Exact Algorithms for NP-hard problems

Advanced Algorithms: Part 2, Lecture 5

Today

- Decision diagrams
- Guest lecture by Matthias Horn

Slides based on slides by
Willem-Jan van Hoeve, Andre Cire, Christian Tjandraatmadja

see https://www.andrew.cmu.edu/user/vanhoeve/mdd/

Mathijs de Weerdt

Decision diagrams

Decision diagrams

- Search trees: trying to prevent computing same subproblems, but these could still occur
- Dynamic programming: no recomputation because of identification of subproblems, but runtime depends on state space (e.g. 2^n in TSP)
- DP over tree decomposition: use tree (like) structure of graph to prevent recomputation & use DP where not tree-like, but not all input is a graph and not easy to find good tree decomposition

An attempt at unifying and generalizing these ideas: *decision diagrams*.

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]

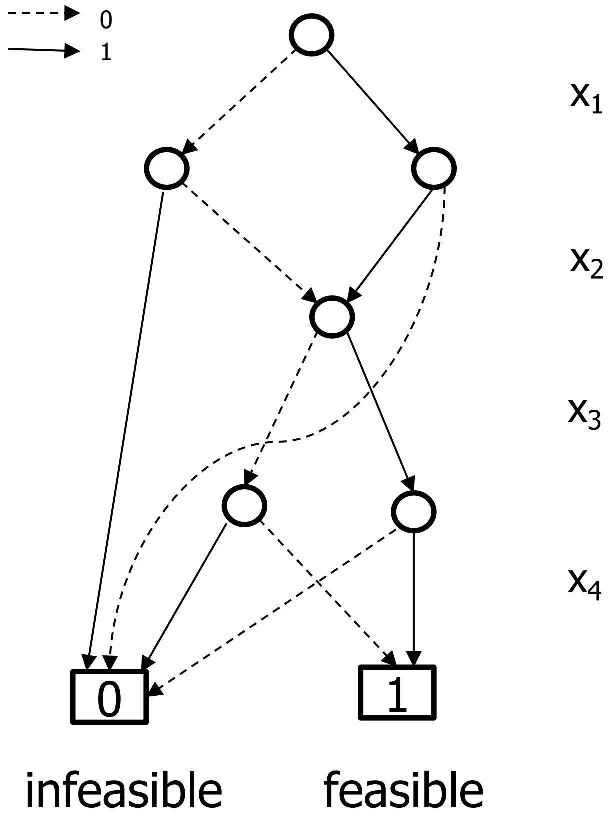
Decision Diagrams: Optimization View

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

subject to

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

- like a search tree, but a DAG where multiple arcs may lead to the same node (state)
- like DP, but only represent states that occur as a subproblem

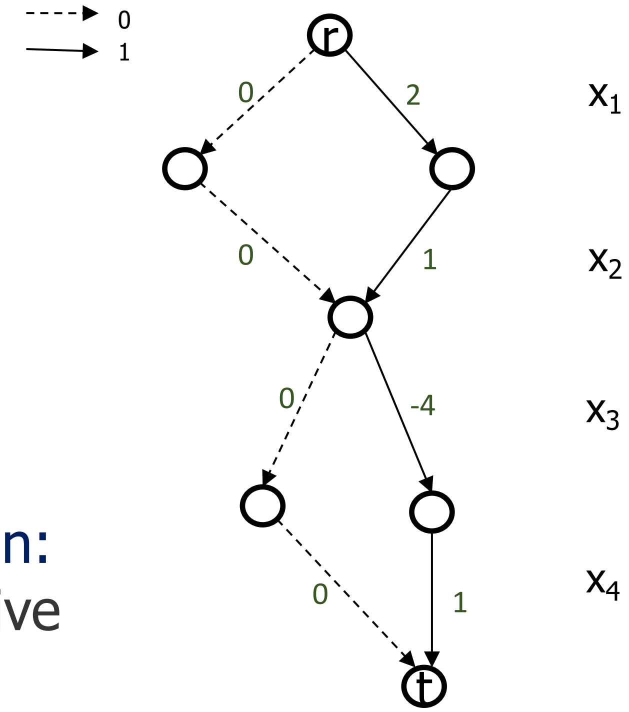


Decision Diagrams: Optimization View

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

subject to

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

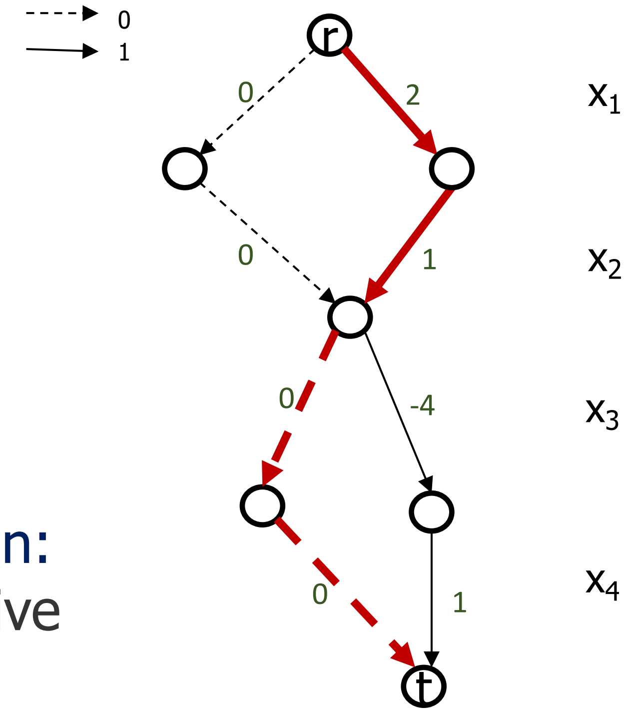


- Maximizing a linear (or separable) function:
  - Arc lengths: contribution to the objective
  - Longest path: optimal solution

Q. What is the longest path here?

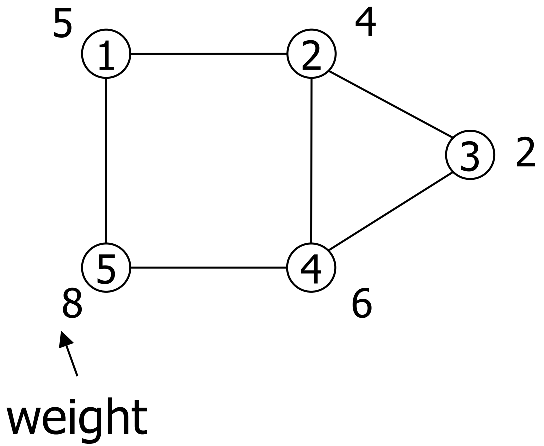
Decision Diagrams: Optimization View

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

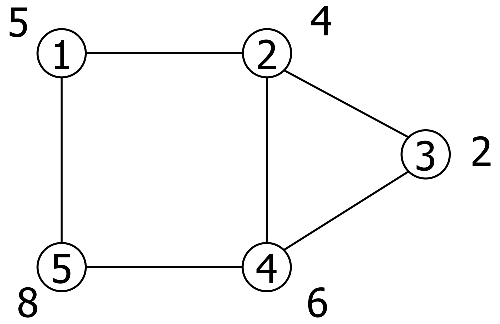
Example: Maximum-weight Independent Set Problem



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

Q. How to write as integer linear program?

Example: Maximum-weight Independent Set Problem



Integer Programming Formulation:

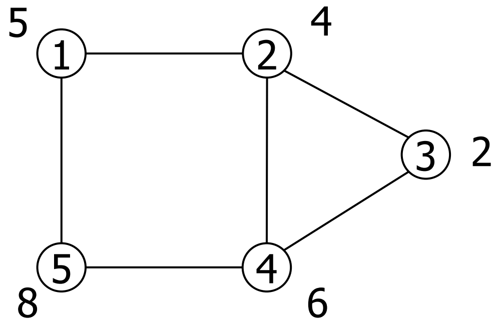
max $5x_1 + 4x_2 + 2x_3 + 6x_4 + 8x_5$

subject to

$$
\begin{aligned}
x_1 + x_2 &\le 1 \\
x_1 + x_5 &\le 1 \\
x_2 + x_3 &\le 1 \\
x_2 + x_4 &\le 1 \\
x_3 + x_4 &\le 1 \\
x_4 + x_5 &\le 1 \\
x_1, x_2, x_3, x_4, x_5 &\in \{0,1\}
\end{aligned}
$$

Q. What are the most important elements in a dynamic programming approach?

Example: Maximum-weight Independent Set Problem



Dynamic Programming:

- Exploit recursiveness of subproblems
- Subproblems are represented by a *state*
- Decisions (or *controls*) define *state transitions*

Decision diagram: State-Transition Graph

- Nodes corresponds to states (subproblems)
- Arcs are state transitions (decisions)
- Arc weights are transition costs

A *graphical representation* of dynamic programming!

