Stochastic Binary Decision Diagrams

John Hooker
Carnegie Mellon University

ISAIM 2022

Motivation

- Binary decision diagrams (BDDs) have proved useful for solving **discrete optimization** problems.
  - Especially those with recursive **dynamic programming** (DP) models.

- Yet many (most) DP models are **stochastic**.
  - We therefore generalize BDDs to **stochastic BDDs** (SDDs) by adding probabilities to arcs.

Deterministic BDDs

- A deterministic BDD is a graphical representation of a **Boolean function**.
  - Often used for logic circuit design, product configuration.
  - We use **ordered** BDDs.
  - Easily extended to **multivalued** DDs (MDDs).

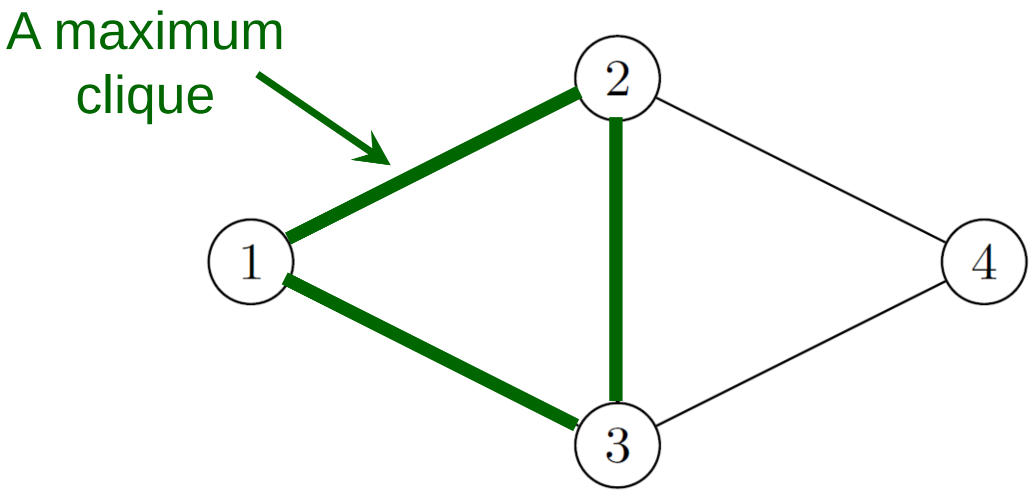
- A **weighted** BDD can represent a discrete **optimization problem**.

Hadzic and JH (2006, 2007)

  - For example, the **maximum clique** problem...

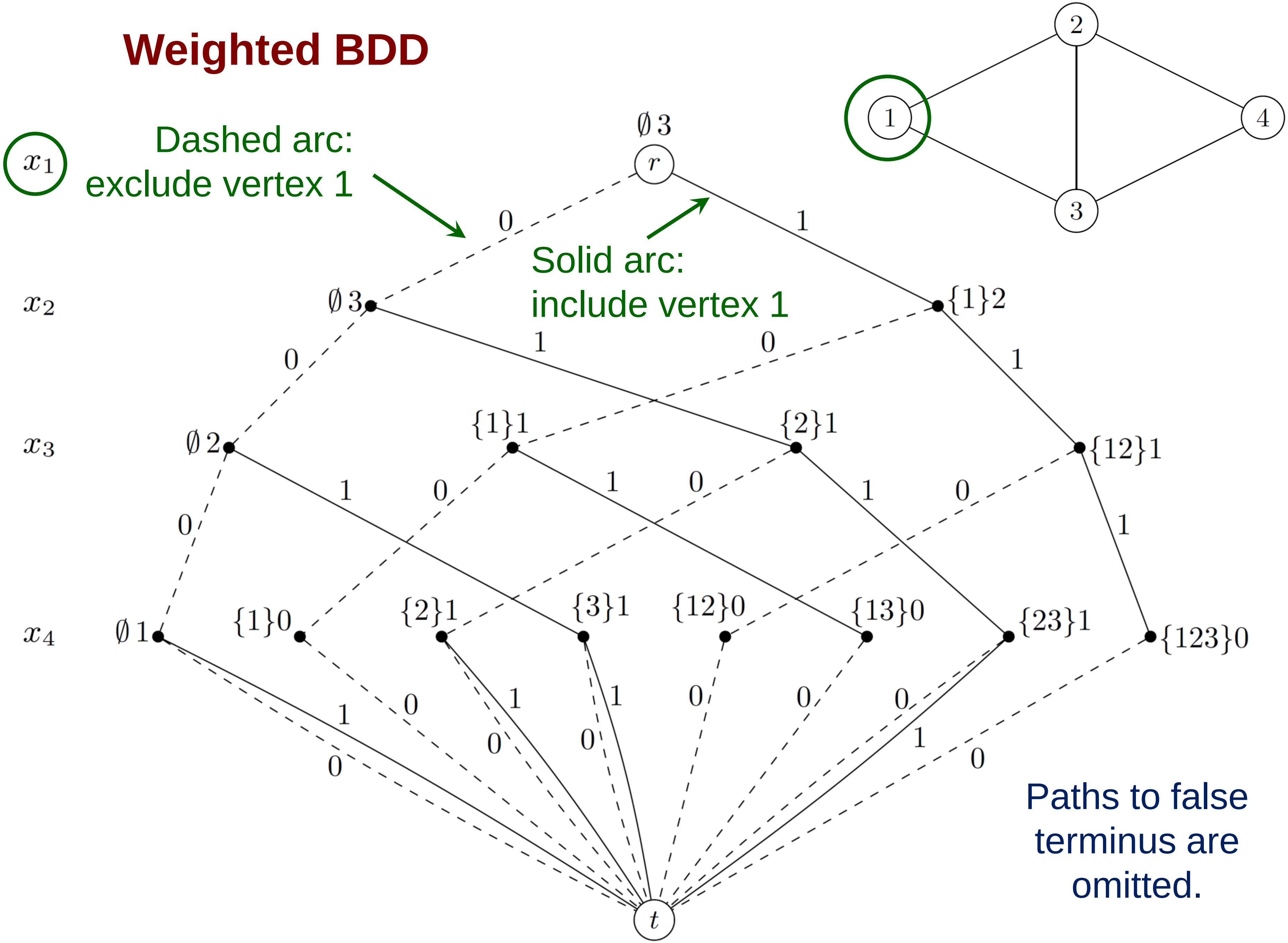
Maximum clique

Max clique example



Let binary variable $x_i = 1$
when vertex i is in the clique.

Weighted BDD

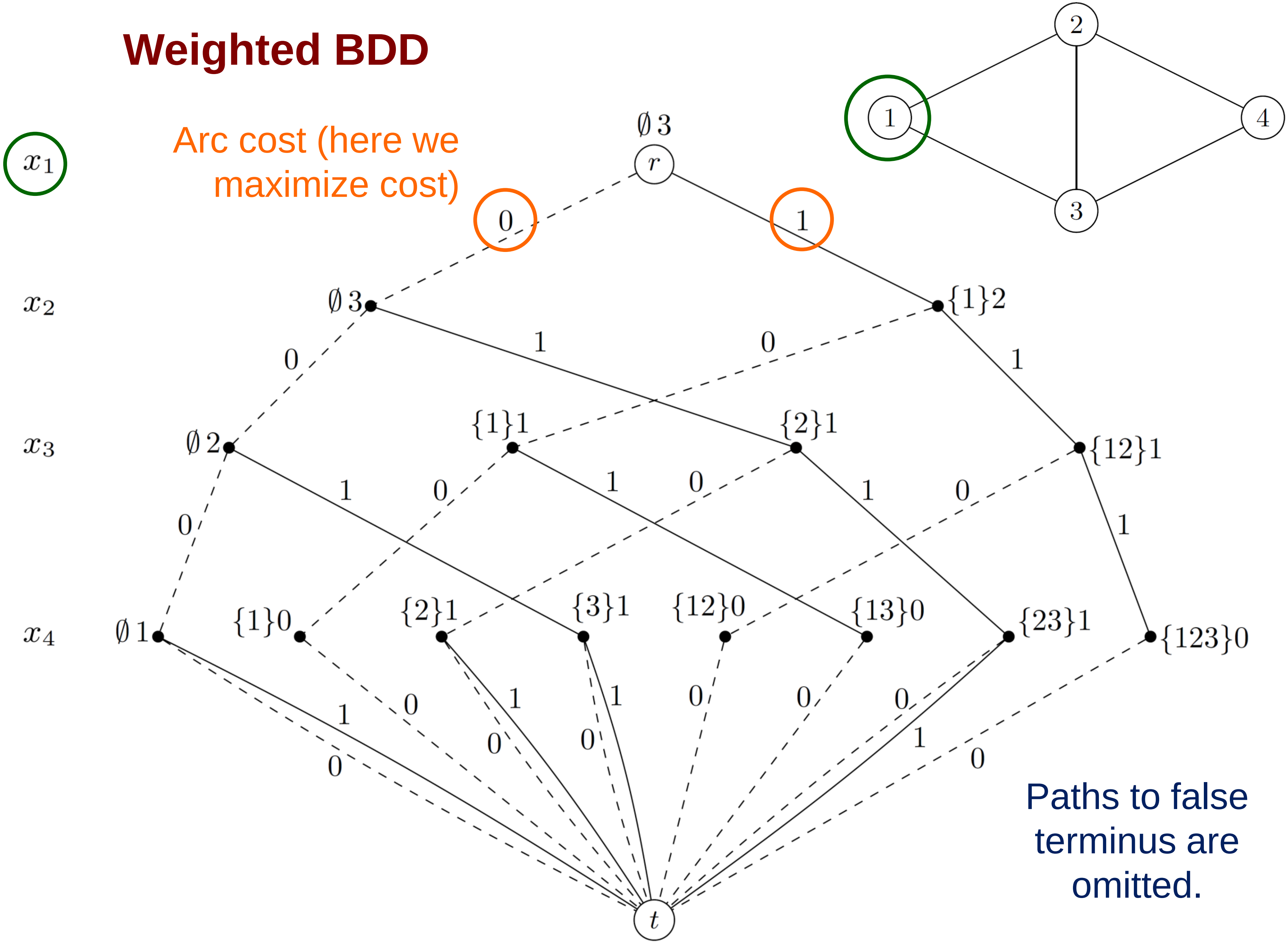


Dashed arc: exclude vertex 1

Solid arc: include vertex 1

Paths to false terminus are omitted.

