Decision Diagrams for Constraint Programming

Part 3

Willem-Jan van Hoeve

Tepper School of Business
Carnegie Mellon University
www.andrew.cmu.edu/user/vanhoeve/mdd/

Plan

What can MDDs do for Combinatorial Optimization?

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

MDDs for Constraint Programming and Scheduling

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

MDDs for Discrete Optimization

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

Many Opportunities: integrated methods, theory, applications,...

MDDs for Discrete Optimization

References

- Bergman, v.H, and Hooker. Manipulating MDD Relaxations for Combinatorial Optimization. In *Proceedings of CPAIOR*, LNCS 6697, pp. 20-35. Springer, 2011.
- Bergman, Cire, v.H,, and Hooker. Optimization Bounds from Binary Decision Diagrams. *INFORMS Journal on Computing* 26(2): 253-258, 2014.
- Bergman, Cire, v.H., and Yunes. BDD-Based Heuristics for Binary Optimization. *Journal of Heuristics*, 20(2): 211-234, 2014.
- Bergman, Cire, v.H., and Hooker. Discrete Optimization with Decision Diagrams. *INFORMS Journal on Computing*, to appear.
- Bergman, Cire, Sabharwal, Samulowitz, Saraswat, and v.H. Parallel Combinatorial Optimization with Decision Diagrams. In *Proceedings of CPAIOR*, LNCS 8451, pp. 351-367. Springer, 2014.

See http://www.andrew.cmu.edu/user/vanhoeve/mdd/

Motivation

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

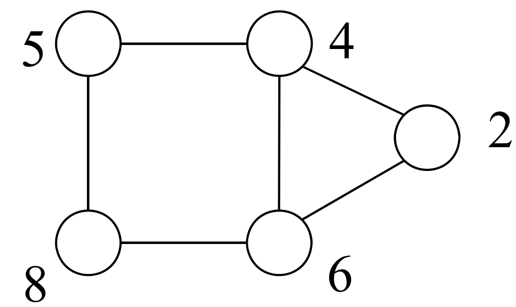
Case Study: Independent Set Problem

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

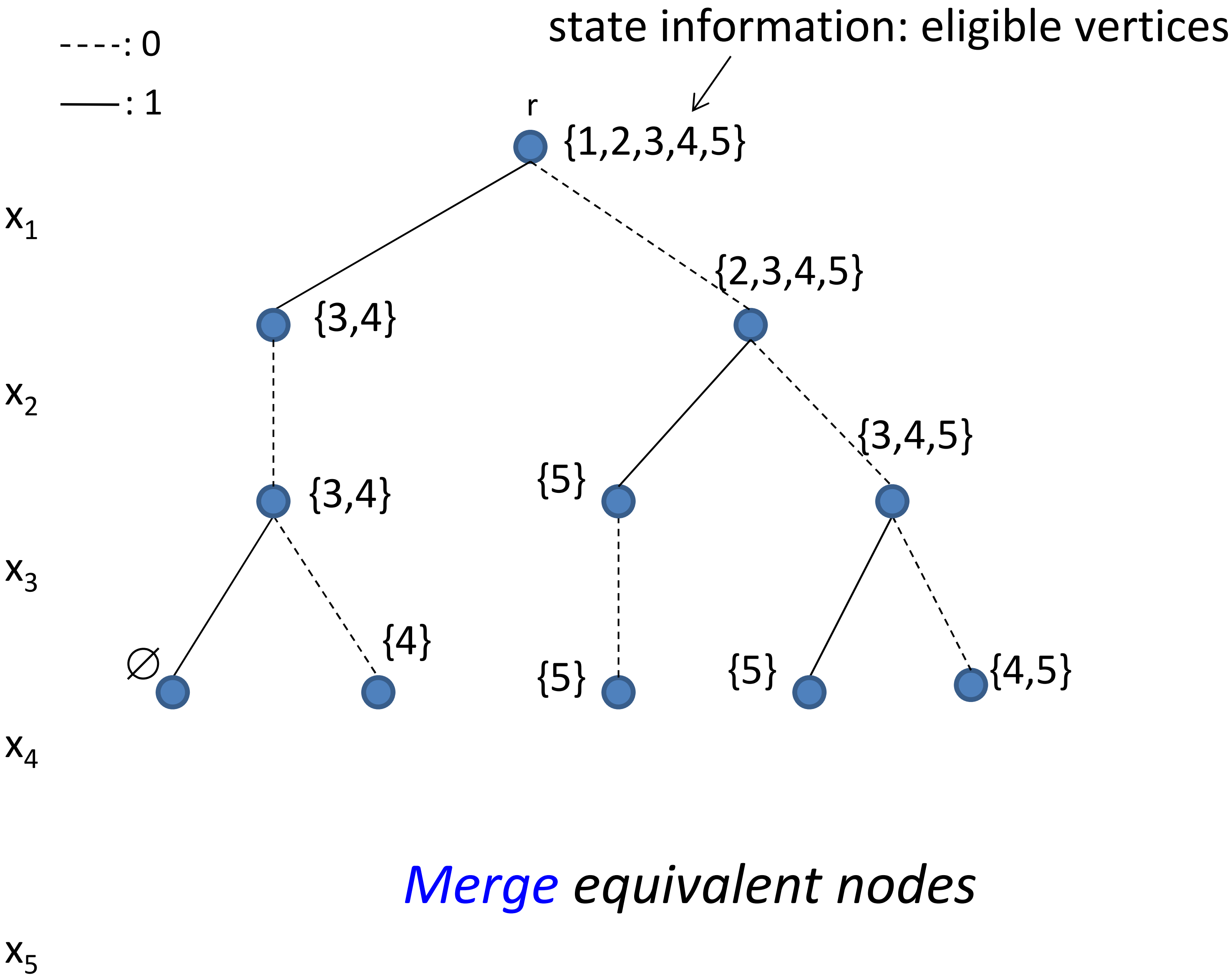
$$\max \quad \sum_i w_i x_i$$

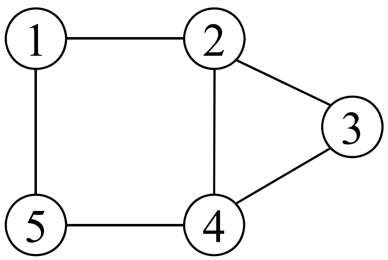
$$\text{s.t.} \quad x_i + x_j \leq 1 \quad \text{for all } (i,j) \text{ in } E$$

$$x_i \text{ binary} \quad \text{for all } i \text{ in } V$$



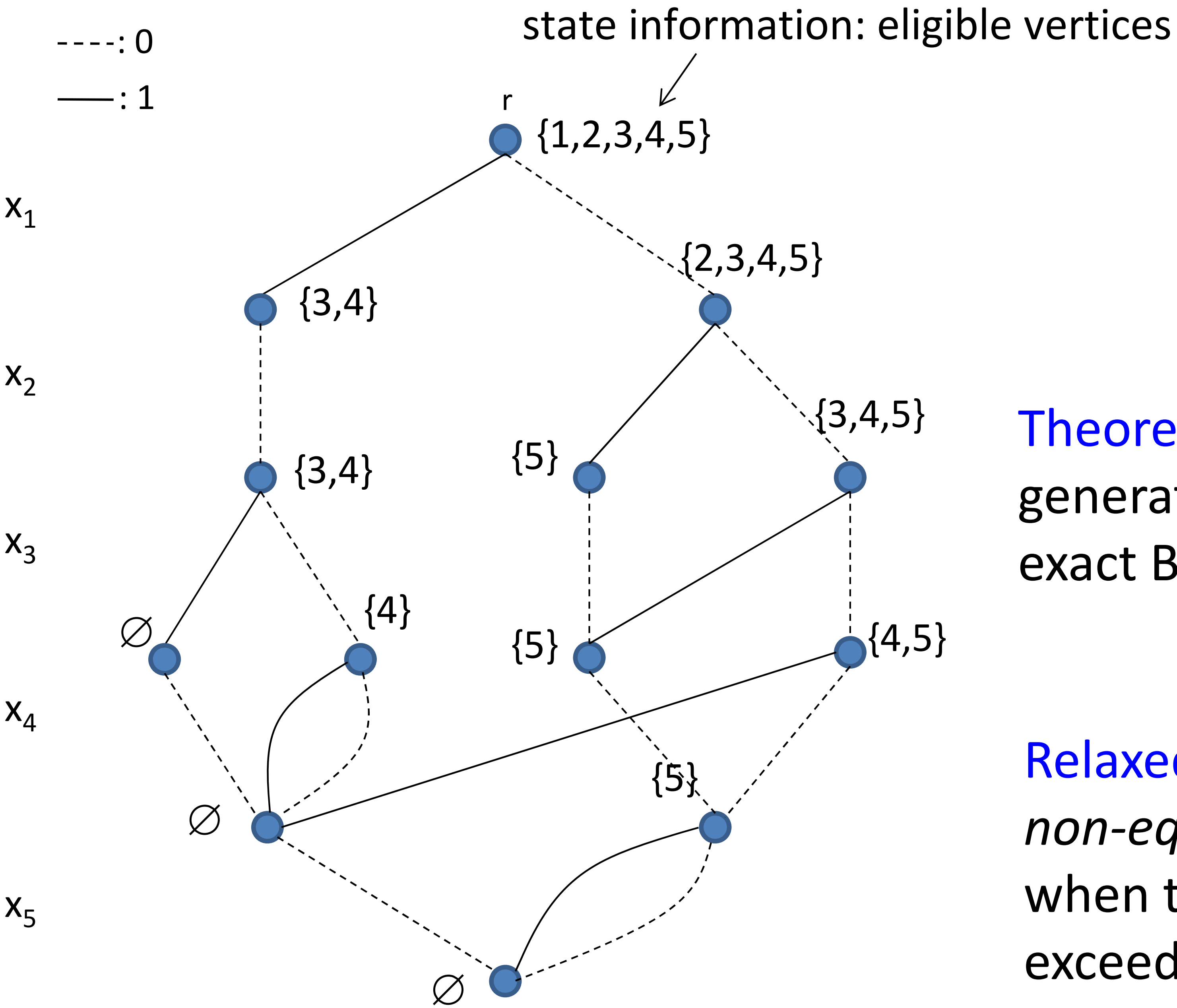
Exact top-down compilation

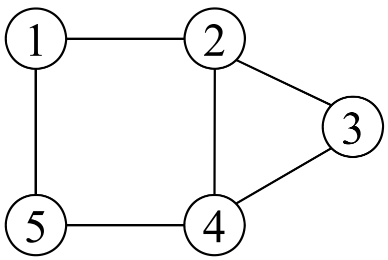




Merge equivalent nodes

Node Merging





Theorem: This procedure generates a reduced exact BDD

[Bergman et al., 2012]

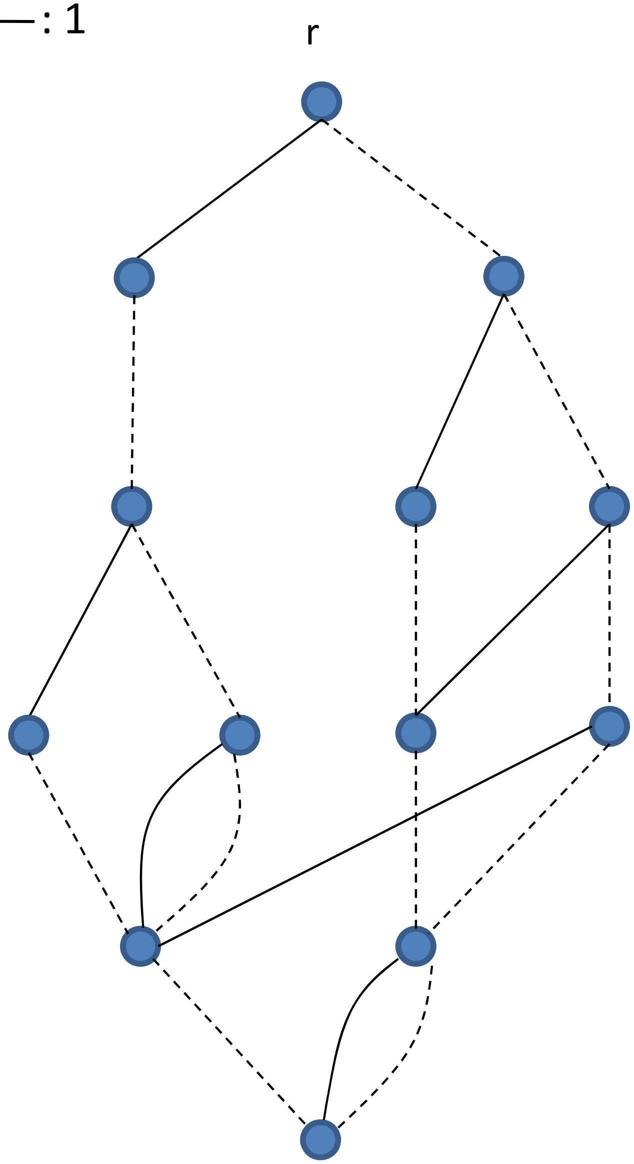
Relaxed BDD: merge *non-equivalent* nodes when the given width is exceeded

