Reinforcement Learning for Discrete Optimization

Elias B. Khalil
Department of Mechanical & Industrial Engineering
SCALE AI Research Chair in Data-Driven Algorithms for Modern Supply Chains
ekhalil.com

UNIVERSITY OF TORONTO

Humans learn to **design** **algorithms**.

Humans learn to **design** **algorithms**.

Can **algorithms** “learn” to **design** **algorithms**?

Humans learn to **design algorithms**.

Machine Learning

Can **algorithms** “learn” to **design algorithms**?

Humans learn to **design** **algorithms**.

Machine Learning

Can **algorithms** “learn” to **design** **algorithms**?

Discrete Optimization

Data Center Resource Management

 Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits

Data Center Resource Management

Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits

Data Center Resource Management

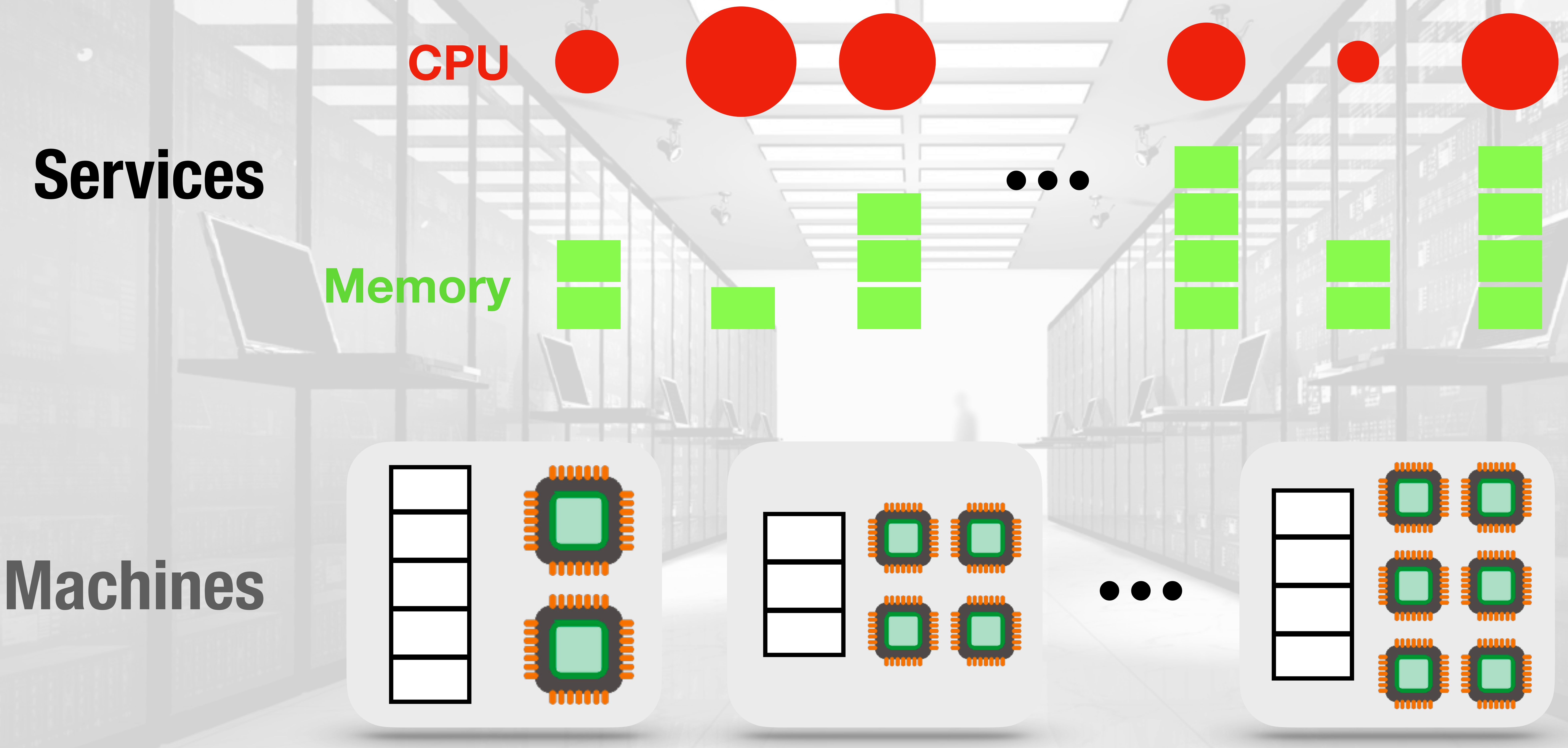


 Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits

Data Center Resource Management

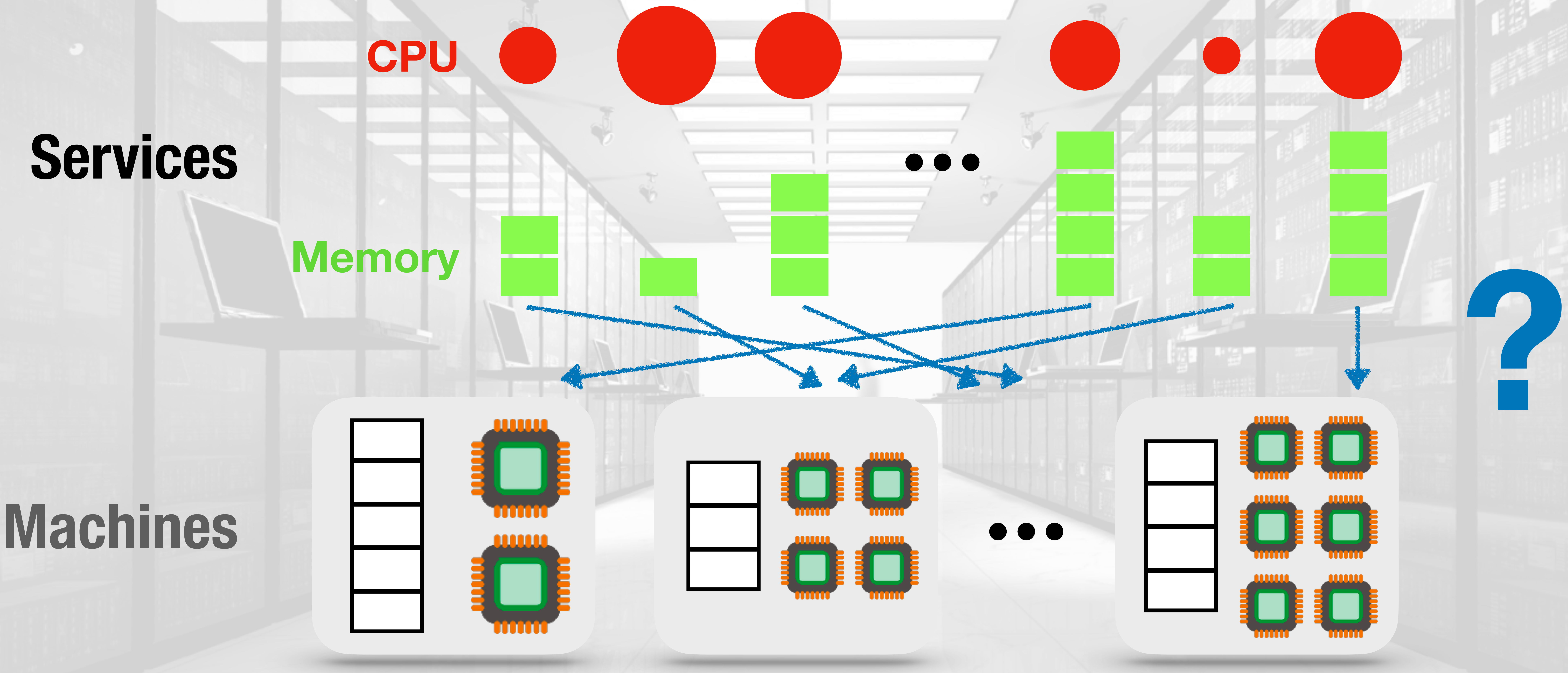
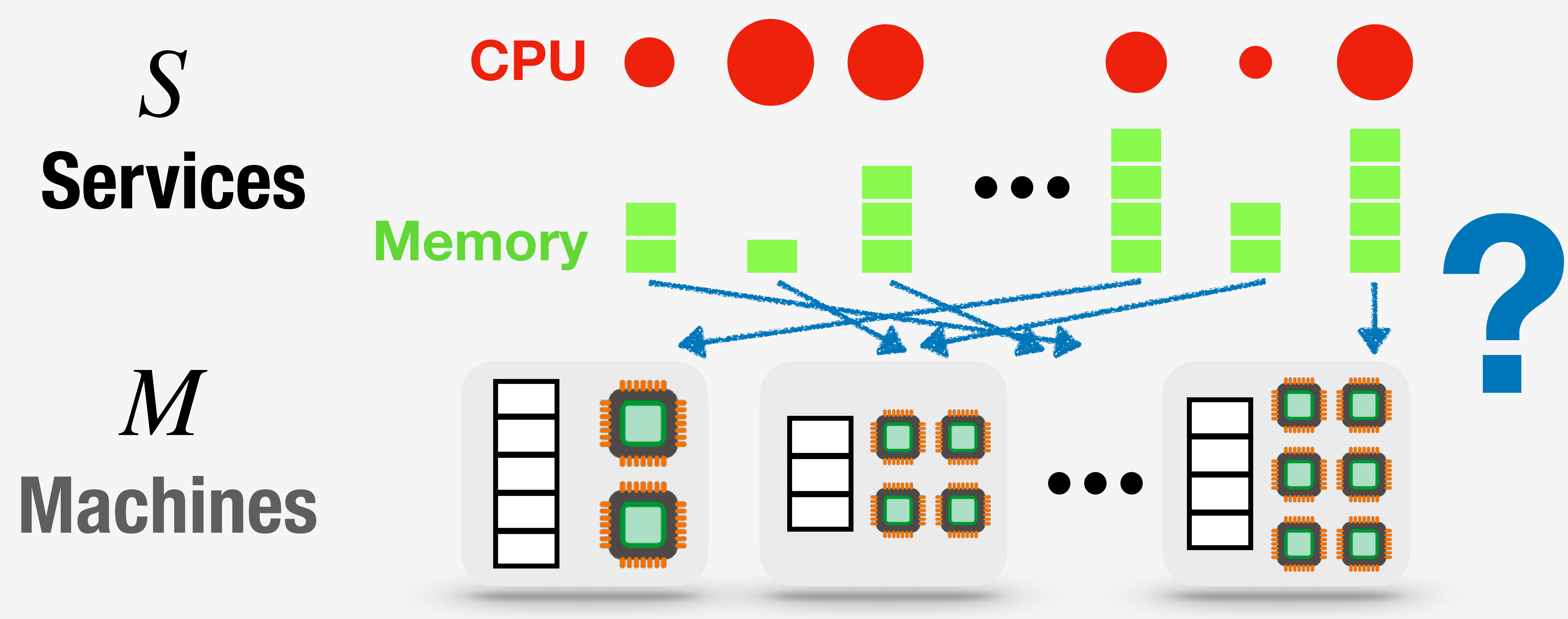
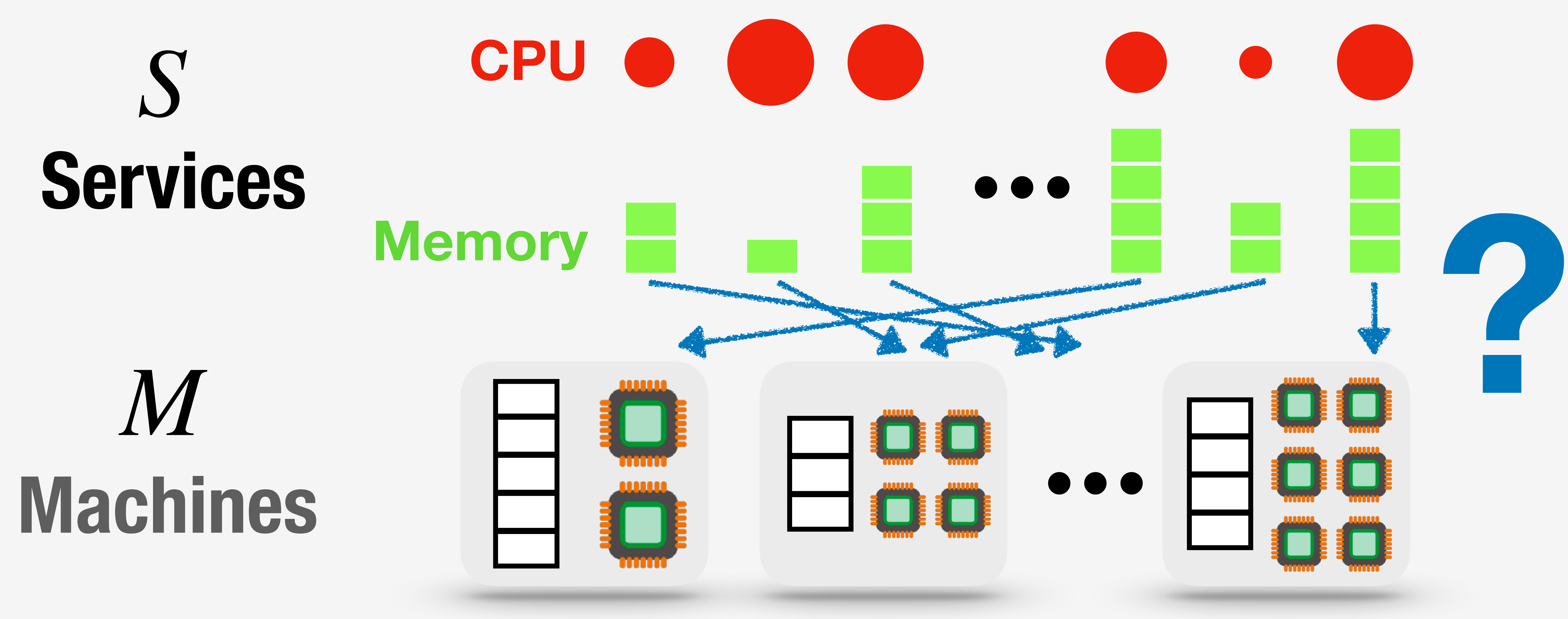


Photo from: https://www.reit.com/what-reit/reit-sectors/data-center-reits


S
Services
CPU
Memory
M
Machines
?

