Overview of Decision Diagrams for Optimization

J. N. Hooker
Carnegie Mellon University

IOS Conference
2016

Today's Session

- **Overview** of decision diagrams for optimization
  - JH
- Decisions diagrams for **sequencing and scheduling**
  - Andre Cire
- Decision diagram **decompositions**
  - David Bergman

Decision Diagrams

- Used in **computer science** and **AI** for decades
  – Logic circuit design
  – Product configuration
- A **new perspective** on optimization
  – Constraint programming
  – **Discrete optimization**

Decision Diagrams

- Advantages:
  - No need for **inequality** formulations.
  - No need for **linear** or **convex** relaxations.
  - New approach to solving **dynamic programming** models.
  - Very effective **parallel** computation.
  - Ideal for **postoptimality** anaylsis
- Disadvantage:
  - Developed only for **discrete**, **deterministic** optimization.
  - …so far.

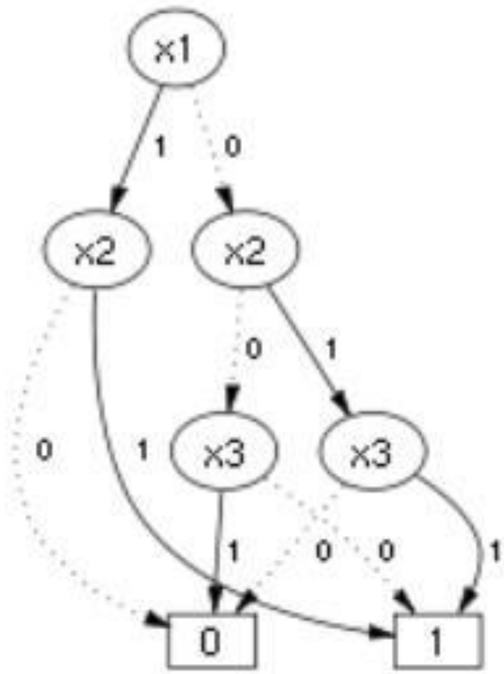
Outline

- Decision diagram **basics**
- Optimization with **exact** decision diagrams
- A **general-purpose** solver that scales up
  - **Relaxed** decision diagrams
  - **Restricted** decision diagrams
  - **Dynamic programming** model
  - A new **branching** algorithm
  - Computational **performance**
- Modeling the objective function
  - Inventory management example
- References

Decision Diagram Basics

- Binary decision diagrams encode Boolean functions

x_1	x_2	x_3	f
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Lee (1959), Akers (1978)

Decision Diagram Basics

- Binary decision diagrams encode Boolean functions
  - Historically used for circuit design & verification

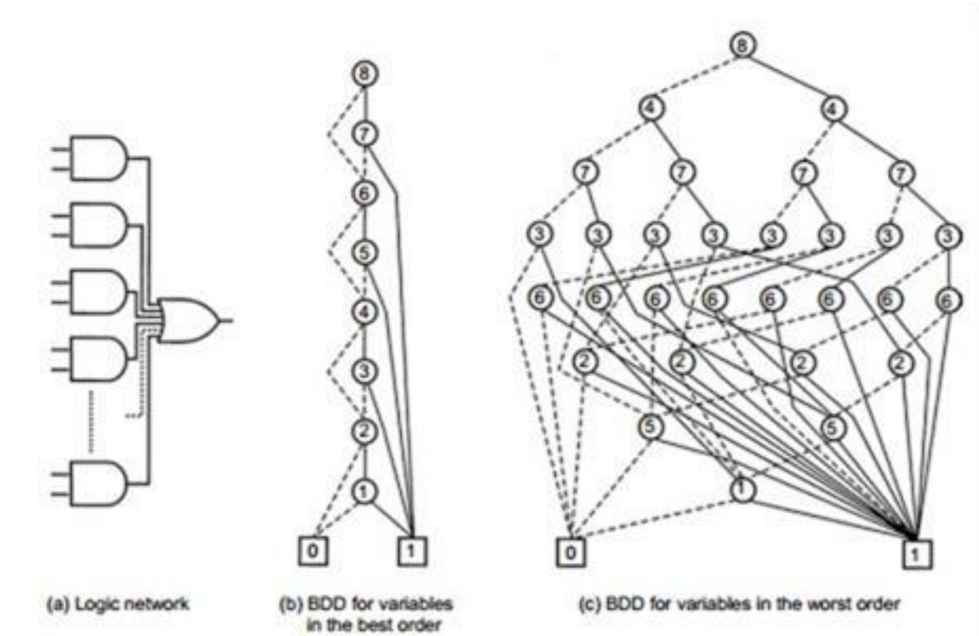
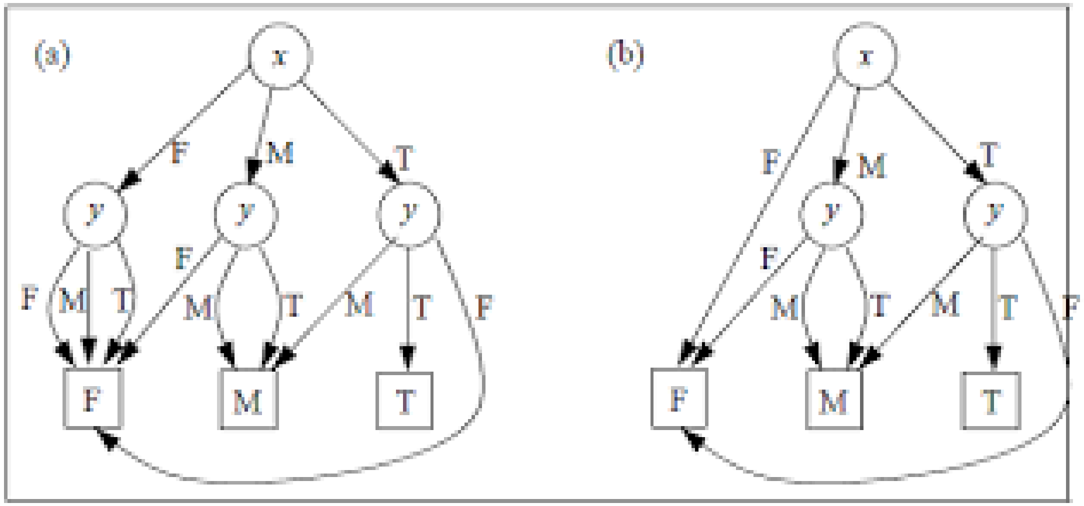


FIGURE 29.11 BDDs for 2-level logic network with AND/OR gates.

Bryant (1986), etc.

Decision Diagram Basics

- Binary decision diagrams encode Boolean functions
  - Historically used for circuit design & verification
  - Easily generalized to multivalued decision diagrams



Reduced Decision Diagrams

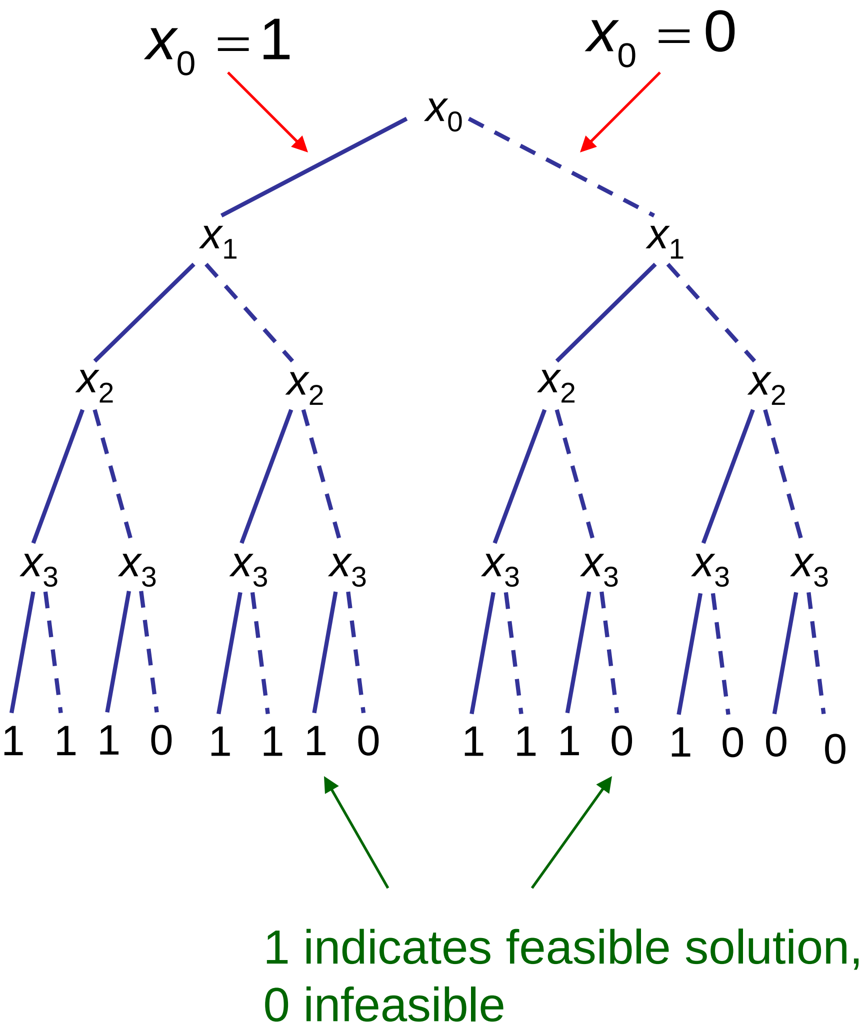
- There is a **unique reduced** DD for any given constraint.
  - Once the variable ordering is specified.

- The reduced DD can be viewed as a branching tree with **redundancy** removed.
  - Superimpose isomorphic subtrees.
  - Remove redundant nodes.

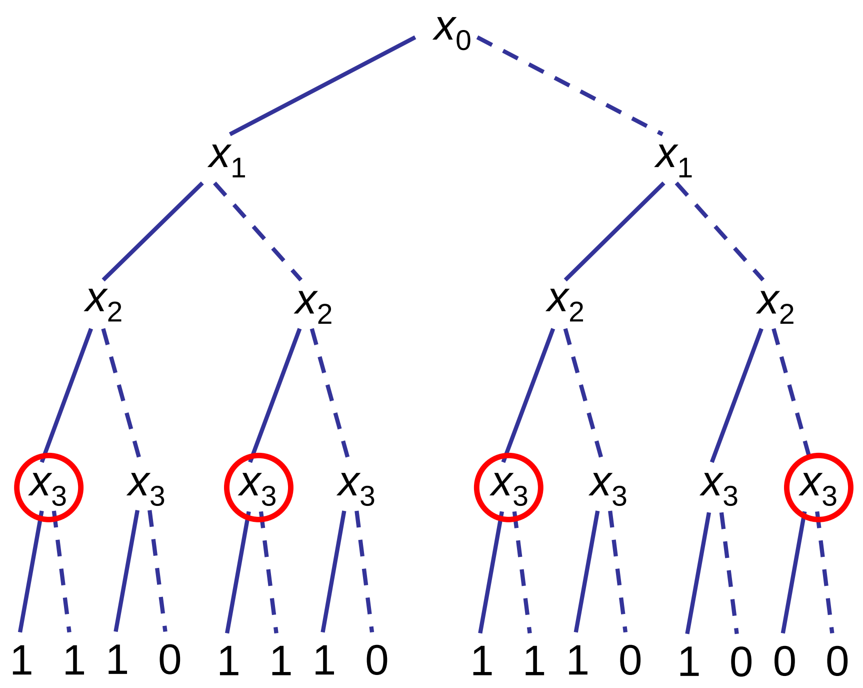
Bryant (1986)

Branching tree for 0-1 inequality

$$2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$$



1

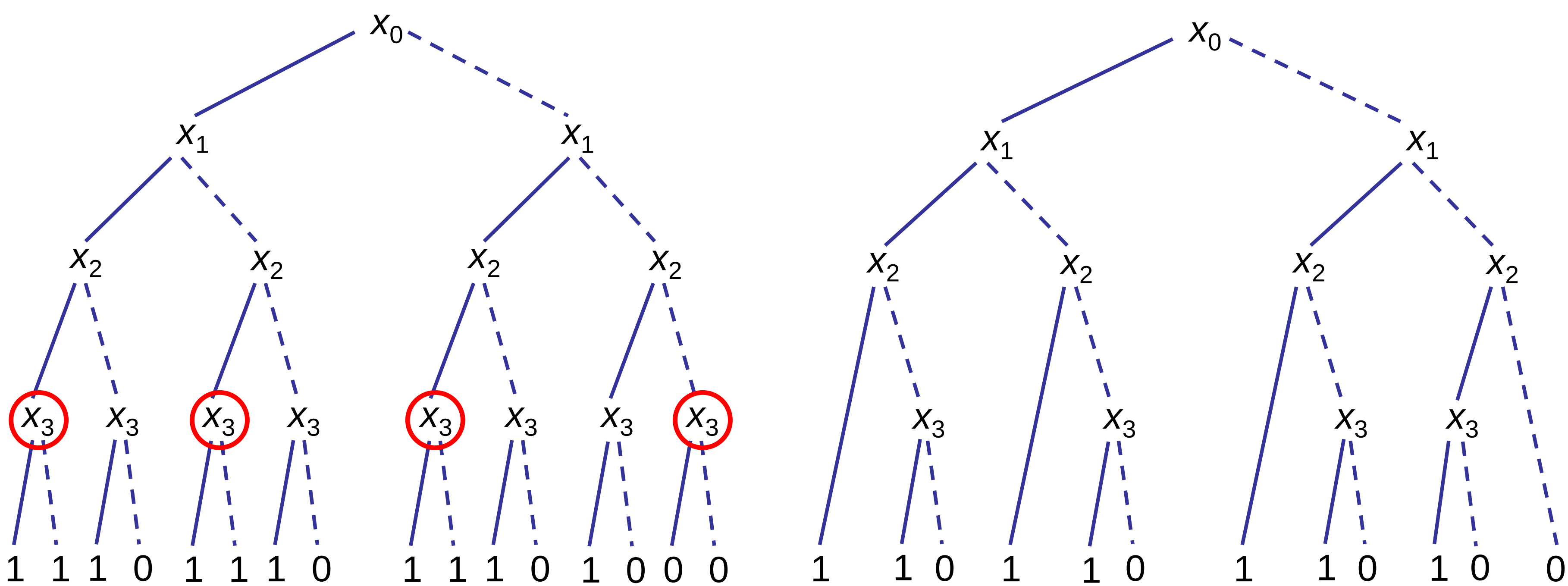


Branching tree for 0-1 inequality

$$2x_0 + 3x_1 + 5x_2 + 5x_3 \geq 7$$

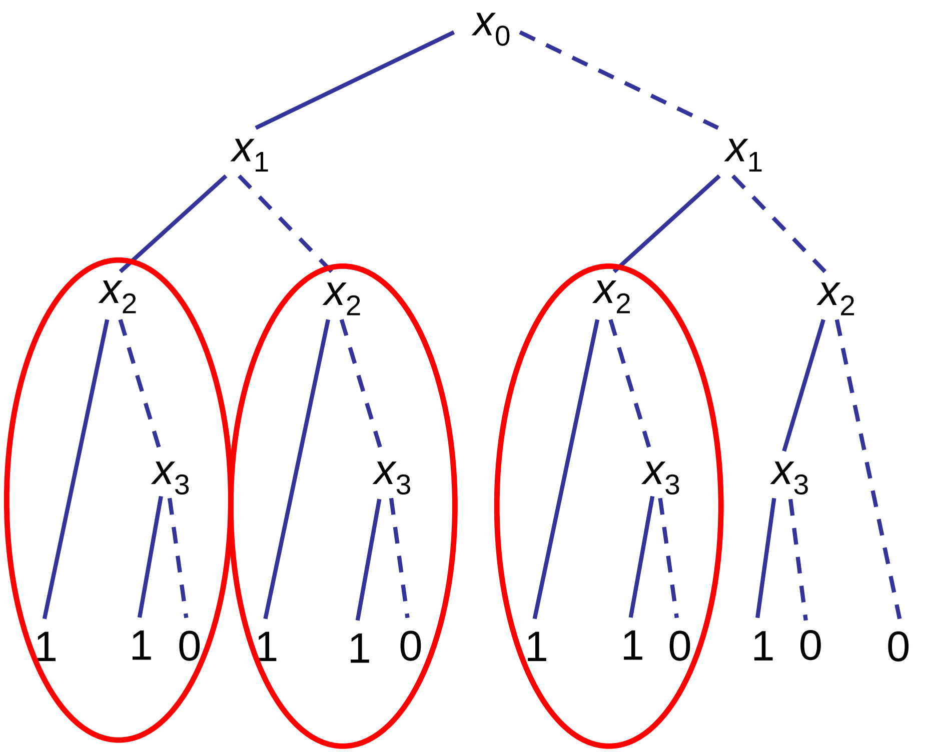
Remove redundant nodes…

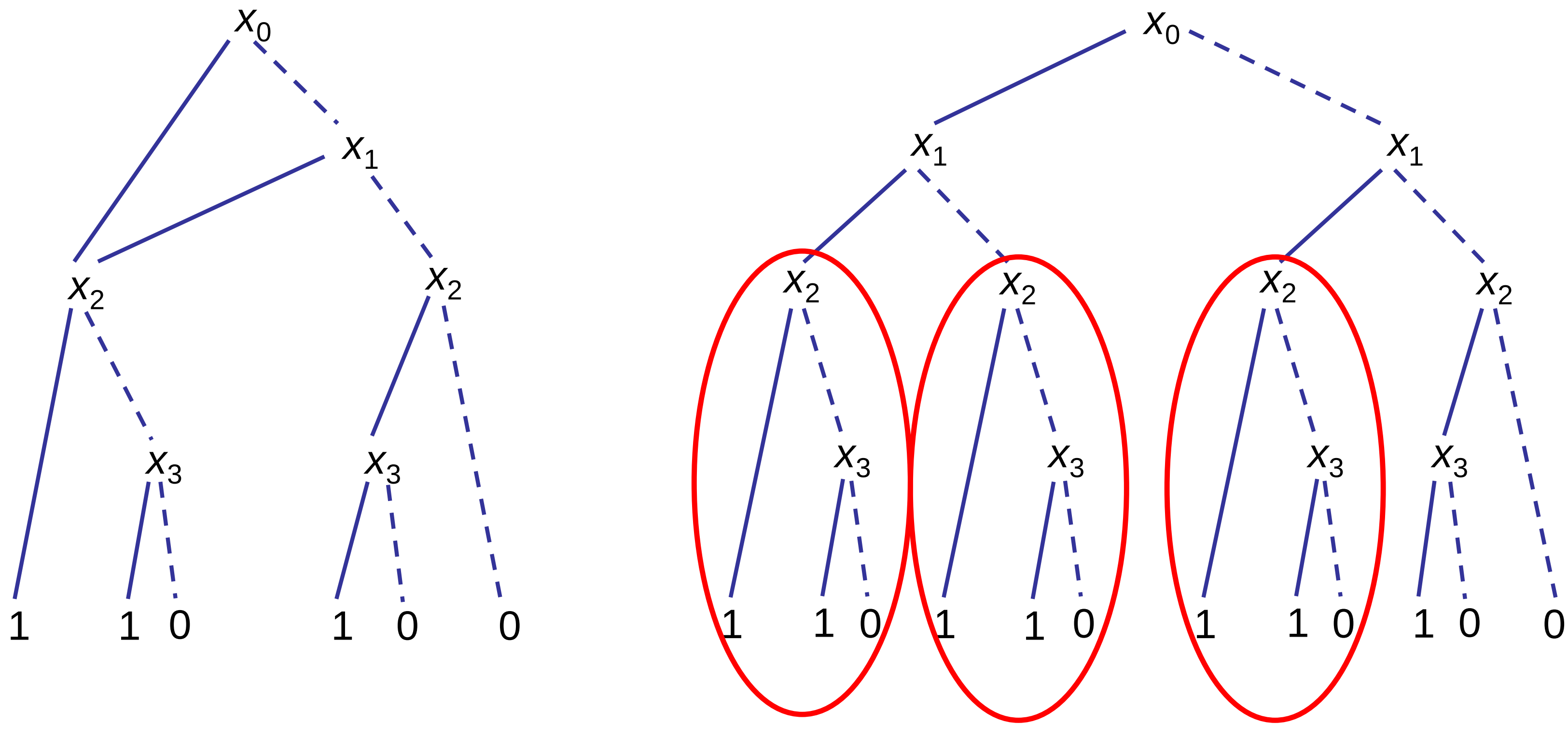
1



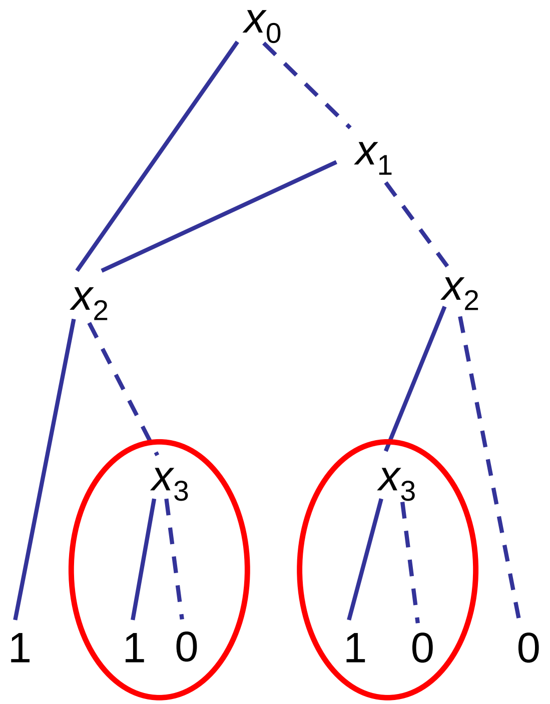
x_0
x_1 x_1
x_2 x_2 x_2 x_2
x_3 x_3 x_3 x_3 x_3 x_3 x_3 x_3
1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0
x_0
x_1 x_1
x_2 x_2 x_2 x_2
x_3 x_3 x_3 x_3
1 1 0 1 1 0 1 1 0 1 0 0

Superimpose identical subtrees…

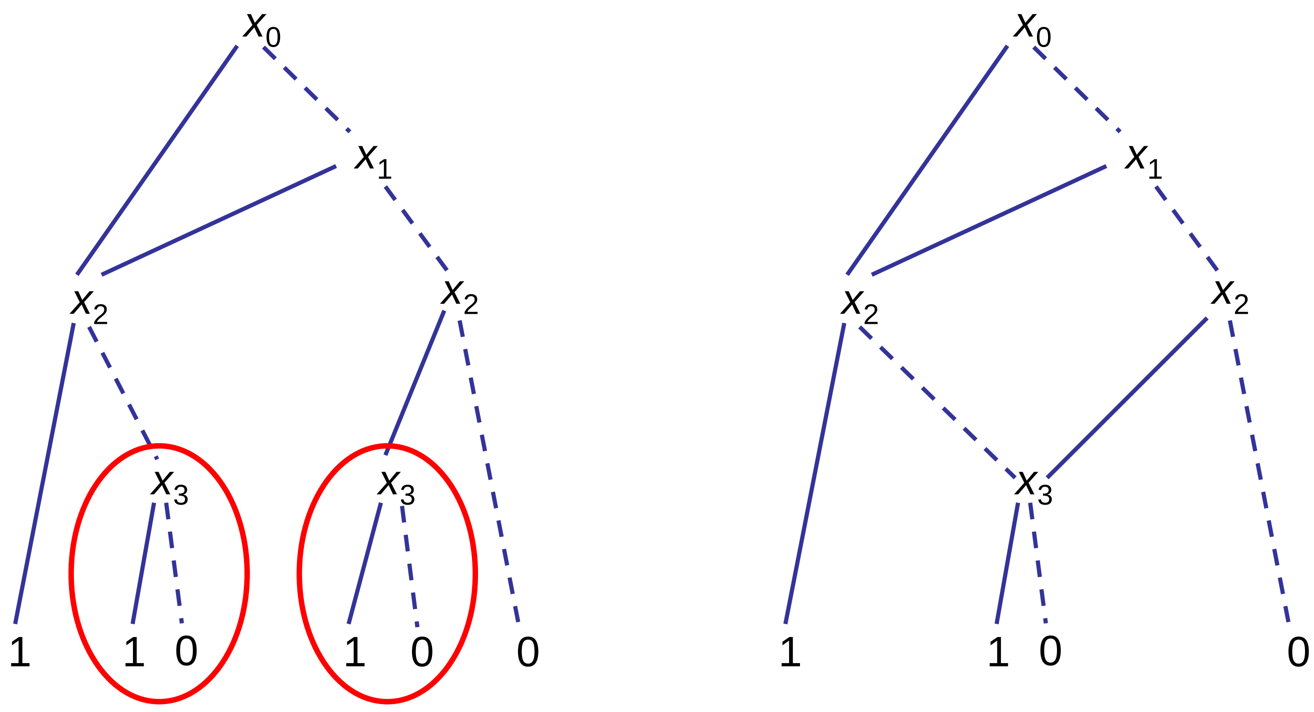




x_0
x_1
x_2
x_2
x_3
x_3
1 1 0 1 0 0
x_0
x_1
x_1
x_2
x_2
x_2
x_2
x_3
x_3
x_3
x_3
1 1 0 1 1 0 1 1 0 1 0 0

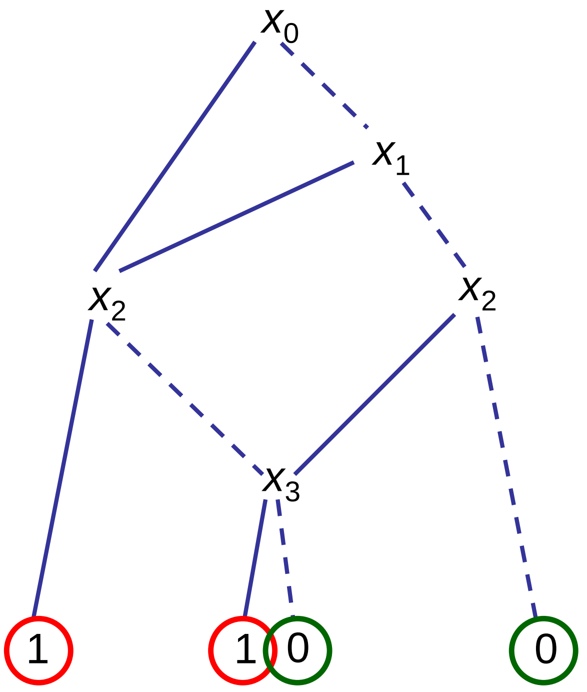


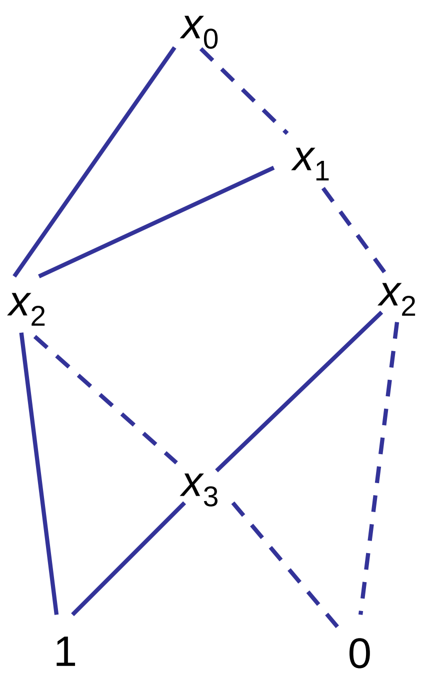
Superimpose identical subtrees…

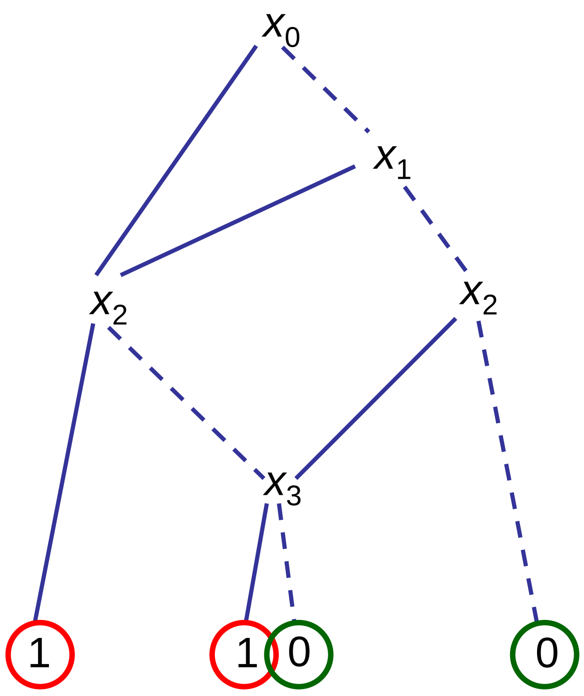


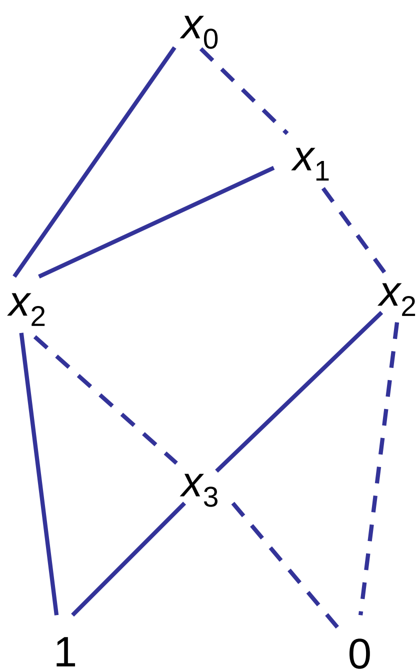
x0
x1
x2
x2
x3
x3
1
1
0
1
0
0
x0
x1
x2
x2
x3
1
1
0
0

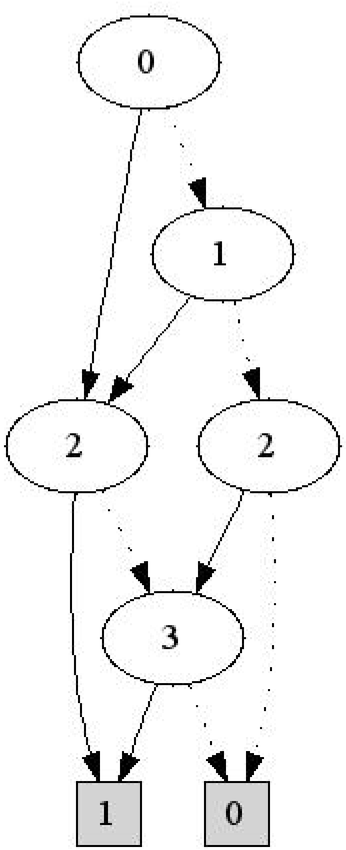
Superimpose identical leaf nodes…




x_0
x_1
x_2
x_2
x_3
1
0


x_0
x_1
x_2
x_2
x_3
1
1
0
0





as generated by software

Reduced Decision Diagrams

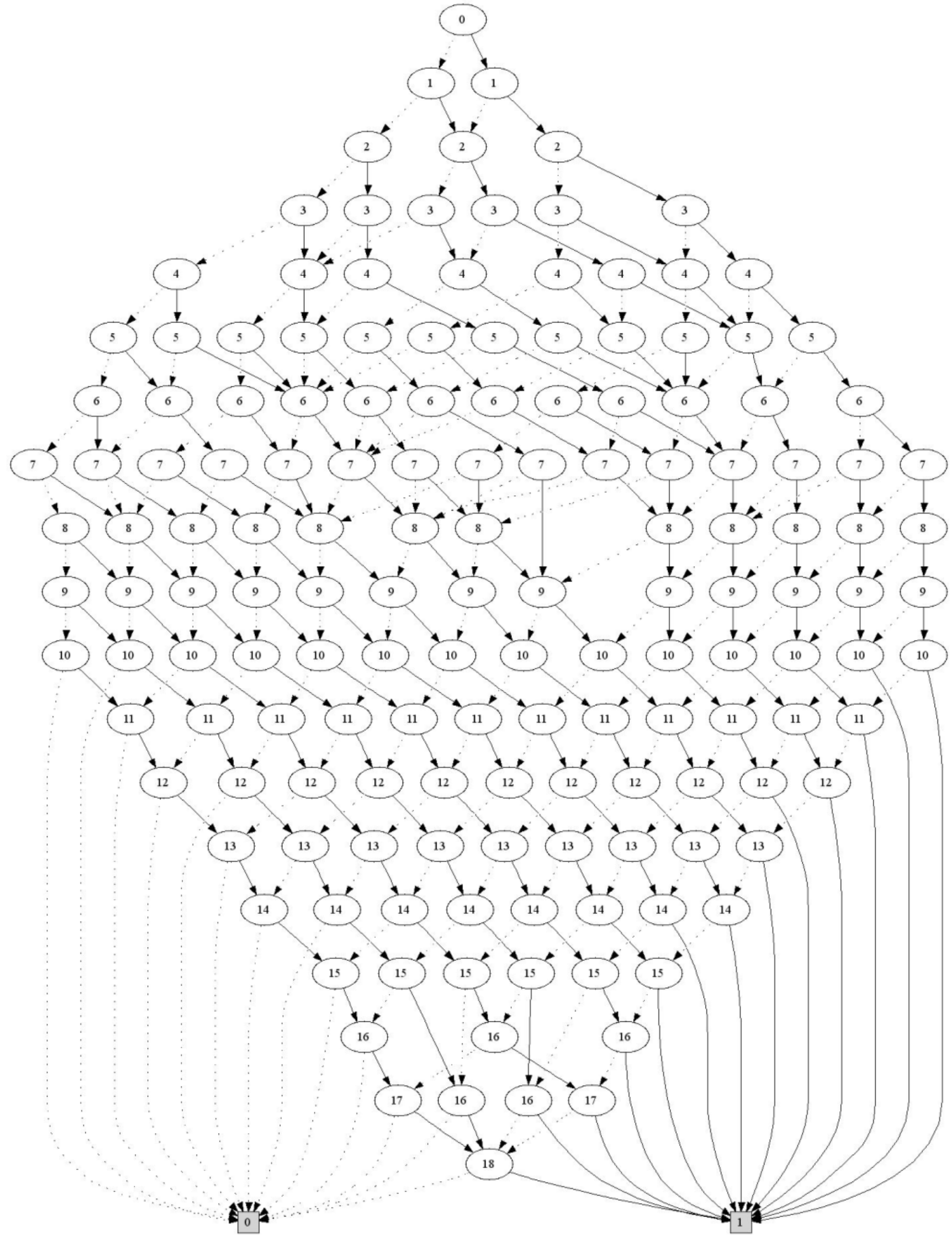
- Reduced DD for a knapsack constraint can be surprisingly small…

