Optimization Bounds from Binary Decision Diagrams

J. N. Hooker

Joint work with
David Bergman, André Ciré, Willem-Jan van Hoeve

Carnegie Mellon University

Journal Presentation Track
CP 2014, Lyon, France

Based on…

D. Bergman, A.A. Ciré, W.-J. van Hoeve, J.N. Hooker,
Optimization bounds from binary decision diagrams,
INFORMS Journal on Computing **26** (2014) 253–268.

Binary Decision Diagrams

- **BDD**s historically used for circuit design and verification.
  - Lee 1959, Akers 1978, Bryant 1986.

Binary Decision Diagrams

- **BDD**s historically used for circuit design and verification.
  - Lee 1959, Akers 1978, Bryant 1986.
- **Compact** graphical representation of **boolean** function.
  - Can also represent **feasible set** of problem with binary variables.
  - Slight generalization (MDDs) represents **finite domain** variables.

Binary Decision Diagrams

- **BDD**s historically used for circuit design and verification.
  - Lee 1959, Akers 1978, Bryant 1986.
- **Compact** graphical representation of **boolean** function.
  - Can also represent **feasible set** of problem with binary variables.
  - Slight generalization (MDDs) represents **finite domain** variables.
- **Reduced** BDD is result of superimposing isomorphic subtrees in a search tree.
  - Unique reduced BDD for given variable ordering.

A little history…

- First CP workshop was in 1993 (first *conference* 1995)
  - I was there! My paper *competed* with BDDs.
    - Logic circuit verification by Benders decomposition
      The Newport Papers: Proceedings of PPCP 1993

A little history…

- First CP workshop was in 1993 (first conference 1995)
  - I was there! My paper *competed* with BDDs.
    - Logic circuit verification by Benders decomposition
      The Newport Papers: Proceedings of PPCP 1993
- First International Conference on AI and OR (1995)
  - *Organizers:* M. Ginsberg, JNH
  - Several CP papers

A little history…

- First CP workshop was in 1993 (first conference 1995)
  - I was there! My paper *competed* with BDDs.
    - Logic circuit verification by Benders decomposition
      The Newport Papers: Proceedings of PPCP 1993
- First International Conference on AI and OR (1995)
  - *Organizers:* M. Ginsberg, JNH
  - Several CP papers
- First CPAIOR workshop (1999)
  - *Organizers:* F. Focacci, A. Lodi, M. Milano, D. Vigo

Binary Decision Diagrams

- Goal: derive bound on optimal value from BDD.
  - Good bounds are essential to optimization.

Binary Decision Diagrams

- Goal: derive bound on optimal value from BDD.
  - Good bounds are essential to optimization.
- BDD can grow exponentially with problem size.
  - So we use a smaller, **relaxed** BDD that represents **superset** of feasible set.
    - Andersen, Hadžić, Hooker, Tiedemann 2007.
  - Introduced as alternative to domain store in CP.
    - For alldiff systems, reduced search tree from >1 million nodes to 1 node.
    - Subsequent papers by…
      - Bergman, Ciré, Hadžić, Hoda, van Hoeve, Hooker, Kell, O'Sullivan, Sabharwal, Samulowitz, Saraswat, Tiedemann, Yunes

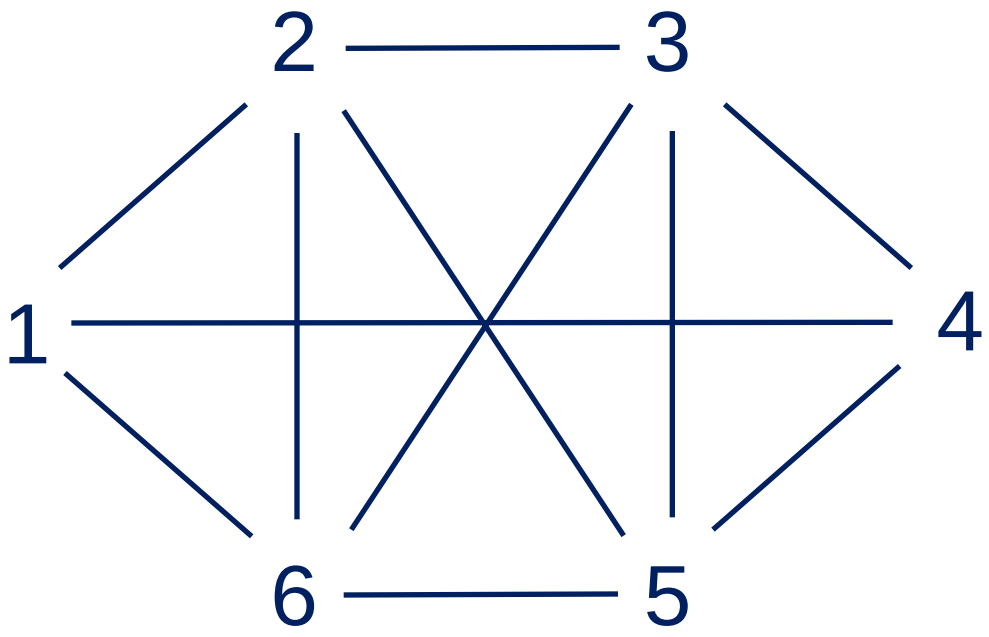
Binary Decision Diagrams

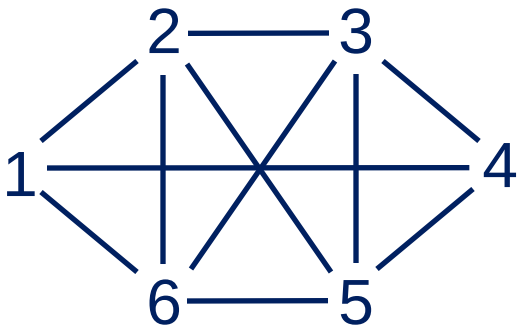
- Goal: derive bound on optimal value from BDD.
  - Good bounds are essential to optimization.
- BDD can grow exponentially with problem size.
  - So we use a smaller, **relaxed** BDD that represents **superset** of feasible set.
    - Andersen, Hadžić, Hooker, Tiedemann 2007.
  - Introduced as alternative to domain store in CP.
    - For alldiff systems, reduced search tree from >1 million nodes to 1 node.
    - Subsequent papers by…
      - Bergman, Ciré, Hadžić, Hoda, van Hoeve, Hooker, Kell, O'Sullivan, Sabharwal, Samulowitz, Saraswat, Tiedemann, Yunes
- We focus on **stable set problem** on a graph…

Stable Set Problem

Let each vertex have weight w_i

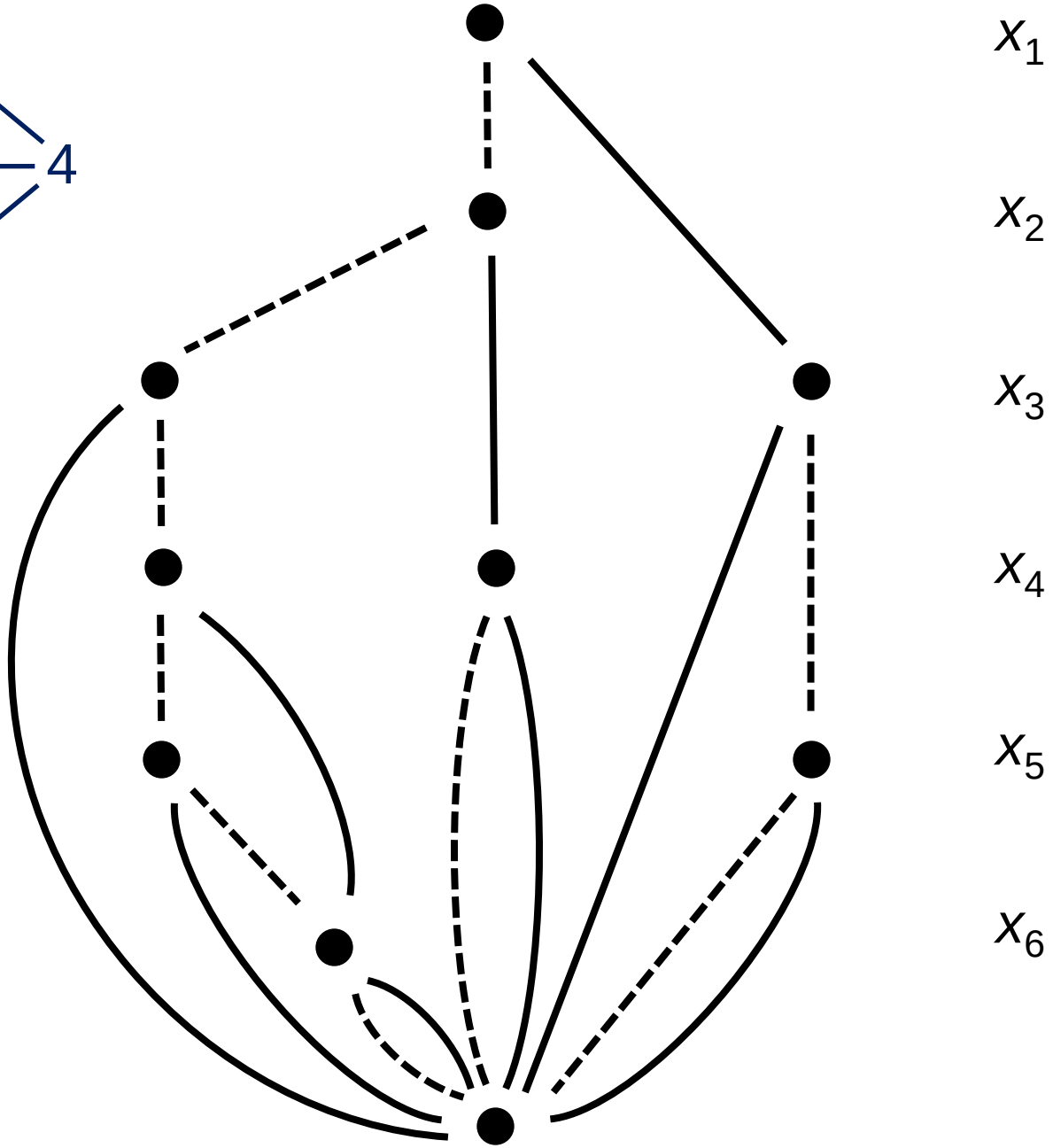
Select set of nonadjacent vertices to maximize $\sum_i w_i x_i$

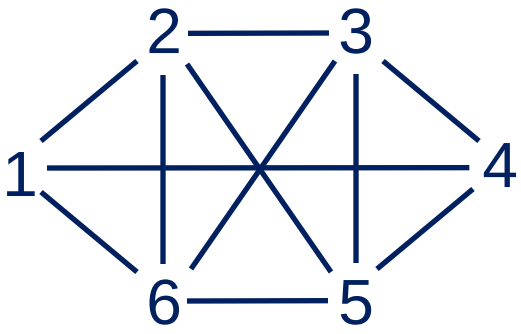




Exact BDD for stable set problem

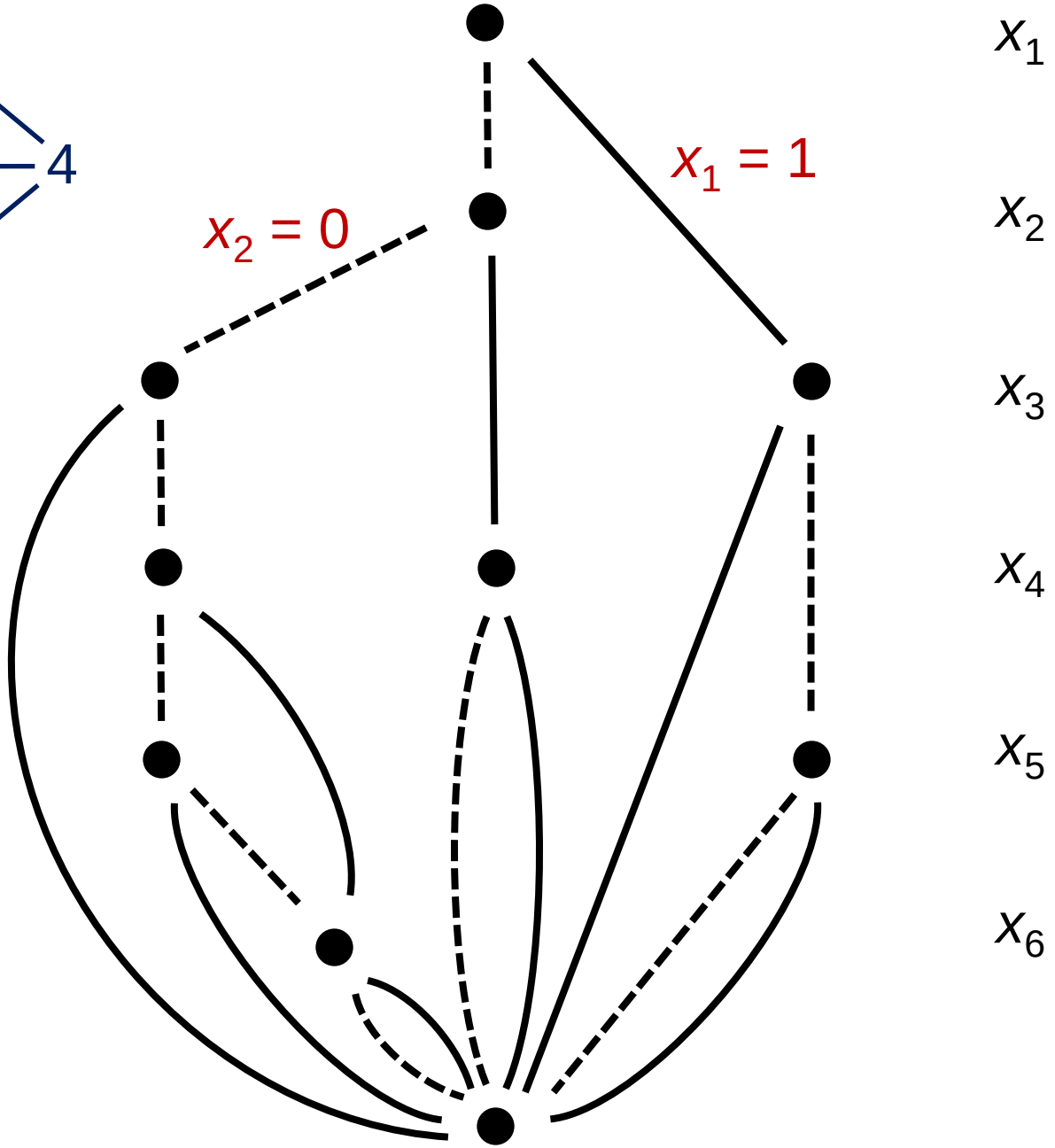
"zero-suppressed" BDD

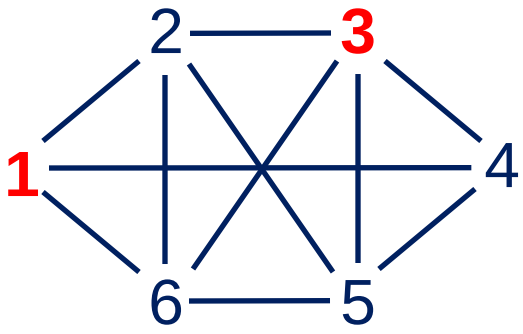




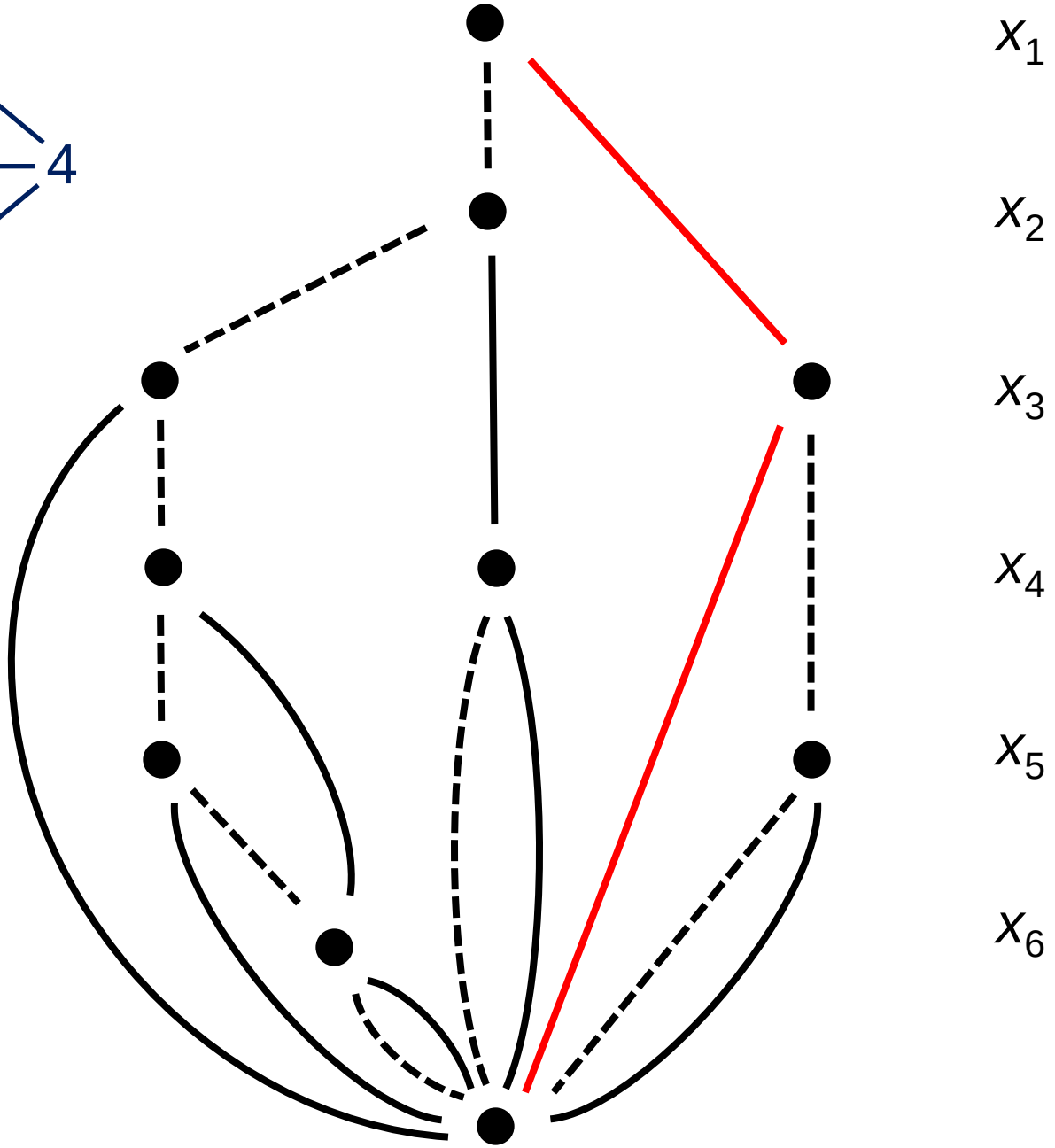
Exact BDD for stable set problem

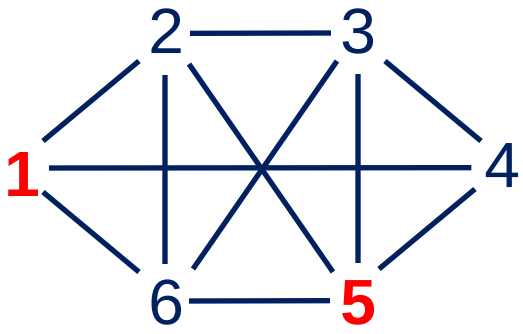
"zero-suppressed" BDD



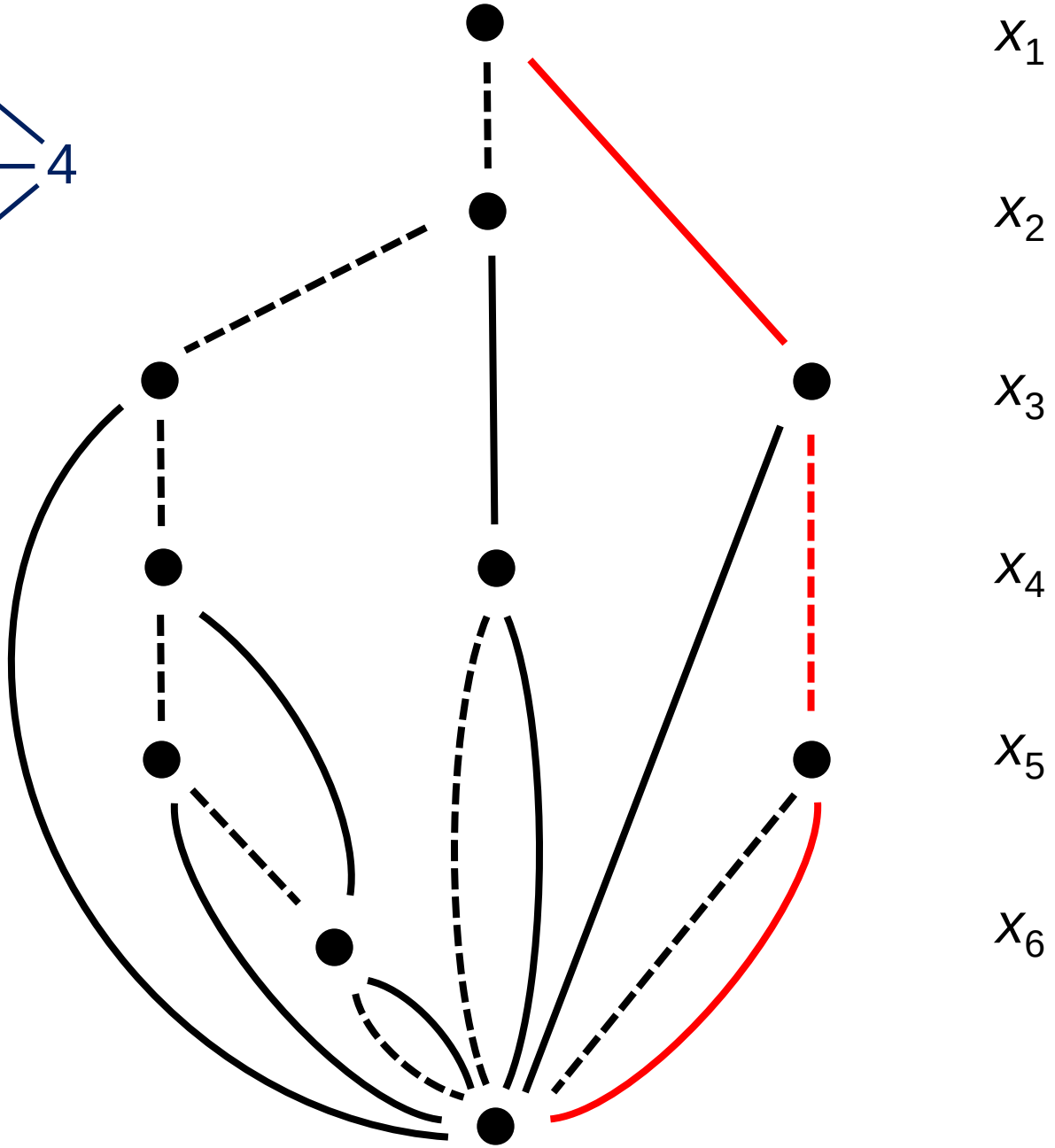


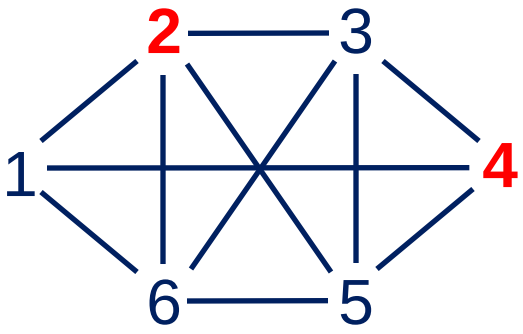
Paths from top to bottom correspond to the 11 feasible solutions



