Optimization Bounds from Binary Decision Diagrams

J. N. Hooker

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

Carnegie Mellon University

Boolean Workshop, April 2013

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 is result of superimposing isomorphic subtrees in a search tree.
  - Unique reduced BDD for given variable ordering.
  - “Caching” is a popular theme in knowledge representation.
- Constraints need not have an inequality representation.

The 0-1 inequality

$$300x_0 + 300x_1 + 285x_2 + 285x_3 + 265x_4 + 265x_5 + 230x_6$$
$$+230x_7 + 190x_8 + 200x_9 + 400x_{10} + 200x_{11} + 400x_{12} + 200x_{13}$$
$$+400x_{14} + 200x_{15} + 400x_{16} + 200x_{17} + 400x_{18} \leq 2700$$

has 117,520 minimal solutions

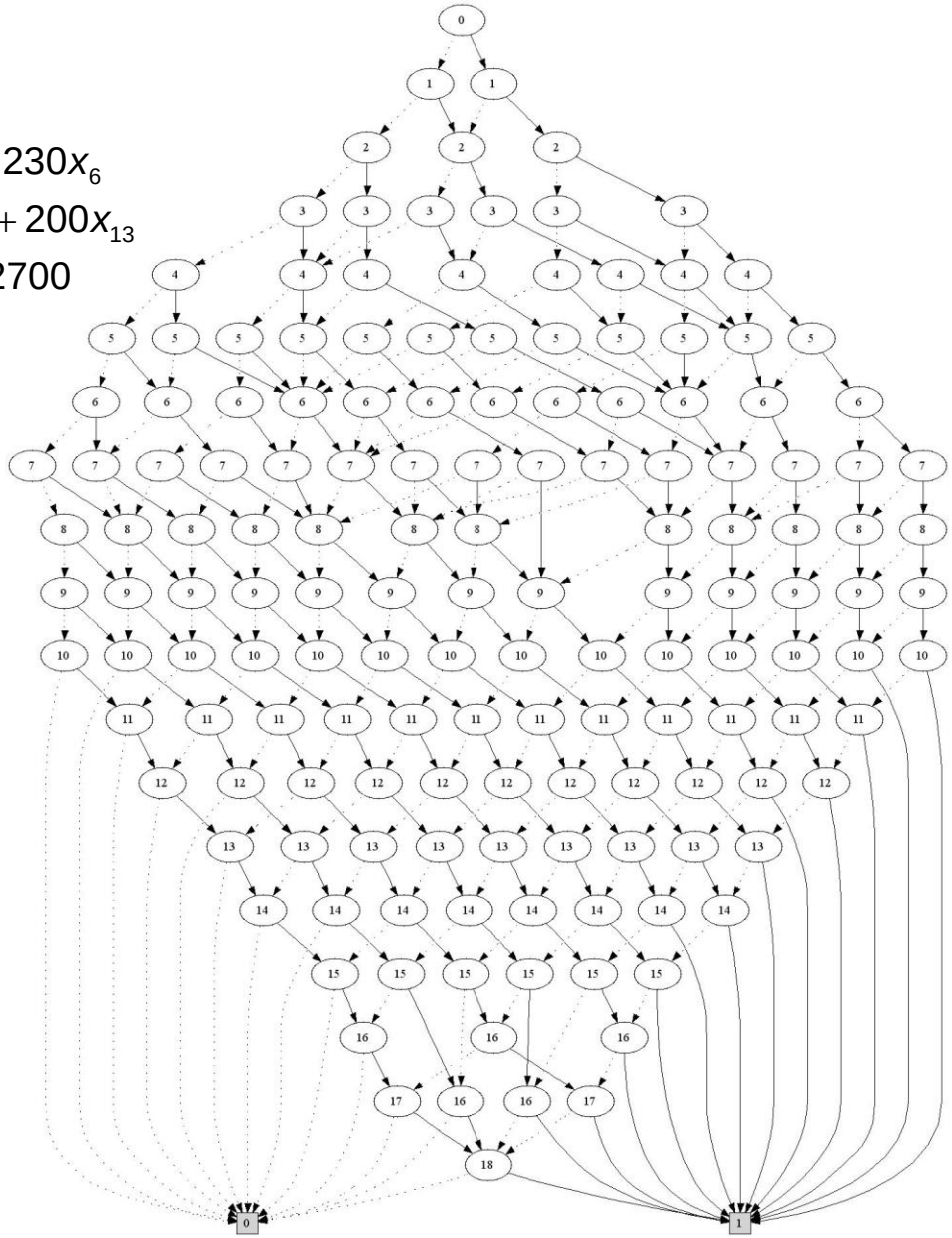
The 0-1 inequality

$$300x_0 + 300x_1 + 285x_2 + 285x_3 + 265x_4 + 265x_5 + 230x_6$$
$$+230x_7 + 190x_8 + 200x_9 + 400x_{10} + 200x_{11} + 400x_{12} + 200x_{13}$$
$$+400x_{14} + 200x_{15} + 400x_{16} + 200x_{17} + 400x_{18} \leq 2700$$

has 117,520 minimal solutions

The BDD has only 152 nodes.

Paths from top to bottom right correspond to feasible solutions



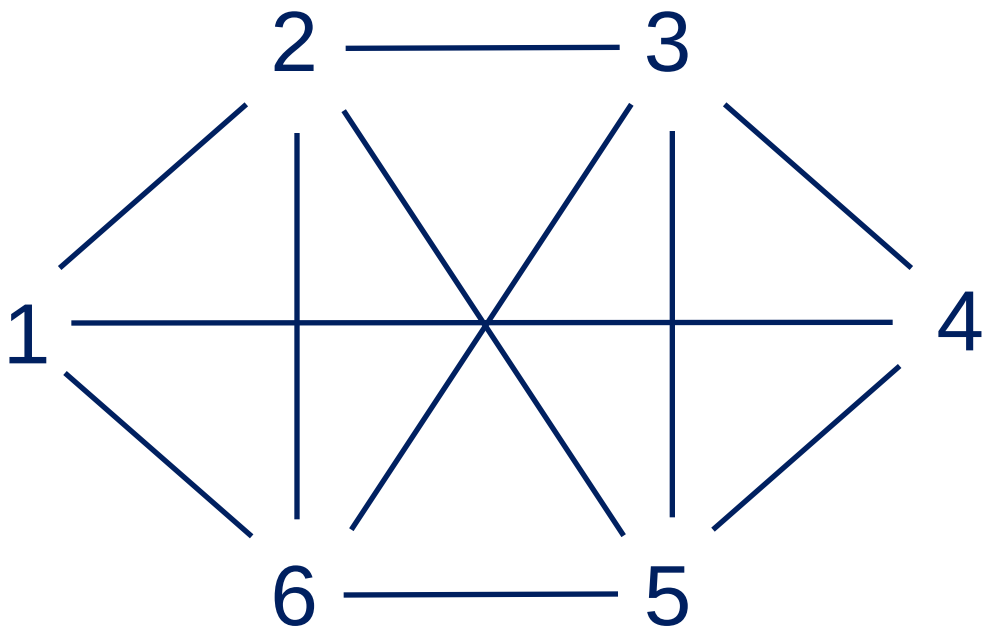
Binary Decision Diagrams

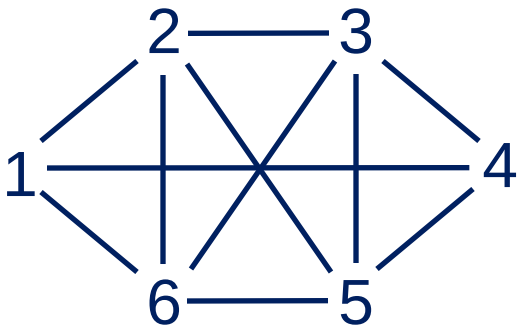
- BDD can grow exponentially with problem size.
  - So we use a smaller, **relaxed** BDD that represents **superset** of feasible set.
    - Andersen, Hadzic, Hooker, Tiedemann 2007.
    - For alldiff systems, reduced search tree from >1 million nodes to 1 node.
    - Subsequent papers with Hadzic, Hoda, van Hoeve, O'Sullivan.
- We focus on **independent set problem** on a graph…

Independent Set Problem

Let each vertex have weight w_i

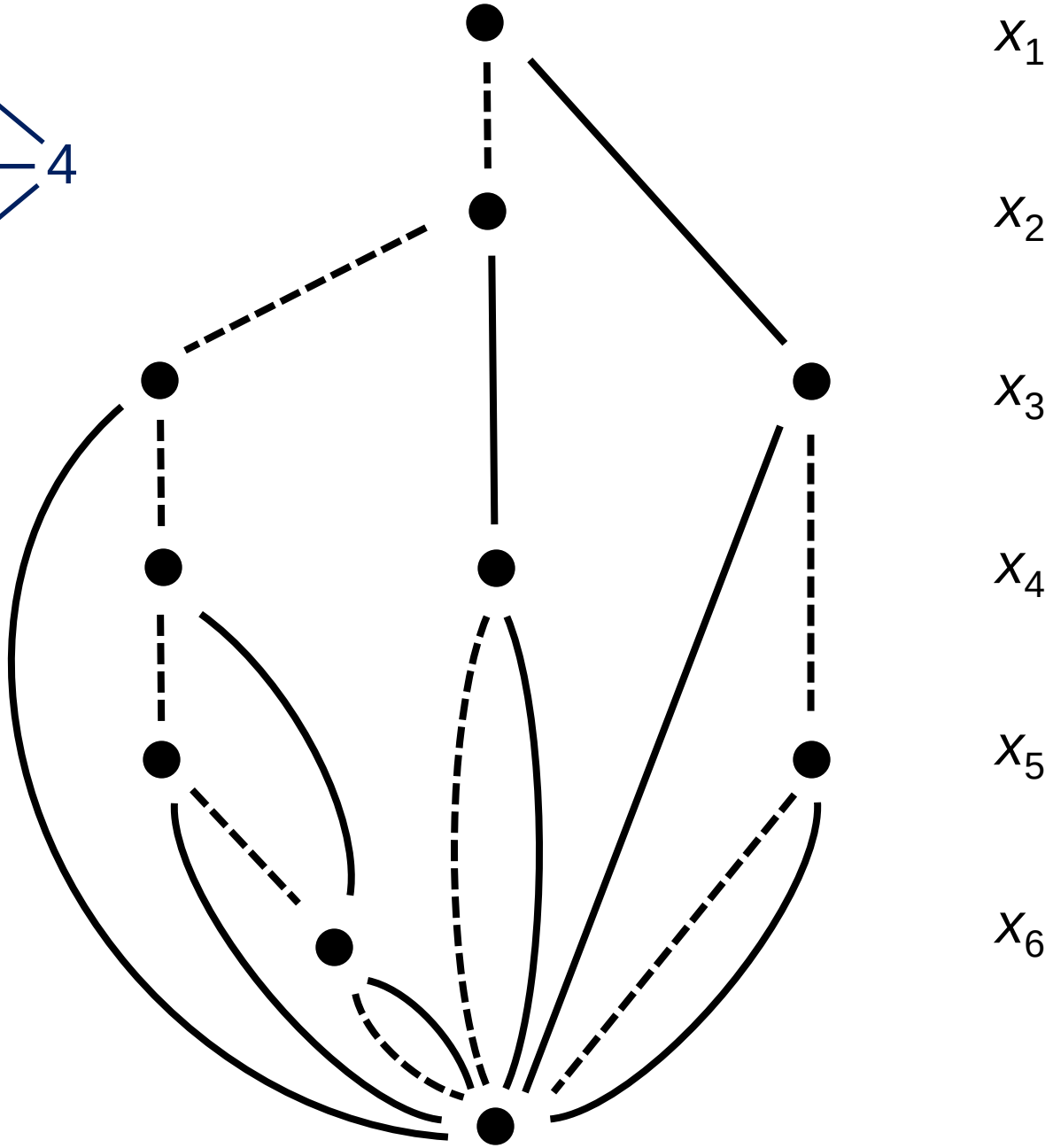
Select nonadjacent vertices to maximize $\sum_i w_i x_i$

