Stochastic Decision Diagrams

John Hooker
Carnegie Mellon University

ICS 2022

Motivation

- Decision diagrams (DDs) 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 DDs to **stochastic DDs** (SDDs) by adding probabilities to arcs.

Deterministic DDs

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

- A **weighted** DD can represent a discrete **optimization problem**. Hadzic and JH (2006, 2007)

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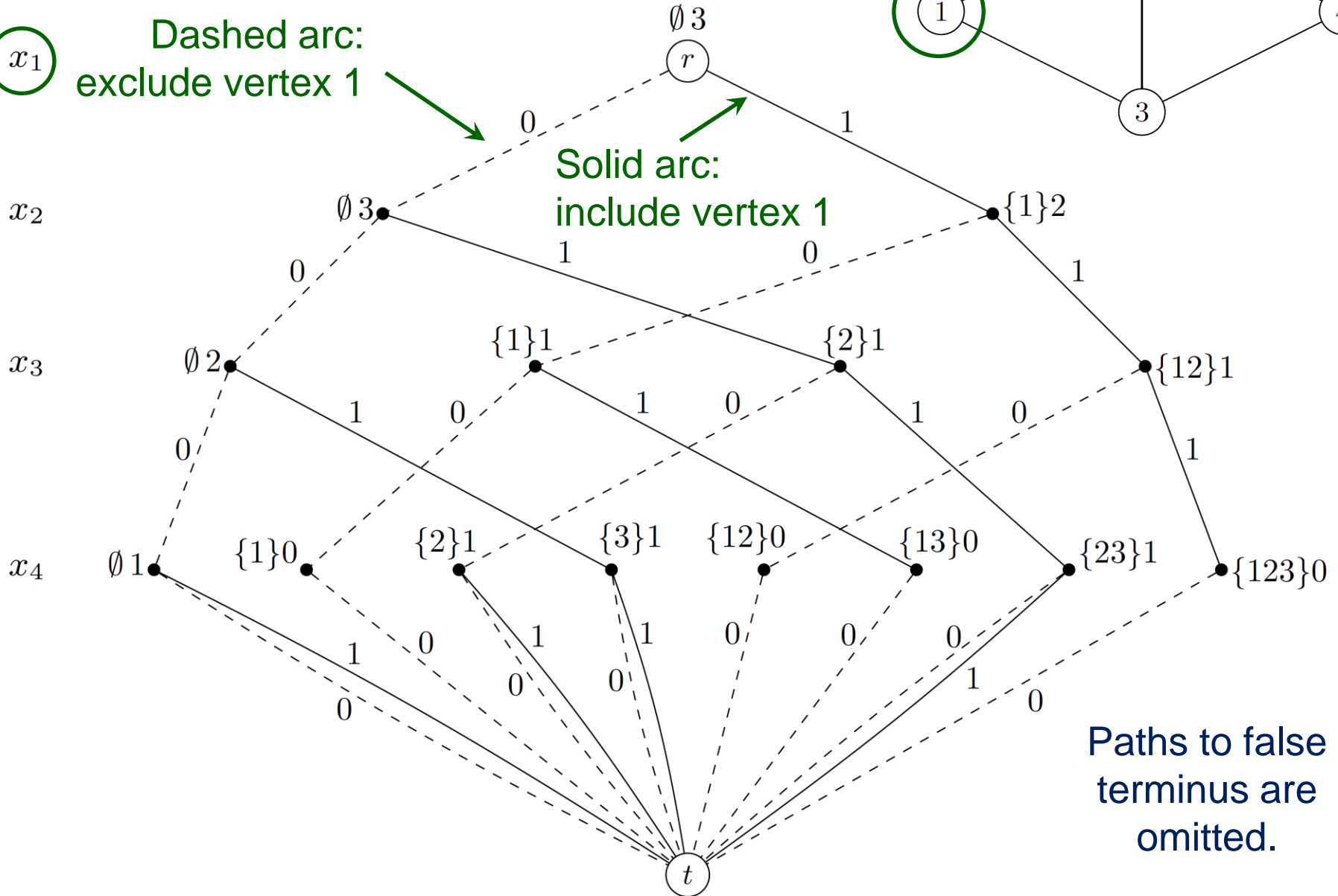
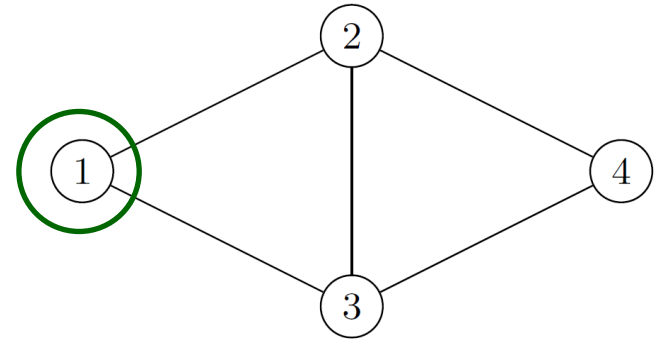# Maximum clique

Max clique example

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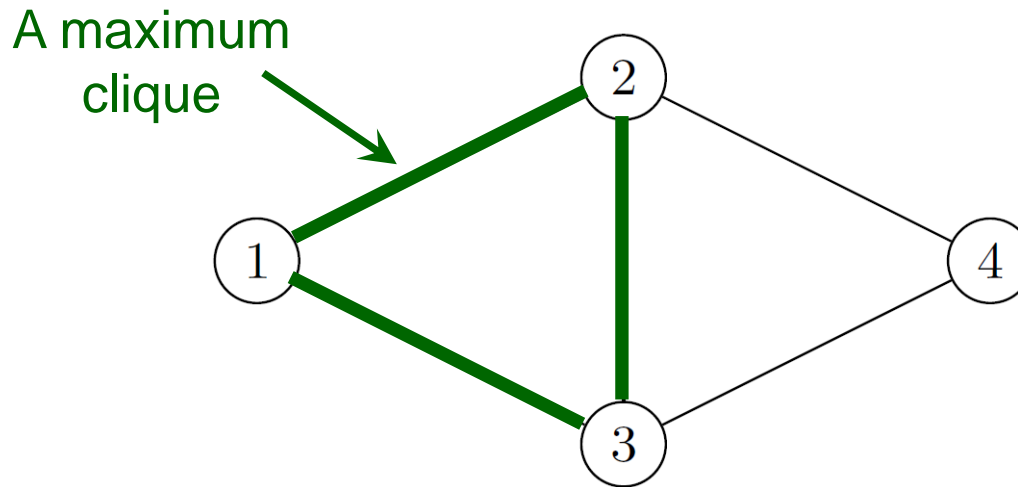
Let binary variable $x_i = 1$
when vertex i is in the clique.

Weighted DD

Dashed arc:
exclude vertex 1

Solid arc:
include vertex 1

Paths to false
terminus are
omitted.

Weighted DD

Arc cost (here we maximize cost)
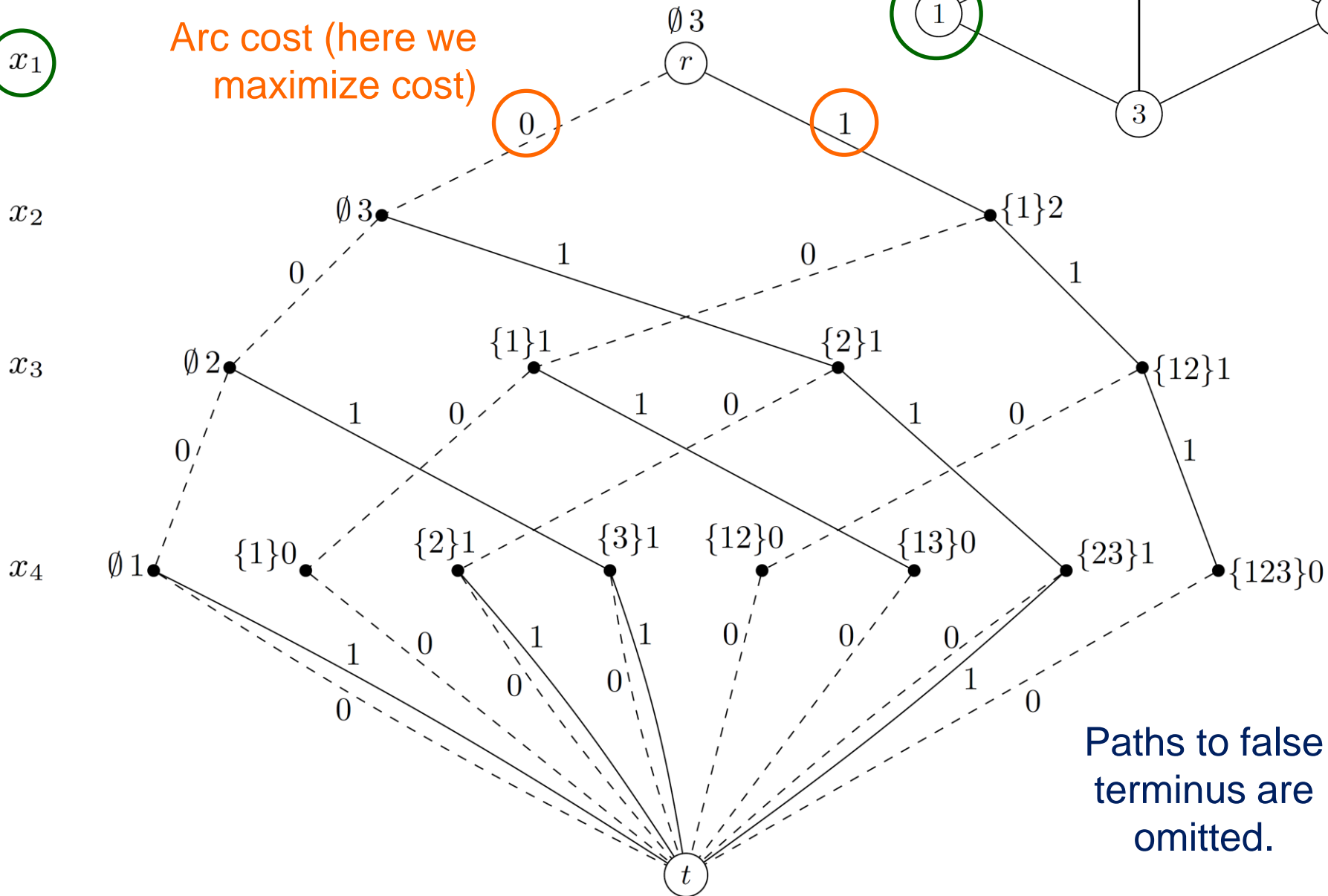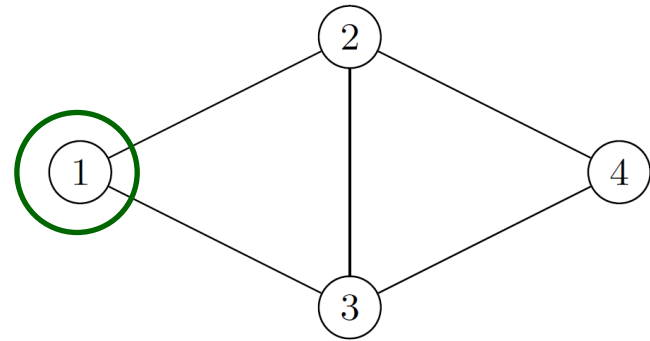
Paths to false terminus are omitted.