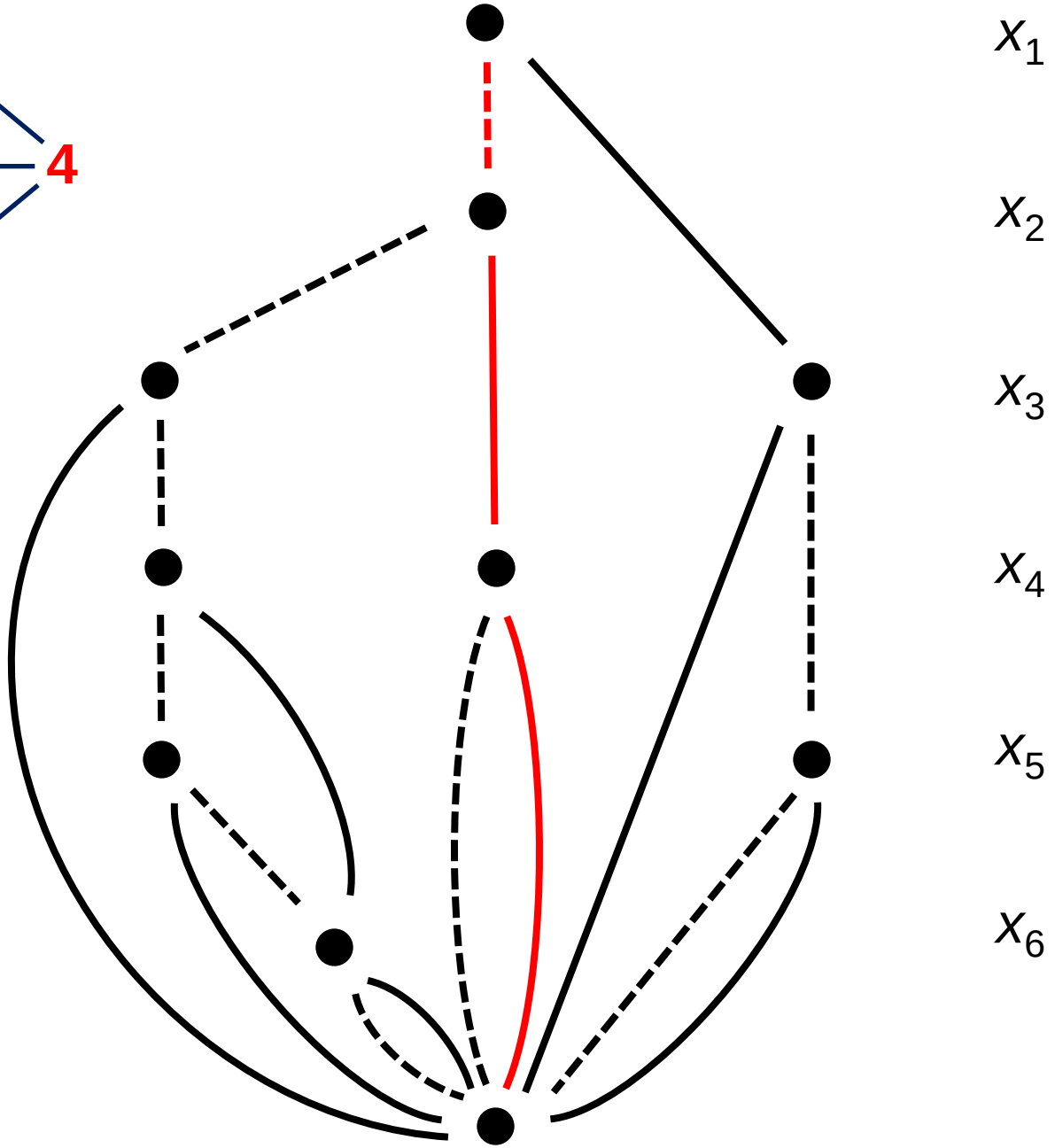


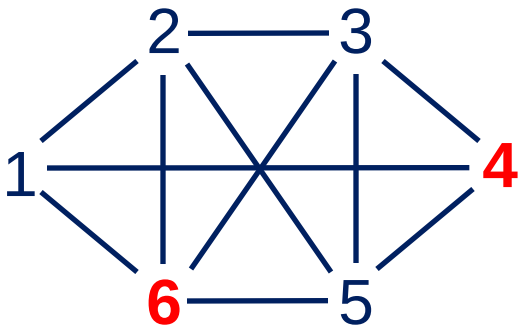
Paths from top to bottom correspond to the 11 feasible solutions



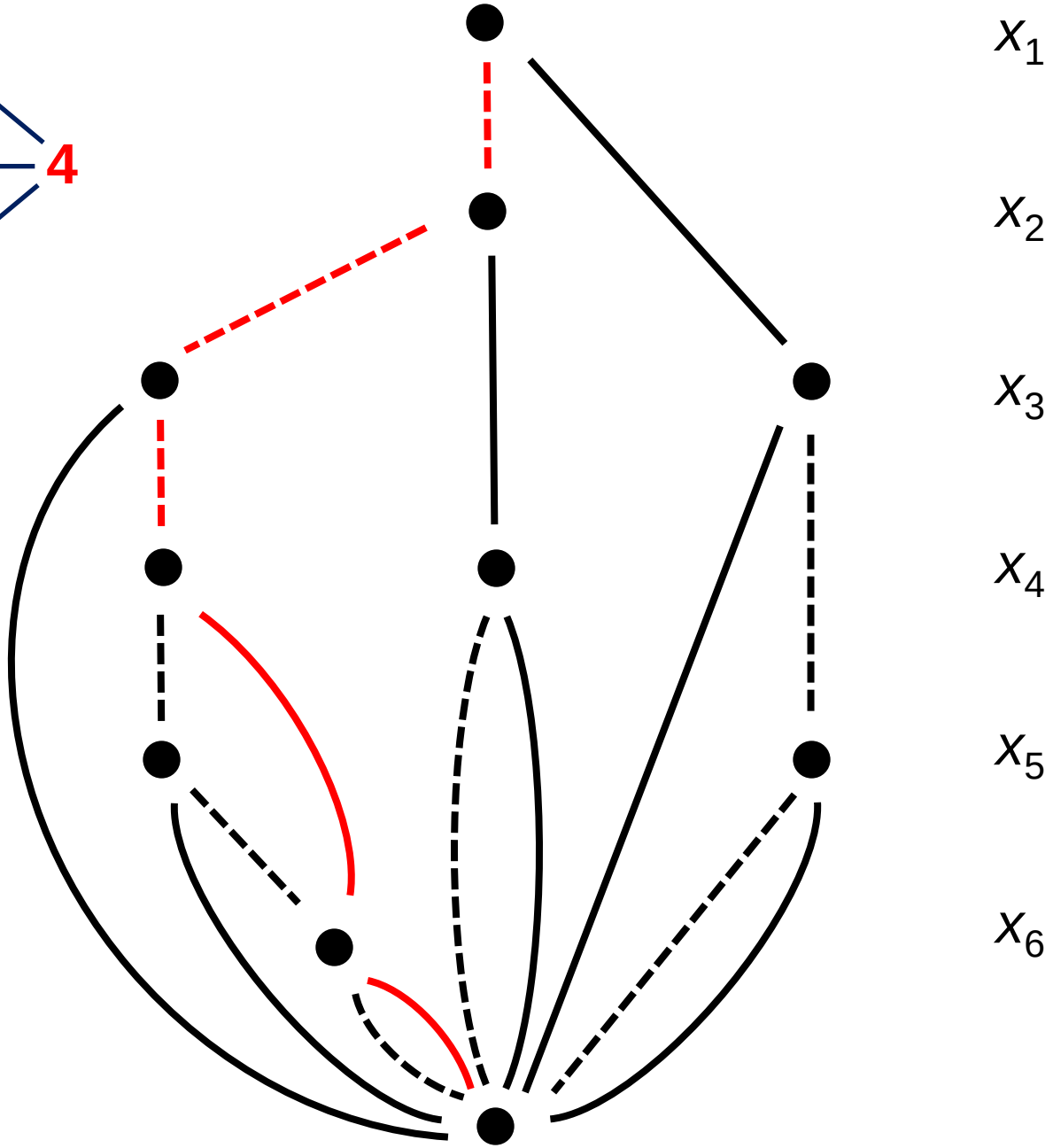


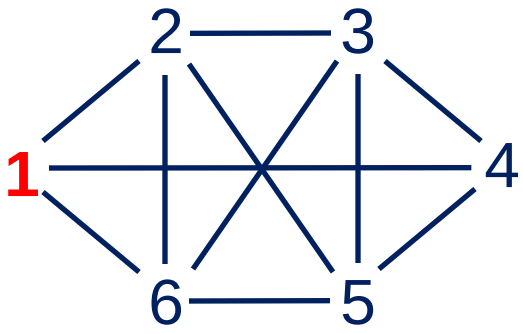
Paths from top to bottom correspond to the 11 feasible solutions





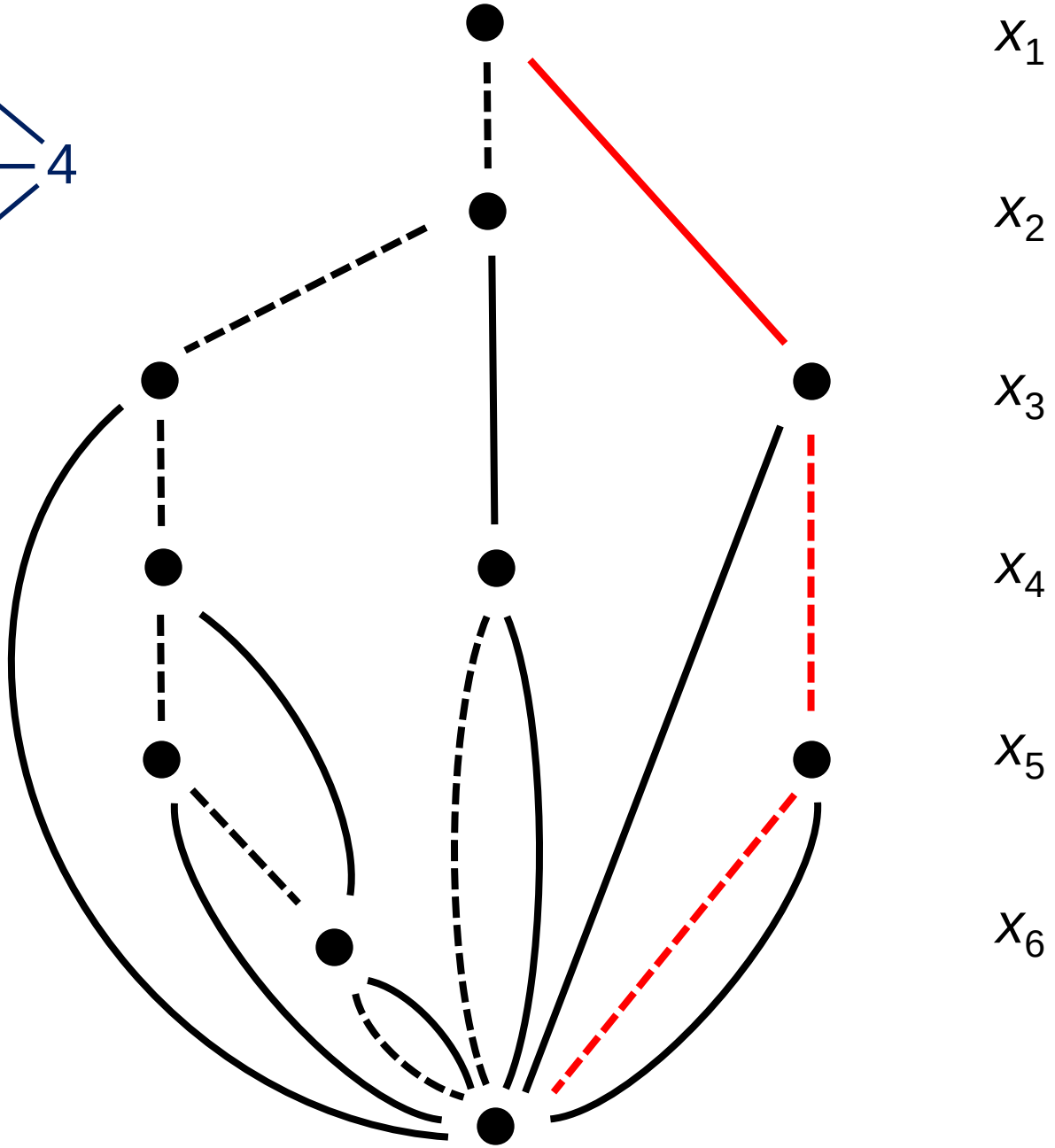
Paths from top to bottom correspond to the 11 feasible solutions

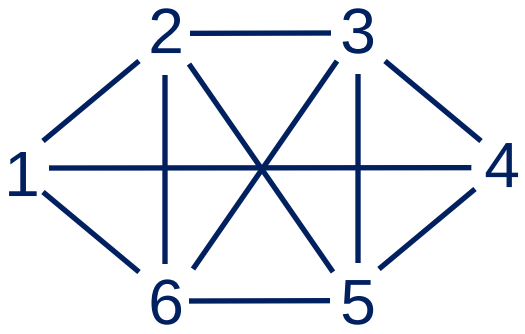




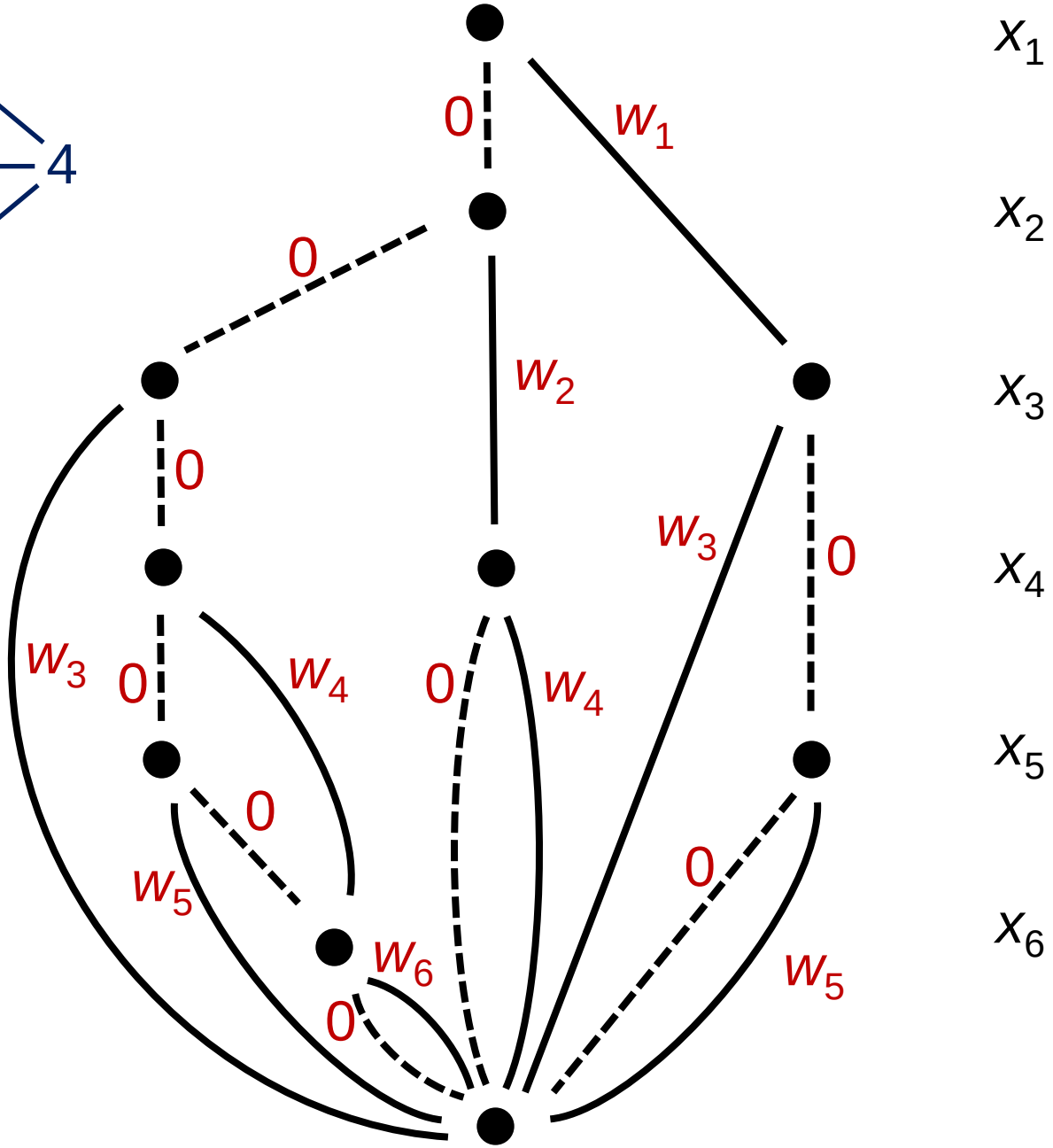
Paths from top to bottom correspond to the 11 feasible solutions

