Toward Stochastic Optimization with Decision Diagrams

John Hooker

ISMP
Pittsburgh, July 2015

Objective

- **Relaxed decision diagrams** provide a general-purpose method for discrete optimization.
  - When the problem has a **dynamic programming** model.
  - It can **outperform MIP** even on problems with natural MIP formulations.
- Goal: extend the method to **stochastic optimization.**

Motivation

- Historical focus on **inequality models**.
  - Problem can be solved by **branch and bound**.
  - Good bounds from **cutting planes**.
- **Recursive (DP) models** are less common.
  - Powerful modeling paradigm – nonlinear, nonconvex.
  - But must enumerate **exponential** state space.

Motivation

- Solution: solve **recursive model** with **branch and bound**!
  - Decision diagrams allow this.
  - Good bounds from **relaxed decision diagrams**.
- Today's goal: conceptual basis to extend DDs to **stochastic problems**.
  - Define **relaxed stochastic decision diagrams**.

Outline

- Review **discrete optimization** with relaxed decision diagrams
  - Some previous results.

Outline

- Review **discrete optimization** with relaxed decision diagrams
  - Some previous results.
- Define **stochastic decision diagrams** (SDDs).
  - This is easy.

Outline

- Review **discrete optimization** with relaxed decision diagrams
  - Some previous results.
- Define **stochastic decision diagrams** (SDDs).
  - This is easy.
- Main task: define **relaxed** SDDs.
  - Not so easy.

Outline

- Review **discrete optimization** with relaxed decision diagrams
  - Some previous results.
- Define **stochastic decision diagrams** (SDDs).
  - This is easy.
- Main task: define **relaxed** SDDs.
  - Not so easy.
- Show how to relax SDDs by **node merger**.

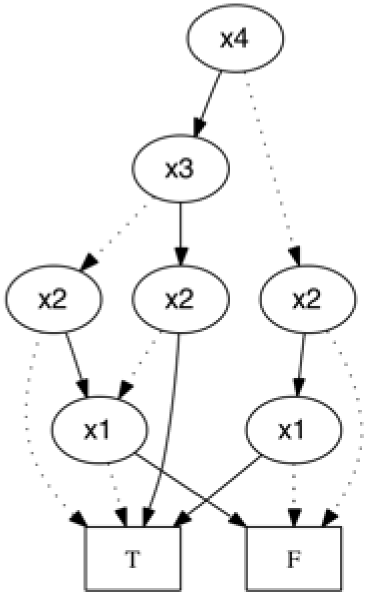
Outline

- Review **discrete optimization** with relaxed decision diagrams
  - Some previous results.
- Define **stochastic decision diagrams** (SDDs).
  - This is easy.
- Main task: define **relaxed** SDDs.
  - Not so easy.
- Show how to relax SDDs by **node merger**.
- Illustrate with a **sequencing** problem
  - No computational results yet.

Decision Diagrams

- Graphical encoding of a boolean function
  - Historically used for circuit design & verification
  - Adapt to optimization and constraint programming

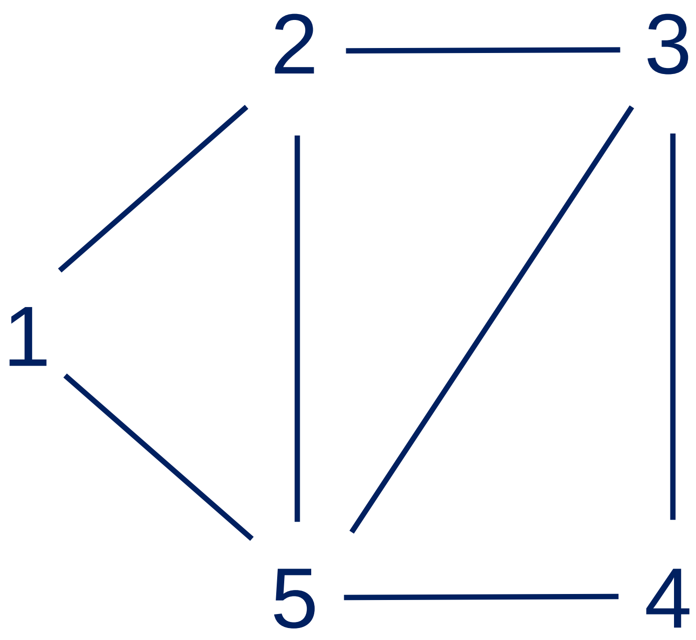
Hadžić and JH (2006, 2007)

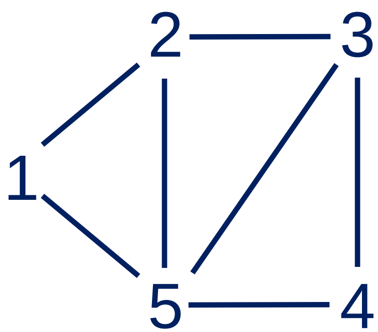


Stable Set Problem

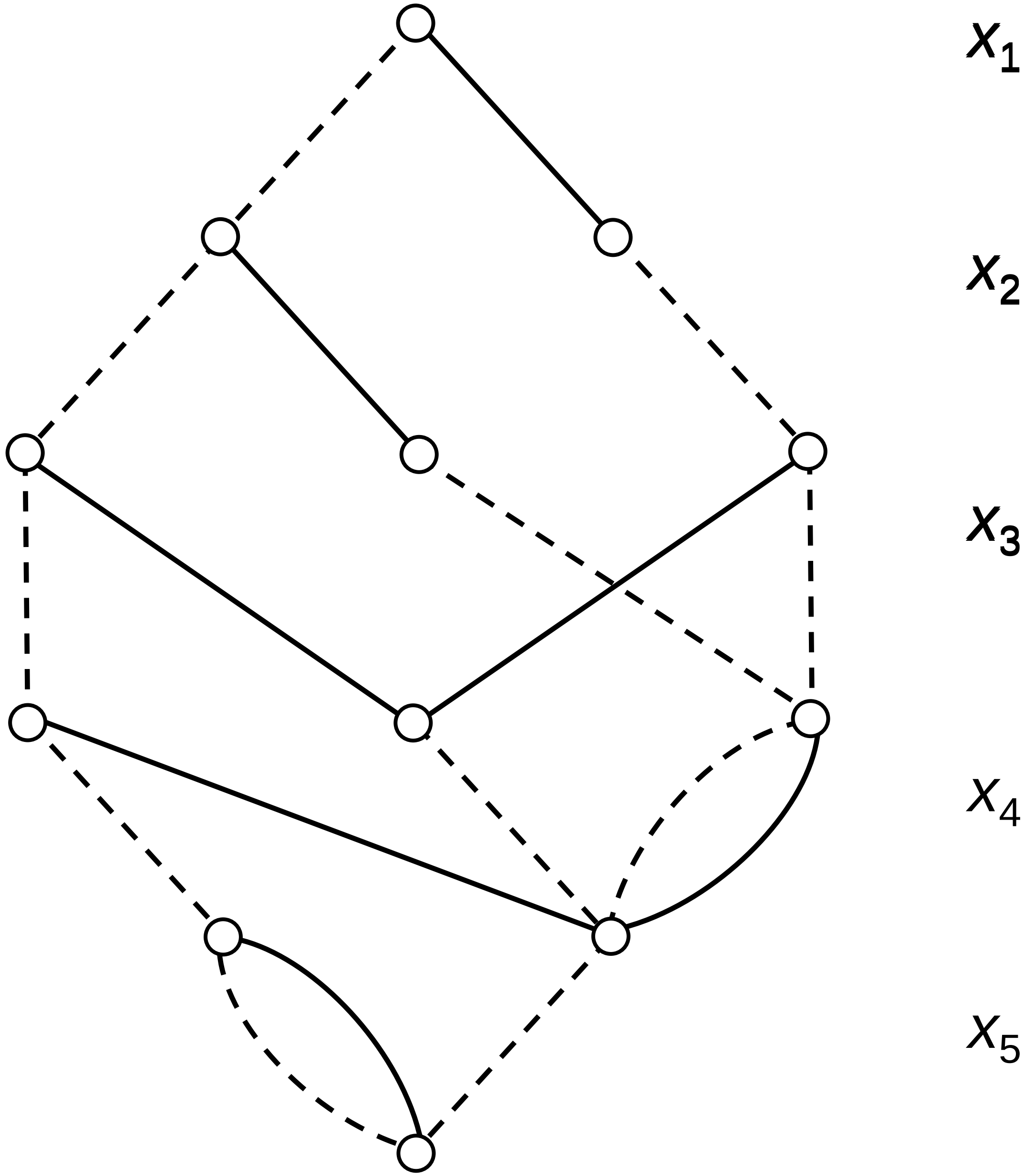
Let each vertex have weight w_i

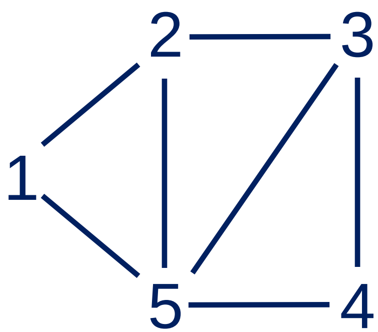
Select nonadjacent vertices to maximize $\sum_i w_i x_i$



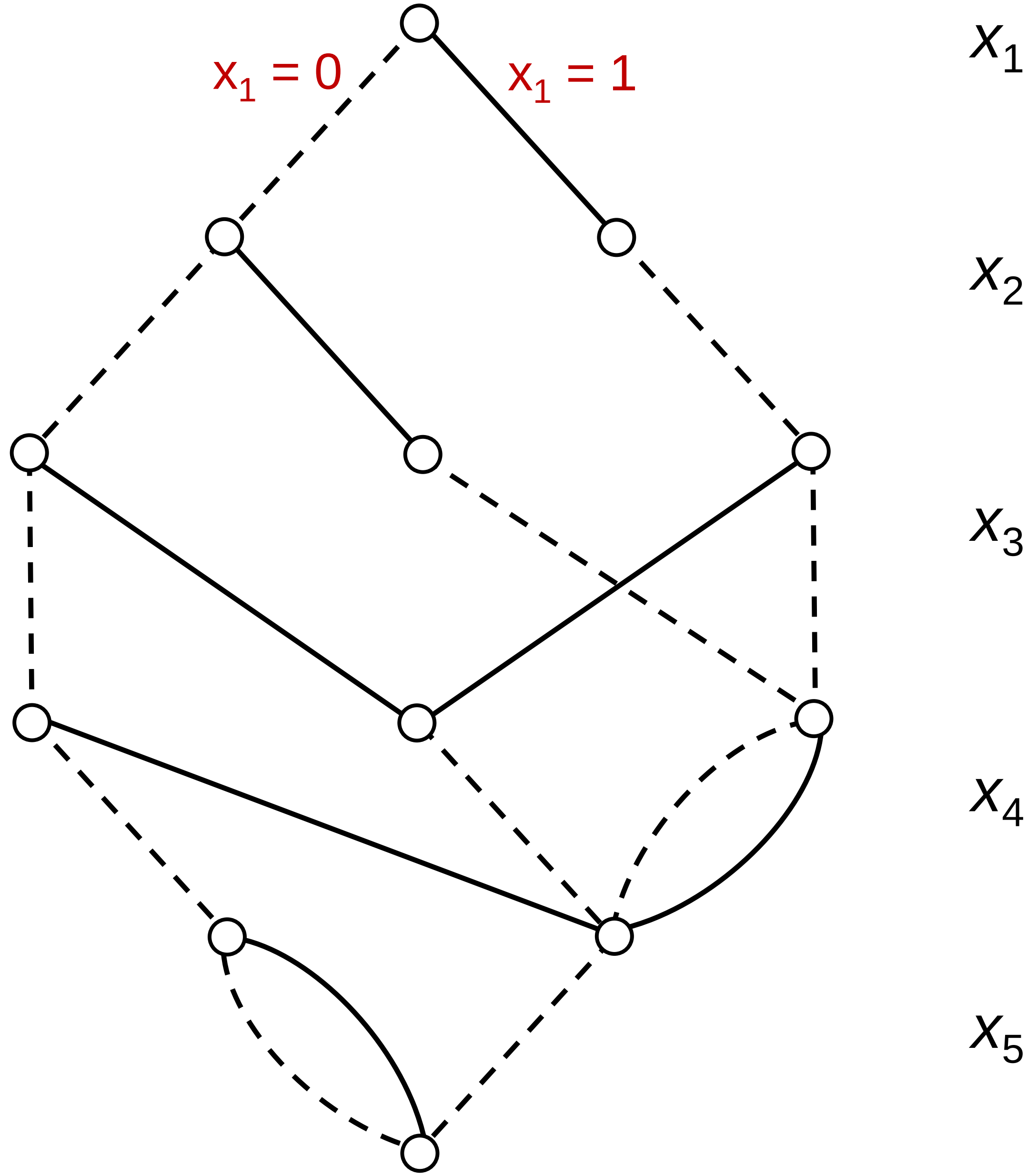


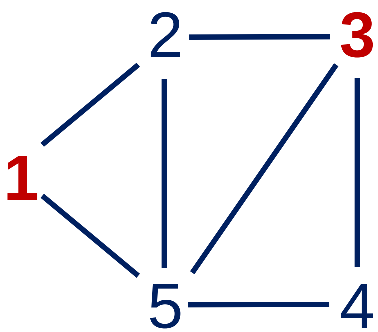
Exact DD for stable set problem



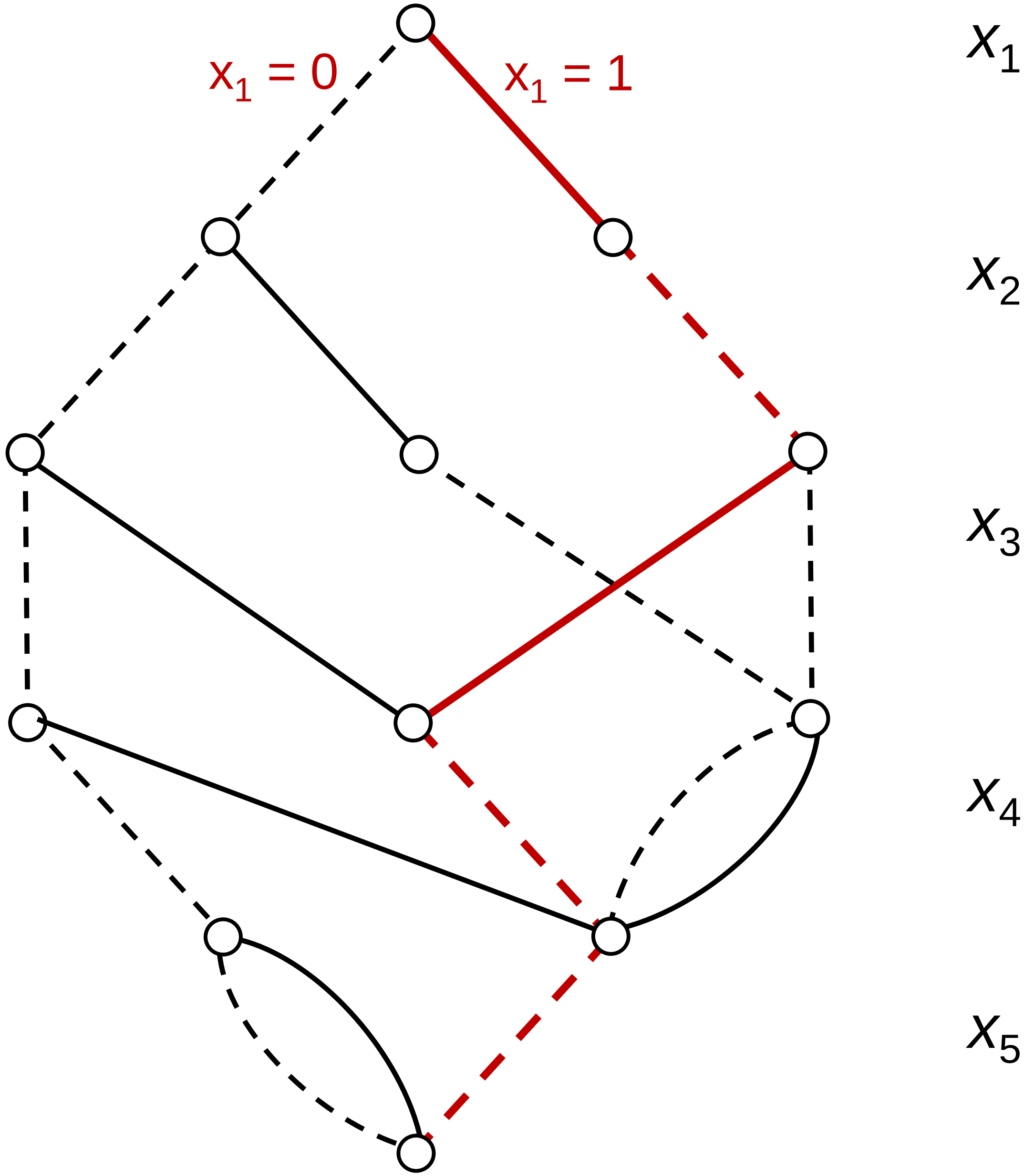


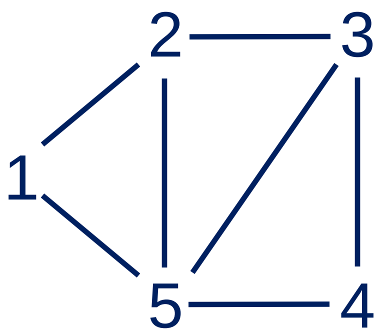
Exact DD for stable set problem



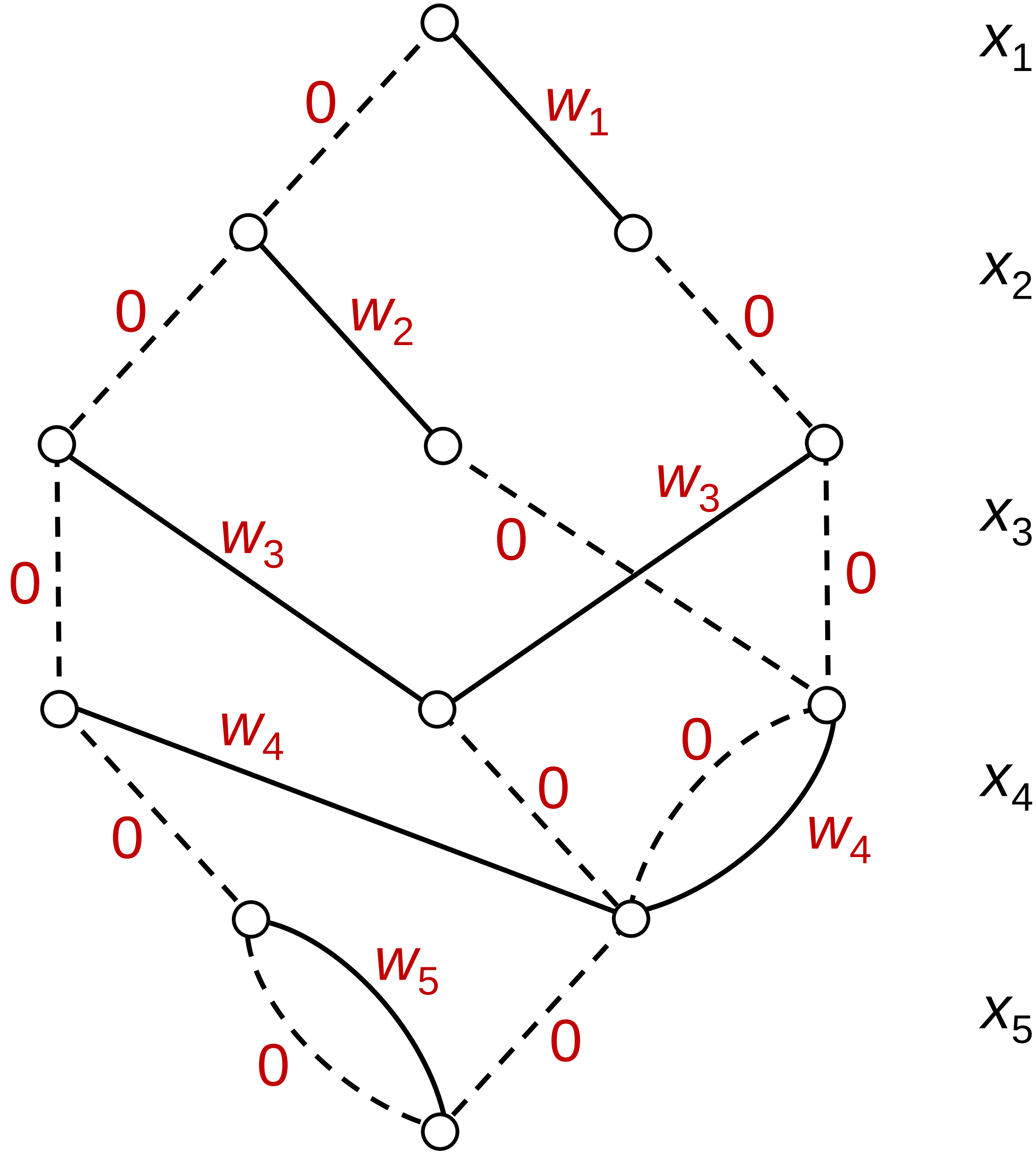


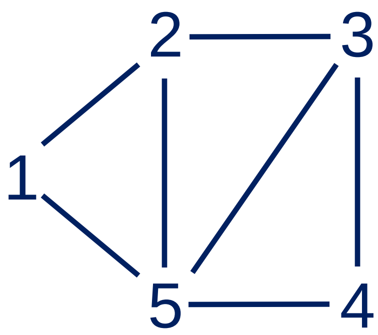
Paths from top to bottom correspond to the 9 feasible solutions



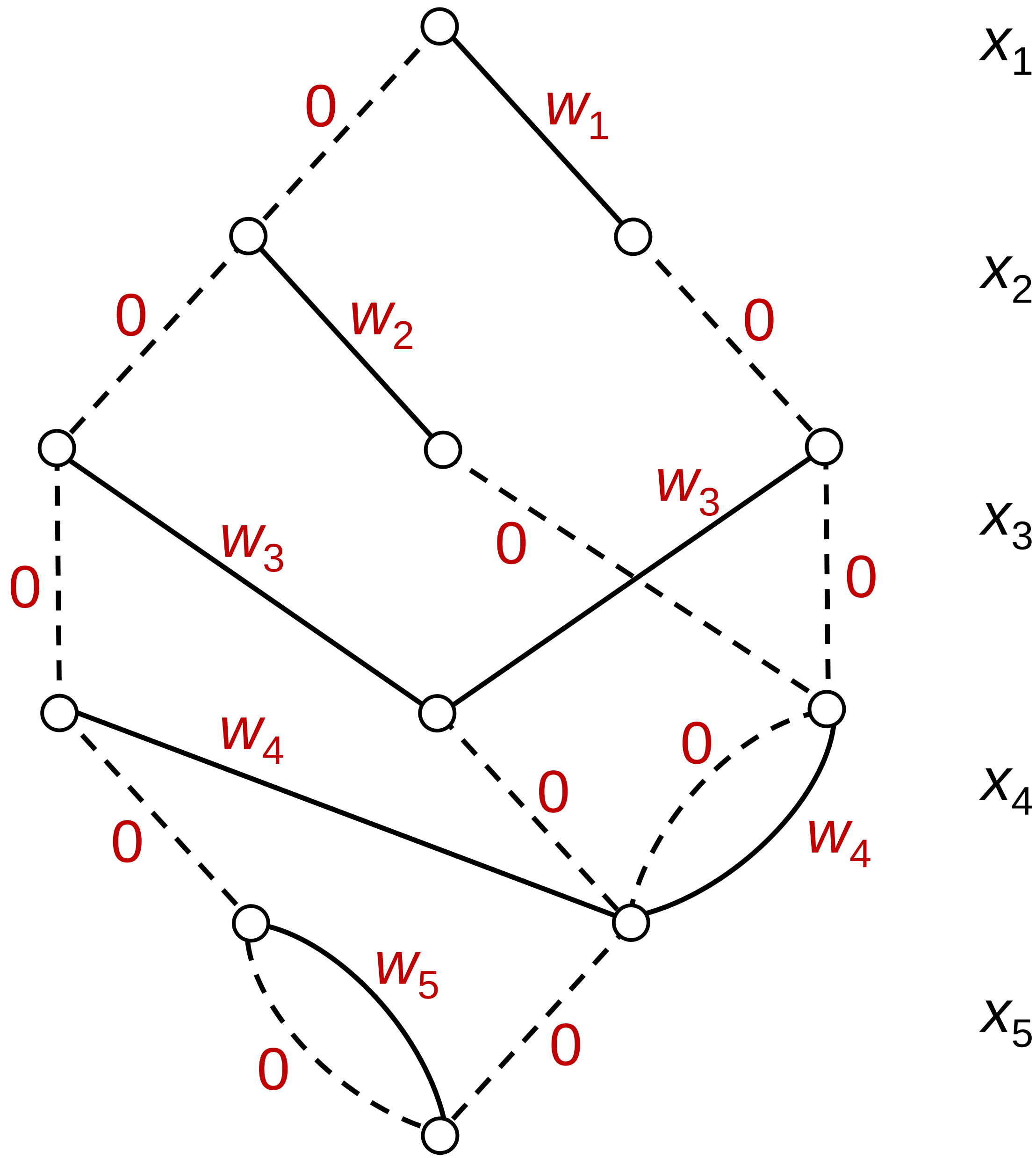


For objective function, associate weights with arcs





For objective function, associate weights with arcs



Optimal solution is **longest path**

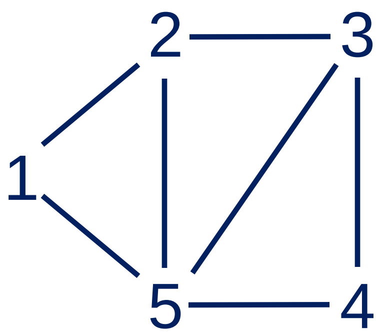
Objective Function

- 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.