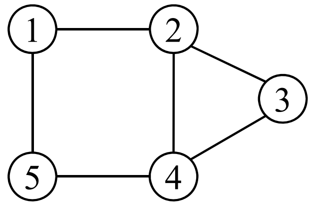
Relaxed BDD

- - - - : 0

——— : 1

Exact BDD





x_1

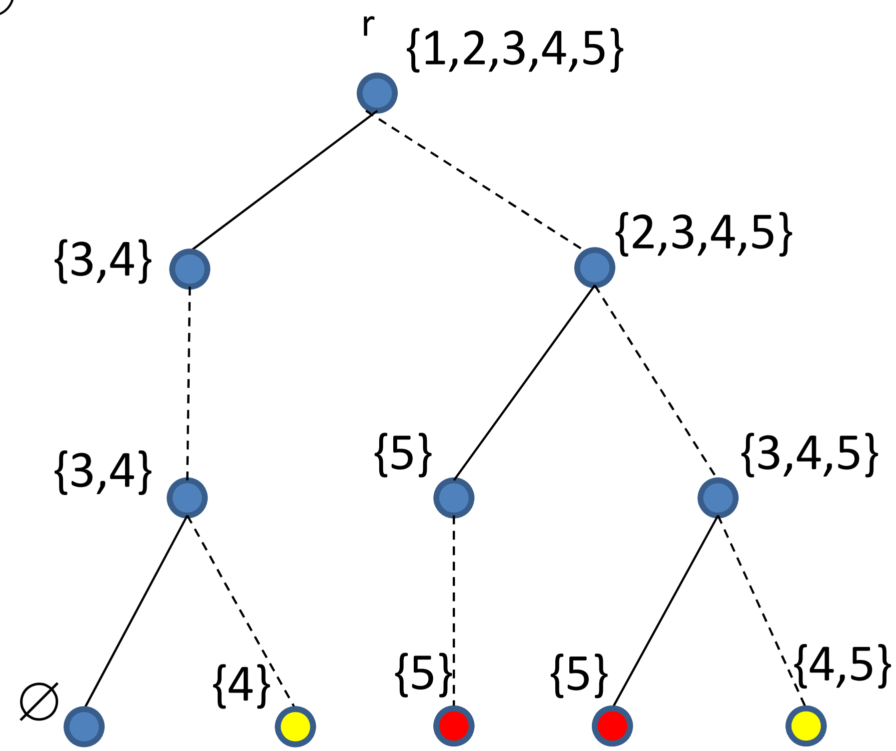
x_2

x_3

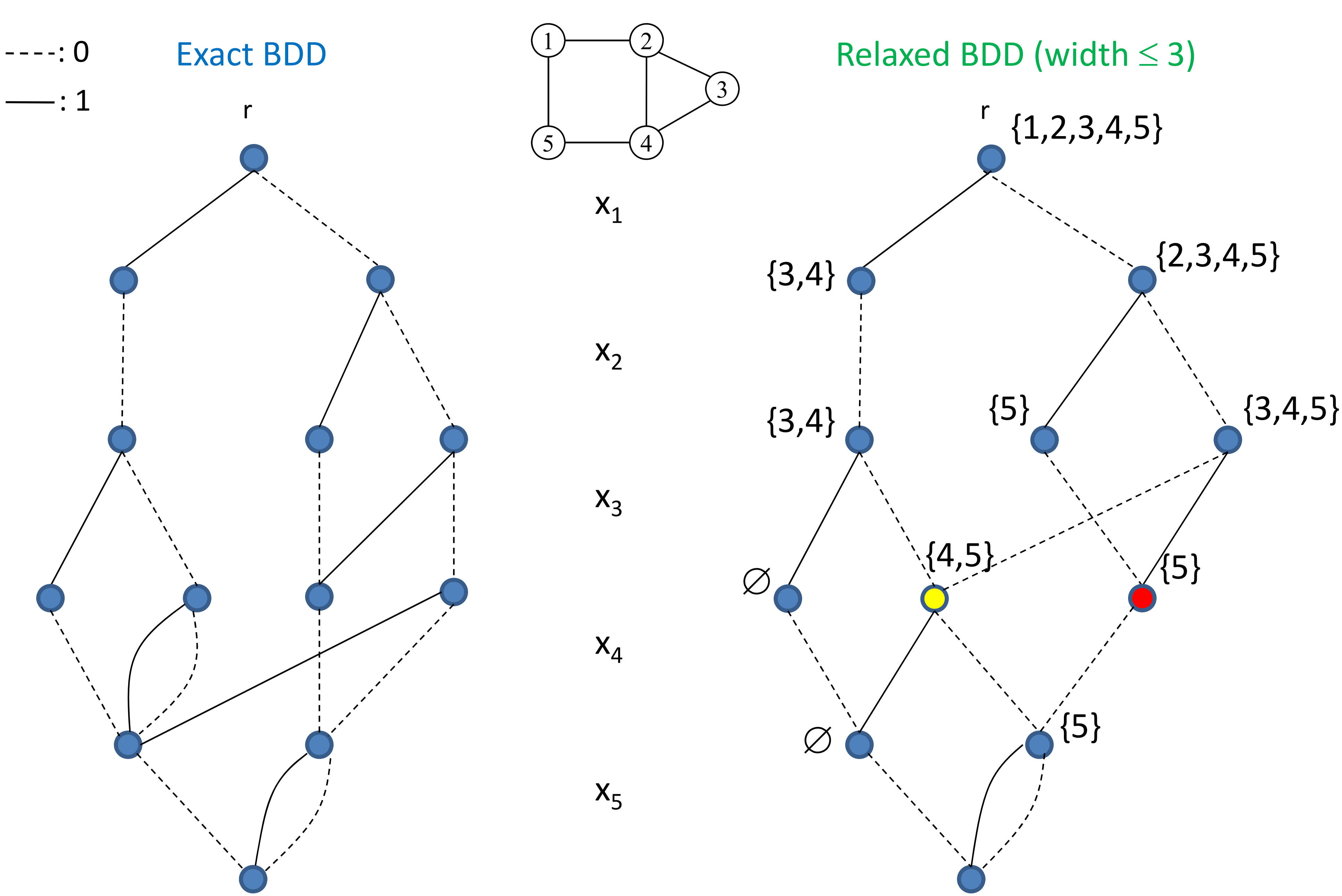
x_4

x_5

Relaxed BDD (width $\leq$ 3)

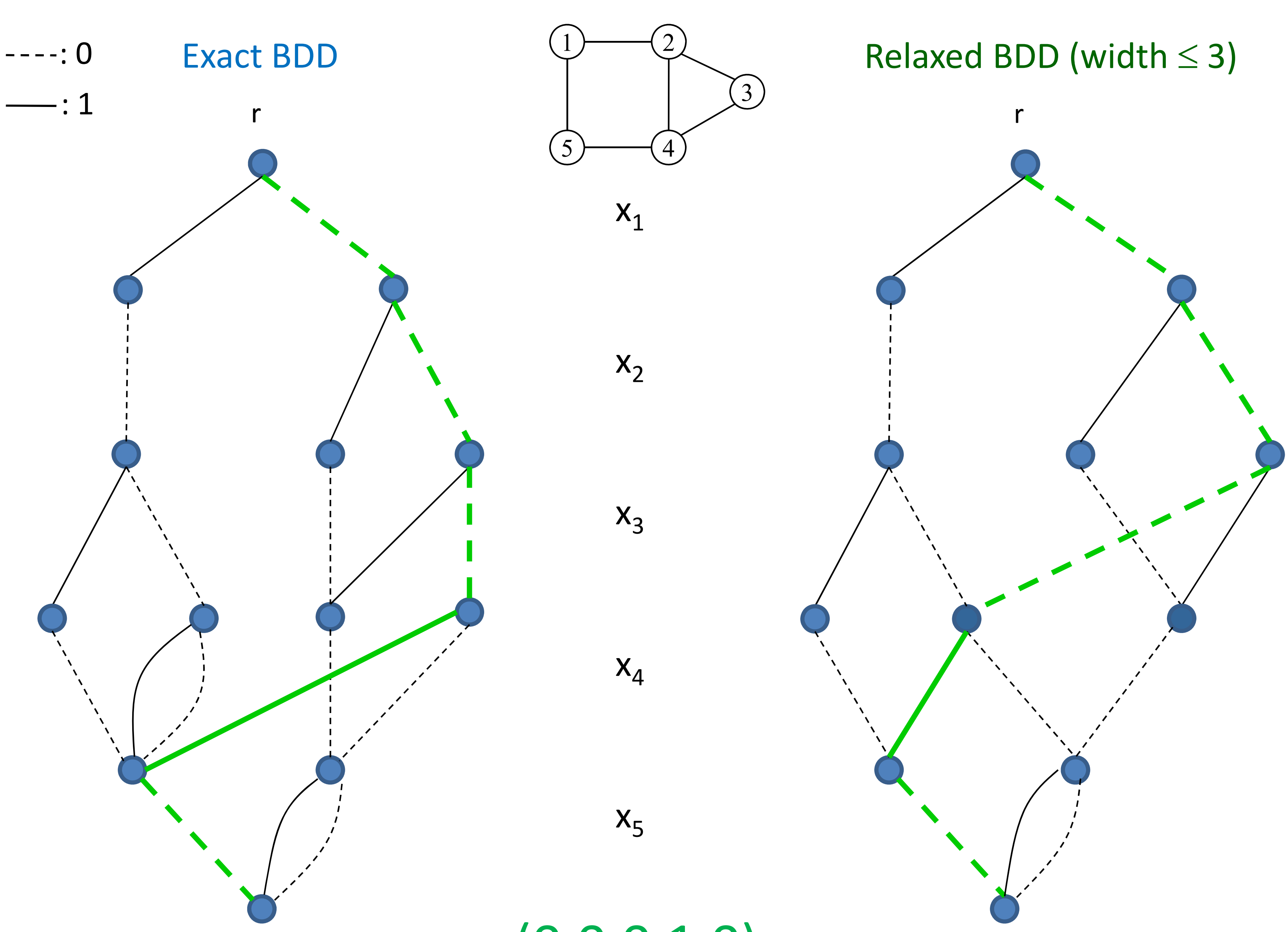


Relaxed BDD



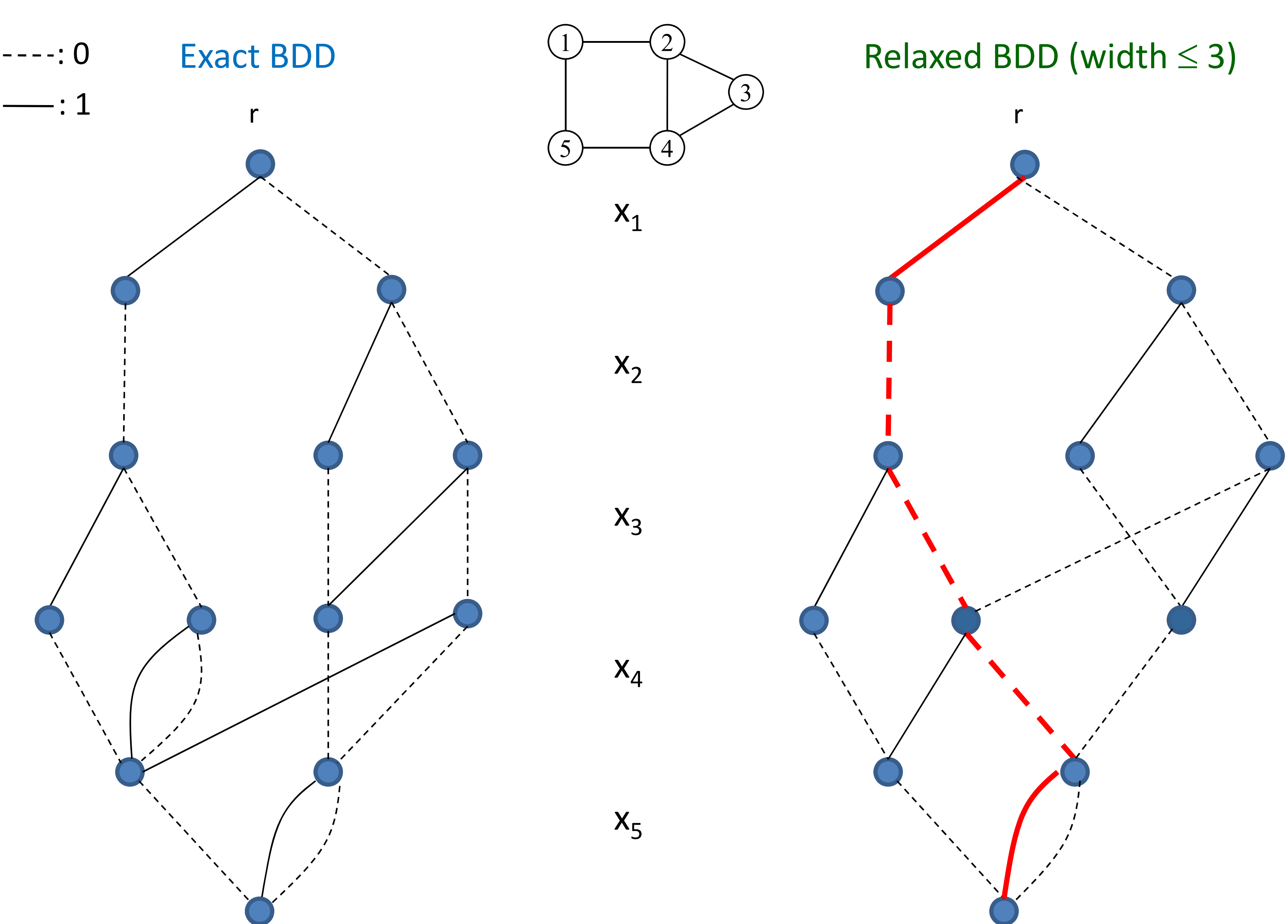
----: 0
——: 1
Exact BDD
r
1
2
3
4
5
x1
x2
x3
x4
x5
Relaxed BDD (width ≤ 3)
r
{1,2,3,4,5}
{3,4}
{2,3,4,5}
{3,4}
{5}
{3,4,5}
∅
{4,5}
{5}
∅
{5}

