Decision Diagrams for Discrete Optimization, Constraint Programming, and Integer Programming

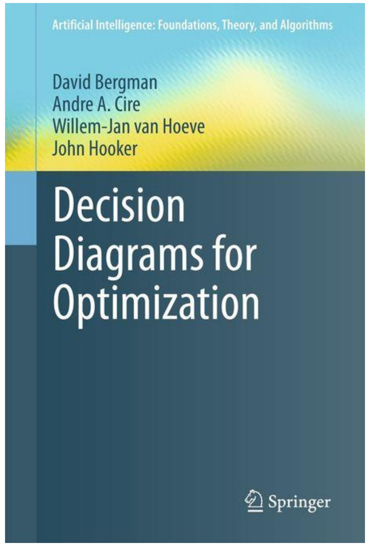
Willem-Jan van Hoeve

Master Class in Hybrid Methods for Combinatorial/Mixed Optimization

Toulouse, June 4-5, 2018

Acknowledgments

David Bergman, Andre Cire, Danial Davarnia, Samid Hoda, John Hooker, Amin Hosseininasab, Brian Kell, Joris Kinable, Ashish Sabharwal, Horst Samulowitz, Vijay Saraswat, Marla Slusky, Christian Tjandraatmadja, Tallys Yunes



Agenda

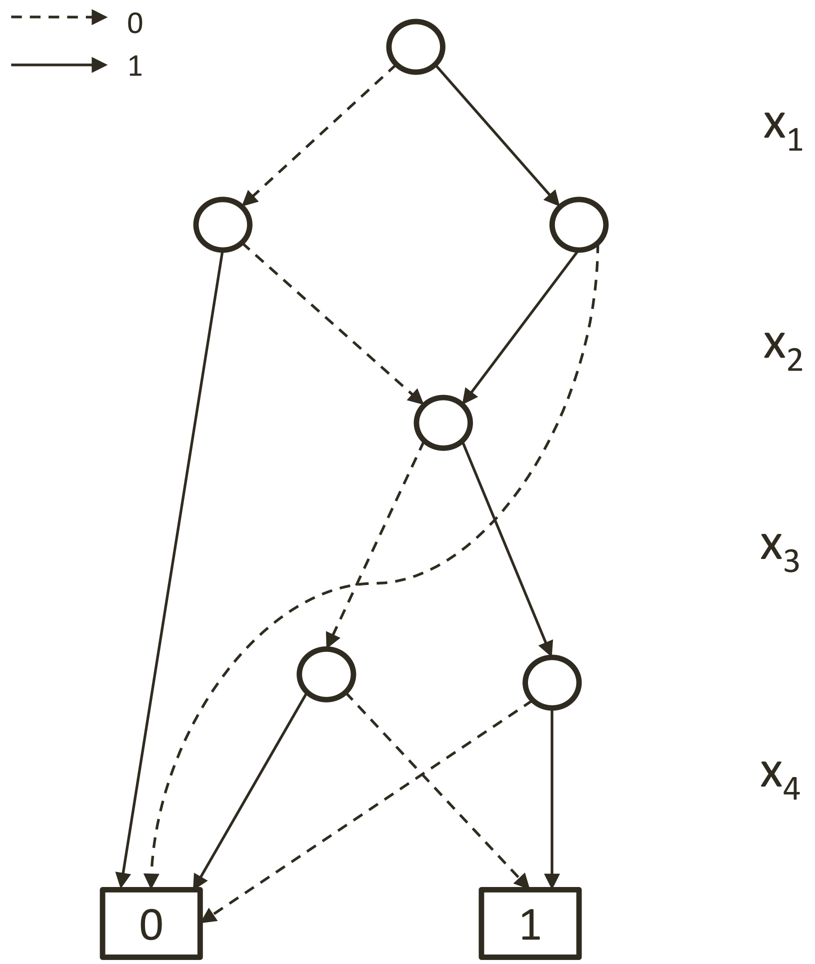
- What are Decision Diagrams?
- Discrete Optimization with Decision Diagrams
  - Modeling, Relaxation/Restriction, Search
- Constraint Programming with Decision Diagrams
  - Constraint Propagation, Scheduling Applications
- Integer Programming with Decision Diagrams
  - Integrate Decision Diagrams in Branch-and-Bound

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**
…	…	…	…	**…**

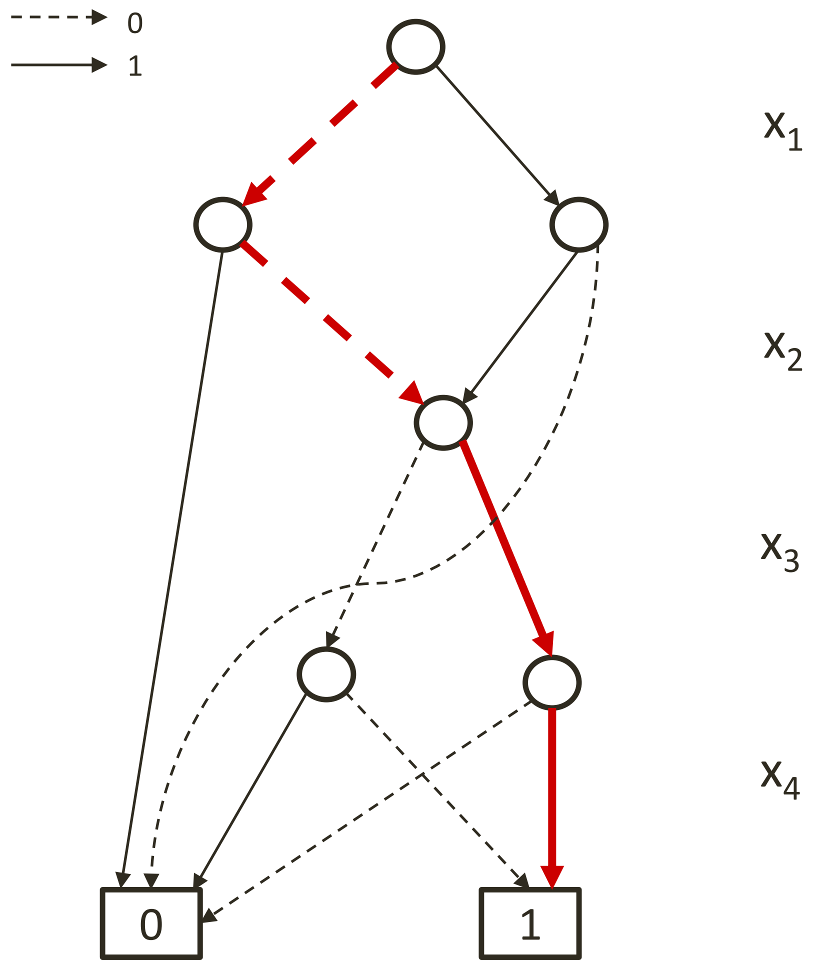


Decision Diagrams

- Graphical representation of **Boolean functions**

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

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**
…	…	…	…	**…**

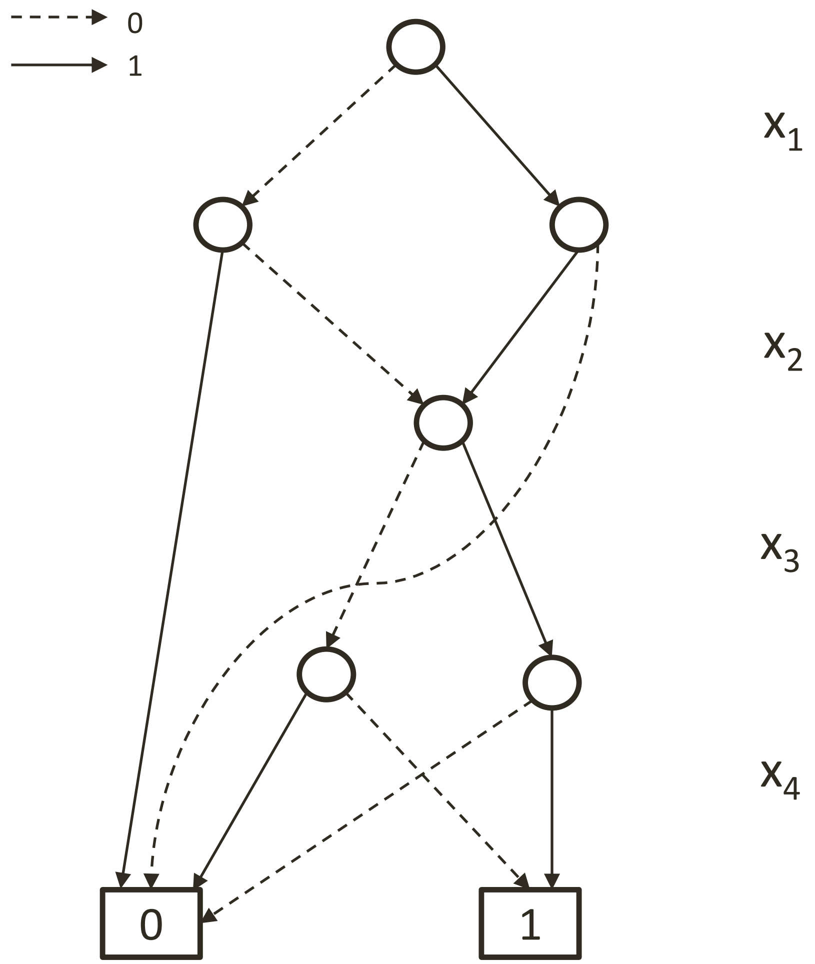


Decision Diagrams

- Graphical representation of **Boolean functions**

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

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



Brief Historic Background

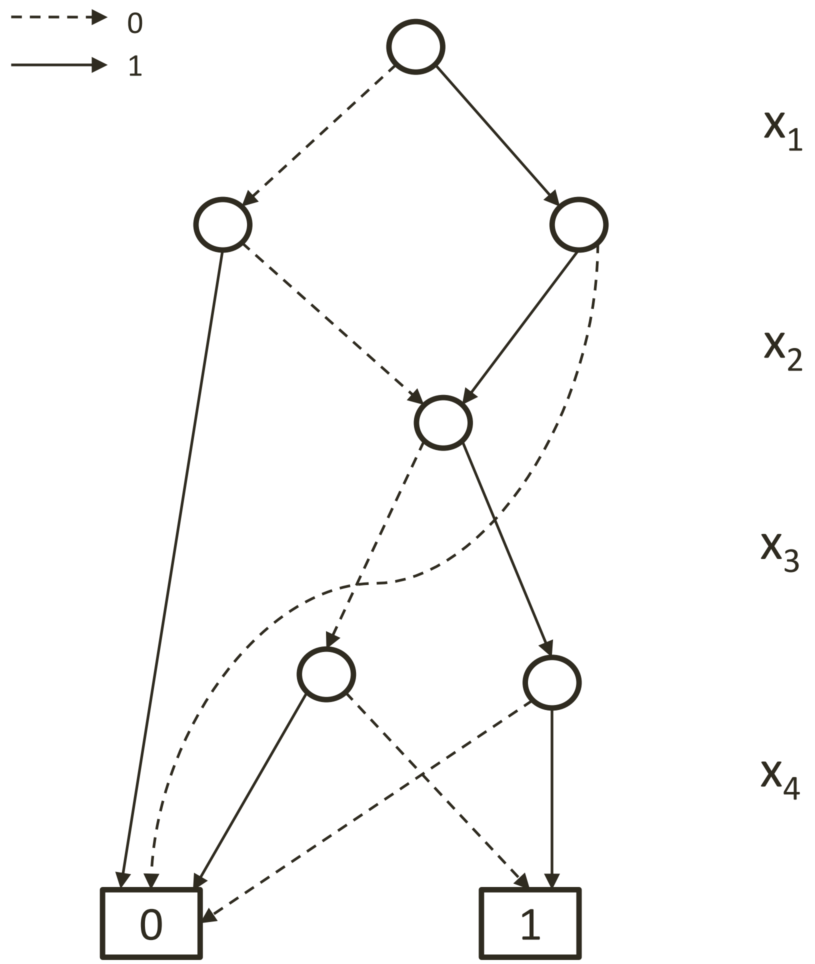
- Widely used in computer science [Lee, 1959; Akers, 1978; Bryant, 1986]
  - original application areas: circuit design, verification
- Usually *reduced ordered* BDDs/MDDs are applied
  - fixed variable ordering; minimal exact representation
- First applications to discrete optimization problems
  - BDD-based IP solver [Lai et al., 1994]
  - set bounds propagation in CP [Hawkins, Lagoon, Stuckey, 2005]
  - IP cut generation [Becker et al., 2005] [Behle & Eisenbrand, 2007] [Behle, 2007]
  - post-optimality analysis [Hadzic & Hooker, 2006, 2007]
- *Relaxed* Decision Diagrams [Andersen, Hadzic, Hooker & Tiedemann, CP 2007]

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



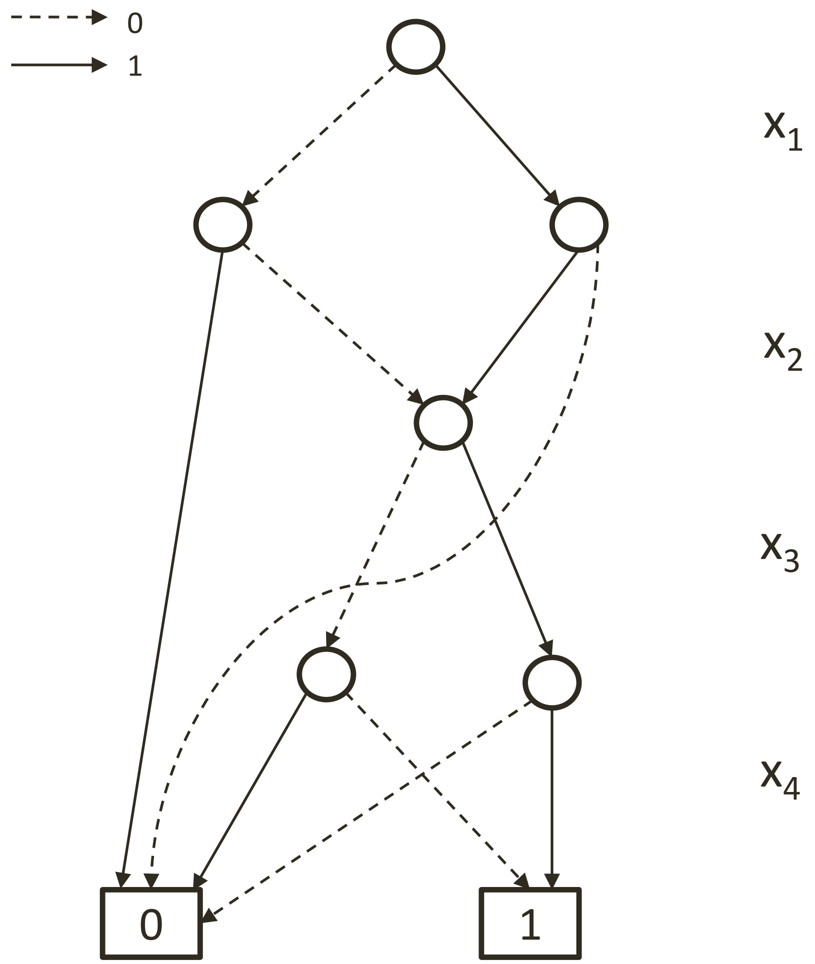
Decision Diagrams: Optimization View

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

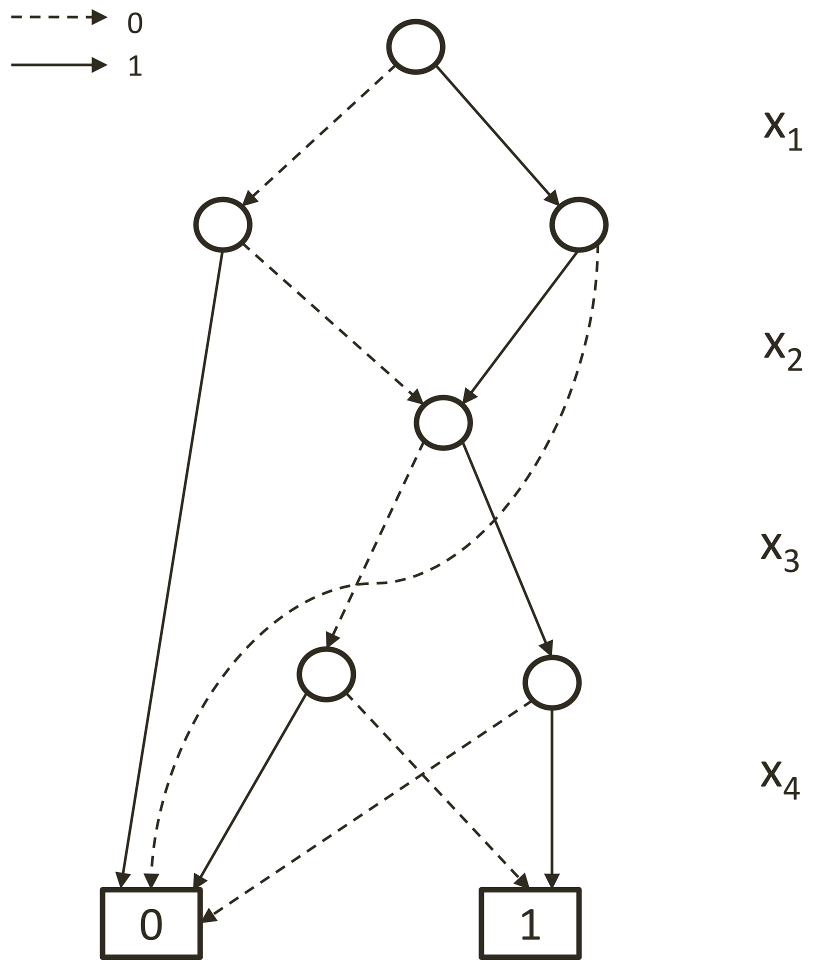
max $2x_1 + x_2 - 4x_3 + x_4$

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

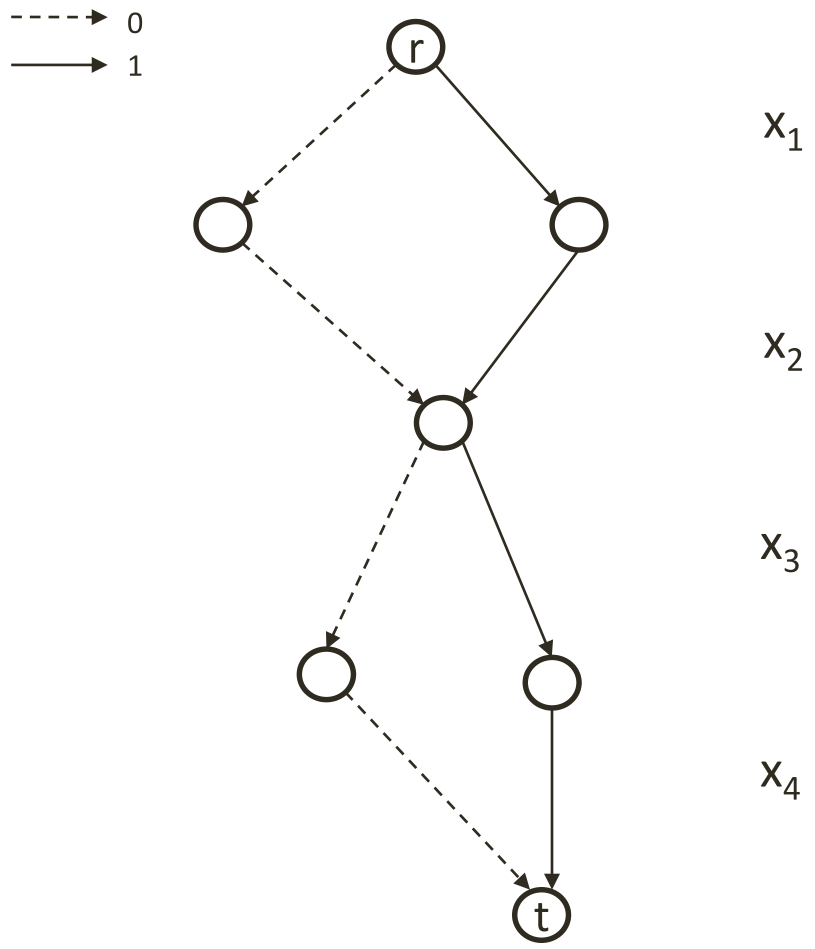
max $2x_1 + x_2 - 4x_3 + x_4$

subject to

$x_1 - x_2 = 0$

$x_3 - x_4 = 0$

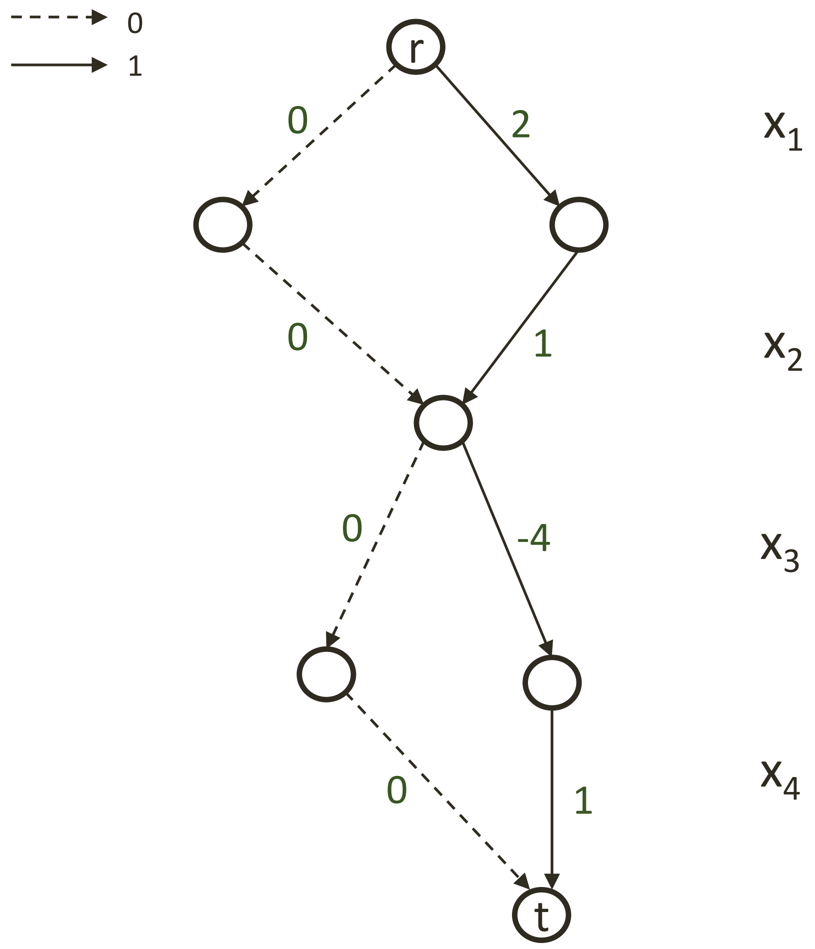
$x_1, x_2, x_3, x_4 \in \{0,1\}$



Decision Diagrams: Optimization View

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

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



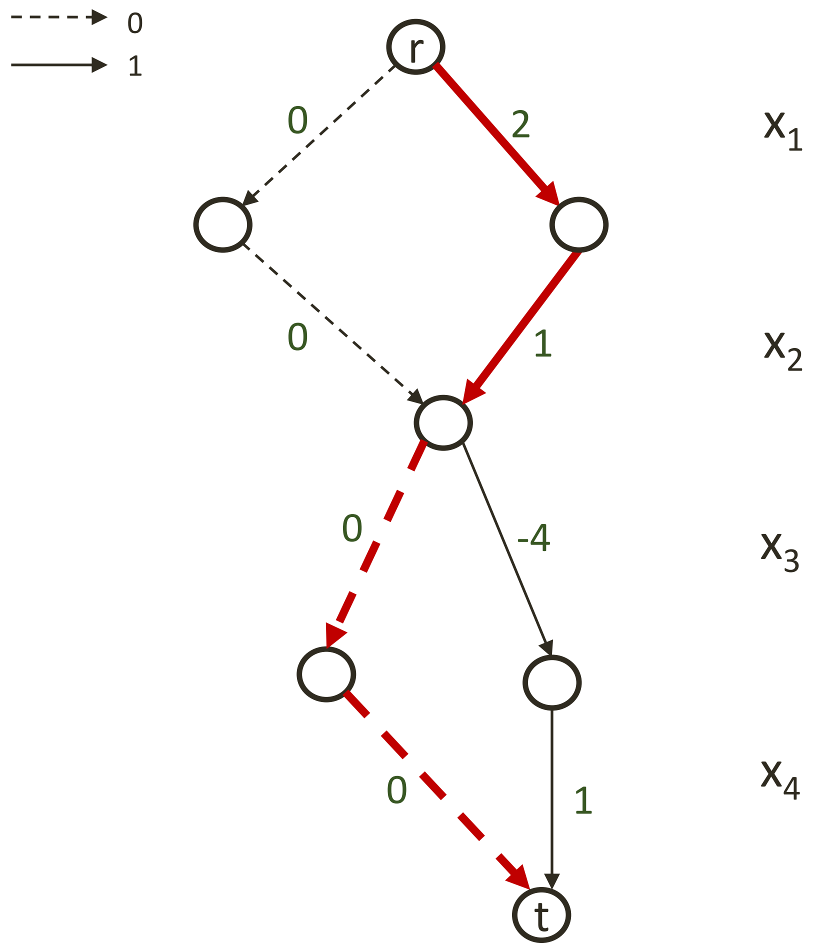
- Maximizing a linear (or separable) function:
  - Arc lengths: contribution to the objective
  - Longest path: optimal solution

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

Towards Generic Discrete Optimization

Relaxation Methods

MIP: Linear Programming Relaxation
DD: Relaxed Decision Diagram

Modeling Framework

MIP: Linear Inequalities
DD: Dynamic Programming

Primal Heuristics

MIP: Feasibility Pump, RINS, …
DD: Restricted Decision Diagrams

Generic Optimization Techniques

E.g., MIP, MINLP, CP, SAT, …

Inference

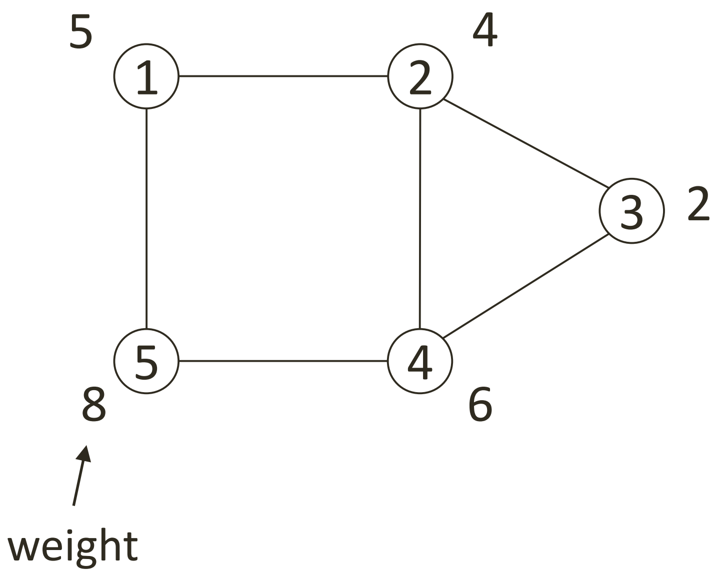
MIP: Valid linear cuts
DD: Propagation, cuts

Search

MIP: Branch and bound (variable-based)
DD: Branch and bound (state-based)

Modeling Framework

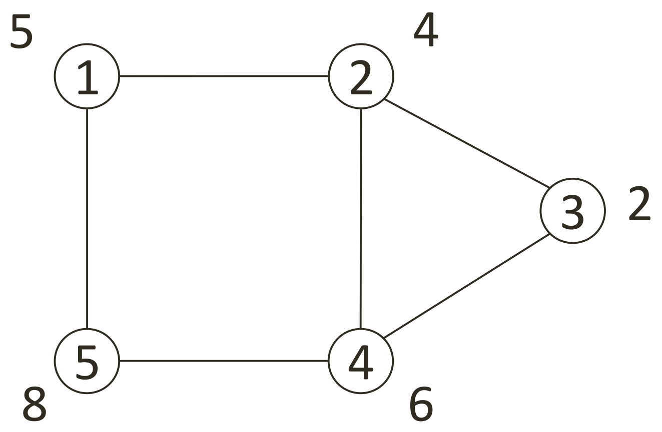
Example: Maximum Independent Set Problem



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

Modeling Framework

Example: Maximum Independent Set Problem

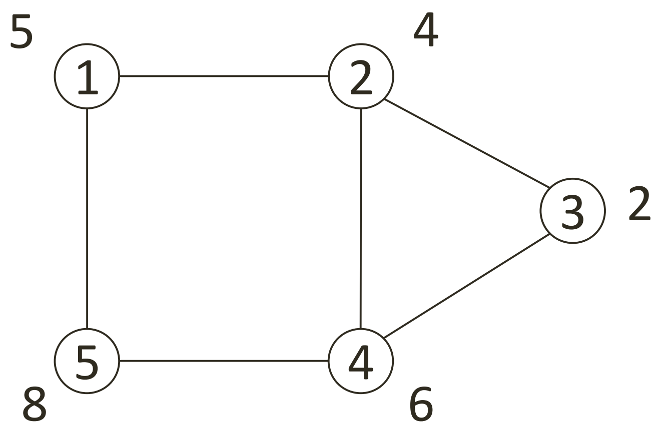


Integer Programming Formulation:

$$
\begin{aligned}
\text{max} \quad & 5x_1 + 4x_2 + 2x_3 + 6x_4 + 8x_5 \\
\text{subject to} \quad & x_1 + x_2 \leq 1 \\
& x_1 + x_5 \leq 1 \\
& x_2 + x_3 \leq 1 \\
& x_2 + x_4 \leq 1 \\
& x_3 + x_4 \leq 1 \\
& x_4 + x_5 \leq 1 \\
& x_1, x_2, x_3, x_4, x_5 \in \{0,1\}
\end{aligned}
$$

Modeling Framework

Example: 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

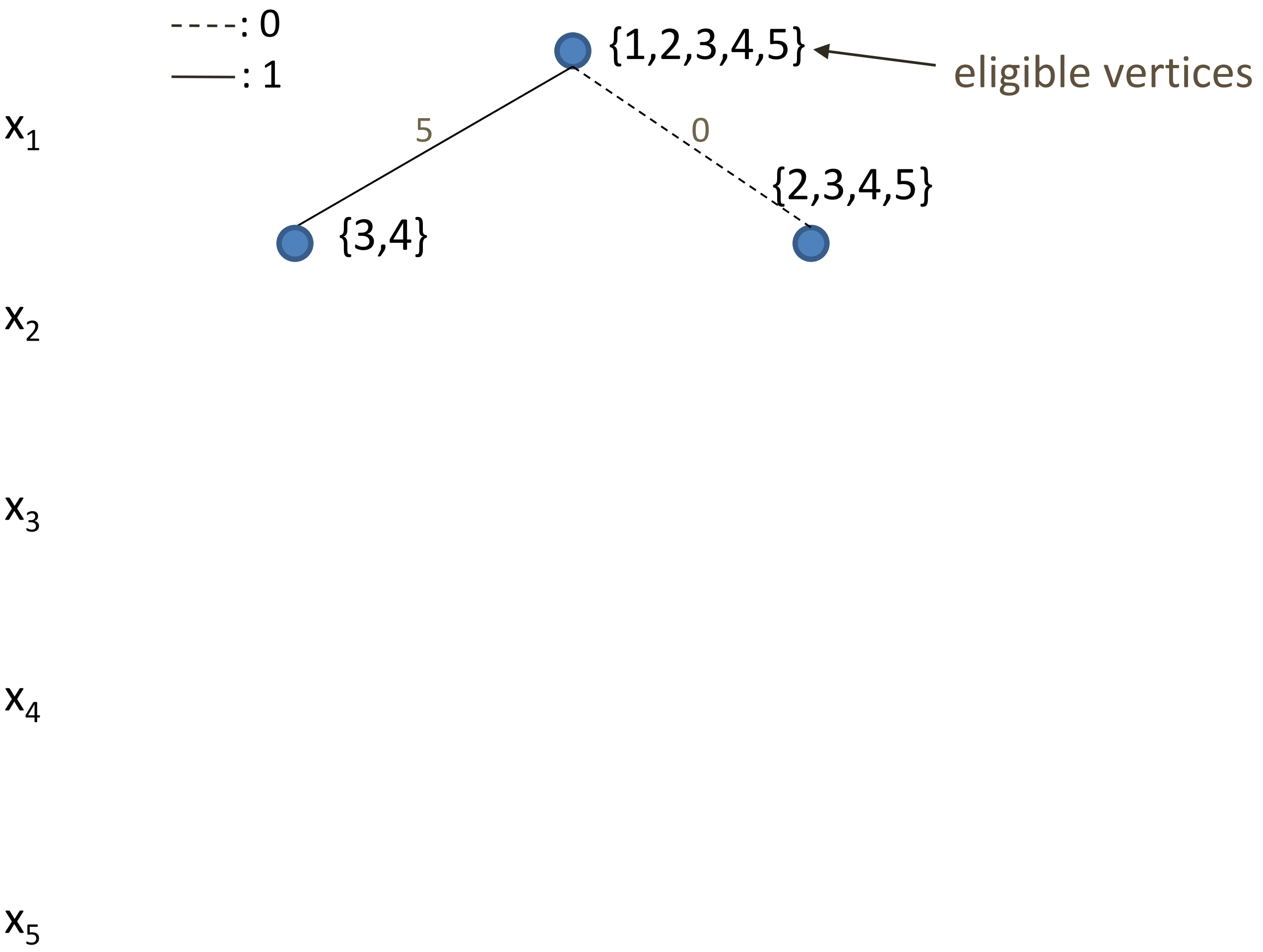
- Formal model:

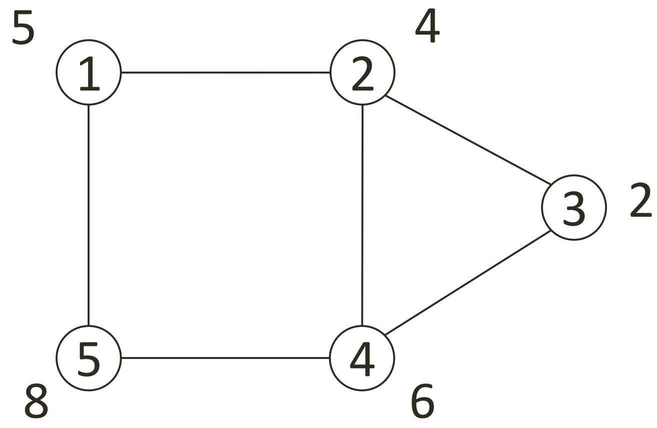
$$V_i(S) = \begin{cases} max\left\{V_{i-1}(S \setminus \{i\}), V_{i-1}\big(S \setminus N(i)\big) + w_i\right\}, & i \in S \\ V_{i-1}(S), & o.w. \end{cases}$$

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

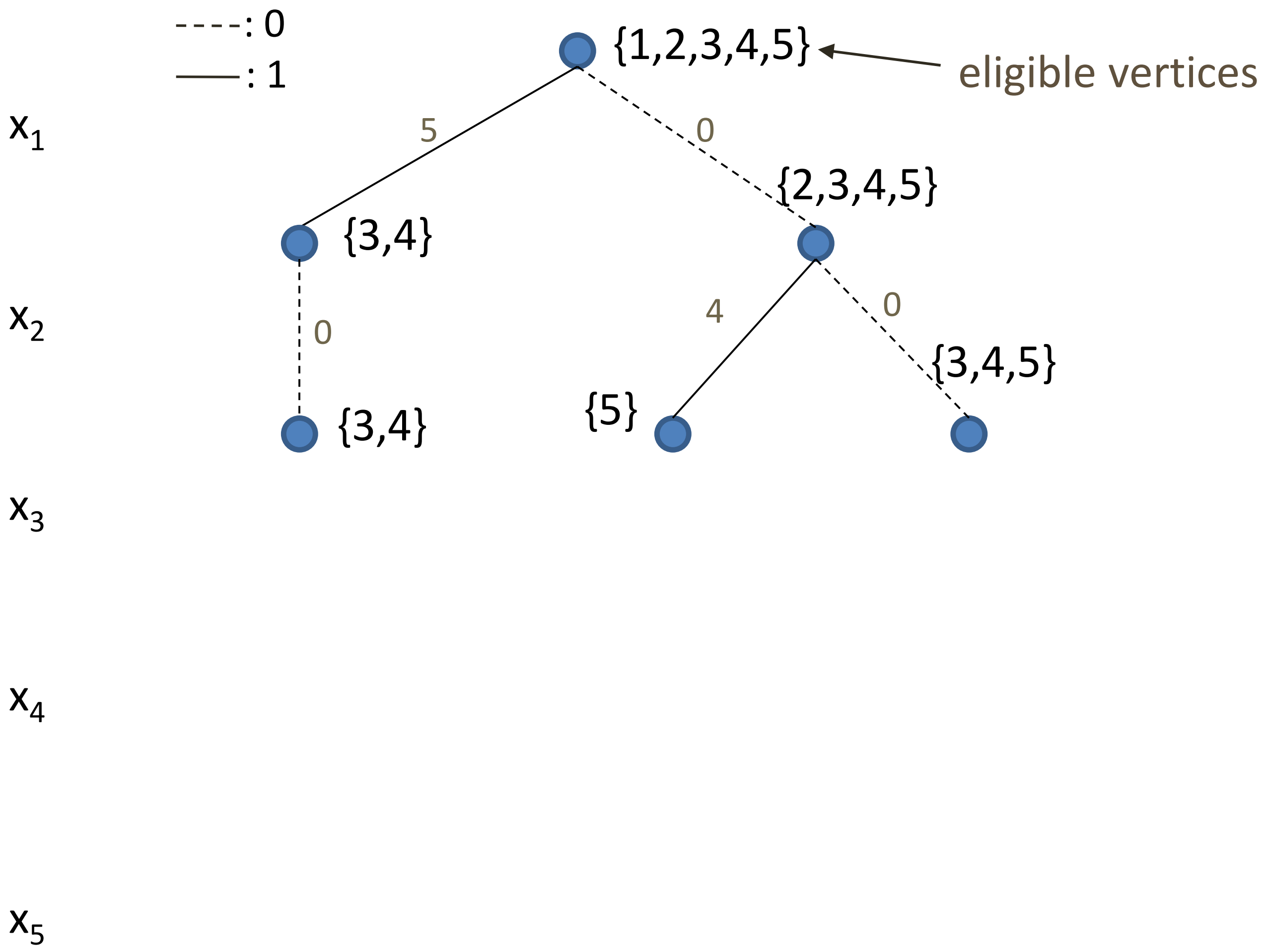
($N(i)$ = i + its neighbors)