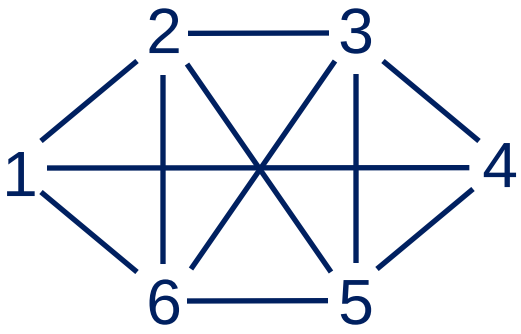




Exact BDD for independent set problem

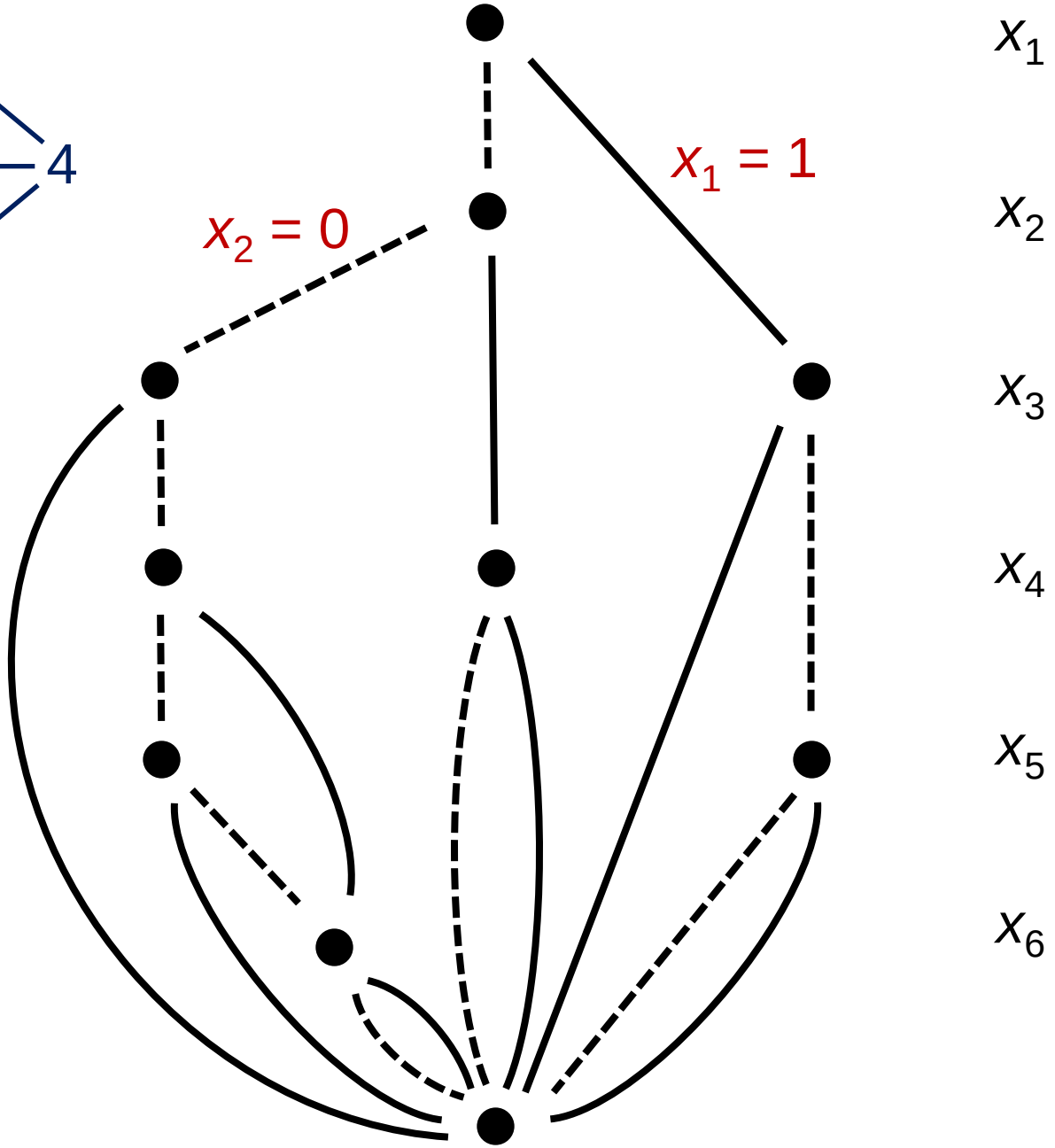
"zero-suppressed" BDD

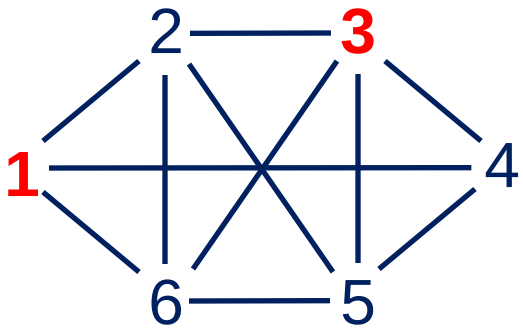




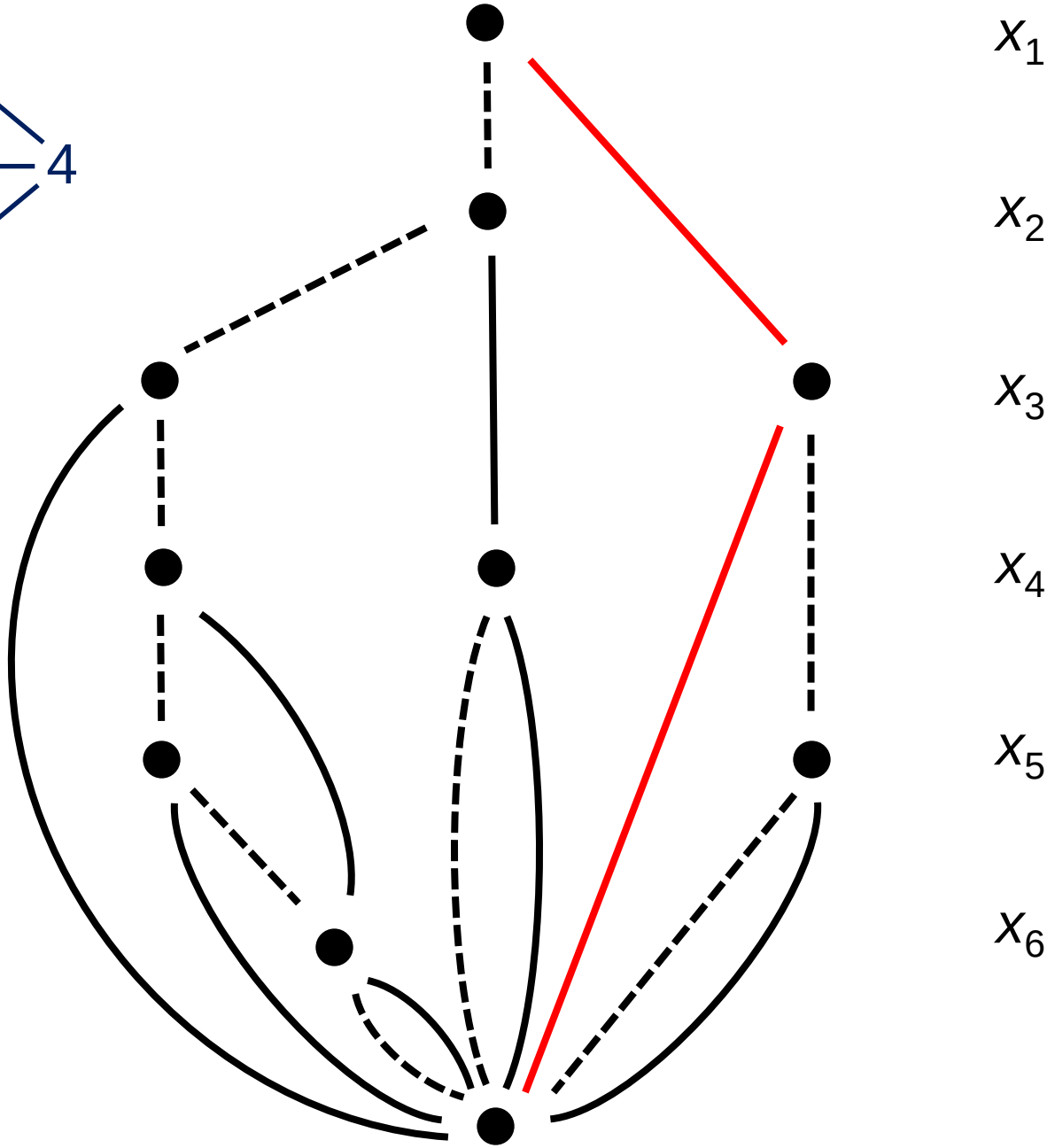
Exact BDD for independent set problem

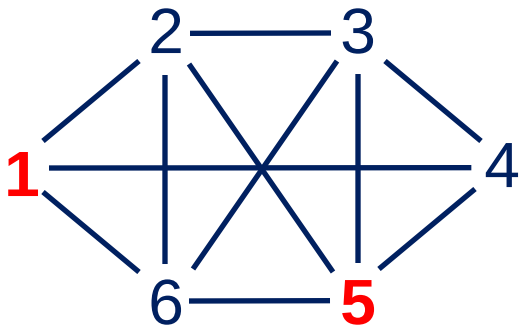
"zero-suppressed" BDD



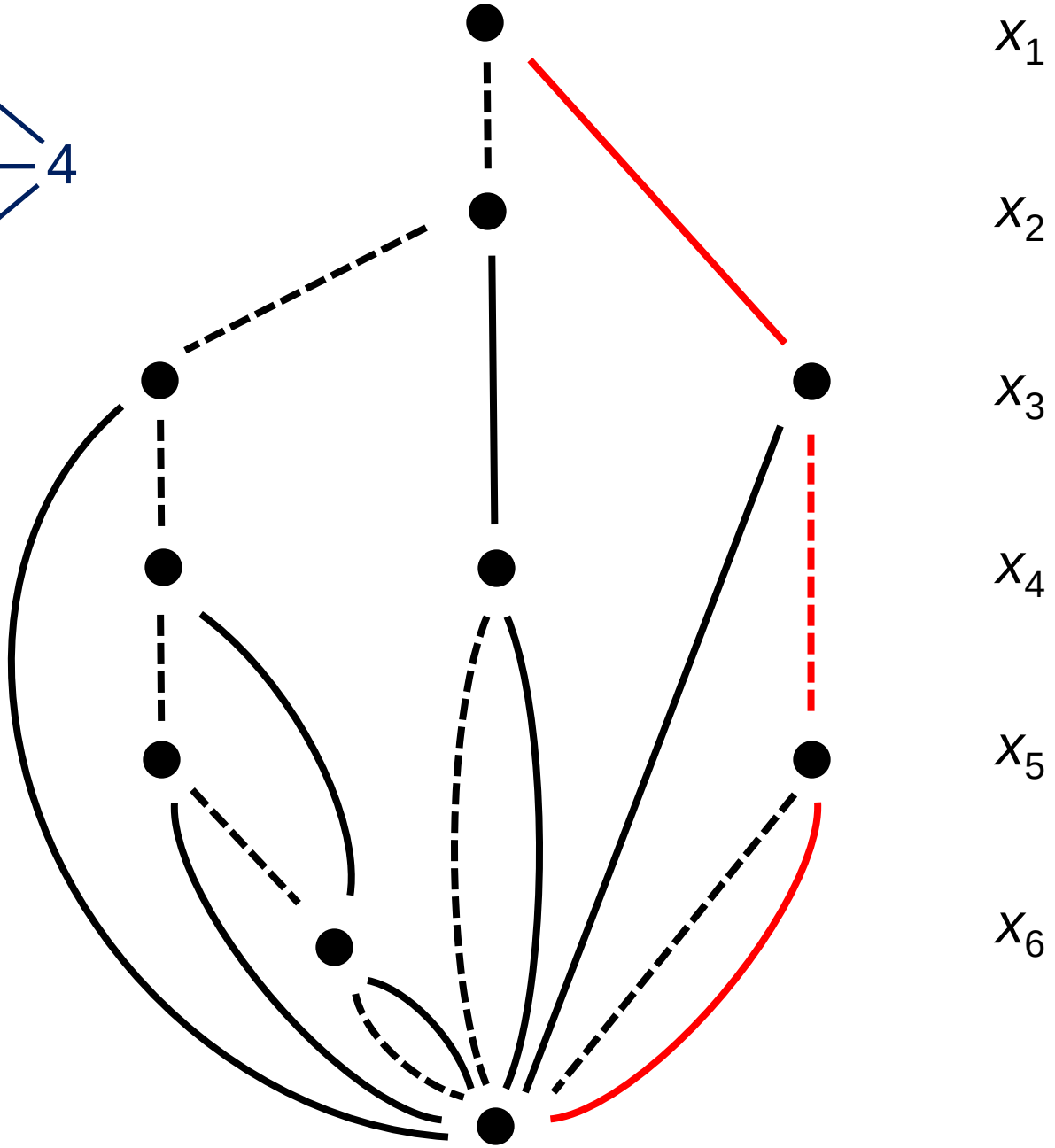


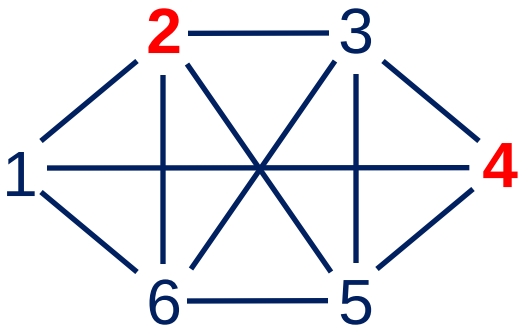
Paths from top to bottom correspond to the 11 feasible solutions



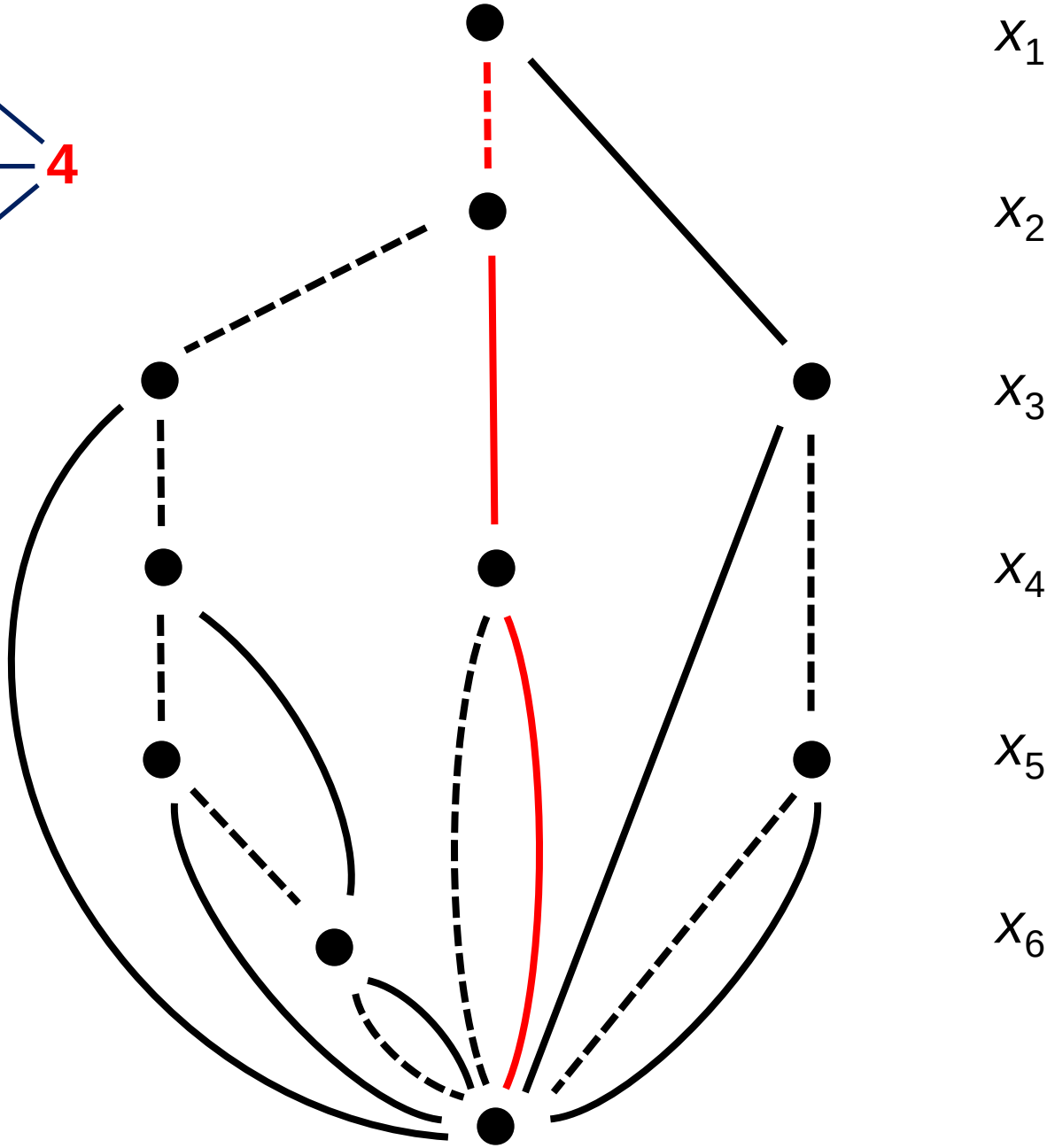


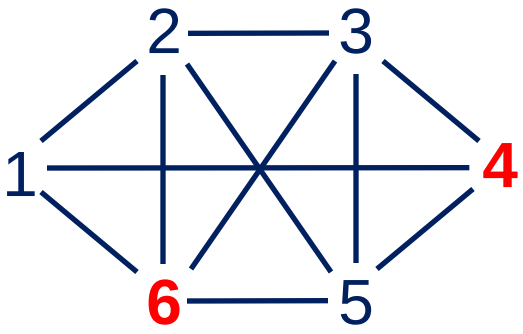
Paths from top to bottom correspond to the 11 feasible solutions



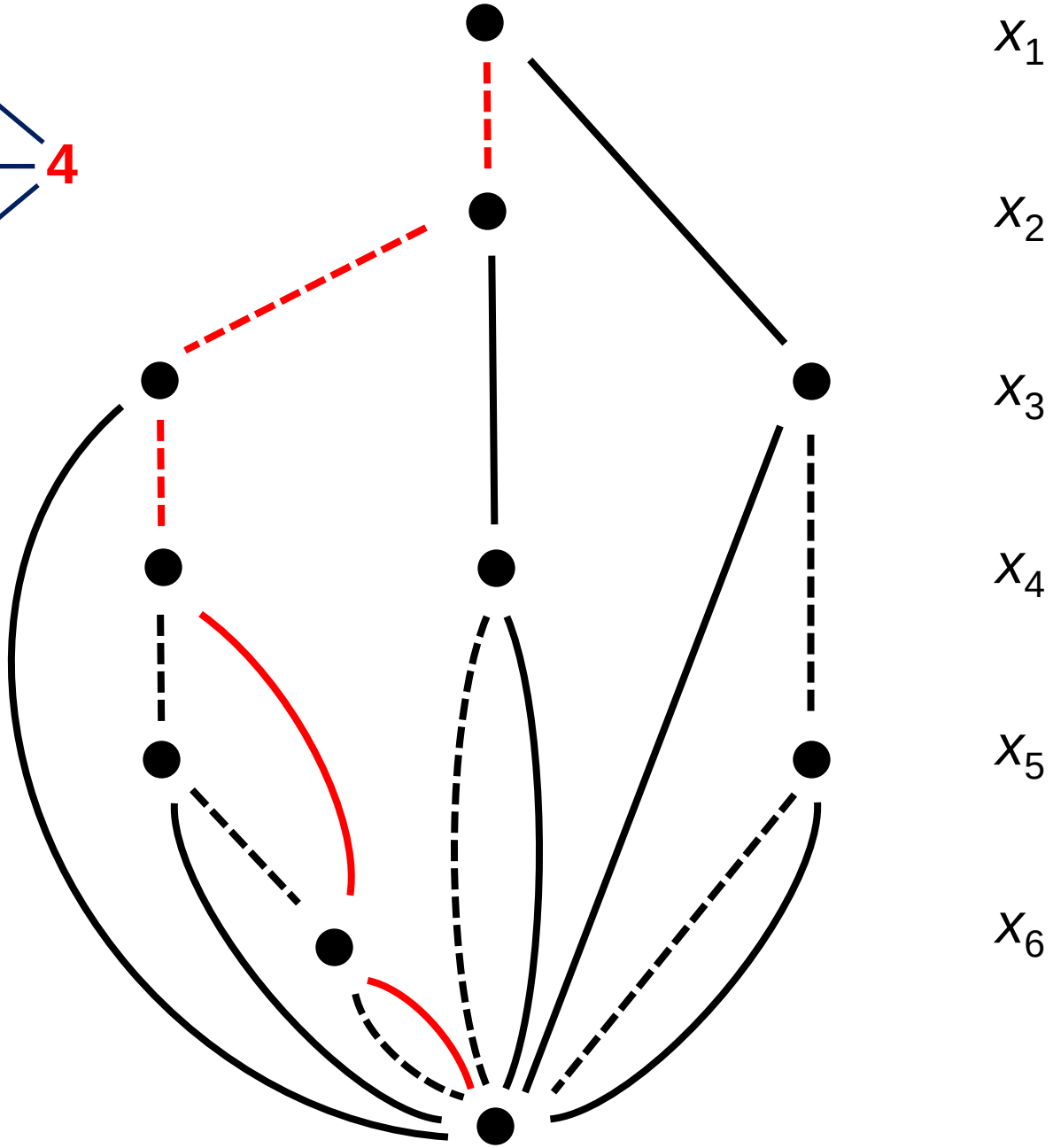


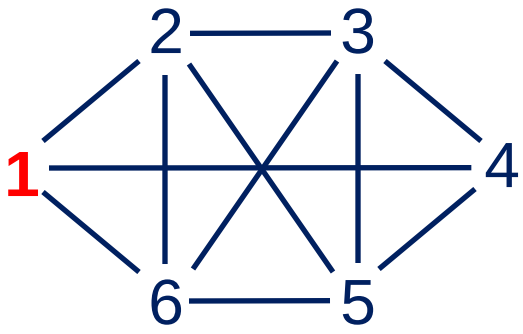
Paths from top to bottom correspond to the 11 feasible solutions





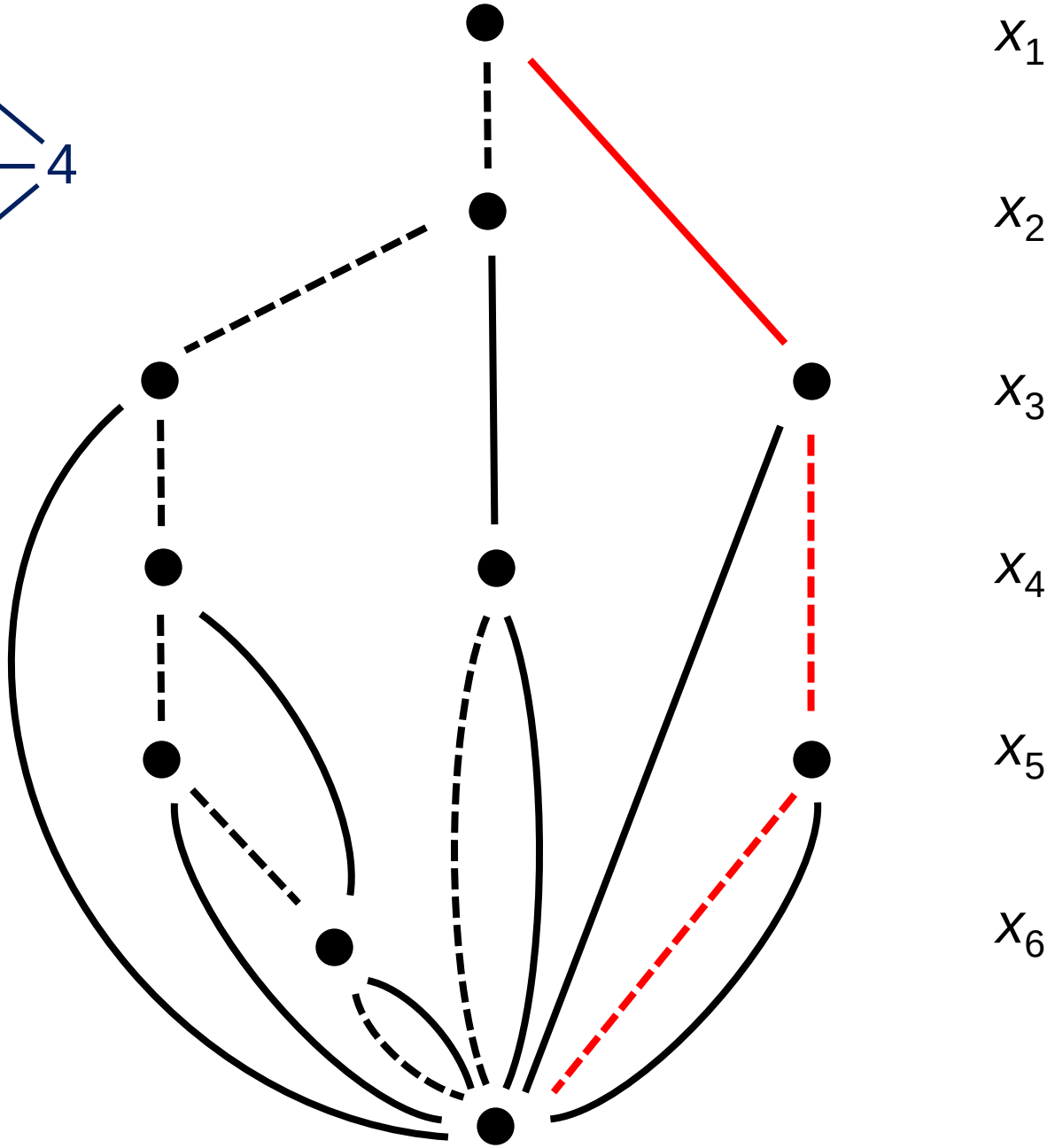
Paths from top to bottom correspond to the 11 feasible solutions

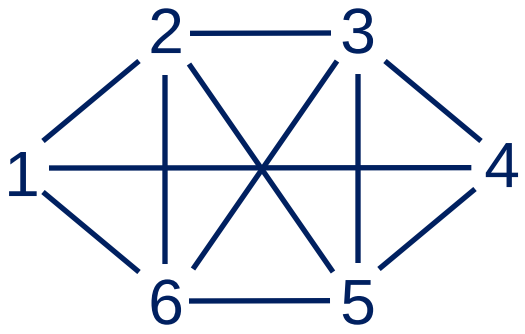




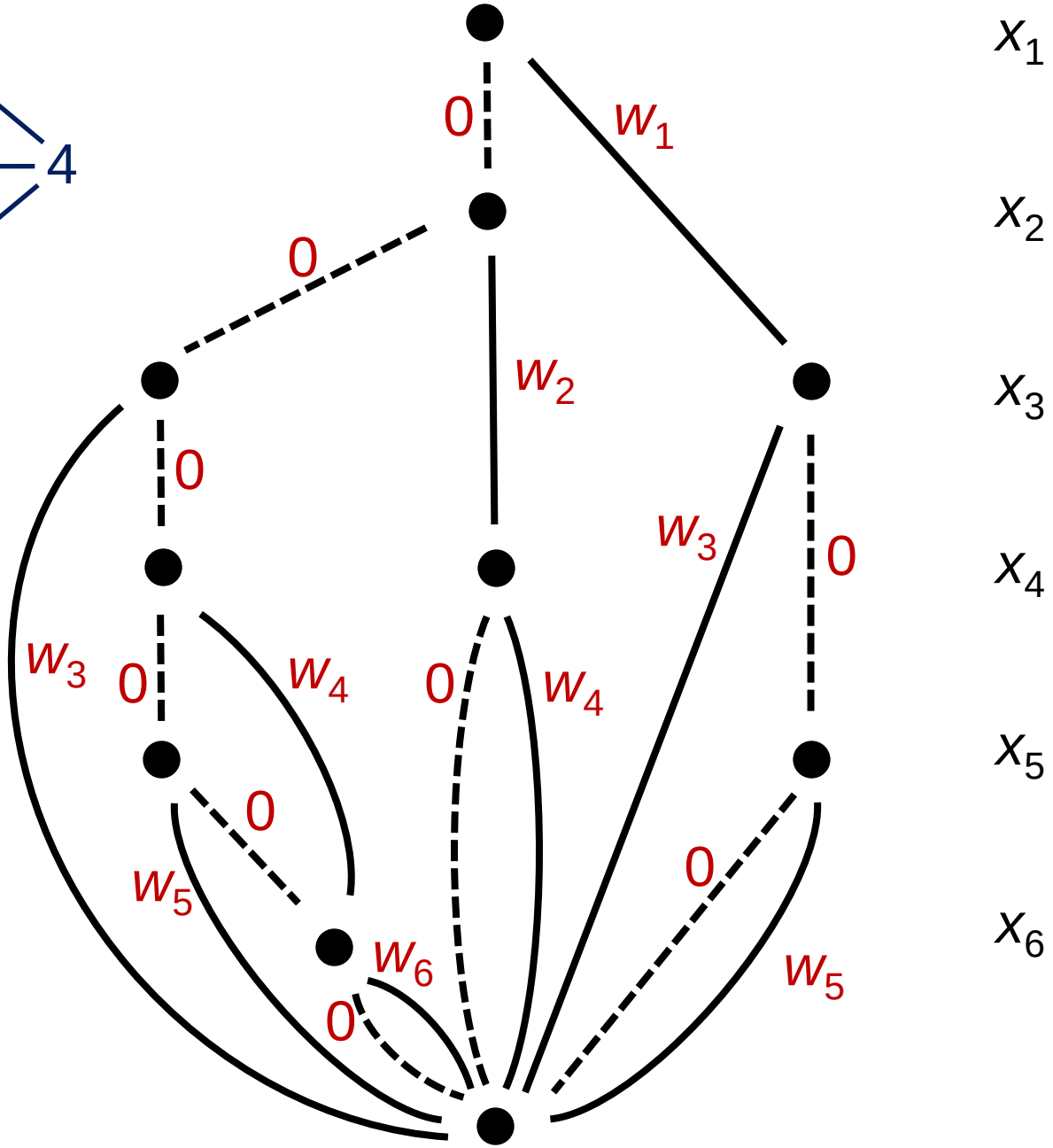
Paths from top to bottom correspond to the 11 feasible solutions