Relaxed BDD



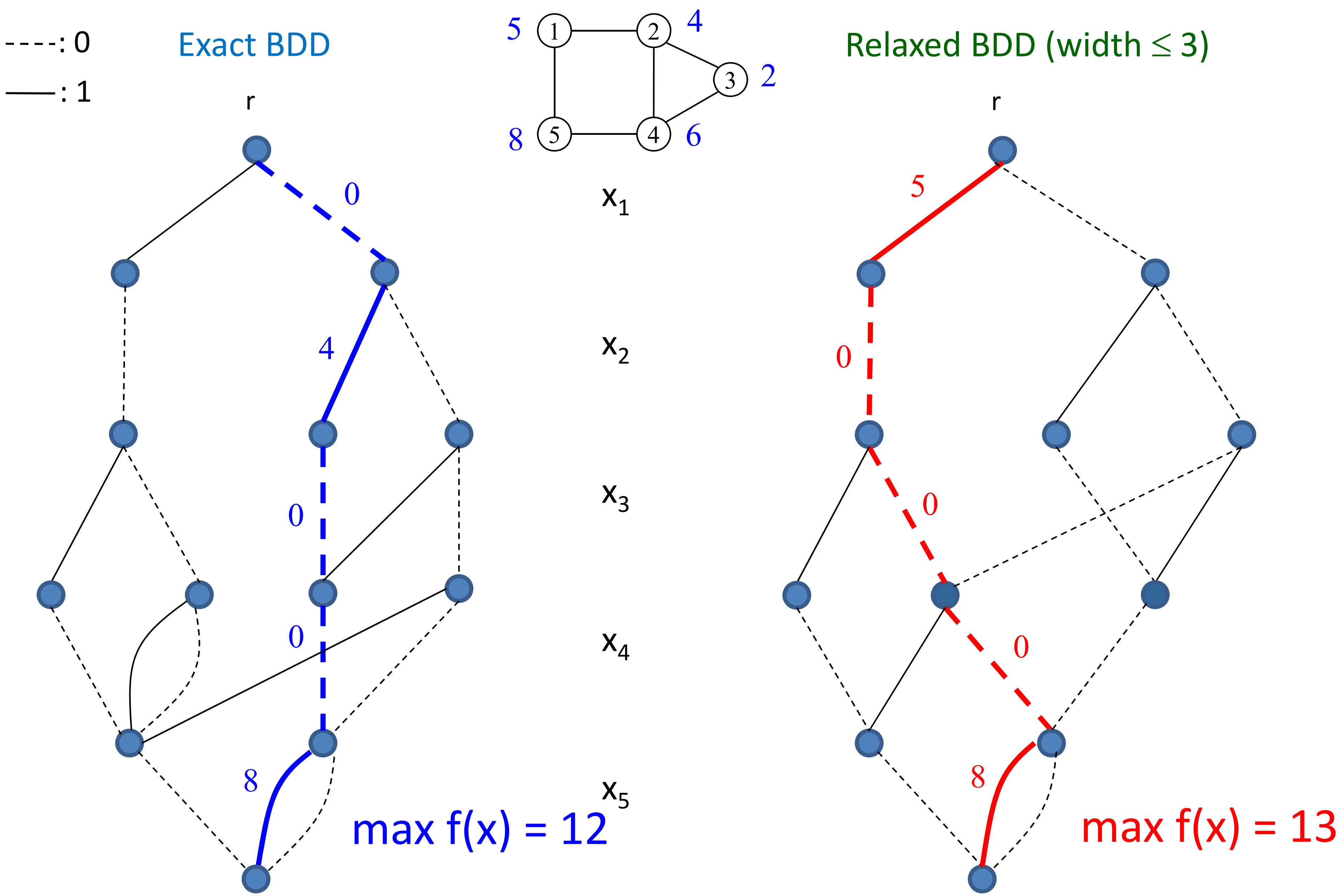
(0,0,0,1,0)

Relaxed BDD



(1,0,0,0,1)

Evaluate Objective Function

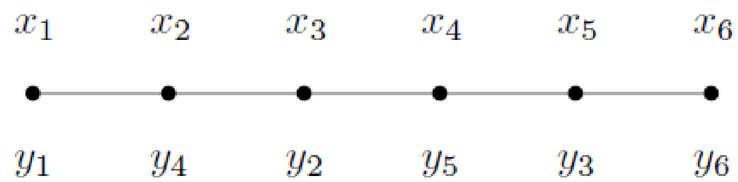


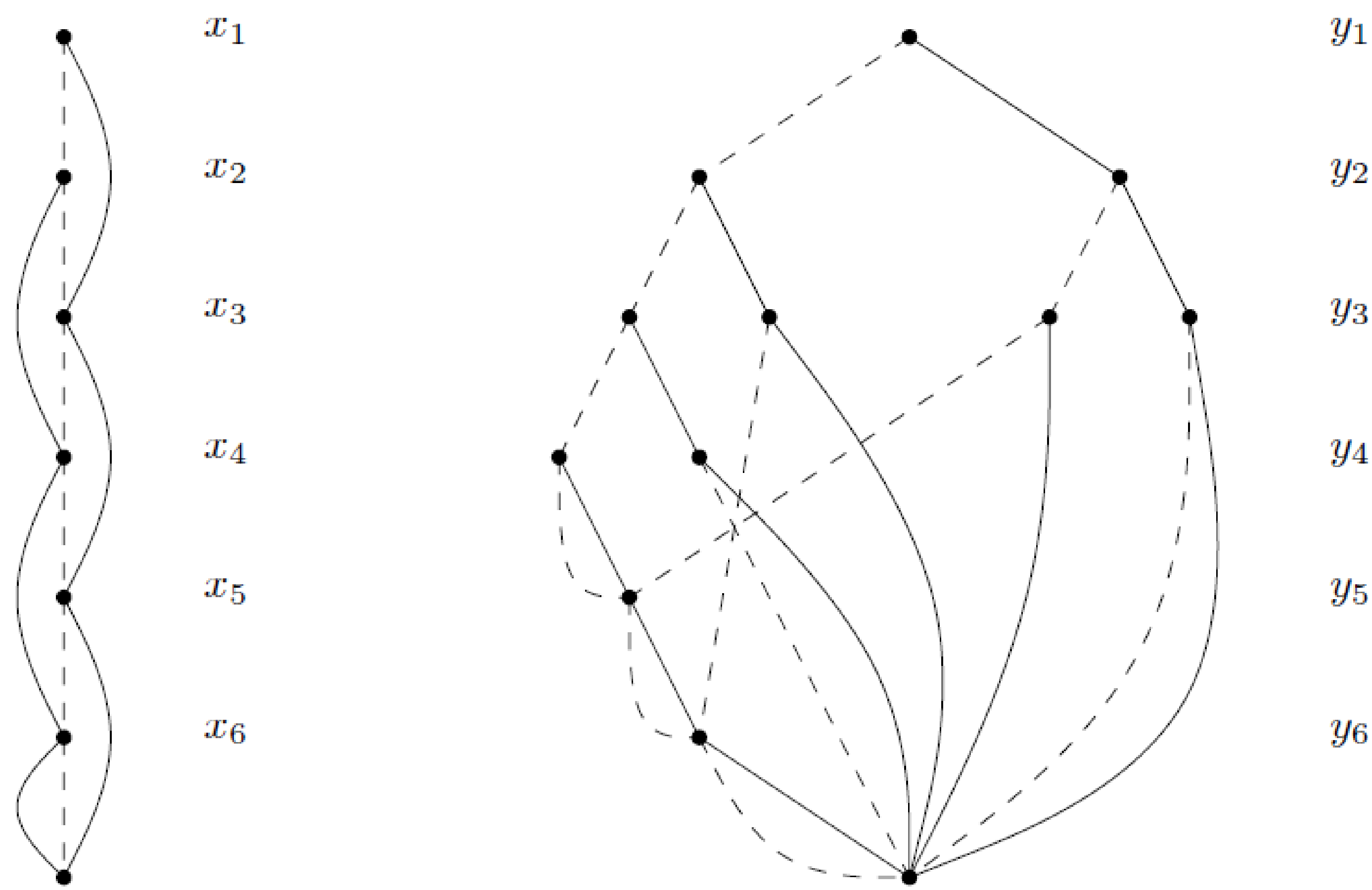
Variable Ordering

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

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

Exact BDD orderings for Paths





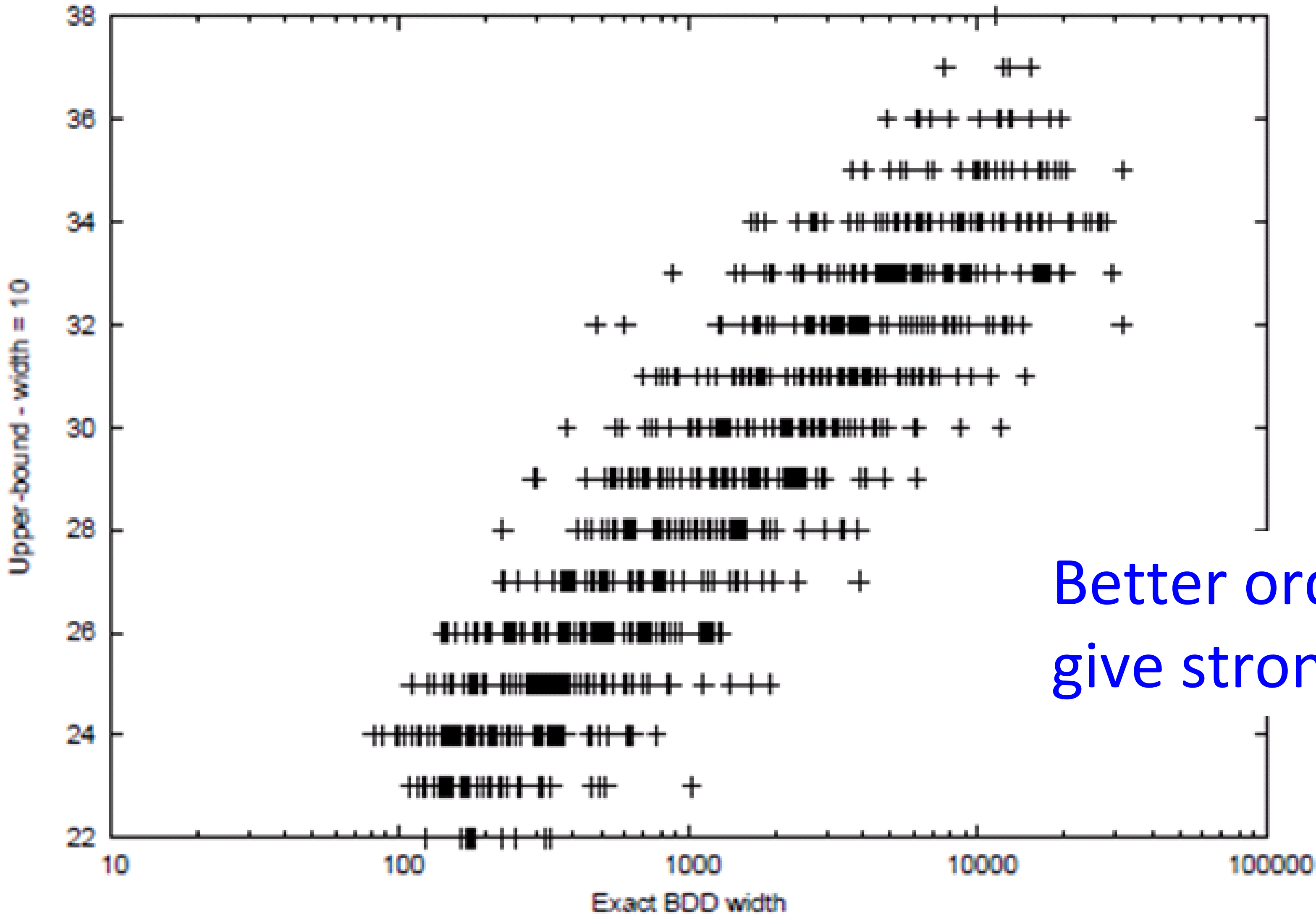
Formal Results for Independent Set

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

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

INFORMS J. Computing (2014)

Many Random Orderings

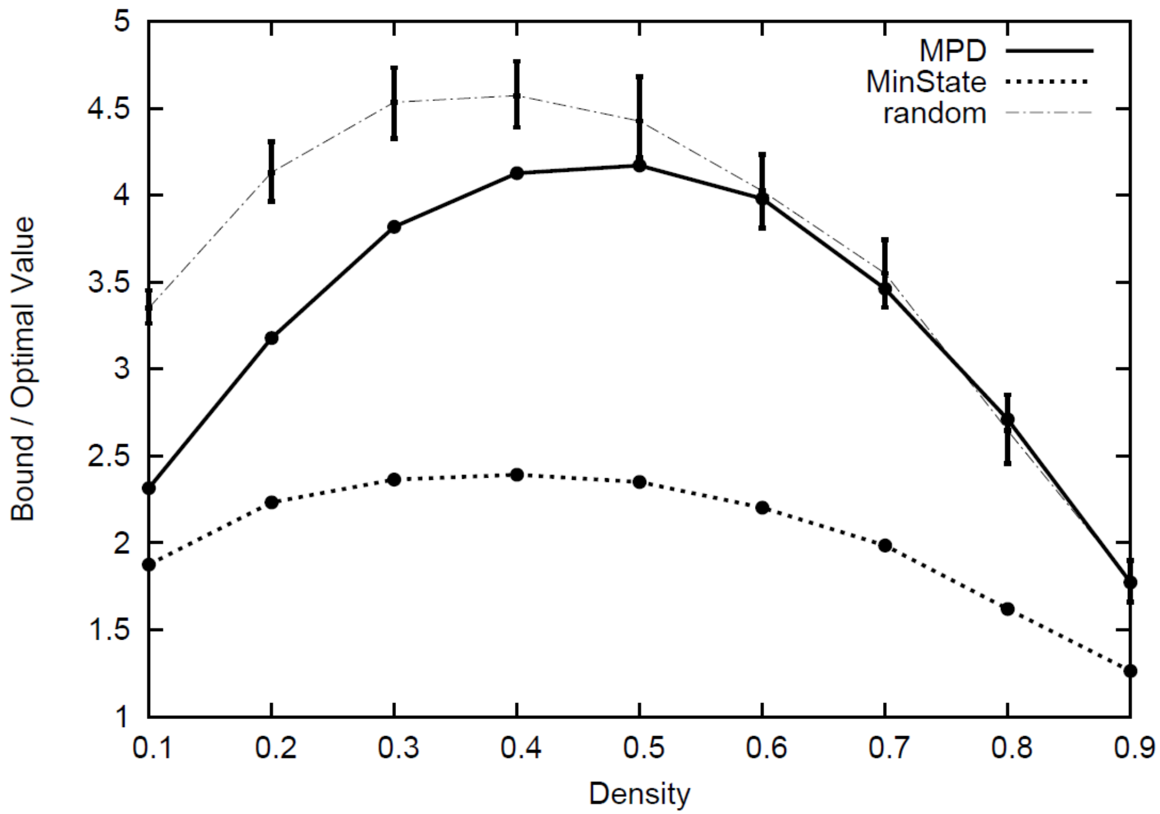


Better orderings
give stronger bounds

For each random ordering, plot the exact BDD width
and the bound from width-10 BDD relaxation