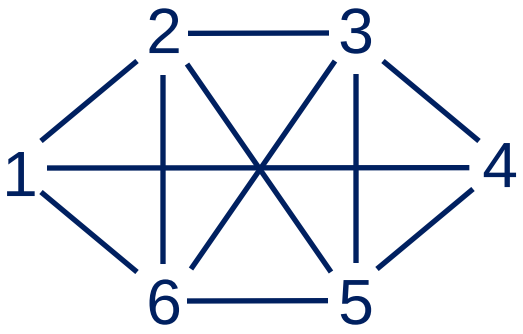
…and so forth





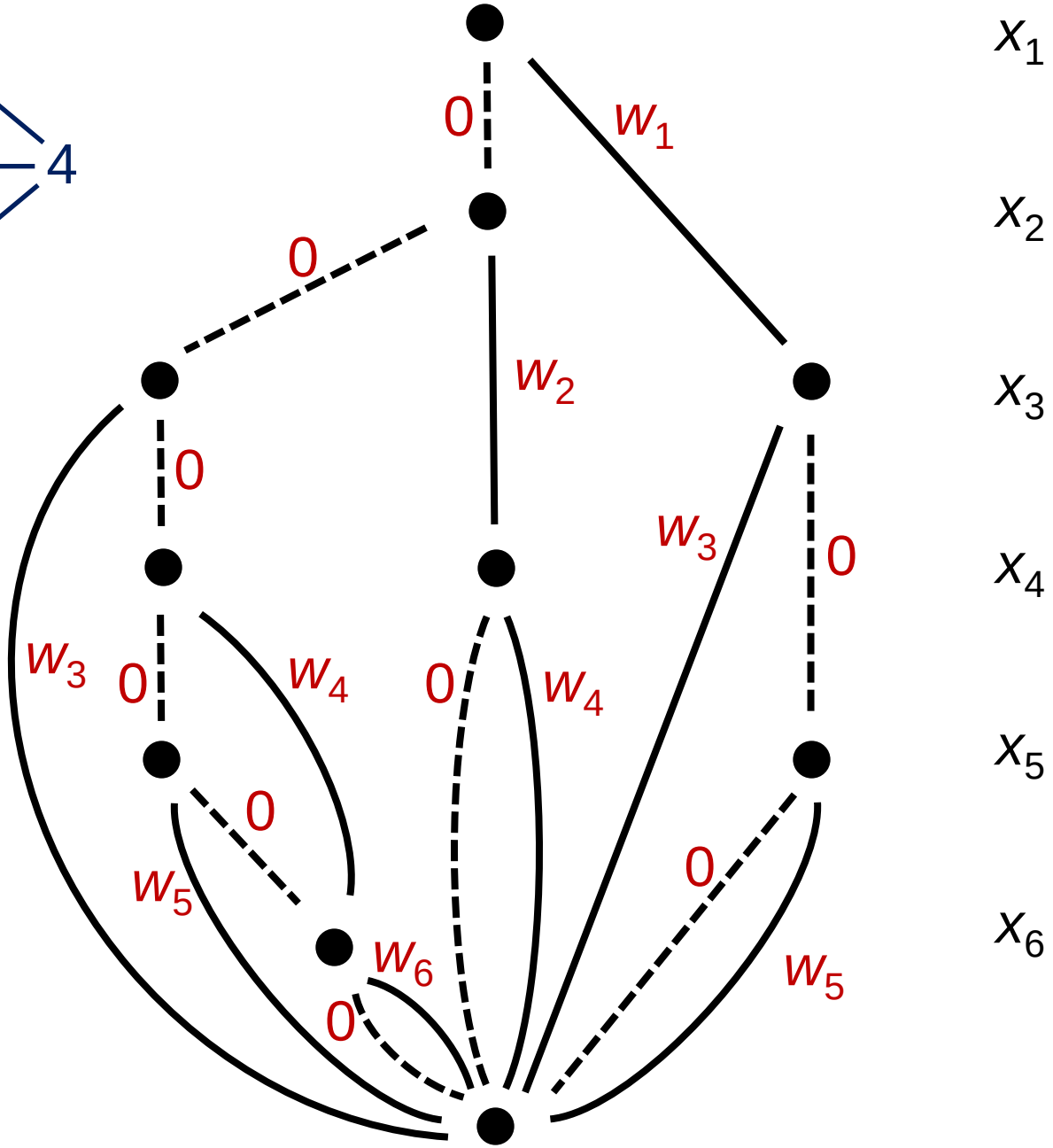
For objective function, associate weights with arcs