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 feasible solutions (or minimal covers)

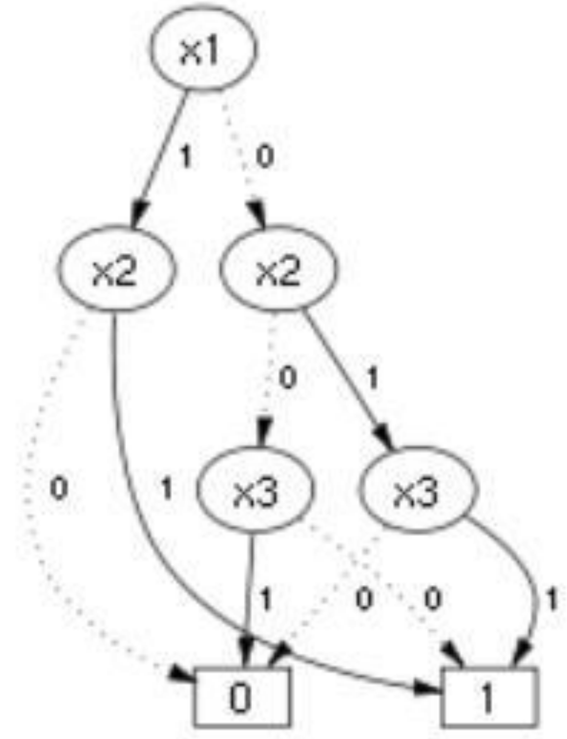
But its reduced BDD has only 152 nodes…



Optimization with Exact Decision Diagrams

- Decision diagrams can represent feasible set
  - Remove paths to 0.
  - Paths to 1 are feasible solutions.
  - Associate costs with arcs.
  - Find longest/shortest path

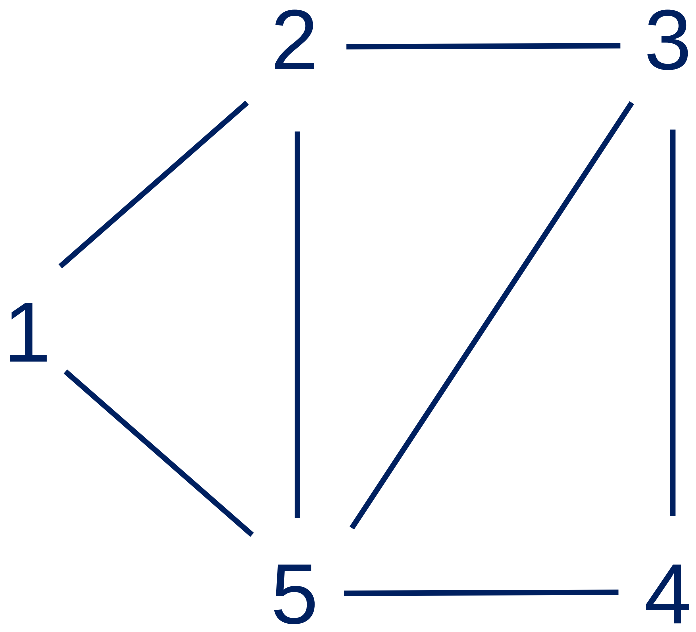
Hadžić and JH (2006, 2007)

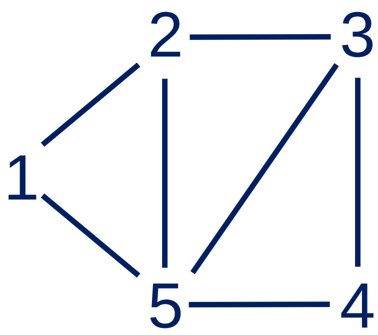


Stable Set Problem

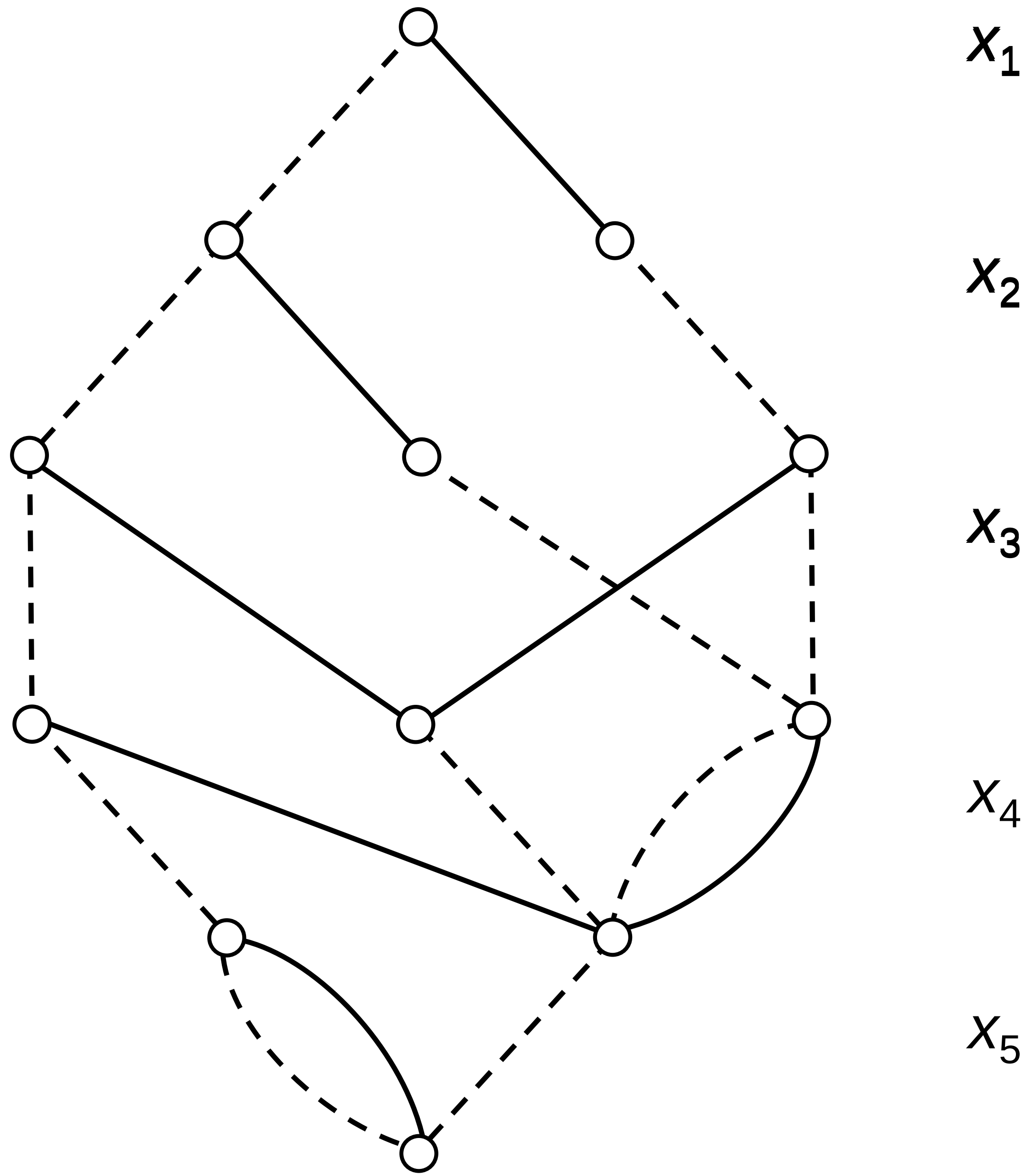
Let each vertex have weight w_i

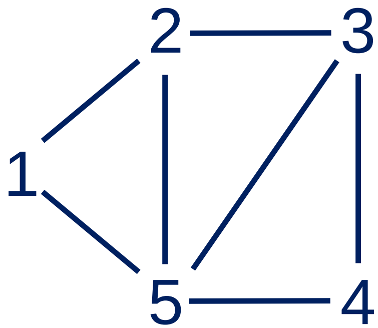
Select nonadjacent vertices to maximize $\sum_i w_i x_i$

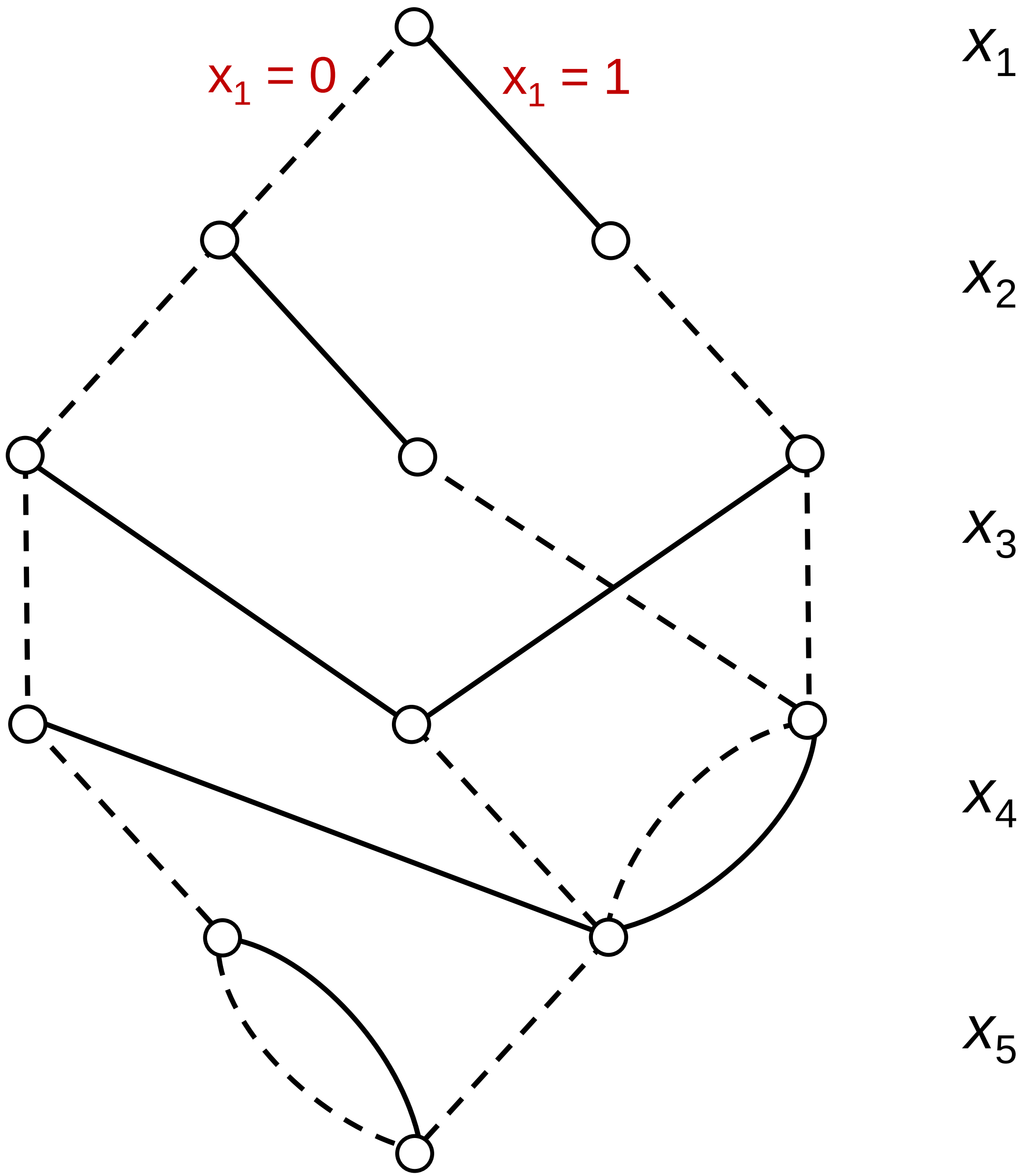




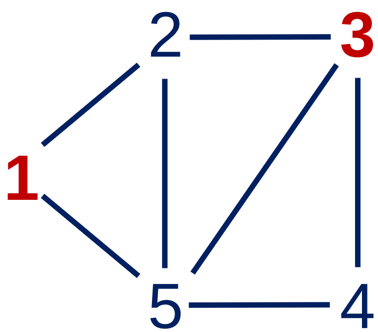
Exact DD for stable set problem



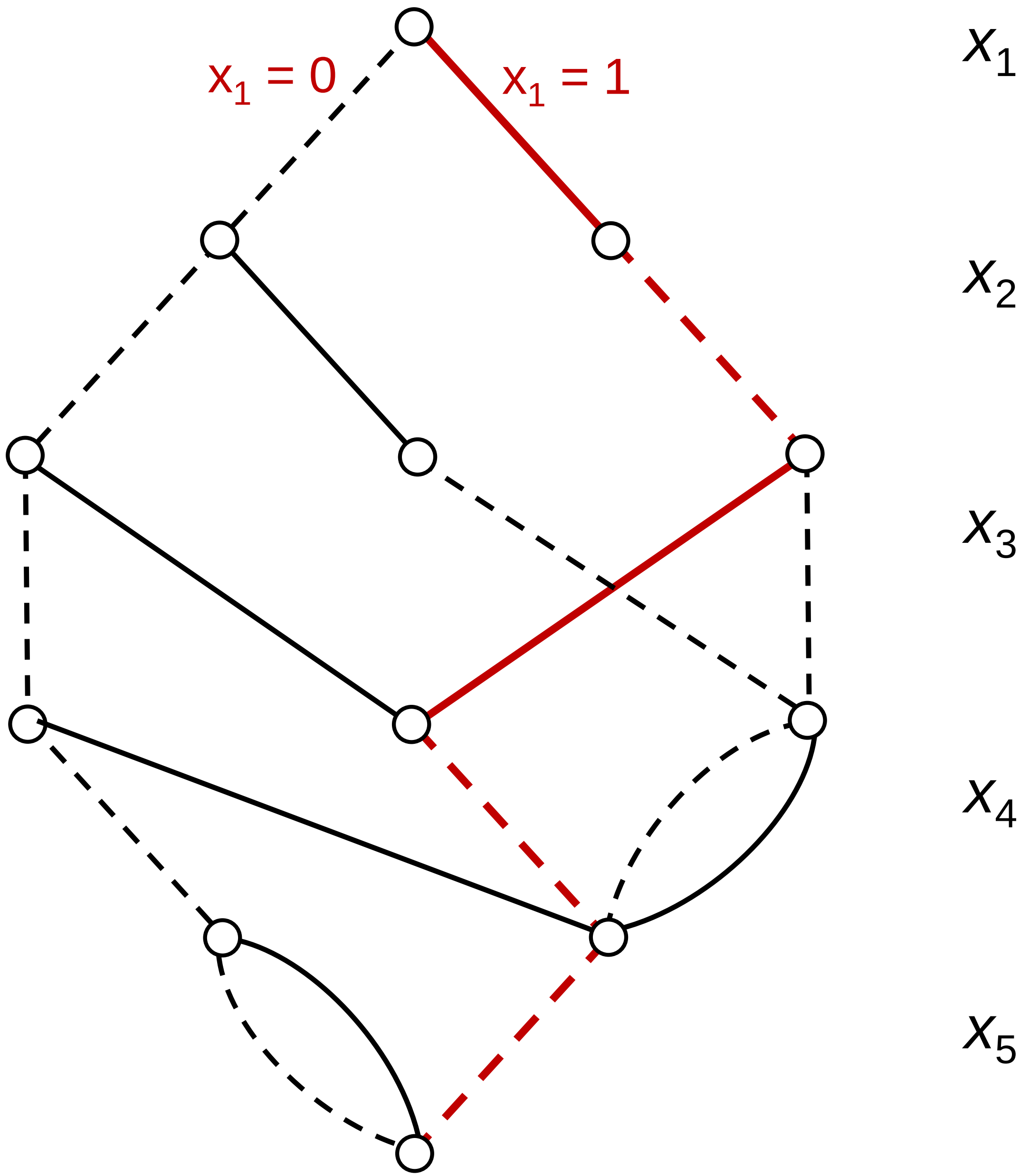


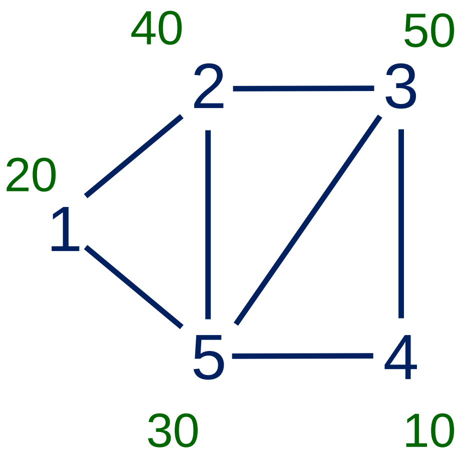


Exact DD for stable set problem

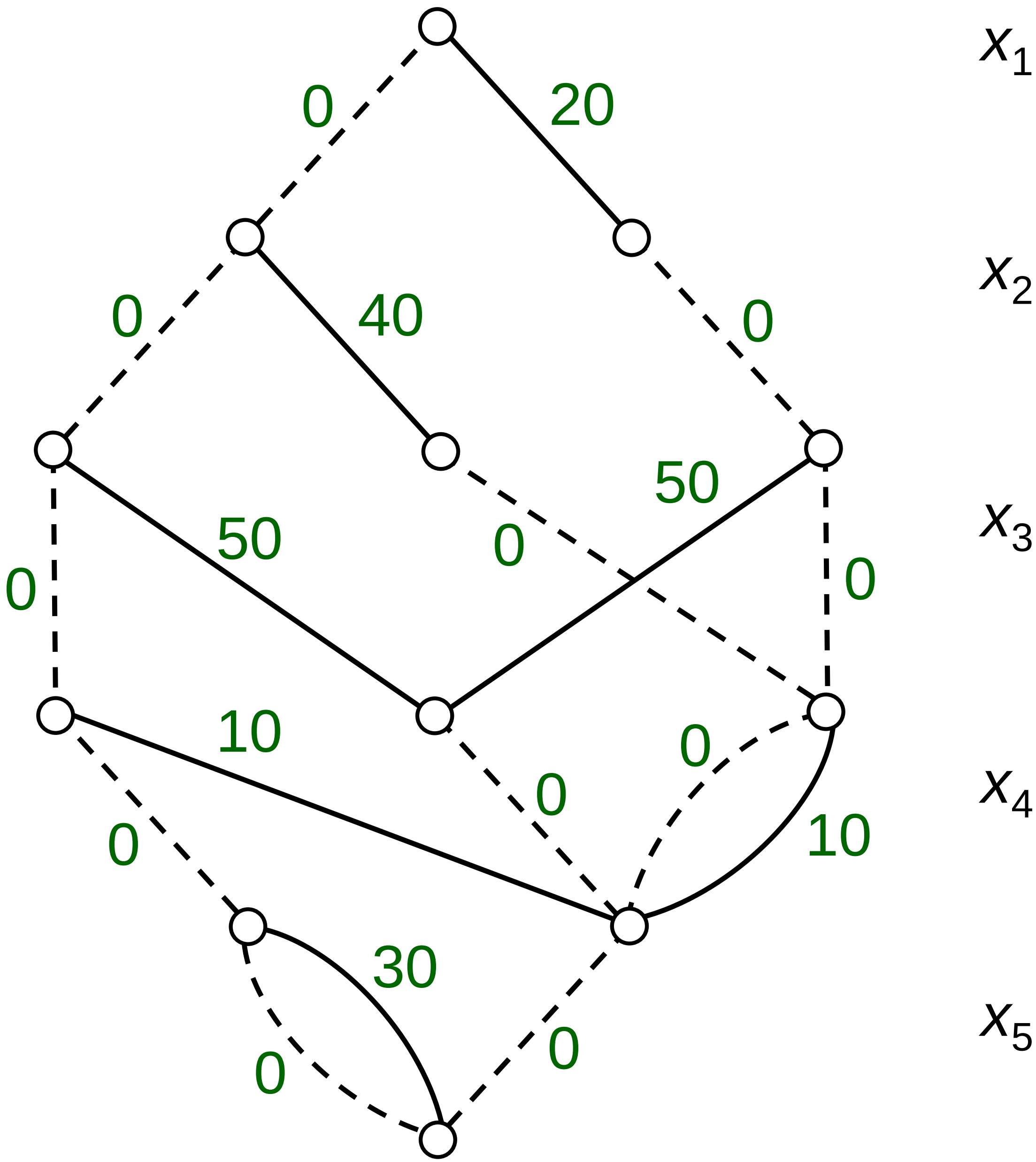


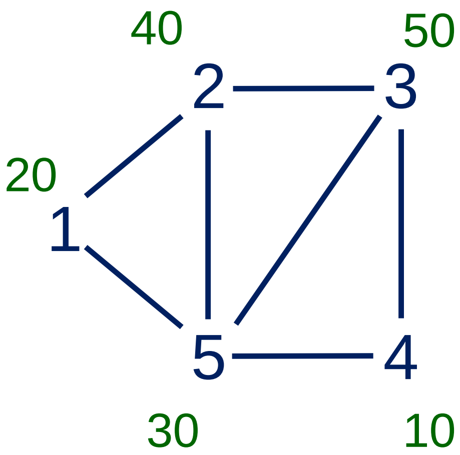
Paths from top to bottom correspond to the 9 feasible solutions

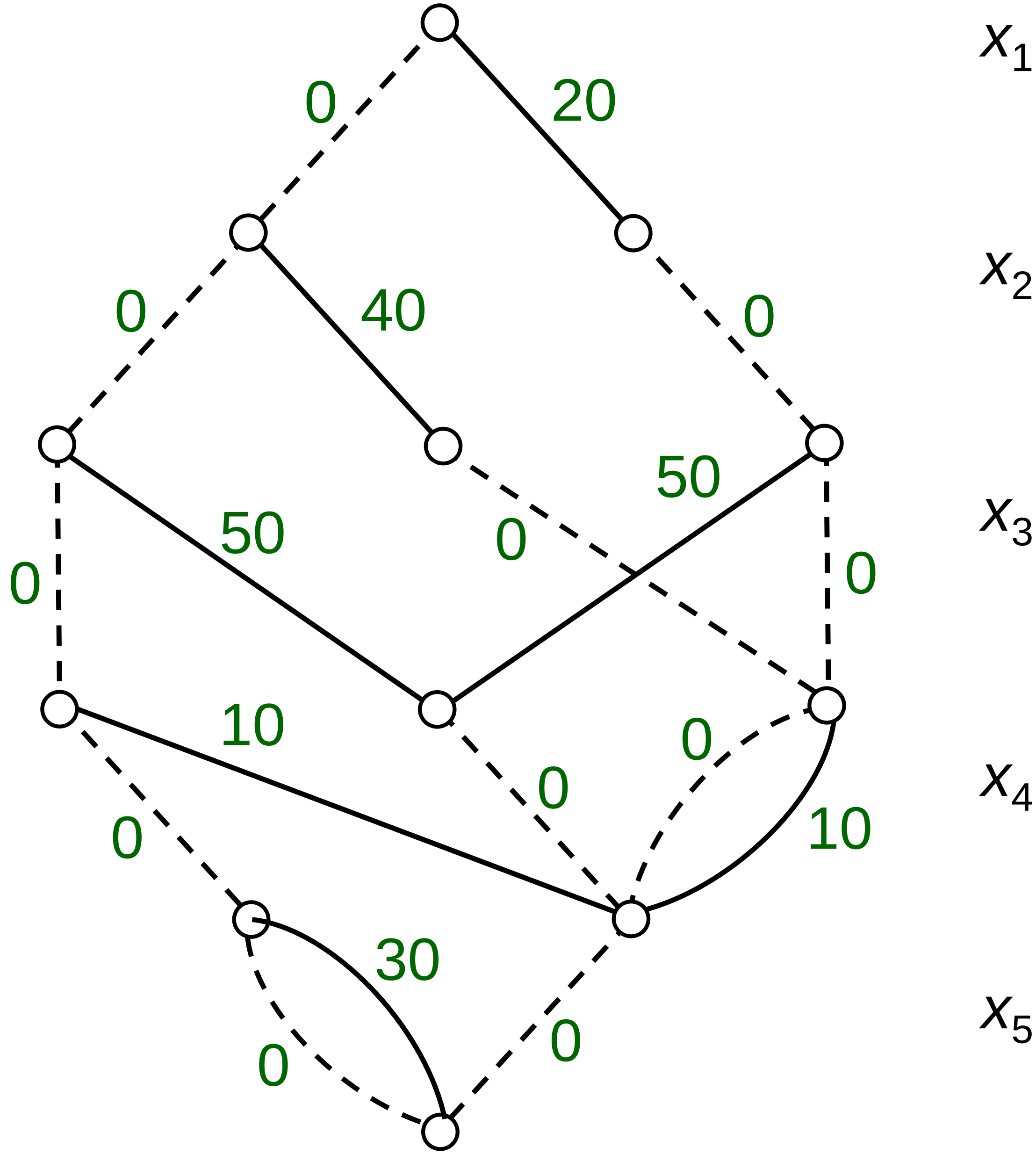




For objective function, associate weights with arcs

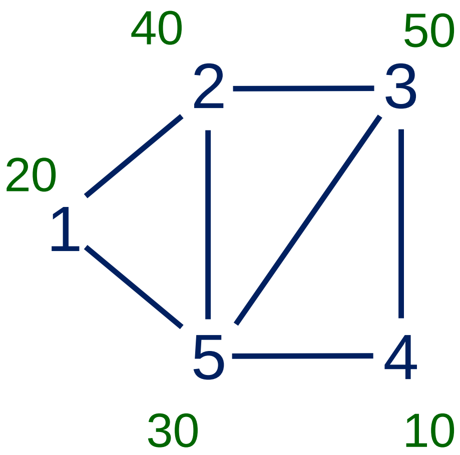




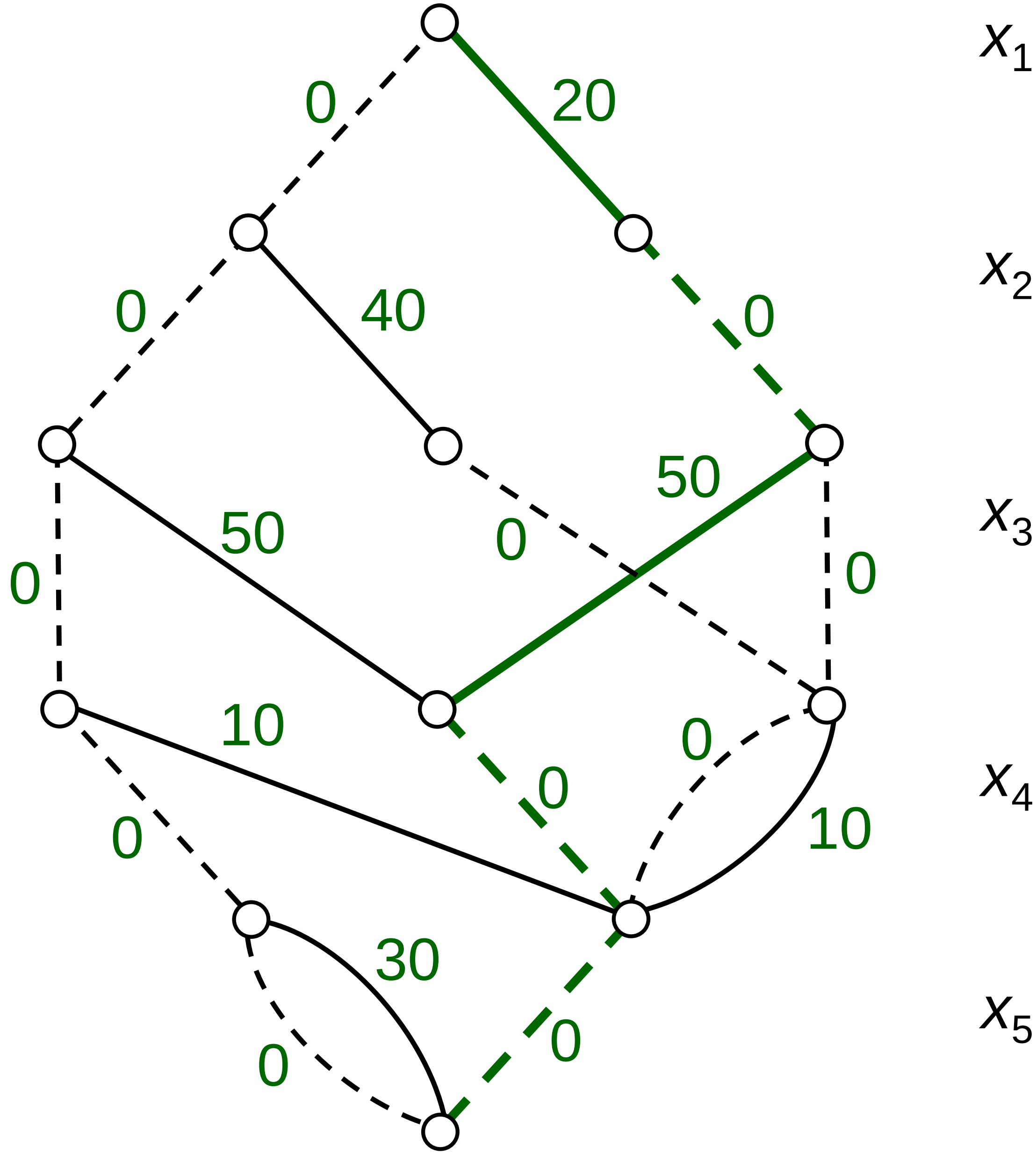


For objective function, associate weights with arcs

Optimal solution is **longest path**



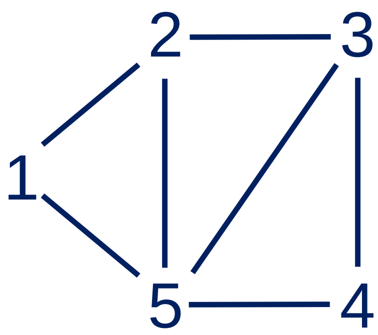
For objective function, associate weights with arcs

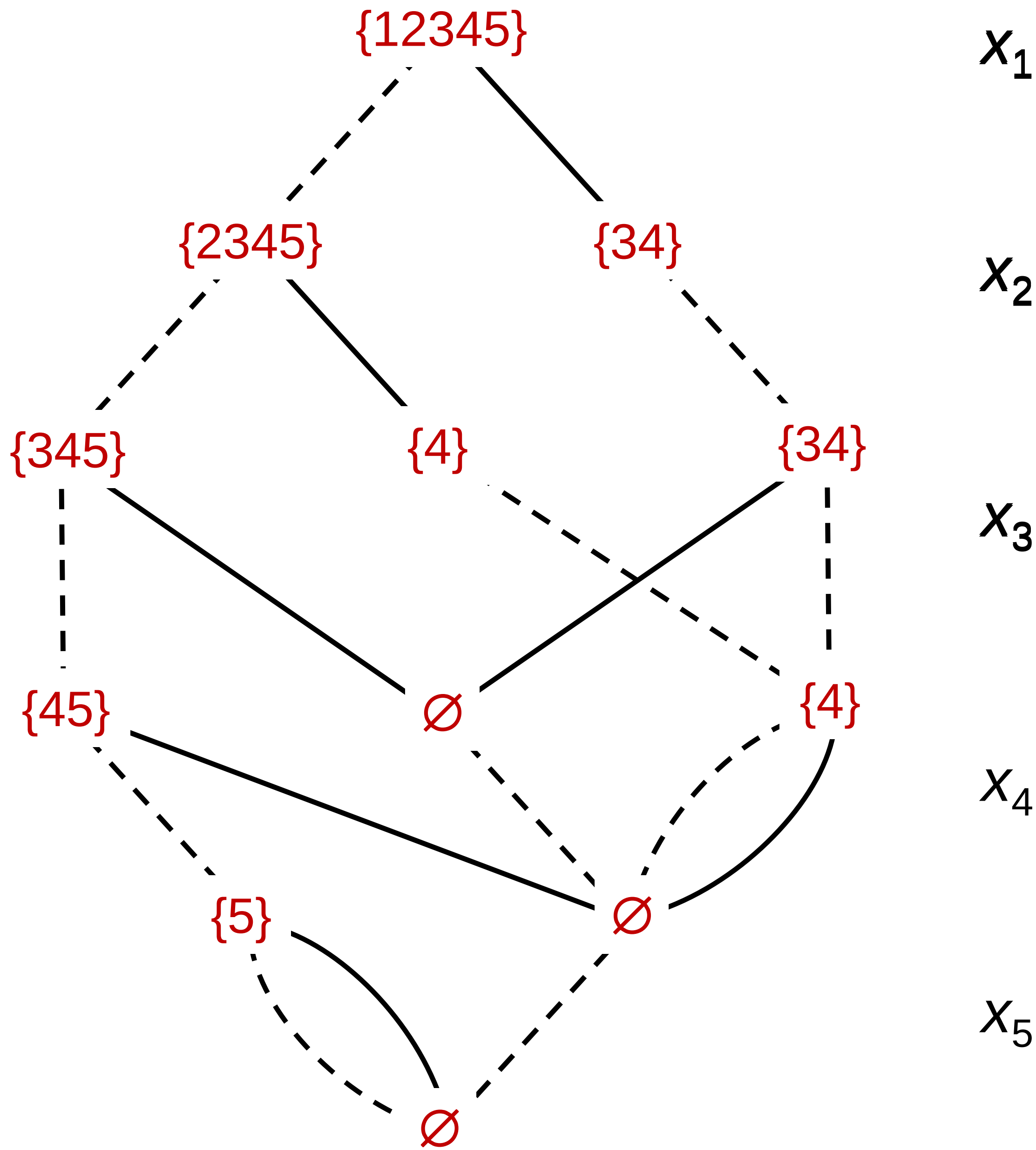


Optimal solution is **longest path**

Exact DD Compilation

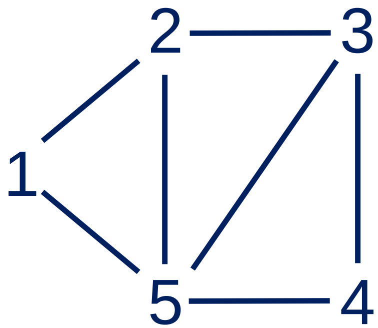
- Build an exact DD by associating a **state** with each node.
  - Merge nodes with **identical states**.





