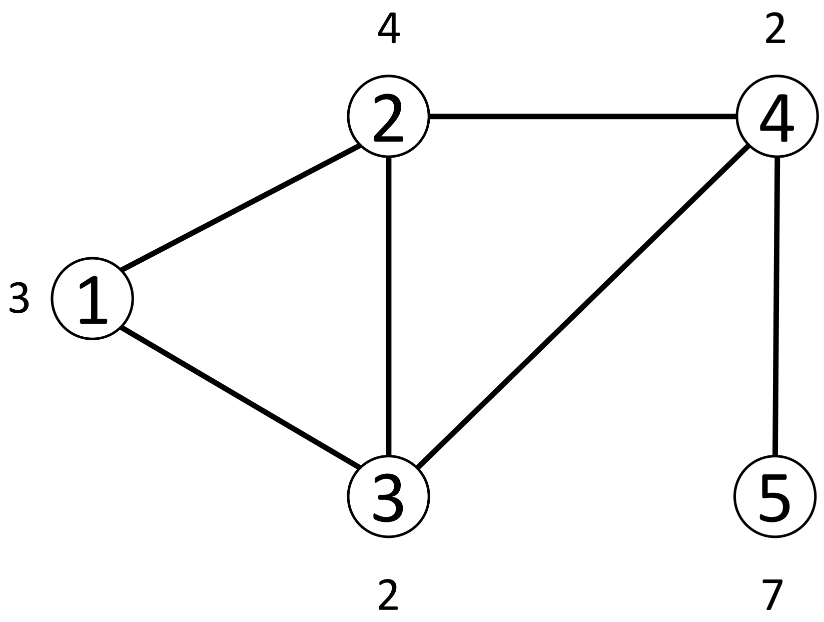
Building Relaxed Decision Diagrams



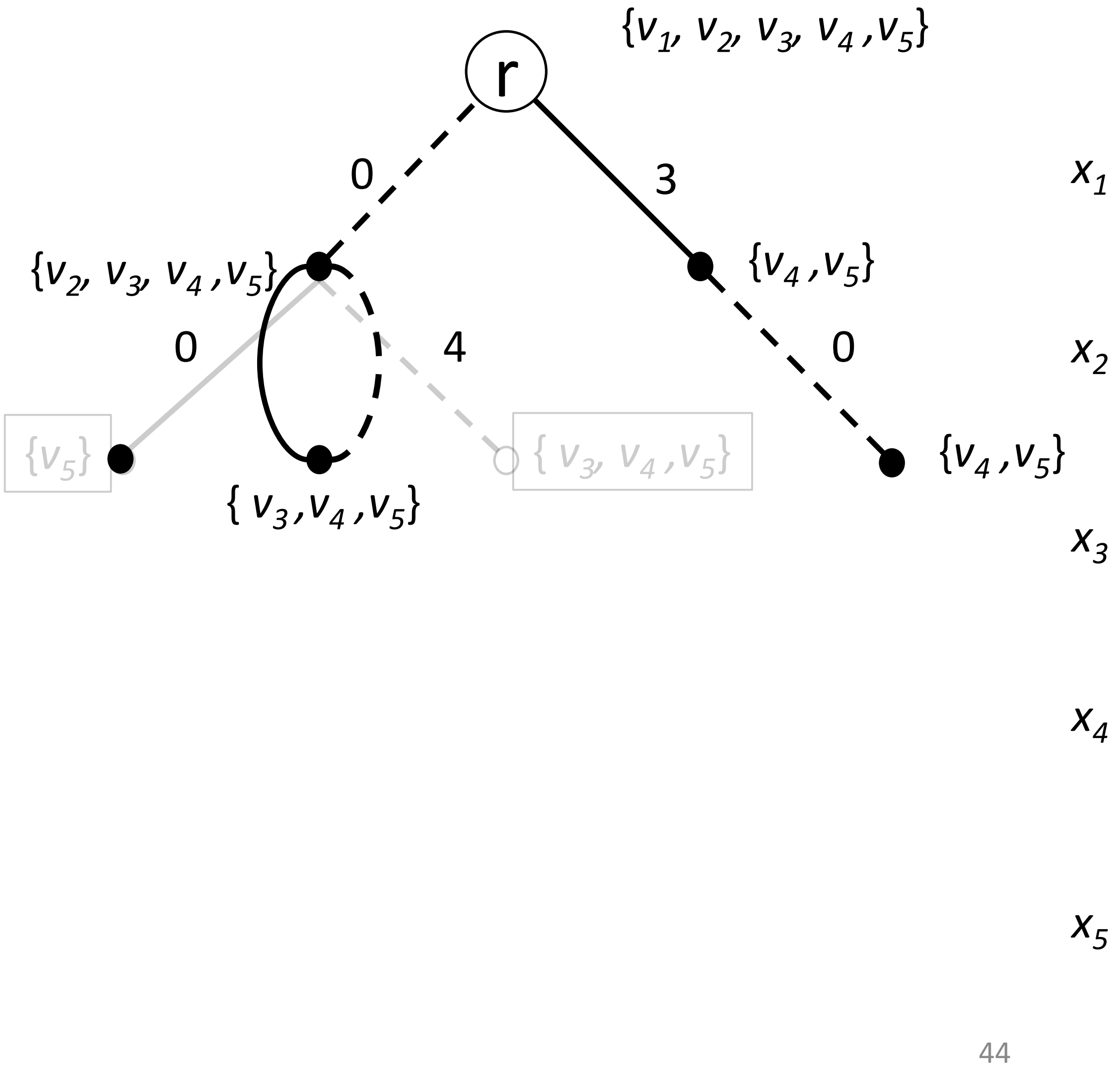
Max Width = 2



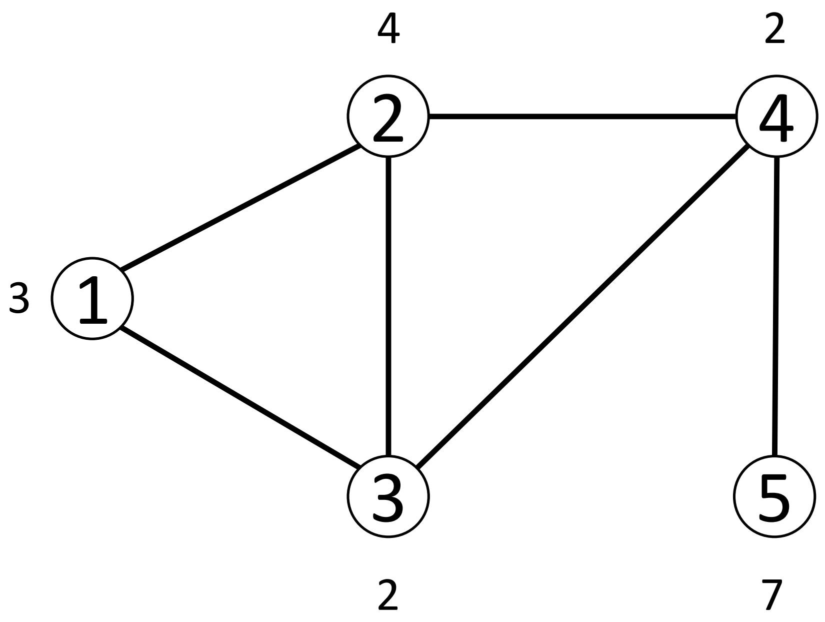
Building Relaxed Decision Diagrams



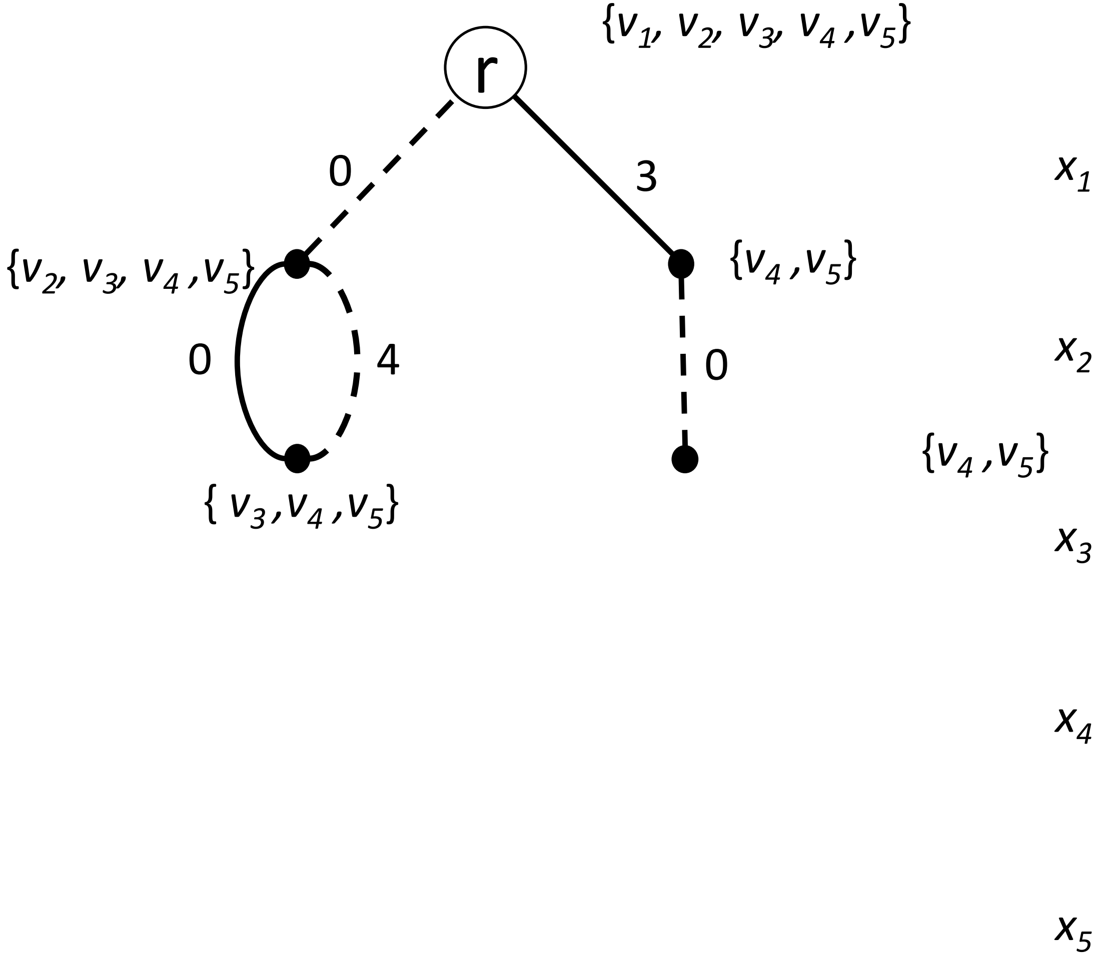
Max Width = 2



Building Relaxed Decision Diagrams

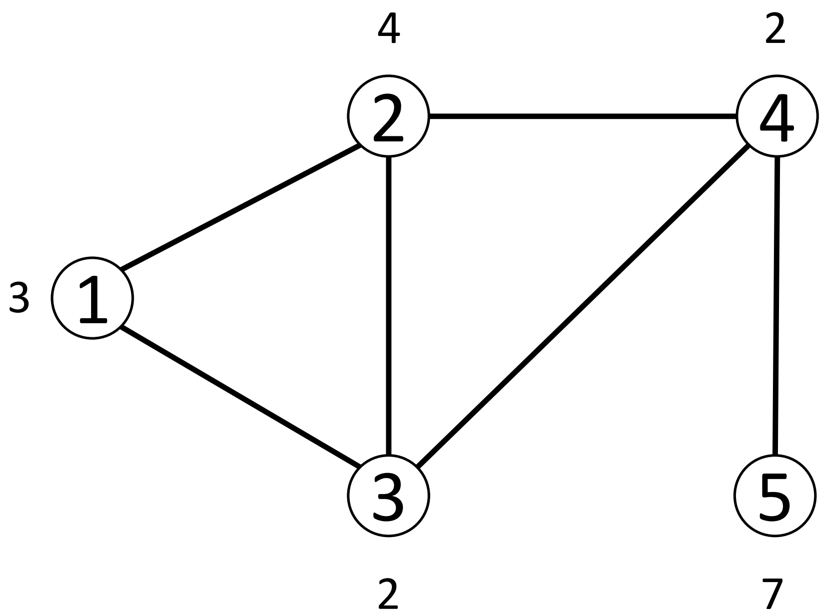


Max Width = 2

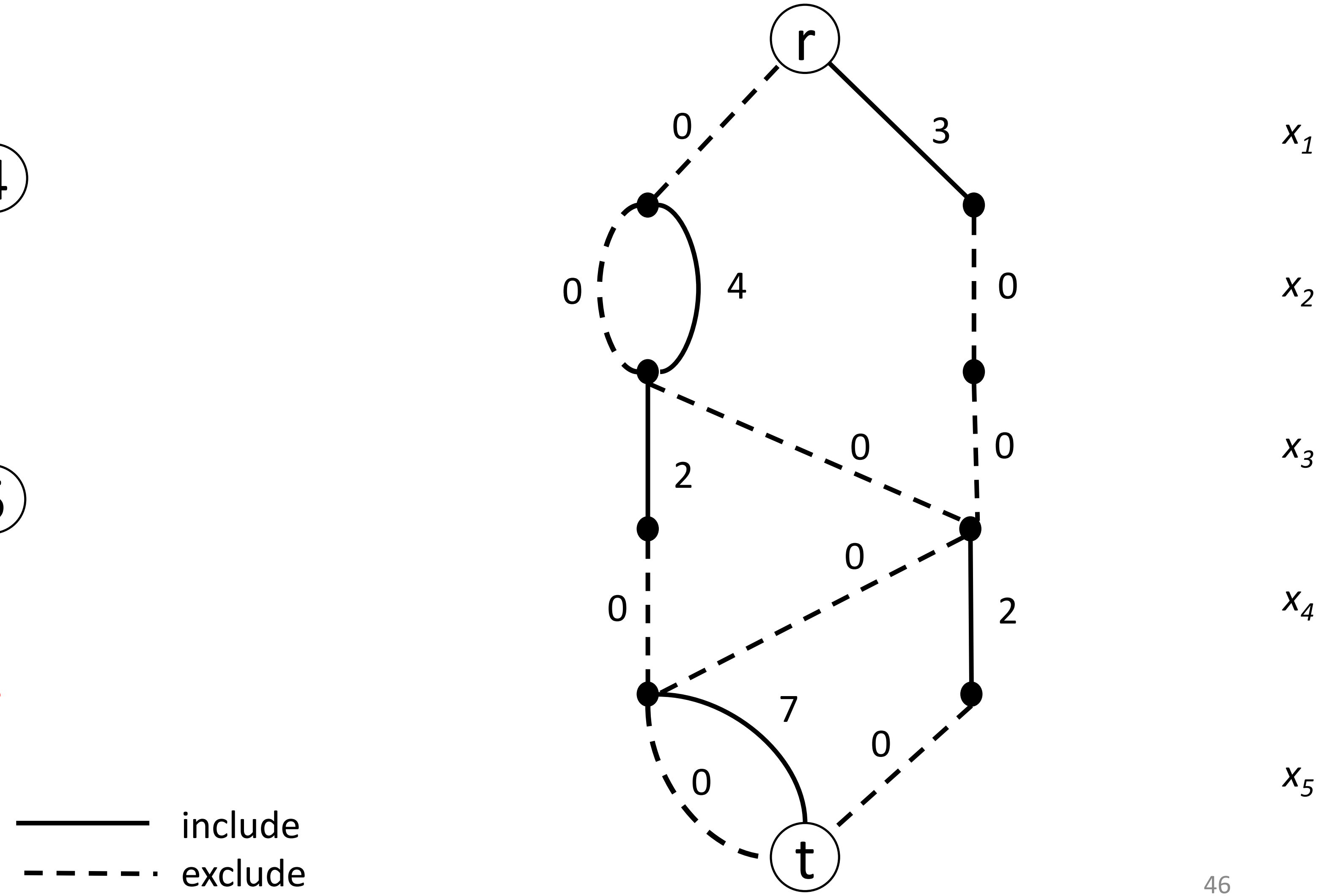


include
exclude

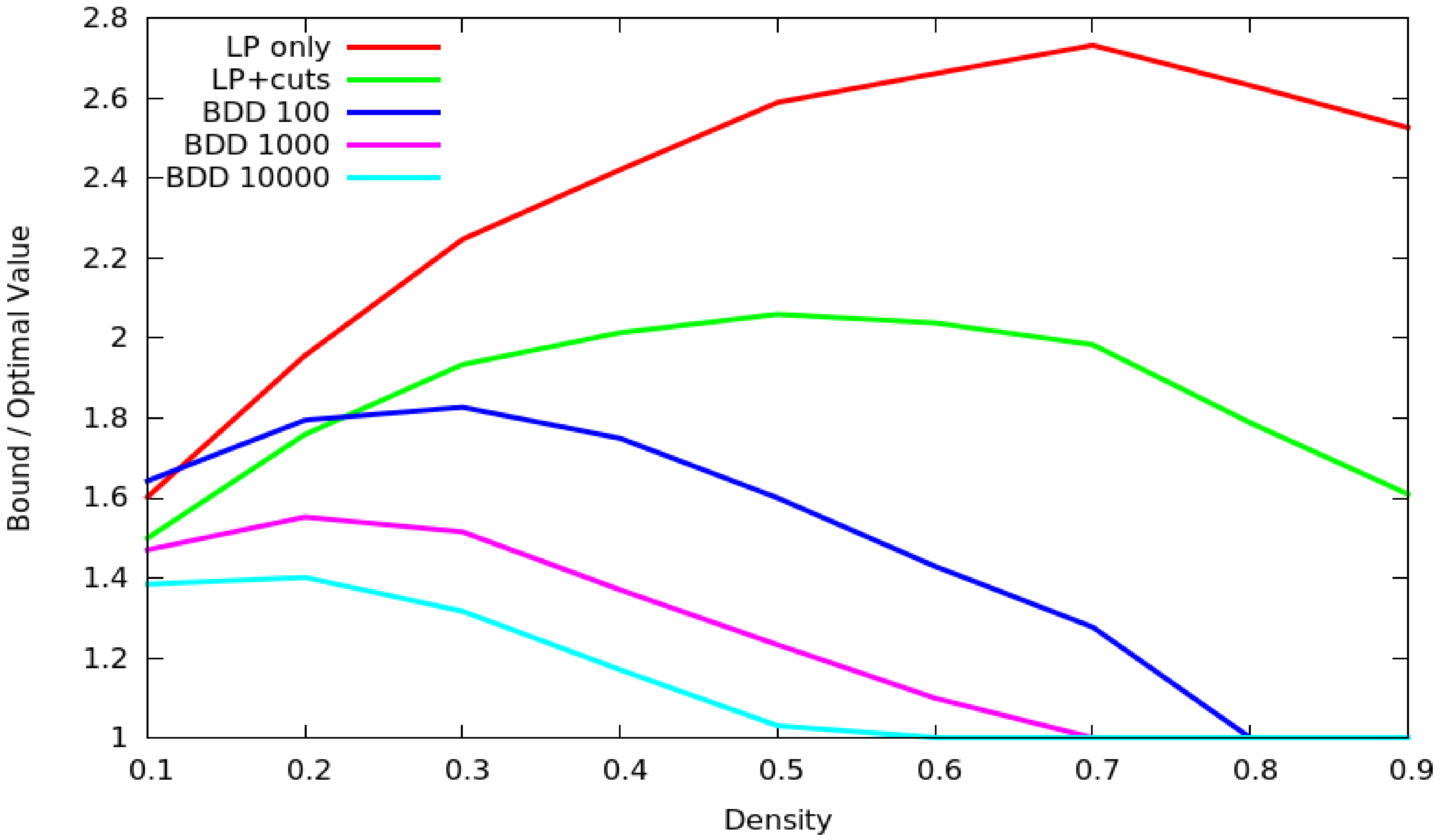
Building Relaxed Decision Diagrams



Max Width = 2



Relaxation Bound: Maximum Independent Set



Strengthening Diagram Relaxations

- Filtering operations
  - “Redundant” constraints

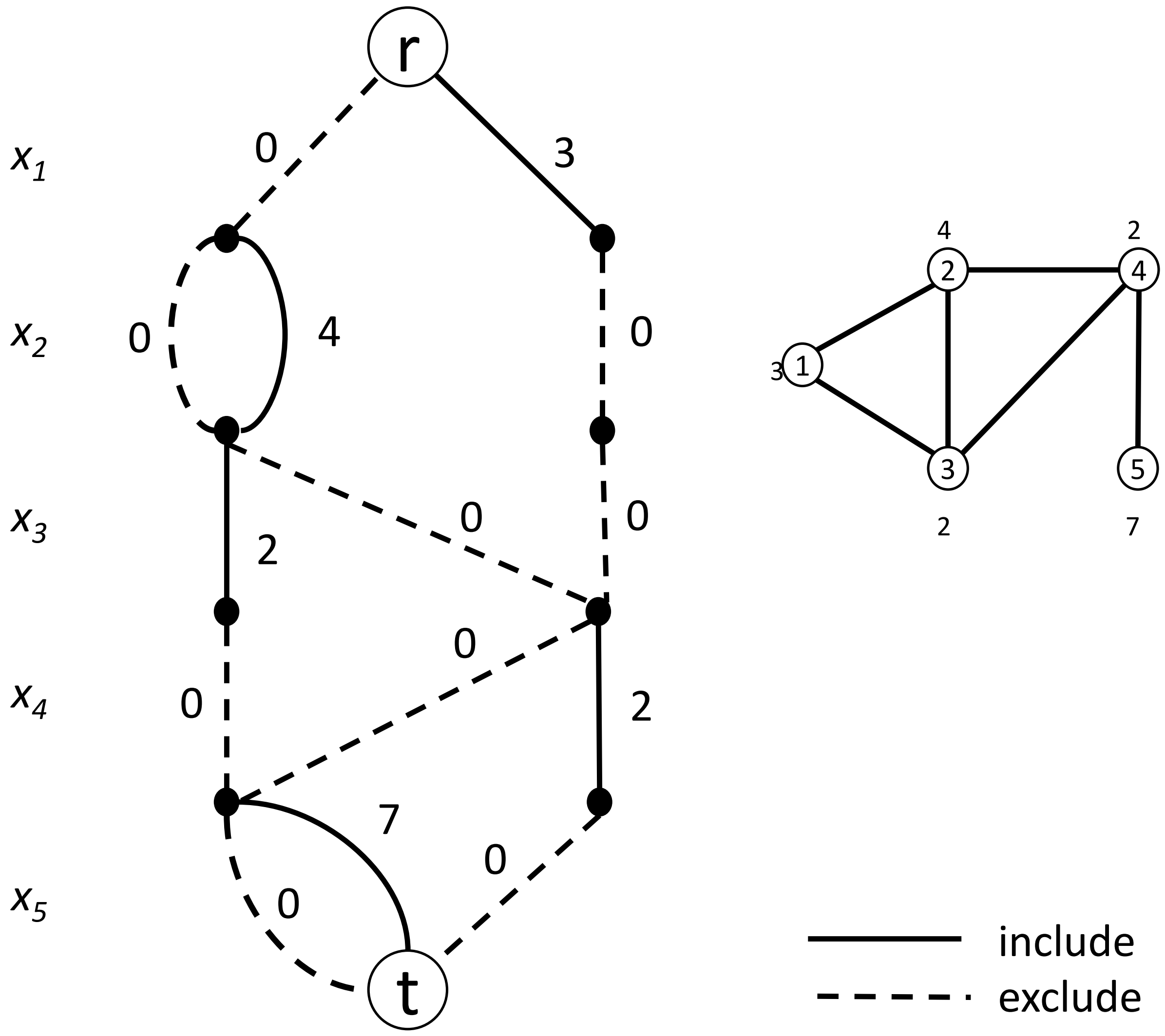
- Additive Bounding
  - Incorporate dual information from LP relaxations