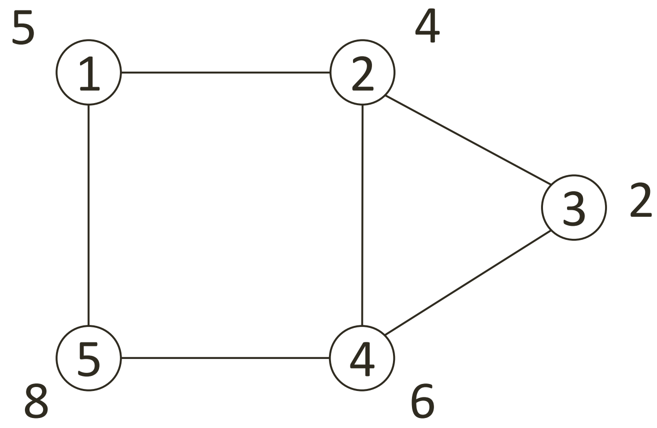
Maximum Independent Set Problem



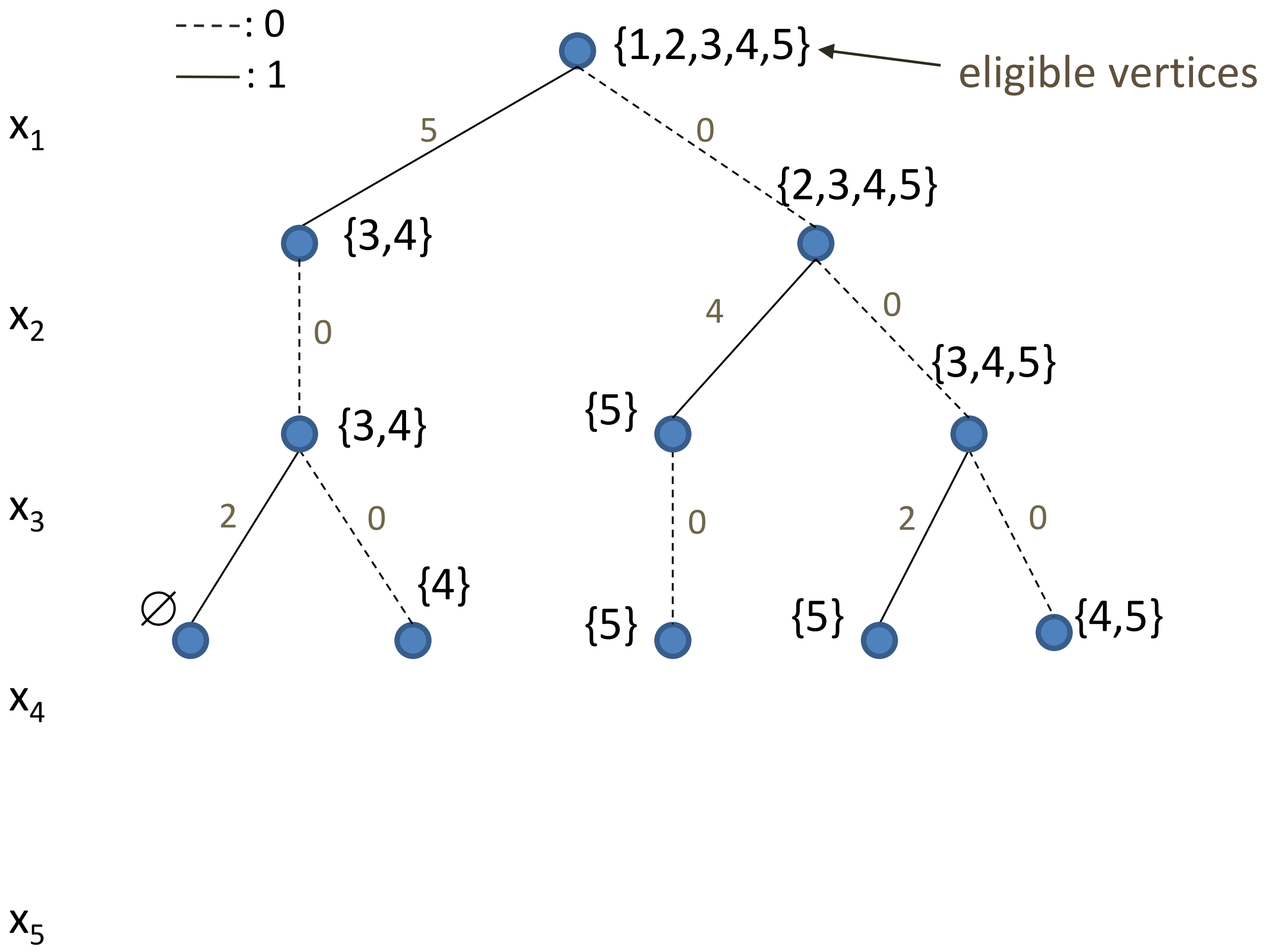


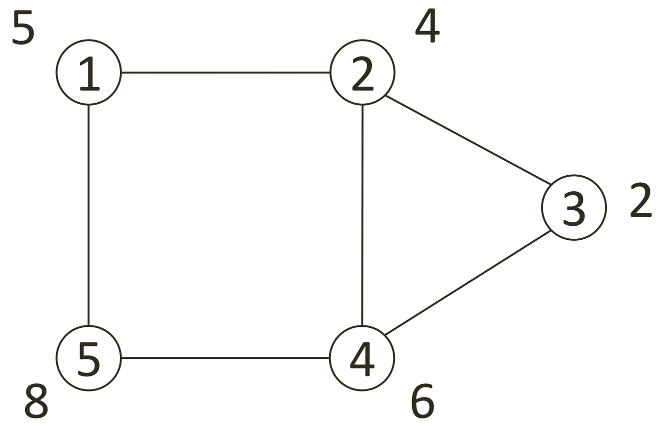
Maximum Independent Set Problem



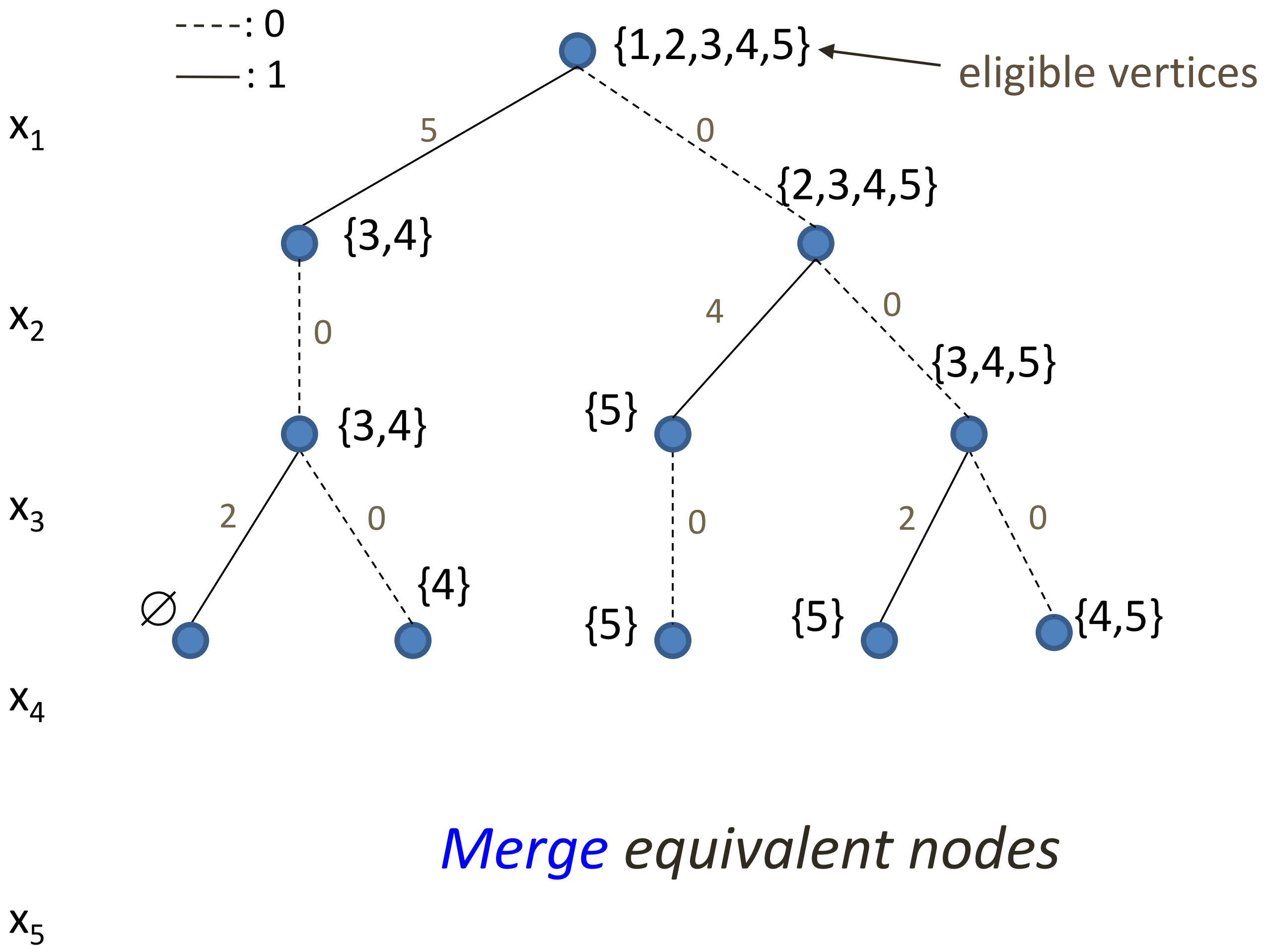


Maximum Independent Set Problem

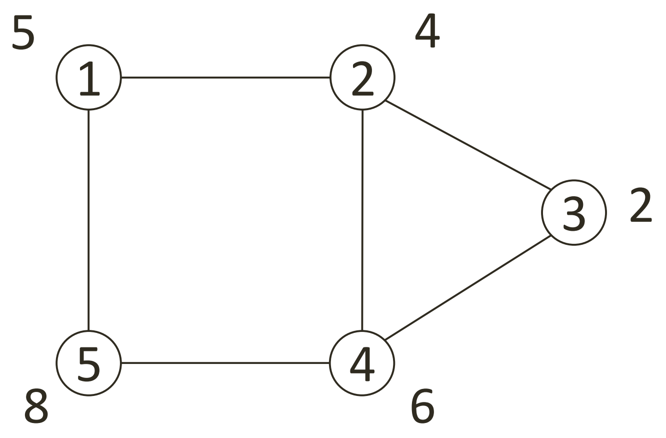




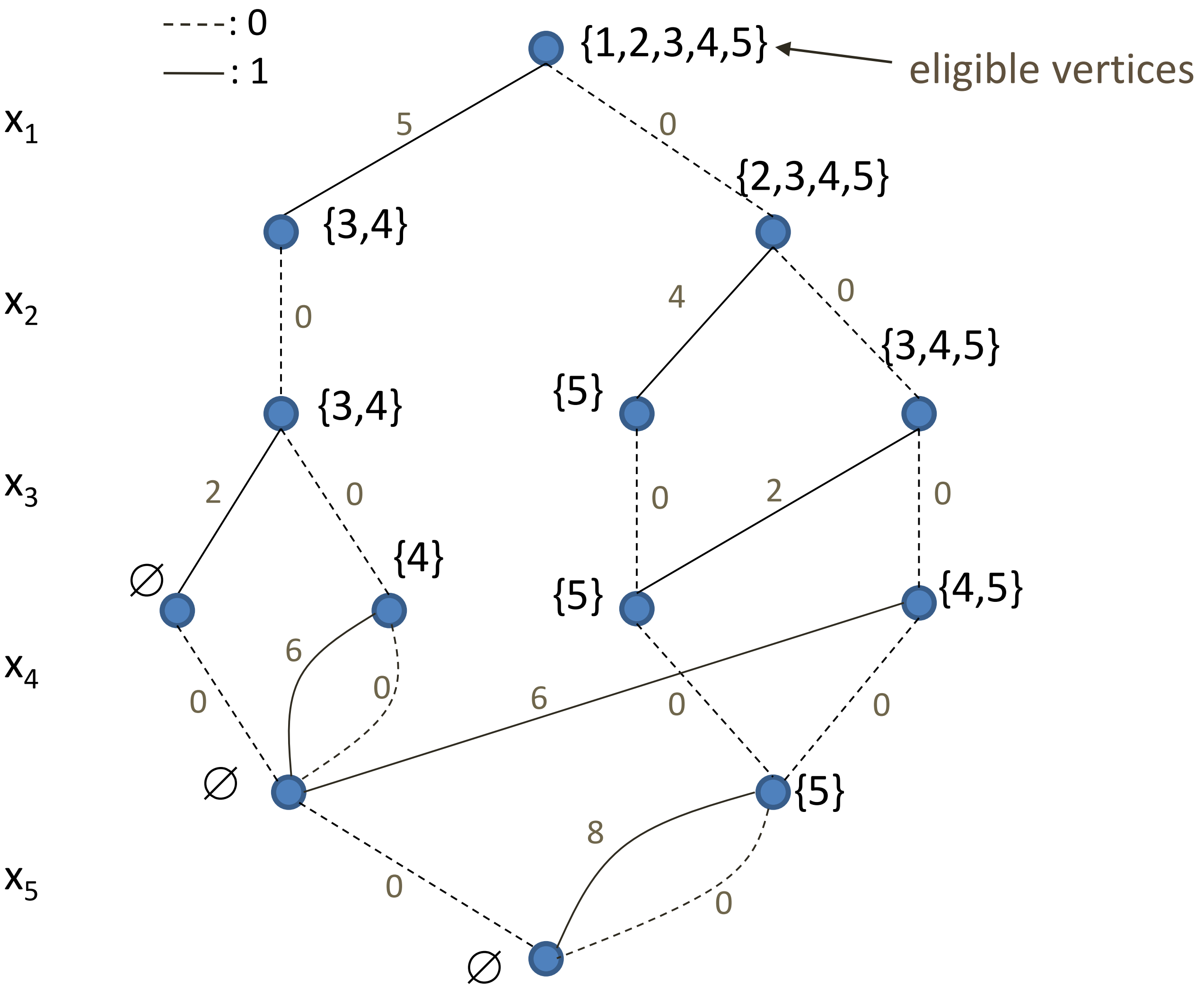
Maximum Independent Set Problem

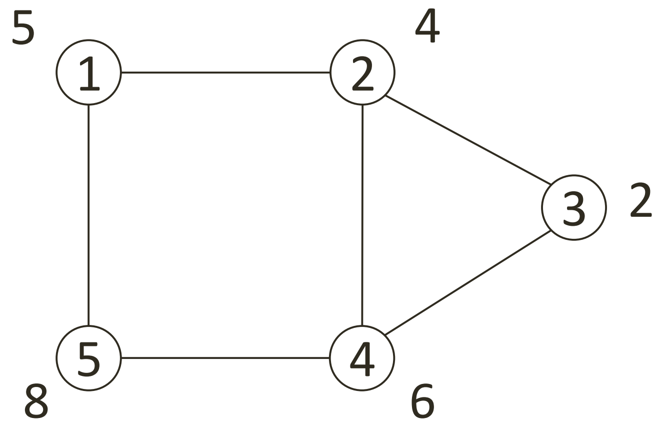


Merge equivalent nodes



Maximum Independent Set Problem





Theorem: This procedure generates a reduced exact BDD

[Bergman, Cire, vH, Hooker, IJOC 2013]

Observations

- In general, decision diagrams grow exponentially large

- Variable ordering impacts size of diagrams
  - Closely connected to treewidth and bandwidth
  - Independent Set: polynomial for certain classes of graphs

    [Bergman, Cire, vH, Hooker, IJOC 2014]
  - TSP: parameterized-size depending on precedence relations

    [Cire & vH, OR 2013]

Towards Generic Discrete Optimization

Relaxation Methods

MIP: Linear Programming Relaxation
DD: Relaxed Decision Diagram

Modeling Framework

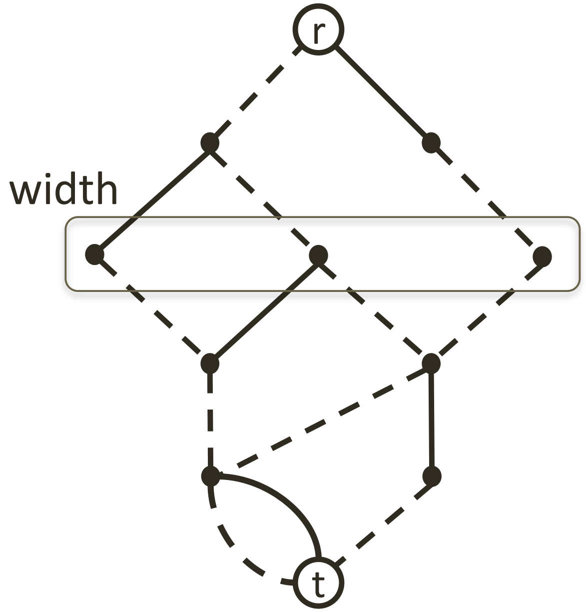
MIP: Linear Inequalities
DD: Dynamic Programming

Generic Optimization Techniques

Relaxed Decision Diagrams

- How to handle exponential size of diagram?
- Explicitly limit the size (e.g., the width)
  - – while ensuring that no solution is lost
  - – over-approximation of the solution space
  - – provides discrete relaxation:

    Relaxed Decision Diagram
  - – strength is controlled by the maximum width



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

Compiling Relaxed Decision Diagrams

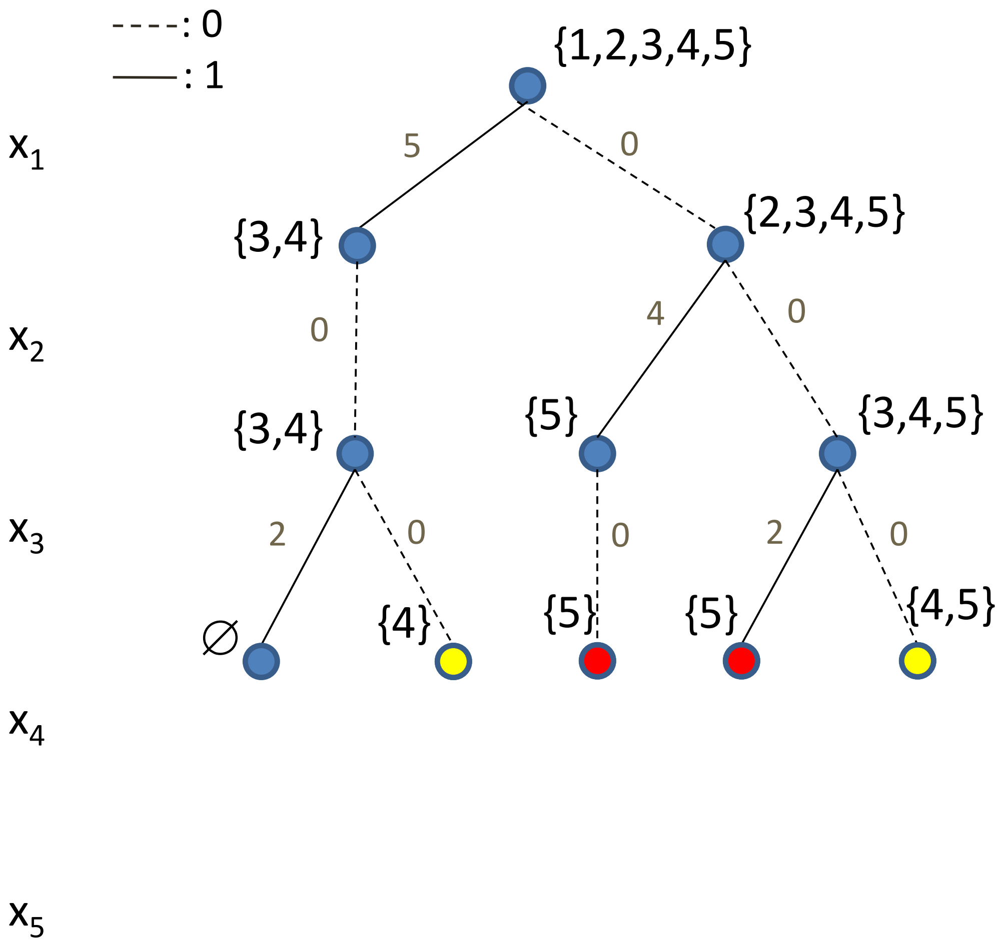
- Model is augmented with a *state aggregation* operator
  - Defines how to merge nodes so that no feasible solution is lost
  - Example for maximum independent set:

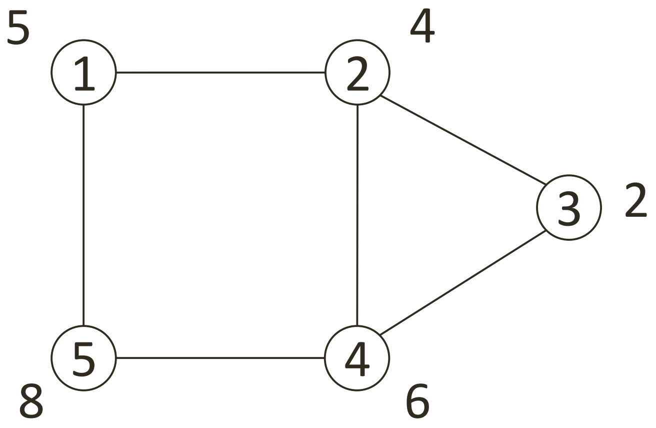
$$V_i(S) = \begin{cases} max\,\{V_{i-1}(S \setminus \{i\}), V_{i-1}\big(S \setminus N(i)\big) + wi\}, & i \in S \\ V_{i-1}(S), & o.w. \end{cases}$$

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

$$\oplus\,(S_1, S_2) = S_1 \cup S_2$$

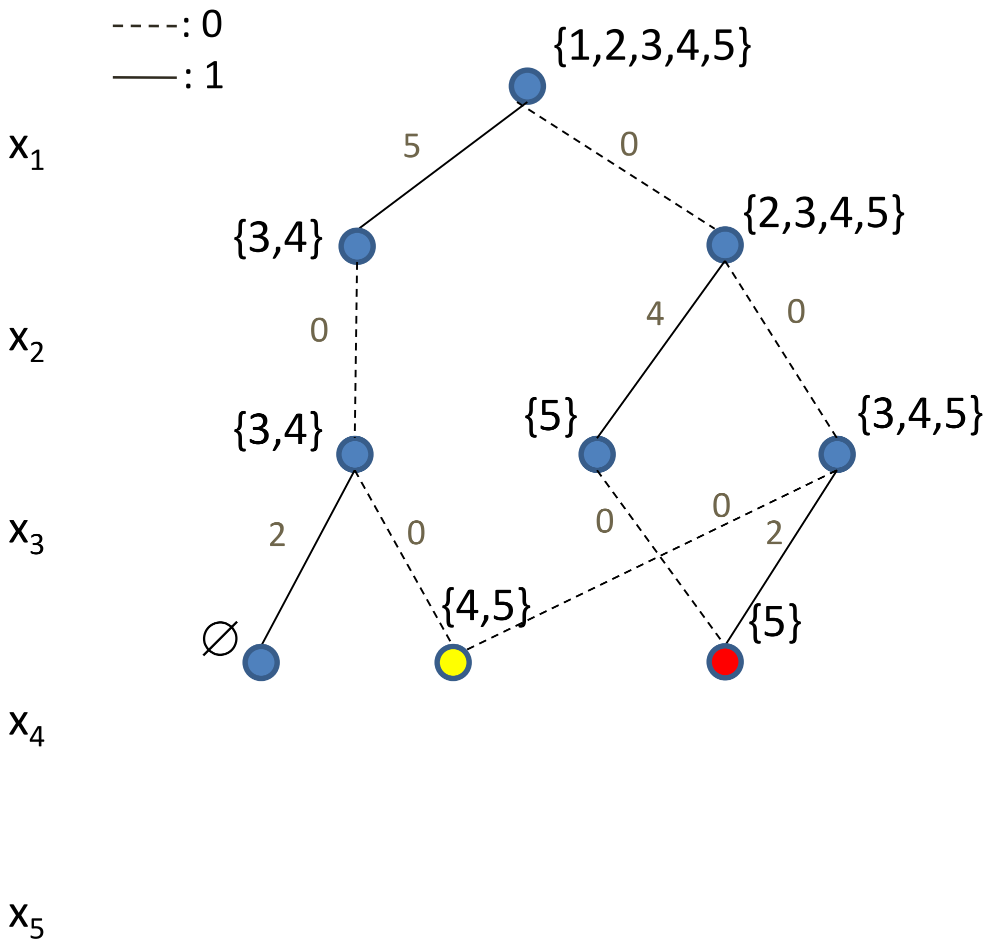
Independent Set Problem: Relaxed DD

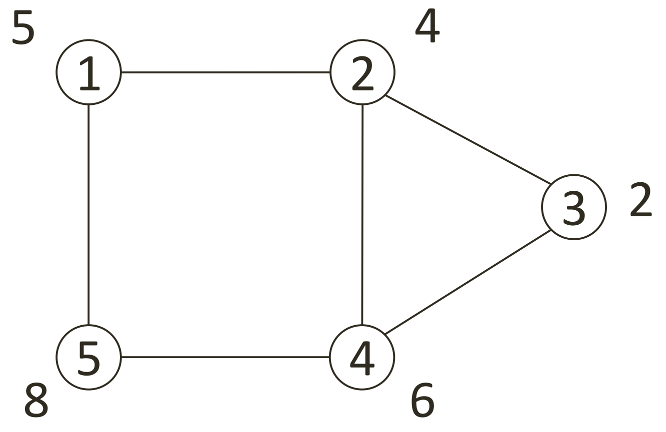




Maximum width = 3

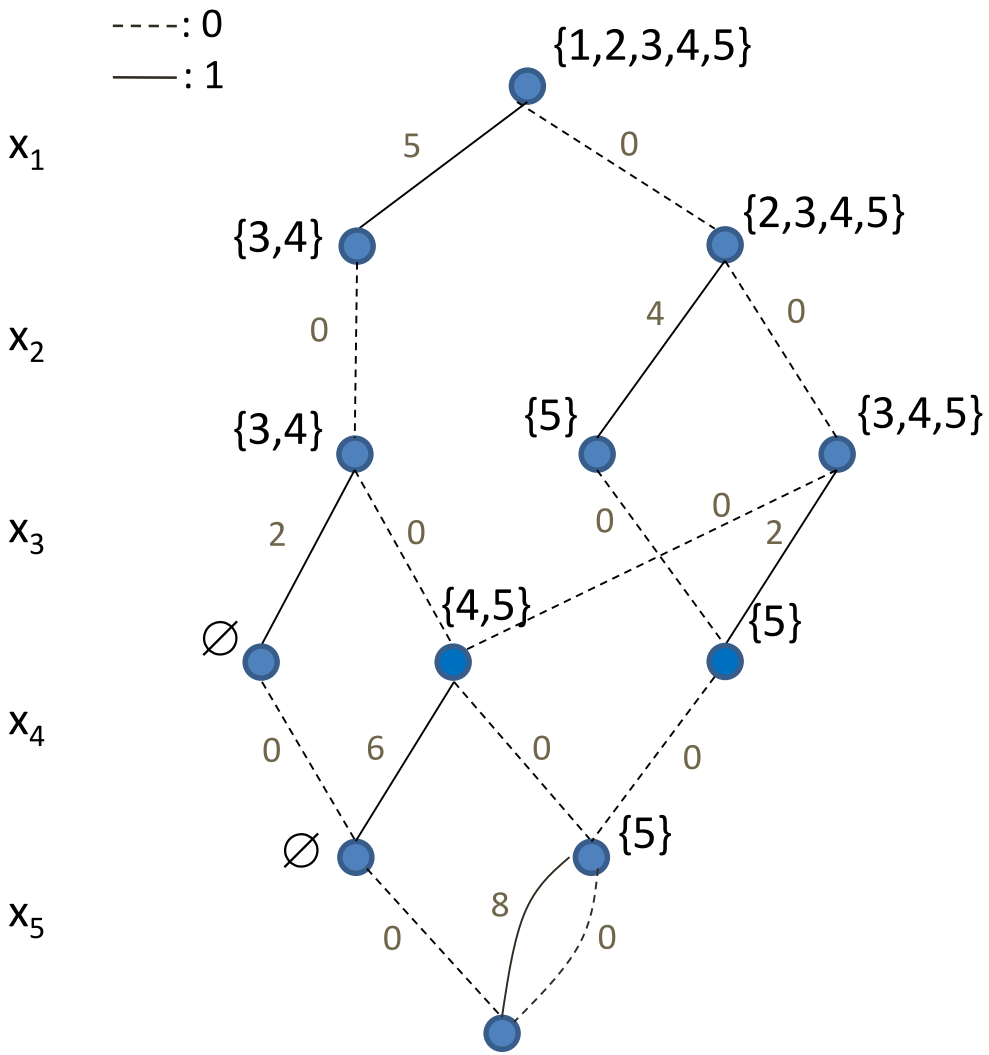
Independent Set Problem: Relaxed DD

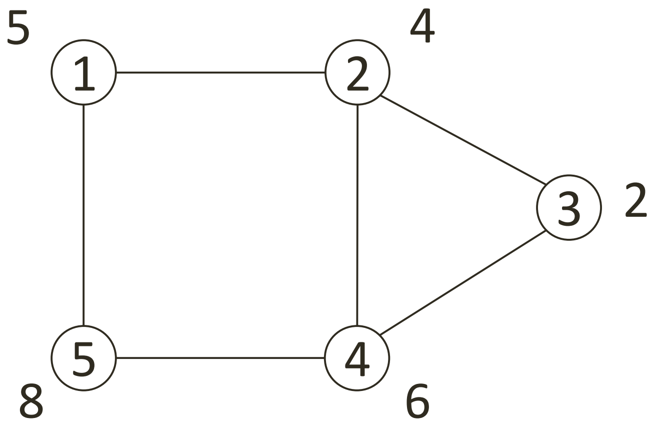




Maximum width = 3

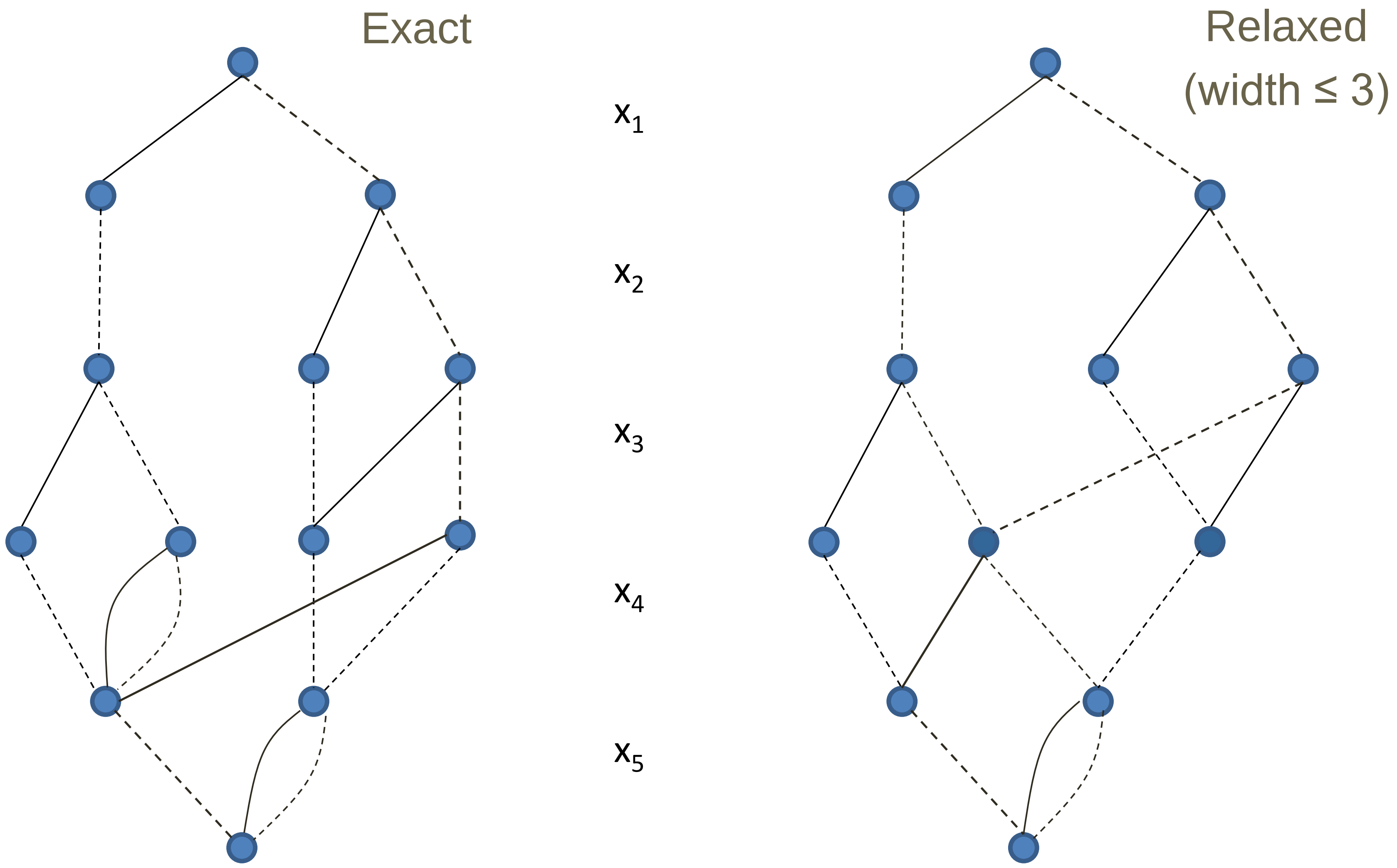
Independent Set Problem: Relaxed DD

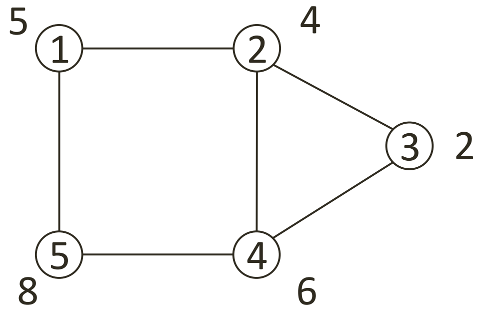




Maximum width = 3

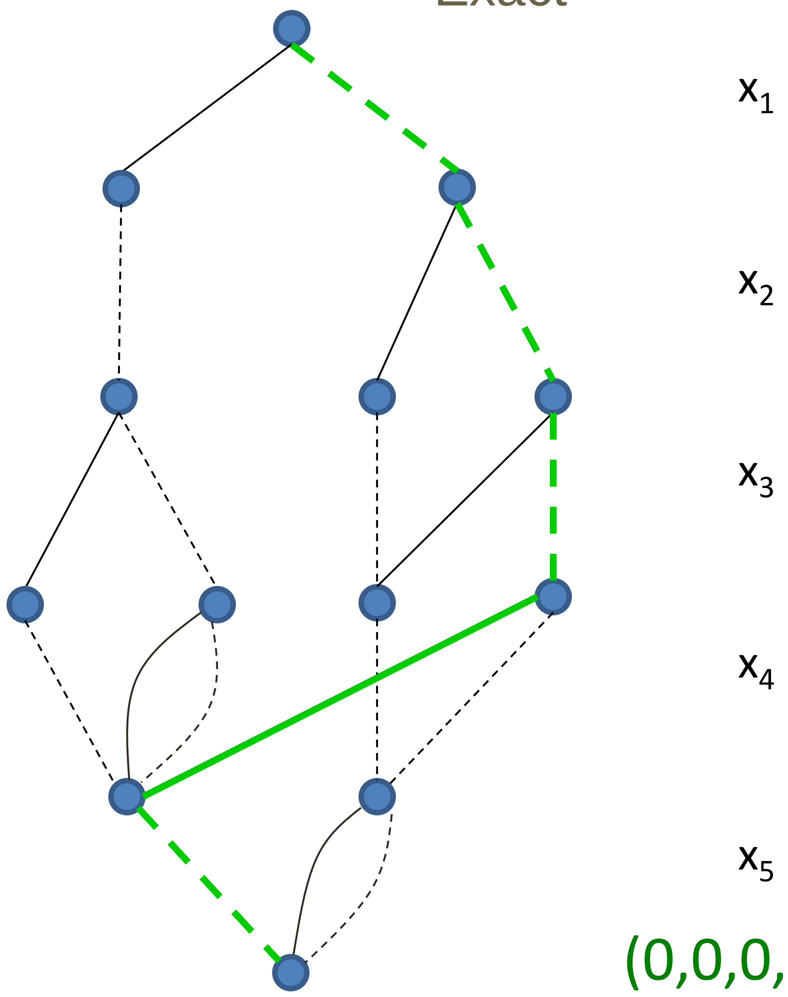
Exact vs. Relaxed Decision Diagrams





Exact vs. Relaxed Decision Diagrams

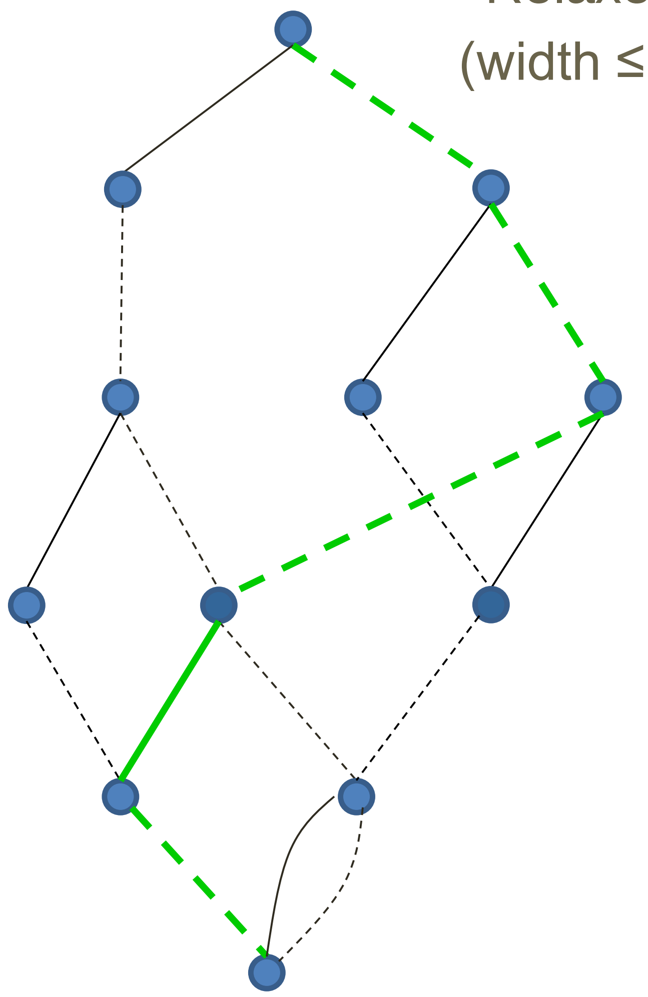
Exact

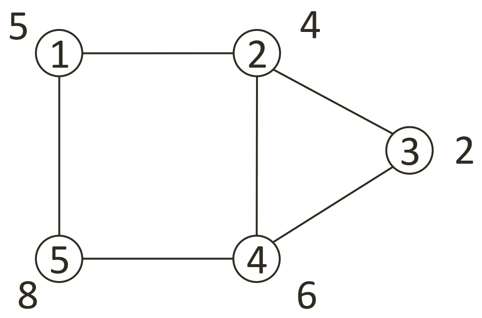


(0,0,0,1,0)