Weighted DD

Each *r-t* path corresponds to a clique

x_1

$\emptyset\,3$
r

0

1

x_2

$\emptyset\,3$

$\{1\}2$

0

1

0

1

x_3

$\emptyset\,2$

$\{1\}1$

$\{2\}1$

$\{12\}1$

0

1

0

1

0

1

0

1

x_4

$\emptyset\,1$

$\{1\}0$

$\{2\}1$

$\{3\}1$

$\{12\}0$

$\{13\}0$

$\{23\}1$

$\{123\}0$

1

0

0

1

1

0

0

0

0

1

0

t
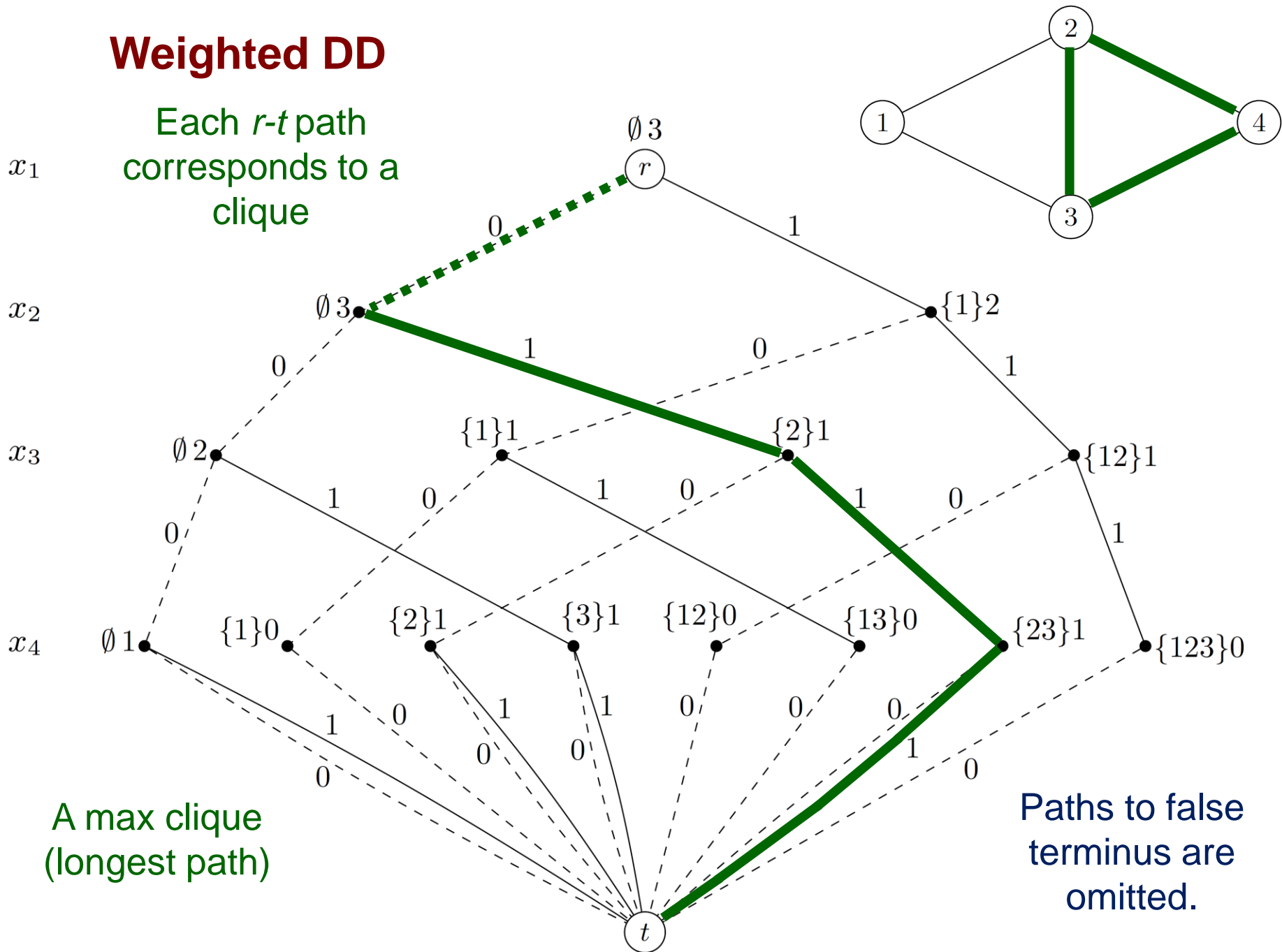
A max clique (longest path)

Paths to false terminus are omitted.

Weighted DD

Each *r-t* path corresponds to a clique

x_1

x_2

x_3

x_4

$\emptyset\, 3$

r

0

1

$\emptyset\, 3$

0

1

0

$\{1\}2$

1

$\emptyset\, 2$

0

$\{1\}1$

1

0

$\{2\}1$

1

0

$\{12\}1$

1

$\emptyset\, 1$

1

0

$\{1\}0$

1

0

$\{2\}1$

1

0

$\{3\}1$

1

0

$\{12\}0$

0

$\{13\}0$

0

$\{23\}1$

1

0

$\{123\}0$

t

Another max clique

Paths to false terminus are omitted.
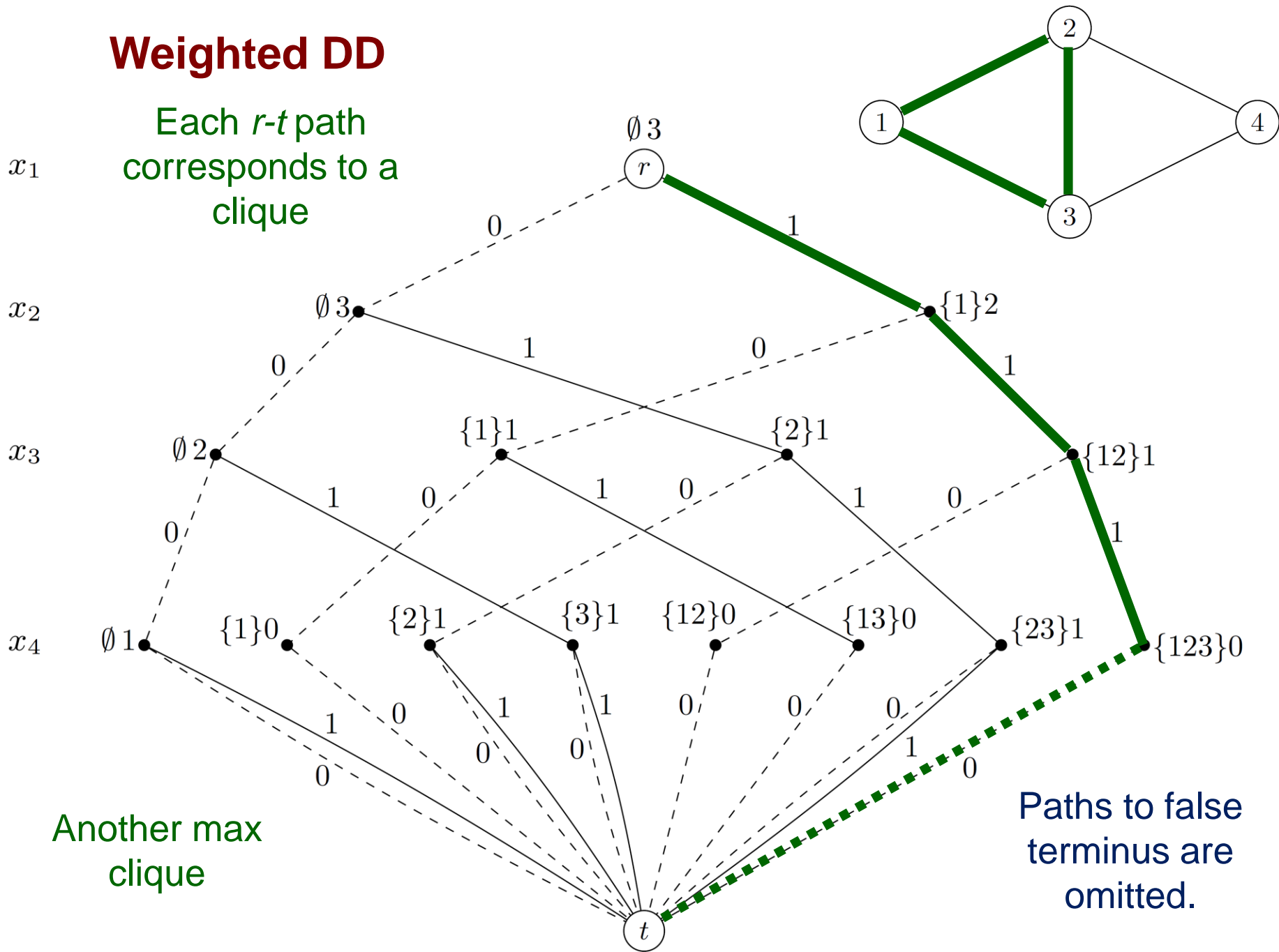
Dynamic Programming

- The **state transition graph** of a dynamic programming (DP) problem can be interpreted as a DD (or MDD).