JH (2014)

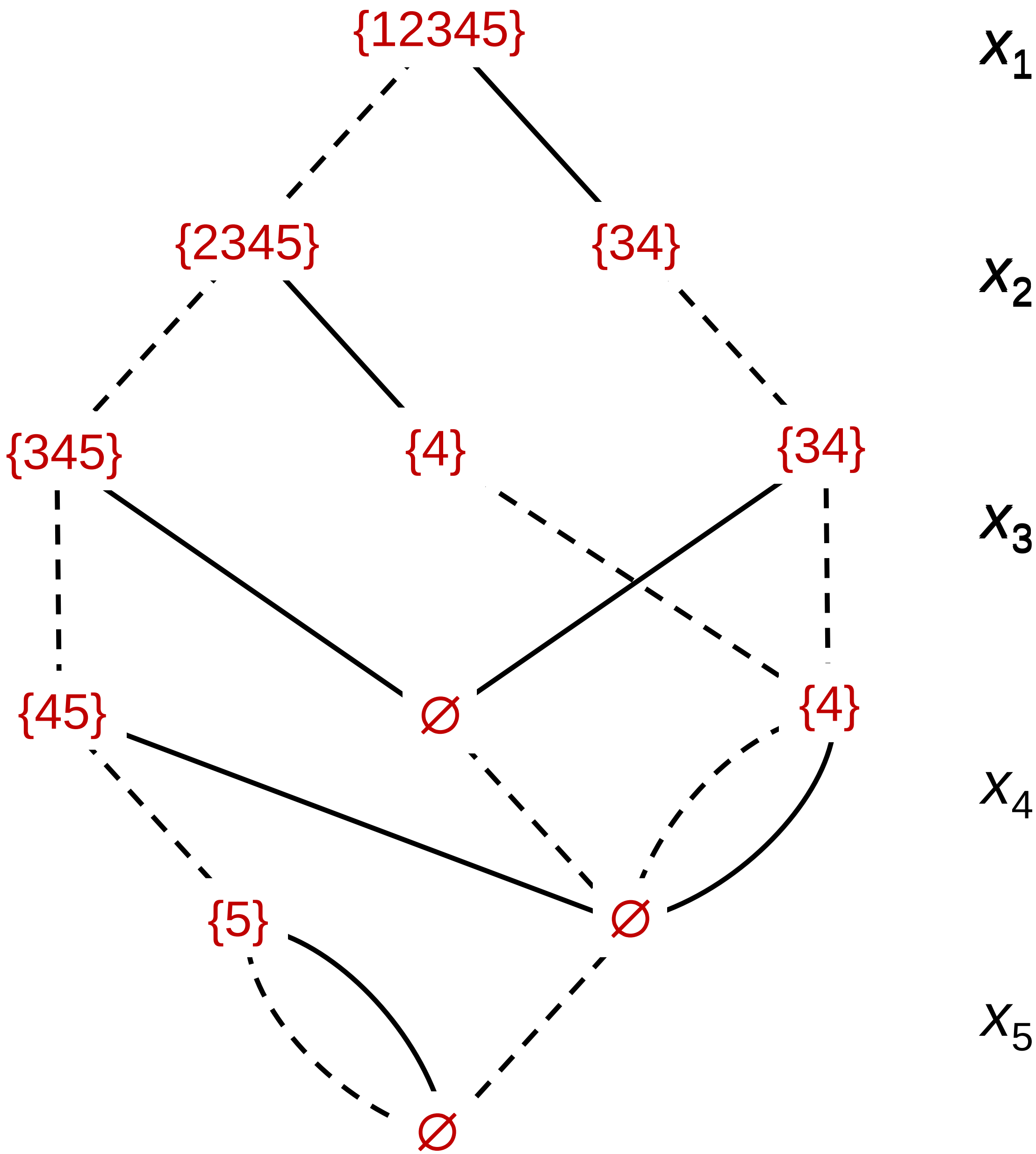
DP-Style Modeling

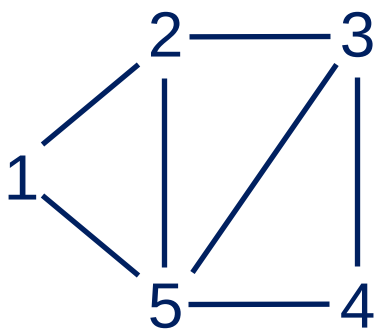
- Model has two components.
  - **DP model** of problem, using **state variables**.
    - Analogous to inequality model in IP.
  - Rule for **merging states** to create relaxed DD.
    - Analogous to adding valid inequalities in IP.



Exact DD for stable set problem

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





{12345}

x_1

x_2

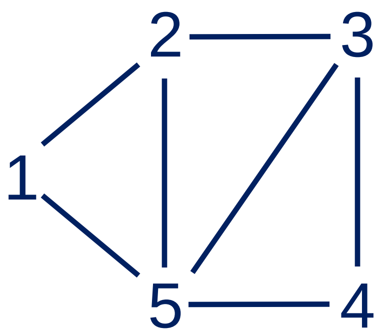
Exact DD for stable set problem

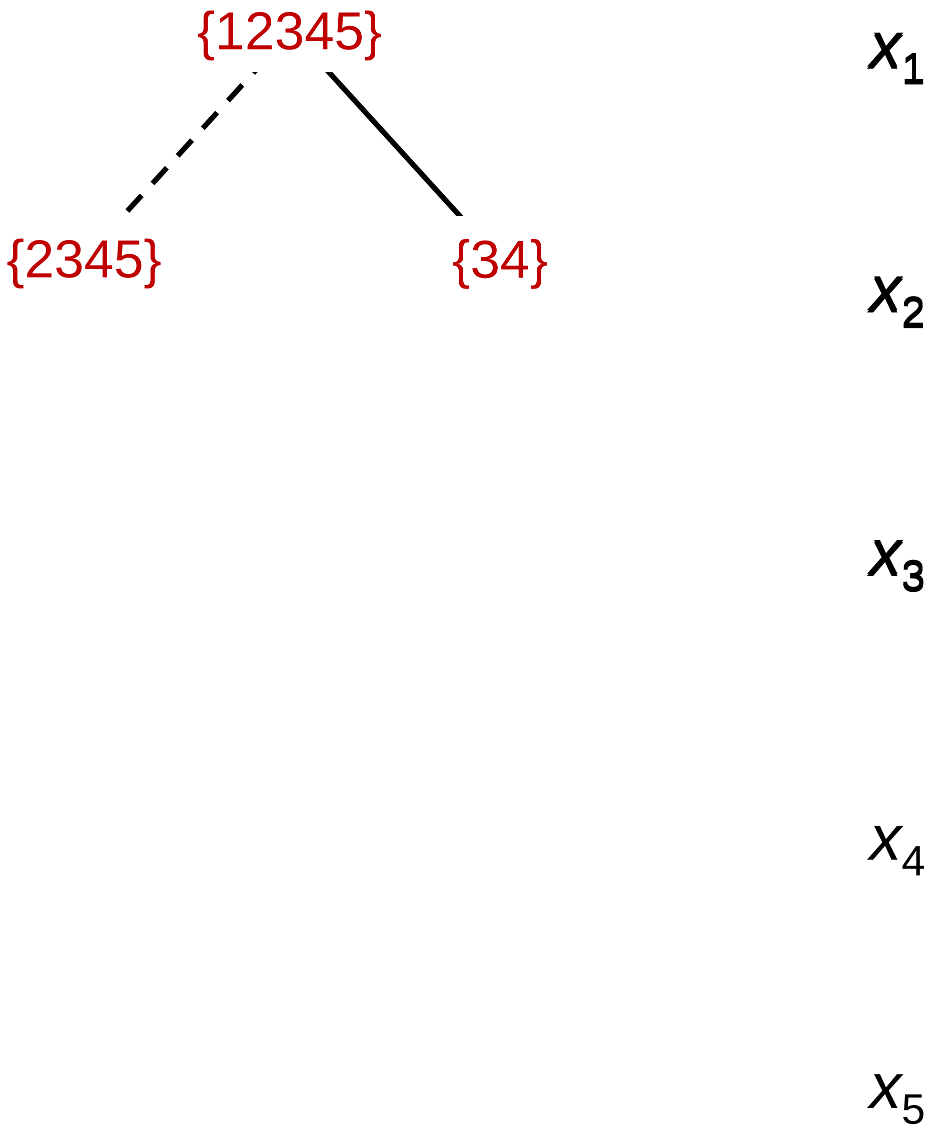
x_3

x_4

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

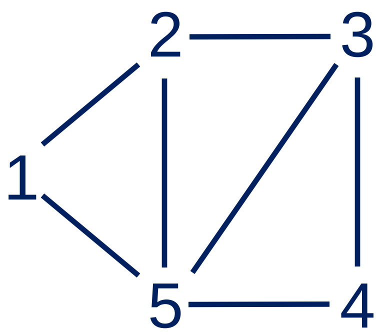
x_5

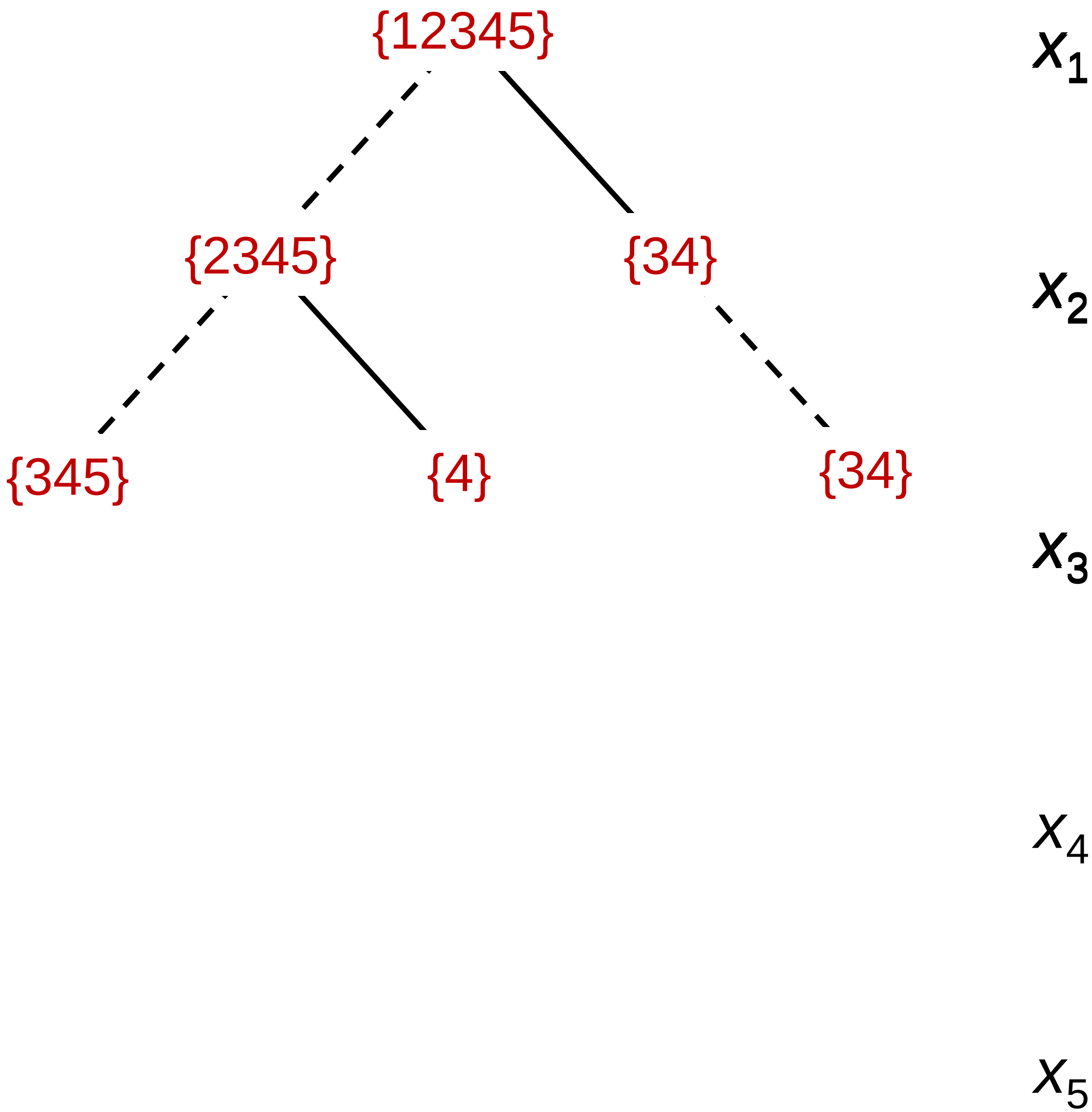




Exact DD for stable set problem

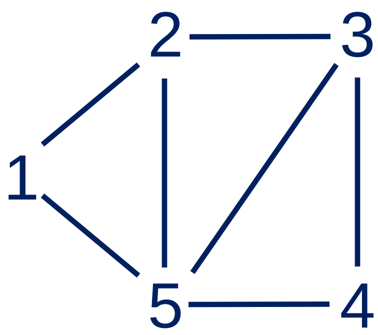
To build DD, associate **state** with each node

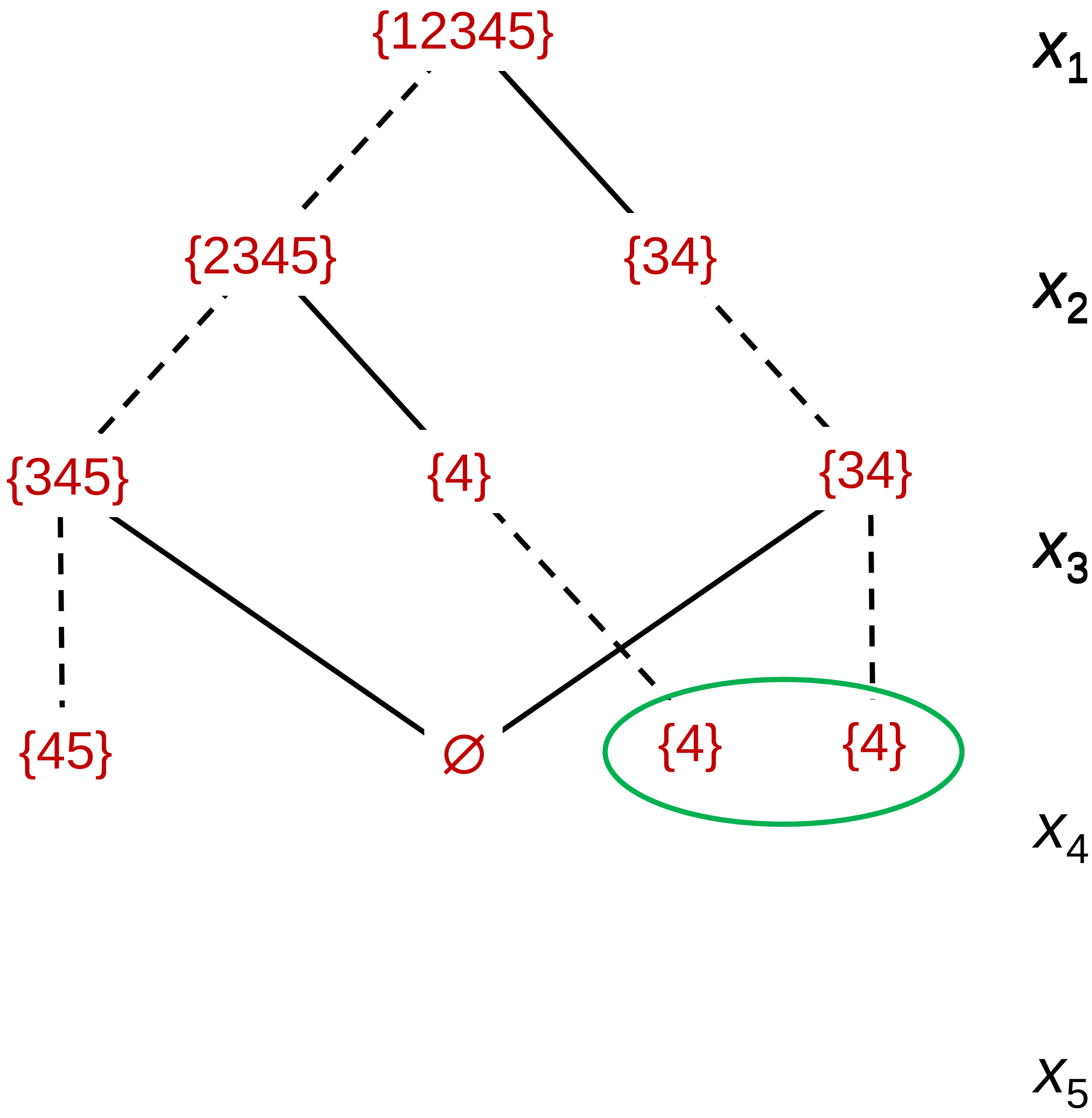




Exact DD for stable set problem

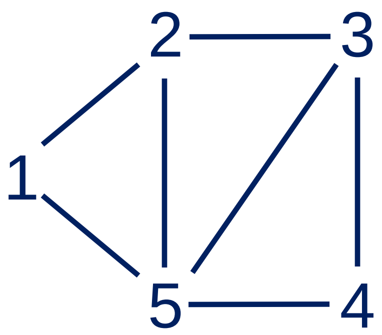
To build DD, associate **state** with each node

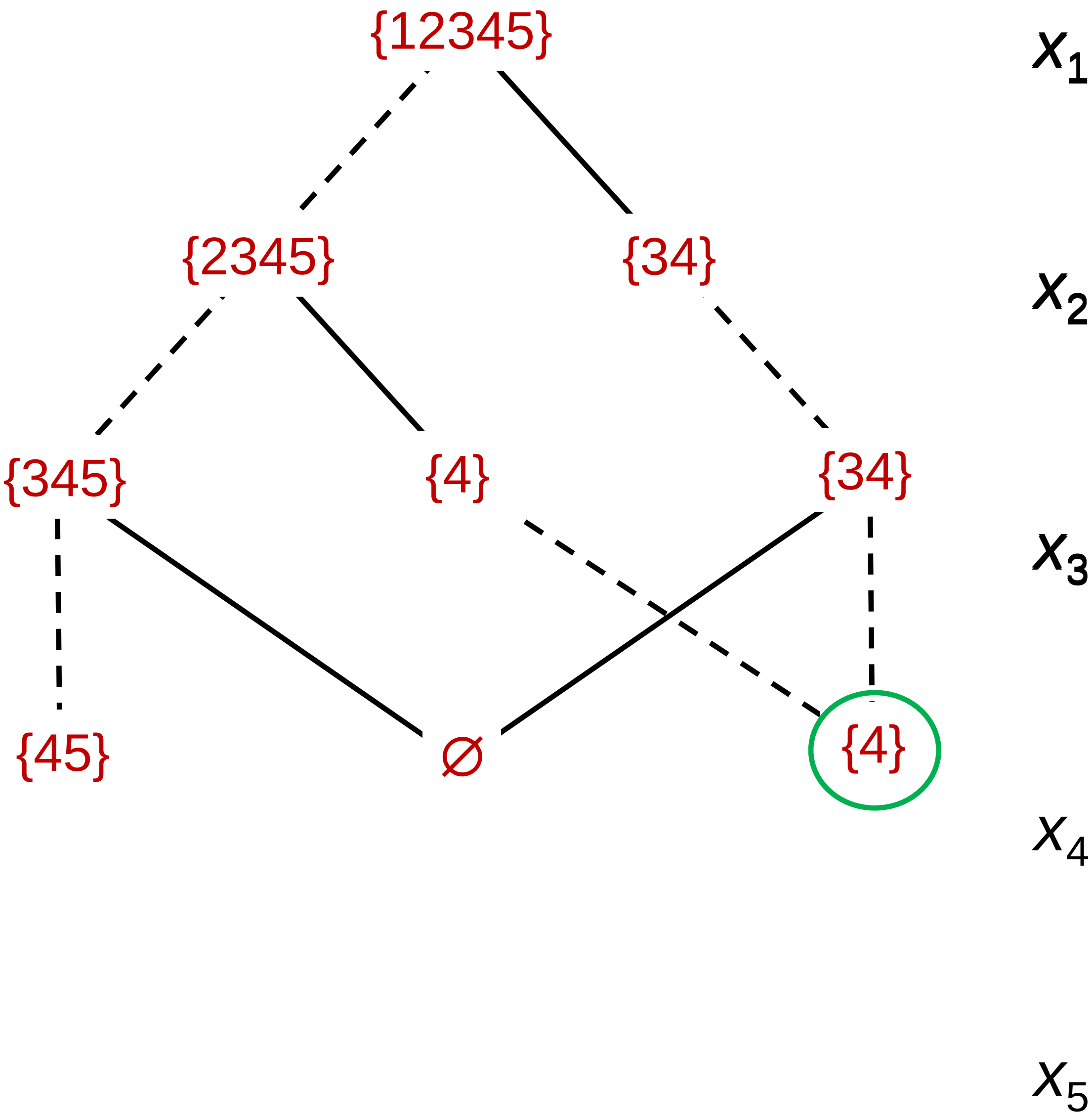




Exact DD for stable set problem

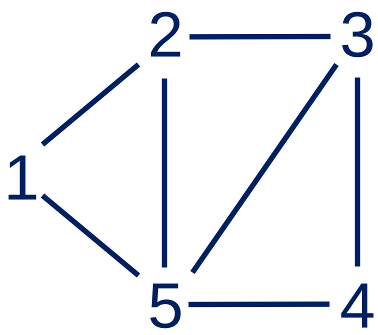
Merge nodes that correspond to the same state

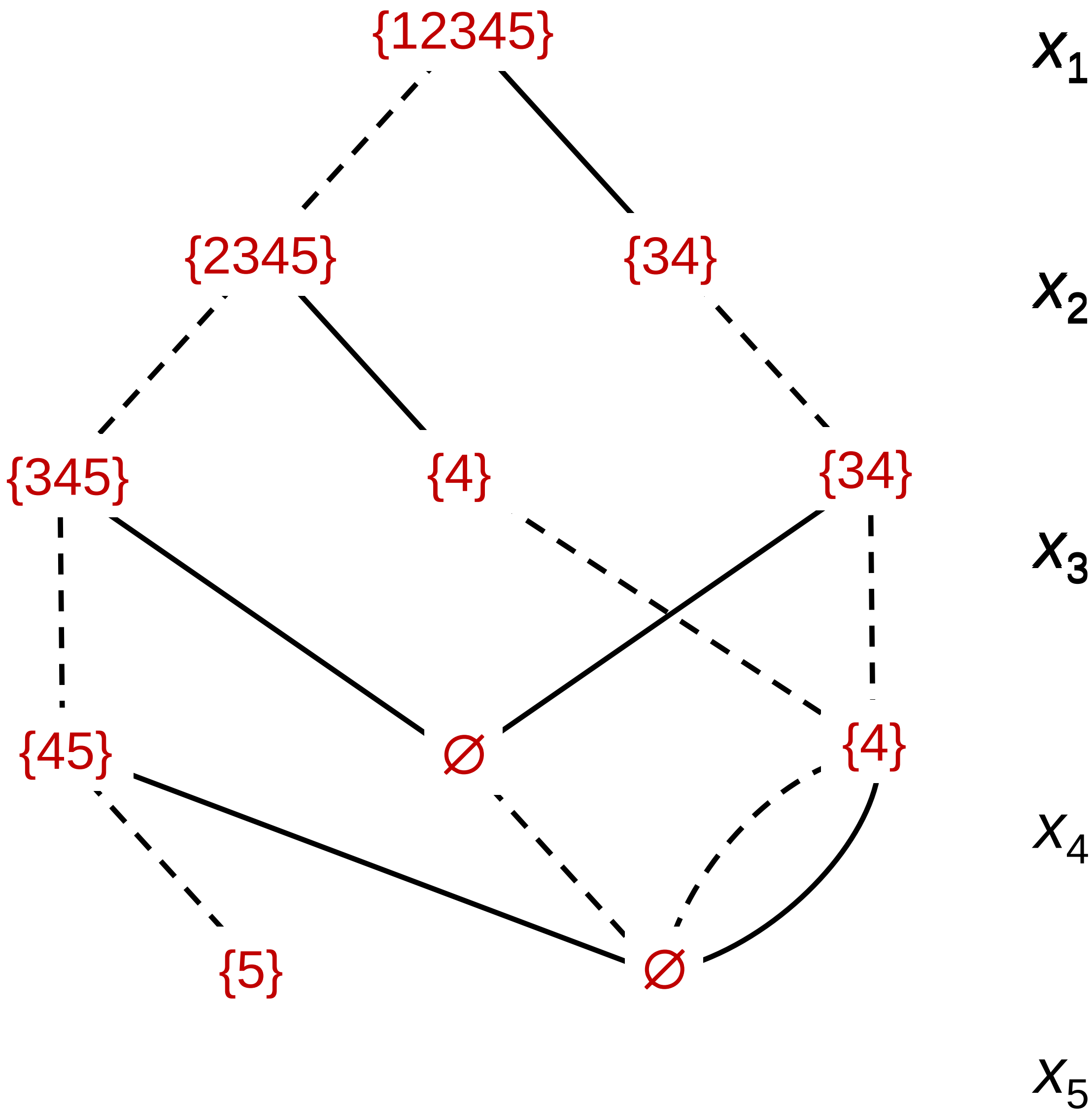




Exact DD for stable set problem

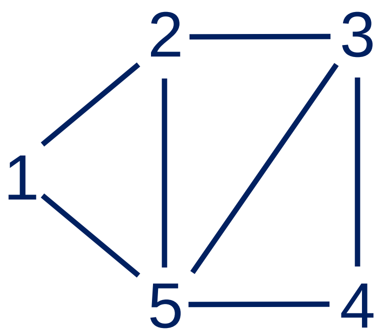
Merge nodes that correspond to the same state