Relaxed

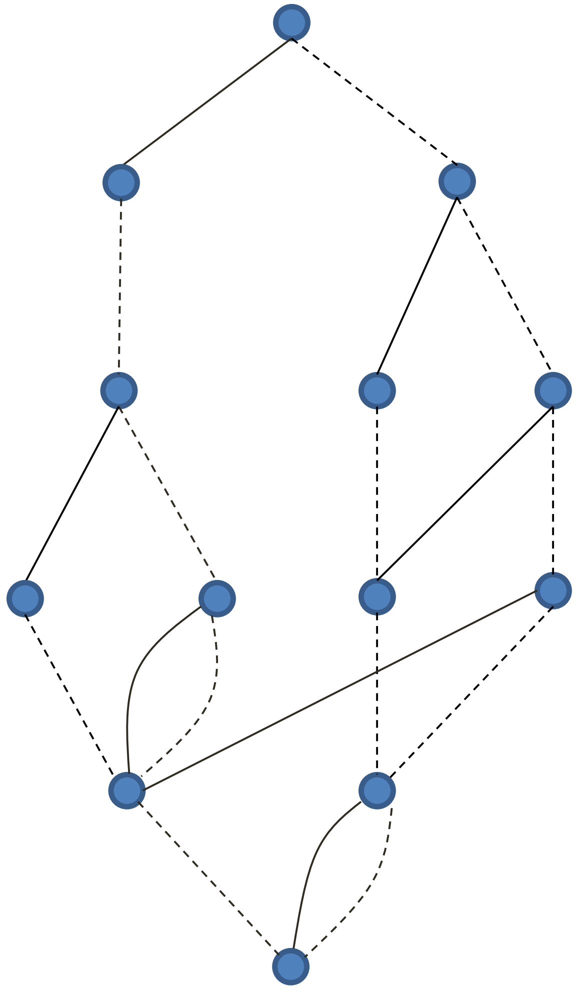
(width ≤ 3)





Exact vs. Relaxed Decision Diagrams

Exact



x_1

x_2

x_3

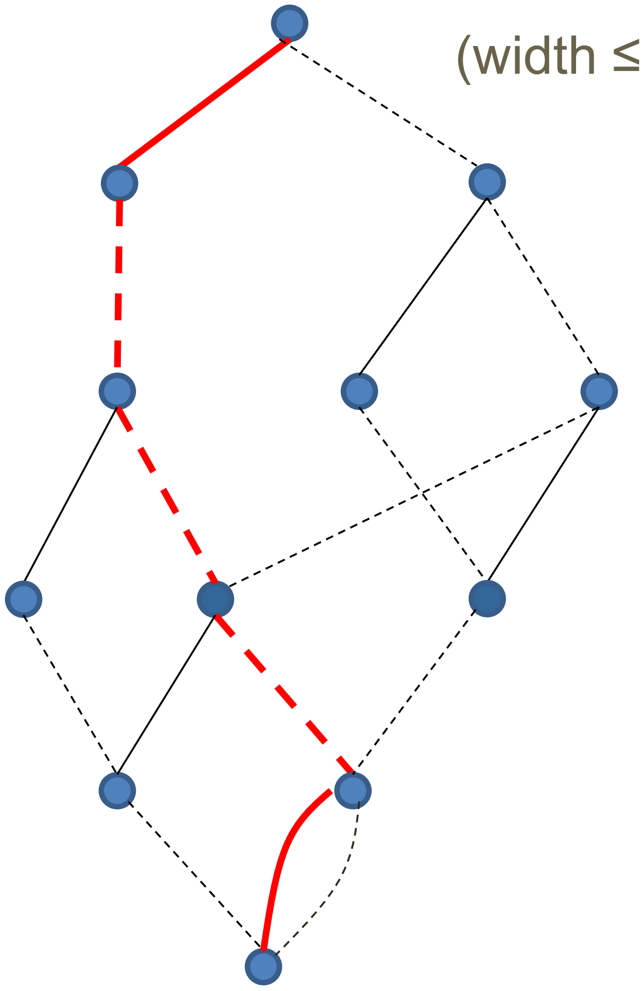
x_4

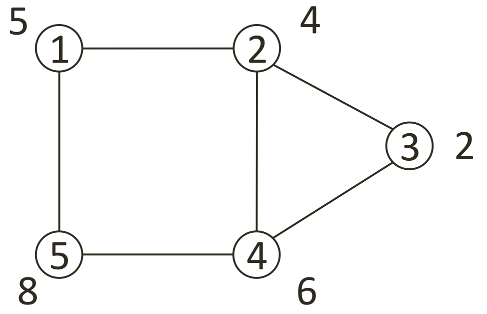
x_5

(1,0,0,0,1)

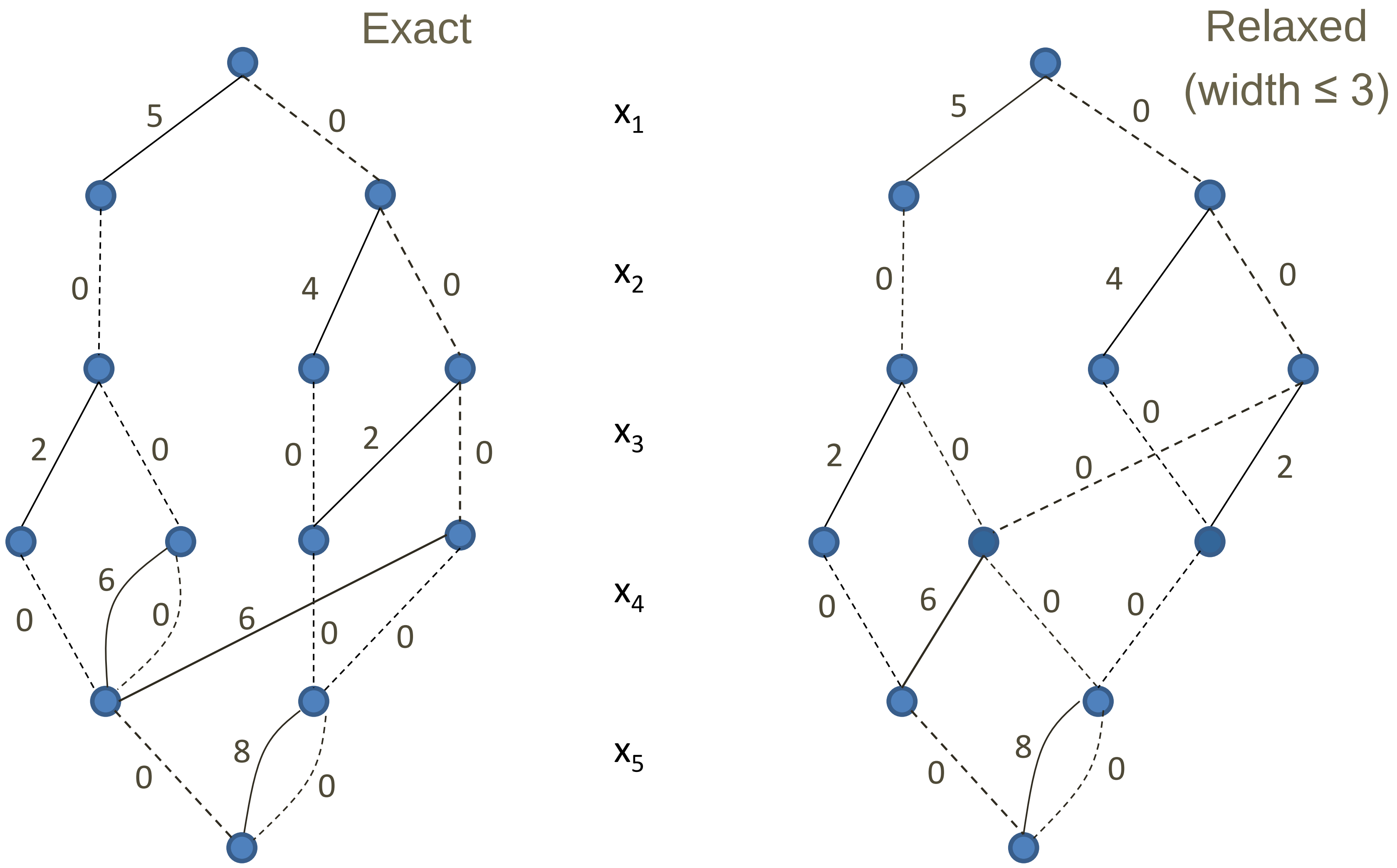
Relaxed

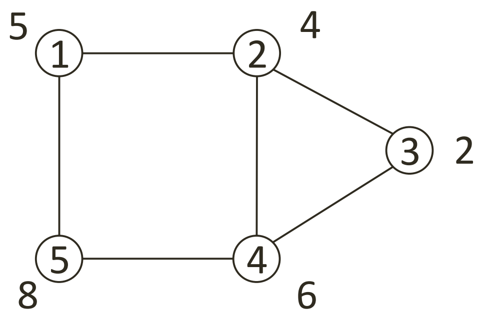
(width ≤ 3)



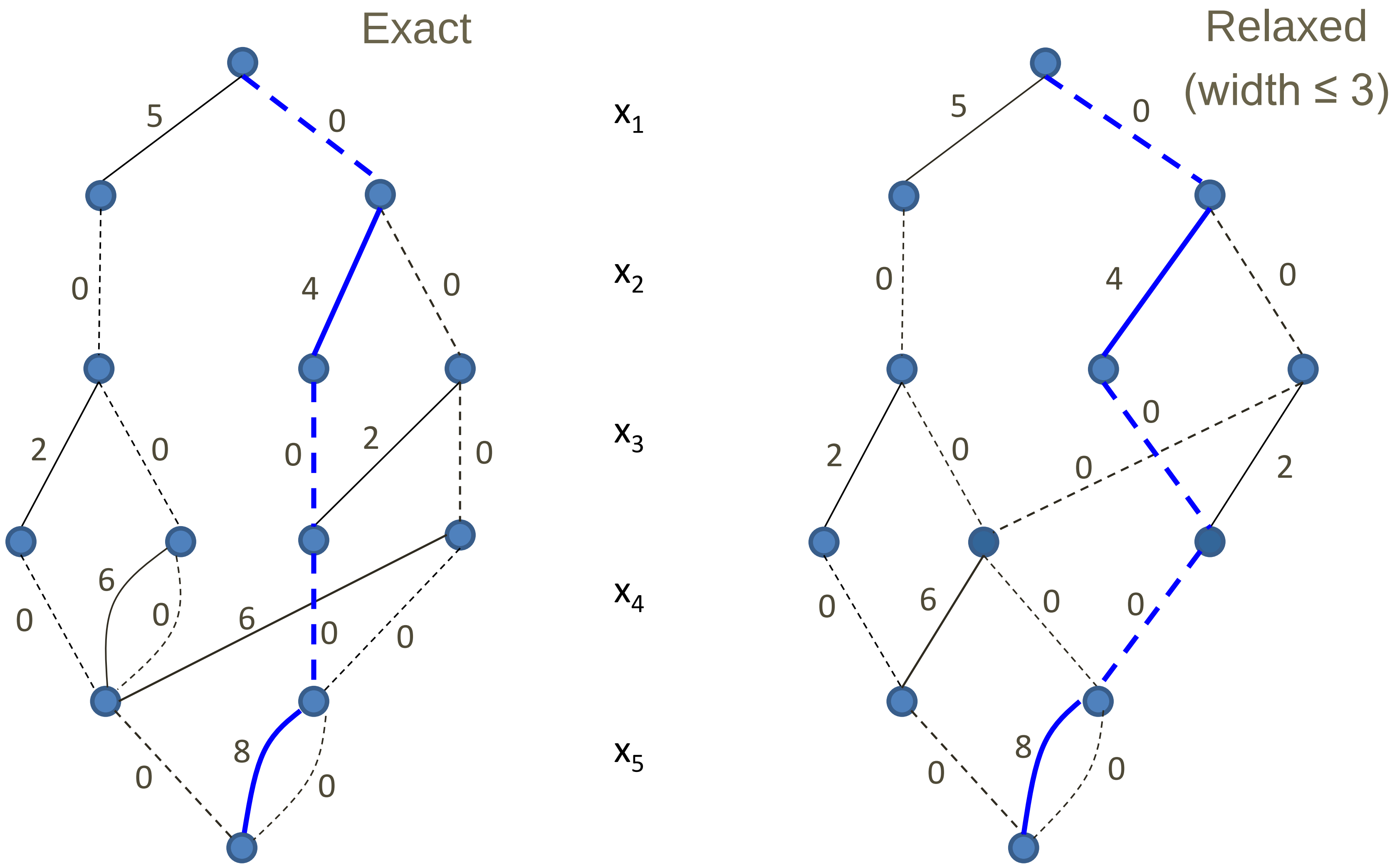


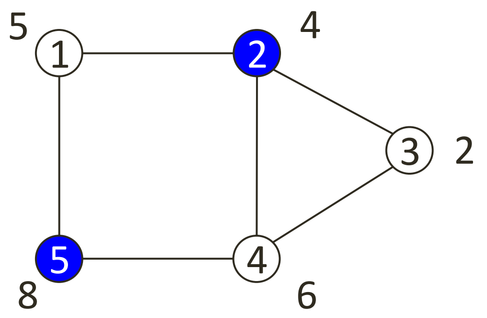
Exact vs. Relaxed Decision Diagrams





Exact vs. Relaxed Decision Diagrams

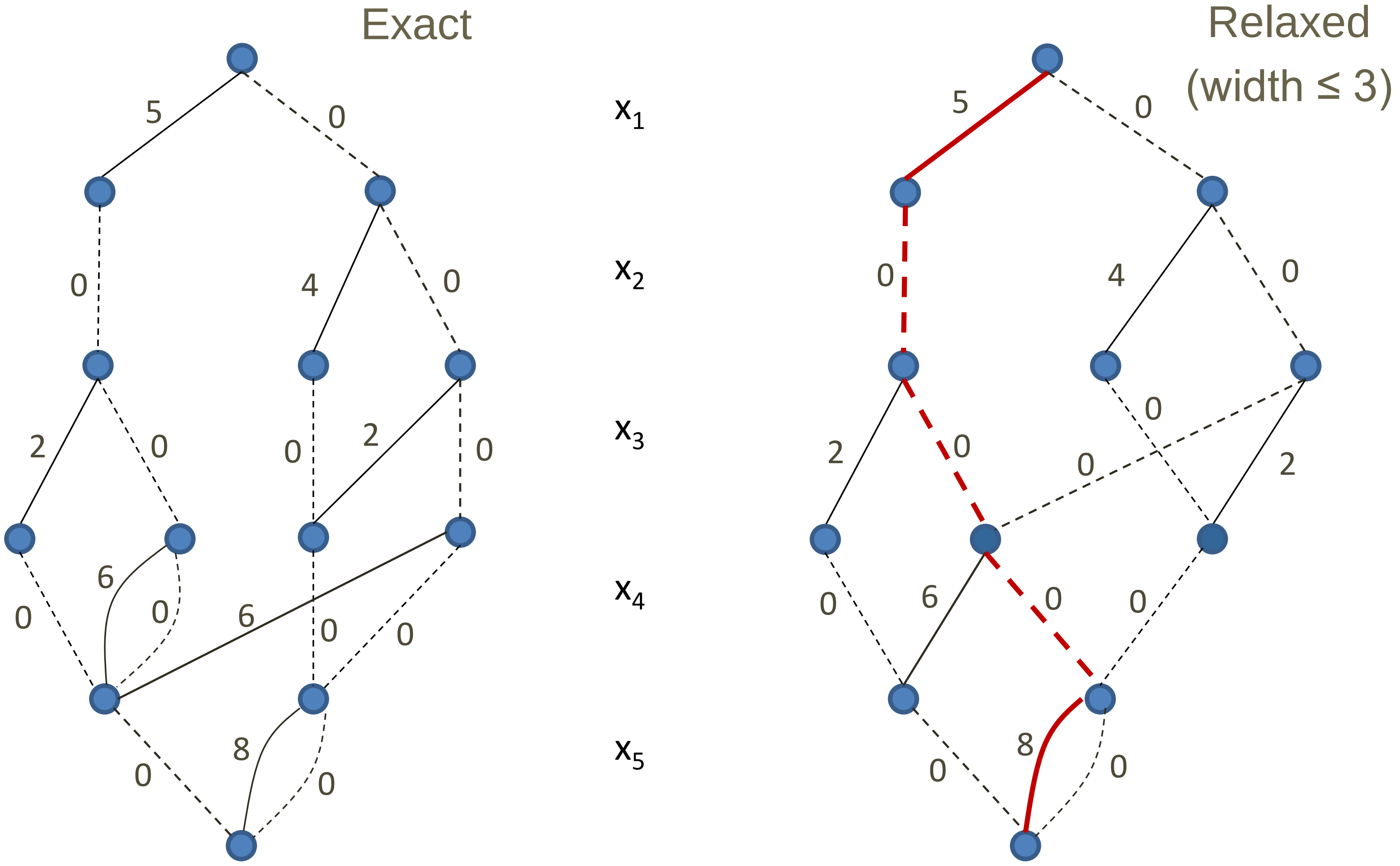


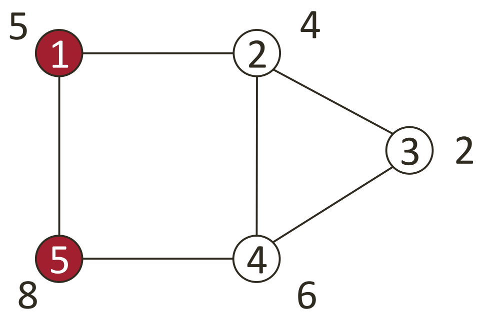


$x = (0, 1, 0, 0, 1)$

Solution value = 12

Exact vs. Relaxed Decision Diagrams

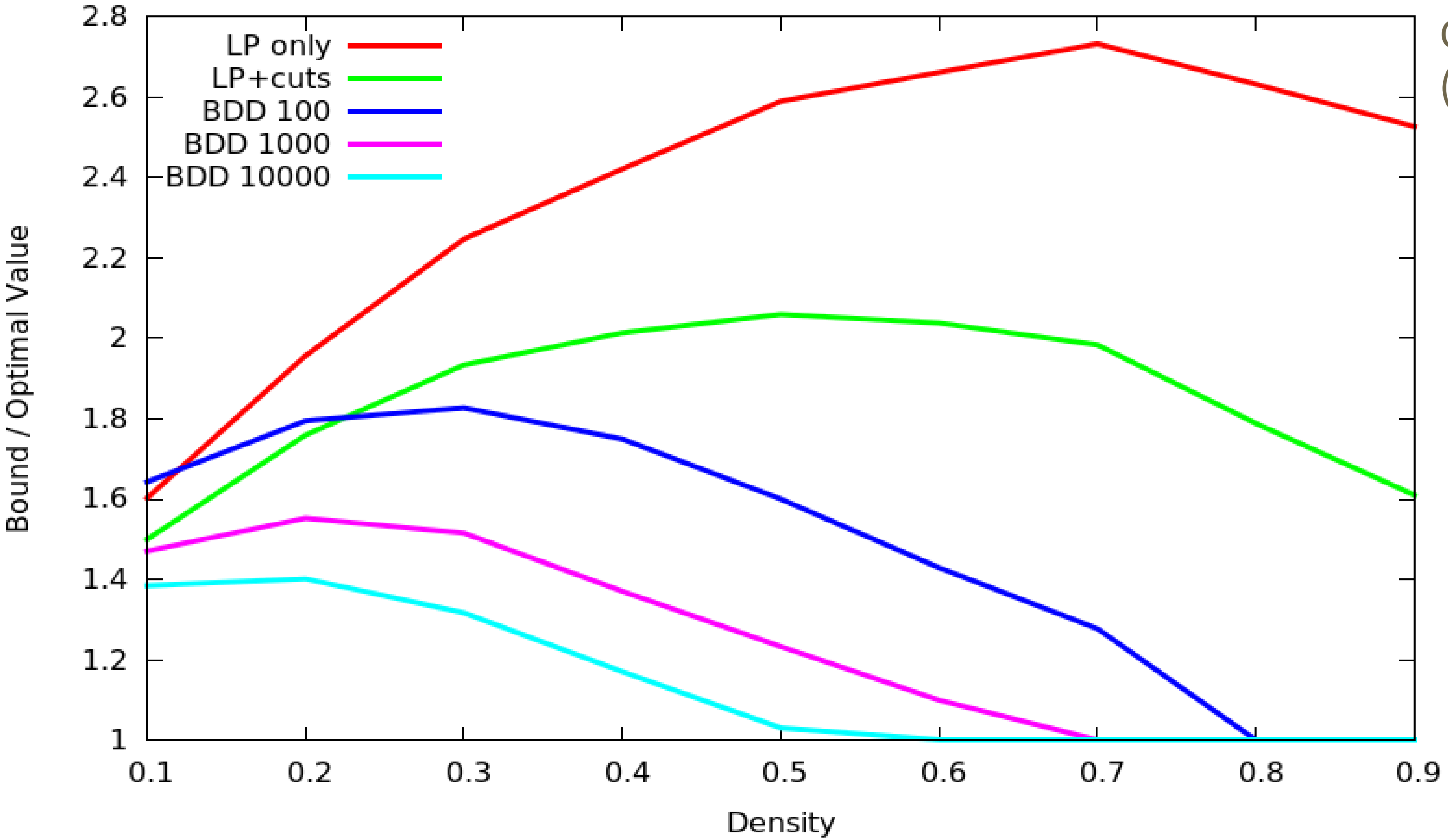




$x = (1, 0, 0, 0, 1)$

Upper bound = 13

Relaxation Bound: Independent Set



Compare with LP bound (CPLEX)

Towards Generic Discrete Optimization

Relaxation Methods

MIP: Linear Programming Relaxation
DD: Relaxed Decision Diagram

Modeling Framework

MIP: Linear Inequalities
DD: Dynamic Programming

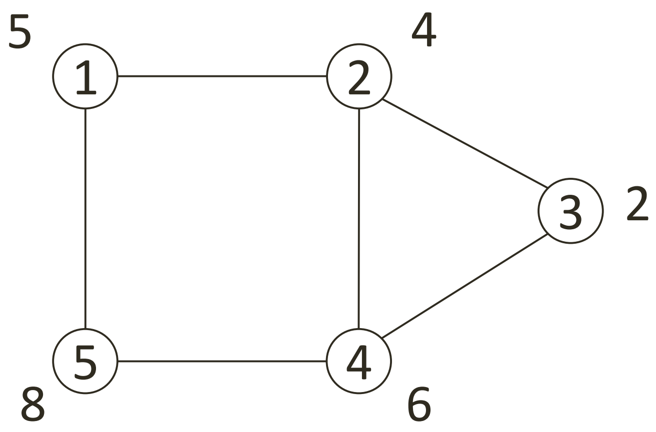
Primal Heuristics

MIP: Feasibility Pump, RINS, …
DD: Restricted Decision Diagrams

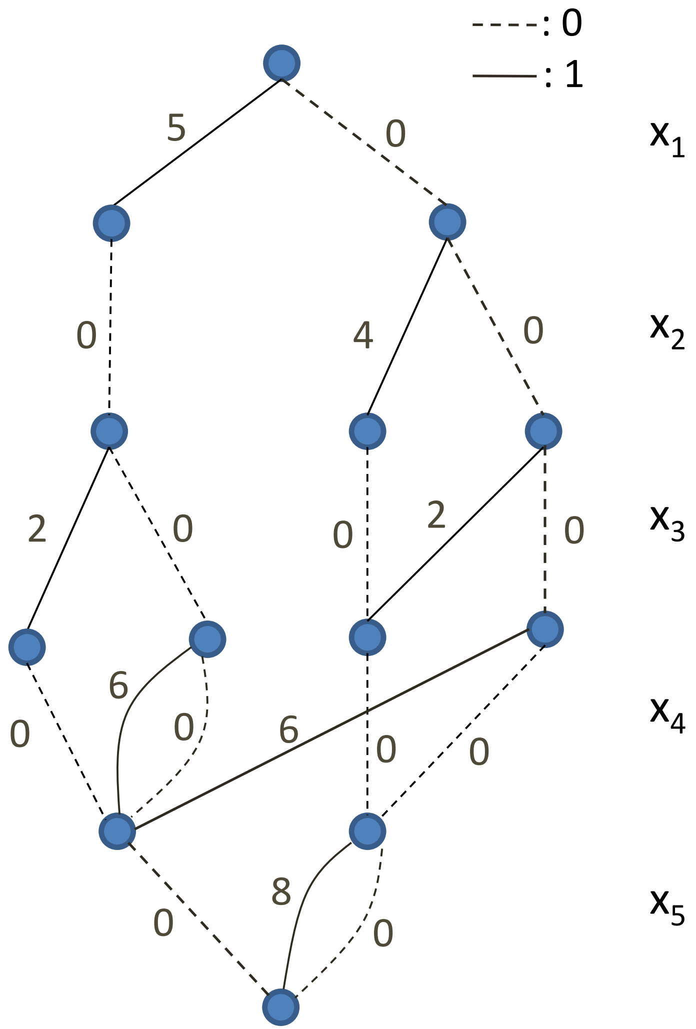
Generic Optimization Techniques

Restricted Decision Diagrams

- Under-approximation of the feasible set



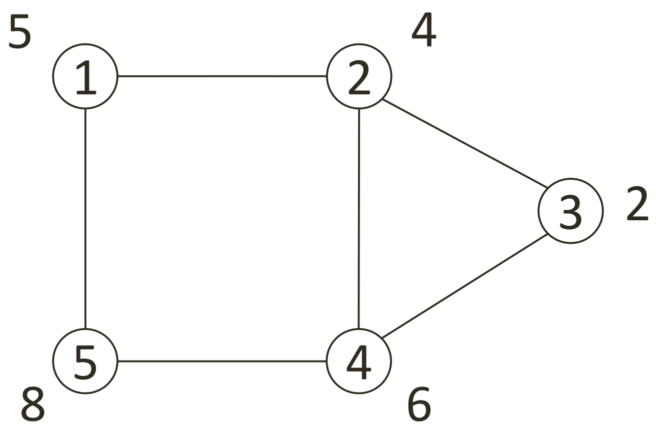
Maximum width = 3



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

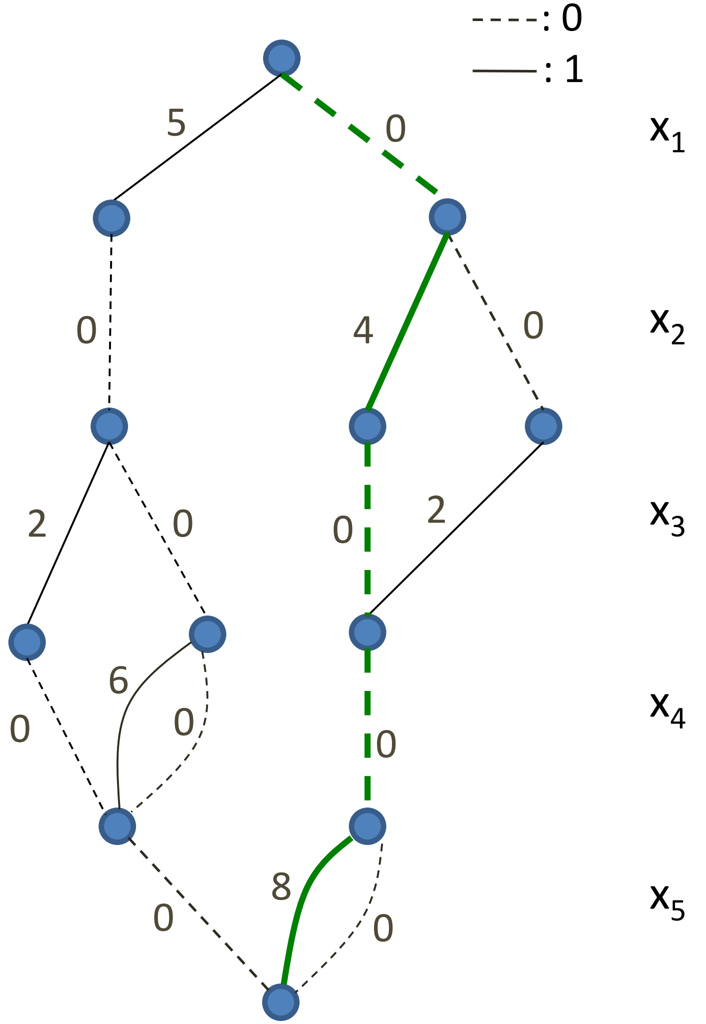
Restricted Decision Diagrams

- Under-approximation of the feasible set



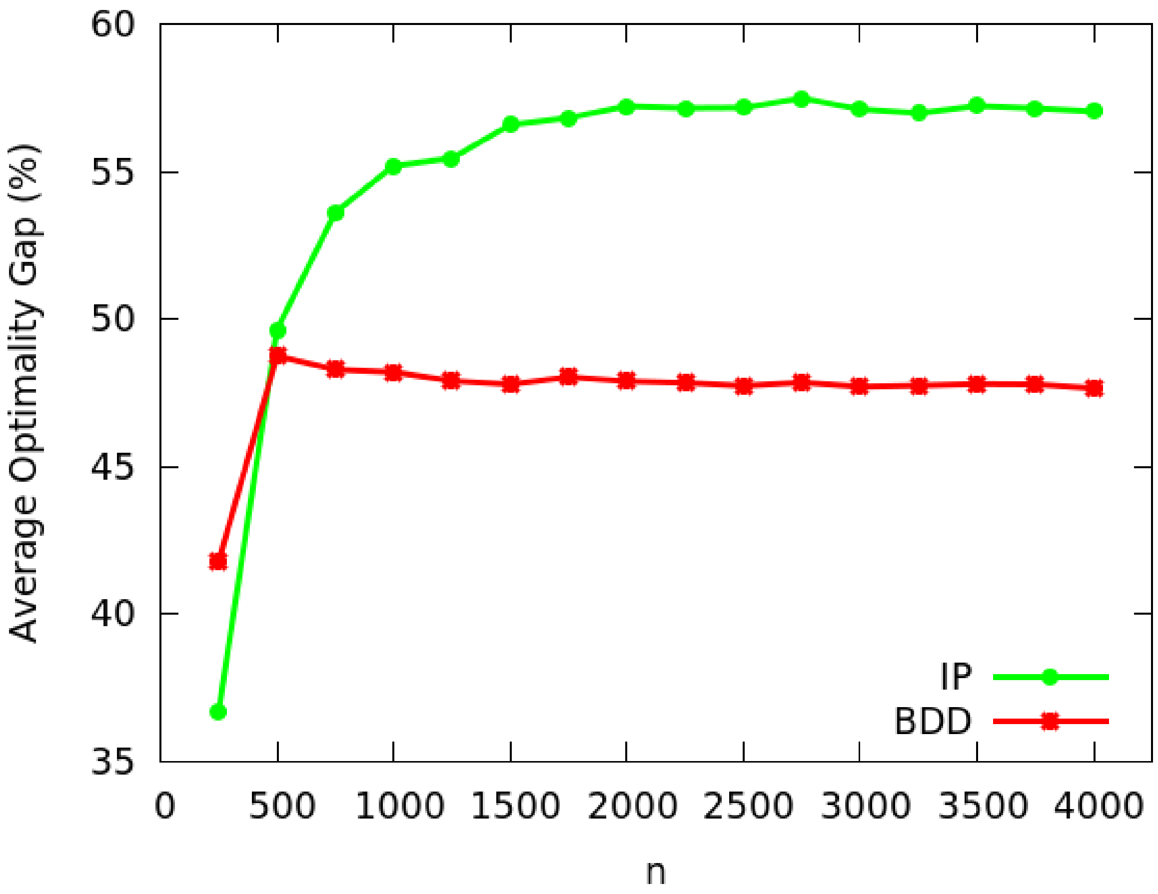
Maximum width = 3

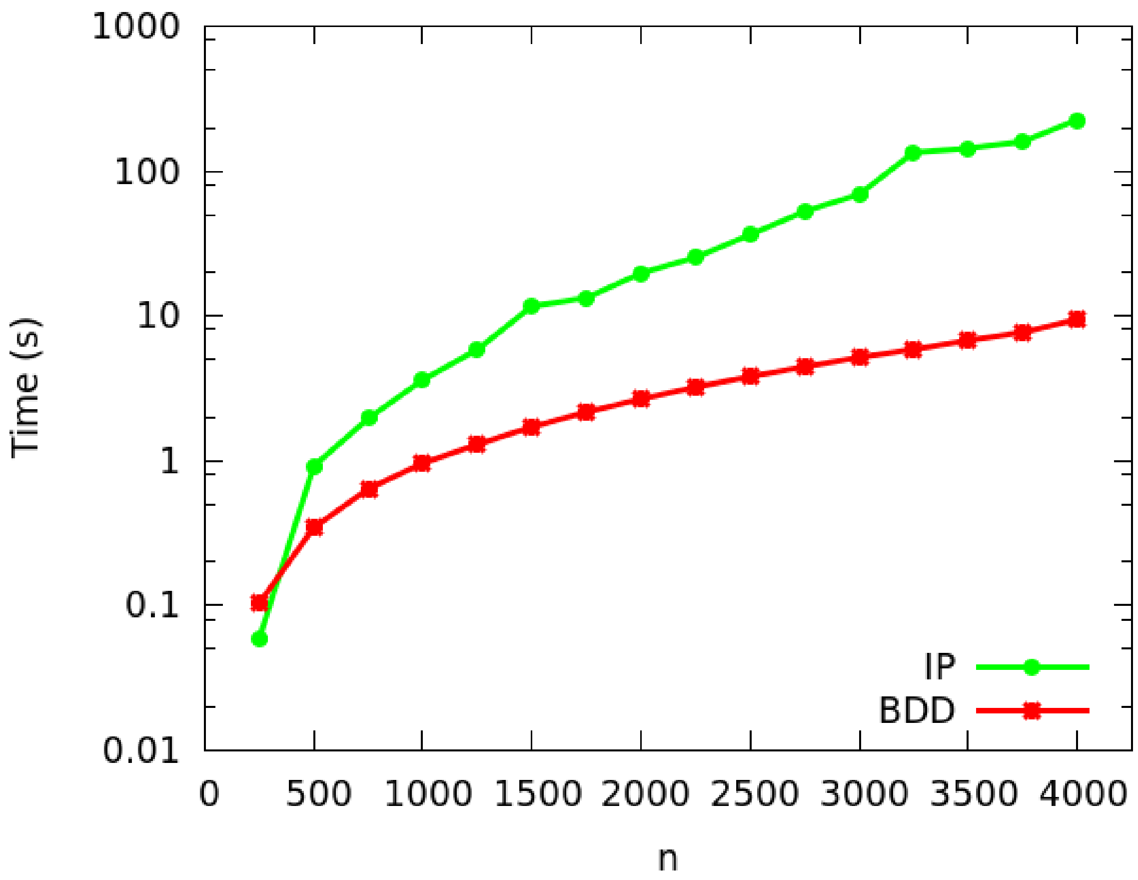
x = (0, 1, 0, 0, 1)
Lower bound = 12



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

Primal Bound: Set Covering Problem





Towards Generic Discrete Optimization

Relaxation Methods

MIP: Linear Programming Relaxation
DD: Relaxed Decision Diagram

Modeling Framework

MIP: Linear Inequalities
DD: Dynamic Programming

Primal Heuristics

MIP: Feasibility Pump, RINS, …
DD: Restricted Decision Diagrams

Generic Optimization Techniques

Inference

MIP: Valid linear cuts
DD: Propagation, cuts

Inference Techniques from DDs

- Cut generation for general MIPs
  - Idea first proposed in [Becker et al., 2005] [Behle, PhD 2007]
  - Facet-defining cuts [Tjandraatmadja & vH, IJOC to appear]
  - Extension to MINLP [Davarnia & vH]
- Clause learning for SAT [Kell et al., CPAIOR 2015]
- Problem-specific cuts
  - Precedence constraints for scheduling problem [Cire&vH, OR 2013]
- Constraint Propagation in Constraint Programming
  - Several constraint types: Alldiff, Among, Sequence, Markov, Statistical, …

[Hoda, vH, Hooker, CP 2010] [Bergman, Cite, vH, JAIR 2014]
[Perez & Regin, IJCAI2015, CP2016, AAAI2017, CPAIOR 2017]