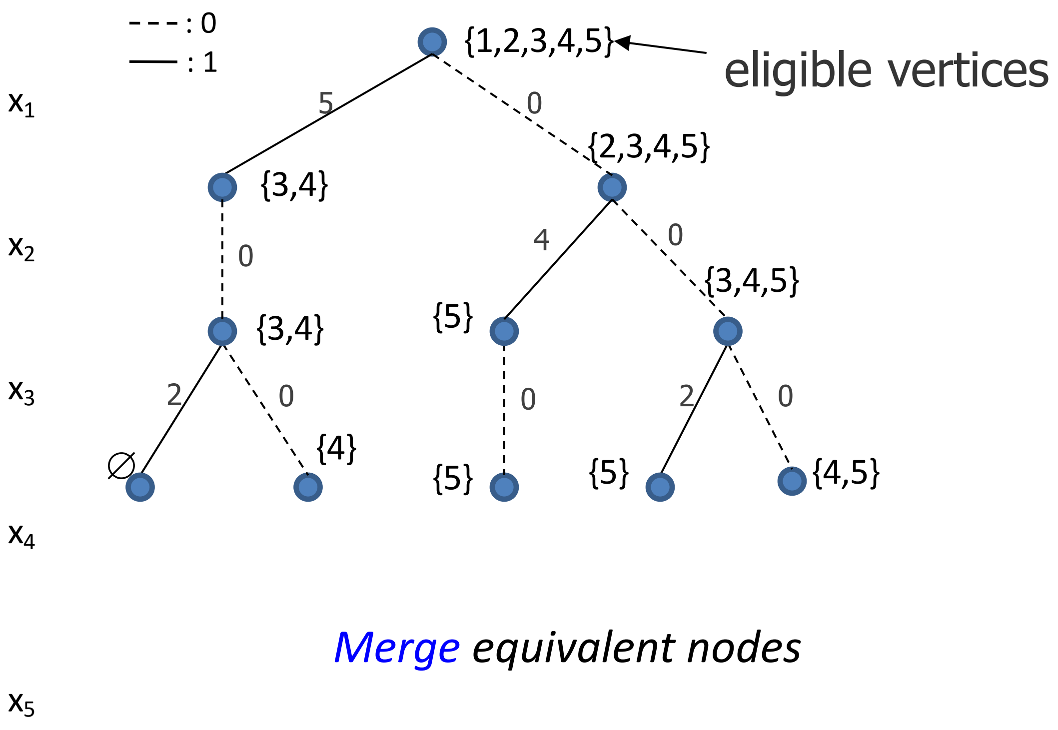
Modeling Framework

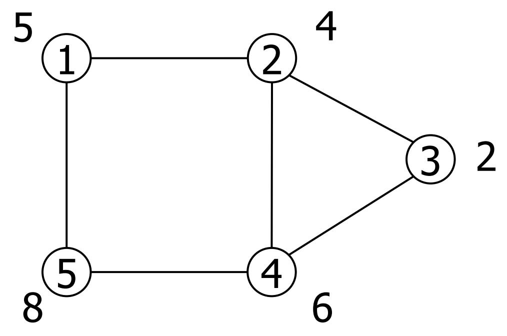
Decision diagram for the maximum independent set:

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

Order of (decisions on) vertices i is fixed (**variable ordering**).

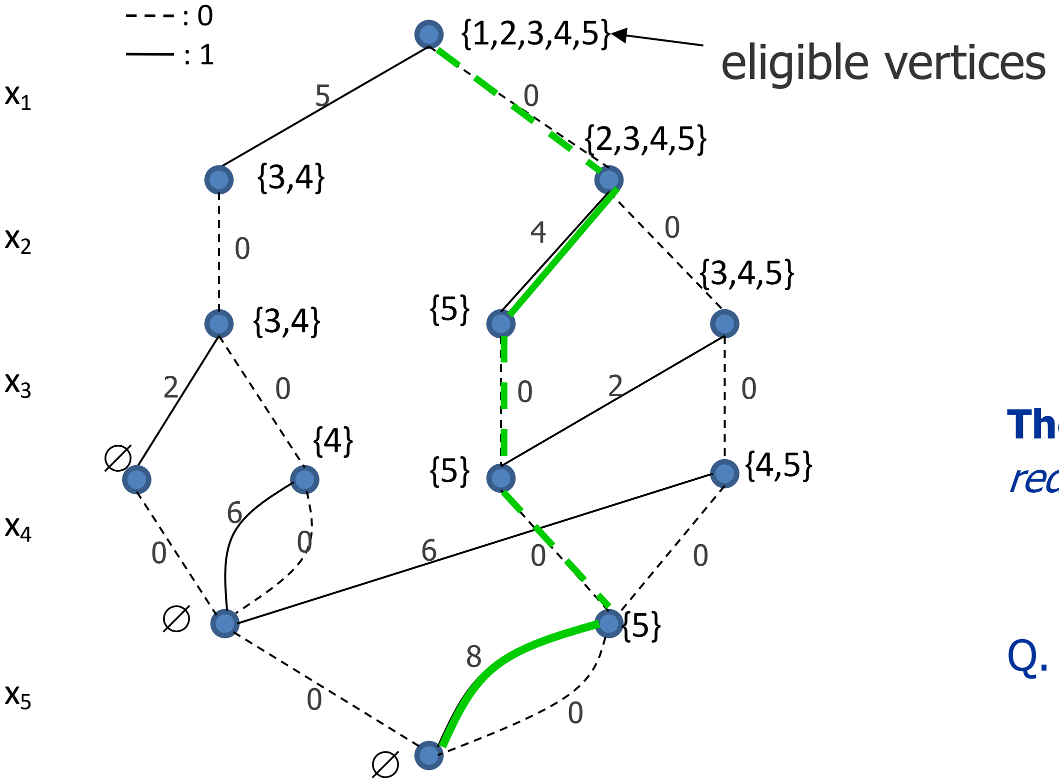
Maximum Independent Set Problem

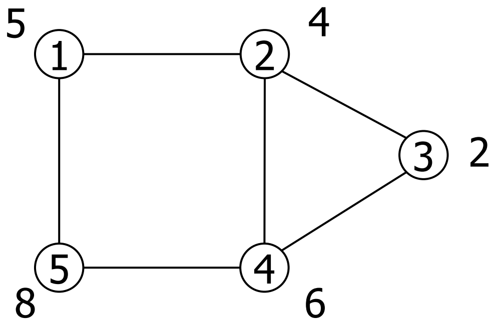




Merge equivalent nodes

Maximum Independent Set Problem



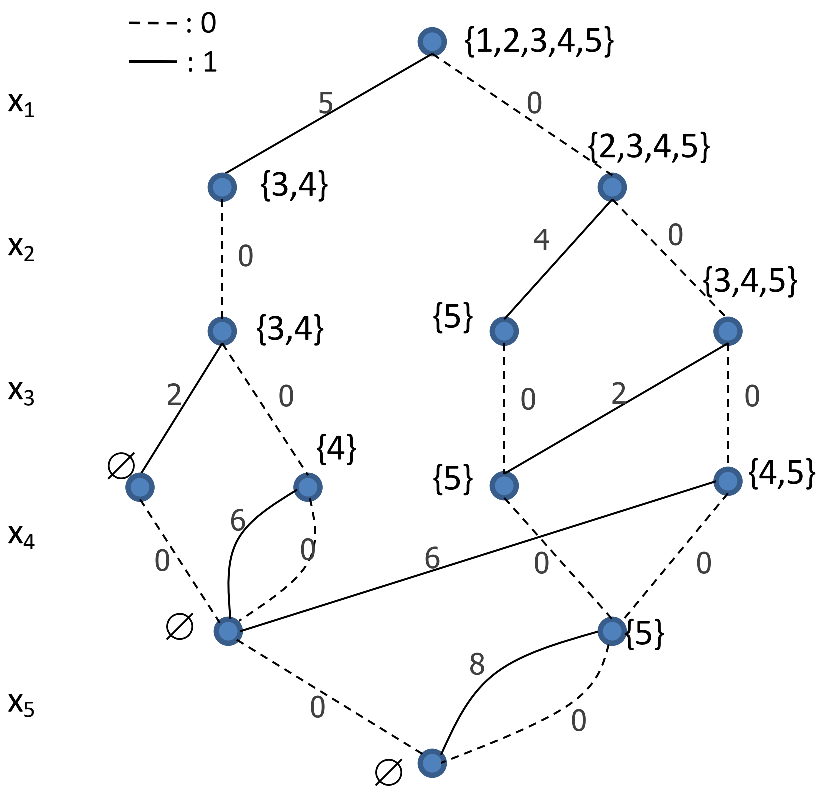


Theorem: This procedure generates a *reduced* exact BDD

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

Q. What is the optimal solution?

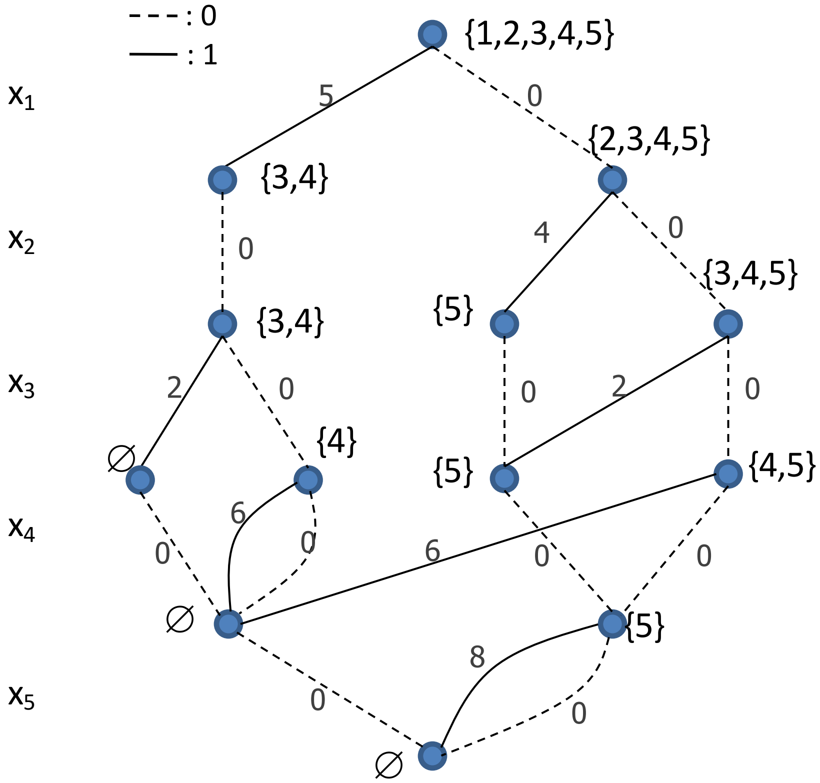
Maximum Independent Set Problem



Formal model: $OPT_i(S)$ represents optimal value of subproblem for selecting independent set from eligible S starting with i.