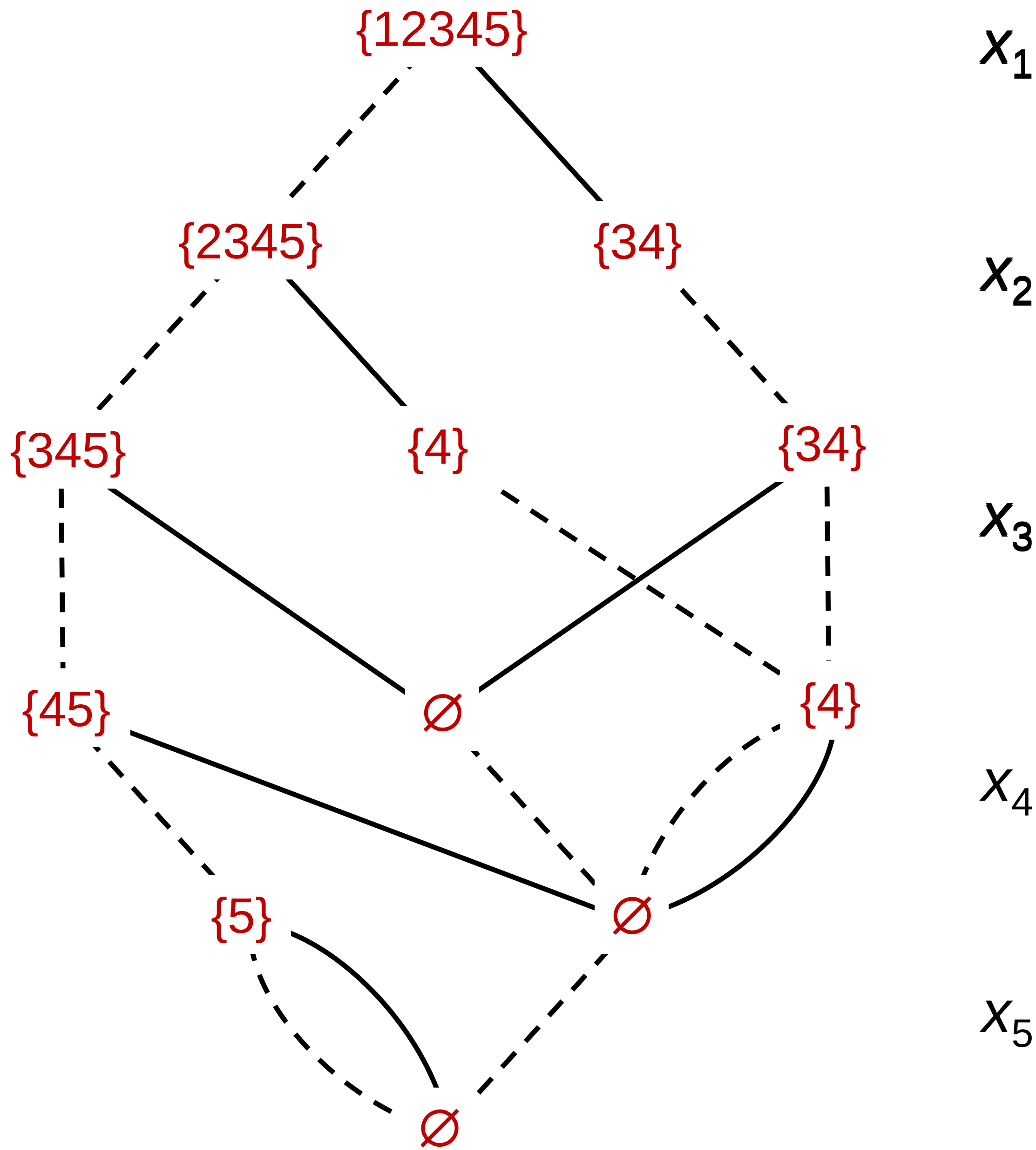


Exact DD for stable set problem

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



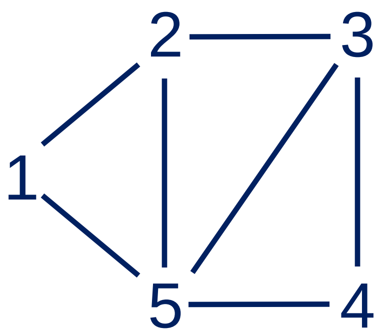
Exact DD for stable set problem



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

Relaxation Bounding

- To obtain a bound on the objective function:
  - Use a **relaxed** BDD
  - Analogous to LP relaxation in IP
  - This relaxation is **discrete**.
  - Doesn’t require the linear inequality formulation of IP.



{12345}

x_1

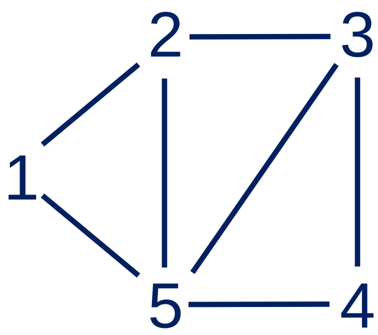
x_2

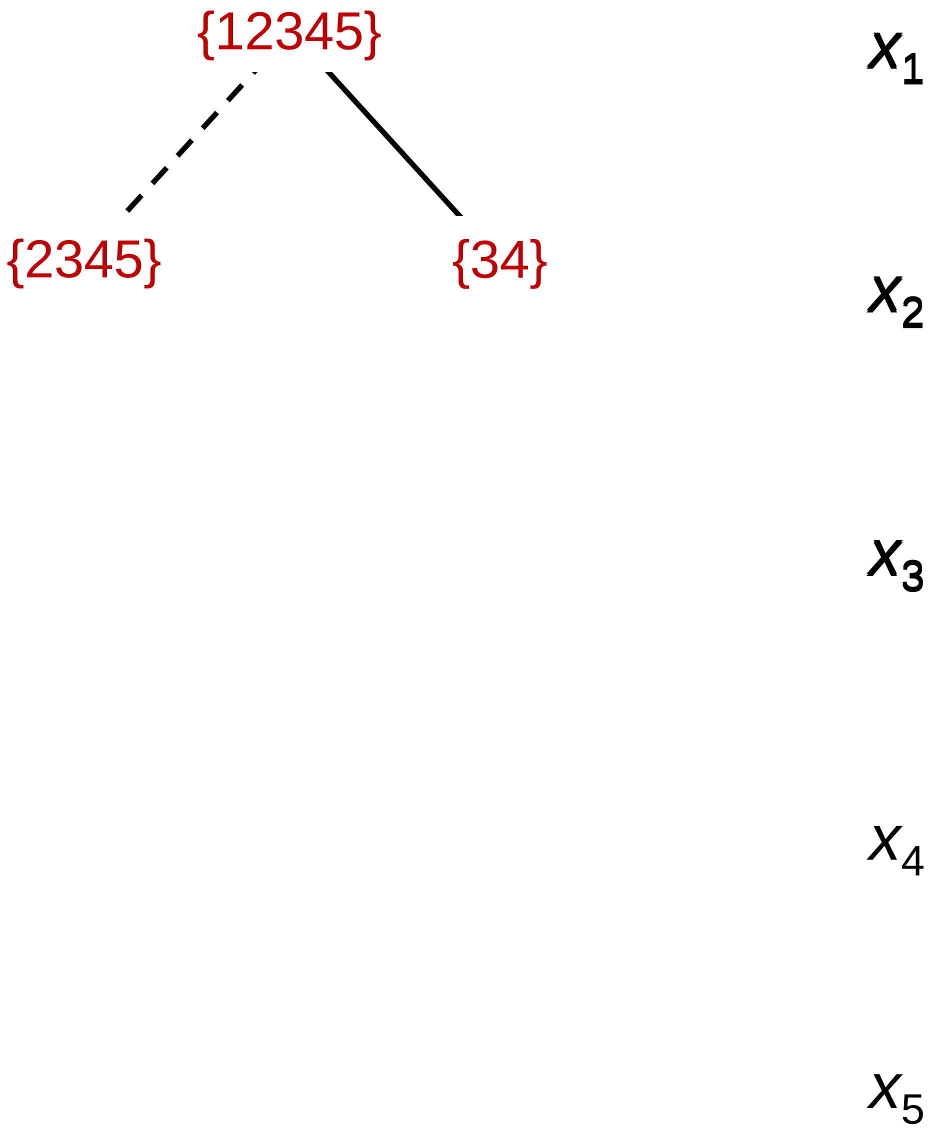
x_3

x_4

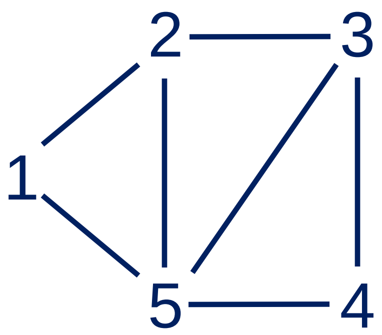
x_5

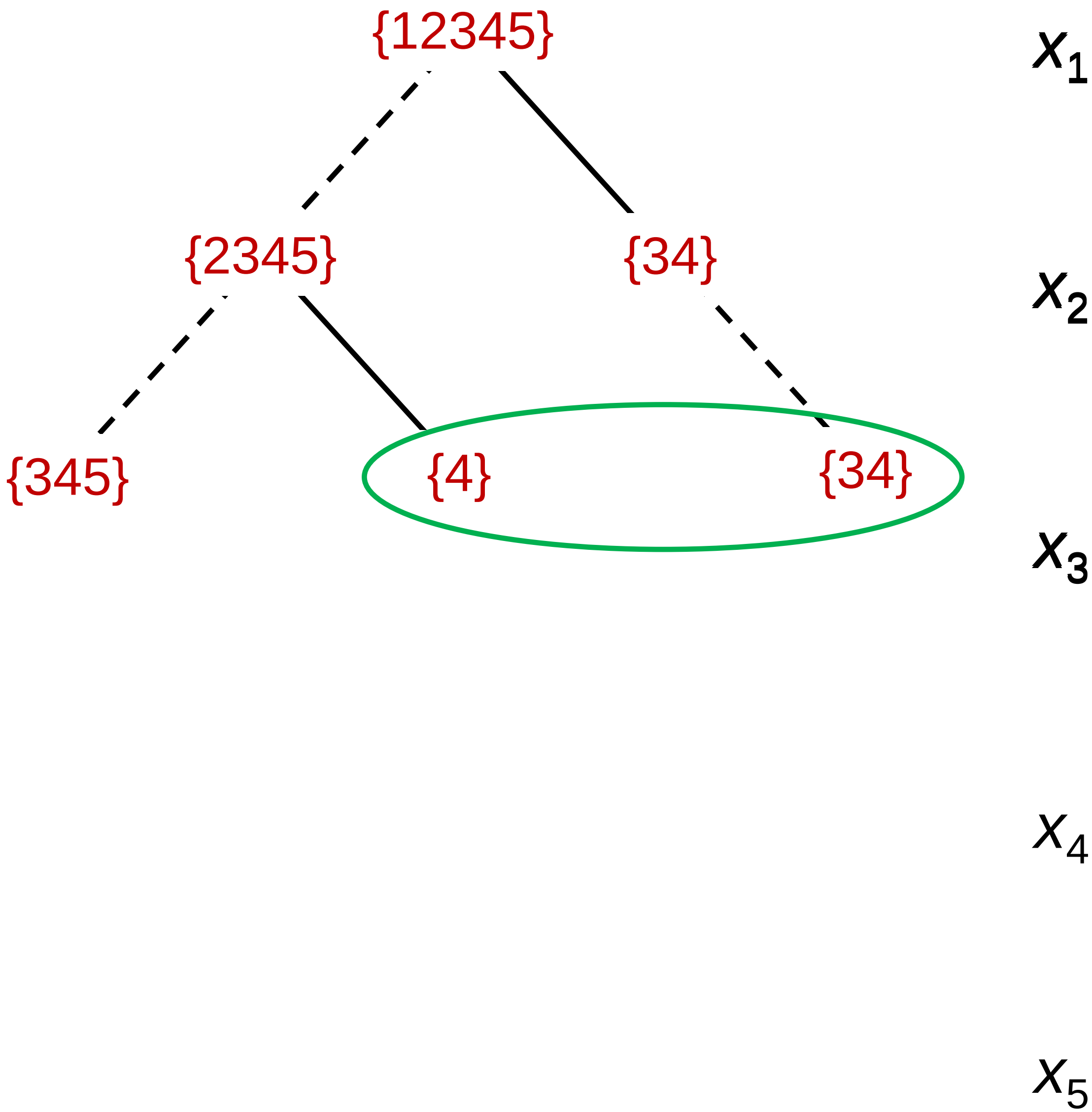
To build **relaxed** BDD, merge some additional nodes as we go along



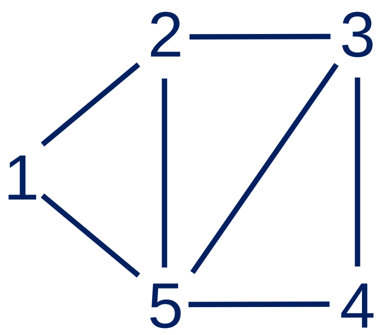


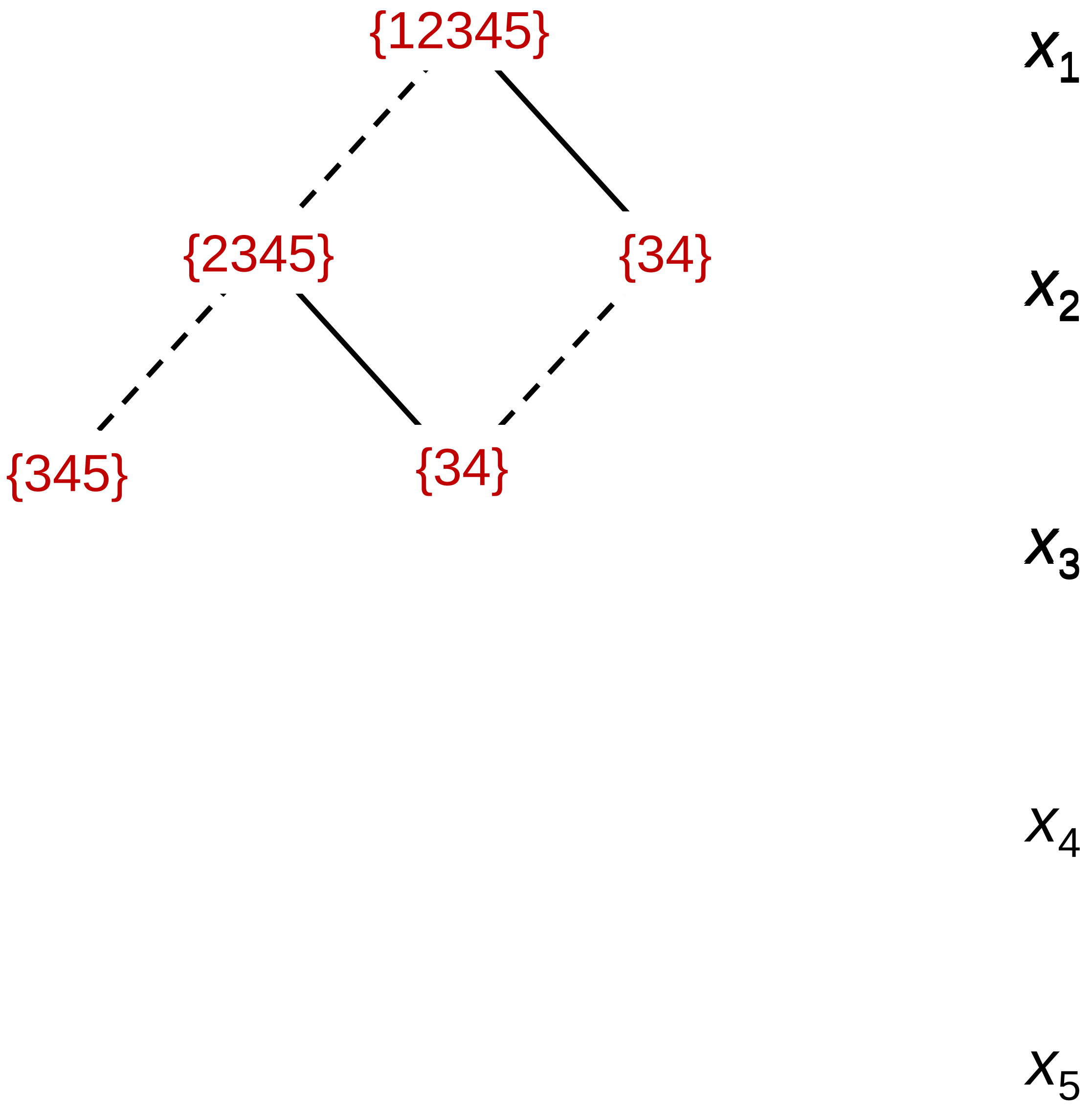
To build **relaxed** BDD, merge some additional nodes as we go along



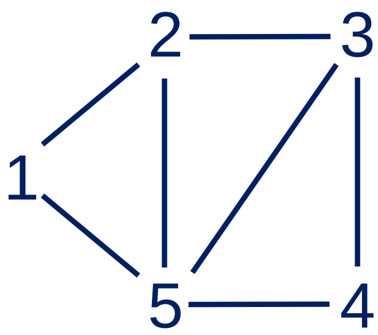


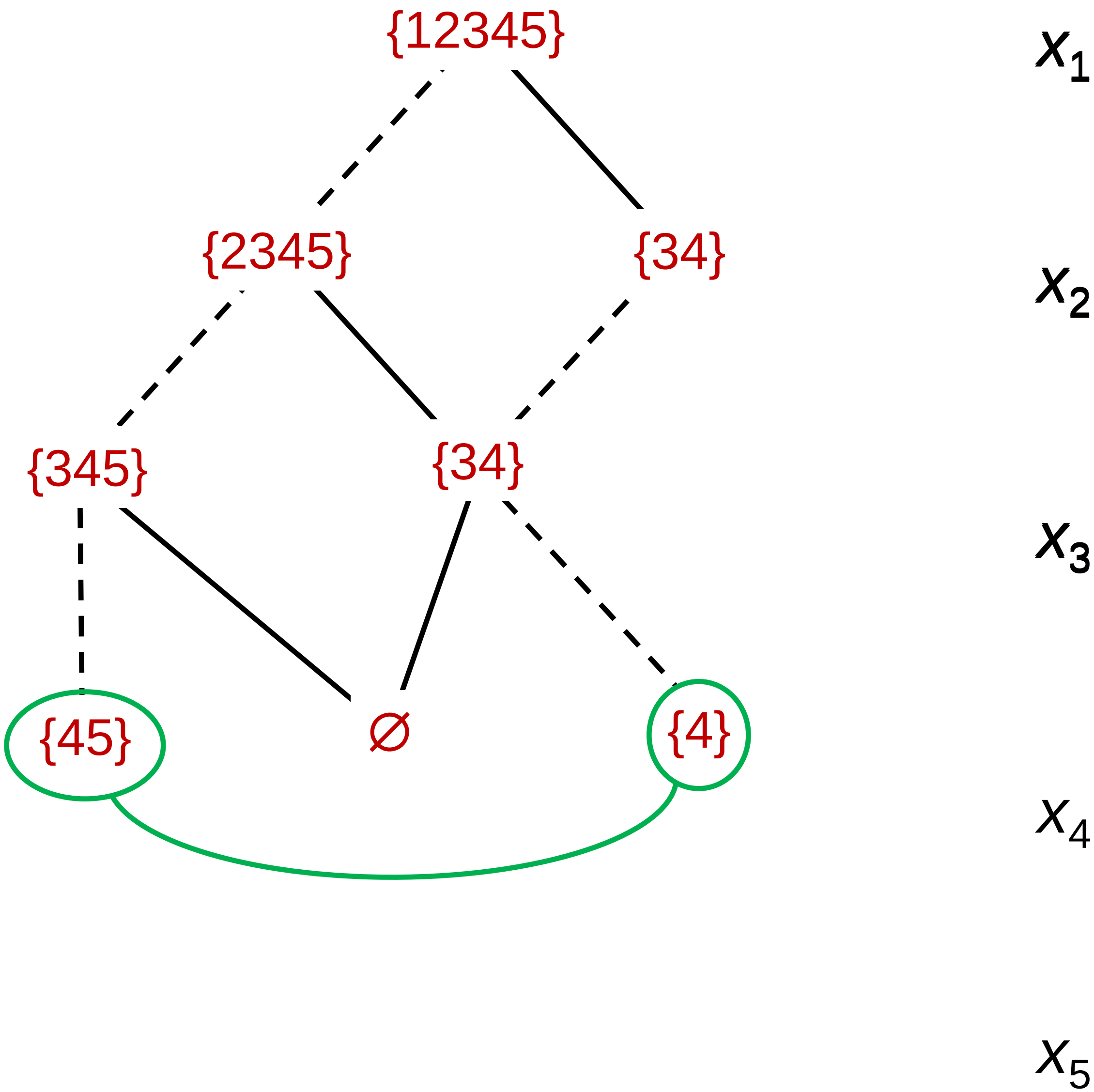
To build **relaxed** BDD, merge some additional nodes as we go along



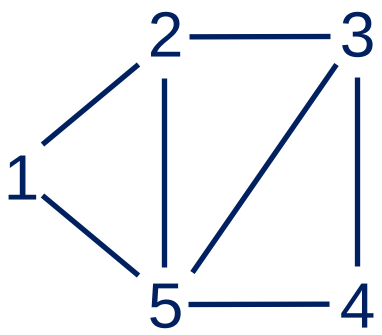


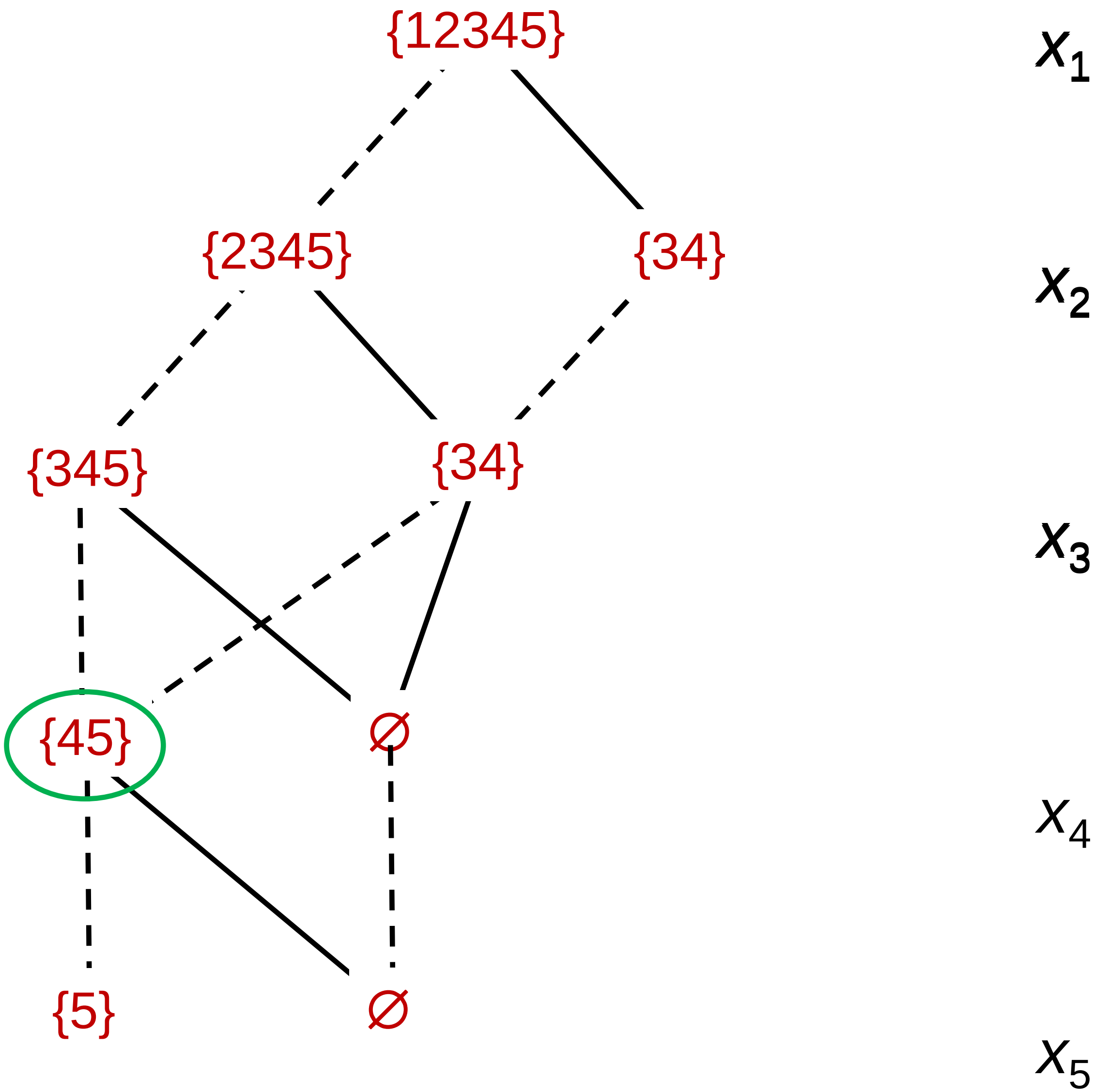
To build **relaxed** BDD, merge some additional nodes as we go along