Towards Generic Discrete Optimization

Relaxation Methods

MIP: Linear Programming Relaxation
DD: Relaxed Decision Diagram

Modeling Framework

MIP: Linear Inequalities
DD: Dynamic Programming

Primal Heuristics

MIP: Feasibility Pump, RINS, …
DD: Restricted Decision Diagrams

Generic Optimization Techniques

Inference

MIP: Valid linear cuts
DD: Propagation, cuts

Search

MIP: Branch and bound (variable-based)
DD: Branch and bound (state-based)

Exact Search Method

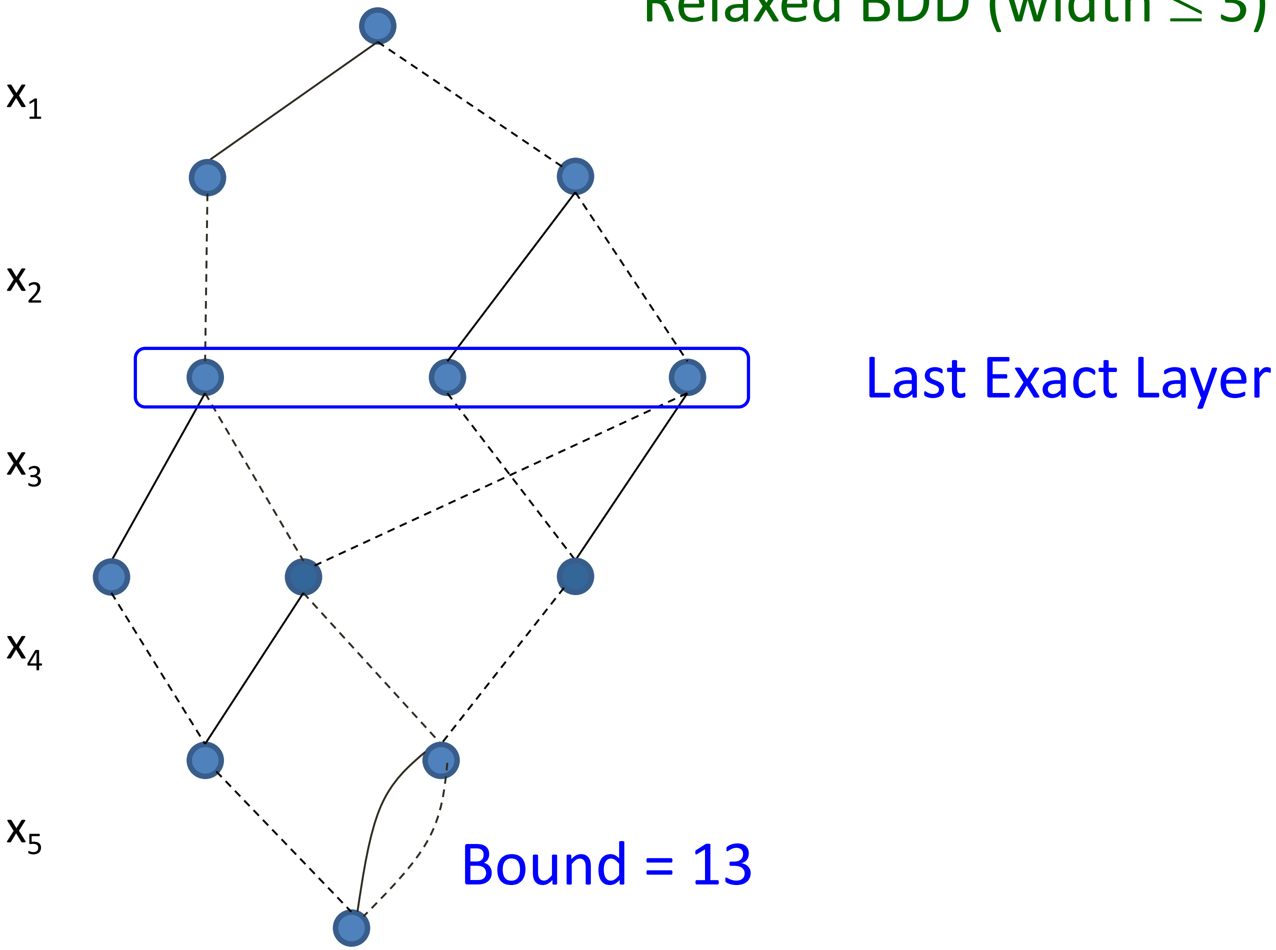
- Novel decision diagram branch-and-bound scheme
  - Relaxed diagrams play the role of the LP relaxation
  - Restricted diagrams are used as primal heuristics

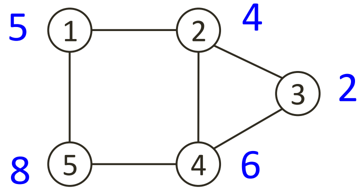
- Branching is done on the *nodes* of the diagram
  - Branching on pools of partial solutions
  - Eliminate search symmetry

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

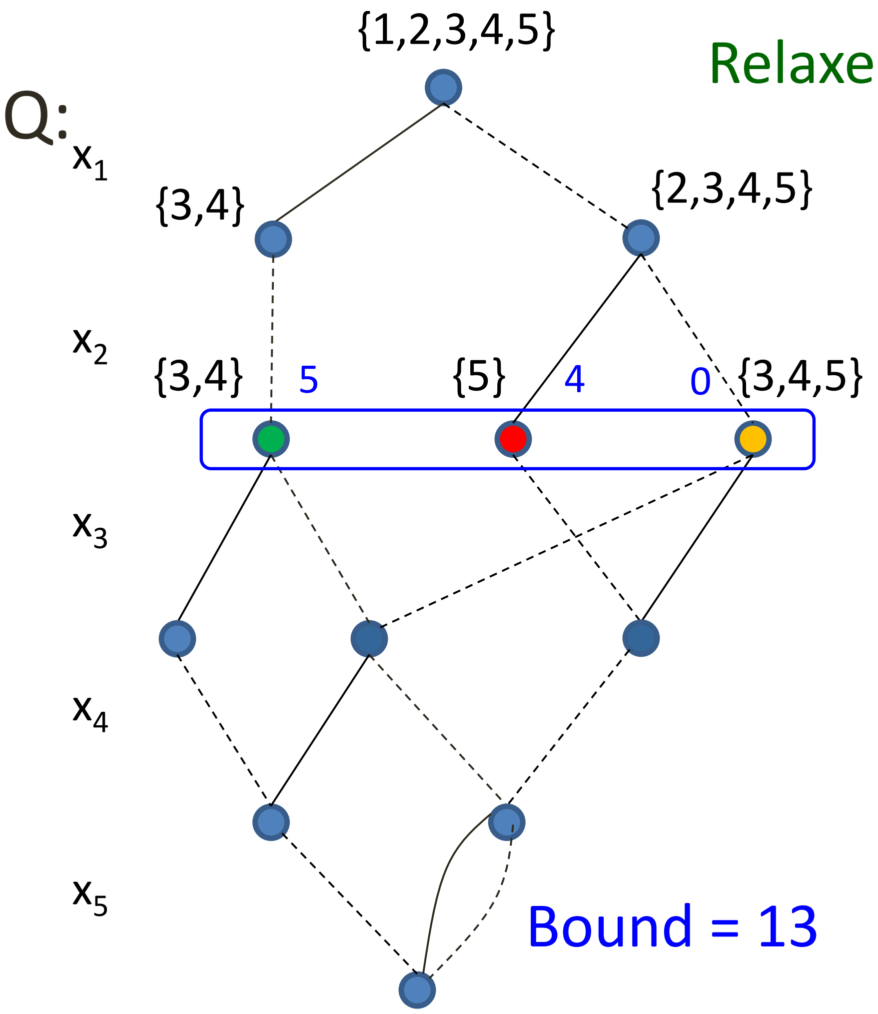
Branch and Bound

Relaxed BDD (width $\leq$ 3)

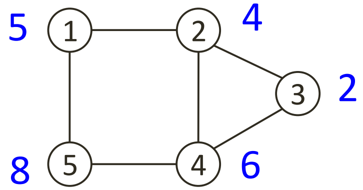




Node Queue



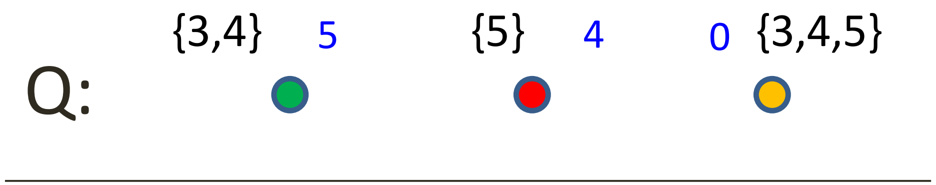
Relaxed BDD (width ≤ 3)

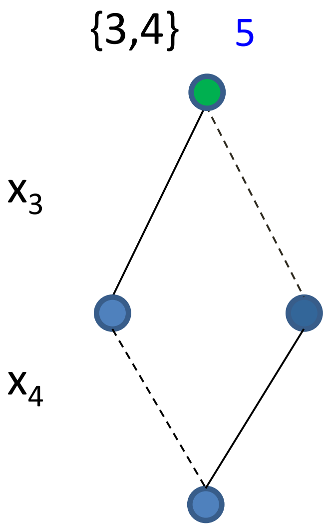


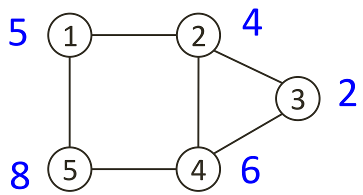
Upper bound = 13

Last Exact Layer

Node Queue





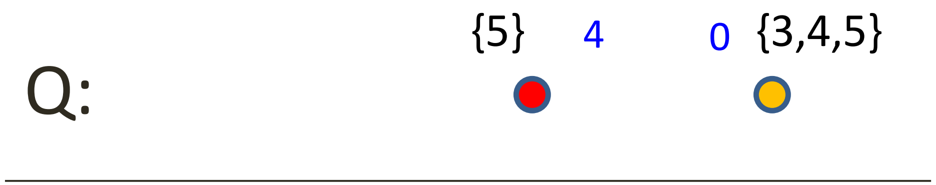


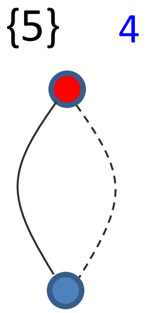
Upper bound = 13

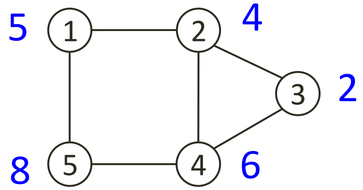
Lower bound = 11

Exact solution: 11

Node Queue







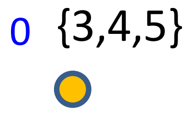
Upper bound = 13

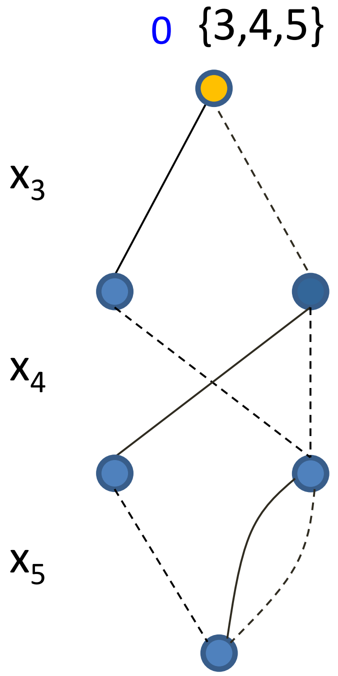
Lower bound = 12

Exact solution: 12

Node Queue

Q:

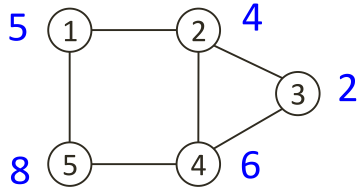




Exact solution: 10

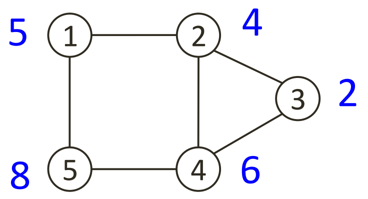
Upper bound = 13

Lower bound = 12



Node Queue

Q:

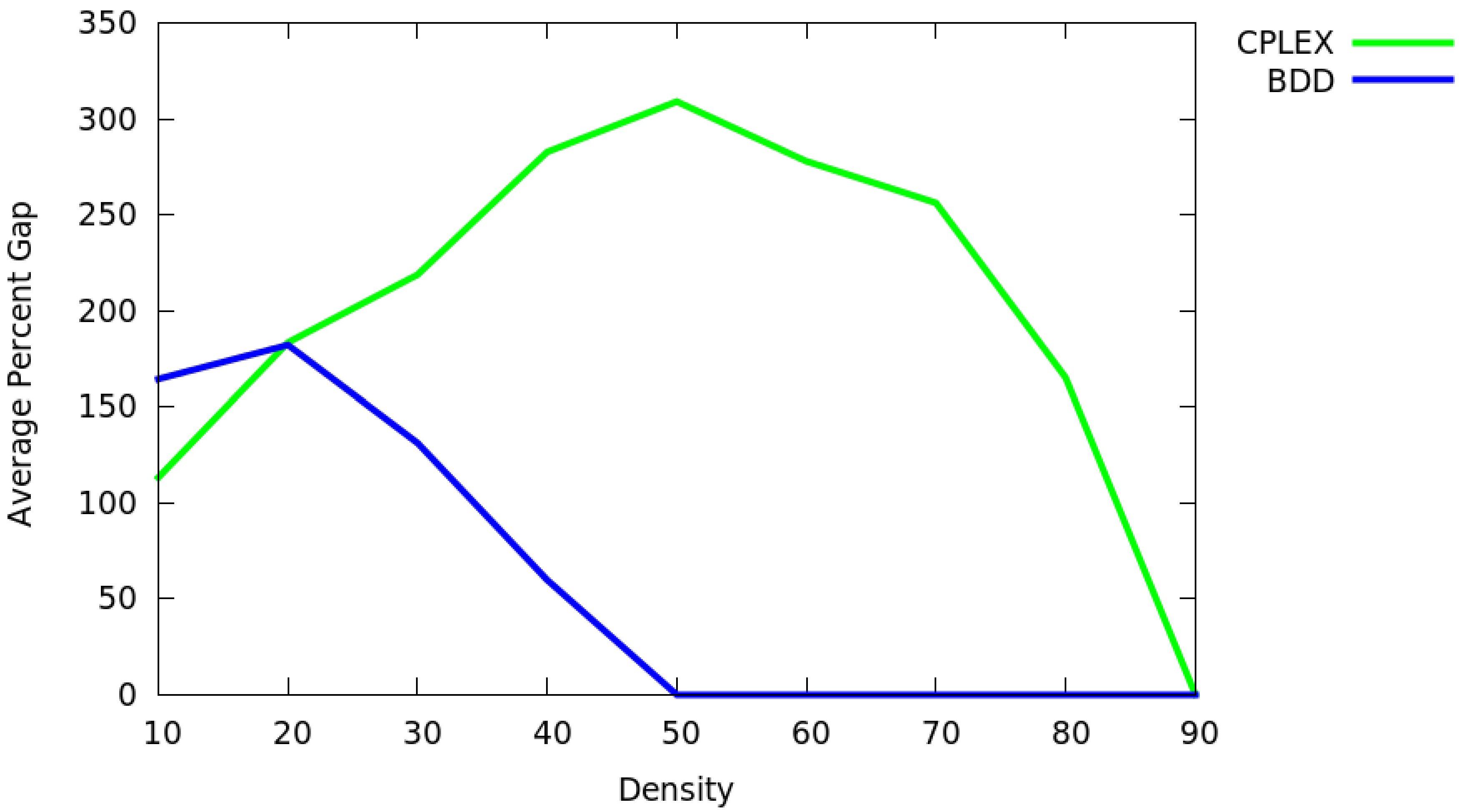


Optimal solution: 12

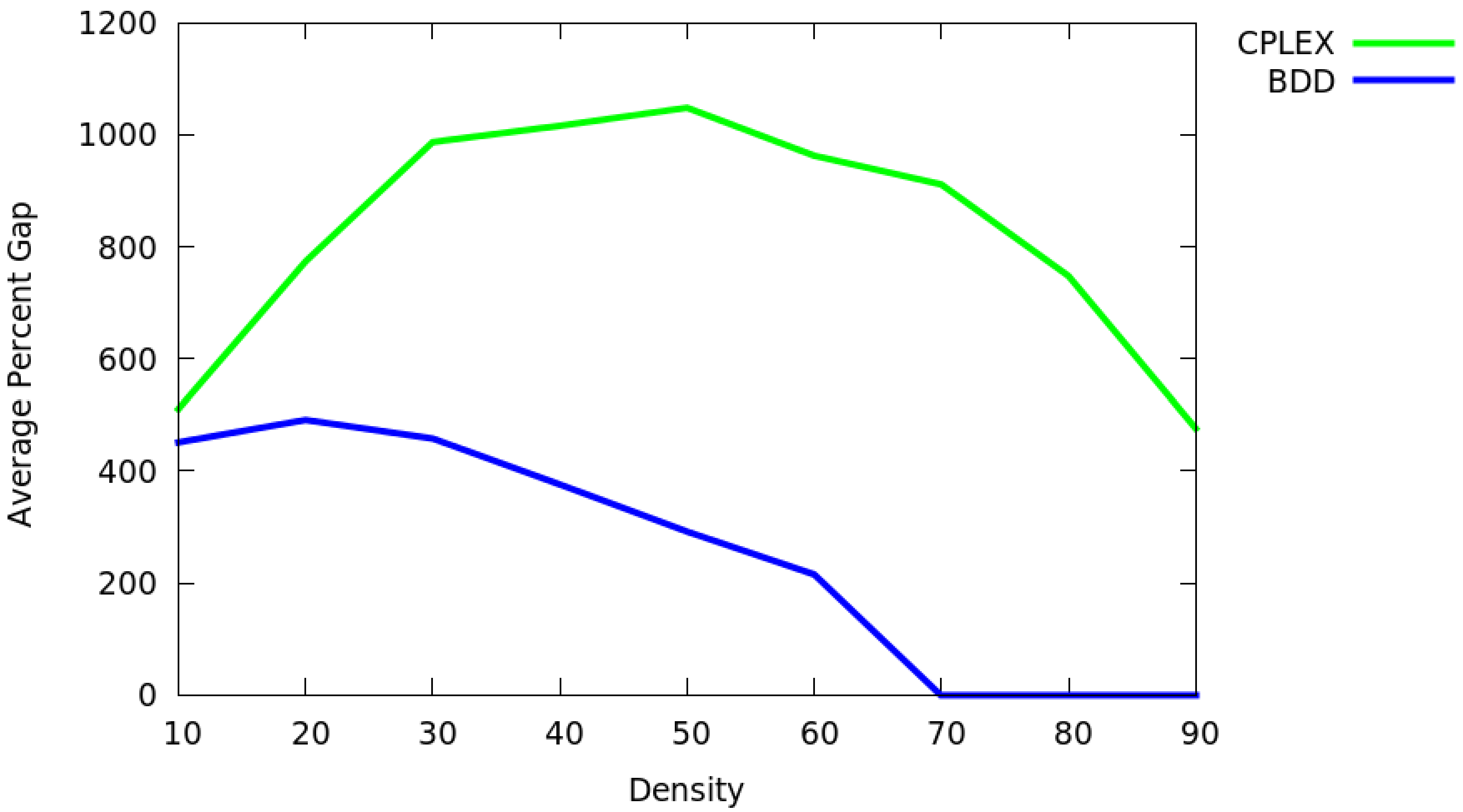
New Branching Scheme

- 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
    - Define CP search with MDD inside global constraint
  - Immediate **parallelization**
    - Send nodes in the queue to different workers, recursive application
    - DDX10 [Bergman et al. CPAIOR 2014]

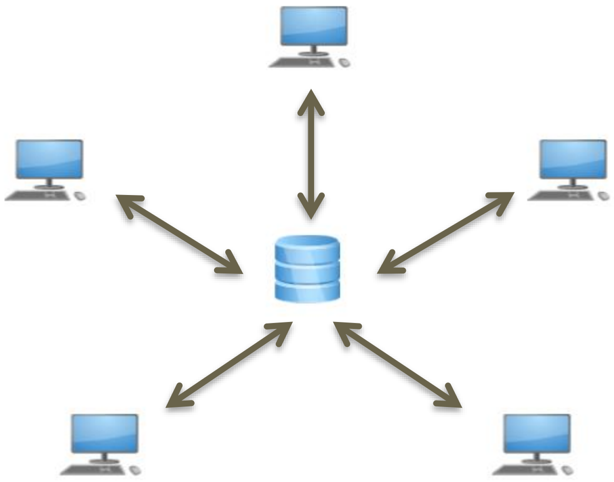
Maximum Independent Set: 500 variables



Maximum Independent Set: 1500 variables



Parallelization: Centralized Architecture



Master maintains a **pool** of BDD nodes to process

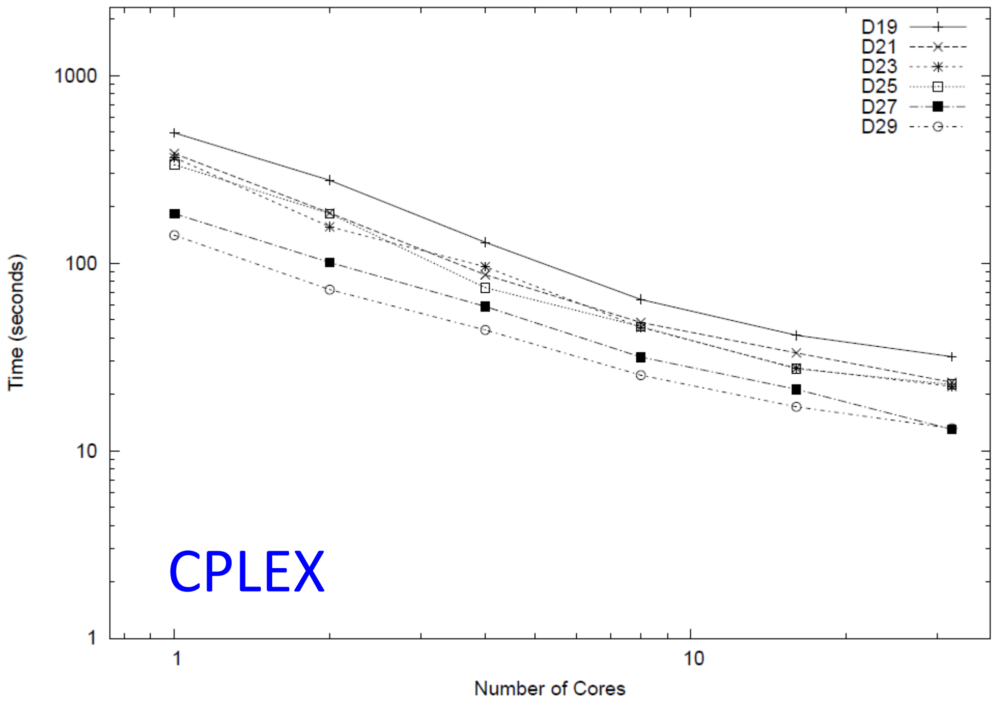
- nodes with larger upper bound have higher priority

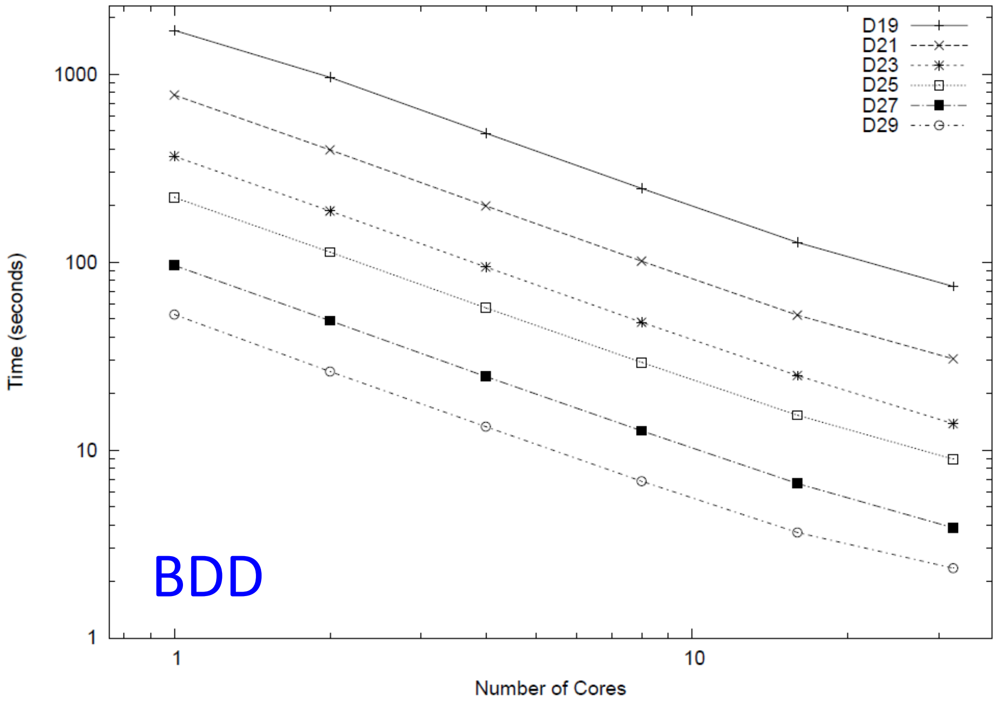
Workers receive BDD nodes, generate *restricted* & *relaxed* BDDs, and send new BDD nodes and bounds to master

- they also maintain a **local pool** of nodes

[Bergman et al. CPAIOR 2014]

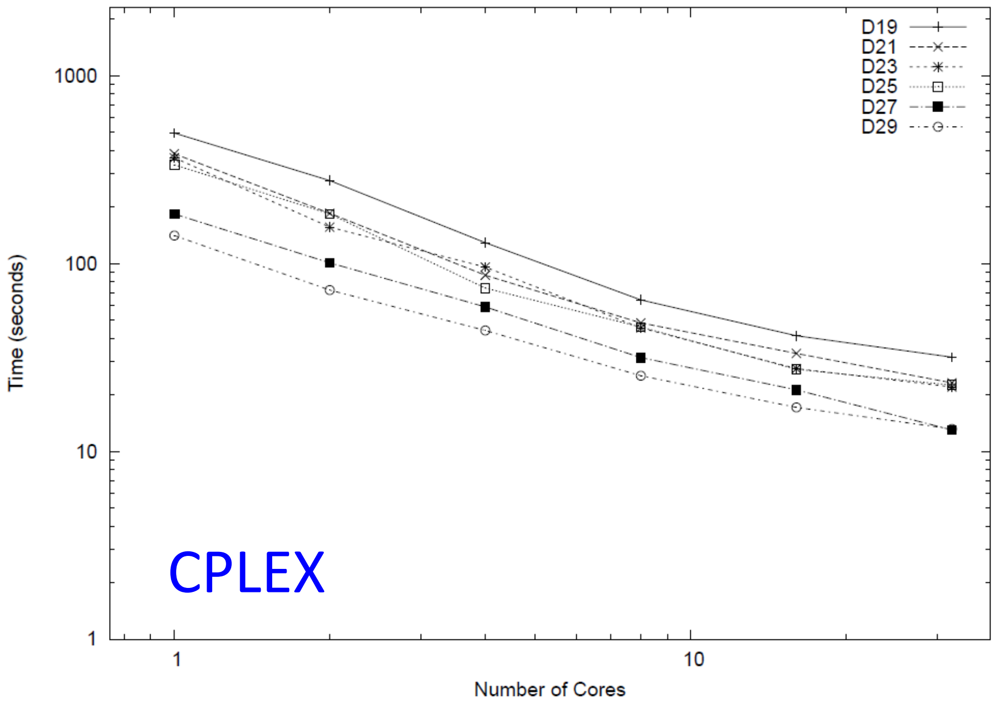
Parallelization: BDD vs CPLEX

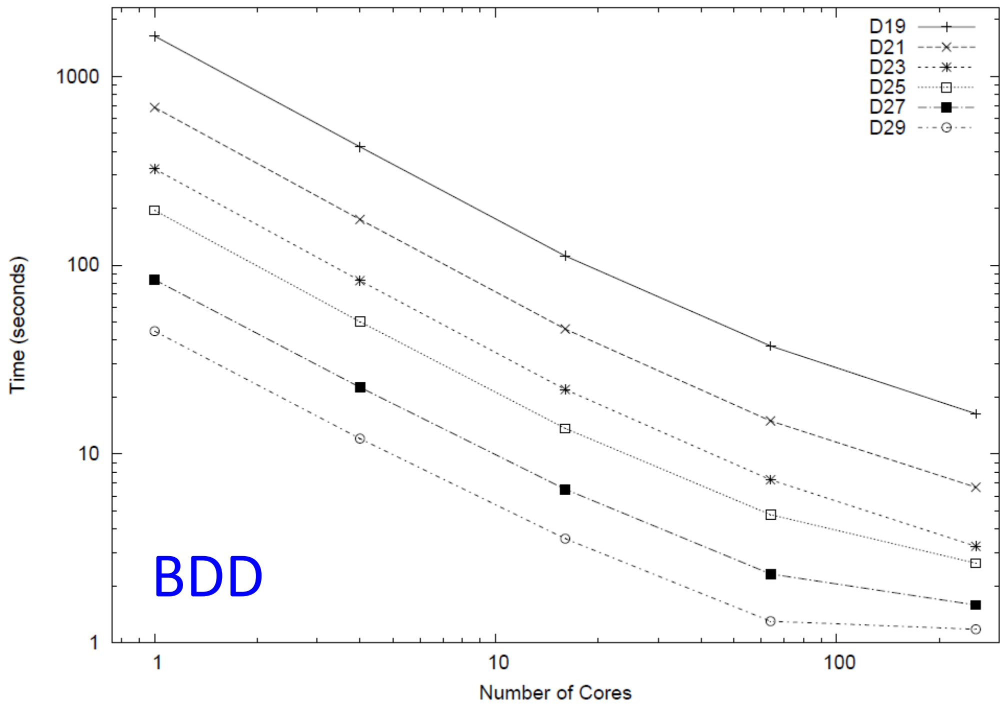




- n = 170, each data point avg over 30 instances
- 1 worker: BDD 1.25 times faster than CPLEX (density 0.29)
- 32 workers: BDD 5.5 times faster than CPLEX (density 0.29)

Parallelization: BDD vs CPLEX



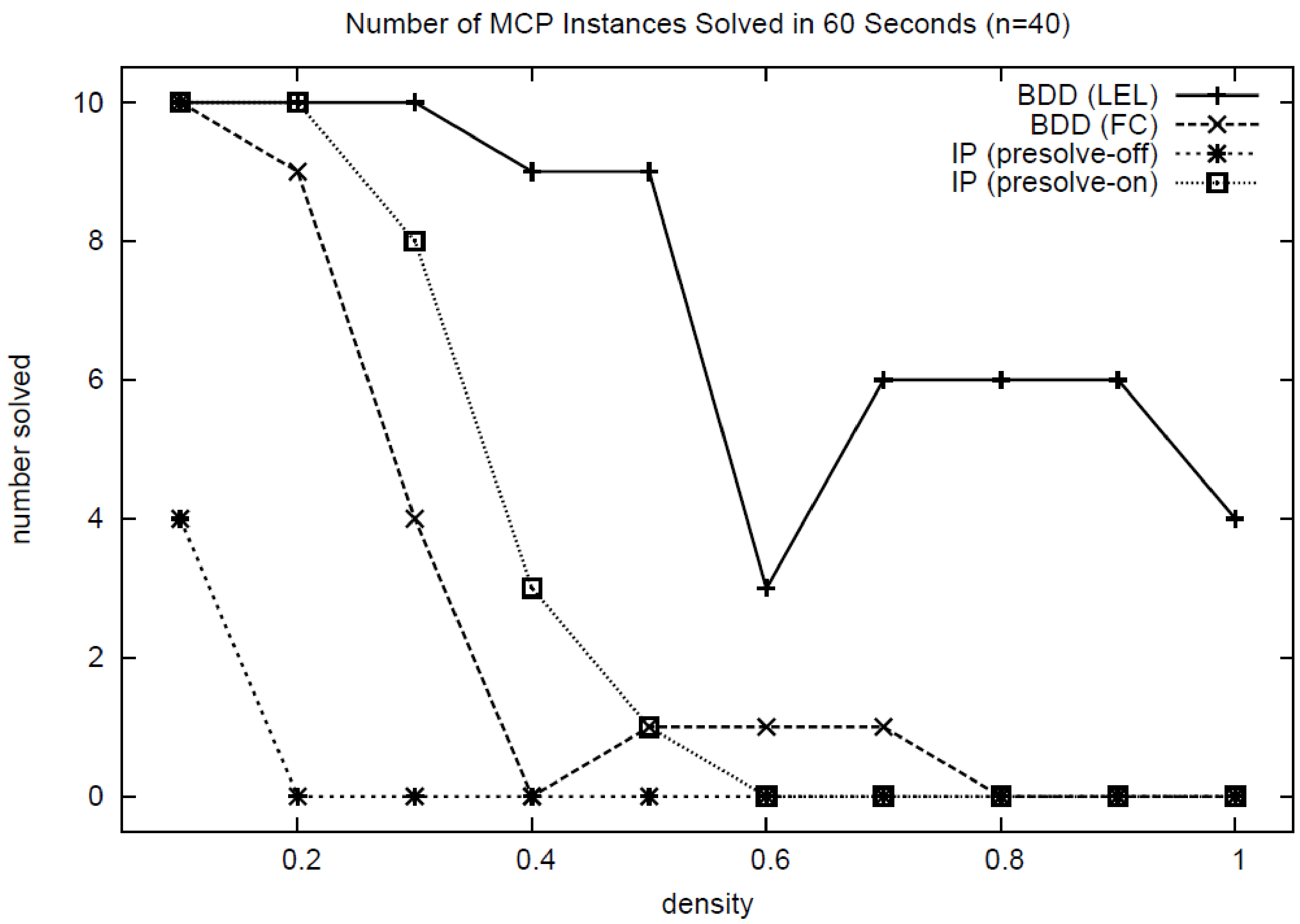


- n = 170, each data point avg over 30 instances
- 1 worker: BDD 1.25 times faster than CPLEX (density 0.29)
- 32 workers: BDD 5.5 times faster than CPLEX (density 0.29)
- BDDs scale to well to (at least) 256 workers

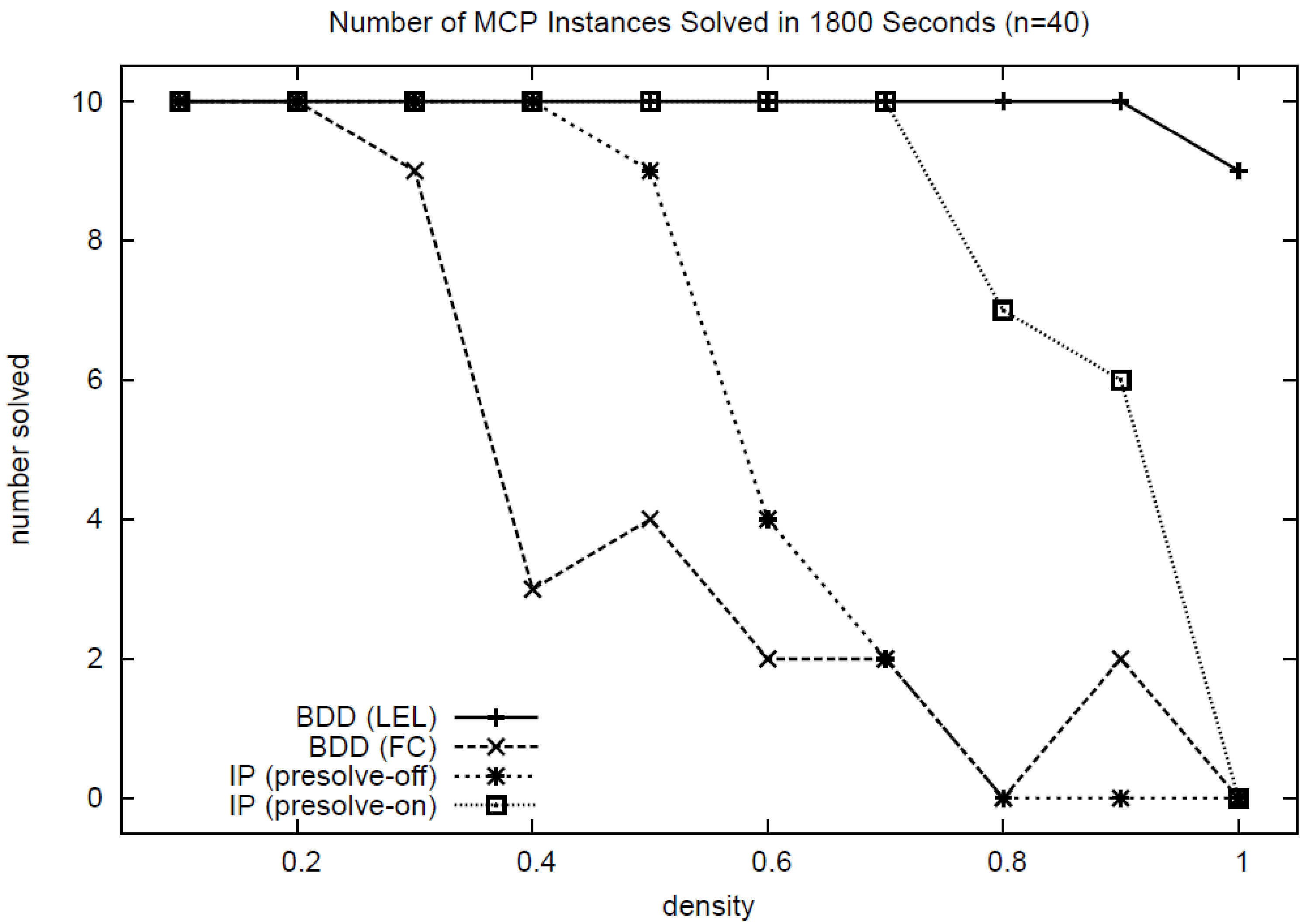
MAX-CUT Results

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

MIP vs BDD: 60 seconds (n=40)