Weighted BDD

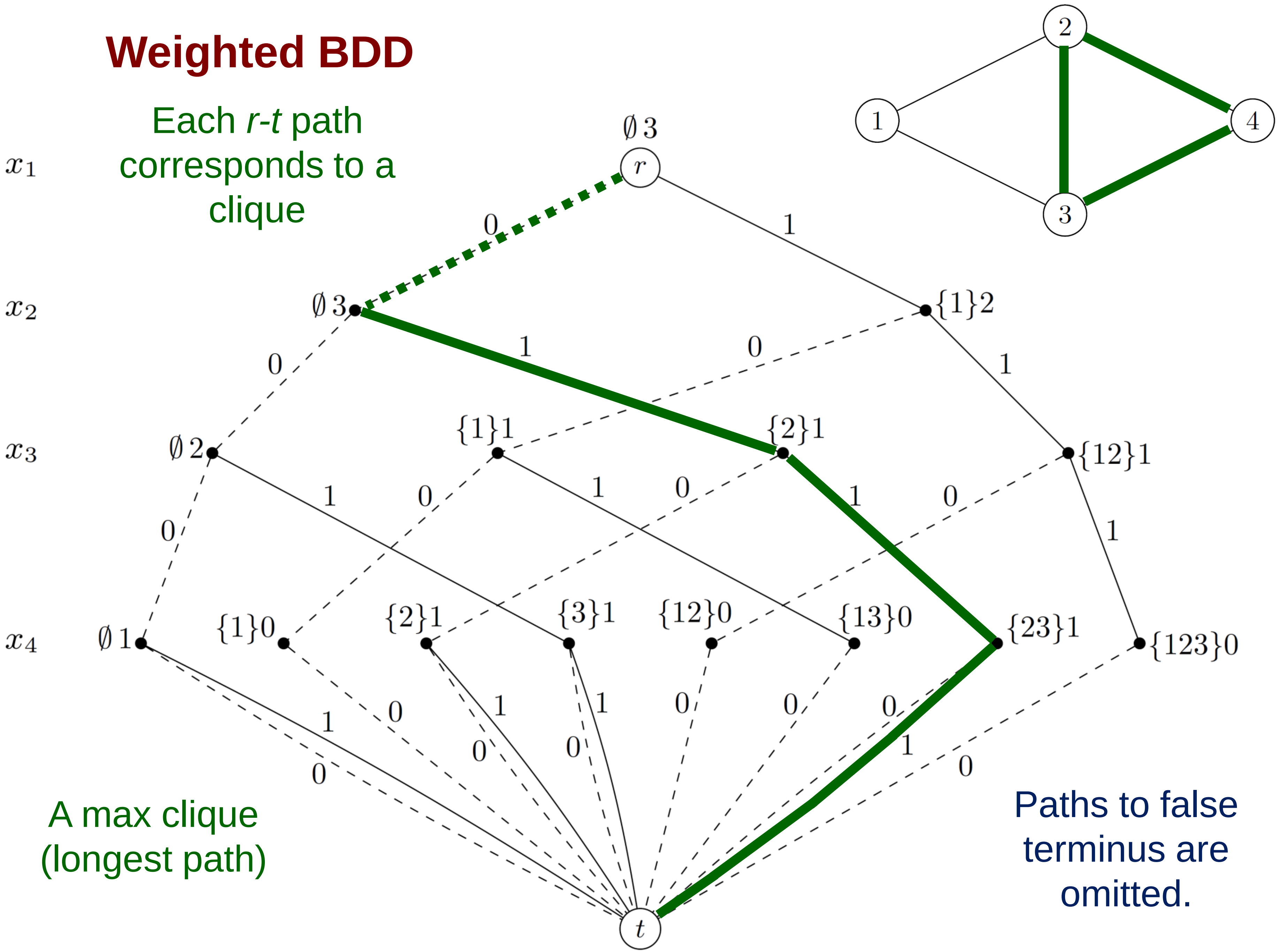


Arc cost (here we maximize cost)

Paths to false terminus are omitted.

Weighted BDD

Each *r*-*t* path corresponds to a clique

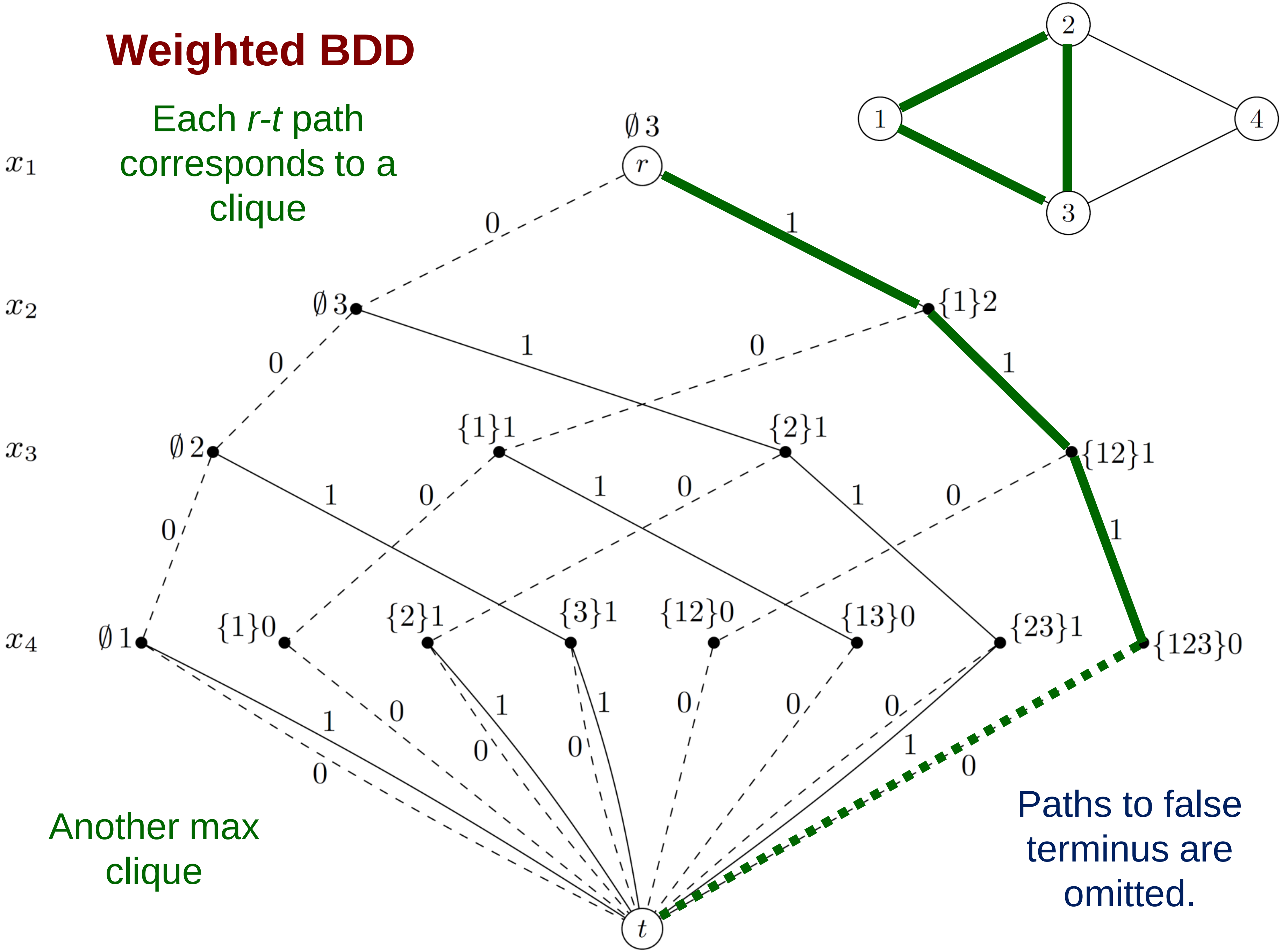


A max clique (longest path)

Paths to false terminus are omitted.

Weighted BDD

Each *r*-*t* path corresponds to a clique



Another max clique

Paths to false terminus are omitted.

Dynamic Programming

- The **state transition graph** of a dynamic programming (DP) problem can be interpreted as a BDD (or MDD).
  - By associating states with nodes of the BDD.
  - This opens the door to using BDD **relaxation** techniques to obtain bounds for DPs.

Andersen, Hadzic, JH, Tiedemann (2007)
Bergman, Cire, van Hoeve, JH (2013)

  - …and to solving the DPs by **branch and bound**.

Bergman, Cire, van Hoeve, JH (2014)

  - For example, the **maximum clique** problem...

Deterministic BDDs

Max clique DP model

State variable $S_i = \{\text{vertices selected so far}\}$.

The recursion is

$$h_i(S_i) = \max\left\{\underset{\substack{\uparrow \\ x_i = 0}}{h_{i+1}(S_i)},\ \ 1 + \underset{\substack{\uparrow \\ x_i = 1}}{h_{i+1}\big(S_i \cup \{i\}\big)}\right\},\ \ i = n, \ldots, 1$$

Deterministic BDDs

Max clique DP model

State variable $S_i = \{\text{vertices selected so far}\}$.

The recursion is

$$h_i(S_i) = \max\Big\{\underset{x_i = 0}{h_{i+1}(S_i)},\ \ 1 + \underset{x_i = 1}{h_{i+1}\big(S_i \cup \{i\}\big)}\Big\},\ \ i = n, \ldots, 1$$

In general,

$$h_i(\boldsymbol{S}_i) = \min_{x_i}\Big\{c_i(\boldsymbol{S}_i, x_i) + h_{i+1}\big(\phi_i(\boldsymbol{S}_i, x_i)\big)\Big\},\ \ i = n, \ldots, 1$$