  - By associating states with nodes of the DD.

  - This opens the door to using DD **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 DDs

Max clique DP model

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

The recursion is

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

$x_i = 0$ $x_i = 1$

Deterministic DDs

Max clique DP model

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

The recursion is

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

$x_i = 0$ $x_i = 1$

In general,

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

control   immediate cost   cost to go   state transition function
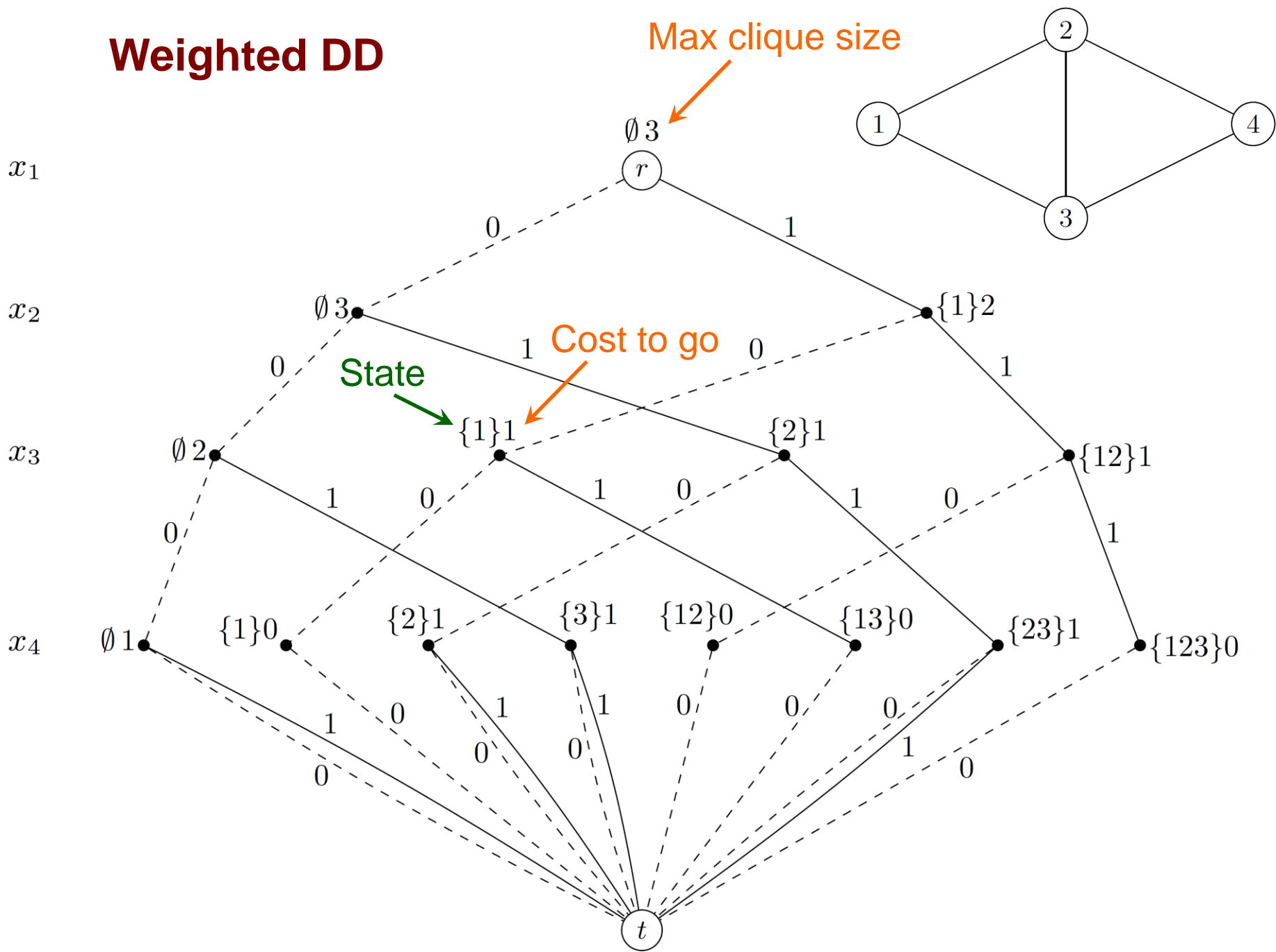
Weighted DD

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 DD

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

$\{\}0(4)$

Job

Tardiness

x_1

Cost to go

State

r

$3(0)$

$1(0)$

$2(0)$

x_2

$\{1\}3(4)$ $\{2\}3(4)$ $\{3\}3(6)$

$3(0)$

$2(2)$ $3(0)$ $1(2)$ $2(2)$ $1(2)$

x_3 $\{12\}5(2)$ $\{13\}5(4)$ $\{12\}6(3)$ $\{23\}5(4)$ $\{13\}6(5)$

$2(4)$

$3(3)$ $1(4)$

$\boldsymbol{r} = (0, 1, 1)$
$\boldsymbol{d} = (5, 3, 5)$ $3(2)$ $2(5)$
$\boldsymbol{t} = (3, 2, 2)$

t
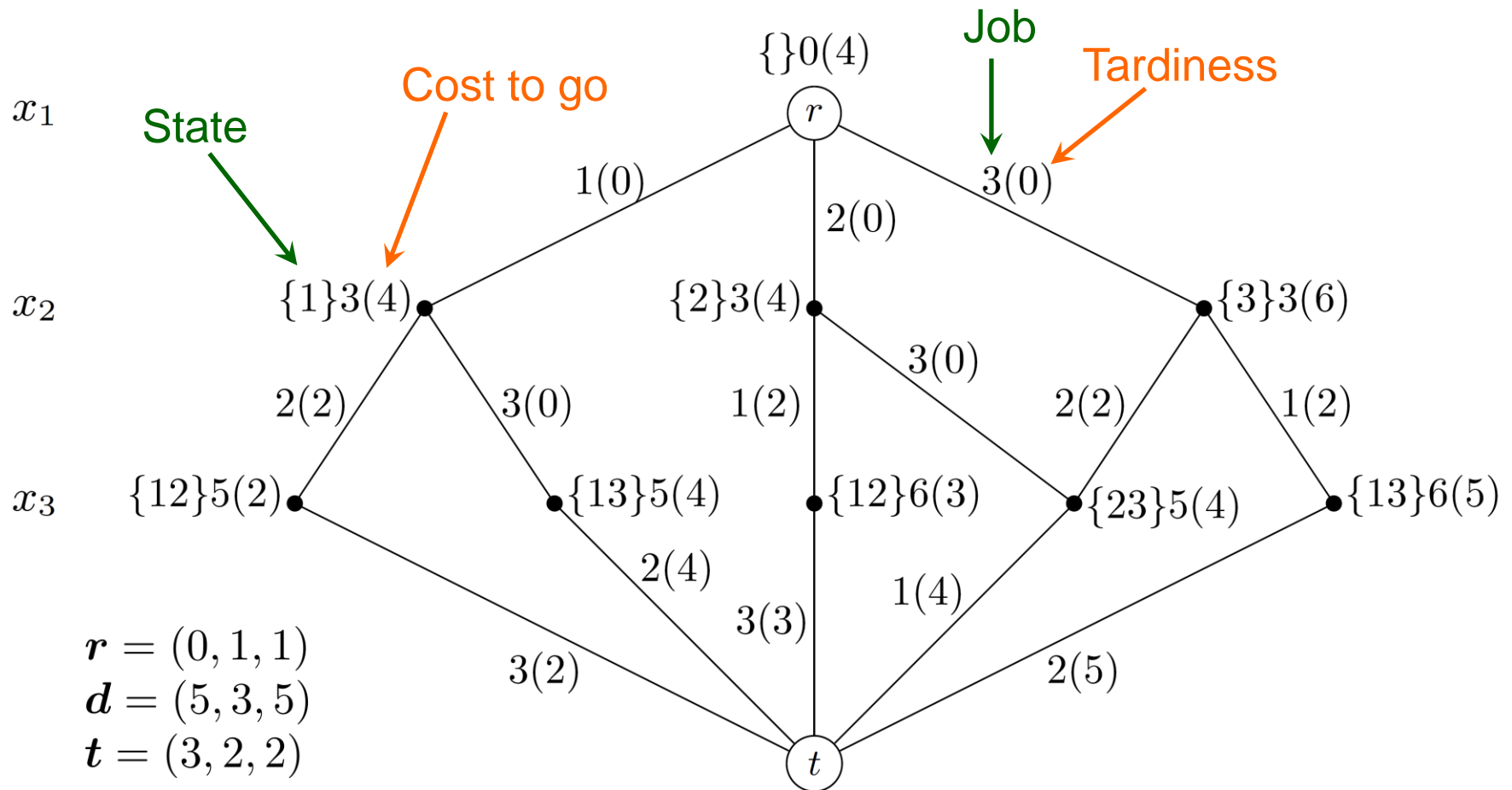
Deterministic DDs

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

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

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

Dynamic Programming

- Note: a state transition graph is a **different concept** than a DD.