$y_m = 1$ if machine m is used

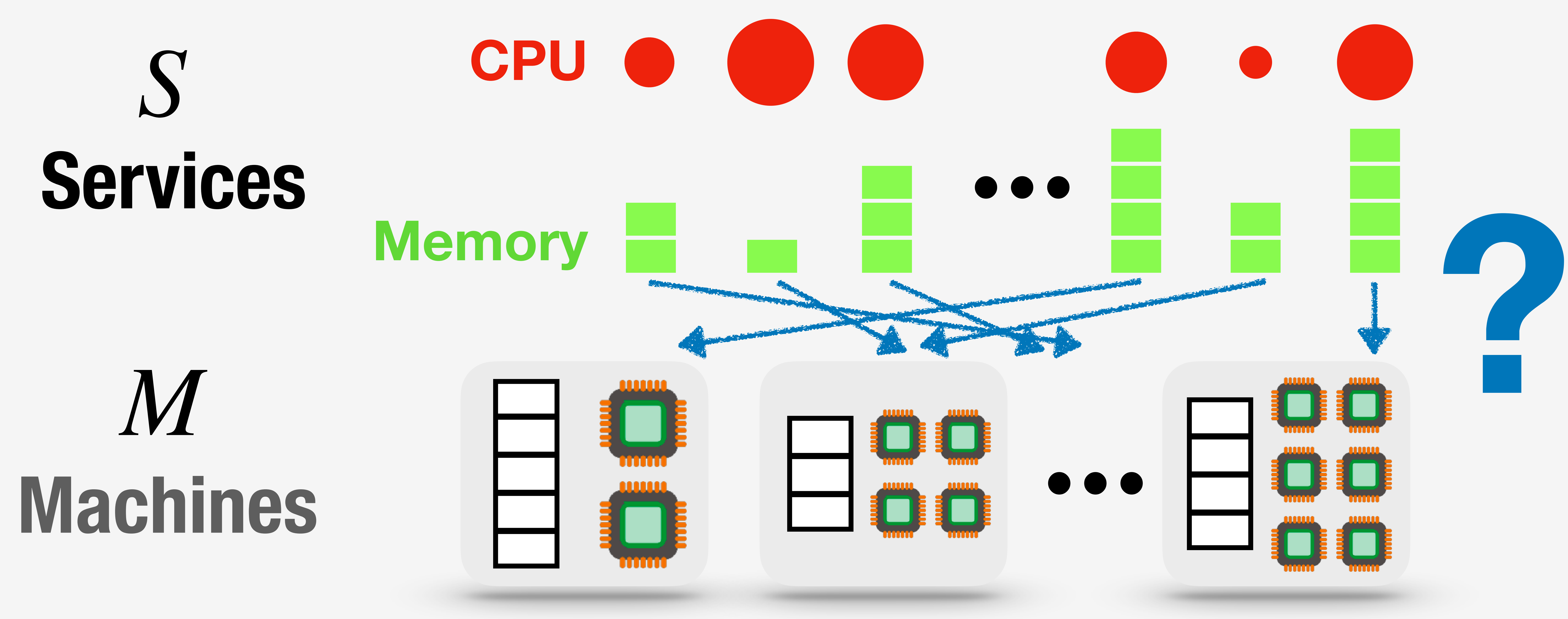
$x_{s,m} = 1$ if service s runs on m



$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M}, y \in \{0,1\}^M$

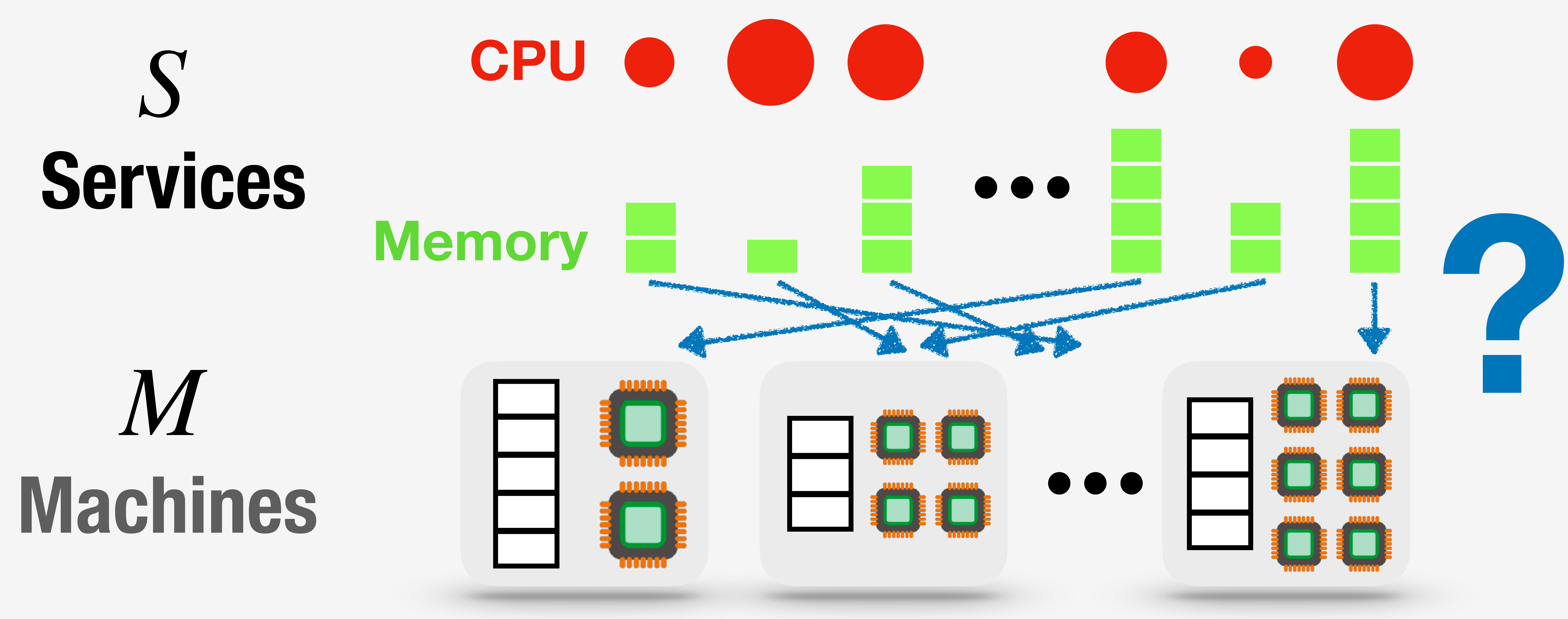


$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M}$, $y \in \{0,1\}^M$

minimize $\sum_{m=1}^{M} y_m$



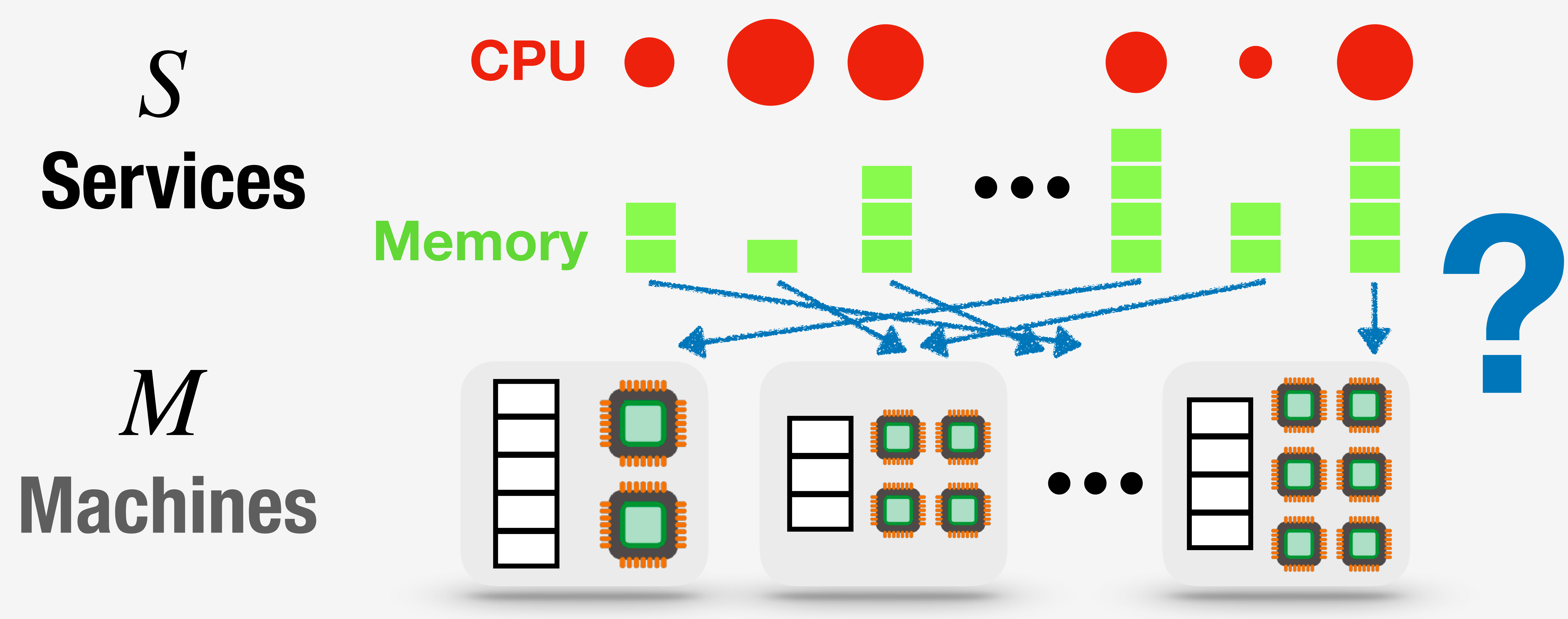
$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M}, y \in \{0,1\}^M$

minimize $\sum_{m=1}^{M} y_m$

Constraints:

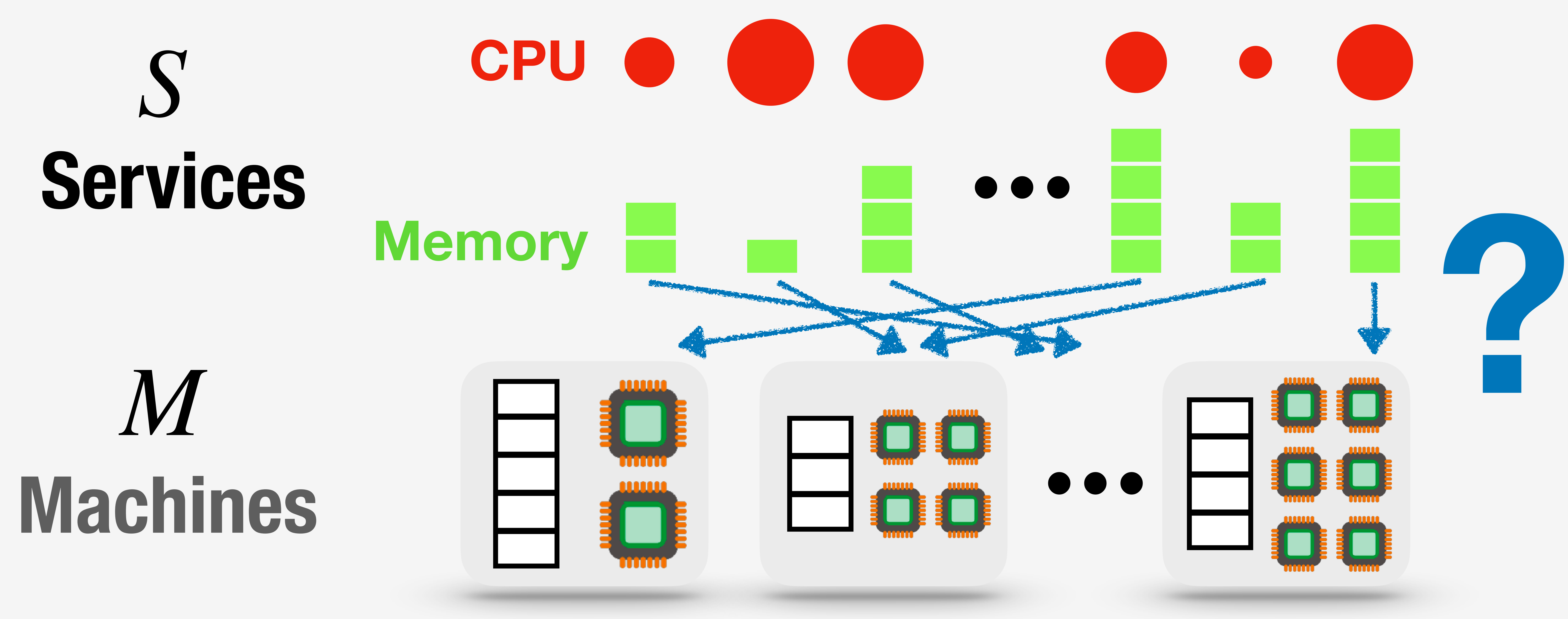


$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M}, \; y \in \{0,1\}^M$

minimize $\sum_{m=1}^{M} y_m$



Constraints:

Each service on one machine only

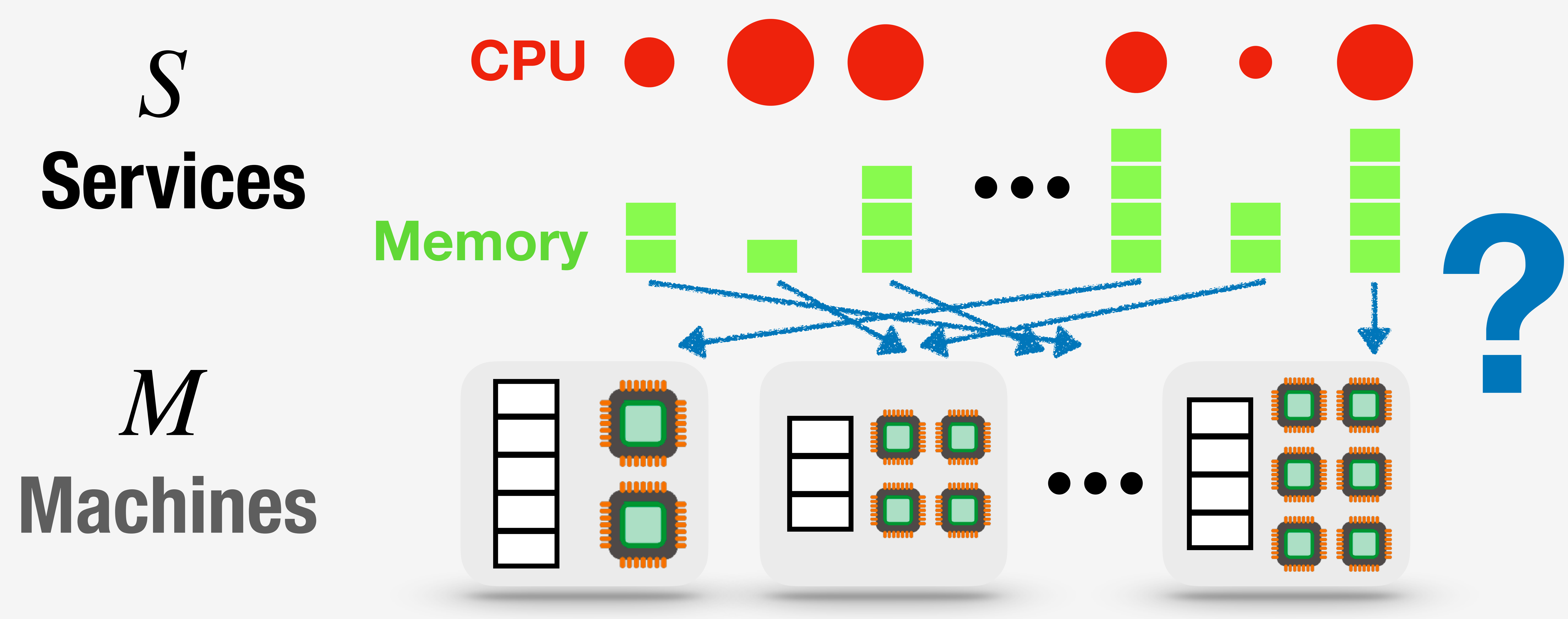
$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$$x \in \{0,1\}^{S \times M},\ y \in \{0,1\}^M$$

minimize $\sum_{m=1}^{M} y_m$



Constraints:

Each service on one machine only

$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

$$y_m \geq x_{s,m} \quad \forall s, m$$

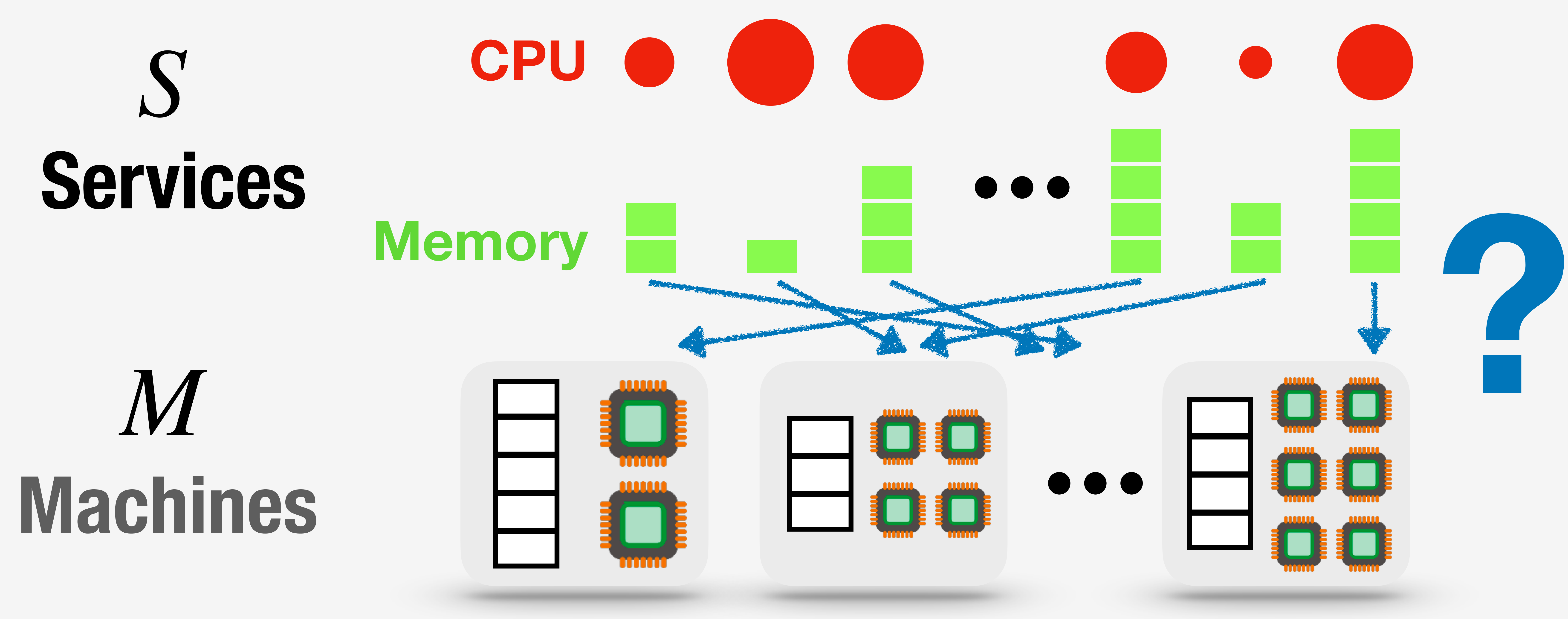
Machine is "ON" if a job is assigned to it

$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$x \in \{0,1\}^{S \times M}, y \in \{0,1\}^M$

minimize $\sum_{m=1}^{M} y_m$



Constraints:

Each service on one machine only

$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

Memory capacity

$$\sum_{s=1}^{S} \textbf{mem}(s) \cdot x_{s,m} \leq \textbf{cap-mem}(m) \quad \forall m$$

$$y_m \geq x_{s,m} \quad \forall s, m$$

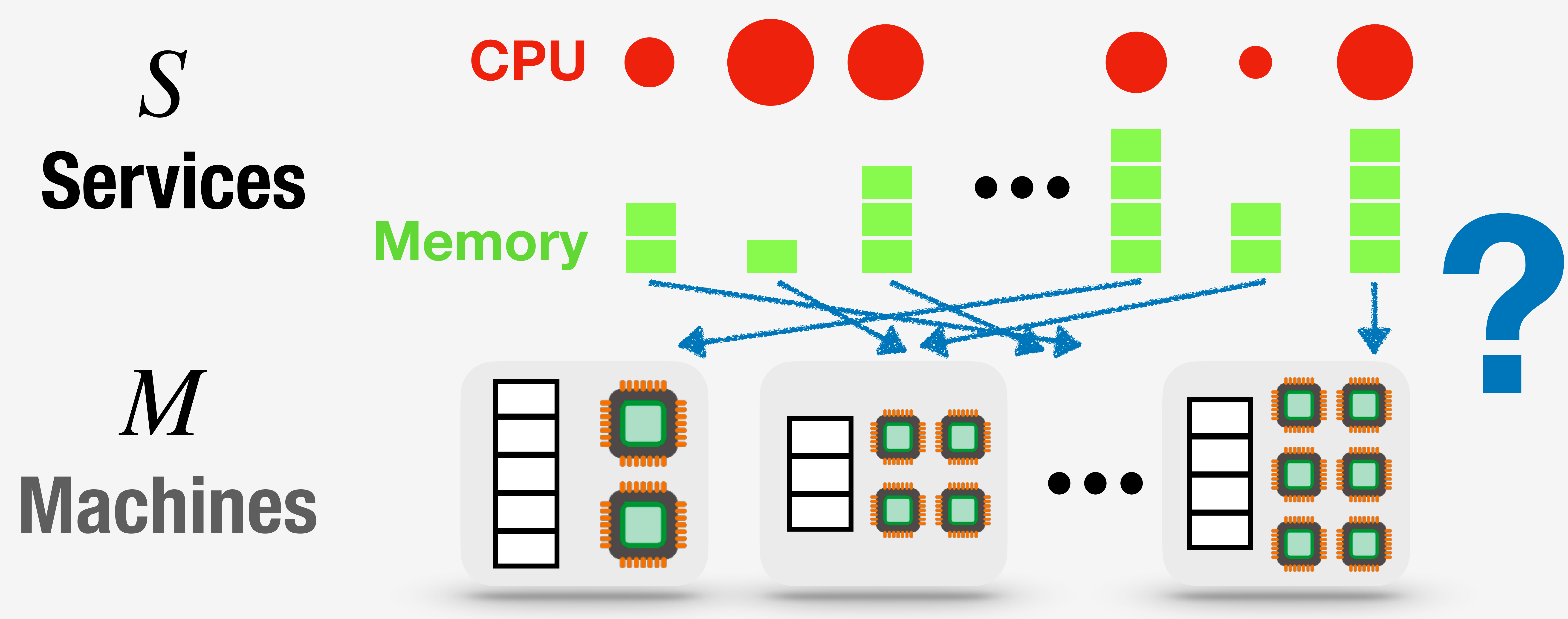
Machine is "ON" if a job is assigned to it

$y_m = 1$ if machine m is used

$x_{s,m} = 1$ if service s runs on m

$$x \in \{0,1\}^{S \times M},\ y \in \{0,1\}^M$$

$$\textbf{minimize} \sum_{m=1}^{M} y_m$$



Constraints:

Each service on one machine only

$$\sum_{m=1}^{M} x_{s,m} = 1 \quad \forall s$$

$$y_m \geq x_{s,m} \quad \forall s, m$$

Machine is "ON" if a job is assigned to it

Memory capacity

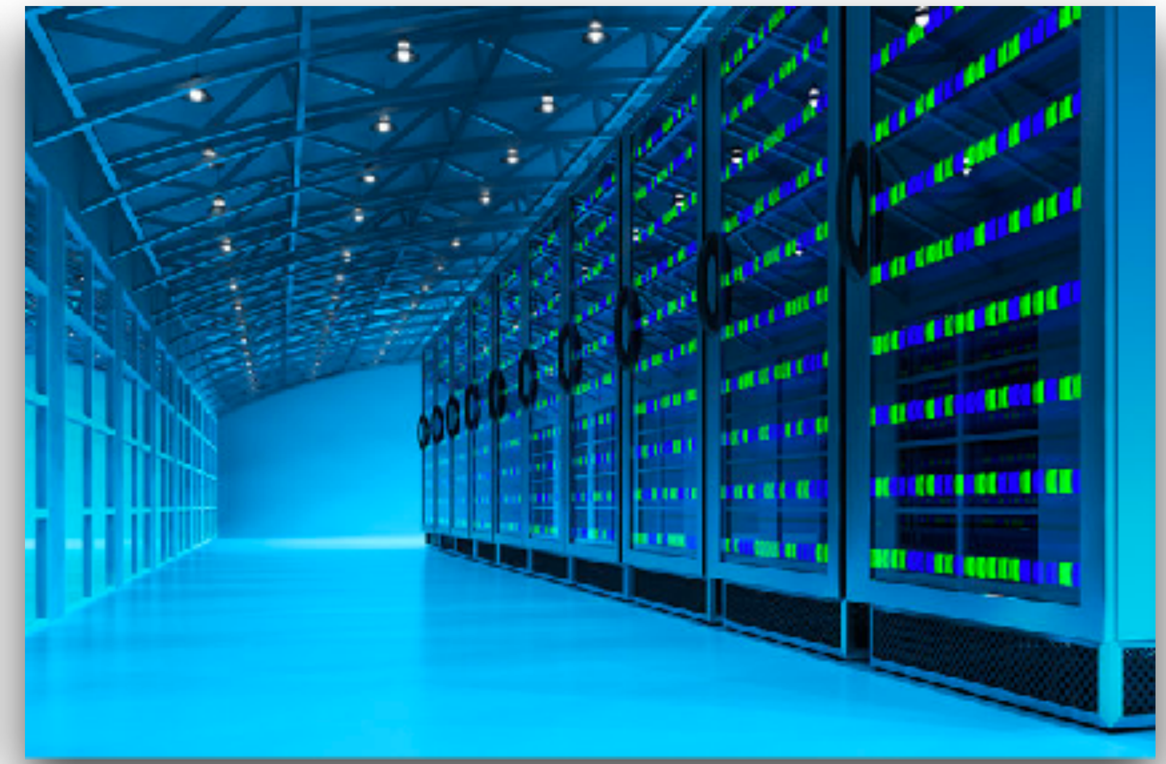
$$\sum_{s=1}^{S} \textbf{mem}(s) \cdot x_{s,m} \leq \textbf{cap-mem}(m) \quad \forall m$$

$$\sum_{s=1}^{S} \textbf{cpu}(s) \cdot x_{s,m} \leq \textbf{cap-cpu}(m) \quad \forall m$$