- DD-Based Lagrangian Relaxations

Strengthening Diagram Relaxations

- Filtering operations
  - “Redundant” constraints
- Additive Bounding
  - Incorporate dual information from LP relaxations
- DD-Based Lagrangian Relaxations

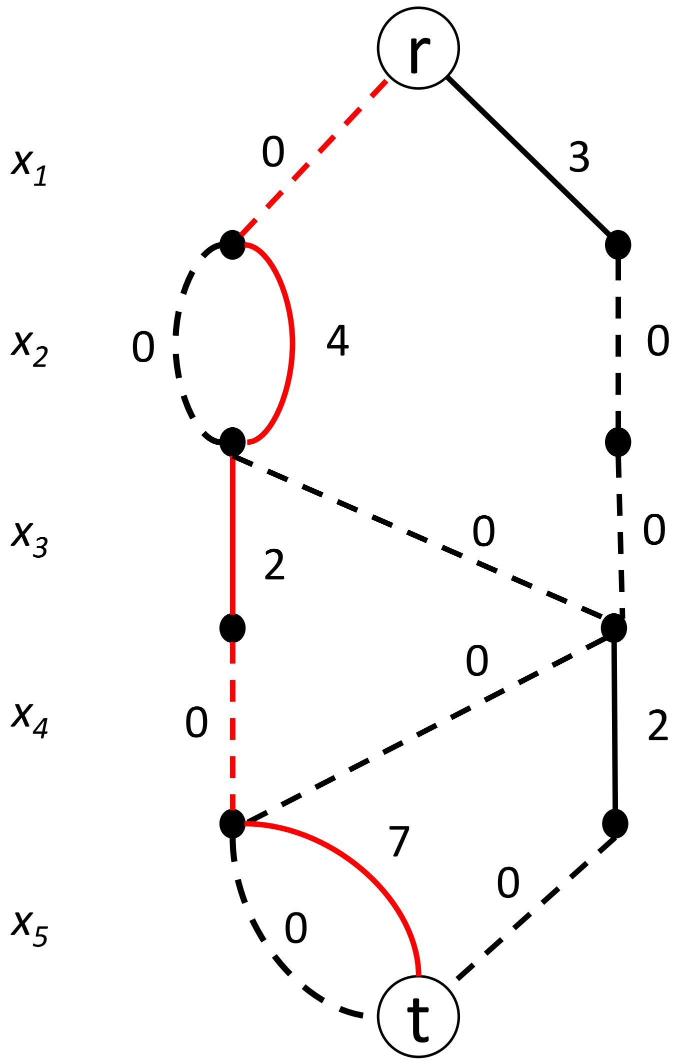
DD-Based Lagrangian Relaxation

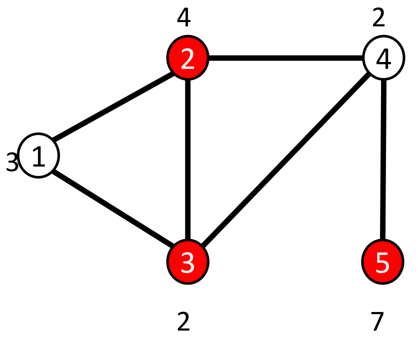


- We are solving

  max f(x)
  subject to
  x ∈ *RelaxedDD*

DD-Based Lagrangian Relaxation





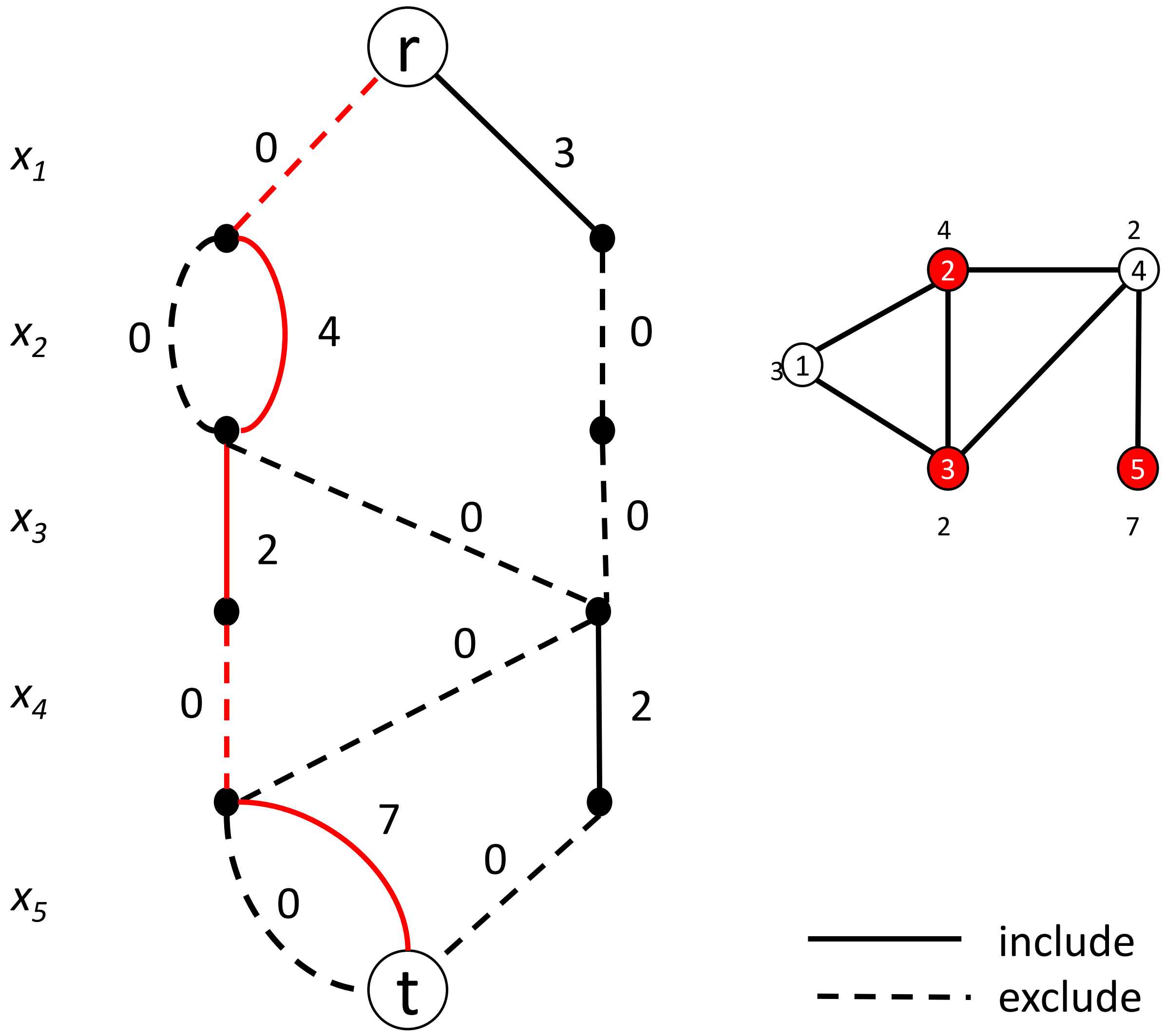
x = (0, 1, 1, 0, 1)
Upper bound = 13

include
exclude

- We are solving

  max f(x)
  subject to
  x ∈ *RelaxedDD*

DD-Based Lagrangian Relaxation



- Let A, b be such that:

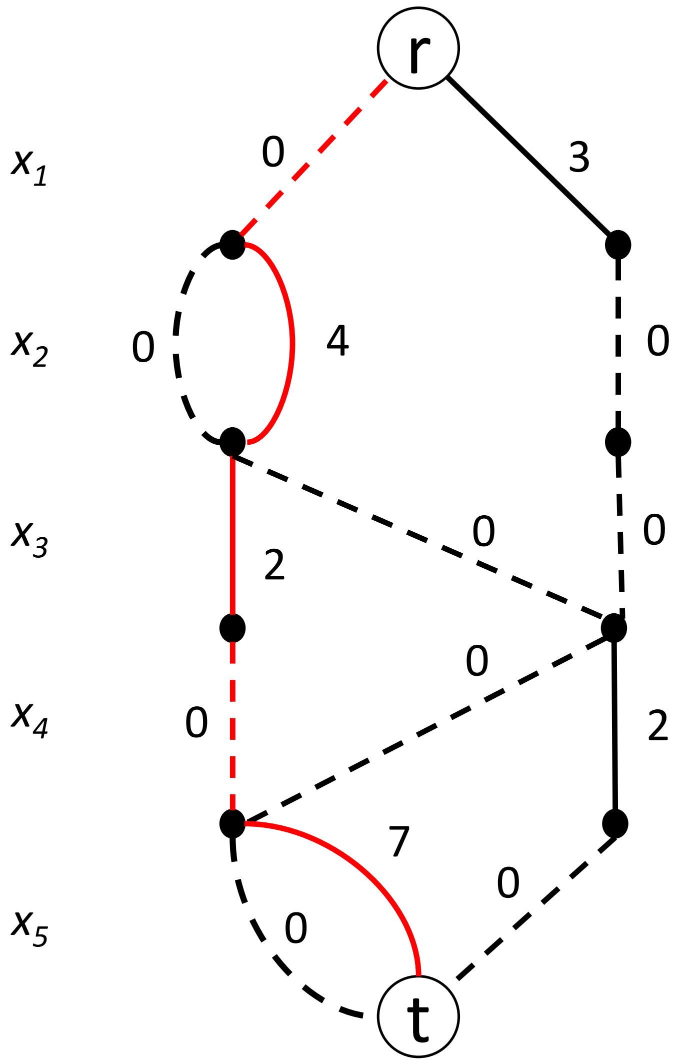
  Ax ≤ b for any **feasible** x

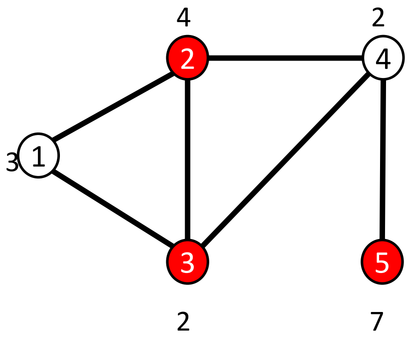
- DD-Based Lagrangian:

  max f(x) + λ(b – Ax)
  subject to
  x ∈ *RelaxedDD*

  - Gives an upper bound for any λ ≥ 0

DD-Based Lagrangian Relaxation





x = (0, 1, 1, 0, 1)
Upper bound = 13

include
exclude

- Solution (0,1,1,0,1) violates constraint

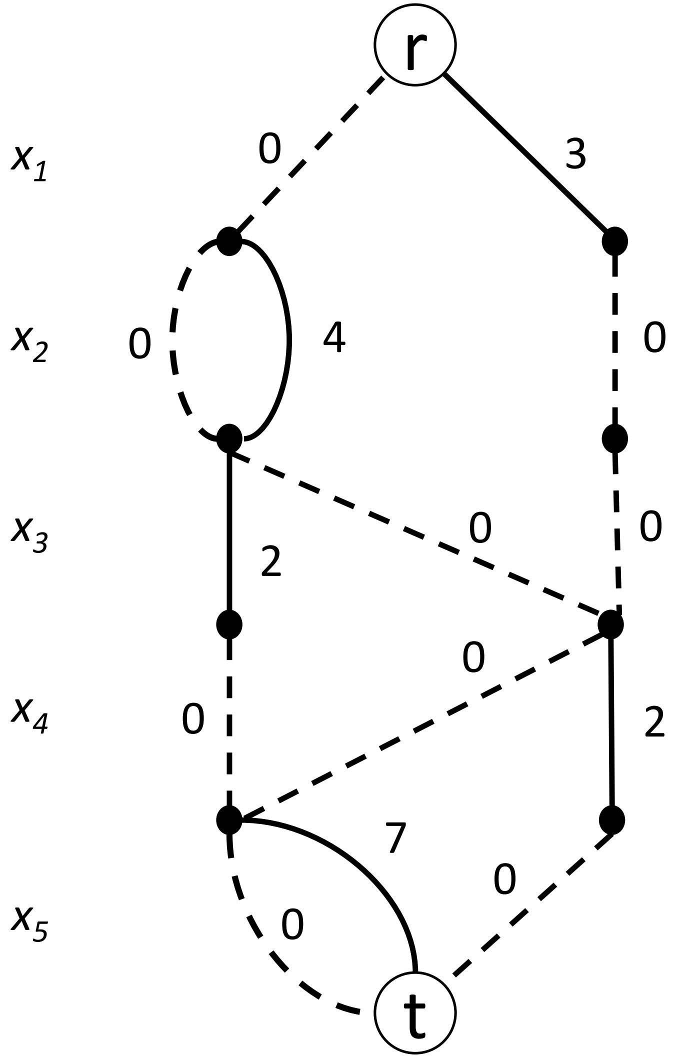
$$x_2 + x_3 \leq 1$$

- We penalize with term

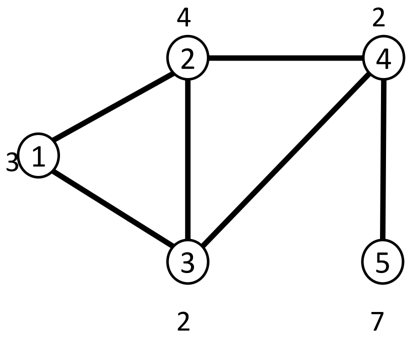
$$+ \lambda (1 - x_2 - x_3)$$

by simply changing the **cost structure** of the DD

DD-Based Lagrangian Relaxation



λ = 2



include
exclude

- Solution (0,1,1,0,1) violates constraint

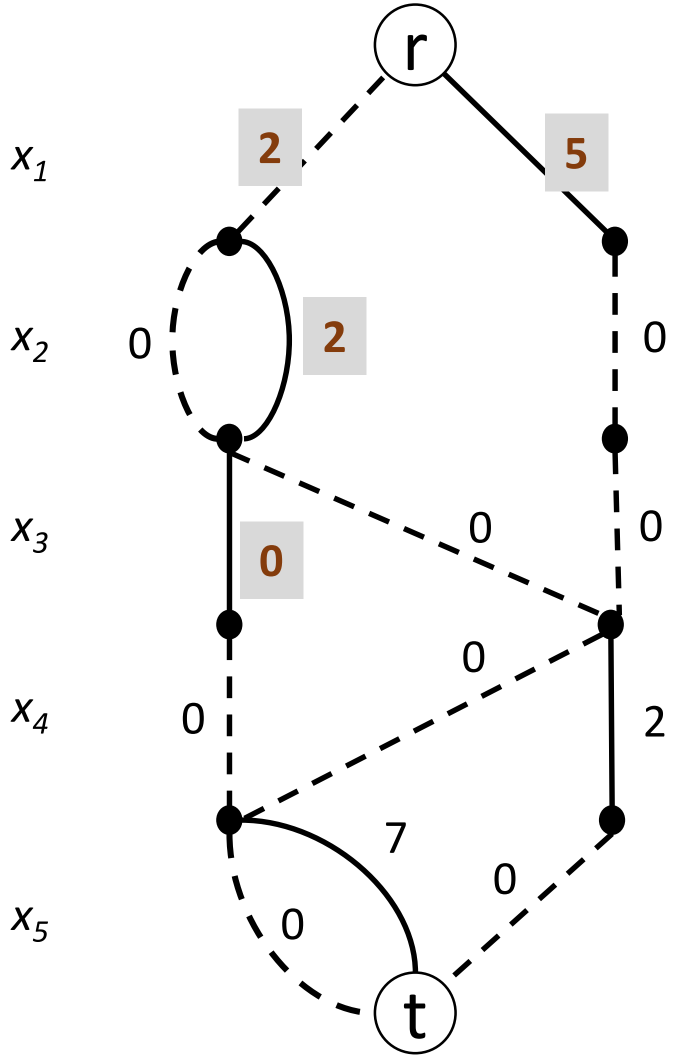
$$x_2 + x_3 \leq 1$$

- We penalize with term

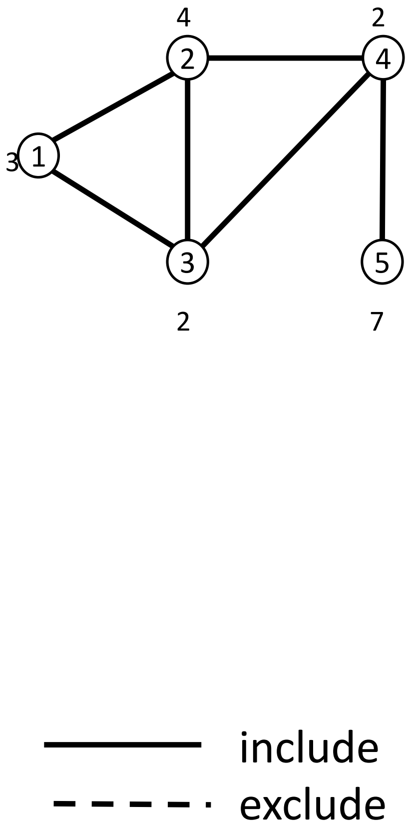
$$+ \lambda\,(1 - x_2 - x_3)$$

by simply changing the **cost structure** of the DD

DD-Based Lagrangian Relaxation



λ = 2



- Solution (0,1,1,0,1) violates constraint

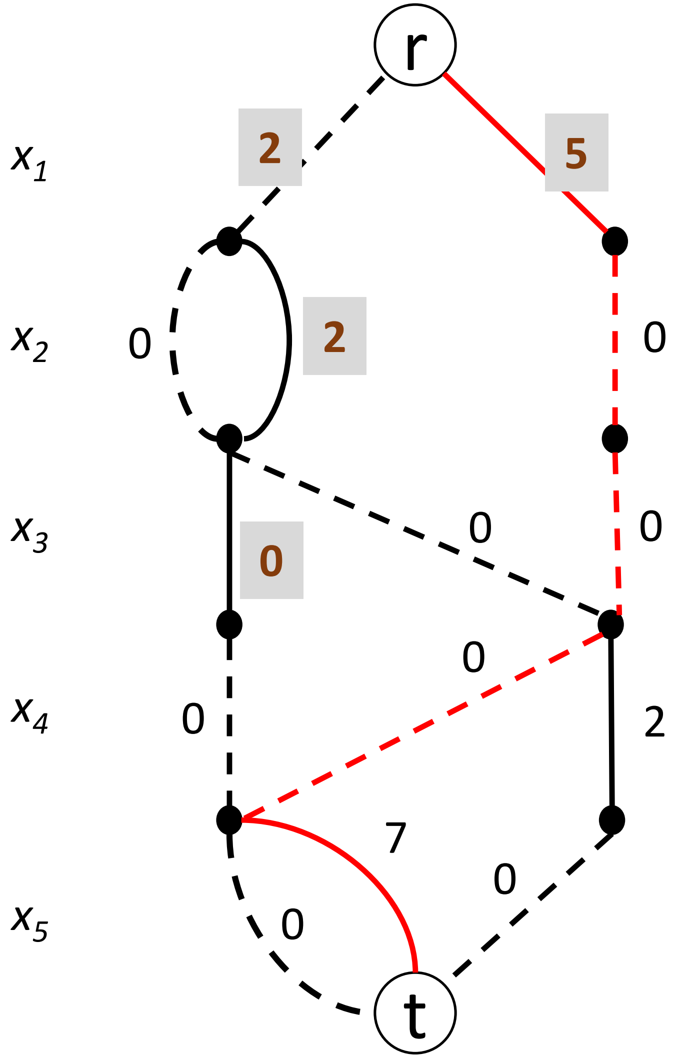
$$x_2 + x_3 \leq 1$$

- We penalize with term

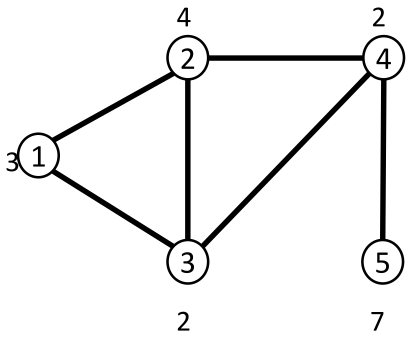
$$+ \lambda\,(1 - x_2 - x_3)$$

by simply changing the **cost structure** of the DD

DD-Based Lagrangian Relaxation



λ = 2



Better upper bound: 12 !

—— include
- - - - exclude

- Solution (0,1,1,0,1) violates constraint

$$x_2 + x_3 \leq 1$$

- We penalize with term

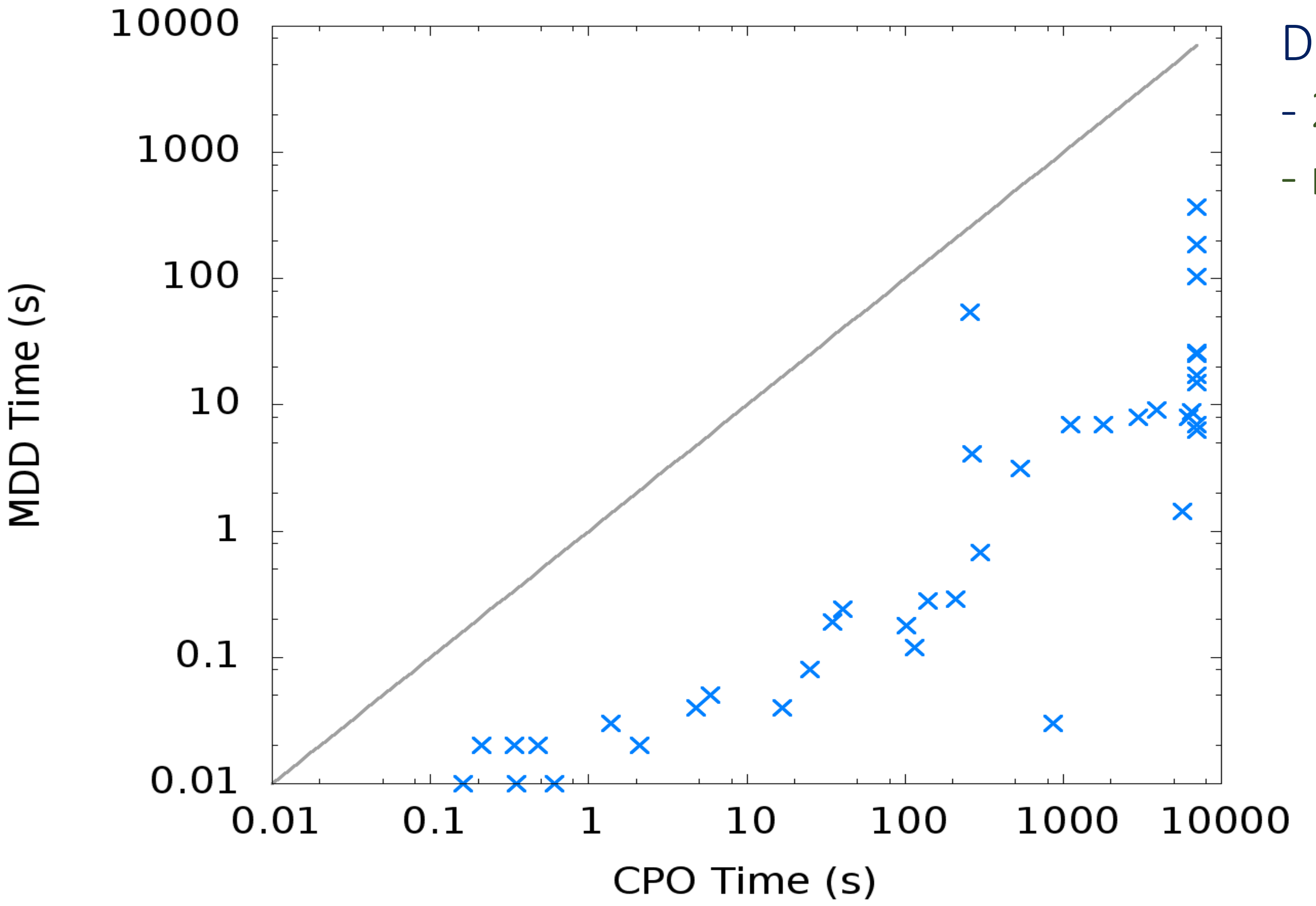
$$+ \lambda (1 - x_2 - x_3)$$

by simply changing the **cost structure** of the DD

Computational Analysis

- Incorporated into IBM ILOG CP Optimizer (CPO)
  - State-of-the-art constraint-based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxations

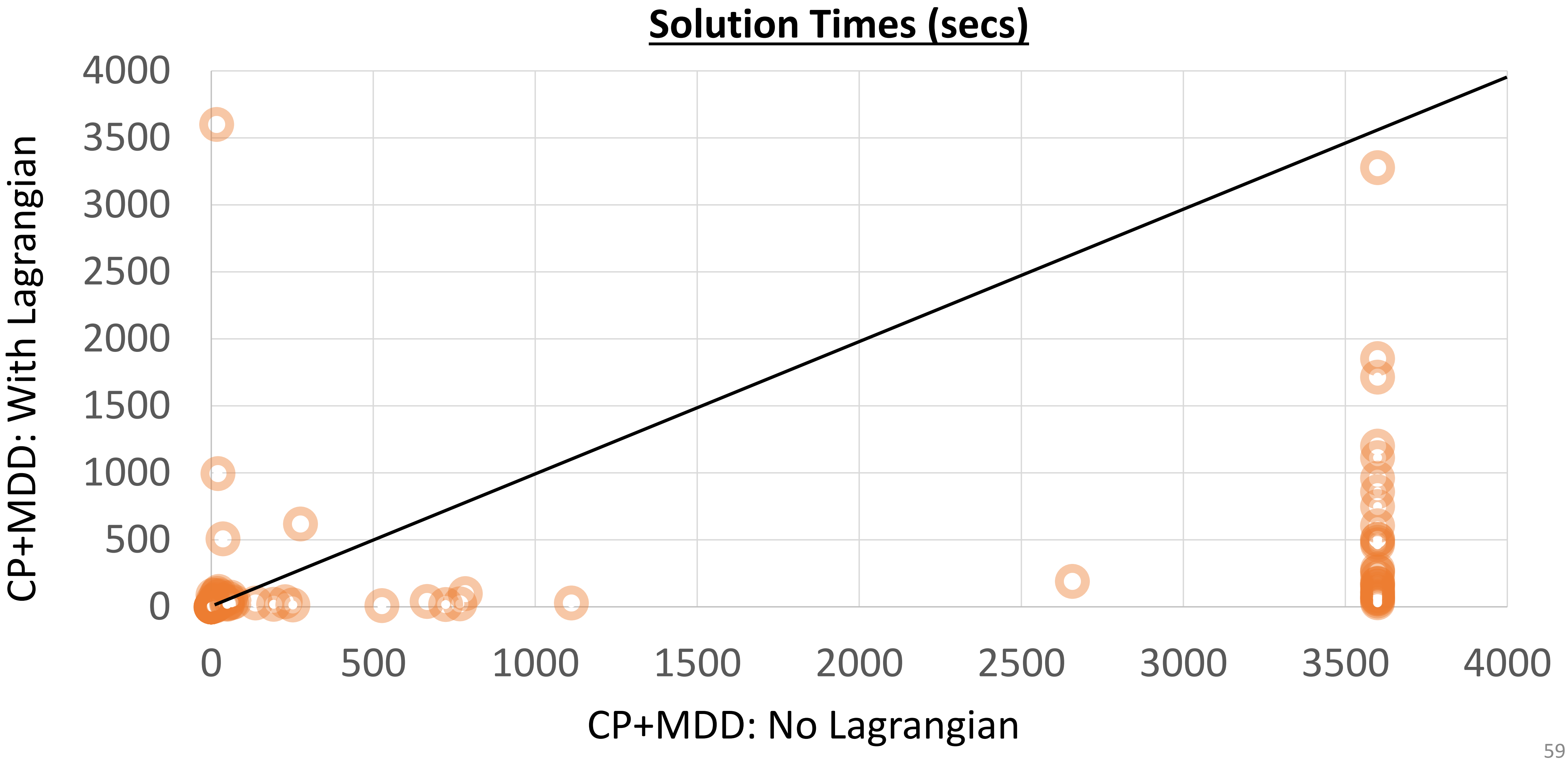
TSP with Time Windows



Dumas/Ascheuer instances

- 20-100 jobs
- maximum width: 16

DD-Based Lagrangian



Other Results

- Asymmetric TSP with Precedence Constraints
  - Closed 3 TSPLIB open instances

- Easy modeling for certain problems
  - Example: *Time-Dependent TSPs*

Relaxation Methods

E.g., Linear Programming Relaxation

Modeling Framework

E.g., Linear Inequalities

Primal Heuristics

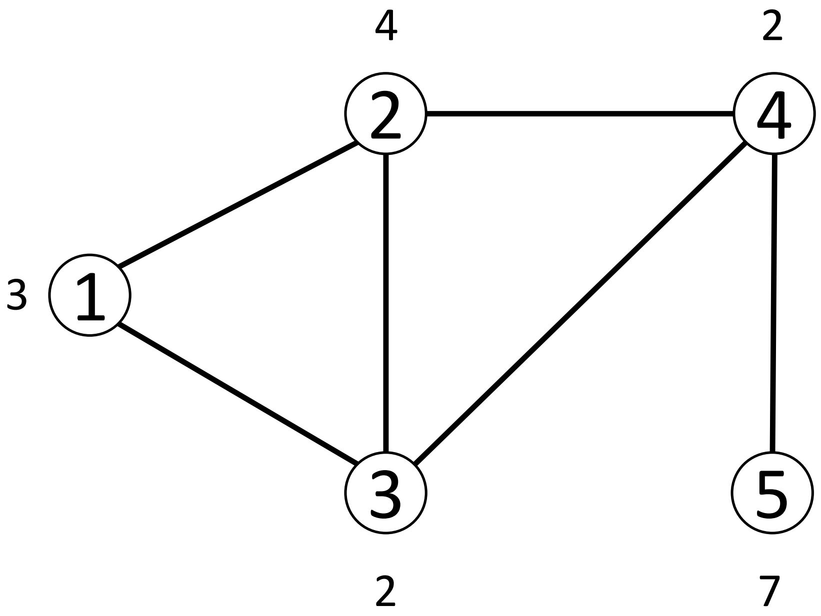
E.g., Feasibility Pump

Generic Optimization Techniques

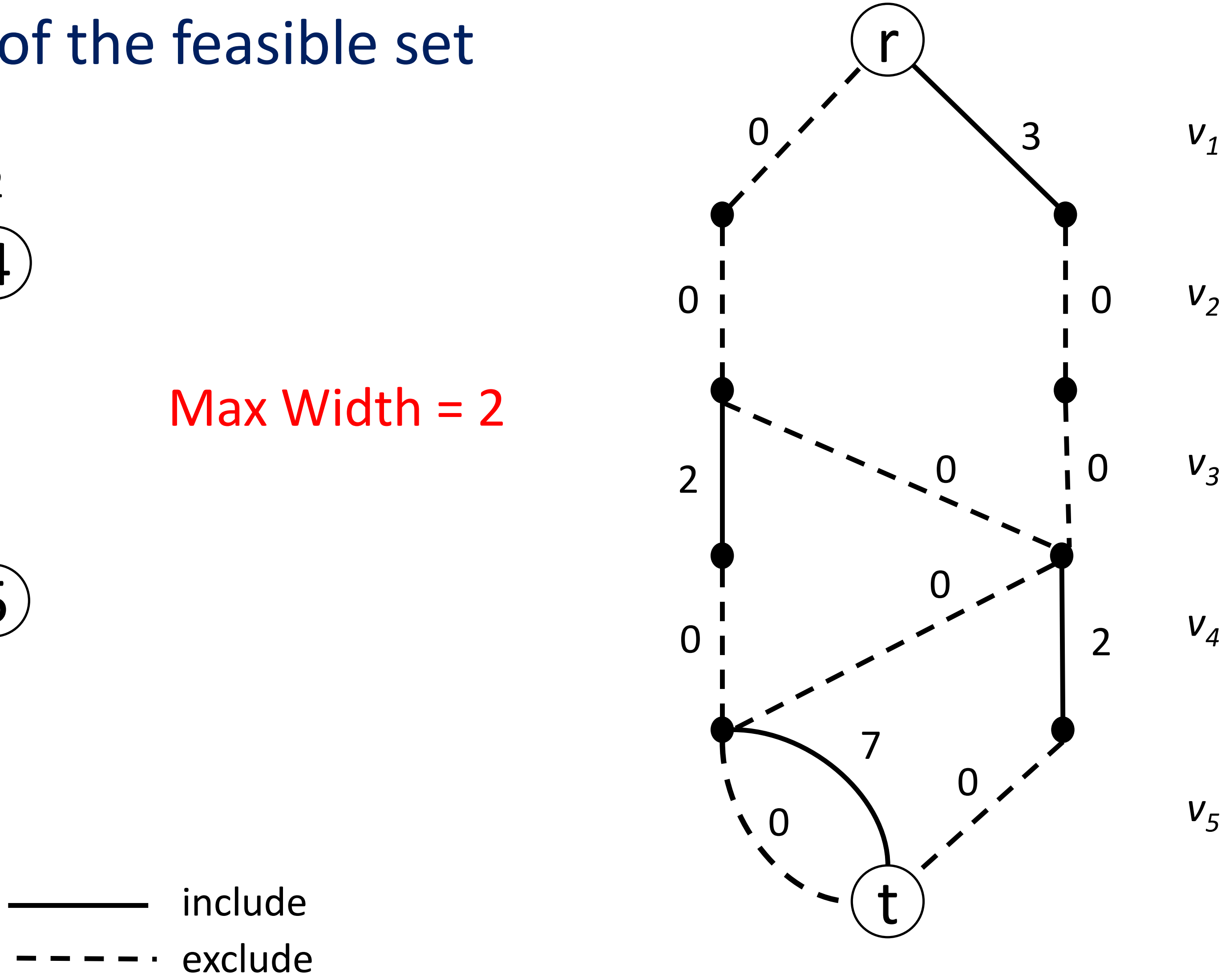
E.g., Mixed-integer Programming

Restricted Decision Diagrams

- Under-approximation of the feasible set

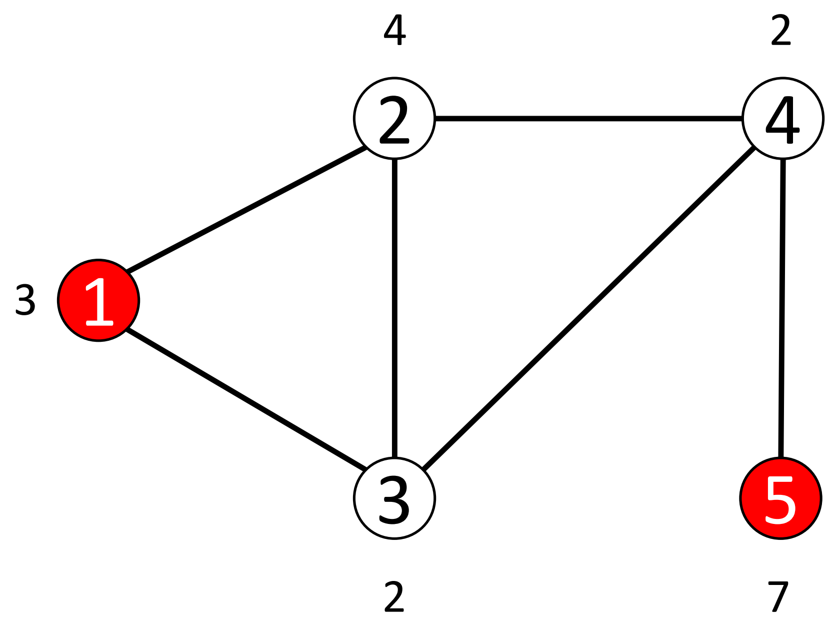


Max Width = 2

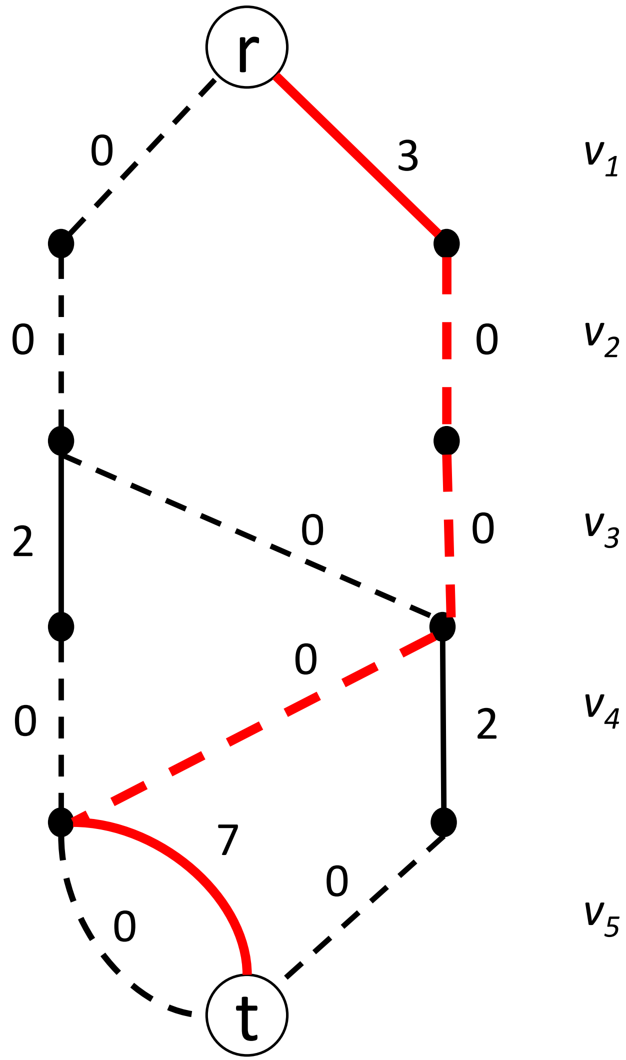


Restricted Decision Diagrams

- Under-approximation of the feasible set



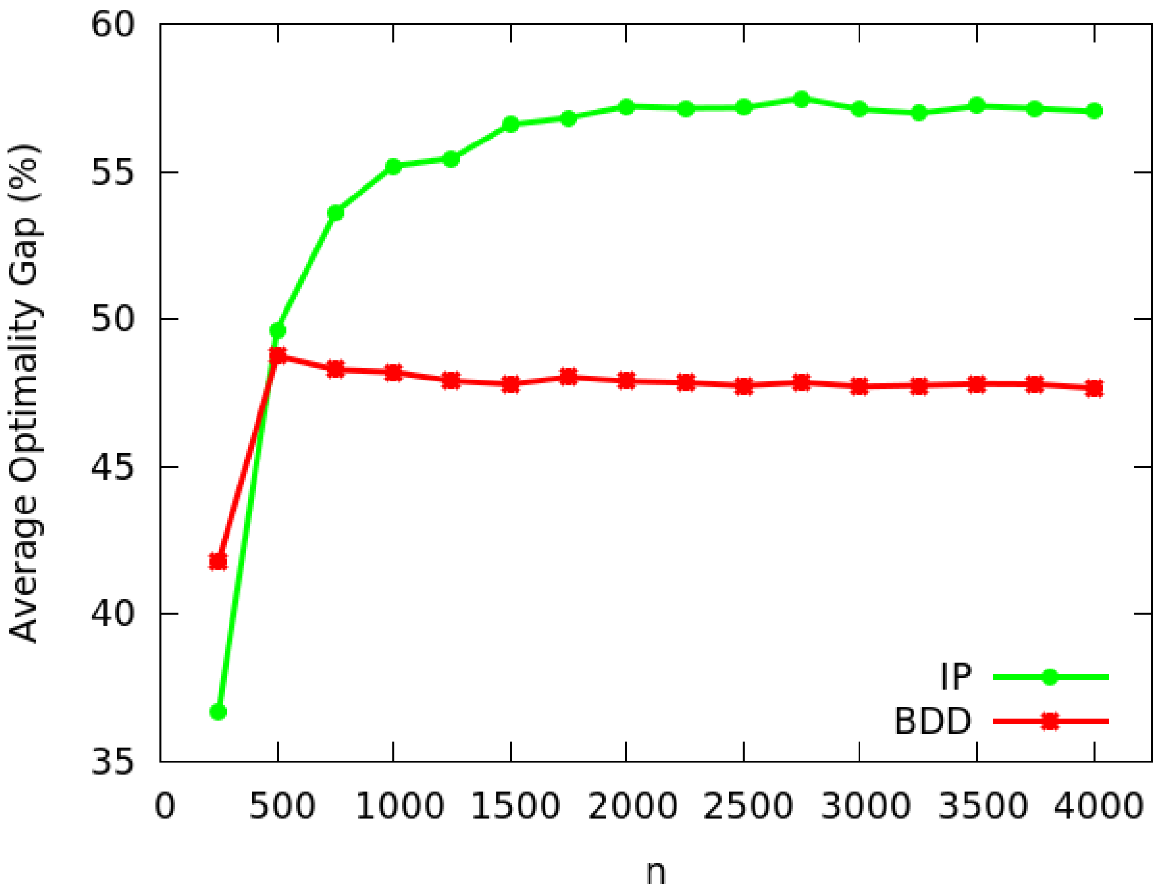
Max Width = 2

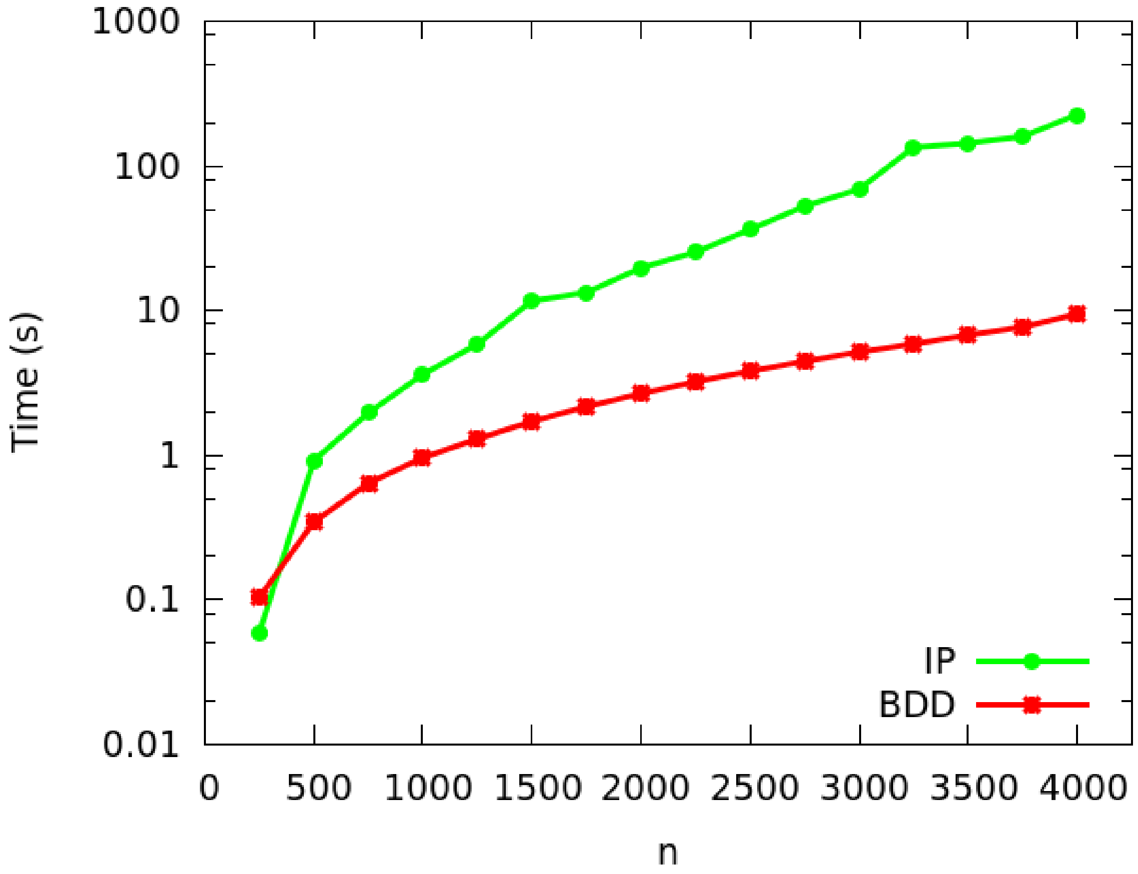


(1,0,0,0,1) → Lower bound = 10

include
exclude

Primal Bound: Set Covering





Relaxation Methods

E.g., Linear Programming Relaxation

Modeling Framework

E.g., Linear Inequalities

Primal Heuristics

E.g., Feasibility Pump

Generic Optimization Techniques

E.g., Mixed-integer Programming

Inference

E.g., valid cuts

Quick Notes on Inference

- Cut generation for MIPs
  - Several techniques from Behle'07
  - Recent: **Polar set cuts** from Relaxed Decision Diagrams
    - Talk to Christian Tjandraatmadja! (poster yesterday!)

- Highly-structured Cuts
  - Precedence relations that must hold in scheduling problems

- We are still exploring notion of *decision diagram separation*
  - Cire & Hooker, ISAIM 2014

Relaxation Methods

E.g., Linear Programming Relaxation

Modeling Framework

E.g., Linear Inequalities

Primal Heuristics

E.g., Feasibility Pump

Generic Optimization Techniques

E.g., Mixed-integer Programming

Inference

E.g., valid cuts

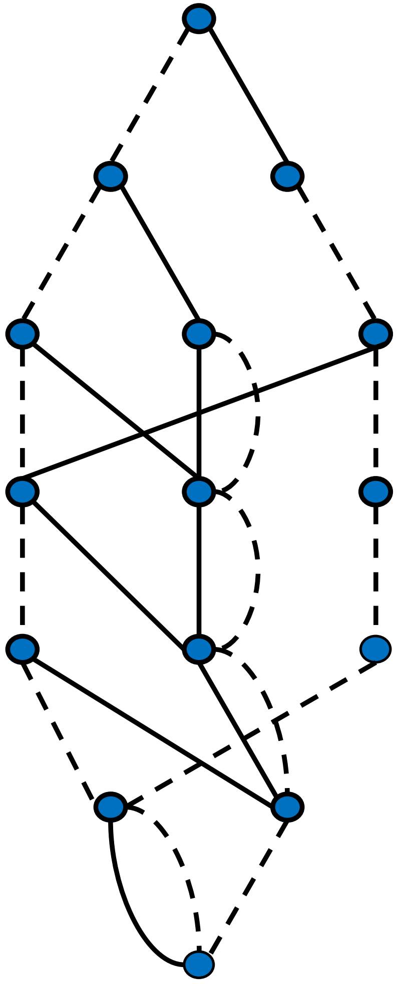
Search

E.g., Branch and bound

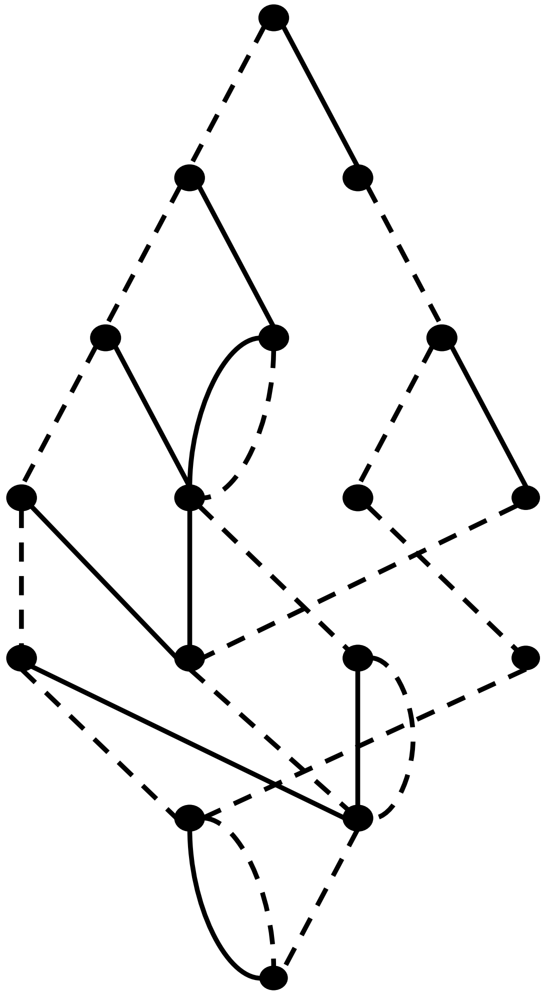
Exact Method

- Novel decision diagram branch-and-bound scheme
  - Relaxed diagrams play the role of the LP relaxation
  - Restricted diagrams are used as primal heuristics

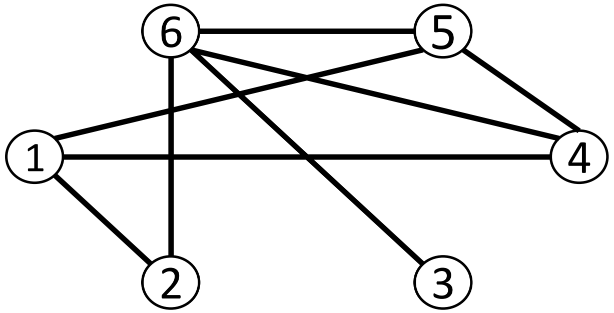
- Branching is done on the **nodes** of the diagram
  - Branching on **pools** of partial solutions
  - Eliminate search symmetry

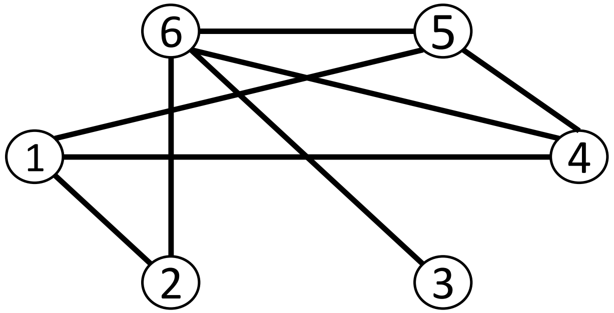


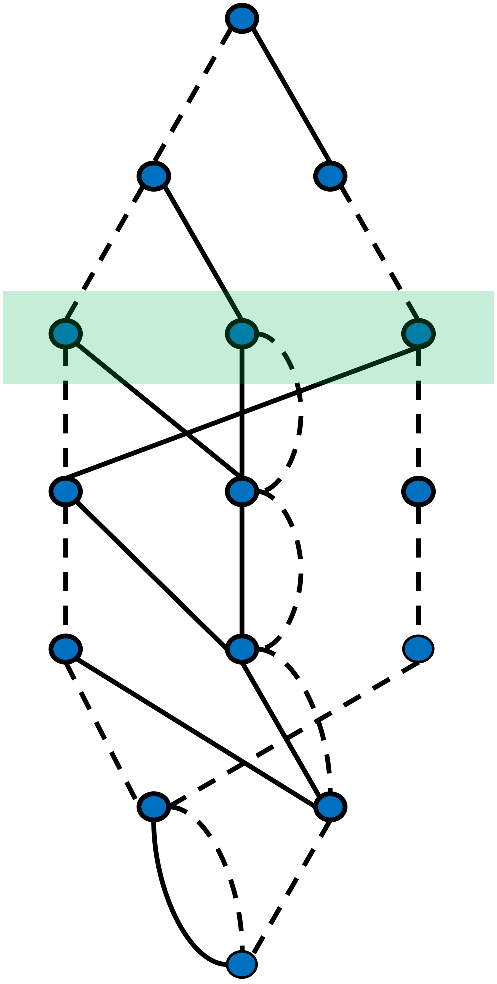
Relaxed



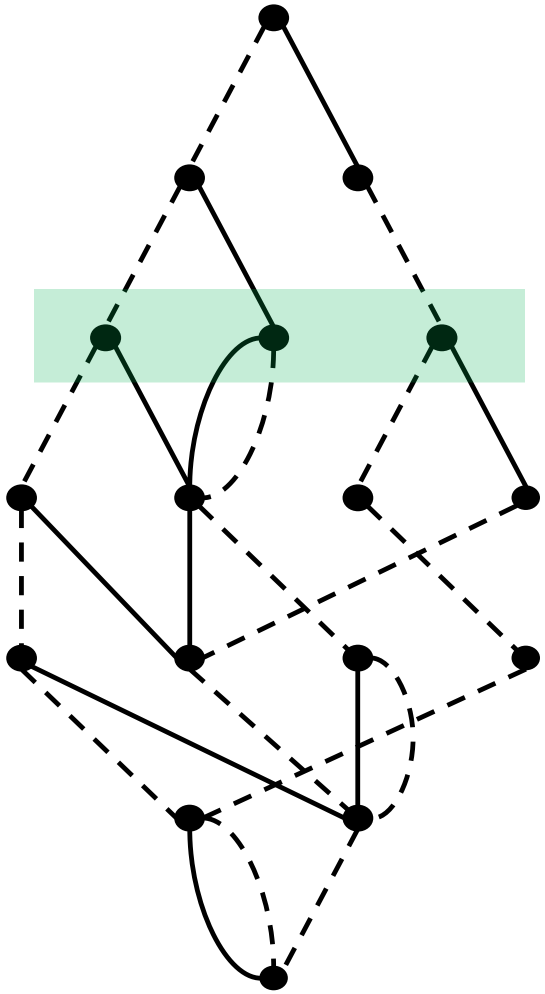
Exact





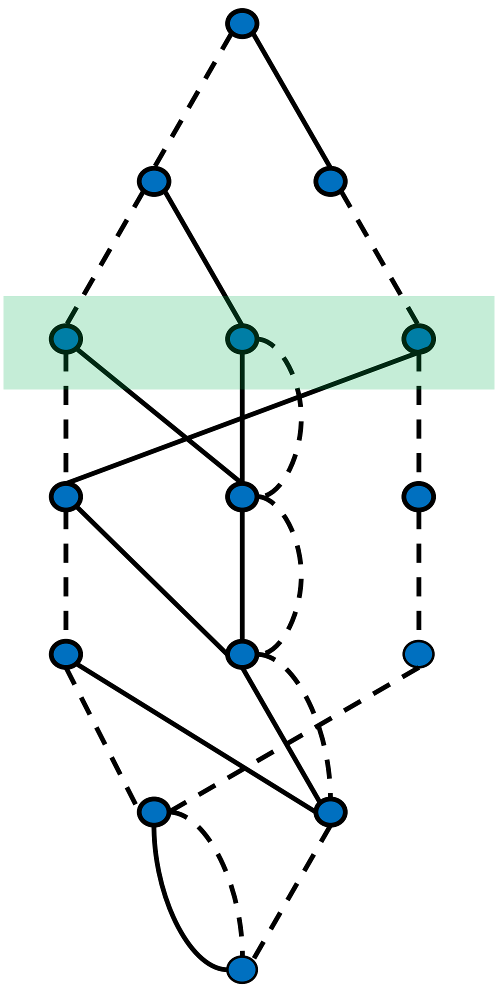


Relaxed

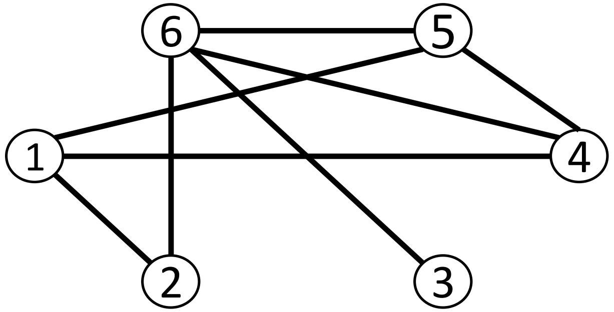


Exact

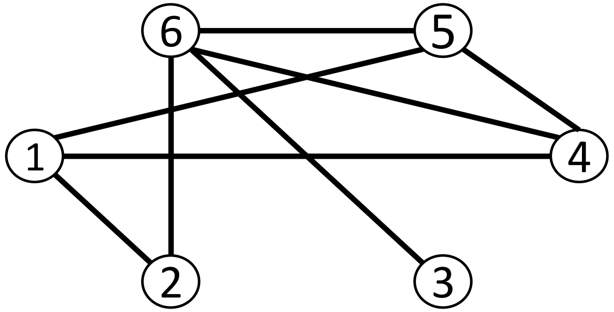
Up to a certain layer, the diagrams are the same (i.e., one layer before you start forcefully merging)