    - A DD does not need **state information**.
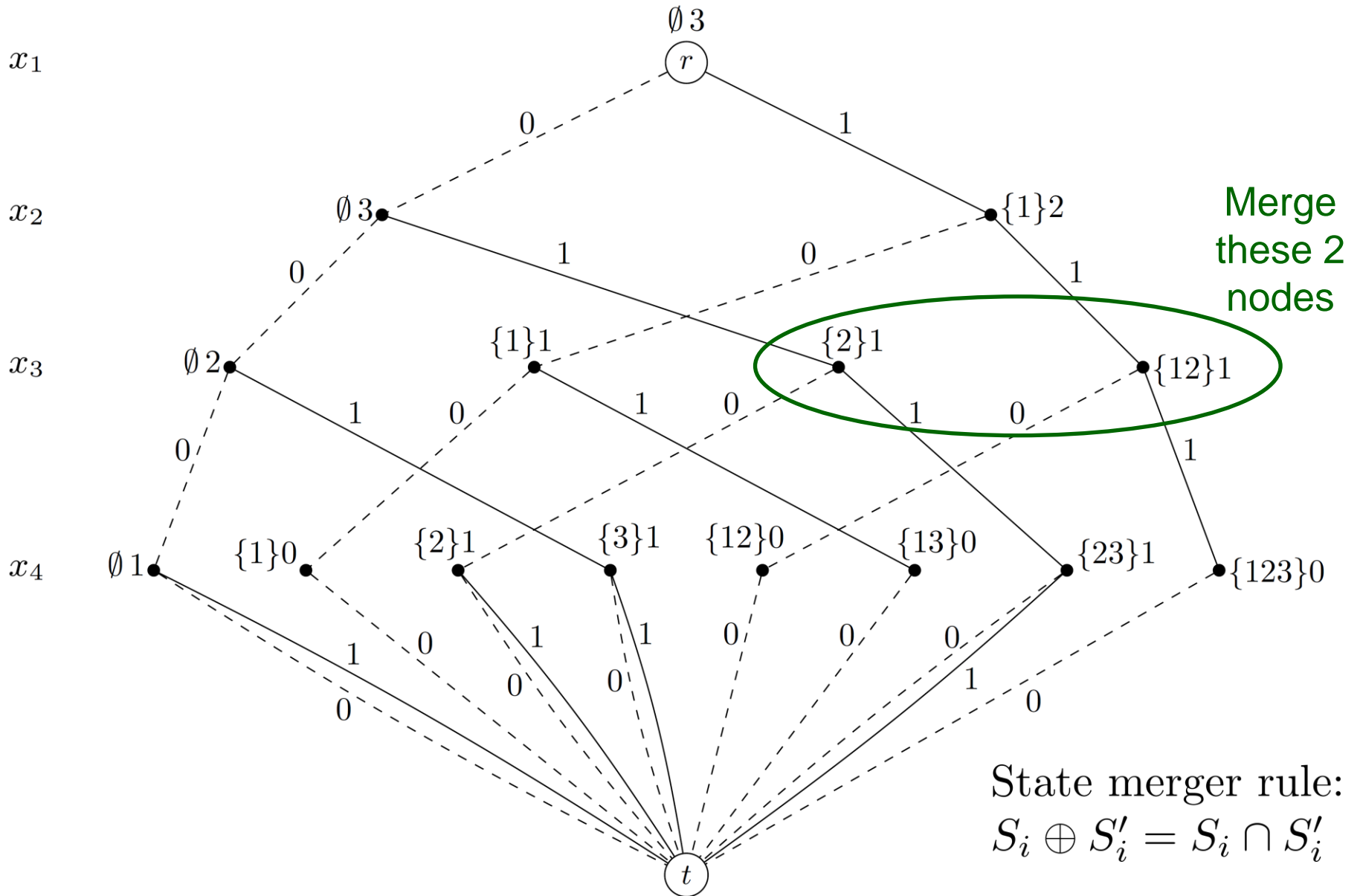
Dynamic Programming

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

- The DD perspective yields advantages:
  - A DD can be often be **reduced** by identifying isomorphic portions of the DD that are associated with different states. Bryant (1986 etc.)
    - This occasionally results in **radical** simplification (e.g., inventory management). JH (2013)
  - DDs 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 DDs…

Relaxed DDs

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

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

Weighted DD for max clique

x_1

x_2

x_3

x_4

$\emptyset\,3$

r

$\{1\}2$

Merge these 2 nodes

$\{1\}1$

$\{2\}1$

$\{12\}1$

$\emptyset\,2$

$\{1\}0$

$\{2\}1$

$\{3\}1$

$\{12\}0$

$\{13\}0$

$\{23\}1$

$\{123\}0$

$\emptyset\,1$

t

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

Relaxed DD for max clique

Longest path length of 4 is bound on optimal value 3

Result of merger
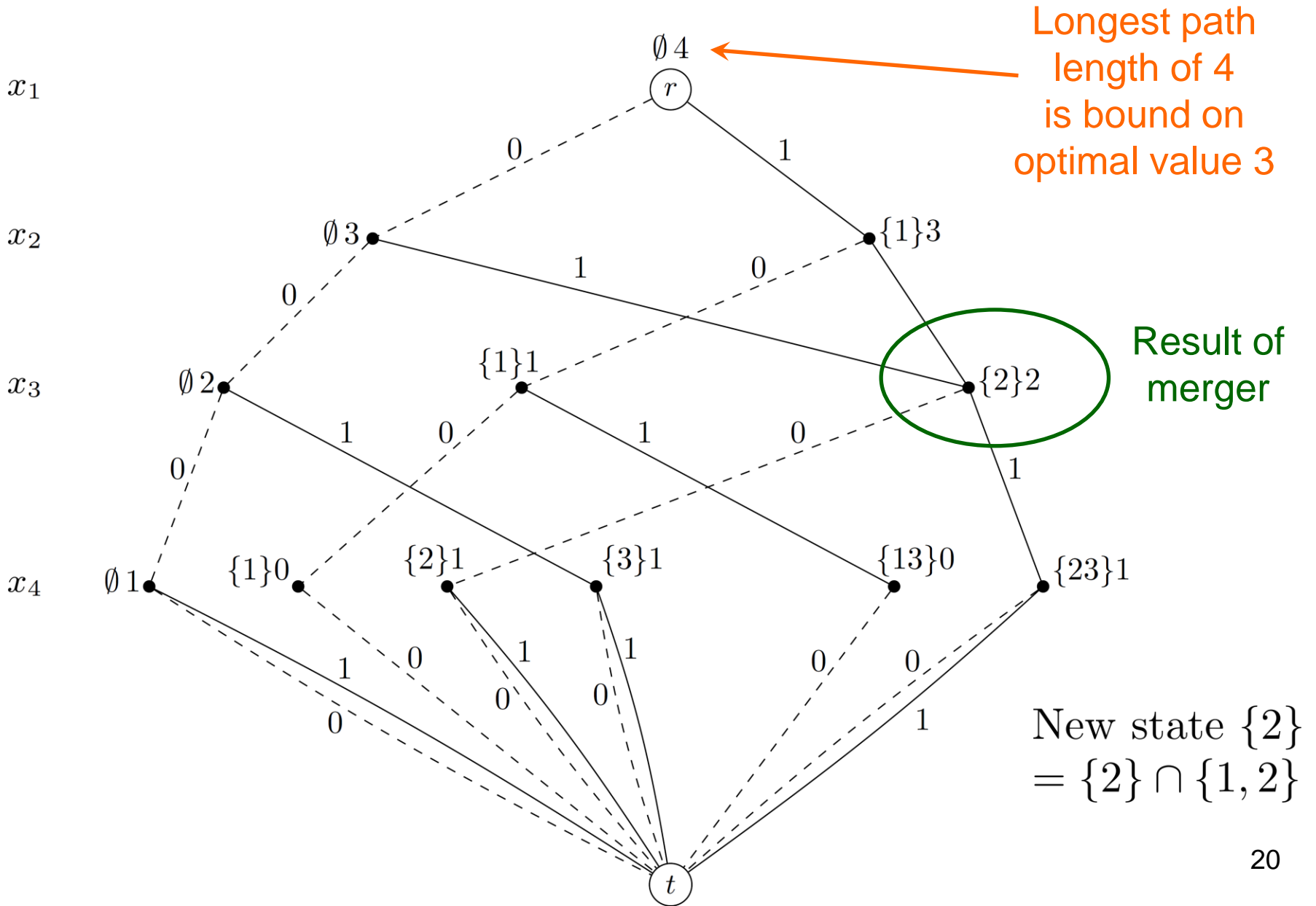
New state $\{2\}$
$= \{2\} \cap \{1, 2\}$

Weighted DD for job sequencing
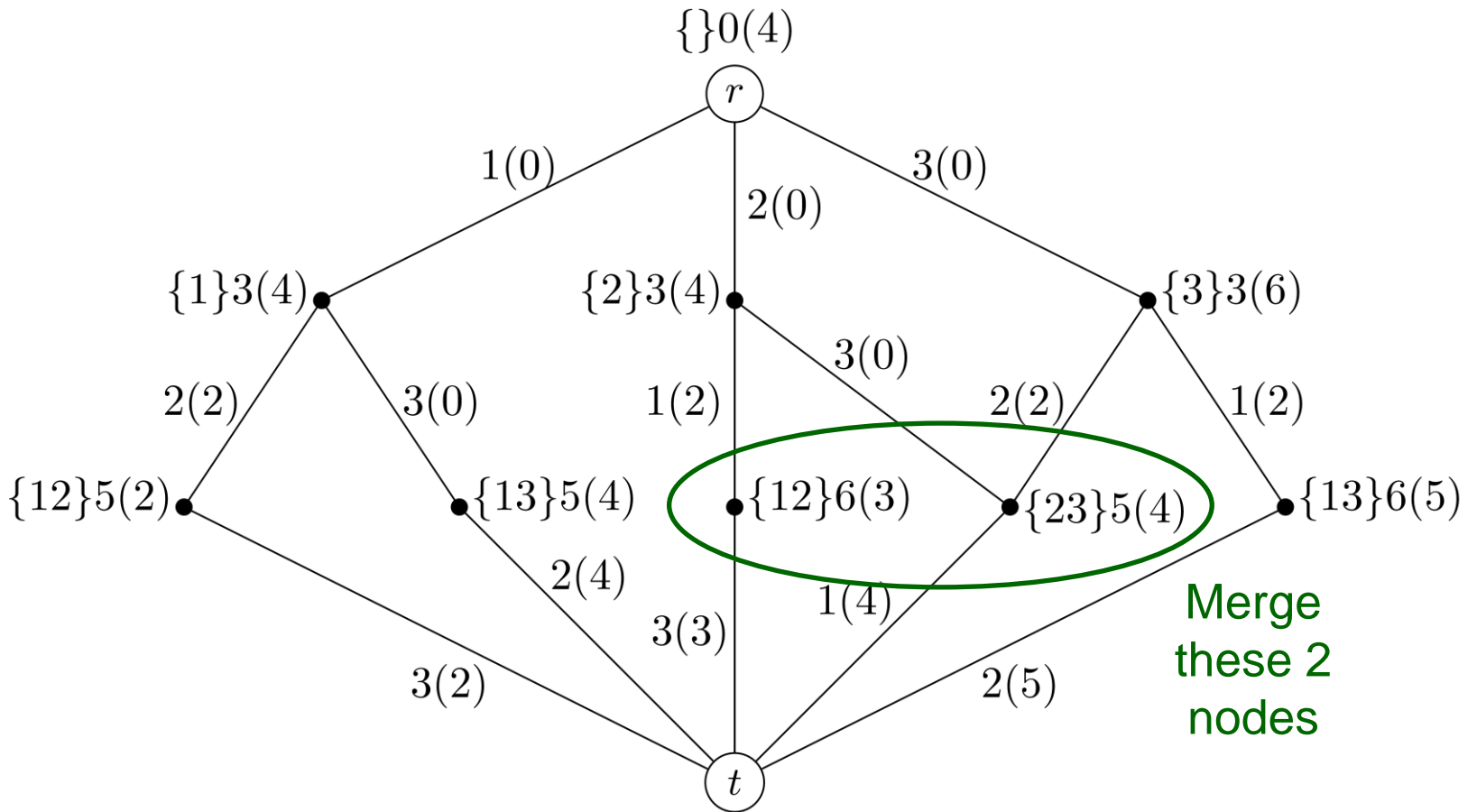
State merger rule: $(S_i, f_i) \oplus (S_i', f_i') = \left((S_i \cap S_i'), \min\{f_i, f_i'\}\right)$