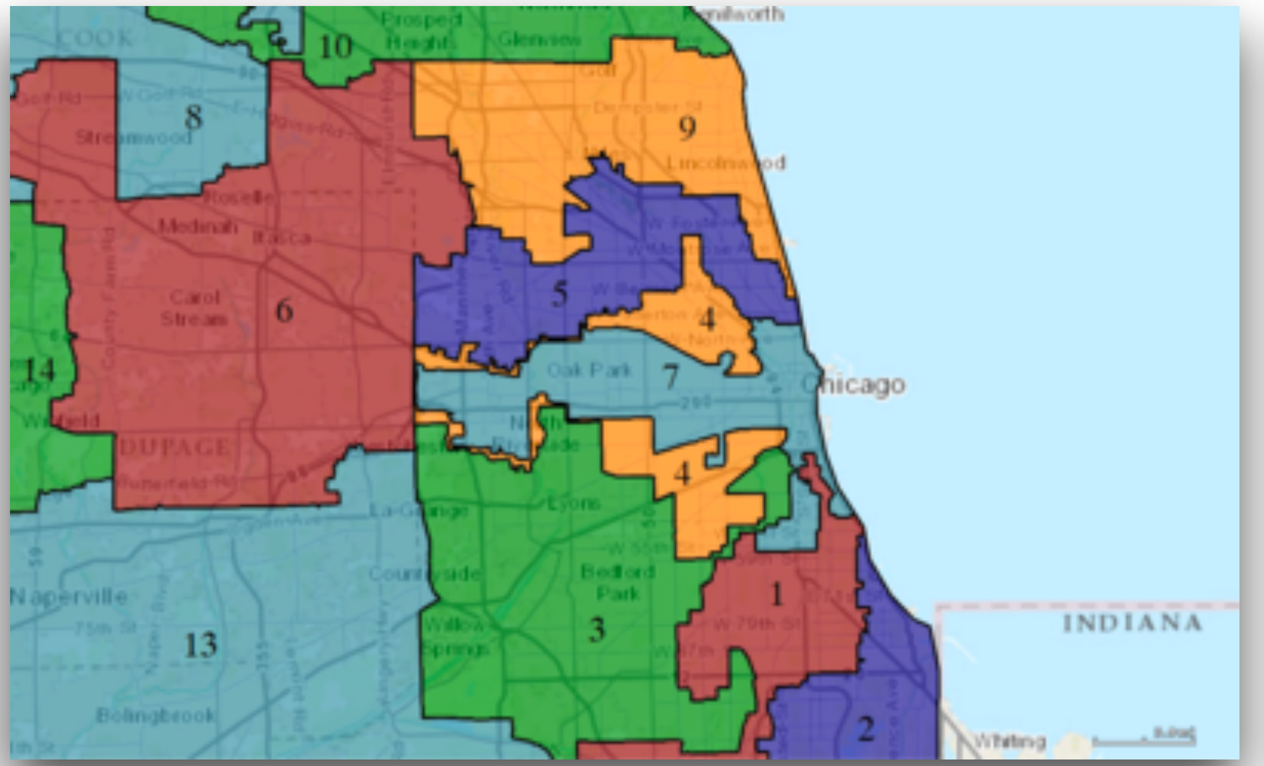
Processor capacity

Auction Design

Data Center Management



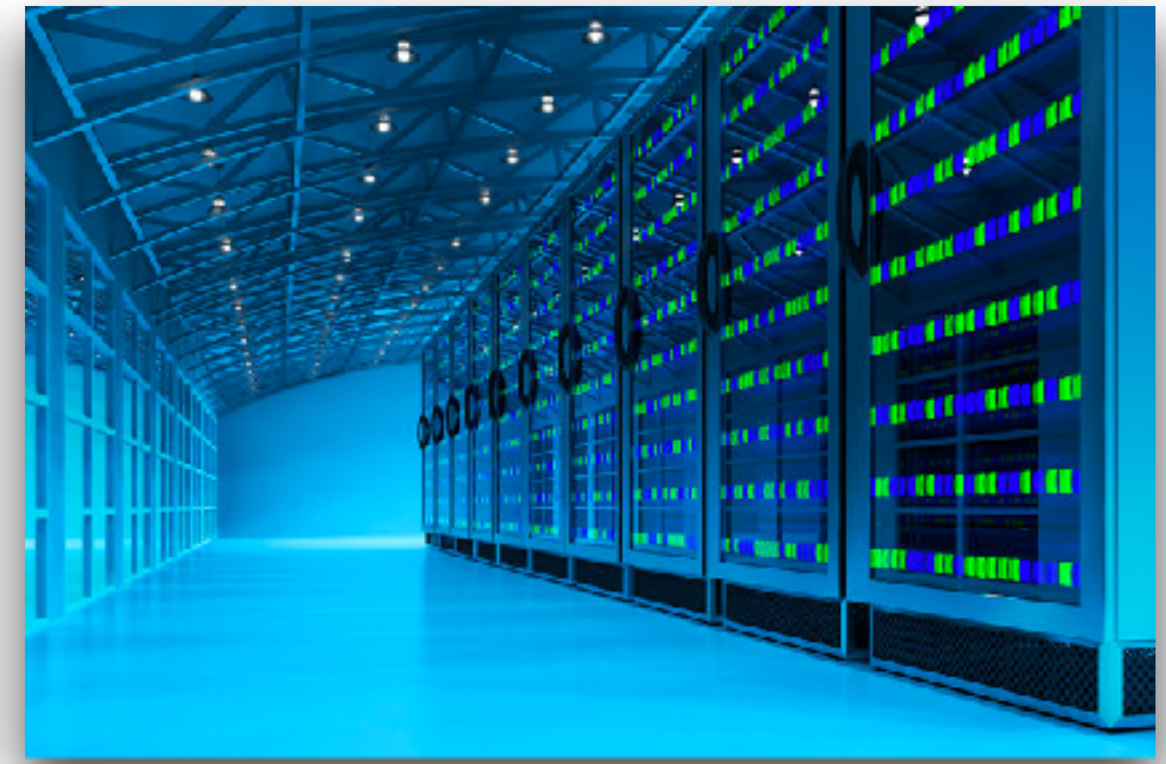
Political Districting



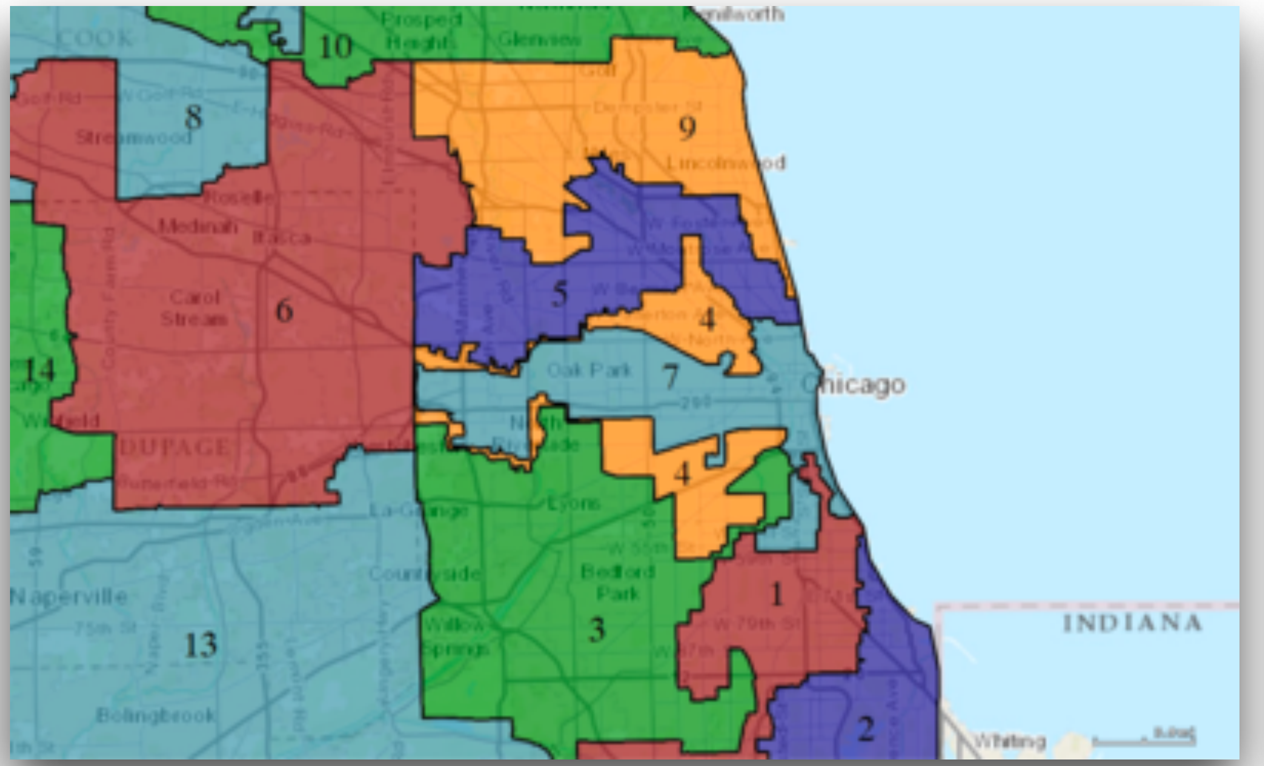
Kidney Exchange

Auction Design

Data Center Management

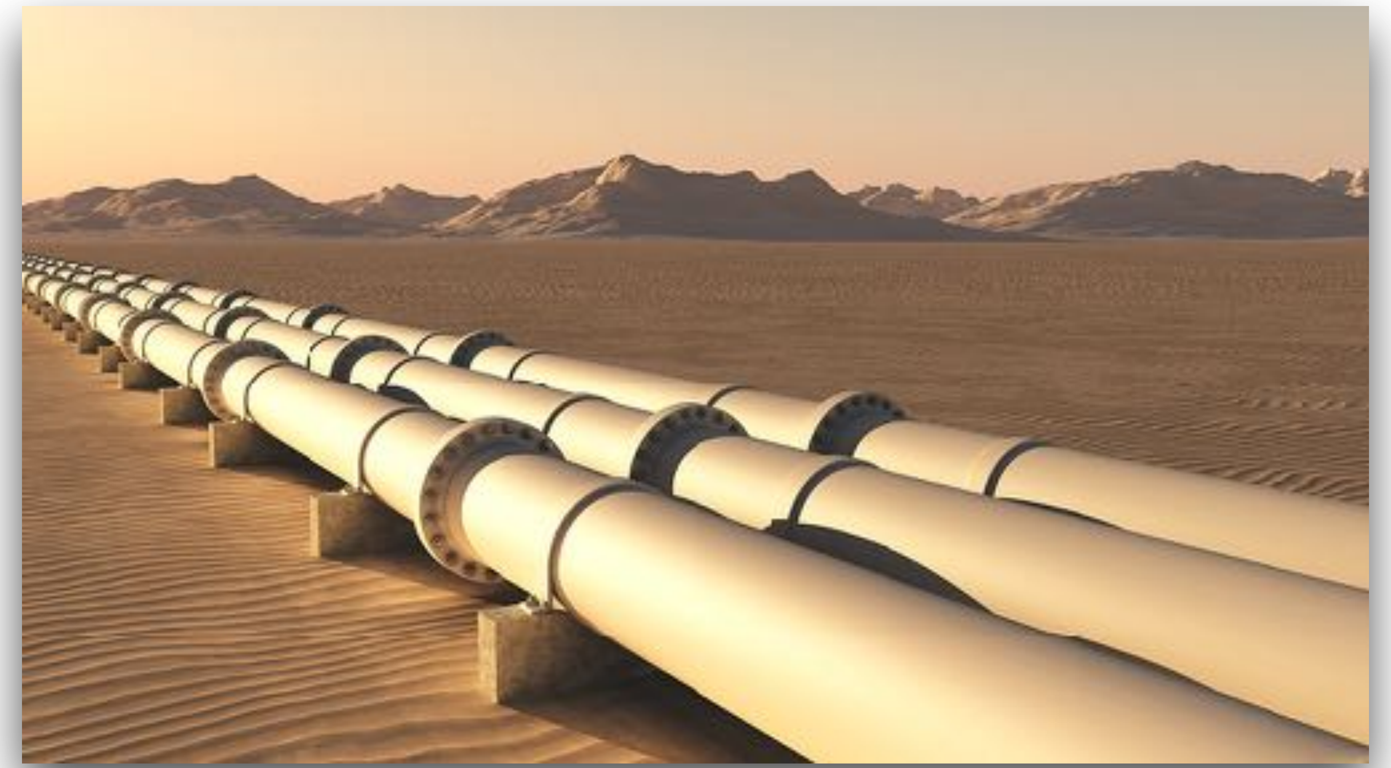


Political Districting

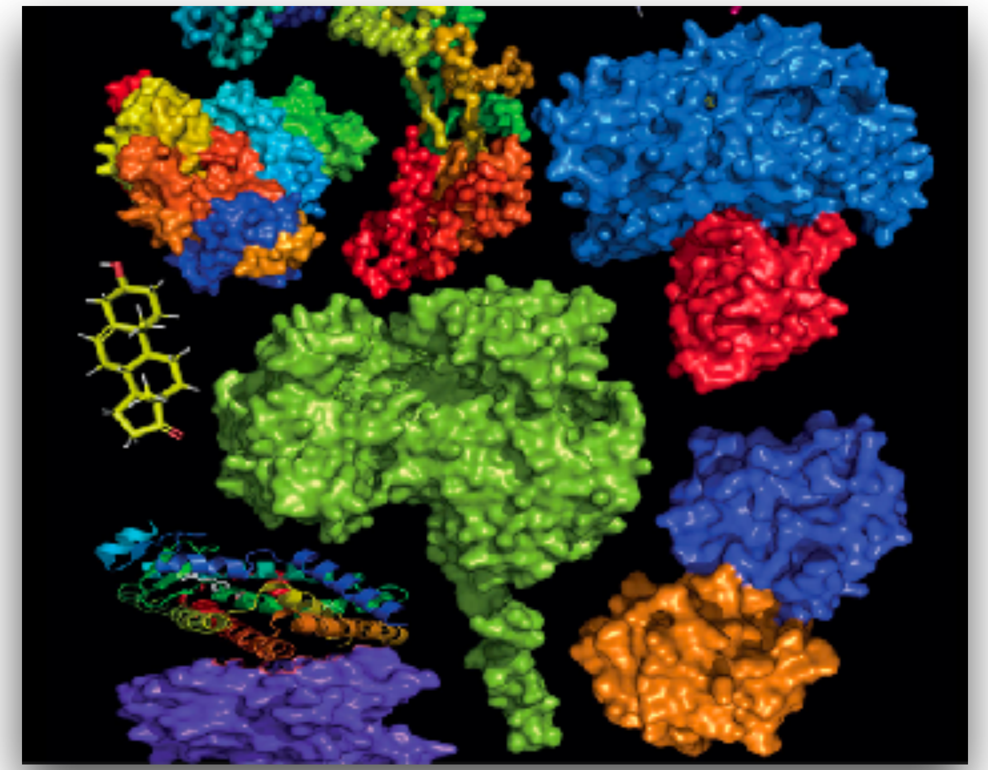


Kidney Exchange

Energy Systems



Scientific Discovery

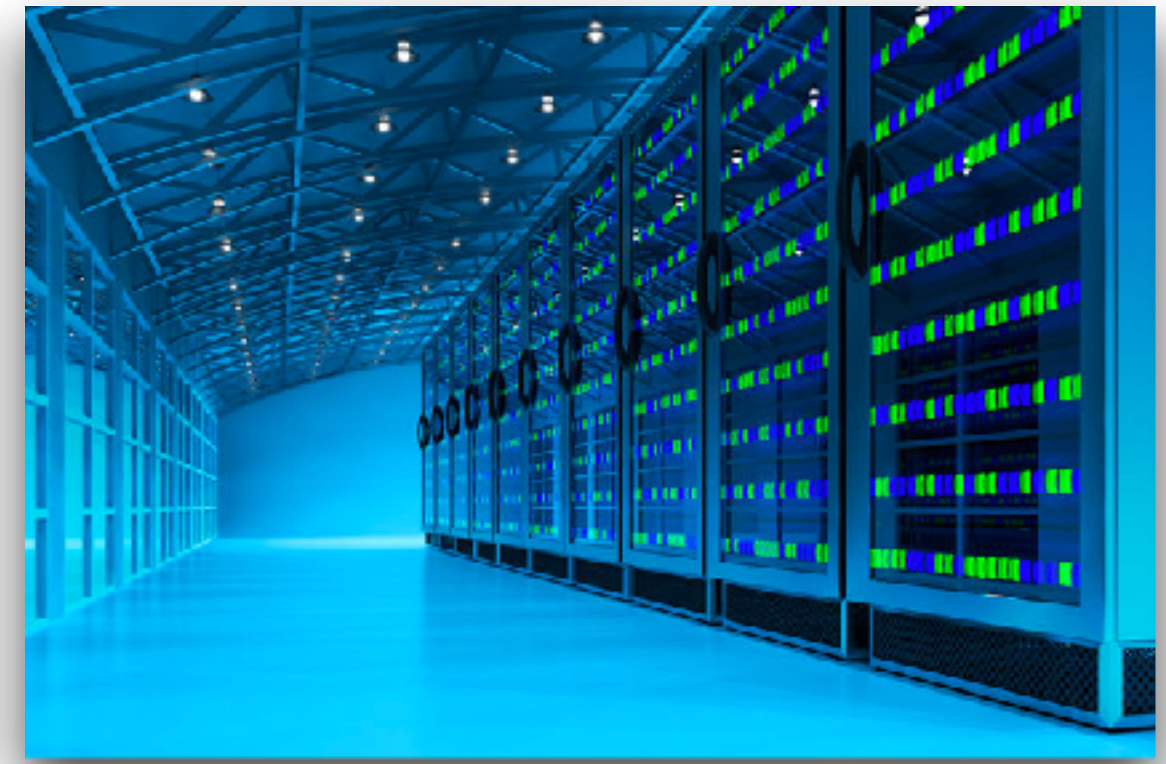


Ridesharing

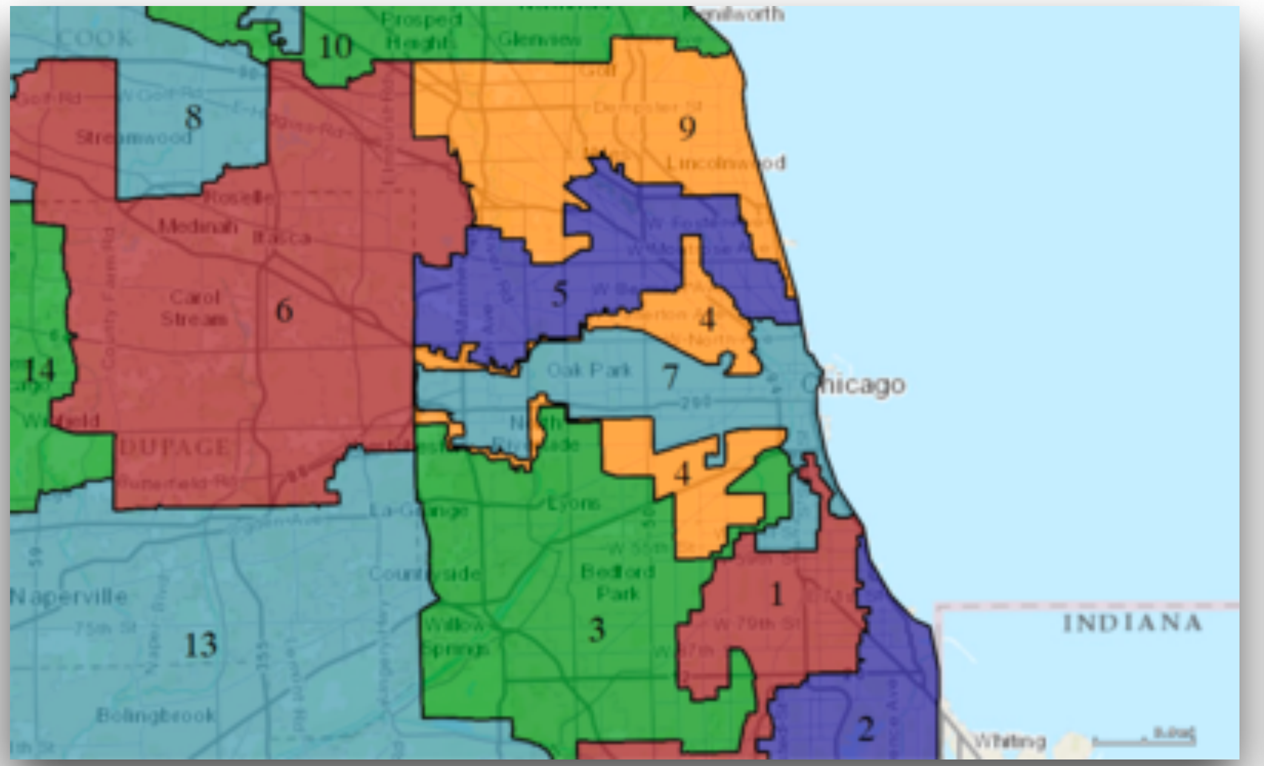
Cancer Therapeutics

Auction Design

Data Center Management

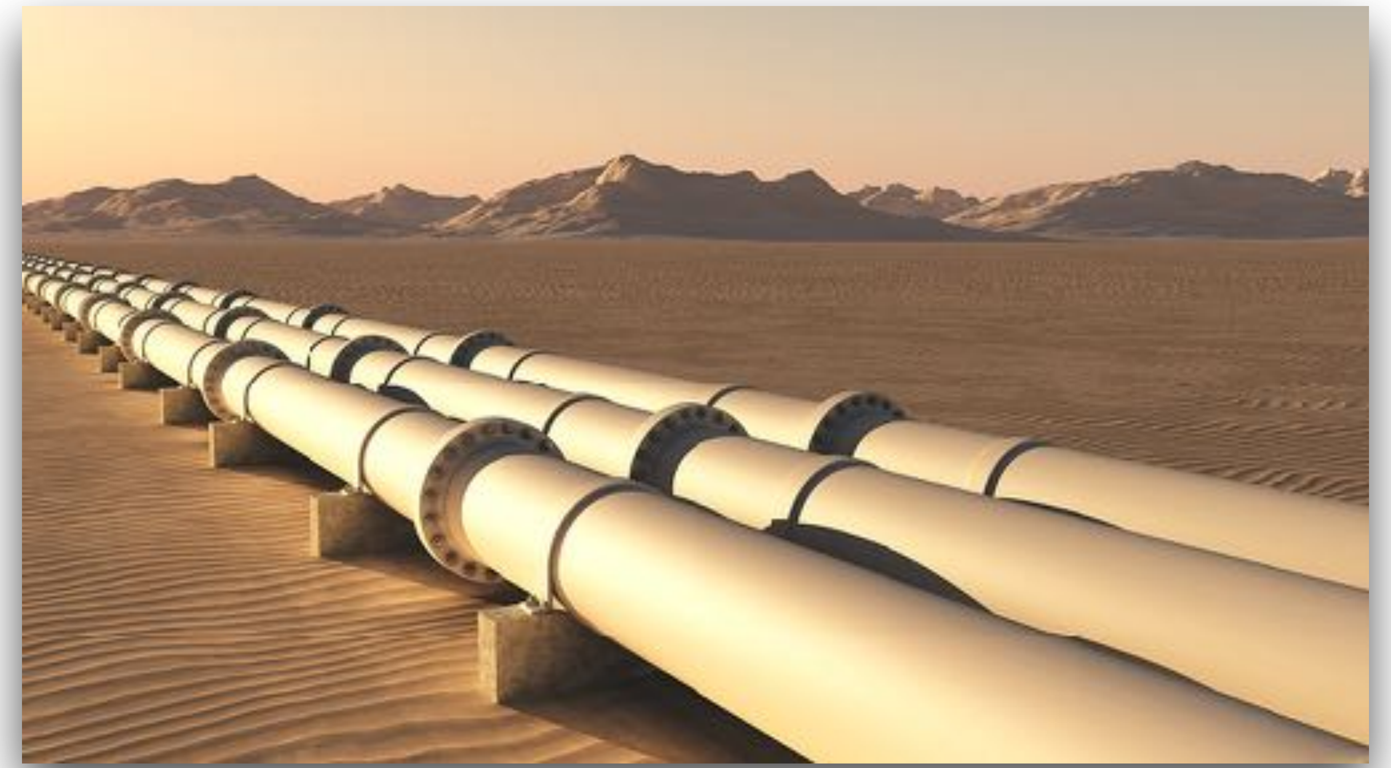


Political Districting

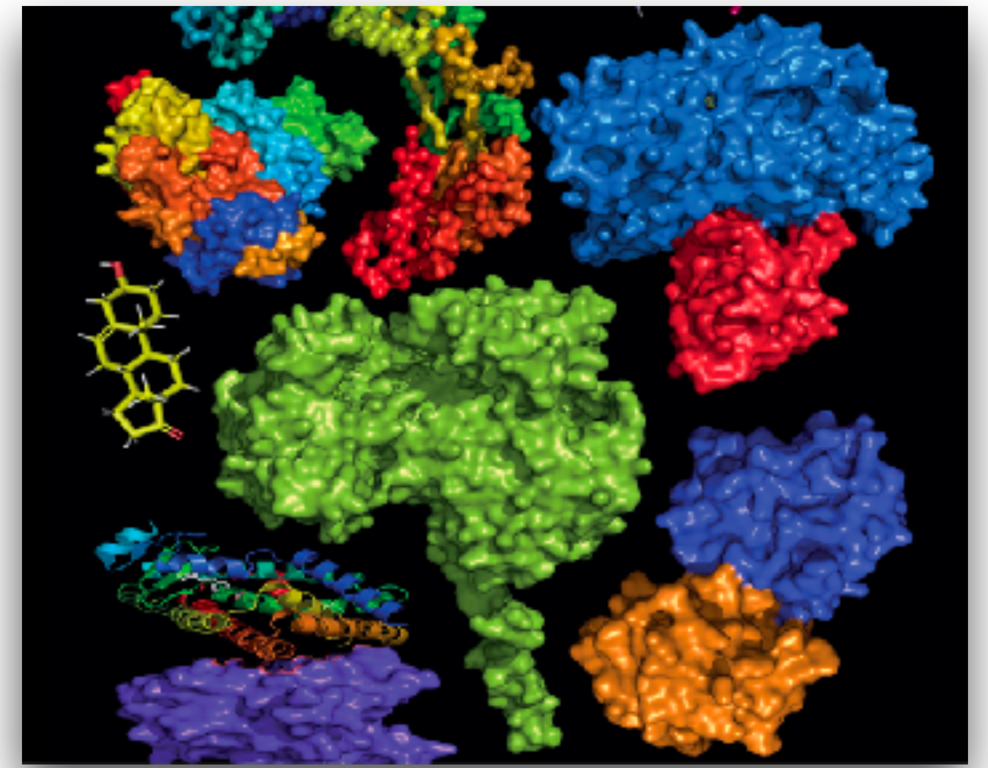


Kidney Exchange

Energy Systems



Scientific Discovery



Ridesharing

Cancer Therapeutics

Airline Scheduling

Conservation Planning

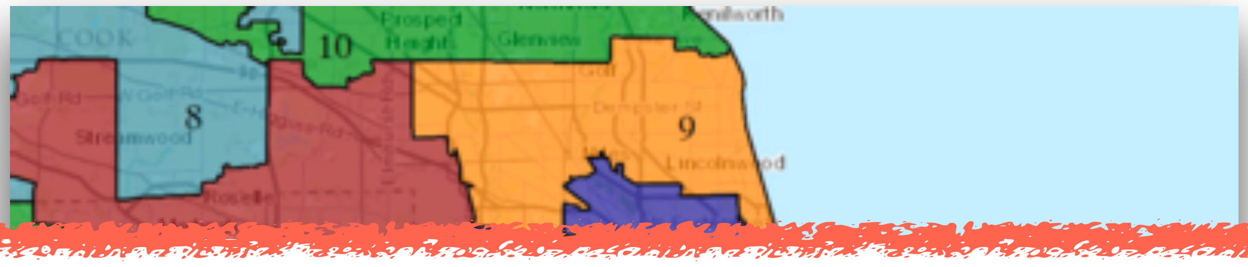
Disaster Response

College Admissions

Auction Design

Data Center Management

Political Districting