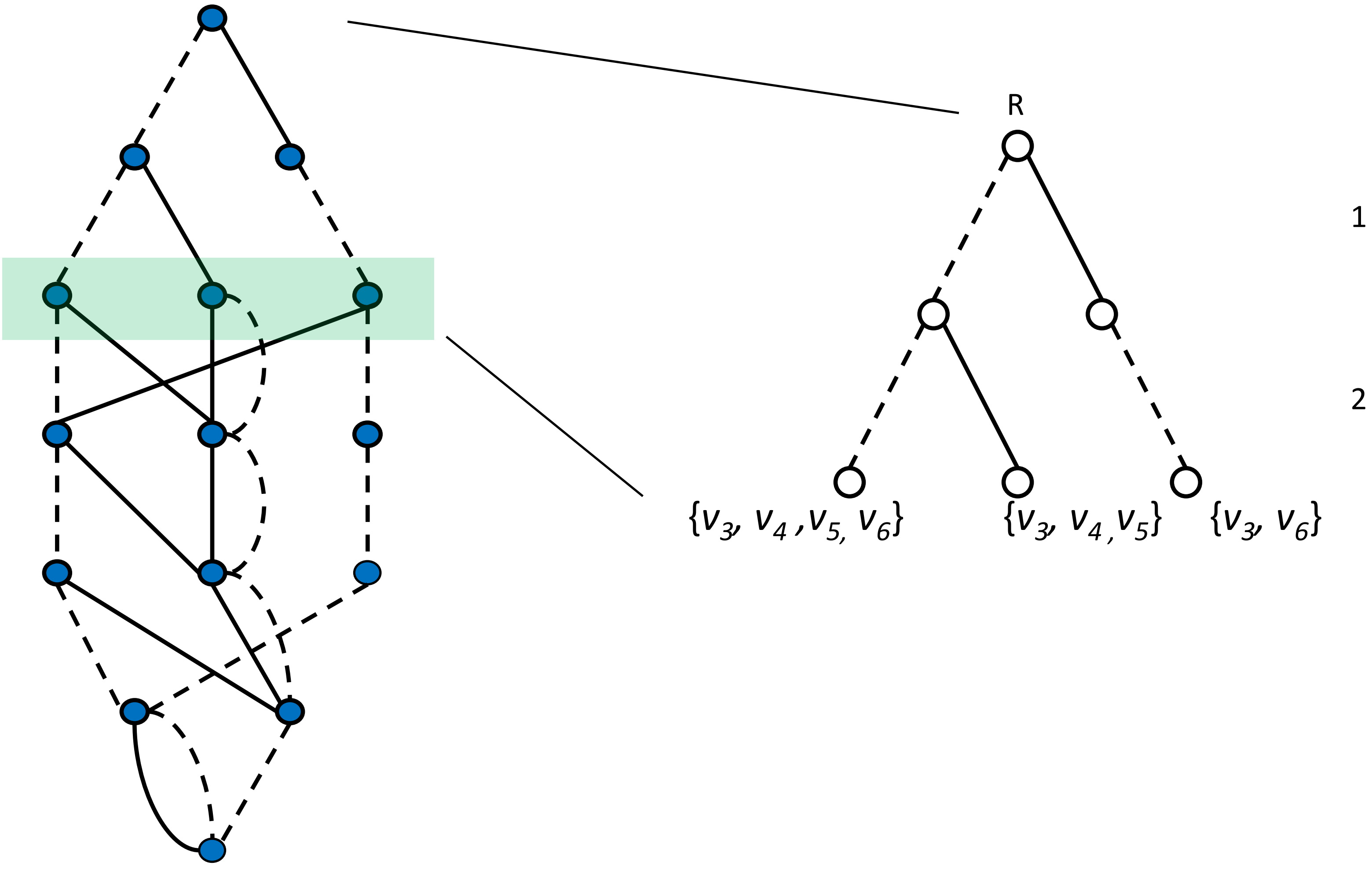


Relaxed

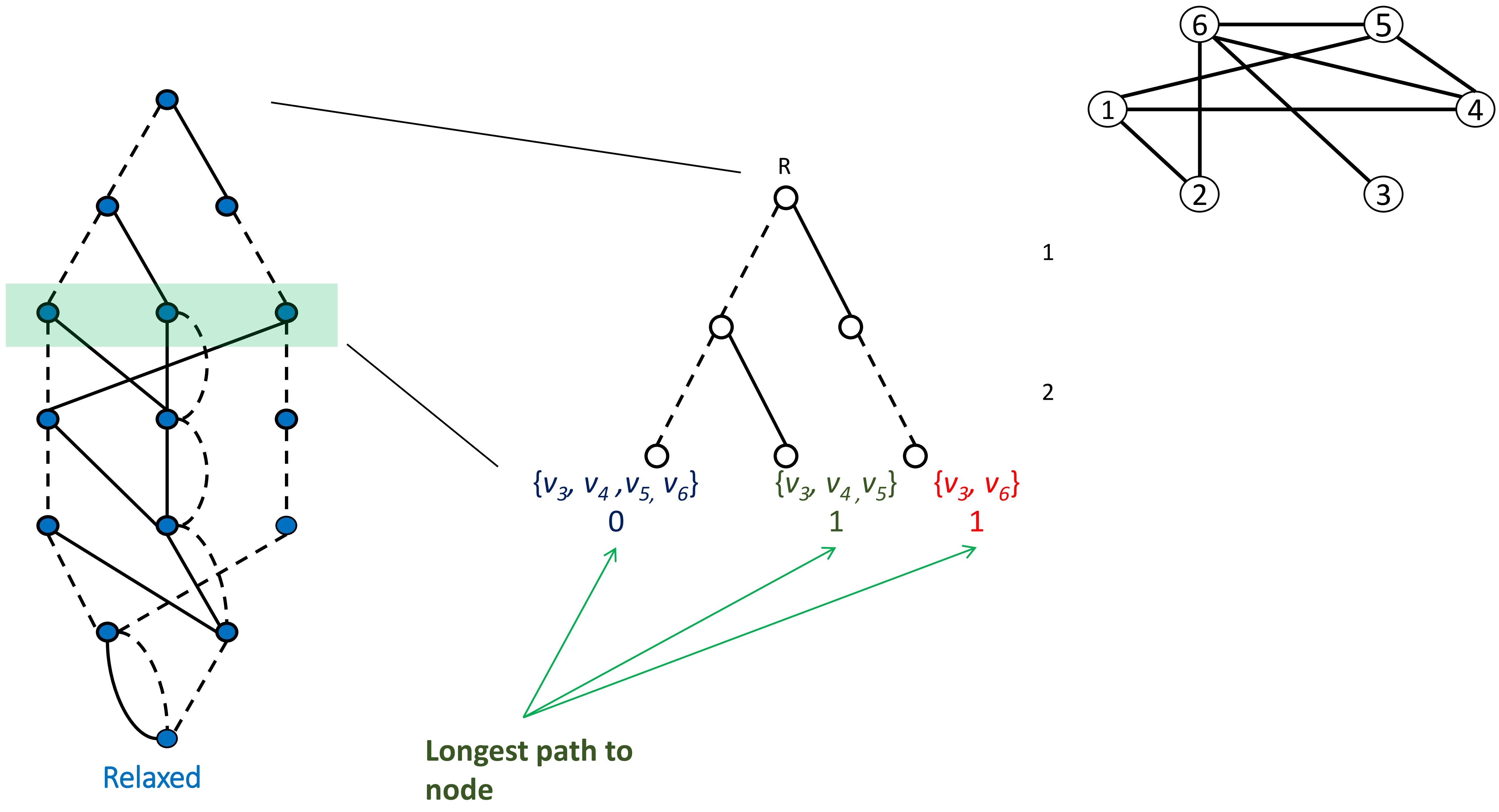


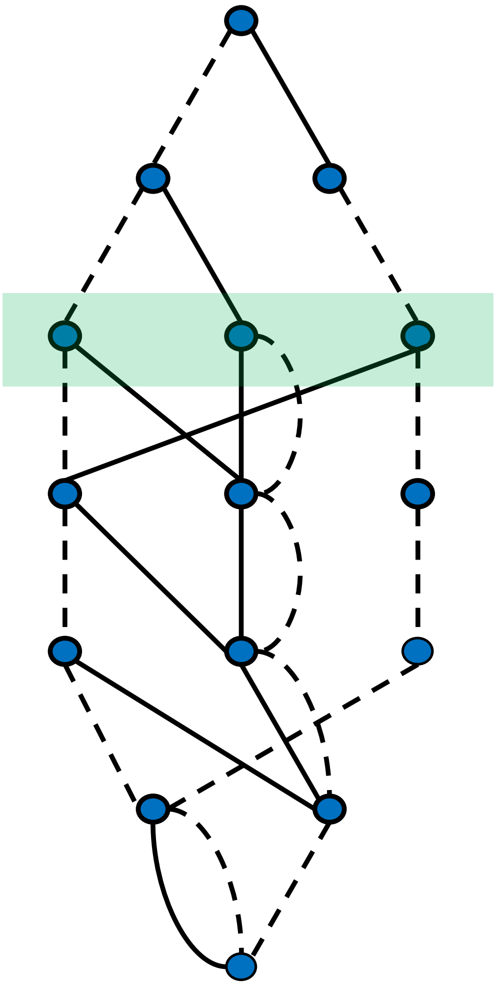
Thus, an optimum solution must necessarily pass through one of these nodes



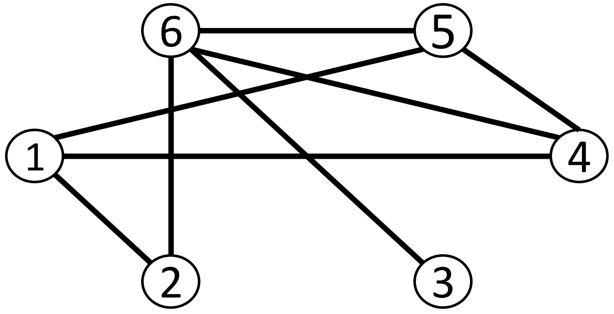


Relaxed


R
1
2
{v_3, v_4, v_5, v_6}
0
{v_3, v_4, v_5}
1
{v_3, v_6}
1
Longest path to node
Relaxed
6
5
1
4
2
3



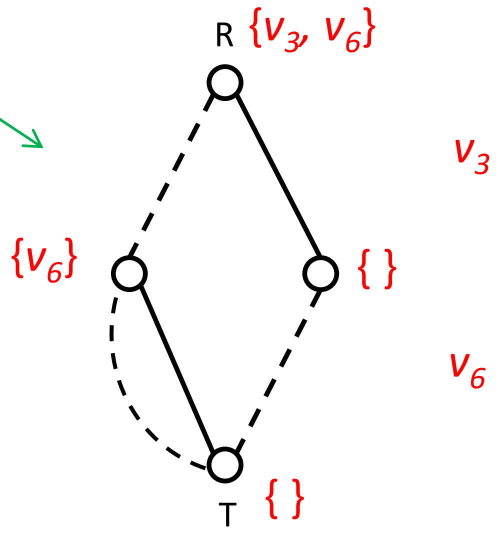
Relaxed

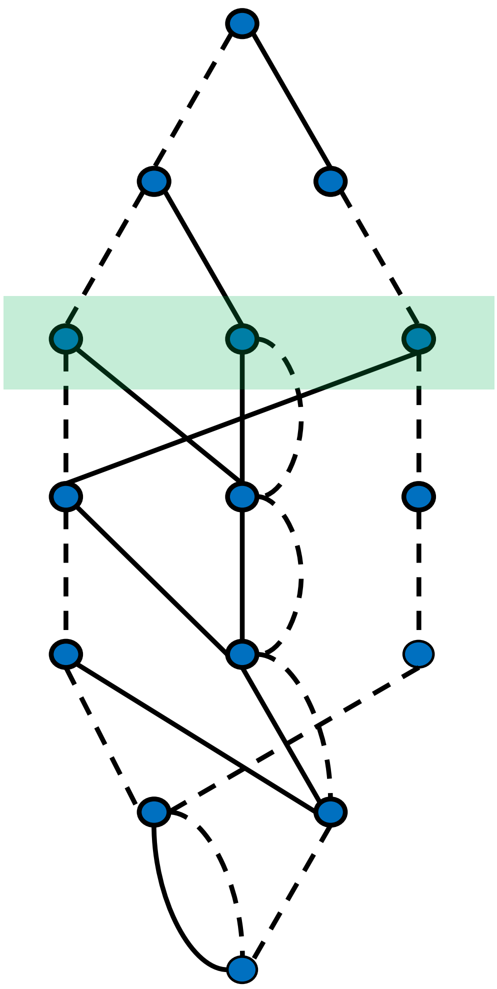


$\{v_3, v_4, v_5, v_6\}$
0

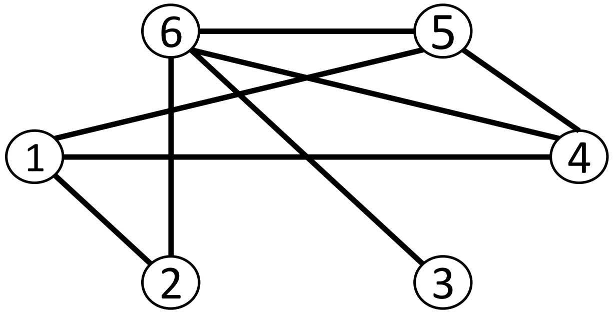
$\{v_3, v_4, v_5\}$
1

$\{v_3, v_6\}$
1





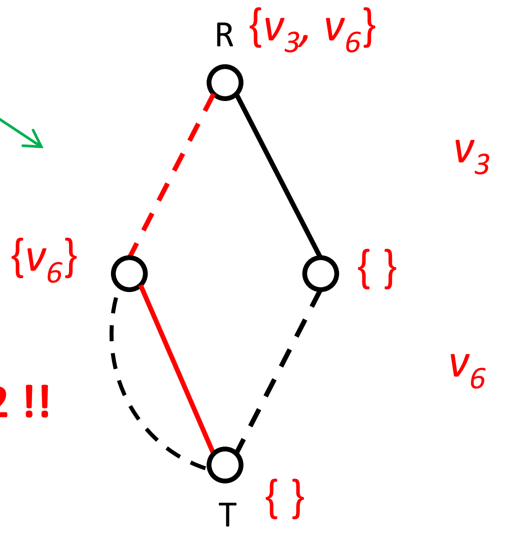
Relaxed



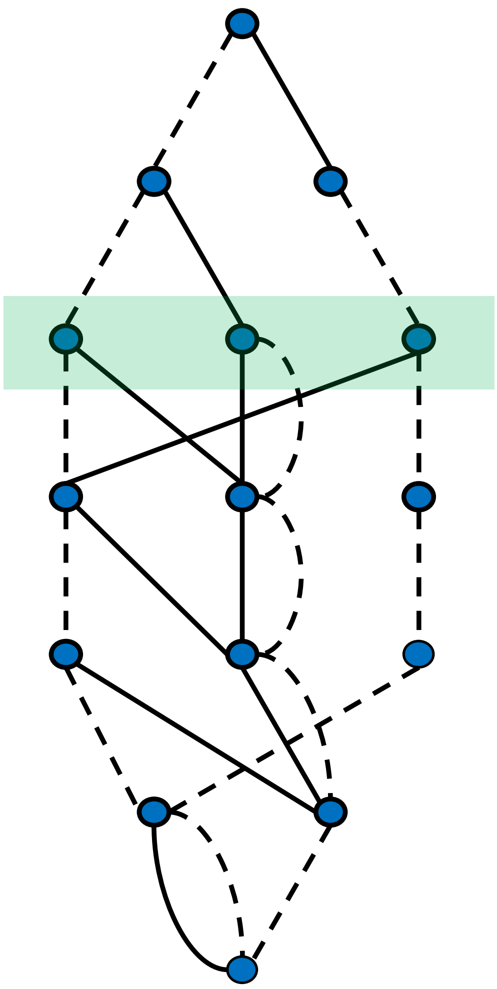
$\{v_3, v_4, v_5, v_6\}$ 0

$\{v_3, v_4, v_5\}$ 1

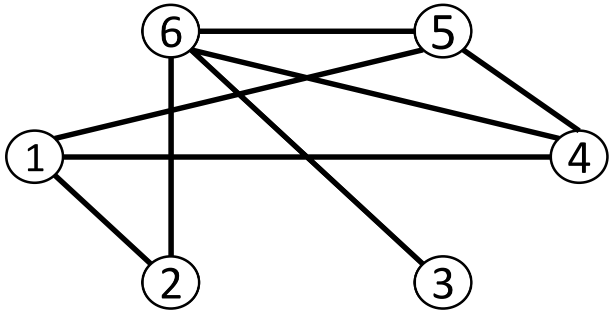
$\{v_3, v_6\}$ 1



Solution of value 2 !!