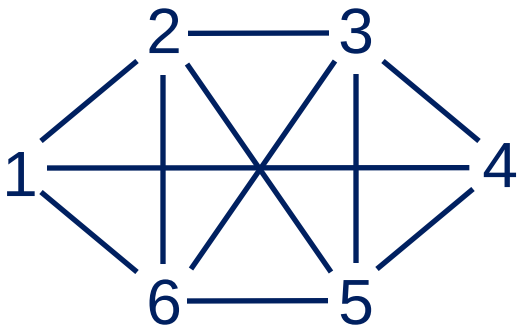


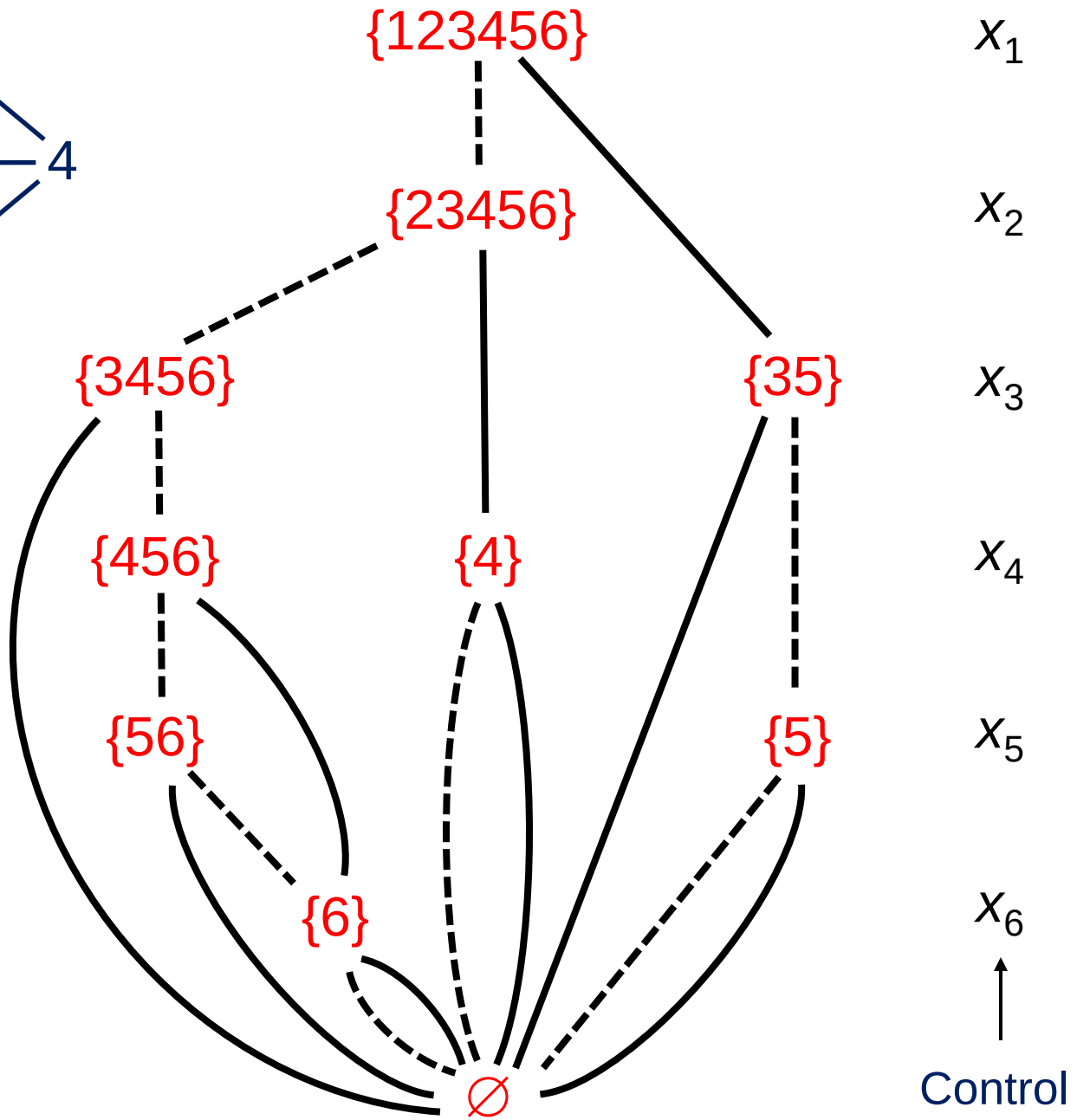


For objective function, associate weights with arcs

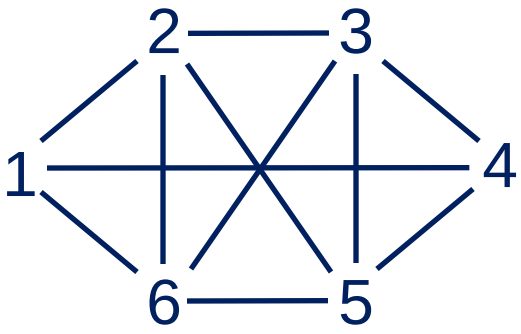
Optimal solution is **longest path**







To build BDD, associate **state** with each node as in dynamic programming



{123456}

x_1

x_2

x_3

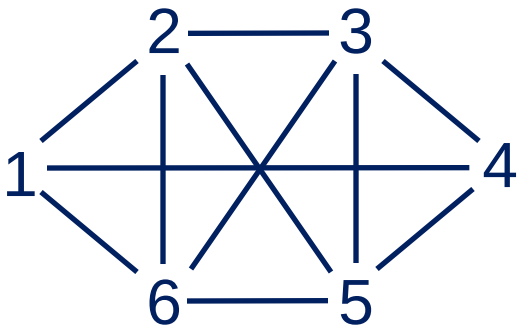
x_4

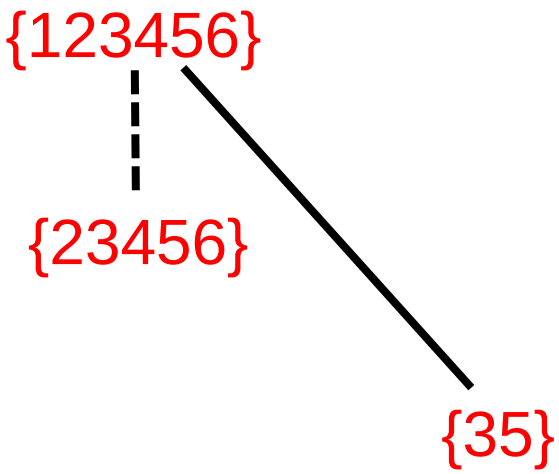
x_5

x_6

Control

To build BDD, associate **state** with each node





x_1

x_2

x_3

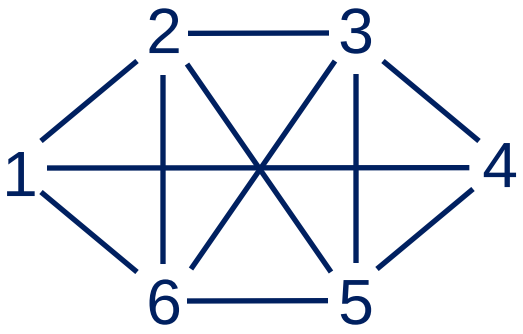
x_4

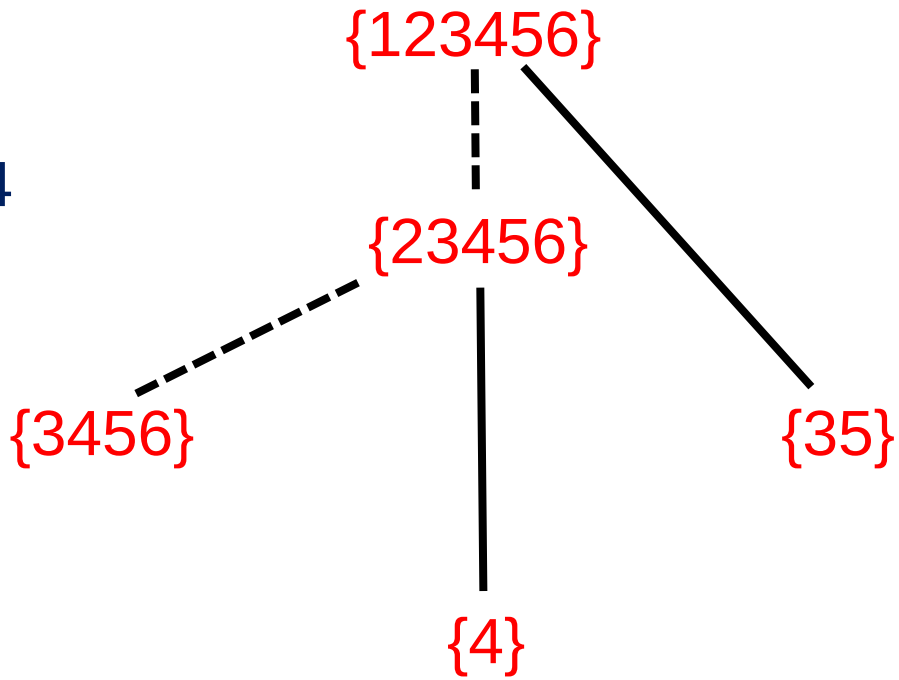
x_5

x_6

Control

To build BDD, associate **state** with each node





x_1

x_2

x_3

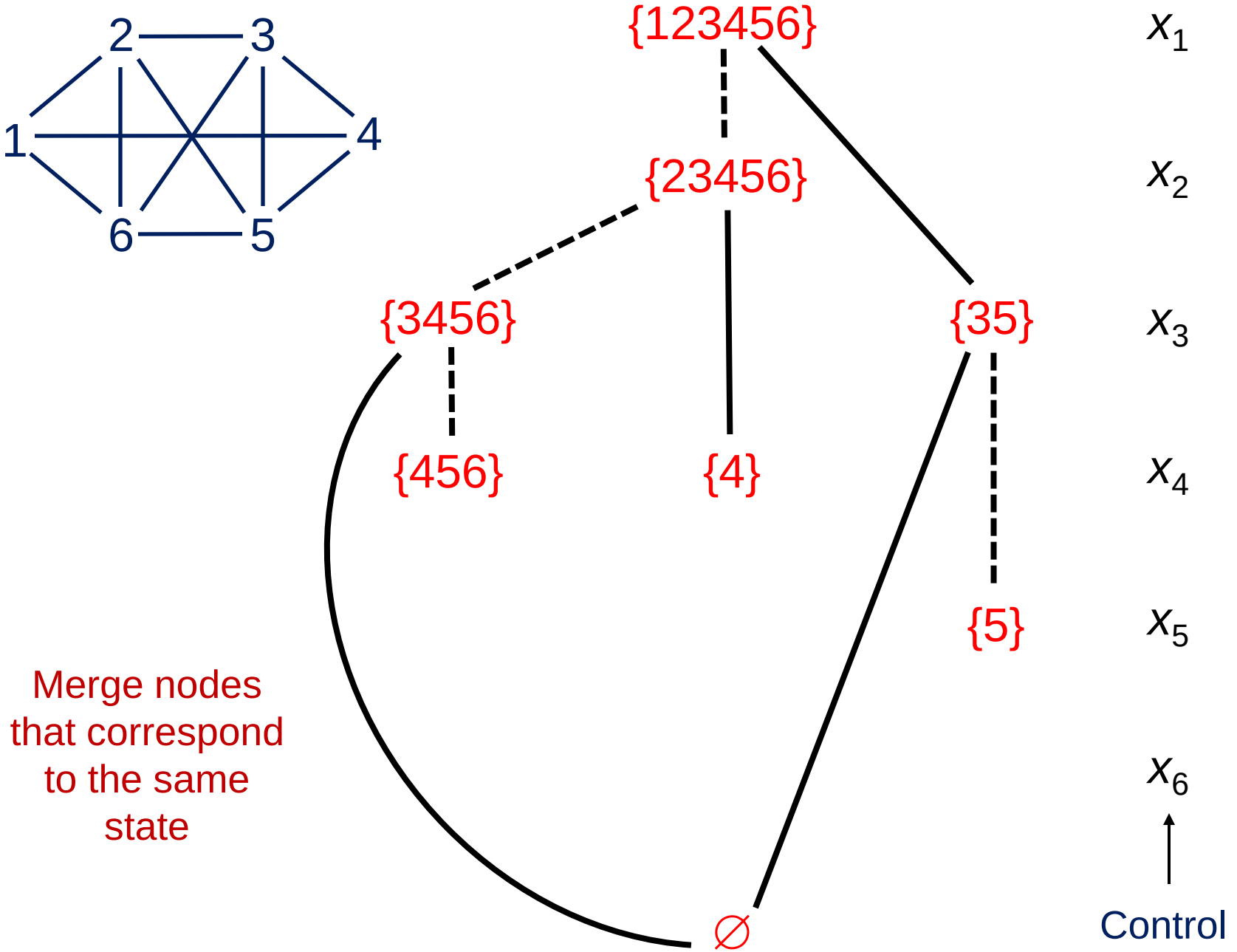
x_4

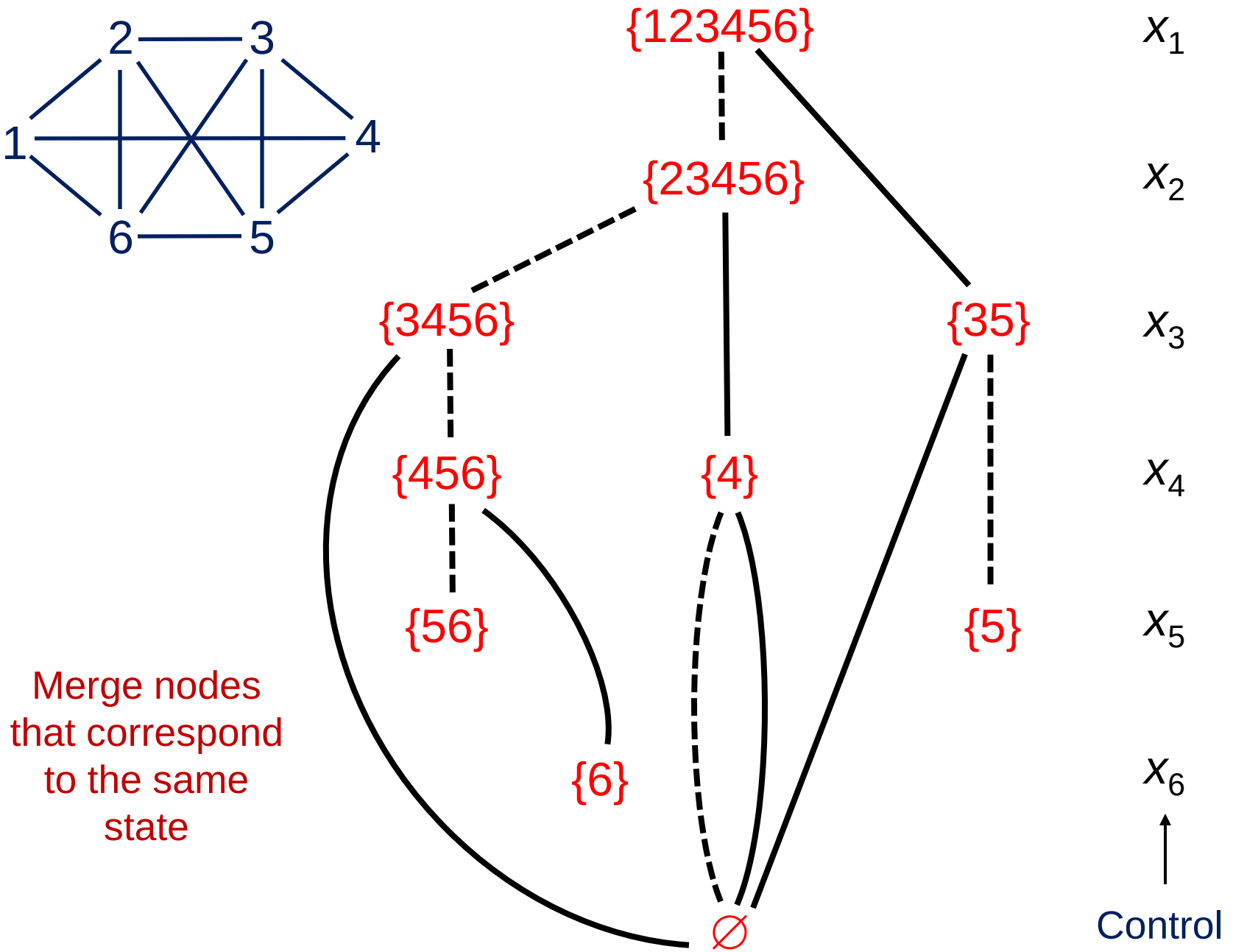
x_5

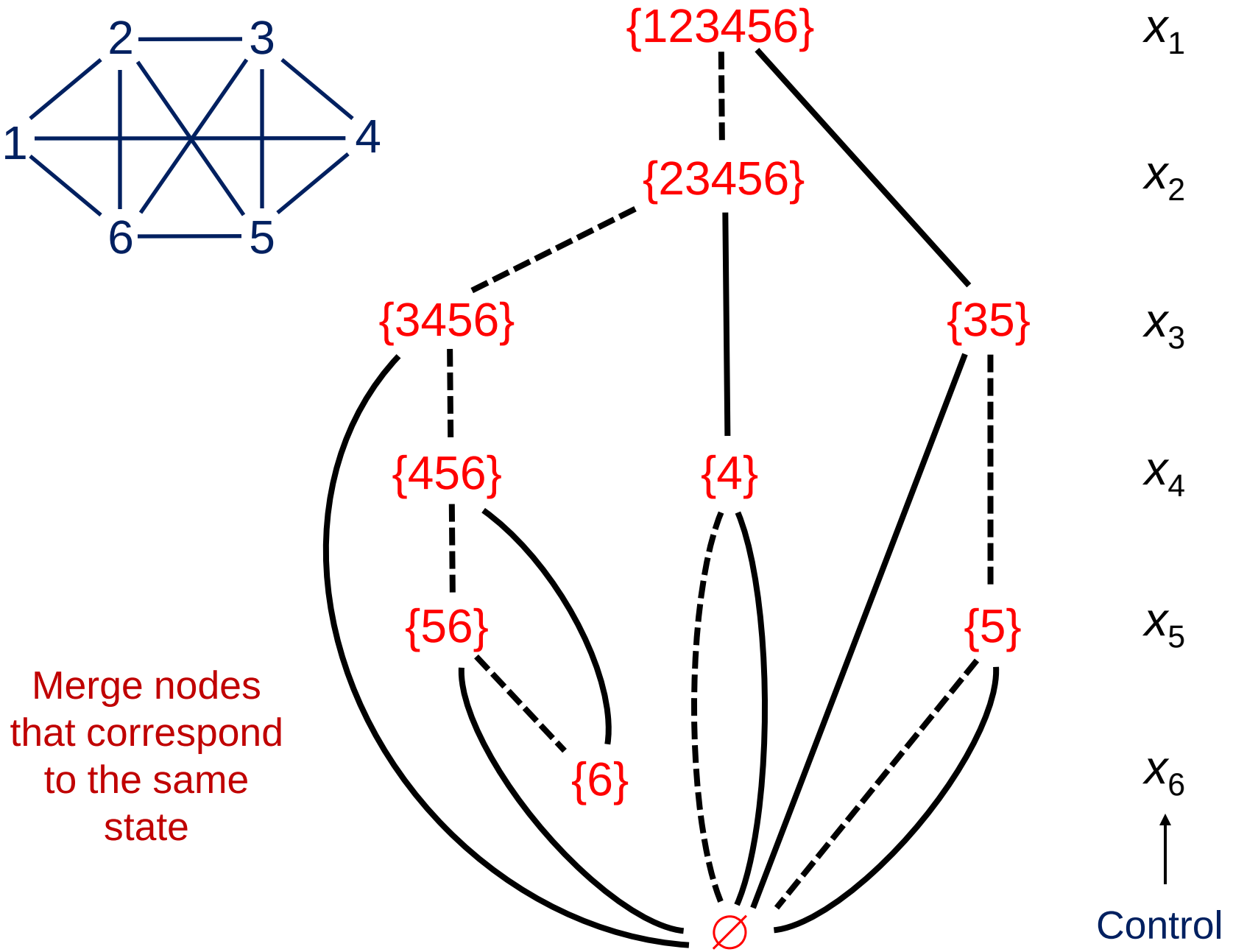
x_6

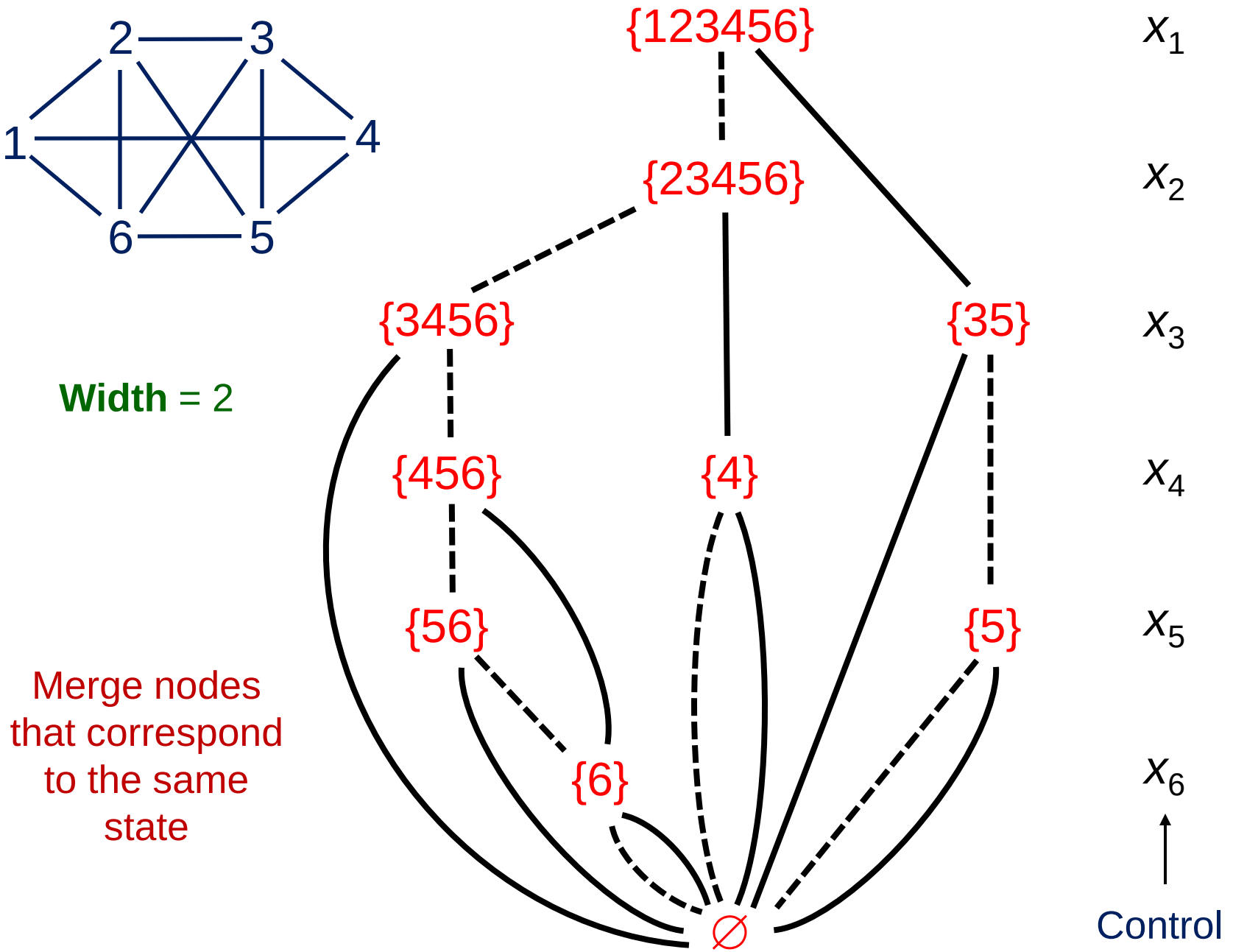
Control

To build BDD, associate **state** with each node


2
3
1
4
6
5
{123456}
{23456}
{3456}
{35}
{456}
{4}
{5}
∅
x_1
x_2
x_3
x_4
x_5
x_6
Control
Merge nodes that correspond to the same state

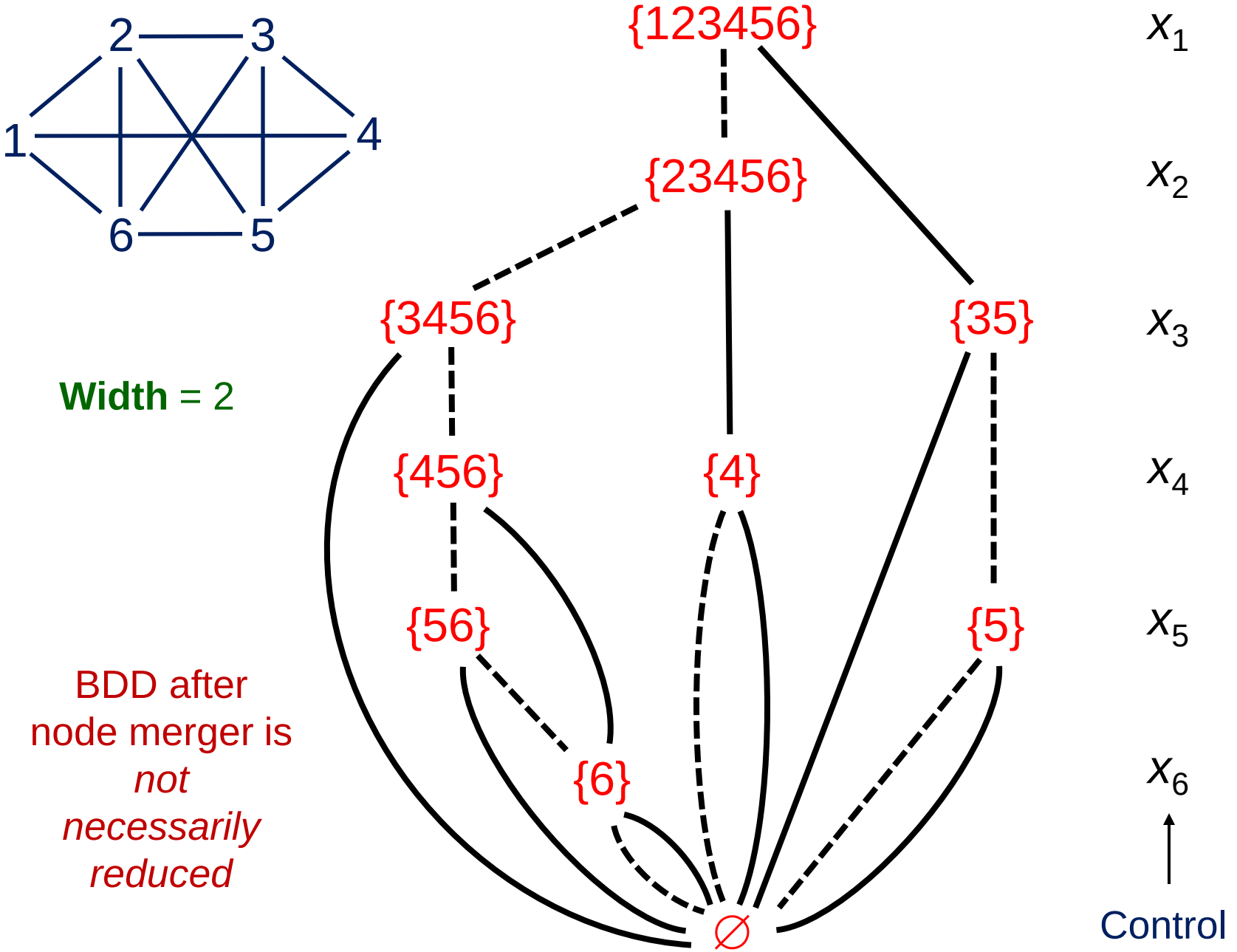

1
2
3
4
5
6
{123456}
{23456}
{3456}
{35}
{456}
{4}
{56}
{5}
{6}
∅
x_1
x_2
x_3
x_4
x_5
x_6
Control
Merge nodes that correspond to the same state


1
2
3
4
5
6
{123456}
{23456}
{3456}
{35}
{456}
{4}
{56}
{5}
{6}
∅
x_1
x_2
x_3
x_4
x_5
x_6
Control
Merge nodes that correspond to the same state



Width = 2

Merge nodes that correspond to the same state

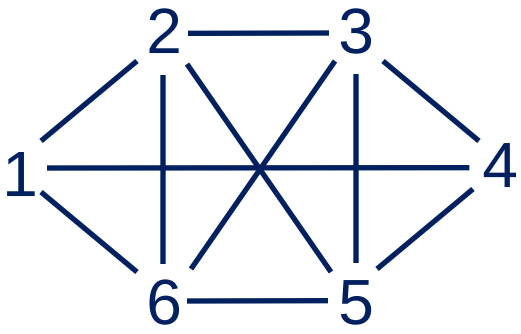


Width = 2

BDD after node merger is *not necessarily reduced*

Objective Function

- In general, objective function can be **any function of the current state.**
  - Linear or nonlinear, convex or nonconvex
  - There is a unique reduced BDD with **canonical** edge costs.