Exact DD for stable set problem

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



{12345}

x_1

x_2

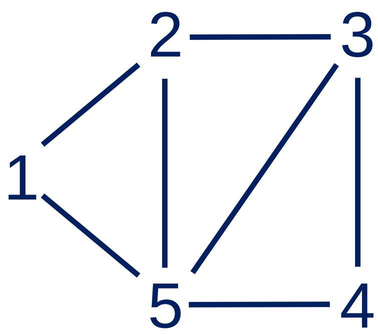
Exact DD for stable set problem

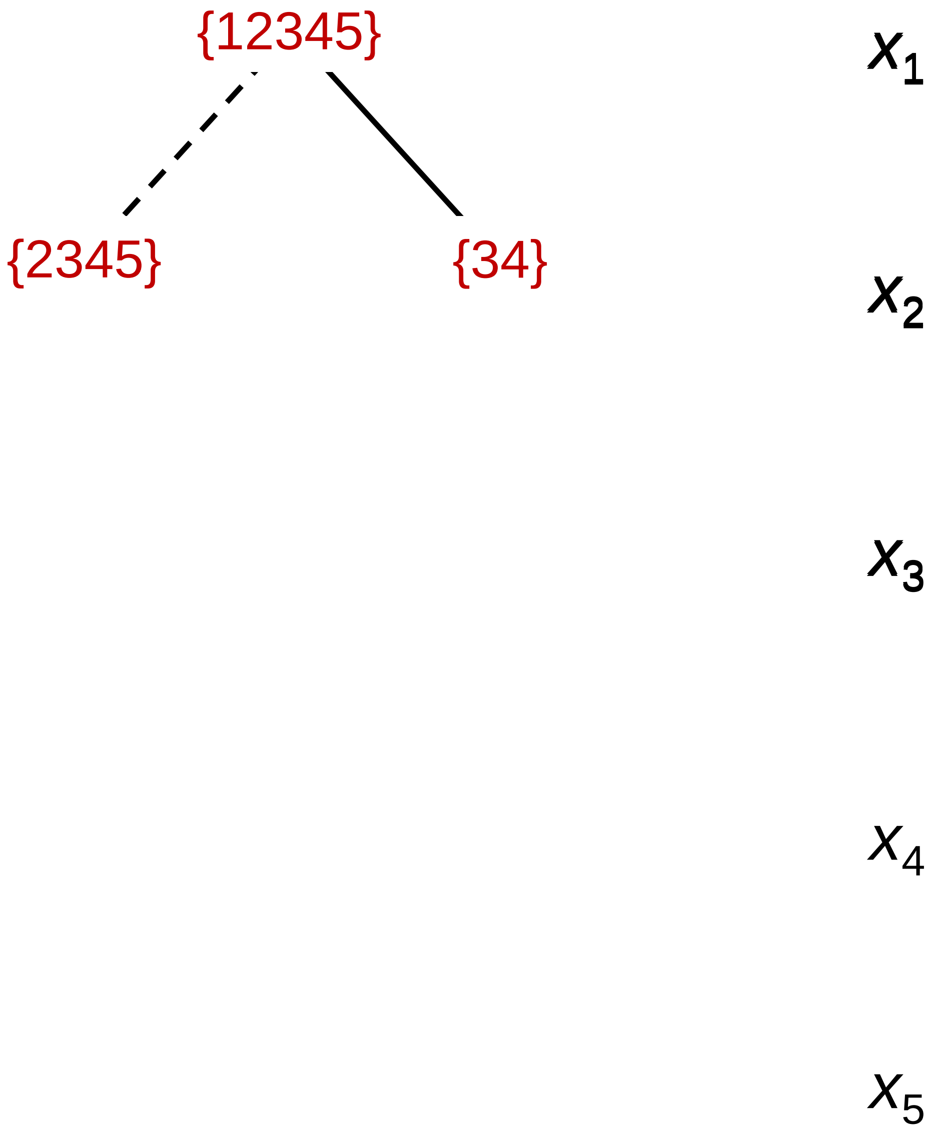
x_3

x_4

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

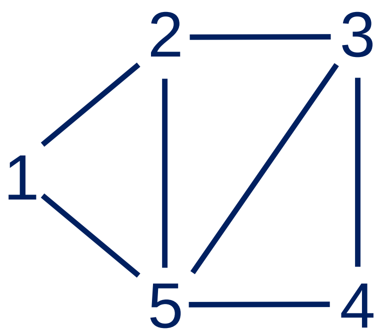
x_5

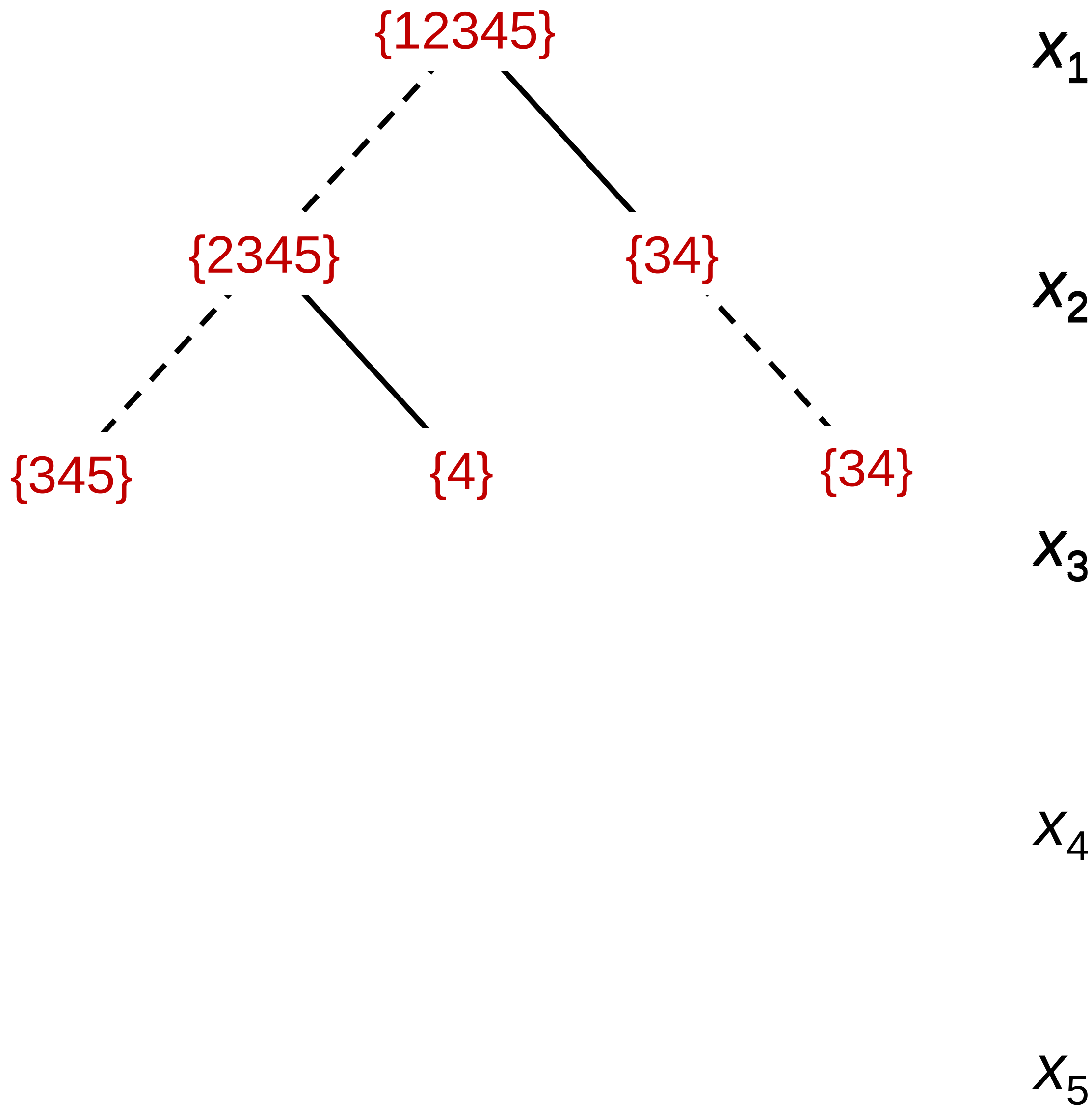




Exact DD for stable set problem

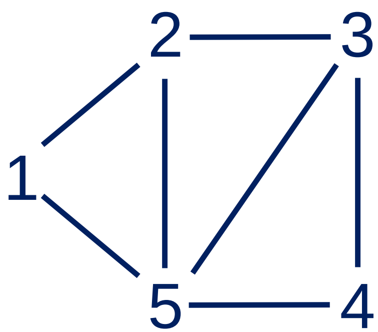
To build DD, associate **state** with each node



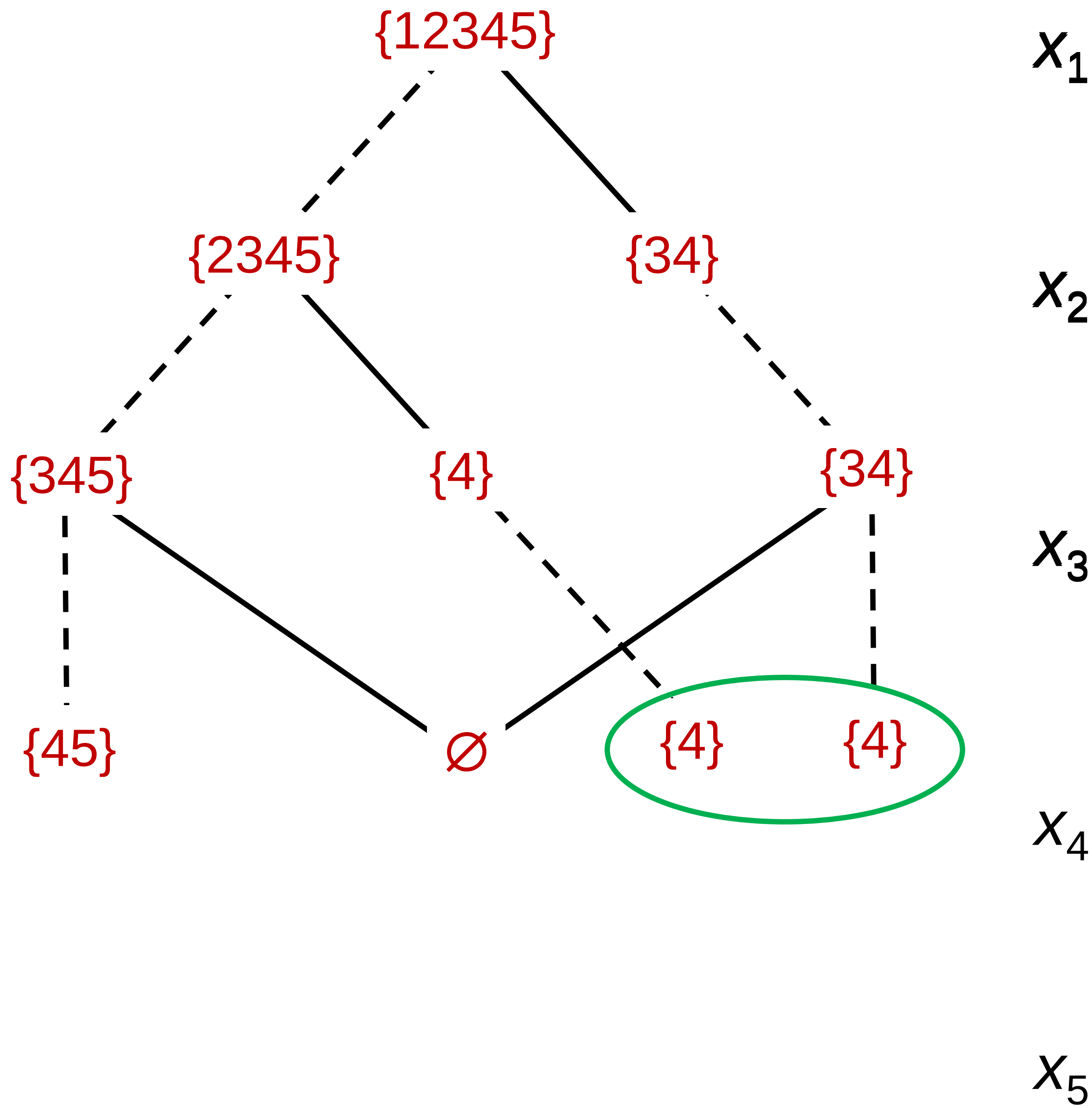


Exact DD for stable set problem

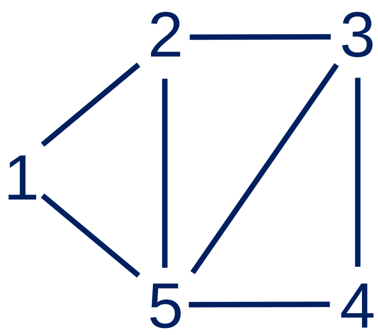
To build DD, associate **state** with each node

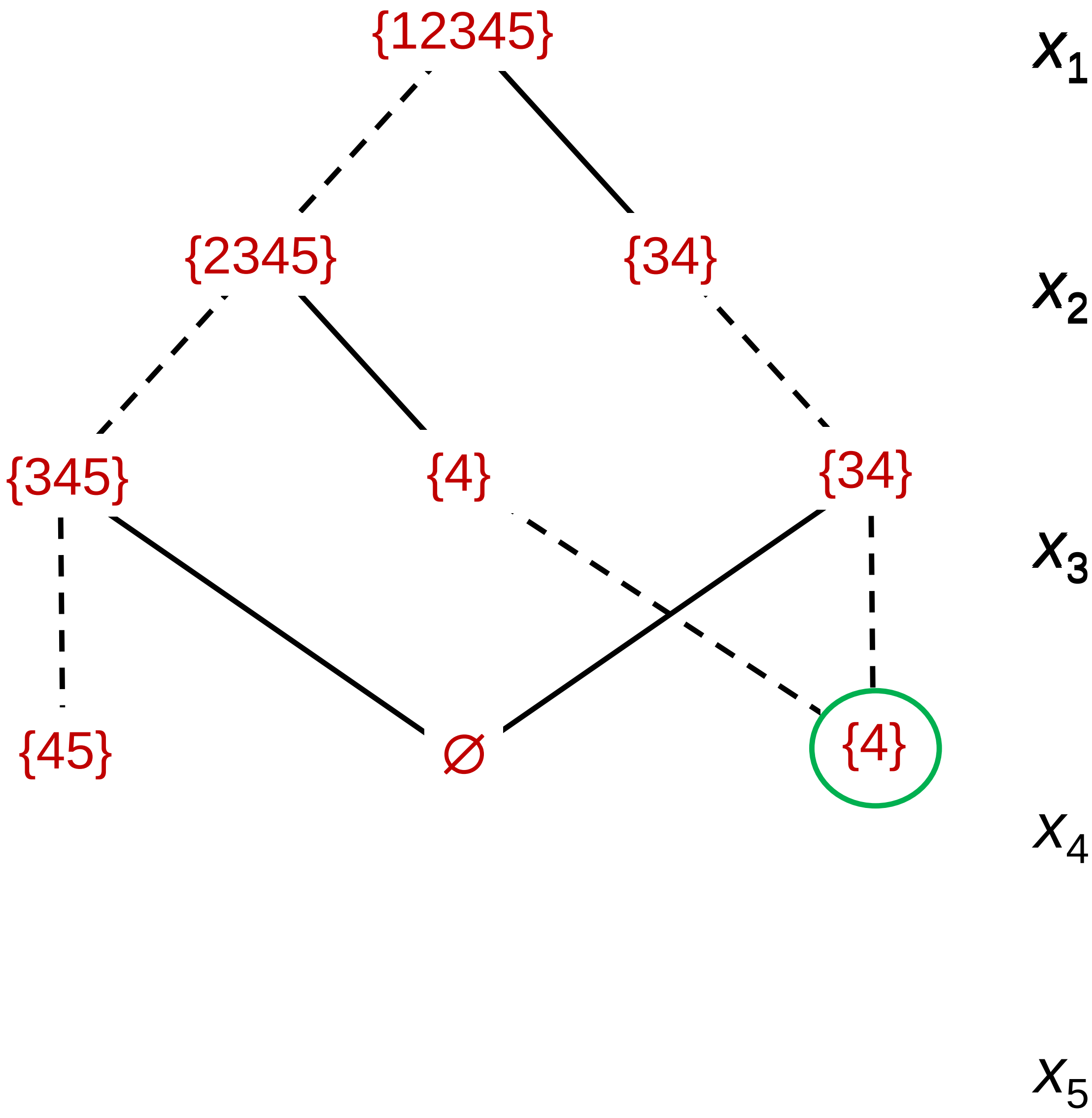


Exact DD for stable set problem



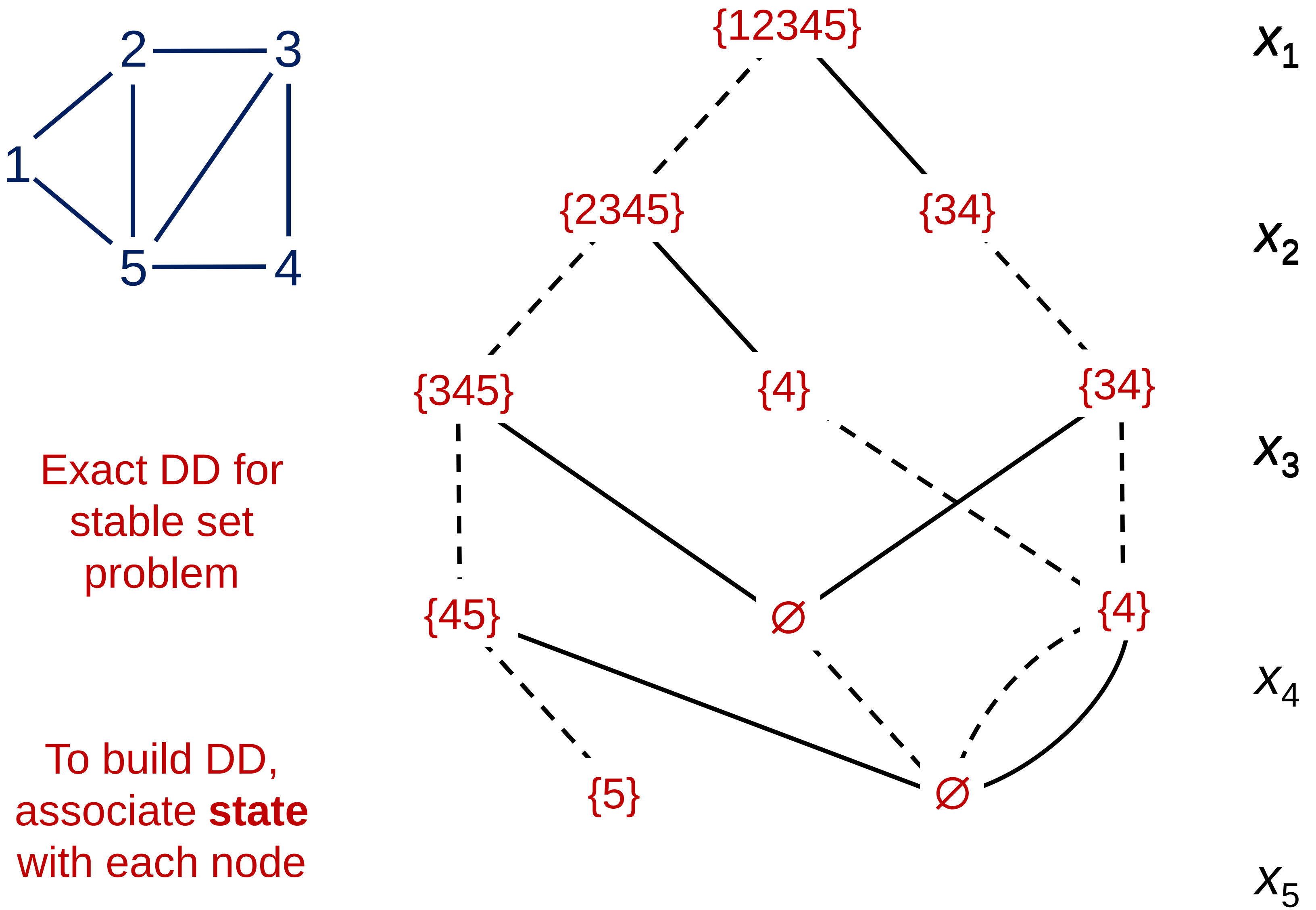
Merge nodes that correspond to the same state





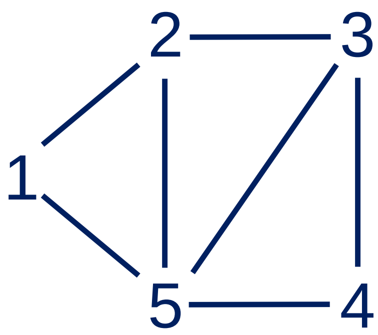
Exact DD for stable set problem

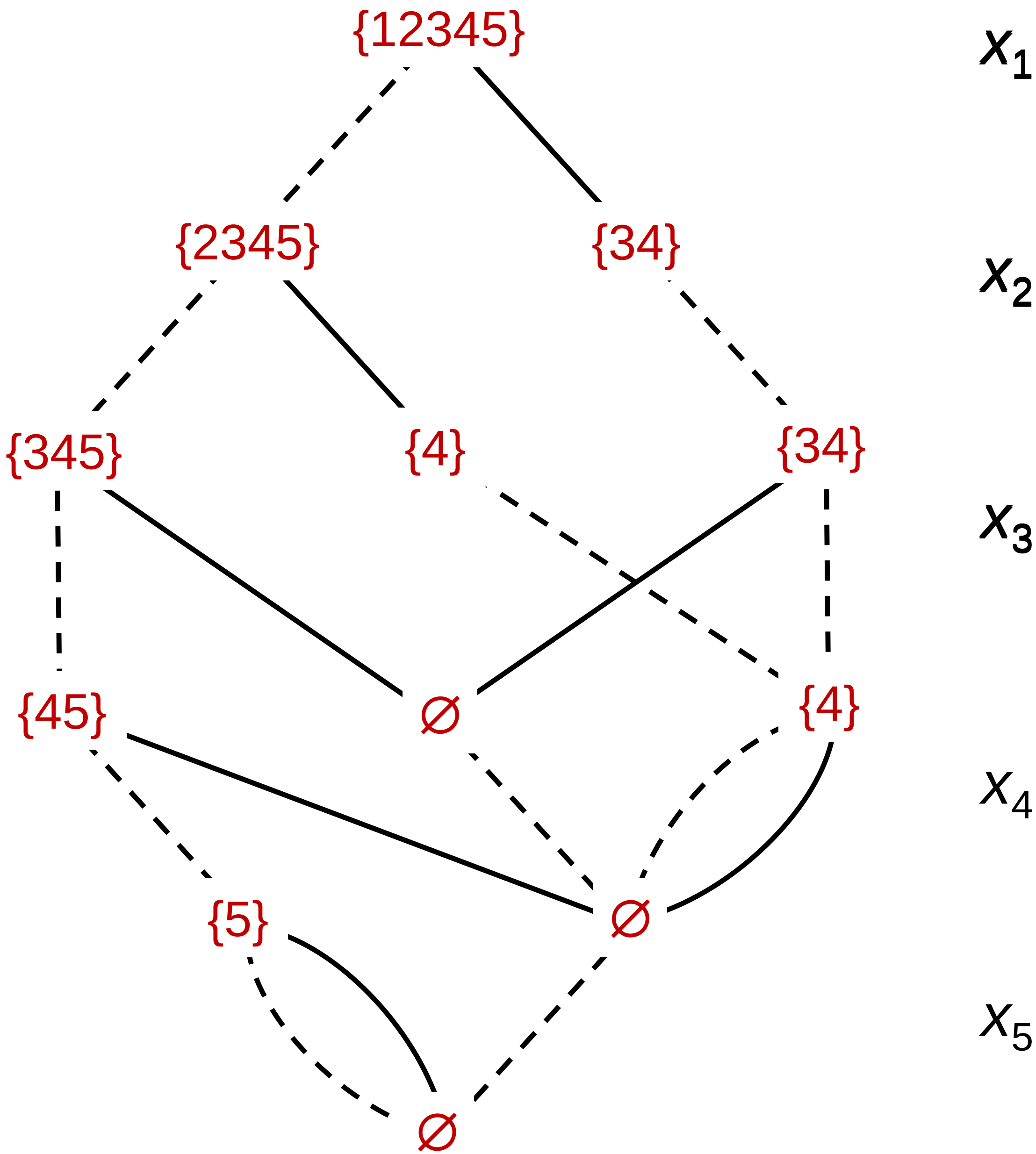
Merge nodes that correspond to the same state



Exact DD for stable set problem

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





Exact DD for stable set problem

Resulting DD is **not necessarily reduced** (it is in this case).

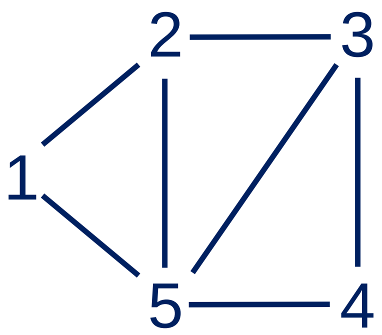
A General-Purpose Solver

- The decision diagram tends to grow exponentially**.**
- To build a **practical solver:**
  - Use limited-width **relaxed** decision diagrams to bound the objective value.
  - Use limited-width **restricted** decision diagrams for primal heuristic
  - Use a **recursive dynamic programming model**.
  - Use **novel branching scheme** within relaxed decision diagrams.

Relaxed Decision Diagrams

- A **relaxed** DD represents a superset of feasible set.
  - Shortest (longest) path length is a **bound** on optimal value.
  - **Size of DD is controlled**.
  - Analogous to LP relaxation in IP, but **discrete**.
  - Does **not** require **linearity**, **convexity**, or **inequality** constraints.