Relaxed



$\{v_3, v_4, v_5, v_6\}$ 0

$\{v_3, v_4, v_5\}$ 1

$\{v_3, v_6\}$ 1

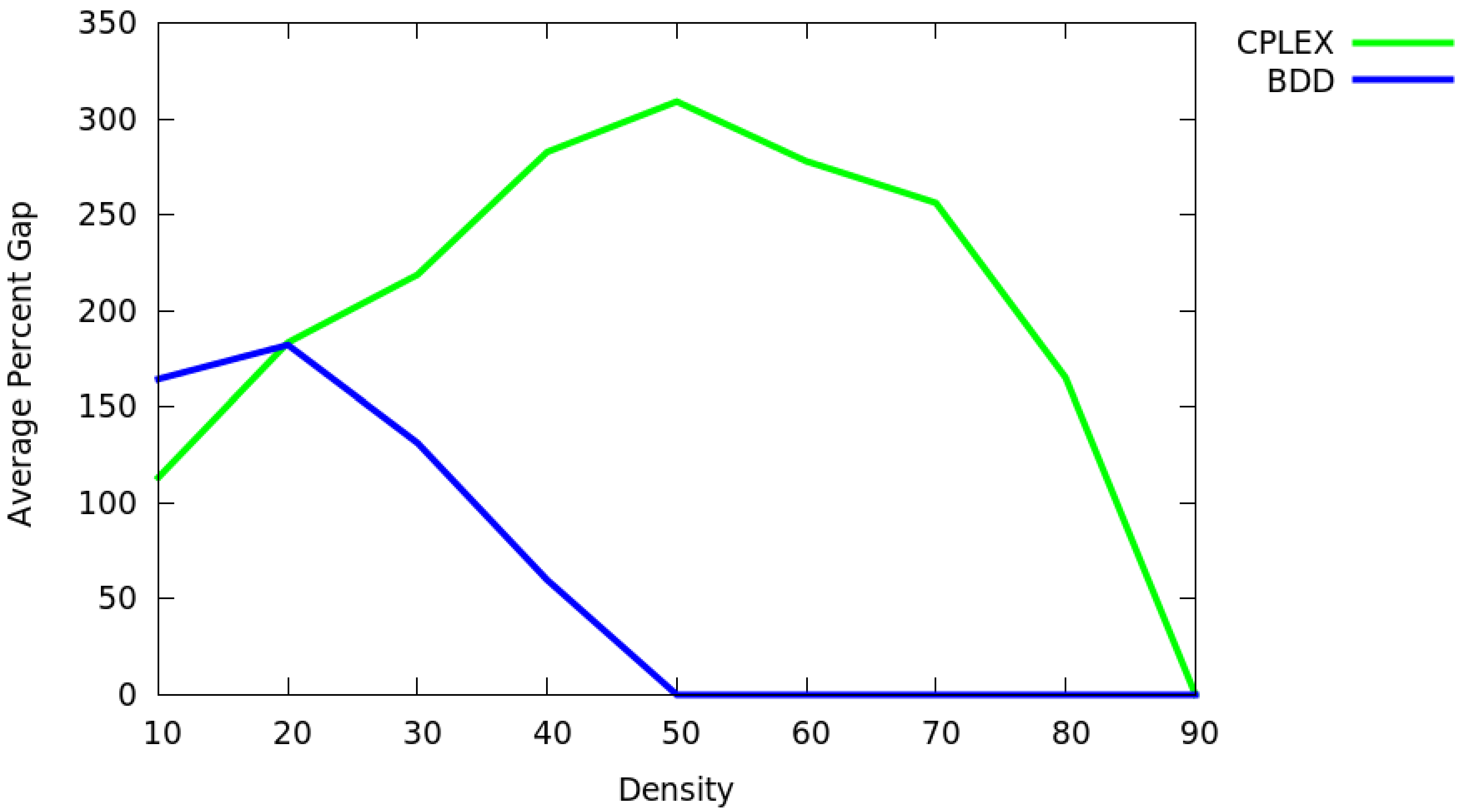
Explore each separately, saving the best solution/bound found

Maximum Cut

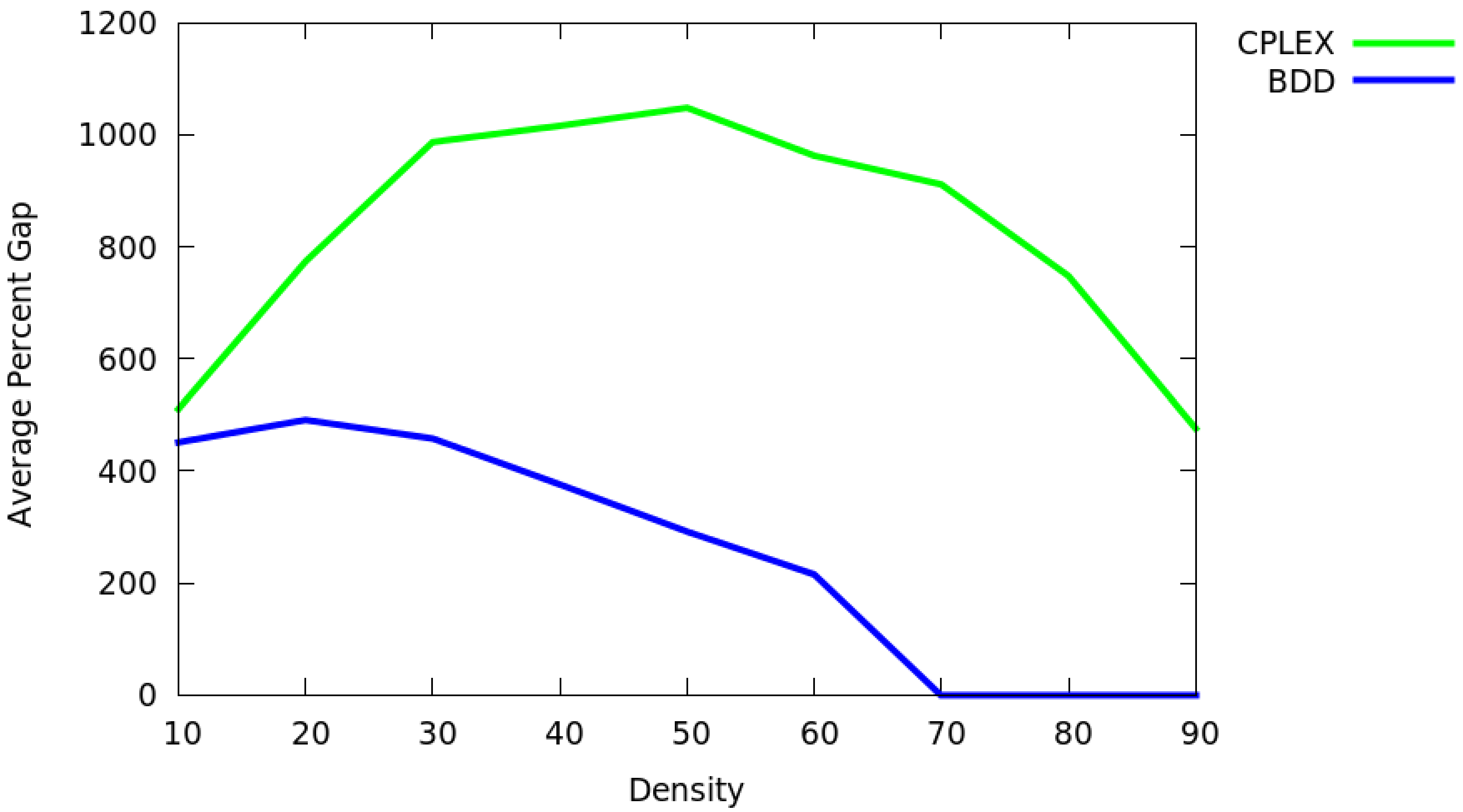
- Reduced certain optimality gaps

instance	old % gap	new % gap	% reduction
g11	11.17	0.53	95.24
g50	1.84	0.32	82.44
g32	11.59	10.64	8.20
g12	11.69	10.79	7.69
g33	11.70	11.30	3.39
g34	12.32	11.99	2.65

Maximum Independent Set: **500 variables**



Maximum Independent Set: **1500 variables**



Parallel Search with Decision Diagrams

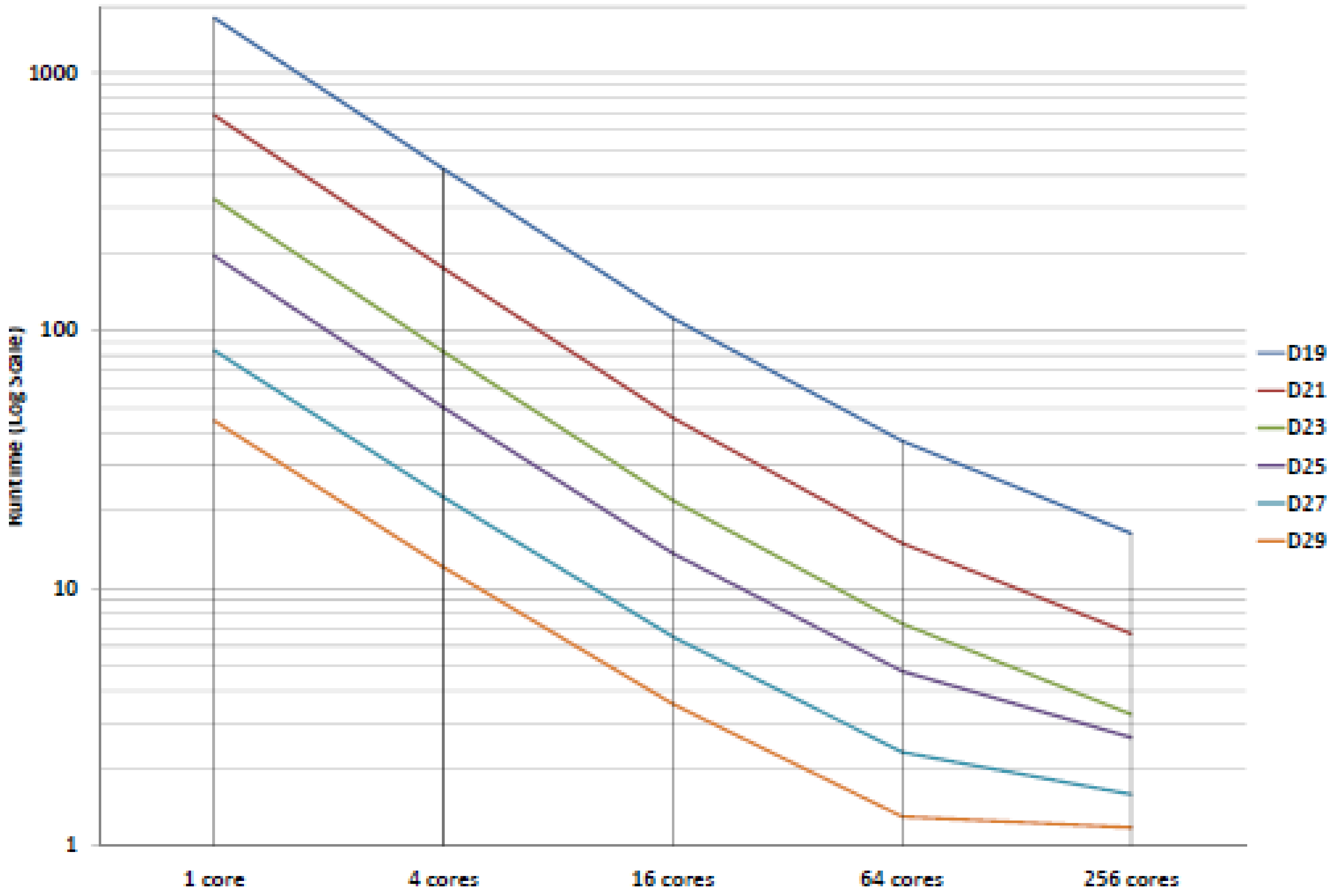
- New branching scheme is very suitable to parallelism
- Idea: explore **DP States** in different cores
  - Relatively little information needs to be shared
  - Most of the computational work involves computing relaxations/restrictions, done locally by each computer core
  - Easier to distribute load
- Joint work with Horst Samulowitz, Vijay Saraswat (IBM Research), and Ashish Sabharwal (Allen Inst.)

Parallel Search: Why bother?

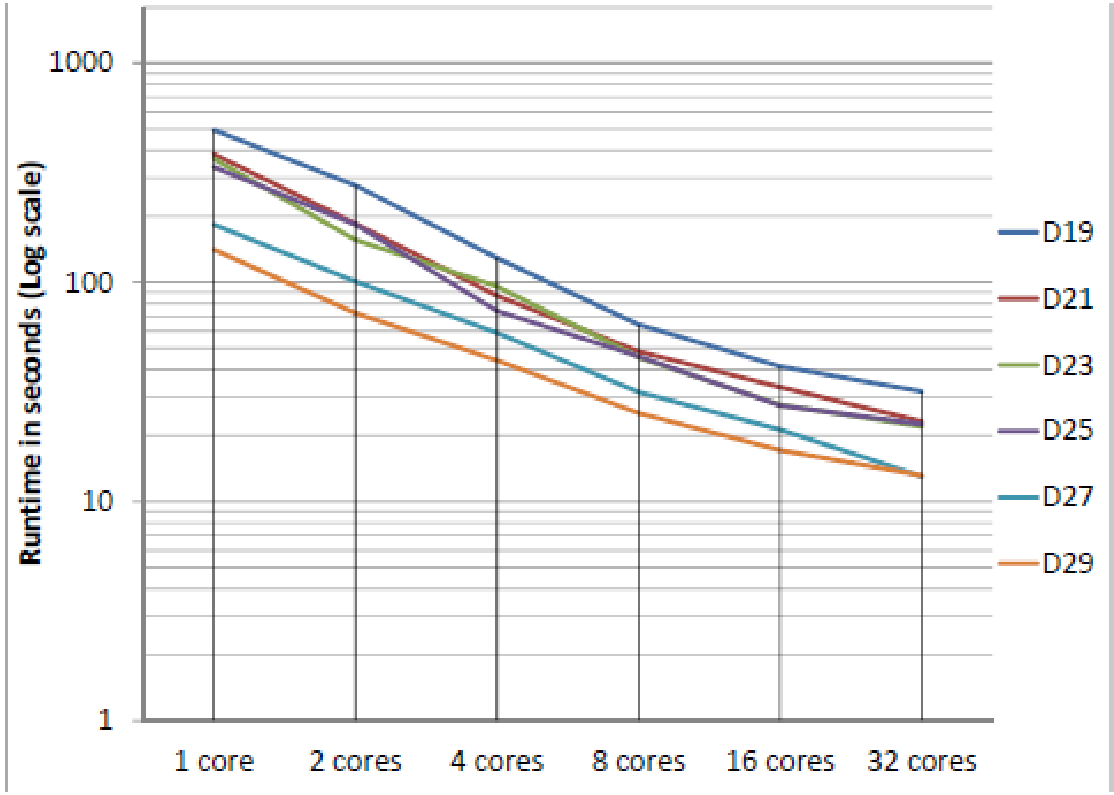
- Current technology
  - Integer Programming
    - Gurobi: Average speedup factor (Gu, 2013)
      - 1.7x on 5 cores
      - 1.8x on 25 cores
    - CPLEX (Mittleman, 2009)
      - 1.67x on 4 cores
  - SAT
    - 2013 SAT competition
      - 8x on 32 cores
  - Constraint Programming
    - Only focus on infeasible instances/finding all solutions

Parallel Search with Decision Diagrams

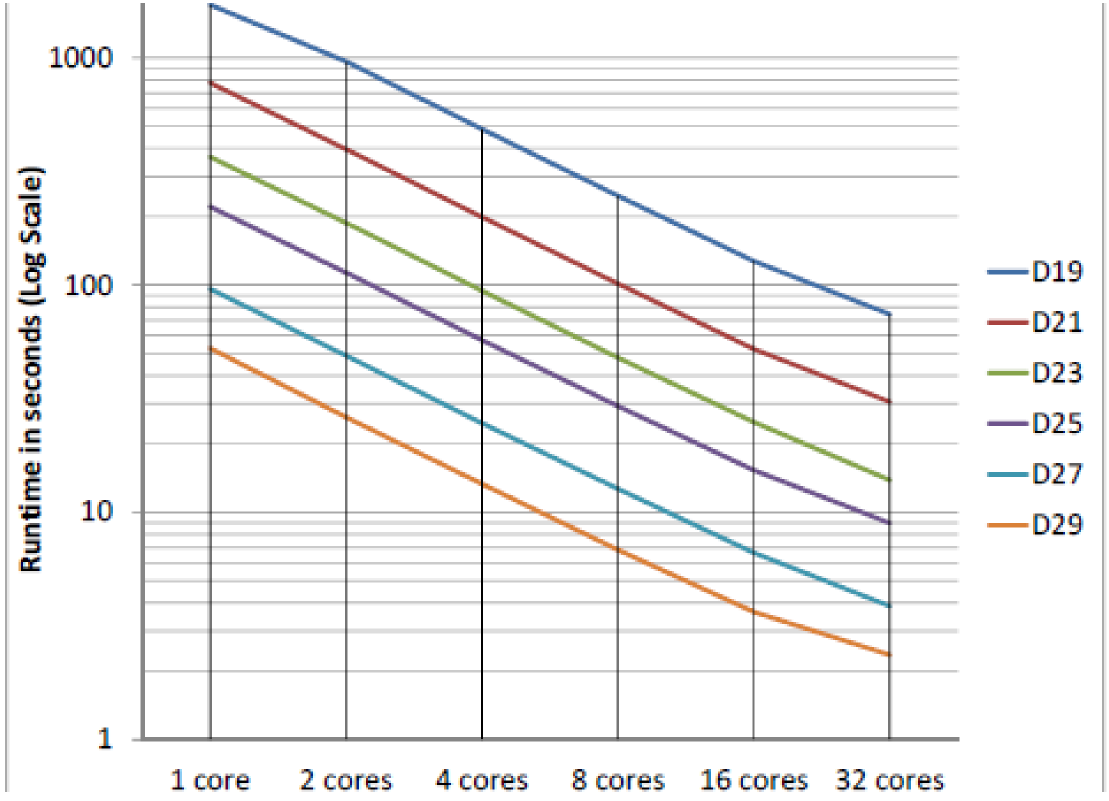
C125.9	**1 core**	**4 cores**	**16 cores**	**64 cores**	**256 cores**
Time to solve (s)	1100.91	277.07	70.74	19.53	8.07
Speedup	-	*3.97x*	*15.56x*	*56.37x*	*136.42x*



Runtime (Log Scale)
1000
100
10
1
1 core
4 cores
16 cores
64 cores
256 cores
D19
D21
D23
D25
D27
D29



CPLEX



Decision Diagrams

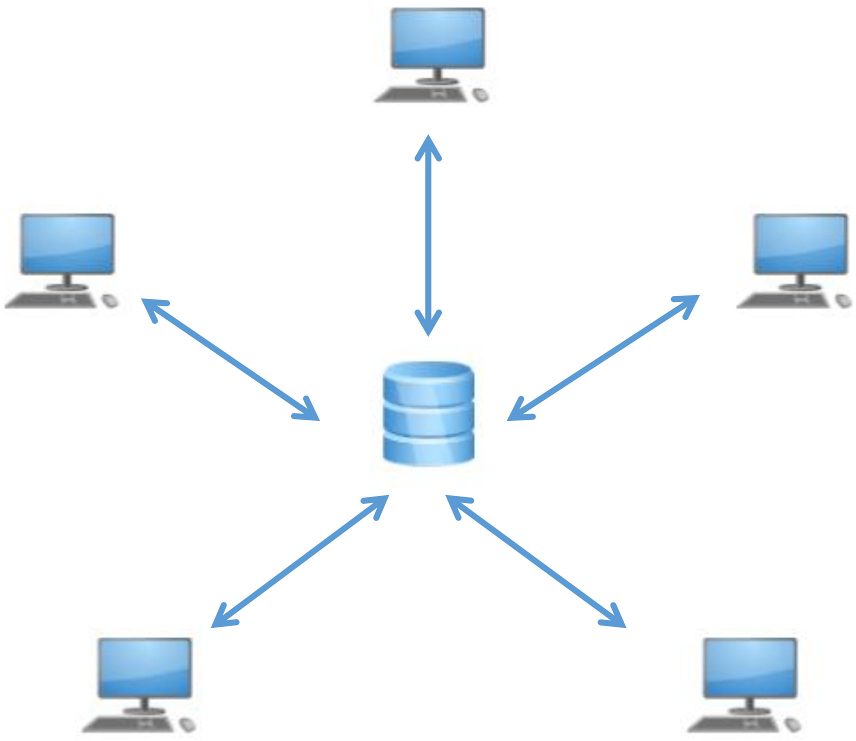
Thank you!