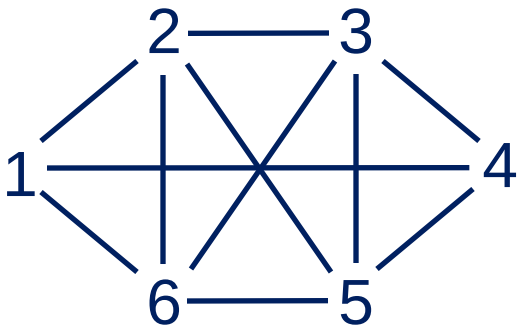
…and so forth





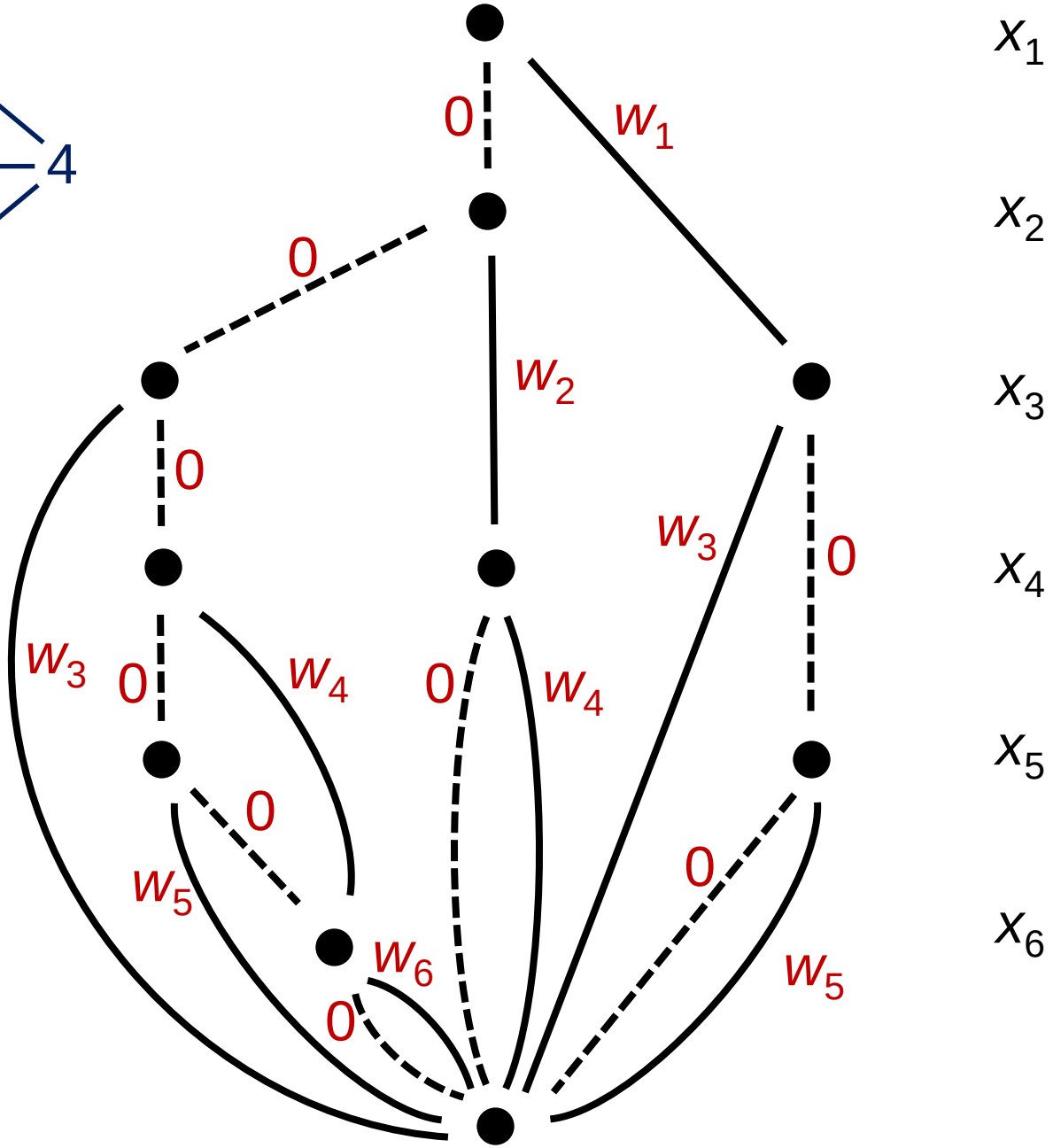
For objective function, associate weights with arcs





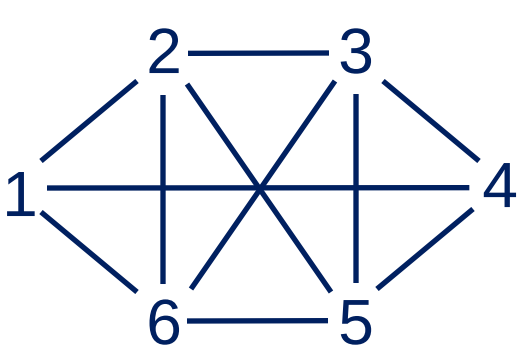
For objective function, associate weights with arcs

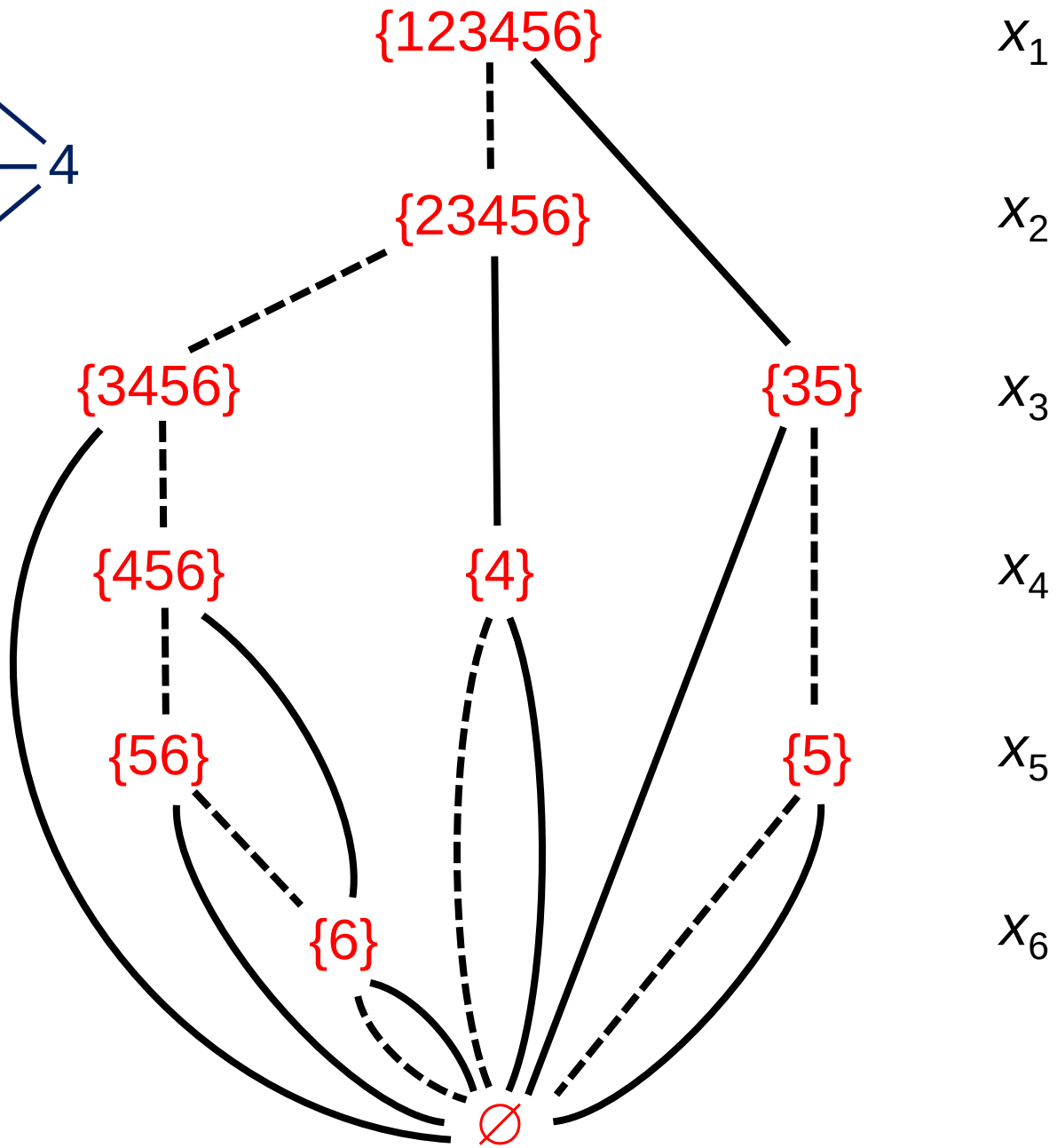
Optimal solution is **longest path**



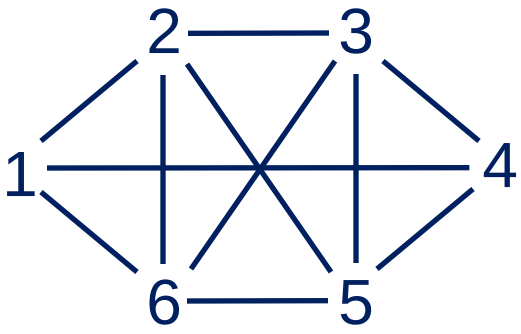
Objective Function

- In general, objective function can be any **separable function.**
  - Linear or nonlinear, convex or nonconvex
- BDDs can be generalized to **nonseparable** objective functions**.**
  - There is a unique reduced BDD with **canonical** edge costs.





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



{123456}

x_1

x_2

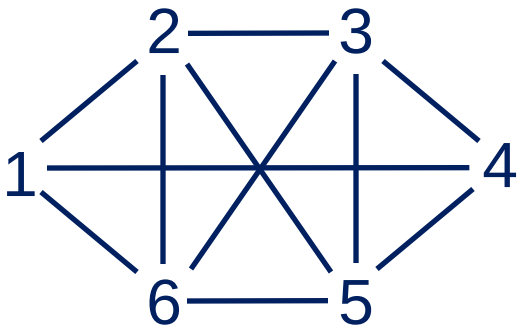
x_3

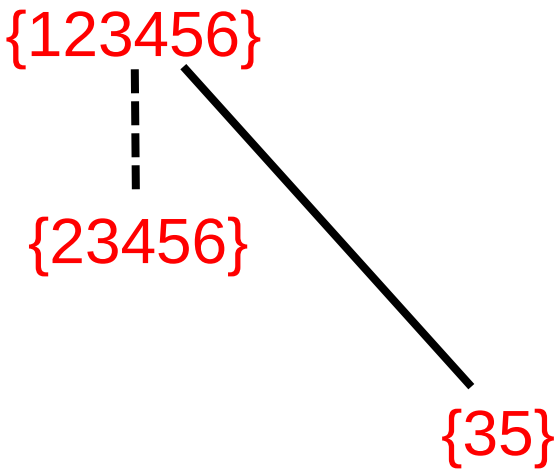
x_4

x_5

x_6

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





x_1

x_2

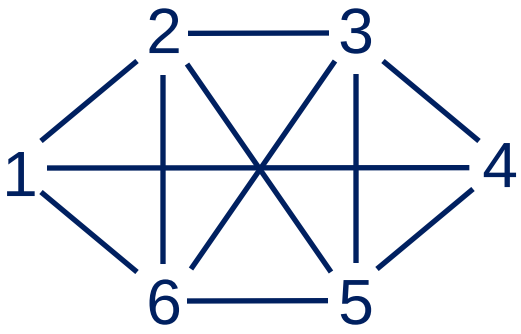
x_3

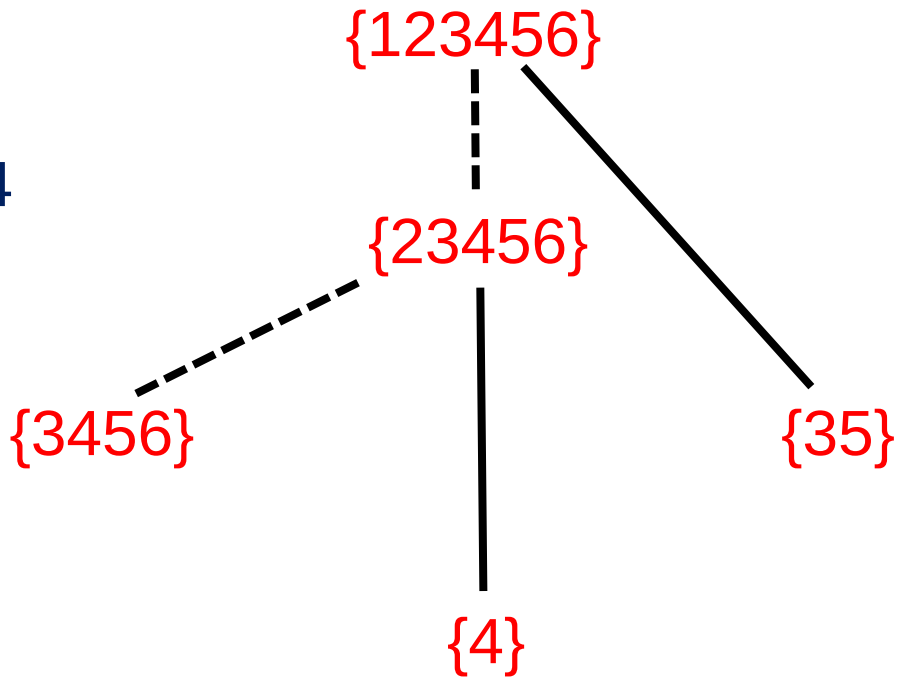
x_4

x_5

x_6

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





x_1

x_2

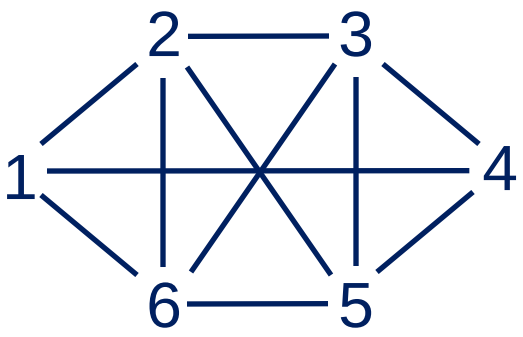
x_3

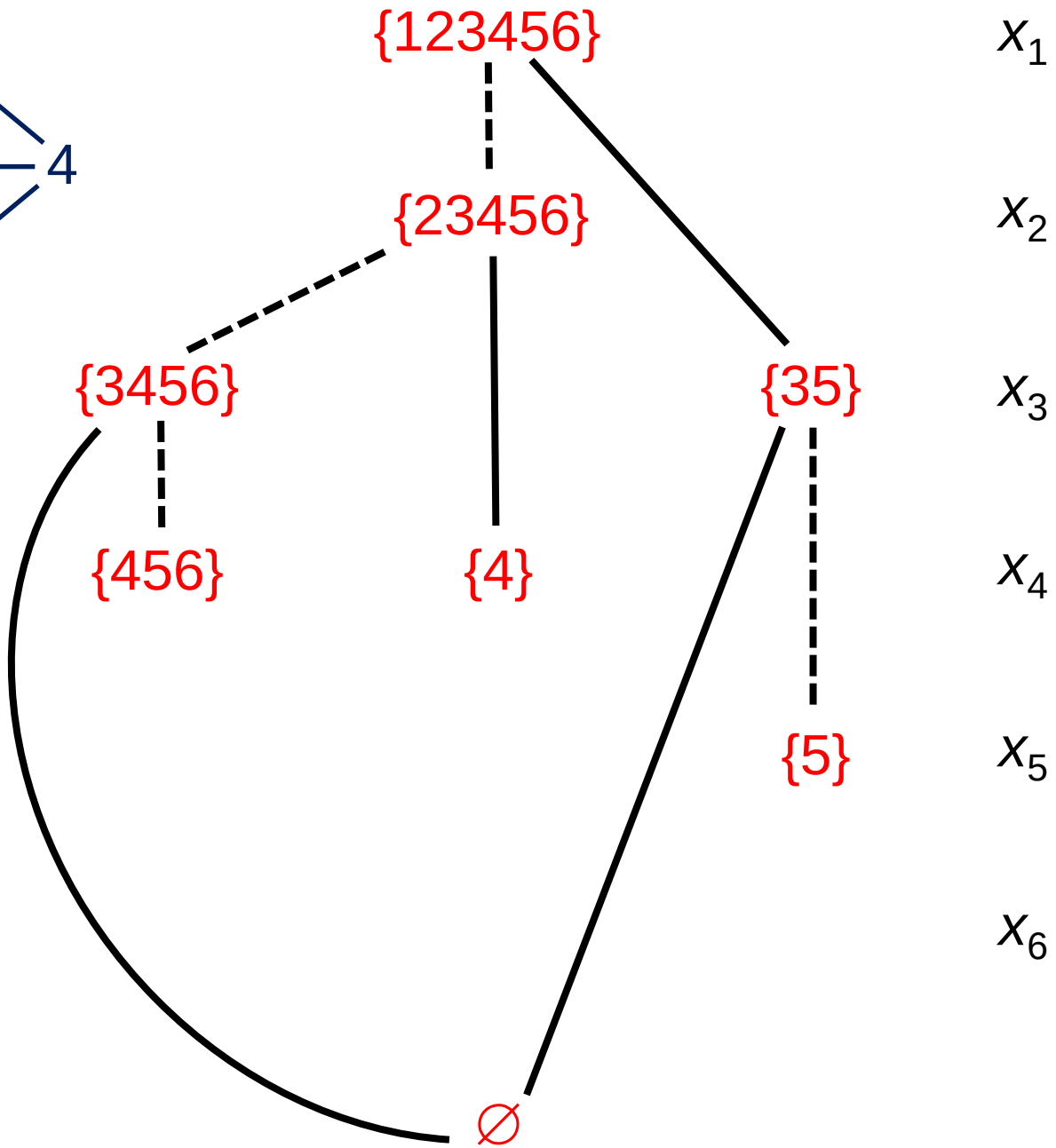
x_4

x_5

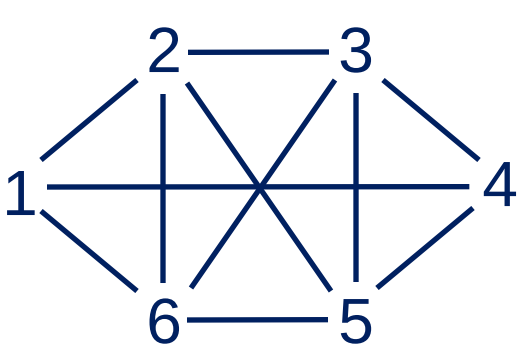
x_6

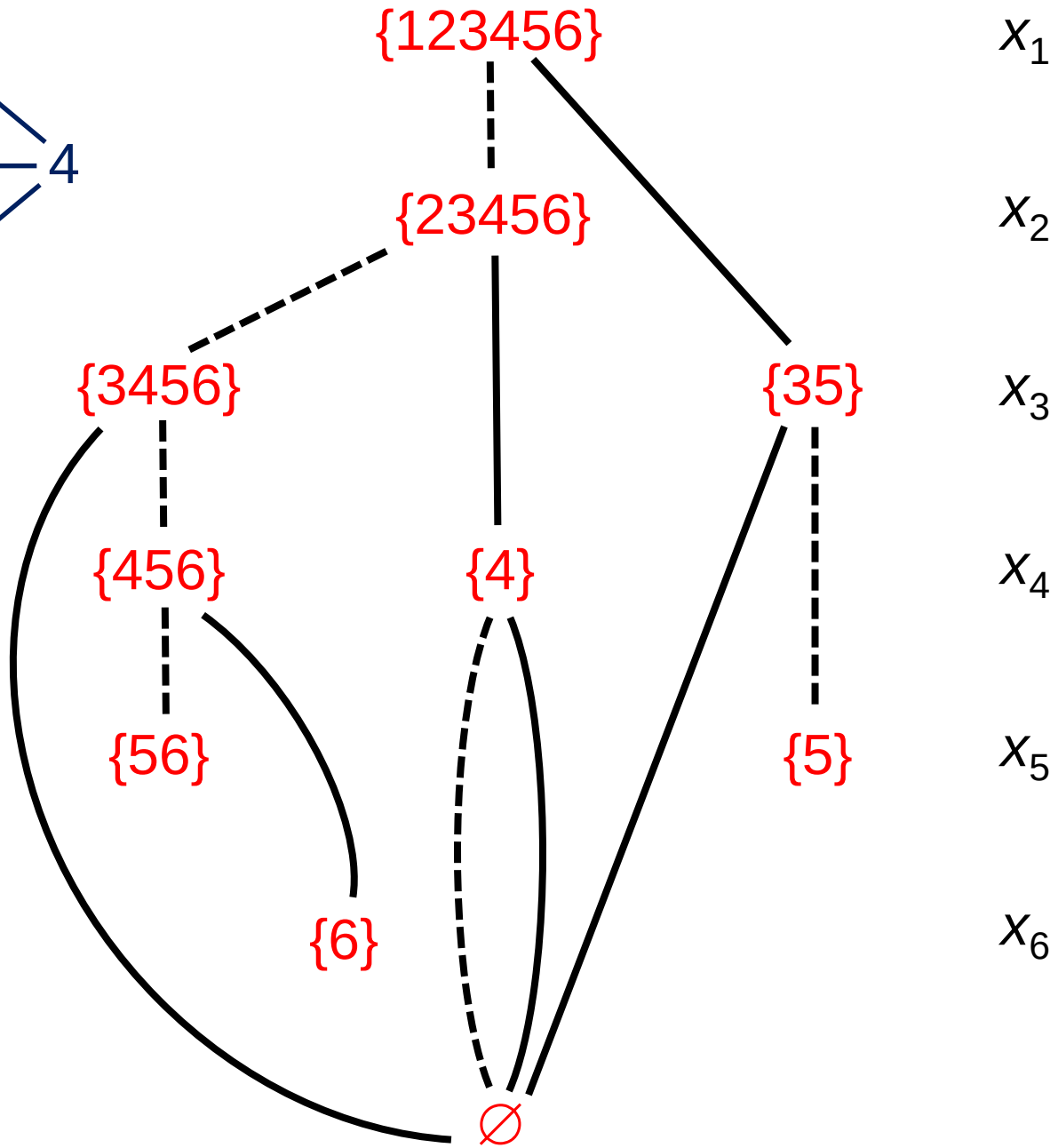
To build BDD, associate **state** with each node