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



BiqMac vs BDD

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

Optimality Gap Improvements

- Reduced optimality gap for several benchmark instances

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

Constraint Programming with Decision Diagrams

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{1,2\}$, $x_2 \in \{0,1,2,3\}$, $x_3 \in \{2,3\}$, $x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$\boxed{x_1 > x_2}$

$x_1 + x_2 = x_3$

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

$x_1 \in \{1,2\}$, $x_2 \in \{0,1,2,3\}$, $x_3 \in \{2,3\}$, $x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$\boxed{x_1 > x_2}$

$x_1 + x_2 = x_3$

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

$x_1 \in \{1,2\},\ x_2 \in \{0,1,\cancel{2},\cancel{3}\},\ x_3 \in \{2,3\},\ x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$\boxed{x_1 + x_2 = x_3}$

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

$x_1 \in \{1,2\}$, $x_2 \in \{0,1,\cancel{2},\cancel{3}\}$, $x_3 \in \{2,3\}$, $x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{1,2\},\ x_2 \in \{0,1,\cancel{2},\cancel{3}\},\ x_3 \in \{2,3\},\ x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{\cancel{1},2\}$, $x_2 \in \{0,1,\cancel{2},\cancel{3}\}$, $x_3 \in \{2,3\}$, $x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{\cancel{1}, 2\}$, $x_2 \in \{0, 1, \cancel{2}, \cancel{3}\}$, $x_3 \in \{\cancel{2}, 3\}$, $x_4 \in \{0, 1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$\boxed{x_1 > x_2}$

$x_1 + x_2 = x_3$

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

$x_1 \in \{\cancel{1}, 2\}$, $x_2 \in \{0, 1, \cancel{2}, \cancel{3}\}$, $x_3 \in \{\cancel{2}, 3\}$, $x_4 \in \{0, 1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$\boxed{x_1 + x_2 = x_3}$

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

$x_1 \in \{\cancel{1},2\},\ x_2 \in \{0,1,\cancel{2},\cancel{3}\},\ x_3 \in \{\cancel{2},3\},\ x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$\boxed{x_1 + x_2 = x_3}$

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

$x_1 \in \{\cancel{1},2\}$, $x_2 \in \{\cancel{0},1,\cancel{2},\cancel{3}\}$, $x_3 \in \{\cancel{2},3\}$, $x_4 \in \{0,1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{\cancel{1}, 2\}$, $x_2 \in \{\cancel{0}, 1, \cancel{2}, \cancel{3}\}$, $x_3 \in \{\cancel{2}, 3\}$, $x_4 \in \{0, 1\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{\cancel{1},2\},\ x_2 \in \{\cancel{0},1,\cancel{2},\cancel{3}\},\ x_3 \in \{\cancel{2},3\},\ x_4 \in \{0,\cancel{1}\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{\cancel{1}, 2\}$, $x_2 \in \{\cancel{0}, 1, \cancel{2}, \cancel{3}\}$, $x_3 \in \{\cancel{2}, 3\}$, $x_4 \in \{0, \cancel{1}\}$

Motivation

- Constraint Programming applies constraint propagation
  - Remove provably inconsistent values from variable domains
  - Propagate updated domains to other constraints

$x_1 > x_2$

$x_1 + x_2 = x_3$

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

$x_1 \in \{\cancel{1},2\},\ x_2 \in \{\cancel{0},1,\cancel{2},\cancel{3}\},\ x_3 \in \{\cancel{2},3\},\ x_4 \in \{0,\cancel{1}\}$

domain propagation
can be weak, however…

Illustrative example

$$\mathit{alldifferent}(x_1,x_2,x_3,x_4) \quad (1)$$

$$x_1 + x_2 + x_3 \geq 9 \quad (2)$$

$$x_i \in \{1,2,3,4\}$$

Illustrative example

$$\textit{alldifferent}(x_1, x_2, x_3, x_4) \quad (1)$$

$$x_1 + x_2 + x_3 \geq 9 \quad (2)$$

$$x_i \in \{1,2,3,4\}$$

(1) and (2) are both domain consistent

(i.e., no propagation)

Illustrative example

$alldifferent(x_1,x_2,x_3,x_4)$ (1)

$x_1 + x_2 + x_3 \geq 9$ (2)

$x_i \in \{1,2,3,4\}$

(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

x_1	x_2	x_3	x_4
1	2	3	4
1	2	4	3
1	3	2	4
	…		
4	3	2	1

Illustrative example

$$alldifferent(x_1,x_2,x_3,x_4) \quad (1)$$

$$x_1 + x_2 + x_3 \geq 9 \quad (2)$$

$$x_i \in \{1,2,3,4\}$$

(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

x_1	x_2	x_3	x_4
1	2	3	4
1	2	4	3
1	3	2	4
	…		
4	3	2	1

domain projection: $D(x_i) = \{1,2,3,4\}$

Illustrative example

alldifferent(x_1,x_2,x_3,x_4) (1)

$x_1 + x_2 + x_3 \geq 9$ (2)

$x_i \in \{1,2,3,4\}$

(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

x_1	x_2	x_3	x_4
1	2	3	4
1	2	4	3
1	3	2	4
	…		
4	3	2	1

Suppose we could evaluate (2) on this list

domain projection: $D(x_i) = \{1,2,3,4\}$

Illustrative example

$alldifferent(x_1,x_2,x_3,x_4)$ (1)

$x_1 + x_2 + x_3 \geq 9$ (2)

$x_i \in \{1,2,3,4\}$

(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

x_1	x_2	x_3	x_4
~~1~~	~~2~~	~~3~~	~~4~~
1	2	4	3
1	3	2	4
	…		
4	3	2	1

Suppose we could evaluate (2) on this list

domain projection: $D(x_i) = \{1,2,3,4\}$

Illustrative example

$alldifferent(x_1,x_2,x_3,x_4)$ (1)

$x_1 + x_2 + x_3 \geq 9$ (2)

$x_i \in \{1,2,3,4\}$

(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

x_1	x_2	x_3	x_4
~~1~~	~~2~~	~~3~~	~~4~~
~~1~~	~~2~~	~~4~~	~~3~~
1	3	2	4
	…		
4	3	2	1

Suppose we could evaluate (2) on this list

domain projection: $D(x_i) = \{1,2,3,4\}$

Illustrative example

$alldifferent(x_1,x_2,x_3,x_4)$ (1)

$x_1 + x_2 + x_3 \geq 9$ (2)

$x_i \in \{1,2,3,4\}$

(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

x_1	x_2	x_3	x_4
~~1~~	~~2~~	~~3~~	~~4~~
~~1~~	~~2~~	~~4~~	~~3~~
~~1~~	~~3~~	~~2~~	~~4~~
	…		
4	3	2	1

Suppose we could evaluate (2) on this list

domain projection: $D(x_i) = \{1,2,3,4\}$

Illustrative example

$alldifferent(x_1,x_2,x_3,x_4)$ (1)

$x_1 + x_2 + x_3 \geq 9$ (2)

$x_i \in \{1,2,3,4\}$

(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

	x_1	x_2	x_3	x_4
✓	2	3	4	1
✓	2	4	3	1
✓	3	2	4	1
		…		
✓	4	3	2	1

Suppose we could evaluate (2) on this list

domain projection: $D(x_4) = \{1\}$
$D(x_1) = D(x_2) = D(x_3) = \{2,3,4\}$

Illustrative example

$alldifferent(x_1,x_2,x_3,x_4)$ (1)

$x_1 + x_2 + x_3 \geq 9$ (2)

$x_i \in \{1,2,3,4\}$

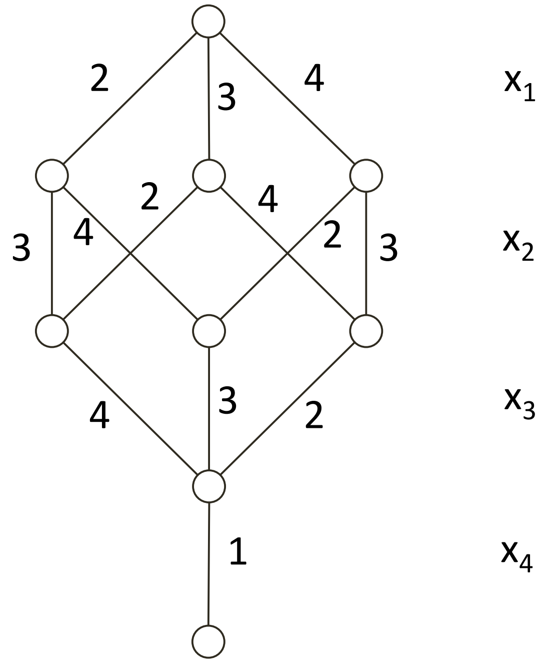
(1) and (2) are both domain consistent

(i.e., no propagation)

List of all solutions to *alldifferent*:

	x_1	x_2	x_3	x_4
✓	2	3	4	1
✓	2	4	3	1
✓	3	2	4	1
		…		
✓	4	3	2	1

Use MDD!



Motivation for MDD propagation

- Conventional domain propagation: all structural relationships among variables are lost after domain projection
- Potential solution space is implicitly defined by Cartesian product of variable domains (very coarse relaxation)

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

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

MDD-based Constraint Programming

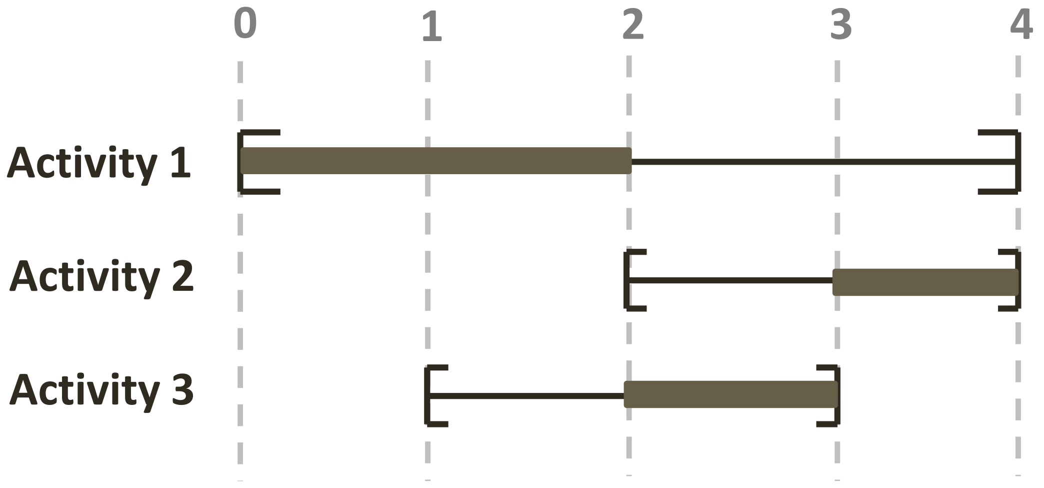
- Maintain limited-width MDD
  - Serves as relaxation
  - Typically start with width 1 (initial variable domains)
  - Dynamically adjust MDD, based on constraints
- Constraint Propagation
  - Edge filtering: Remove provably inconsistent edges (those that do not participate in any solution)
  - Node refinement: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

Specific MDD propagation algorithms

- Linear equalities and inequalities [Hadzic et al., 2008] [Hoda et al., 2010]
- *Alldifferent* constraints [Andersen et al., 2007]
- *Element* constraints [Hoda et al., 2010]
- *Among* constraints [Hoda et al., 2010]
- Disjunctive scheduling constraints [Hoda et al., 2010] [Cire & v.H., 2011, 2013]
- *Sequence* constraints (combination of *Amongs*) [Bergman et al., 2014]
- Generic re-application of existing domain filtering algorithm for any constraint type [Hoda et al., 2010]

Application to 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



Scheduling: Model 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
  - …

DDs for Disjunctive Scheduling

Three main considerations:

- Representation
  - – How to represent solutions of disjunctive scheduling in a DD?
- Construction
  - – How to construct the DD?
- Inference techniques
  - – What can we infer using the DD?

Decision Diagram Representation

- 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.:

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

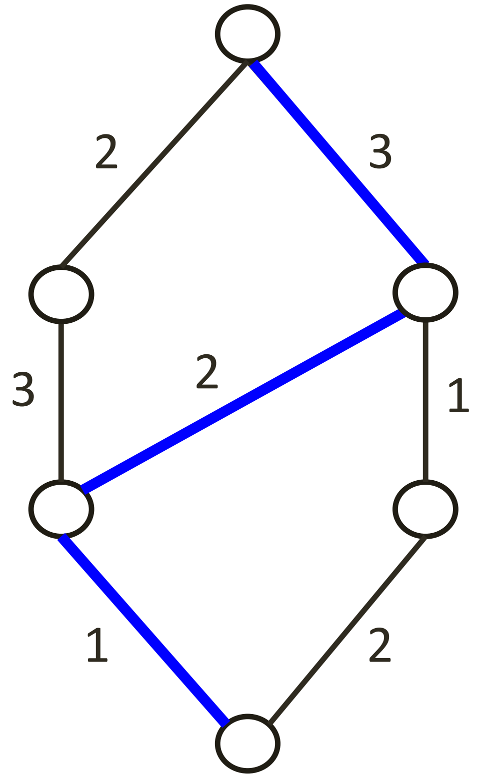
- Represent feasible permutations with **multi-valued** decision diagram (MDD)

[Cire&vH, OR 2013]

MDD Representation: Example

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

precedence: $3 \ll 1$



π_1

π_2

π_3

Path 3 – 2 – 1 :

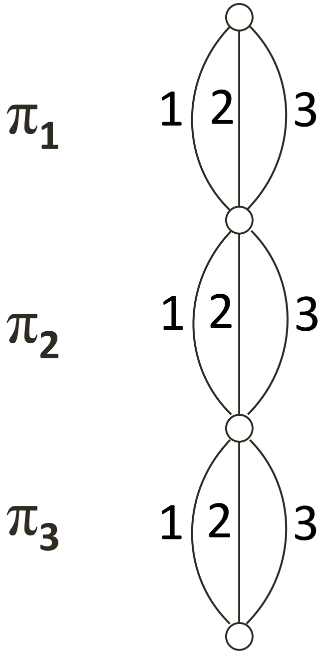
$6 \leq start_1 \leq 8$

$3 \leq start_2 \leq 5$

$1 \leq start_3 \leq 3$

Top-down MDD compilation

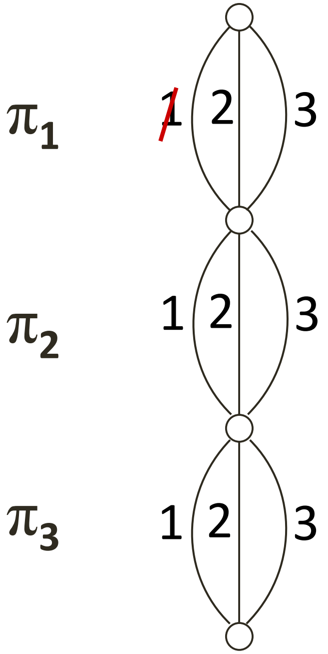
precedence:
$3 \ll 1$



Top-down MDD compilation

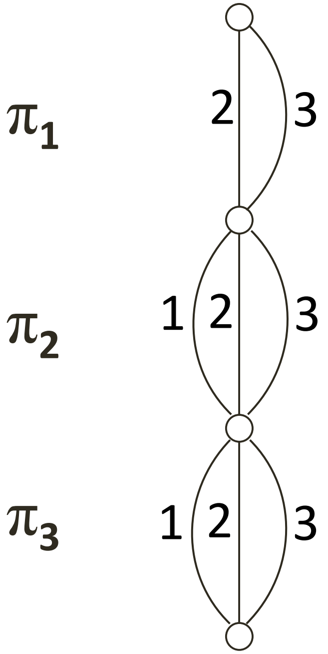
precedence:

$3 \ll 1$



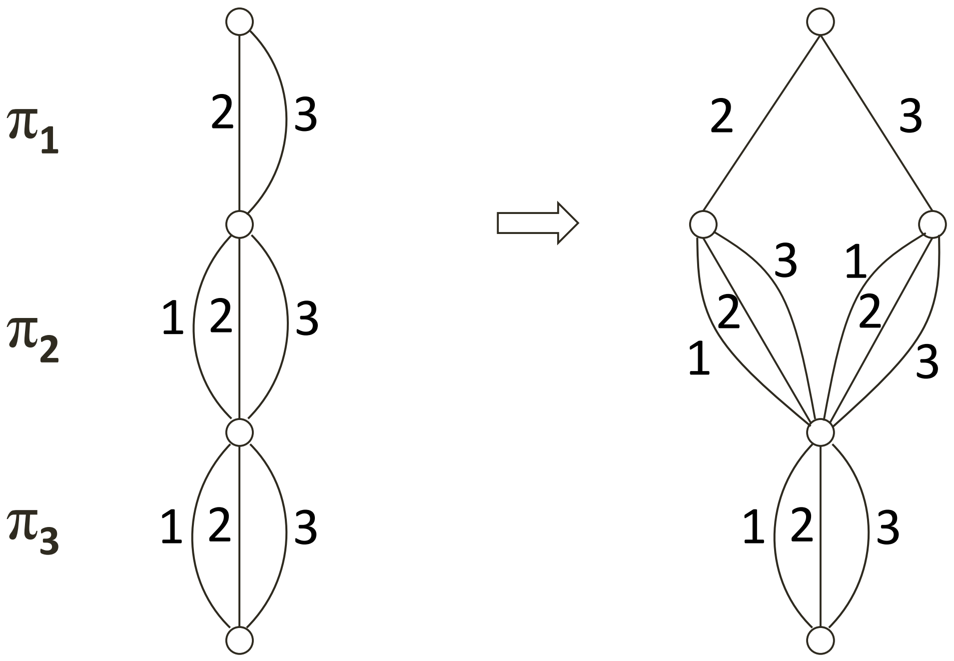
Top-down MDD compilation

precedence:
$3 \ll 1$



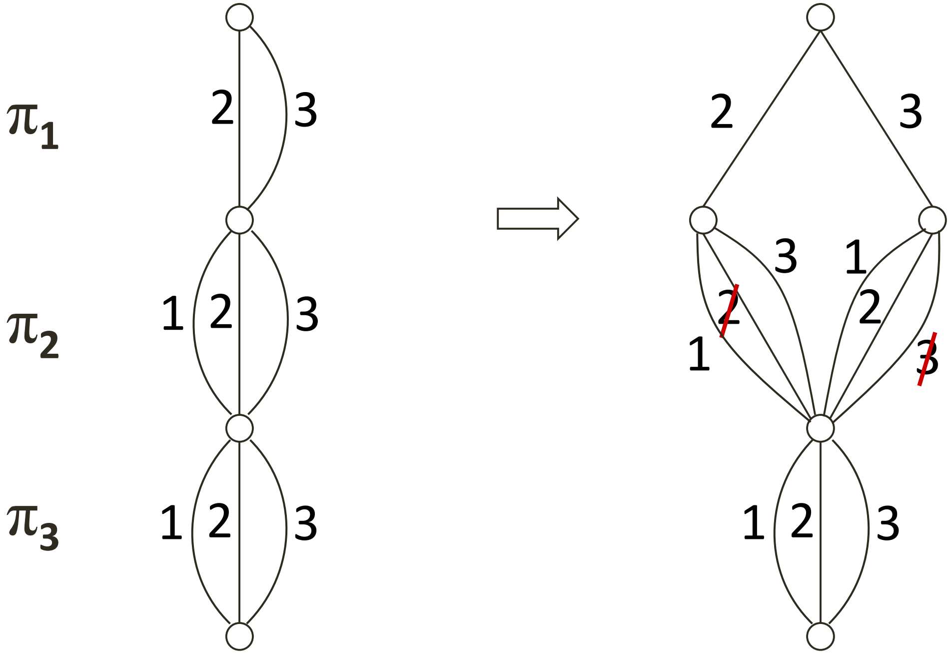
Top-down MDD compilation

precedence:
$3 \ll 1$



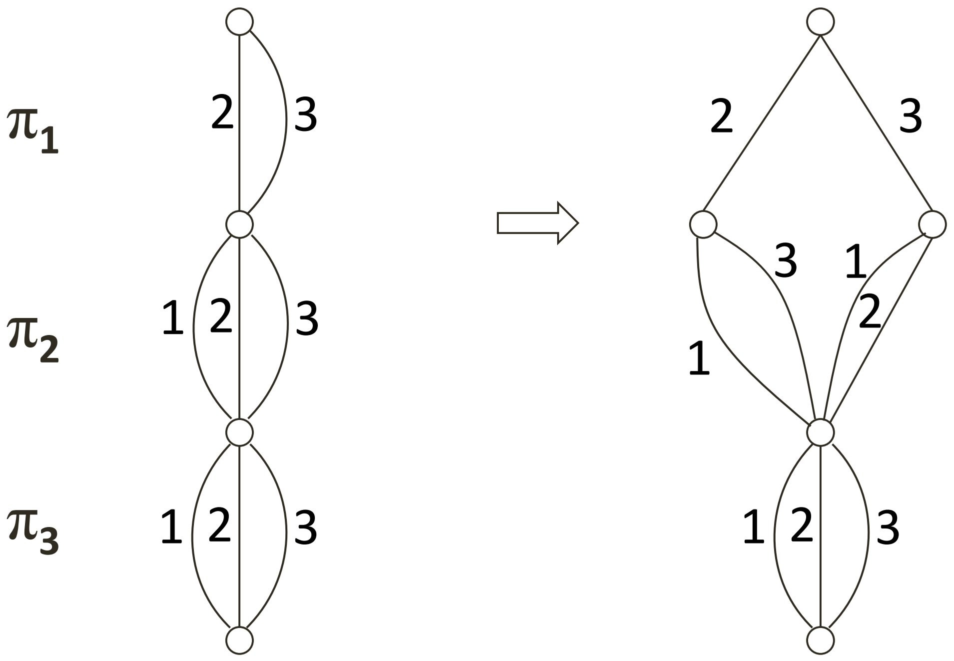
Top-down MDD compilation

precedence:
$3 \ll 1$



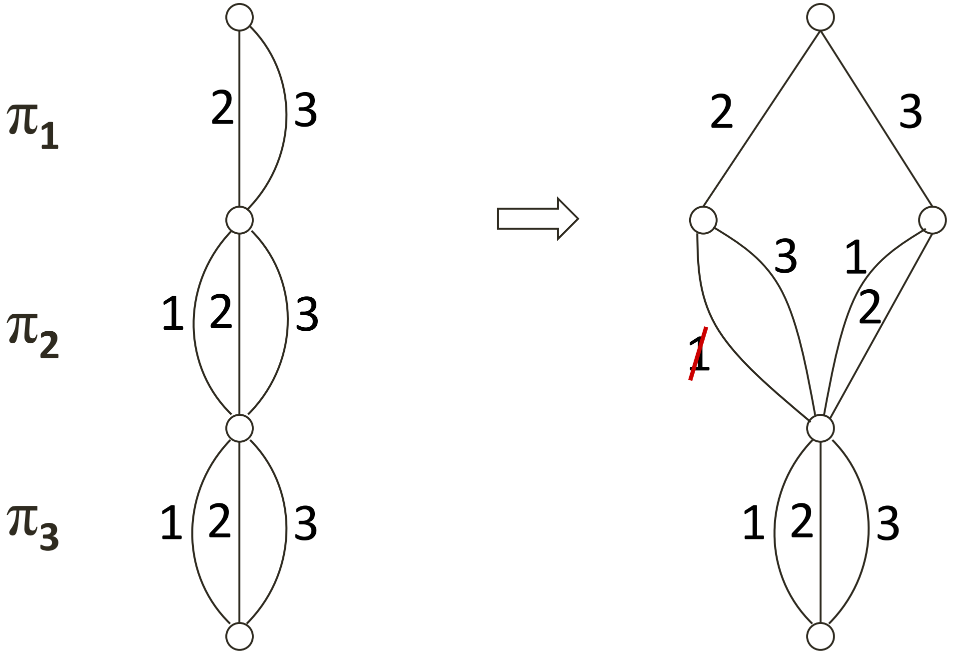
Top-down MDD compilation

precedence:
$3 \ll 1$



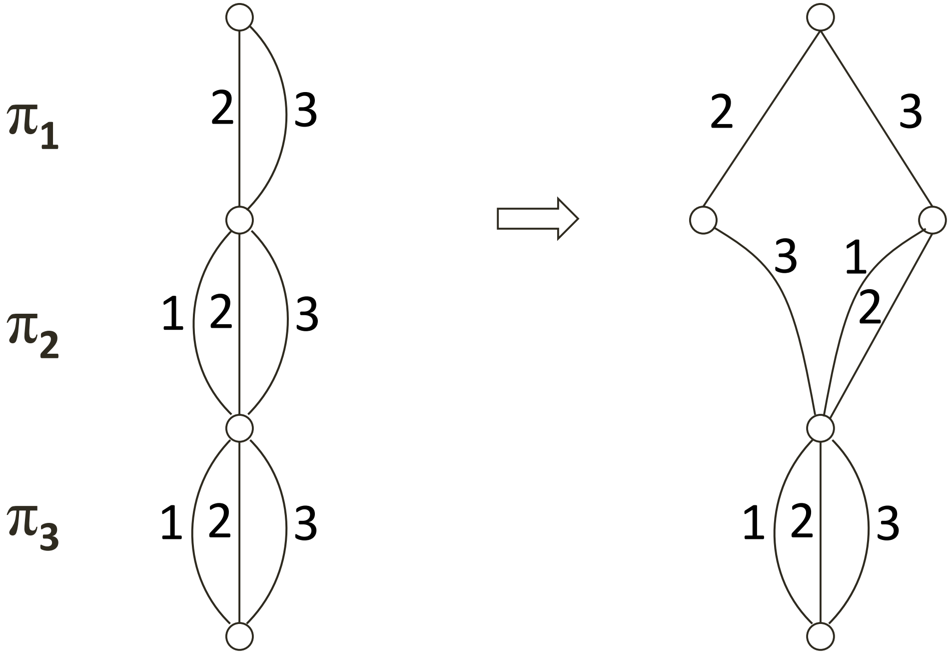
Top-down MDD compilation

precedence:
$3 \ll 1$



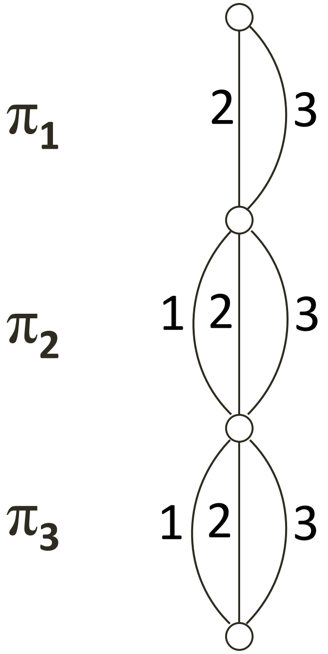
Top-down MDD compilation

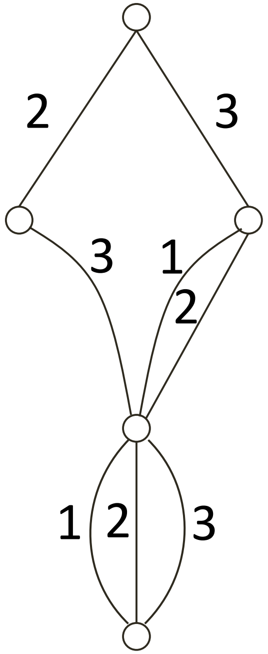
precedence:
$3 \ll 1$



Top-down MDD compilation

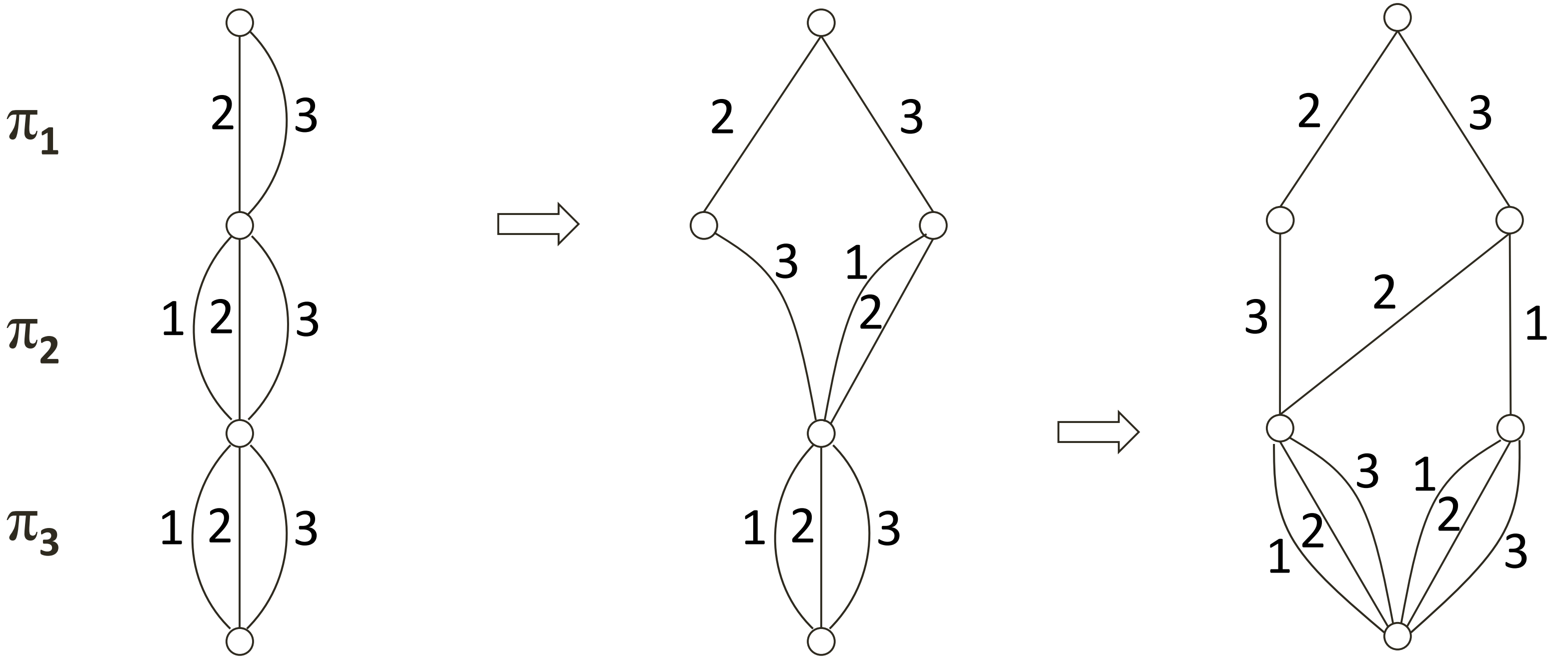
precedence:
$3 \ll 1$





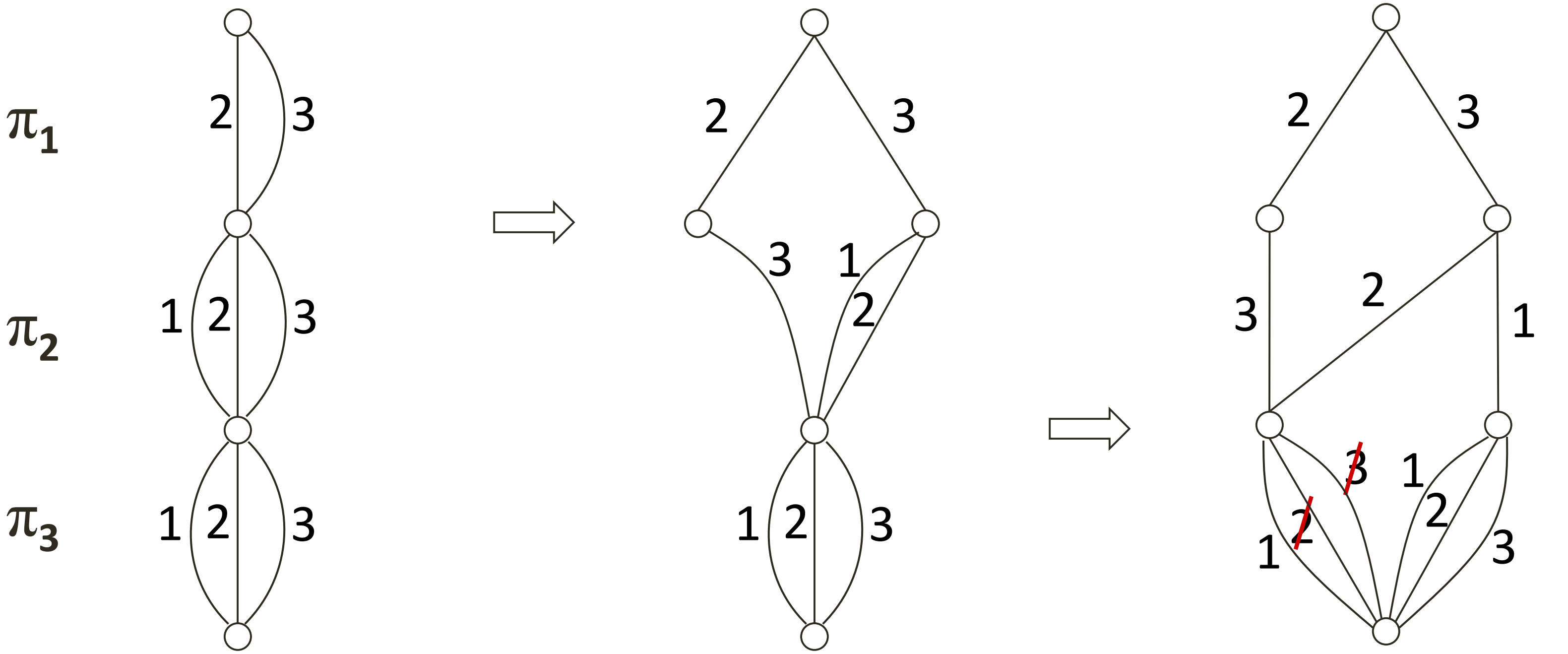
Top-down MDD compilation

precedence:
$3 \ll 1$



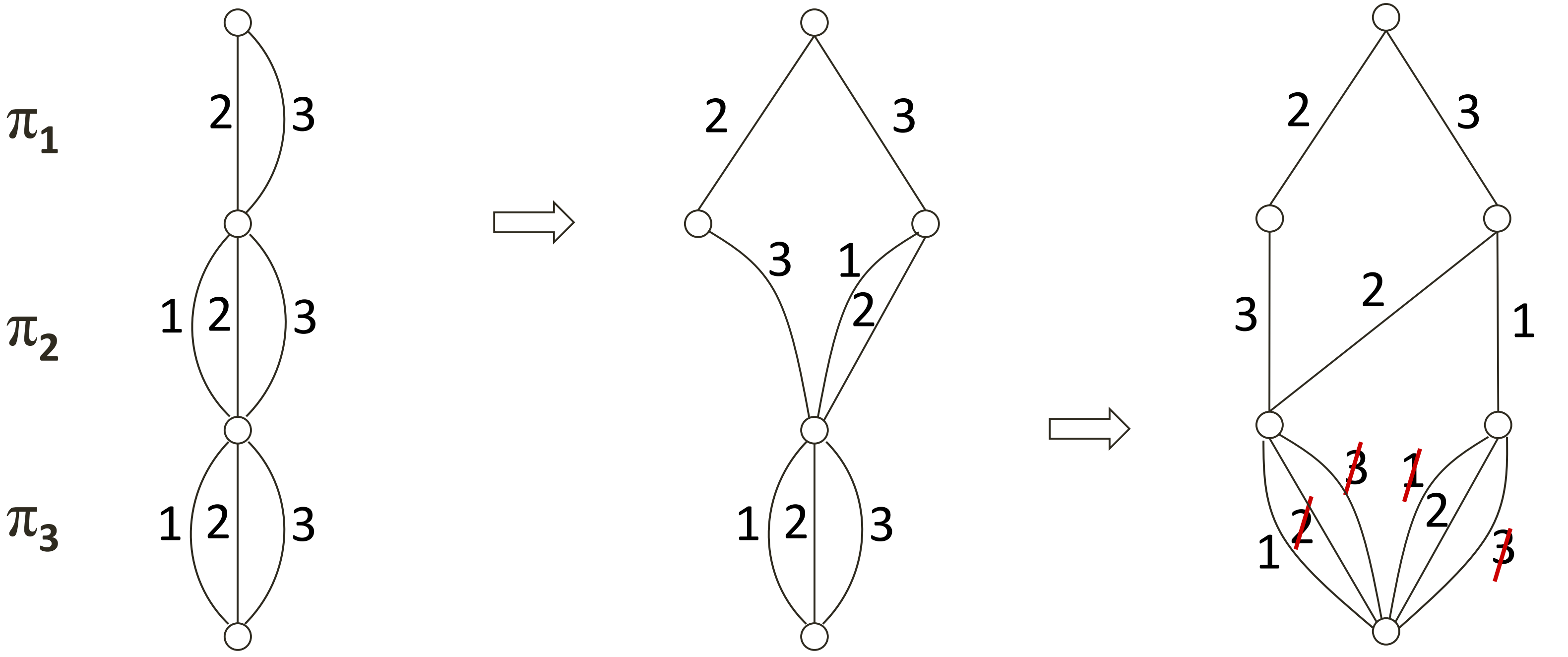
Top-down MDD compilation

precedence:
$3 \ll 1$



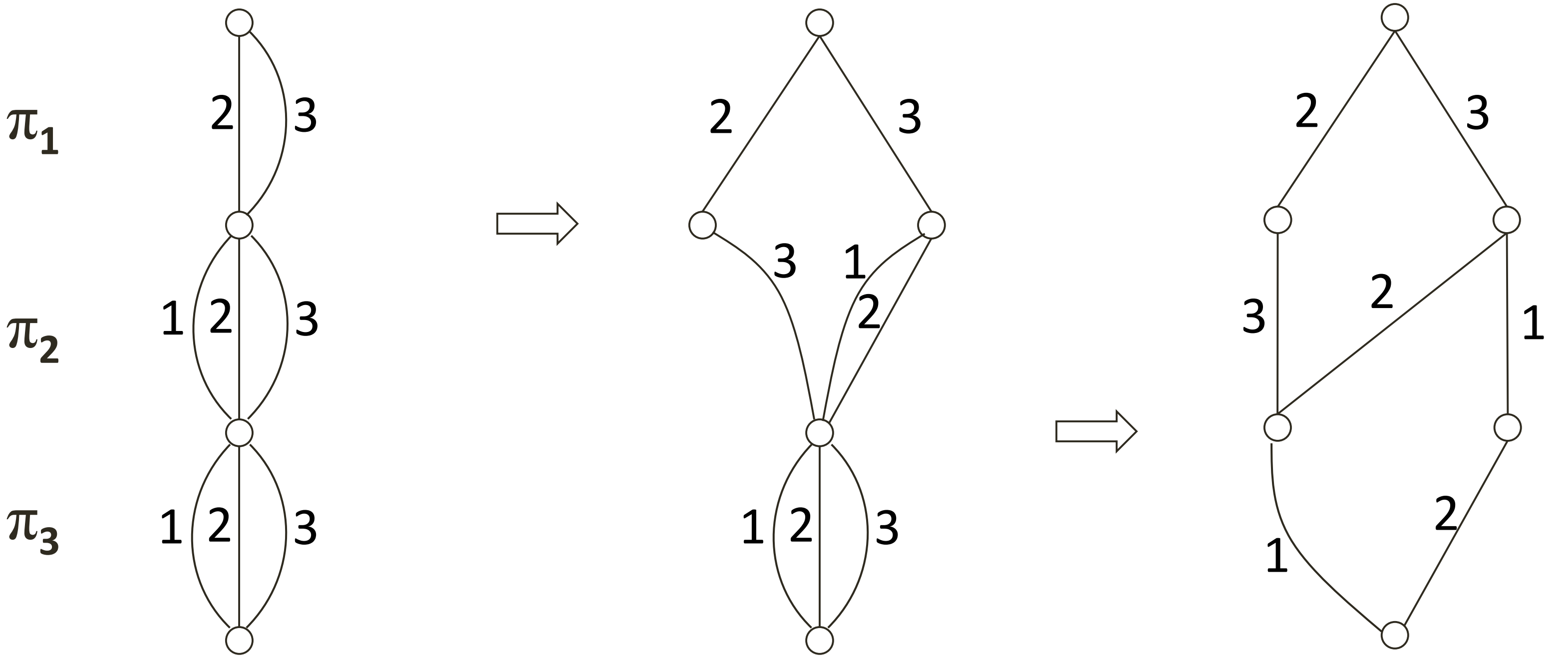
Top-down MDD compilation

precedence:
$3 \ll 1$



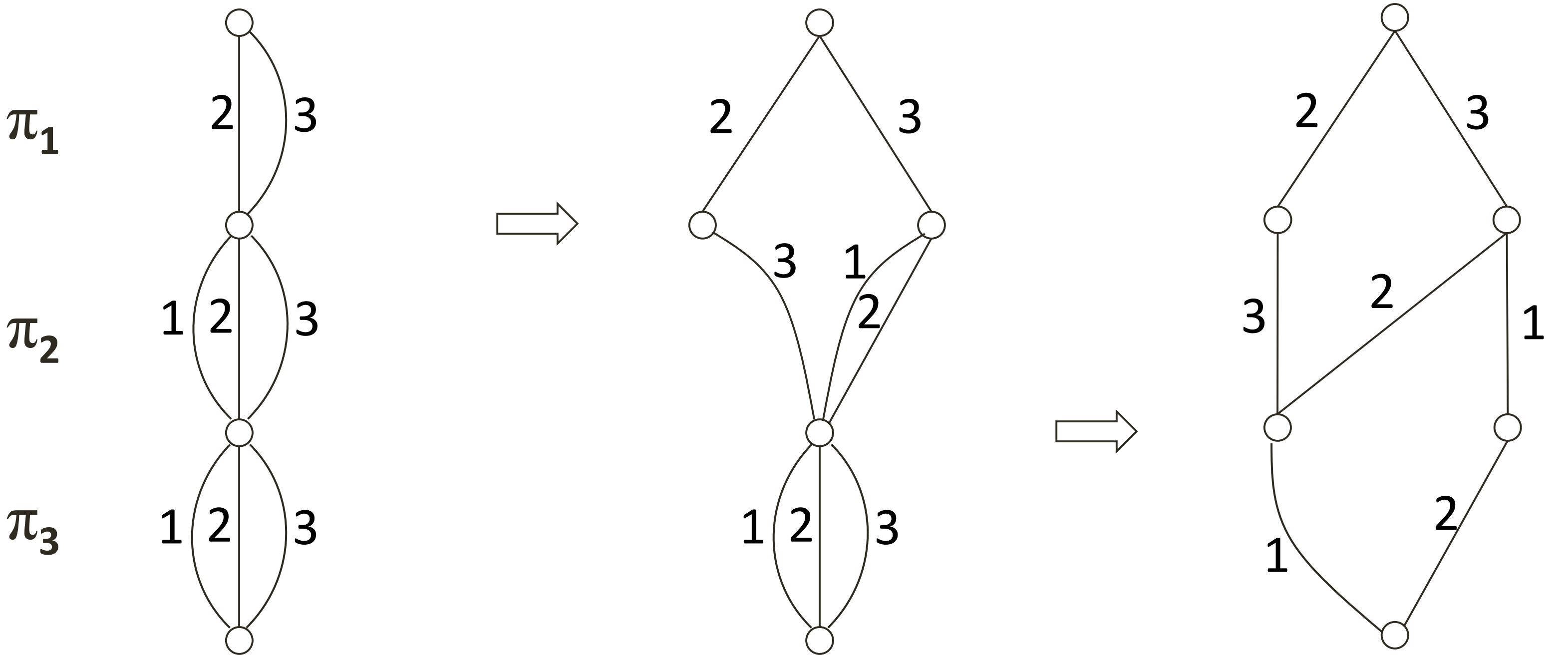
Top-down MDD compilation

precedence:
$3 \ll 1$



Top-down MDD compilation

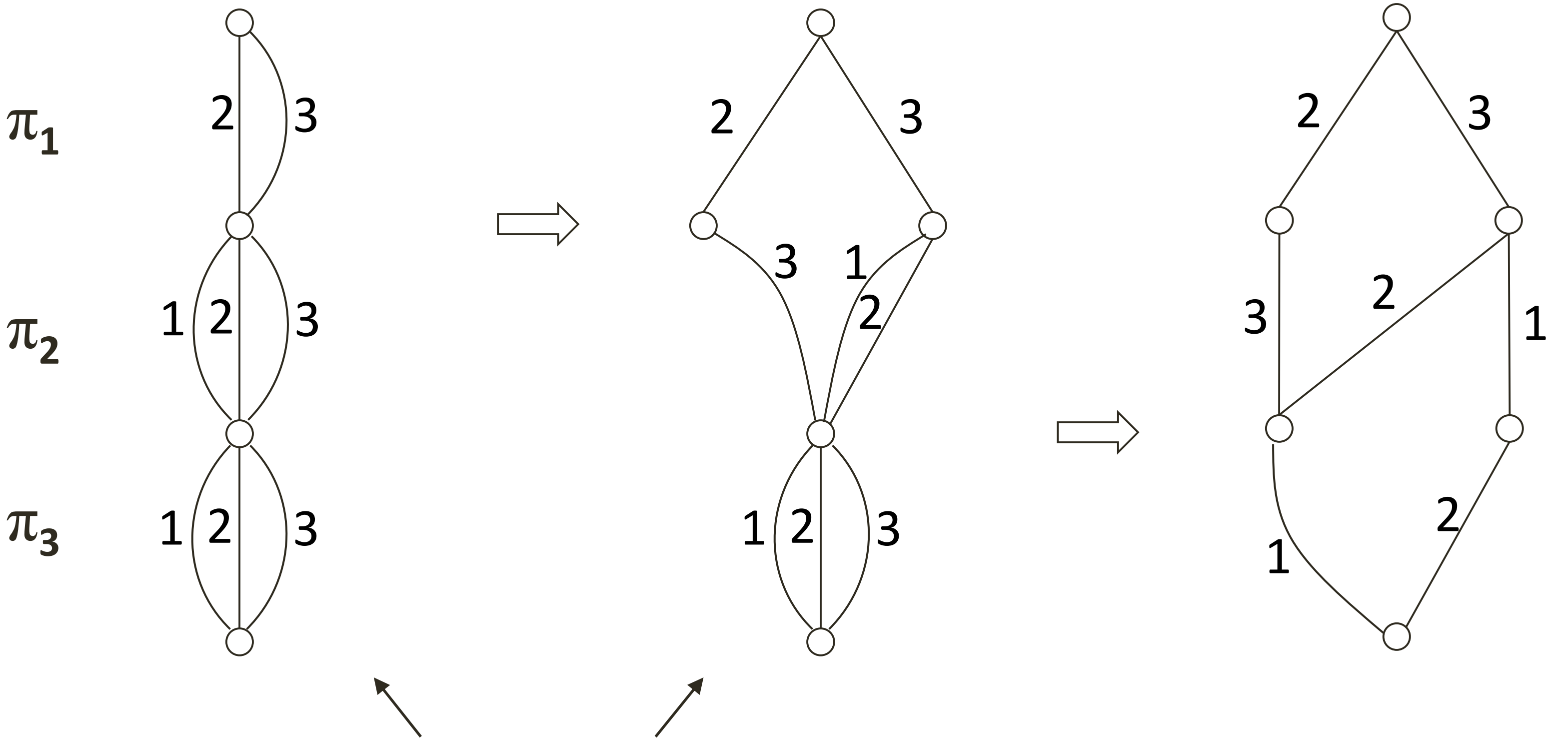
precedence:
$3 \ll 1$



exact MDD

Top-down MDD compilation

precedence:
3 ≪ 1

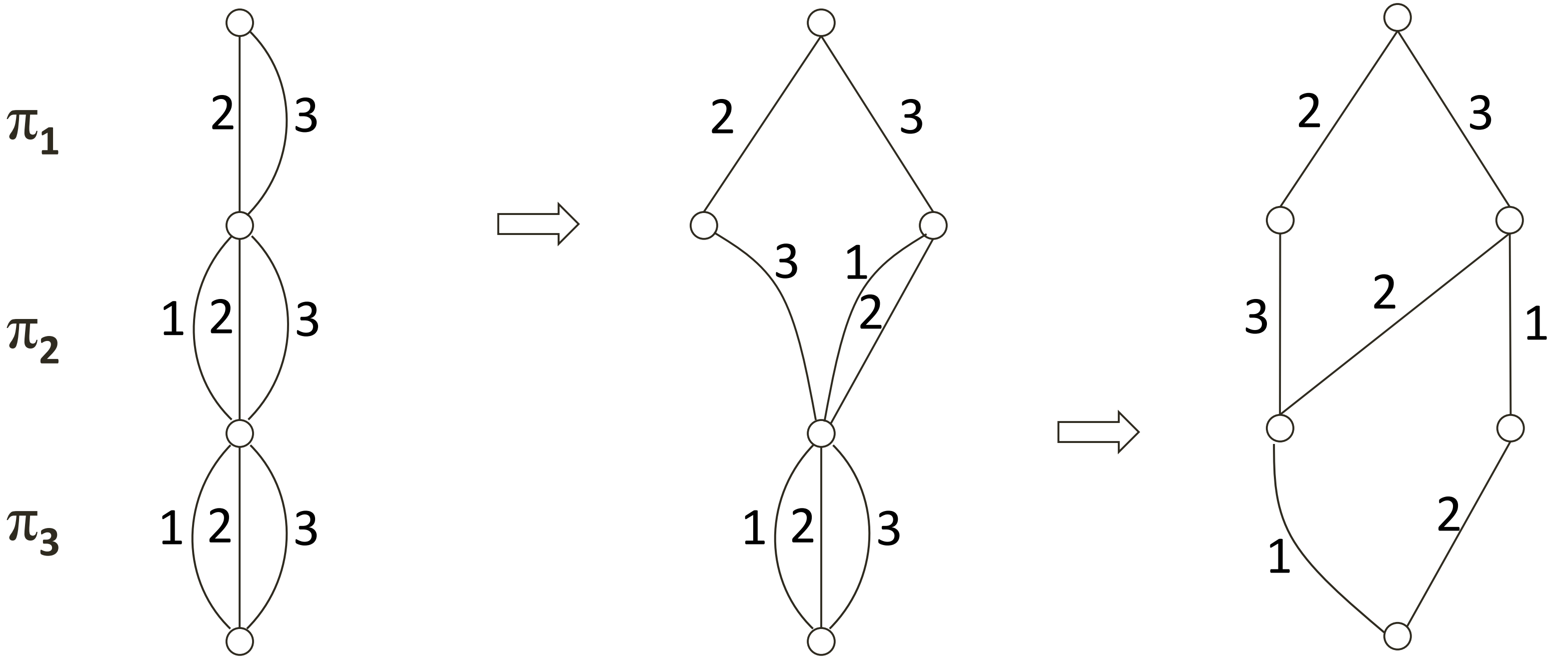


relaxed MDDs
(strength is controlled by maximum width)

exact MDD

Top-down MDD compilation

precedence:
$3 \ll 1$

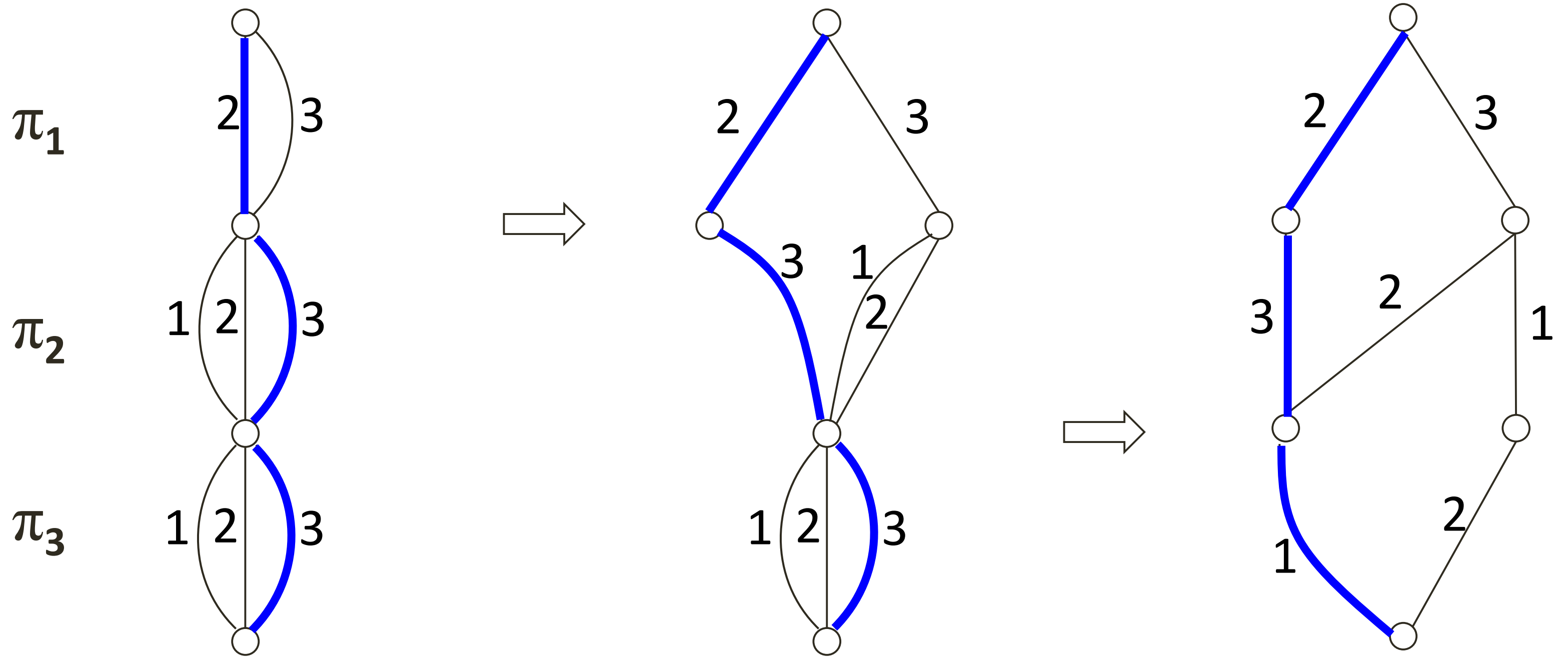


Top-down MDD compilation

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:

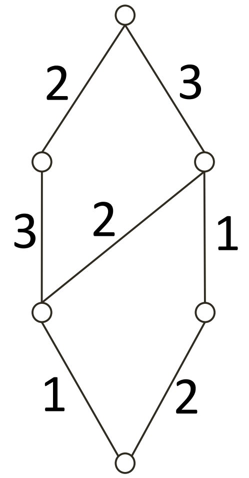


lower bound = 7 | lower bound = 7 | optimum = 9

MDD State Information

We need to represent several problem components:

- Permutation structure (“AllDifferent”)
  - state information: set of values taken on paths from root to state
- Earliest start time (similar for latest end time)
  - state information: minimum completion time of all paths from root
- Precedence relations
  - can be enforced using the state information for AllDifferent



Exact MDD Compilation

- Theorem: *Constructing the exact MDD for a Disjunctive Instance is NP-Hard*

  (In fact, determining state equivalence is already NP-hard)
- Therefore we use relaxed MDDs
  - specify a maximum width
- MDDs of bounded width exist for special cases
  - for example for structured precedence relations

Inference from the MDD

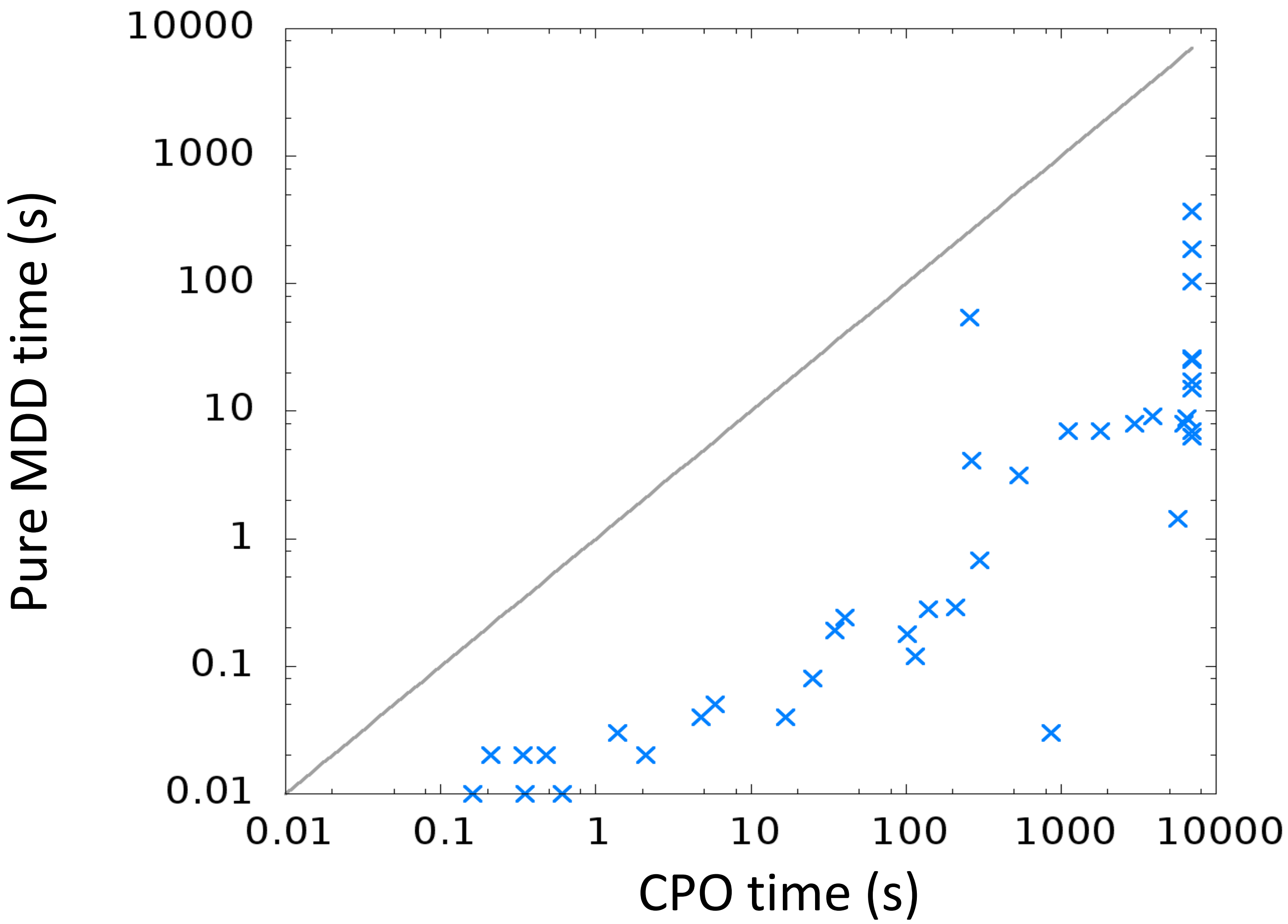
- Theorem: *Given exact MDD M, we can deduce all implied activity precedences in* $O(n^2|M|)$ *time*

- The algorithm can also be applied to *relaxed* MDD to find a subset of precedences
  – can be stronger than edge-finding, not-first/not-last, etc.

Performance

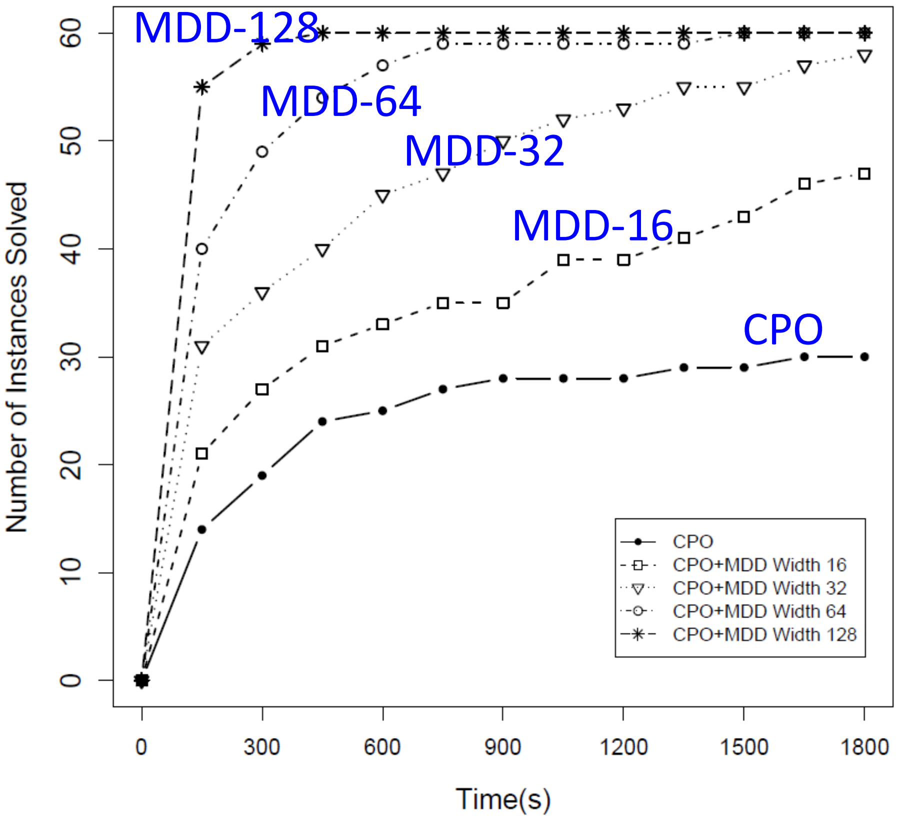
- MDD propagation implemented in IBM ILOG CPLEX CP Optimizer 12.4 (CPO)
  - State-of-the-art constraint based scheduling solver
  - Uses a portfolio of inference techniques and LP relaxation
  - MDD is added as user-defined propagator
- Compare three different variants
  - CPO (only use CPO propagation)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

TSP with Time Windows

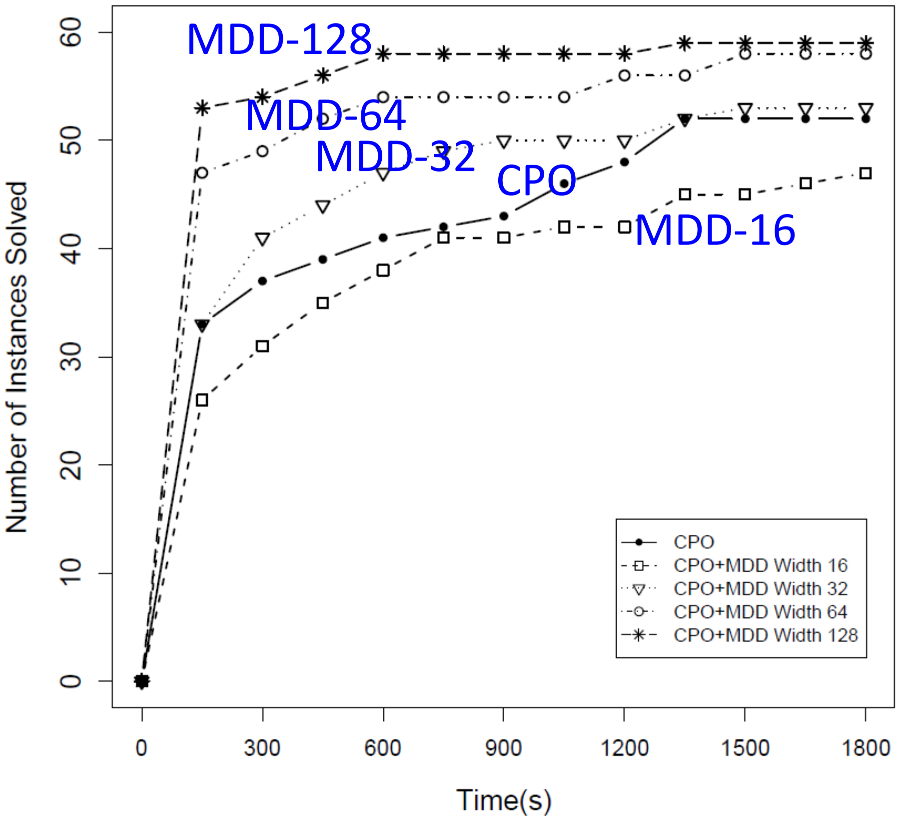


Dumas/Ascheuer instances
- 20-60 cities
- max MDD width: 16

Total Tardiness



total tardiness



total weighted tardiness

Sequential Ordering Problem (TSPLIB)

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

* solved for the first time

Strengthening Relaxed Decision Diagrams

- Lagrangian relaxation
  - penalize constraint violations by modifying arc weights

- Additive bounding
  - incorporate dual information from LP relaxations
  - e.g., aggregate reduced costs along path from root to terminal

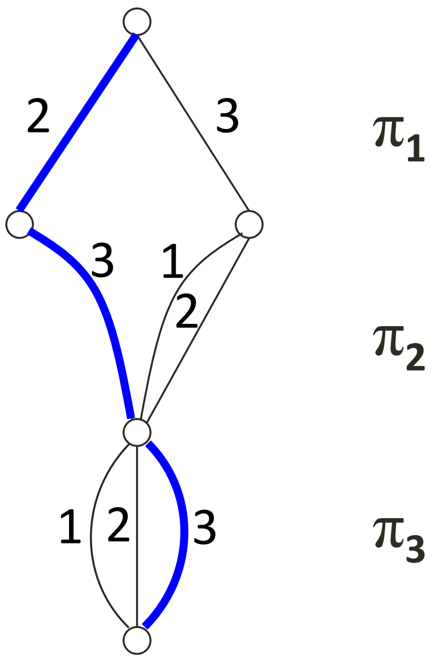
Extension: Lagrangian bounds

- Observation: MDD bounds can be very loose
  - main cause: repetition of activities
- Apply Lagrangian relaxation
  - penalize repeated activities; reward unused activities

$$\min z + \sum_{j=1}^{n} \lambda_j \left(\sum_{i=1}^{n} (\pi_i = j) - 1 \right)$$

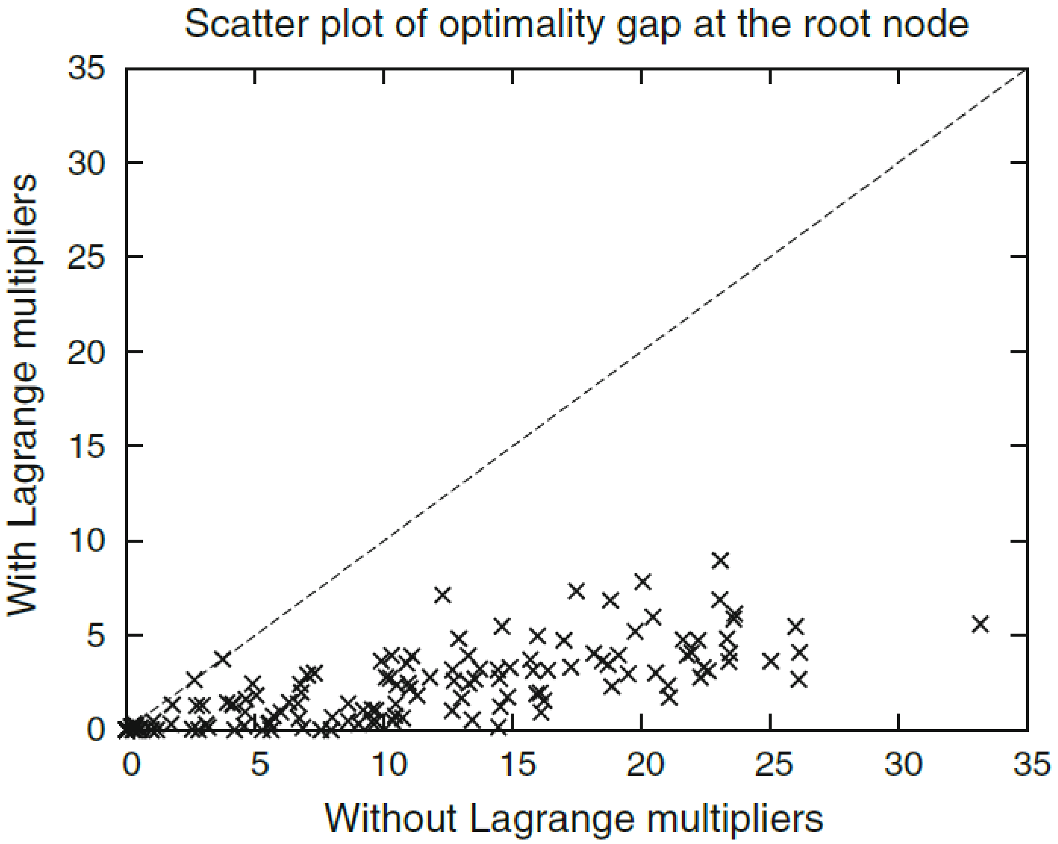
$$= z + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_j (\pi_i = j) - \sum_{j=1}^{n} \lambda_j$$

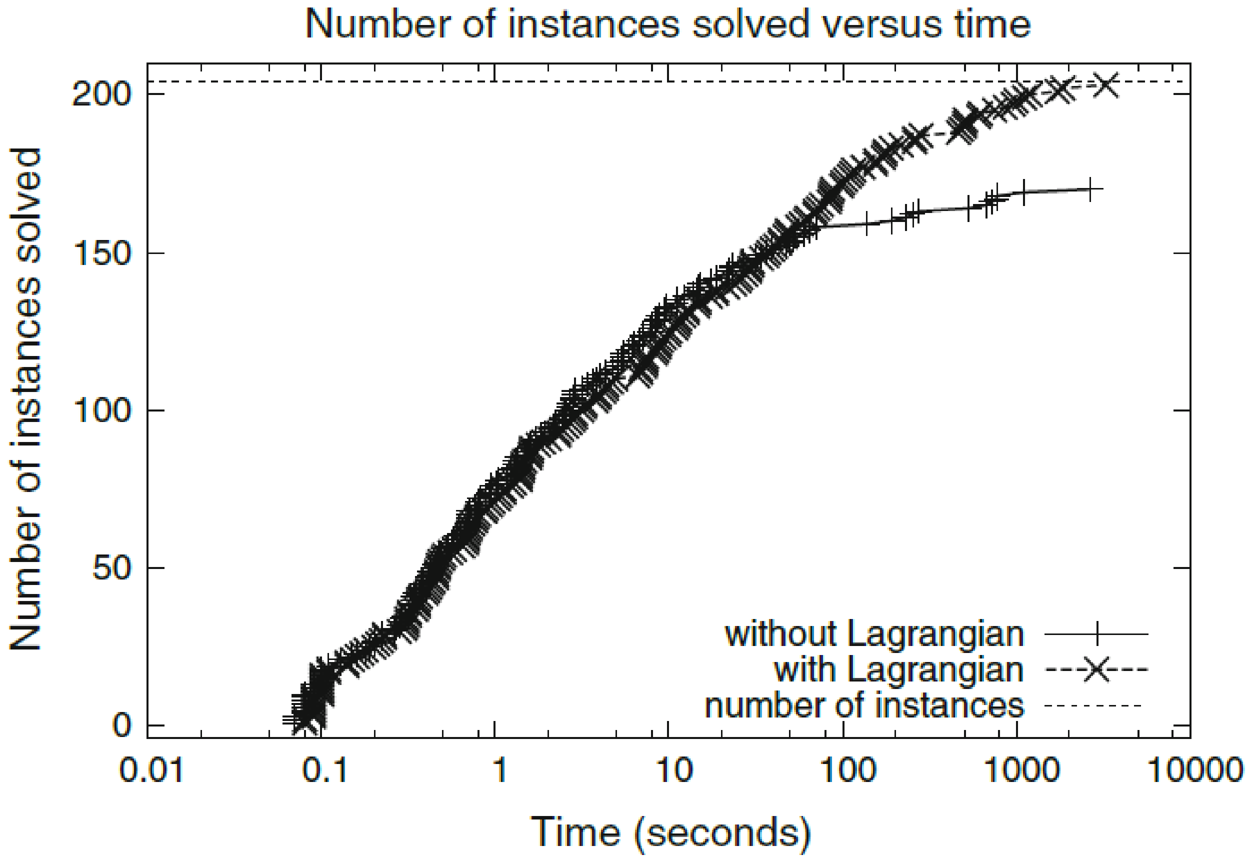
  - shortest path with updated weights



$$\sum_{i=1}^{n} (\pi_i = j) = 1 \quad \forall j$$

Impact of Lagrangian Relaxation (TSPTW)



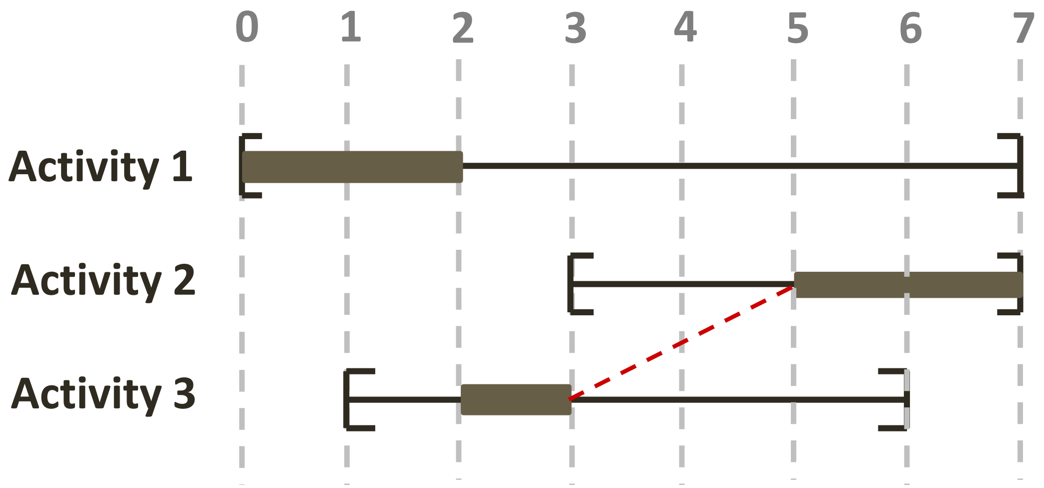


[Bergman, Cire, vH, 2015]

Extension: Additive Bounding

- Case: time-dependent sequencing
  - sequence-dependent setup times also depend on position!
  - $\delta_{i,j}^{t}$ = setup time between i and j if i is at position t

- MDD representation
  - state-dependent costs



[Kinable, Cire, vH, EJOR 2017]

Additive Bounding: LP + MDD

- Add LP reduced costs to MDD relaxation [Fischetti &Toth, 1989]
- Effectivess depends on the quality of the LP relaxation
- LP can be made stronger for specific problem class
  - TD-TSP [Picard & Queyranne, 1978] [Vander Wiel and Sahinidis, 1995] [Gouveia and Voss, 1995] [Abeledo et al. 2013] [Miranda-Bront et al., 2014]
  - TD-TSP-TW (time windows) [Miller,Tucker, Zemlin, 1960] [Desrocher & Laporte, 2014]
  - TD-SOP (precedence constraints) [Sarin, Sherali, Bhootra, 2005]