Q. How to define $OPT_i(S)$ recursively?

Maximum Independent Set Problem



Formal model: or $OPT_i(S)$ represents optimal value of subproblem for selecting independent set from eligible S starting with i.

$$OPT_i(S) = \begin{cases} max \left\{OPT_{i+1}(S \setminus \{i\}),\ OPT_{i+1}\big(S \setminus N(i)\big) + w_i\right\}, & i \in S \\ OPT_{i+1}(S), & otherwise \end{cases}$$

$$OPT_i(\emptyset) = 0, \text{for } \ i = 1, \ldots, n$$

where

$N(i)$ = {i} and its neighbors

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, van Hoeve, Hooker, IJOC 2014]
- TSP: parameterized-size depending on precedence relations

  [Cire & van Hoeve, OR 2013]

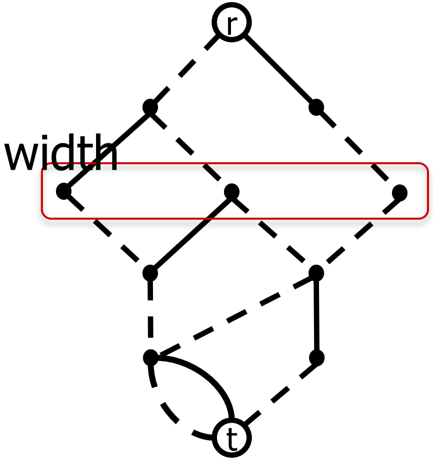
Next topics today:

- Relaxed diagrams -> upper bounds (infeasible)
- Restricted diagrams -> lower bounds (feasible)
- Alternative to branch & bound

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:

$$OPT_i(S) = \begin{cases} max\,\{OPT_{i+1}(S \setminus \{i\}),\ OPT_{i+1}\big(S \setminus N(i)\big) + w_i\}, & i \in S \\ OPT_{i+1}(S), & otherwise \end{cases}$$

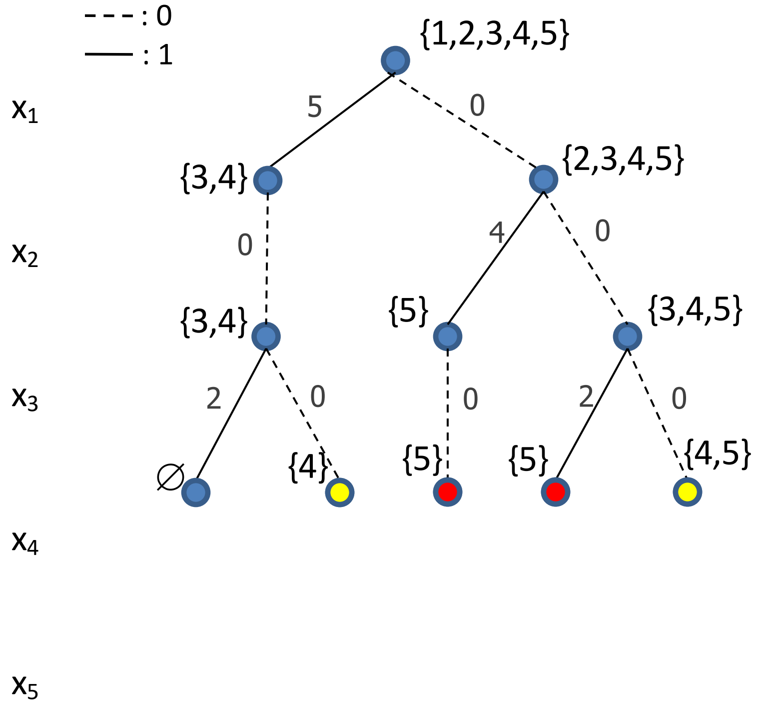
$$OPT_i(\emptyset) = 0, \text{for } i = 1, \dots, n$$

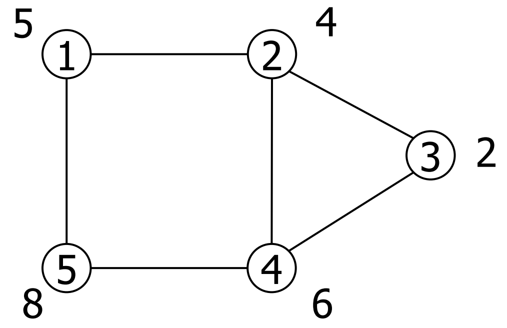
$$\oplus\,(S_1, S_2) = S_1 \cup S_2$$

Observations on this state aggregation operator:

- It allows infeasible solutions
- It does not remove feasible solutions
- Potential for heuristic use