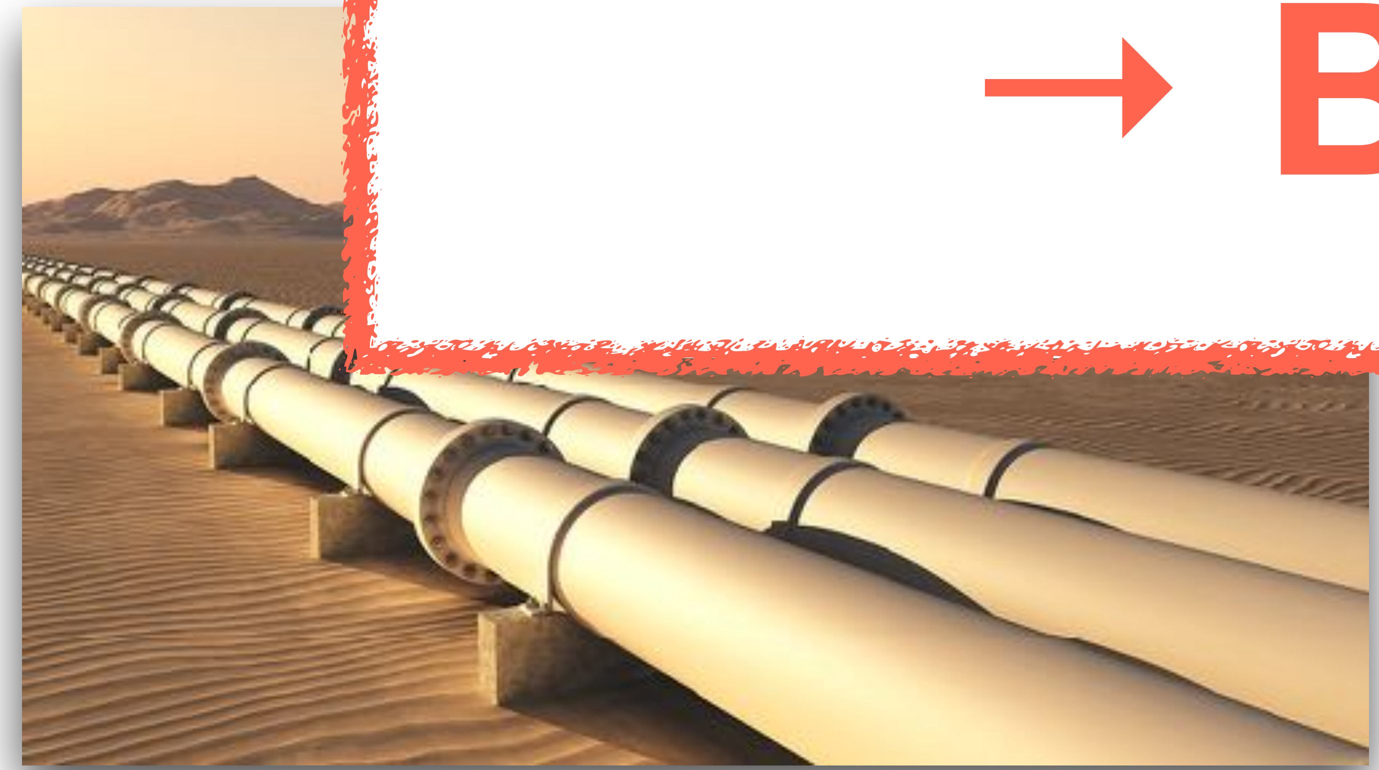
Kidney Exchange

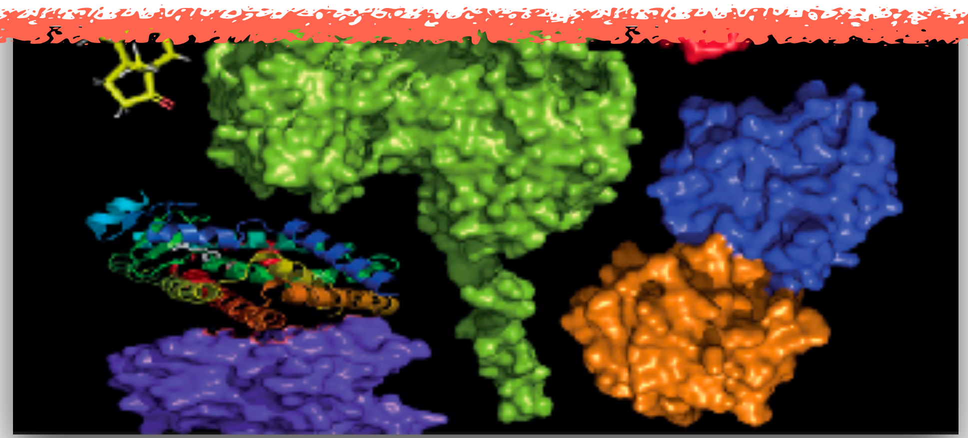
> 50% of INFORMS Edelman Award winners use Discrete Optimization

→ **Billions ($) in savings/profit**

George Nemhauser, Plenary at EURO INFORMS, 2013

utics





Airline Scheduling

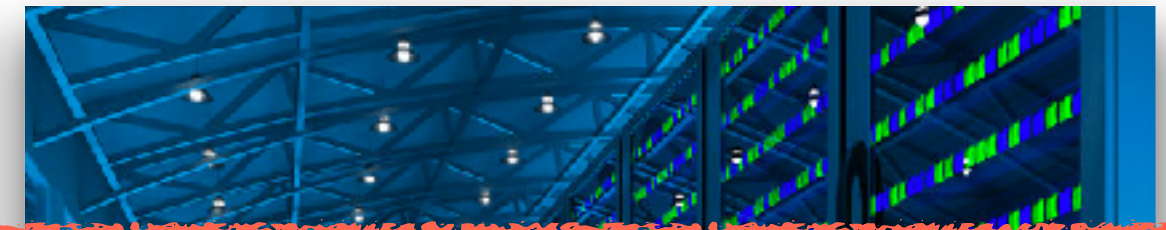
Conservation Planning

Disaster Response

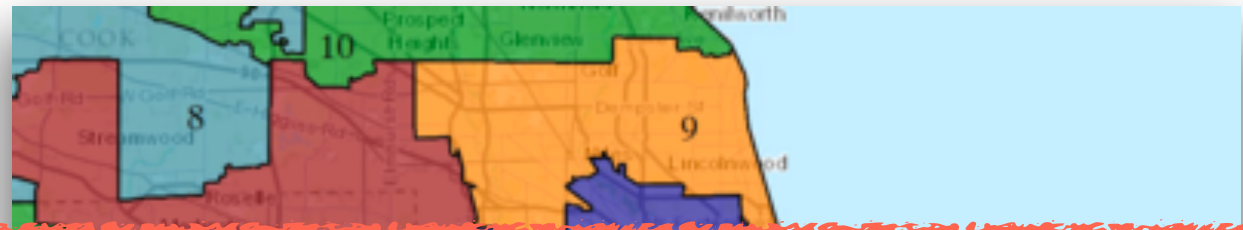
College Admissions

Auction Design

Data Center Management



Political Districting



Kidney Exchange

Airline Sc

utics

> 50% of INFORMS Edelman Award winners use Discrete Optimization

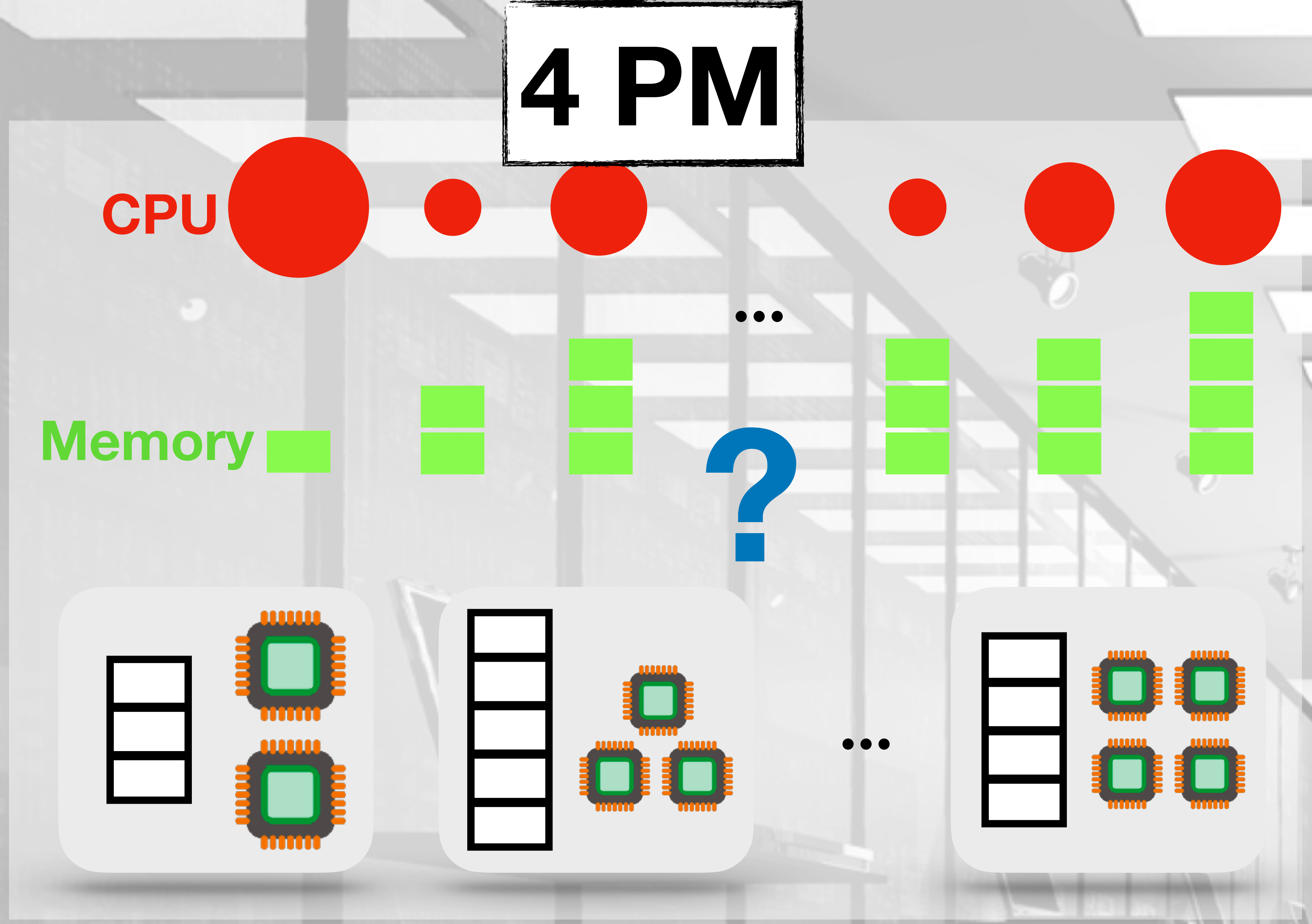
> Billions ($) in savings/profit

“[...] the overall value of linear optimization to the economy probably surpasses 5% overall or more than $1 trillion each year in the United States alone.”

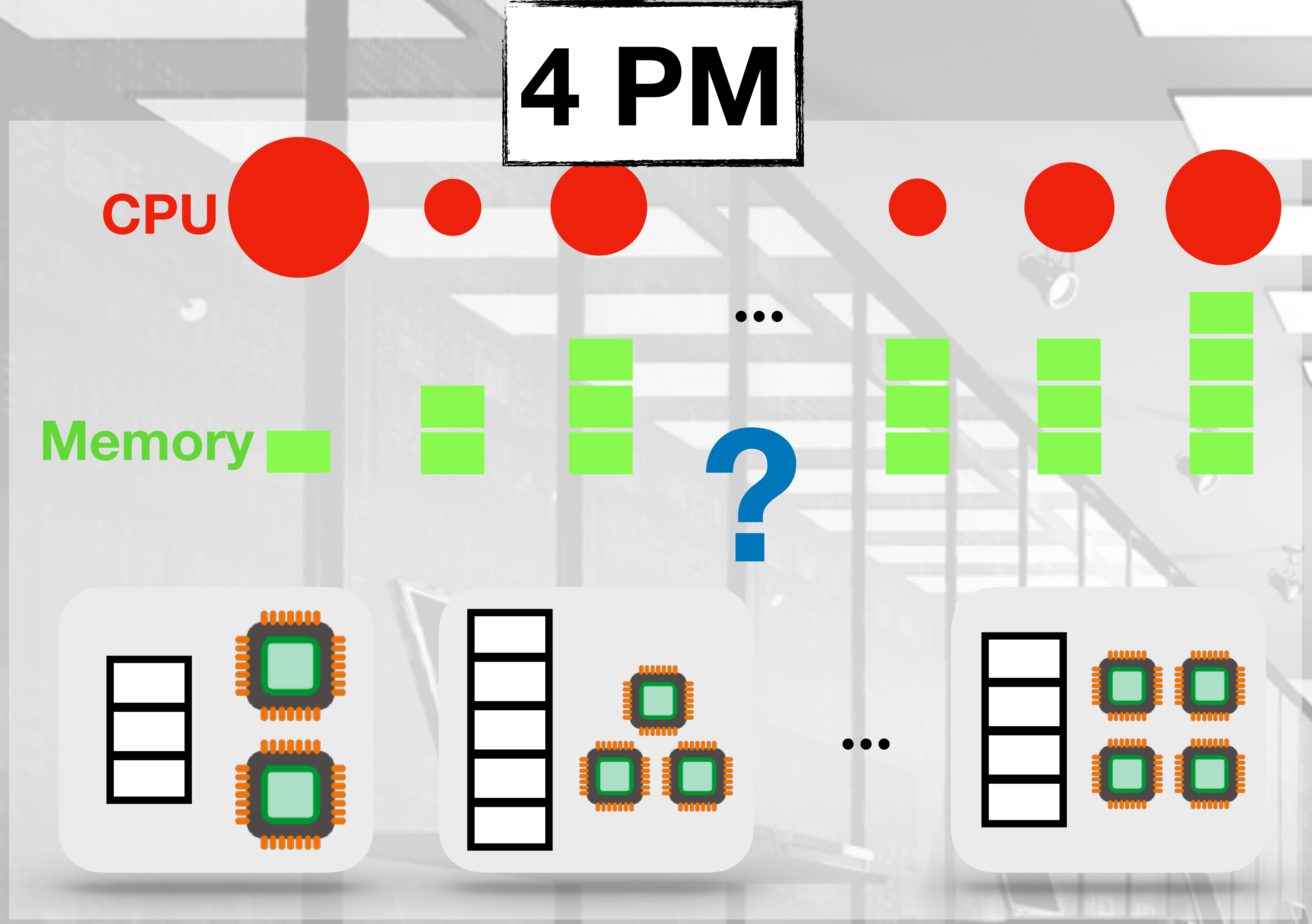
Birge, J. R. (2022). George Bernard Dantzig. Production and Operations Management, 31, 1909– 1911. https://doi.org/10.1111/poms.13751