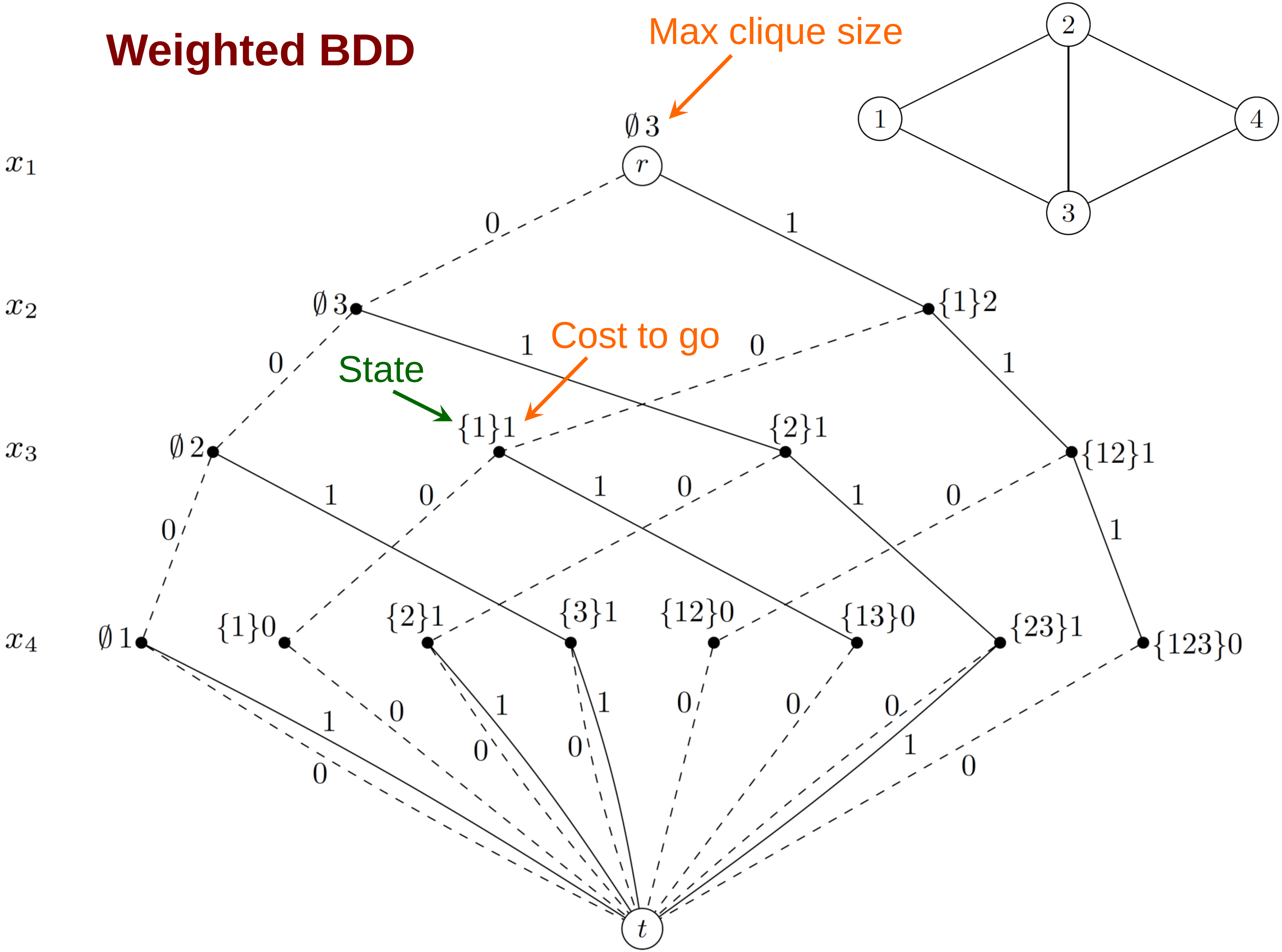
control

immediate cost

cost to go

state transition function

Weighted BDD



Deterministic MDDs

Job sequencing example

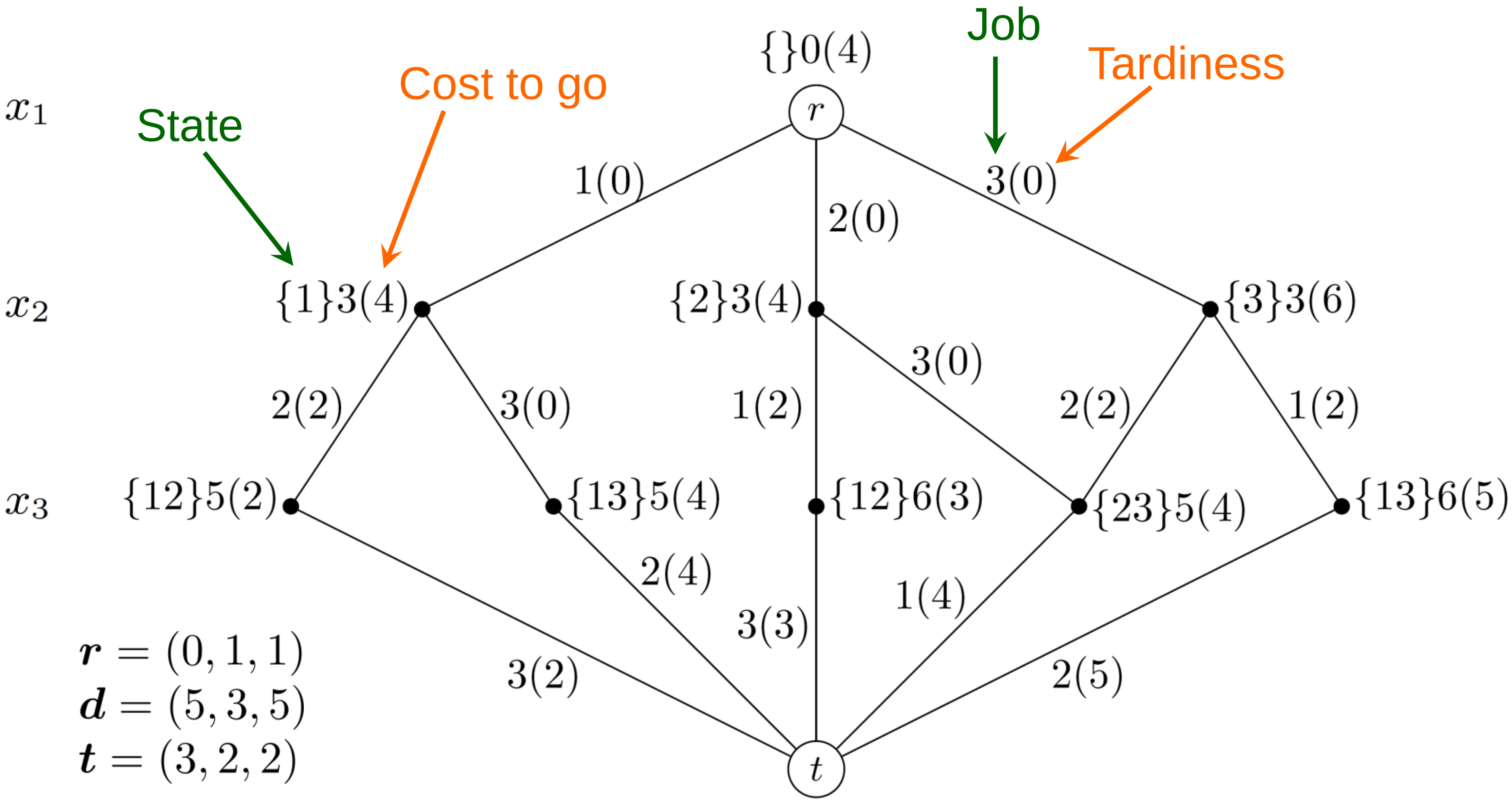
Let control x_i be ith job in sequence.

Time window $[r_i, d_i]$ and processing time t_i for each job i.

Minimize total tardiness.

Weighted MDD

State $= (S_i, f_i)$, where
$S_i = \{\text{jobs sequenced so far}\}$, $f_i =$ finish time of previous job



Deterministic BDDs

Job sequencing DP model

State is (S_i, f_i).

The recursion is

$$h_i(S_i, f_i) = \min_{x_i \notin S_i} \left\{ c_i(S_i, f_i) + h_{i+1}\big(\phi_i((S_i, f_i), x_i)\big) \right.$$

where

$$c_i\big((S_i, f_i), x_i\big) = \max\left\{0,\ \max\{r_{x_i}, f_i\} + t_{x_i} - d_{x_i}\right\}$$

$$\phi_i\big((S_i, f_i), x_i\big) = \big(S_i \cup \{x_i\}, \max\{r_{x_i}, f_i\} + t_{x_i}\big)$$

Dynamic Programming

- Note: a state transition graph is a **different concept** than a BDD.
  - A BDD does not need **state information**.

Dynamic Programming

- Note: a state transition graph is a **different concept** than a BDD.
  - A BDD does not need **state information**.
- The BDD perspective yields advantages:
  - A BDD can be often be **reduced** by identifying isomorphic portions of the BDD that are associated with different states.

    Bryant (1986 etc.)
    - This occasionally results in **radical** simplification (e.g., inventory management).

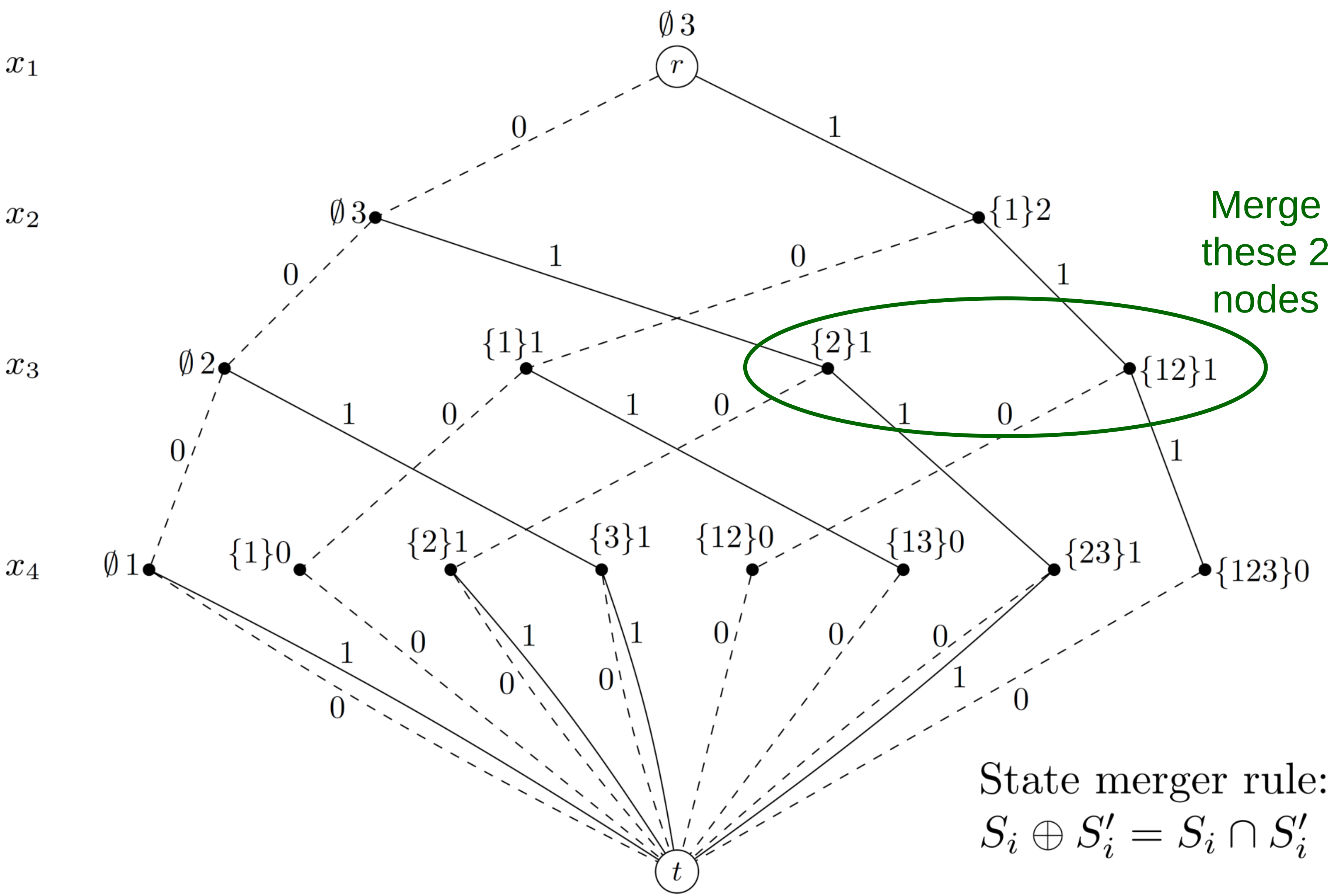
      JH (2013)
  - BDDs can also have **different** nodes that correspond to the **same state** (we will do this later).
  - DP can benefit from **relaxation techniques** that have been developed for BDDs…

Relaxed BDDs

- A **relaxed** BDD represents a superset of feasible solutions.
  - Can provide a **bound** on the optimal value.
- Created during top-down compilation of the BDD.
  - By **node merger** or **node splitting**.
  - We focus on **node merger**.
  - Mergers result in **smaller BDD** but weaker bound.
  - Can obtain bound of **any desired quality** by controlling width of relaxed BDD.