Relaxed DD for job sequencing

Shortest path length of 2 is bound on optimal value 4

$\{\}0(2)$

r

$1(0)$

$2(0)$

$3(0)$

$\{1\}3(4)$

$\{2\}3(2)$

$3(0)$

$\{3\}3(4)$

$2(2)$

$3(0)$

$1(2)$

$2(2)$

$1(2)$

$\{12\}5(2)$

$\{13\}5(4)$

$\{2\}5(2)$

$\{13\}6(5)$

$2(4)$

$1(4)$

$3(2)$

$3(2)$

$2(5)$

Result of merger

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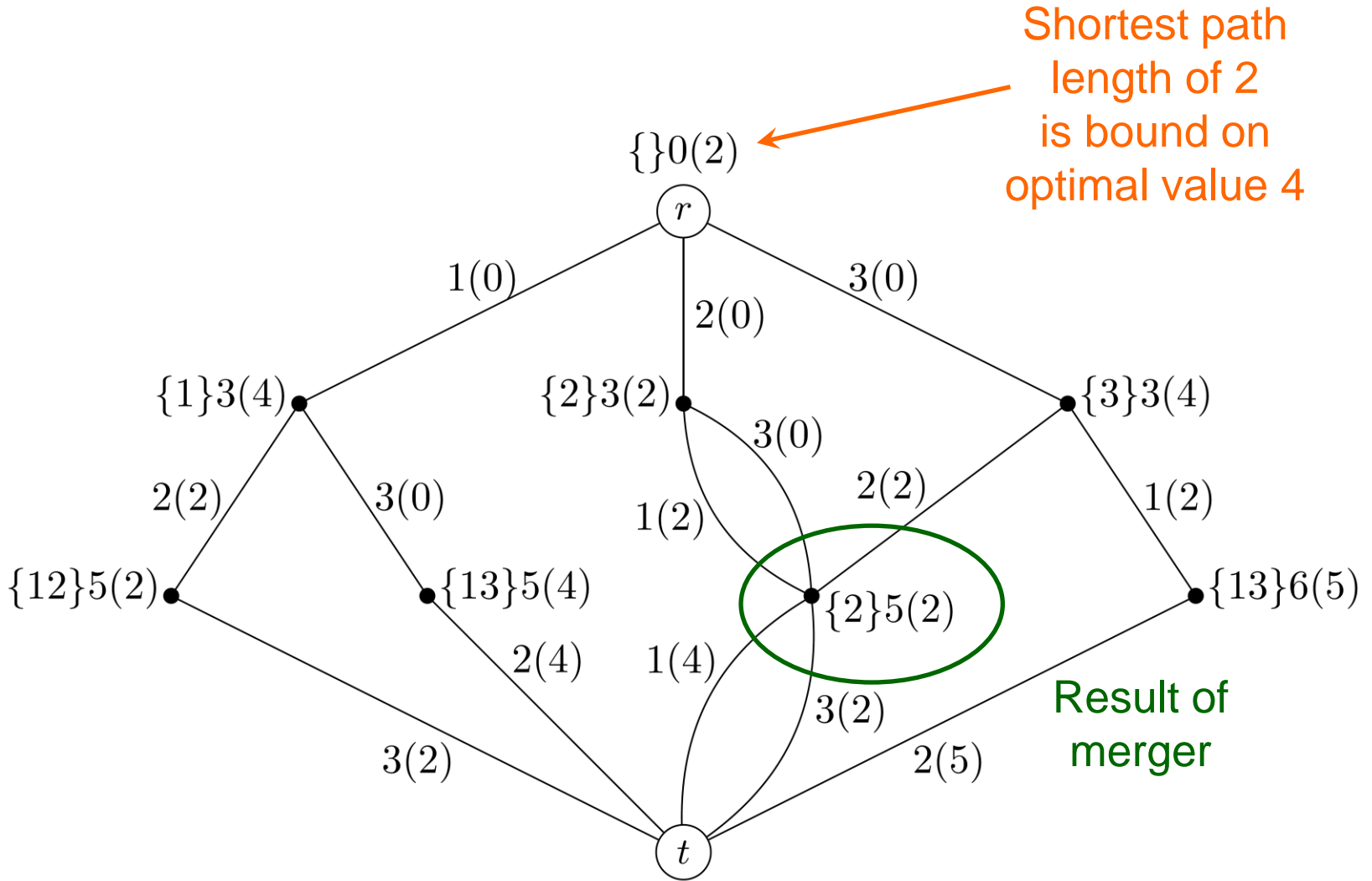
New state $\big(\{2\}, 5\big) = \big(\{1, 2\} \cap \{2, 3\}, \min\{6, 5\}\big)$

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 DD-based relaxation**.**
  - Uses **same state variables** as original problem.
  - This allows DD-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**.

Stochastic DDs

- 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 DDs

Max clique example

Defined on a
random graph
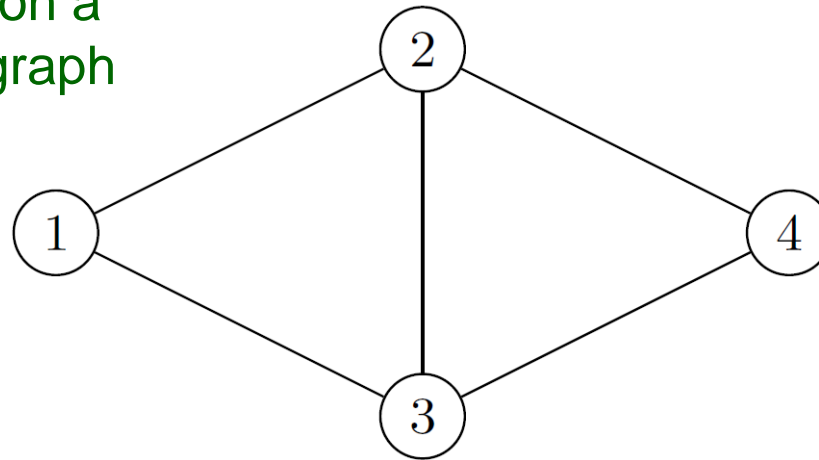
Each arc has probability 0.6

Objective is to maximize expected clique size.

Stochastic DD

Control $x_1 = 1$ has 2 possible outcomes 0,1

Transition probabilities

State S_i

$\emptyset\, 2.056$ — r

x_1

x_2 — $\emptyset\, 2.056$ — $\{1\}1.056$

x_3 — $\emptyset\, 1.6$ — $\{1\}0.6$ — $\{2\}1.056$ — $\{12\}0.36$

x_4 — $\emptyset\, 1$ — $\{1\}0$ — $\{2\}0.6$ — $\{3\}0.6$ — $\{12\}0$ — $\{13\}0$ — $\{23\}0.36$ — $\{123\}0$

t

0.4   0.6   0.64   0.36

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

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

Stochastic DD

Transition probabilities are state-dependent

x_1

$\emptyset\,2.056$ — r

0   1

x_2

$\emptyset\,2.056$ $\{1\}1.056$

State S_i

0   1   0   0   0.4   0.6   1

Transition probabilities

x_3

$\emptyset\,1.6$ $\{1\}0.6$ $\{2\}1.056$ $\{12\}0.36$

1   0   0.6   0   0.6   0   0.64   0.36

0   0.4   0   0.4   0   0

0   0   1   1   1

x_4

$\emptyset\,1$ $\{1\}0$ $\{2\}0.6$ $\{3\}0.6$ $\{12\}0$ $\{13\}0$ $\{23\}0.36$ $\{123\}0$