{123456}

x_1

x_2

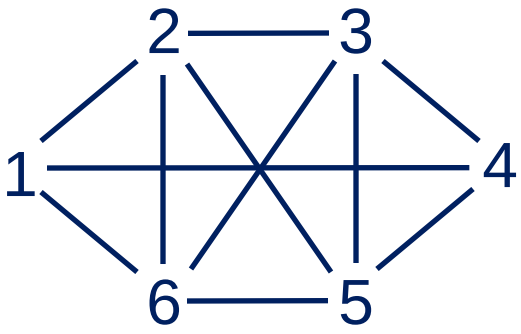
x_3

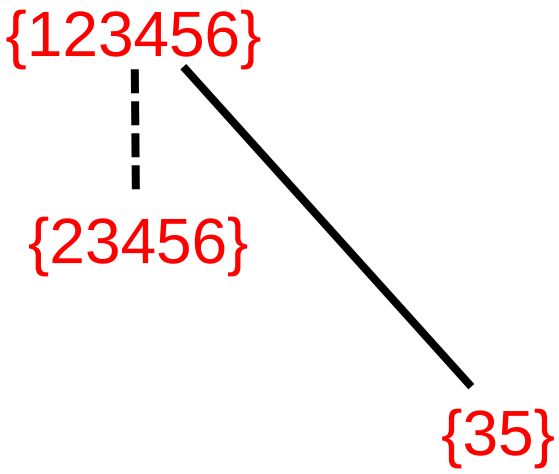
x_4

x_5

x_6

To build **relaxed** BDD, merge some additional nodes as we go along





x_1

x_2

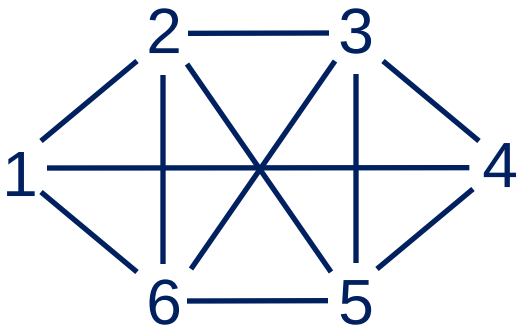
x_3

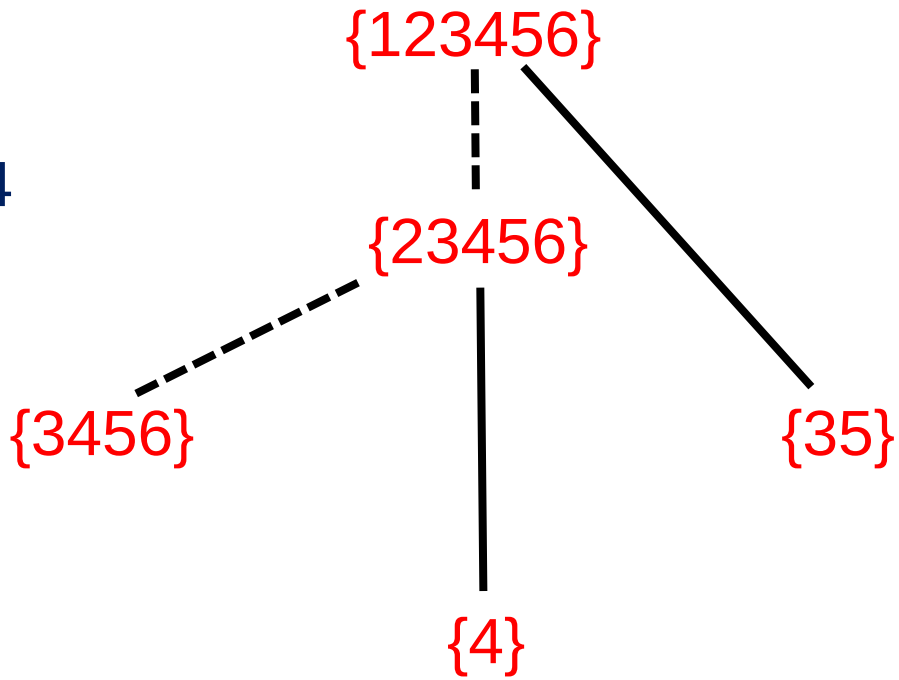
x_4

x_5

x_6

To build **relaxed** BDD, merge some additional nodes as we go along





x_1

x_2

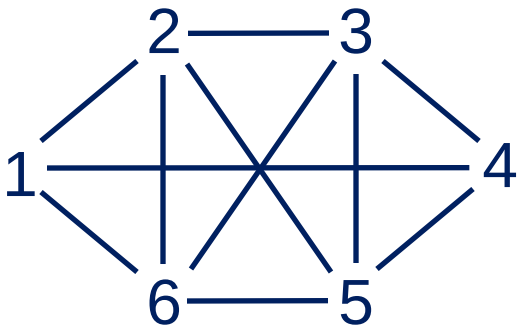
x_3

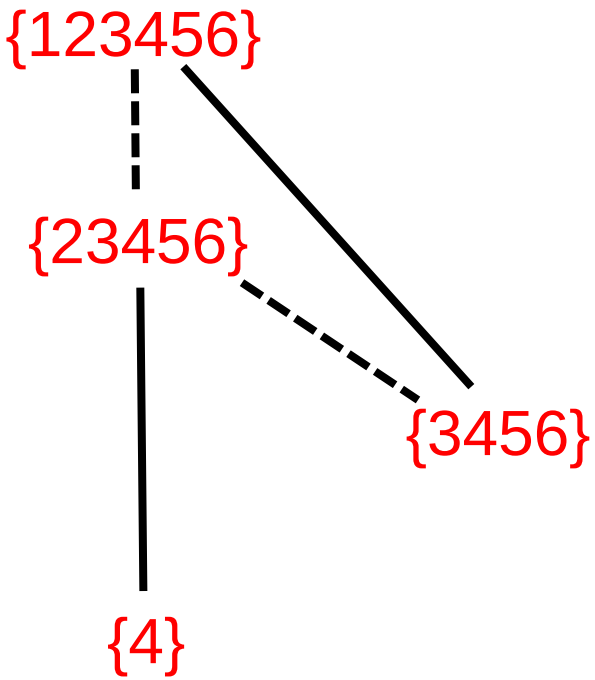
x_4

x_5

x_6

To build **relaxed** BDD, merge some additional nodes as we go along





x_1

x_2

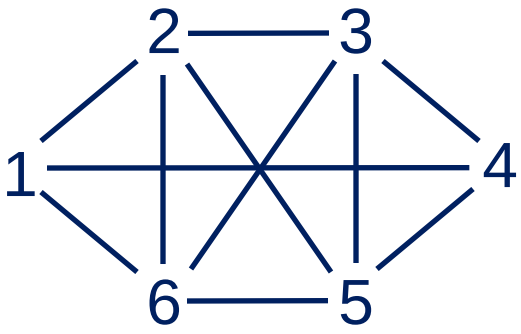
x_3

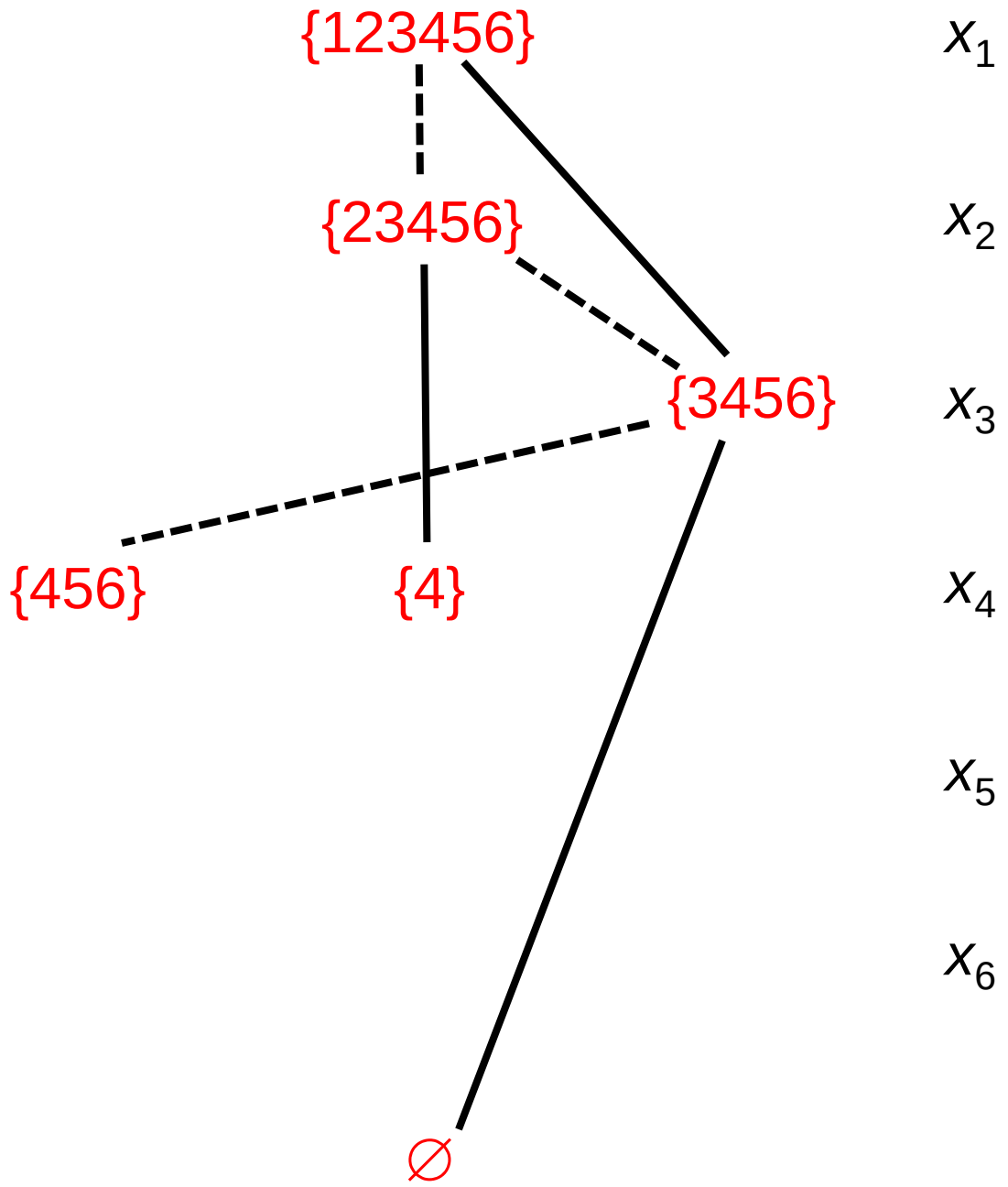
x_4

x_5

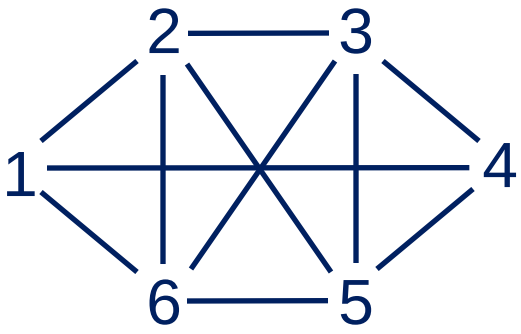
x_6

To build **relaxed** BDD, merge some additional nodes as we go along

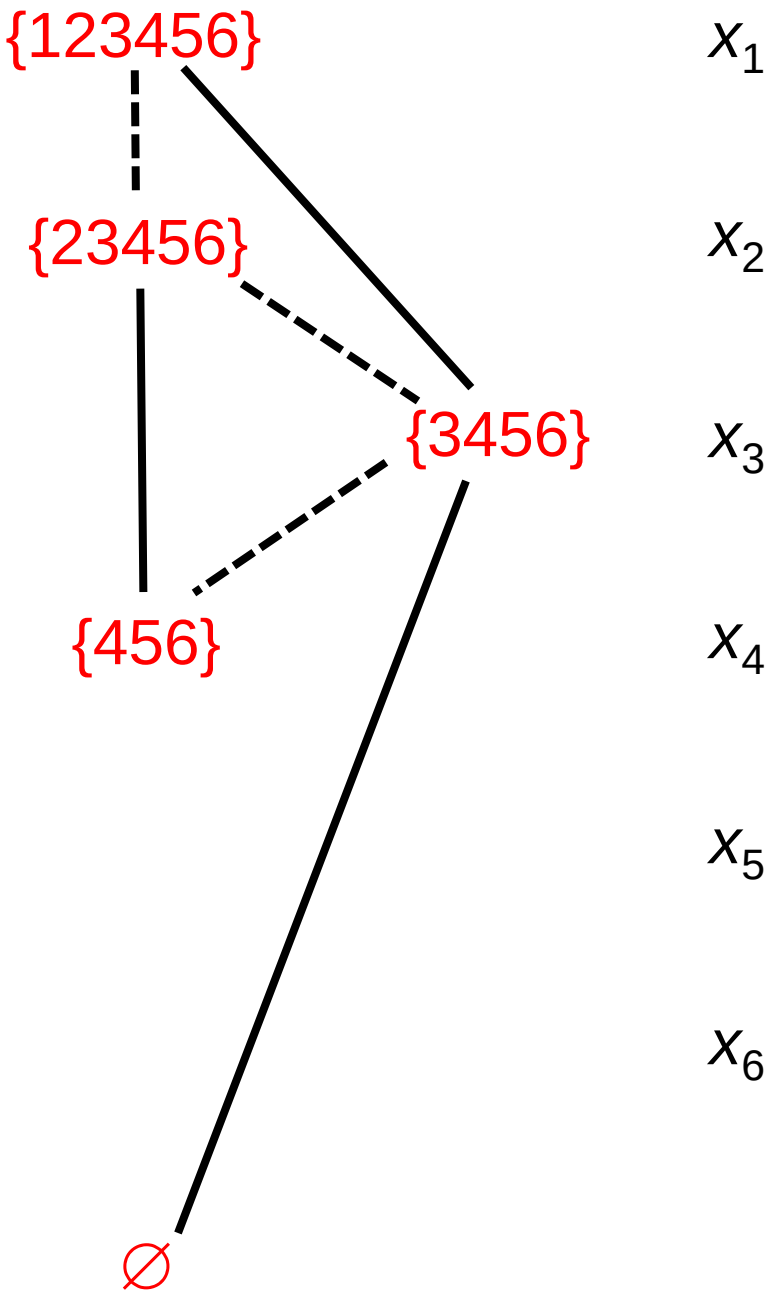


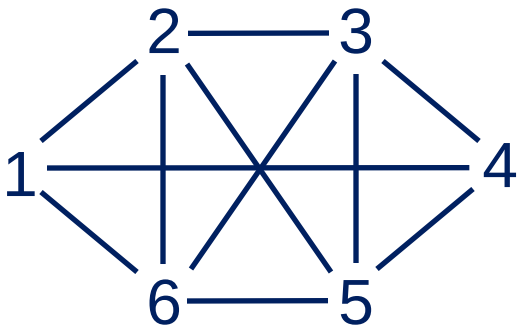


To build **relaxed** BDD, merge some additional nodes as we go along

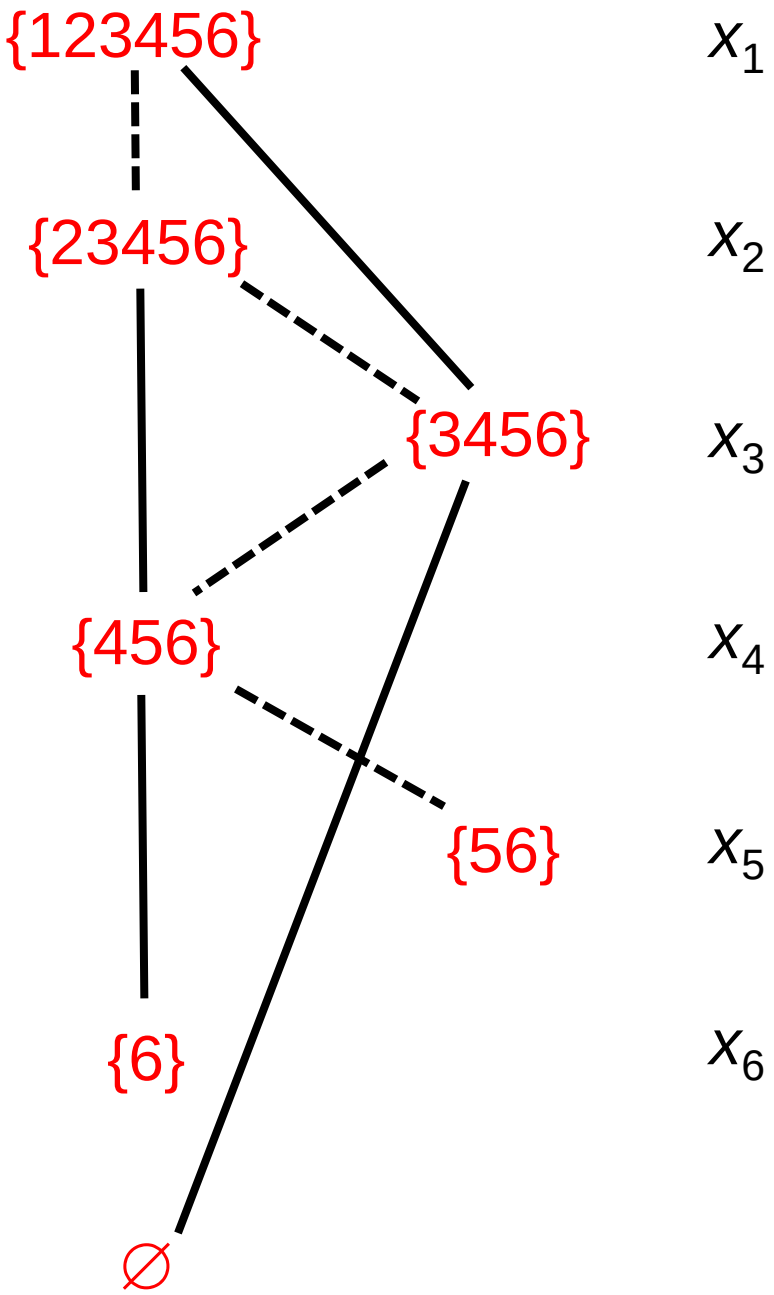


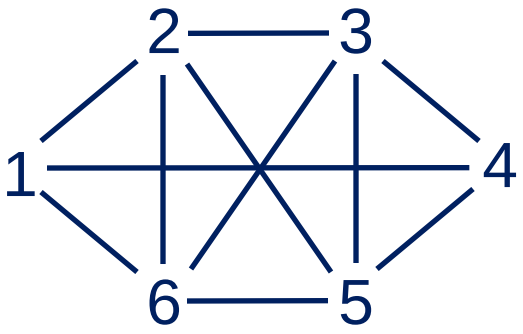
To build **relaxed** BDD, merge some additional nodes as we go along



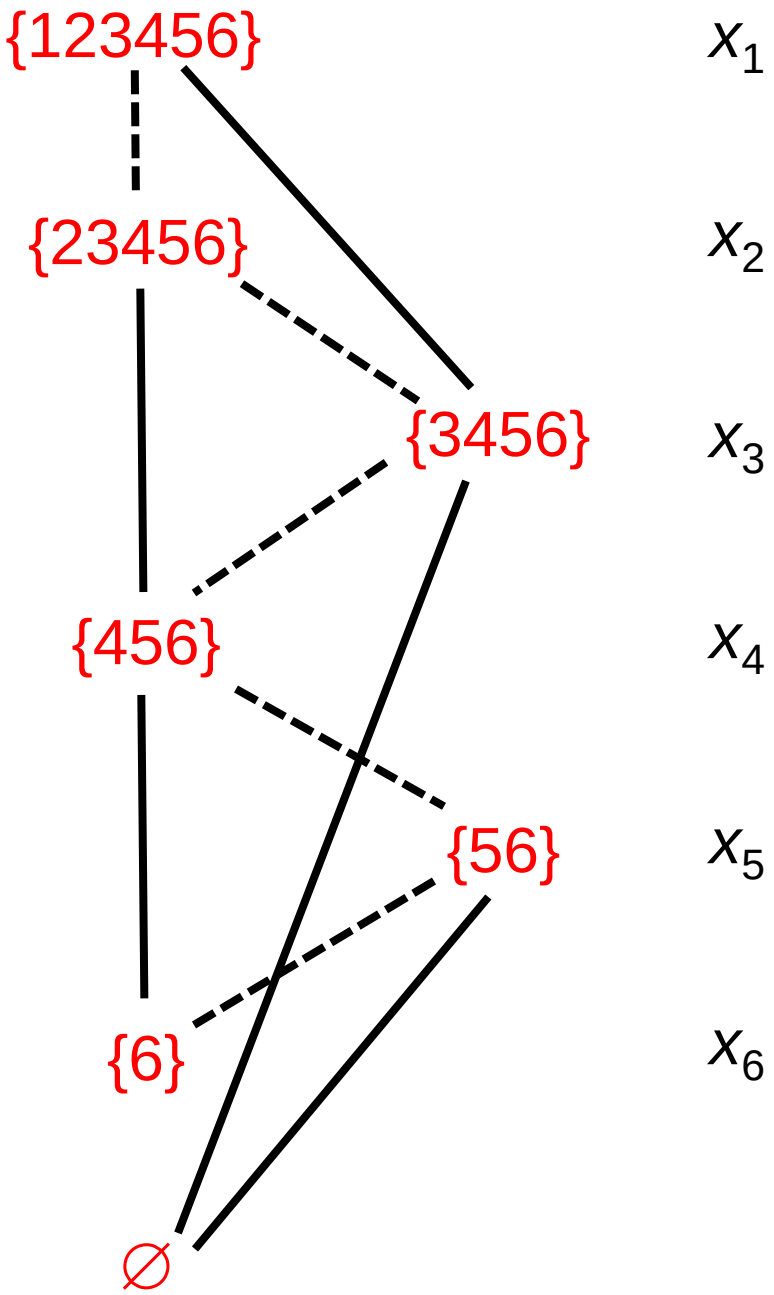


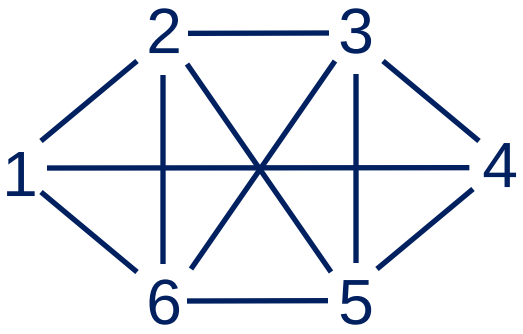
To build **relaxed** BDD, merge some additional nodes as we go along





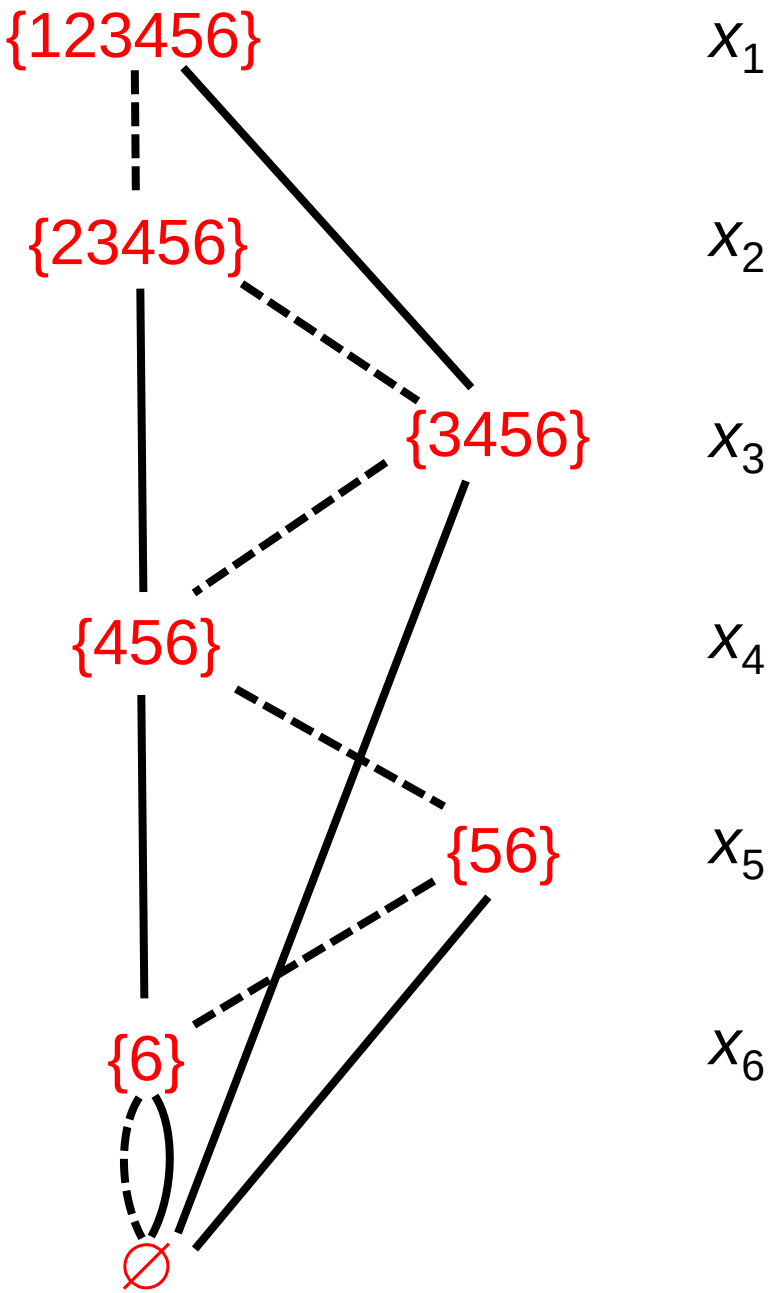
To build **relaxed** BDD, merge some additional nodes as we go along

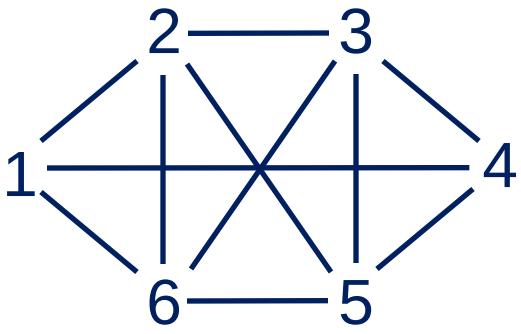




Width = 1

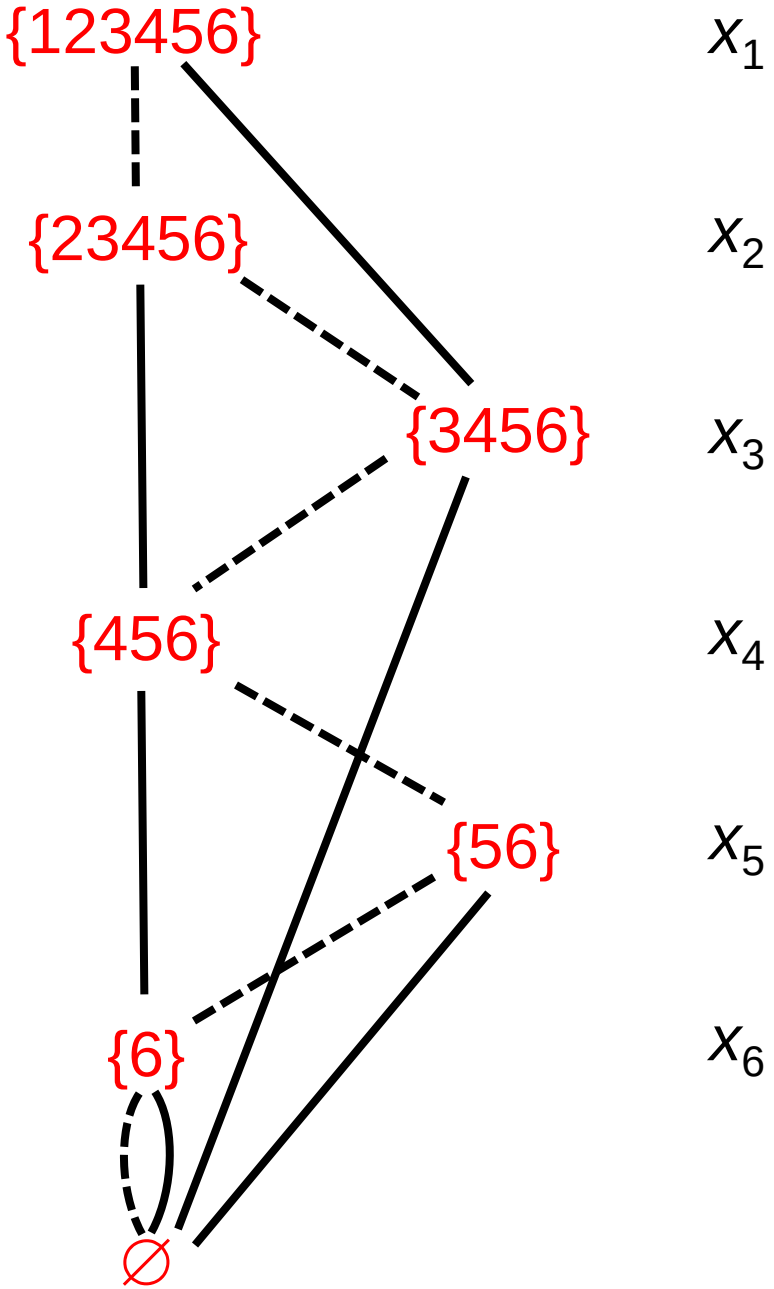
To build **relaxed** BDD, merge some additional nodes as we go along

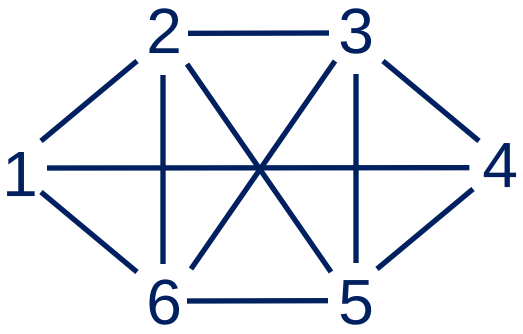




Width = 1

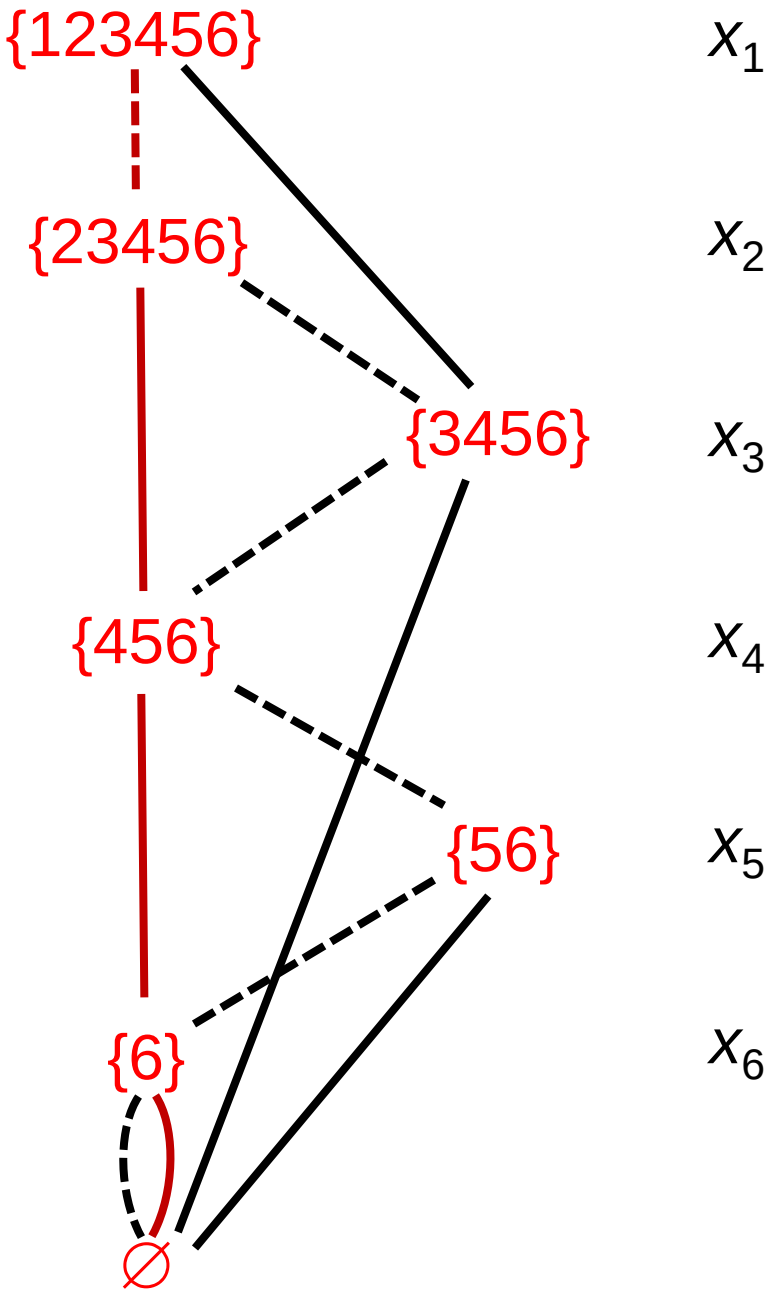
Represents 18 solutions, including 11 feasible solutions





Width = 1

Longest path
gives bound
of 3 on optimal
value of 2



Variable Ordering

- Variable ordering is important.
  - Just as in branching methods.
  - Some orderings provide tighter bounds.
  - Recursive model can dictate natural variable ordering.