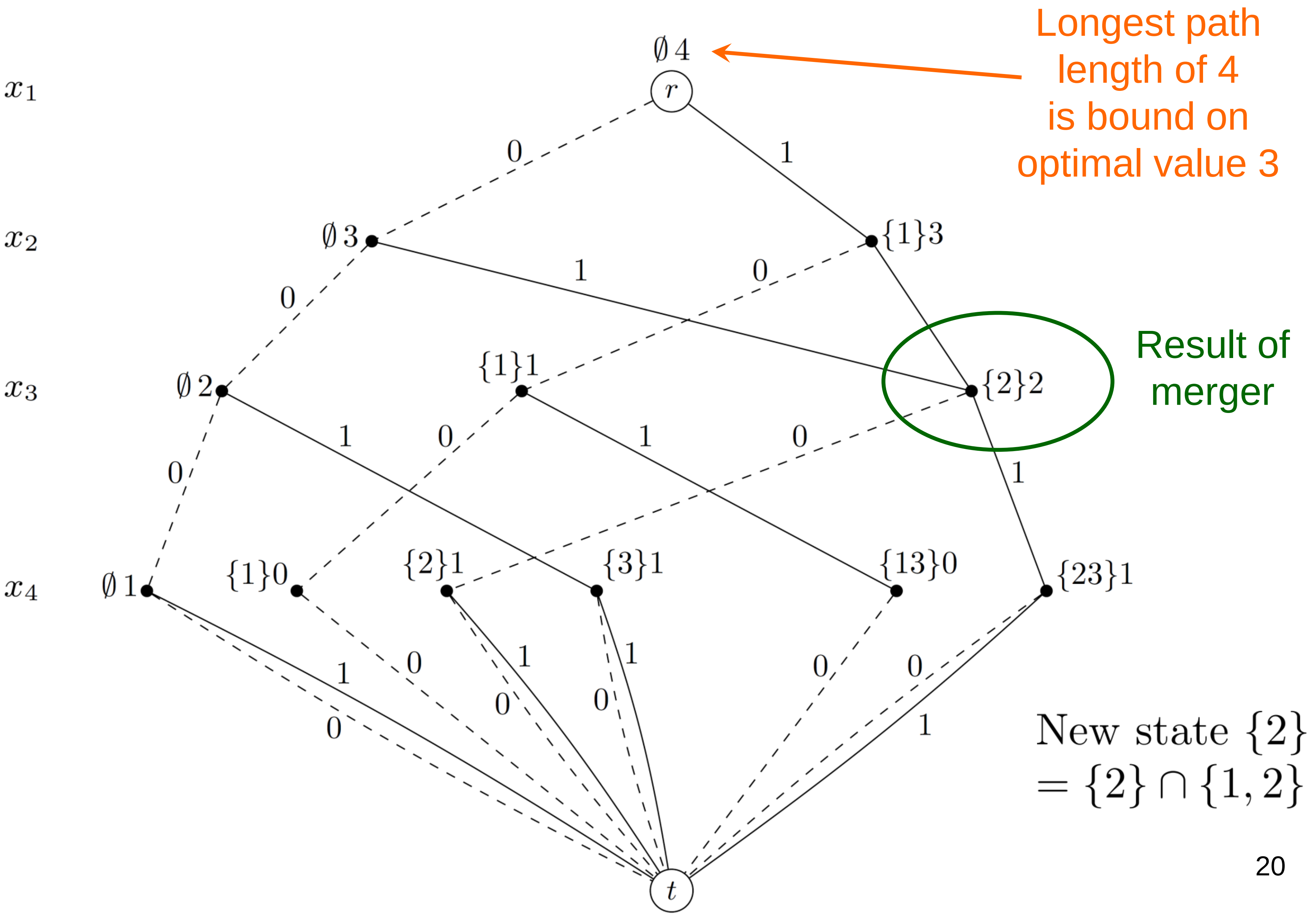
Andersen, Hadzic, JH, Tiedemann (2007)
Bergman, Cire, van Hoeve, JH (2013)

Weighted BDD for max clique

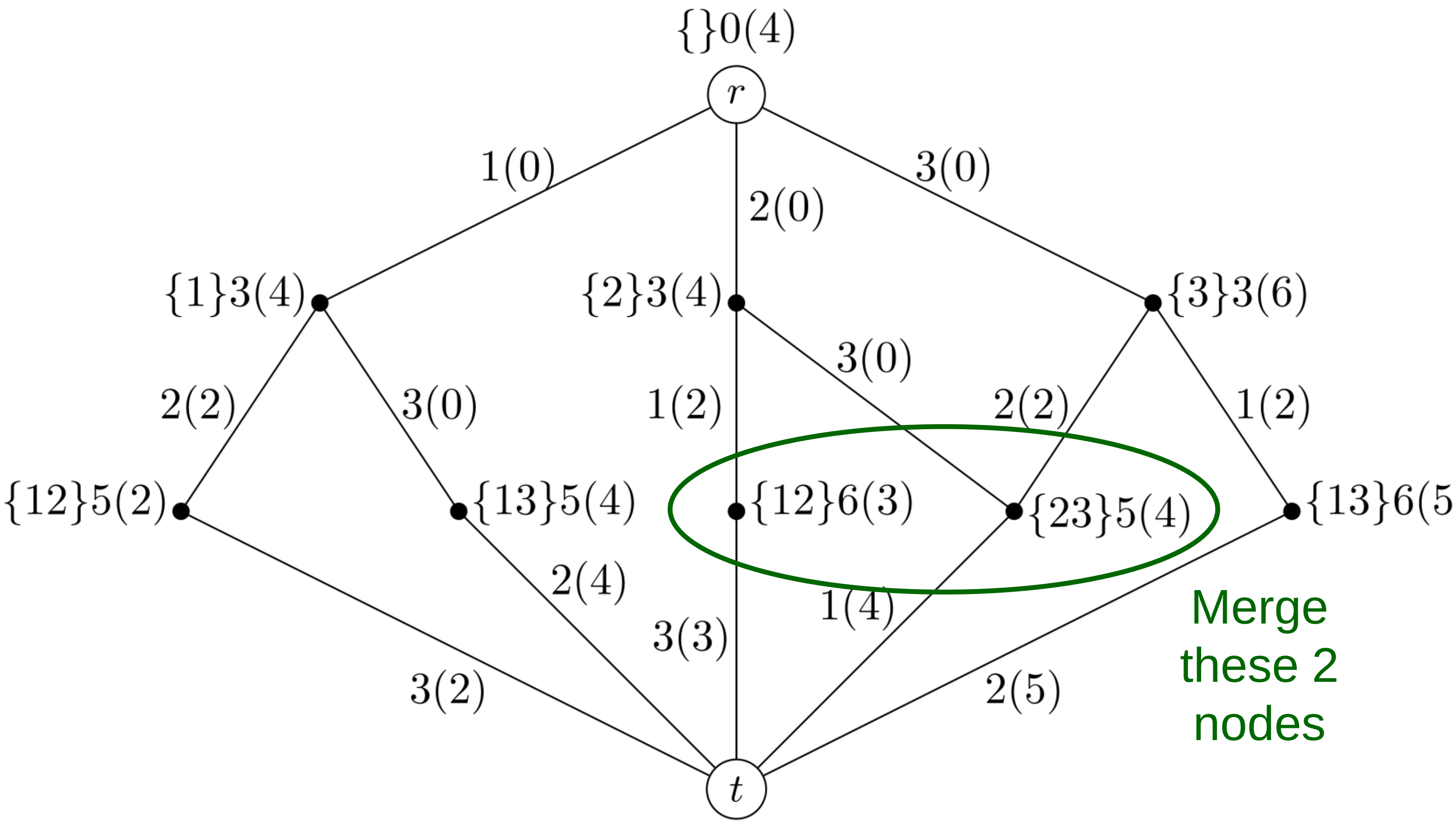


State merger rule:
$S_i \oplus S'_i = S_i \cap S'_i$

Relaxed BDD for max clique

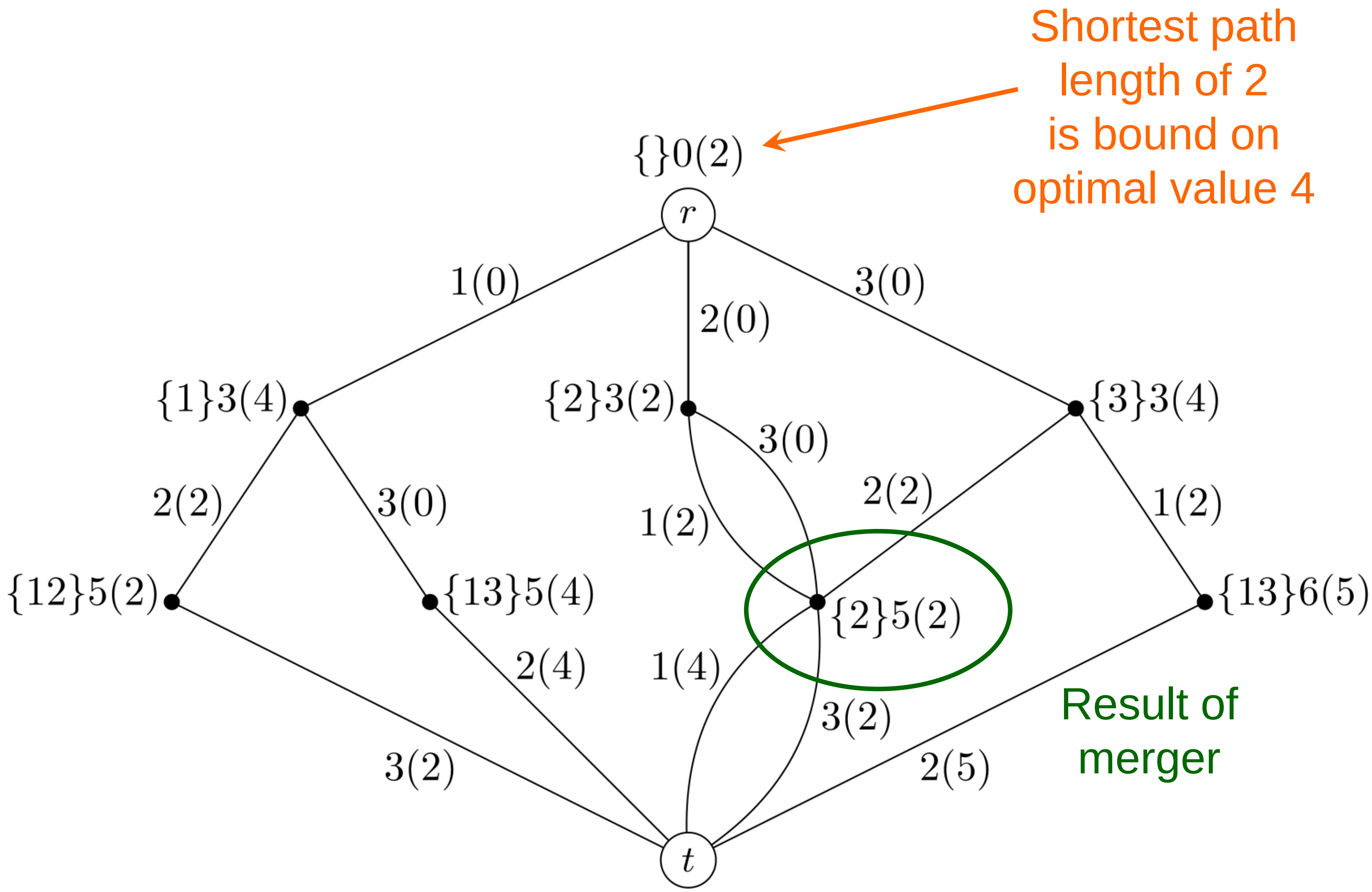


Weighted MDD for job sequencing



State merger rule: $(S_i, f_i) \oplus (S'_i, f'_i) = \big((S_i \cap S'_i), \min\{f_i, f'_i\}\big)$

Relaxed MDD for job sequencing



$$\text{New state } (\{2\}, 5) = (\{1,2\} \cap \{2,3\}, \min\{6,5\})$$

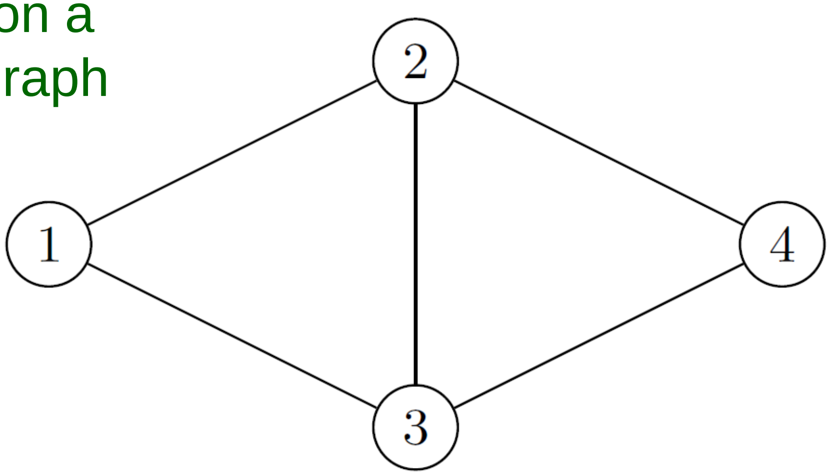
Stochastic BDDs

- A stochastic decision diagram (SDD) has **probabilistic** transitions to the next layer.
  - A control can have several possible **outcomes**, each with a known probability.
  - The outcome determines which arc is followed.
- A solution is now a **policy**.
  - The control in a given layer depends on the **state** (node).