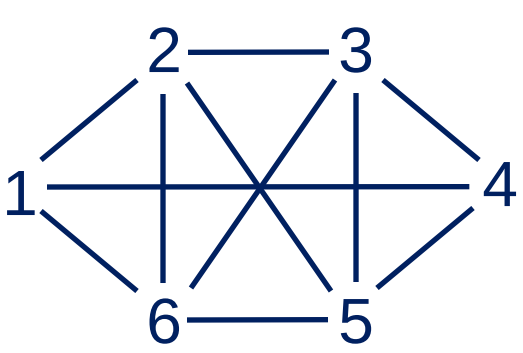


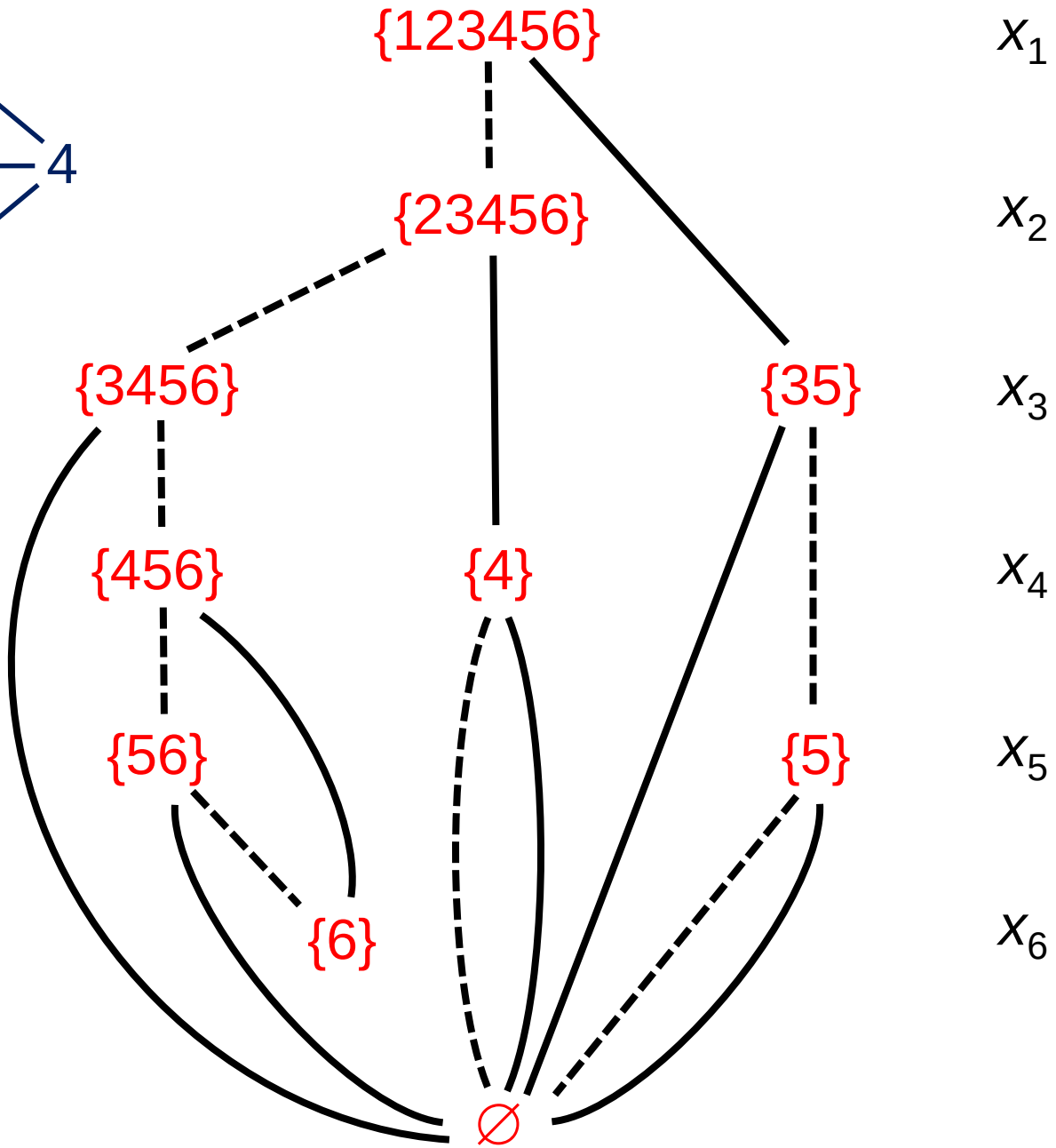
Merge nodes that correspond to the same state



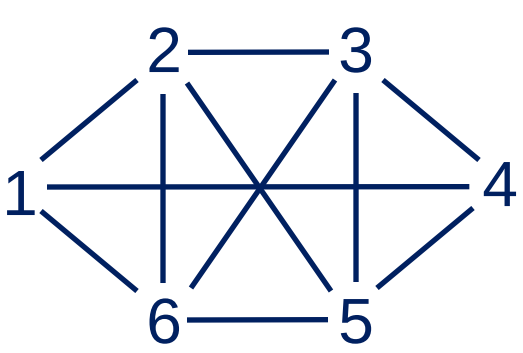


Merge nodes that correspond to the same state



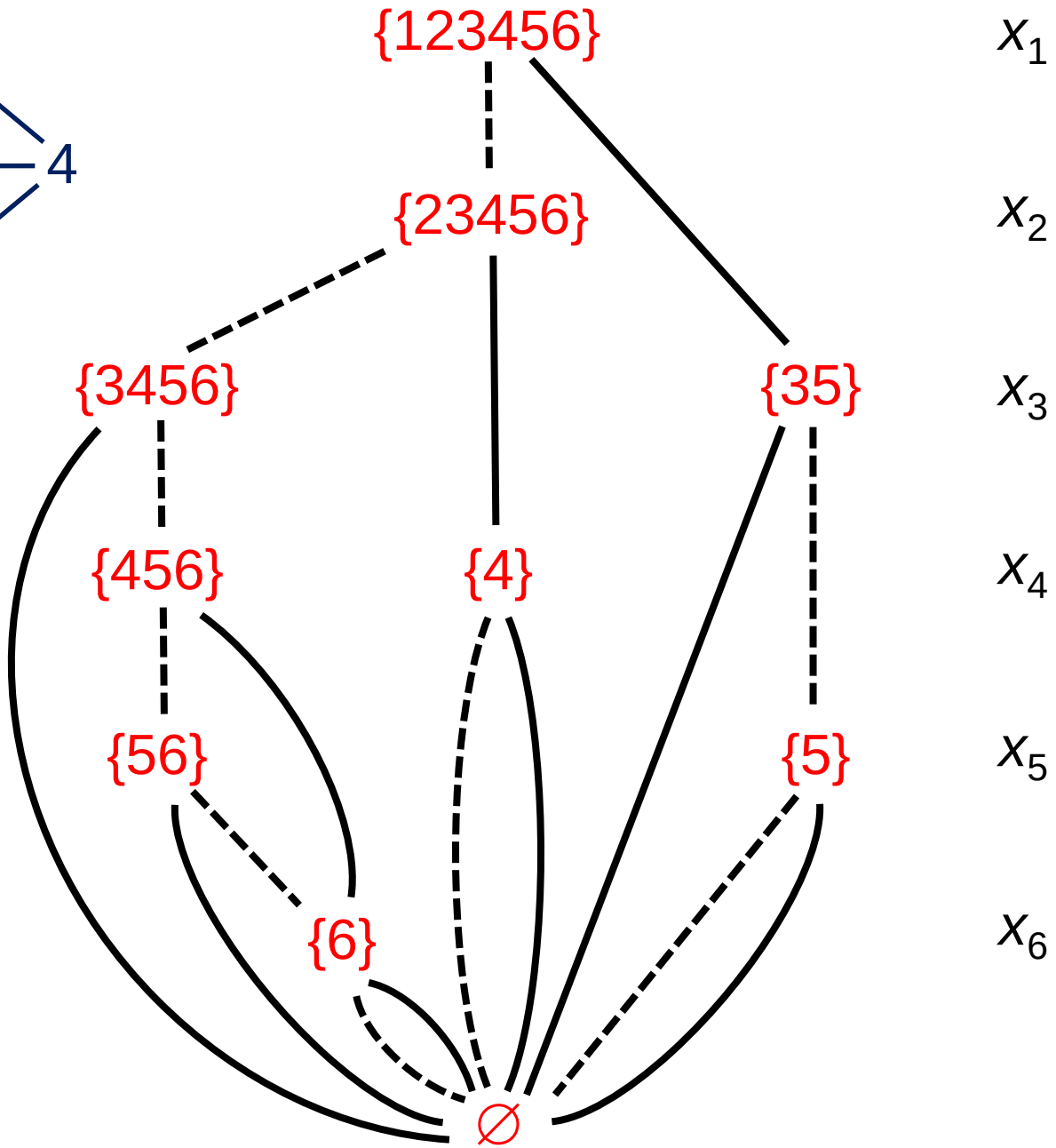


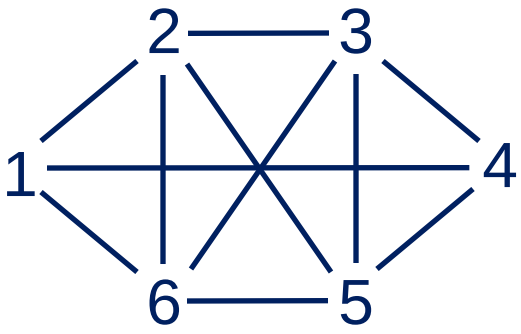
Merge nodes that correspond to the same state