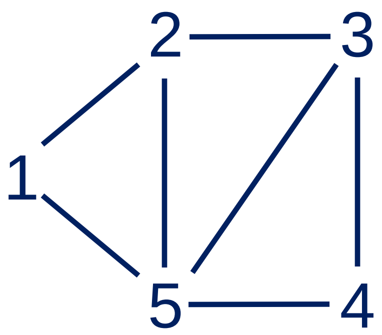


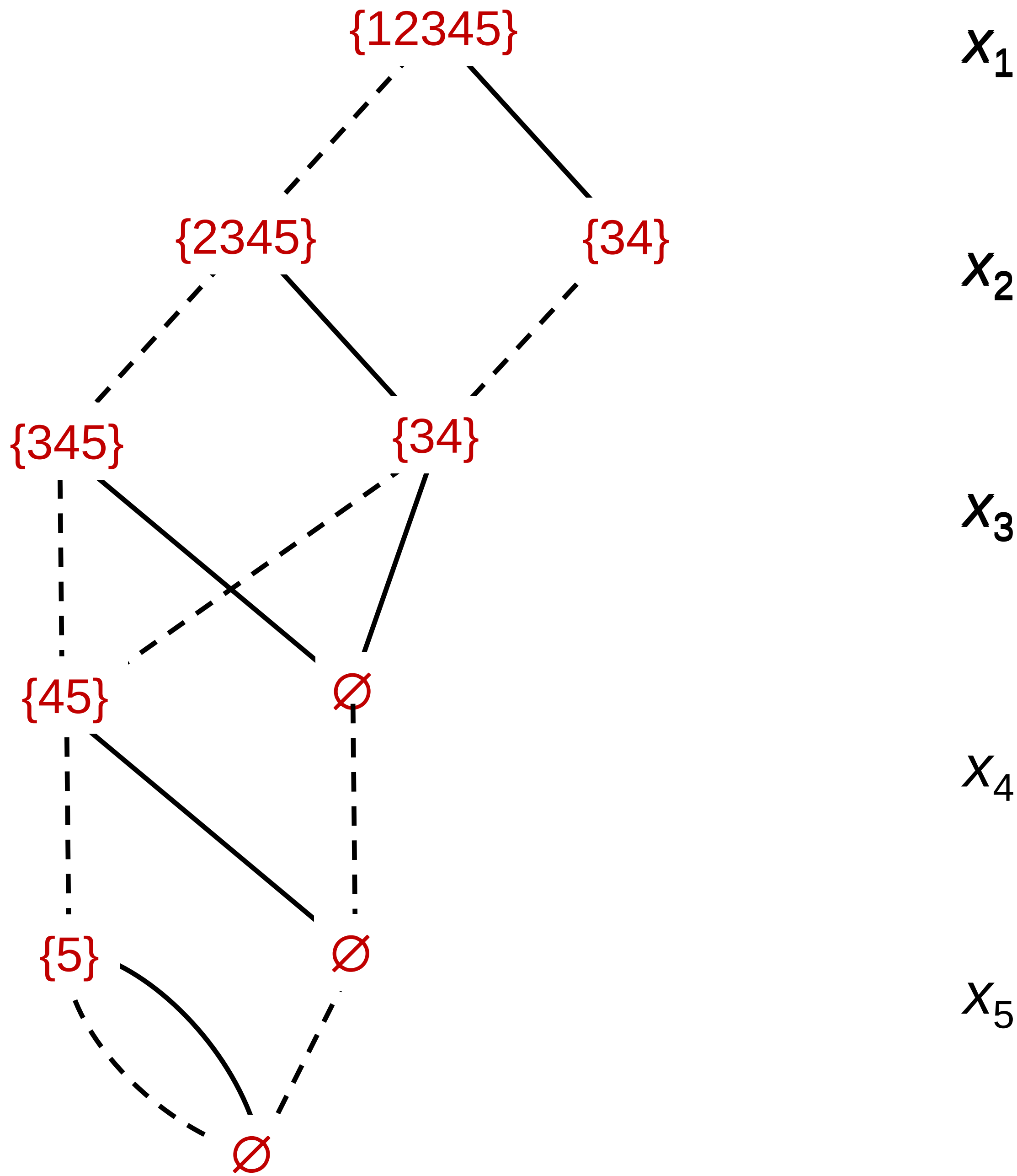
To build **relaxed** BDD, merge some additional nodes as we go along





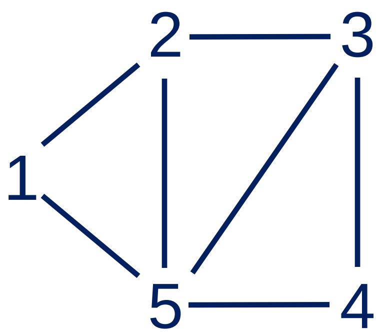
To build **relaxed** BDD, merge some additional nodes as we go along





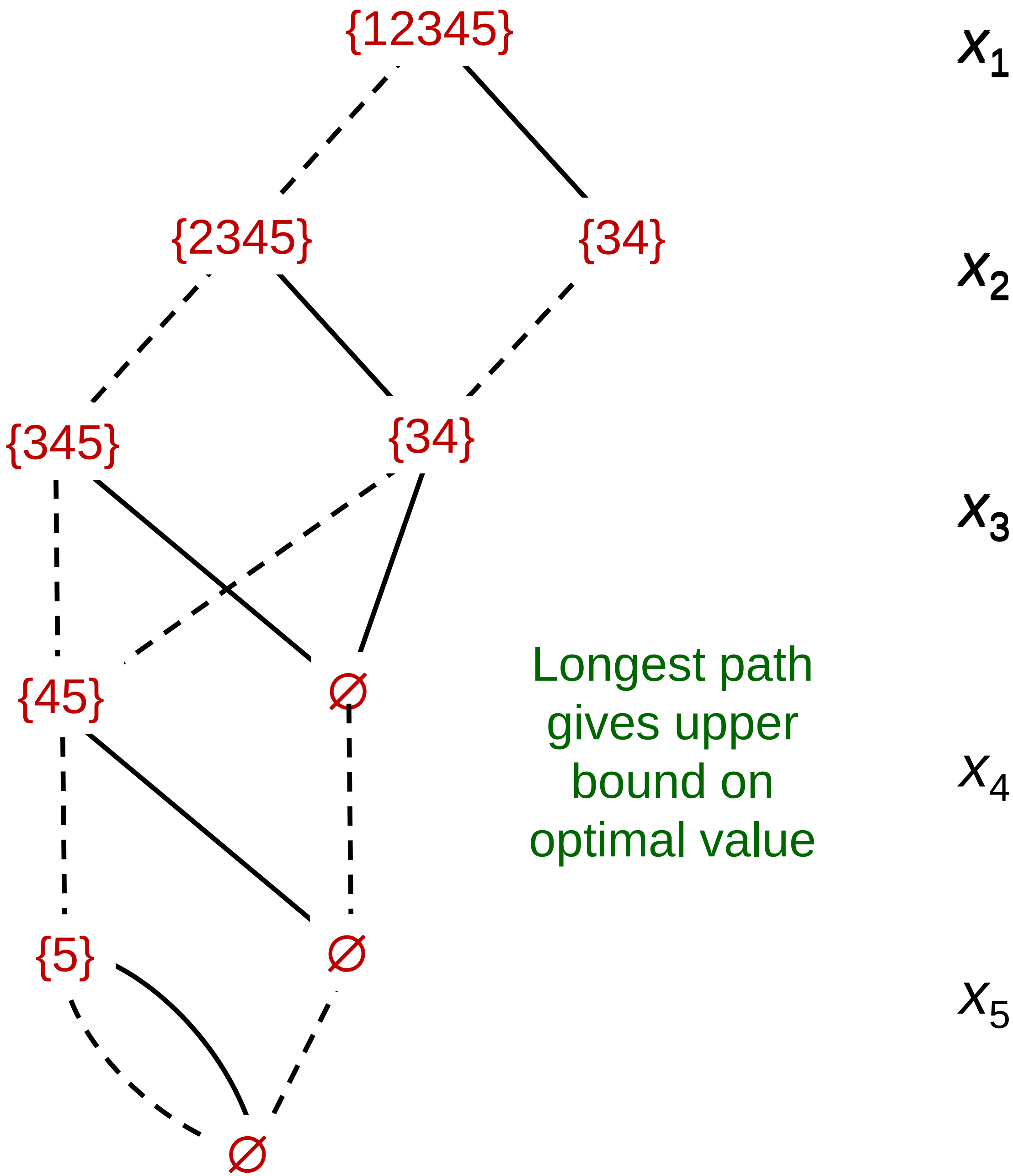
Width = 2

Represents 11 solutions, including 9 feasible solutions



Width = 2

Represents 11 solutions, including 9 feasible solutions



Longest path gives upper bound on optimal value

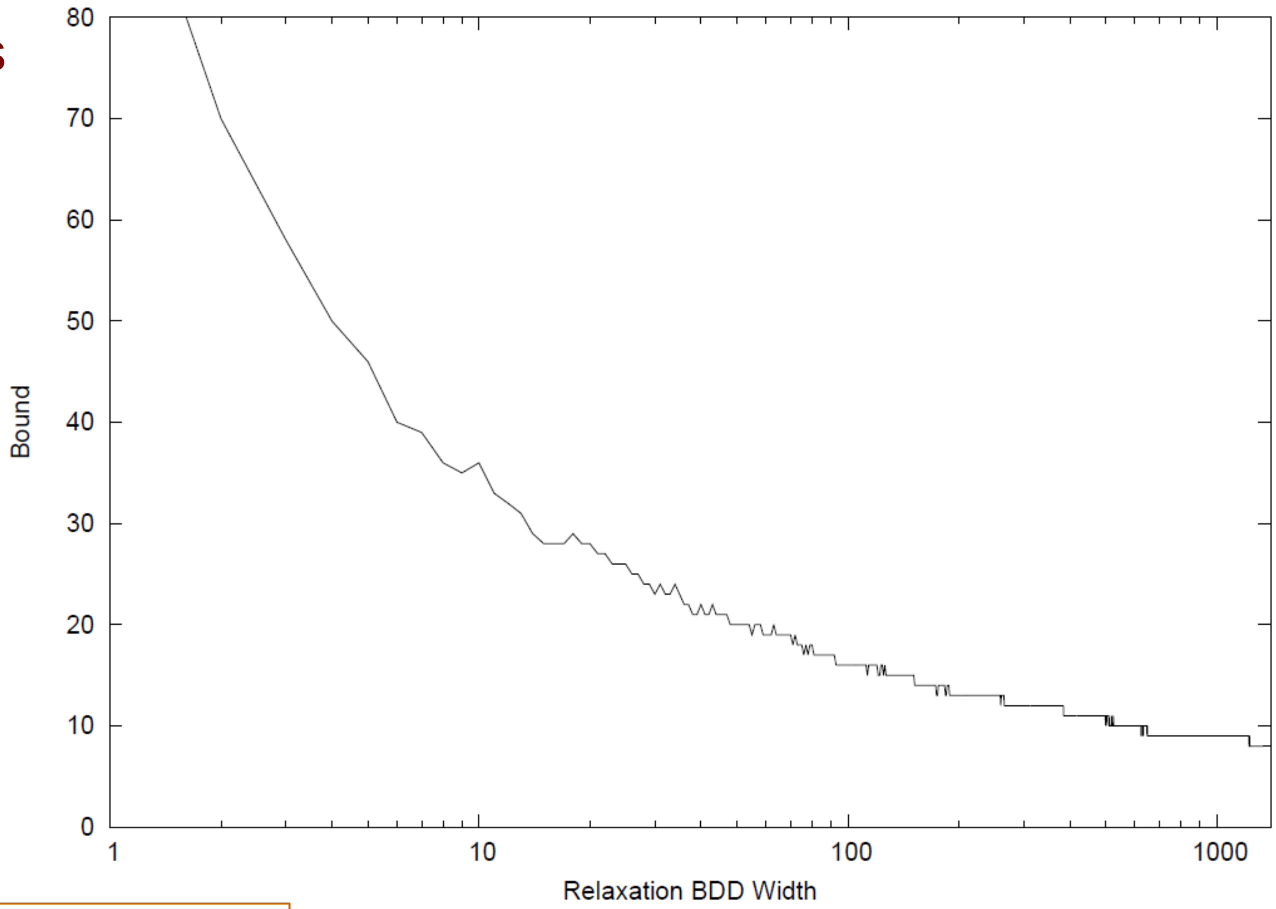
Decision Diagrams

- 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)

Decision Diagrams

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



Bergman, Ciré, van Hoeve, JH (2013)

Decision Diagrams

- 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.

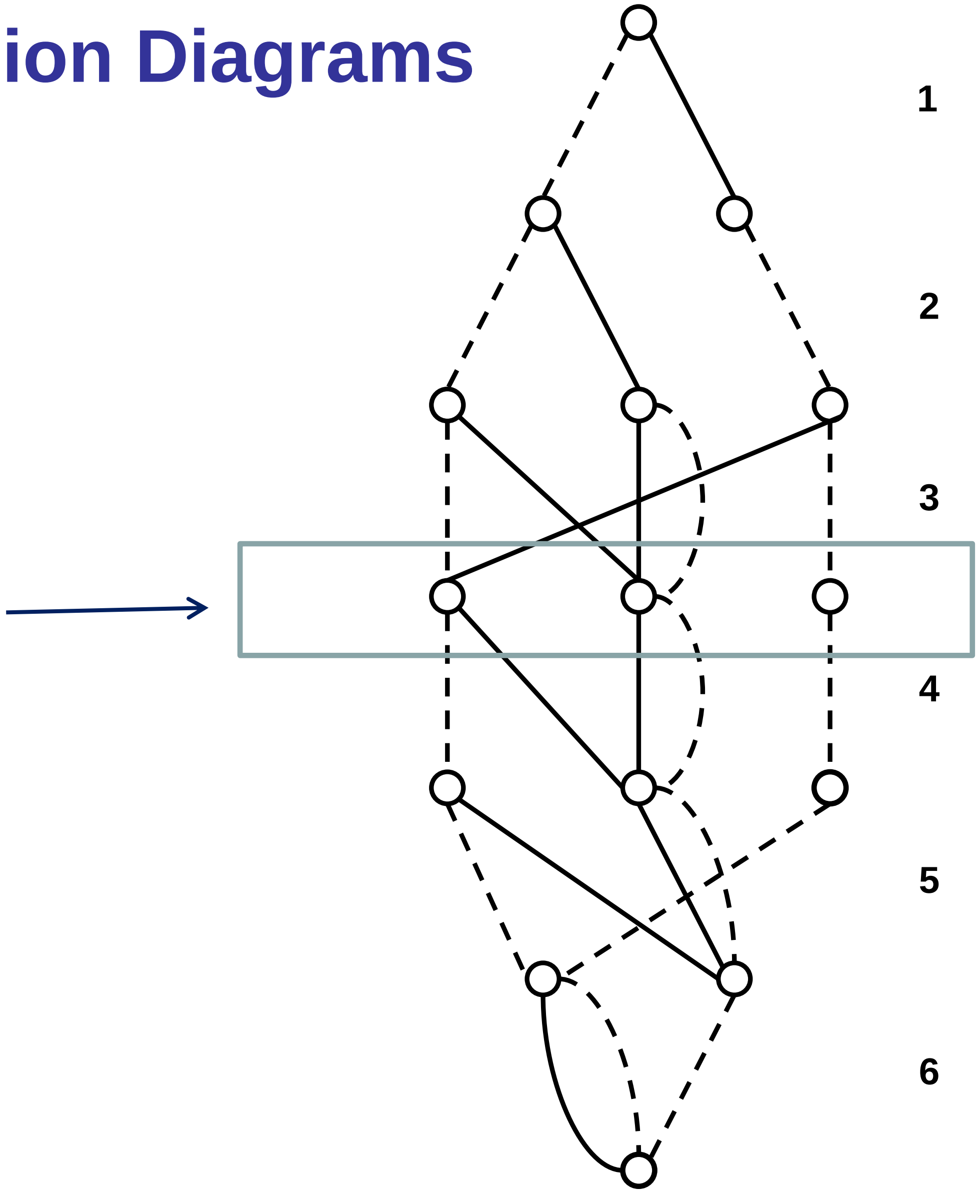
Decision Diagrams

- 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**.