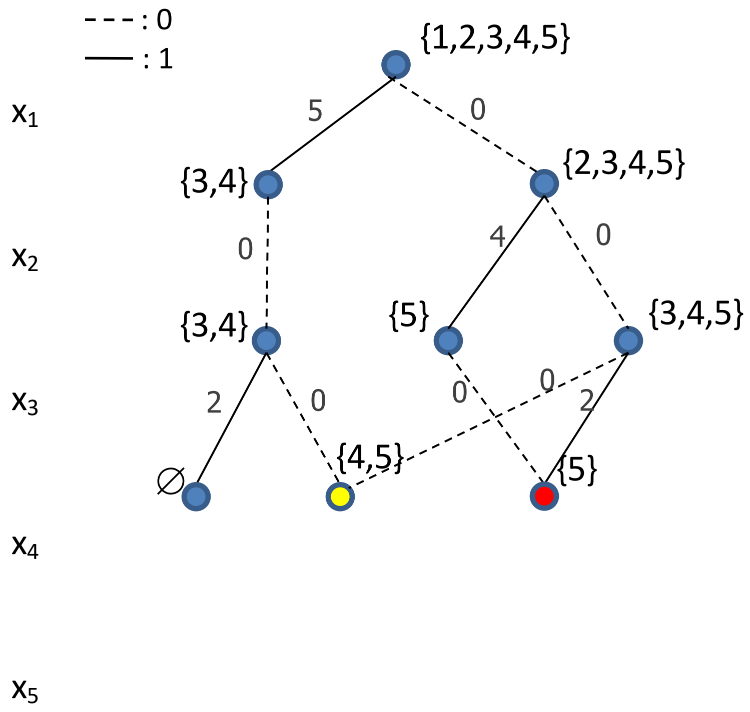
Independent Set Problem: Relaxed DD

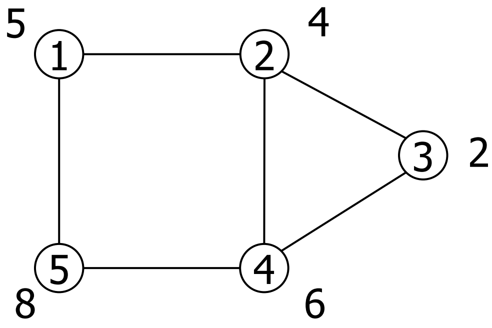




Maximum width = 3

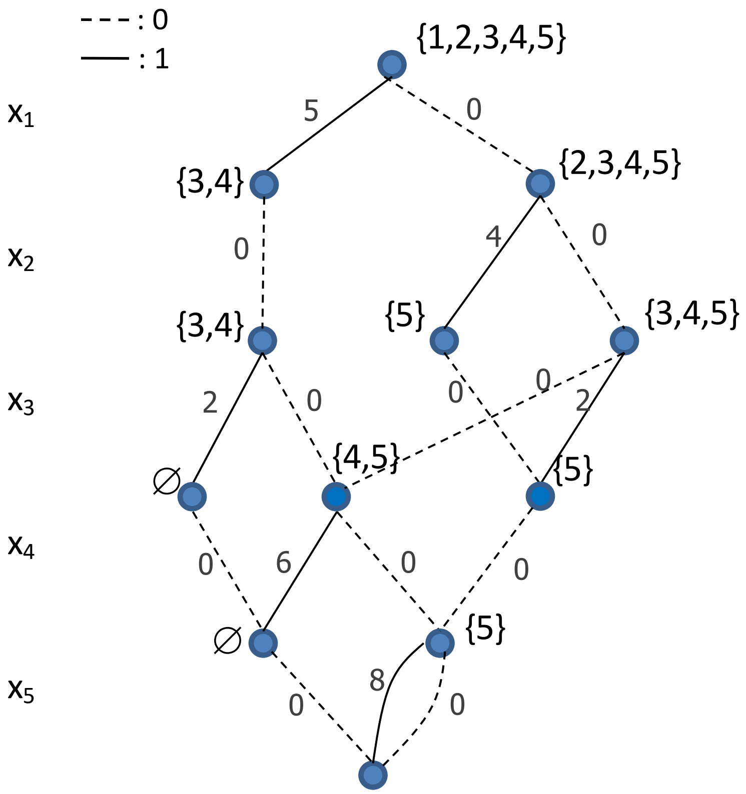
Independent Set Problem: Relaxed DD

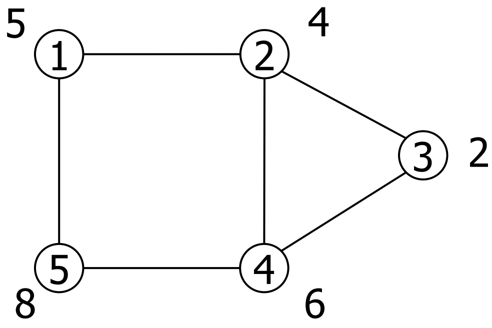




Maximum width = 3

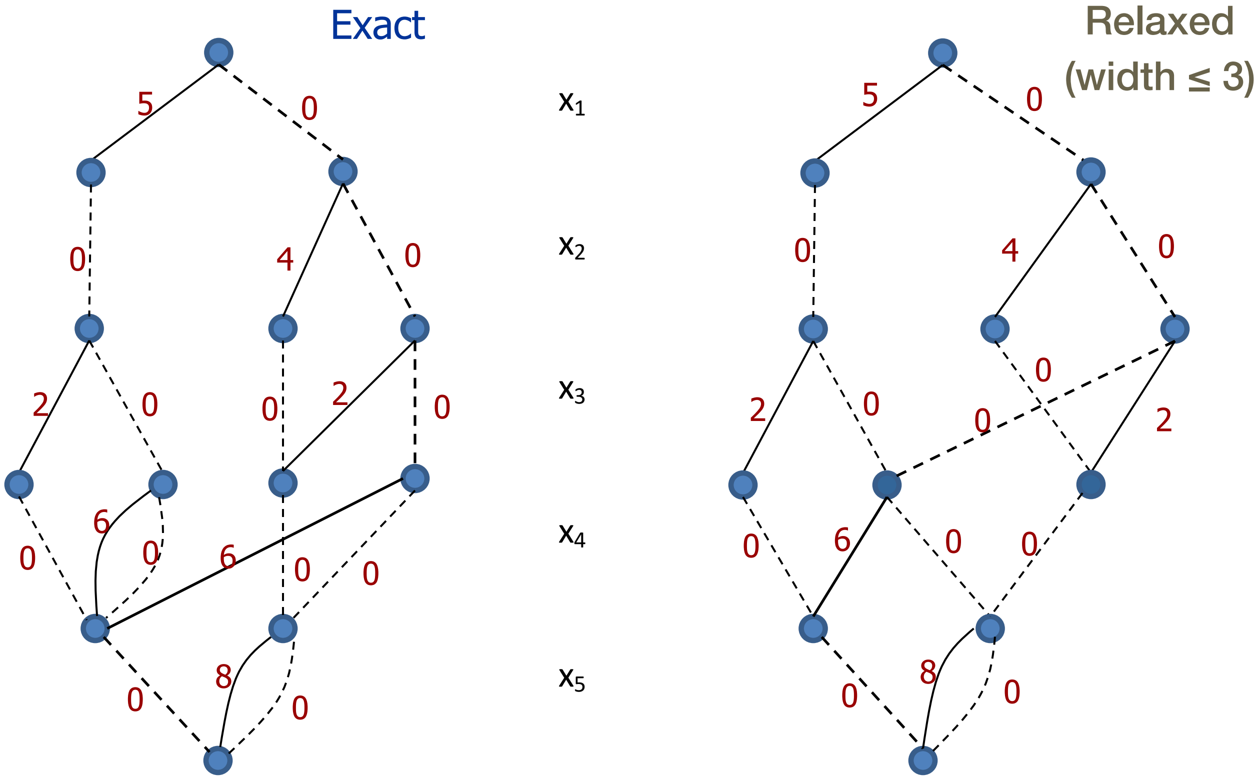
Independent Set Problem: Relaxed DD

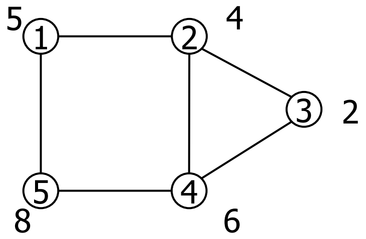




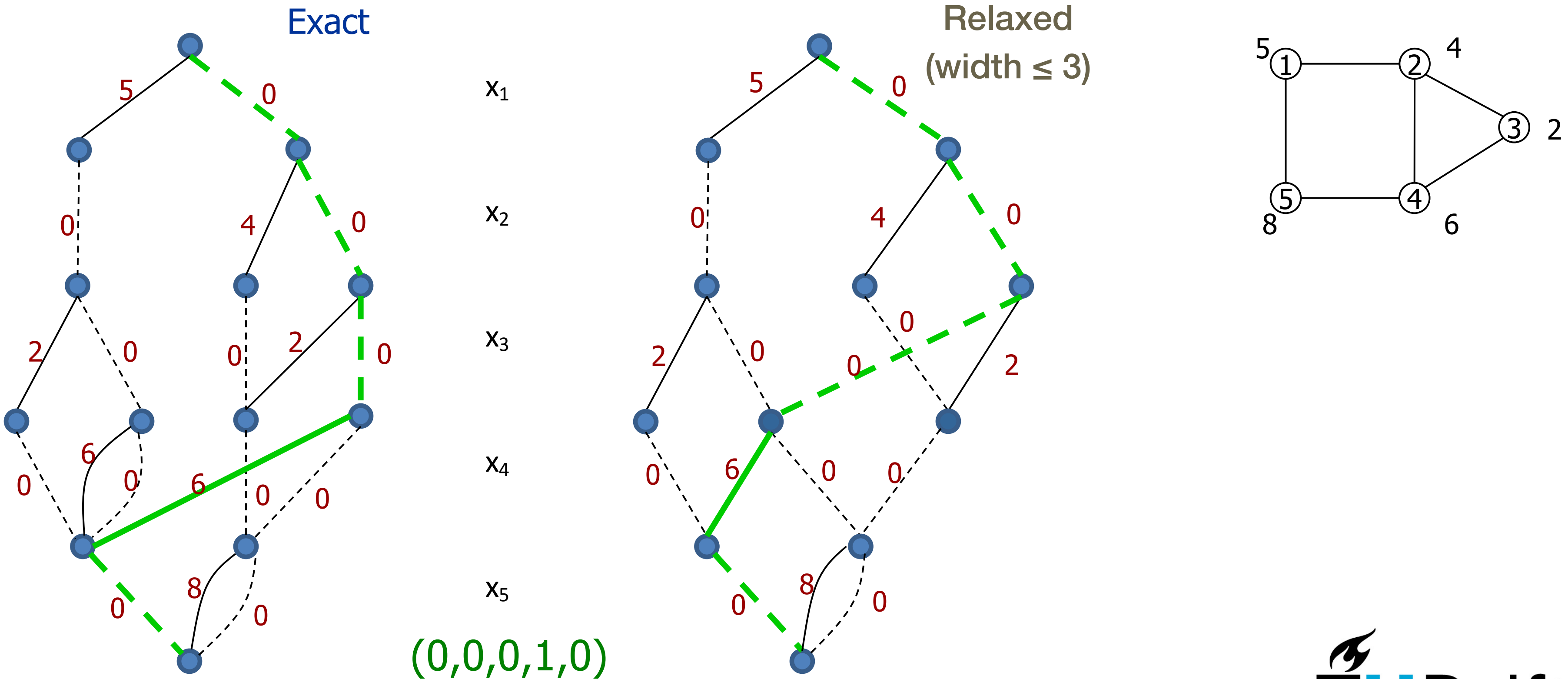
Maximum width = 3

Exact vs. Relaxed Decision Diagrams

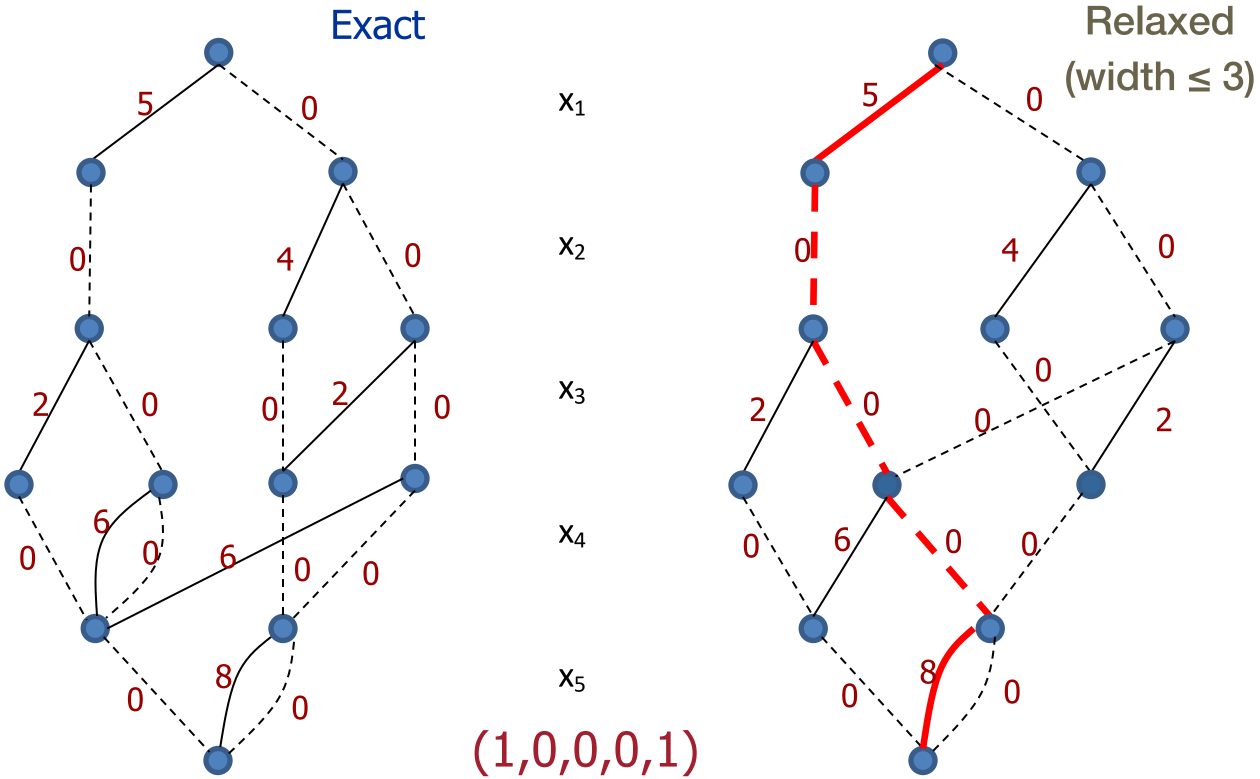