Width = 2

Merge nodes that correspond to the same state





{123456}

x_1

x_2

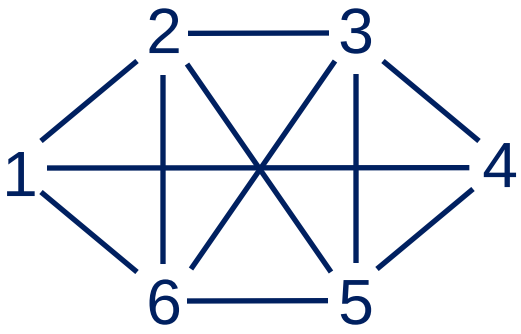
x_3

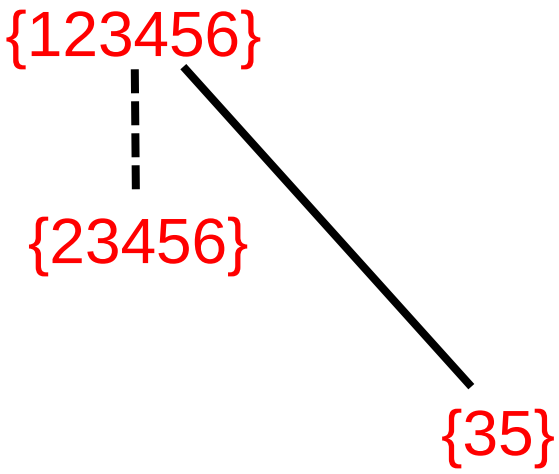
x_4

x_5

x_6

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





x_1

x_2

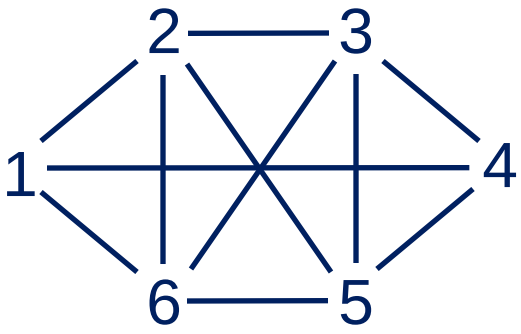
x_3

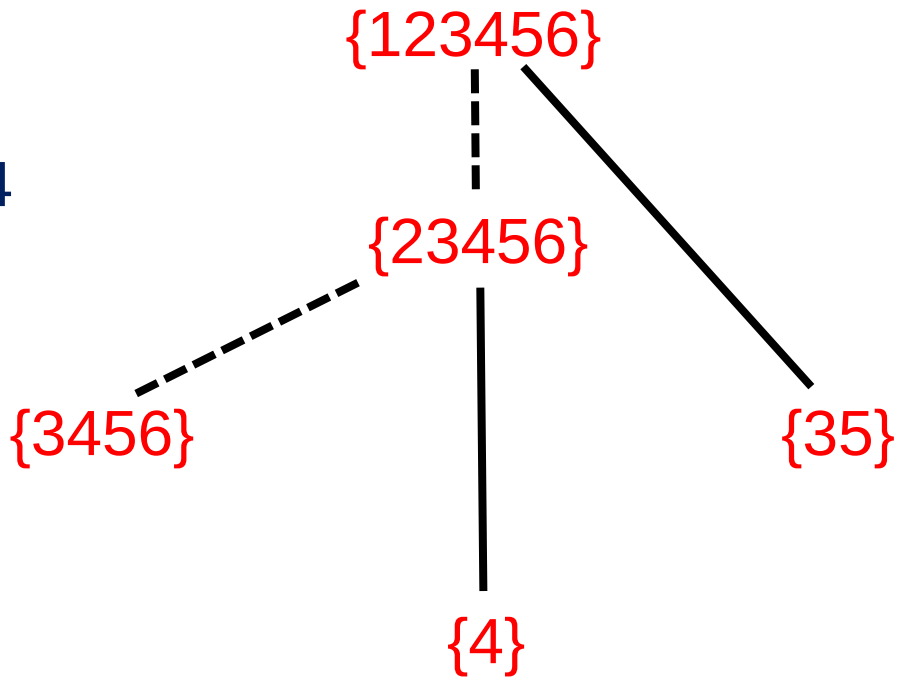
x_4

x_5

x_6

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





x_1

x_2

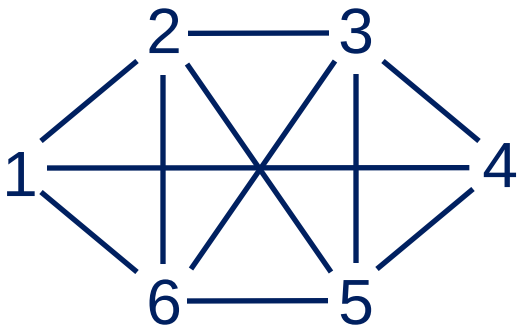
x_3

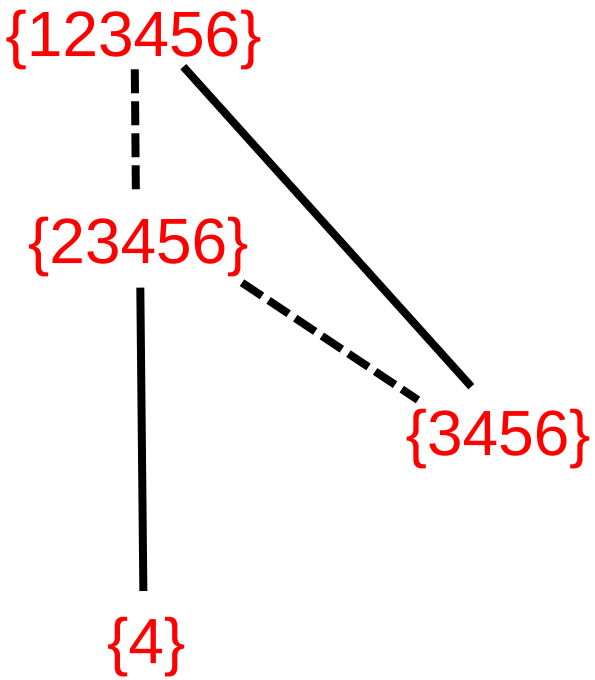
x_4

x_5

x_6

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





x_1

x_2

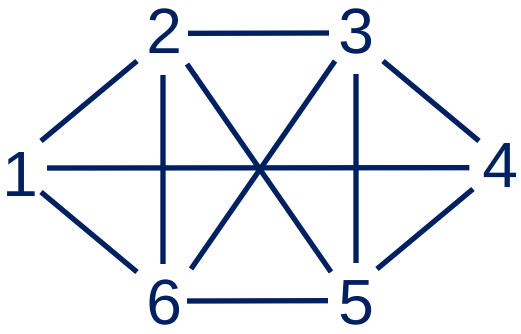
x_3

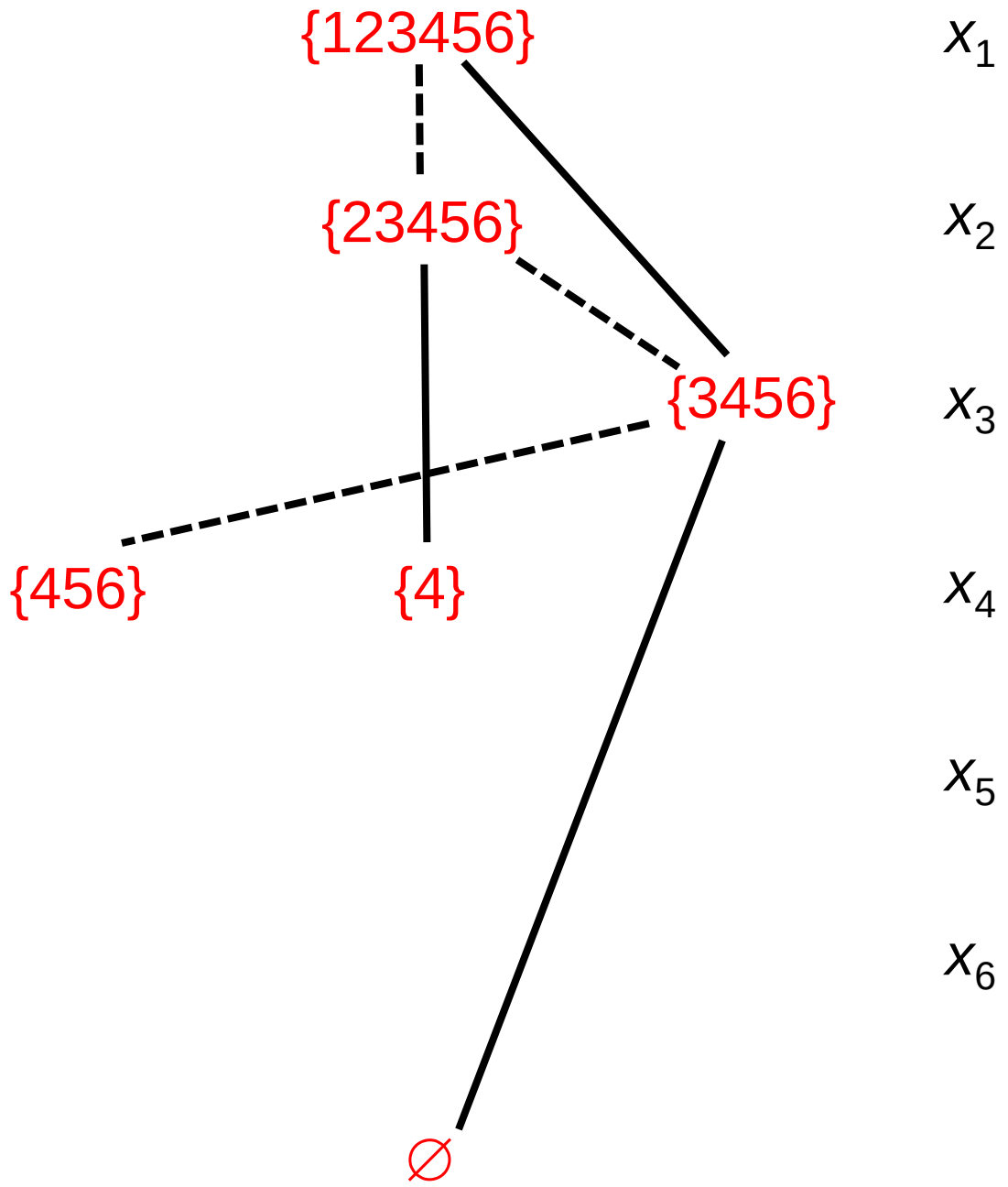
x_4

x_5

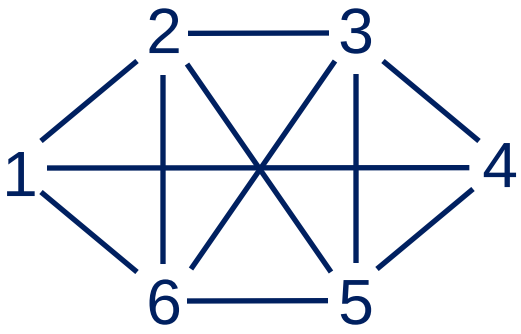
x_6

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

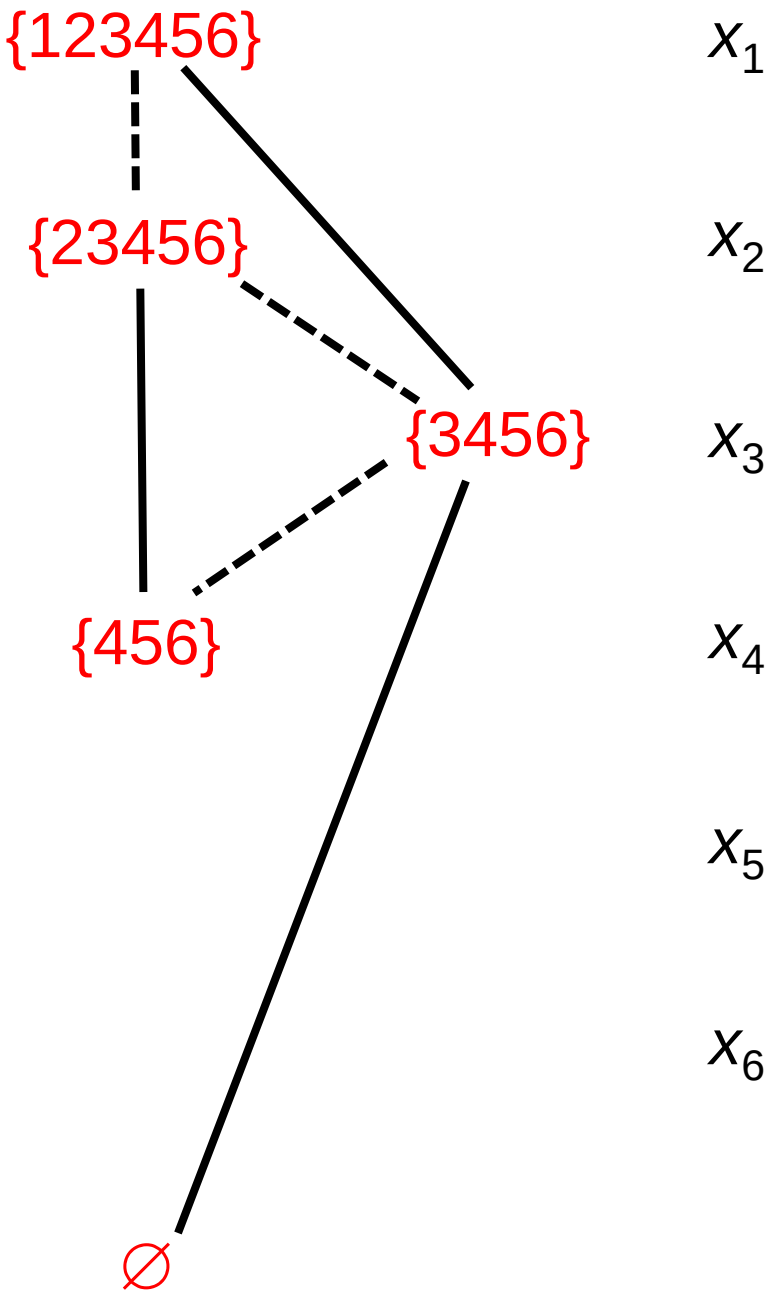


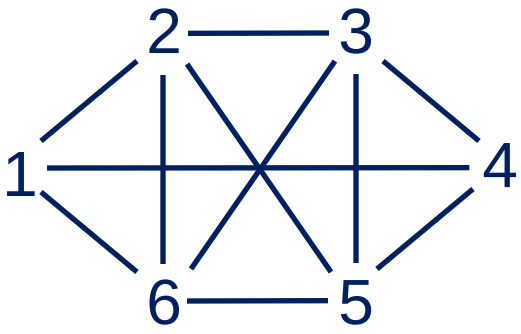


To build **relaxed** BDD, merge some additional nodes as we go along

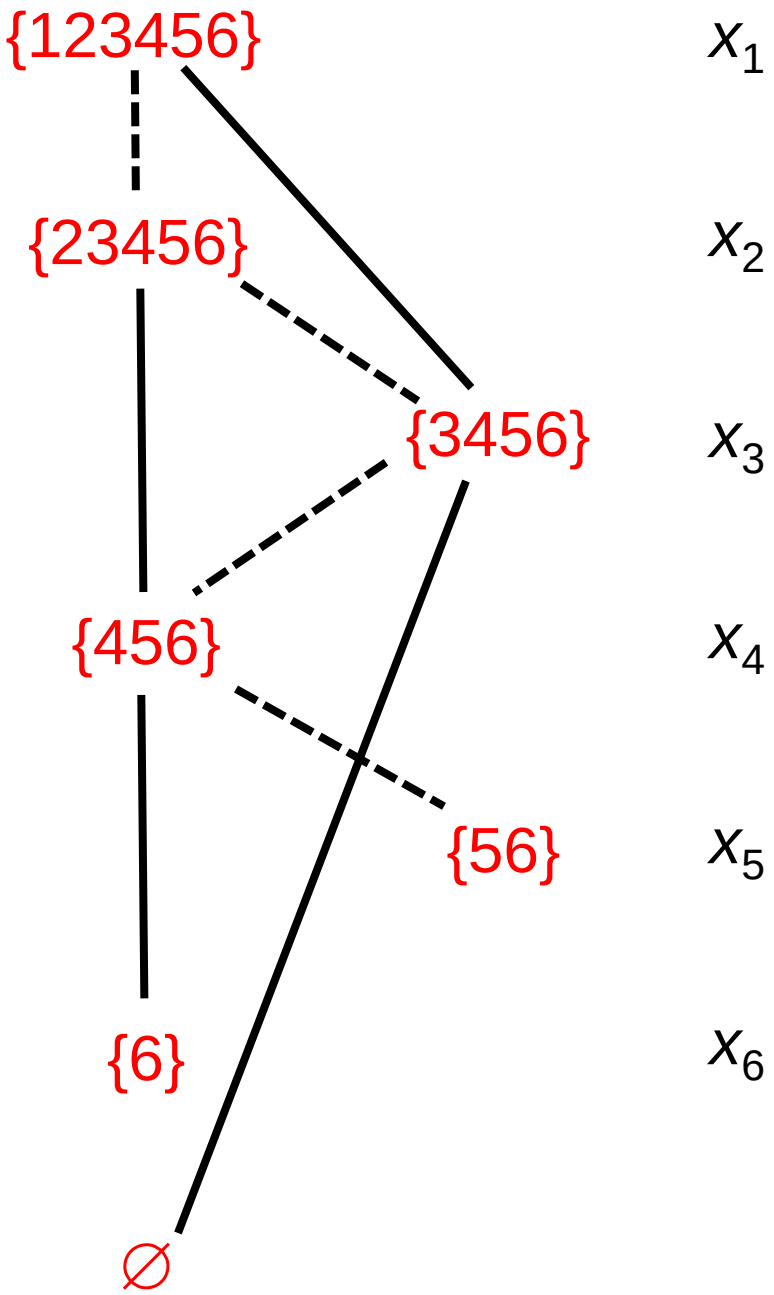


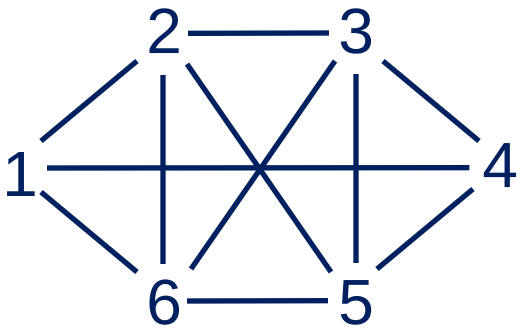
To build **relaxed** BDD, merge some additional nodes as we go along



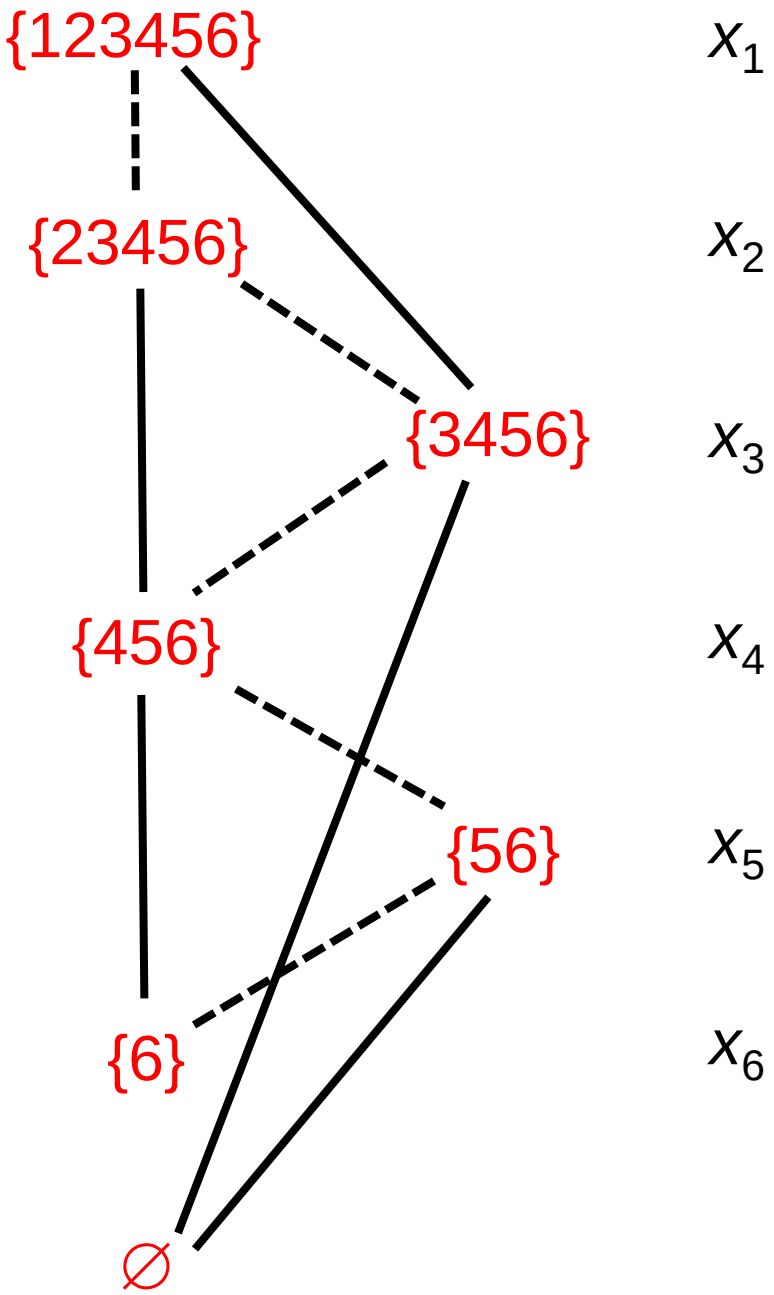


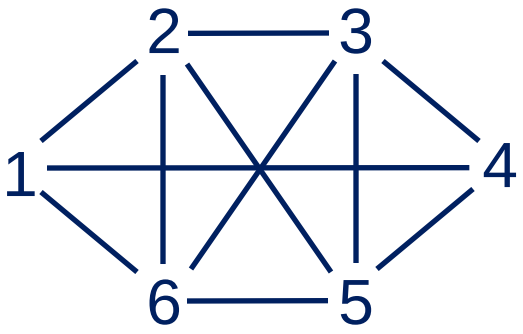
To build **relaxed** BDD, merge some additional nodes as we go along





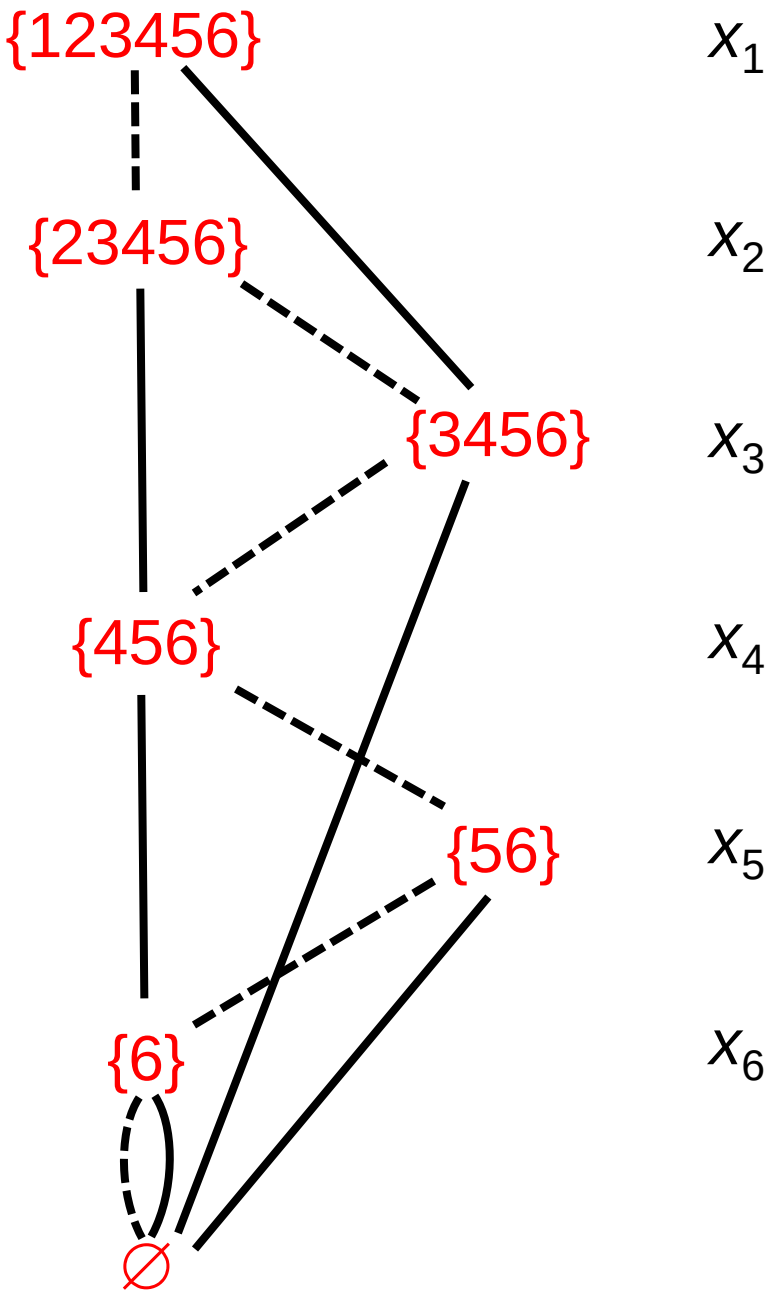
To build **relaxed** BDD, merge some additional nodes as we go along

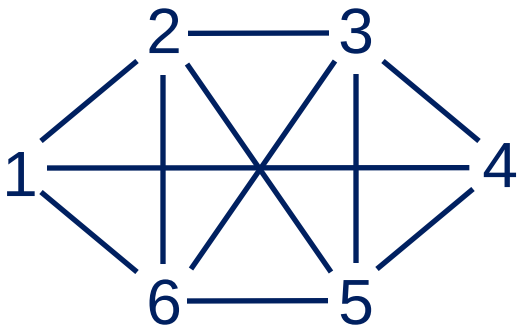




Width = 1

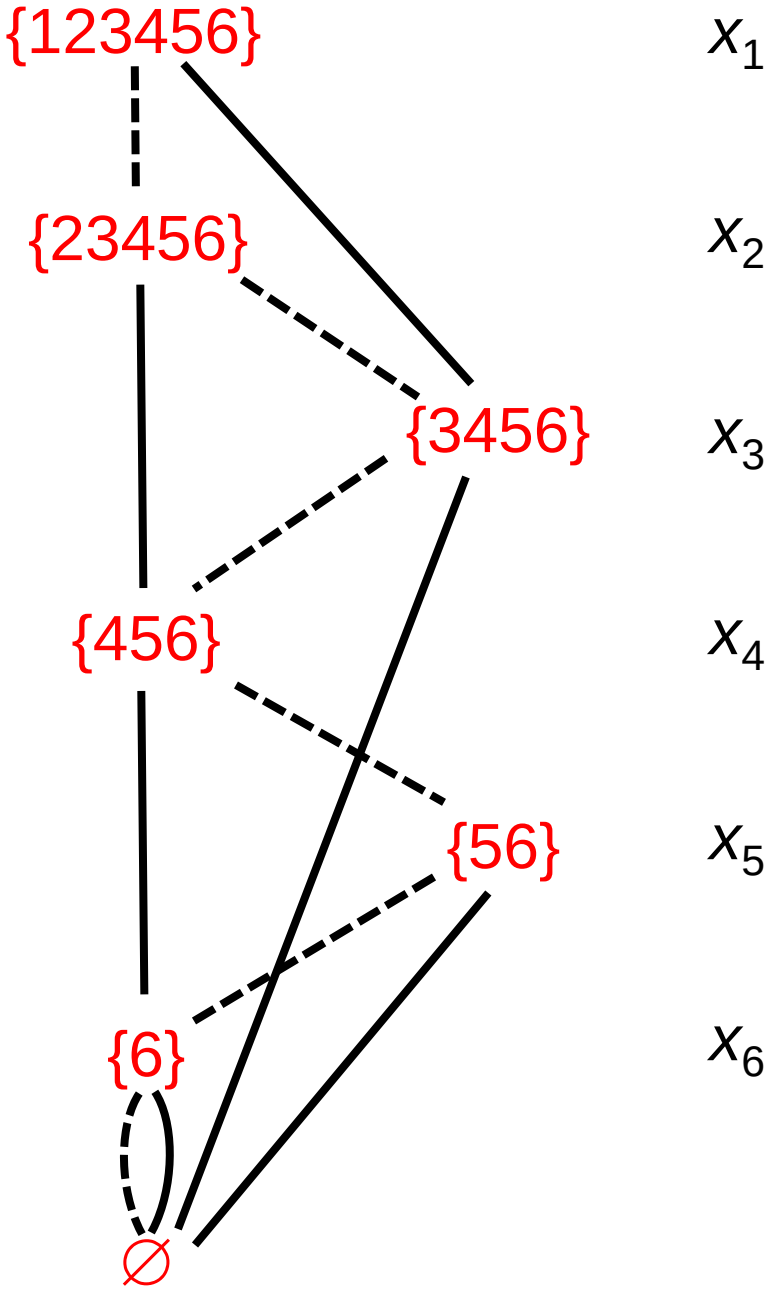
To build **relaxed** BDD, merge some additional nodes as we go along

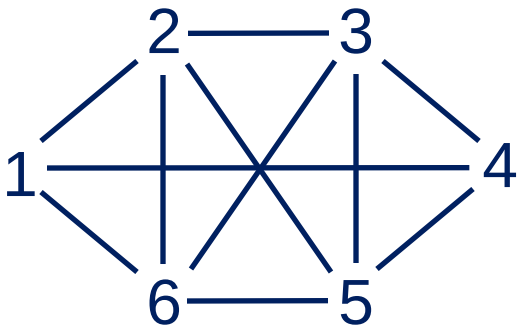




Width = 1

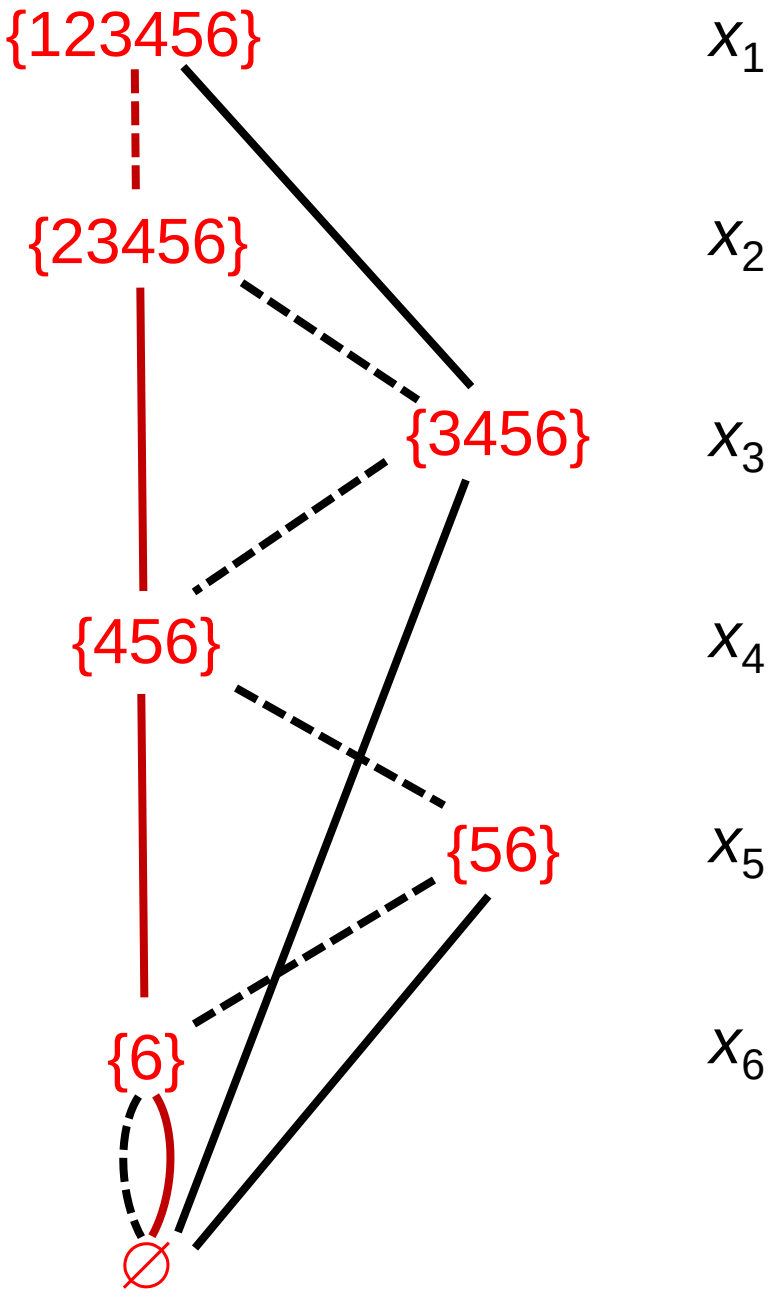
Represents 18 solutions, including 11 feasible solutions