Decision Diagram Page:
http://www.andrew.cmu.edu/user/vanhoeve/mdd/

acire@utsc.utoronto.ca

Parallel Architecture

- We consider a **centralized architecture**
  - **Master** maintains a **pool of states** to process
  - **Workers** receive states, generate relaxed diagrams, and send new states to master
- Suitable to small architectures (up to 256 cores)



Master & Workers Pools

- Master keeps a **priority queue** of states
  - States with better optimization bounds have a higher priority of being explored
- Workers also keep a local priority queue
  - Relaxed (and restricted) decision diagrams are computed very quickly
  - Reduce communication to master
- Key issue: large memory consumption
  - Pools may grow quickly for very large problems
  - If memory is almost exceeded, priority queue becomes a regular queue (depth-first search)

Load Balancing

- Crucial question in many parallelization scheme
- In our case: How to distribute states among workers?
  - Too many nodes at once: many workers will be idle
  - Too few nodes: communication becomes bottleneck

Load Balancing

$$nodes\ to\ send = \min\left(c.\frac{size\ of\ pool}{number\ of\ cores}, \frac{avg\ states\ added}{c'}\right)$$

where **c** and **c’** are some constants (in our experiments, c = c’ = 2)

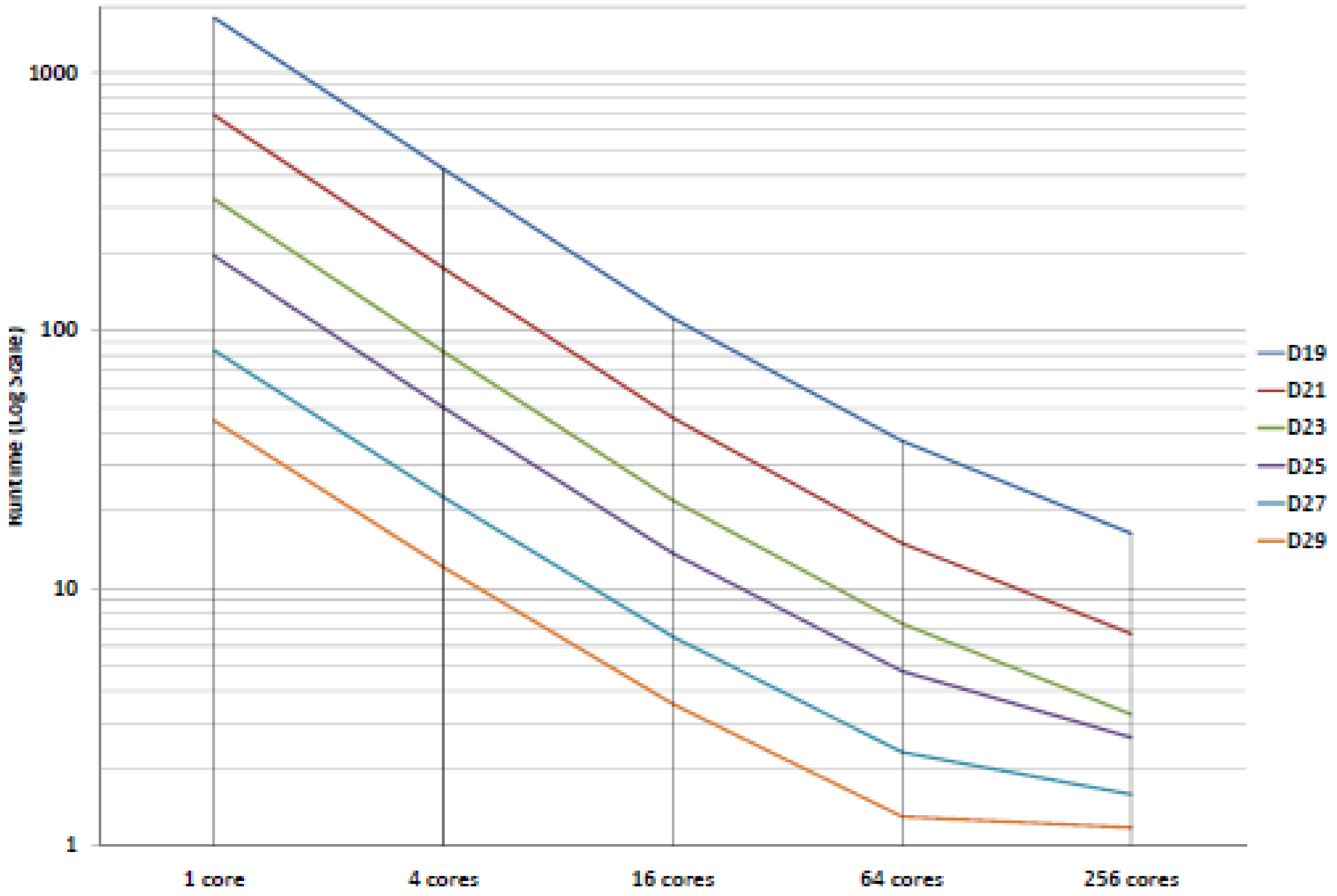
Load Balancing

75% of nodes with best optimization bounds

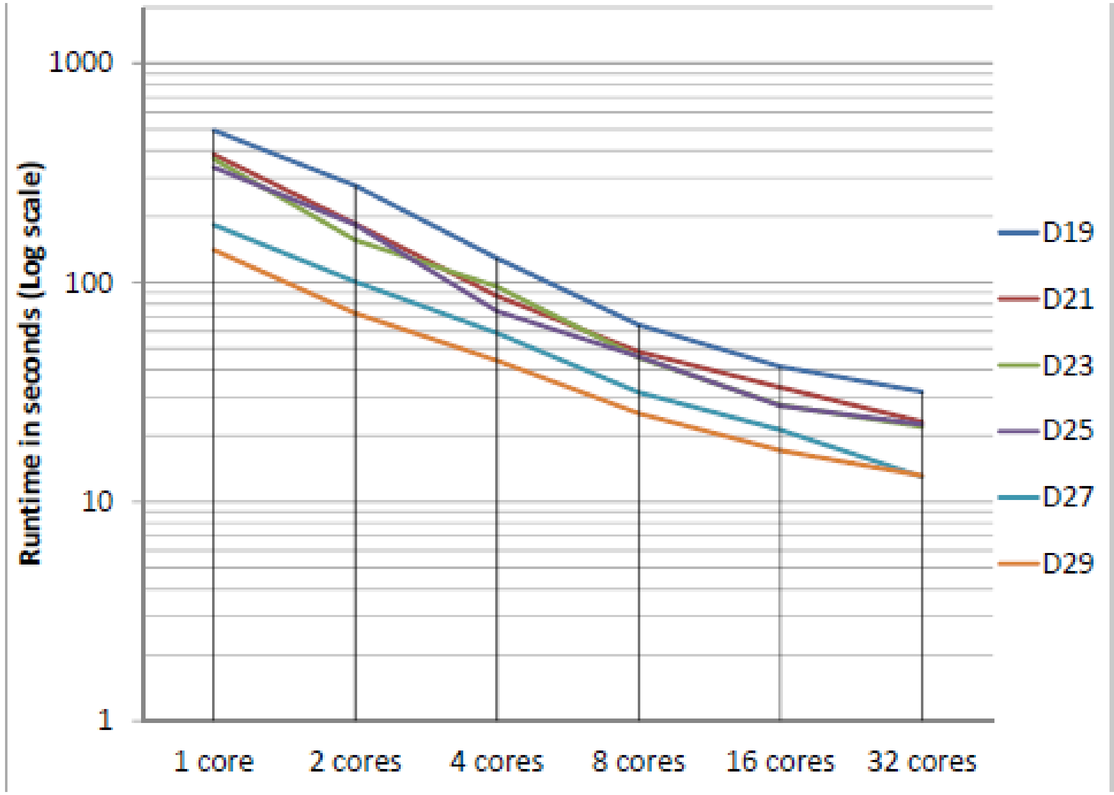
- Speed up the processing of promising nodes

Computational Results

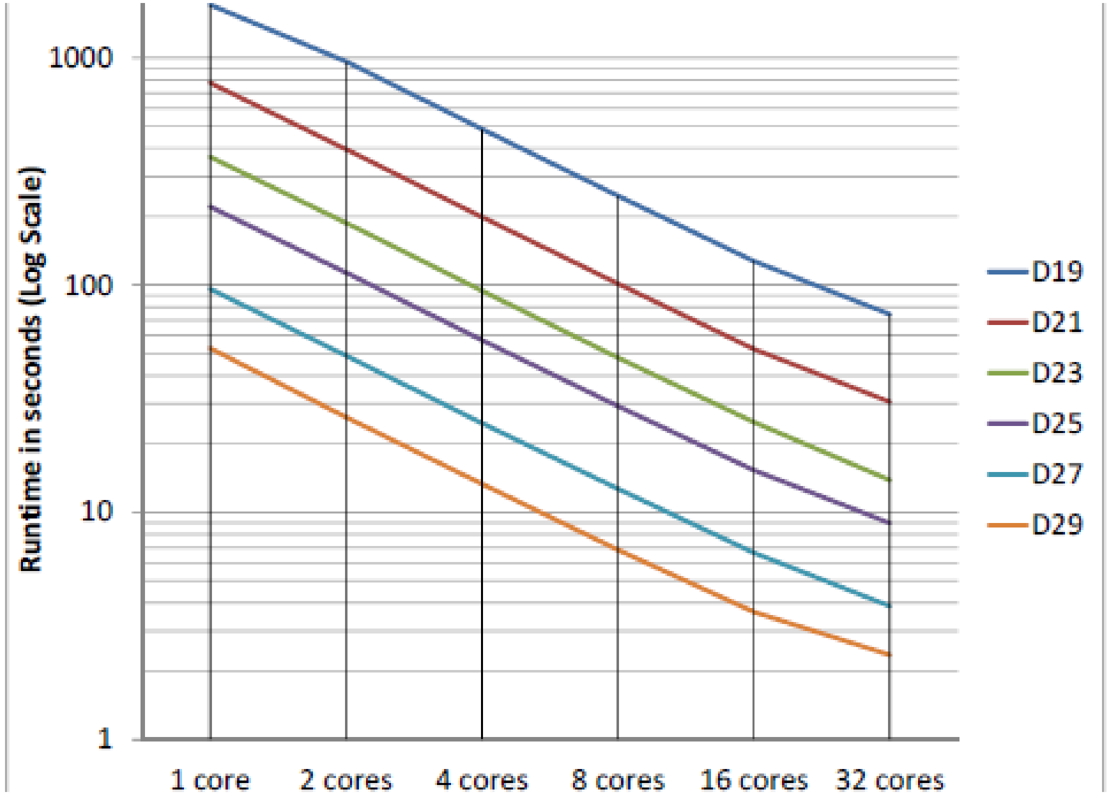
- Relaxed decision diagrams implemented in **C++**
- Parallel architecture implemented in **X10**
  - IBM X10 Team: Vijay Saraswat et al
  - x10-lang.org
- Tested in a computer cluster with 256 cores
  - 16 computers, each with 32 cores, 64 GB RAM



1000
100
10
1
Runtime (Log Scale)
1 core
4 cores
16 cores
64 cores
256 cores
D19
D21
D23
D25
D27
D29



CPLEX



Decision Diagrams

Other results

- Also observe same behaviour for other problem classes
  - Proved optimality for some maxcut instances for the first time
  - Testing on some variations of constrained TSP

- Other architectures
  - Work-stealing models

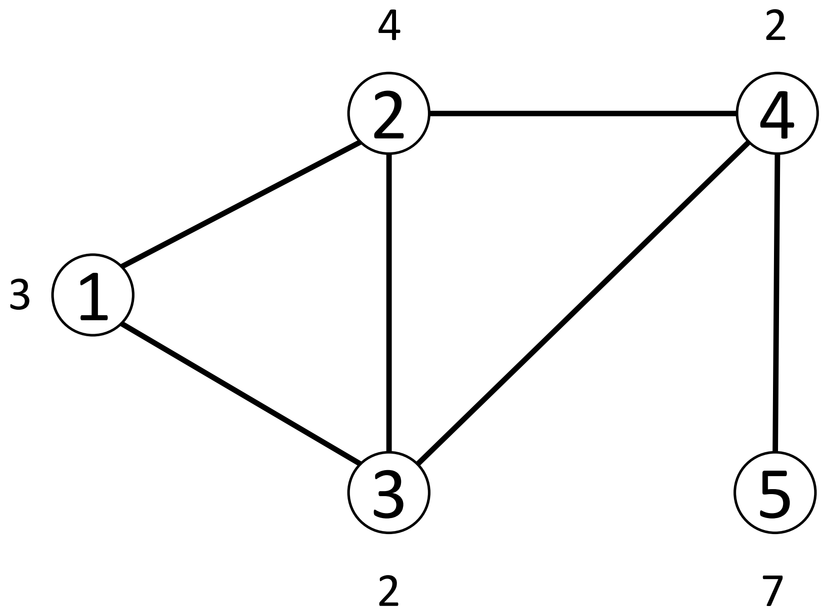
Thank you!

Relaxed Decision Diagrams

- Computational study on the max. independent set problem
  - Able to provide tighter bounds than integer programming models
- Application on Single-Machine Scheduling Problems
  - Closed open TSPLib instances, orders of magnitude improvement over constraint programming models, plus theoretical properties
- Application on Timetabling Problems
  - Orders of magnitude speed up in solving times compared to state-of-the-art approaches, plus theoretical properties

Modeling Framework

Ex.: Maximum independent set problem

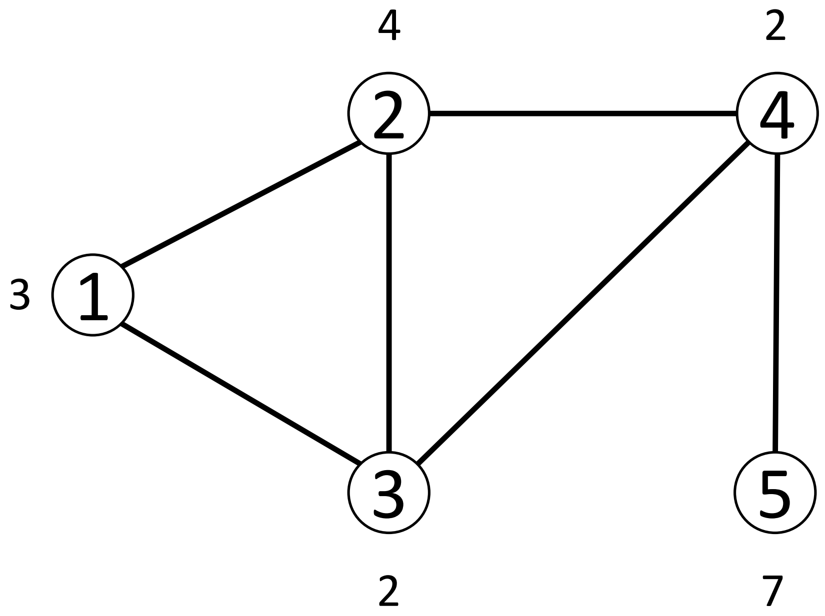


- Our model: **Dynamic Programming**
  - Exploit recursiveness
  - Solved by **stages**
  - Passing from one stage to another corresponds to transitioning from a **state** to another

- Decision diagram: **State-Transition Graph**
  - **Nodes** corresponds to **states**
  - **Arcs** are **state transitions**
  - **Arc weights** are **transition costs**

Modeling Framework

Ex.: Maximum independent set problem



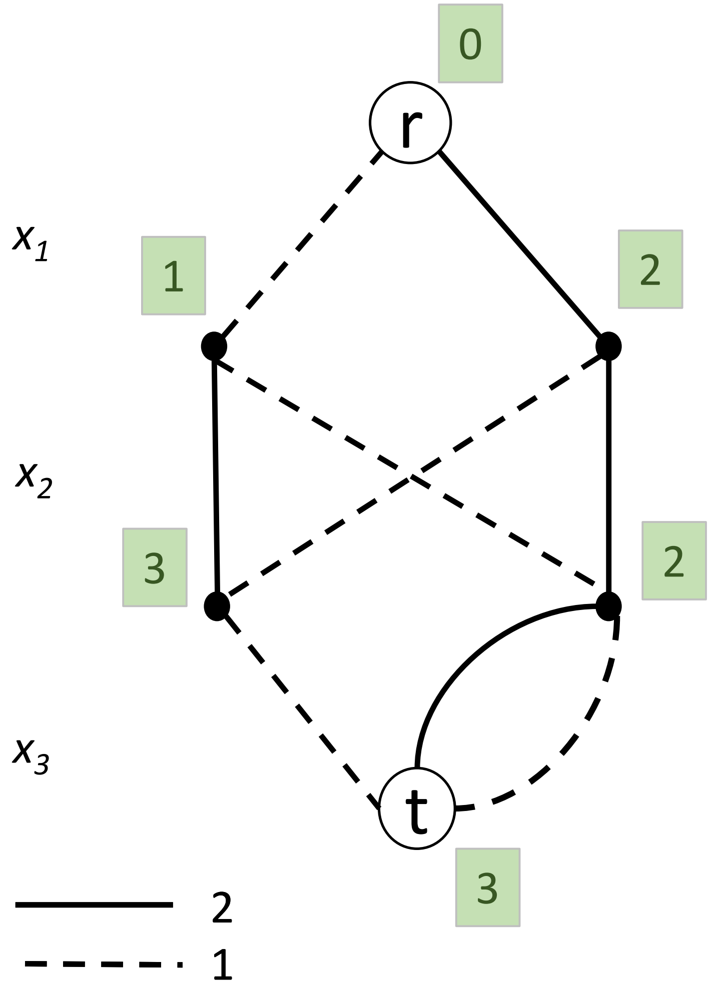
- DP model for the maximum independent set:

$$V_i(S) = \begin{cases} max\,\{V_{i-1}(S \setminus \{i\}), V_{i-1}(S \setminus N(i)) + 1\}, & i \in S \\ V_{i-1}(S \setminus N(i)), & o.w. \end{cases}$$

$$V_i(\emptyset) = 0, \qquad i = 1, \ldots, 5$$

- **Highlights:**
  - Stage i: **select** vertex i
  - State: set of **eligible** vertices