Andersen, Hadžić, JH, Tiedemann (2007)



{12345}

x_1

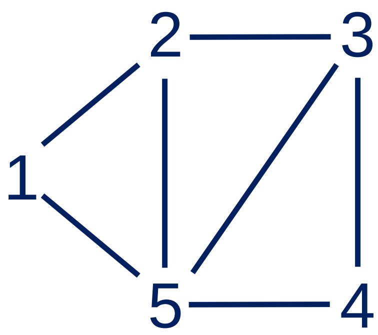
x_2

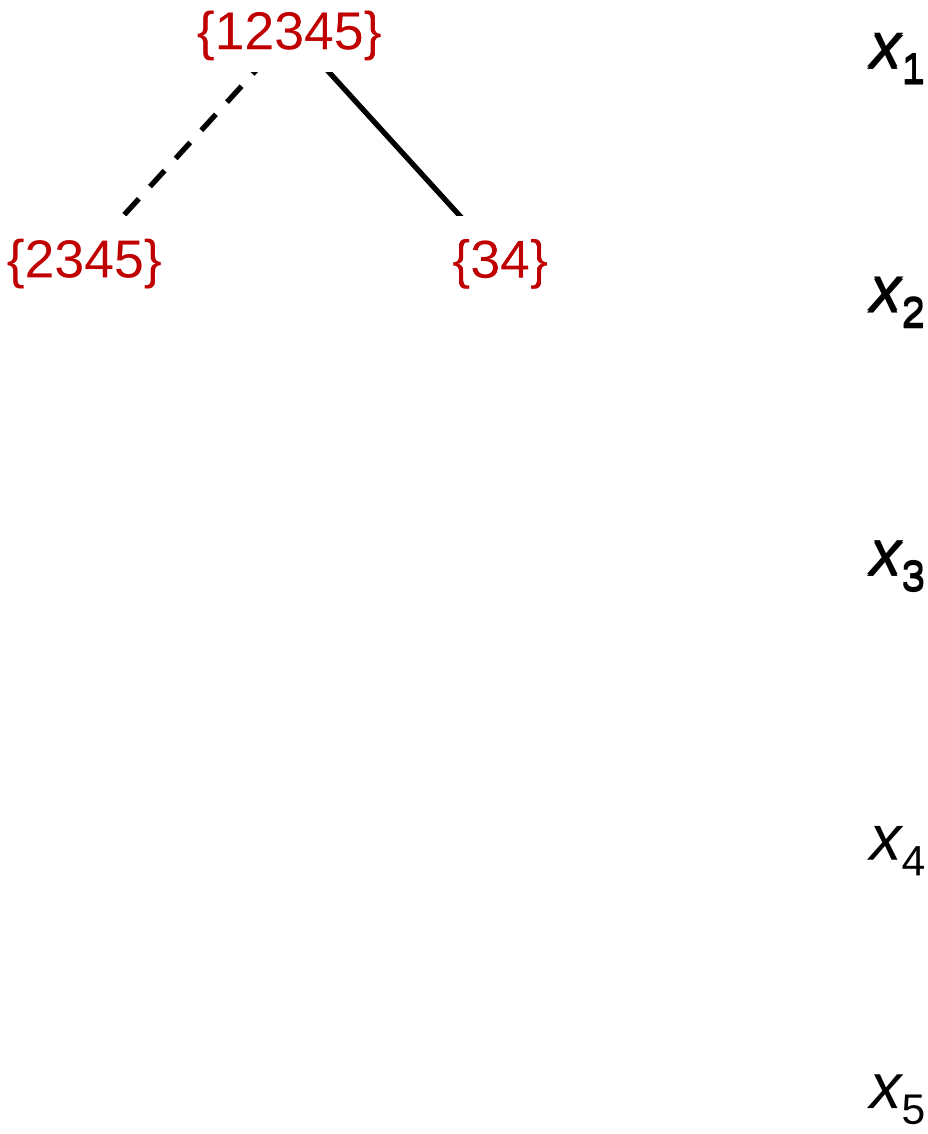
x_3

x_4

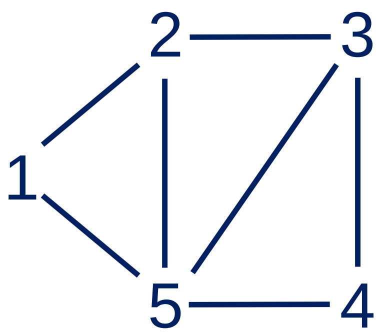
x_5

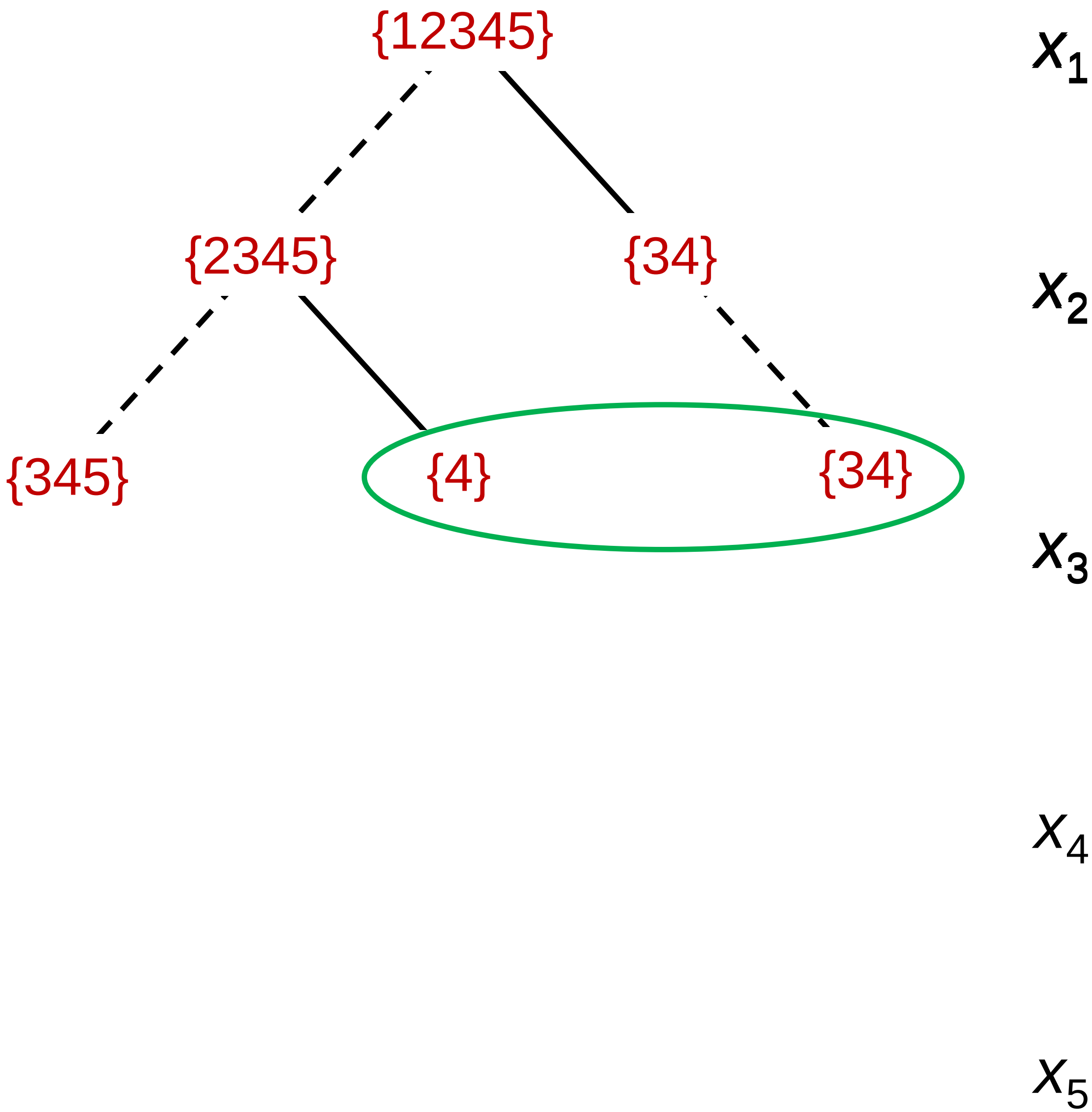
To build **relaxed** DD, merge some additional nodes as we go along





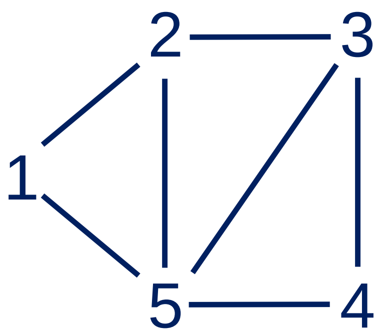
To build **relaxed** DD, merge some additional nodes as we go along

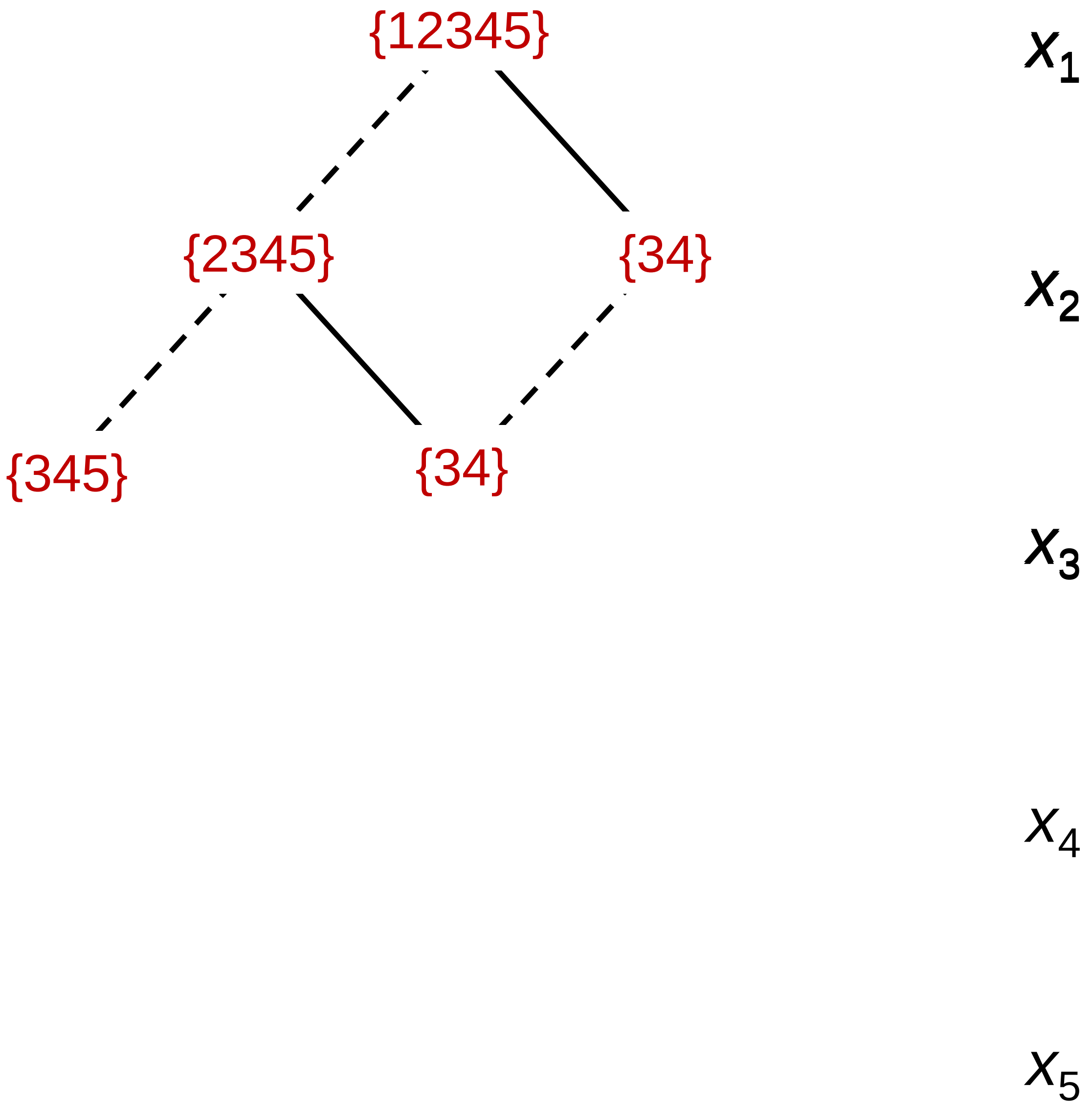




To build **relaxed** DD, merge some additional nodes as we go along.

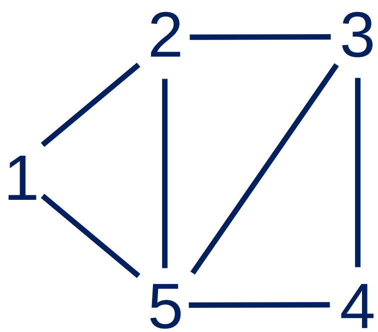
Take the **union** of merged states

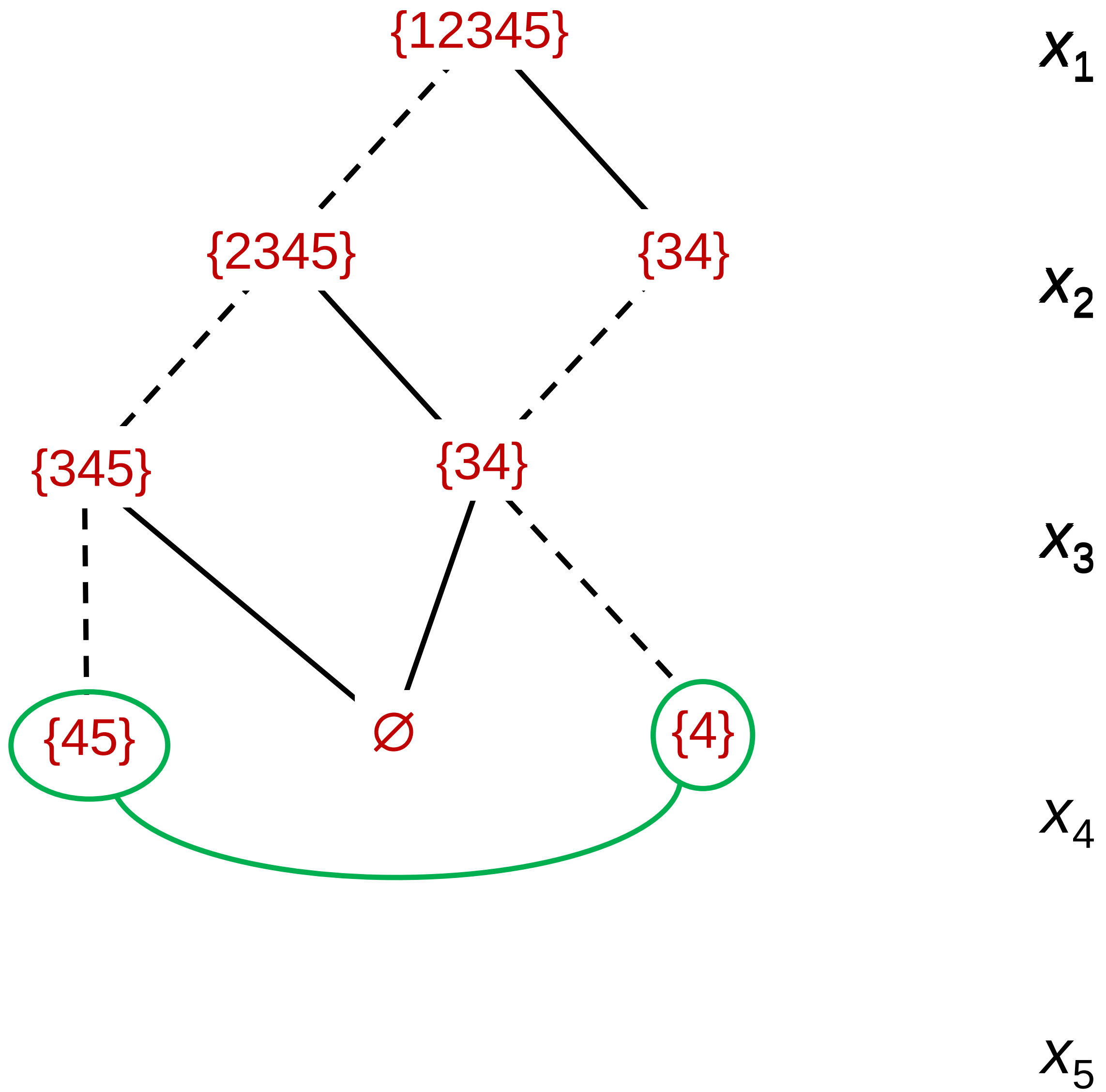




To build **relaxed** DD, merge some additional nodes as we go along.

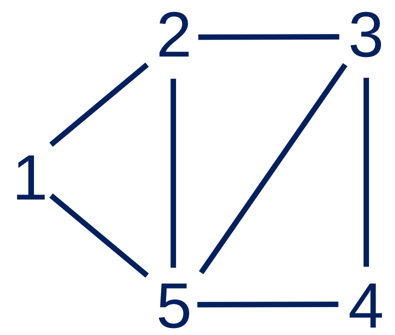
Take the **union** of merged states.

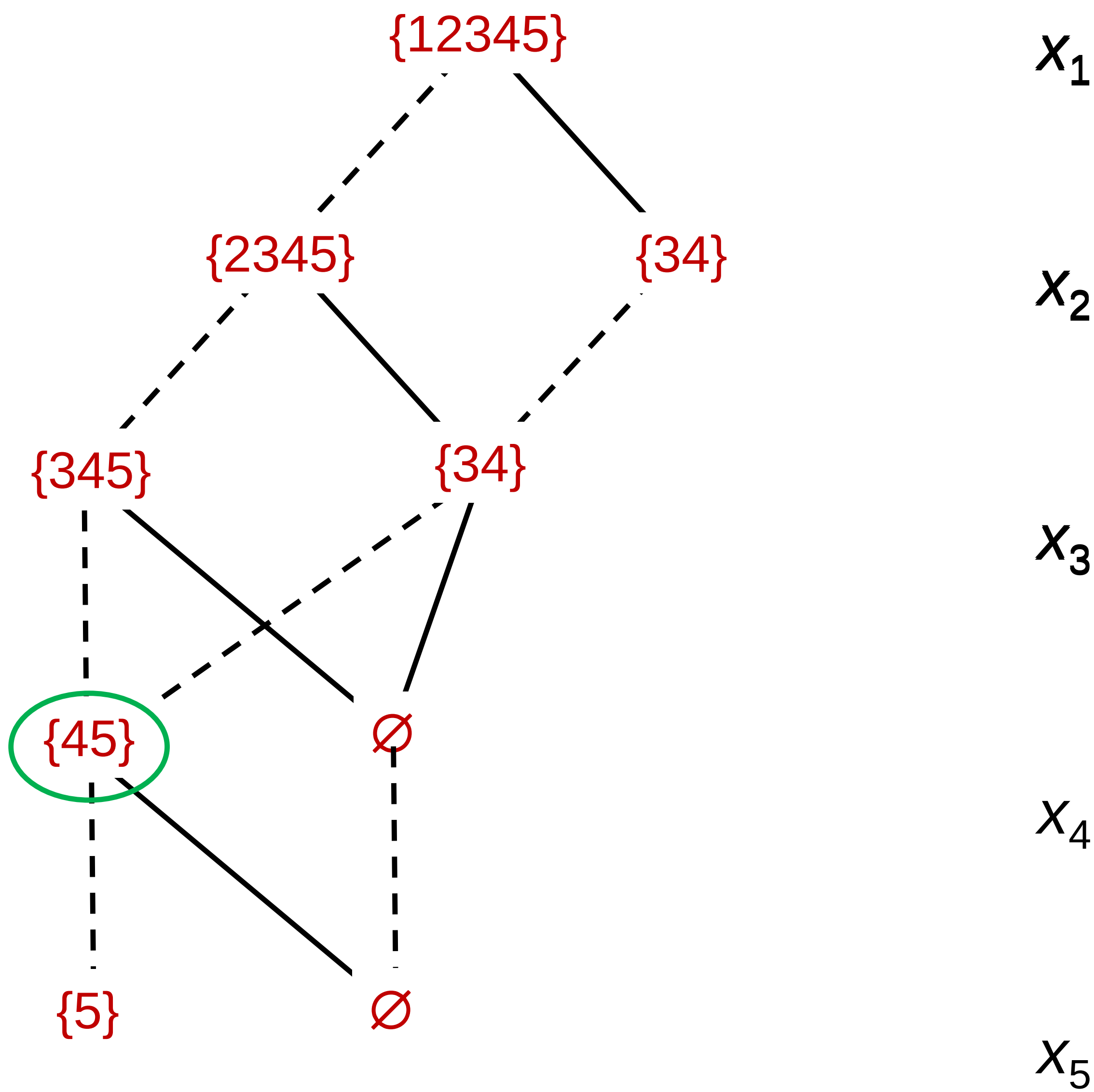




To build **relaxed** DD, merge some additional nodes as we go along.

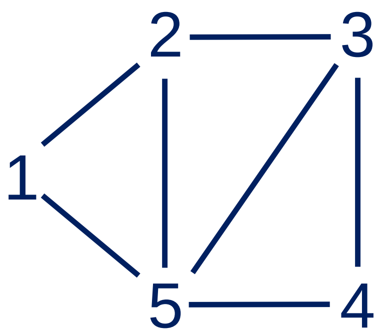
Take the **union** of merged states.





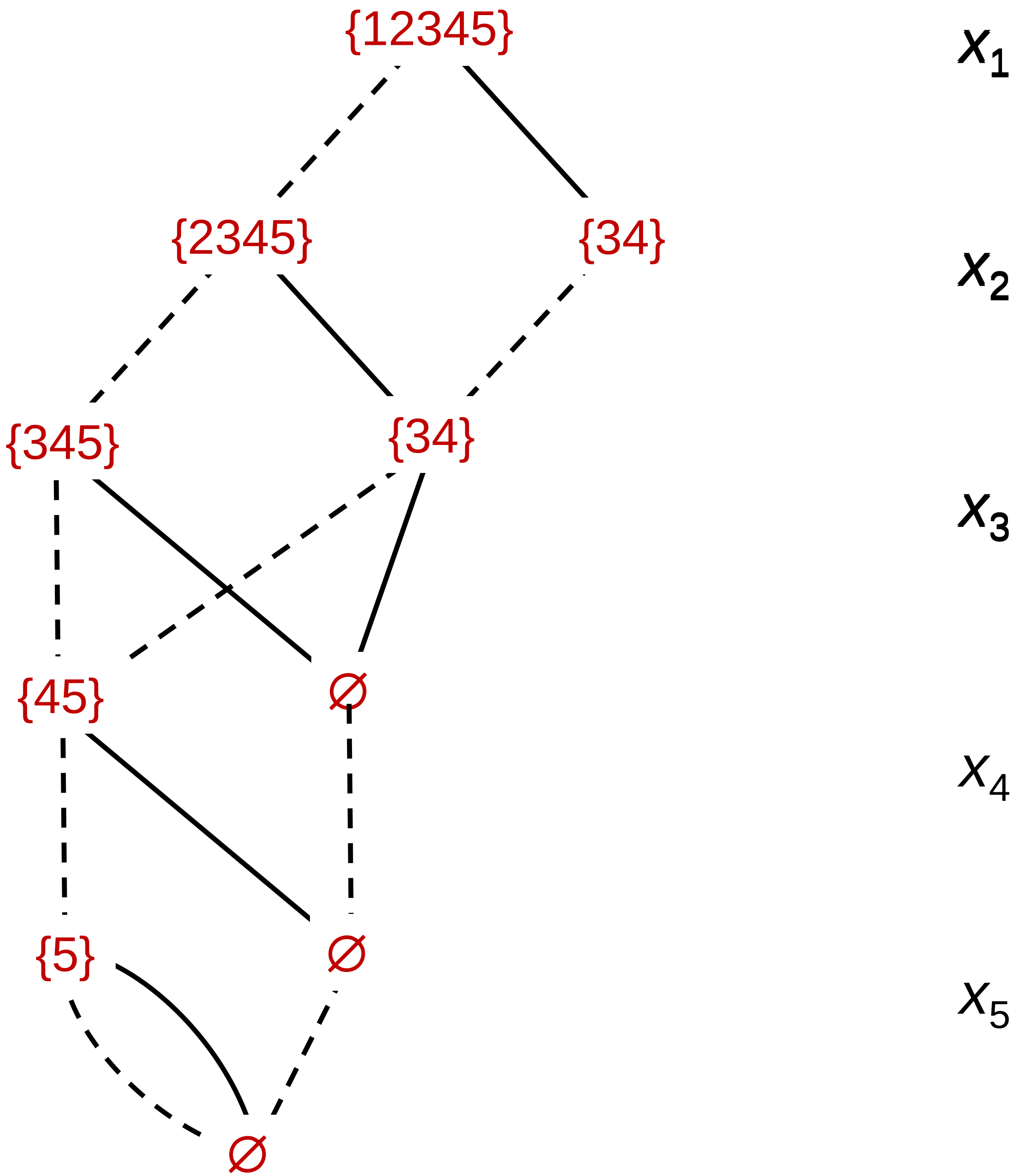
To build **relaxed** DD, merge some additional nodes as we go along.

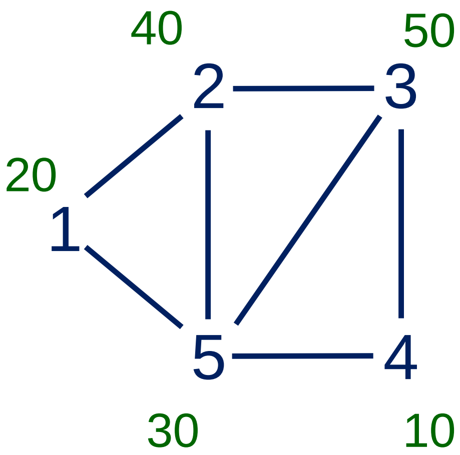
Take the **union** of merged states.



Width = 2

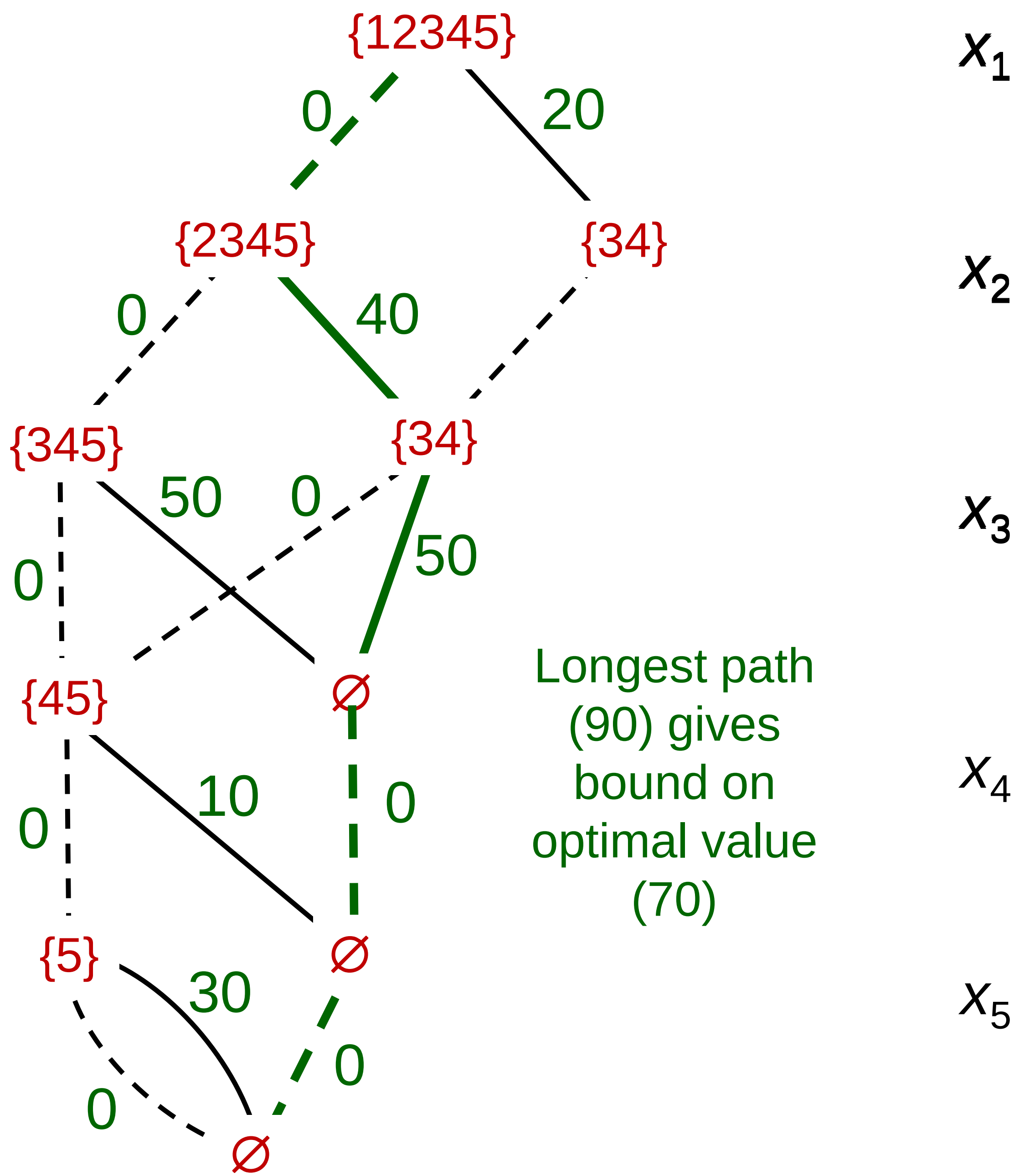
Represents 11 solutions, including 9 feasible solutions





Width = 2

Represents
11 solutions,
including
9 feasible
solutions



Longest path
(90) gives
bound on
optimal value
(70)

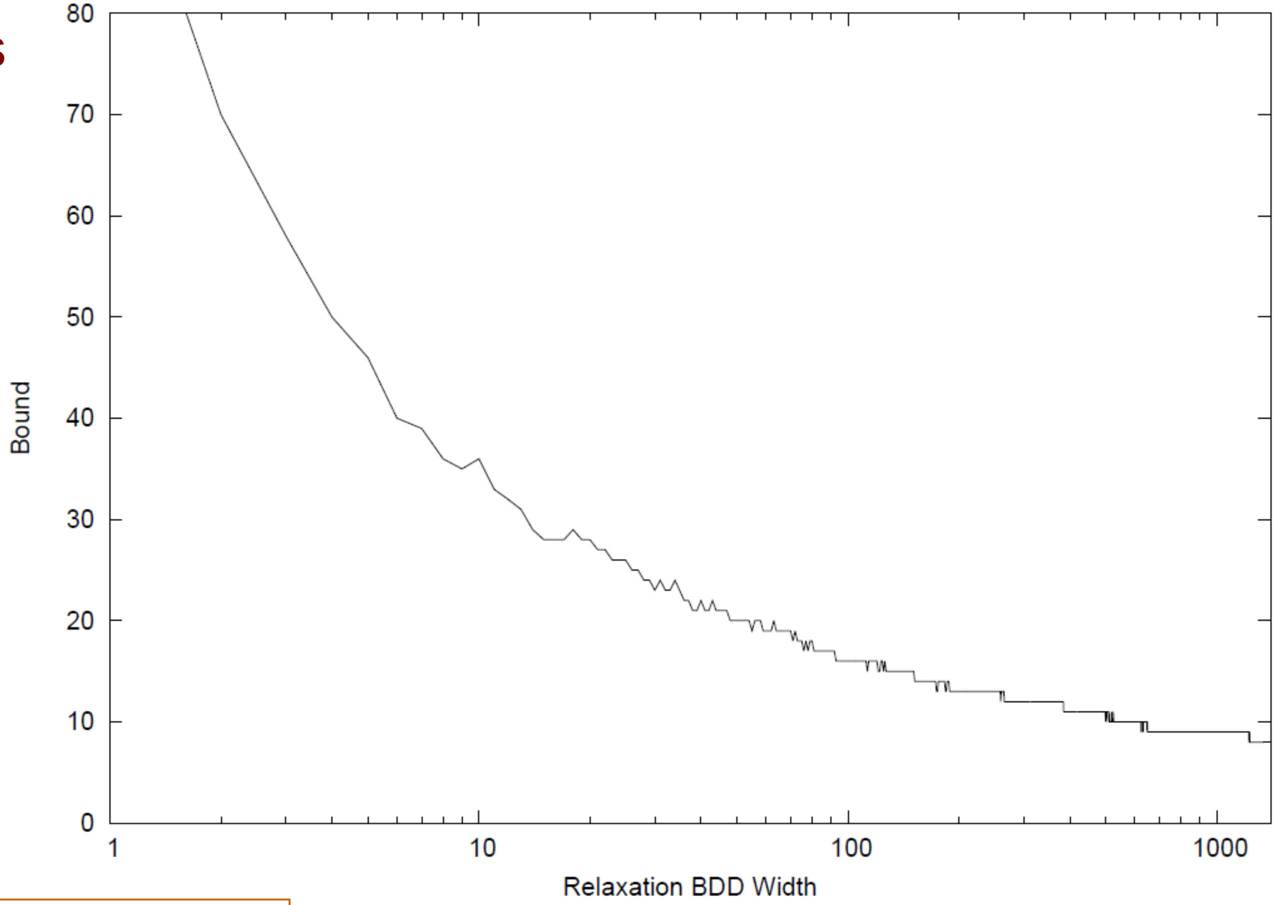
Relaxed Decision Diagrams

- Original application: enhanced propagation in constraint programming
  - In multiple alldiff problem (graph coloring), reduced 1 million node search trees to 1 node.

Andersen, Hadžić, JH, Tiedemann (2007)

Relaxed Decision Diagrams

- Wider diagrams yield tighter bounds
  - But take longer to build.
  - Adjust width dynamically.