Width = 1

Longest path
gives bound
of 3 on optimal
value of 2



Variable Ordering

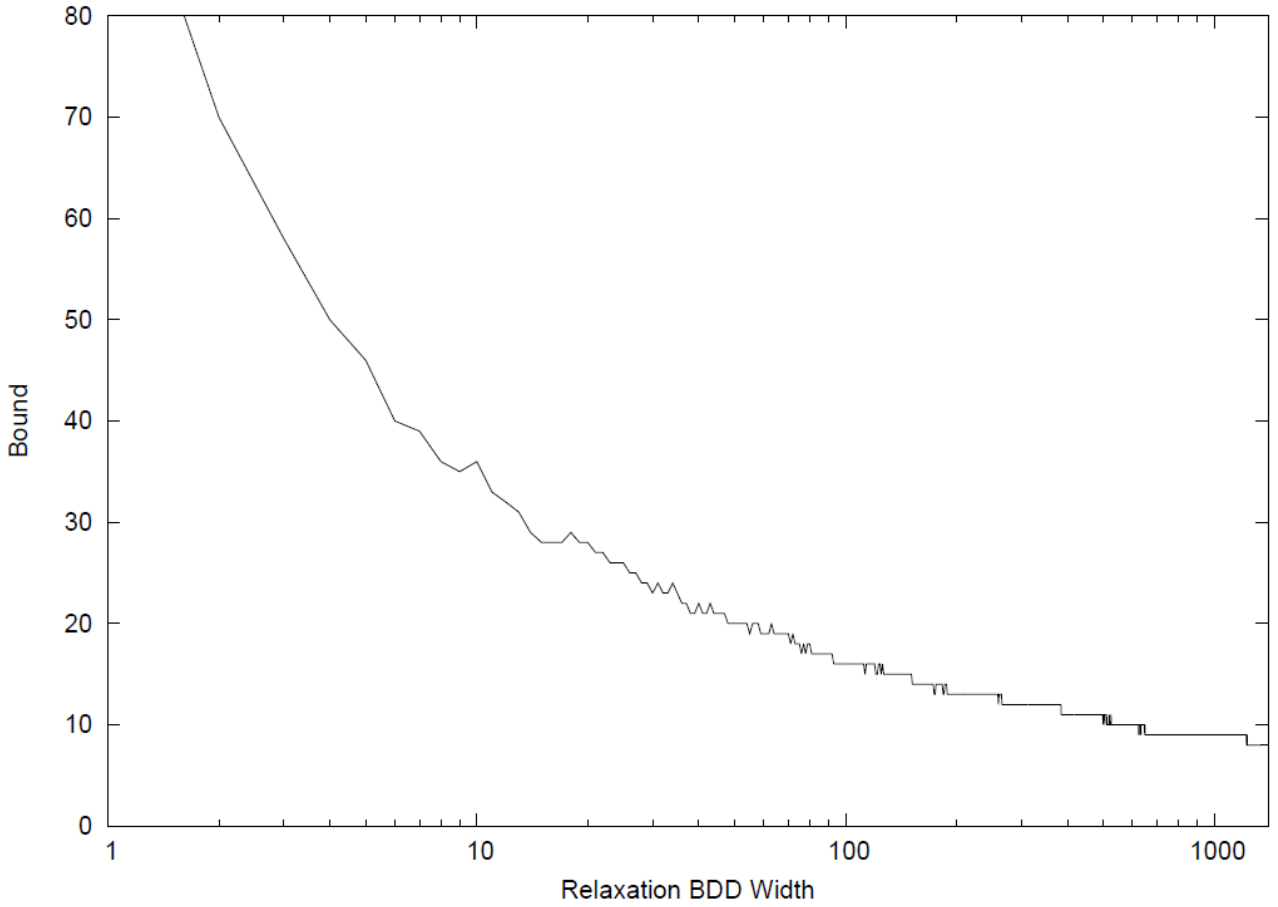
- Variable ordering is key.
  - Just as in branching methods.
  - Some orderings provide much tighter bounds.
- Width of exact BDD is bounded by Fibonacci numbers.
  - For an ordering induced by maximal path decomposition of the graph.
- We used a dynamic ordering heuristic.
  - Next variable is the 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 longest path heuristic seems by far the best.
  - Order nodes on current layer by increasing length of longest path into each node.
  - Merge nodes in this order.
  - We lose information in areas not likely to be part of the solution.
- This is better than merging nodes with more vertices in the corresponding states.

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.

$$\max \sum_i w_i y_i$$

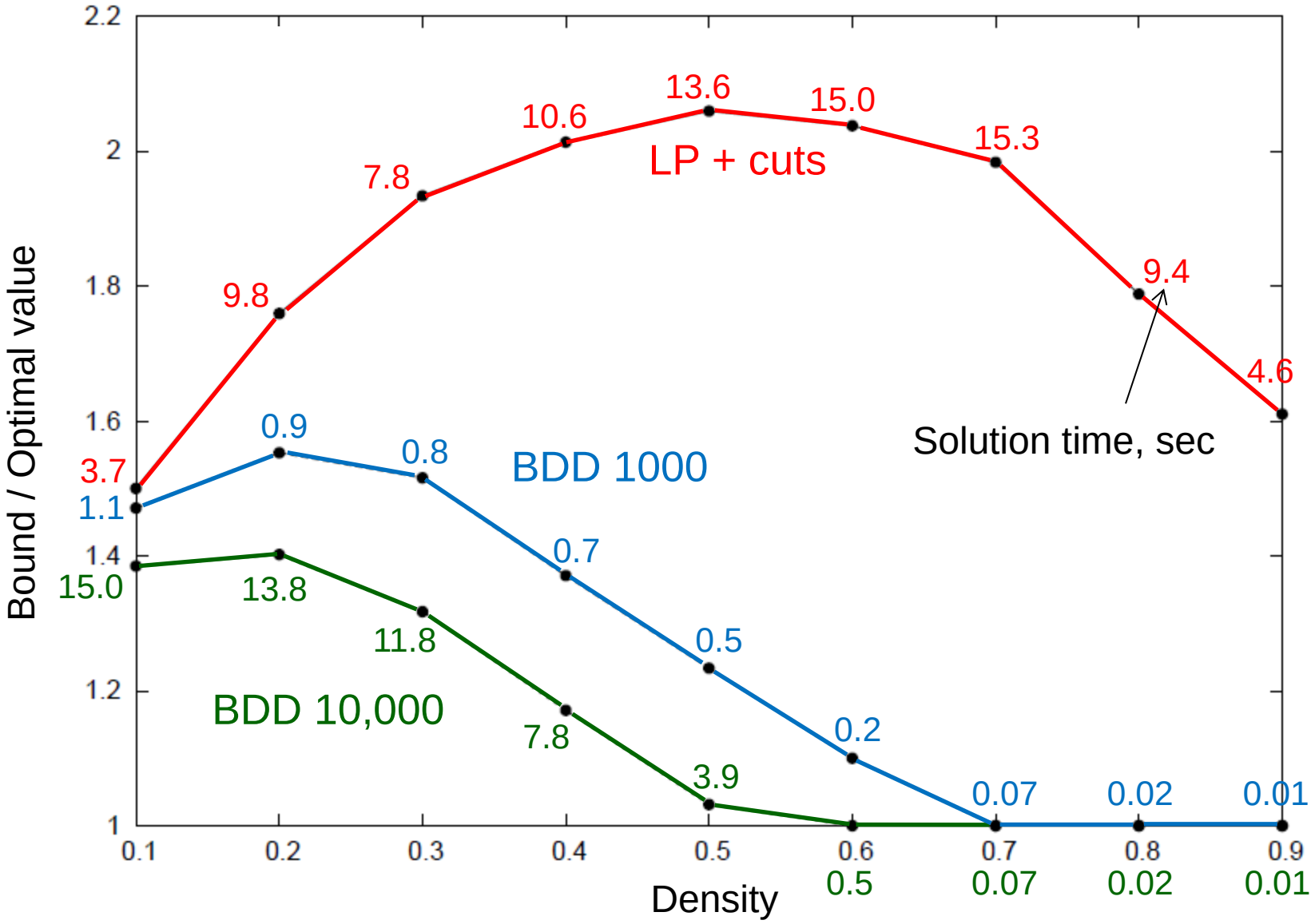
$$\sum_{i \in C_k} y_i \leq 1, \quad \text{all cliques } C_k \text{ in clique cover}$$

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

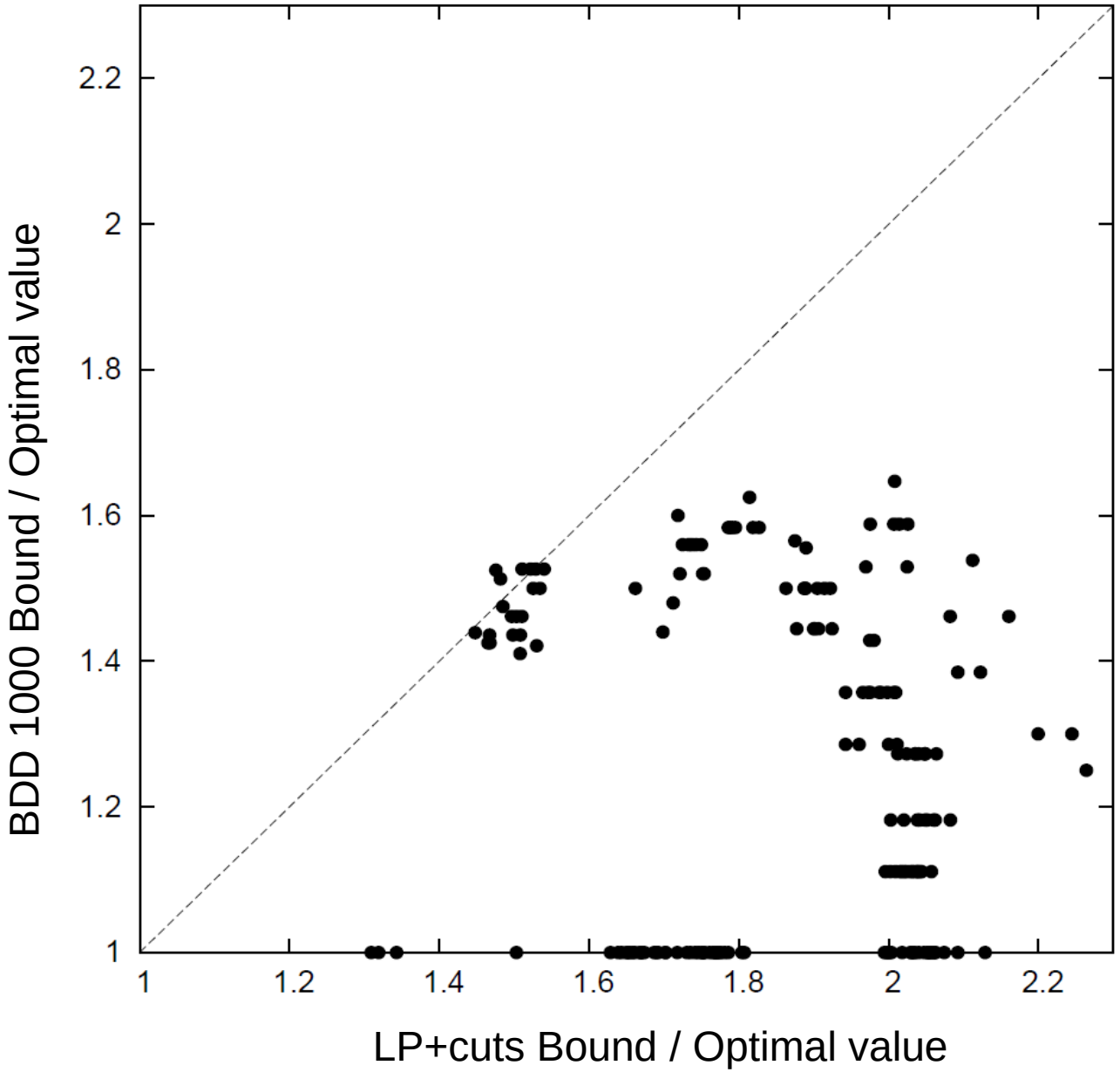
Comparison with LP Bound

- 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
  - Use interior point (barrier) LP solver
    - Faster than simplex on these instances.