Variable Ordering

- Variable ordering is important.
  - Just as in branching methods.
  - Some orderings provide tighter bounds.
  - Recursive model can dictate natural variable ordering.
- For stable set problem, width of exact BDD is bounded by Fibonacci numbers.
  - For an ordering induced by maximal path decomposition of the graph.

Variable Ordering

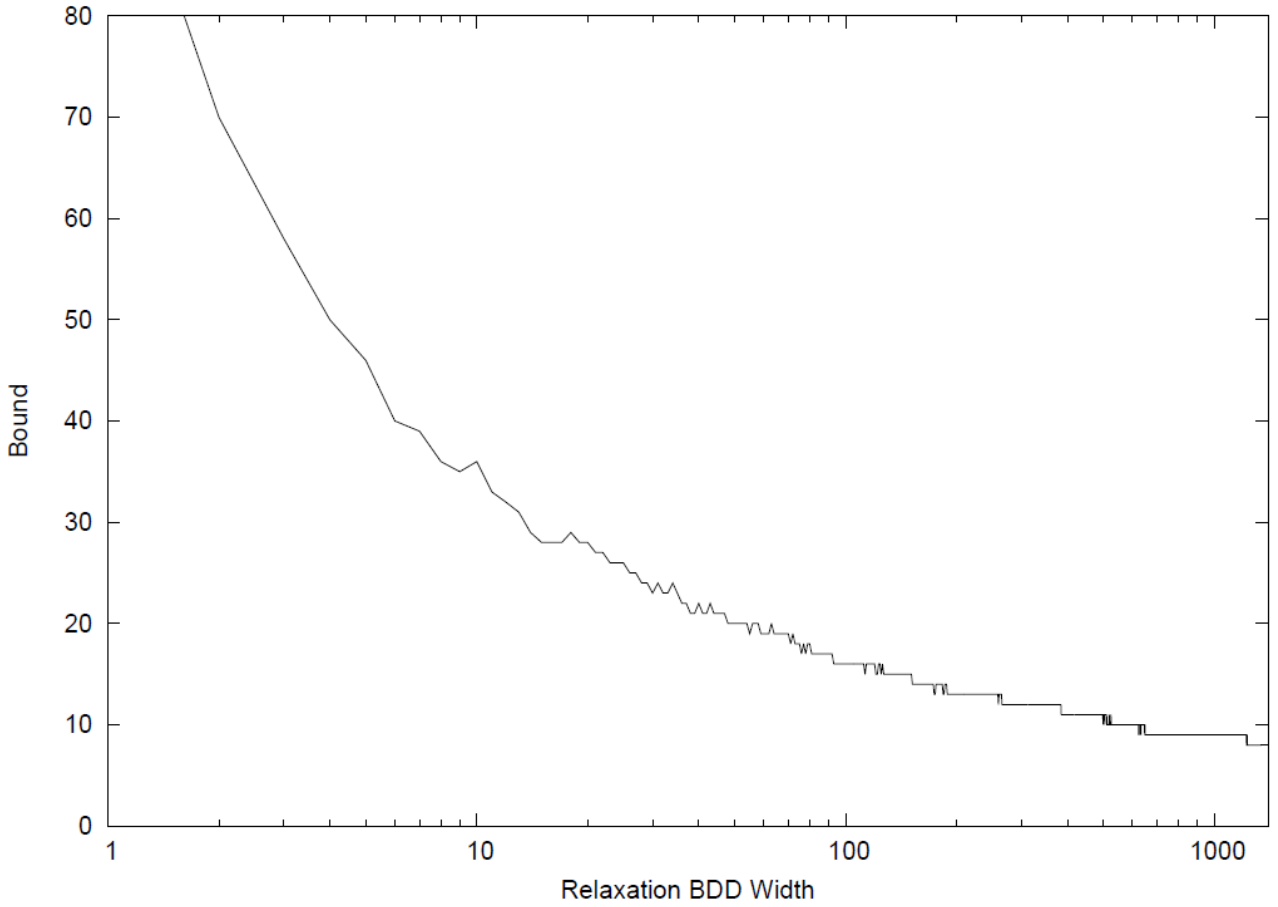
- Variable ordering is important.
  - Just as in branching methods.
  - Some orderings provide tighter bounds.
  - Recursive model can dictate natural variable ordering.
- For stable set problem, width of exact BDD is bounded by Fibonacci numbers.
  - For an ordering induced by maximal path decomposition of the graph.
- We used a dynamic ordering heuristic.
  - Next variable is the one appearing in the smallest number of states on the current level.
    - Better than maximal path ordering.

Merging Heuristic

- Which nodes to merge when building relaxation?
- A quality-of-solution heuristic seems by far the best.
  - Define a state variable that indicates solution quality.
  - Merge nodes associated with poor solution quality
  - In this case, solution quality = longest path into node.
  - We lose information in areas not likely to be part of the solution.

Width of BDD

- Wider BDDs yield tighter bounds.
  - But take longer to build.



Comparison with LP Bound

- Random and benchmark instances
- Compare with LP bound at root node in CPLEX
  - Use clique cover IP formulation
    - Requires precomputing clique cover.

$$\max \sum_i w_i y_i$$

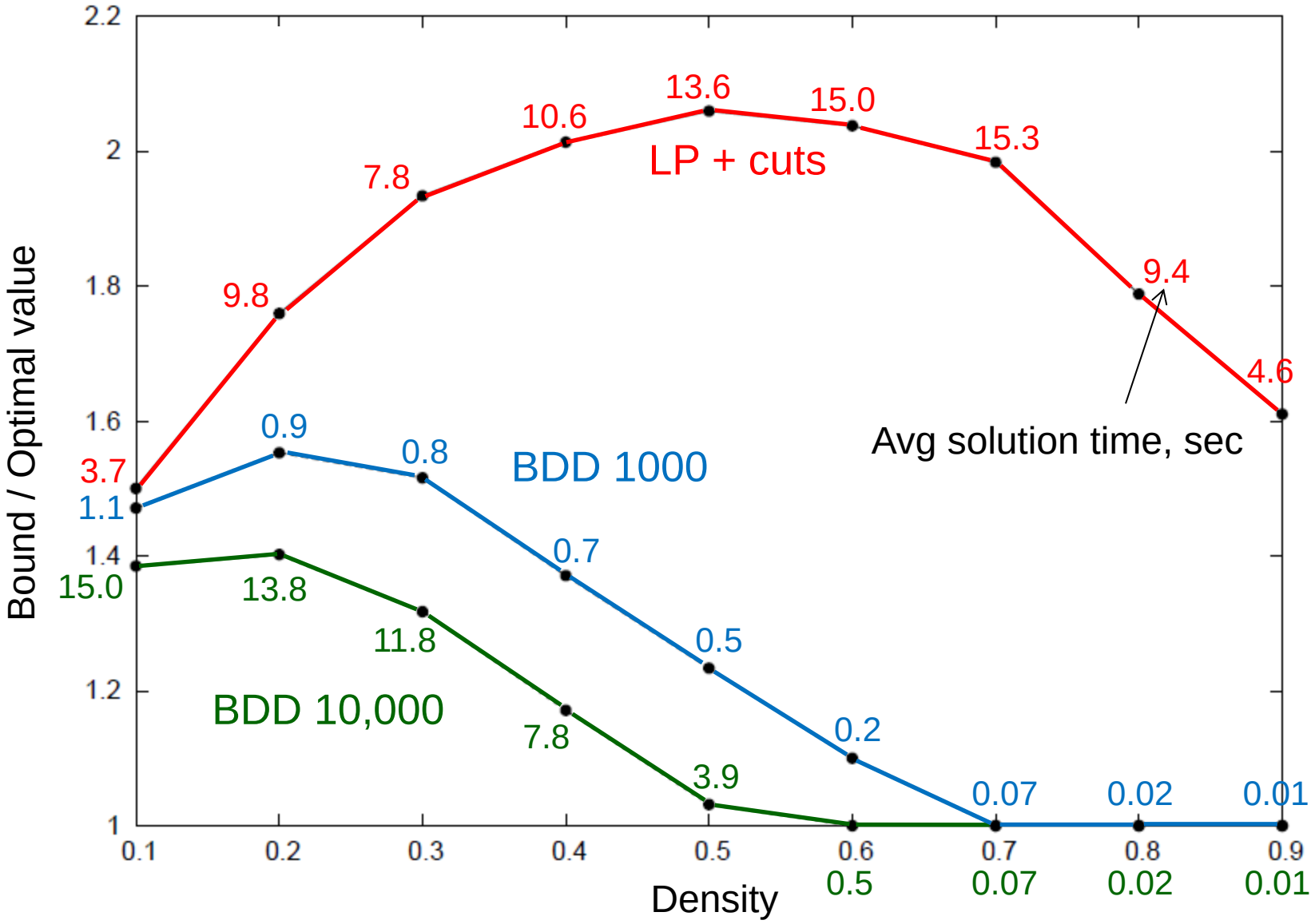
$$\sum_{i \in C_k} y_i \leq 1, \quad \text{all cliques } C_k \text{ in clique cover}$$

$$y_i \in \{0,1\}, \quad \text{all } i$$

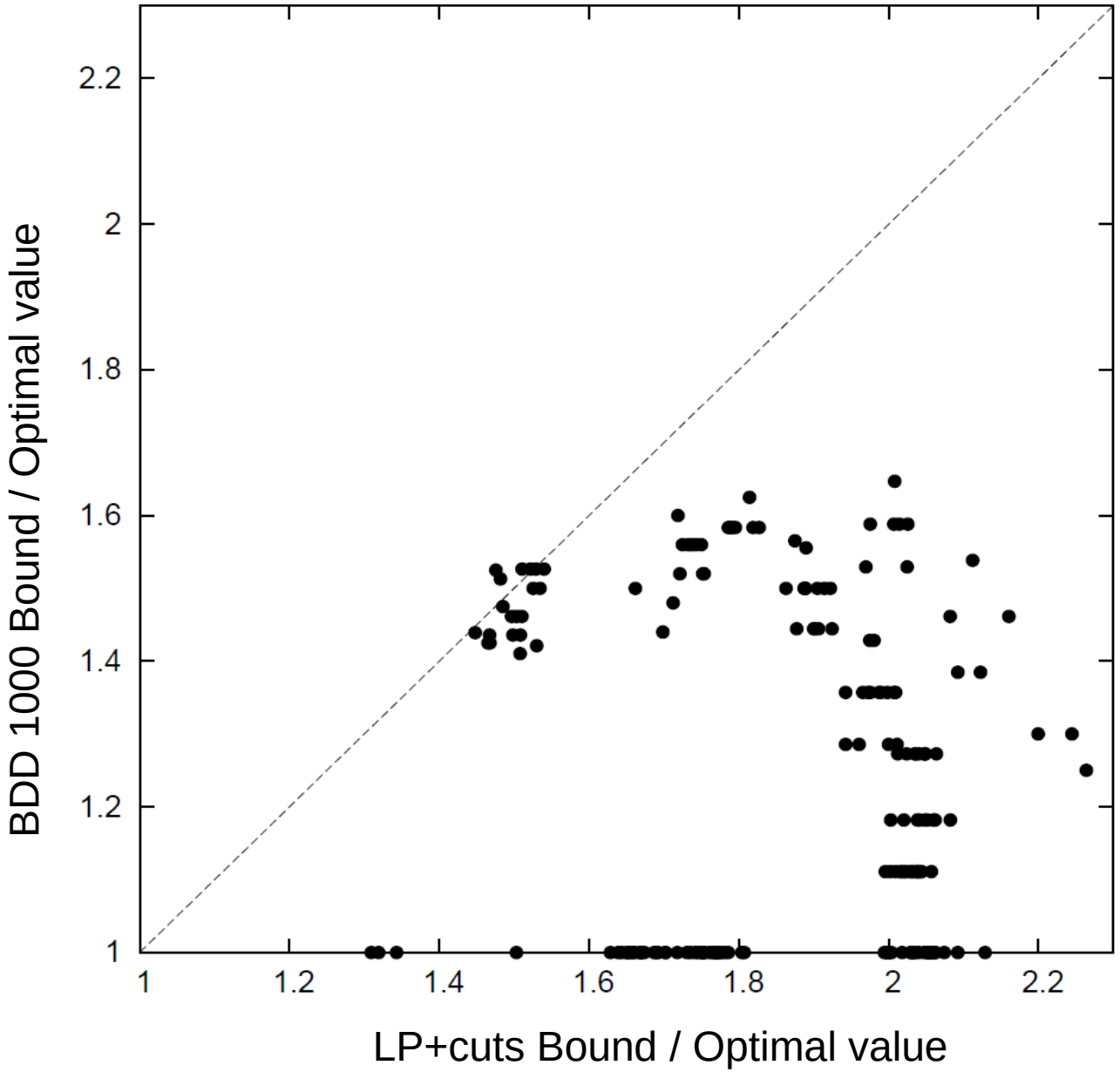
Comparison with LP Bound

- CPLEX settings
  - Turn off presolve
    - It makes CPLEX bound worse or at most 1 better.
    - BDD bounds don’t use presolve.
  - Use full cutting plane resources (50 years of development)
  - Use interior point (barrier) LP solver
    - Faster than simplex on these instances.