Variable ordering heuristics

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



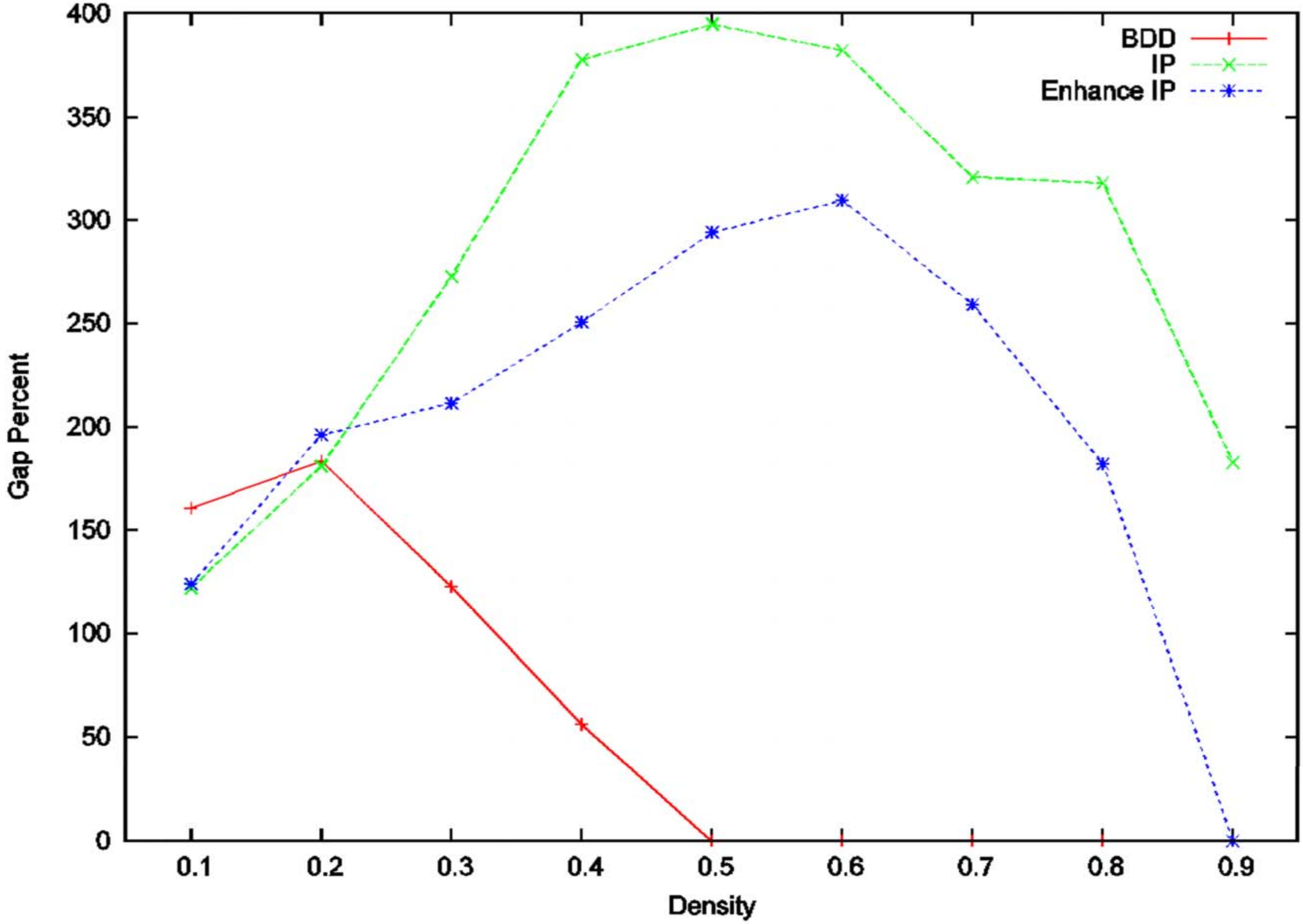
- Each data point is average over 20 instances
- For random, line segment indicates range over 5 instances

Quality of the bound in practice

- Benchmarks
  - Random Erdös-Rényi G(n,p) graphs
  - DIMACS clique graphs (87 instances)
  - Compare with CPLEX 12.5
    (standard MIP model and clique cover model)

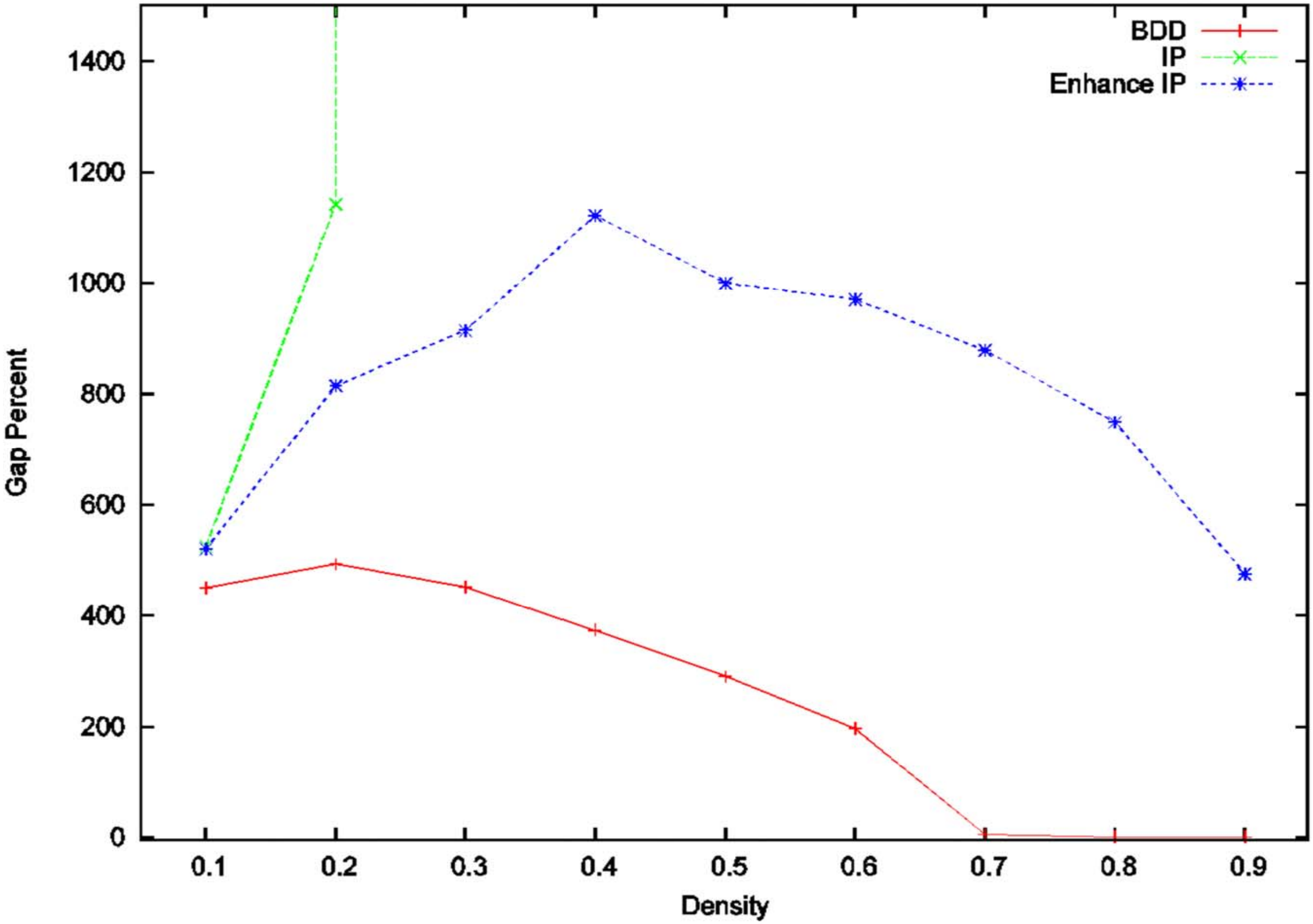
Bounds in practice

random graphs (n=500)



Bounds in practice

random graphs (n=1500)



Restricted BDDs

- Relaxed BDDs find upper bounds for independent set problem
- Can we use BDDs to find lower bounds as well (i.e., good feasible solutions)?
- Restricted BDDs represent a subset of feasible solutions
  - we require that every r-t path corresponds to a feasible solution
  - but not all solutions need to be represented
- Goal: Use restricted BDDs as a heuristic to find good feasible solutions

Creating Restricted BDDs

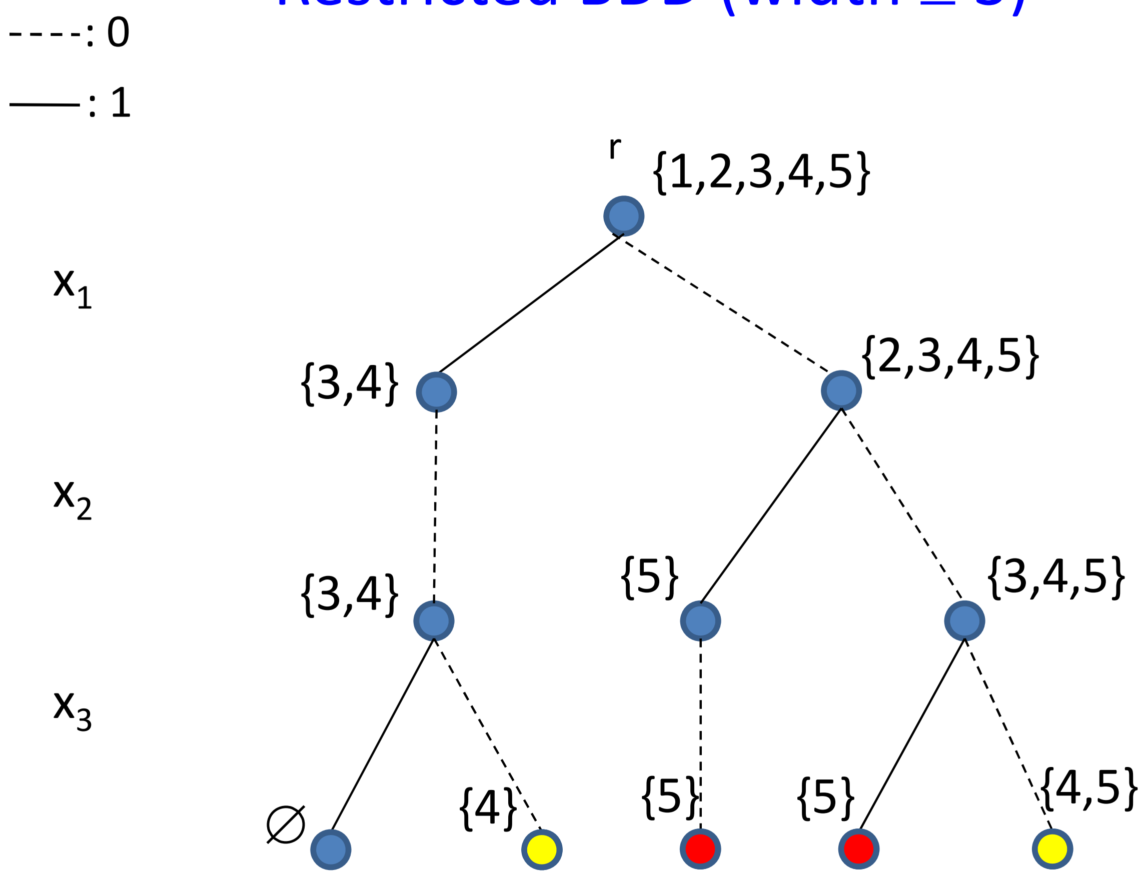
Using an exact top-down compilation method, we can create a limited-width restricted BDD by