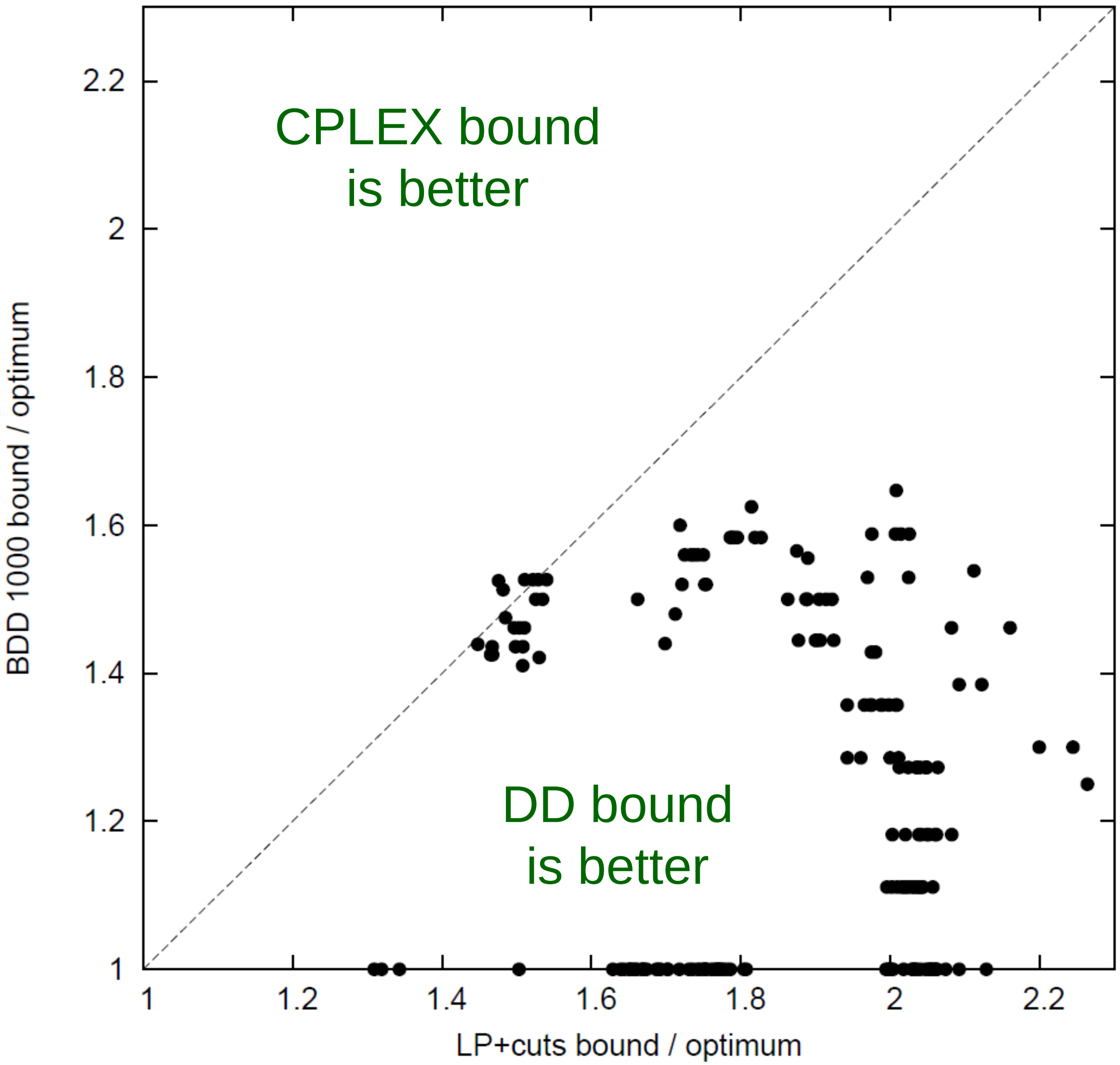


Bergman, Ciré, van Hoeve, JH (2013)

Relaxed Decision Diagrams

- DDs vs. CPLEX bound at root node for max stable set problem
  - Using CPLEX default cut generation
  - DD max width of 1000.
  - DDs require about 5% the time of CPLEX

Bergman, Ciré, van Hoeve, JH (2013)



Restricted Decision Diagrams

- A **restricted** DD represents a **subset** of the feasible set.
- Restricted DDs provide a basis for a **primal heuristic**.
  - Shortest (longest) paths in the restricted DD provide good feasible solutions.
  - Generate a **limited-width** restricted DD by deleting nodes that appear unpromising.

Bergman, Ciré, van Hoeve, Yunes (2014)

Set covering problem

$$x_1 + x_2 + x_3 \geq 1$$

$$x_1 + x_4 + x_5 \geq 1$$

$$x_2 + x_4 + x_6 \geq 1$$

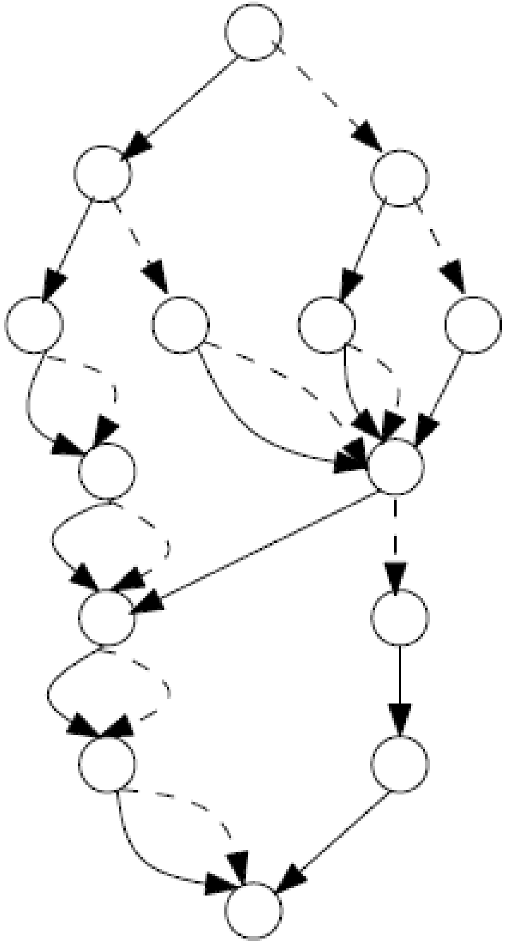
Sets

	1	2	3	4	5	6
A	●	●	●			
B	●			●	●	
C		●		●		●

52 feasible solutions.

Minimum cover of 2, e.g. x_1, x_2

Restricted DD of width 4

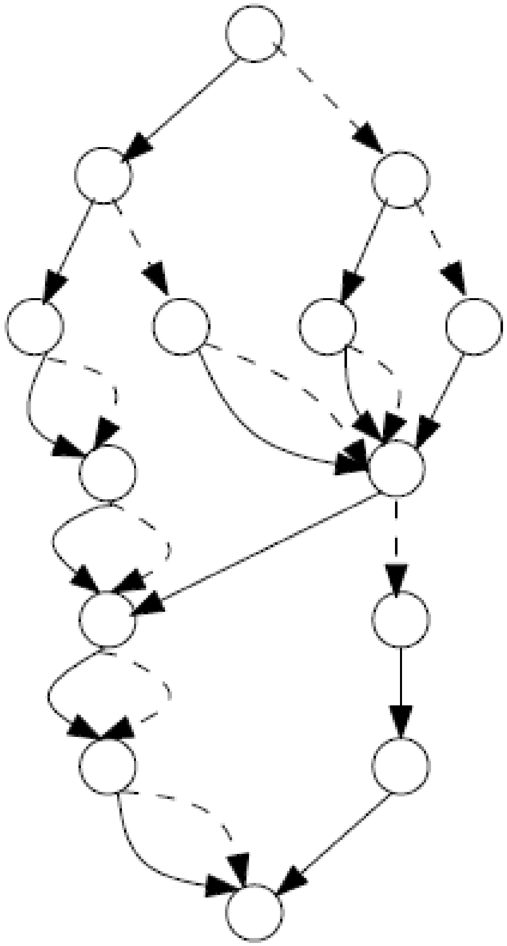


Several shortest paths have length 2.

All are minimum covers.

41 paths (< 52 feasible solutions)

Restricted DD of width 4



Several shortest paths have length 2.

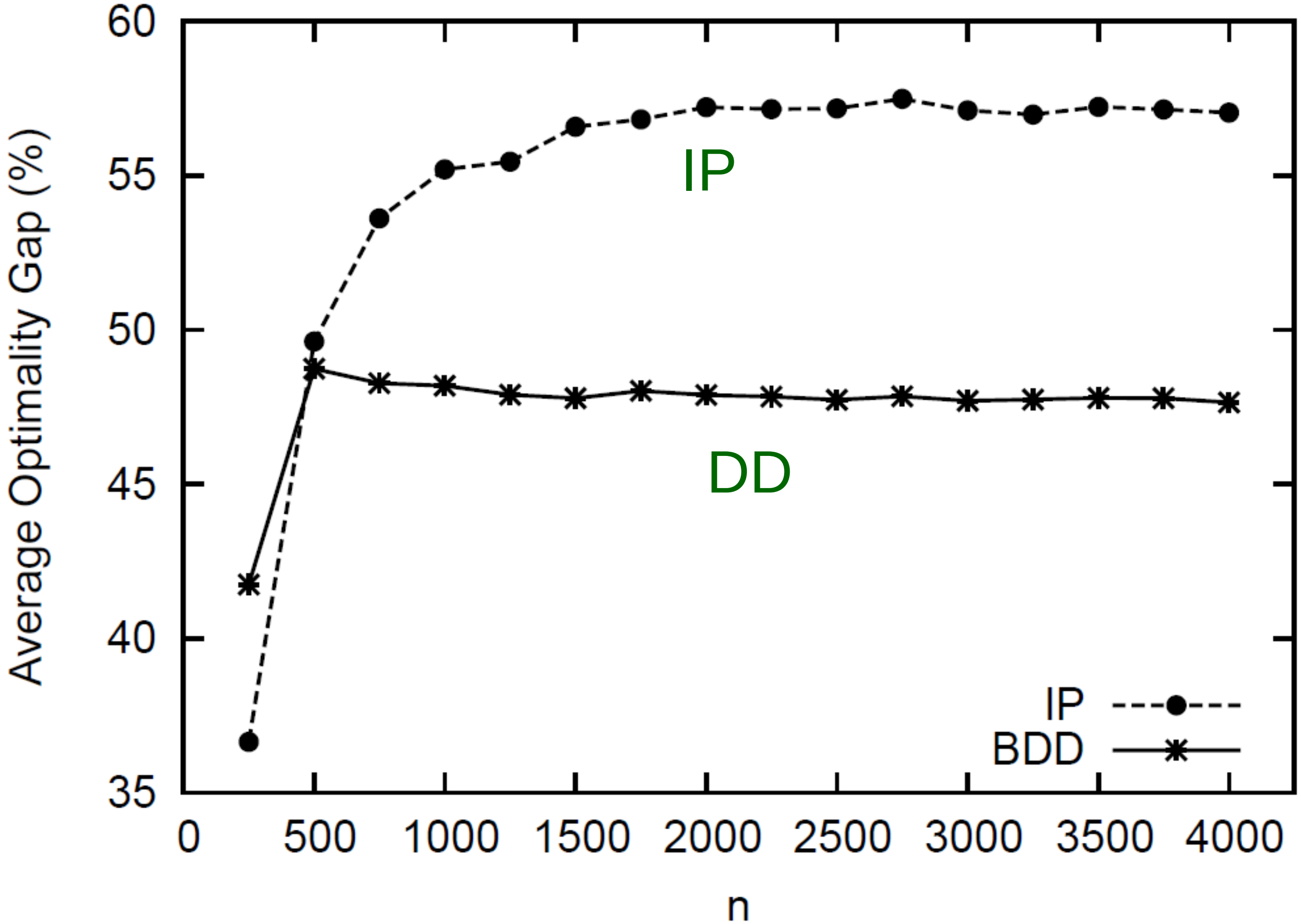
All are minimum covers.

In this case, restricted DD delivers optimal solutions.

41 paths (< 52 feasible solutions)

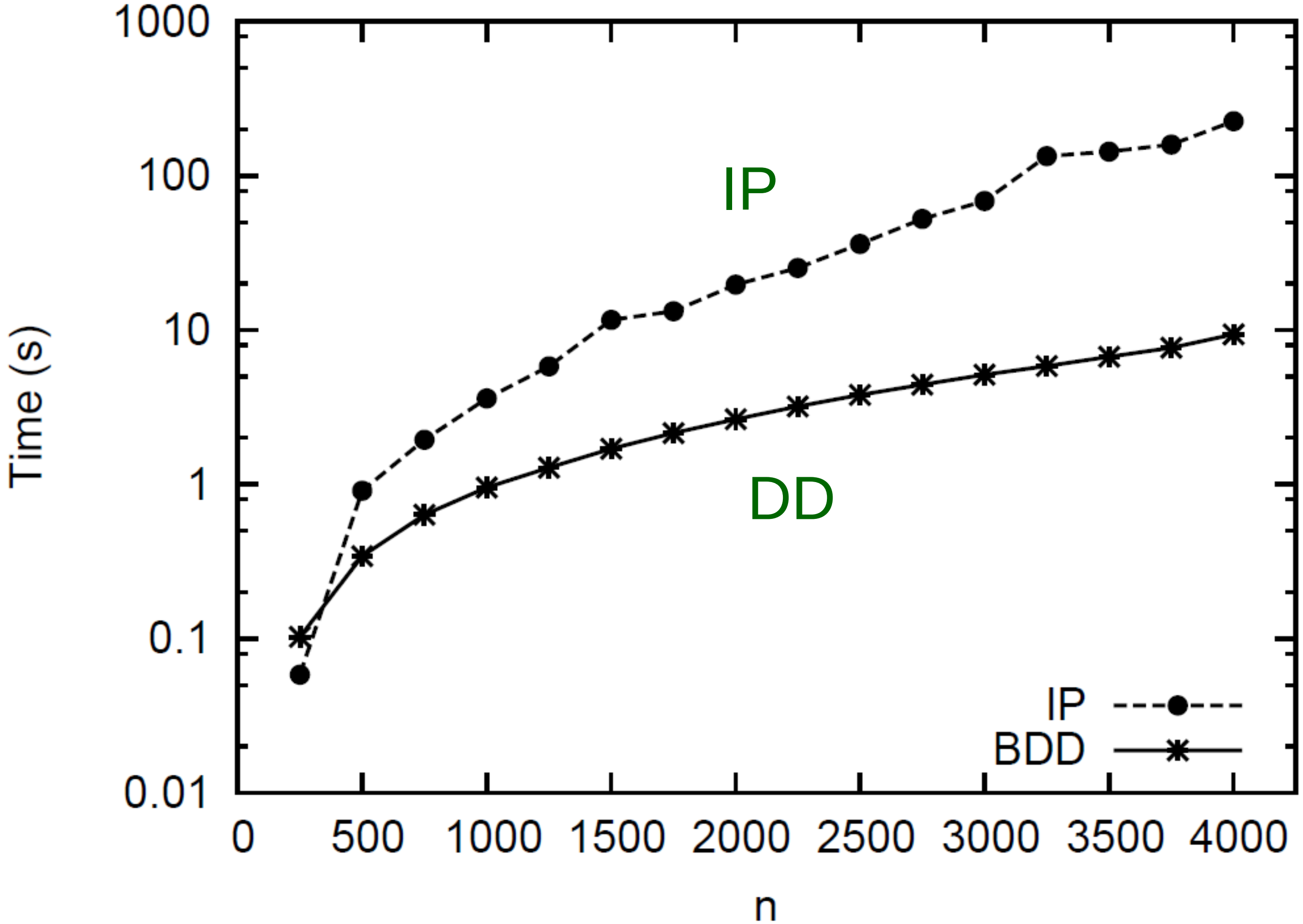
Optimality gap for set covering, *n* variables

Restricted DDs vs
Primal heuristic at root node of CPLEX



Computation time

Restricted DDs vs
Primal heuristic at root node of CPLEX (cuts turned off)



Dynamic Programming Model

- Formulate problem with **dynamic programming** model.
  - Rather than constraint set.
  - Problem must have **recursive** structure
  - But there is great **flexibility** to represent constraints and objective function.
  - Any function of **current state** is permissible.
  - We **don’t care** if state space is **exponential**, because we don’t solve the problem by dynamic programming.

Dynamic Programming Model

- Formulate problem with **dynamic programming** model.
  - Rather than constraint set.
  - Problem must have **recursive** structure
  - But there is great **flexibility** to represent constraints and objective function.
  - Any function of **current state** is permissible.
  - We **don't care** if state space is **exponential**, because we don't solve the problem by dynamic programming.
- State variables are the same as in relaxed DD.
  - Must also specify **state merger** rule.
  - Much as one must **linearize** IP constraints, or perhaps add valid inequalities.

Dynamic Programming Model

- Max stable set problem on a graph.
  - **State** = set of vertices that can be added to stable set.
  - **State merger** = union
- Max cut problem on a graph.
  - **State** = marginal benefit of placing each remaining vertex on left side of cut..
  - **State merger** =
    - Componentwise min if all components ≥ 0 or all ≤ 0; 0 otherwise
    - Adjust incoming arc weights
- Max 2-SAT.
  - Similar to max cut.

Branching Algorithm

- Solve optimization problem using a novel **branch-and-bound** algorithm.
  - Branch on nodes in **last exact layer** of relaxed decision diagram.
    - …rather than branch on variables.
    - Create a new **relaxed DD rooted** at each branching node.
    - Prune search tree using bounds from relaxed DD.

Bergman, Ciré, van Hoeve, JH (2016)

Branching Algorithm

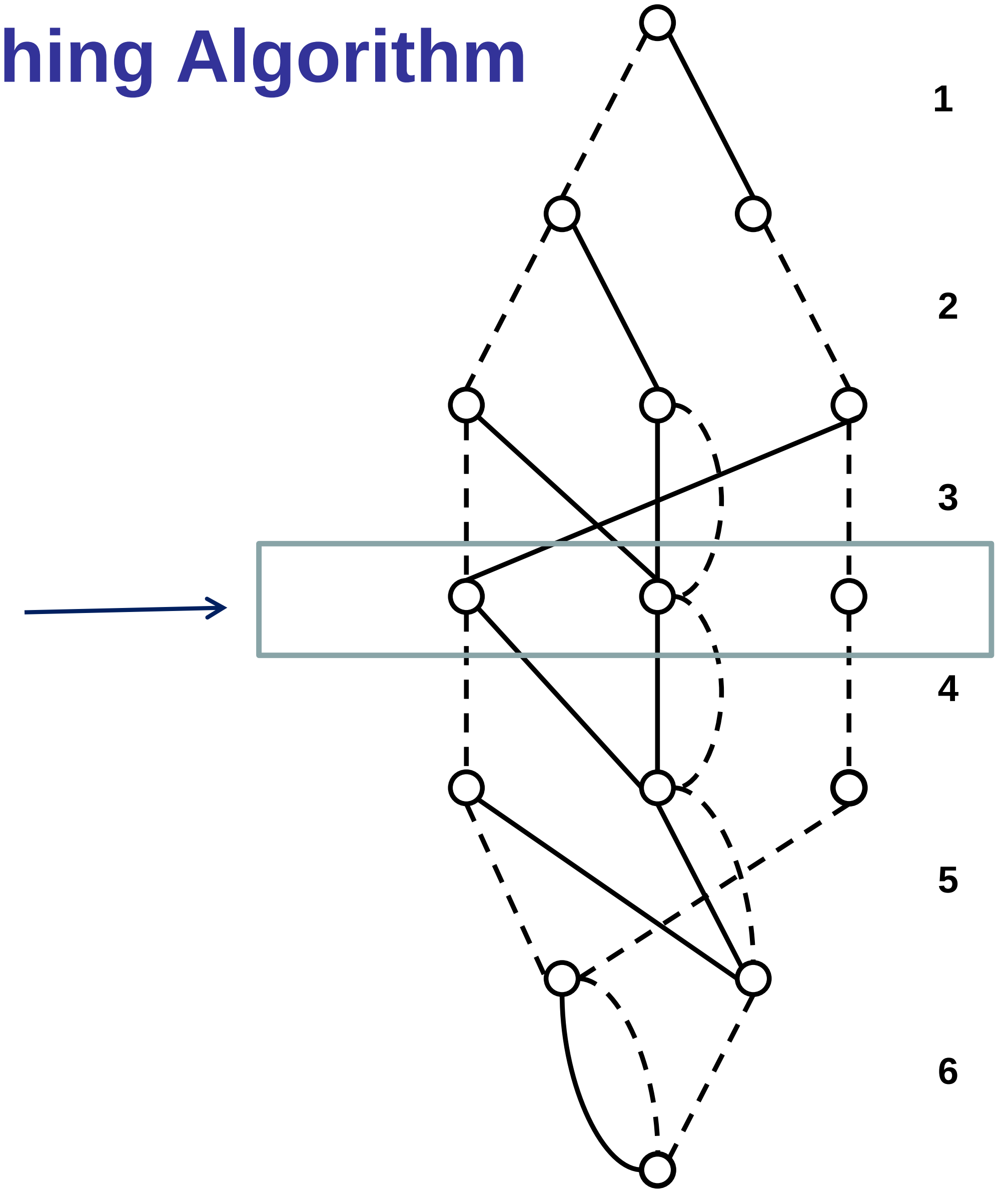
- Solve optimization problem using a novel **branch-and-bound** algorithm.
  - Branch on nodes in **last exact layer** of relaxed decision diagram.
    - …rather than branch on variables.
    - Create a new **relaxed DD rooted** at each branching node.
    - Prune search tree using bounds from relaxed DD.
  - Advantage: a manageable number states may be reachable in first few layers.
    - …even if the state space is **exponential**.
    - Alternative way of dealing with **curse of dimensionality**.

Bergman, Ciré, van Hoeve, JH (2016)

Branching Algorithm

Branching in a relaxed decision diagram

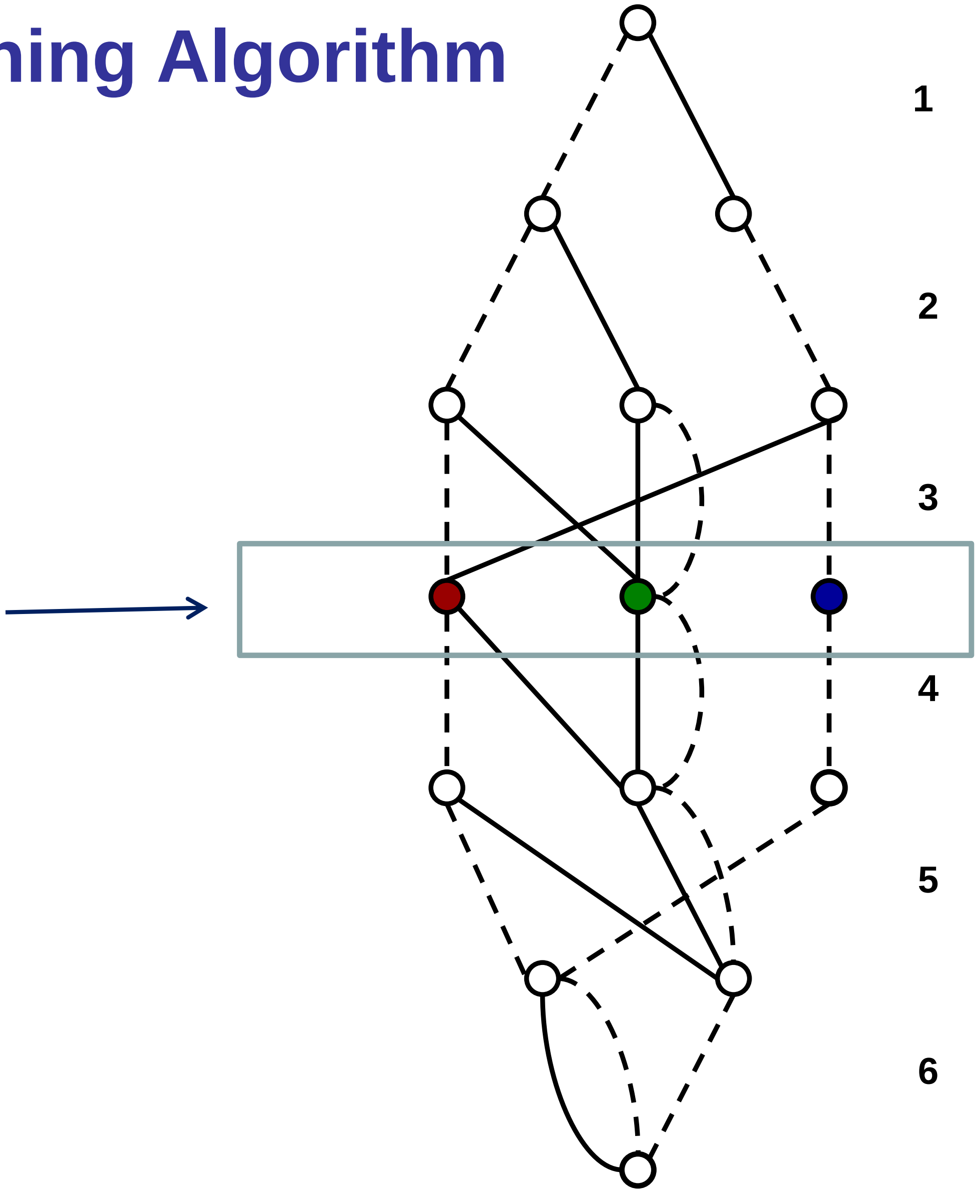
Diagram is exact down to here



Branching Algorithm

Branching in a relaxed decision diagram

Branch on nodes in this layer

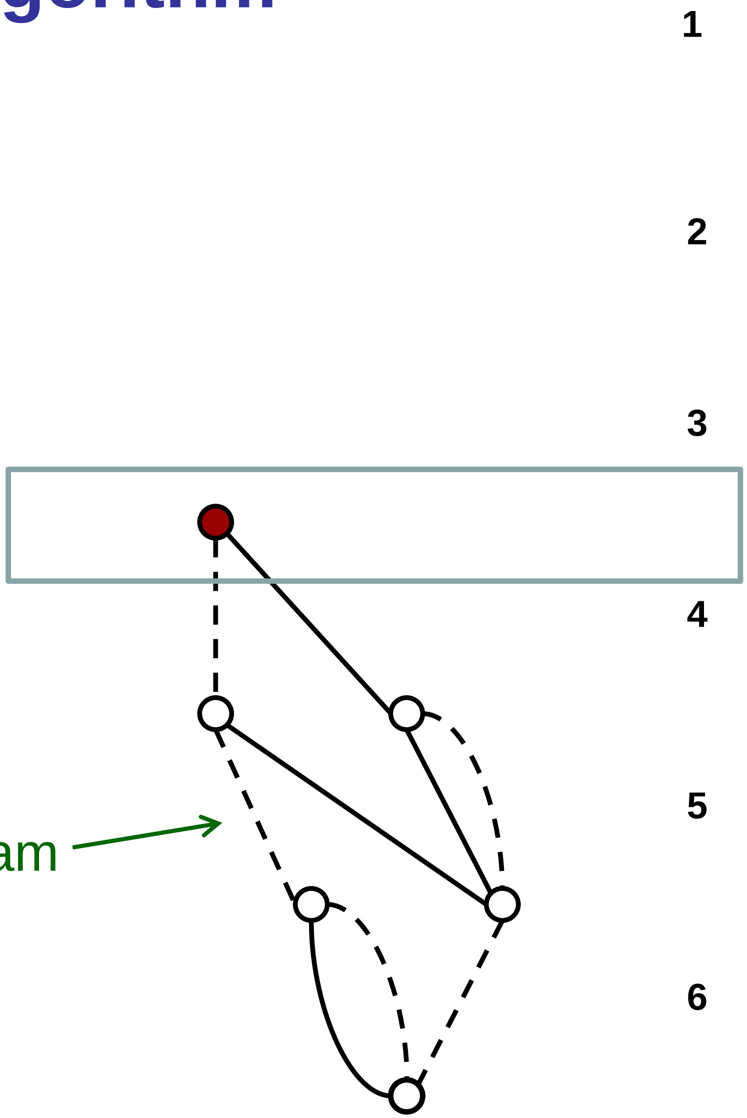


Branching Algorithm

Branching in a relaxed
decision diagram

First branch

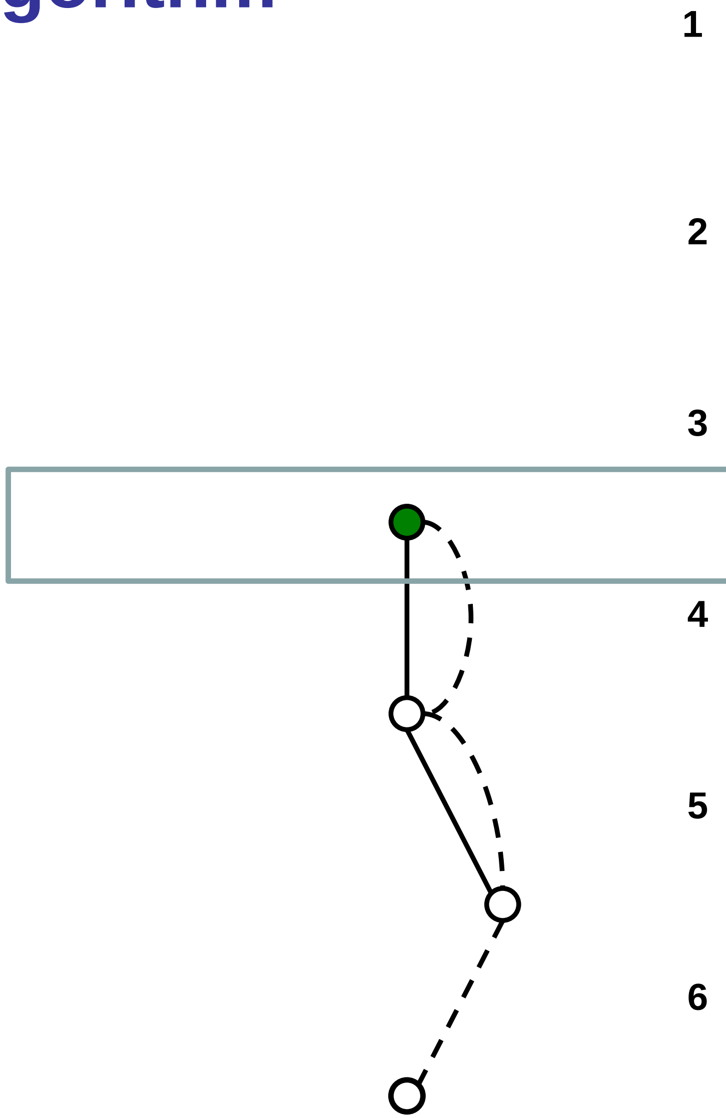
New relaxed decision diagram



Branching Algorithm

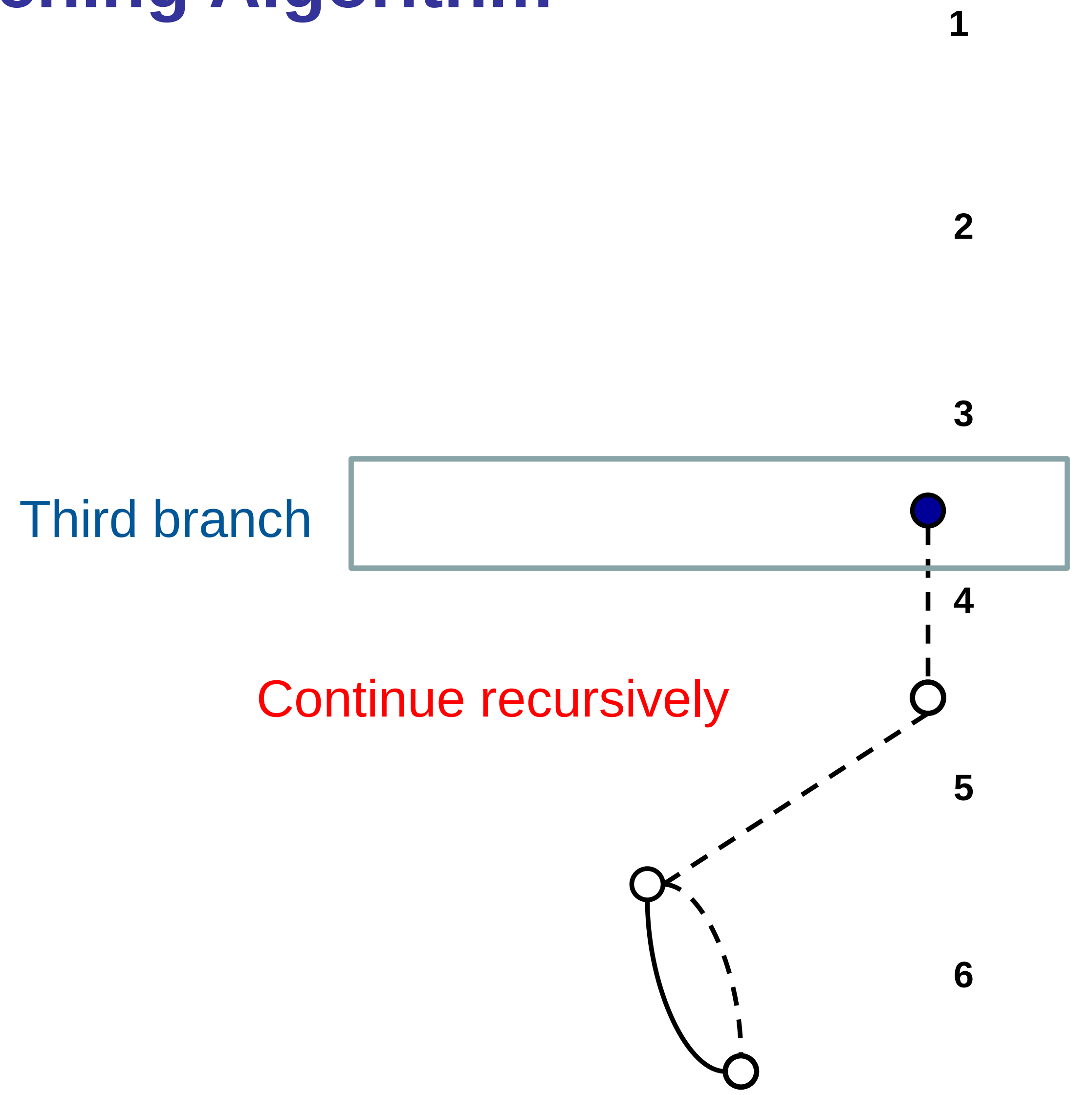
Branching in a relaxed
decision diagram

Second branch



Branching Algorithm

Branching in a relaxed decision diagram



State Space Relaxation?