Decision Diagrams

Branching in a relaxed decision diagram

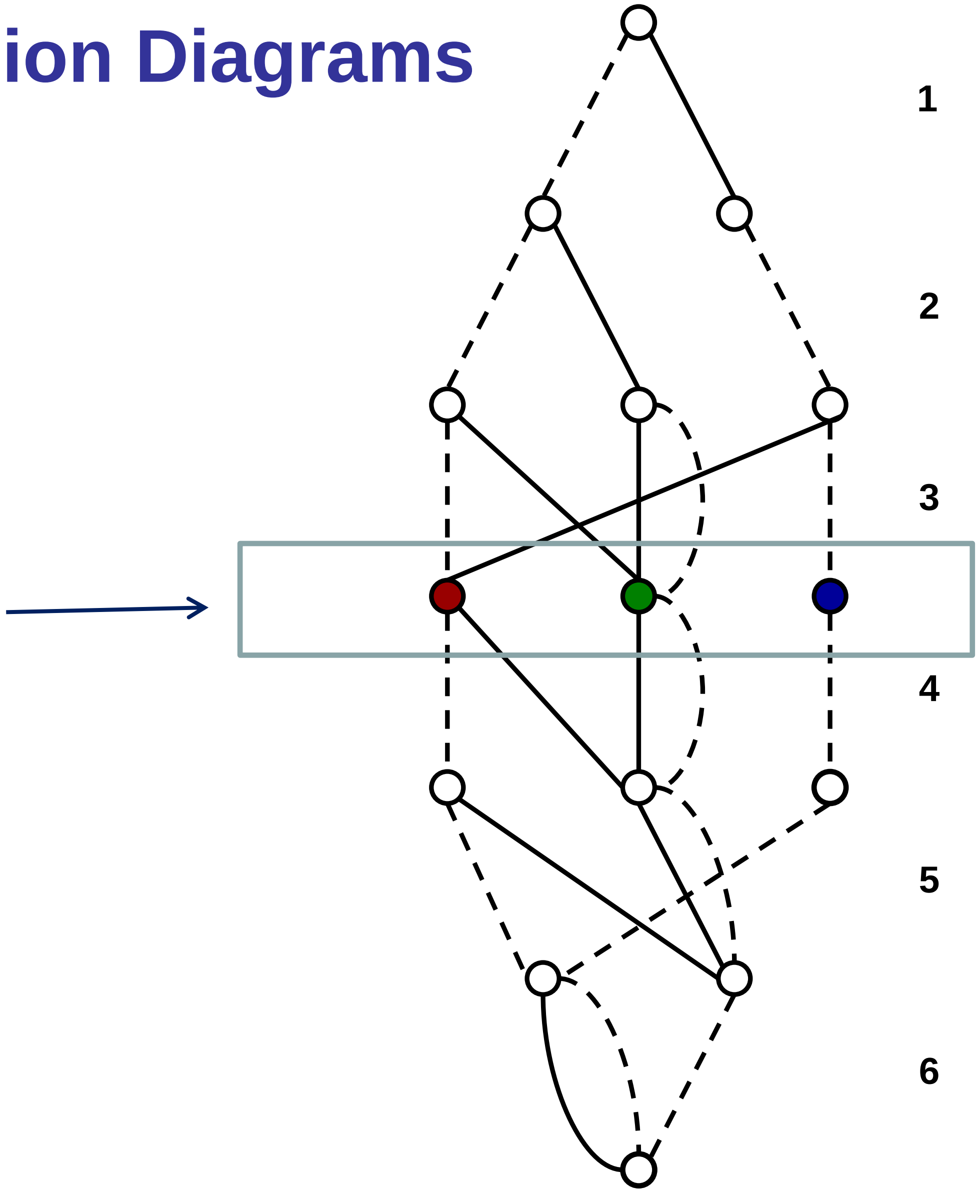
Diagram is exact down to here



Decision Diagrams

Branching in a relaxed decision diagram

Branch on nodes in this layer

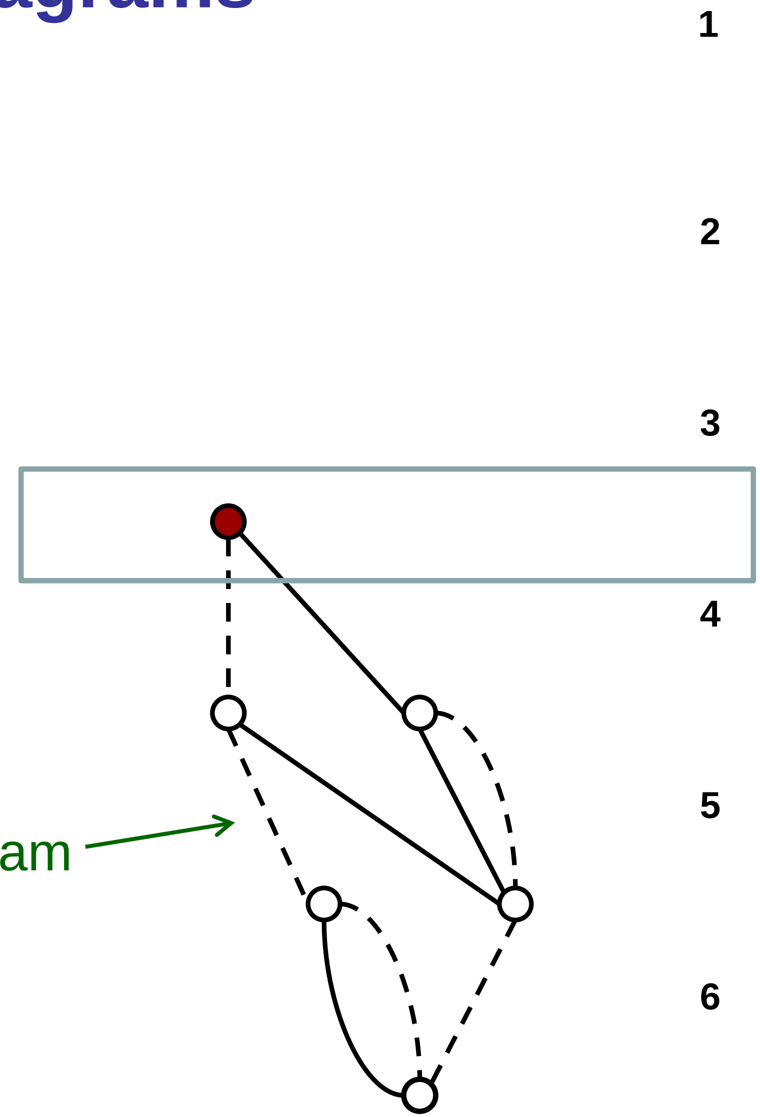


Decision Diagrams

Branching in a relaxed decision diagram

First branch

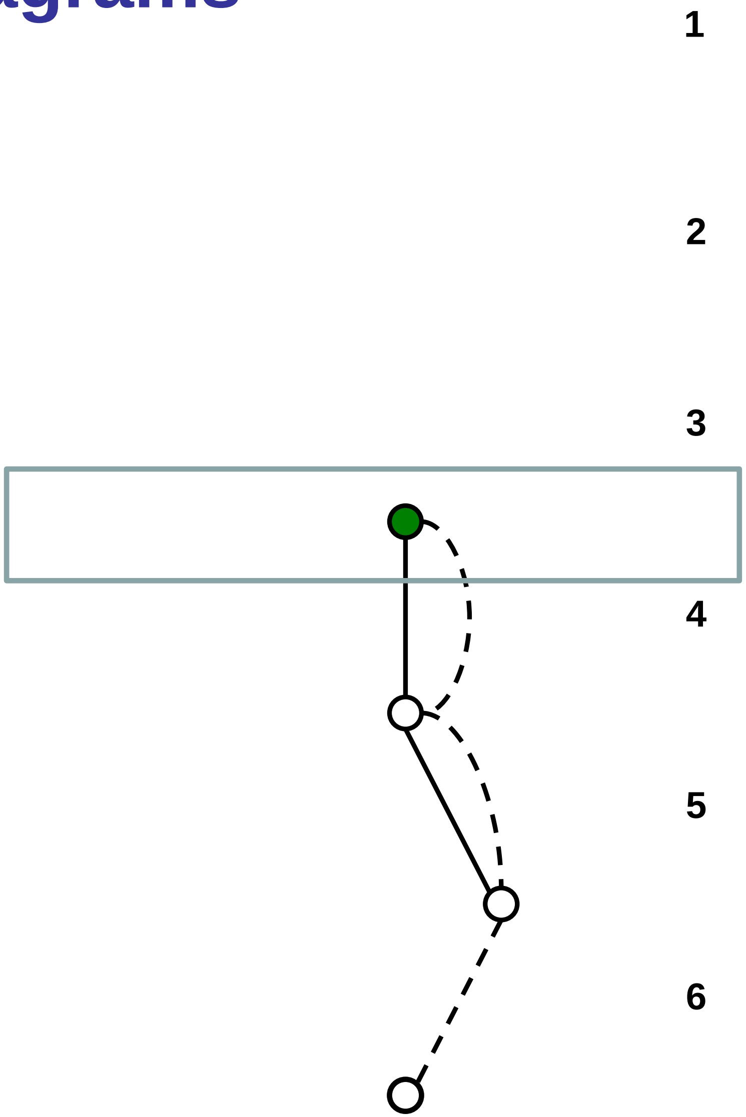
New relaxed decision diagram



Decision Diagrams

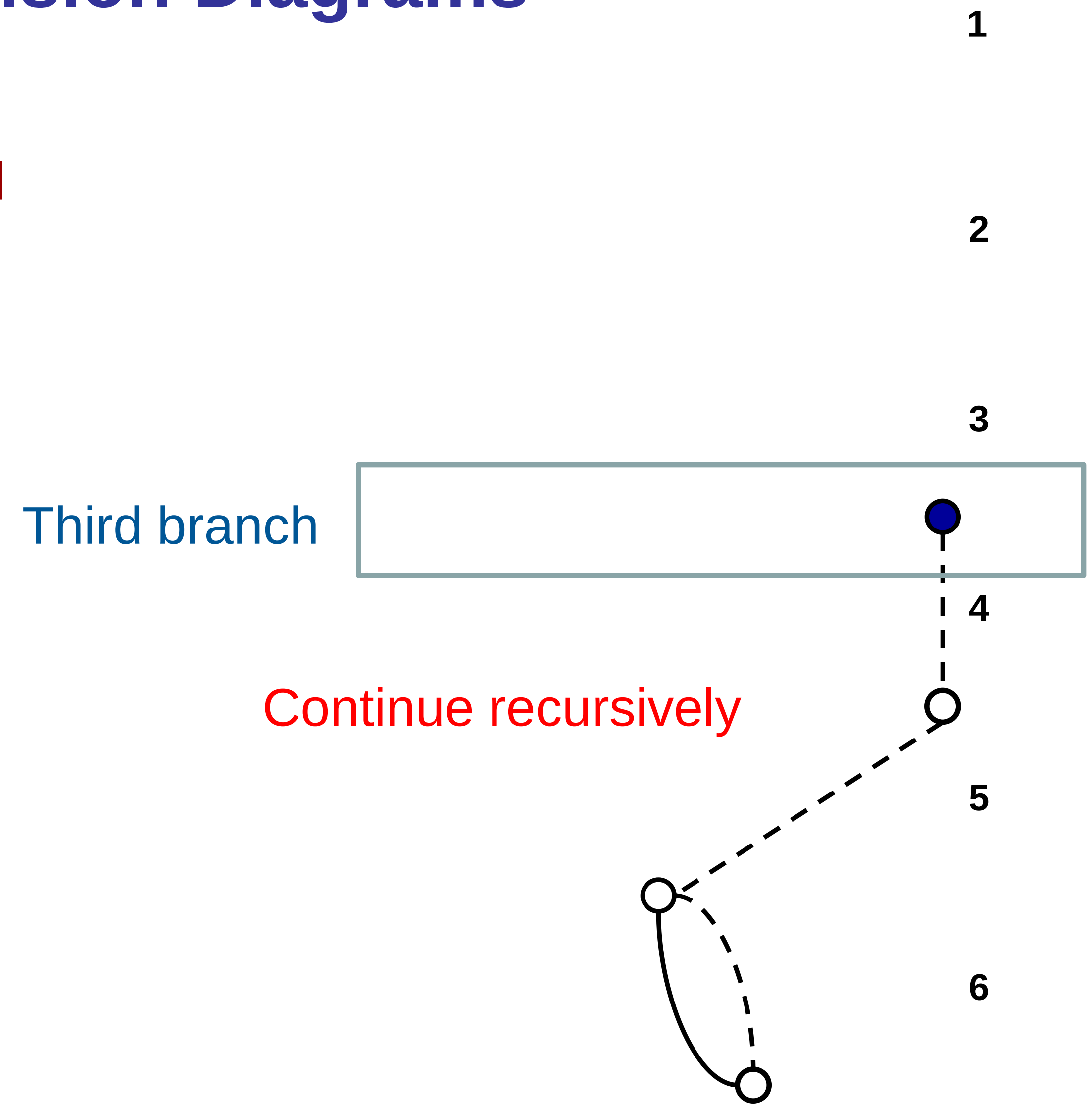
Branching in a relaxed decision diagram

Second branch



Decision Diagrams

Branching in a relaxed decision diagram



Decision Diagrams

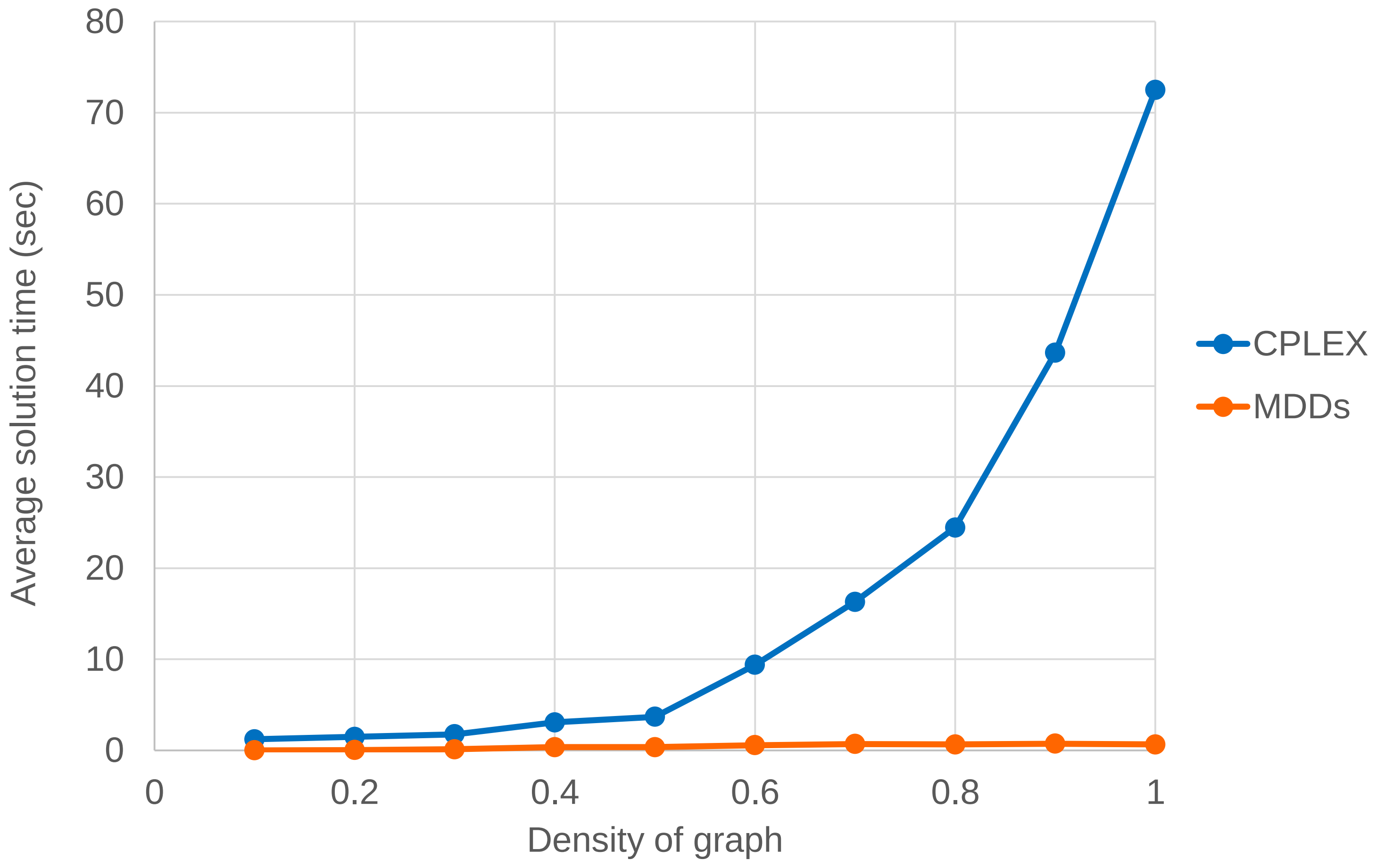
- Computational results…
  - Applied to stable set, max cut, max 2-SAT.
    - Superior to commercial MIP solver (CPLEX) on most instances.
    - Even though the problems have **natural MIP models**.
    - Obtained best known solution on some max cut instances.
  - Slightly slower than MIP on stable set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

Decision Diagrams

Max cut
on a graph

Avg. solution time
vs
graph density

30 vertices

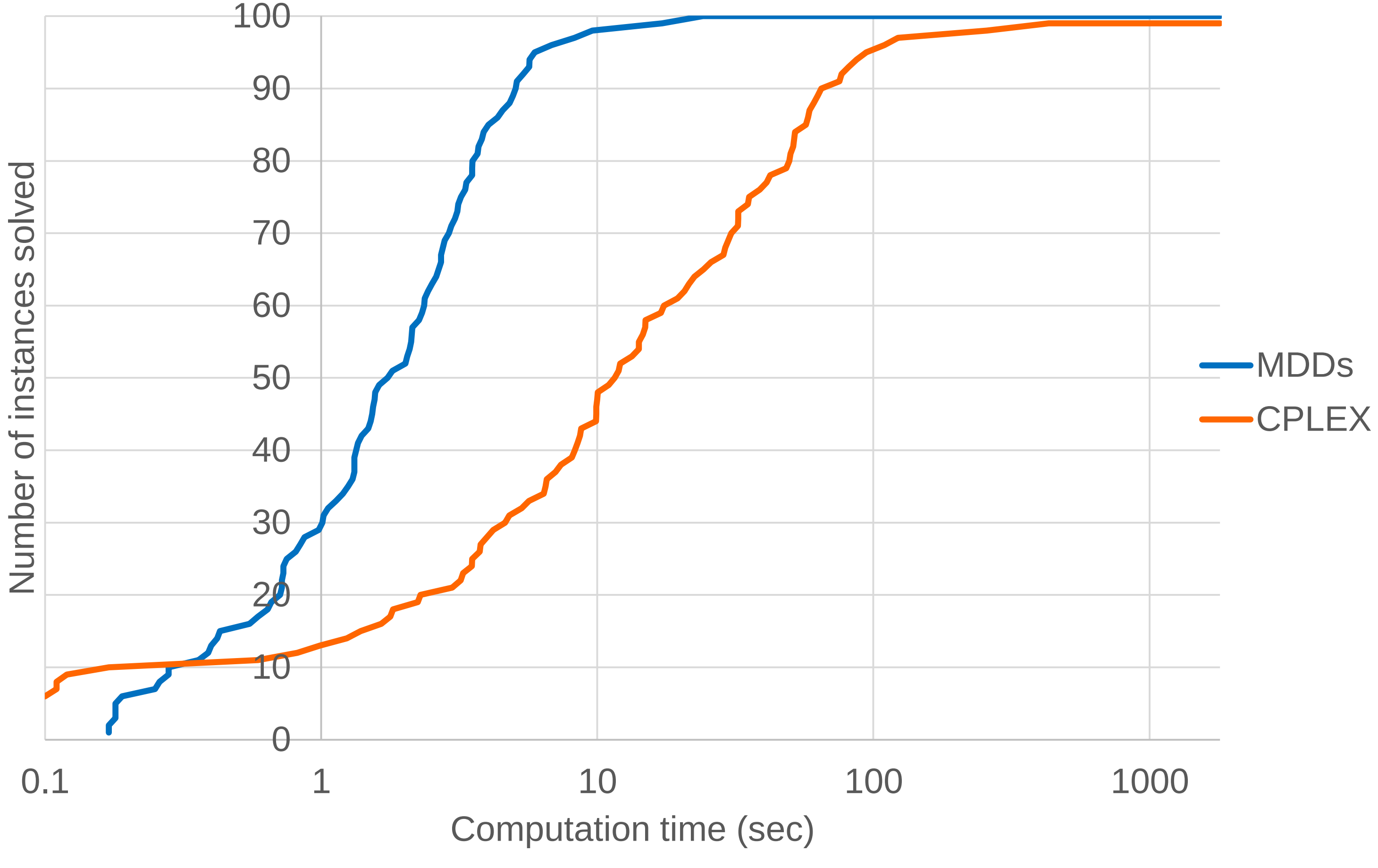


Decision Diagrams

Max 2-SAT

Performance profile

30 variables

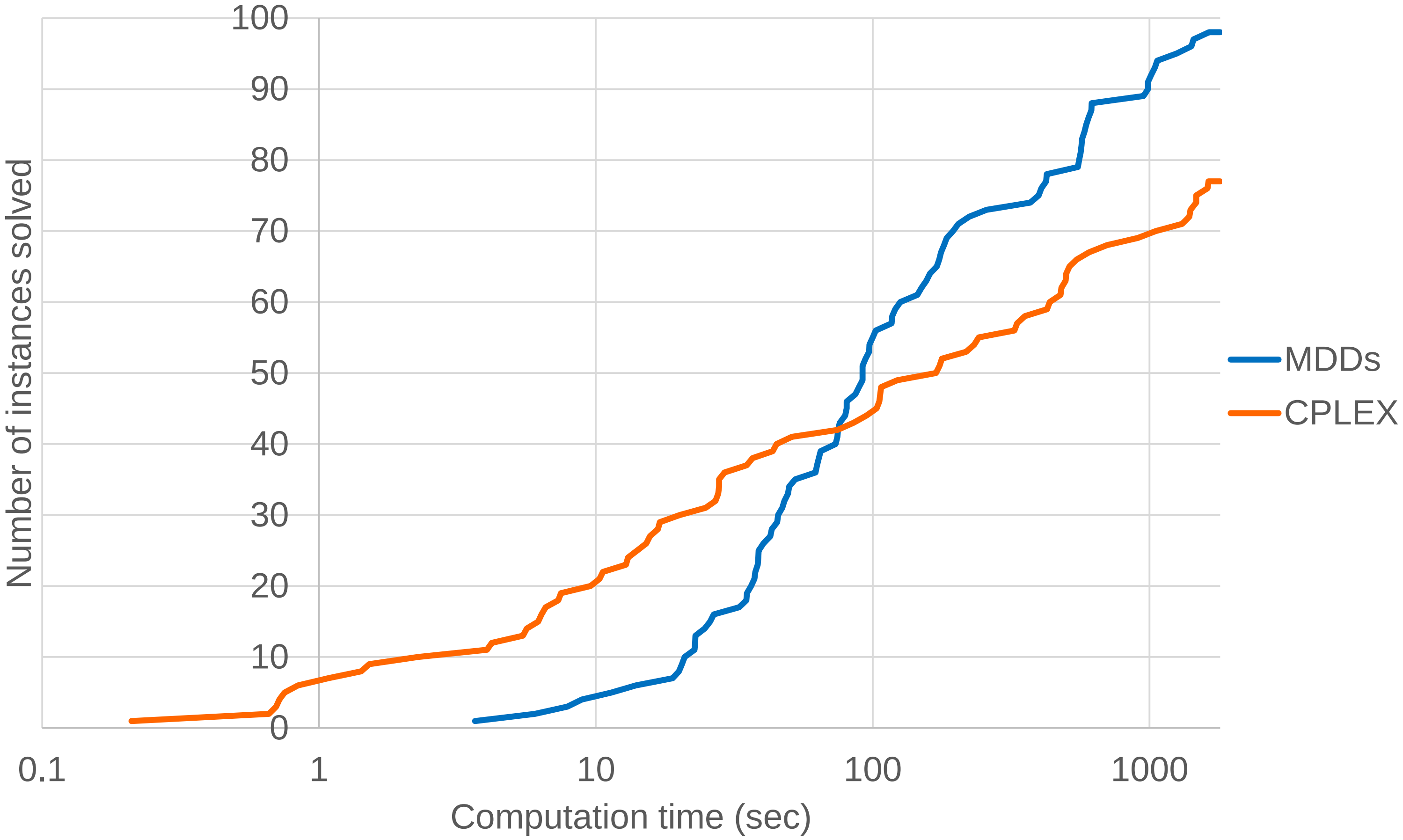


Decision Diagrams

Max 2-SAT

Performance profile

40 variables

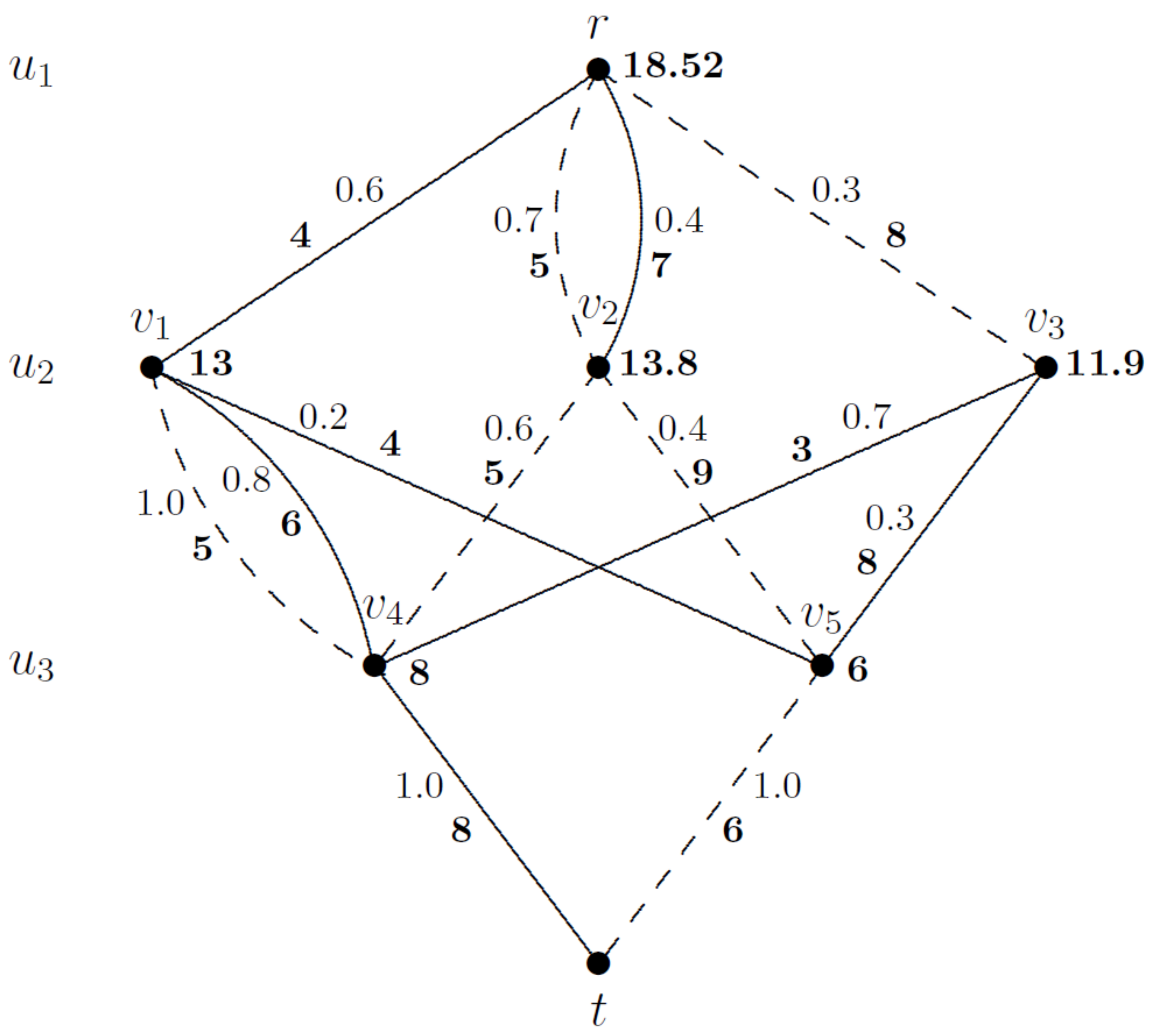


Decision Diagrams

- 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.

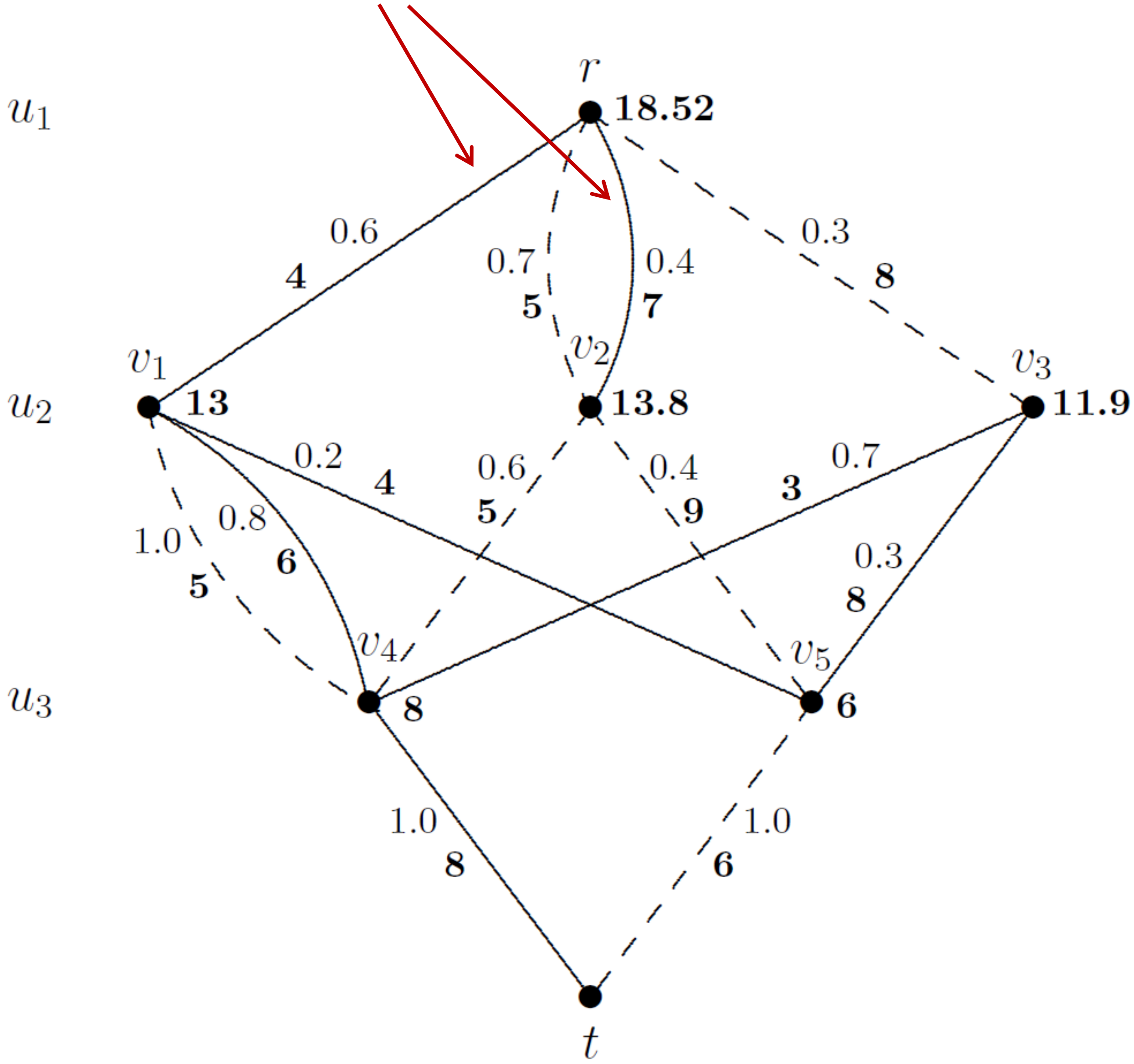
Stochastic Decision Diagram

- Each decision (control) has probabilistic results.
  - Several possible outcomes.
  - A solution is a **policy** (not a path).
    - Specifies control **at each node** of DD.