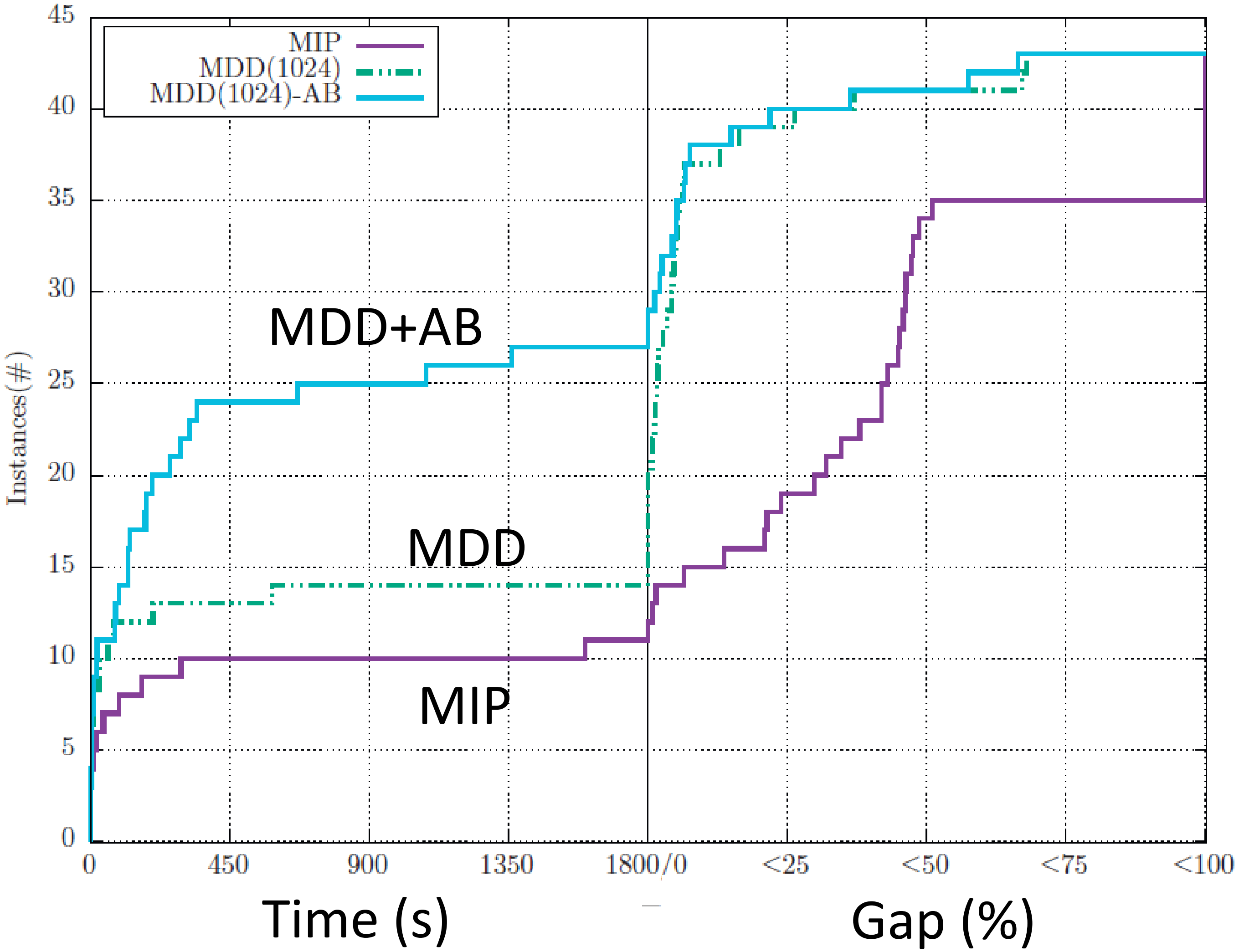
Experimental Setup

- Solvers: IBM ILOG CPLEX and CP Optimizer 12.6.3
  - MDD added to CP Optimizer (Cire & v.H., 2013)
  - maximum width 1024
  - time limit: 30 minutes

- TD-TSP 38 instances from TSPLIB (n=14-107 jobs)

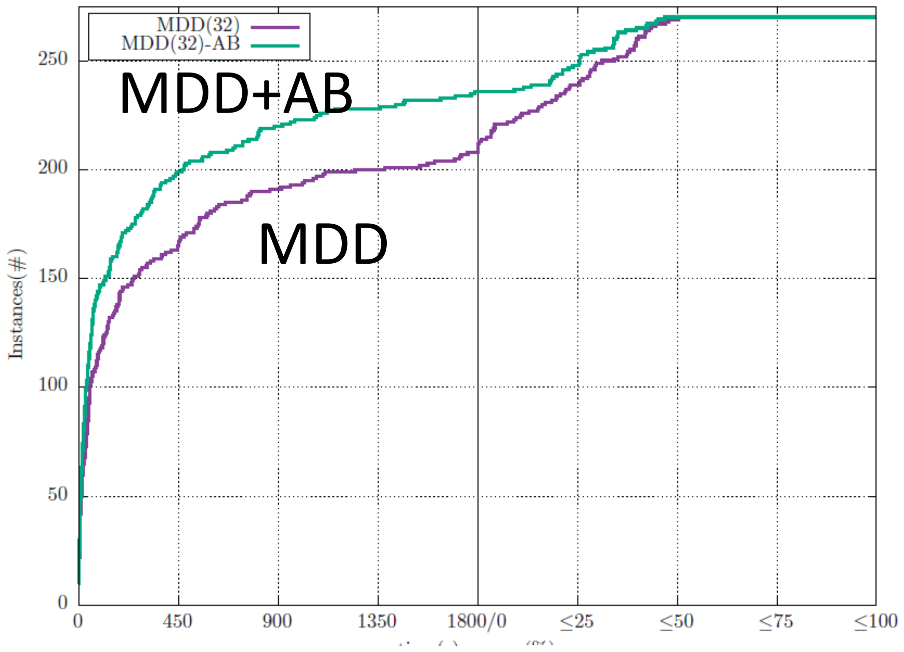
  $\delta^{t}_{i,j}$ = (n-t)*$\delta_{i,j}$ [Abeledo et al., 2013]
- TD-TSPTW based on Dumas et al. (n=30, 35, 40), 270 total
- TD-SOP 29 instances from SOP dataset in TSPLib (n=7 to 100)

TD-TSP: Performance Plot



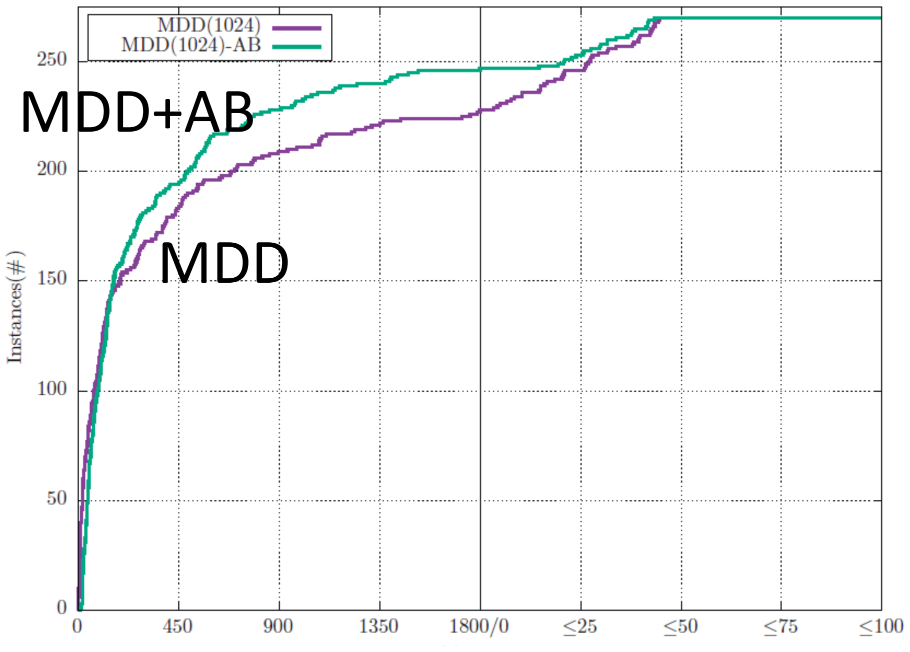
TD-TSPTW: Performance Plot

max width 32



Time (s) Gap (%)

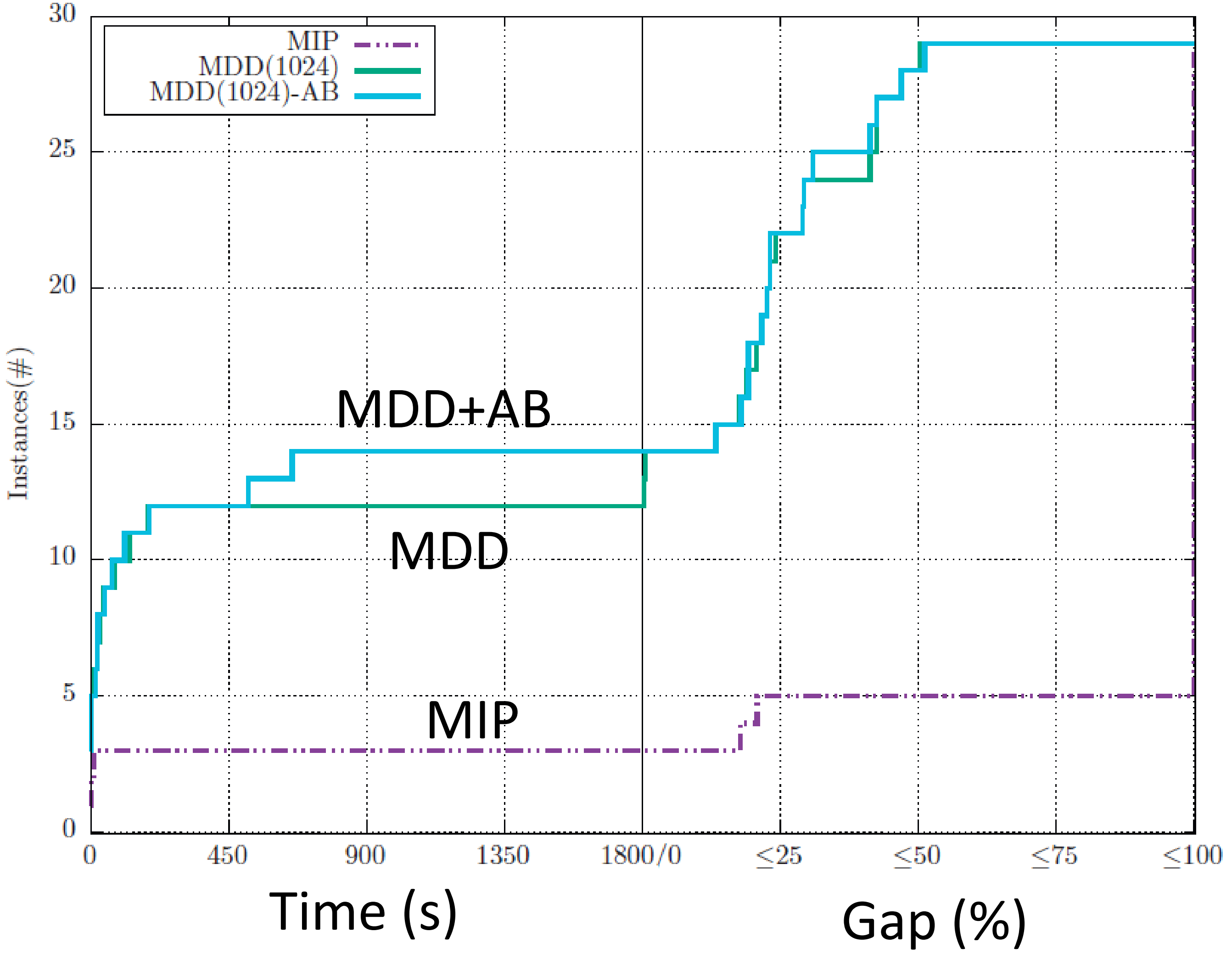
max width 1024



Time (s) Gap (%)

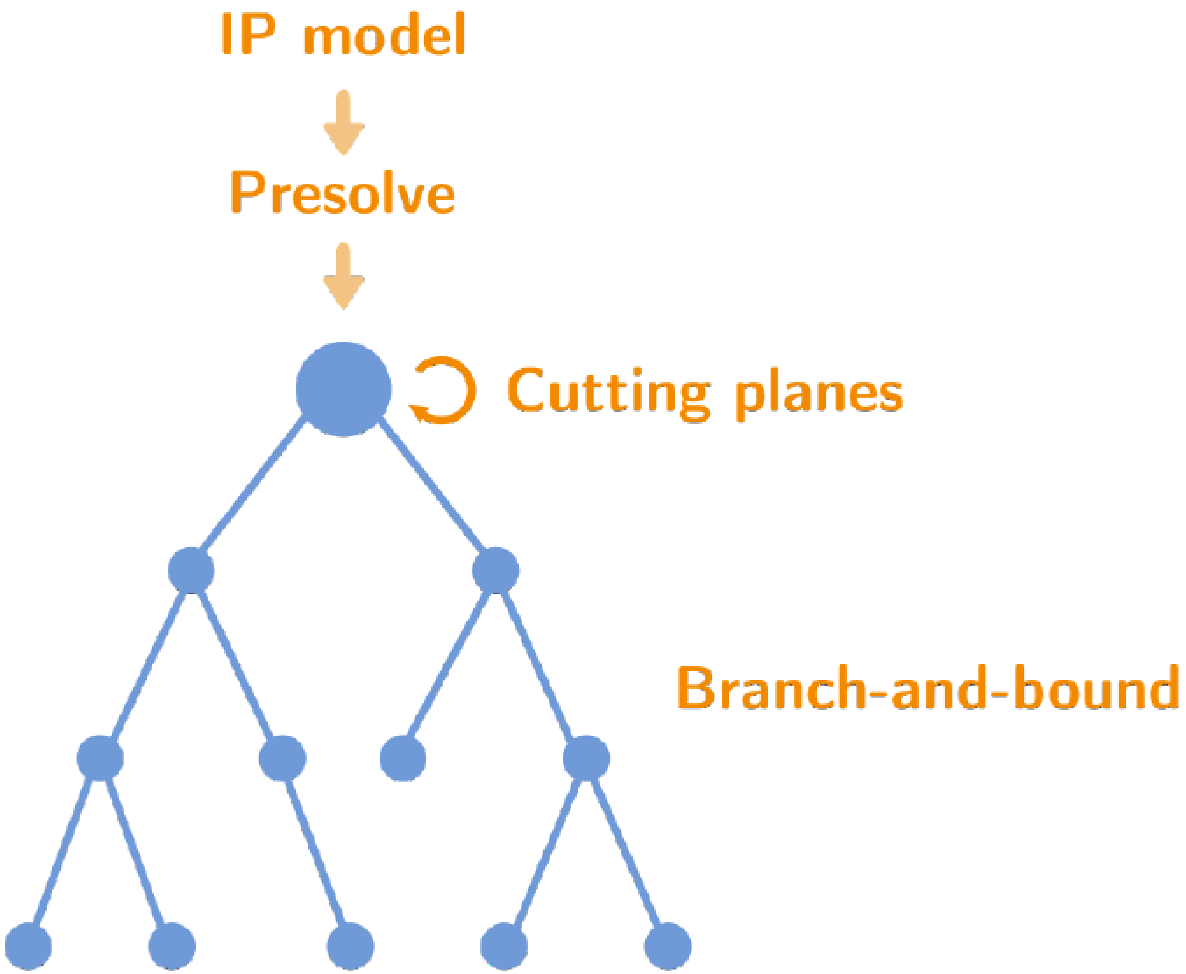
(MIP was unable to find any single integer solution)

TD-SOP: Performance Plot

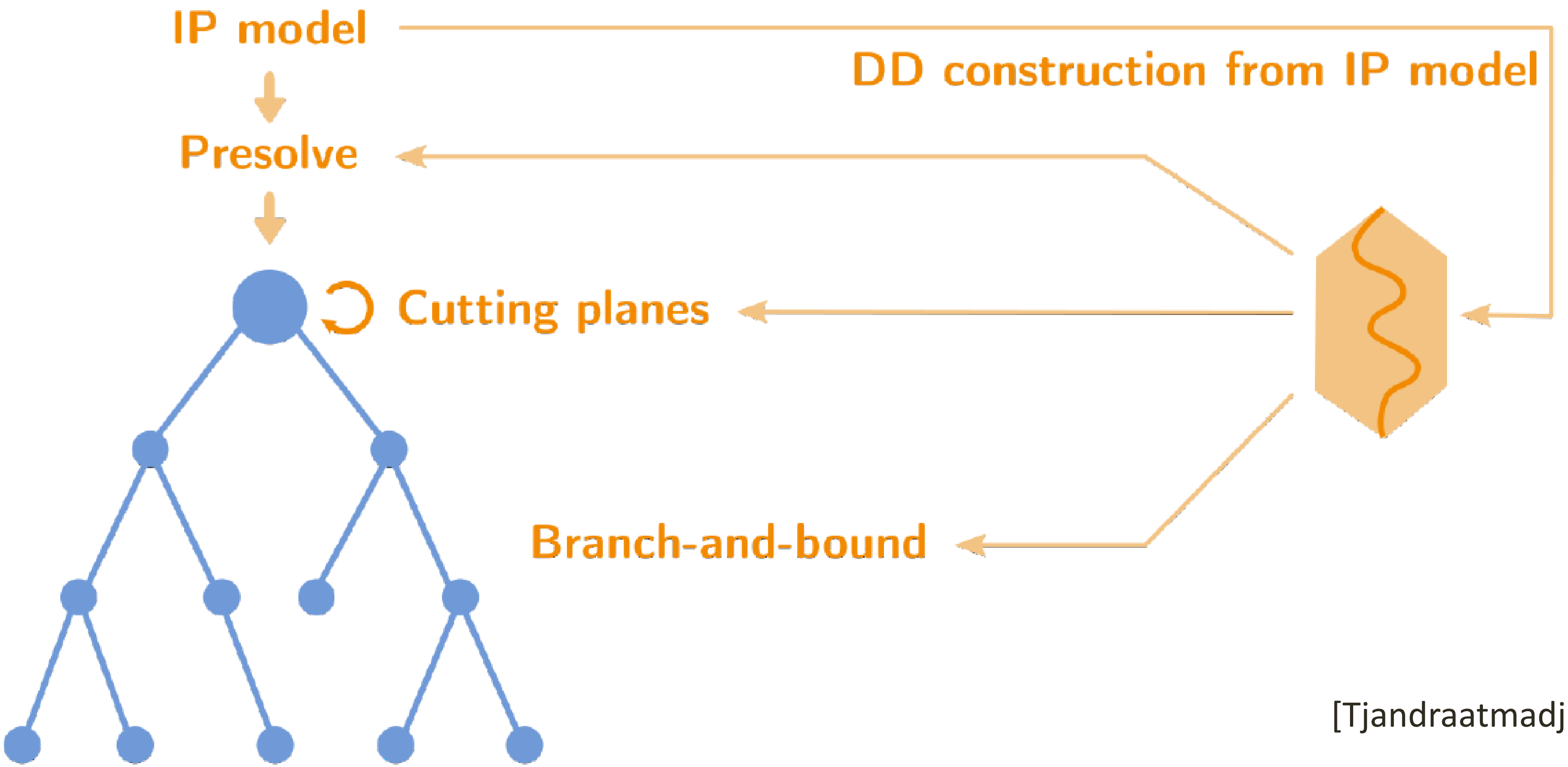


Integer Programming with Decision Diagrams

Motivation



Motivation

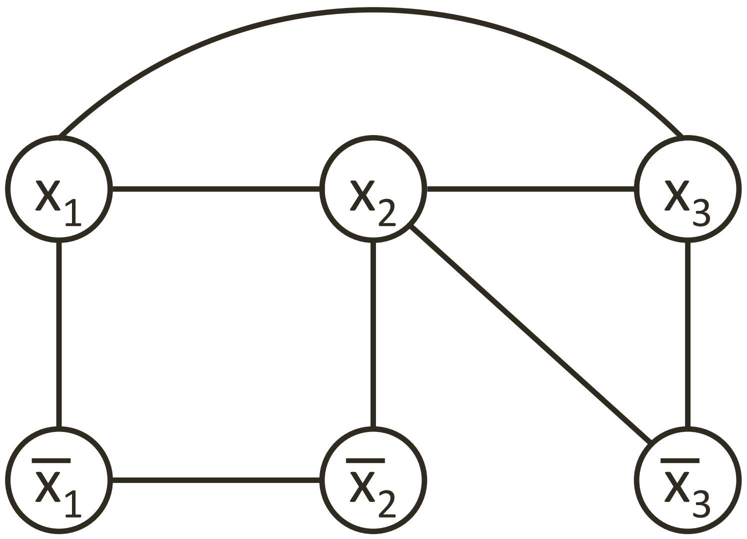


[Tjandraatmadja, PhD 2018]

Decision Diagram for IP Model?

- Option 1: use linear constraints to build DD
  - DD relaxation usually much weaker than LP bound
- Option 2: identify structure in model
  - set covering? set packing? independent set?
  - dedicated DD representing part of the model
- **Option 3:** use structure inferred by solver
  - conflict graph/clique table

Conflict Graph for Binary Problems



$$x_1 + x_2 + x_3 \leq 1$$

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

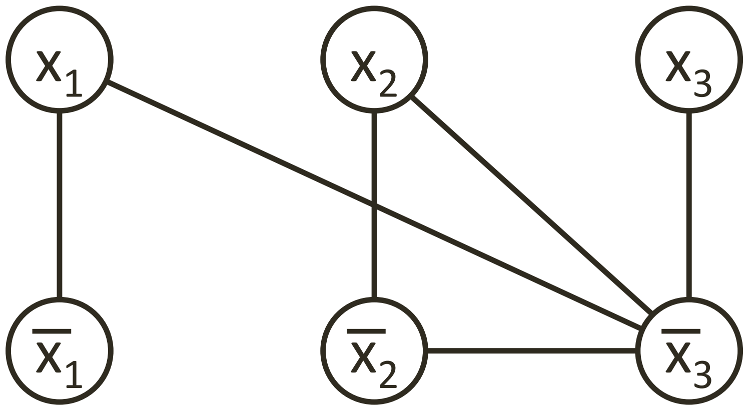
$$(1 - x_1) + (1 - x_2) \leq 1$$

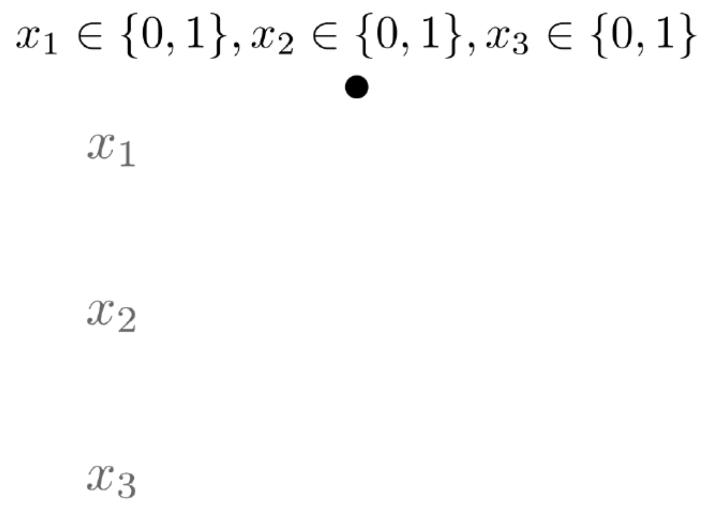
Conflict graphs are inferred and constructed by most modern MIP solvers

[Atamtürk et al., 2000; Achterberg, 2007]

Decision Diagram Compilation

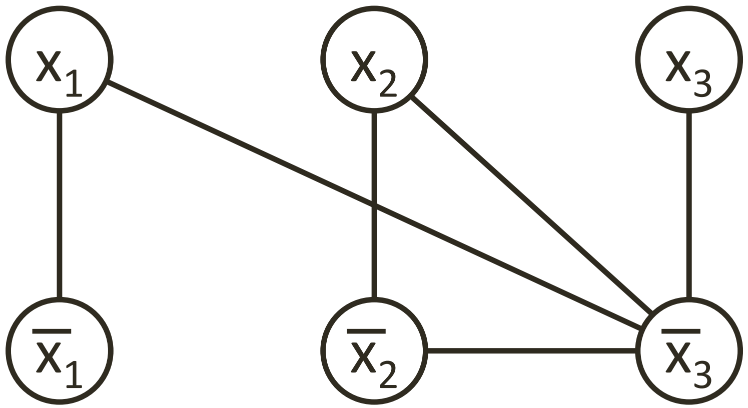
- State: variable domains
- Transition: propagate decision

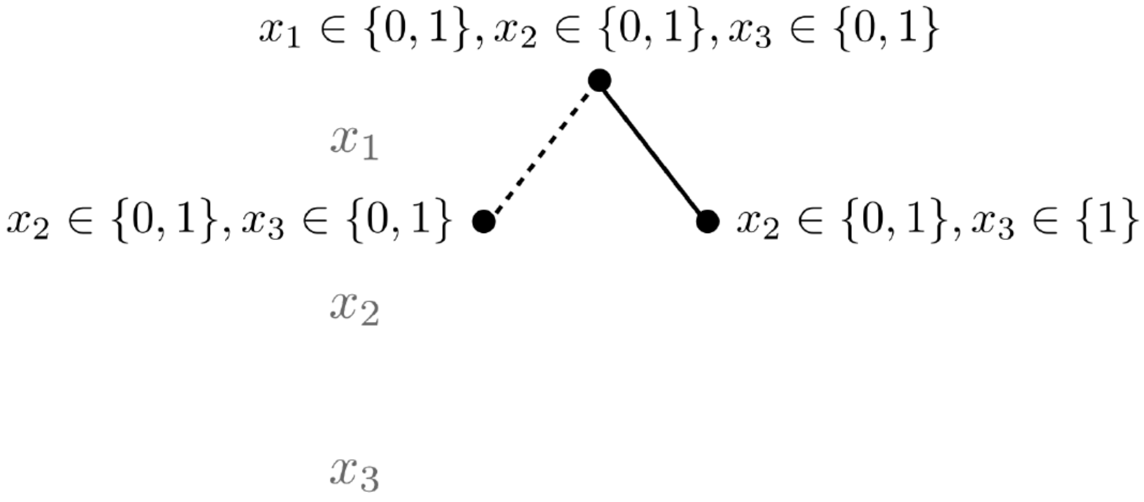




Decision Diagram Compilation

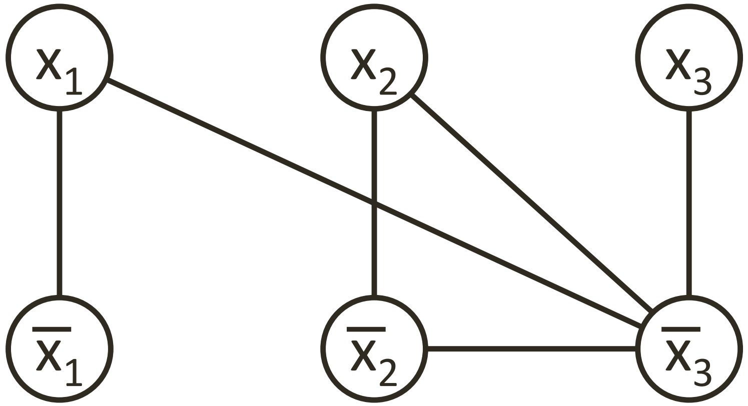
- State: variable domains
- Transition: propagate decision

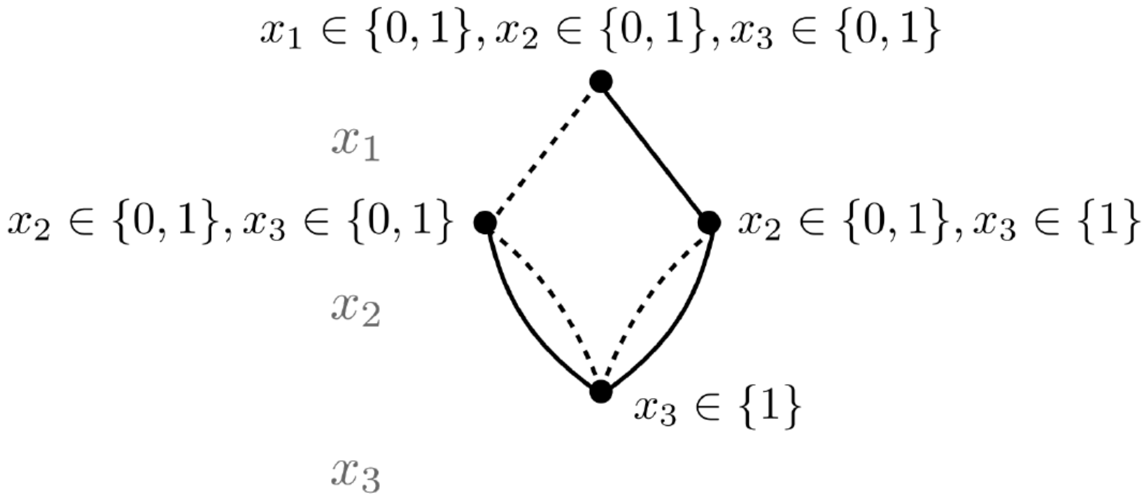




Decision Diagram Compilation

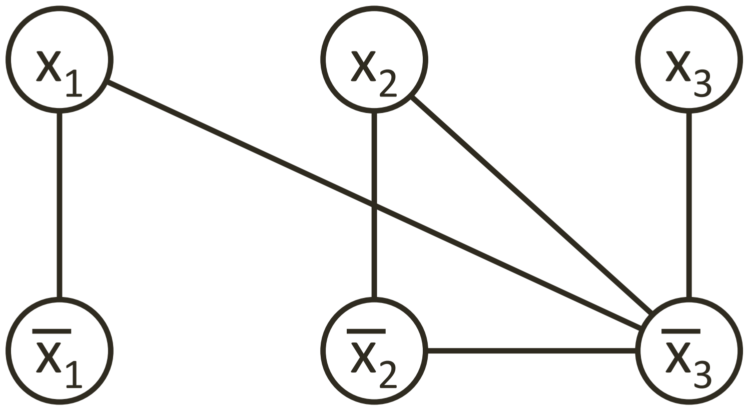
- State: variable domains
- Transition: propagate decision

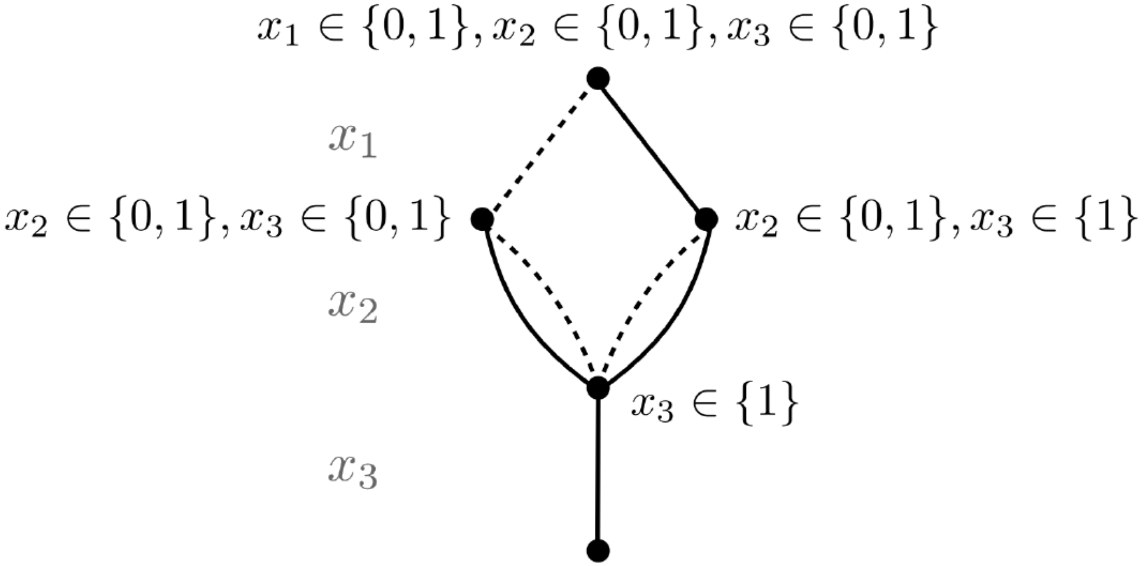




Decision Diagram Compilation

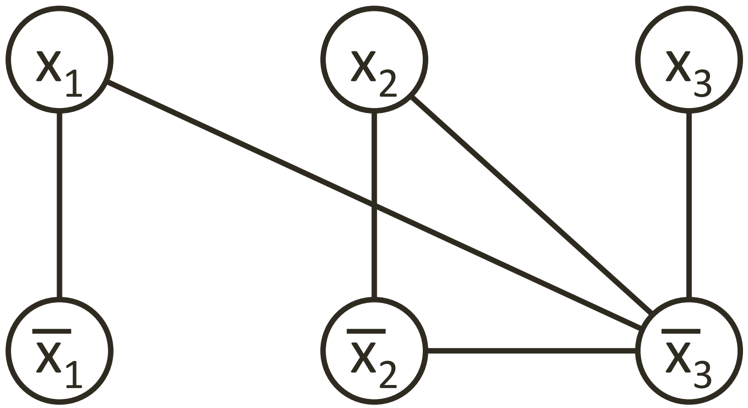
- State: variable domains
- Transition: propagate decision

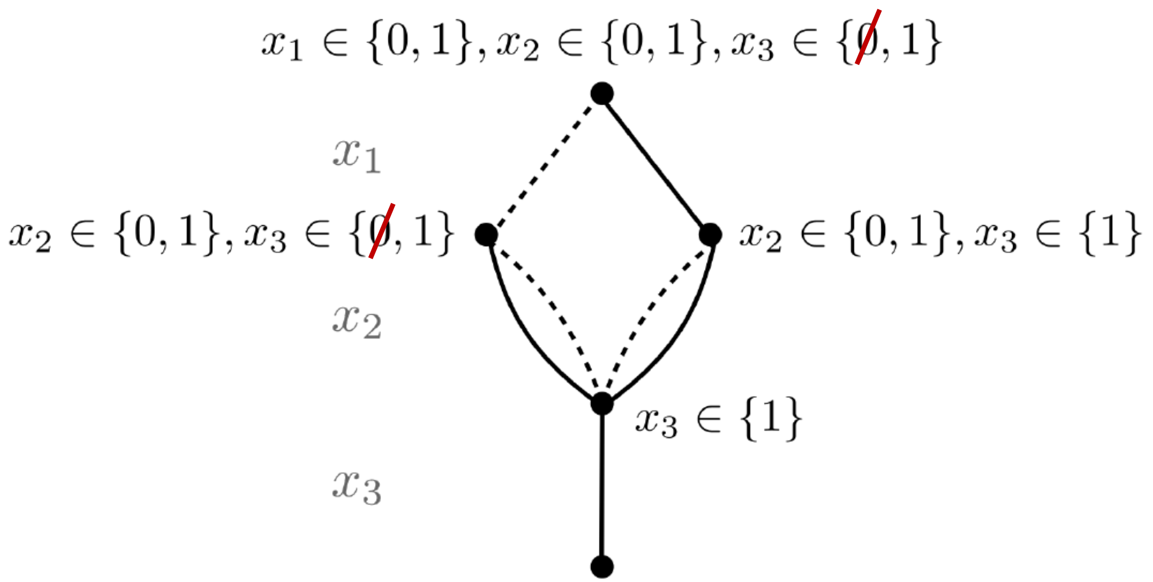




Decision Diagram Compilation

- State: variable domains
- Transition: propagate decision

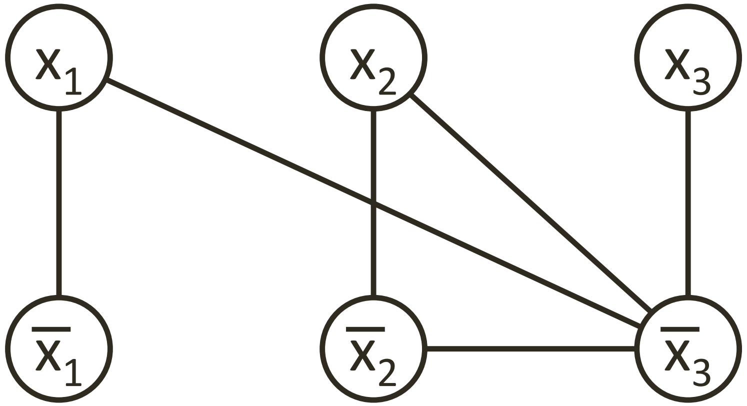


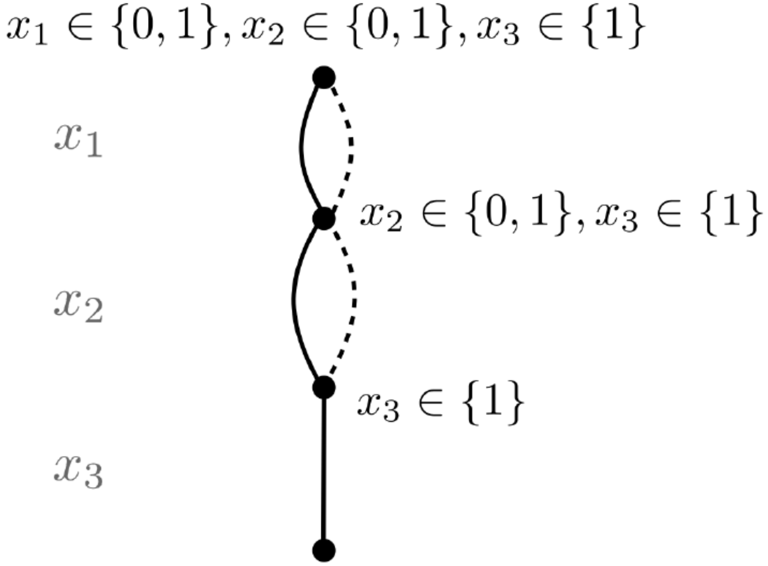


- Theorem: If root state is domain consistent, then this approach yields a reduced exact DD

Decision Diagram Compilation

- State: variable domains
- Transition: propagate decision





- Theorem: If root state is domain consistent, then this approach yields a reduced exact DD

Stronger DD relaxation via Lagrangian

Original IP model

$$\max\ c^\top x$$

$$Fx \le f \quad \leftarrow \text{Structured constraints for DD}$$

$$Ax \le b \quad \leftarrow \text{Any set of linear constraints}$$

$$x \in \mathbb{Z}^n,\ \ell \le x \le u$$

Lagrangian model

$$\min_{\lambda \ge 0} \max\ c^\top x + \lambda^\top (b - Ax)$$

$$Fx \le f$$

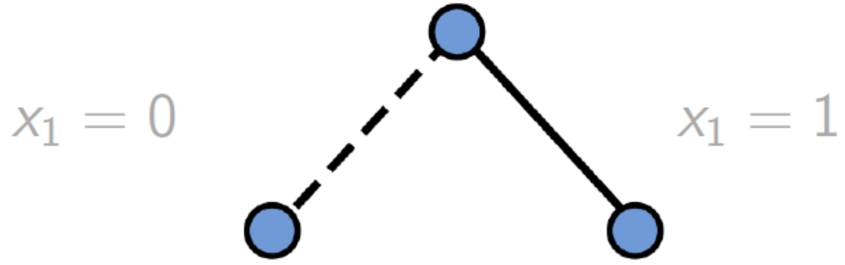
$$x \in \mathbb{Z}^n,\ \ell \le x \le u$$