0.6   0.6   0   0   0   0.36

1   0   1   1   1

0   0   0

t

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

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

Stochastic DDs

Max clique DP model

The recursion is

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

$x_i = 0$

$x_i = 1$

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

Stochastic DDs

Max clique DP model

The recursion is

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

$$\uparrow \qquad\qquad\qquad\qquad \uparrow$$

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

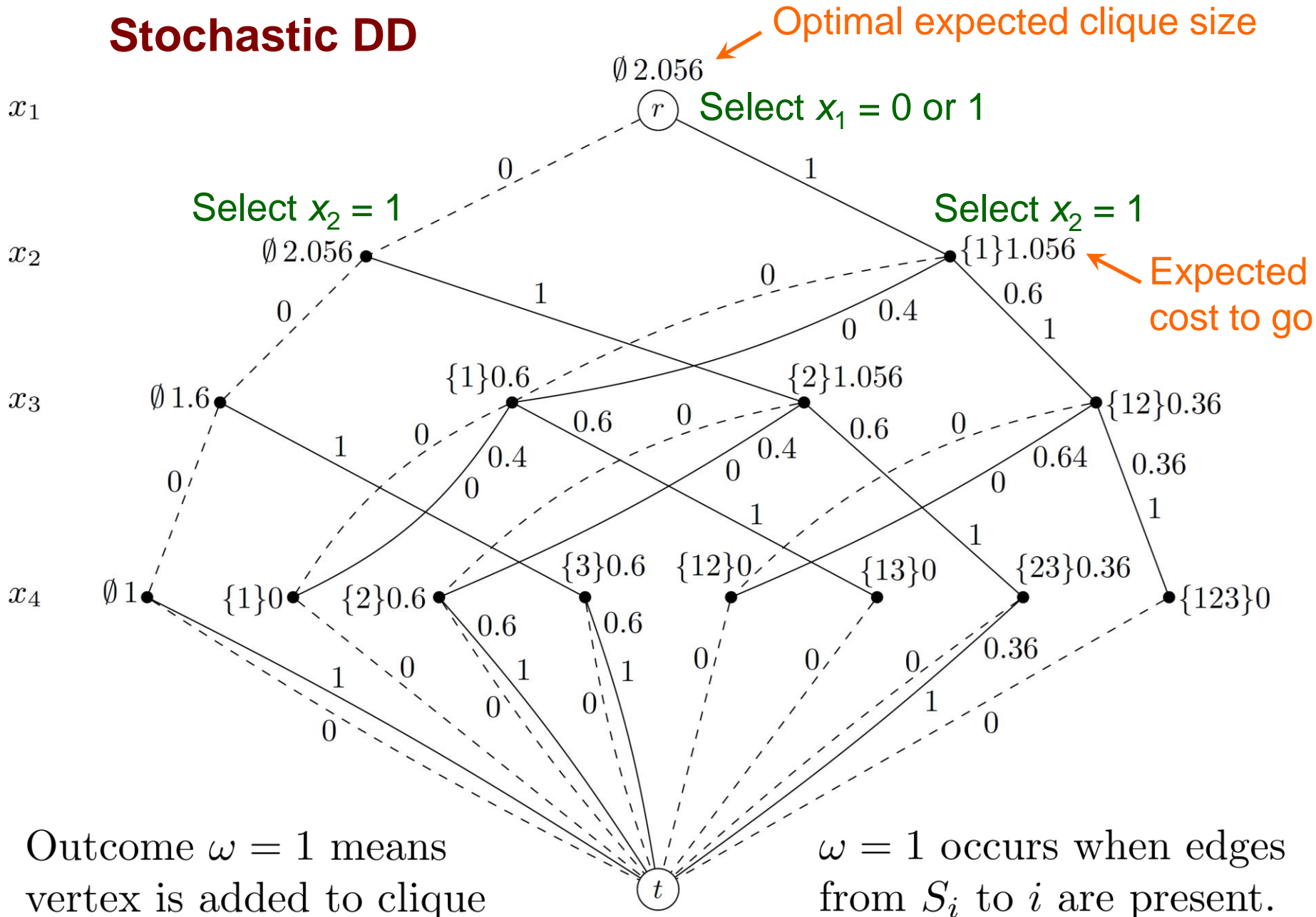where $p(S_i) = $ probability that vertex i can be added to clique

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 DD

Optimal expected clique size

$\emptyset$ 2.056

x_1 — r — Select $x_1 = 0$ or 1

0   1

Select $x_2 = 1$ Select $x_2 = 1$

x_2 $\emptyset$ 2.056 $\{1\}$1.056

0   1   0   0   0.4   0.6   1   Expected cost to go

x_3 $\emptyset$ 1.6 $\{1\}$0.6 $\{2\}$1.056 $\{12\}$0.36

0   1   0   0.4   0.6   0   0   0.4   0.6   0   0.64   0.36

0   0   1   1   0   1

x_4 $\emptyset$ 1 $\{1\}$0 $\{2\}$0.6 $\{3\}$0.6 $\{12\}$0 $\{13\}$0 $\{23\}$0.36 $\{123\}$0

1   0   0.6   1   0.6   1   0   0   0   0.36   1

0   0   0   1   0

t

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

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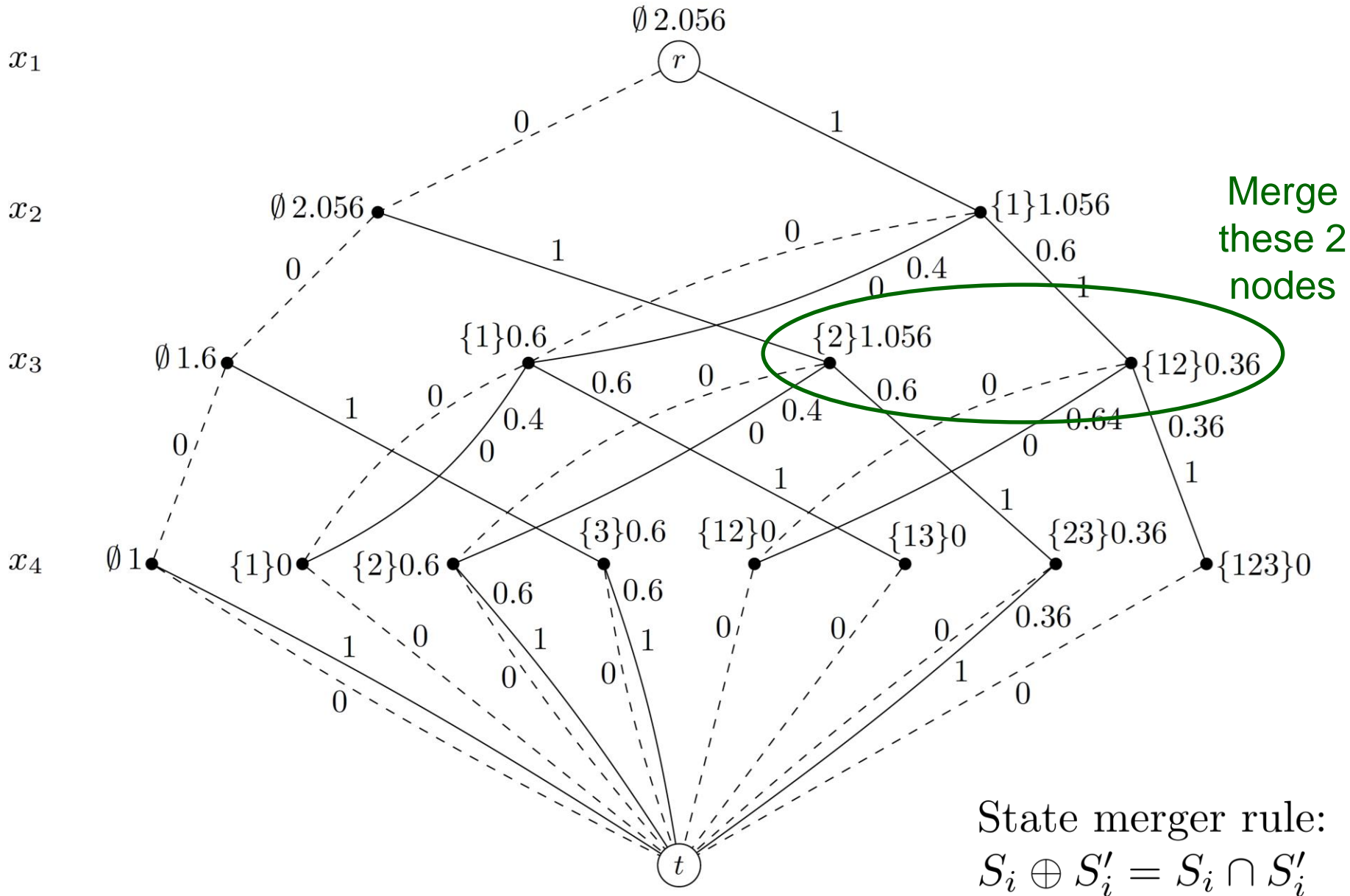
Relaxed SDDs

- 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.
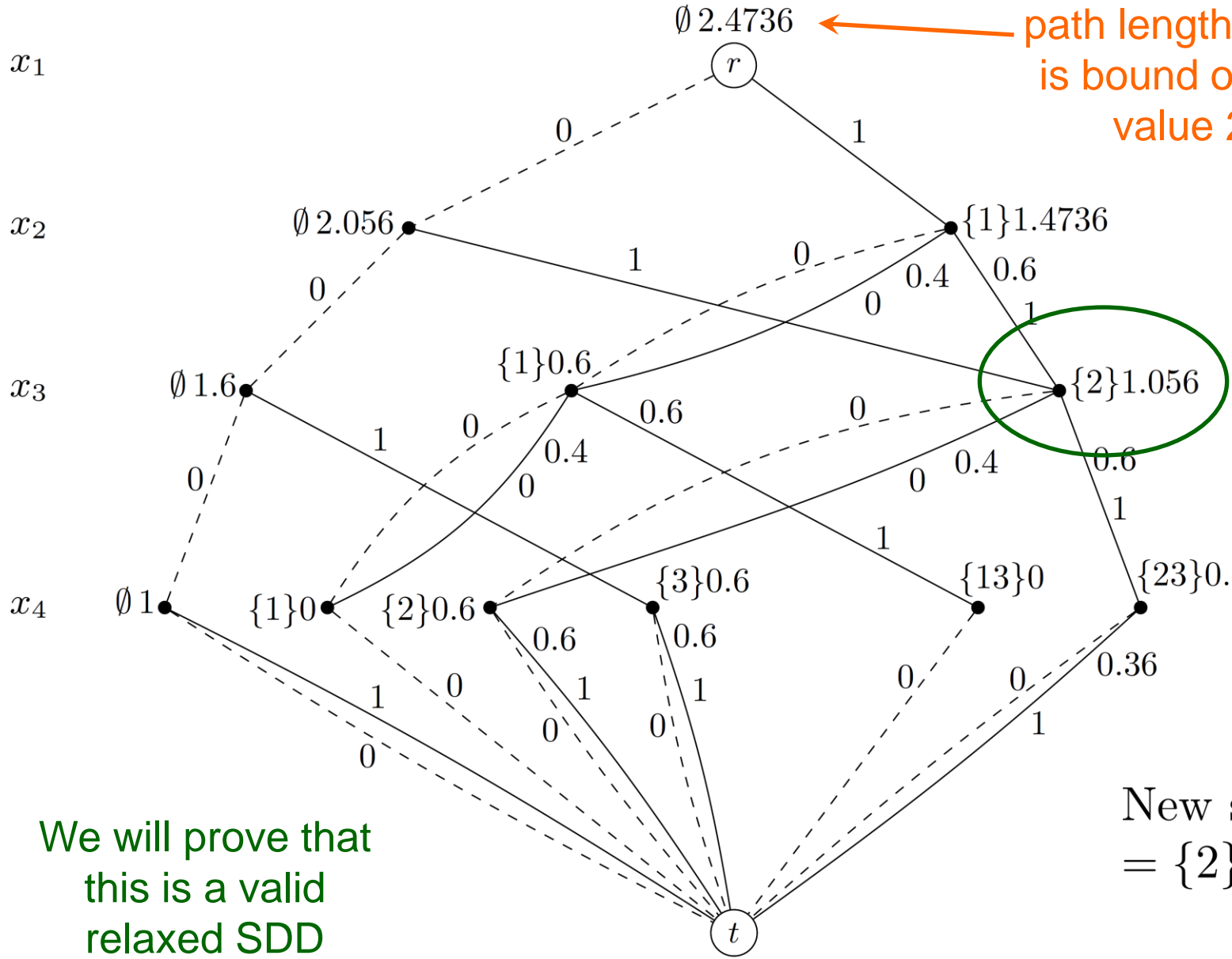
Relaxed SDDs