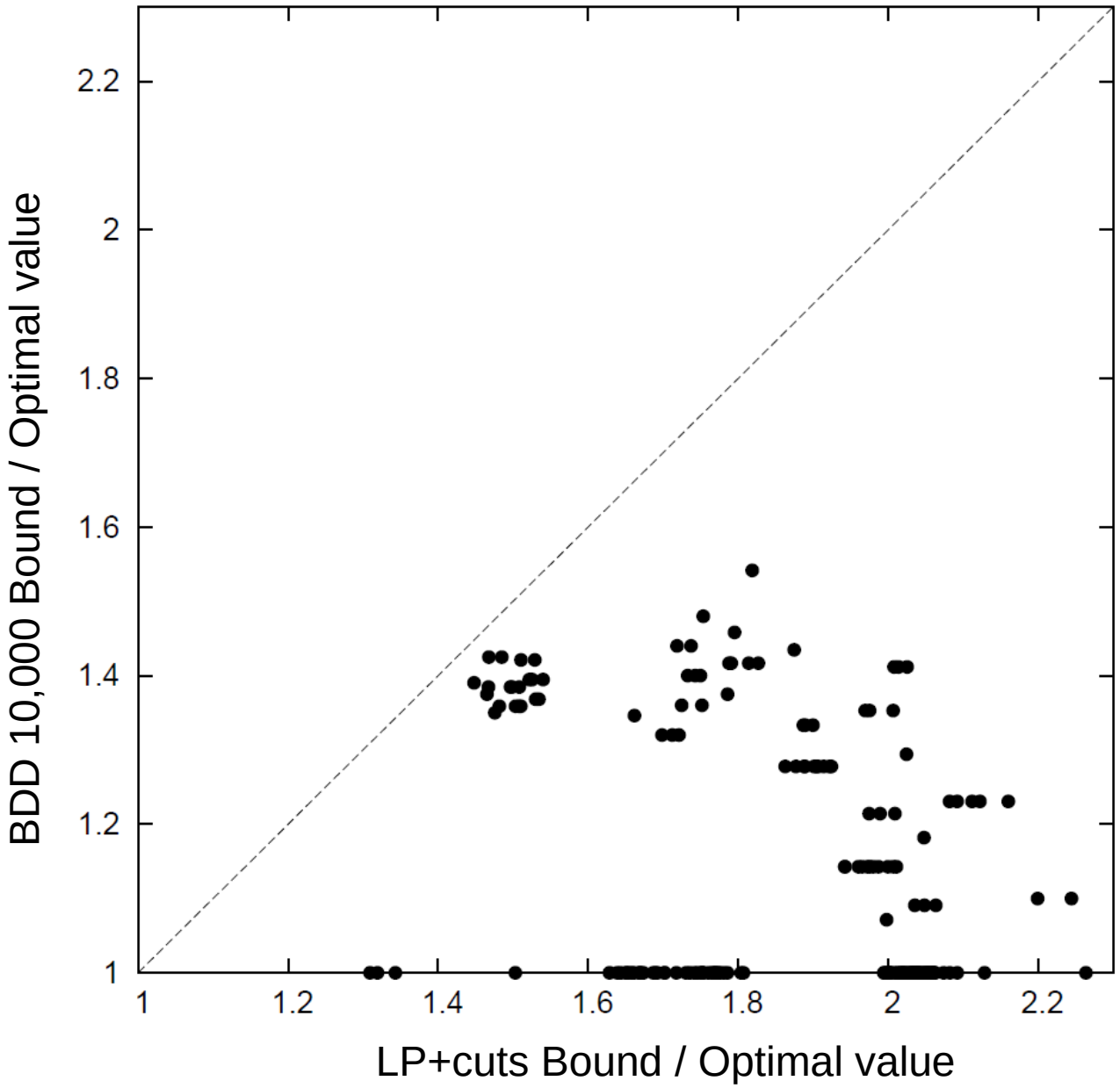
Random instances



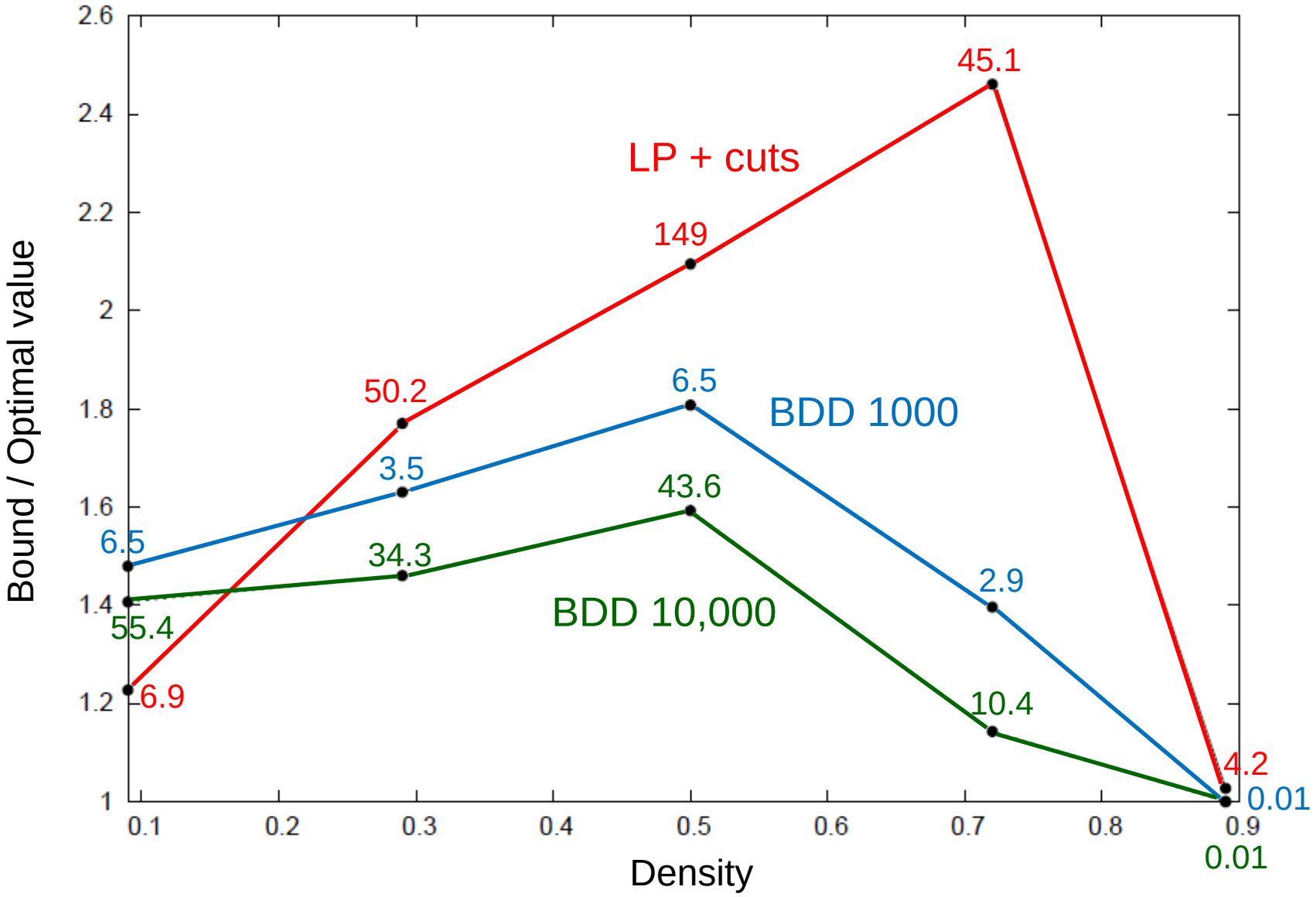
Random instances



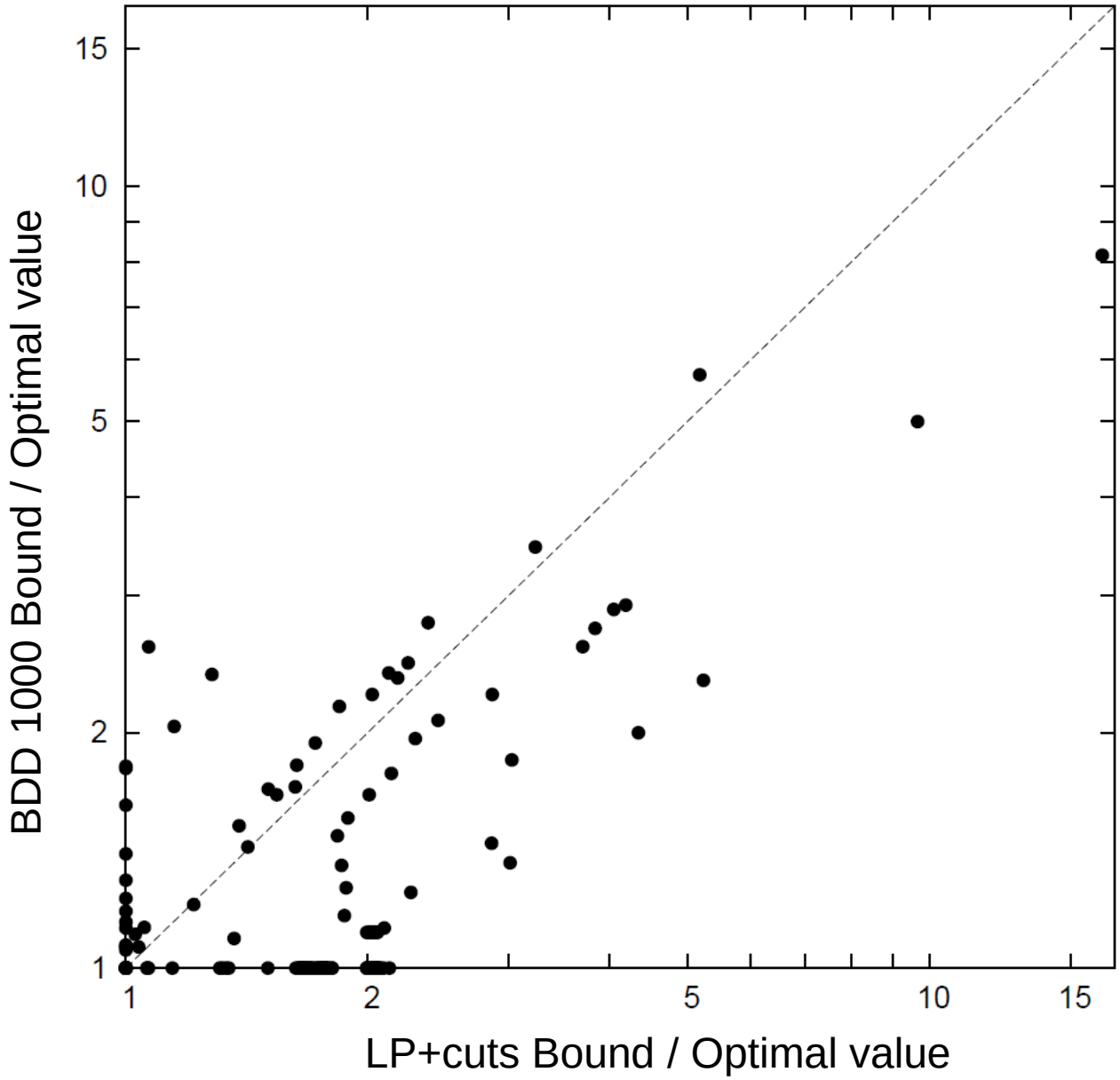
Random instances



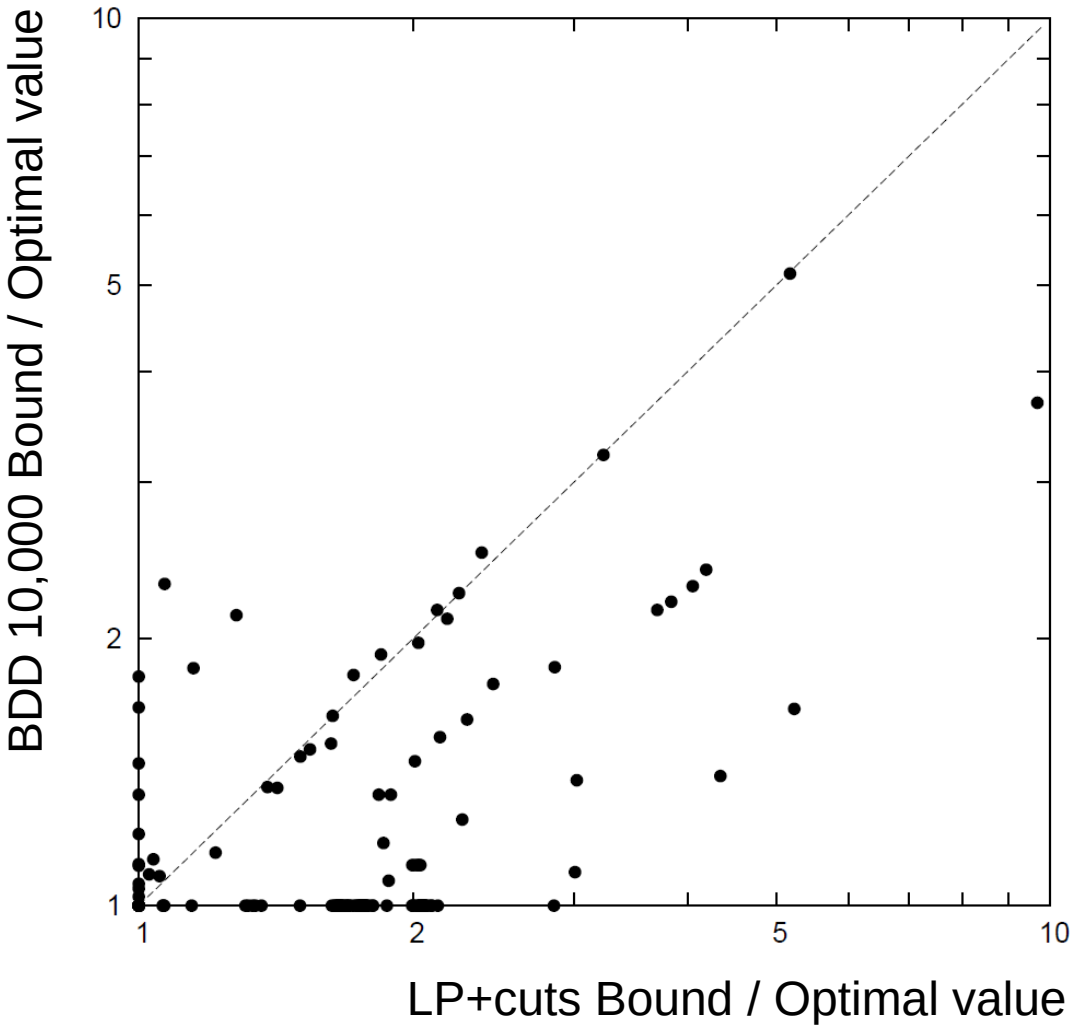
DIMACS instances



DIMACS instances



DIMACS instances



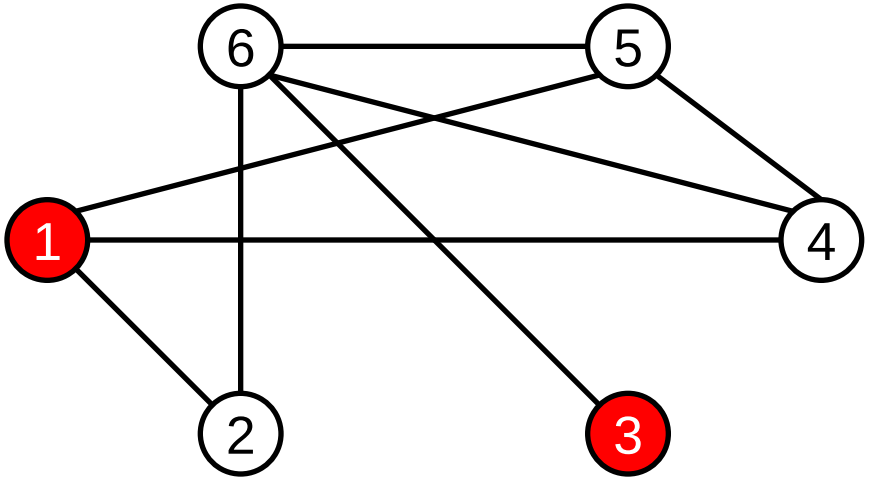
Incrementality

- Fast incremental calculation of bound.
  - Modify relaxed BDD after variable is fixed in branching tree (fast).
  - Recompute longest path (fast).
- However, may be better to rebuild relaxation from scratch.
  - Research issue.

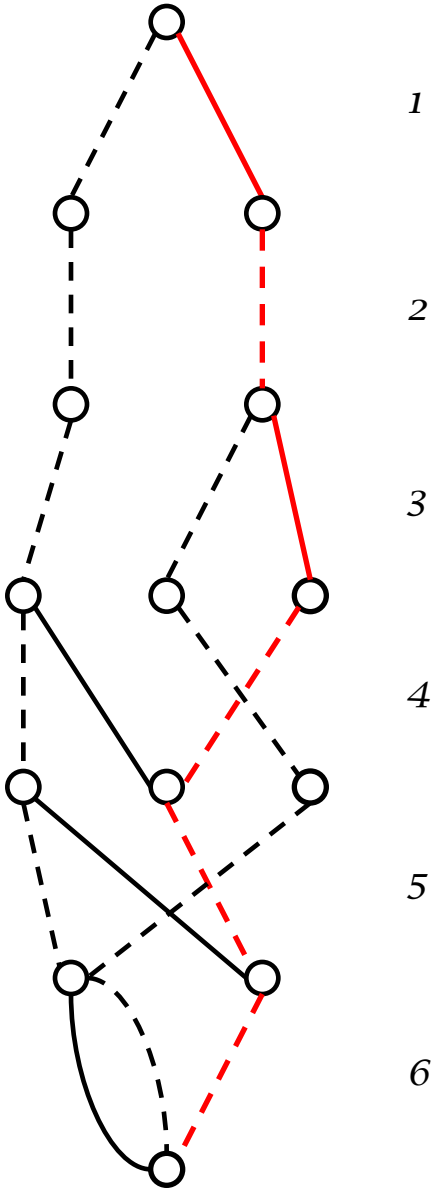
Restricted BDDs

- **Restricted** BDDs as **primal heuristic**
  - All paths are feasible.
  - Longest path is a good feasible solution
  - Reduce width by intelligently removing nodes.
- Apply to stable set problem
  - DIMACS instances
  - Compare with lower bound obtained by CPLEX primal heuristics at rote node.
  - Details and updates in Bergman, Ciré, van Hoeve, Yunes, *J. of Heuristics* 2014.

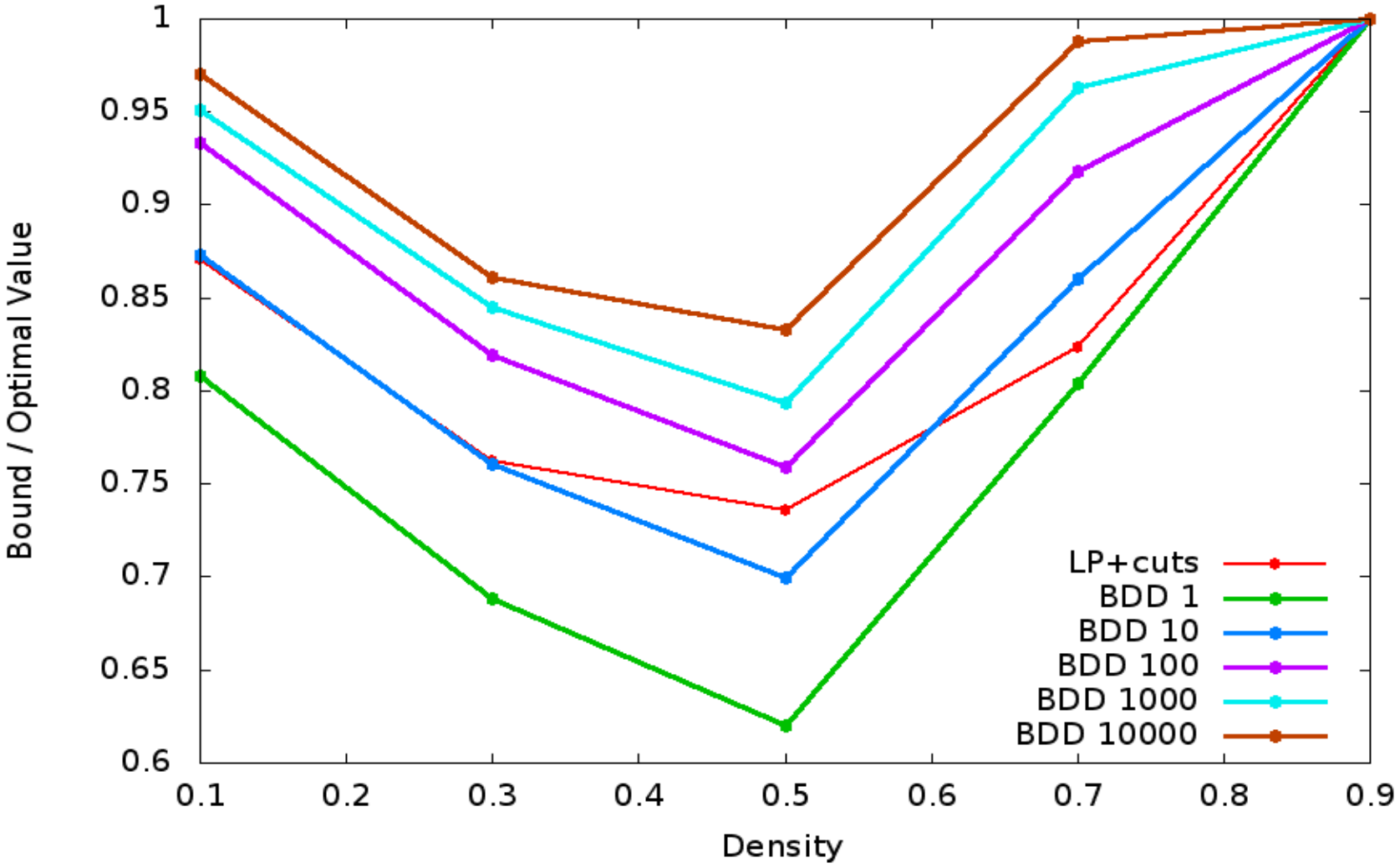
Restricted BDD
Width = 3



Longest path is
feasible solution
and gives upper bound
of **2** on optimal value of **3**



DIMACS instances



How to Use BDD Bounds

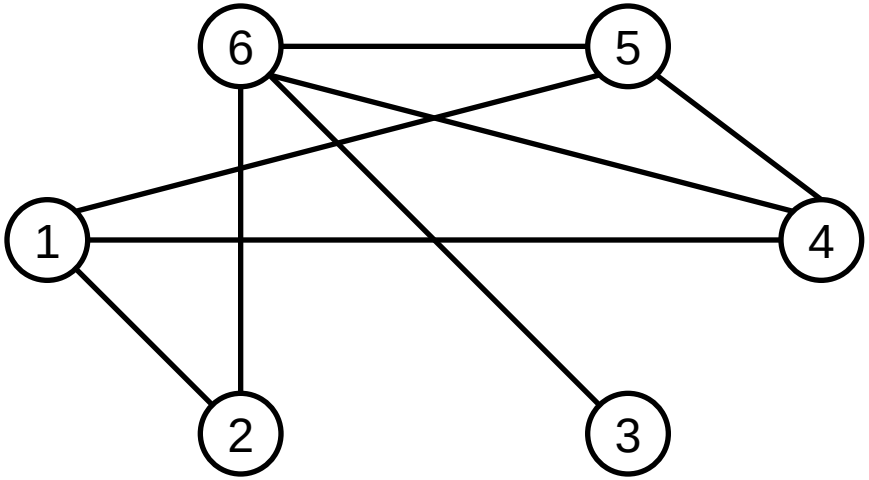
- Assist **existing solvers**
  - MIP solvers.
    - Bound at root node.
    - Can generate BDD bounds at nodes below root node, due to speed of BDD processing.
  - CP solvers.
    - Bound at root node.

How to Use BDD Bounds

- Part of general **BDD-based solver**
  - **Branch in the relaxed BDD**.
  - Combine with BDD-based propagation
    - BDD plays role of LP relaxation, cutting planes
  - BDD-based primal heuristic
  - Dynamic programming style formulation
    - Specify state variable for BDDs
    - Linearity, convexity irrelevant
  - Details in talk by David Bergman tomorrow (ACP Thesis Award).

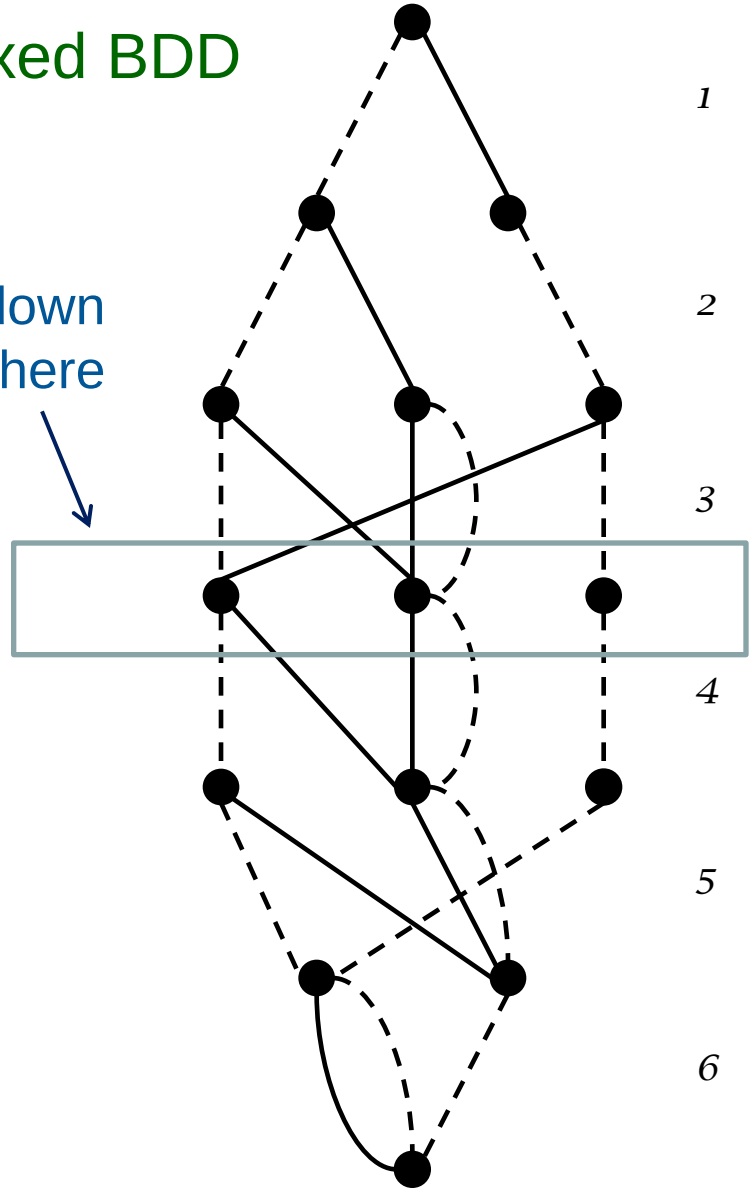
Branching in Relaxed BDD

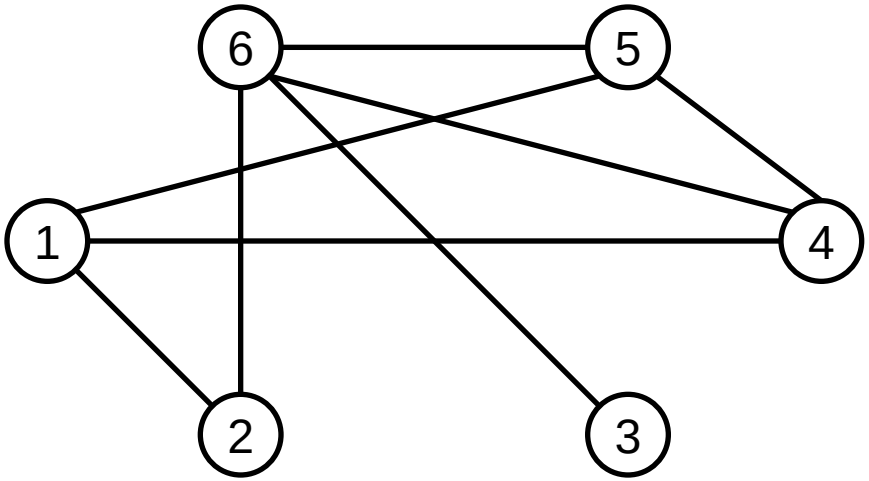
- Novel branching scheme
  - Branch on nodes of relaxed BDD.
  - …rather than on variables.