Stochastic BDDs

Max clique example

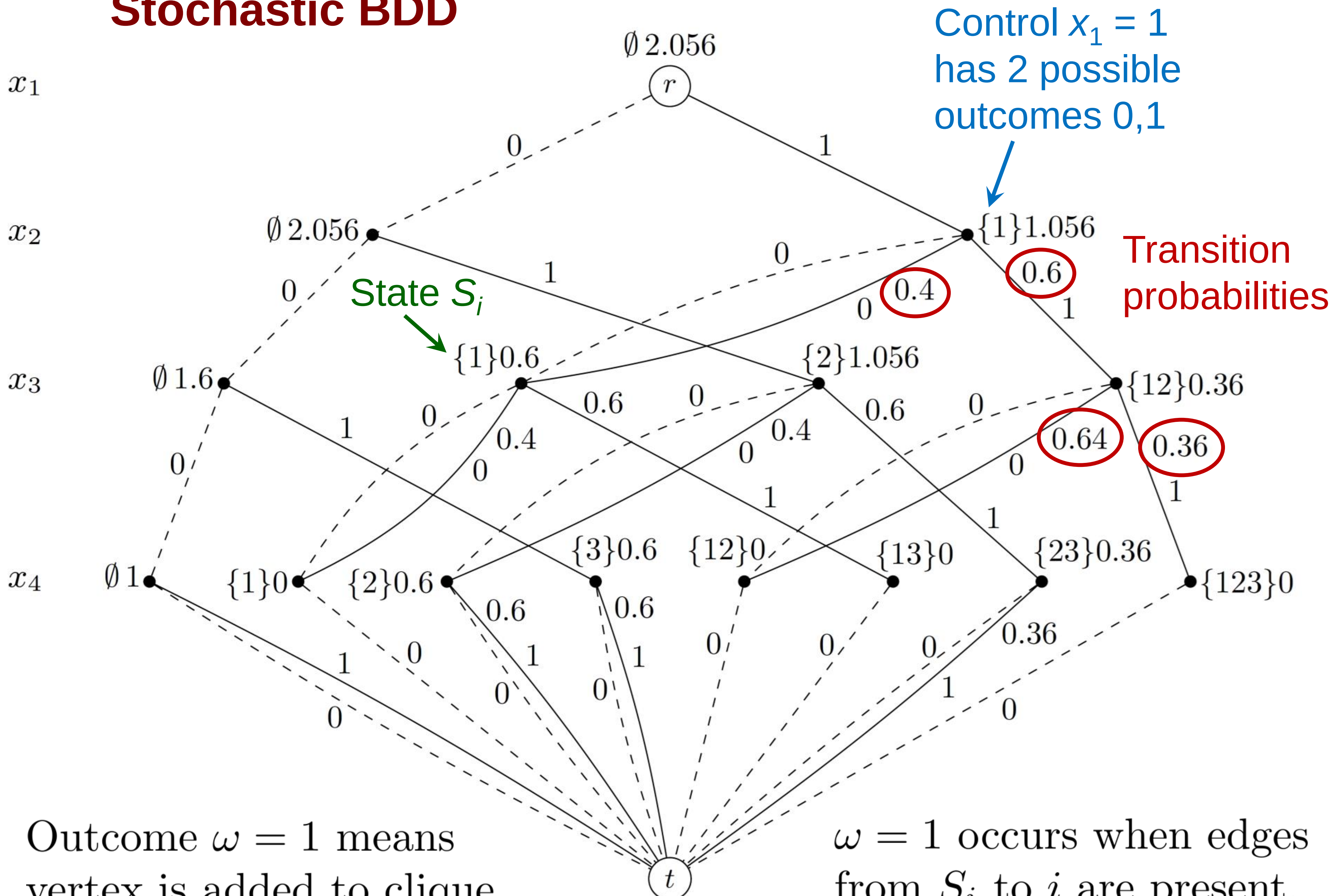
Defined on a
random graph



Each arc has probability 0.6

Objective is to maximize expected clique size.

Stochastic BDD



Outcome $\omega = 1$ means vertex is added to clique

$\omega = 1$ occurs when edges from S_i to i are present.

Stochastic BDDs

Max clique DP model

The recursion is

$$h_i(S_i) = \max\left\{\underbrace{h_{i+1}(S_i)}_{x_i = 0},\ \underbrace{\big(1 - p(S_i)\big)h_{i+1}(S_i) + p(S_i)h_{i+1}\big(S_i \cup \{i\}\big)}_{x_i = 1}\right\}$$

where $p(S_i)$ = probability that vertex i can be added to clique

Stochastic BDDs

Max clique DP model

The recursion is

$$h_i(S_i) = \max\left\{ h_{i+1}(S_i),\ \big(1 - p(S_i)\big) h_{i+1}(S_i) + p(S_i) h_{i+1}\big(S_i \cup \{i\}\big) \right\}$$

$x_i = 0$ $\qquad\qquad$ $x_i = 1$

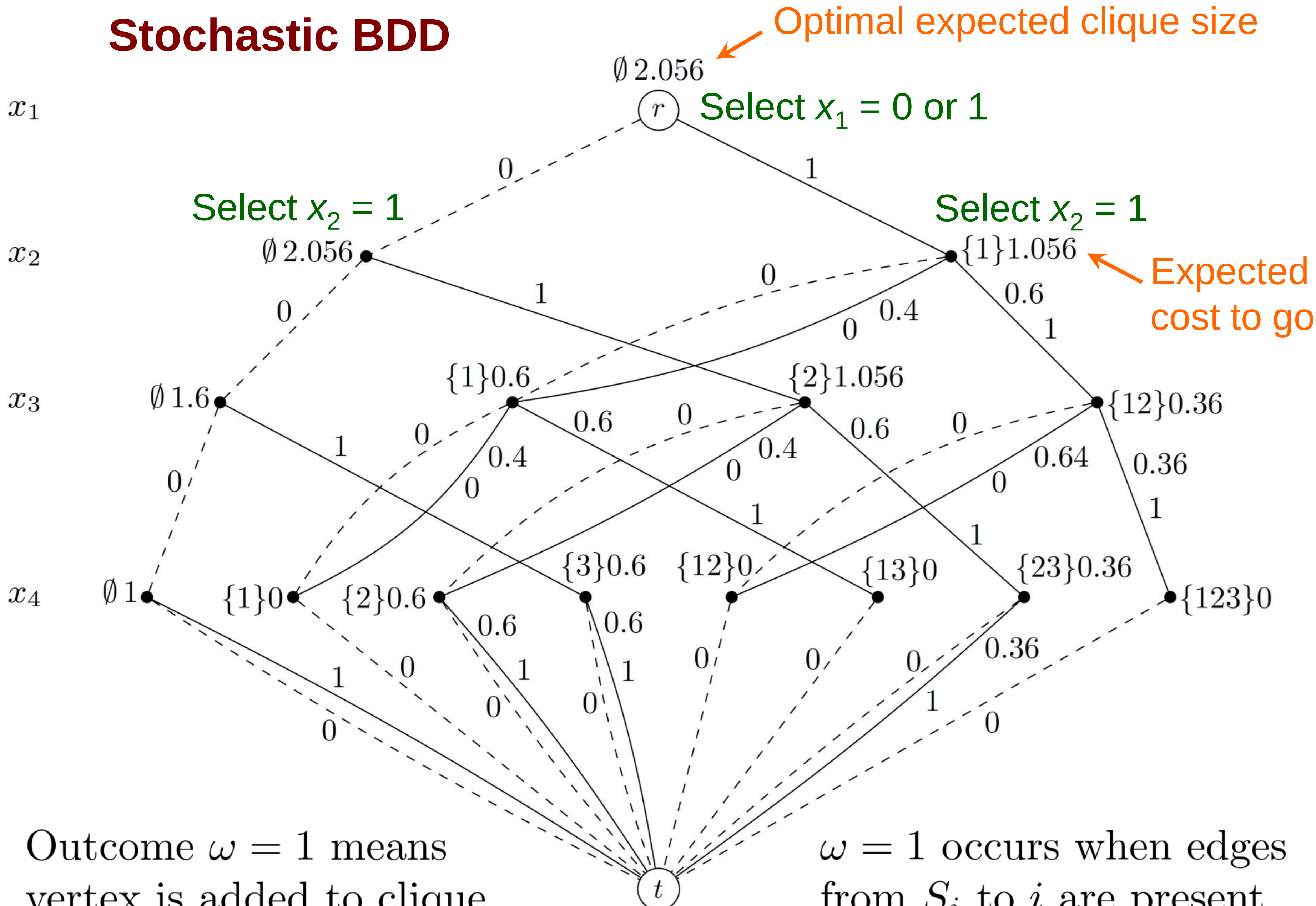
where $p(S_i)$ = probability that vertex i can be added to clique

In general,

$$h_i(\boldsymbol{S}_i) = \min_{x_i} \left\{ \sum_{\omega} p_{i\omega}(\boldsymbol{S}_i, x_i) \big[c_{i\omega}(\boldsymbol{S}_i, x_i) + h_{i+1}\big(\phi_{i\omega}(\boldsymbol{S}_i, x_i)\big) \big] \right\}$$

where $p_{i\omega}(\boldsymbol{S}_i, x_i)$ = prob. of outcome ω given control x_i in state $\boldsymbol{S}_i$ and similarly for $c_{i\omega}(\boldsymbol{S}_i, x_i)$ and $\phi_{i\omega}(\boldsymbol{S}_i, x_i)$

Stochastic BDD



Outcome $\omega = 1$ means vertex is added to clique

$\omega = 1$ occurs when edges from S_i to i are present.