- This is **very different** from state space relaxation.
  - Problem is **not solved by dynamic programming**.
  - Relaxation created by **merging nodes of DD**
    - …rather than mapping into smaller state space.
  - Relaxation is **constructed dynamically**
    - …as relaxed DD is built.
  - Relaxation uses **same state variables** as exact formulation
    - …which allows branching in relaxed DD

Christofides, Mingozzi, Toth (1981)

Computational performance

- Computational results…
  - Applied to stable set, max cut, max 2-SAT.
    - Superior to commercial MIP solver (CPLEX) on most instances.
    - Obtained best known solution on some max cut instances.
  - Slightly slower than MIP on stable set with precomputed clique cover model, but…

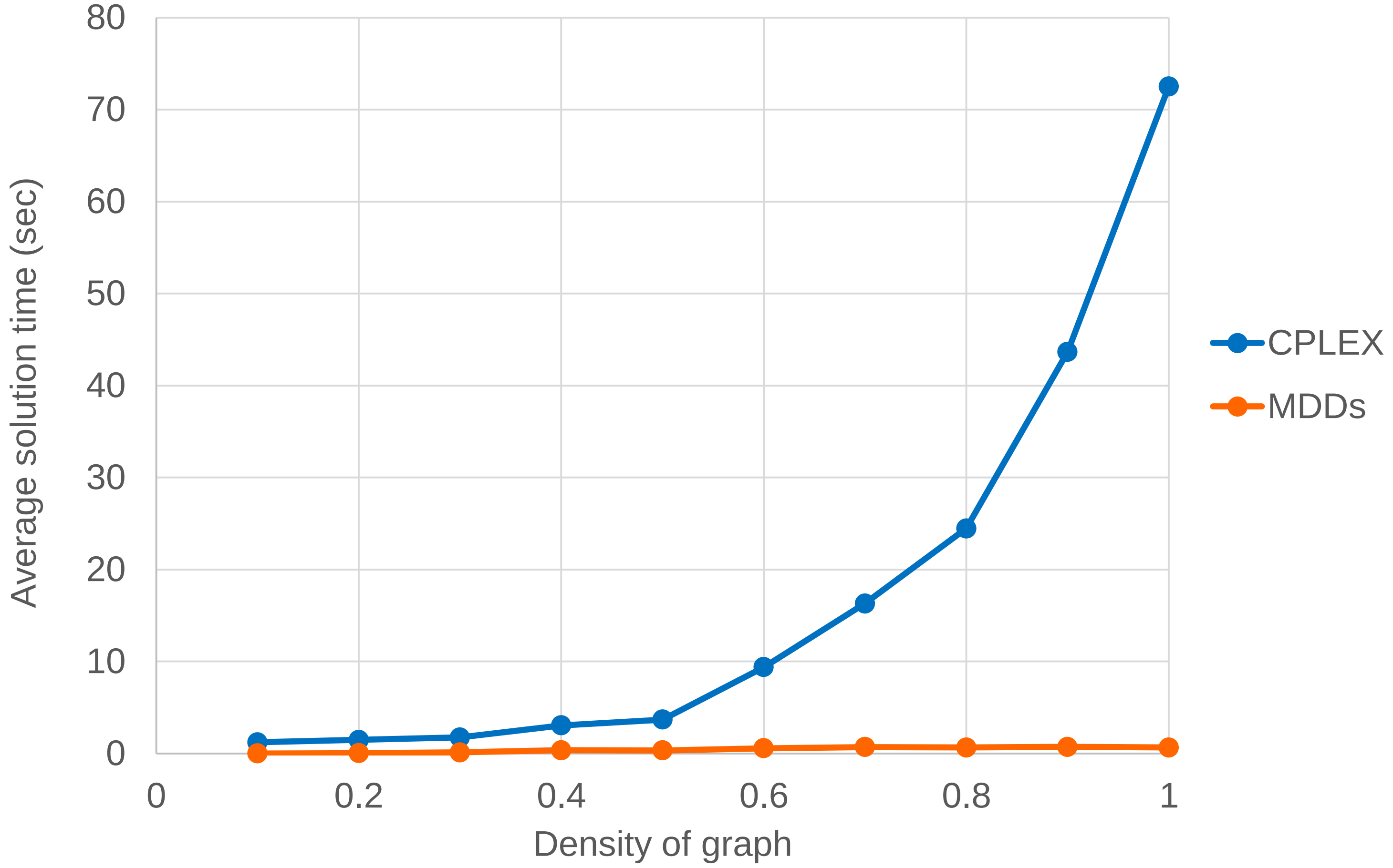
Bergman, Ciré, van Hoeve, JH (2016)

Computational performance

Max cut
on a graph

Avg. solution time
vs
graph density

30 vertices

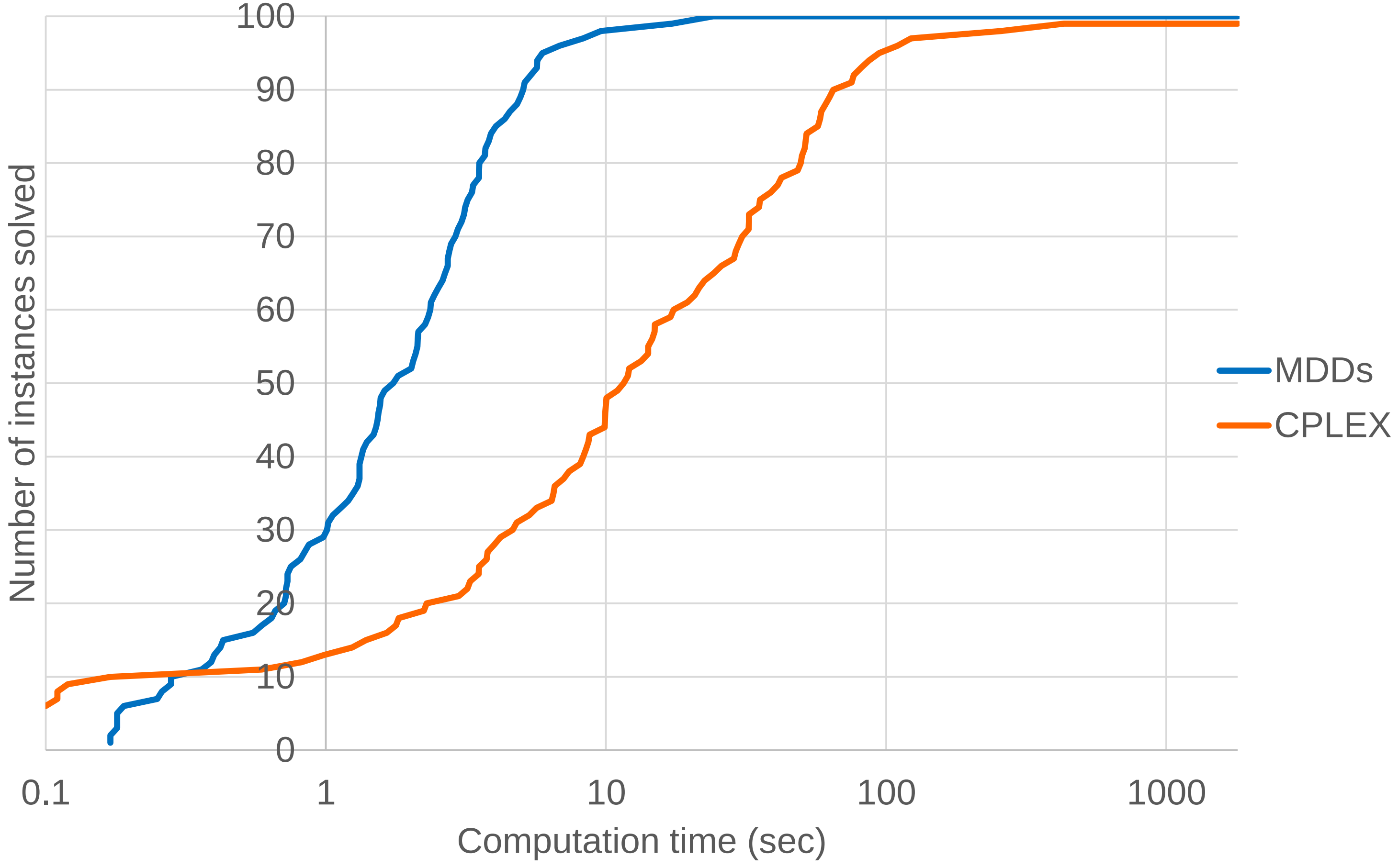


Computational performance

Max 2-SAT

Performance profile

30 variables

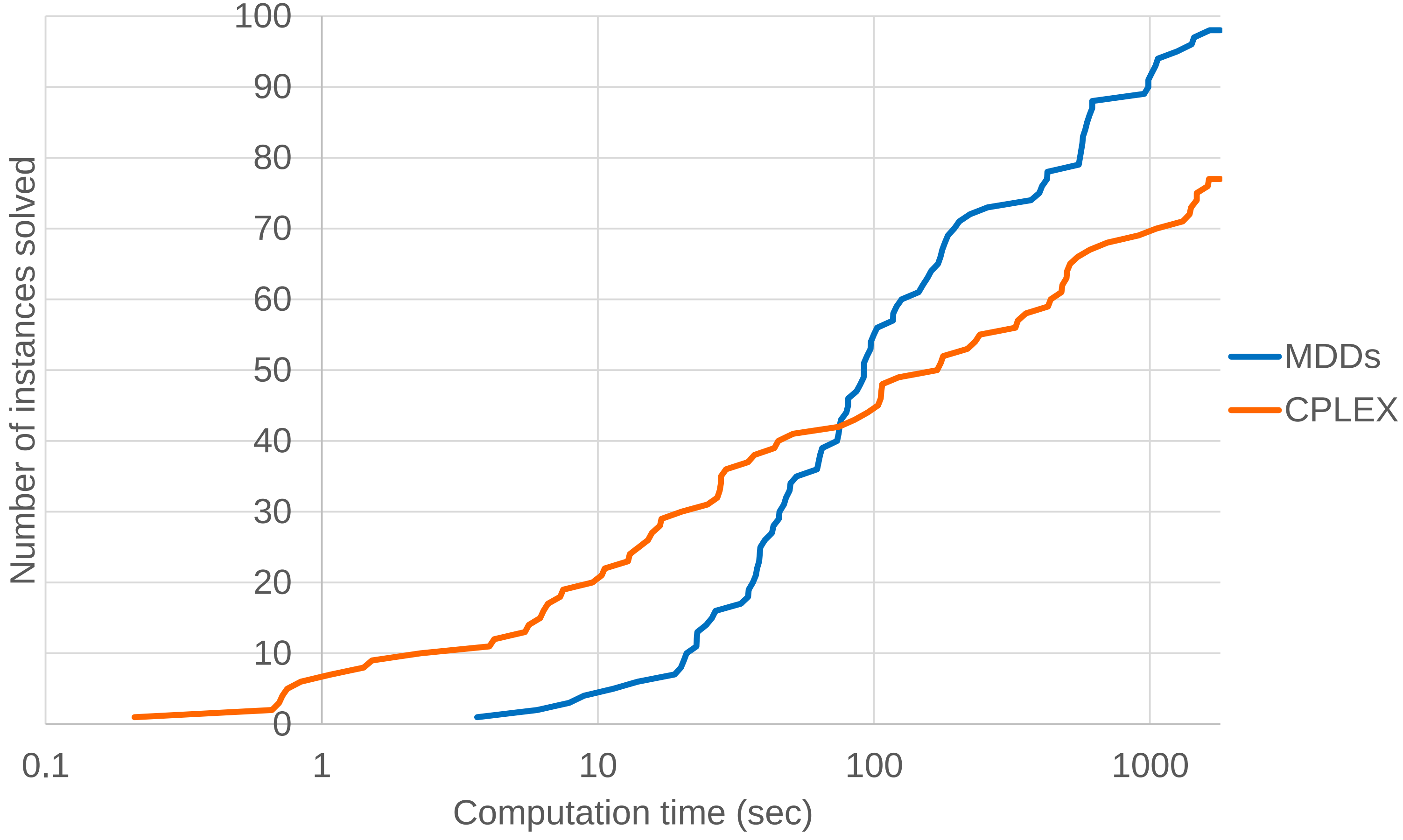


Computational performance

Max 2-SAT

Performance profile

40 variables



Computational performance

- Potential to scale up
  - No need to load large inequality model into solver.
  - **Parallelizes** very effectively
    - **Near-linear** speedup.
    - Much better than mixed integer programming.

Computational performance

- In all computational comparisons so far…
  - Problem is **easily formulated for IP**.
- DD-based optimization is most competitive when…
  - Problem has a recursive dynamic programming model…
  - and **no convenient IP model**.
- Such as…
  - Sequencing and scheduling problems (next talk)
  - DP problems with exponential state space
    - New approach to “curse of dimensionality”
  - Problems with nonconvex, nonseparable objective function…

Modeling the Objective Function

- Weighted DD can represent **any** objective function
  - Separable functions are the easiest, but any nonseparable function is possible.
  - Can be nonlinear, nonconvex, etc.
  - The issue is complexity of resulting DD

Modeling the Objective Function

- Weighted DD can represent **any** objective function
  - Separable functions are the easiest, but any nonseparable function is possible.
  - Can be nonlinear, nonconvex, etc.
  - The issue is complexity of resulting DD
- Multiple encodings
  - A given objective function can be encoded by **multiple** assignments of costs to arcs.
  - There is a **unique canonical** arc cost assignment.
    - Which can **reduce size** of exact DD.
  - Design state variables accordingly

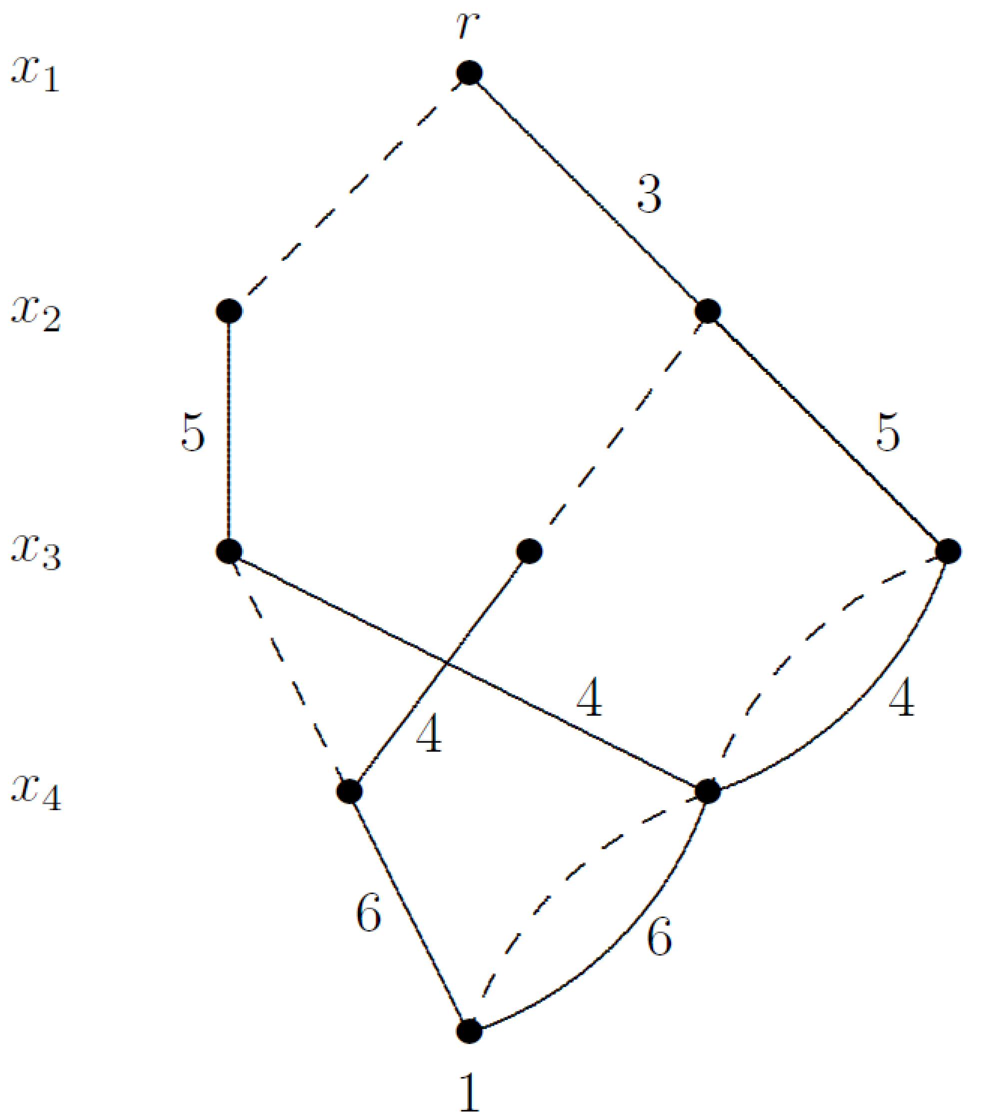
Modeling the Objective Function

Set covering with separable cost function

Easy. Just label arcs with weights.

	Set i			
	1	2	3	4
A	•	•		
B	•		•	•
C		•	•	
D		•		•
Weight	3	5	4	6

$x_i = 1$ when we select set i

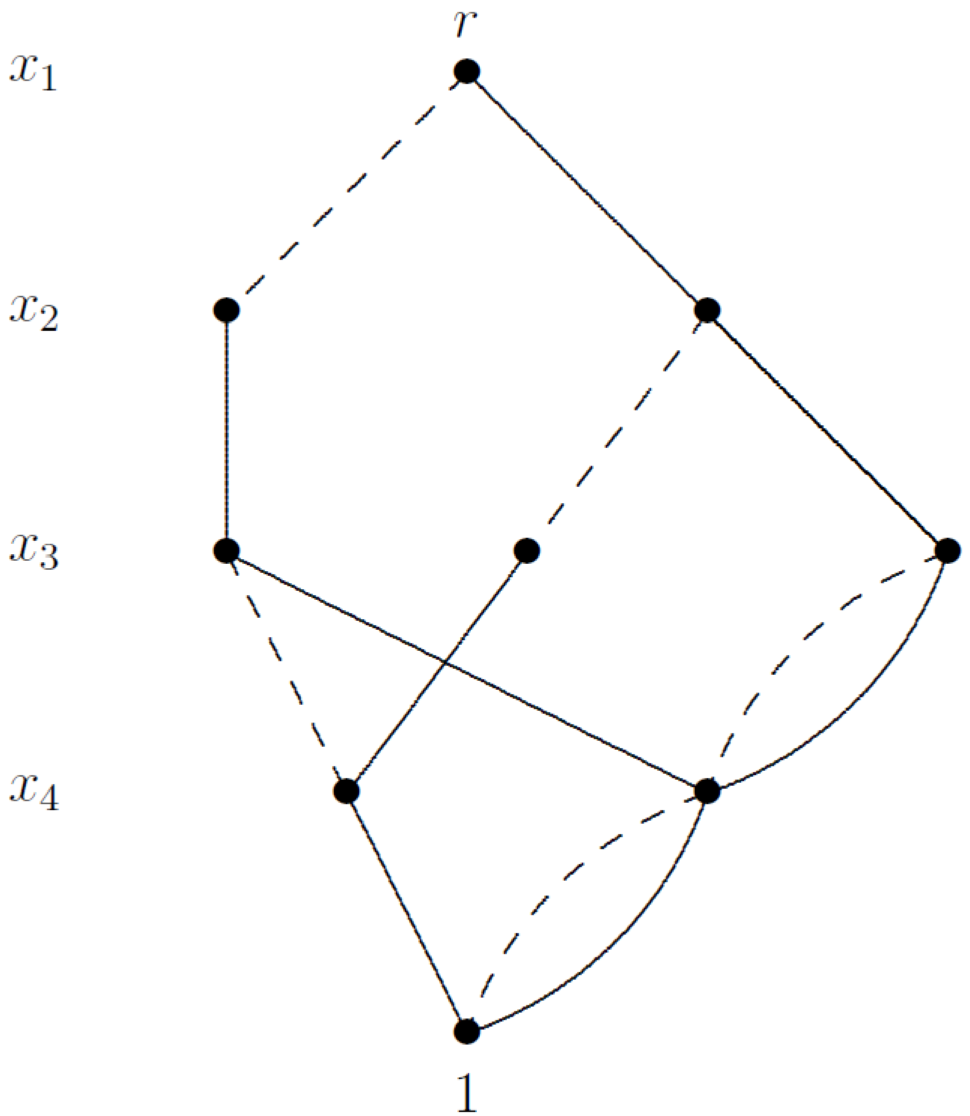


Modeling the Objective Function

Nonseparable cost function

Now what?

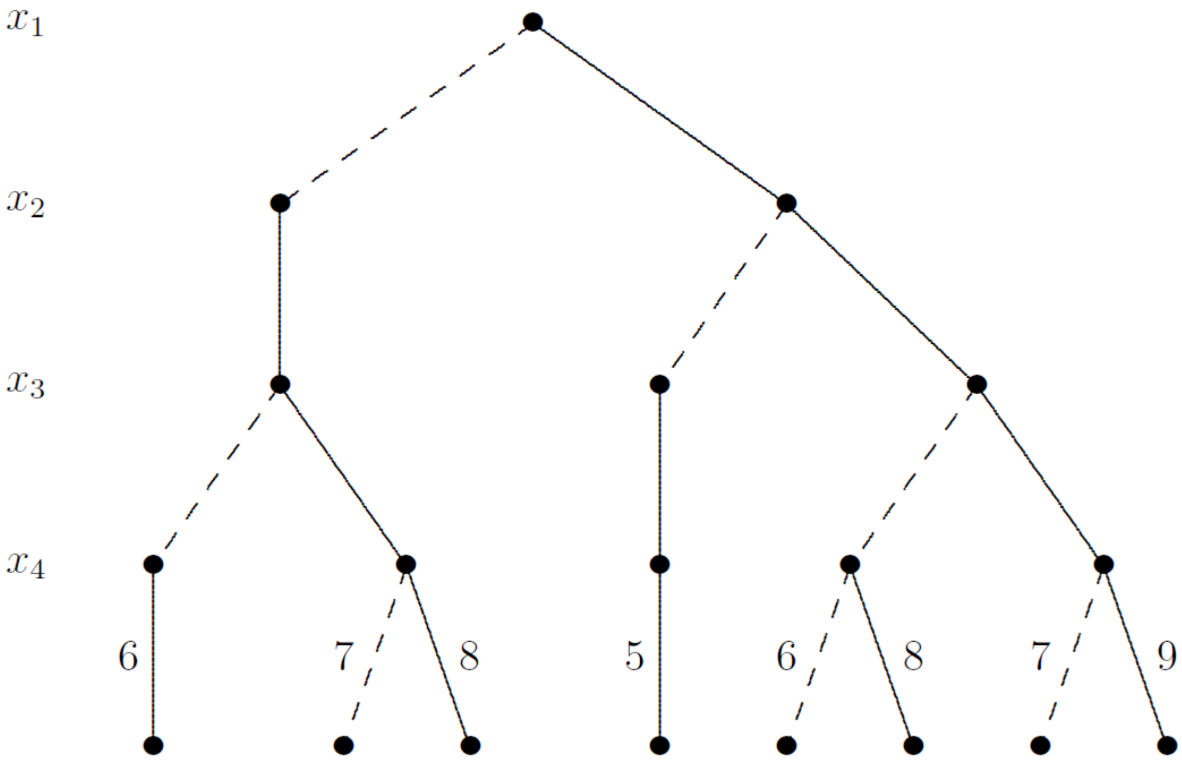
x	$f(x)$
(0,1,0,1)	6
(0,1,1,0)	7
(0,1,1,1)	8
(1,0,1,1)	5
(1,1,0,0)	6
(1,1,0,1)	8
(1,1,1,0)	7
(1,1,1,1)	9



Modeling the Objective Function

Nonseparable cost function

Put costs on leaves of branching tree.



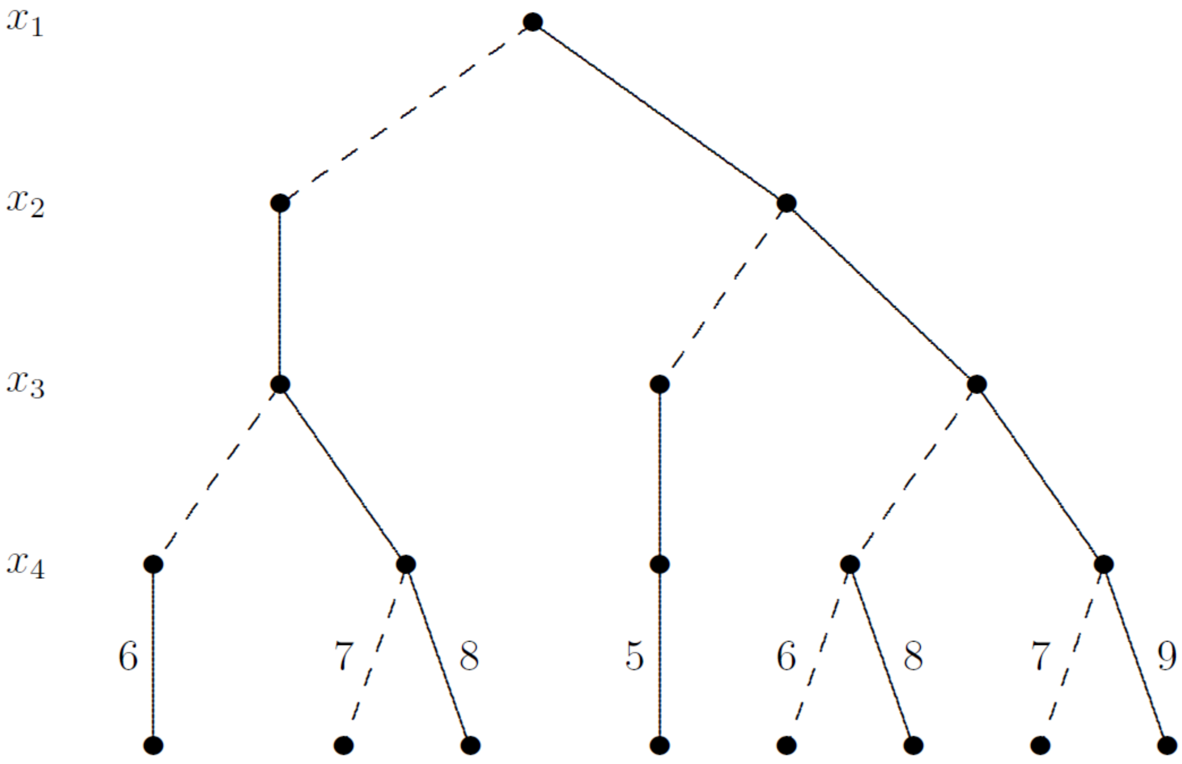
x	$f(x)$
(0,1,0,1)	6
(0,1,1,0)	7
(0,1,1,1)	8
(1,0,1,1)	5
(1,1,0,0)	6
(1,1,0,1)	8
(1,1,1,0)	7
(1,1,1,1)	9

Modeling the Objective Function

Nonseparable cost function

Put costs on leaves of branching tree.

But now we can't reduce the tree to an efficient decision diagram.



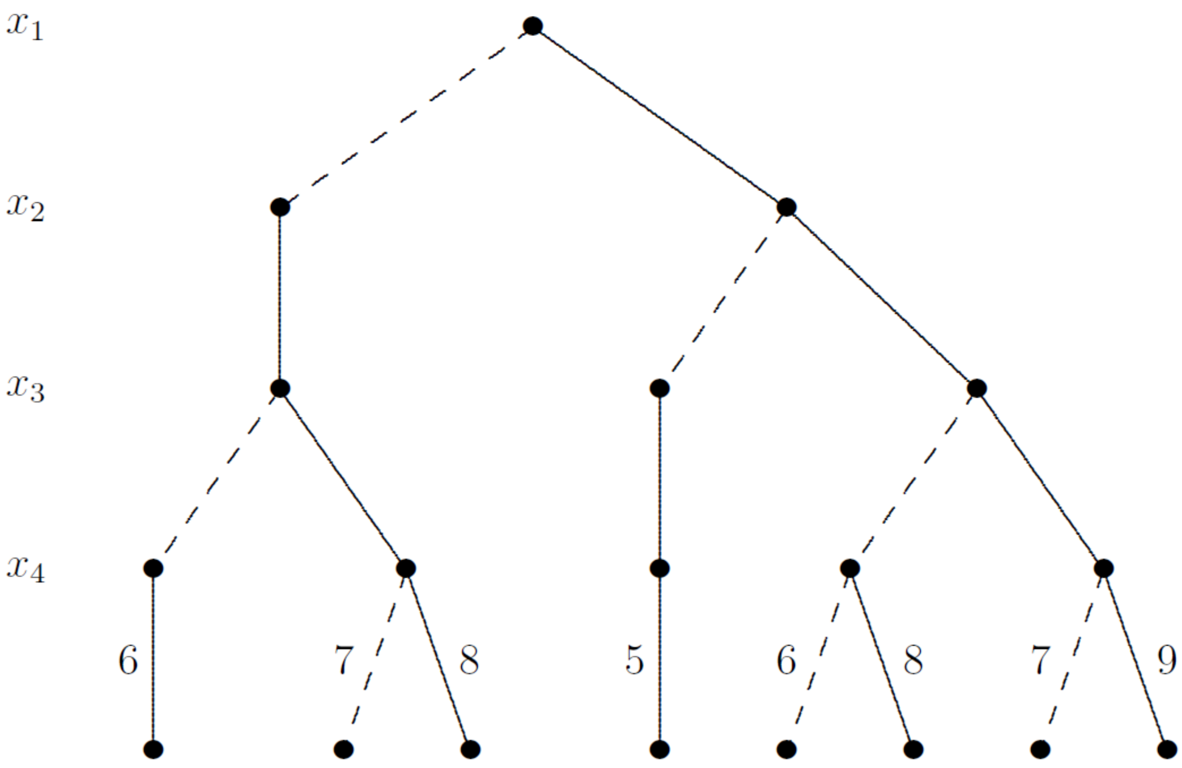
Modeling the Objective Function

Nonseparable cost function

Put costs on leaves of branching tree.

But now we can't reduce the tree to an efficient decision diagram.

We will rearrange costs to obtain **canonical costs**.



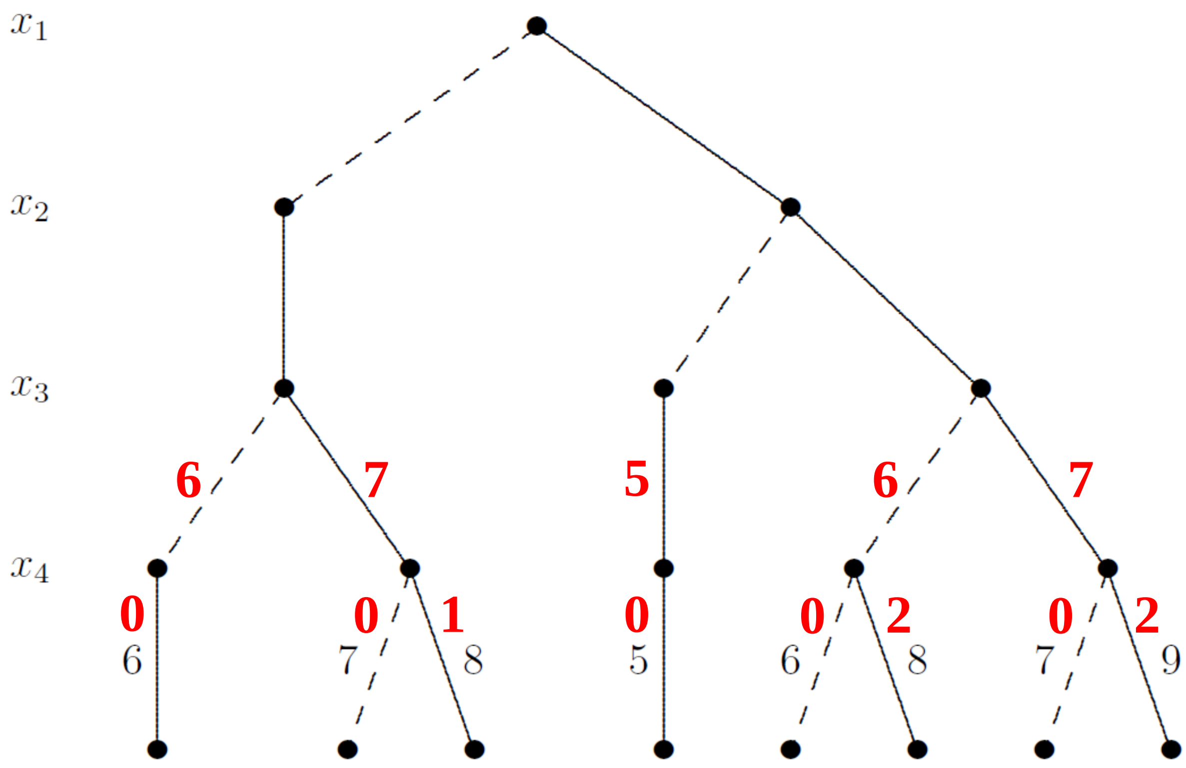
Modeling the Objective Function

Nonseparable cost function

Put costs on leaves of branching tree.