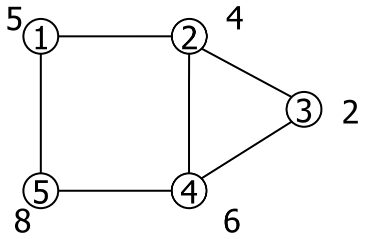


Exact vs. Relaxed Decision Diagrams

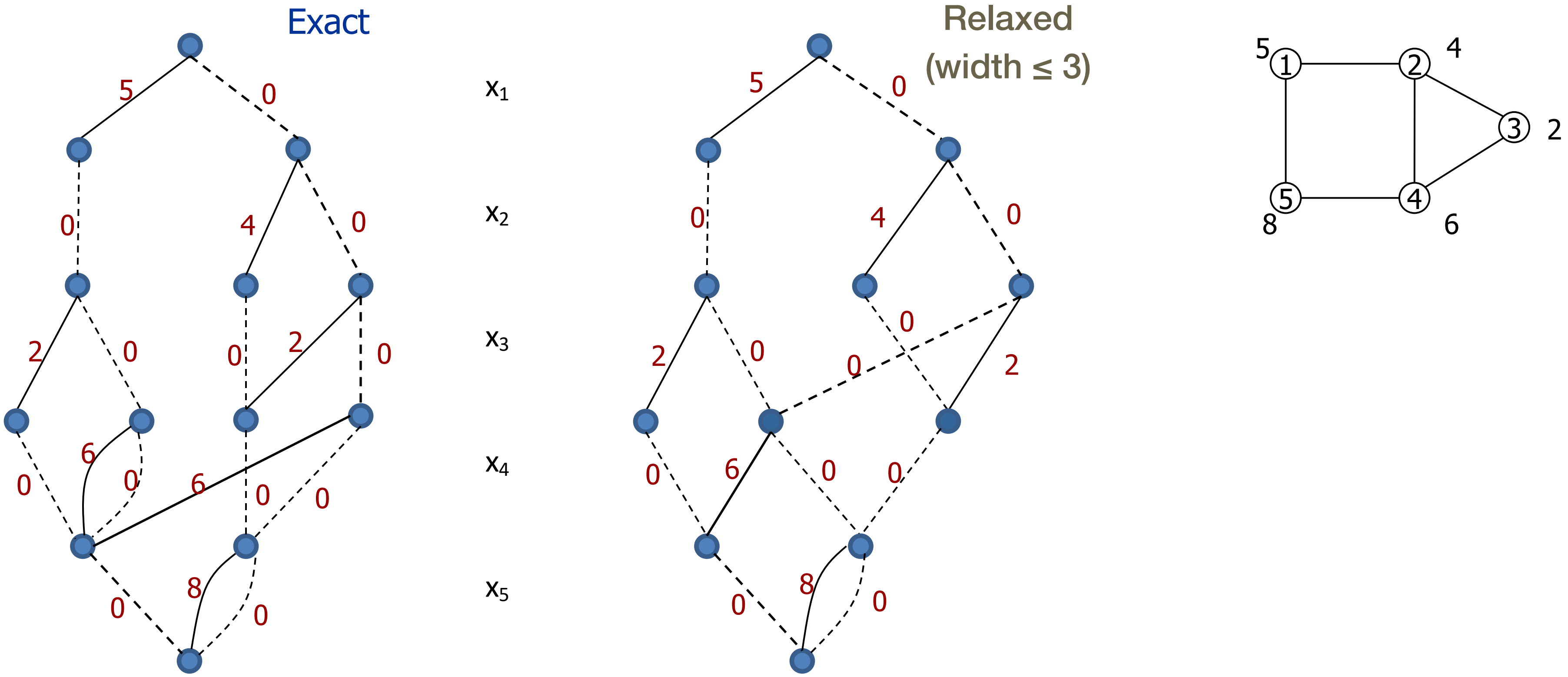


Exact vs. Relaxed Decision Diagrams



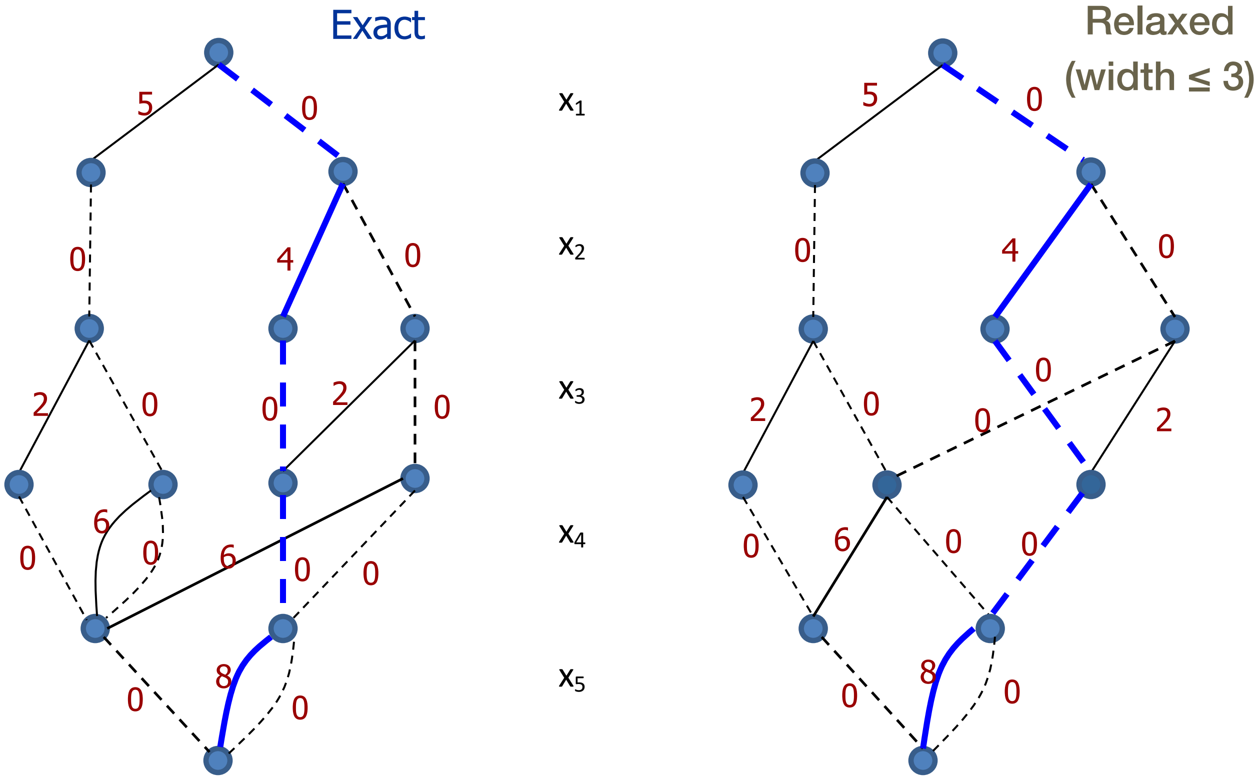


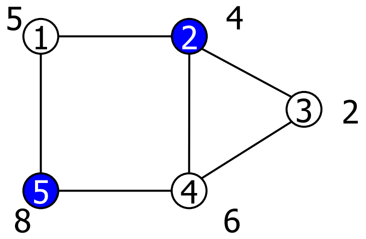
Exact vs. Relaxed Decision Diagrams



Q. What is the length of the longest path in the exact diagram?

Exact vs. Relaxed Decision Diagrams



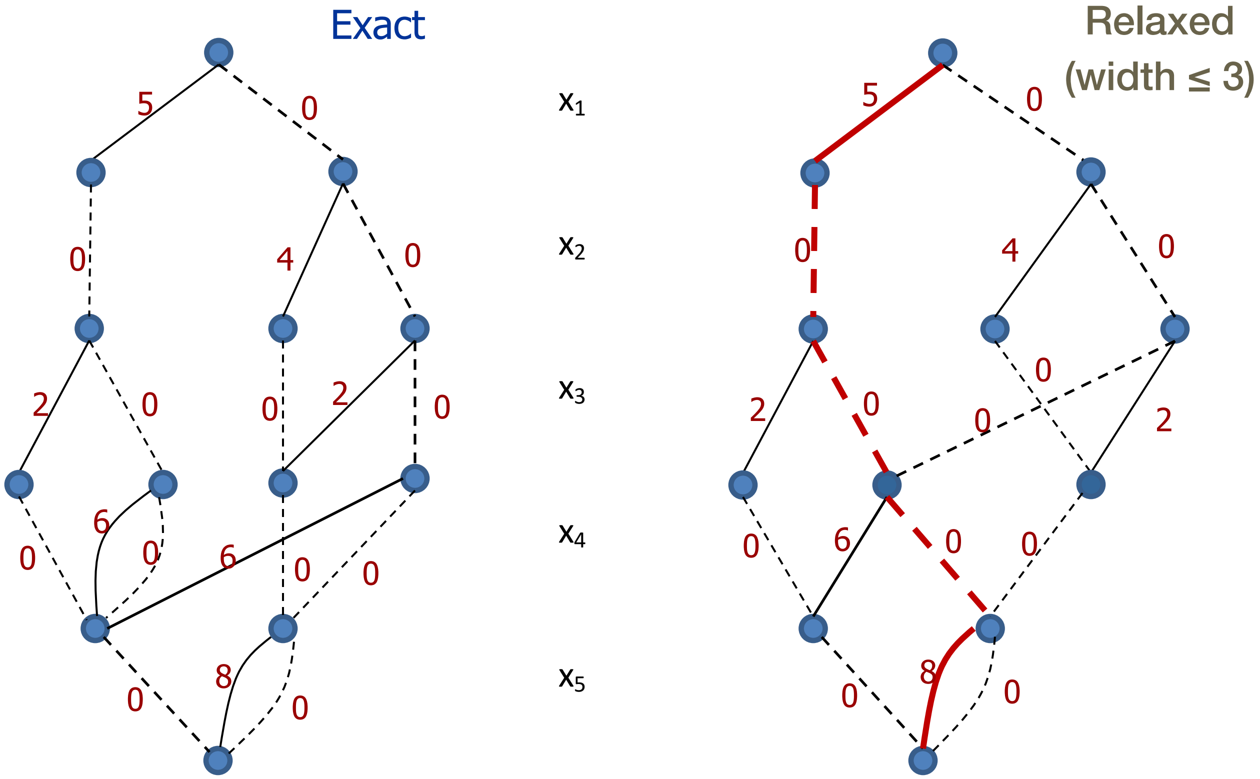


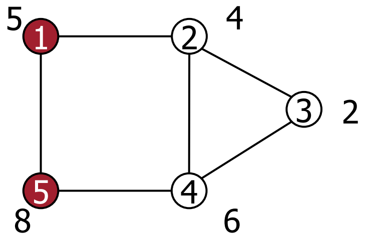
$x = (0, 1, 0, 0, 1)$

Solution value = 12

Q. What is the length of the longest path in the relaxed diagram?