Relaxed BDD

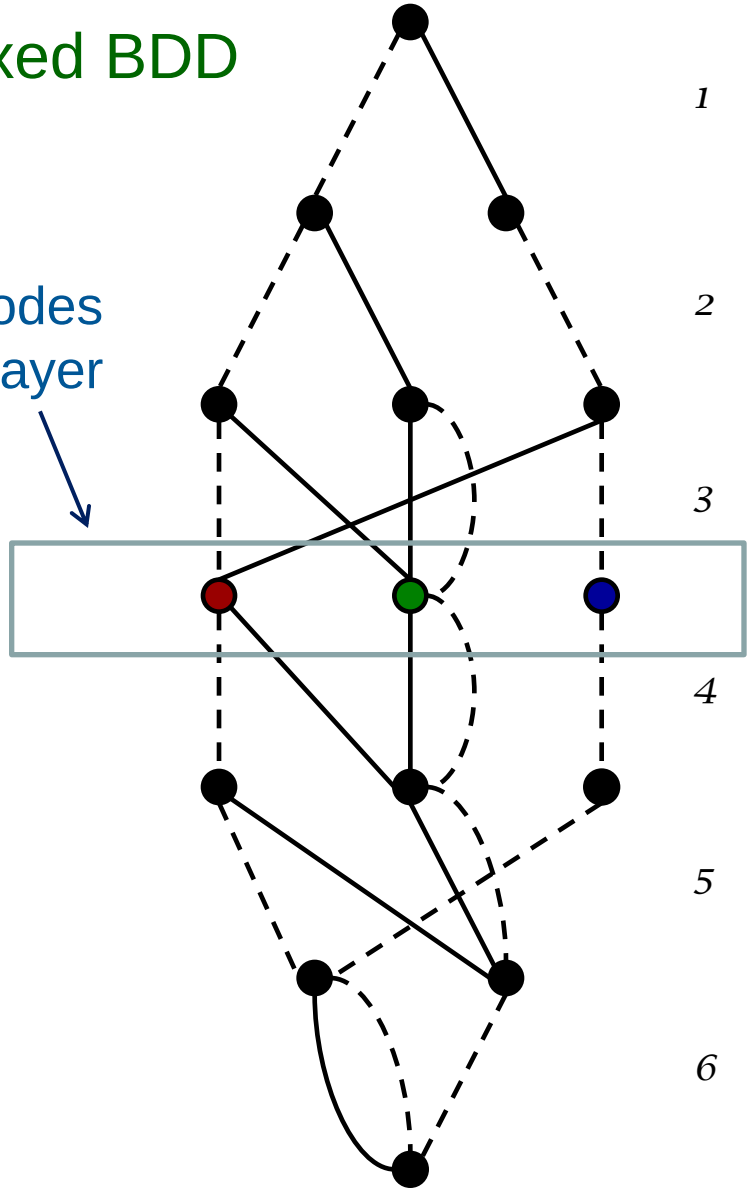
Exact down to here

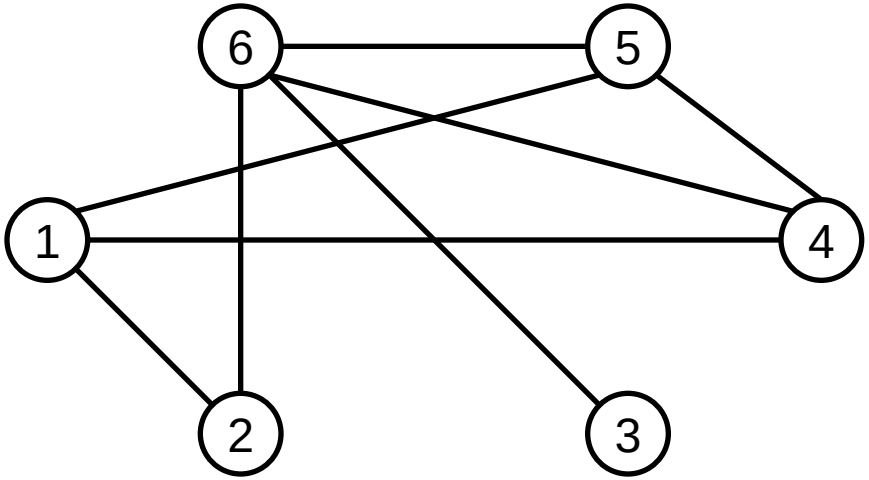




Relaxed BDD

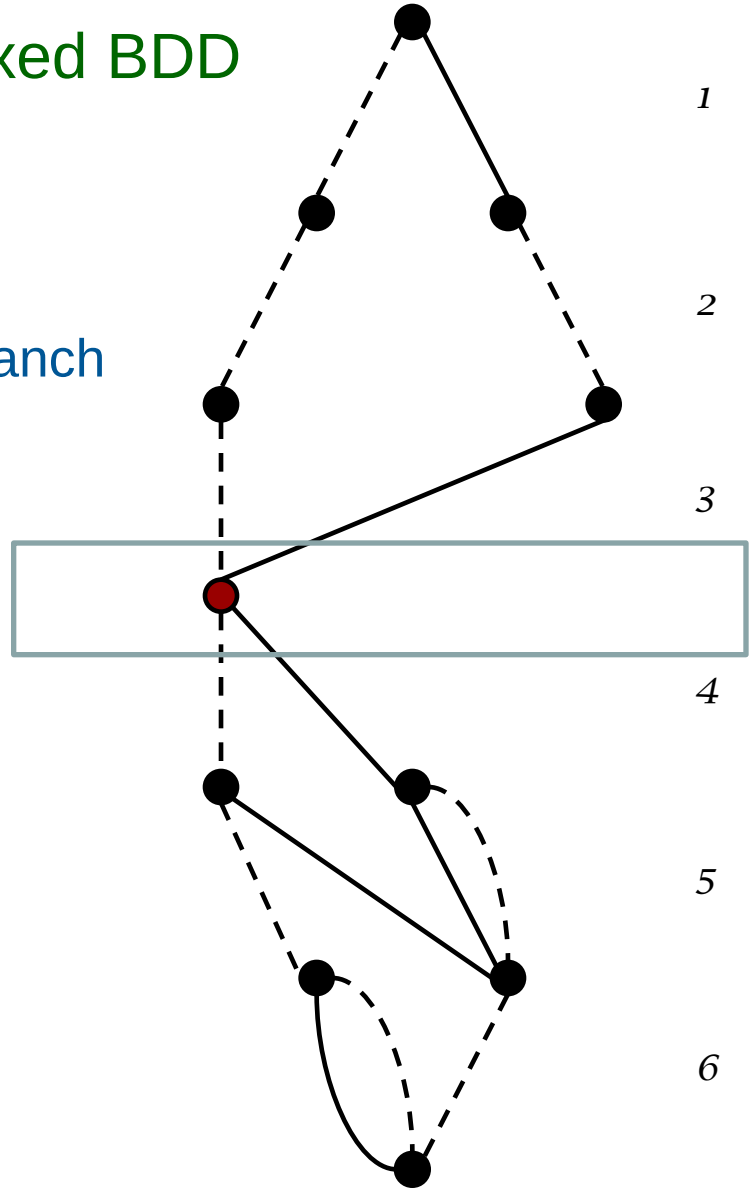
Branch on nodes
in this layer



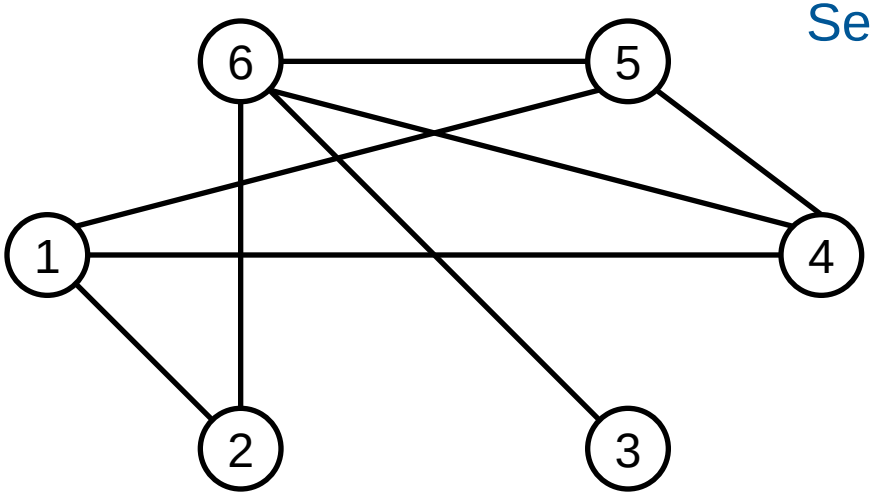


Relaxed BDD

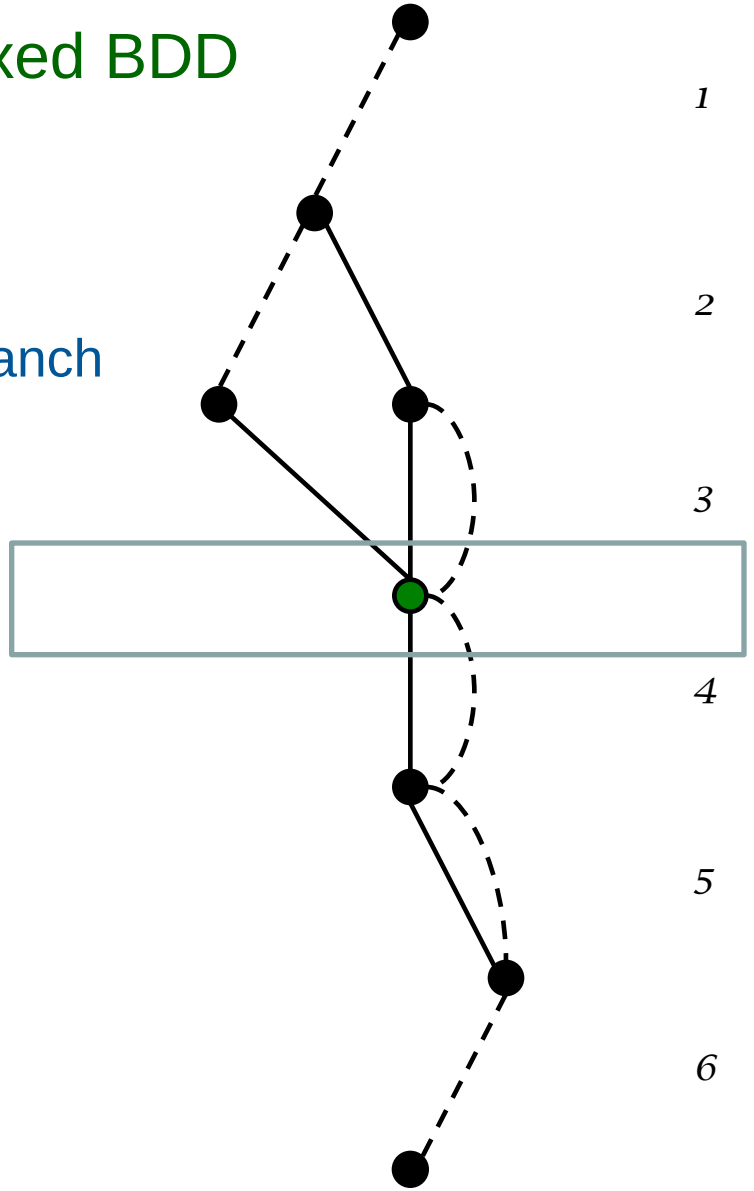
First branch



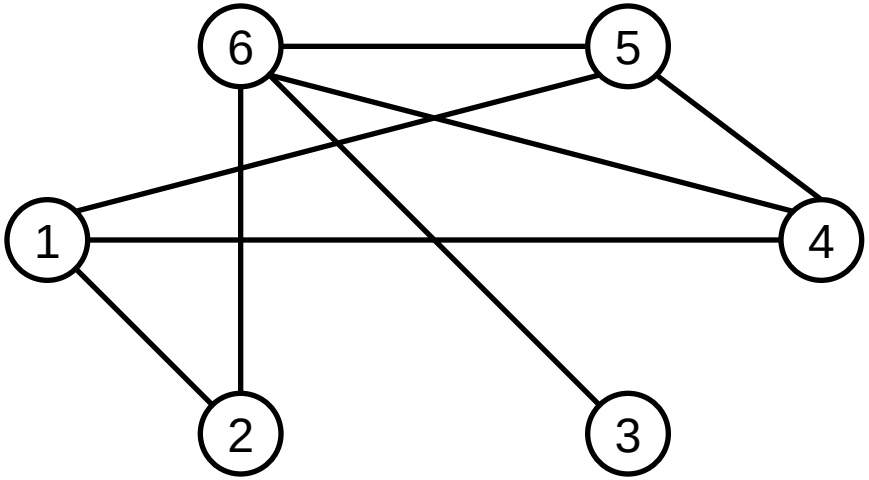
Relaxed BDD



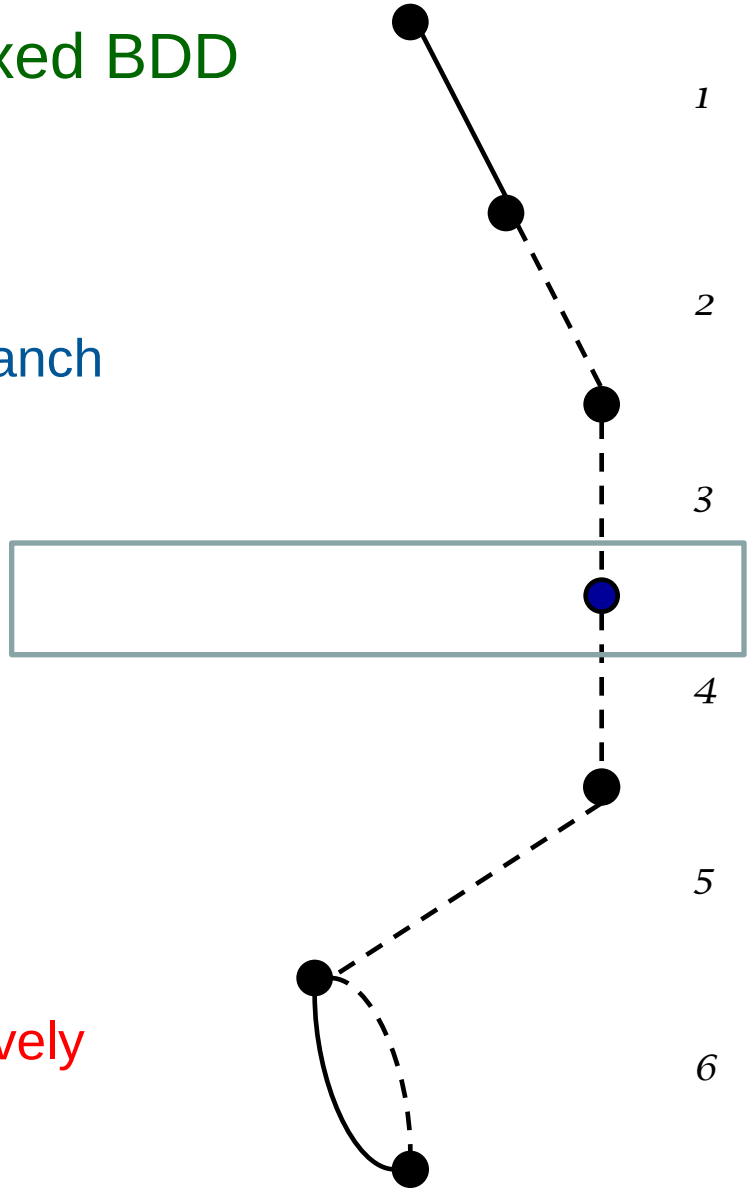
Second branch



Relaxed BDD



Third branch



Continue recursively

Parallelization

- BDD-based optimization parallelizes very efficiently
  - Near-linear speedups.
  - Details in Bergman, Ciré, Sabharwal, Samulowitz, Saraswat, van Hoeve, CPAIOR 2014.

Recursive Modeling

- Flexibility of dynamic programming
  - Without having to search an exponential state space.
- Potential for “big data” optimization
  - No need to “load” a huge model, as in integer programming.
  - Recursive model is essentially constant size.

Recursive Modeling

- Example: simple one-machine sequencing problem.
  - At least 6 integer programming models.
  - Very simple DP state
    - set of jobs not yet sequenced
    - finish time of last job processed.

Recursive Modeling

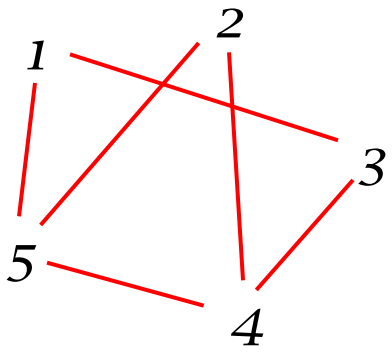
- Example: simple one-machine sequencing problem.
  - At least 6 integer programming models.
  - Very simple DP state
    - set of jobs not yet sequenced
    - finish time of last job processed.
- Formulate constraints that are hard to write in IP.
  - For example, job cannot be processed until certain components are available from earlier jobs.
  - Maintenance schedule, conditional time windows, etc.
  - Easily written in terms of current state.

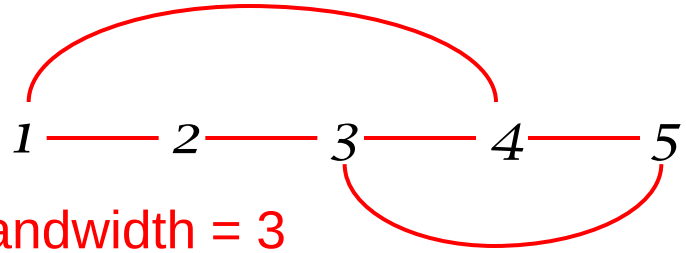
Recursive Modeling

- Example: simple one-machine sequencing problem.
  - At least 6 integer programming models.
  - Very simple DP state
    - set of jobs not yet sequenced
    - finish time of last job processed.
- Formulate constraints that are hard to write in IP.
  - For example, job cannot be processed until certain components are available from earlier jobs.
  - Maintenance schedule, conditional time windows, etc.
  - Easily written in terms of current state.
- Nonstandard objective functions.
  - Job processing cost depends on which components available from earlier jobs.
  - Can be table lookup function.

Recursive Modeling

- Min bandwidth and linear arrangement problems.
  - Sequence electronic components.
  - Some pairs are connected by a wire.
  - Minimize longest wire, or sum of wire lengths.



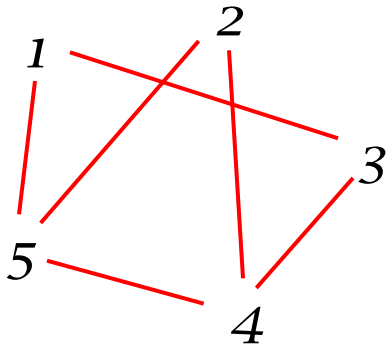


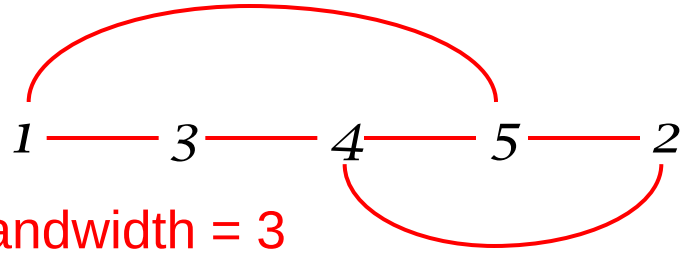
Bandwidth = 3

sum = 9

Recursive Modeling

- Min bandwidth and linear arrangement problems.
  - Sequence electronic components.
  - Some pairs are connected by a wire.
  - Minimize longest wire, or sum of wire lengths.





Bandwidth = 3
sum = 9

- No usable IP model
- Recursive model
  - Control = length of wire connecting pair of components
  - State = (for each component) remaining positions it can be assigned

Future Work

- Continuous global optimization.
  - Recursive models.
  - Massive discretization of continuous variables.
  - Node merger heuristics control discretization density.
  - No need for convexification and Lipschitz bounds.
  - Use “big data” approach of applying simple model to huge dataset (of discrete values).