Filtering



max $4x_1 + 4x_2 + x_3$

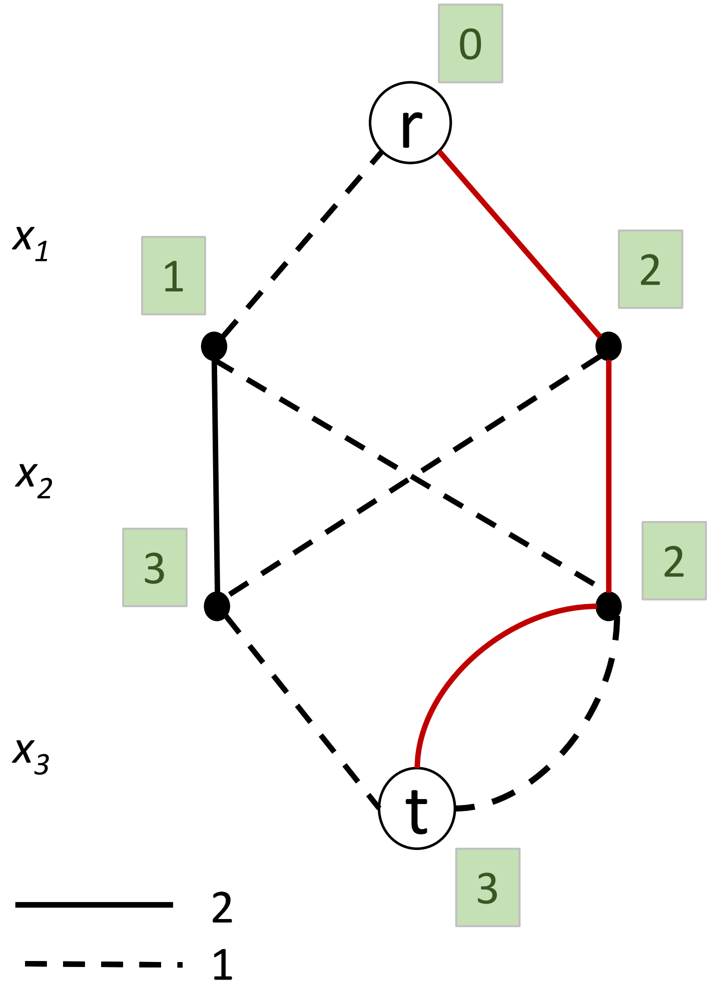
subject to

$$x_1 + x_2 + x_3 \leq 4$$

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

- **Max Width** = 2
- **State**: left-hand side of constraint

Filtering



max $4x_1 + 4x_2 + x_3$

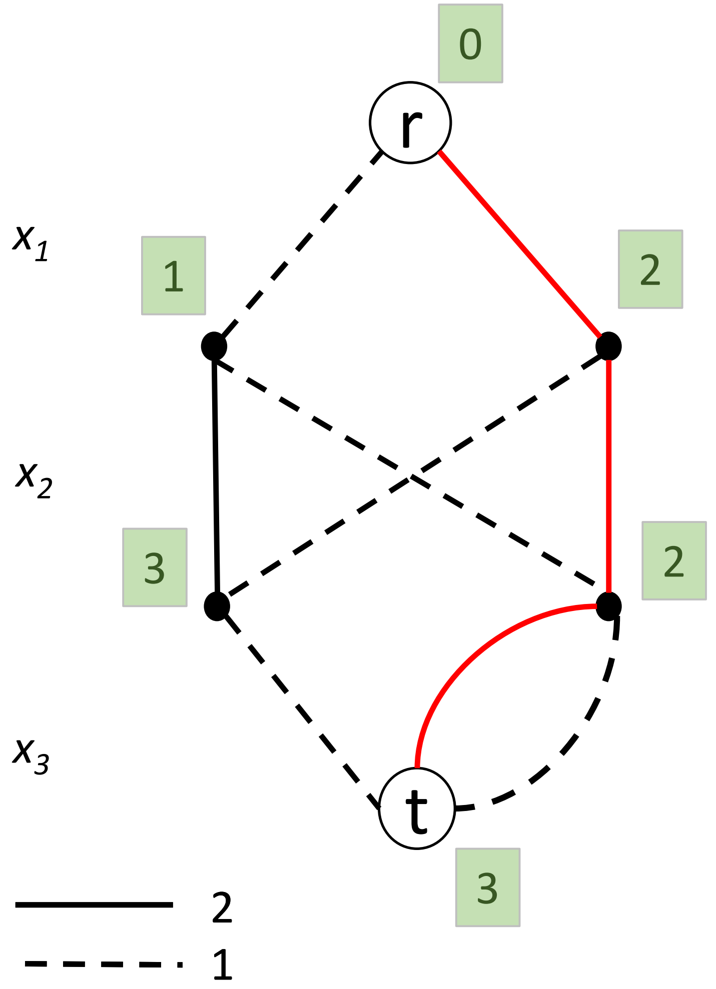
subject to

$$x_1 + x_2 + x_3 \leq 4$$

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

- **Max Width** = 2
- **State**: left-hand side of constraint
- Longest path: $x_1 = x_2 = x_3 = 1$

Filtering

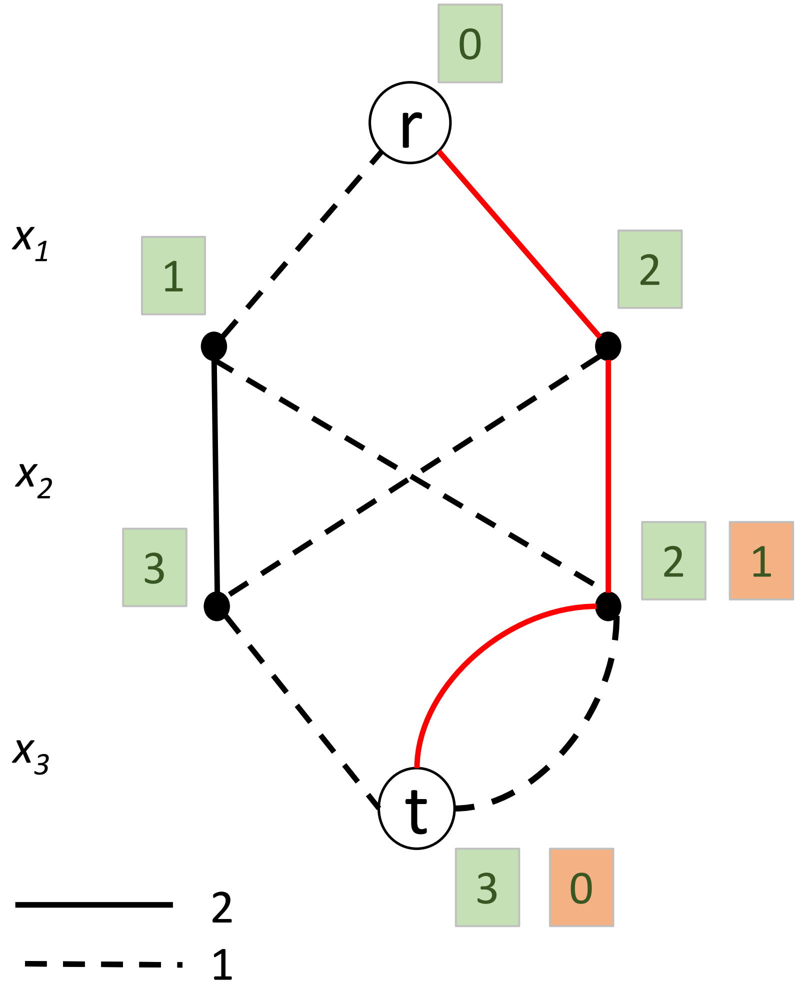


max $4x_1 + 4x_2 + x_3$
subject to
$x_1 + x_2 + x_3 \leq 4$
$x_1, x_2, x_3 \in \{1,2\}$

- Note that **top-down** is a **forward recursion:**

$$V_i(\ldots) = V_{i-1}(\ldots) + \ldots$$

Filtering



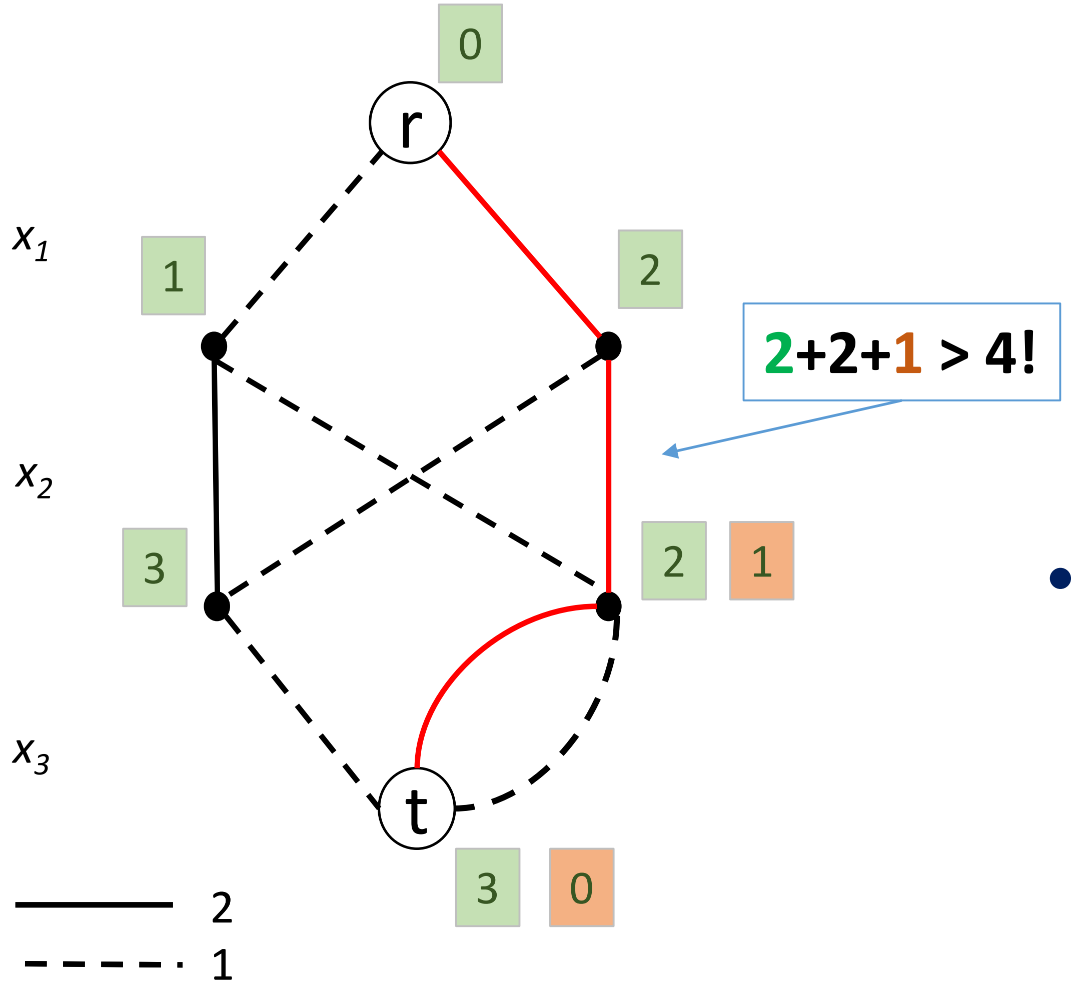
$$\max\ 4x_1 + 4x_2 + x_3$$

subject to

$$x_1 + x_2 + x_3 \leq 4$$
$$x_1, x_2, x_3 \in \{1,2\}$$

- But what happens when we do a **backward recursion**?

Filtering



max $4x_1 + 4x_2 + x_3$

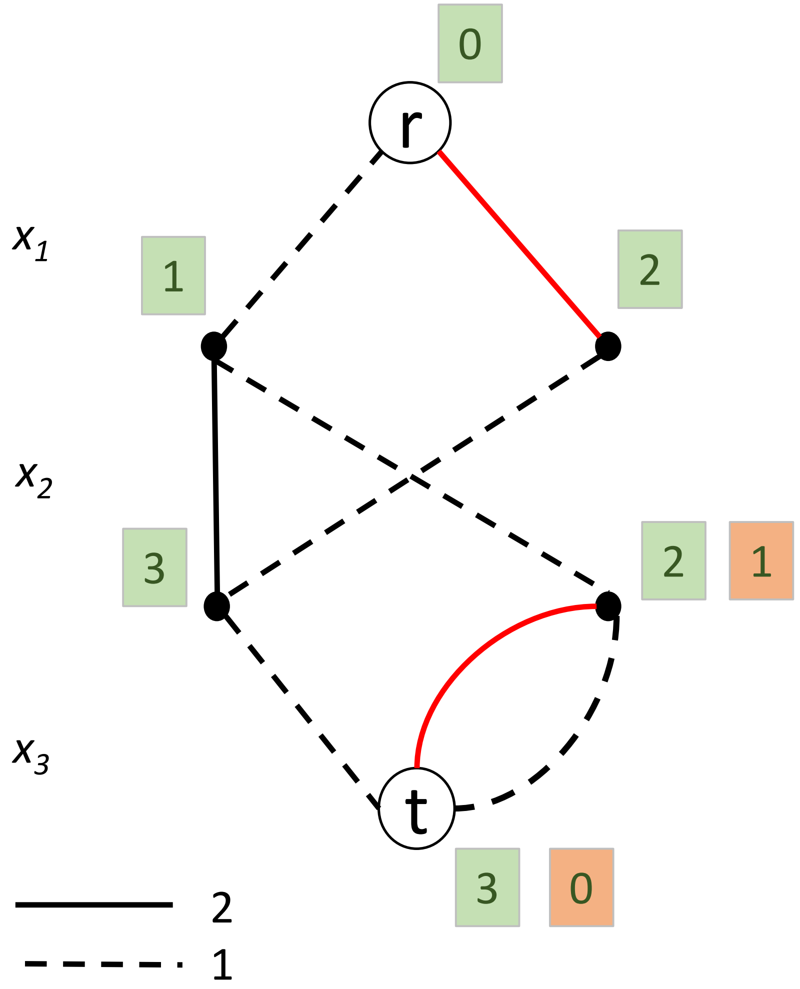
subject to

$$x_1 + x_2 + x_3 \leq 4$$

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

- But what happens when we do a **backward recursion**?

Filtering



max $4x_1 + 4x_2 + x_3$

subject to

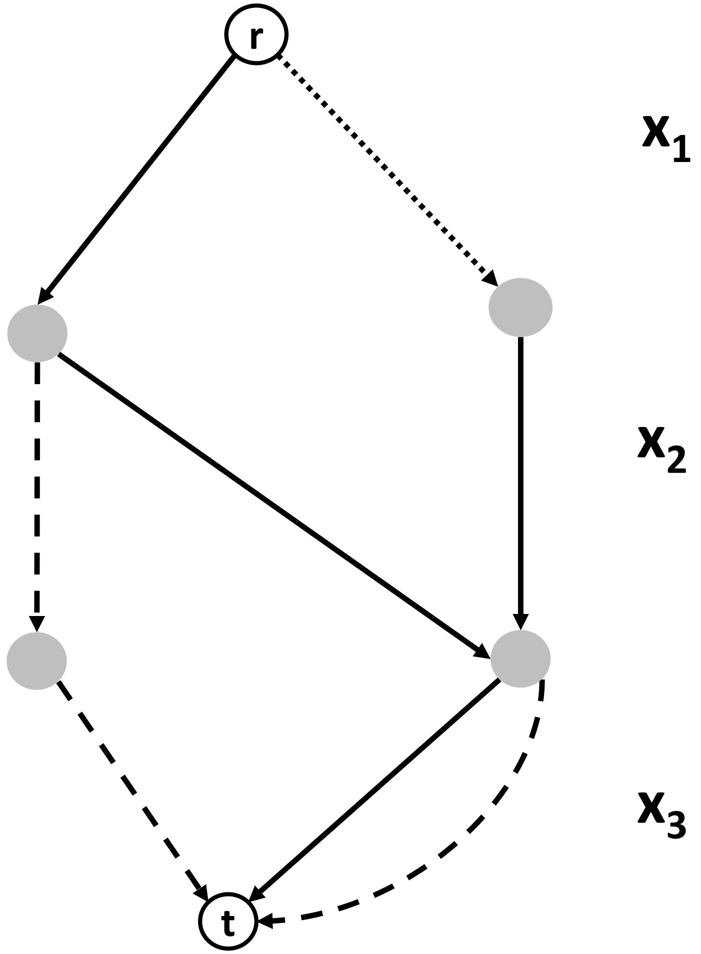
$$x_1 + x_2 + x_3 \leq 4$$

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

- Underlying concept: Use **“redundant” DP formulations** to remove arcs, e.g.:

$$\mathbf{V'_i(\ldots) = V'_{i-1}(\ldots) + V'_{i+1}(\ldots) + \ldots}$$

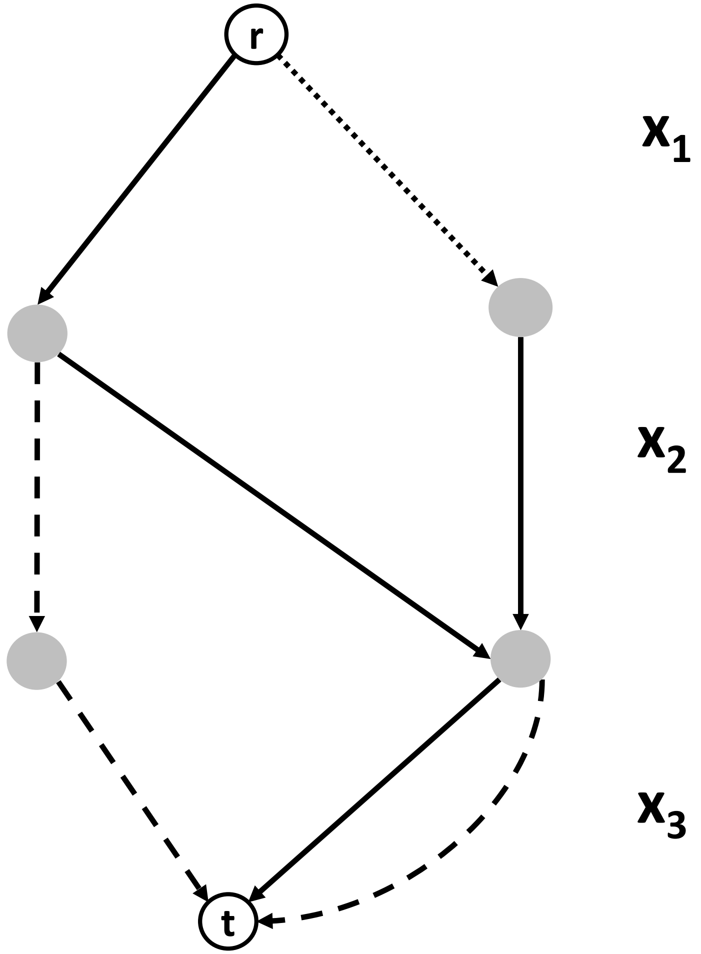
Some theoretical insights



- Let **X** the set of solutions represented by an MDD
- Optimizing a linear function **f** over the MDD is equivalent to solving the **LP problem**:

Minimize cx **subject to** **x is a flow from r to t**	=	**Minimize cx** **subject to** **x ∈ conv(X)**

Some theoretical insights



- Let **Ax ≥ b** be a set of constraints that we dualize over the MDD.
- If z* is the optimal shortest path after dualization, then

$$\mathbf{z^* =} \begin{array}{l} \textbf{Minimize cx} \\ \textbf{subject to} \\ \mathbf{Ax \geq b} \\ \mathbf{x \in conv(X)} \end{array}$$