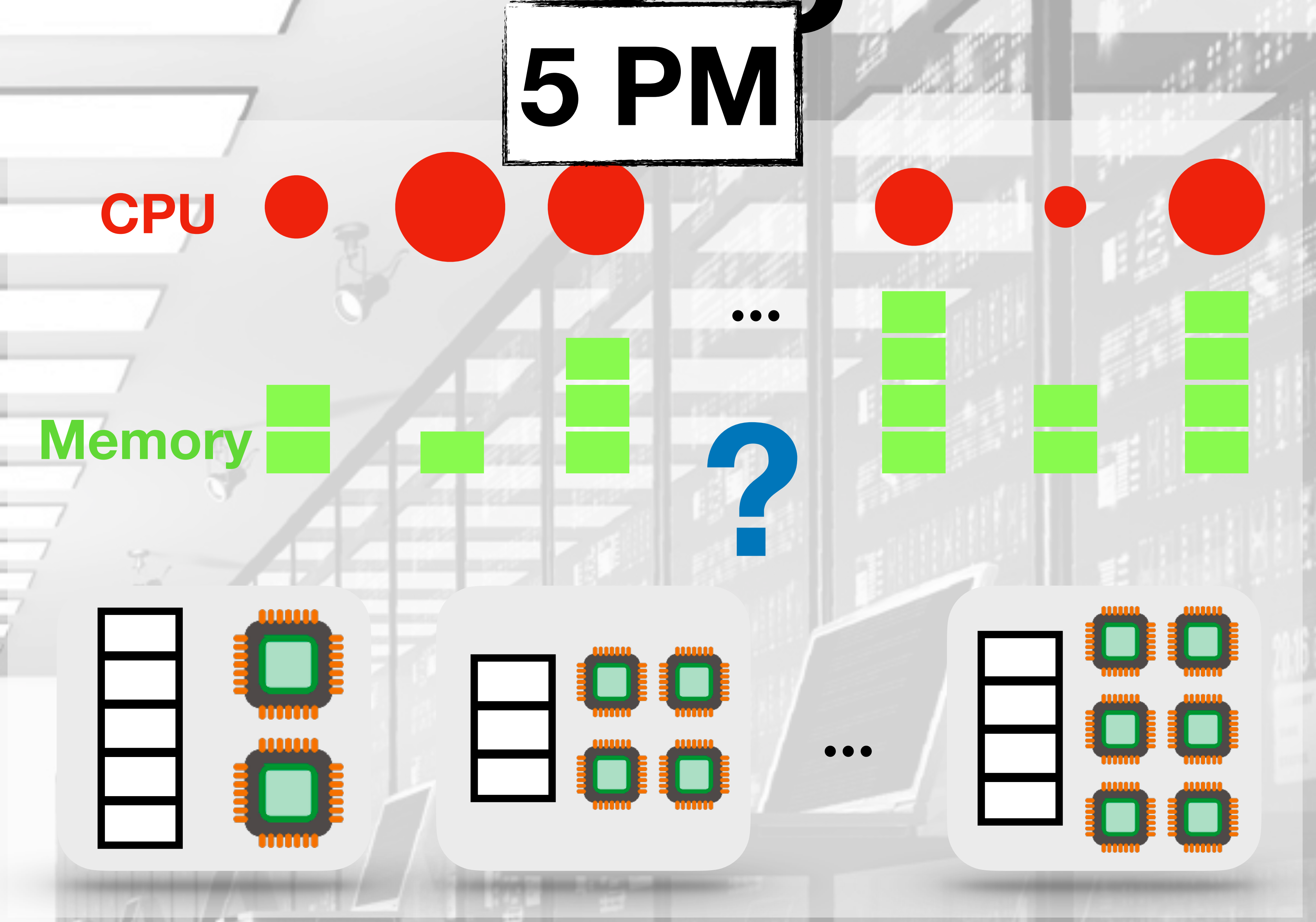
Data Center Resource Management

Data Center Resource Management

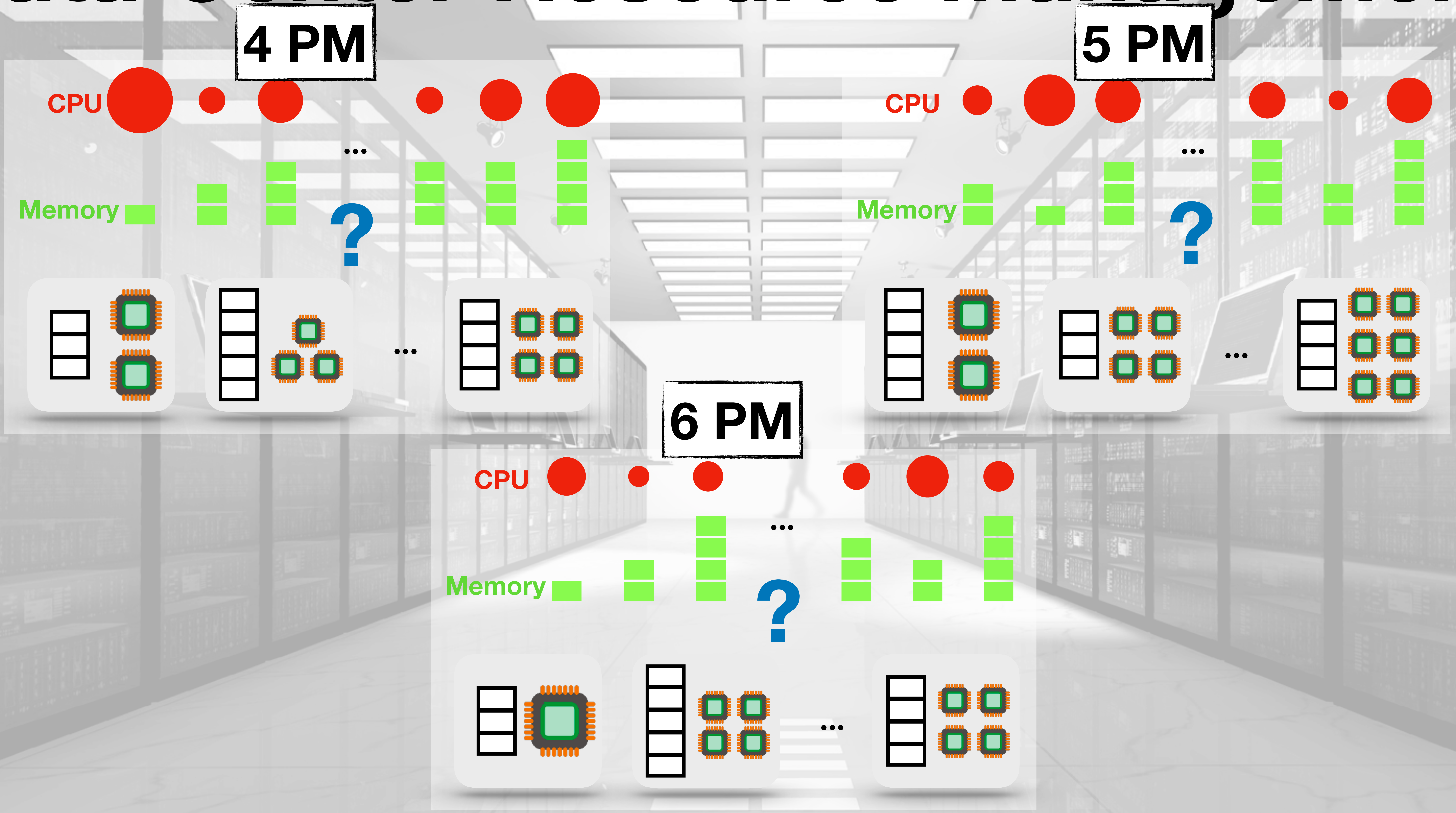


Data Center Resource Management

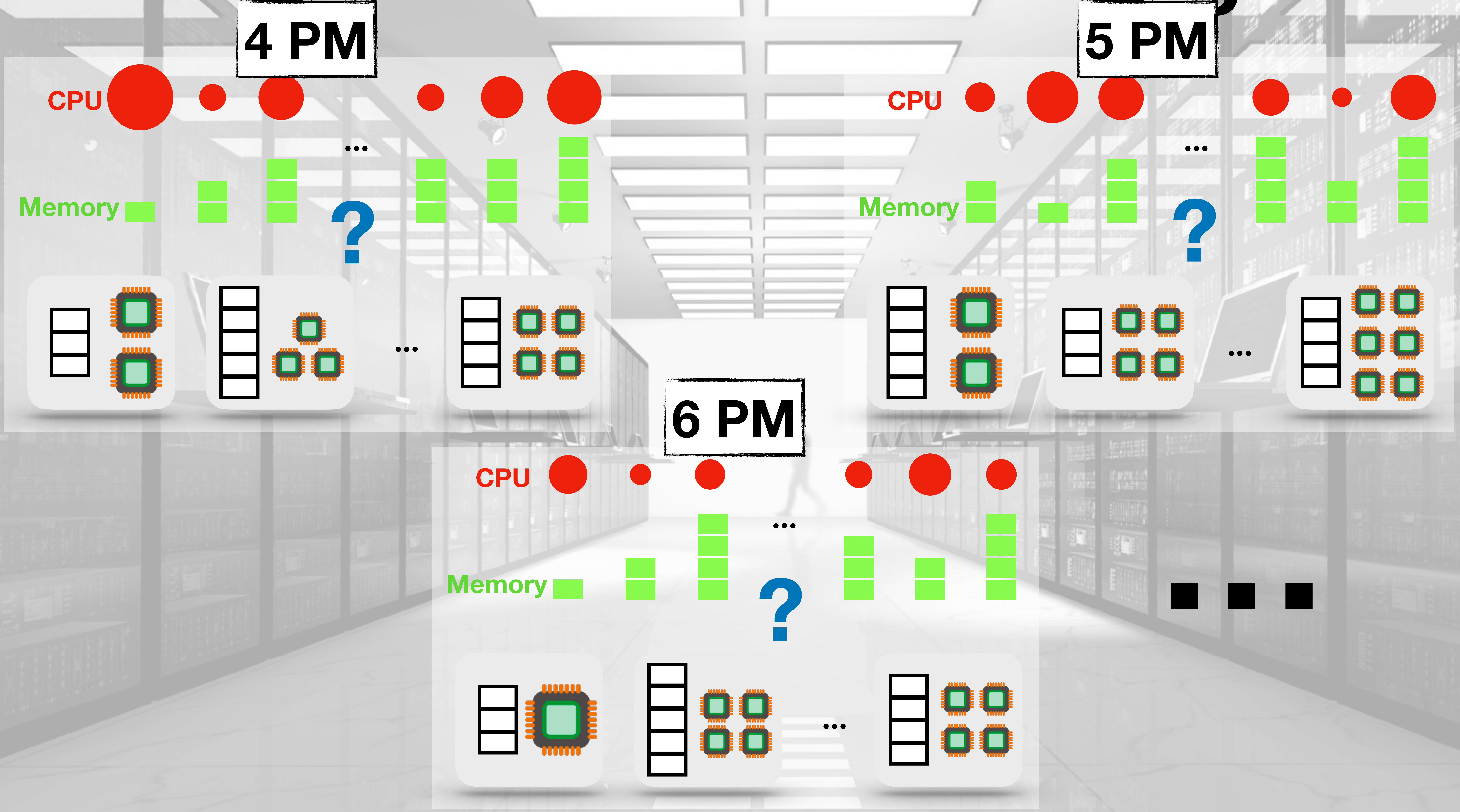




Data Center Resource Management



Data Center Resource Management



Tackling NP-Hard Problems

Paradigm	Design Rationale

Tackling NP-Hard Problems

Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers

Tackling NP-Hard Problems

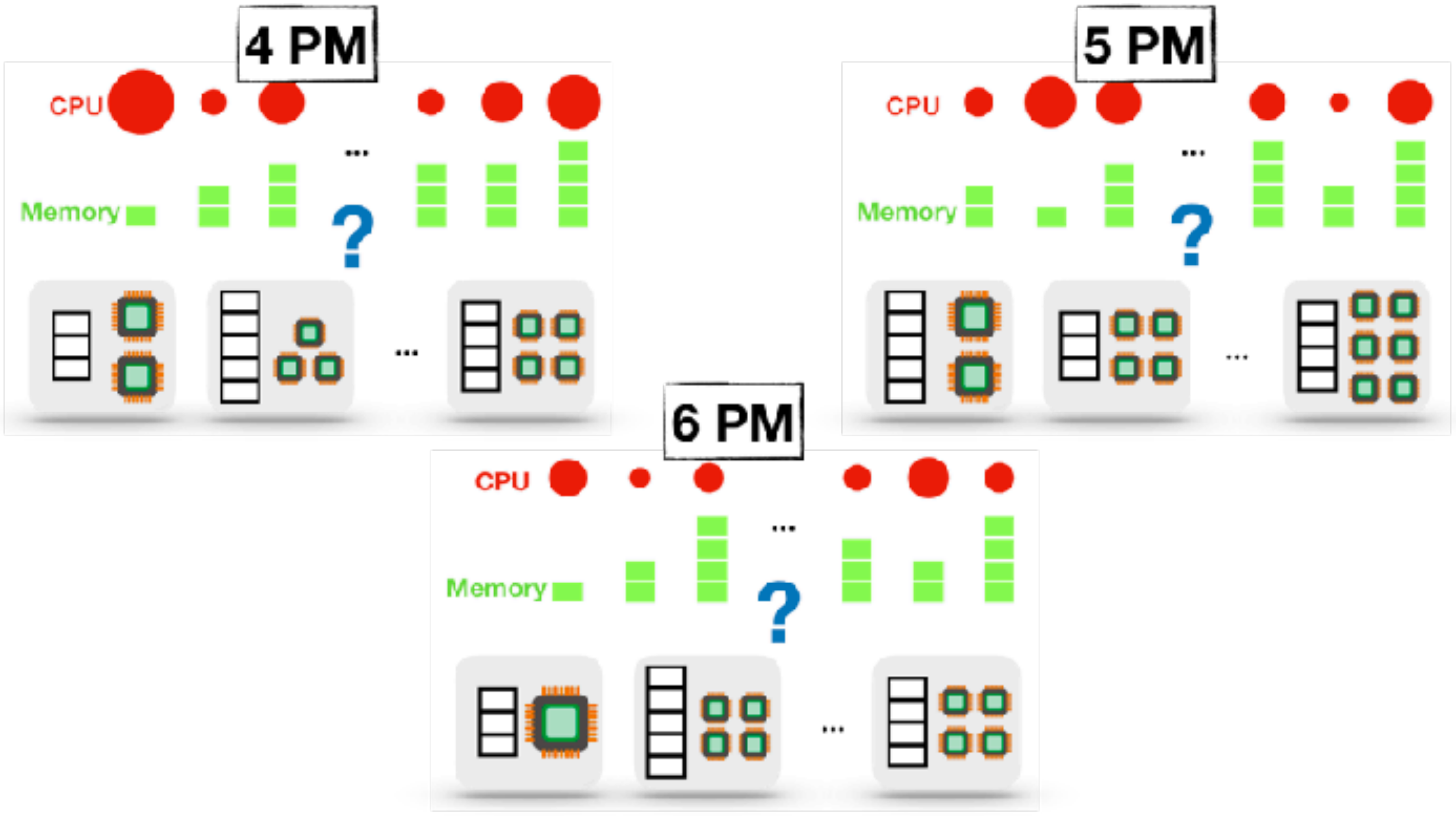
Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees

Tackling NP-Hard Problems

Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

Tackling NP-Hard Problems

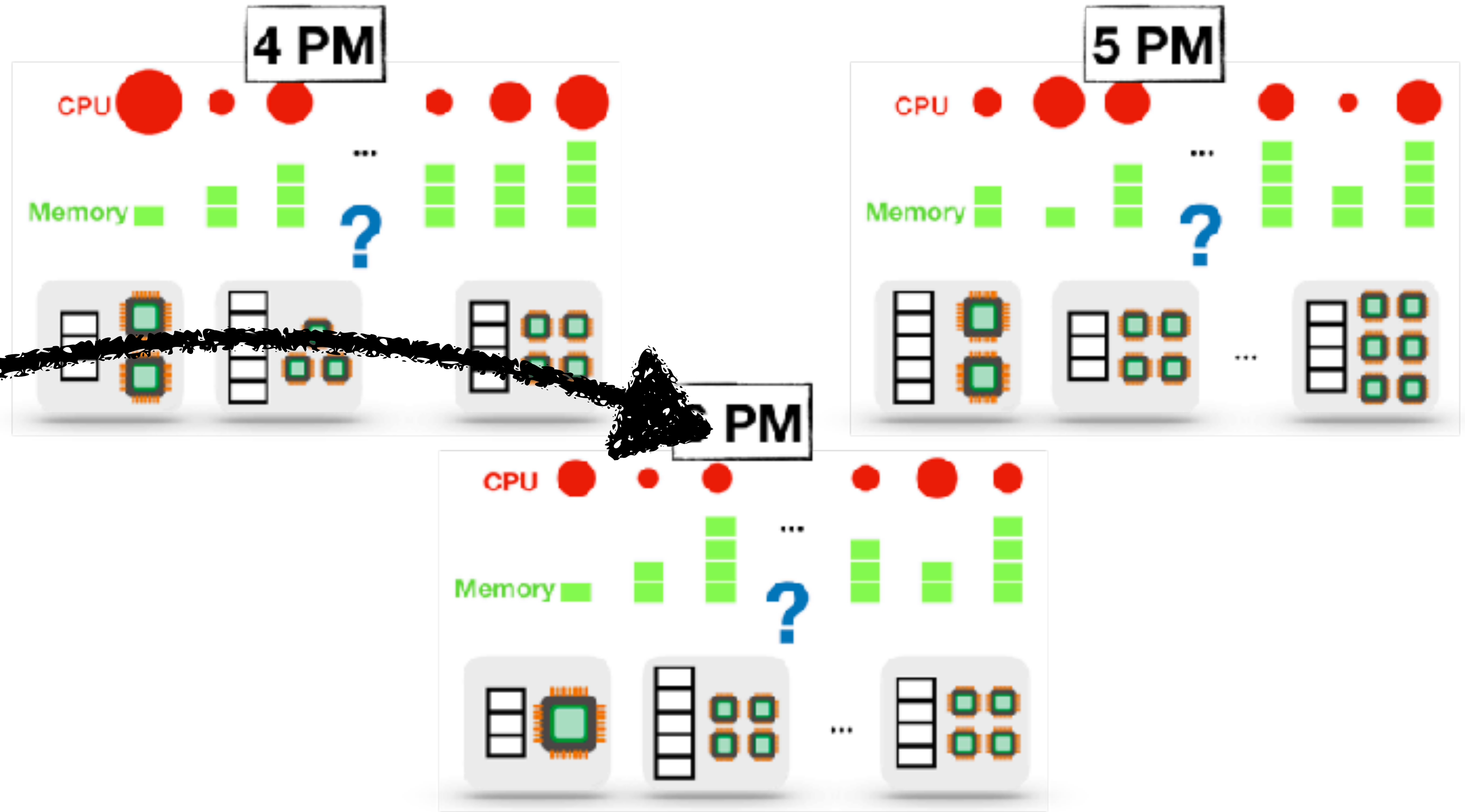
Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**



Tackling NP-Hard Problems

Paradigm	Design Rationale
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

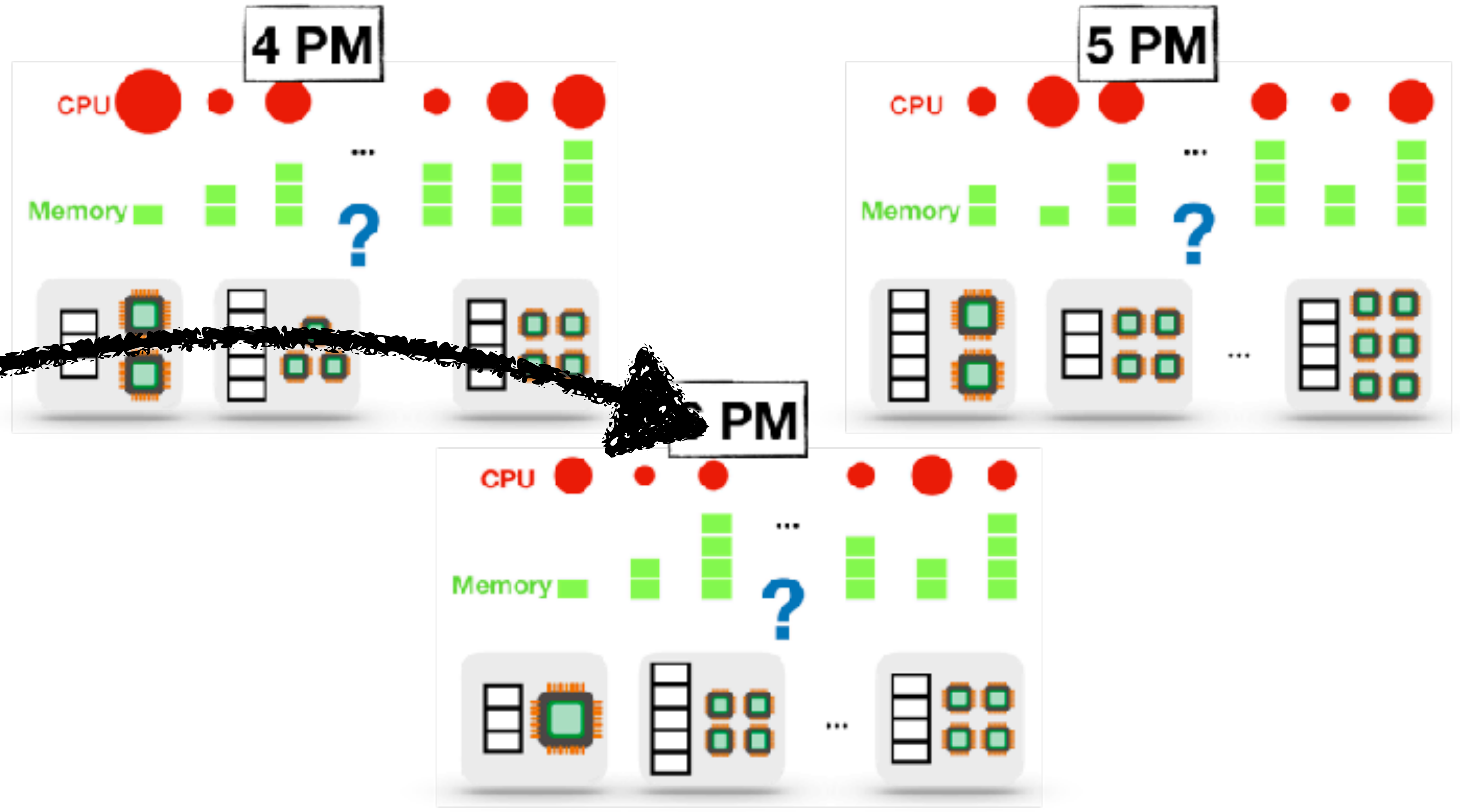
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via...
Exhaustive Search	**Tight formulations** Powerful **Branch-and-Bound** solvers
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