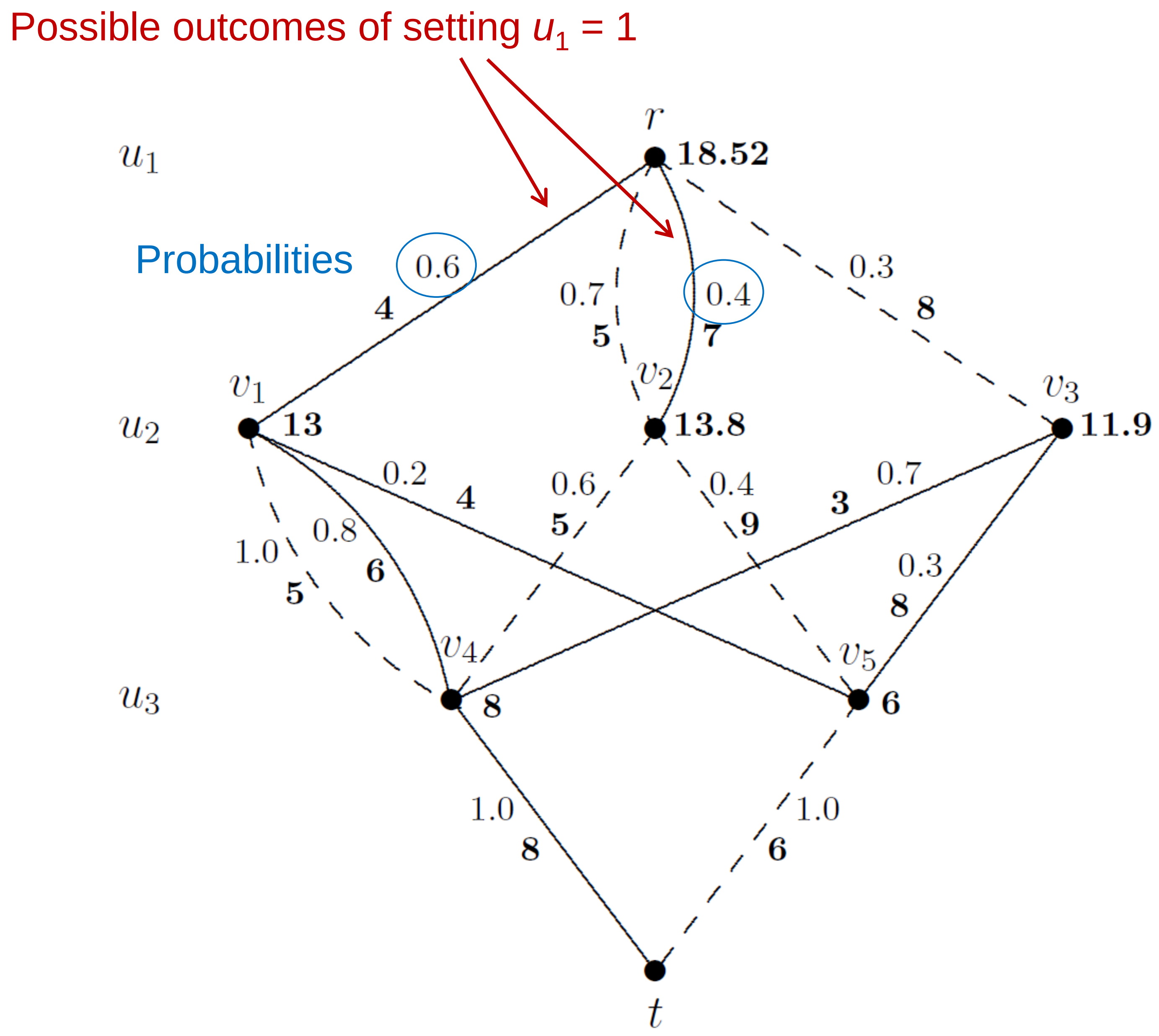


Stochastic decision diagram (SDD)

Possible outcomes of setting $u_1 = 1$



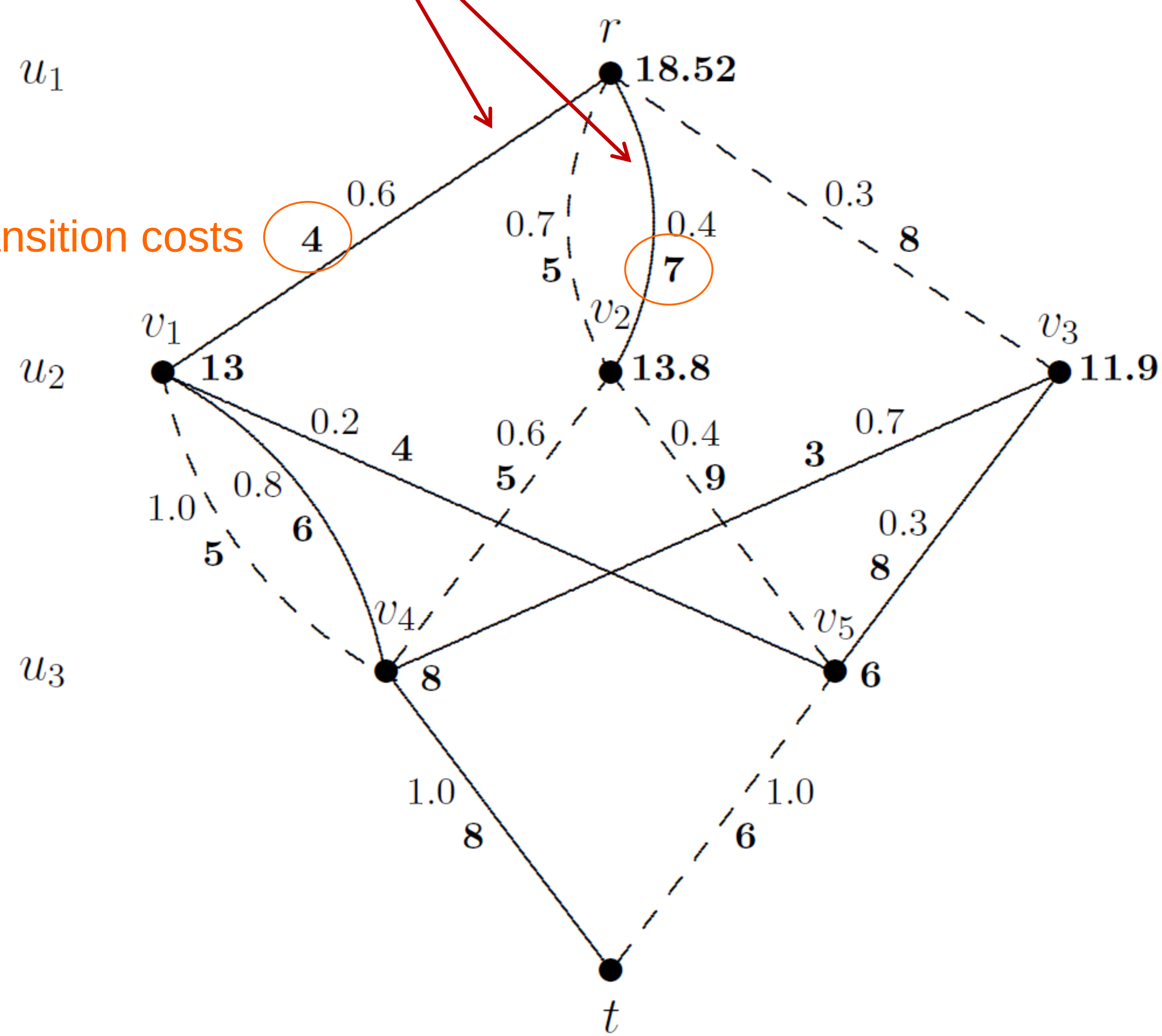
Stochastic decision diagram (SDD)



Stochastic decision diagram (SDD)

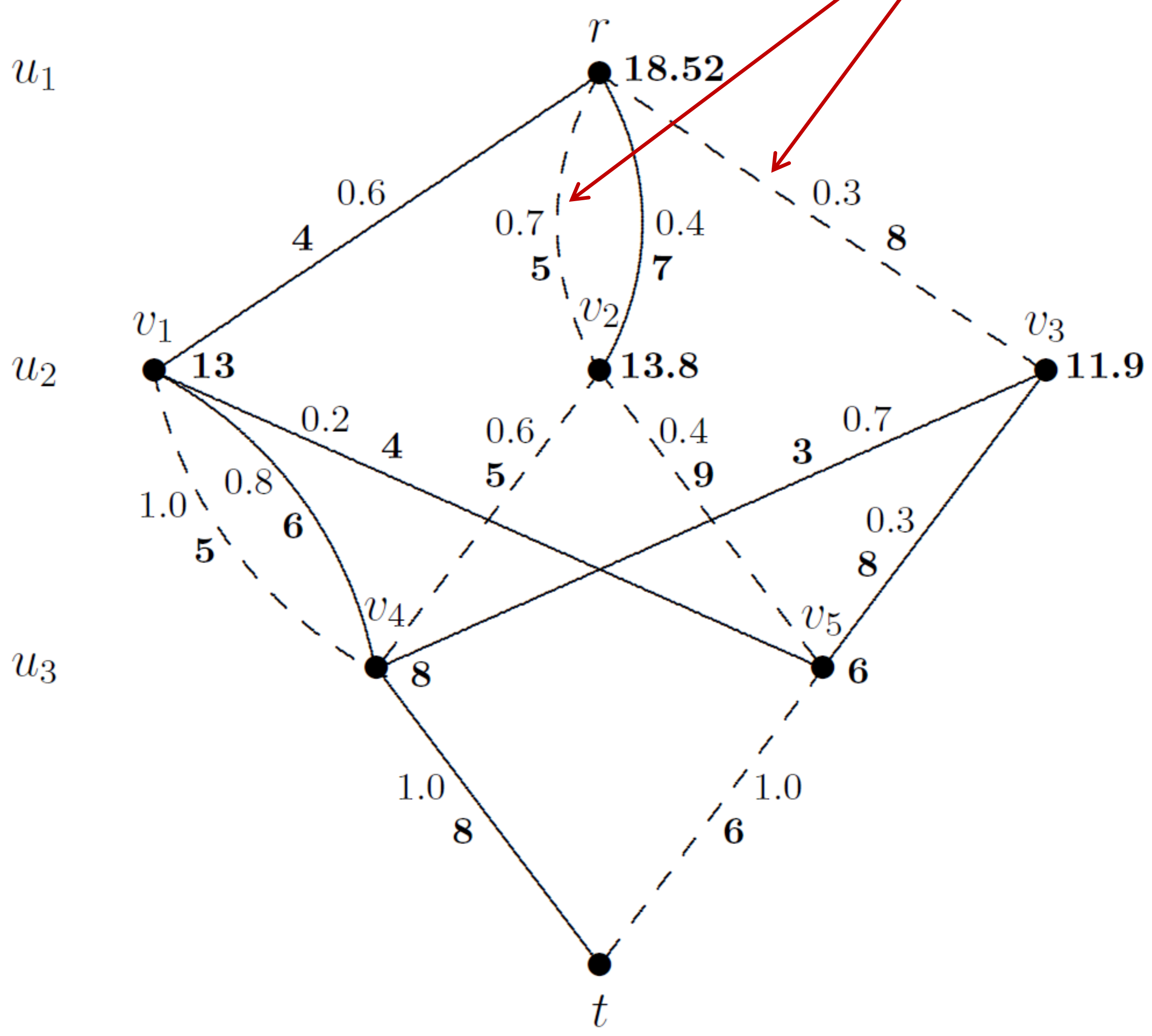
Possible outcomes of setting $u_1 = 1$

Transition costs

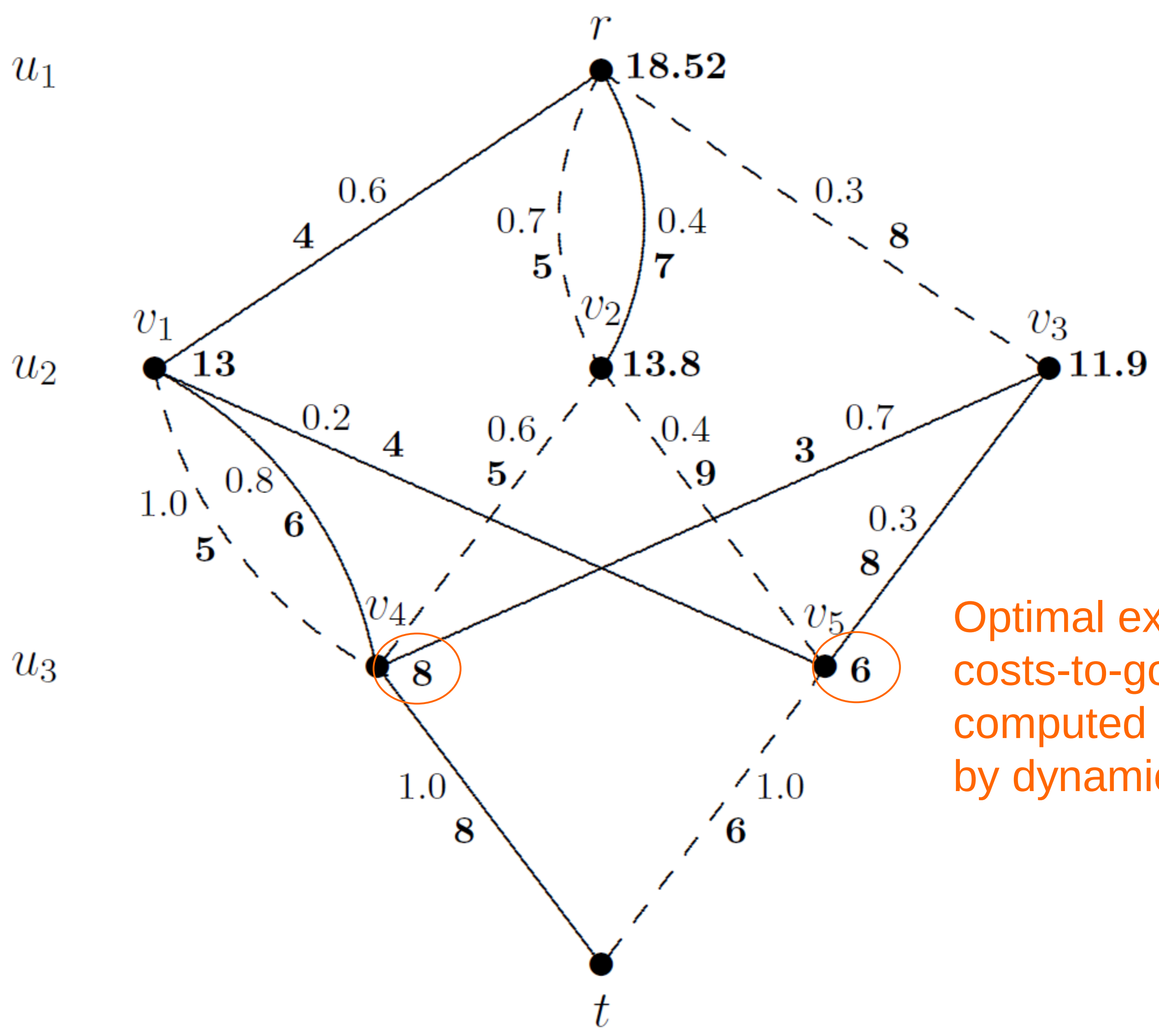


Stochastic decision diagram (SDD)

Possible outcomes of setting $u_1 = 0$

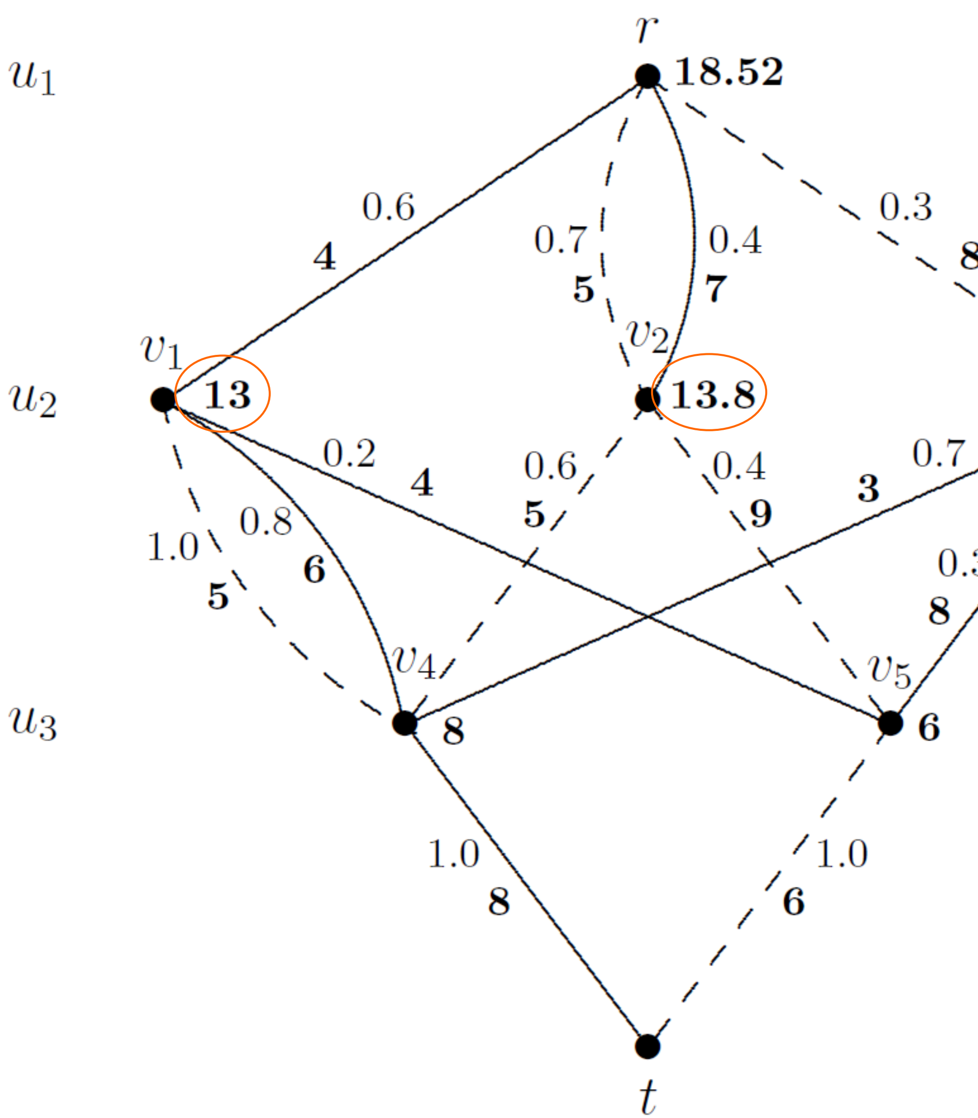


Stochastic decision diagram (SDD)



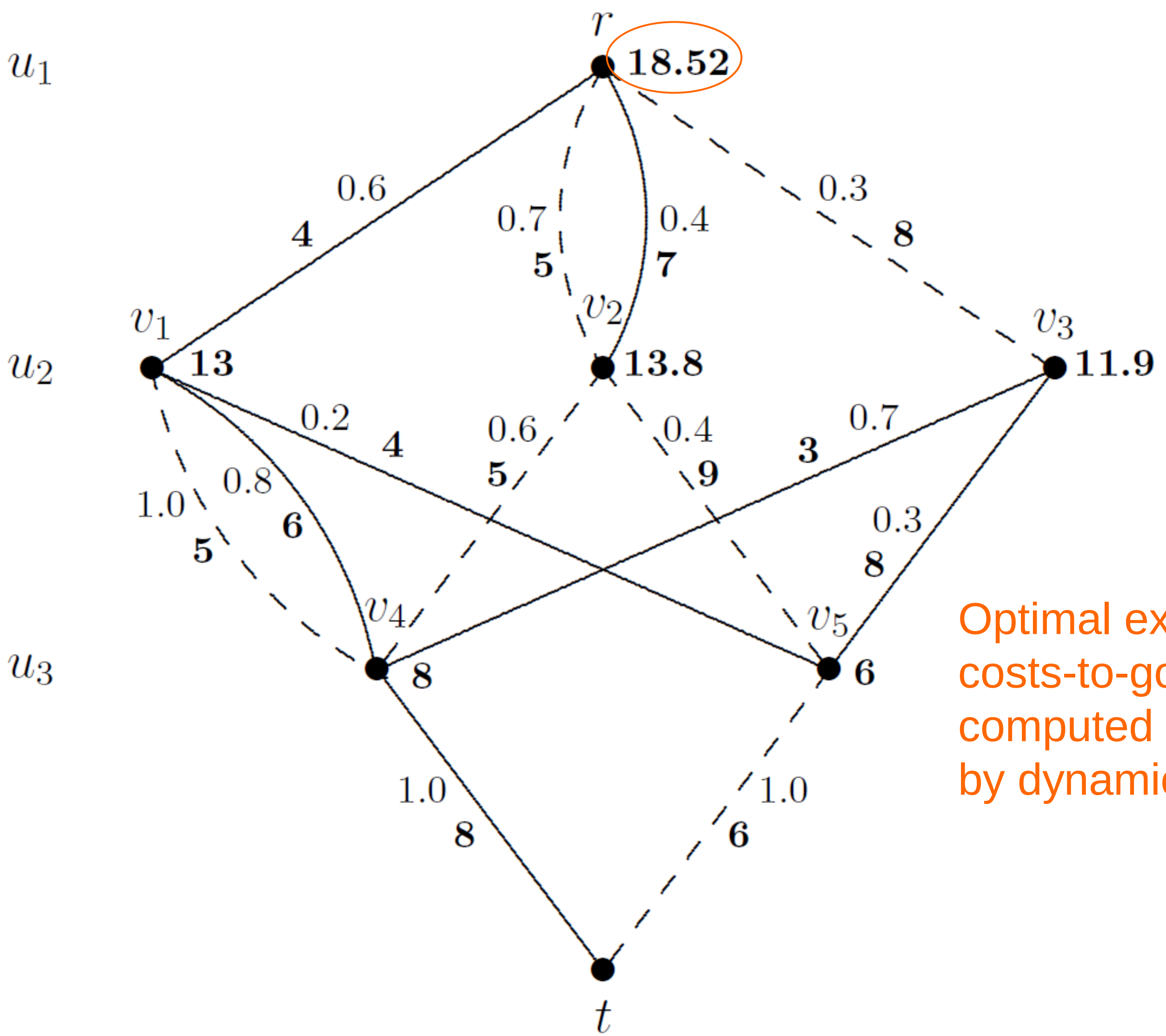
Optimal expected costs-to-go, computed recursively by dynamic programming

Stochastic decision diagram (SDD)



Optimal expected costs-to-go, computed recursively by dynamic programming

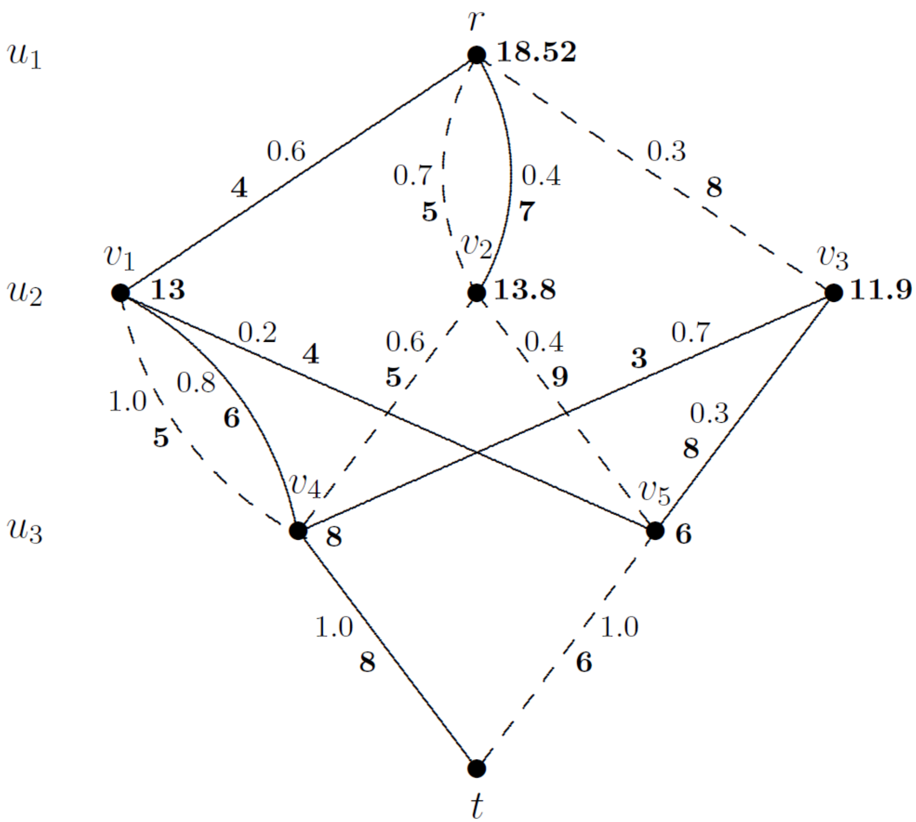
Stochastic decision diagram (SDD)



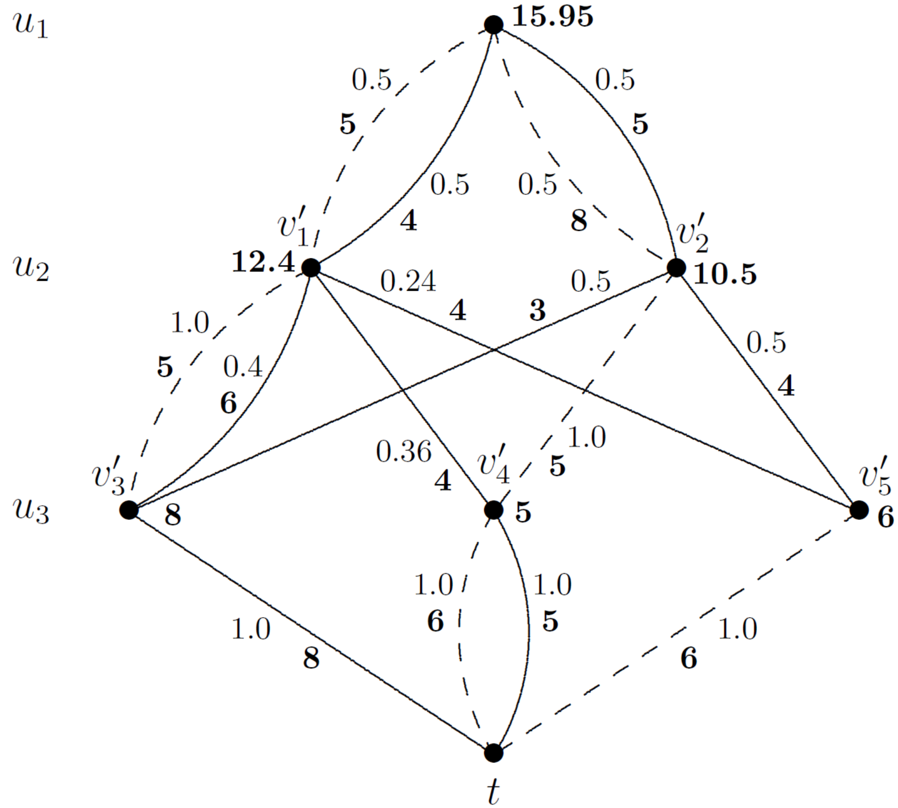
Optimal expected costs-to-go, computed recursively by dynamic programming

Stochastic decision diagram (SDD)