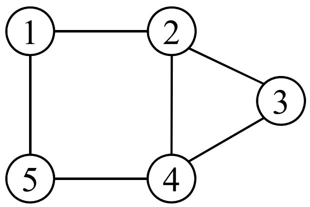
1. merging nodes, or
2. deleting nodes

while ensuring that no non-solutions are introduced

Node merging by example

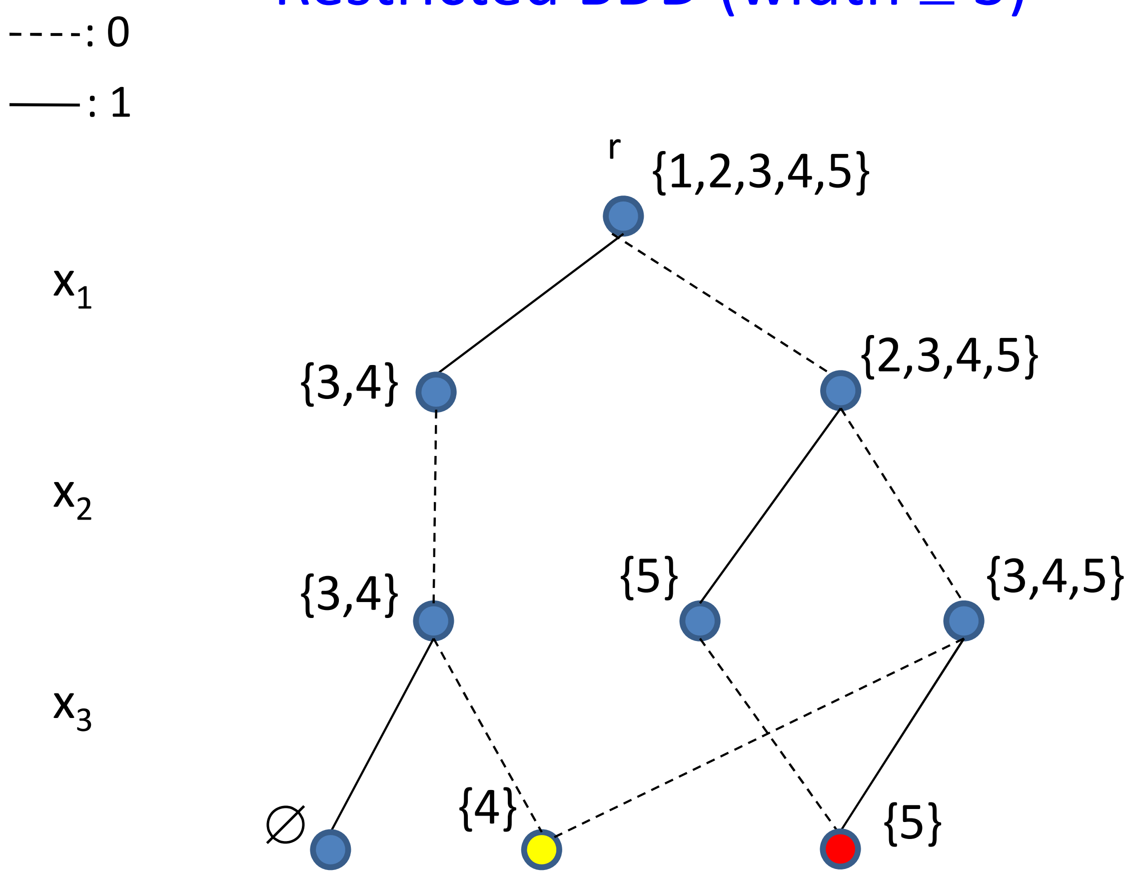
Restricted BDD (width ≤ 3)

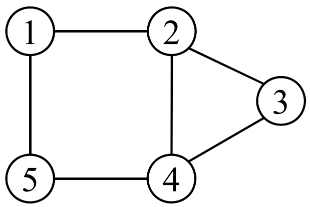




Node merging by example

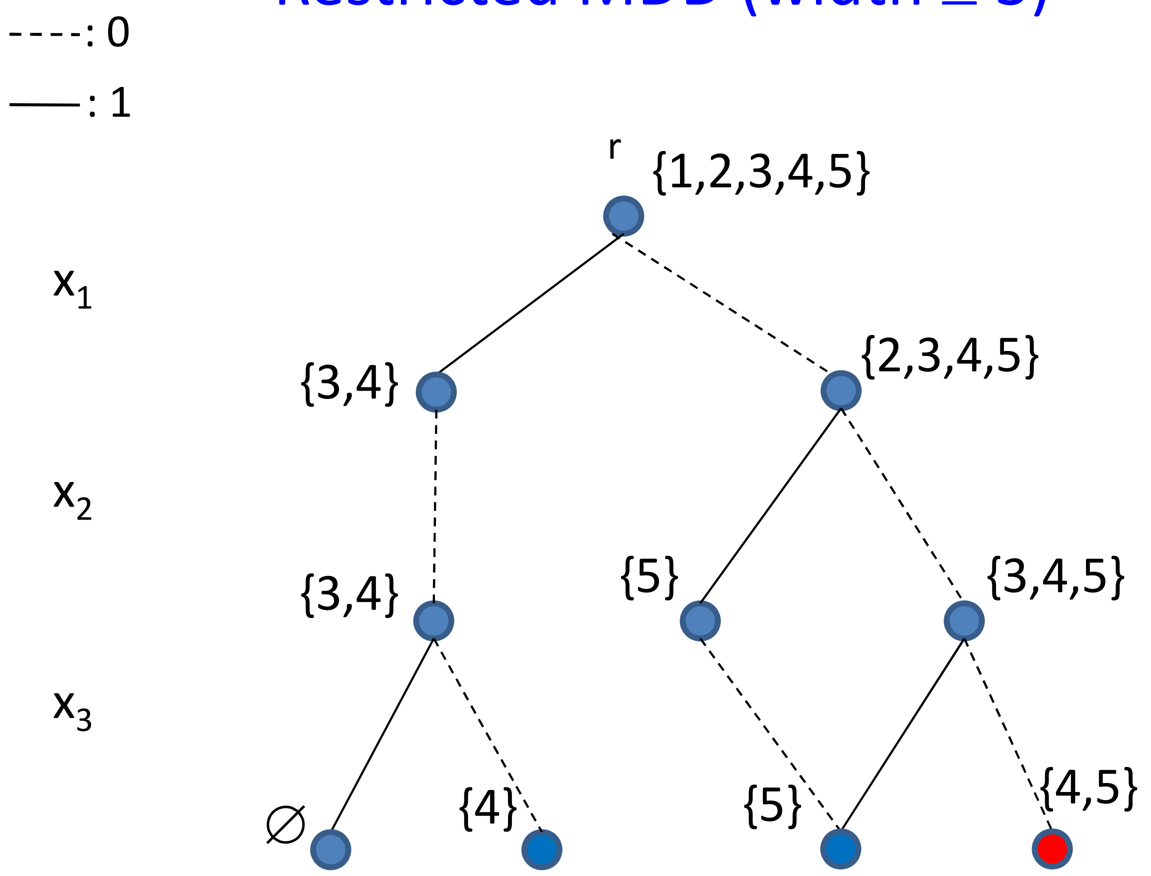
Restricted BDD (width ≤ 3)

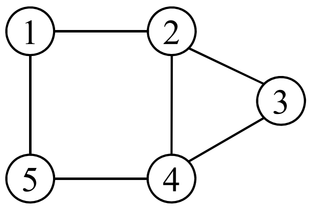




Node deletion by example

Restricted MDD (width ≤ 3)





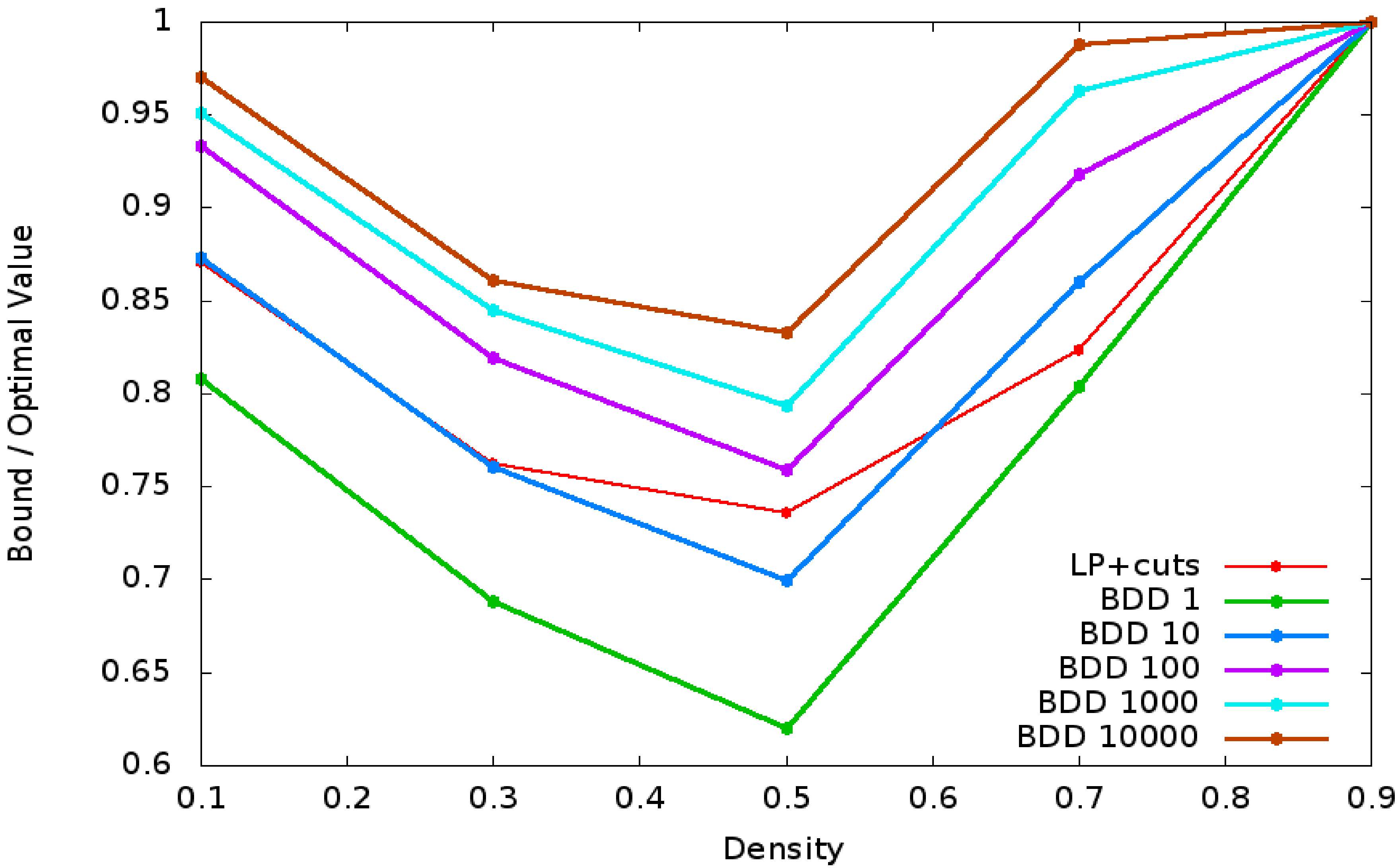
In practice, node deletion superior to node merging

(similar or better bounds, but much faster)

Experimental Evalution

- Compare with Integer Programming (CPLEX)
  - LP relaxation + cutting planes
  - Root node solution
- DIMACS instance set
- Restricted BDDs with varying maximum width

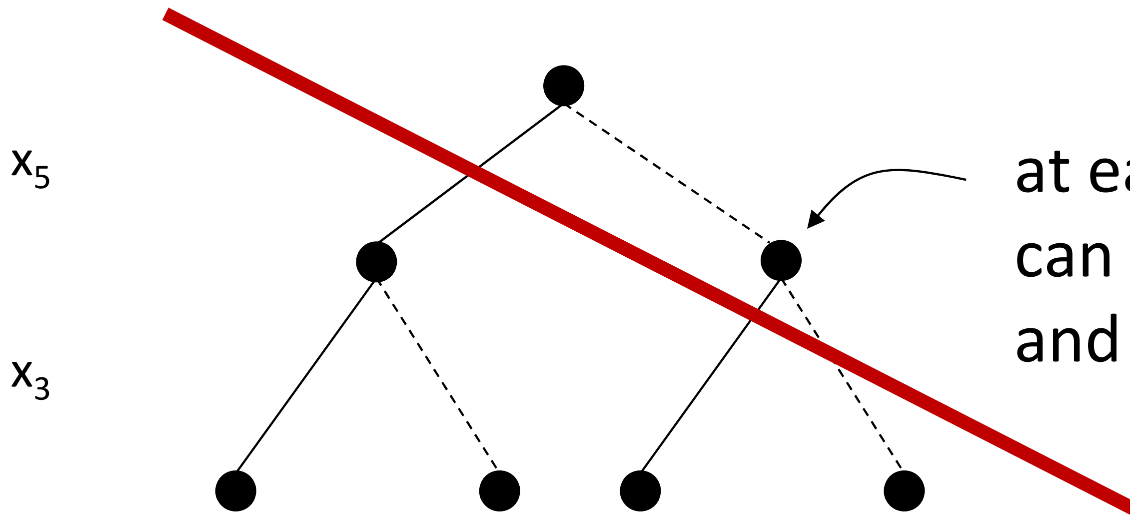
IP versus BDD heuristic



Each data point is geometric mean over 20 instances

BDD-based Branch and Bound

- Search in conventional branch and bound
  - branch on variable ($x \leq v$ or $x \geq v$)
  - branch on constraints ($act_1 << act_2$ or $act_2 << act_1$)