Exact vs. Relaxed Decision Diagrams



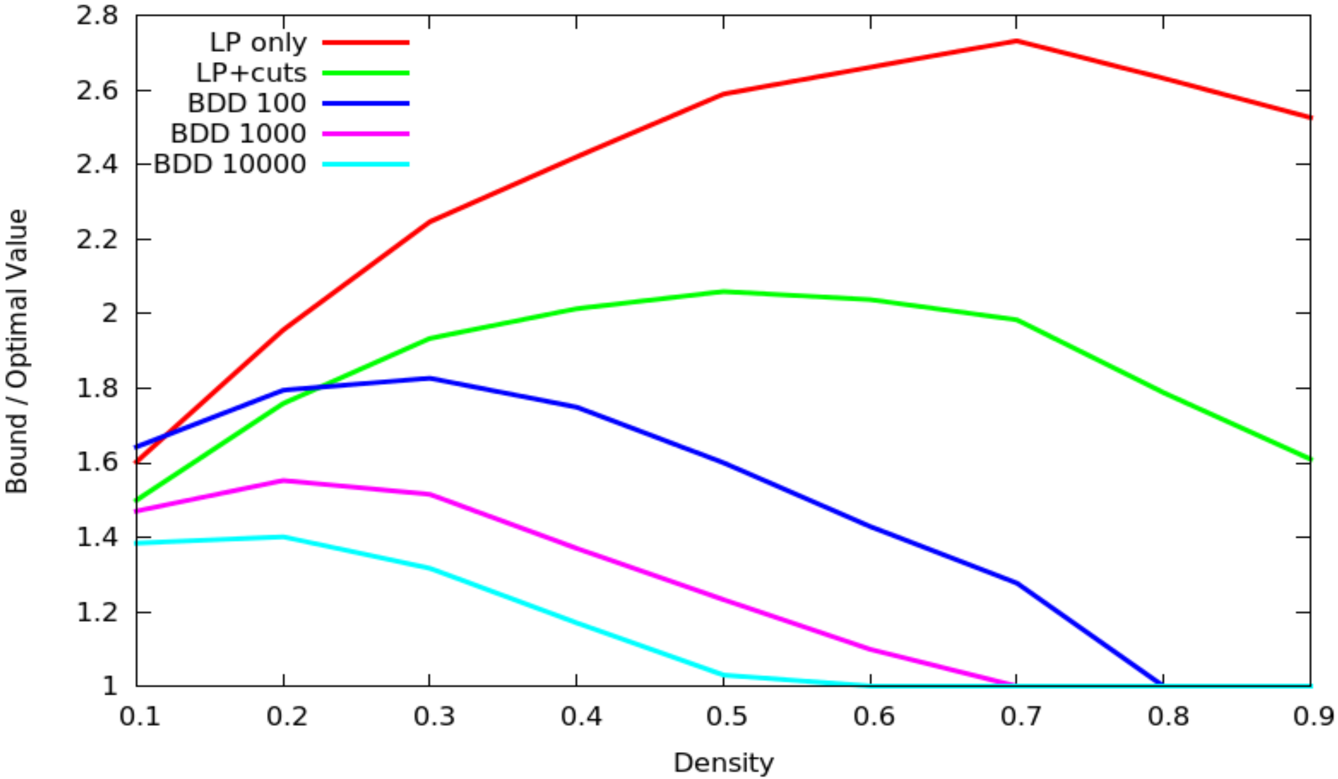


x = (1, 0, 0, 0, 1)

Upper bound = 13

Q. What is the length of the longest path in the relaxed diagram?

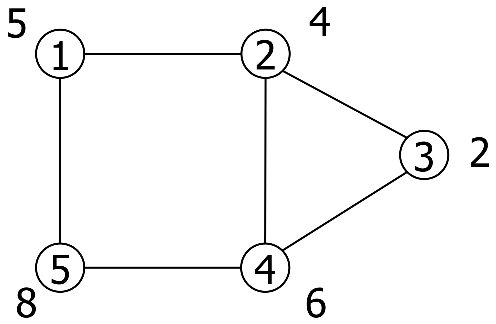
Relaxation Bound: Independent Set



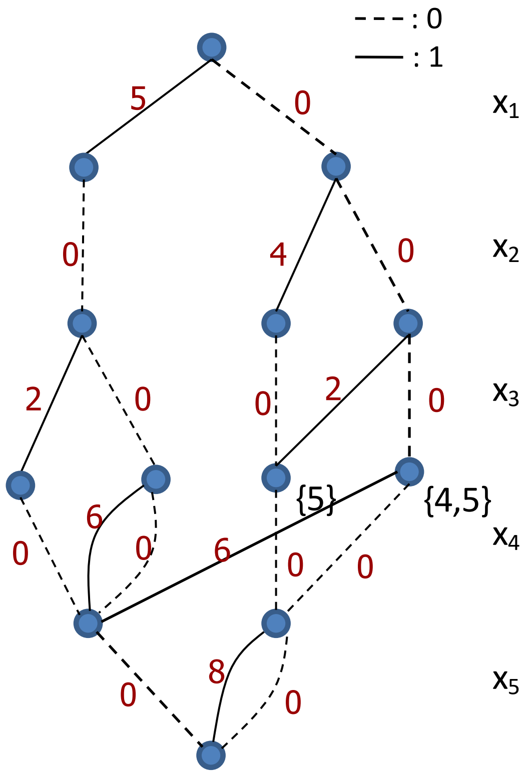
Compare with LP bound (CPLEX)

Restricted Decision Diagrams

- **Under-approximation of the feasible set**



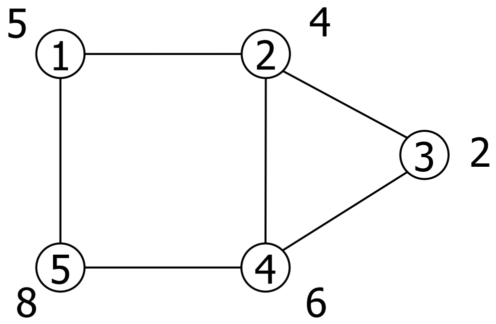
Maximum width = 3



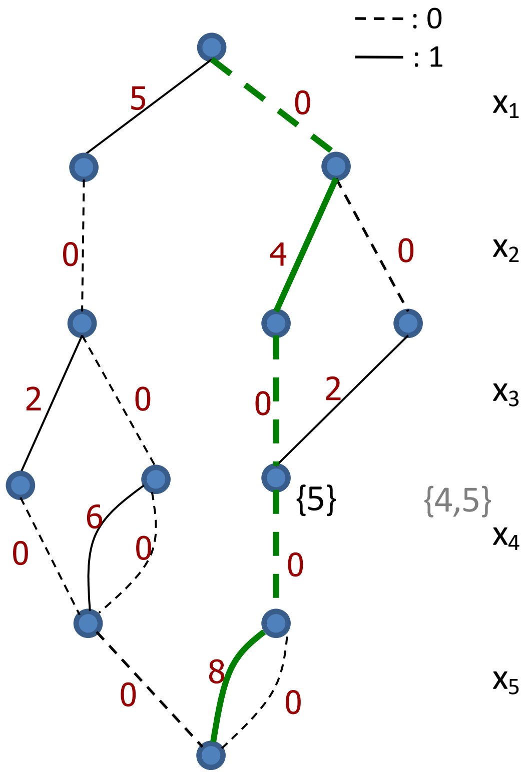
[Bergman, Cire, van Hoeve, Yunes, J Heur. 2014]