A possible relaxation of the SDD

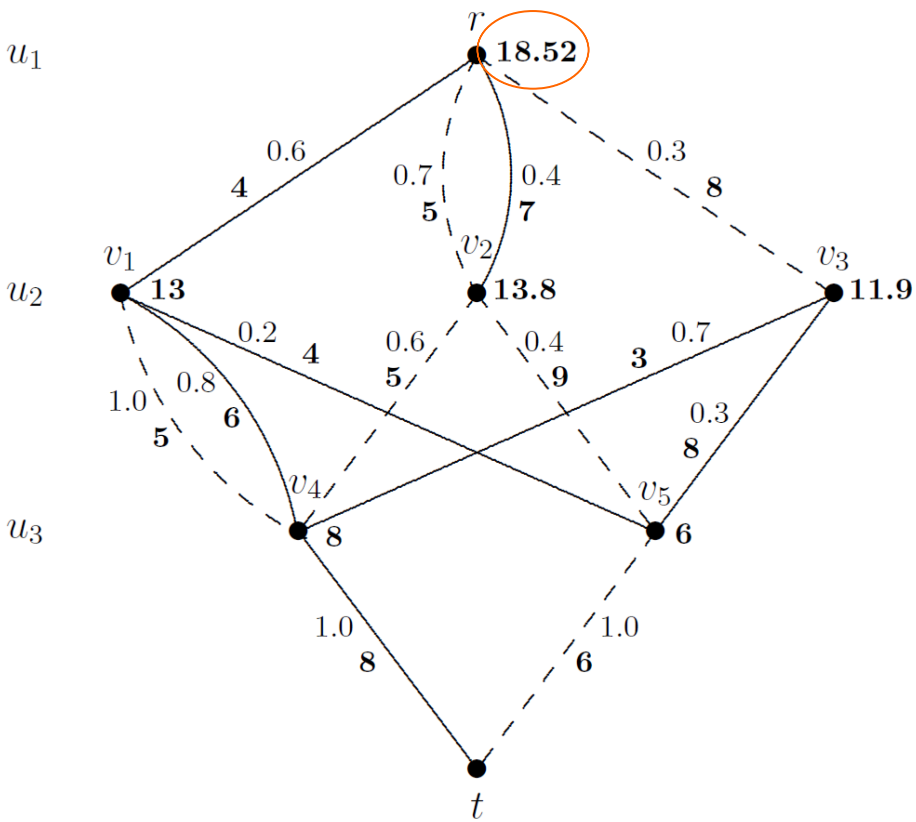


Original SDD

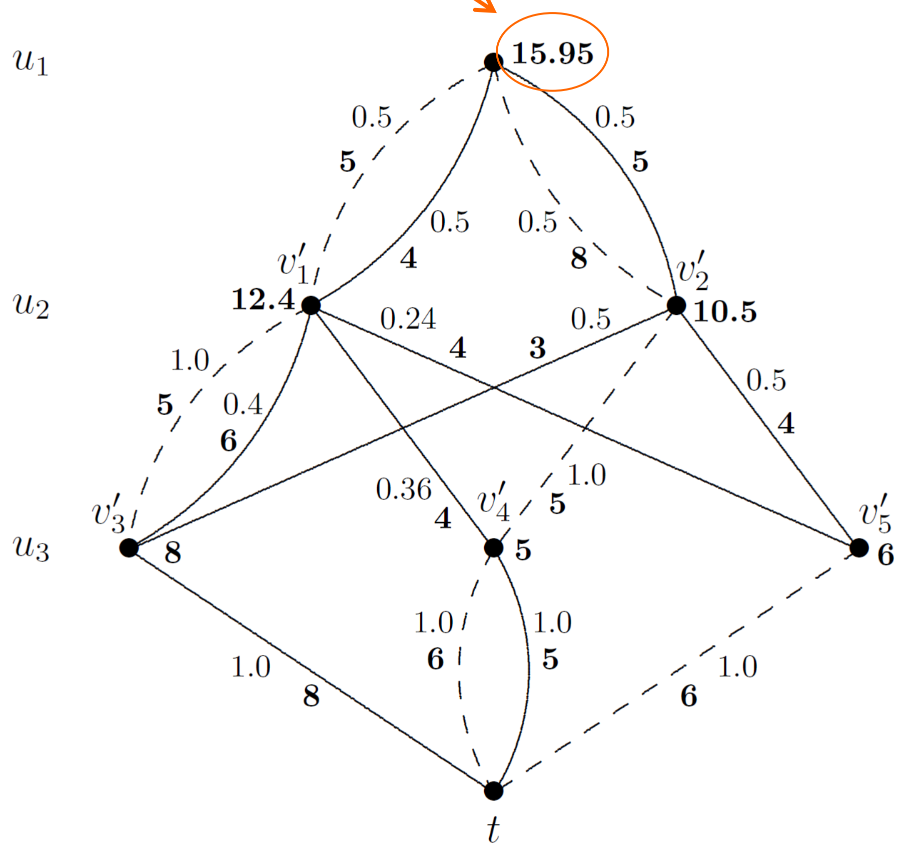


Relaxed SDD

Provides lower bound on optimal cost



Original SDD



Relaxed SDD

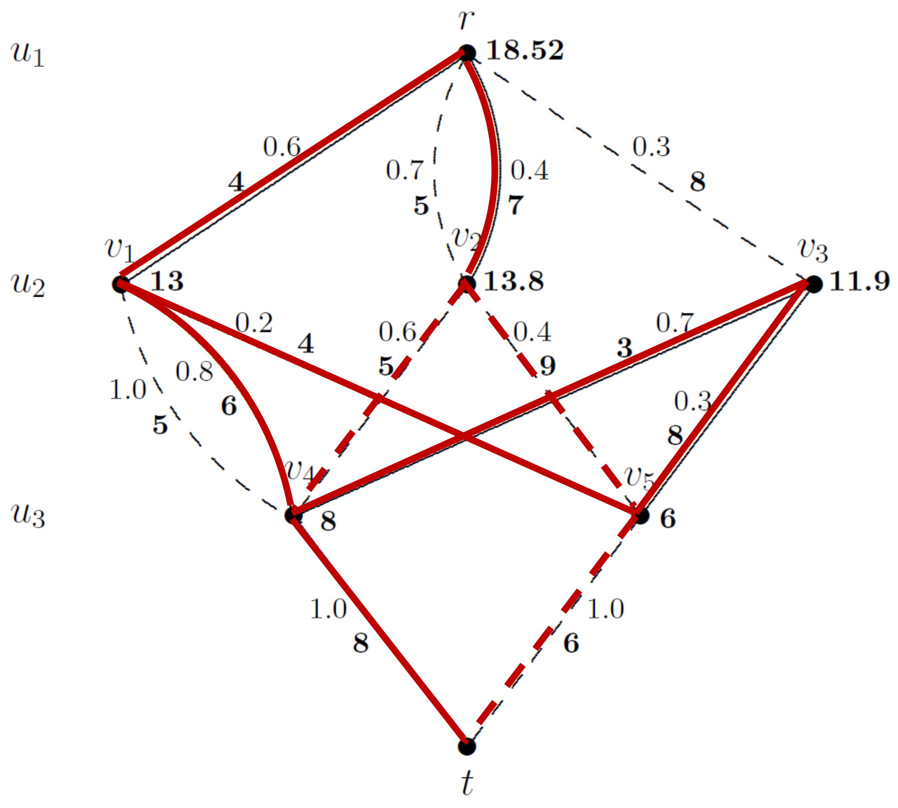
Relaxed SDD

- Not obvious how to define a relaxed SDD.
  - We can't say that every solution of the original is a solution of the relaxation.
  - A solution is a **policy** that defines control at each node.
    - The relaxed DD may have a **completely different** configuration of nodes.
    - …as in the example.

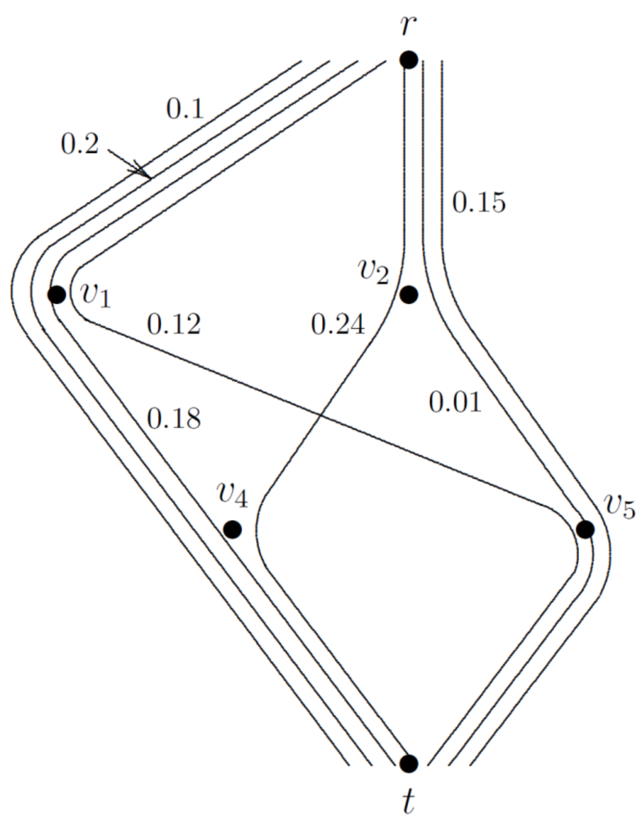
Relaxed SDD

- We need a concept of **flow-path decomposition**
  - …for a given policy.
- A flow-path decomposition is a set of flows from top to bottom such that:
  - Sum of flows is 1.
  - Sum of flows on a given arc is probability of traversing that arc.

A possible flow-path decomposition of the original SDD

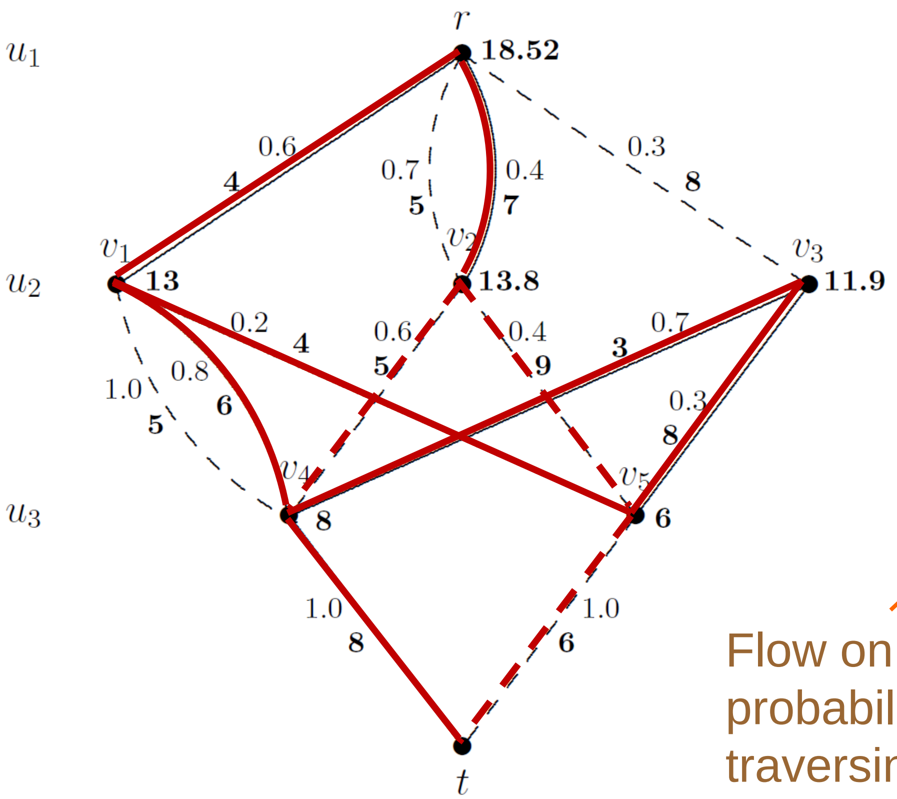


Original SDD



Flow-path decomposition for policy in red

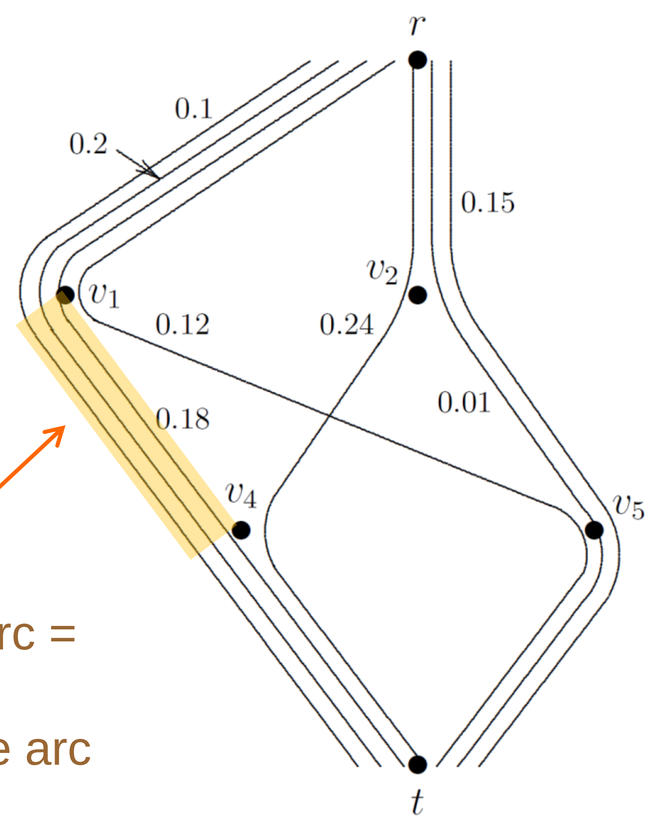
A possible flow-path decomposition of the original SDD



Original SDD

Flow on an arc = probability of traversing the arc

0.1 + 0.2 + 0.18 = (0.6)(0.8)

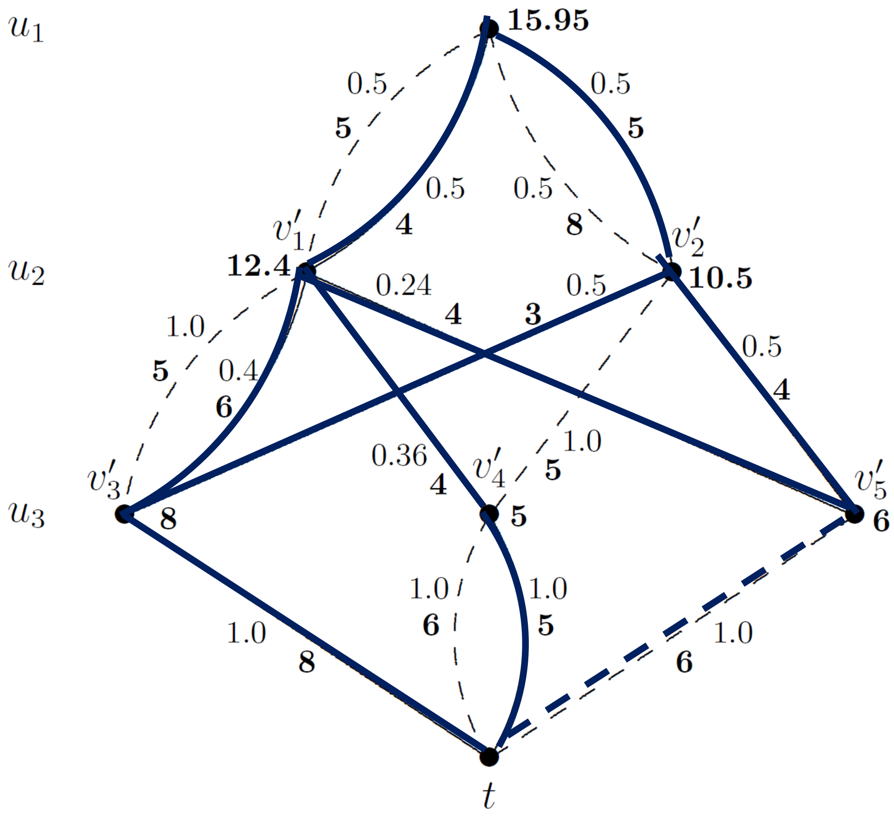


Flow-path decomposition for policy in red

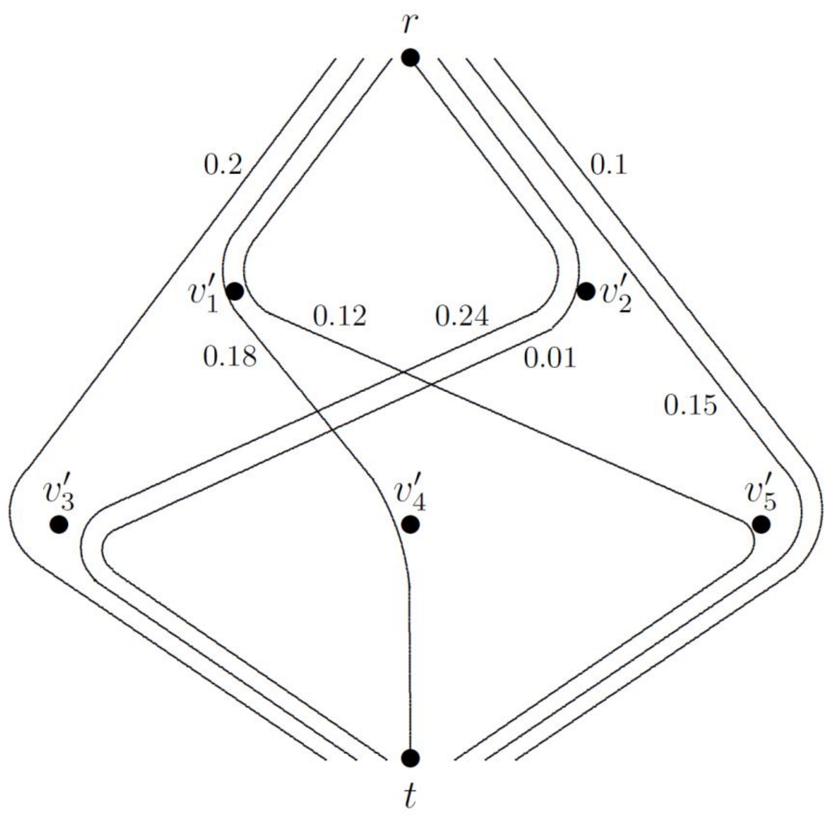
Relaxed SDD

- D' is a relaxation of D if:
  - For every policy u on D:
  - There is a policy u' on D', and flow-path decompositions F, F' on D, D' for policies u, u' such that:
  - There is a 1-1 mapping of flow-paths from F to F' that, on each arc, preserves flow and does not increase cost.

A possible flow-path decomposition of the relaxed SDD

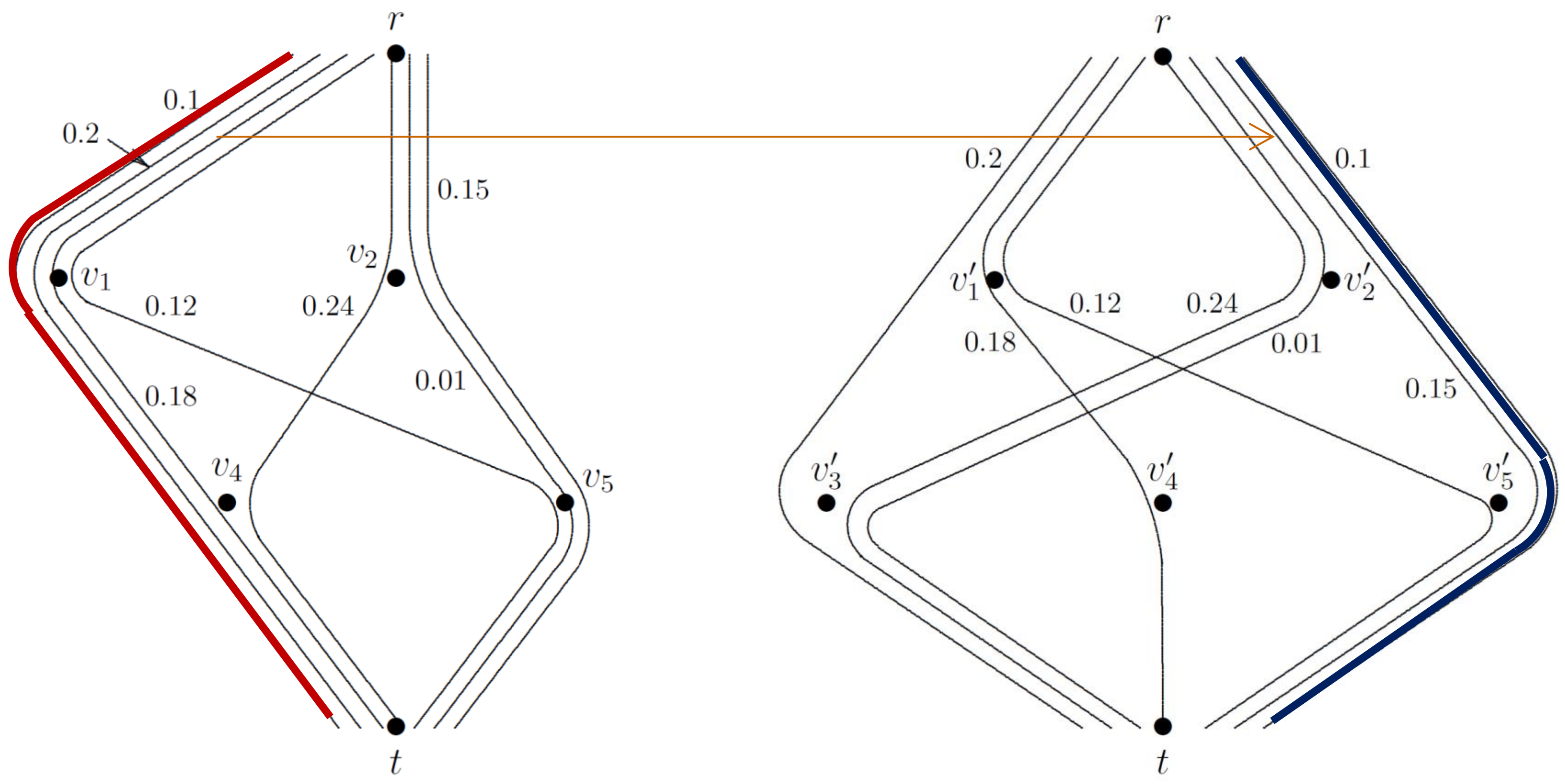


Relaxed SDD



Flow-path decomposition for policy in blue

Must have 1-1 mapping of each path to one of equal flow and no greater cost



Flow-path decomposition for original SDD

Flow-path decomposition for relaxed SDD

Relaxed SDD

Theorem. The expected cost of a relaxed SDD is a bound on the expected cost of the original SDD.

Relaxation by Node Merger

- Can we relax by merging nodes?
  - As in deterministic case?
- Focus on a job sequencing problem.
  - Processing time is stochastic.
  - Minimize penalty for tardiness.

Relaxation by Node Merger

- Can we relax by merging nodes?
  - As in deterministic case?
- Focus on a job sequencing problem.
  - Processing time is stochastic.
  - Minimize penalty for tardiness.
- Decision diagram:
  - Associate a state with each node:

$$(S,t)$$

where S = {jobs still available to schedule}
t = finish time of jobs scheduled so far

Relaxation by Node Merger

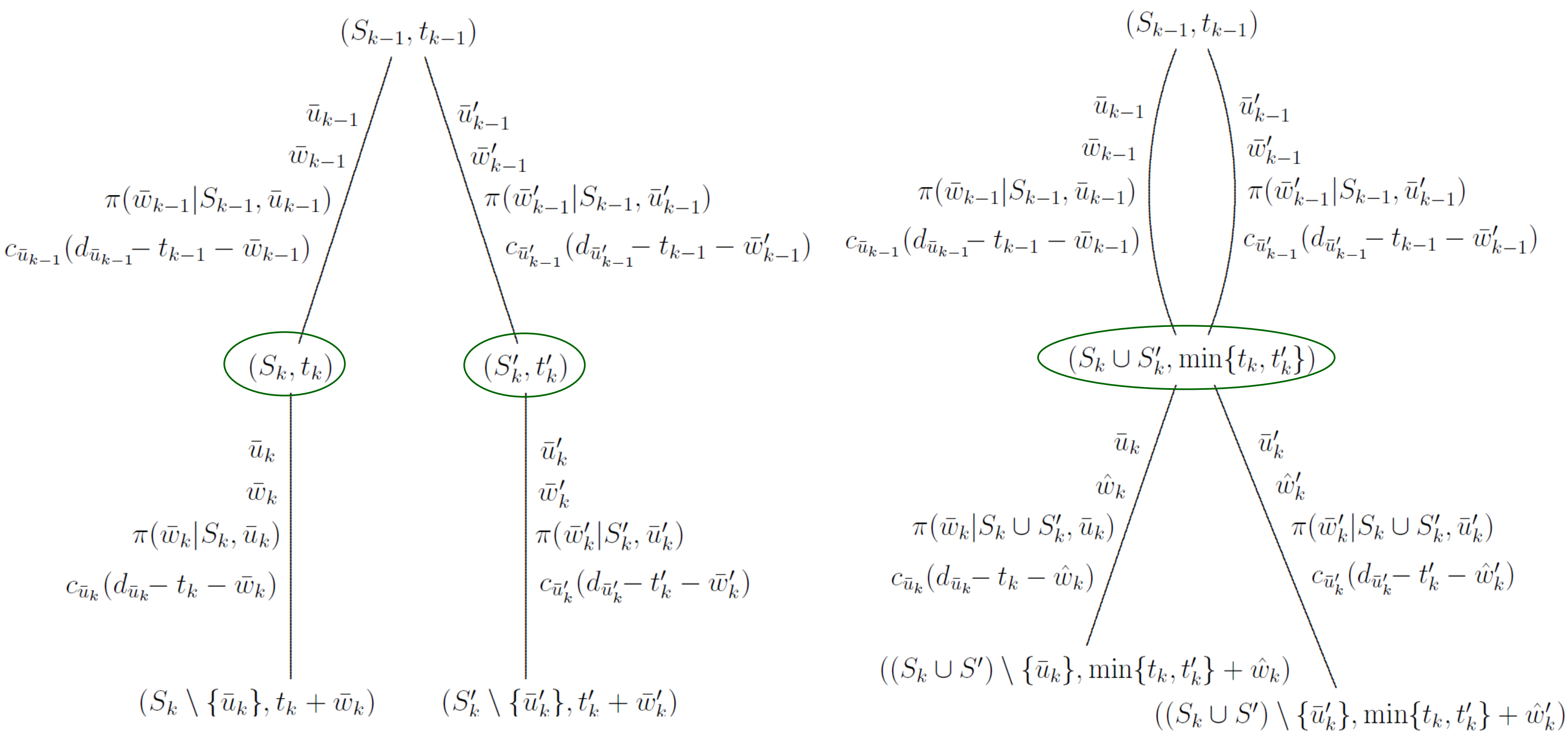
- Merging nodes creates a valid relaxation if:
  - Probability distribution over outcomes is the same at each node, except for an offset.
  - We merge node states as follows:

$$(S,t),\ (S',t') \rightarrow (S \cup S',\ \min\{t,t'\})$$

Union ensures that no feasible policies are excluded.