- 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…

33

Stochastic DD for max clique

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

Relaxed DD for max clique

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

Result of merger

We will prove that this is a valid relaxed SDD

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

x_1

x_2

x_3

x_4

$\emptyset$ 2.4736

r

0

1

$\emptyset$ 2.056

$\{1\}$1.4736

0

1

0

0.4

0.6

0

$\emptyset$ 1.6

$\{1\}$0.6

$\{2\}$1.056

1

0

0.6

0

0.6

0.4

0

0.4

0.6

0

0

1

1

$\emptyset$ 1

$\{1\}$0

$\{2\}$0.6

$\{3\}$0.6

$\{13\}$0

$\{23\}$0.36

0.6

0.6

0

0

0.36

1

0

1

0

1

1

0

0

0.36

0

1

t

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.

Transition probabilities are state-independent

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

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

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

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 $\bar{S}_i$ **relaxes** state S_i only if

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

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 $\bar{S}_i$ **relaxes** state S_i only if

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

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 DDs

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

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

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

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

Theorem. If **probabilities are state-independent**, then a merger operation that satisfies (C1) and (C2) alone yields a relaxed SDD.

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 DDs 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 DDs 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
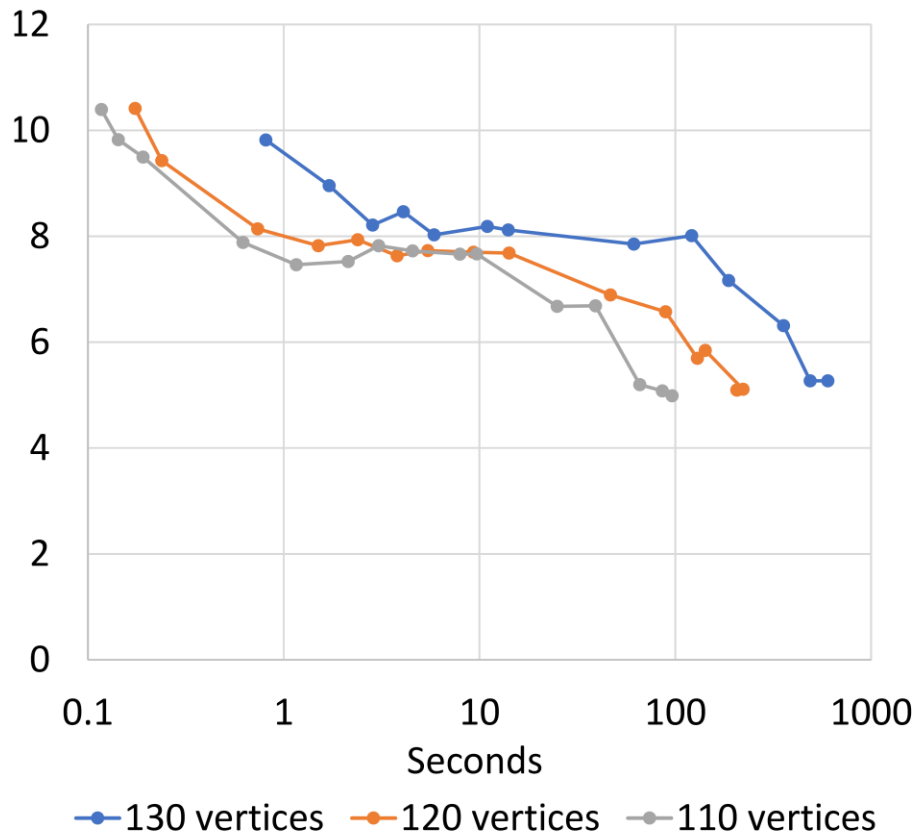## Solved to optimality
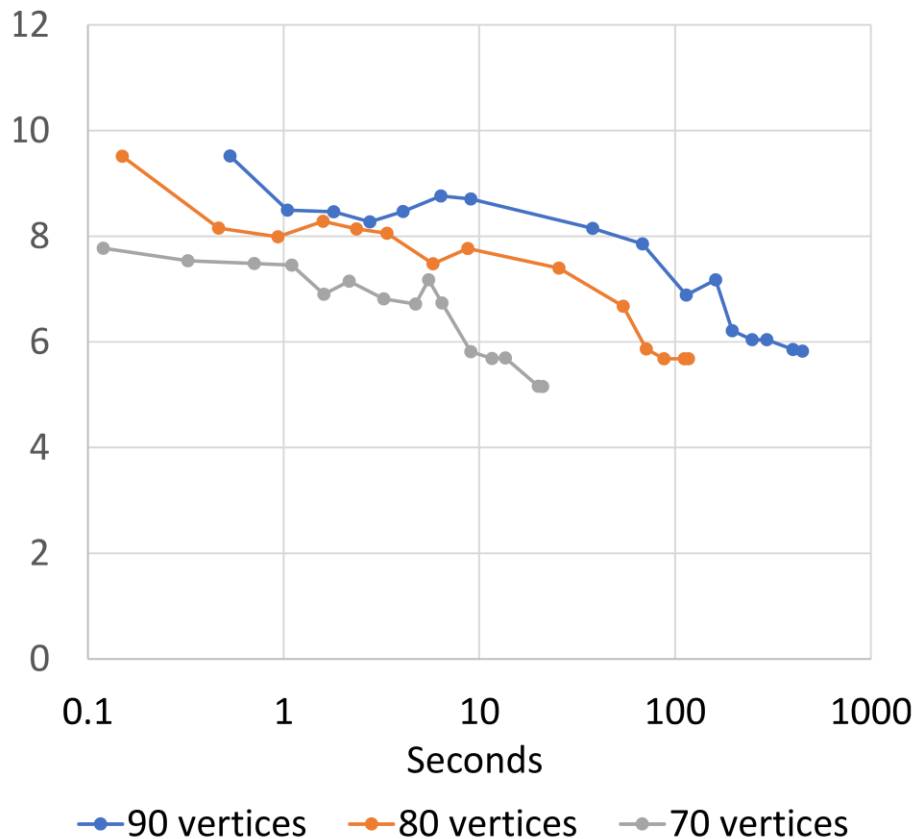## Bound vs time

Density 0.6

Density 0.7

70 vertices   65 vertices   60 vertices

50 vertices   48 vertices   45 vertices
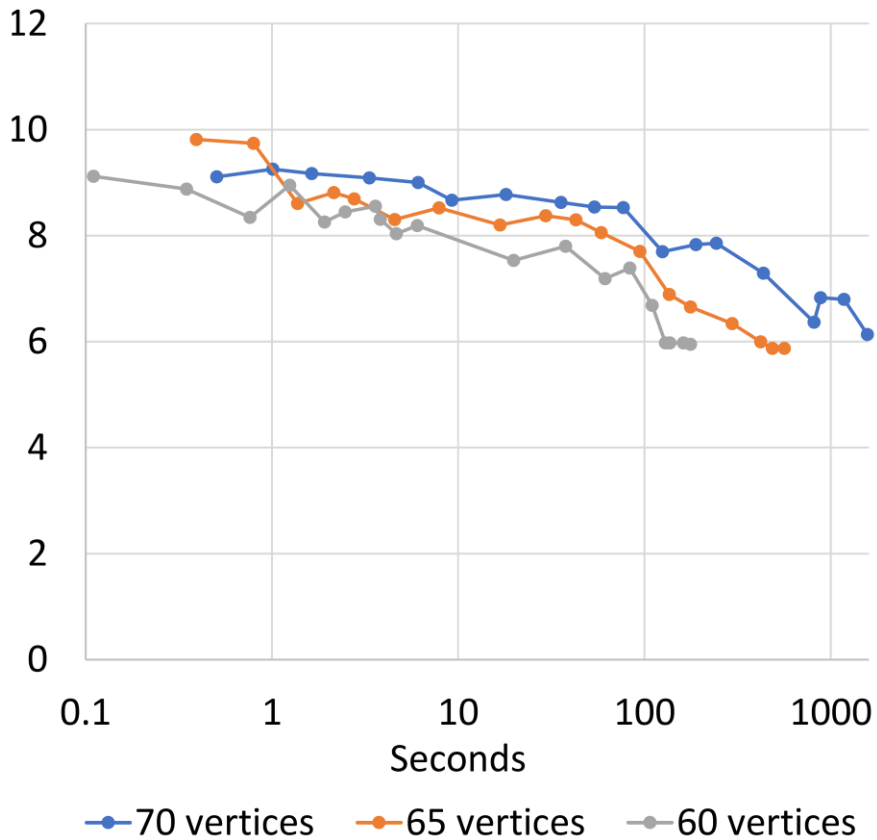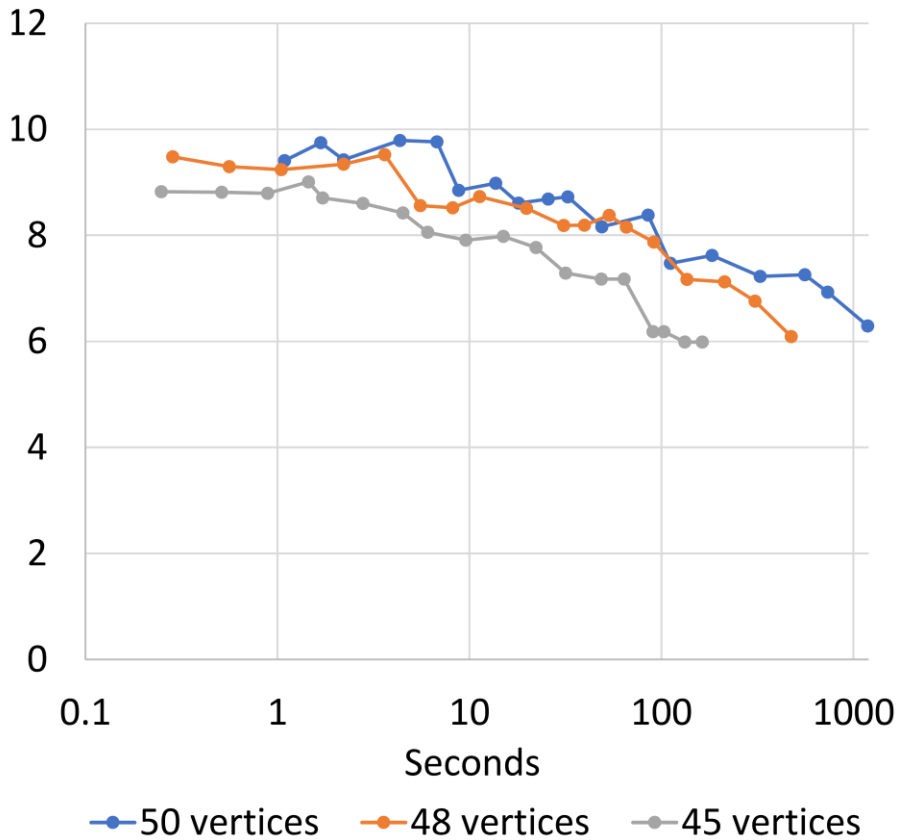
Random instances
Solved to optimality