Relaxed SDDs

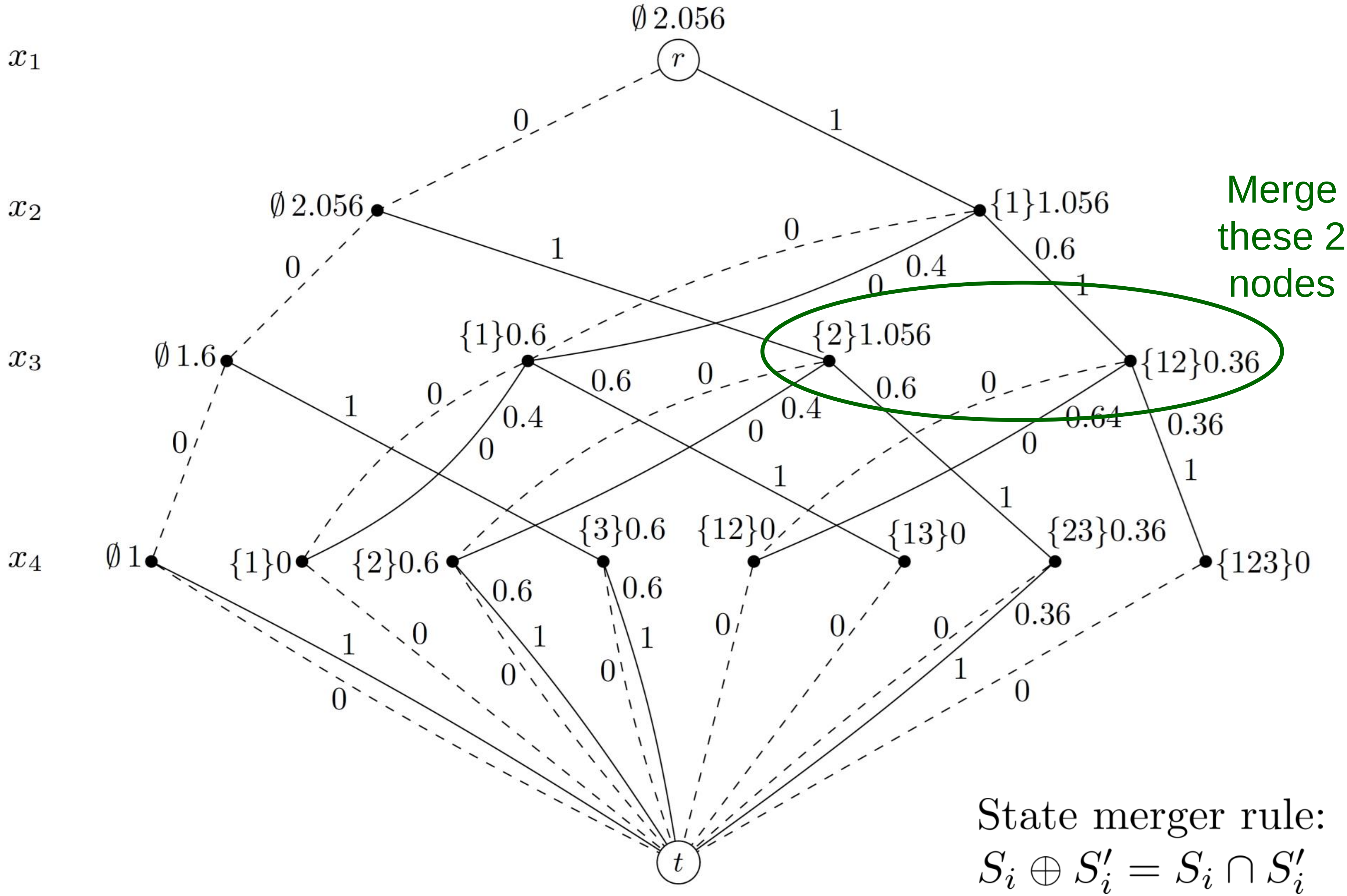
- A relaxed SDD is one that provides a **valid bound** on optimal expected cost.
  - **Unclear** how to define relaxation in terms of **individual solutions**.
  - …since solutions are **policies** defined on the **entire SDD**, and a relaxed SDD may have very different structure.

Stochastic diagram $\bar{D}$ relaxes diagram D when $\bar{D}$ and D have the same variables, controls, and possible outcomes, and when optimal cost of $\bar{D} \leq$ optimal cost of D.

Relaxed SDDs

- We will relax SDDs by node merger.
  - We will also provide **sufficient conditions** under which a given merger operation yields a valid relaxed SDD.
  - Conditions must account for **policy-based** solutions rather than simple control sequences.
  - Examples…

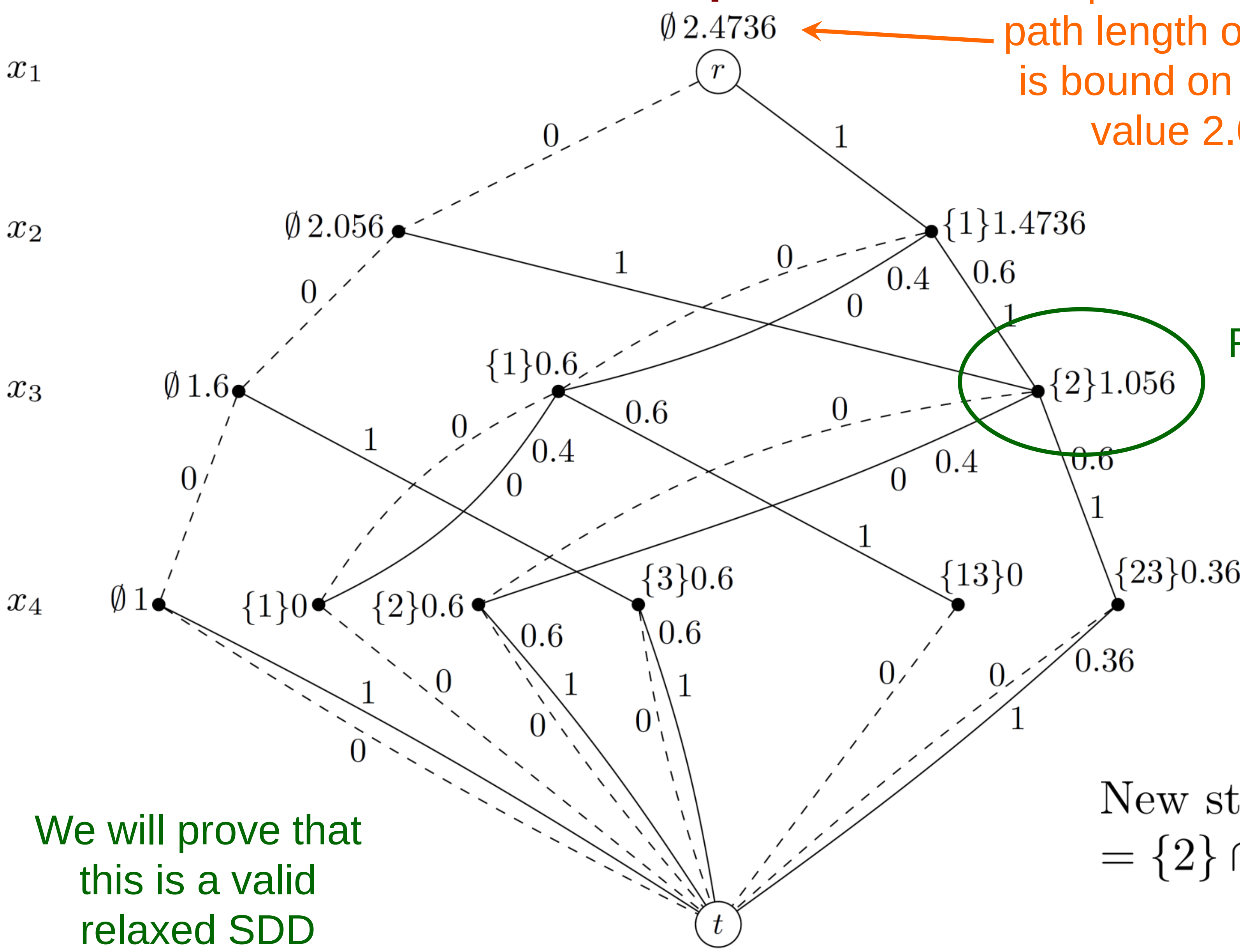
Stochastic BDD for max clique



State merger rule:

$$S_i \oplus S_i' = S_i \cap S_i'$$

Relaxed SDD for max clique



Expected longest path length of 2.4736 is bound on optimal value 2.056

We will prove that this is a valid relaxed SDD

New state $\{2\} = \{2\} \cap \{1,2\}$

Relaxed SDDs

Stochastic job sequencing DP model

Stochastic element is processing time (state independent).
Let $t_{j\omega}$ = job j processing time in outcome ω.
Let $p_{j\omega}$ probability of outcome ω for job j.

The recursion is

$$h_i(S_i, f_i) = \min_{x_i \notin S_i} \left\{ \sum_{\omega} p_{i\omega}(S_i, x_i) \left[c_{i\omega}(S_i, f_i) + h_{i+1}\left(\phi_{i\omega}((S_i, f_i), x_i)\right) \right] \right\}$$

where

$$c_{i\omega}\big((S_i, f_i), x_i\big) = \max\left\{0,\ \max\{r_{x_i}, f_i\} + t_{x_i\omega} - d_{x_i}\right\}$$

$$\phi_{i\omega}\big((S_i, f_i), x_i\big) = \big(S_i \cup \{x_i\}, \max\{r_{x_i}, f_i\} + t_{x_i\omega}\big)$$

We can use the same node merger operation as before.

Node Merger in SDDs

We need a concept of **one state relaxing another**.

Relaxation must have the property that state $\bar{\boldsymbol{S}}_i$ **relaxes** state $\boldsymbol{S}_i$ only if

$$p_{i\omega}(\bar{\boldsymbol{S}}_i, x_i)c_{i\omega}(\bar{\boldsymbol{S}}_i, x_i) \leq p_{i\omega}(\boldsymbol{S}_i, x_i)c_{i\omega}(\boldsymbol{S}_i, x_i)$$

for any control x_i and any outcome ω.

Node Merger in SDDs

We need a concept of **one state relaxing another**.

Relaxation must have the property that state $\bar{\boldsymbol{S}}_i$ **relaxes** state $\boldsymbol{S}_i$ only if

$$p_{i\omega}(\bar{\boldsymbol{S}}_i, x_i)c_{i\omega}(\bar{\boldsymbol{S}}_i, x_i) \leq p_{i\omega}(\boldsymbol{S}_i, x_i)c_{i\omega}(\boldsymbol{S}_i, x_i)$$

for any control x_i and any outcome ω.

Max clique problem: $\bar{S}_i$ relaxes S_i when $\bar{S}_i \subseteq S_i$.

Job sequencing problem: $(\bar{S}_i, \bar{f}_i)$ relaxes (S_i, f_i) when $\bar{S}_i \subseteq S_i$ and $\bar{f}_i \leq f_i$.

These definitions satisfy the property.

Node Merger in SDDs

Jointly sufficient conditions under which node merger yields a relaxed SDD:

(C1) State $\boldsymbol{S}_i \oplus \boldsymbol{S}'_i$ relaxes both $\boldsymbol{S}_i$ and $\boldsymbol{S}'_i$.

(C2) If state $\bar{\boldsymbol{S}}_i$ relaxes state $\boldsymbol{S}_i$, then $\phi_{i\omega}(\bar{\boldsymbol{S}}_i, x_i)$ relaxes $\phi_{i\omega}(\boldsymbol{S}_i, x_i)$ for any ω, x_i.

Note: (C1) and (C2) are sufficient for deterministic BDDs

JH (2017)

Node Merger in SDDs

Jointly sufficient conditions under which node merger yields a relaxed SDD:

(C1) State $\boldsymbol{S}_i \oplus \boldsymbol{S}'_i$ relaxes both $\boldsymbol{S}_i$ and $\boldsymbol{S}'_i$.

(C2) If state $\bar{\boldsymbol{S}}_i$ relaxes state $\boldsymbol{S}_i$, then $\phi_{i\omega}(\bar{\boldsymbol{S}}_i, x_i)$ relaxes $\phi_{i\omega}(\boldsymbol{S}_i, x_i)$ for any ω, x_i.

(C3) If state $\bar{\boldsymbol{S}}_i$ relaxes state $\boldsymbol{S}_i$, then given any control x_i and any set of numbers $\{\eta_\omega \mid \text{all } \omega\}$, there is a control $\bar{x}_i$ such that

$$\sum_\omega p_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i)\big(c_{i\omega}(\bar{\boldsymbol{S}}_i, \bar{x}_i) + \eta_\omega\big) \le \sum_\omega p_{i\omega}(\boldsymbol{S}_i, x_i)\big(c_{i\omega}(\boldsymbol{S}_i, x_i) + \eta_\omega\big)$$

Note: (C1) and (C2) are sufficient for deterministic BDDs

JH (2017)

Node Merger in SDDs

- Key to proofs: work with **fully articulated** SDDs.
  - All states are represented, even those that are reached with zero probability.
  - Node merger becomes rearrangement of probabilities.