Lagrangian subproblem is longest path in DD (efficient)

Stronger DD relaxation via Propagation

- Propagate linear constraints
- Additional state information
  - – variable domains
  - – constraint right-hand sides

$3x_1 + x_2 + 2x_3 \leq \mathbf{4}$
$x_1 \in \{0,1\}, x_2 \in \{0,1\}, x_3 \in \{0,1\}$

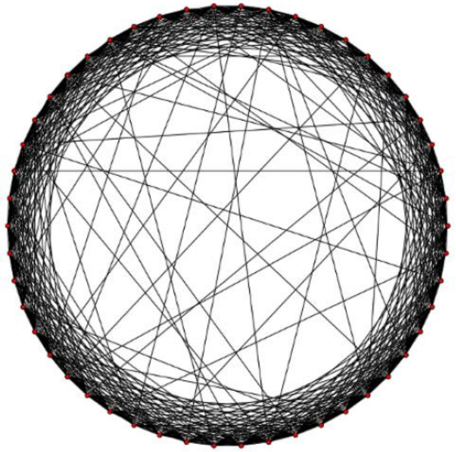


$x_2 + 2x_3 \leq \mathbf{4}$
$x_2 \in \{0,1\}, x_3 \in \{0,1\}$

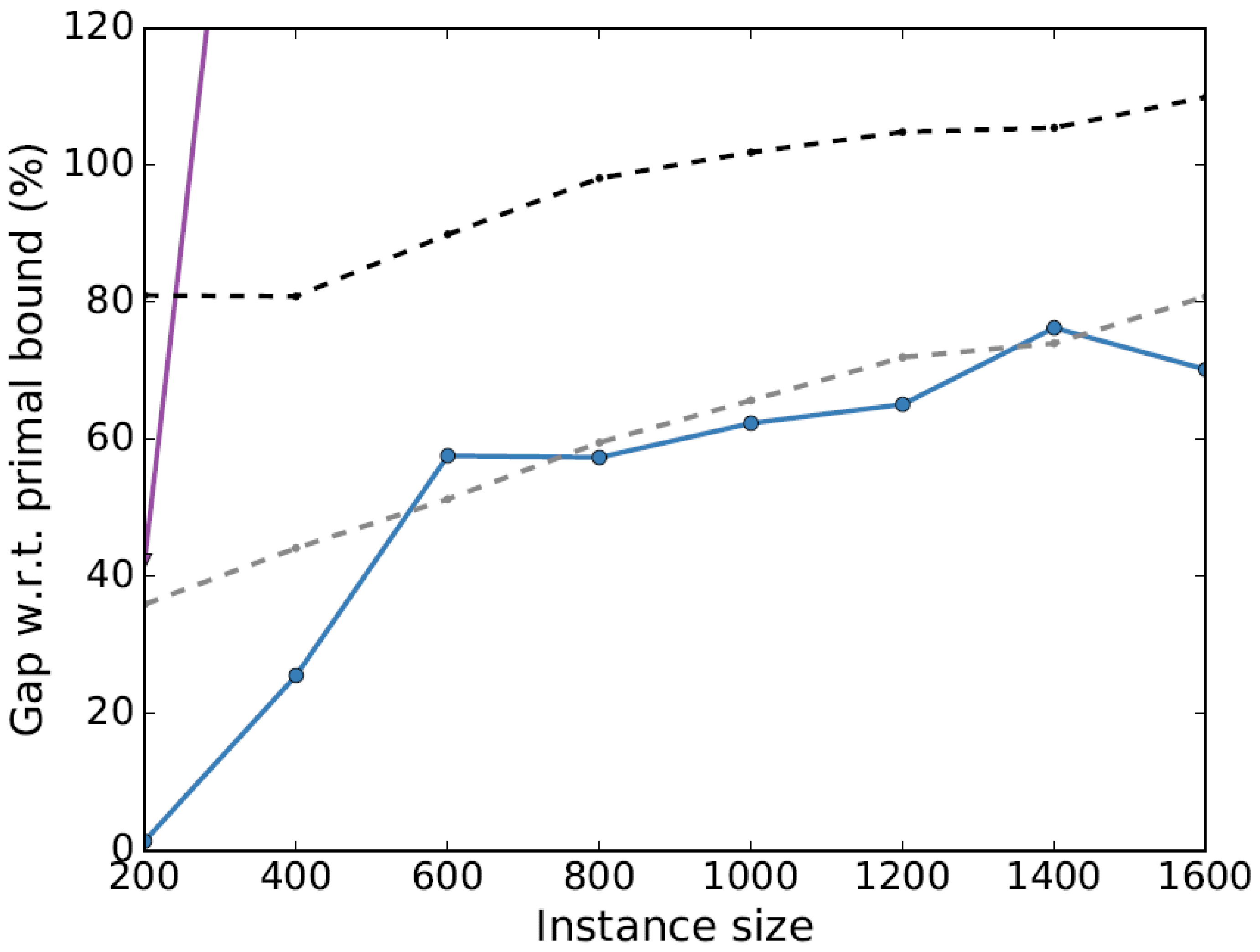
$x_2 + 2x_3 \leq \mathbf{1}$
$x_2 \in \{0,1\}, \mathbf{x_3 \in \{0\}}$

Quality of Bound

- Experimental setup
  - Independent set problem on random graphs (Watts-Strogatz)
  - Add set of random knapsack constraints $\sum_{i \in S} a_i x_i \leq b$
  - Vary number of variables n
  - Vary number of knapsack constraints m
- Implemented in SCIP 5.0.1
  - Only IP model is given to solver
  - DD compiled automatically

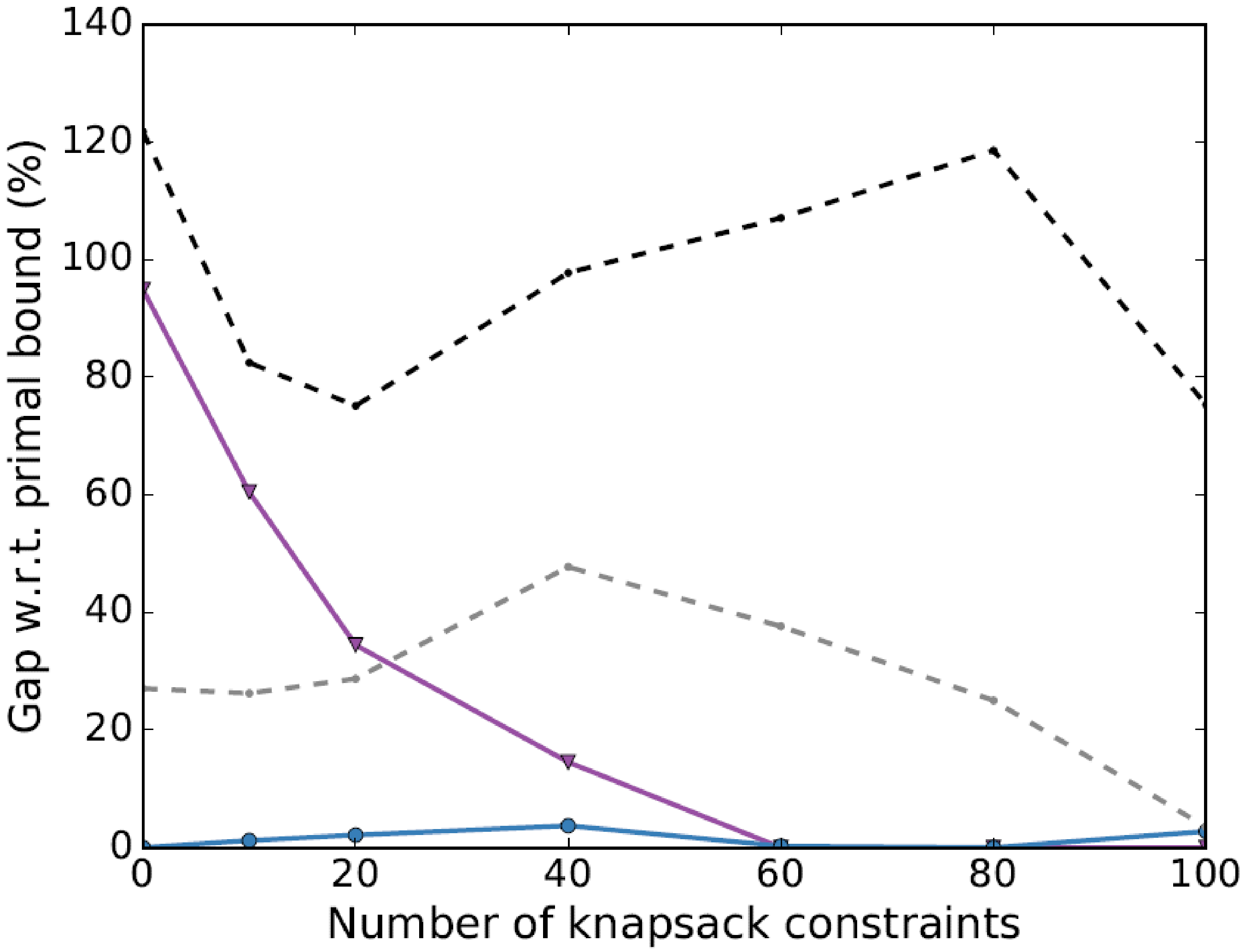


Varying Number of Variables

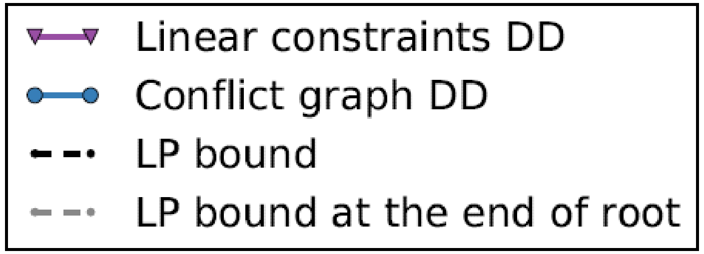


number of knapsack constraints: m = 0.1n

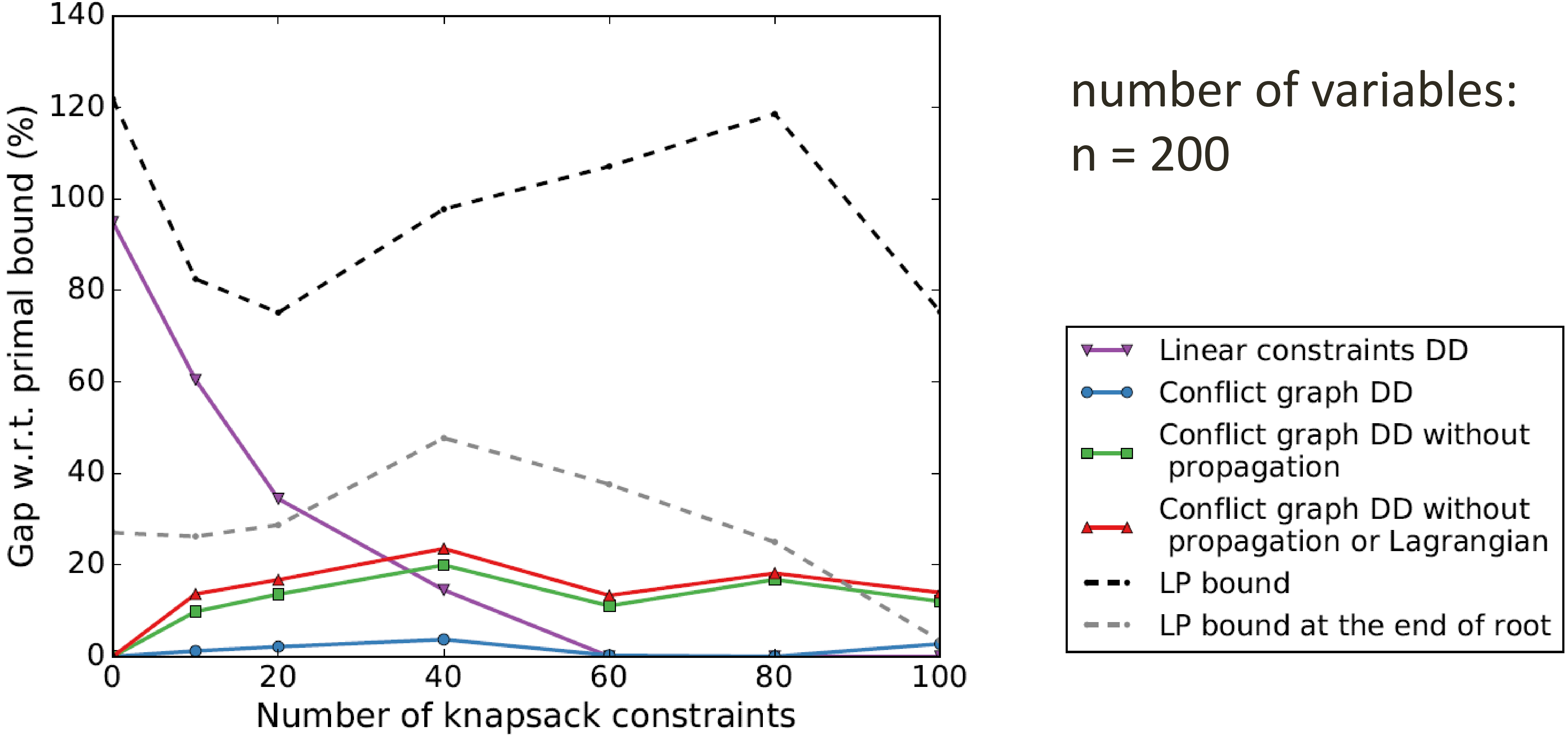
Varying Number of Knapsack Constraints



number of variables:
n = 200

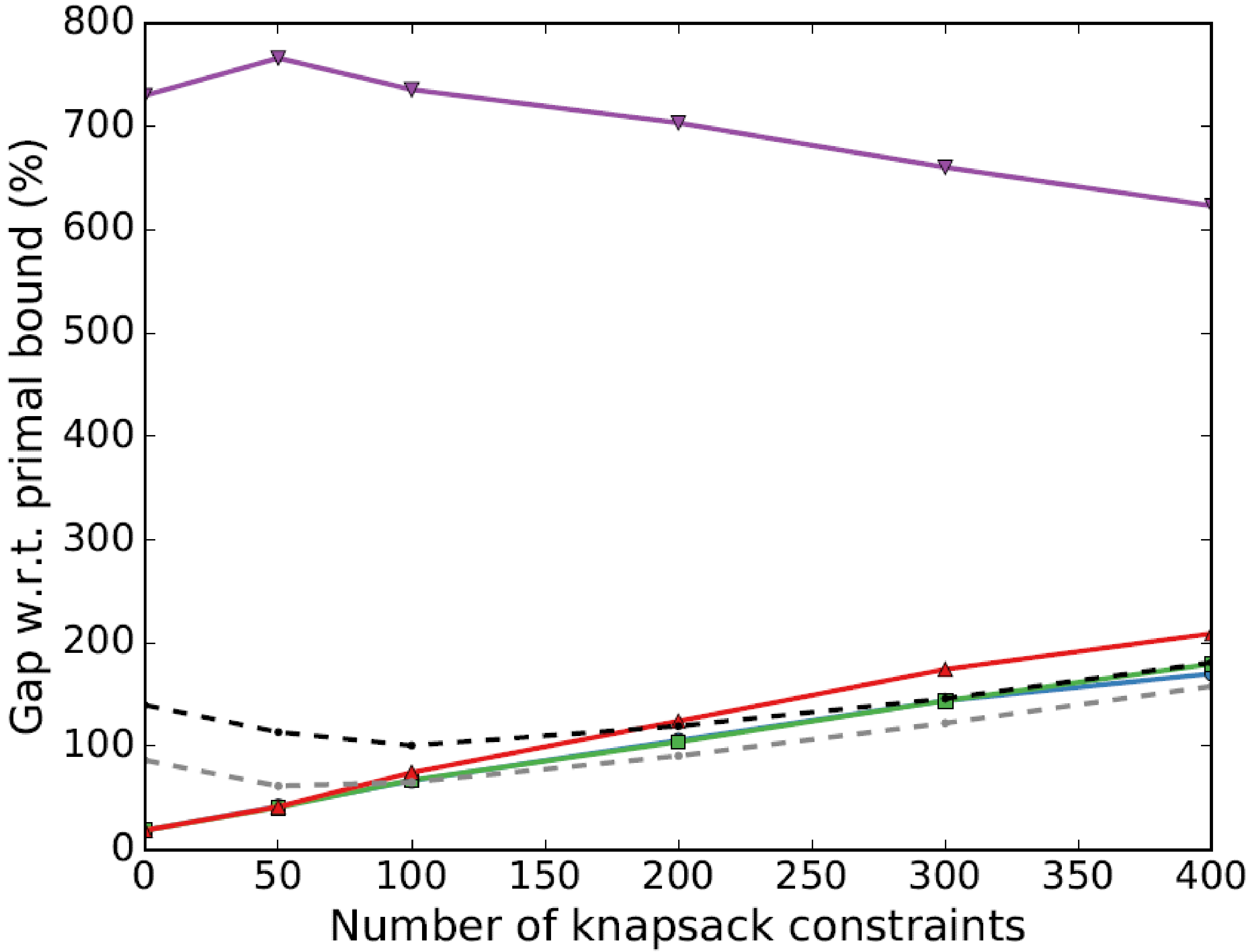


Varying Number of Knapsack Constraints

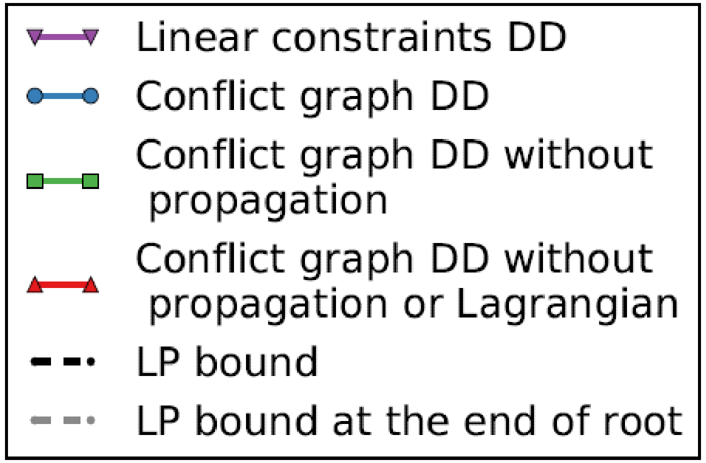


number of variables:
n = 200

Varying Number of Knapsack Constraints



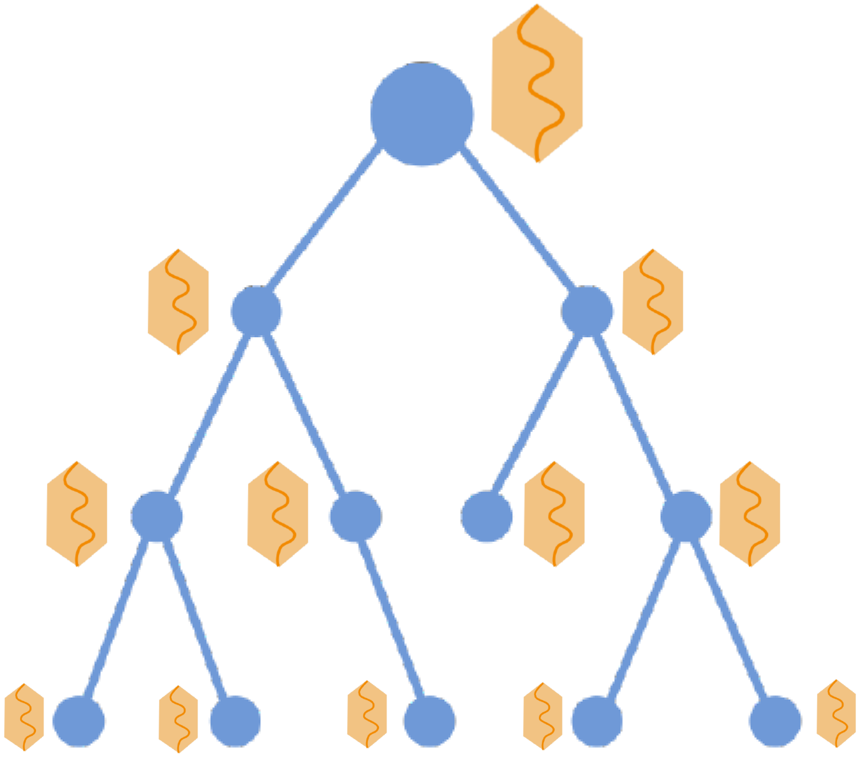
number of variables:
n = 1000



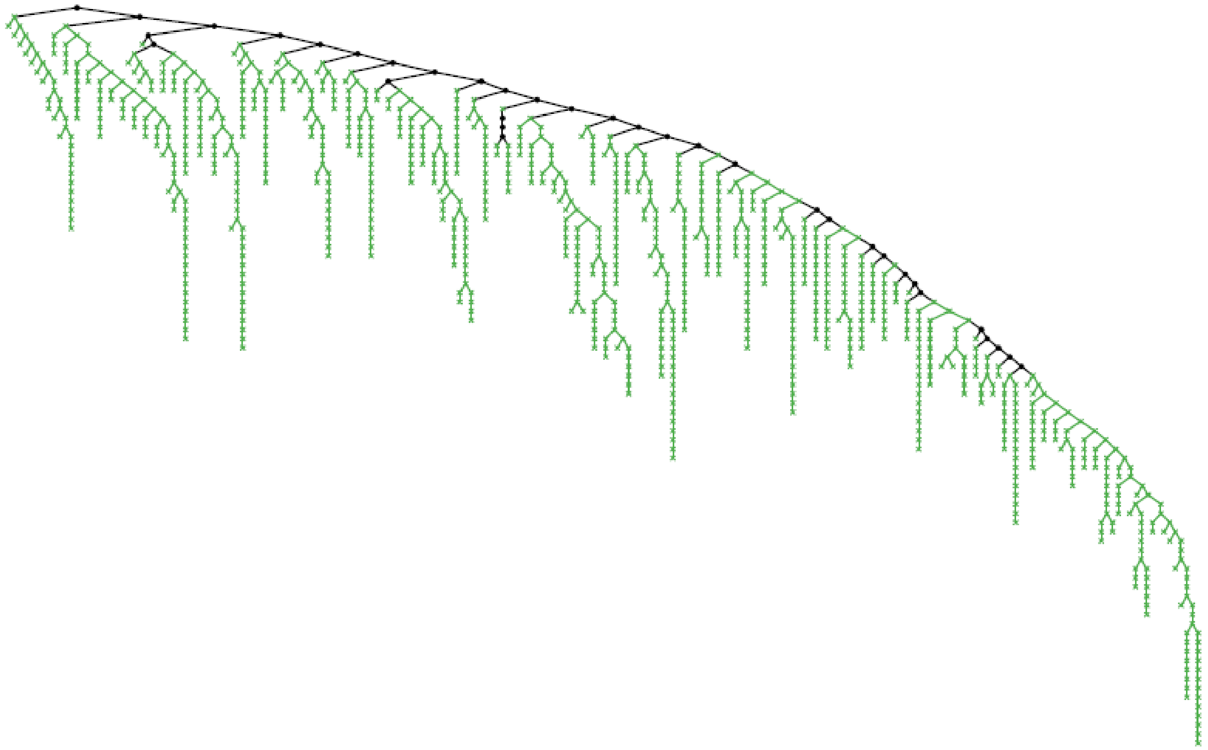
Integrate DDs into IP Branch and Bound

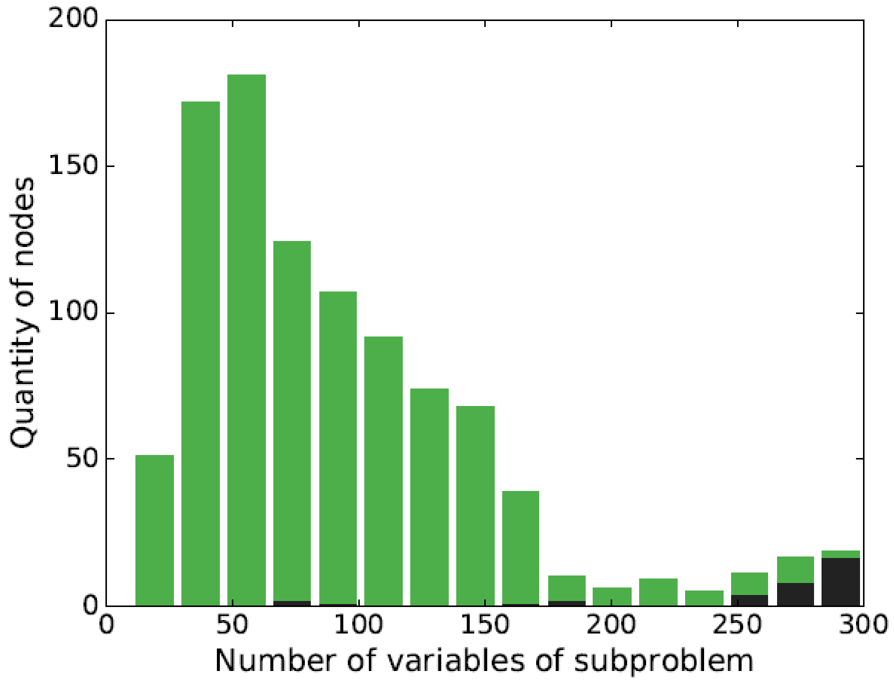
Carnegie Mellon Tepper SCHOOL OF BUSINESS

- Ingredients
  - Dual+Primal bounds from DDs
  - DD compilation based on conflict graph, Lagrangian, and propagation
  - Use MIP primal bound to remove sub-optimal DD arcs



When to apply Decision Diagrams?

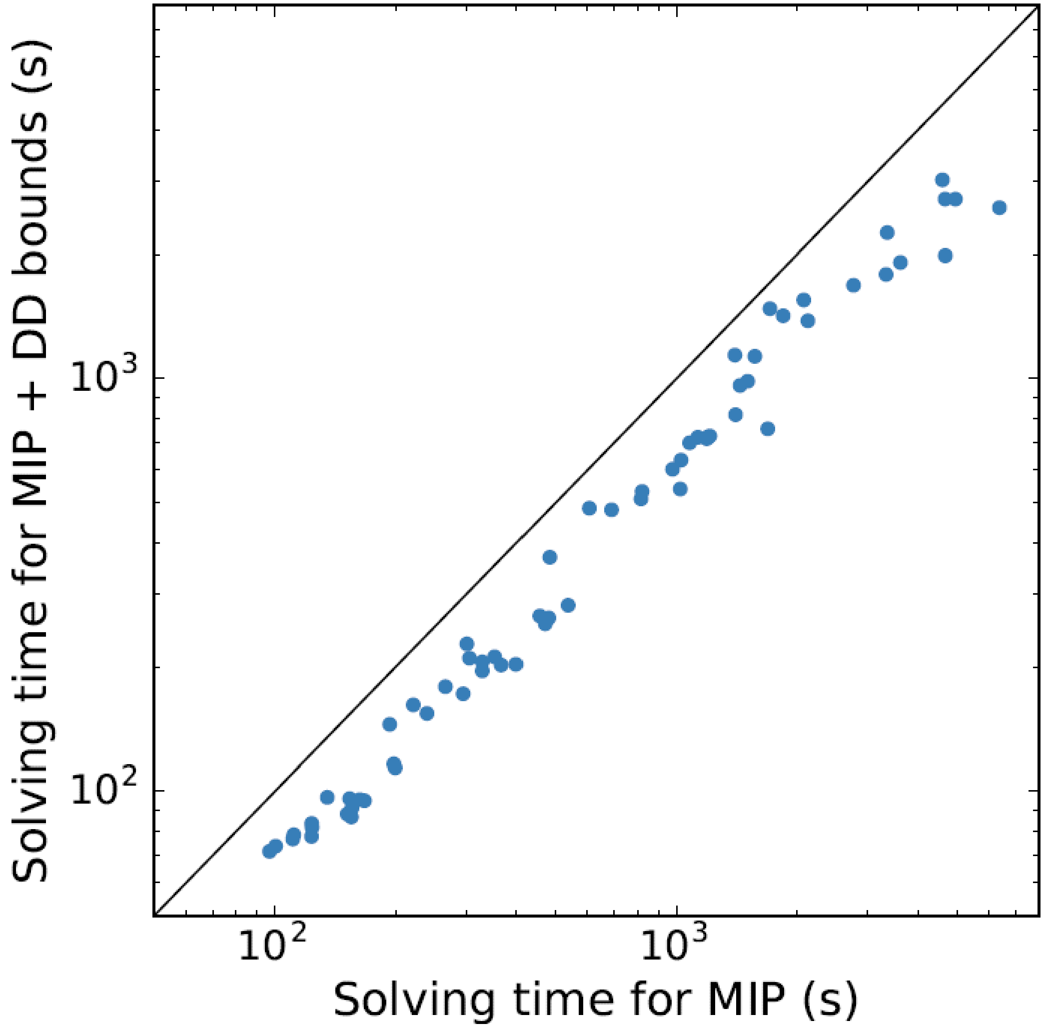




Smaller subproblems are most effective; up to 100~200 variables

– for experiments we used 100 variable threshold, and max width 100

Random Graphs + Knapsack Constraints

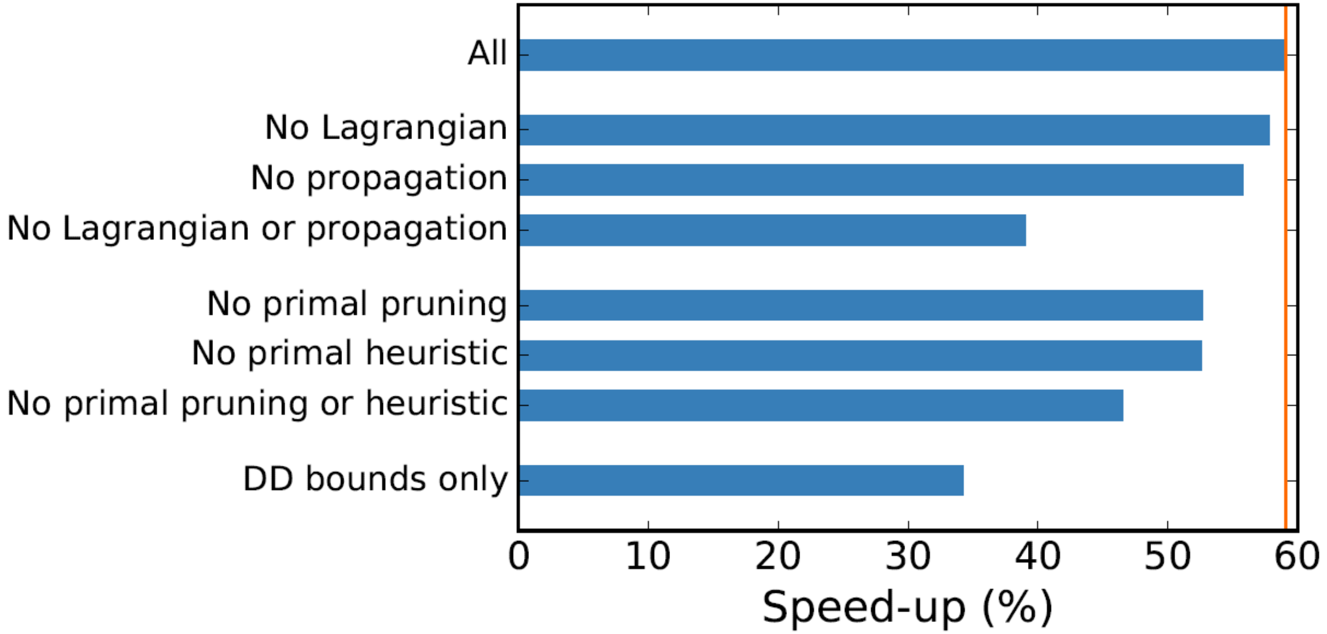


n = 300, 350, 400, 450
m = 0.1n

On average: 65.5% node reduction
1.59x speedup

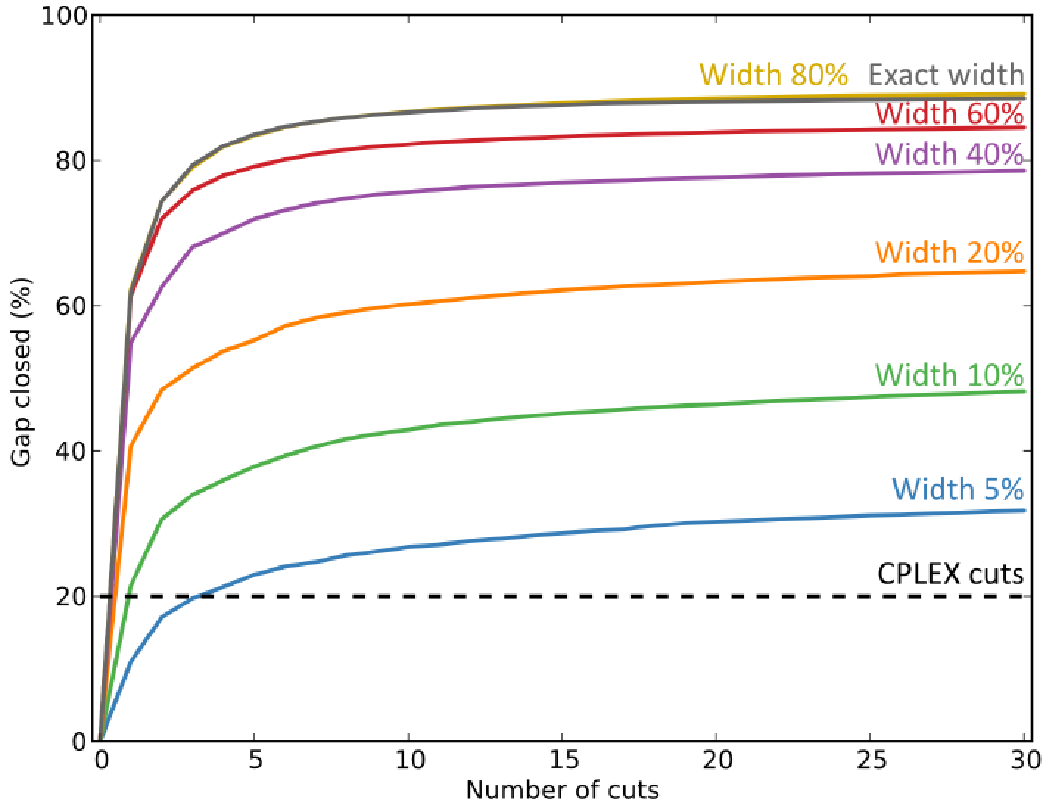
More detailed results

n	300	350	400	450
Average speed-up (%)	57.33	62.90	60.14	60.17
Average node reduction (%)	73.93	67.44	63.63	57.75

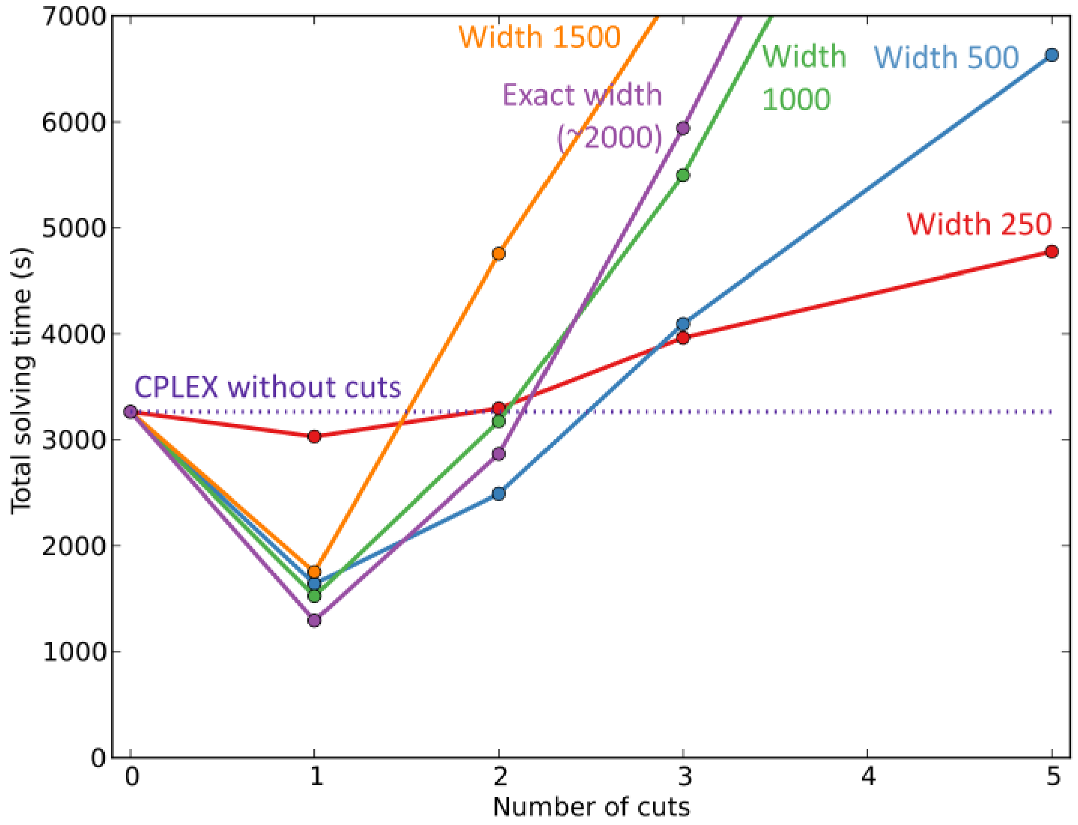


More IP Integration: Cut Generation with DDs

Gap closed for instances with 80% density and 300 vertices (truncated at 30 cuts)



Solving time for instances with 80% density and 600 vertices



[Tjandraatmadja & vH, IJOC to appear]

Summary

- Discrete Optimization with Decision Diagrams
  - new generic solving methodology
  - outperforms integer programming on several classical problems
- Constraint Programming with Decision Diagrams
  - state of the art for sequencing with side constraints
  - closed several open instances from TSPLIB
- Integer Programming with Decision Diagrams
  - generic methodology can improve IP solver with factor 1.59