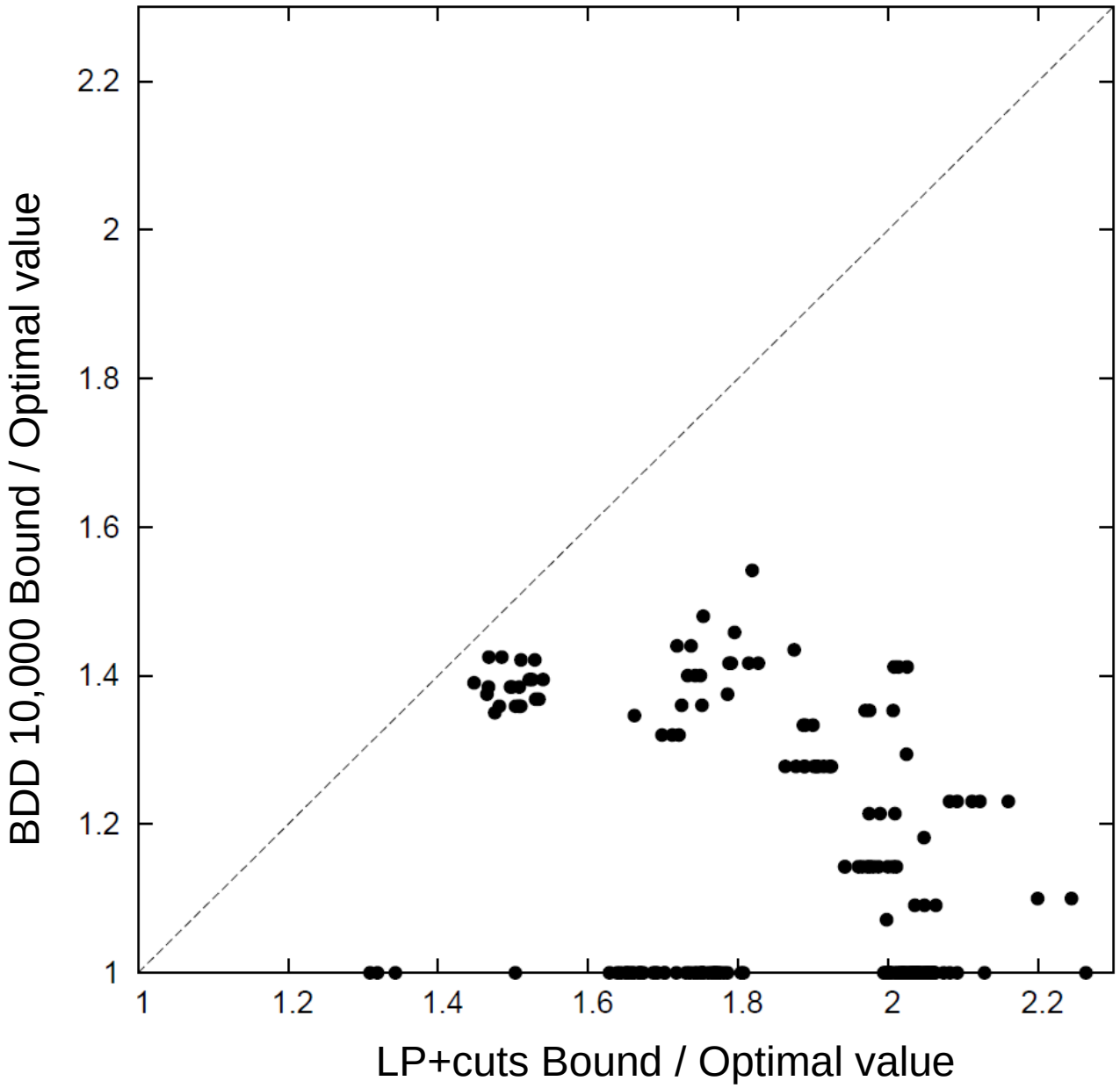
Random instances



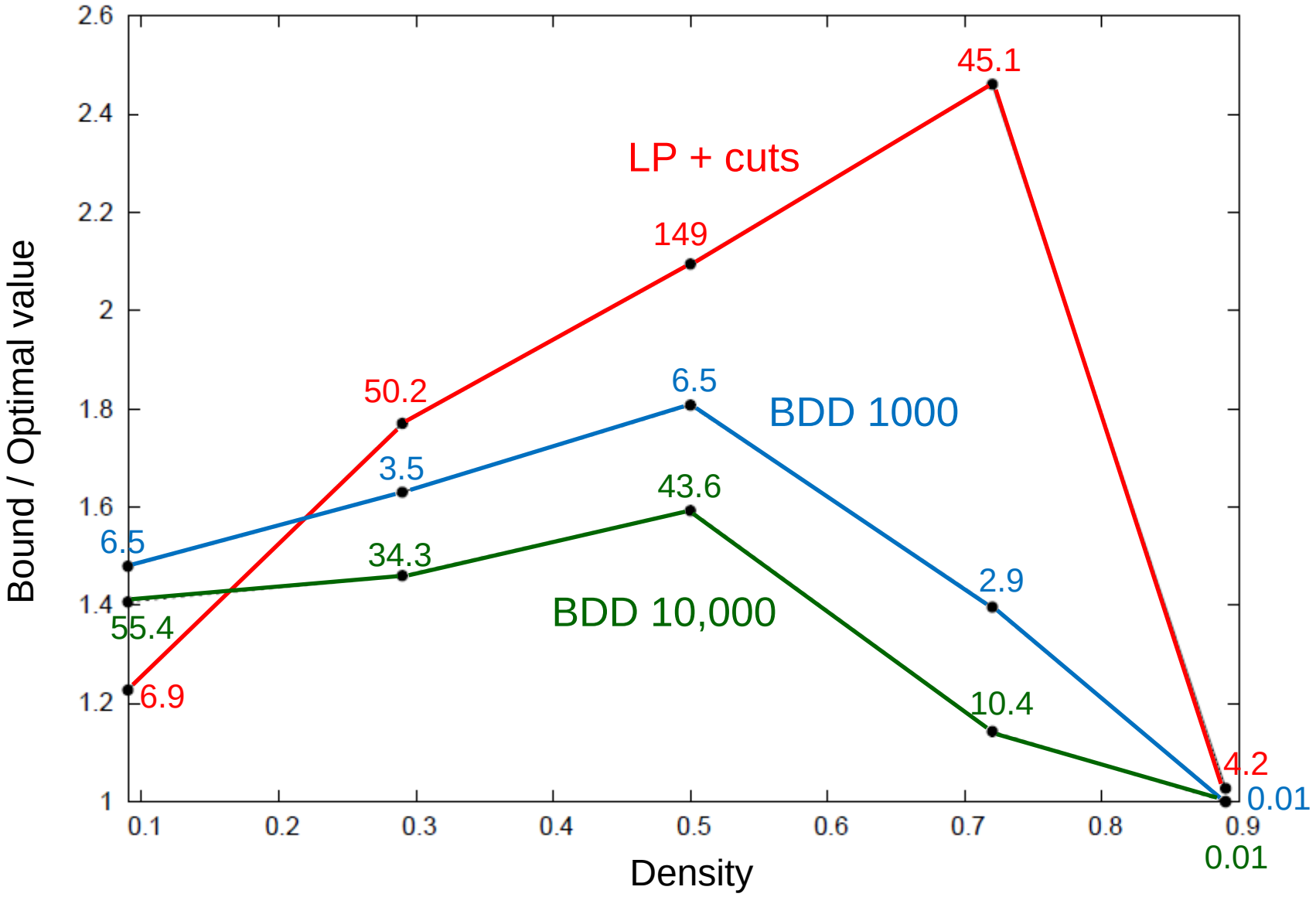
Random instances



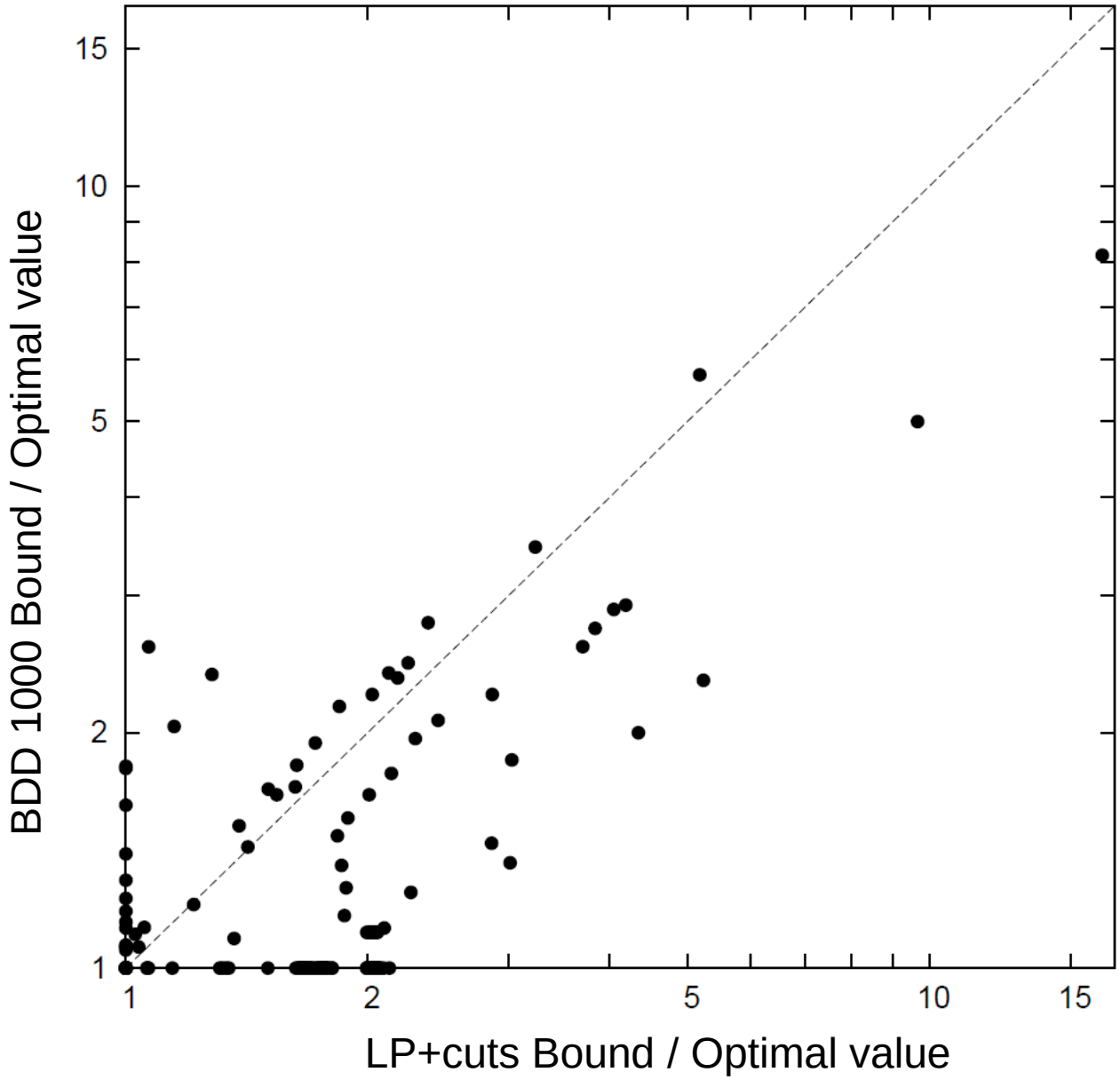
Random instances



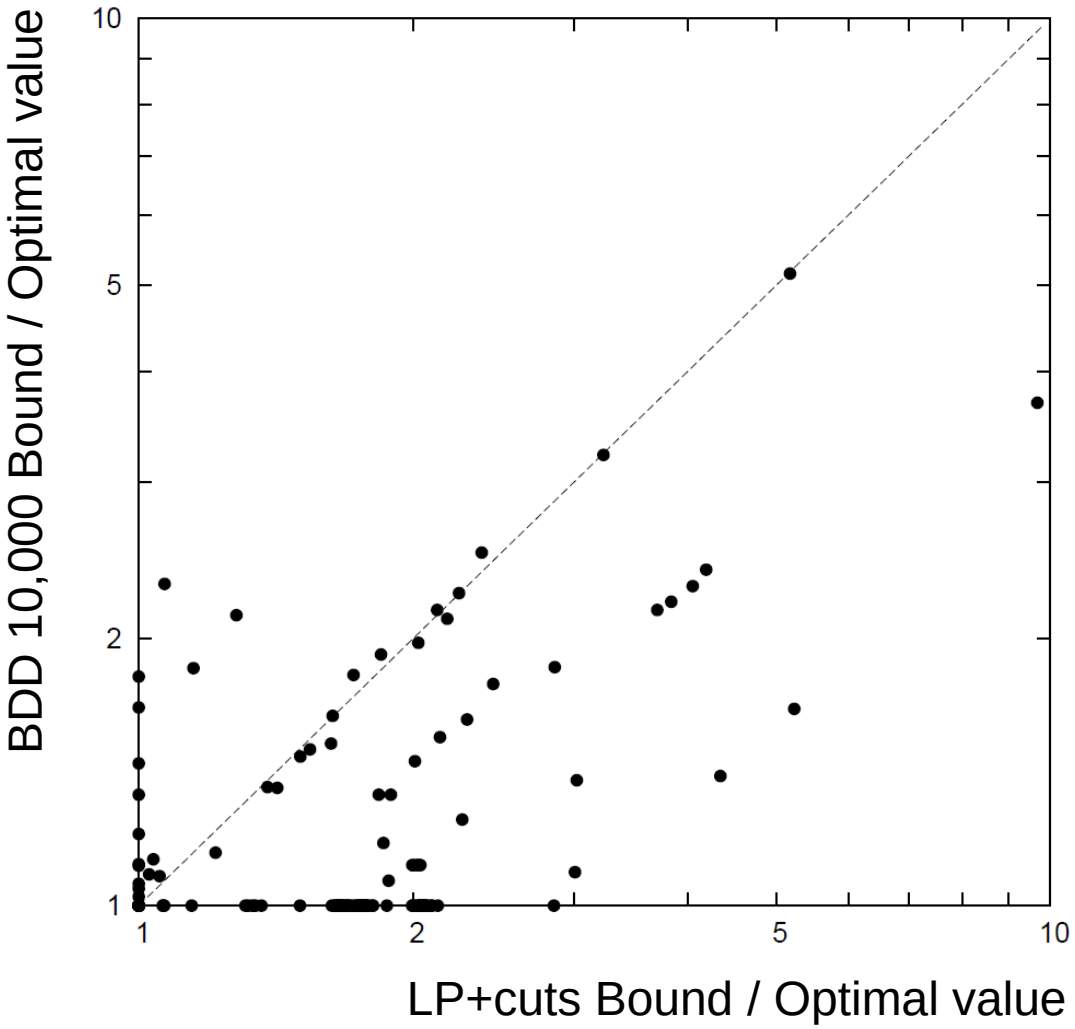
DIMACS instances



DIMACS instances



DIMACS instances



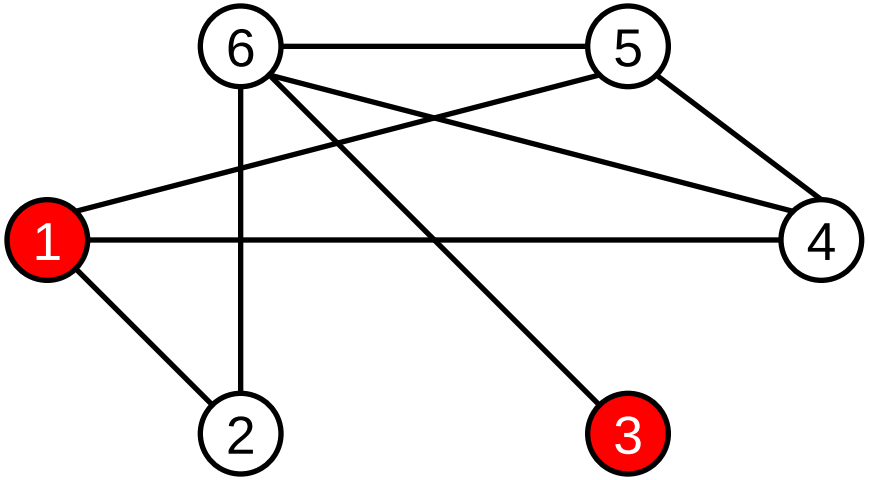
Incrementality

- 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.

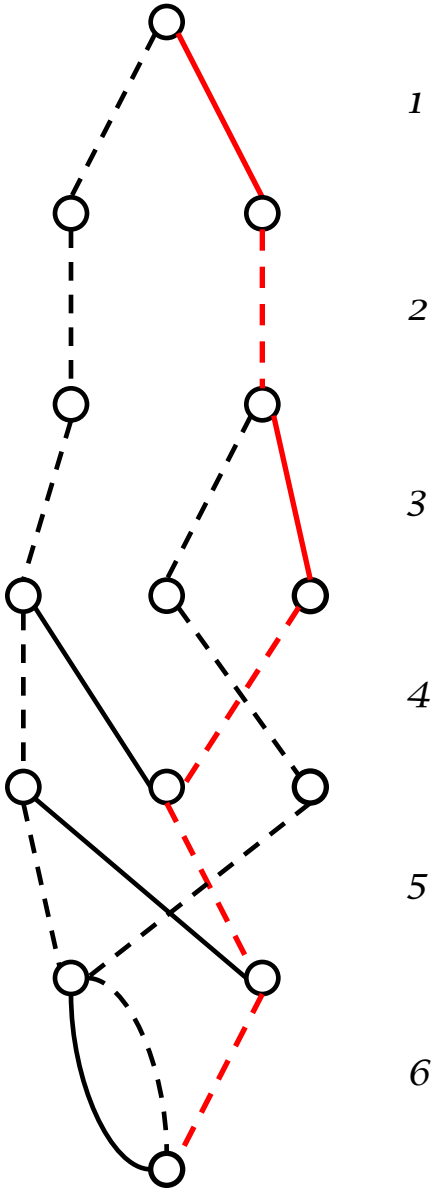
Recent Work

- Restricted BDDs as **primal heuristic**
  - All paths are feasible.
  - Longest path is a good feasible solution
  - Reduce width by intelligently removing nodes.
- Apply to independent set problem
  - DIMACS instances
  - Compare with lower bound obtained by CPLEX primal heuristics at rote node.

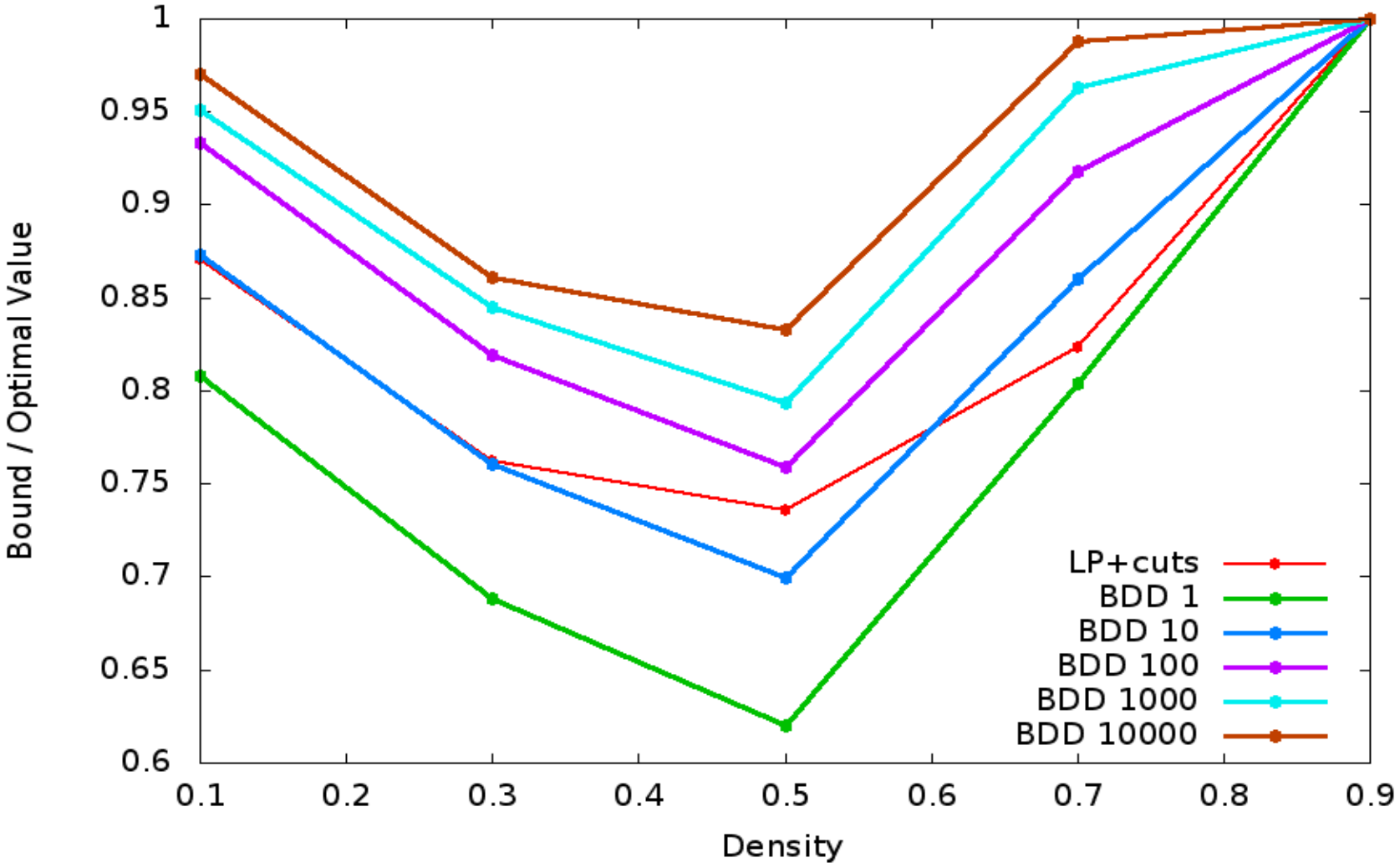
Restricted BDD
Width = 3



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



DIMACS instances

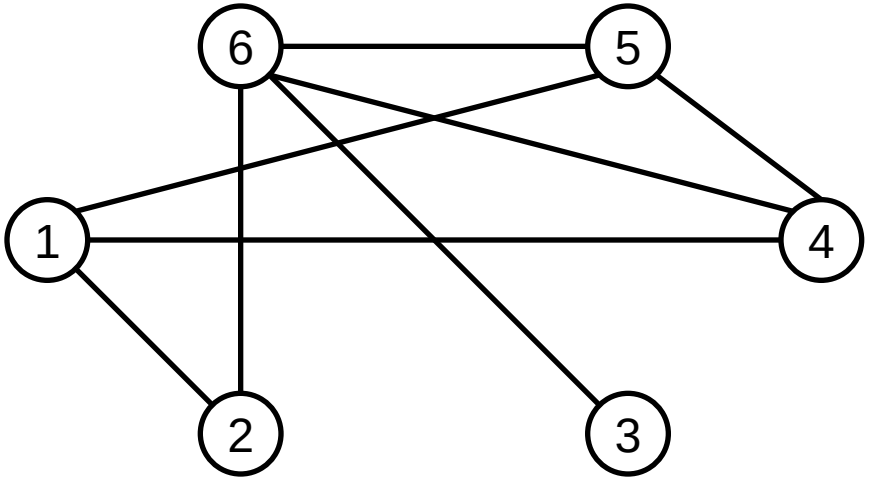


Recent Work

- General **BDD-based solver**
  - Branch in the BDD.
  - Combine with BDD-based propagation
  - BDD-based bounds, primal heuristic
    - No LP relaxation, cutting planes
  - Linearity, convexity irrelevant
    - Possibly nonseparable objective function
  - Dynamic programming style formulation
    - Specify state variable for BDDs
  - Integrate MIP, BDDs, CP, DP.

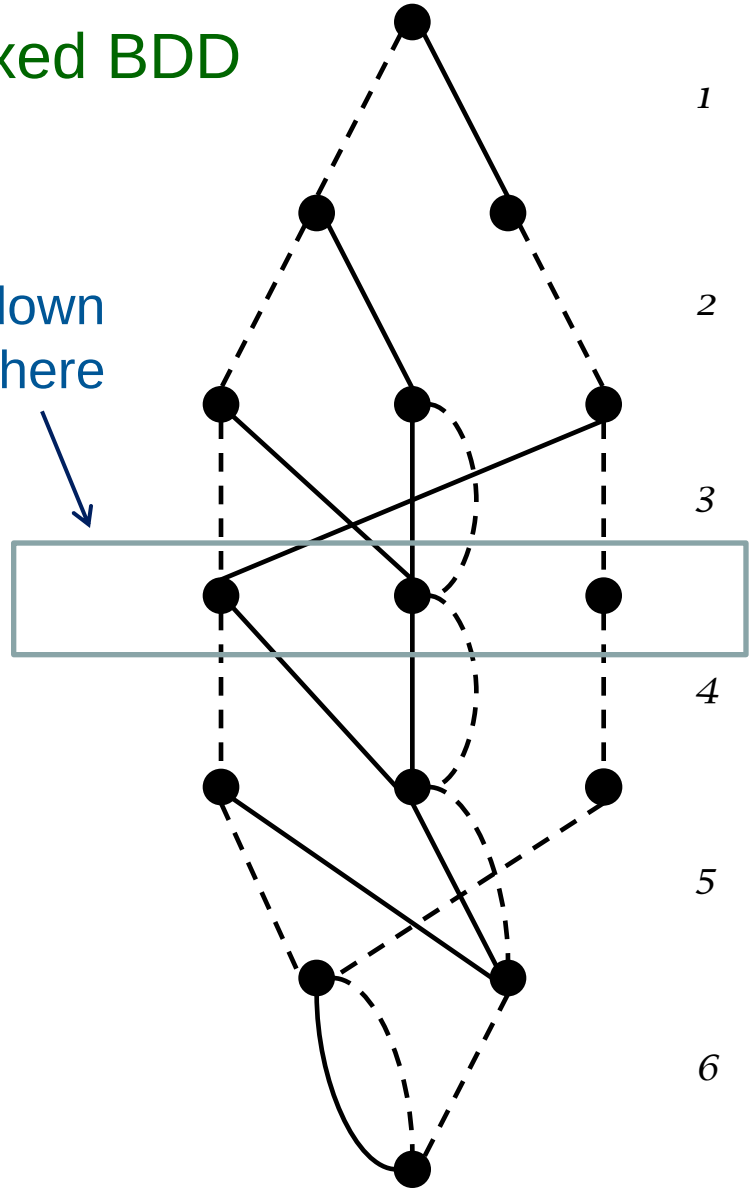
Recent Work

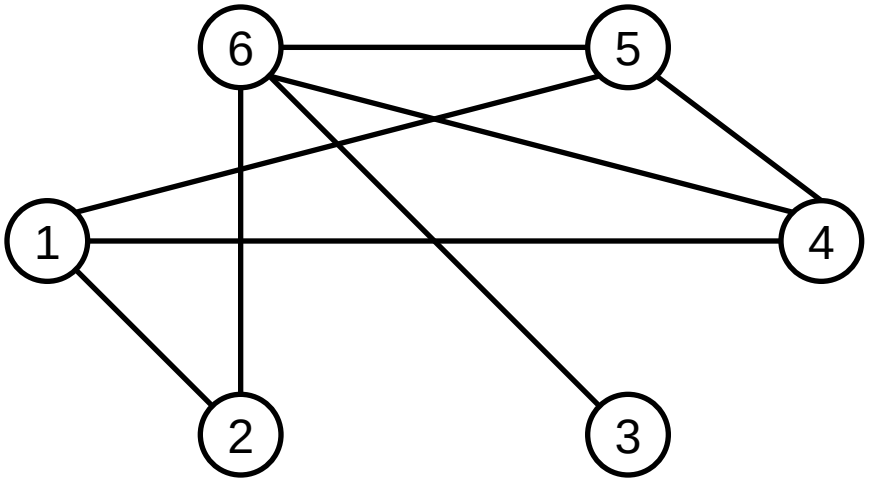
- Novel branching scheme
  - Branch in relaxed BDD.
  - Branch on pools of partial solutions.
  - Reduce symmetry.