Lemma. If condition (C2) is satisfied, and $\bar{\boldsymbol{S}}_i$ relaxes $\boldsymbol{S}_i$, then cost to go of $\bar{\boldsymbol{S}}_i \leq$ cost to go of $\boldsymbol{S}_i$.

Proof by backward induction on layers.

Node Merger in SDDs

Theorem. If (C1)-(C3) are satisfied, then node merger yields a relaxed SDD.

Proof by forward induction on layers of partially compiled SDDs.

Node Merger in SDDs

Theorem. If (C1)-(C3) are satisfied, then node merger yields a relaxed SDD.

Proof by forward induction on layers of partially compiled SDDs.

Corollary. The max clique state merger operation yields a relaxed SDD.

The operation satisfies (C1)–(C3), but the proof is nontrivial due to the strength of condition (C3).

Node Merger in SDDs

Lemma. If probabilities are state-independent, then a merger operation that satisfies (C1) and (C2) also satisfies (C3).

Corollary. The job sequencing merger operation yields a relaxed SDD.

Probabilities are state-independent, and it is easy to show the merger operation satisfies (C1)–(C2).

Computational Experiments

- Use **stochastic max clique problem** as test case.
  - Why? Relaxed BDDs have been shown to provide good bounds for the **deterministic** problem.
  - …tighter bounds than the **full cutting plane resources** of a state-of-the-art MIP solver, in less time, based on DIMACS instances.

Bergman, Cire, van Hoeve, JH (2013)

  - So BDDs may also yield useful bounds for the **stochastic** problem.
  - …although we cannot compare with an MIP solver, since **no practical MIP model** exists for the stochastic problem.

Computational Experiments

- The main challenge is finding **tractable** and yet **nontrivial** instances.
  - **Nearly all** of the DIMACS instance are intractable as a stochastic problem.
  - We solved only **5 instances**, 3 of which were trivial, and 1 of which required 24 hours to solve.

Computational Experiments

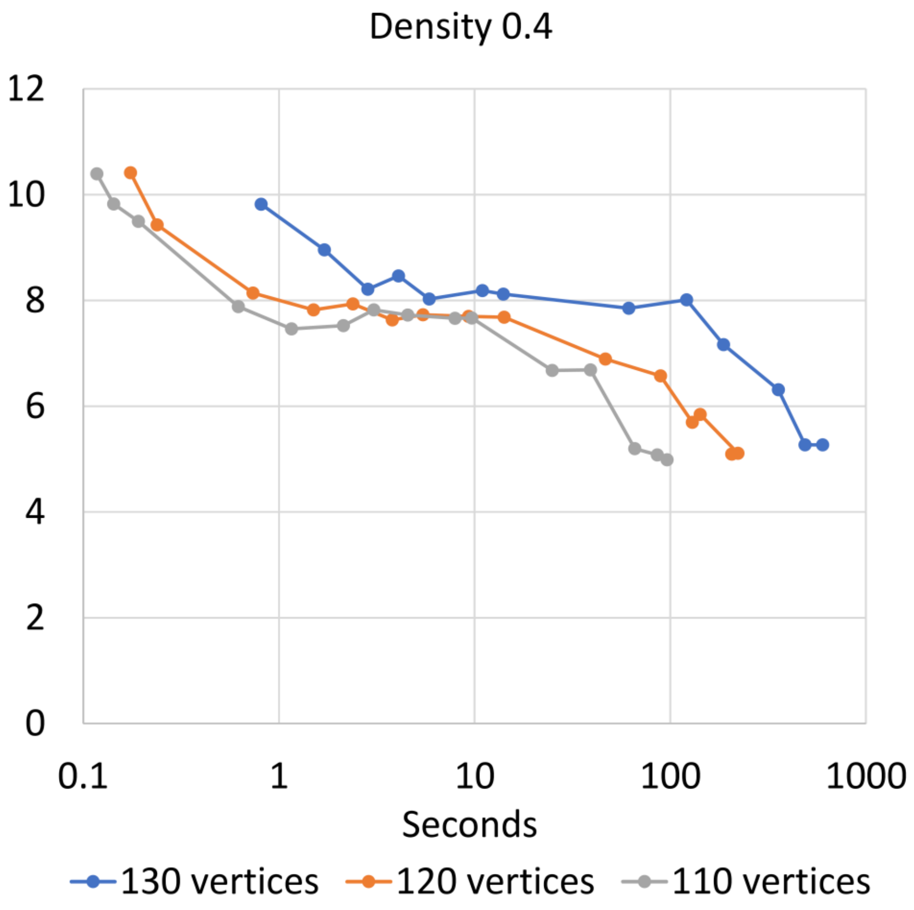
- The main challenge is finding **tractable** and yet **nontrivial** instances.
  - **Nearly all** of the DIMACS instance are intractable as a stochastic problem.
  - We solved only **5 instances**, 3 of which were trivial, and 1 of which required 24 hours to solve.
- We therefore obtained SDD-based bounds for both **random** and **DIMACS** instances.
  - Random instances **sized** to be tractable and nontrivial.
  - Exact solutions found with **complete** SDDs, since state space enumeration is the only available method.
  - DIMACS bounds were not compared with optimal solutions (2 exceptions).

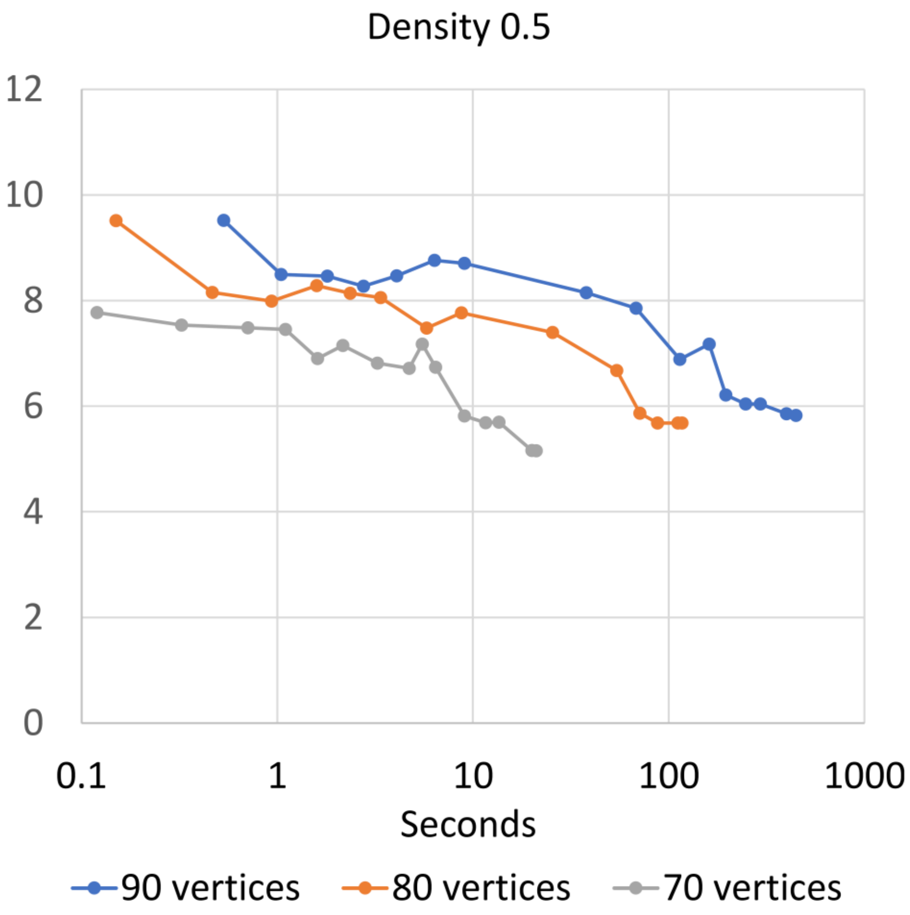
Computational Experiments

- **Merger heuristic** is based on previous experience with deterministic problem.
  - Merge **less attractive** nodes first.
  - That is, nodes with **shortest paths** to root node in deterministic problem.
  - These are less likely to be part of an optimal solution of the relaxed SDD.
- **Control size** of relaxed SDD by limiting **width**.
  - Width = max number of nodes in a layer.
  - To save time, we do not check whether state resulting from a merger already occurs in the layer.