Restricted Decision Diagrams

- **Under-approximation of the feasible set**



Maximum width = 3

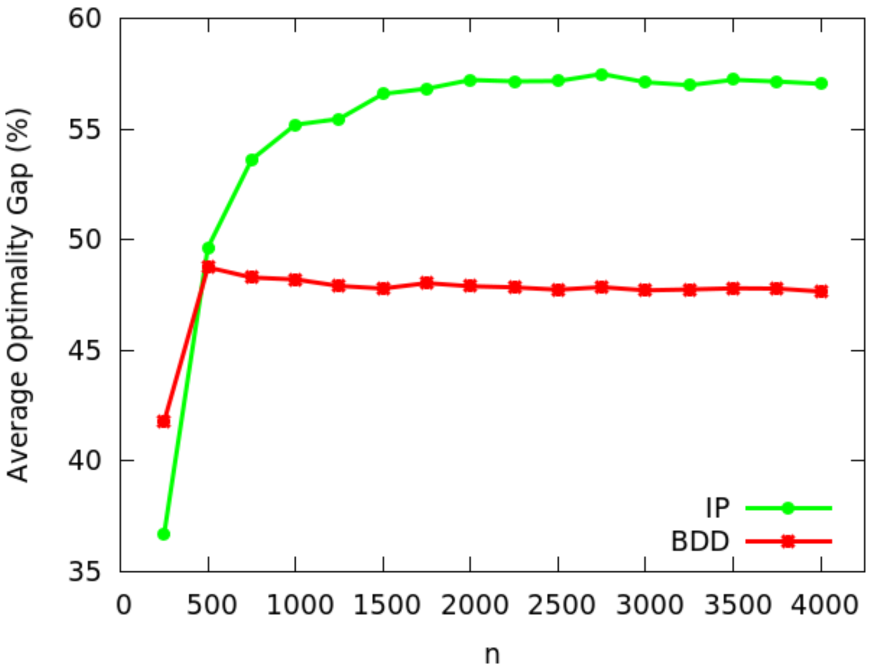


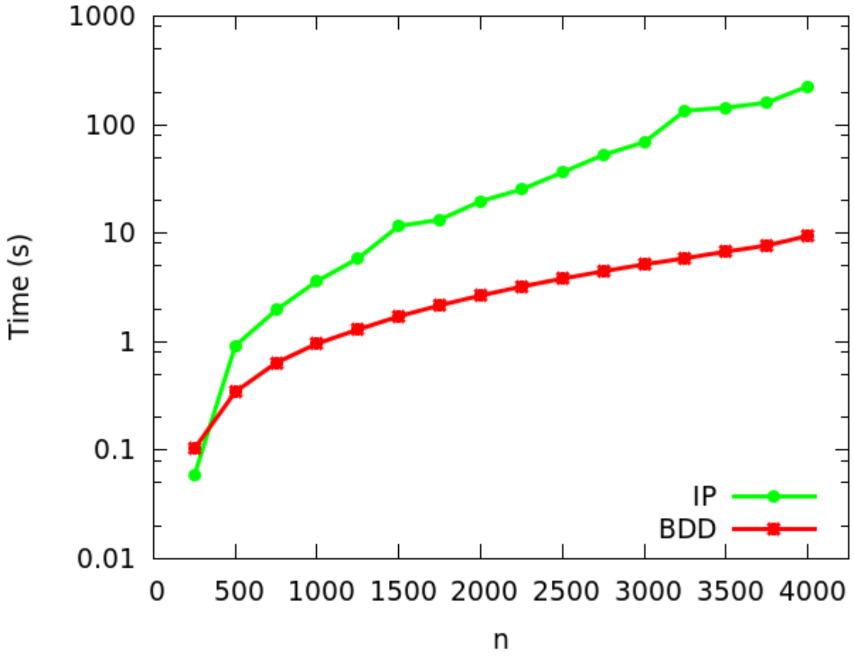
x = (0, 1, 0, 0, 1)

Lower bound = 12

[Bergman, Cire, van Hoeve, Yunes, J Heur. 2014]

Primal Bound: Set Covering Problem





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

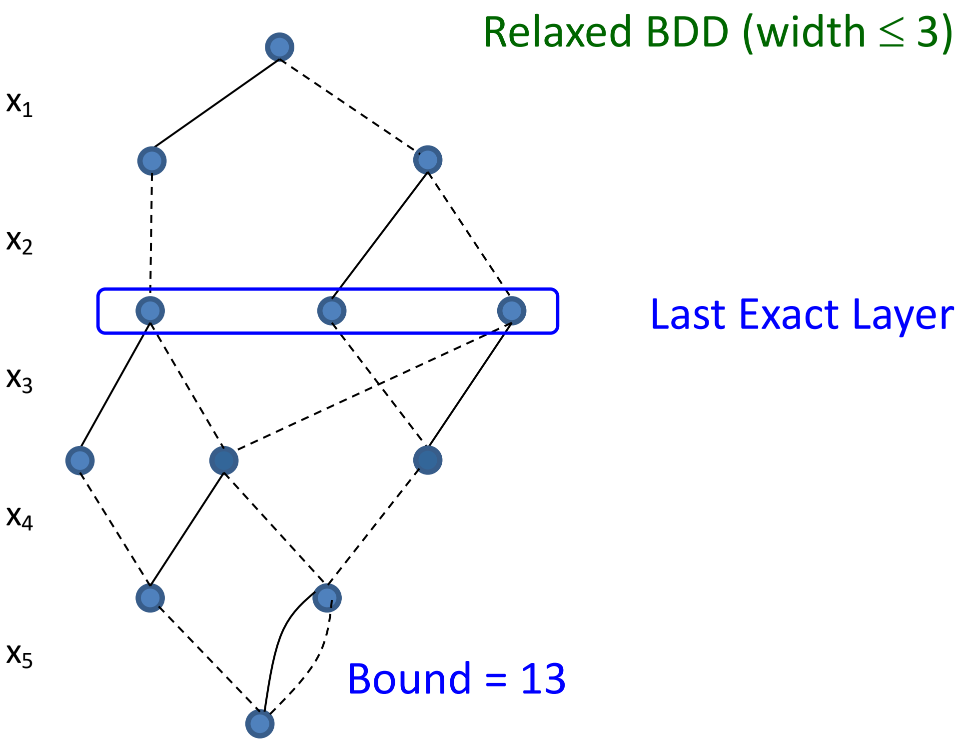
Branching is done on the *nodes* of the diagram

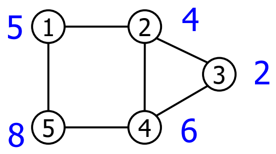
- Branching on pools of partial solutions
- Eliminate search symmetry

[Bergman, Cire, van Hoeve, Hooker, IJOC 2016]

TU Delft

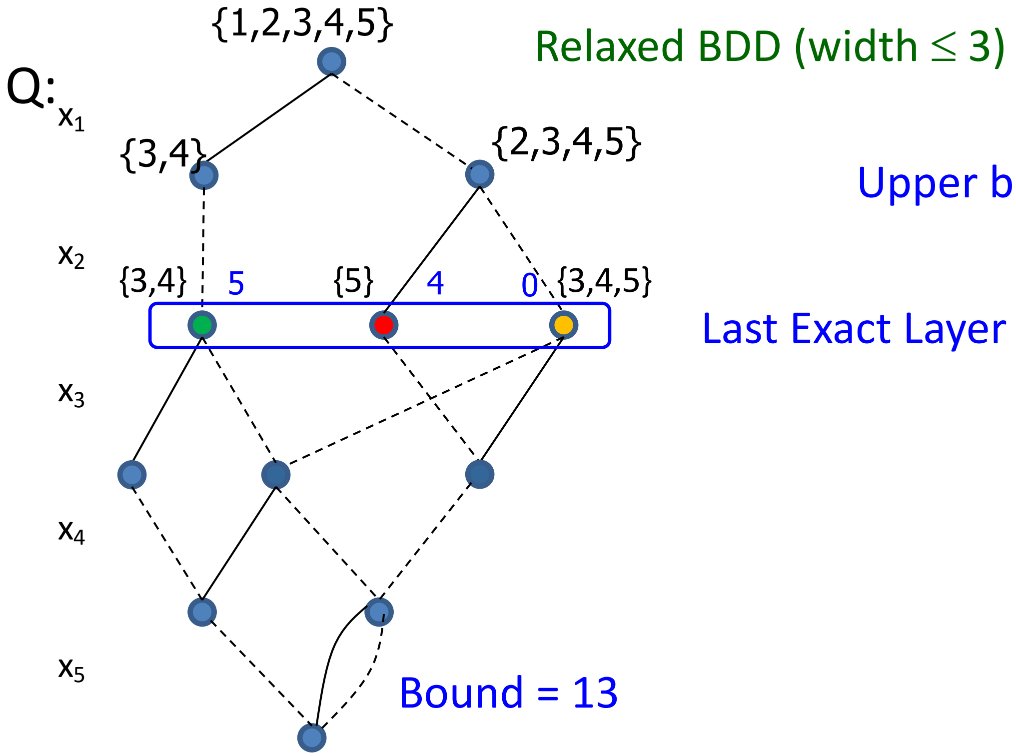
Branch and Bound



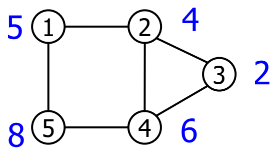


Node Queue

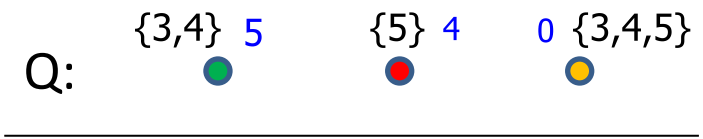
Relaxed BDD (width ≤ 3)



Upper bound = 13

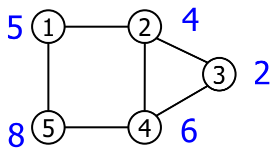


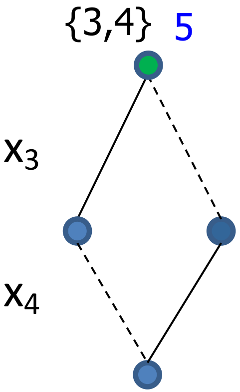
Node Queue



Upper bound = 13

Lower bound = 11



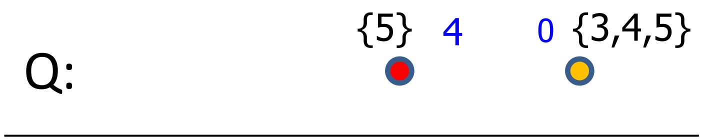


Exact solution: 11

Queue of nodes with exact paths; take one

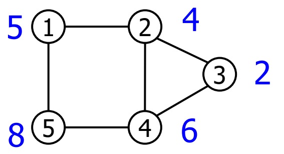
- if relaxed diagram leads to upper bound below lower bound, prune; otherwise:
- build restricted diagram of subproblem to find (possibly) better lower bound, and
- build relaxed diagram to find new "exact layer", add to Q