But now we can't reduce the tree to an efficient decision diagram.

We will rearrange costs to obtain **canonical costs**.



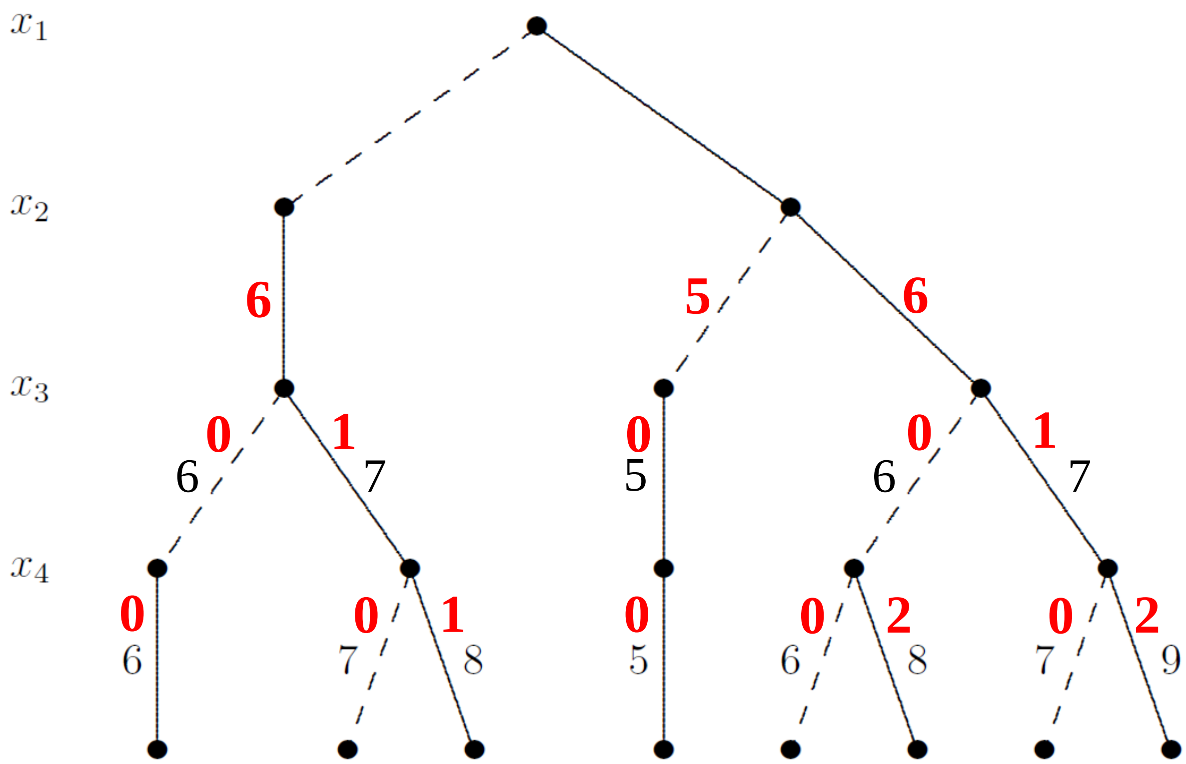
Modeling the Objective Function

Nonseparable cost function

Put costs on leaves of branching tree.

But now we can't reduce the tree to an efficient decision diagram.

We will rearrange costs to obtain **canonical costs**.



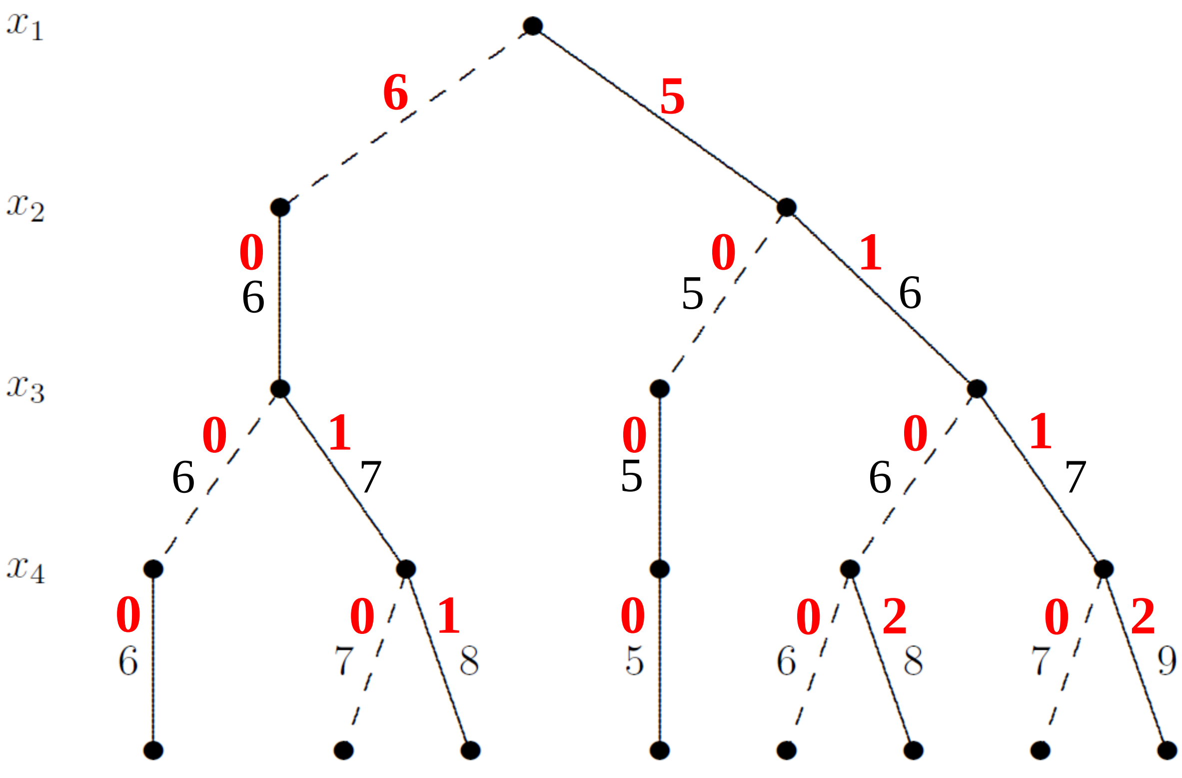
Modeling the Objective Function

Nonseparable cost function

Put costs on leaves of branching tree.

But now we can't reduce the tree to an efficient decision diagram.

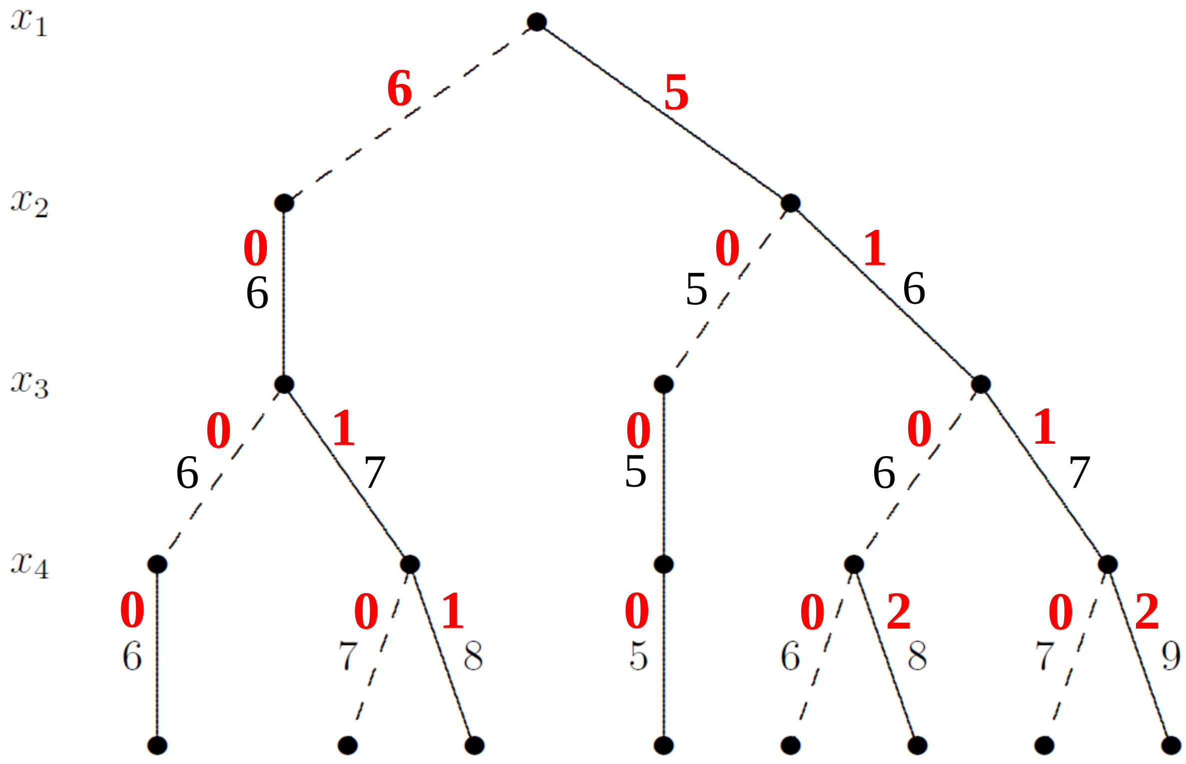
We will rearrange costs to obtain **canonical costs**.



Modeling the Objective Function

Nonseparable cost function

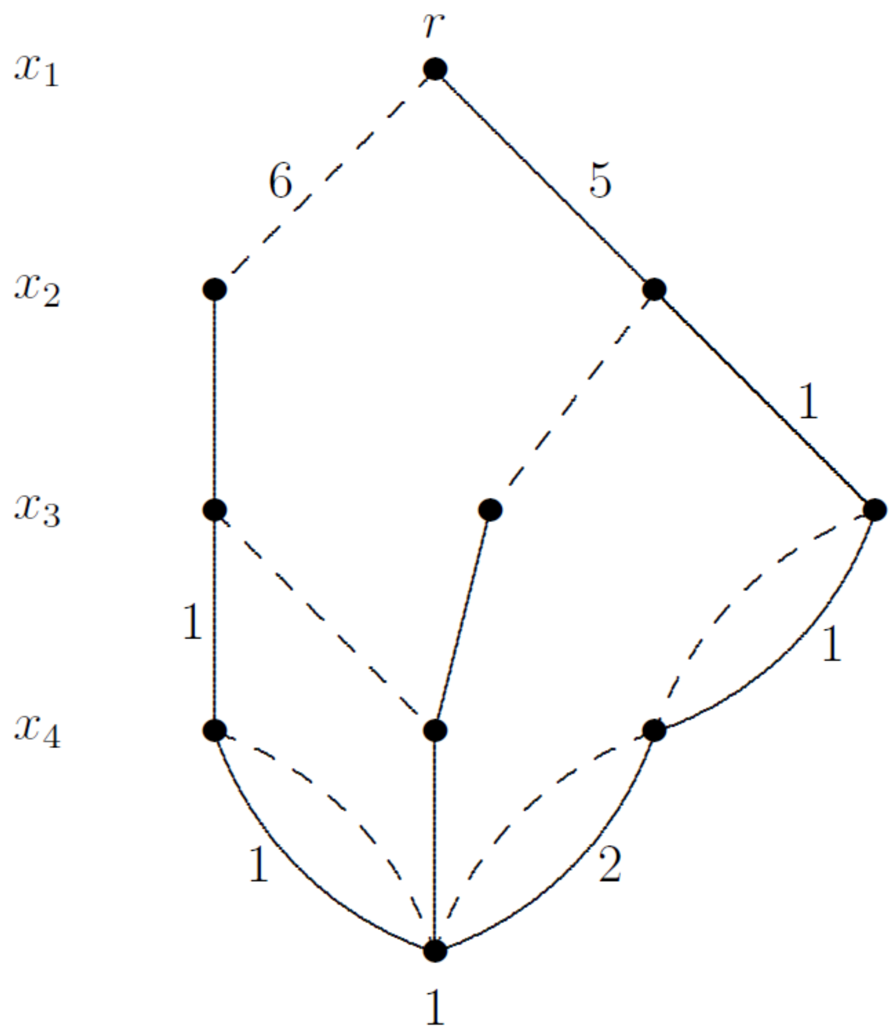
Now the tree can be reduced.



Modeling the Objective Function

Nonseparable cost function

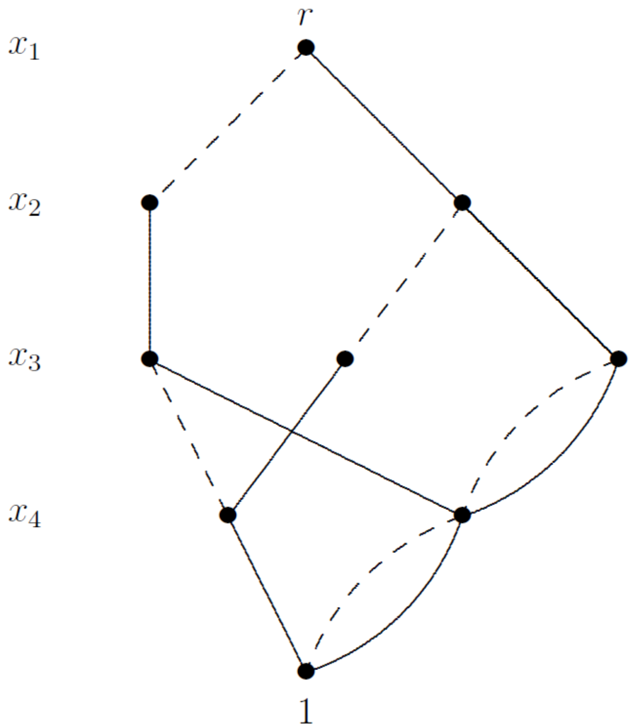
Now the tree can be reduced.

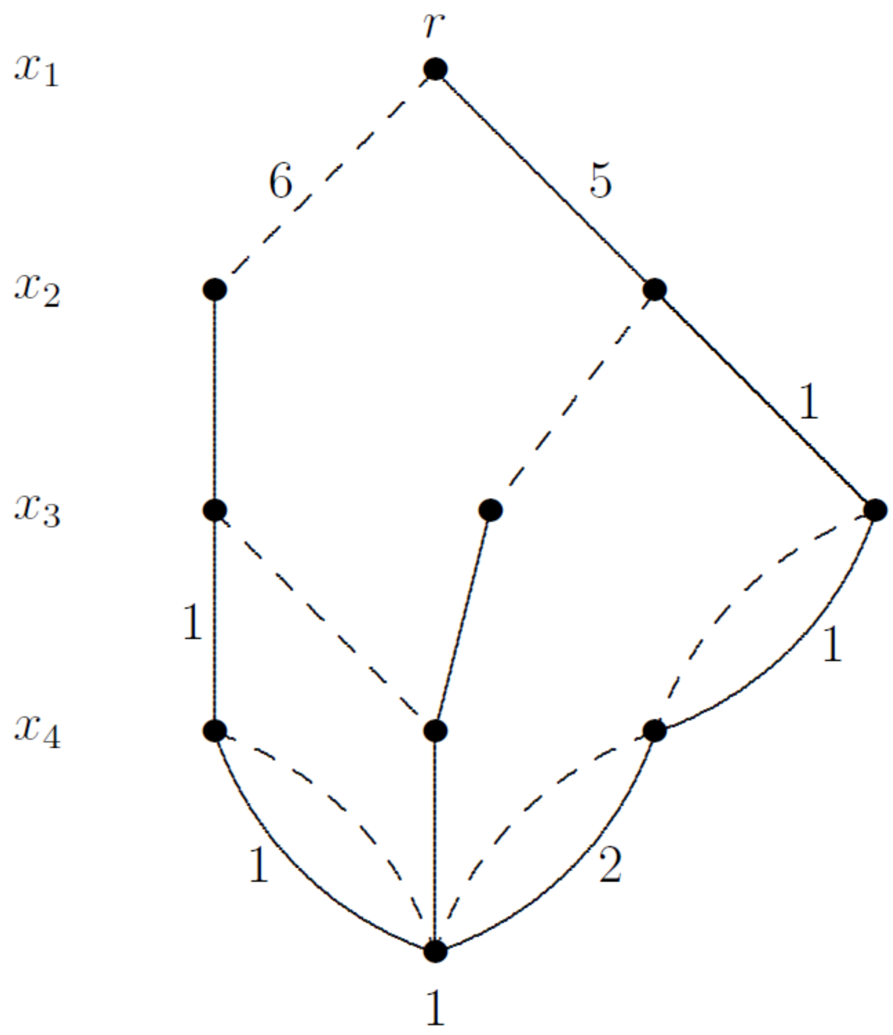


Modeling the Objective Function

Nonseparable cost function

DD is larger than
reduced unweighted DD,
but still compact.





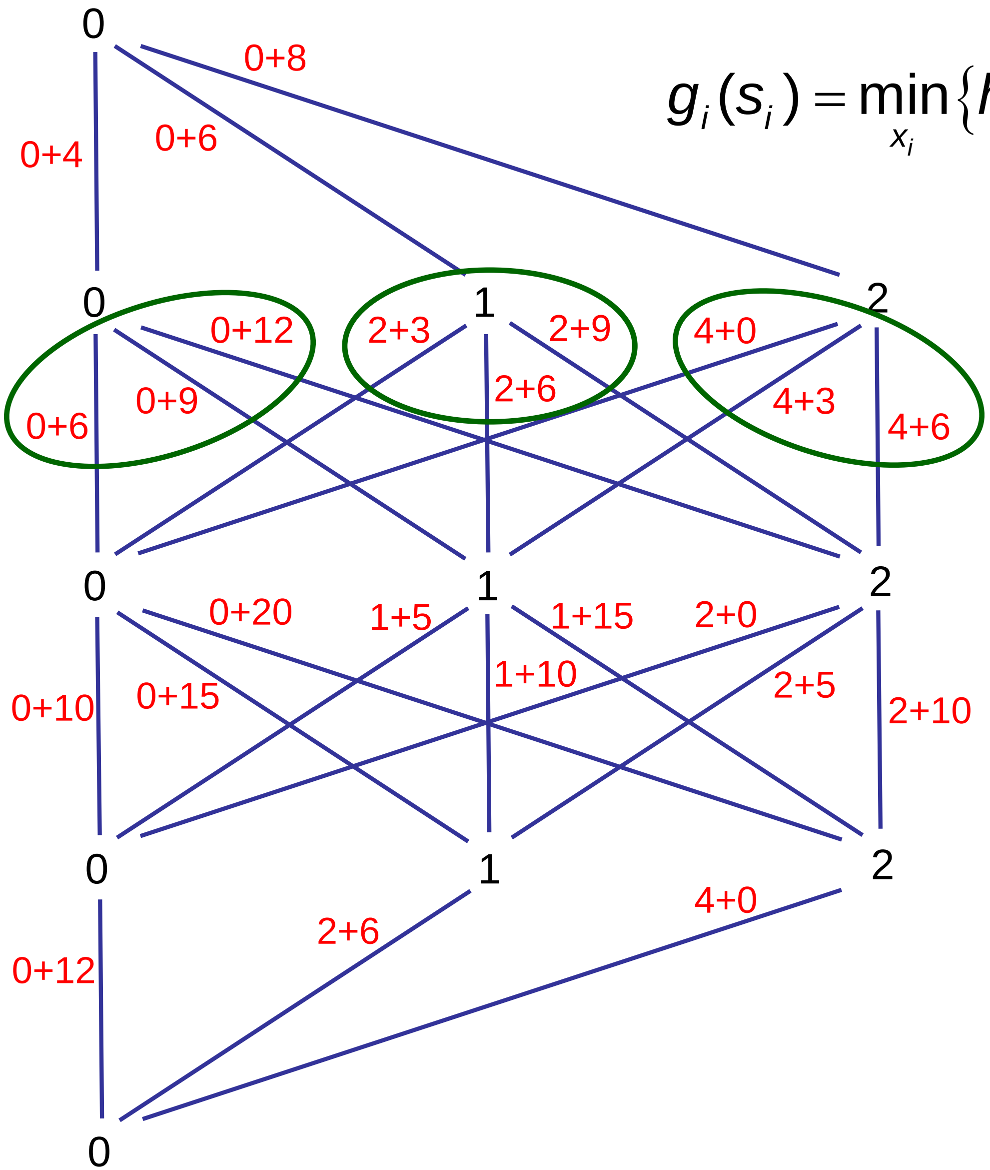
Modeling the Objective Function

Theorem. For a given variable ordering, a given objective function is represented by a **unique** weighted decision diagram with canonical costs.

Inventory Management Example

- In each period *i*, we have:
  - Demand d_i
  - Unit production cost c_i
  - Warehouse space *m*
  - Unit holding cost h_i
- In each period, we decide:
  - Production level x_i
  - Stock level s_i
- Objective:
  - Meet demand each period while minimizing production and holding costs.

Reducing the Transition Graph

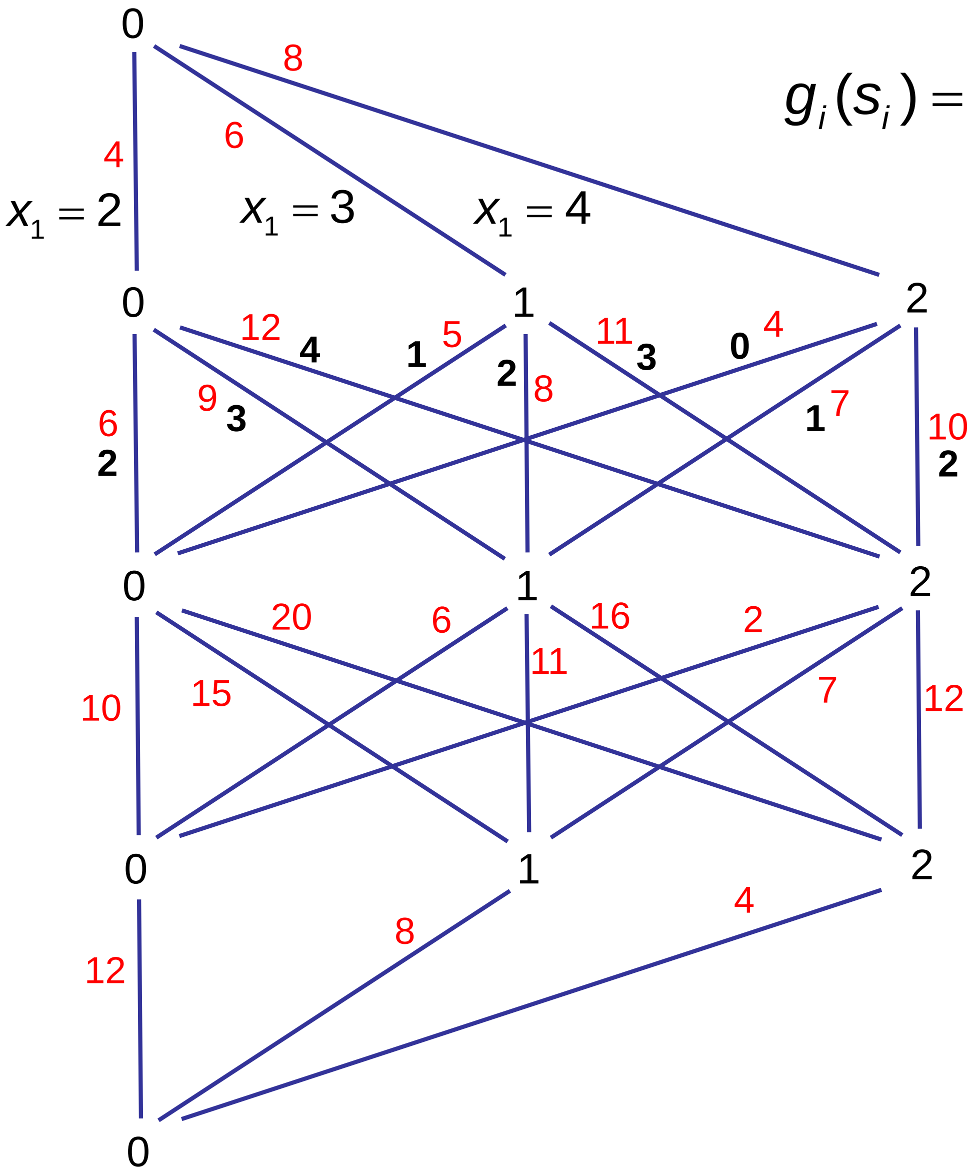


$$g_i(s_i) = \min_{x_i}\{h_i s_i + c_i x_i + g_{i+1}(s_i + x_i - d_i)\}$$

Arcs leaving each node are very similar.

- Transition to the same states.
- Have the same costs, up to an offset.

Inventory Problem



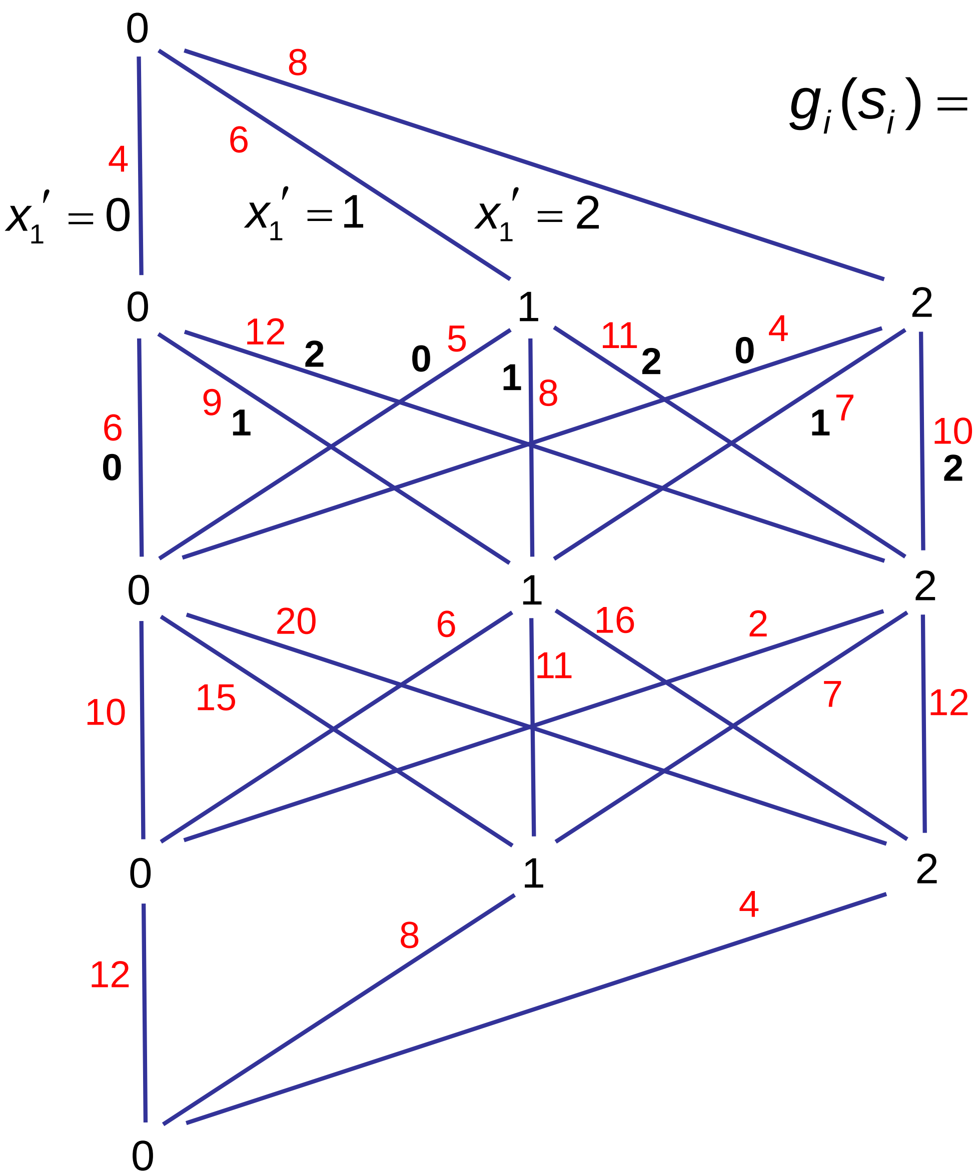
$$g_i(s_i) = \min_{x_i}\{h_i s_i + c_i x_i + g_{i+1}(s_i + x_i - d_i)\}$$

To equalize controls, let

$$x_i' = s_i + x_i - d_i$$

Be the stock level in next period.

Inventory Problem



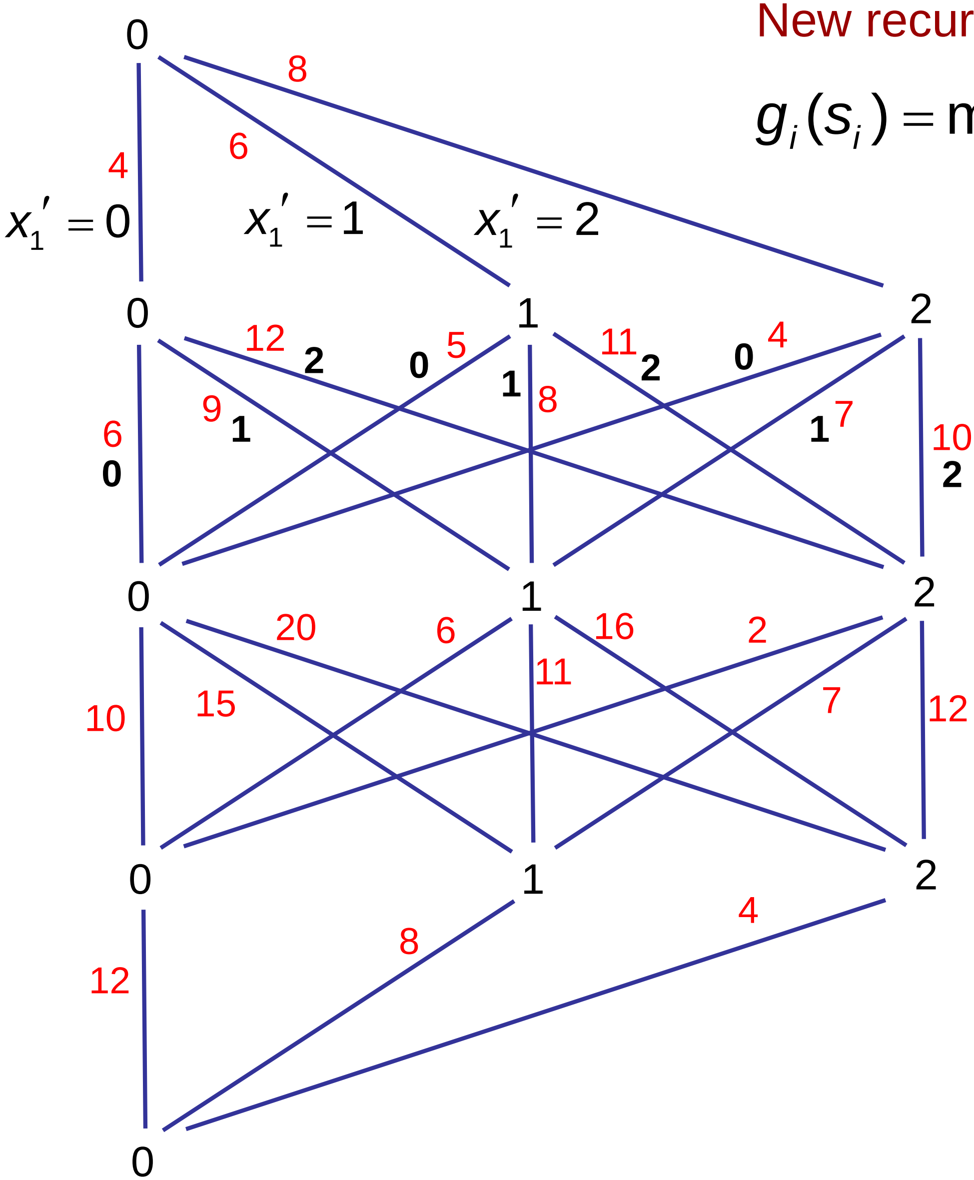
$$g_i(s_i) = \min_{x_i}\{h_i s_i + c_i x_i + g_{i+1}(s_i + x_i - d_i)\}$$

To equalize controls, let

$$x_i{}' = s_i + x_i - d_i$$

Be the stock level in next period.