Random instances

Solved to optimality

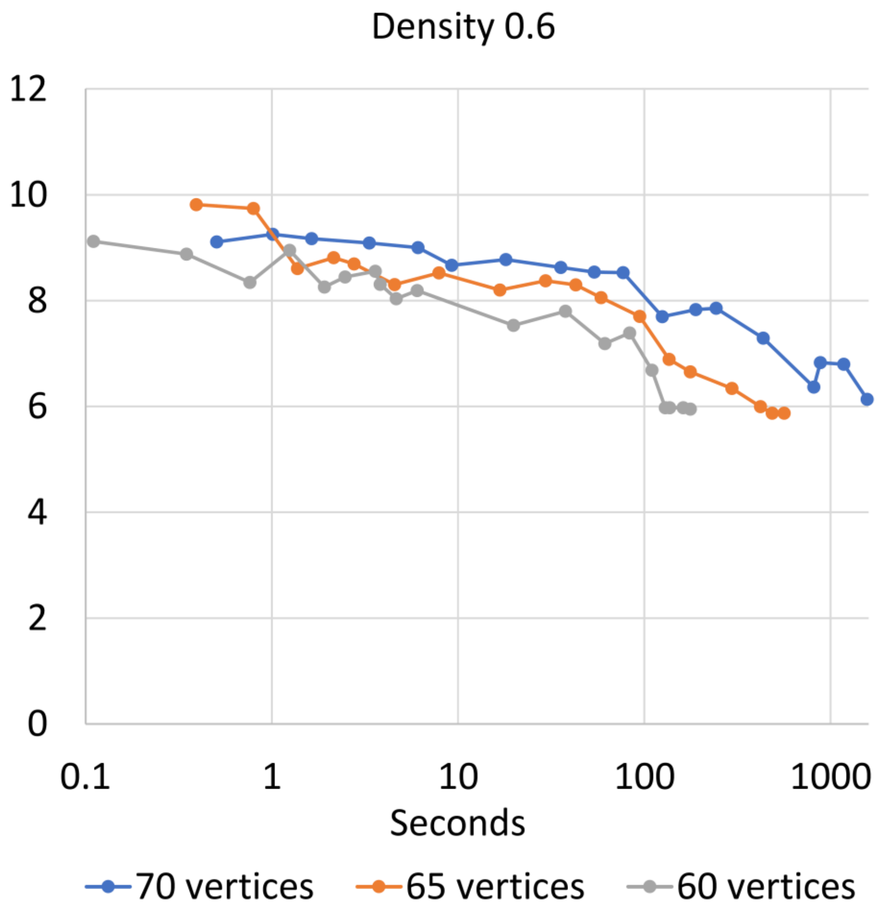


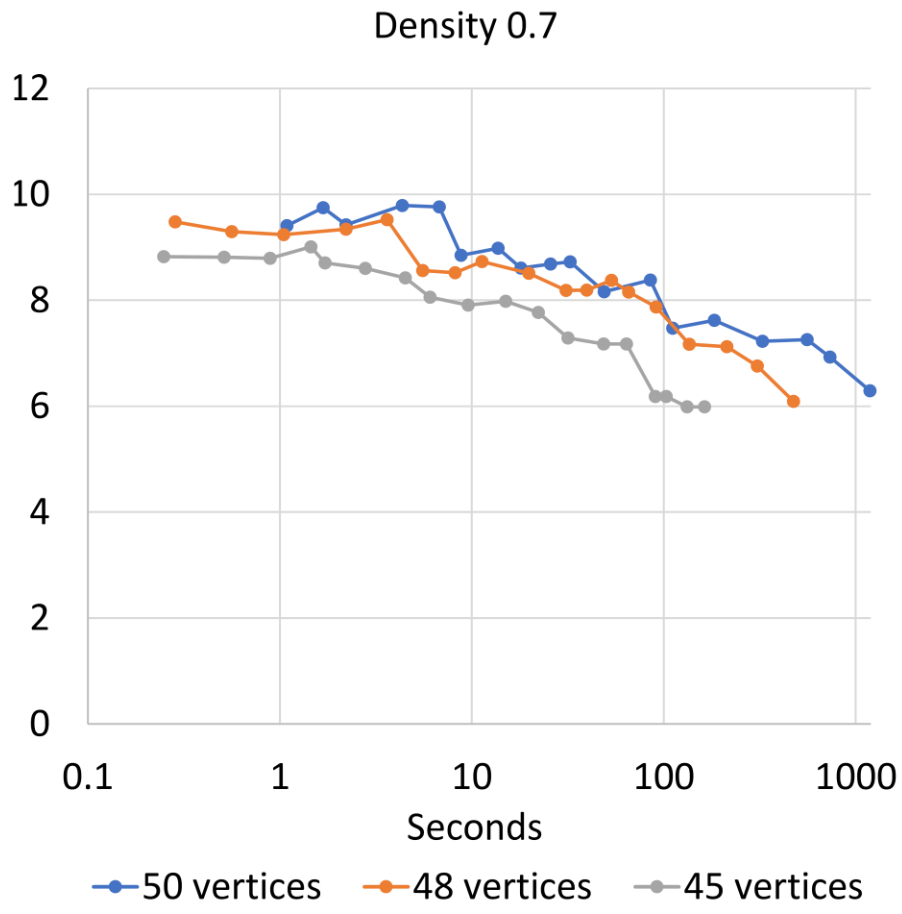


Random instances

Solved to optimality

Bound vs time





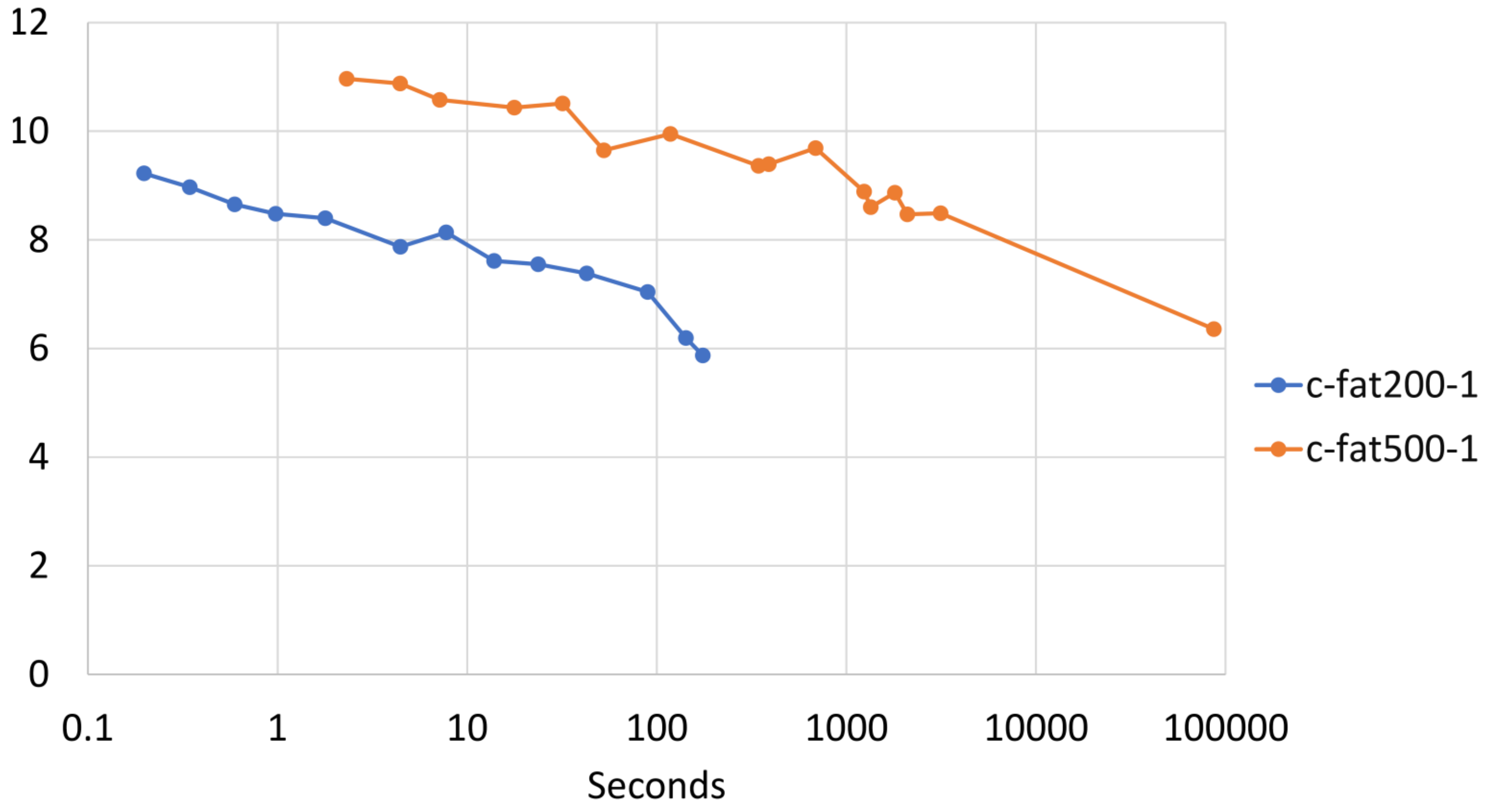
DIMACS instances

Instance	Vertices	Density	Instance	Vertices	Density
brock200_1	200	0.7417	hamming6-2	64	0.8906
cfat200-1	200	0.0767	johnson8-4-4	70	0.7571
cfat500-1	500	0.0357	keller4	171	0.6453
c125.9	125	0.8913	p_hat300-1	300	0.2430
DSJC500_5	500	0.5010	san200_0.7_1	200	0.6965
gen200_p0.9_44	200	0.8955	sanr_0.7	200	0.6934

2 DIMACS instances

Solved to optimality

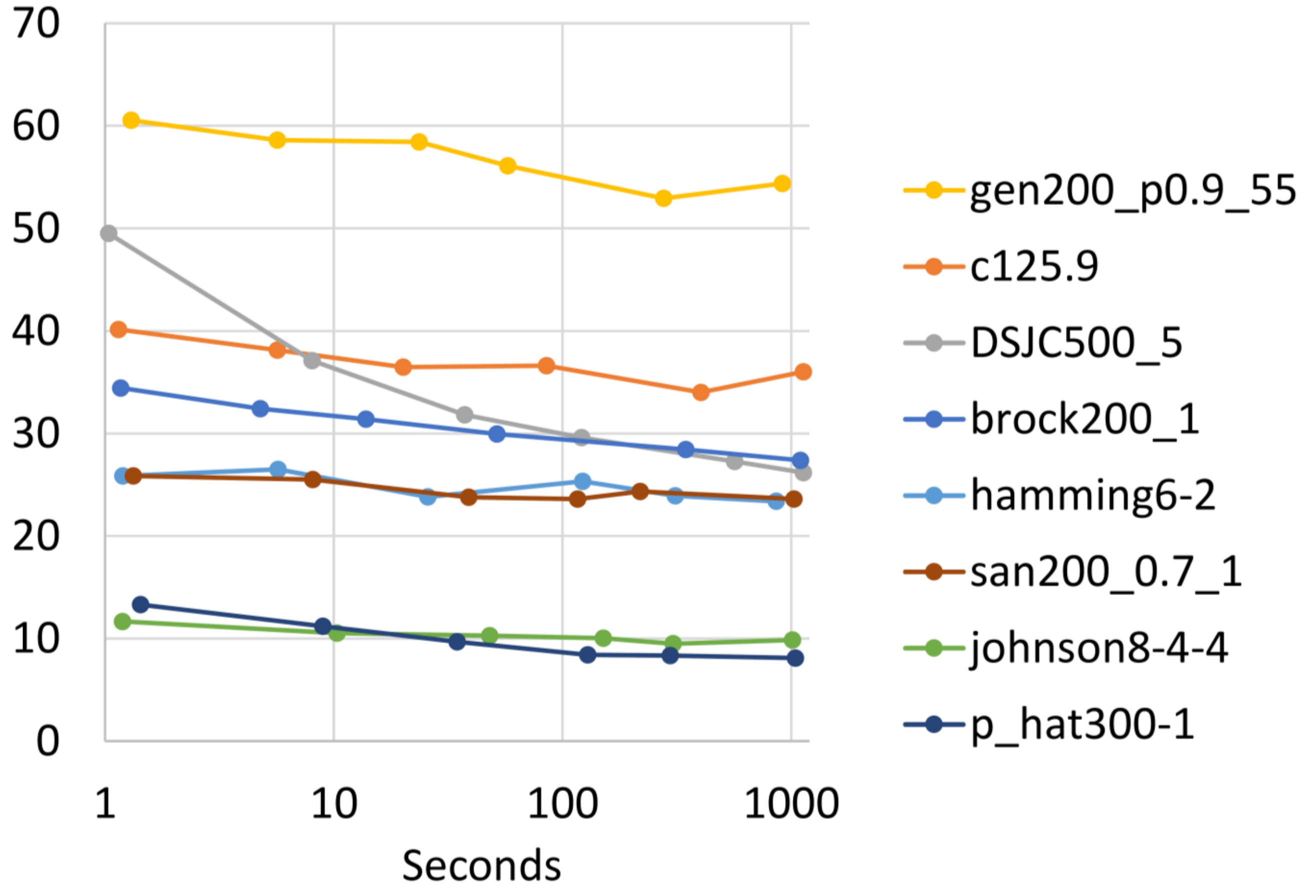
Bound vs time



DIMACS instances

Not solved to optimality

Bound vs time



Computational Experiments

- Bound quality degrades **gradually** with reduction in SDD width/time investment**.**
  - Even reduction down to a **few seconds**.
  - Indicates that SDDs can provide **useful bounds** for DP models.
- Roughly **logarithmic** relationship**.**
  - In most cases.
  - May allow estimate of how bound will **improve** with greater time investment.

Research Issues

- Use SDD bounds to solve moderate-sized problems by **branch and bound.**
  - Based on previous experience with deterministic problems**.**
- Use relaxed SDDs to compute bounds for **approximate DP**.
  - Find solution with traditional approximate DP, which **estimates** costs to go.
  - Use relaxed SDDs to compute **bounds** on costs to go, using same controls as in approximate DP solution.

References and further details are in:

J. N. Hooker, Stochastic decision diagrams, submitted.

Traditional state space relaxation

- Requires creation of alternate (smaller) state space for every problem.

Christofides, Mingozzi, Toth (1981)
Baldacci, Mingozzi, Roberti (2012)

  - General practice is to use **approximate DP** instead.

Powell (2011)

Traditional state space relaxation

- Advantages of SDD-based relaxation**.**
  - Uses **same state variables** as original problem.
  - This allows SDD-based **branch-and-bound** method to solve problem.

Bergman, Cire, van Hoeve, JH (2014)

  - Relaxation constructed **dynamically**.
  - Can be **tightened** by filtering, Lagrangian relaxation.

Bergman, Cire, van Hoeve (2015); JH (2017, 2019)

  - Can be sized to provide bound of **any desired quality**.