Density 0.4

Density 0.5

130 vertices   120 vertices   110 vertices

90 vertices   80 vertices   70 vertices

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

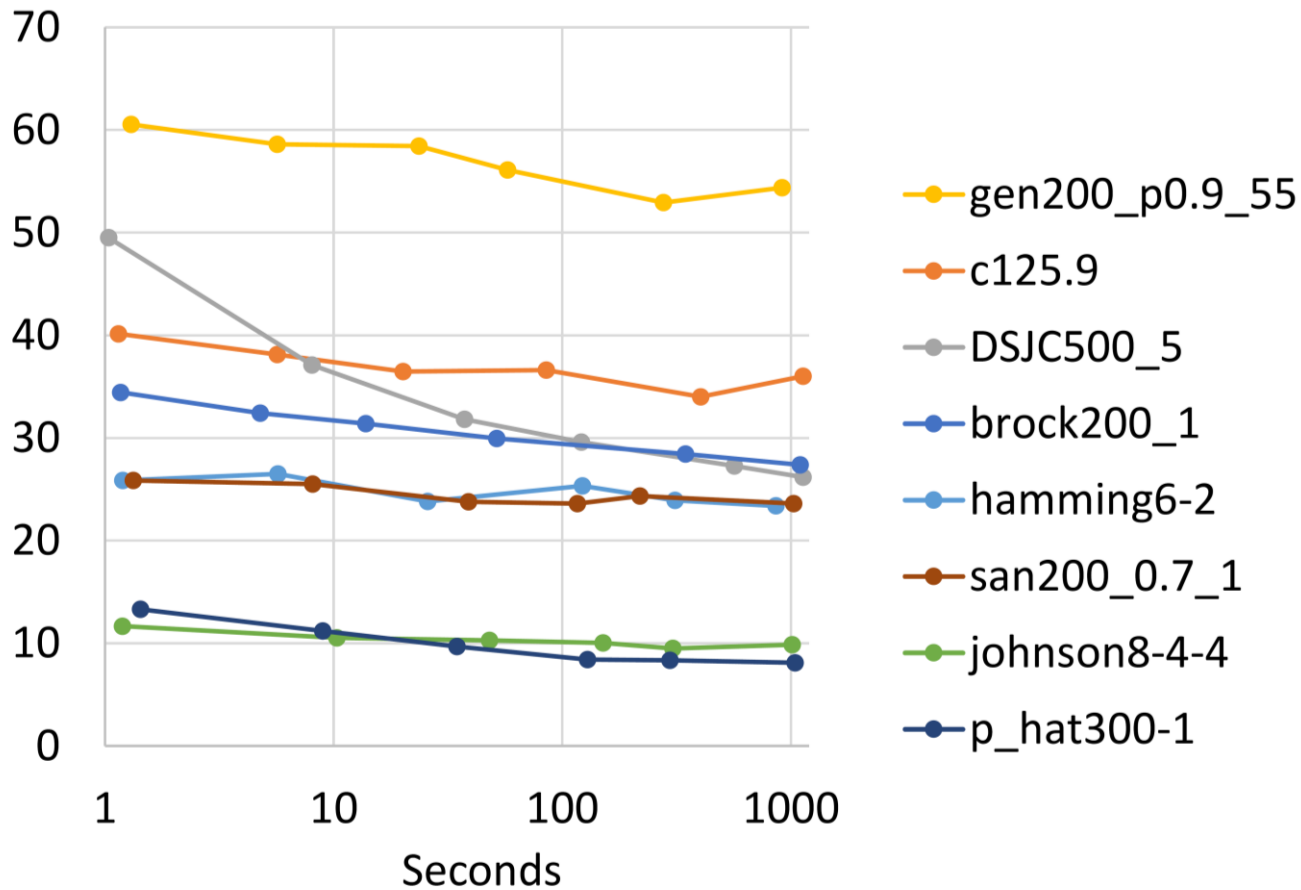# 2 DIMACS instances
Solved to optimality
Bound vs time
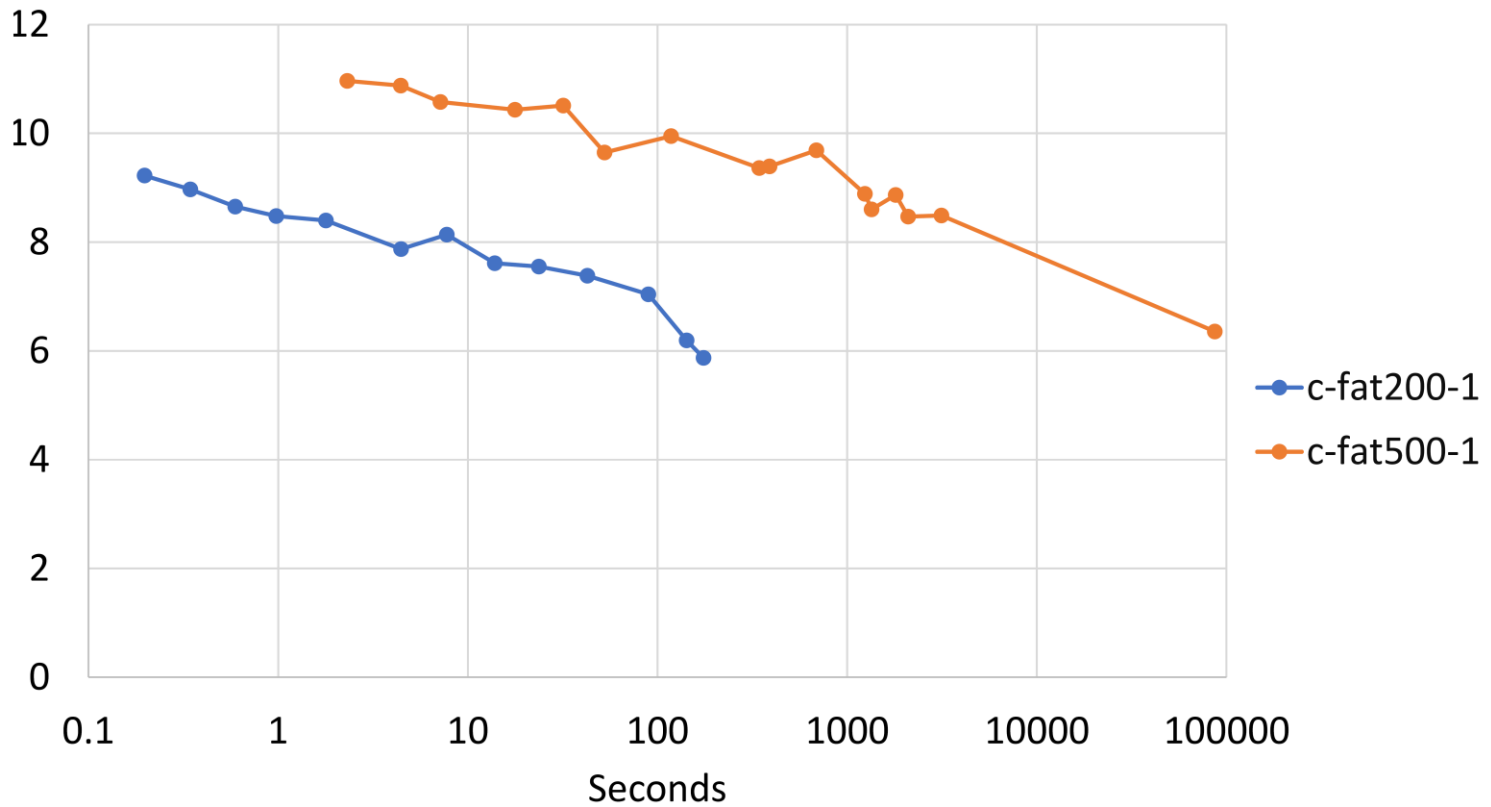
DIMACS instances

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.