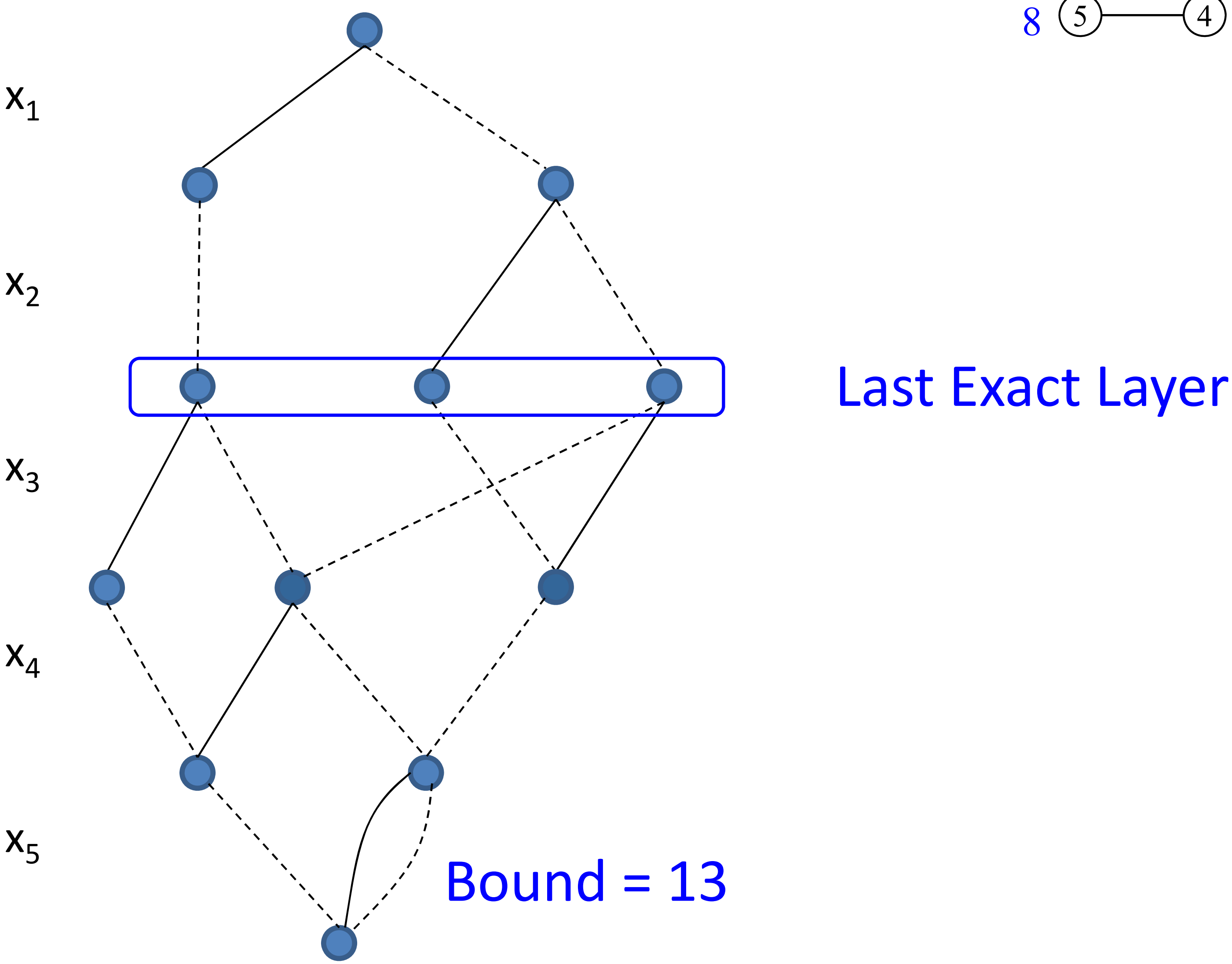


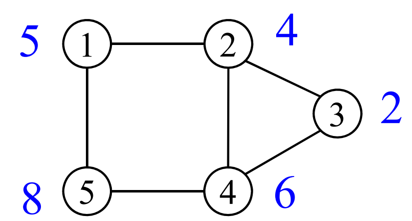
at each search state, can evaluate BDD lower and upper bound

- We will 'branch' on states in the BDD instead

Branch and Bound

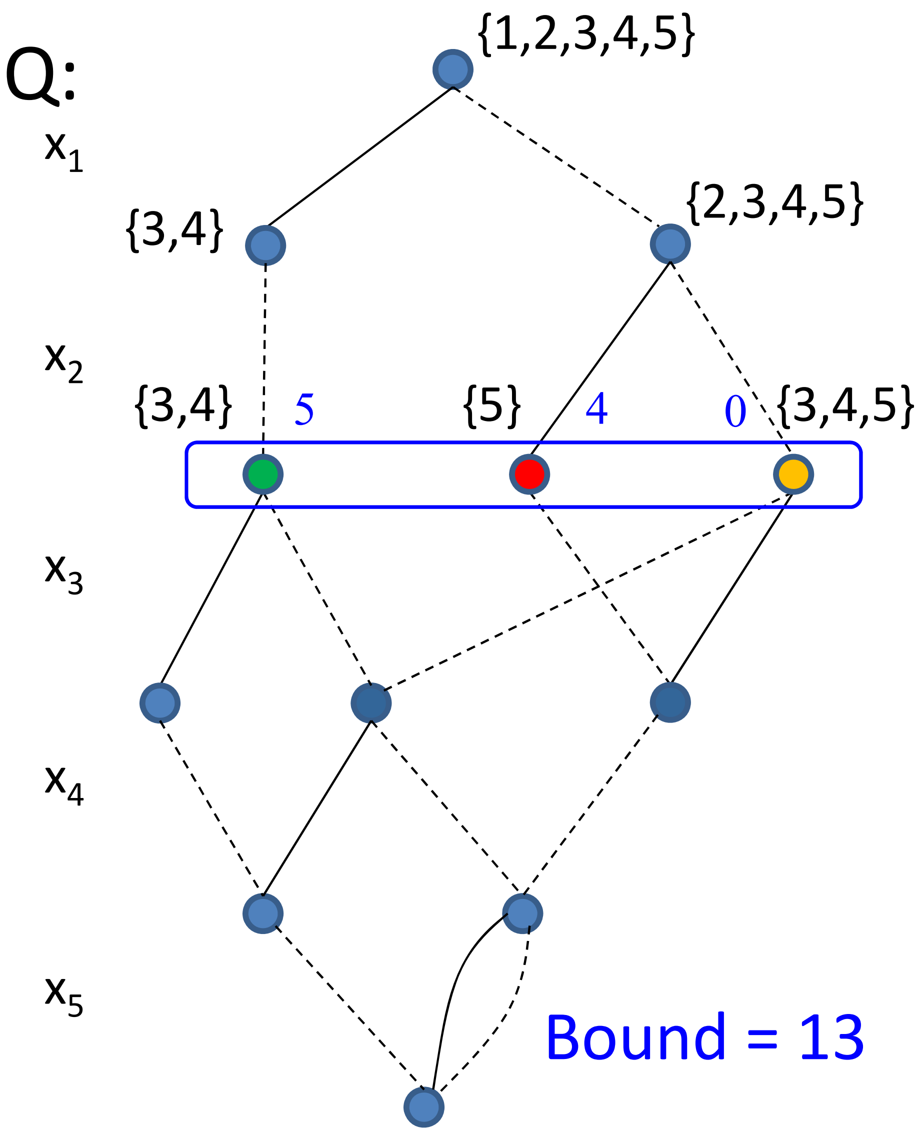
Relaxed BDD (width ≤ 3)



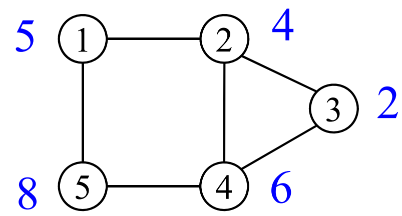


Node Queue

Relaxed BDD (width ≤ 3)

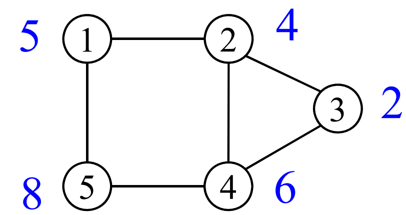


Last Exact Layer

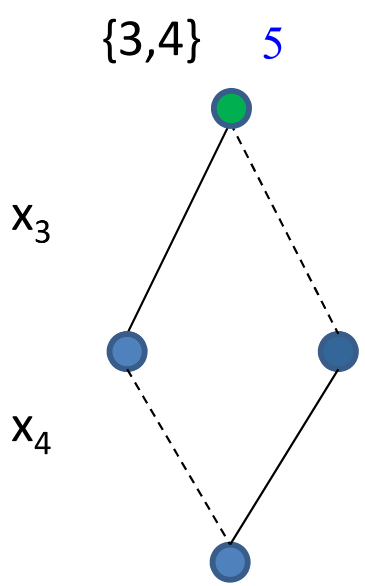


Upper bound = 13

Node Queue



Q: {3,4} 5 {5} 4 0 {3,4,5}



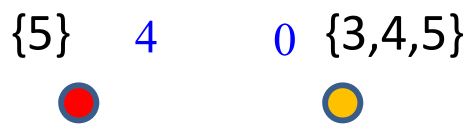
Upper bound = 13

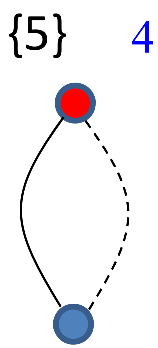
Lower bound = 11

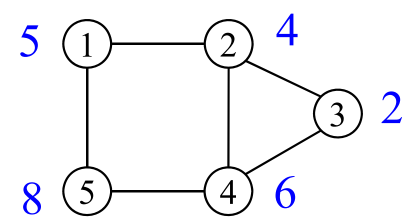
Exact solution: 11

Node Queue

Q:







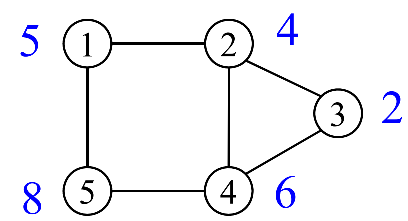
Upper bound = 13

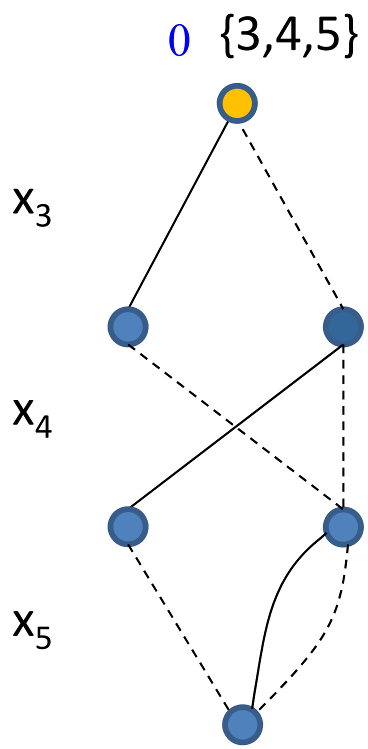
Lower bound = 12

Exact solution: 12

Node Queue

Q:



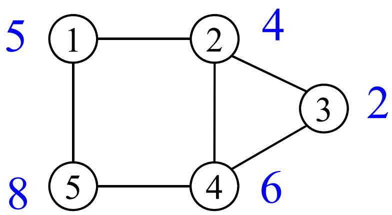


Upper bound = 13

Lower bound = 12

Exact solution: 10

Node Queue



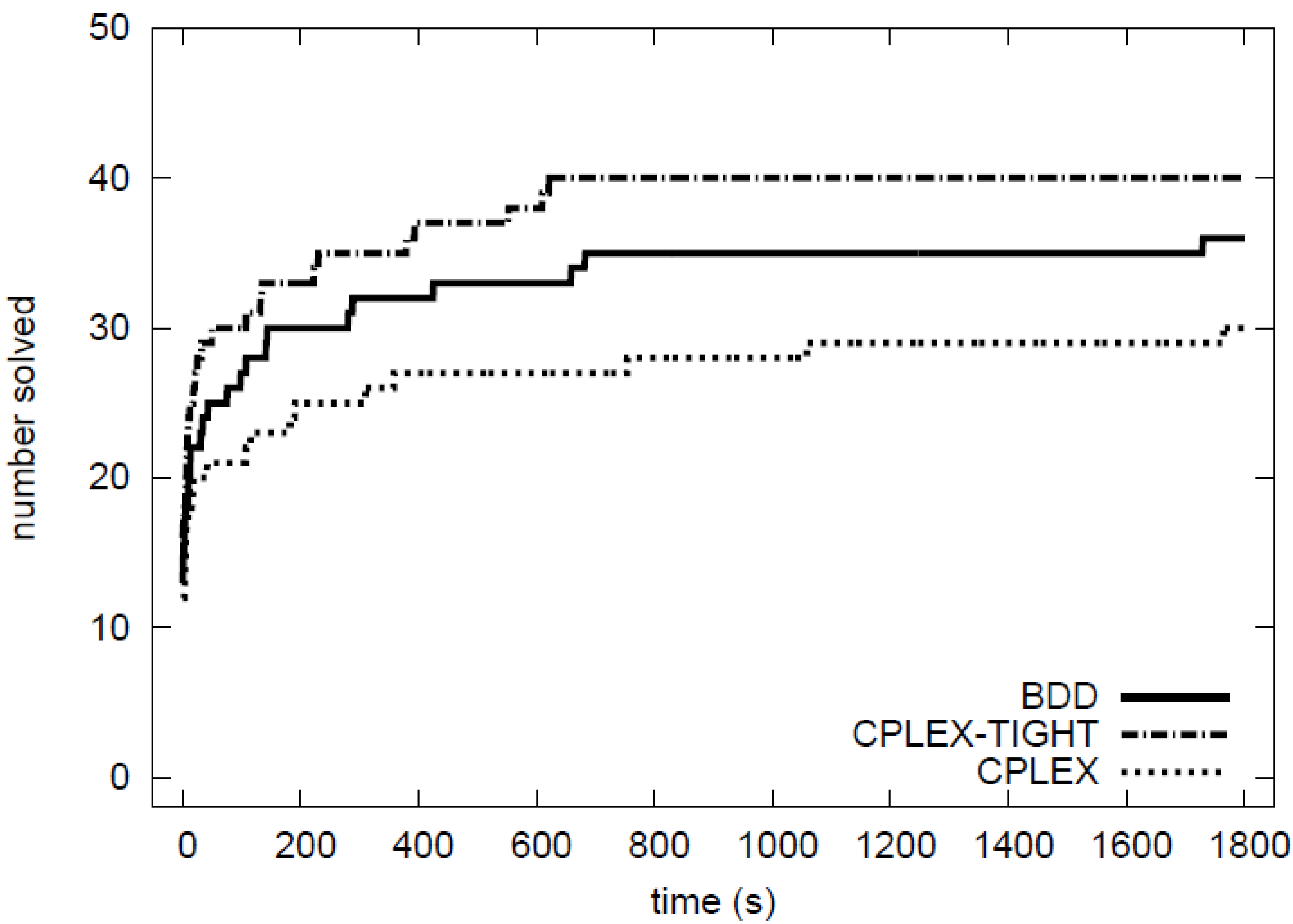
Q:

Optimal Solution: 12

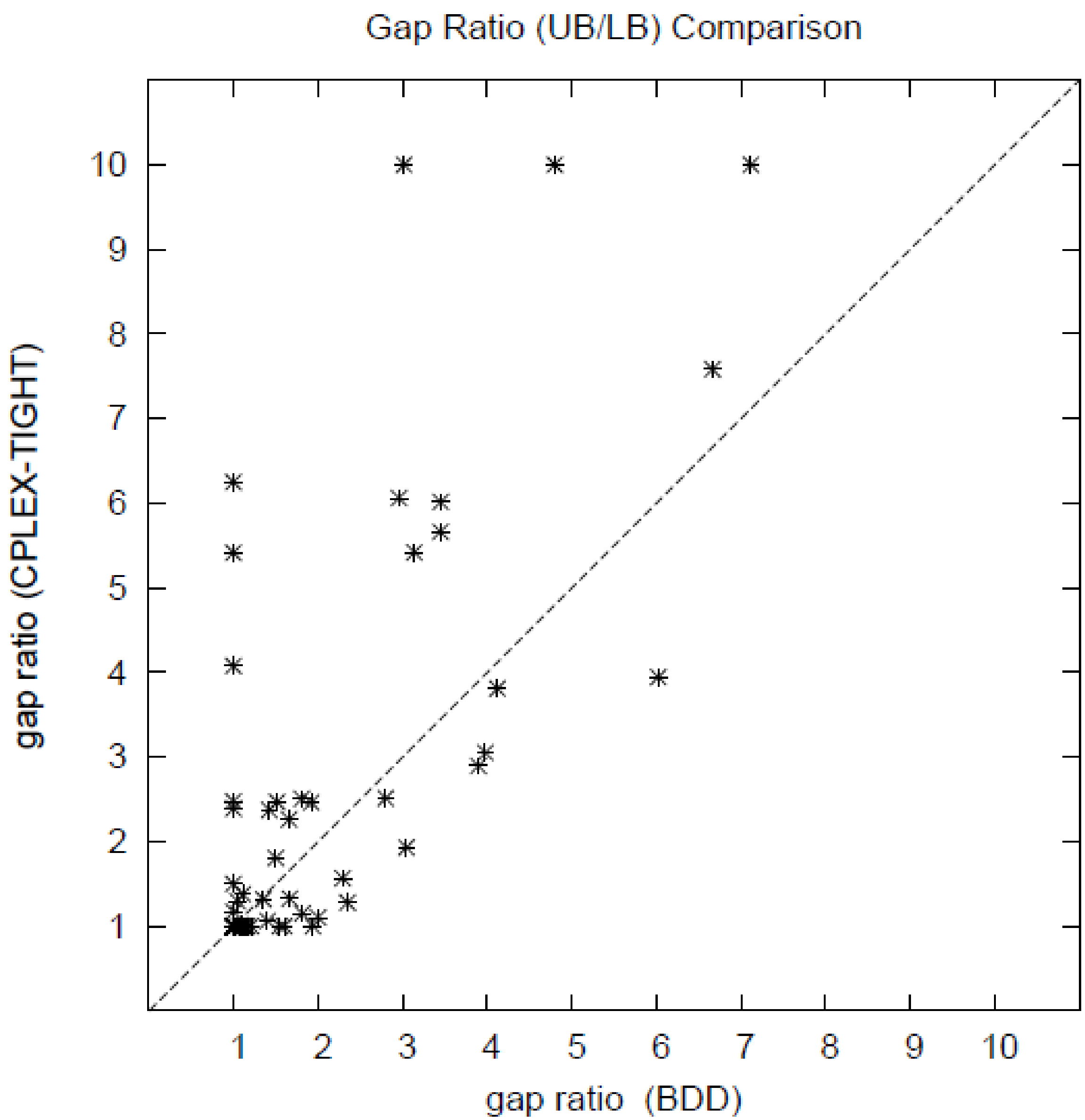
New Branching Scheme

- 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 to different workers, recursive application
    - DDX10 (CPAIOR 2014)

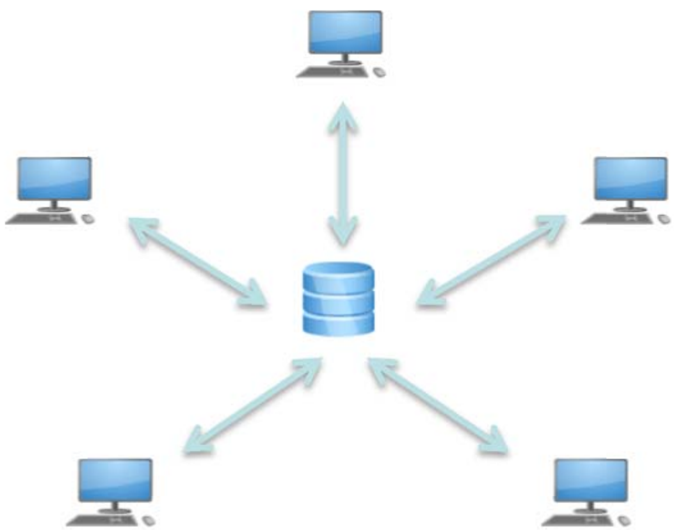
Computational Results: DIMACS



DIMACS Graphs: End Gap (1,800s)



Parallelization: Centralized Architecture



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

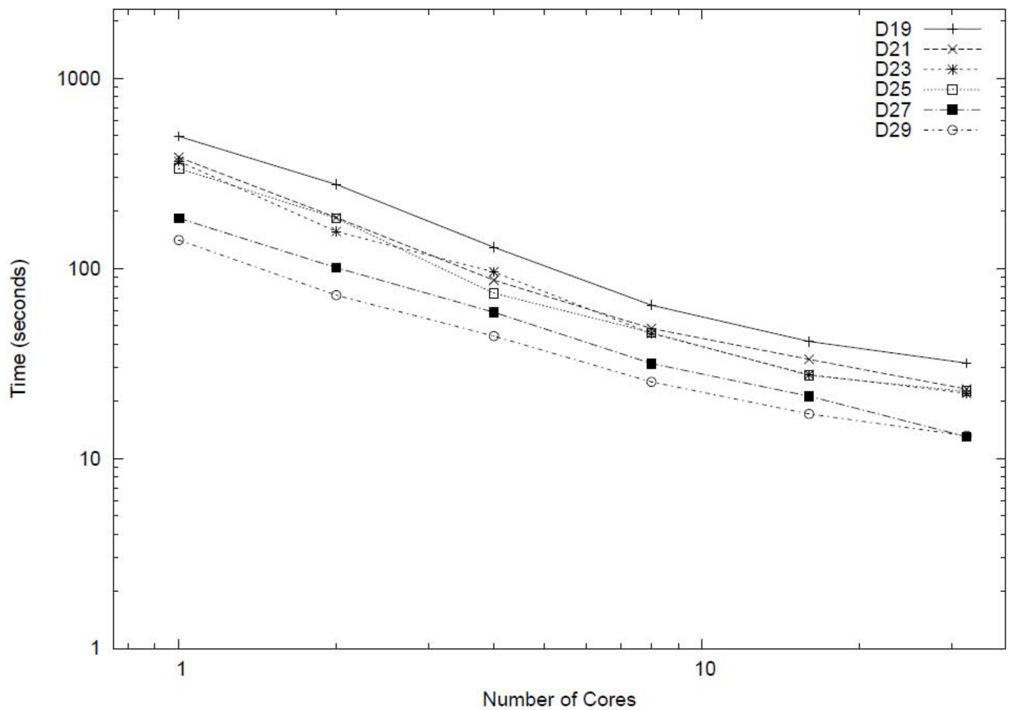
– nodes with larger upper bound have higher priority

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

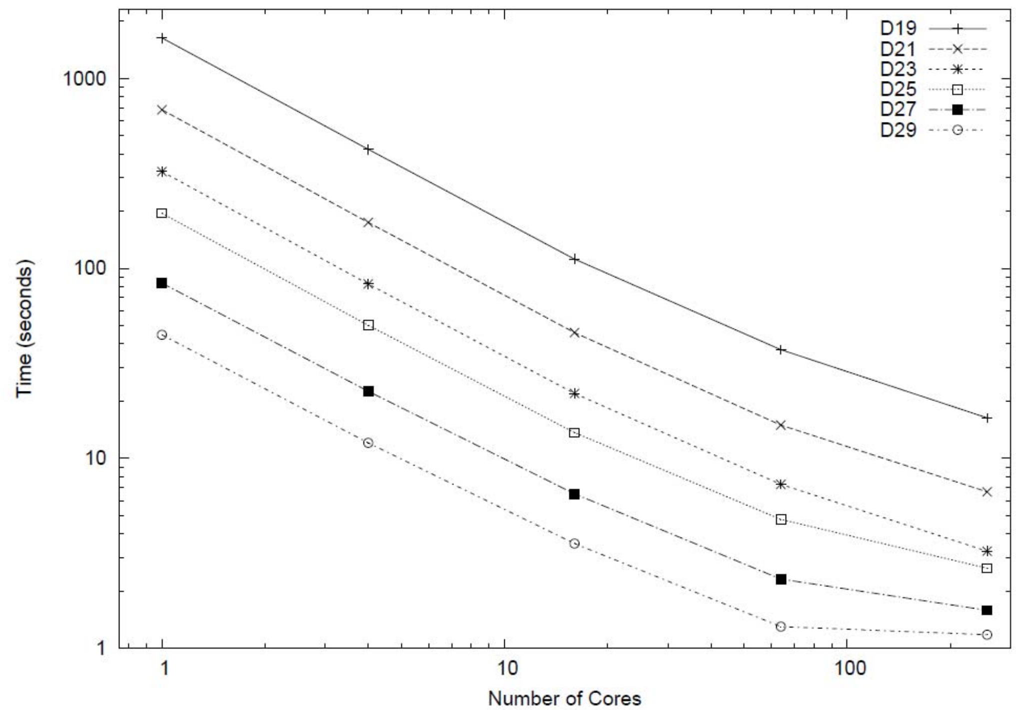
– they also maintain a **local pool** of nodes

Parallelization: BDD vs CPLEX

CPLEX



BDD



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

General Approach

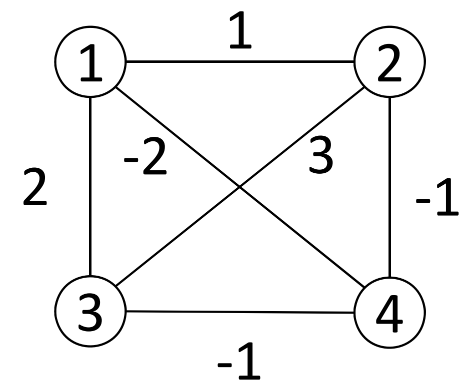
- In general, our approach can be applied when problem is formulated as a dynamic programming model
  - We can build exact BDD from DP model using top-down compilation scheme (exponential size in general)
  - Note that we do not use DP to solve the problem, only to represent it

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

INFORMS J. Computing (to appear)

J. Heuristics (2014)

MAX-CUT representation

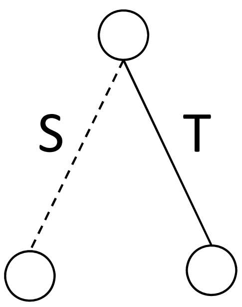


- Value of a cut (S,T) is

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

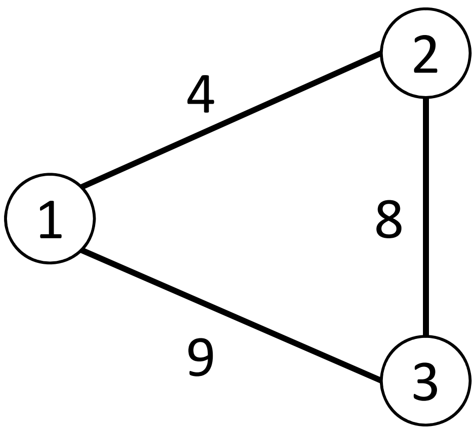
- Example: cut ({1,2}, {3,4)) has value 2
- MAX-CUT: Find a cut with maximum value

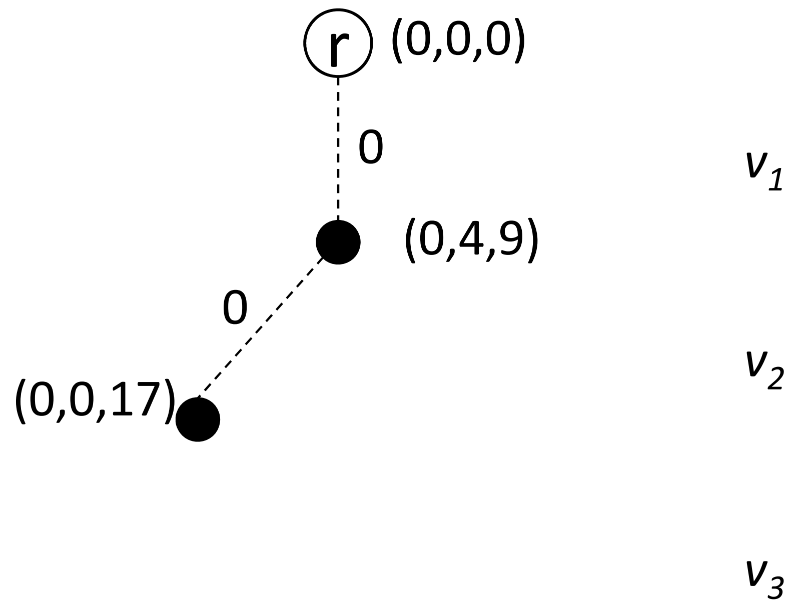
- How can we represent this in a BDD?
  - state represents vertices included in S?
  - we propose a state to represent the marginal cost of including vertex in T