Relaxed BDD

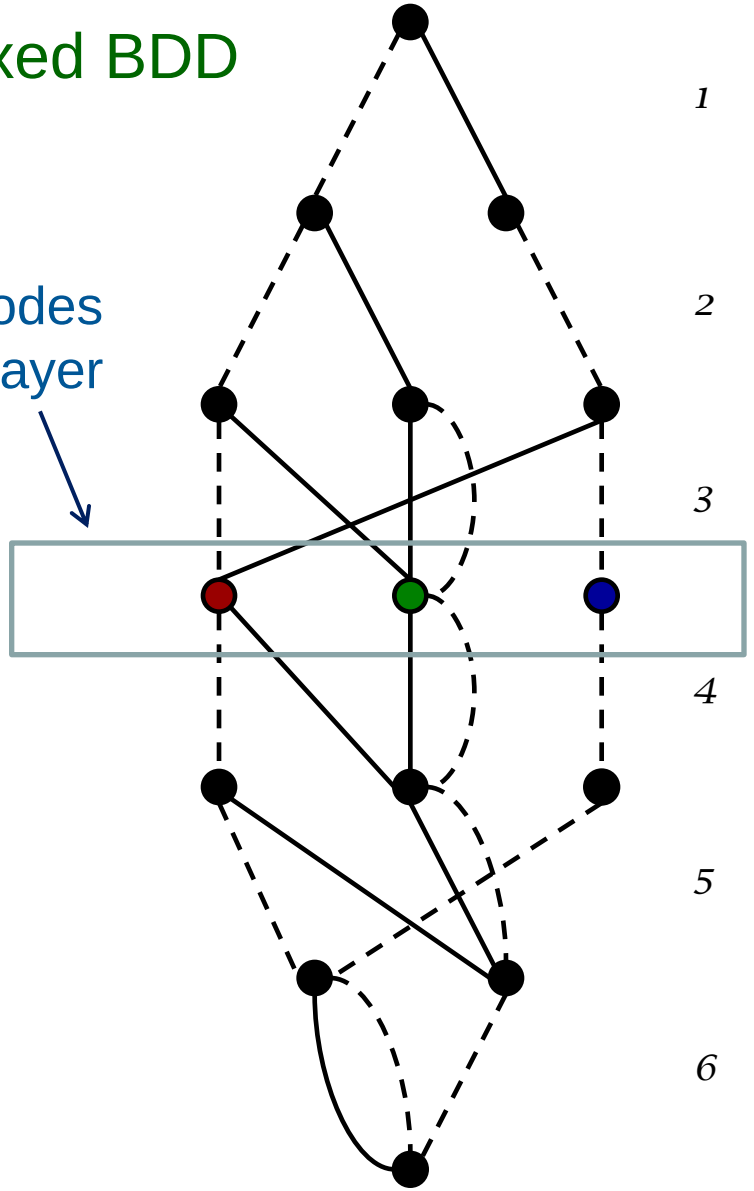
Exact down to here

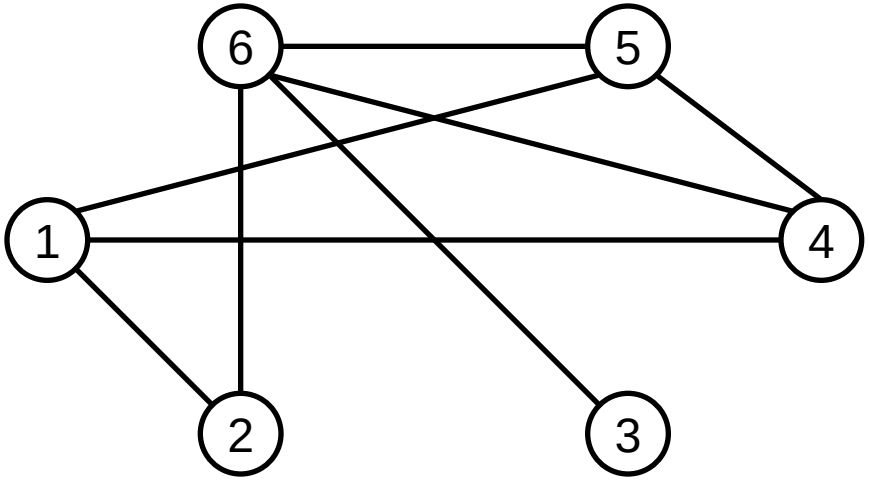




Relaxed BDD

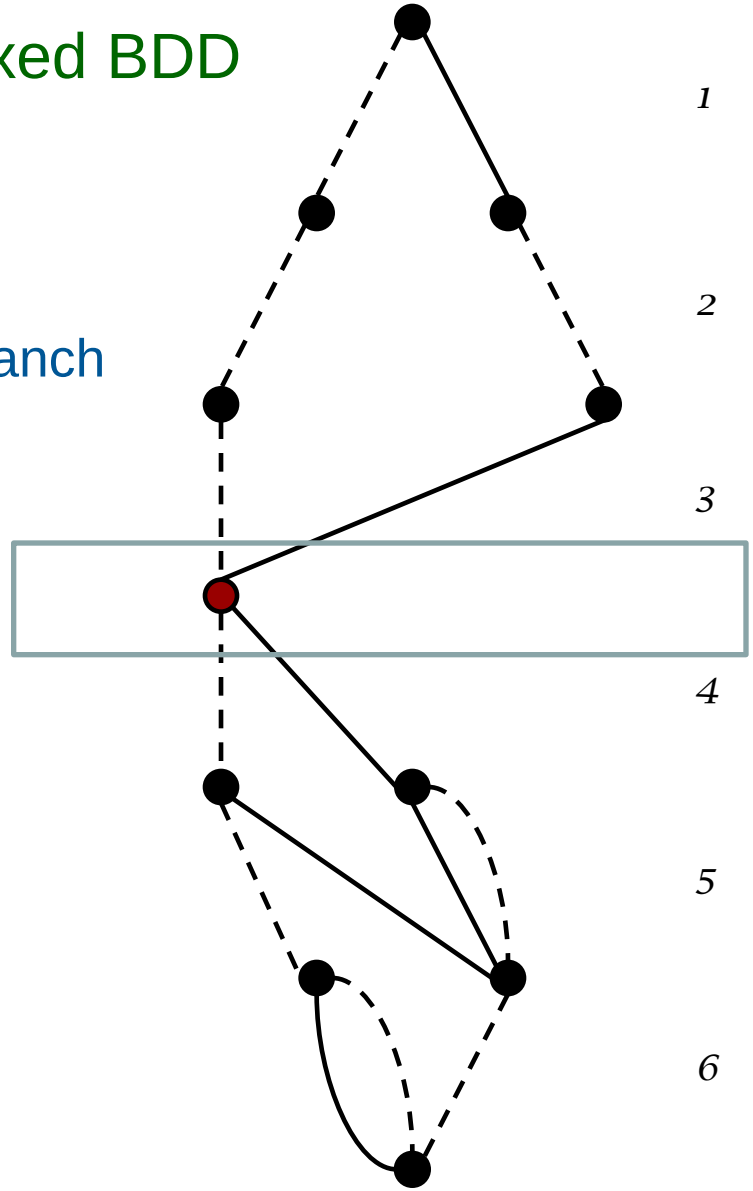
Branch on nodes
in this layer



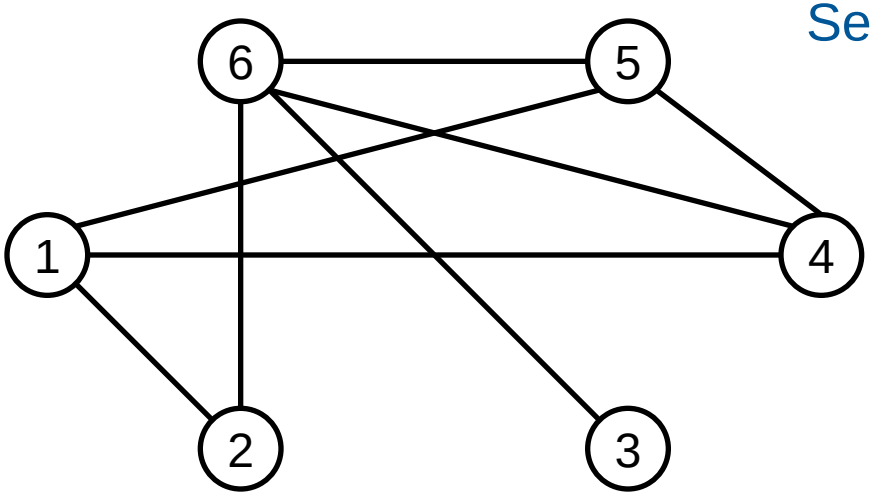


Relaxed BDD

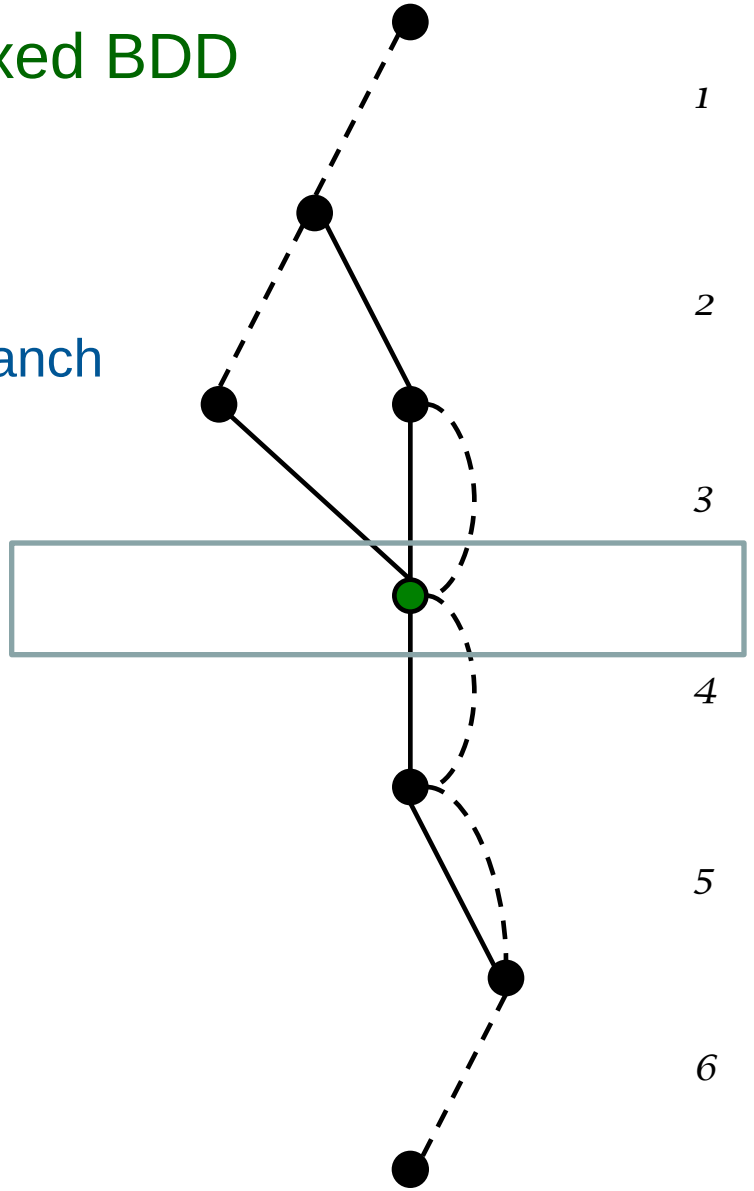
First branch



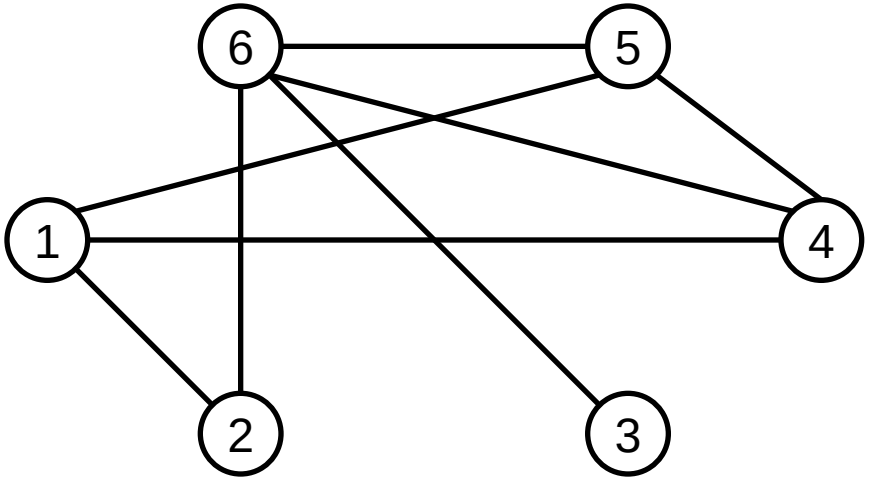
Relaxed BDD



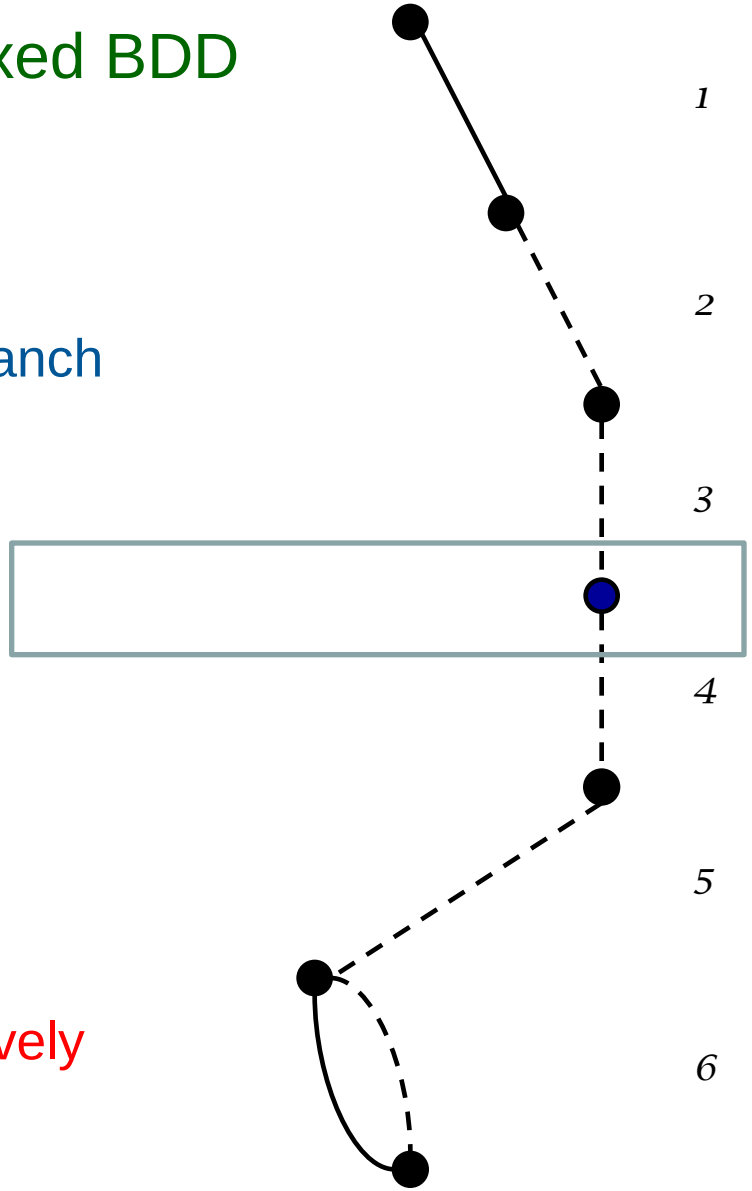
Second branch



Relaxed BDD

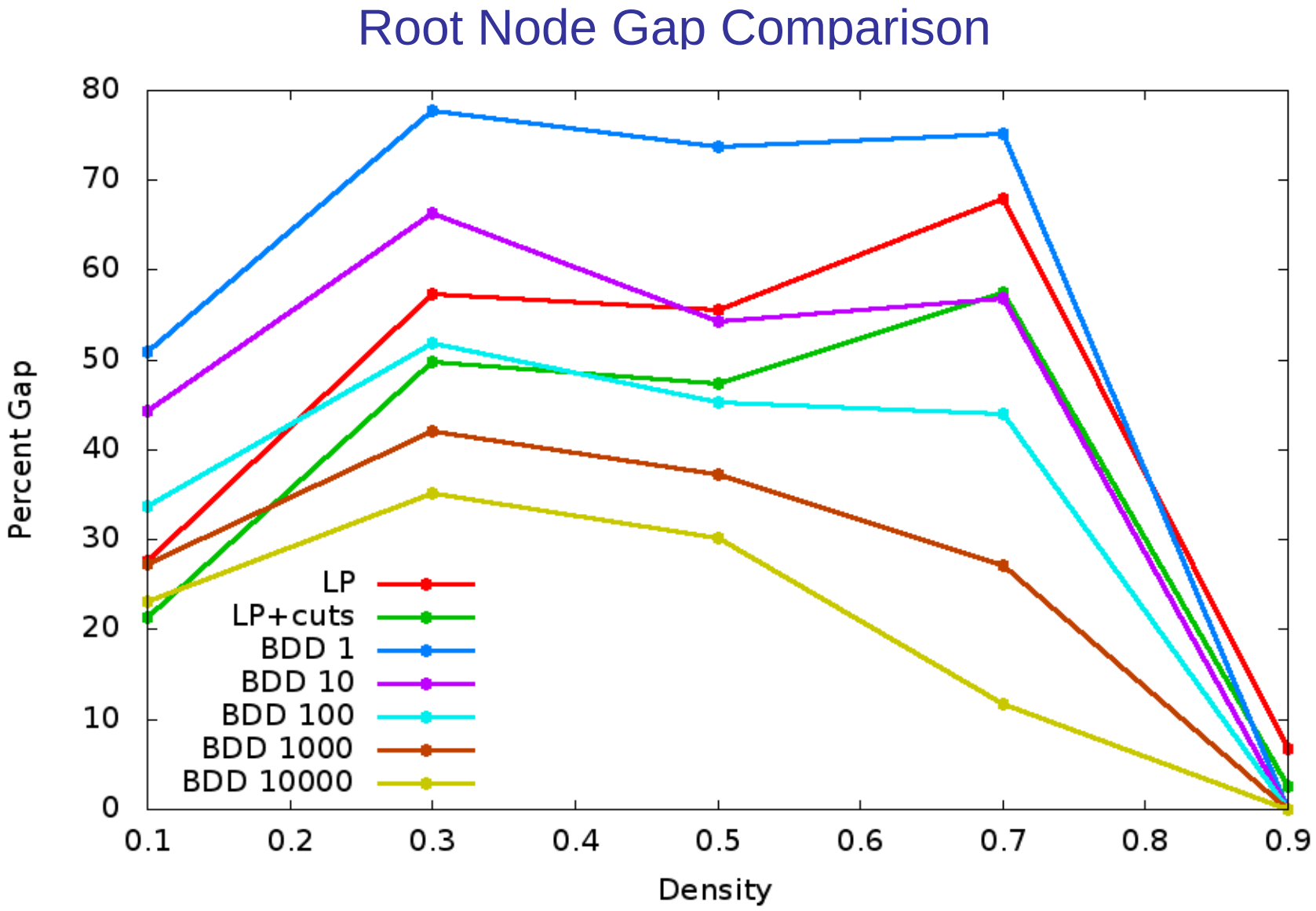


Third branch

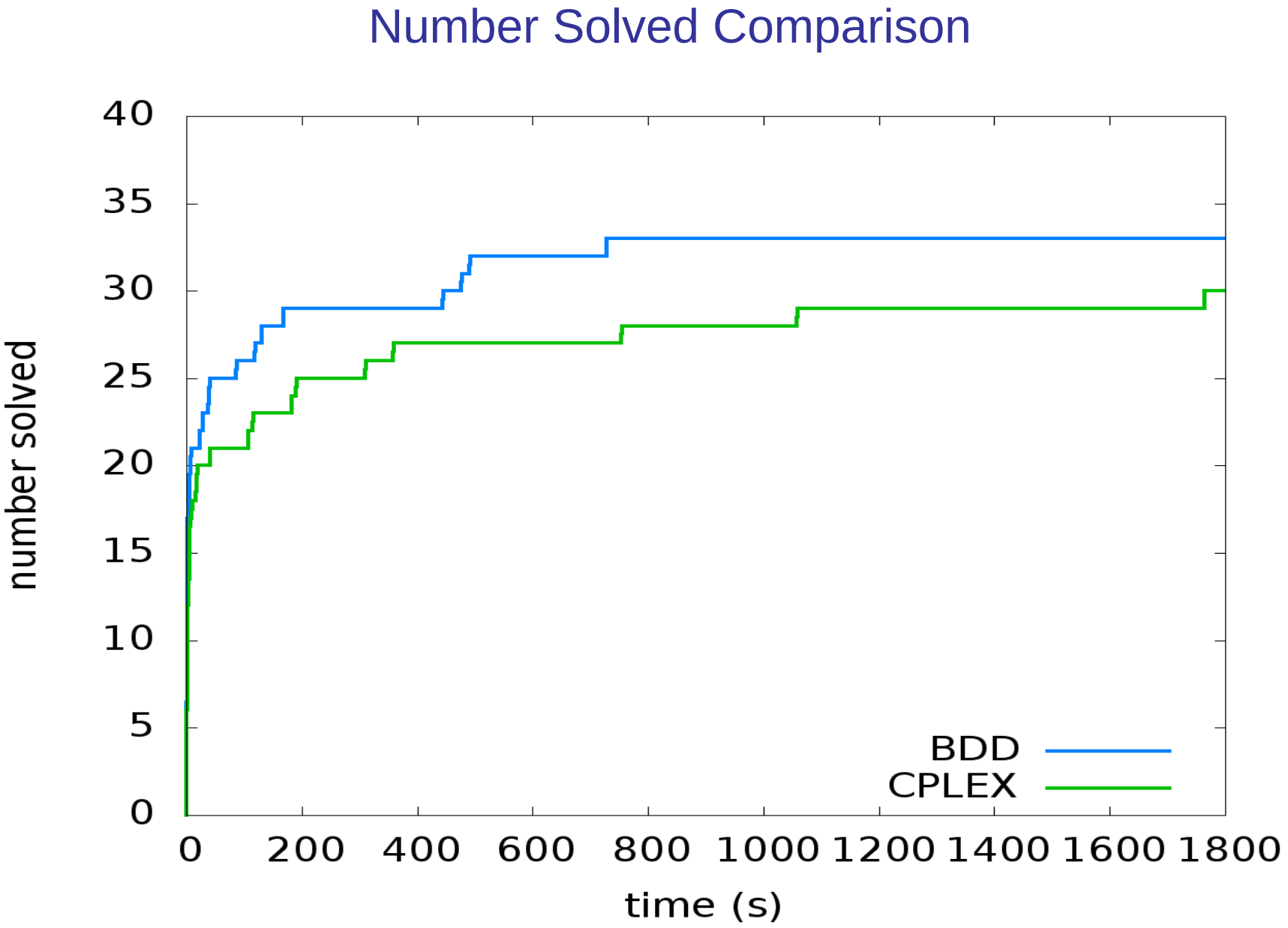


Continue recursively

DIMACS instances



DIMACS instances



End Gap Comparison