Inventory Problem



New recursion:

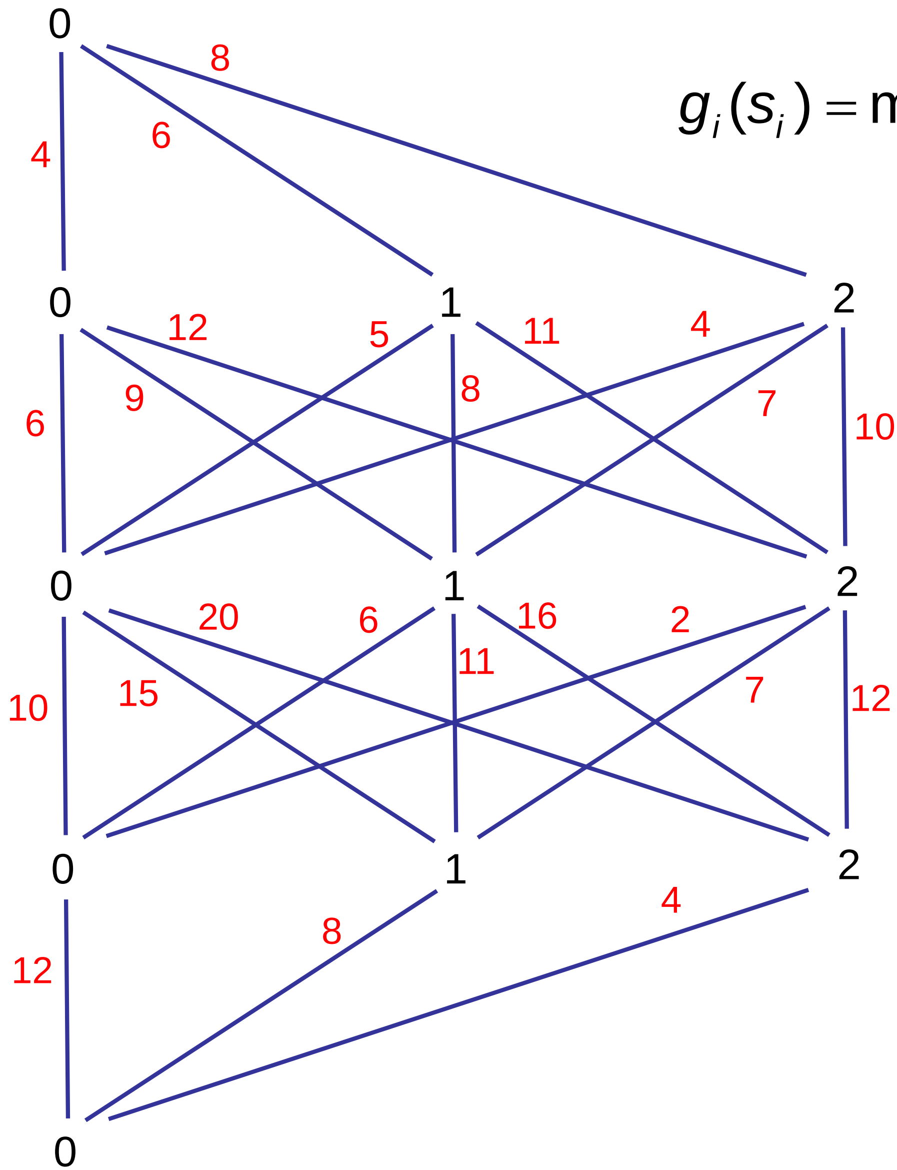
$$g_i(s_i) = \min_{x_i'} \left\{ h_i s_i + c_i(x_i' - s_i + d_i) + g_{i+1}(x_i') \right\}$$

To equalize controls, let

$$x_i' = s_i + x_i - d_i$$

Be the stock level in next period.

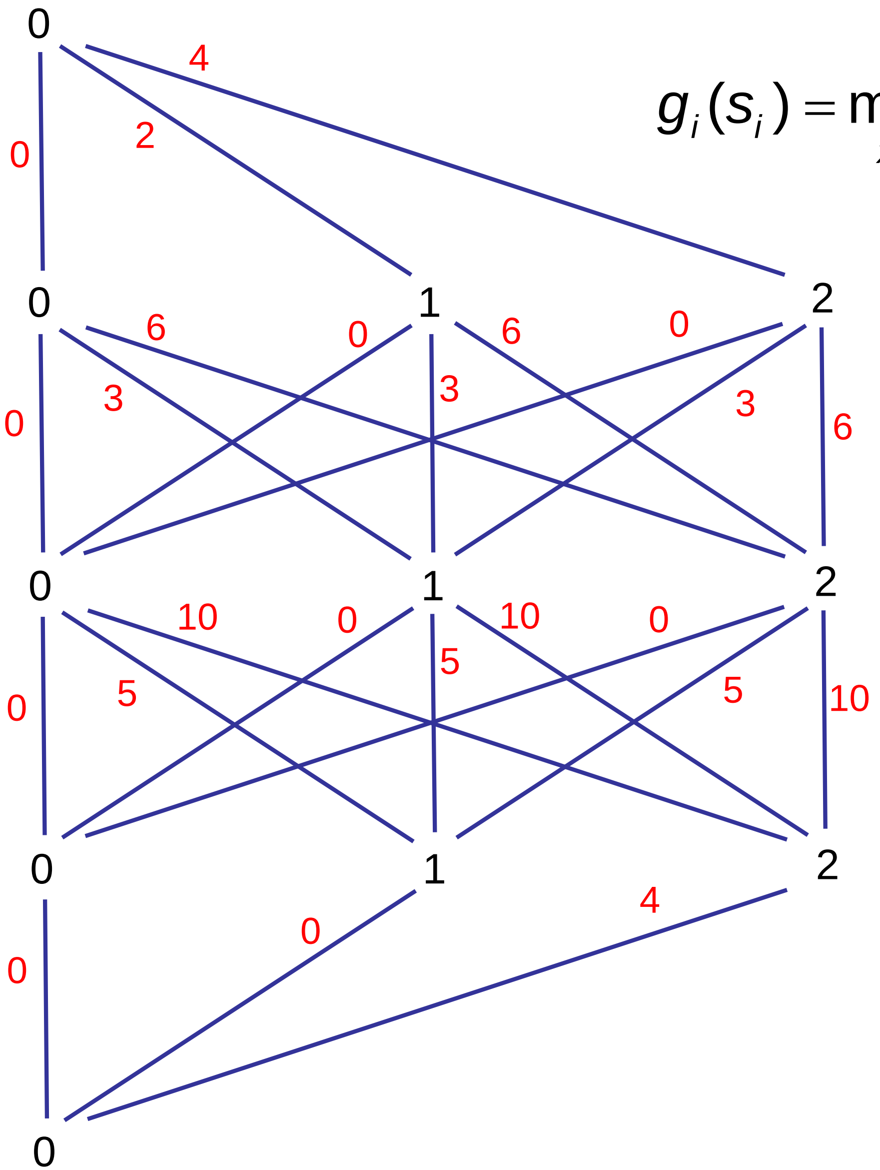
Inventory Problem



$$g_i(s_i) = \min_{x_i'} \left\{ h_i s_i + c_i(x_i' - s_i + d_i) + g_{i+1}(x_i') \right\}$$

To obtain canonical costs, subtract $c_i(m - s_i) + h_i s_i$ from cost on each arc (s_i, s_{i+1}).

Inventory Problem

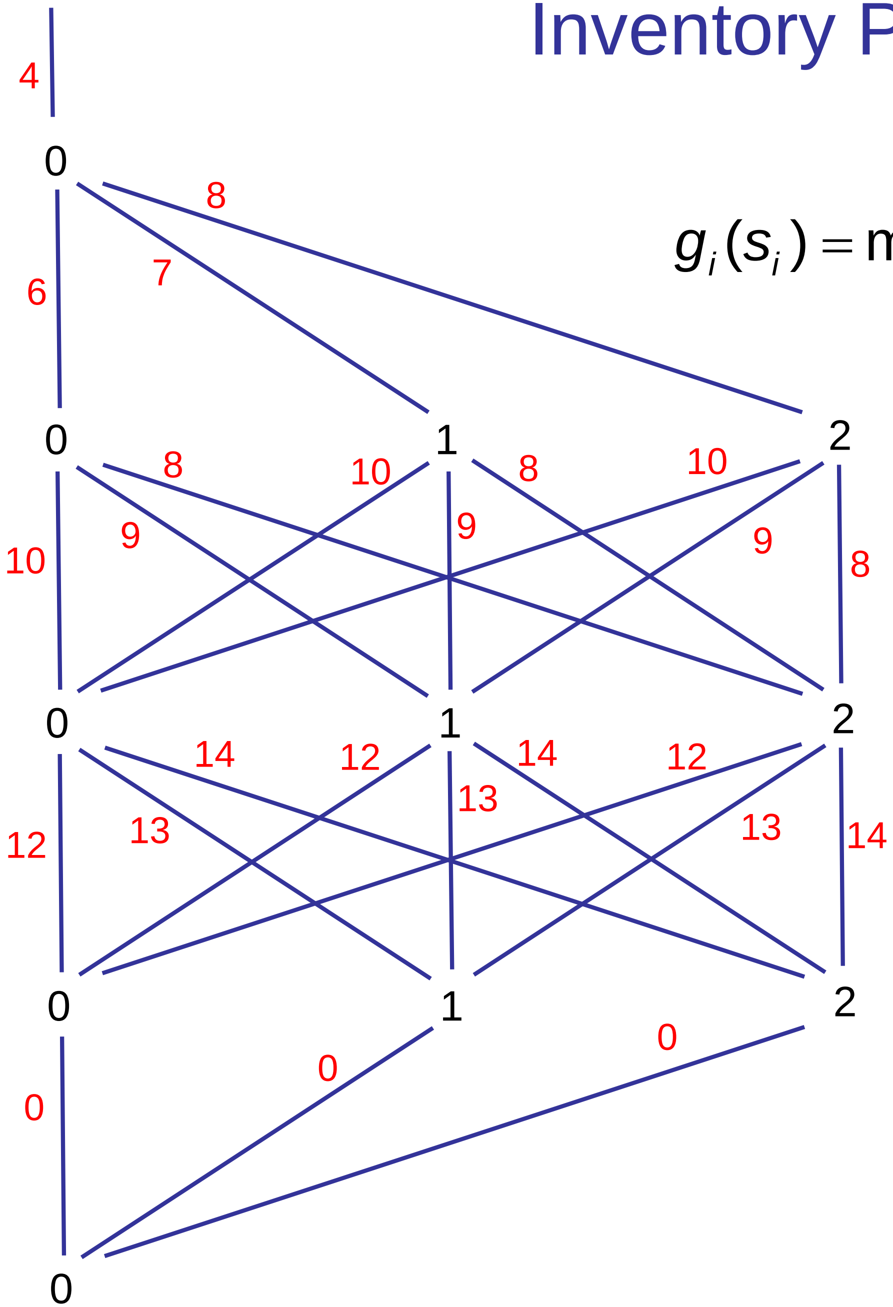


$$g_i(s_i) = \min_{x_i'} \left\{ h_i s_i + c_i(x_i' - s_i + d_i) + g_{i+1}(x_i') \right\}$$

To obtain canonical costs, subtract $c_i(m - s_i) + h_i s_i$ from cost on each arc (s_i, s_{i+1}).

Add these offsets to incoming arcs.

Inventory Problem

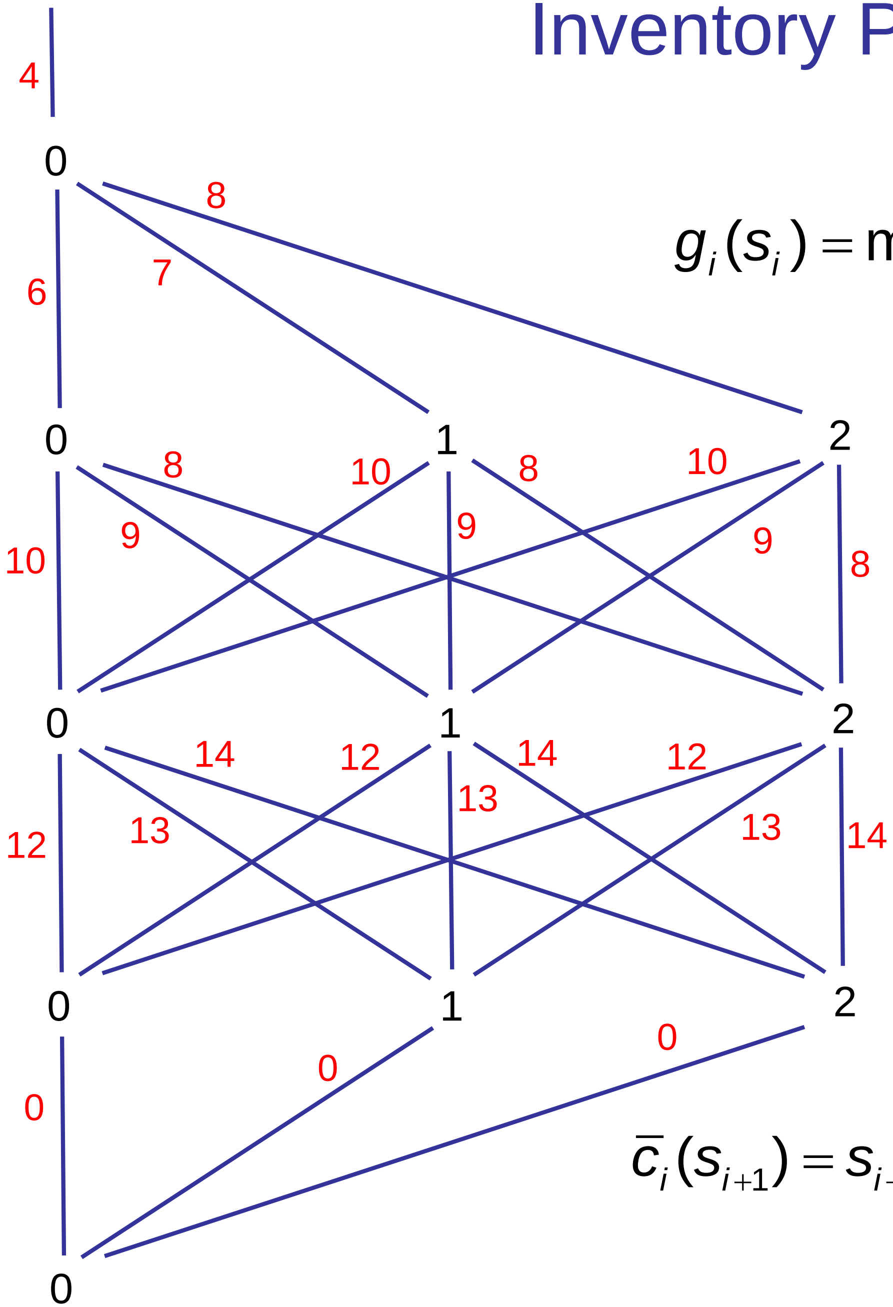


$$g_i(s_i) = \min_{x_i'} \left\{ h_i s_i + c_i(x_i' - s_i + d_i) + g_{i+1}(x_i') \right\}$$

To obtain canonical costs, subtract $c_i(m - s_i) + h_i s_i$ from cost on each arc (s_i, s_{i+1}).

Add these offsets to incoming arcs.

Inventory Problem



$$g_i(s_i) = \min_{x_i'} \left\{ h_i s_i + c_i(x_i' - s_i + d_i) + g_{i+1}(x_i') \right\}$$

To obtain canonical costs, subtract $c_i(m - s_i) + h_i s_i$ from cost on each arc (s_i, s_{i+1}).

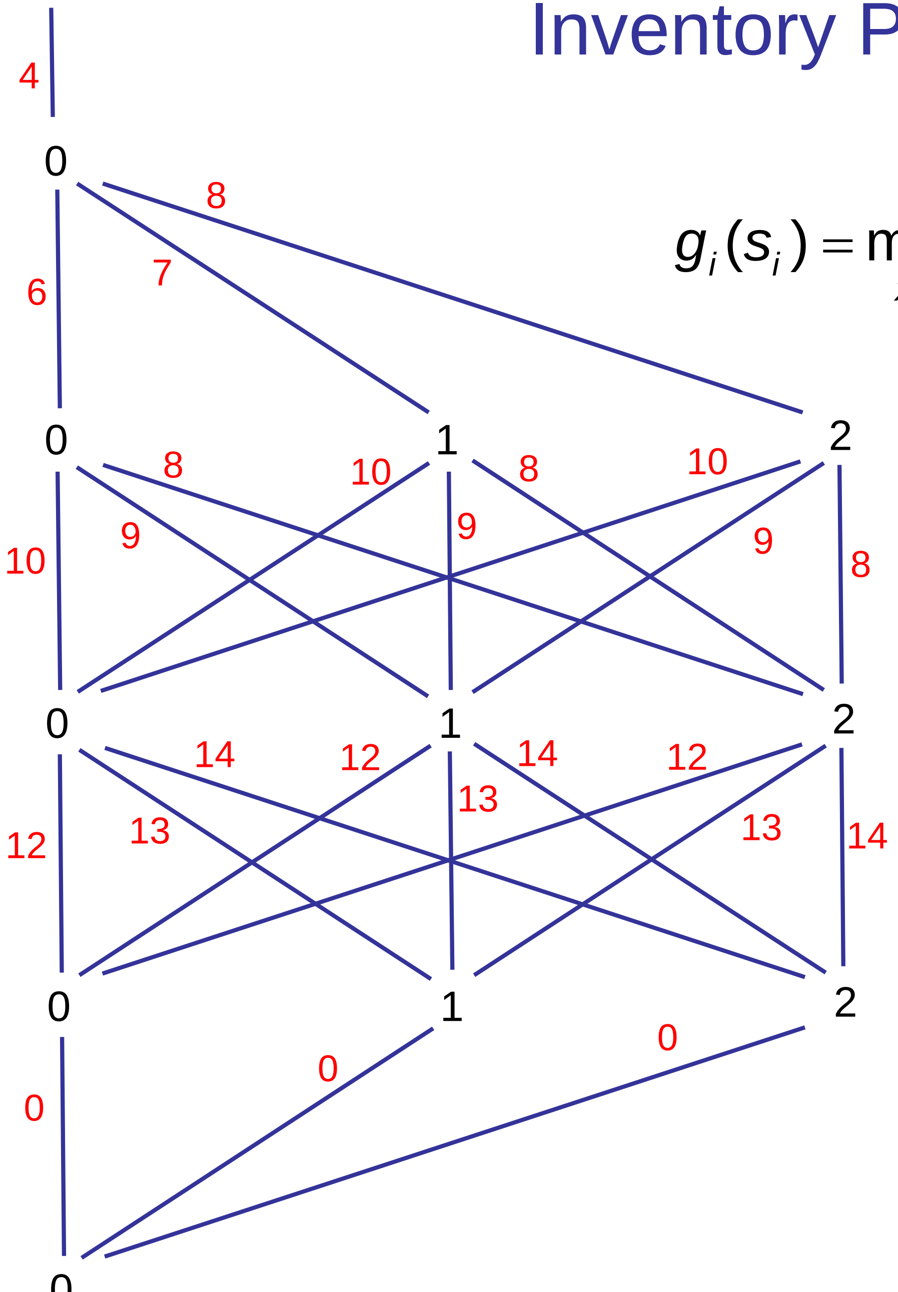
Add these offsets to incoming arcs.

Now outgoing arcs look alike.

And all arcs into state s_i have the same cost

$$\bar{c}_i(s_{i+1}) = s_{i+1} h_{i+1} + c_i(d_i - s_{i+1} - m) + c_{i+1}(m - s_{i+1})$$

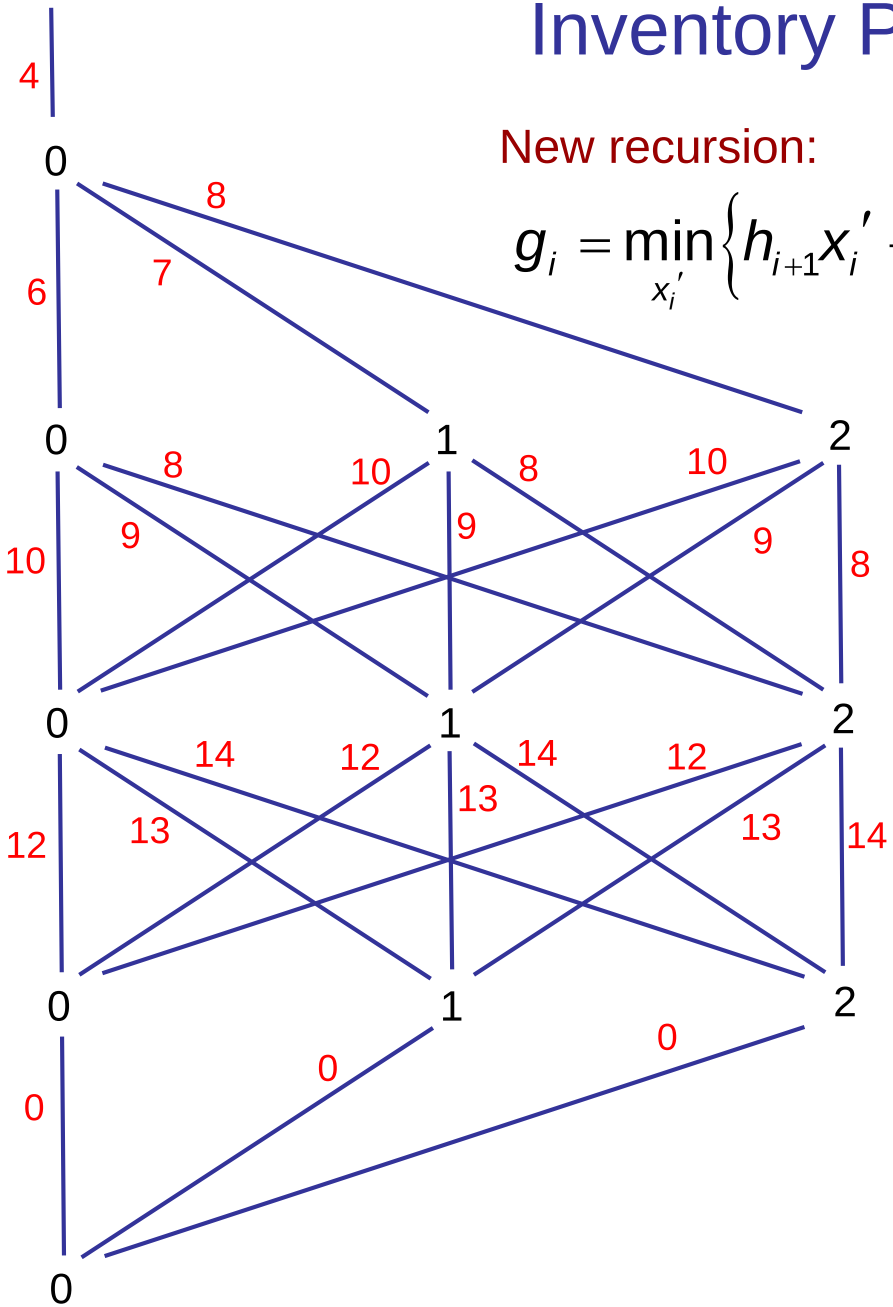
Inventory Problem



$$g_i(s_i) = \min_{x_i'} \left\{ h_i s_i + c_i(x_i' - s_i + d_i) + g_{i+1}(x_i') \right\}$$

These are canonical costs with offset $\min_{s_{i+1}} \left\{ \bar{c}_i(s_{i+1}) \right\}$

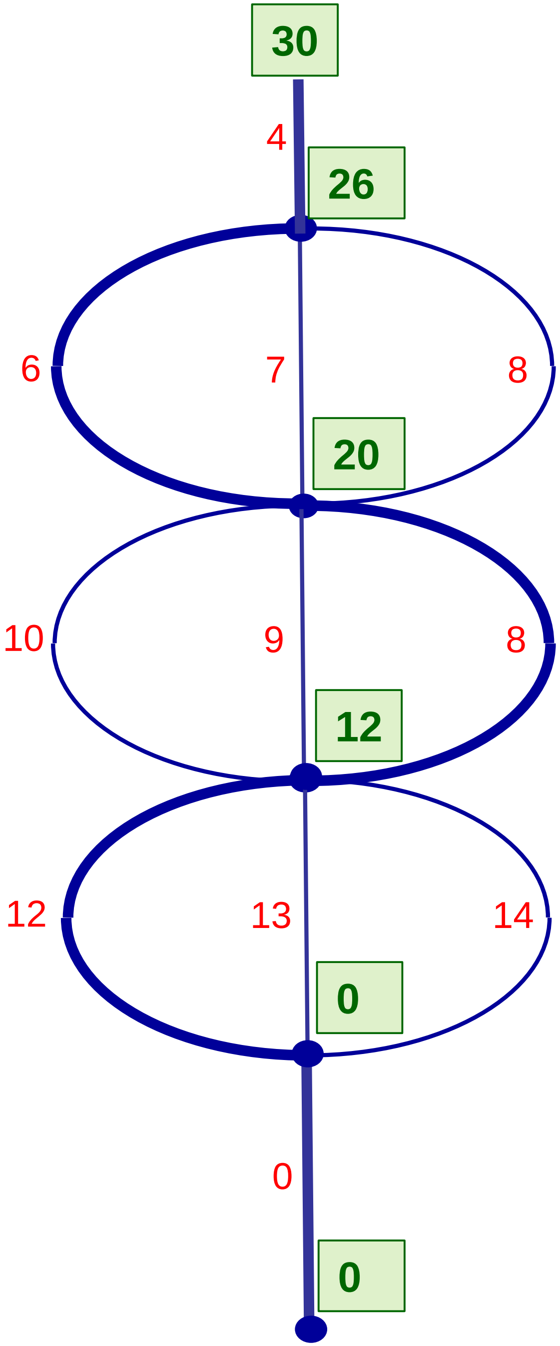
Inventory Problem



New recursion:

$$g_i = \min_{x_i'} \left\{ h_{i+1} x_i' + c_i (x_i' - m + d_i) + c_{i+1}(m - x_i') + g_{i+1} \right\}$$

These are canonical costs with offset $\min_{s_{i+1}} \left\{ \bar{c}_i (s_{i+1}) \right\}$



Inventory Problem

New recursion:

$$g_i = \min_{x_i'} \left\{ h_{i+1} x_i' + c_i (x_i' - m + d_i) + c_{i+1}(m - x_i') + g_{i+1} \right\}$$

Now there is only one state per period.

Current Research

- Broader applicability
  - Stochastic dynamic programming
  - Continuous global optimization
- Combination with other techniques
  - Lagrangean relaxation.
  - Column generation
  - Logic-based Benders decomposition
    - Solve separation problem

References

2006

- T. Hadzic and J. N. Hooker. Discrete global optimization with binary decision diagrams. In *Workshop on Global Optimization: Integrating Convexity, Optimization, Logic Programming, and Computational Algebraic Geometry (GICOLAG)*, Vienna, 2006.

2007

- Tarik Hadzic and J. N. Hooker. Cost-bounded binary decision diagrams for 0-1 programming. In *Proceedings of CPAIOR*. LNCS 4510, pp. 84-98. Springer, 2007.
- Tarik Hadzic and J. N. Hooker. Postoptimality analysis for integer programming using binary decision diagrams. December 2007, revised April 2008 (not submitted).
- M. Behle. Binary Decision Diagrams and Integer Programming. PhD thesis, Max Planck Institute for Computer Science, 2007.
- H. R. Andersen, T. Hadzic, J. N. Hooker, and P. Tiedemann. A constraint store based on multivalued decision diagrams. In *Proceedings of CP*. LNCS 4741, pp. 118-132. Springer, 2007.

2008

- T. Hadzic, J. N. Hooker, B. O'Sullivan, and P. Tiedemann. Approximate compilation of constraints into multivalued decision diagrams. In *Proceedings of CP*. LNCS 5202, pp. 448-462. Springer, 2008.
- T. Hadzic, J. N. Hooker, and P. Tiedemann. Propagating separable equalities in an MDD store. In *Proceedings of CPAIOR*. LNCS 5015, pp. 318-322. Springer, 2008.

References

2010

- S. Hoda. Essays on Equilibrium Computation, MDD-based Constraint Programming and Scheduling. *PhD thesis*, Carnegie Mellon University, 2010.
- S. Hoda, W.-J. van Hoeve, and J. N. Hooker. A Systematic Approach to MDD-Based Constraint Programming. In *Proceedings of CP*. LNCS 6308, pp. 266-280. Springer, 2010.
- T. Hadzic, E. O'Mahony, B. O'Sullivan, and M. Sellmann. Enhanced inference for the market split problem. In *Proceedings, International Conference on Tools for AI (ICTAI)*, pages 716–723. IEEE, 2009.

2011

- D. Bergman, W.-J. van Hoeve, and J. N. Hooker. Manipulating MDD Relaxations for Combinatorial Optimization. In *Proceedings of CPAIOR*. LNCS 6697, pp. 20-35. Springer, 2011.

2012

- A. A. Cire and W.-J. van Hoeve. MDD Propagation for Disjunctive Scheduling. In *Proceedings of ICAPS*, pp. 11-19. AAAI Press, 2012.
- D. Bergman, A.A. Cire, W.-J. van Hoeve, and J.N. Hooker. Variable Ordering for the Application of BDDs to the Maximum Independent Set Problem. In *Proceedings of CPAIOR*. LNCS 7298, pp. 34-49. Springer, 2012.

References

2013

- A. A. Cire and W.-J. van Hoeve. Multivalued Decision Diagrams for Sequencing Problems. *Operations Research* 61(6): 1411-1428, 2013.
- D. Bergman. New Techniques for Discrete Optimization. *PhD thesis*, Carnegie Mellon University, 2013.
- J. N. Hooker. Decision Diagrams and Dynamic Programming. In *Proceedings of CPAIOR*. LNCS 7874, pp. 94-110. Springer, 2013.
- B. Kell and W.-J. van Hoeve. An MDD Approach to Multidimensional Bin Packing. In *Proceedings of CPAIOR*, LNCS 7874, pp. 128-143. Springer, 2013.

2014

- D. R. Morrison, E. C. Sewell, S. H. Jacobson , Characteristics of the maximal independent set ZDD, Journal of Combinatorial Optimization 28 (1) 121-139, 2014
- D. R. Morrison, E. C. Sewell, S. H. Jacobson, Solving the Pricing Problem in a Generic Branch-and-Price Algorithm using Zero-Suppressed Binary Decision Diagrams,
- D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. N. Hooker. Optimization Bounds from Binary Decision Diagrams. *INFORMS Journal on Computing* 26(2): 253-258, 2014.
- A. A. Cire. Decision Diagrams for Optimization. *PhD thesis*, Carnegie Mellon University, 2014.
- D. Bergman, A. A. Cire, and W.-J. van Hoeve. MDD Propagation for Sequence Constraints. *JAIR*, Volume 50, pages 697-722, 2014.
- D. Bergman, A. A. Cire, W.-J. van Hoeve, and T. Yunes. BDD-Based Heuristics for Binary Optimization. *Journal of Heuristics* 20(2): 211-234, 2014.

References

2014

- D. Bergman, A. A. Cire, A. Sabharwal, H. Samulowitz, V. Saraswat, and W.-J. van Hoeve. Parallel Combinatorial Optimization with Decision Diagrams. In *Proceedings of CPAIOR*, LNCS 8451, pp. 351-367. Springer, 2014.
- A. A. Cire and J. N. Hooker. The Separation Problem for Binary Decision Diagrams. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2014.]

2015

- D. Bergman, A. A. Cire, and W.-J. van Hoeve. Lagrangian Bounds from Decision Diagrams. *Constraints* 20(3): 346-361, 2015.
- B. Kell, A. Sabharwal, and W.-J. van Hoeve. BDD-Guided Clause Generation. In *Proceedings of CPAIOR*, 2015.

2016

- D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. N. Hooker, *Decision Diagrams for Optimization*, Springer, to appear.
- D. Bergman, A. A. Cire, W.-J. van Hoeve, and J. N. Hooker. Discrete Optimization with Decision Diagrams. *INFORMS Journal on Computing* 28: 47-66, 2016.