Min ensures that tardiness cost does not increase.

General node merger scheme for sequencing problem



Before merger of circled nodes

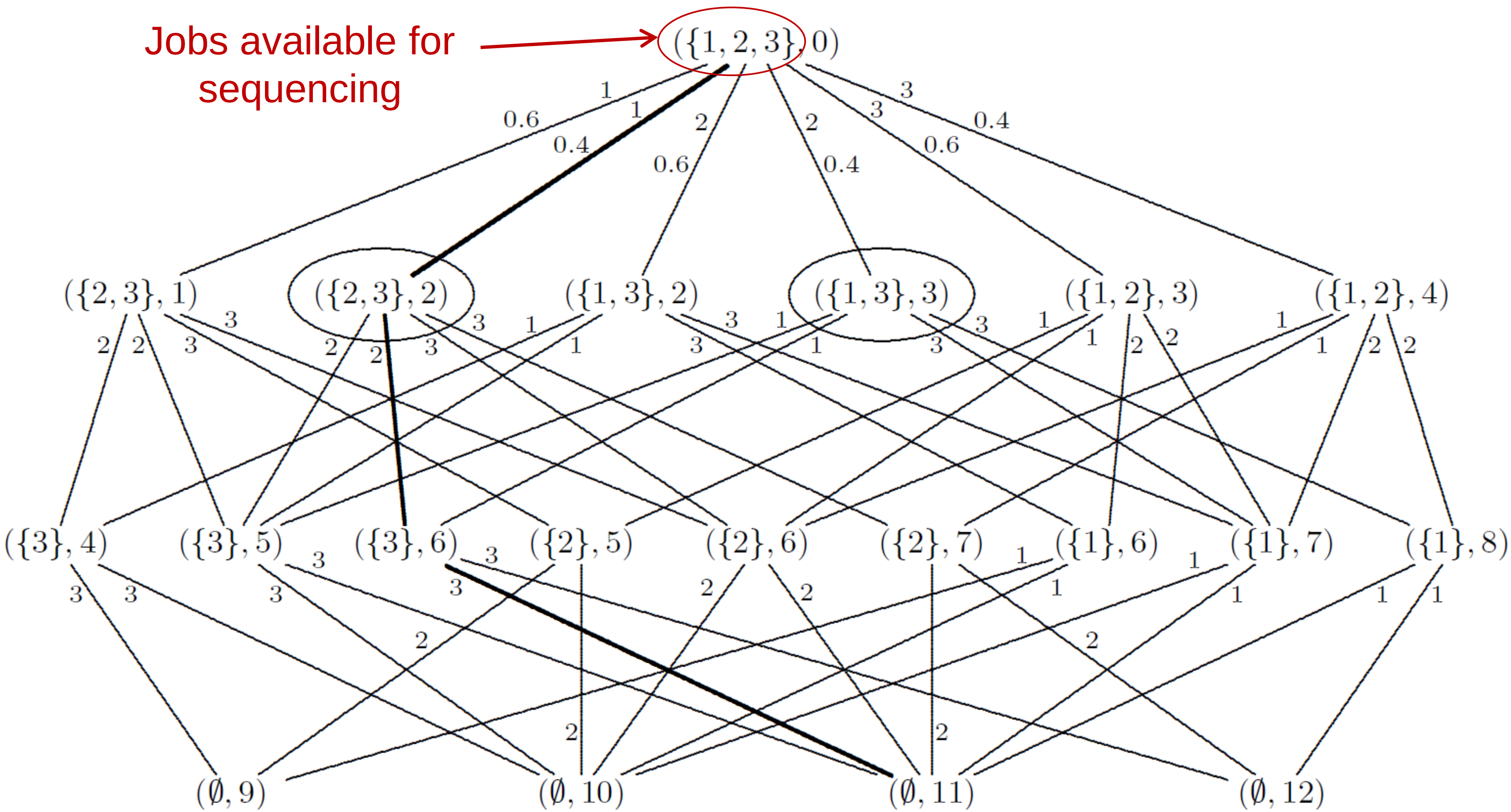
After merger of circled nodes

Relaxation by Node Merger

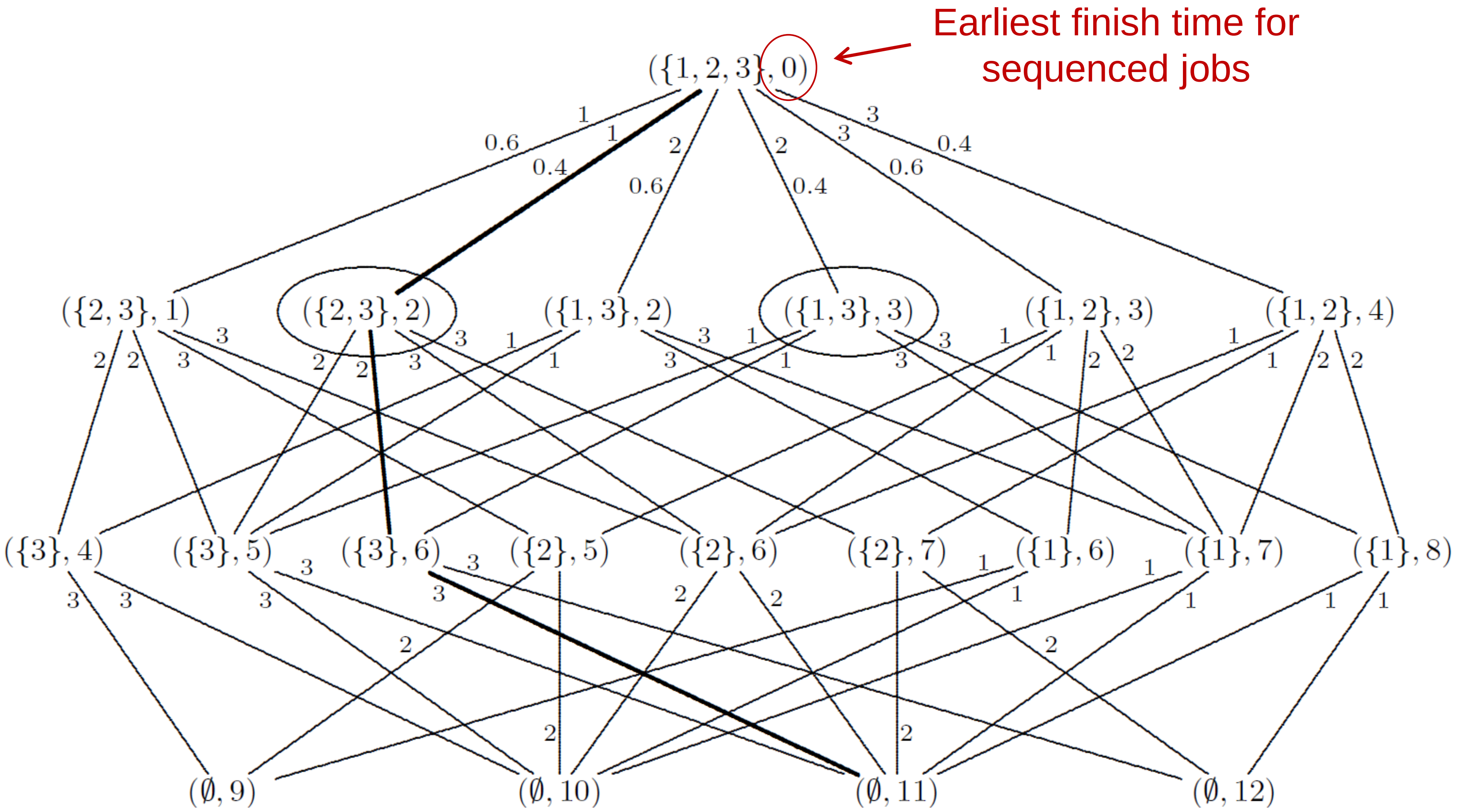
- Consider an instance with 3 jobs.
  - Minimize total tardiness.
  - Transition probabilities have offset pattern:

Job	No. of jobs already processed	Processing time 1	2	3	4	5	6
1	0	0.6	0.4				
	1		0.6	0.4			
	2			0.6	0.4		
2	0		0.6	0.4			
	1			0.6	0.4		
	2				0.6	0.4	
3	0			0.6	0.4		
	1				0.6	0.4	
	2					0.6	0.4

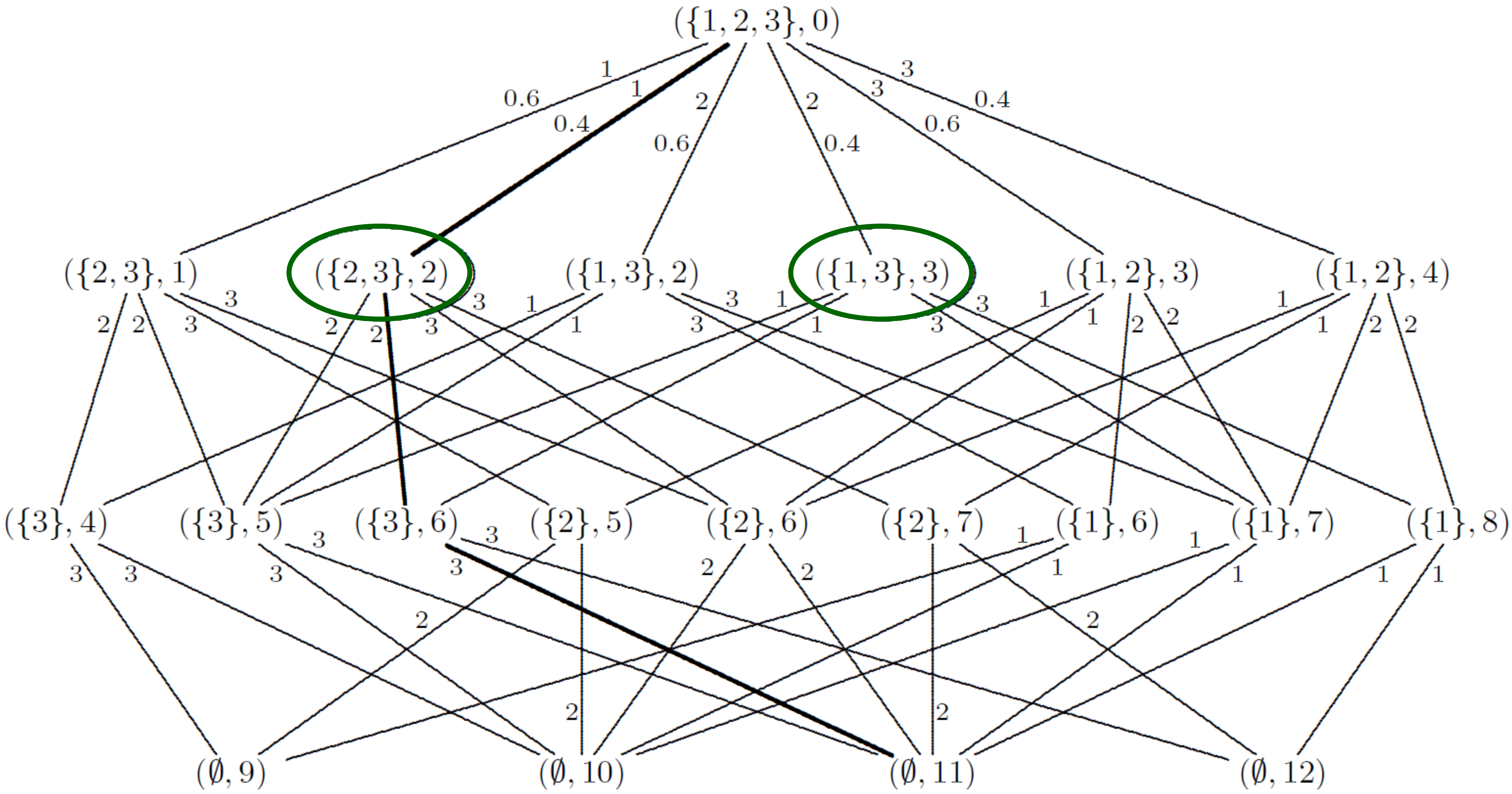
Exact SDD for small sequencing problem



Exact SDD for small sequencing problem

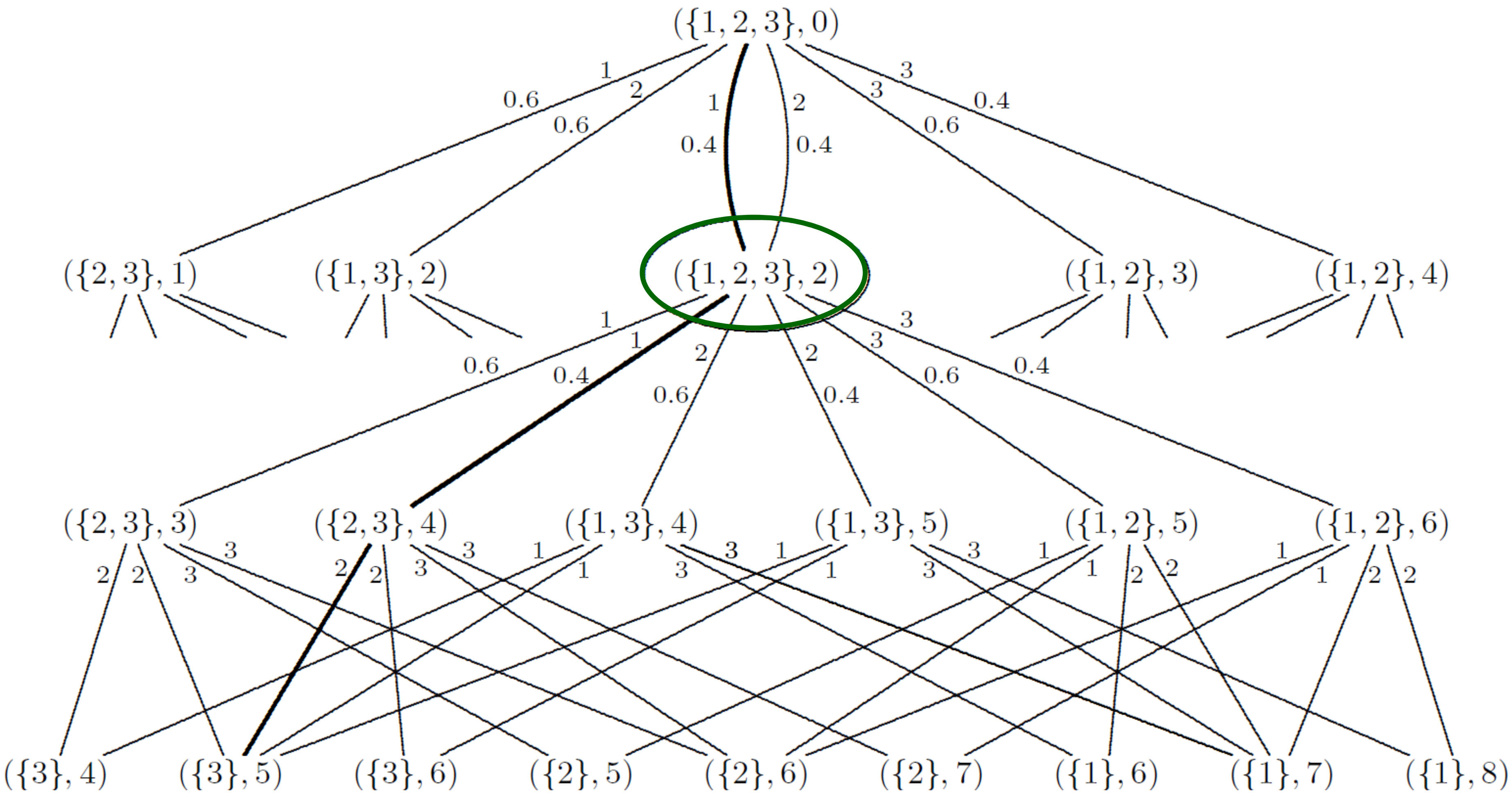


Exact SDD for small sequencing problem



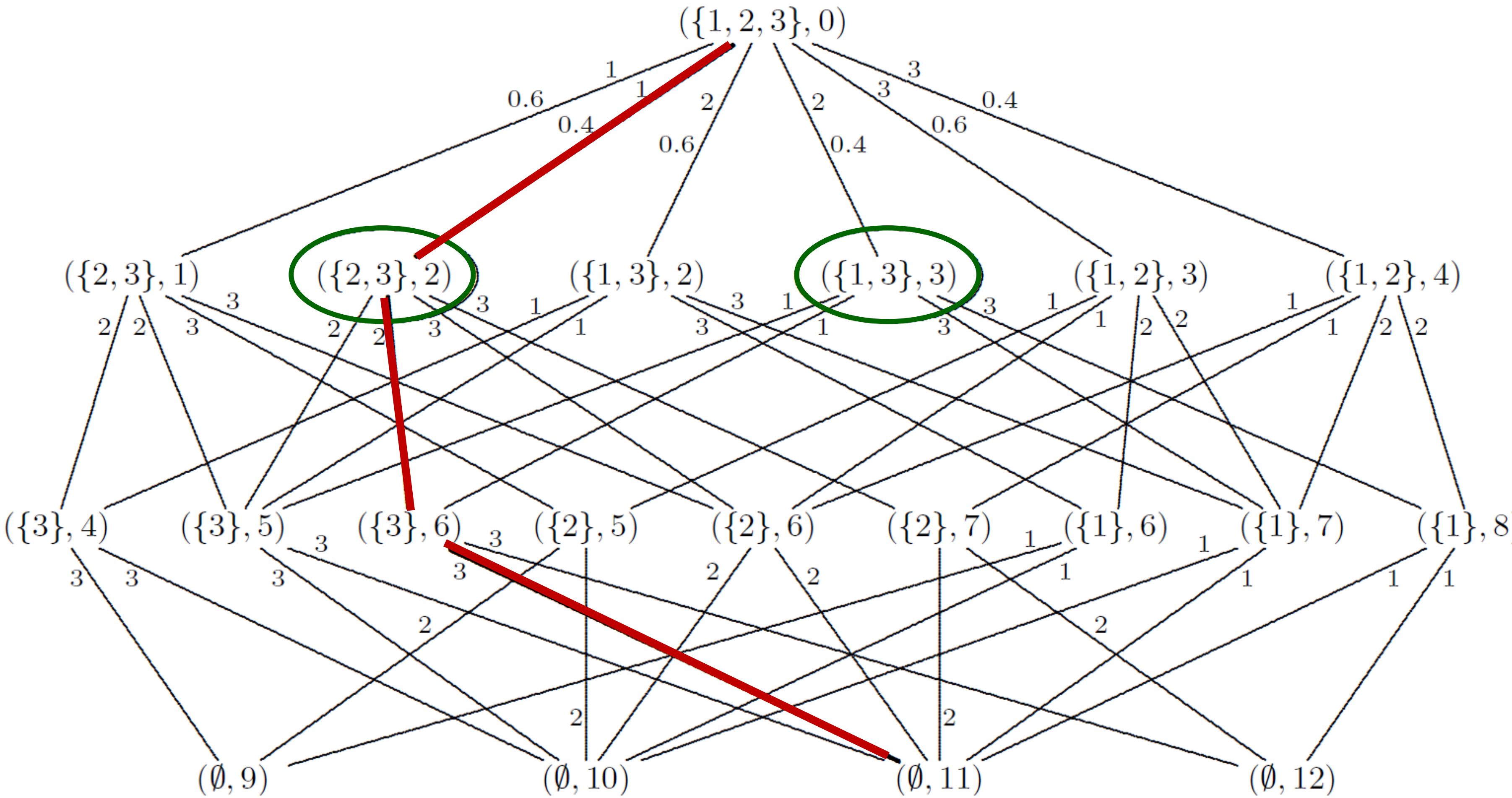
Merge circled nodes

Relaxed SDD for small sequencing problem



Merge circled nodes

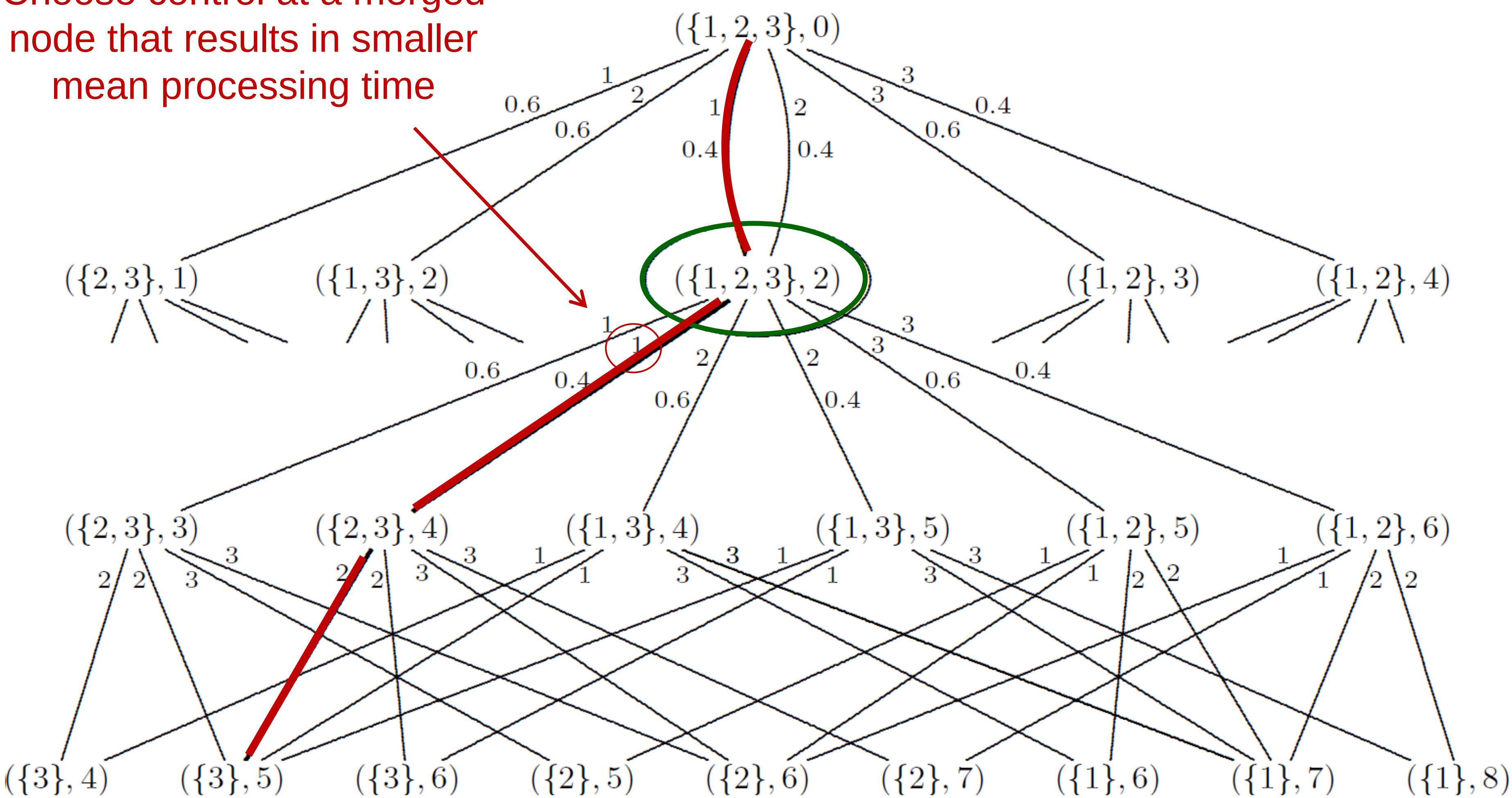
Check validity of relaxation



This path, with flow (0.4)(0.4)(0.6) = 0.096 …

Check validity of relaxation

Choose control at a merged node that results in smaller mean processing time



…maps to this path with same flow, ≤ cost

Relaxed SDD

Theorem. This merger scheme creates a relaxed SDD, even when applied recursively.

Relaxation by Node Merger

- If probability distributions **do not have offset pattern**…
  - Replace each distribution with one that **stochastically dominates** it, so that replacement distributions have the offset pattern..
    - Use original distributions down to last exact later of DD.
    - Weakens relaxation but does not sacrifice optimality.

Partially Observable State Spaces

- Can use relaxed SDDs for this case.
  - State is information vector.
    - such as probability distribution over original states.
- Can again relax by merging nodes.
  - Using similar merger rule.
  - If transition probabilities have offset property.

Relaxing Stochastic DP

- Relaxed SDDs provide a general **relaxation scheme for stochastic DP**.
  - Yields a valid lower bound.
  - …unlike most state space approximation schemes.
- Also a new solution method.
  - The same SDD that provides a relaxation provides a framework for solution by branch and bound.
- Relaxation created dynamically.
  - For example, using node merger.