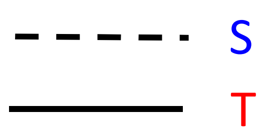


MAX-CUT example BDD

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

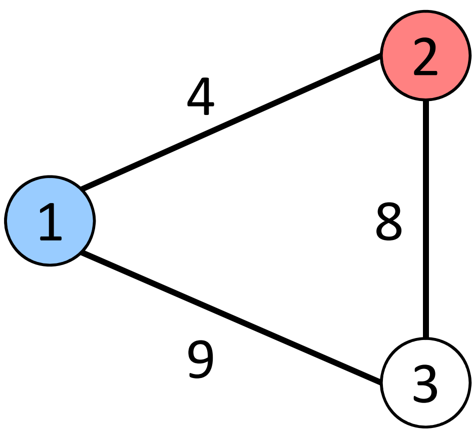


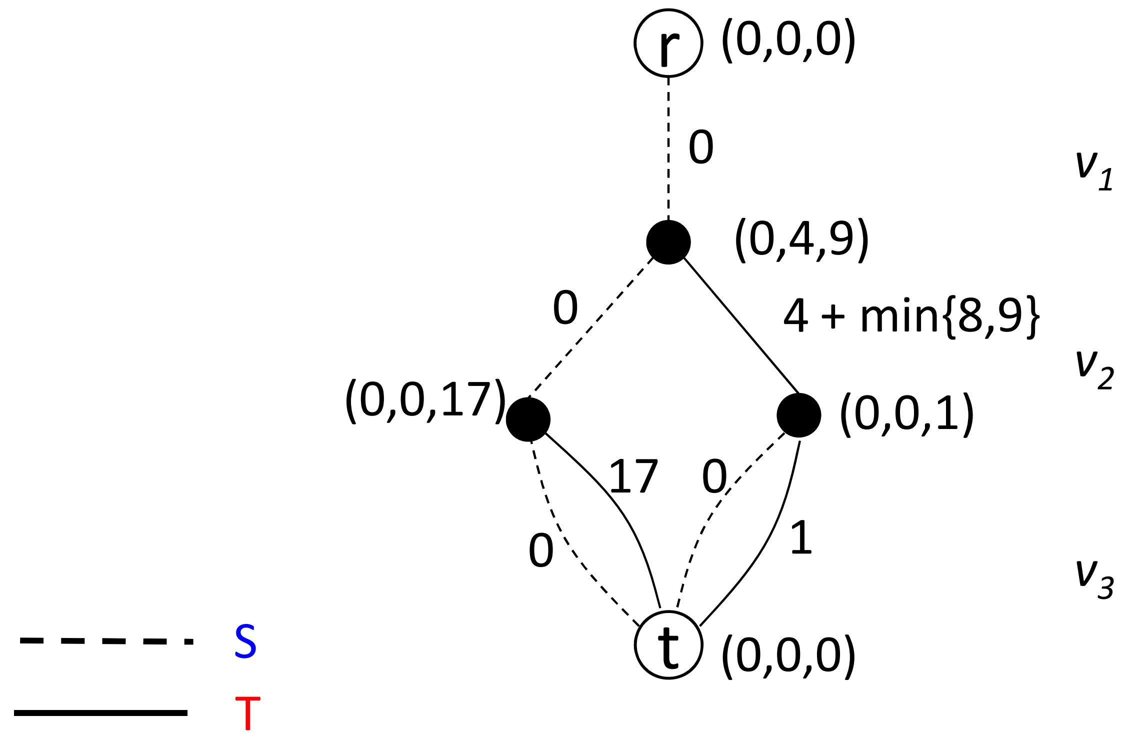




MAX-CUT example BDD

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

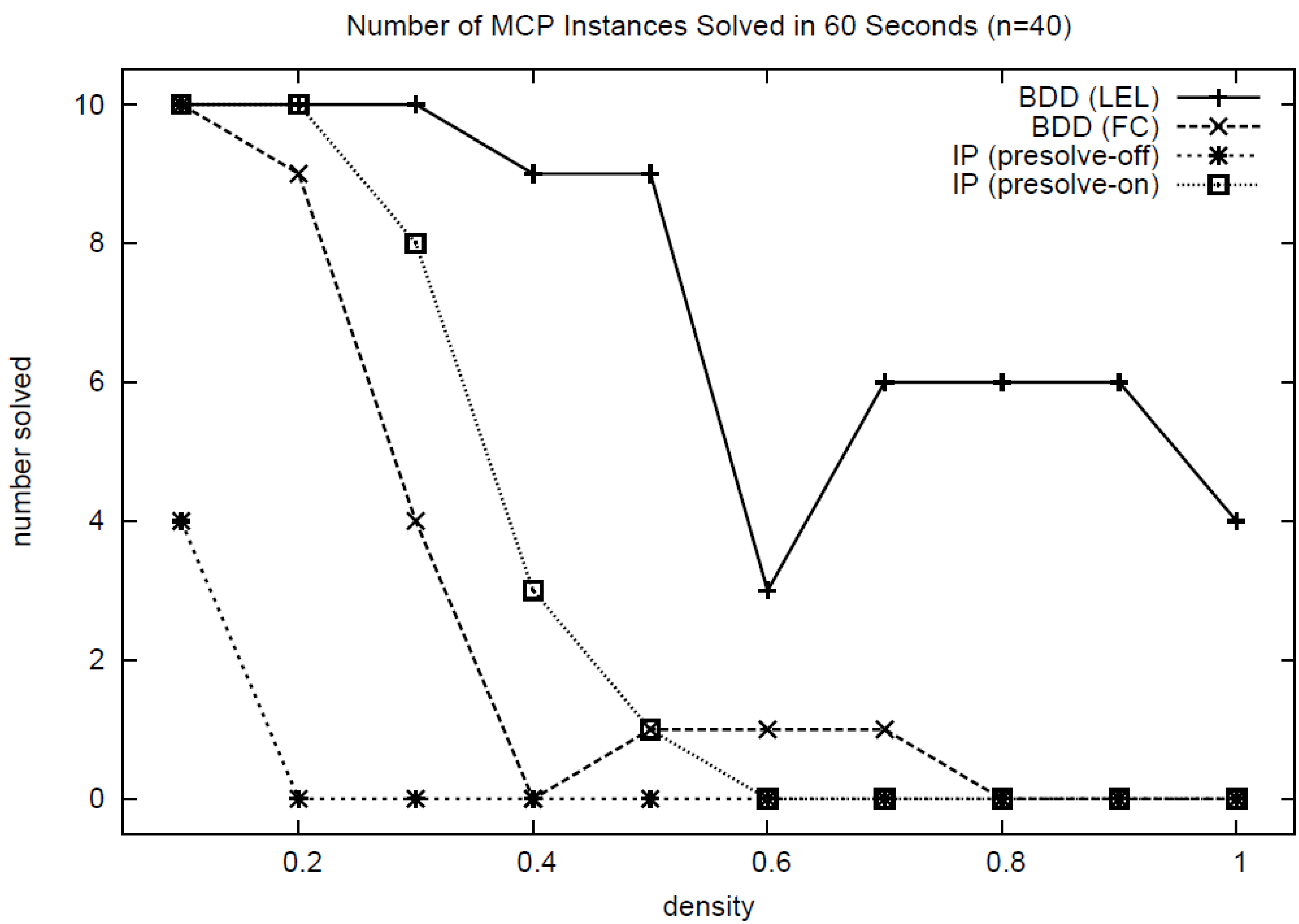




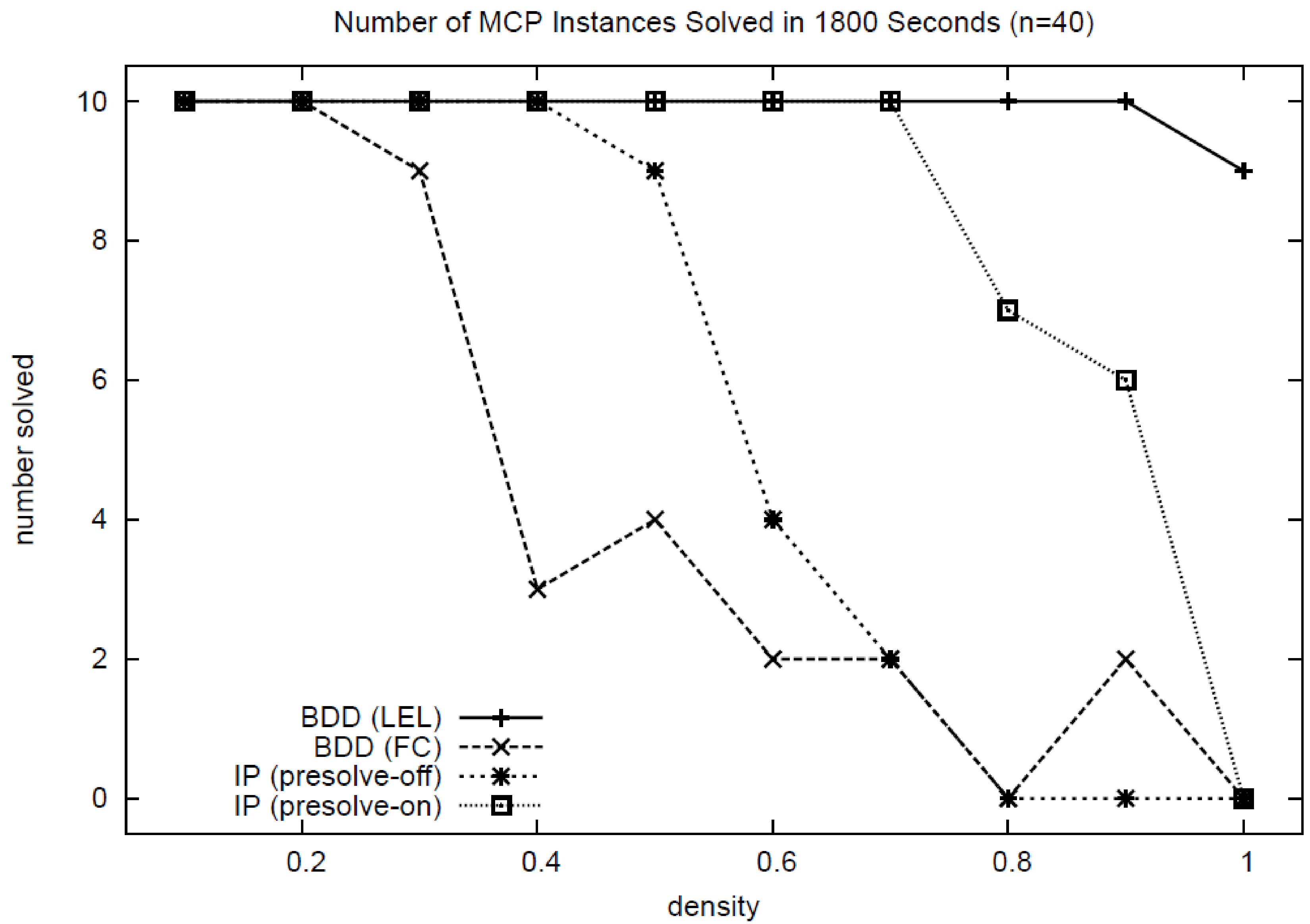
Computational Results

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

MIP vs BDD: 60 seconds (n=40)



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



BiqMac vs BDD

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

Summary

What can MDDs do for Combinatorial Optimization?

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

MDDs for Constraint Programming and Scheduling

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

MDDs for Discrete Optimization

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

Many Opportunities: integrated methods, theory, applications,…

Opportunities / Open issues

- Extend application to CP
  - Which other global constraints are suitable? (Cumulative?)
  - Can we develop search heuristics based on the MDD? (yes)
  - Can we more efficiently store and manipulate approximate MDDs? (Implementation issues)
  - Can we obtain a tighter integration with CP domains?
- MDD technology
  - How should we handle constraints that partially overlap on the variables? Build one large MDD or have partial MDDs communicate?
  - How do we communicate information between MDDs on different subproblems (e.g., jobshop)? (Lagrangians)

Opportunities / Open issues (cont'd)

- Formal characterization
  - Can MDDs be used to identify tractable classes of CSPs?
  - Can we identify classes of global constraints for which establishing MDD consistency is hard/easy?
  - Can MDDs be used to prove approximation guarantees?
  - Can we exploit a connection between MDDs and tight LP representations of the solution space?
- Optimization
  - Relaxed/restricted MDDs can provide bounds for any nonlinear (separable) objective function. Demonstrate the performance on an actual application.

Opportunities / Open issues (cont'd)

- Beyond classical CP
  - How can MDDs be helpful in presence of uncertainty? E.g., can we use approximate MDDs to represent policy trees for stochastic optimization?
  - Can we utilize limited-width BDDs for SAT? (yes)
  - Can MDDs help generate nogoods, e.g., in lazy clause generation? (yes)
  - Tighter integration of MDDs in MIP solvers? (yes)
- Applications
  - So far we have looked mostly at generic problems. Are there specific application areas for which MDDs work particularly well? (Bioinformatics?)

7. Consider the following CSP

$$4x_1 + 2x_2 + x_3 + x_4 + 2x_5 + 4x_6 = 7$$

$$x_1, x_2, \ldots, x_6 \in \{0,1\}$$

a) Draw an exact BDD for this problem using the variable ordering $x_1, x_2, x_3, x_4, x_5, x_6$

b) Draw an exact BDD for this problem using the variable ordering $x_1, x_6, x_2, x_5, x_3, x_4$

c) Which of the two orderings yields the smallest width?

Exercises

8. Consider the following set covering instance:

$$
\begin{array}{llcccccl}
\text{minimize} & 3x_1 + 2x_2 + x_3 + 4x_4 + 2x_5 \\
\text{s.t.} & x_1 + x_2 + x_3 & & & \geq 1 \\
& x_1 & & + x_4 + x_5 & \geq 1 \\
& \quad\; x_2 & & + x_4 & \geq 1
\end{array}
$$

What state representation would you use to define the BDD? Construct a restricted BDD with maximum width 3. Does it yield the optimal solution?