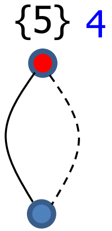
Node Queue



Upper bound = 13

Lower bound = 12





Exact solution: 12

Queue of nodes with exact paths; take one

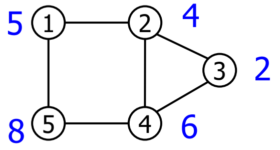
- if relaxed diagram leads to upper bound below lower bound, prune; otherwise:
- build restricted diagram of subproblem to find (possibly) better lower bound, and
- build relaxed diagram to find new "exact layer", add to Q

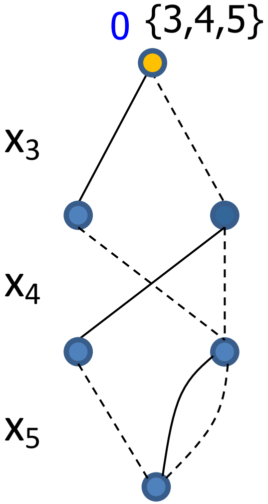
Node Queue

Q:

Upper bound = 13

Lower bound = 12





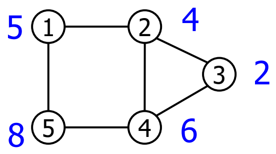
Exact solution/ upper bound: 10

Queue of nodes with exact paths; take one

- if relaxed diagram leads to upper bound below lower bound, prune; otherwise:
- build restricted diagram of subproblem to find (possibly) better lower bound, and
- build relaxed diagram to find new "exact layer", add to Q

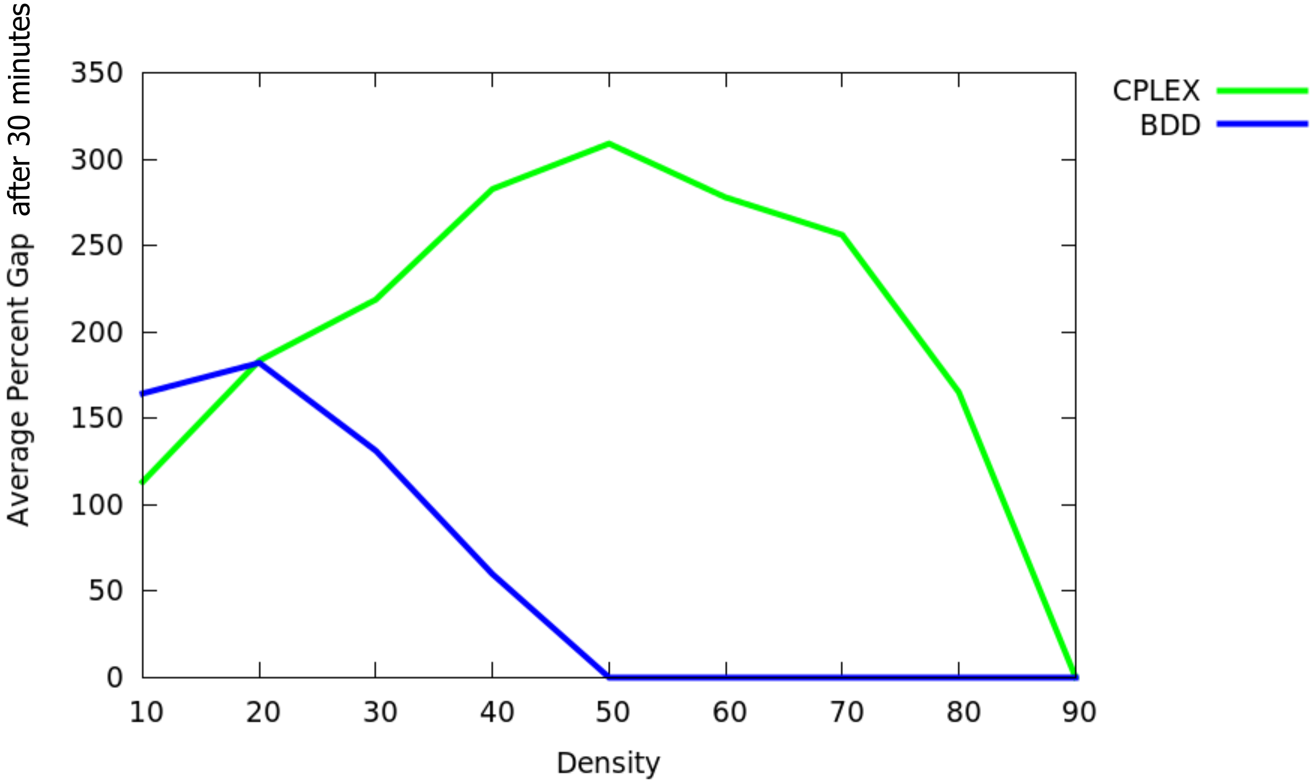
Node Queue

Q:



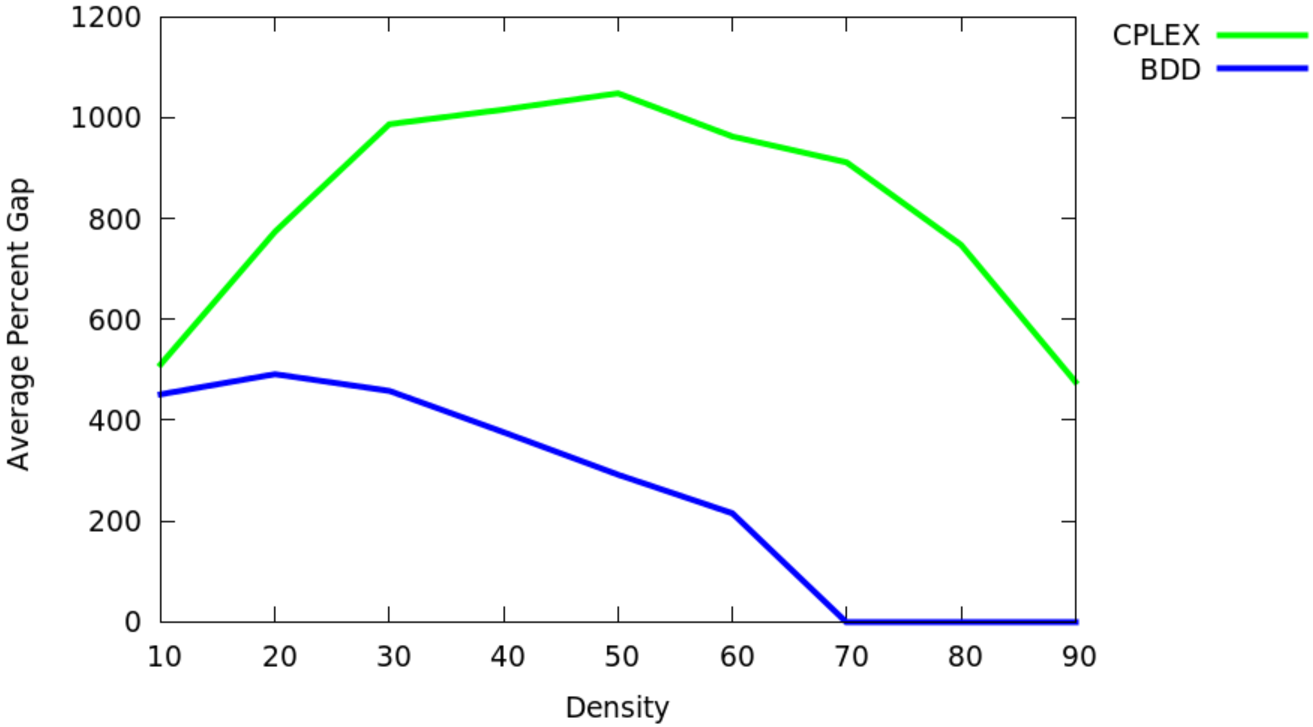
Optimal solution: 12

Maximum Independent Set: 500 variables



TU Delft

Maximum Independent Set: 1500 variables



Novel branching **scheme**

- 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]

1-Slide Summary on Decision Diagrams

A decision diagram (DD) represents *all feasible solutions* by defining a variable ordering, states, decisions and the state transition graph

- a path is a solution, the width is an indication of the size
- a reduced DD has equivalent nodes merged
- a decision diagram for maximum weight independent set is defined by:

$$OPT_i(S) = \begin{cases} max\left\{OPT_{i+1}(S \setminus \{i\}),\ OPT_{i+1}\left(S \setminus N(i)\right) + w_i\right\}, & i \in S \\ OPT_{i+1}(S), & otherwise \end{cases}$$

$$OPT_i(\emptyset) = 0, \text{for } \ i = 1, \dots, n$$

- relaxed and restricted diagrams are obtained by state merging / deletion
- merge operator for independent set is the union
- branch & bound can be done using decision diagrams instead of MILP

Study Advice

Please read

1. Chapters 3 and 4 of Bergman, D., Cire, A. A., Van Hoeve, W. J., & Hooker, J. (2016). *Decision diagrams for optimization*. New York, NY: Springer International Publishing. (Invited also to read later chapters, e.g. Ch.6 on the new branching.)

Homework assignment

- Describe a decision diagram for another problem (maximum weight over edges crossing a subset)