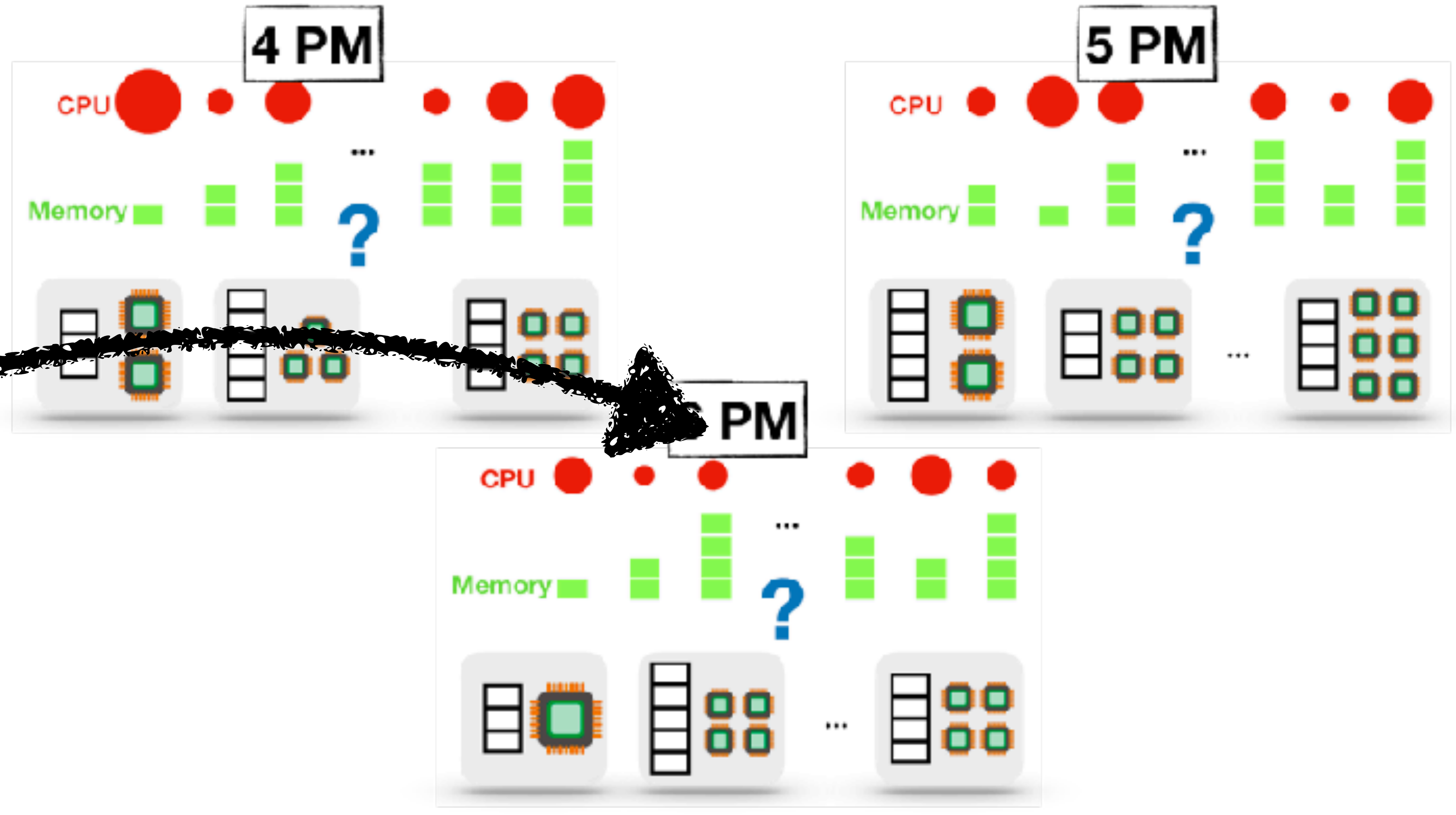
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via...
Exhaustive Search	**Problem-Specific Bounding** functions or **search rules**
Approximation Algorithms	**Good worst-case** guarantees
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

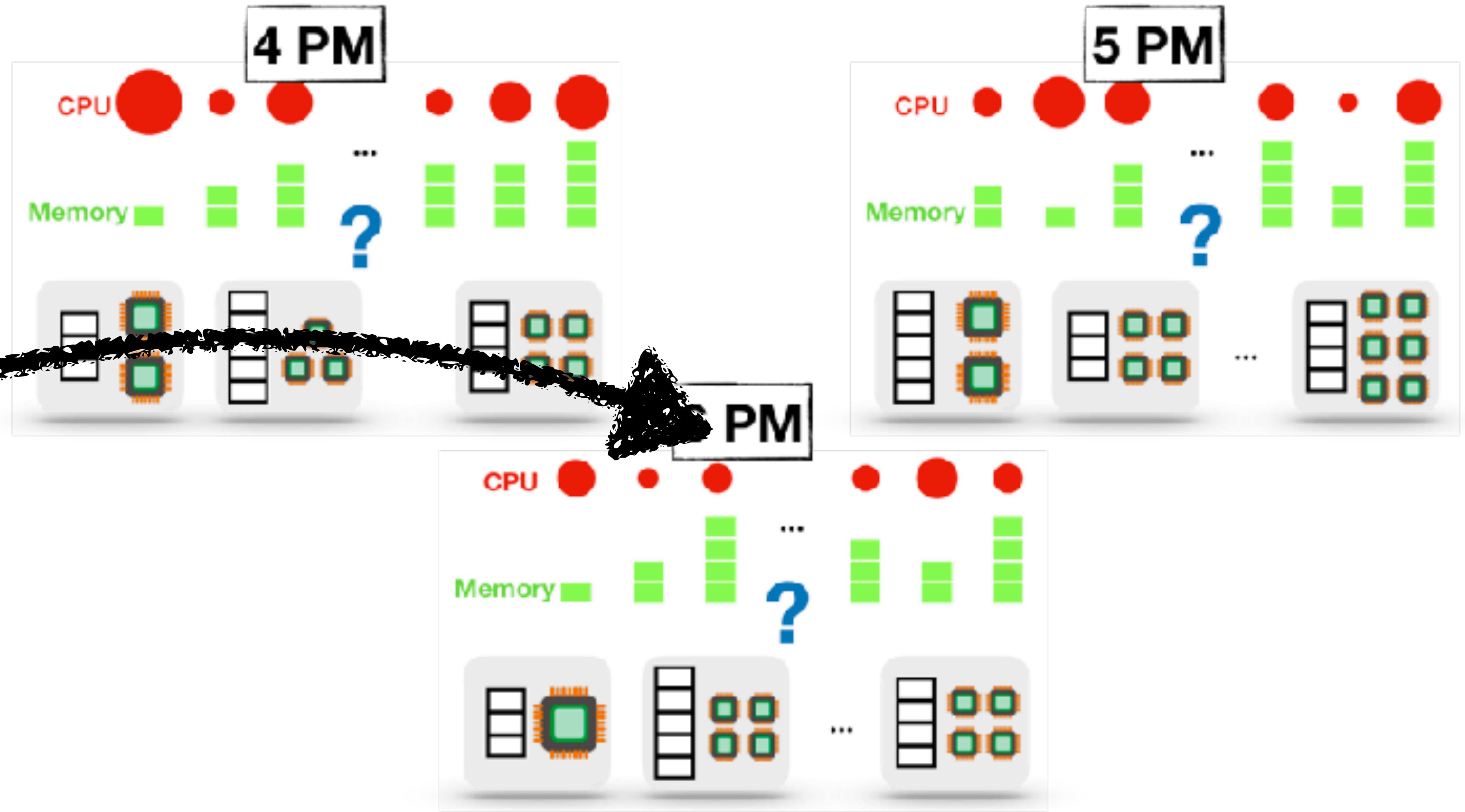
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via...
Exhaustive Search	**Problem-Specific Bounding** functions or **search rules**
Approximation Algorithms	Make explicit **assumptions** on input distribution and **redesign algo.**
Heuristics	**Intuition** exploiting problem structure Empirical **trial-and-error**

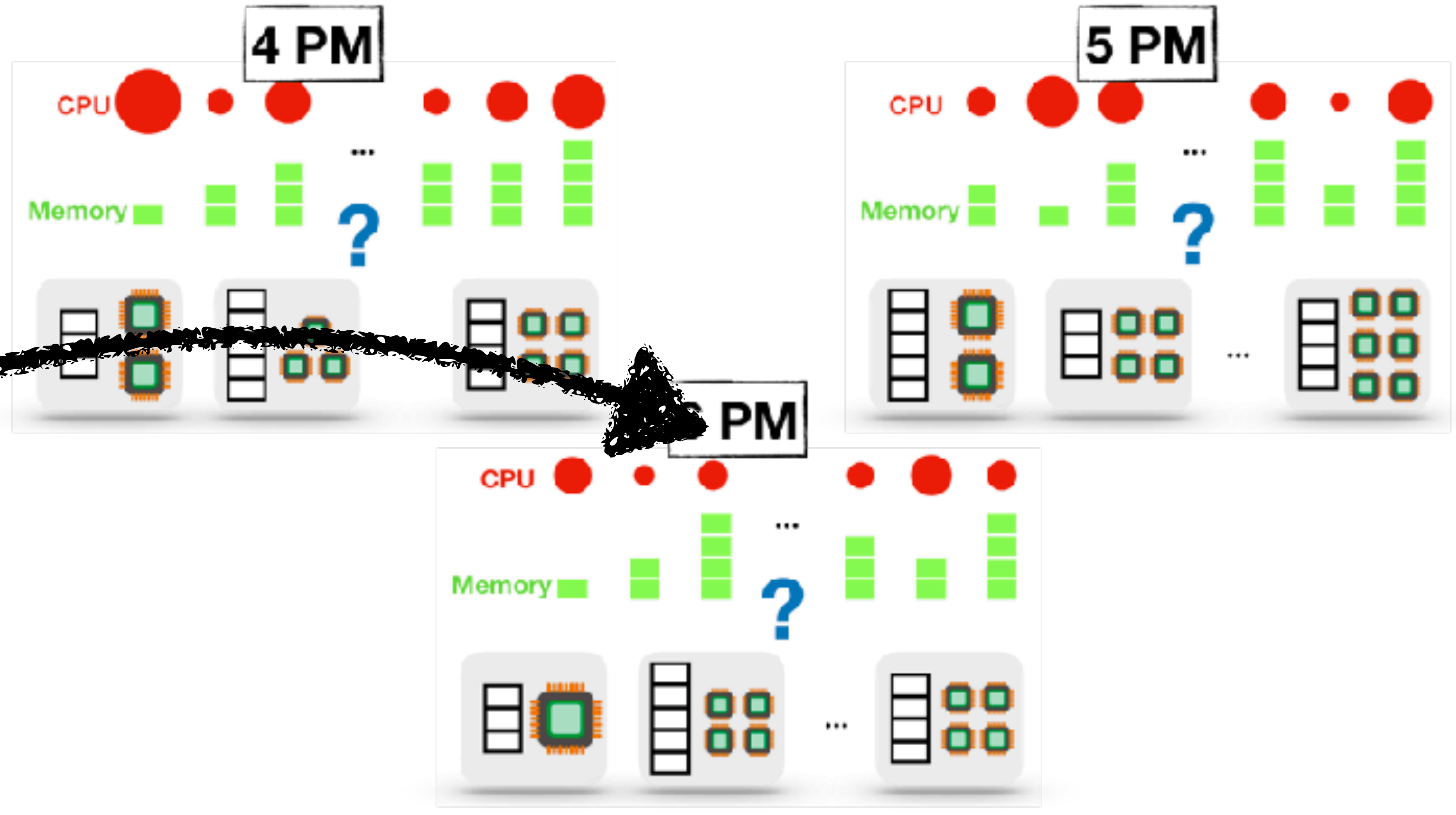
How do you **tailor** the algorithm to **YOUR** instances?



Tackling NP-Hard Problems

Paradigm	Customization via…
Exhaustive Search	**Problem-Specific Bounding** functions or **search rules**
Approximation Algorithms	Make explicit **assumptions** on input distribution and **redesign algo.**
Heuristics	**Analyze** algorithm **behavior** on your inputs; look for **patterns** to exploit

How do you **tailor** the algorithm to **YOUR** instances?



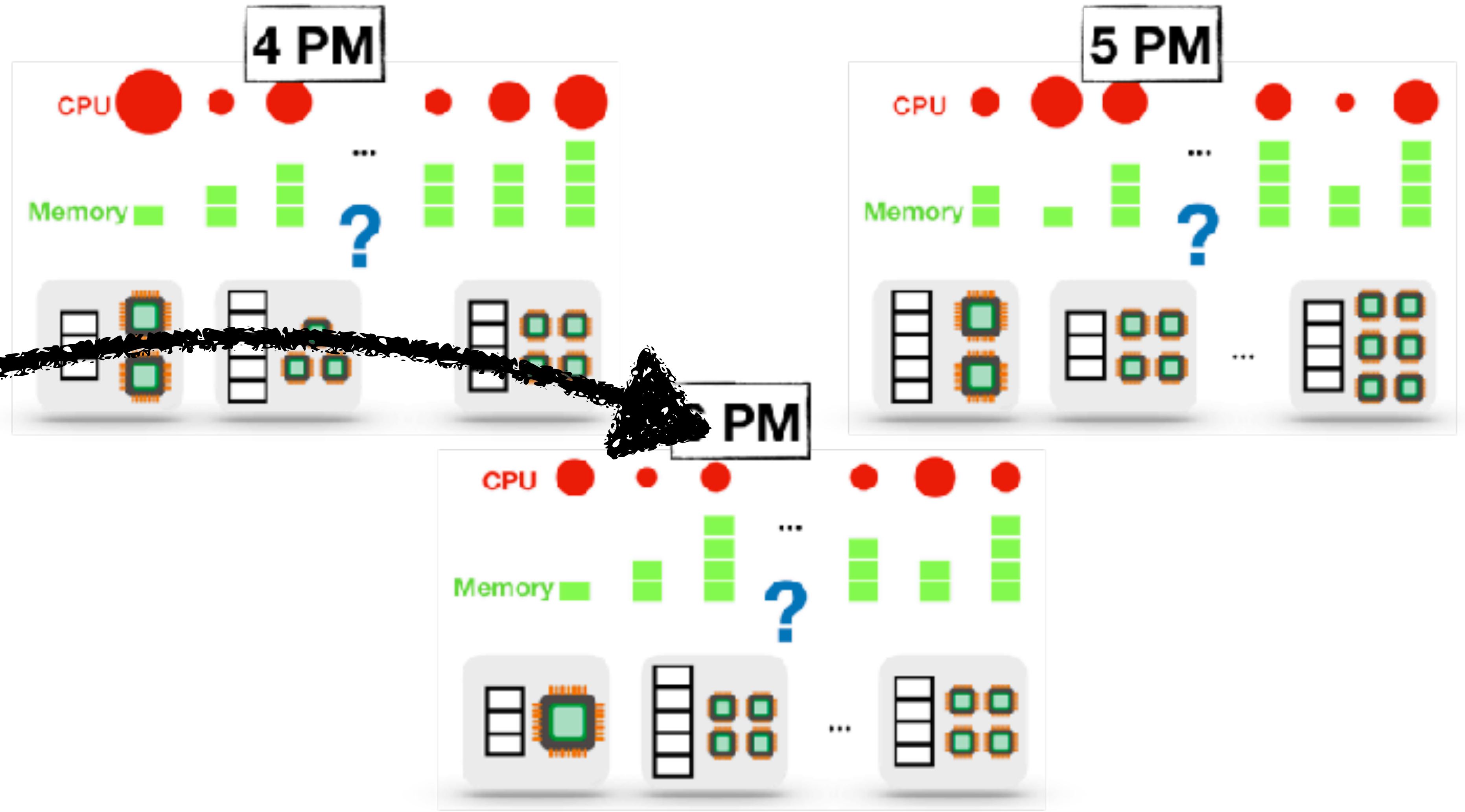
Tackling NP-Hard Problems

Paradigm	Customization via...
	Problem-Specific Bounding functions or **search rules**
	Make explicit **assumptions** on input distribution and **redesign algo.**
	Analyze algorithm **behavior** on your inputs; look for **patterns** to exploit

ANSWER:

Manual intellectual/ experimental **effort** required

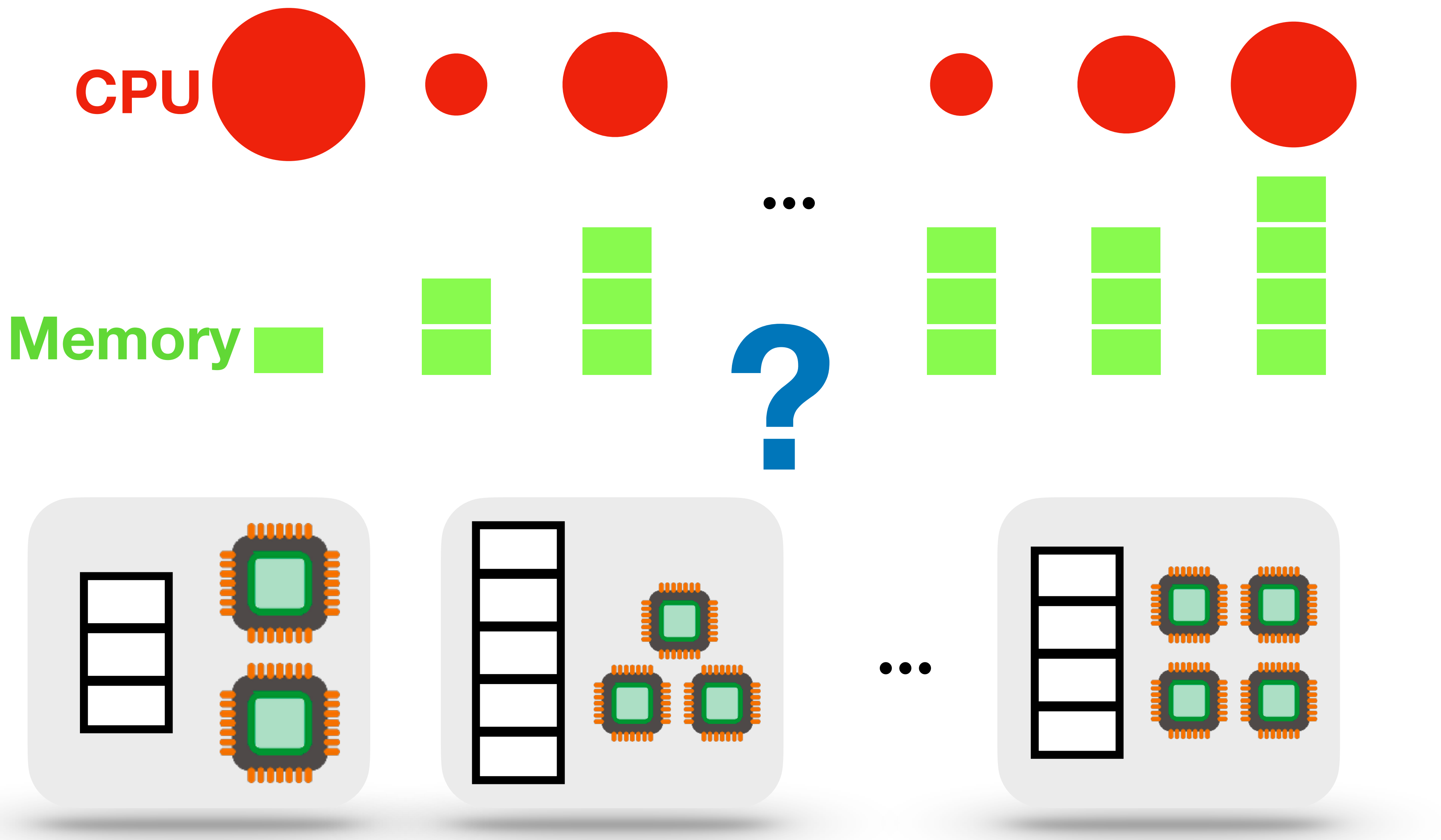
How do you **tailor** the algorithm to **YOUR** instances?



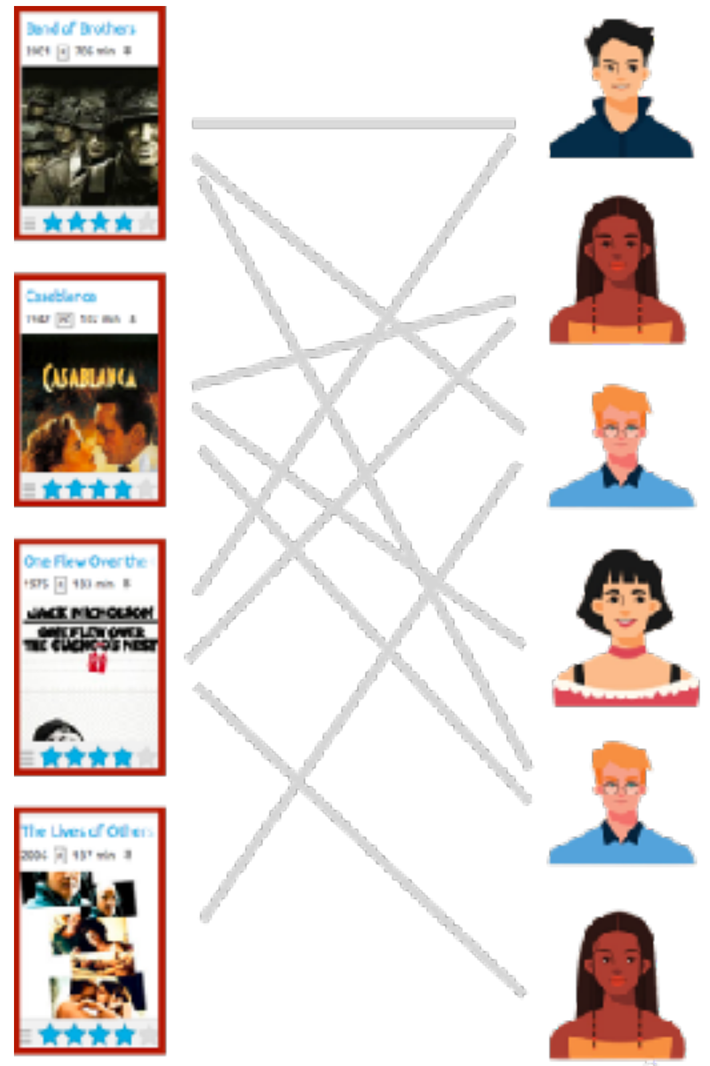
Opportunity

Automatically tailor algorithms to a **family** of **instances**

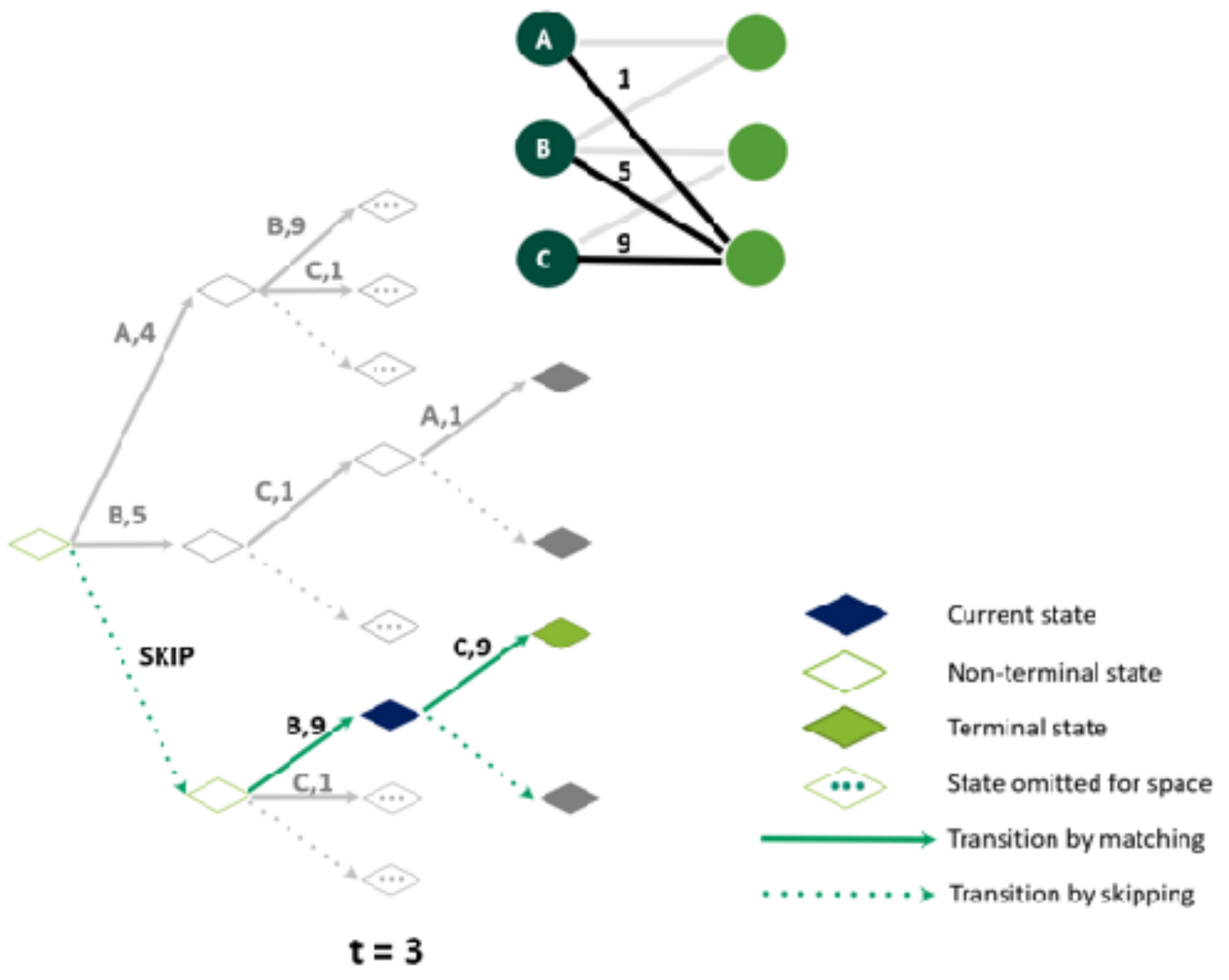
Data Center Resource Management



Online Matching....



Matching users to movies



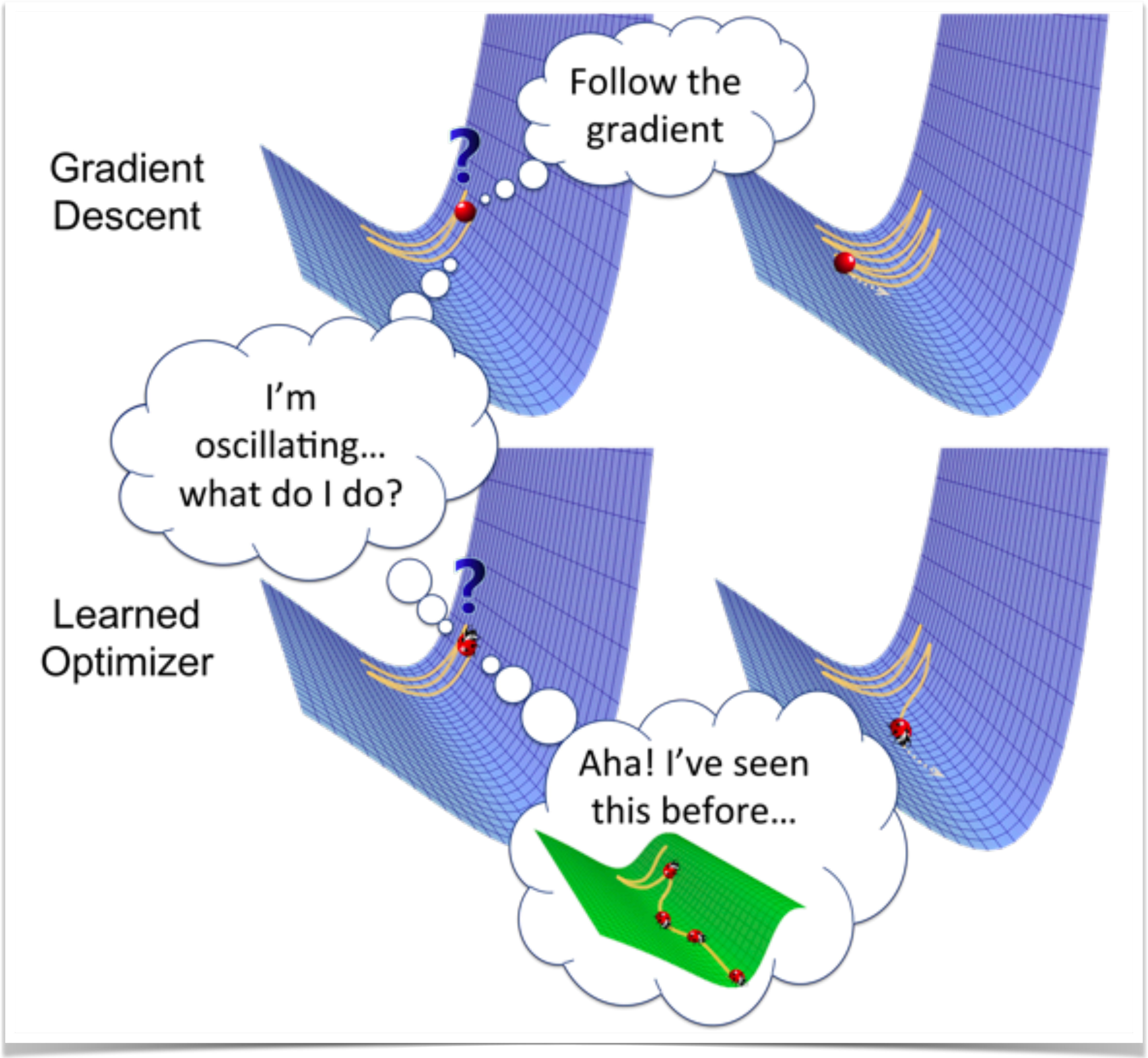
MDP for Online Bipartite Matching

Survey on the topic: Bengio, Yoshua, Andrea Lodi, and Antoine Prouvost. "Machine learning for combinatorial optimization: a methodological tour d’horizon." EJOR 290.2 (2021): 405-421.

Pierre Selim, 2-opt procedure, distributed under a CC BY-SA 3.0 license.

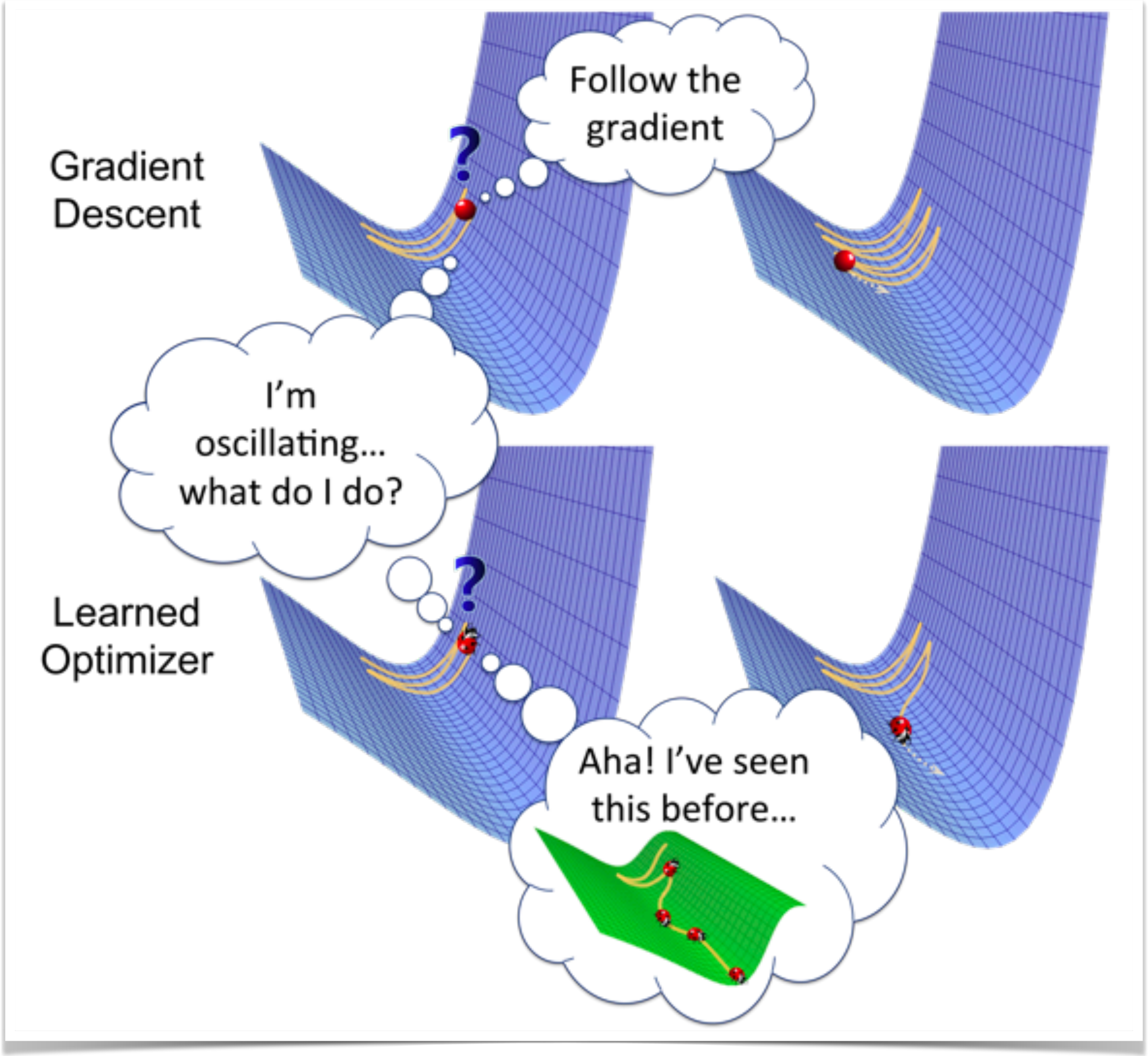
Warm-up: Learning in Gradient Descent

Source: Ke Li, http://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl
Li, Ke, and Jitendra Malik. "Learning to optimize." ICLR, 2017



Warm-up: Learning in Gradient Descent

Source: Ke Li, http://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl
Li, Ke, and Jitendra Malik. "Learning to optimize." ICLR, 2017



Algorithm 1 General structure of optimization algorithms

Require: Objective function f

$x^{(0)} \leftarrow$ random point in the domain of f

for $i = 1, 2, \ldots$ **do**

$\Delta x \leftarrow \boxed{\phi(\{x^{(j)}, f(x^{(j)}), \nabla f(x^{(j)})\}_{j=0}^{i-1})}$

if stopping condition is met **then**

return $x^{(i-1)}$

end if

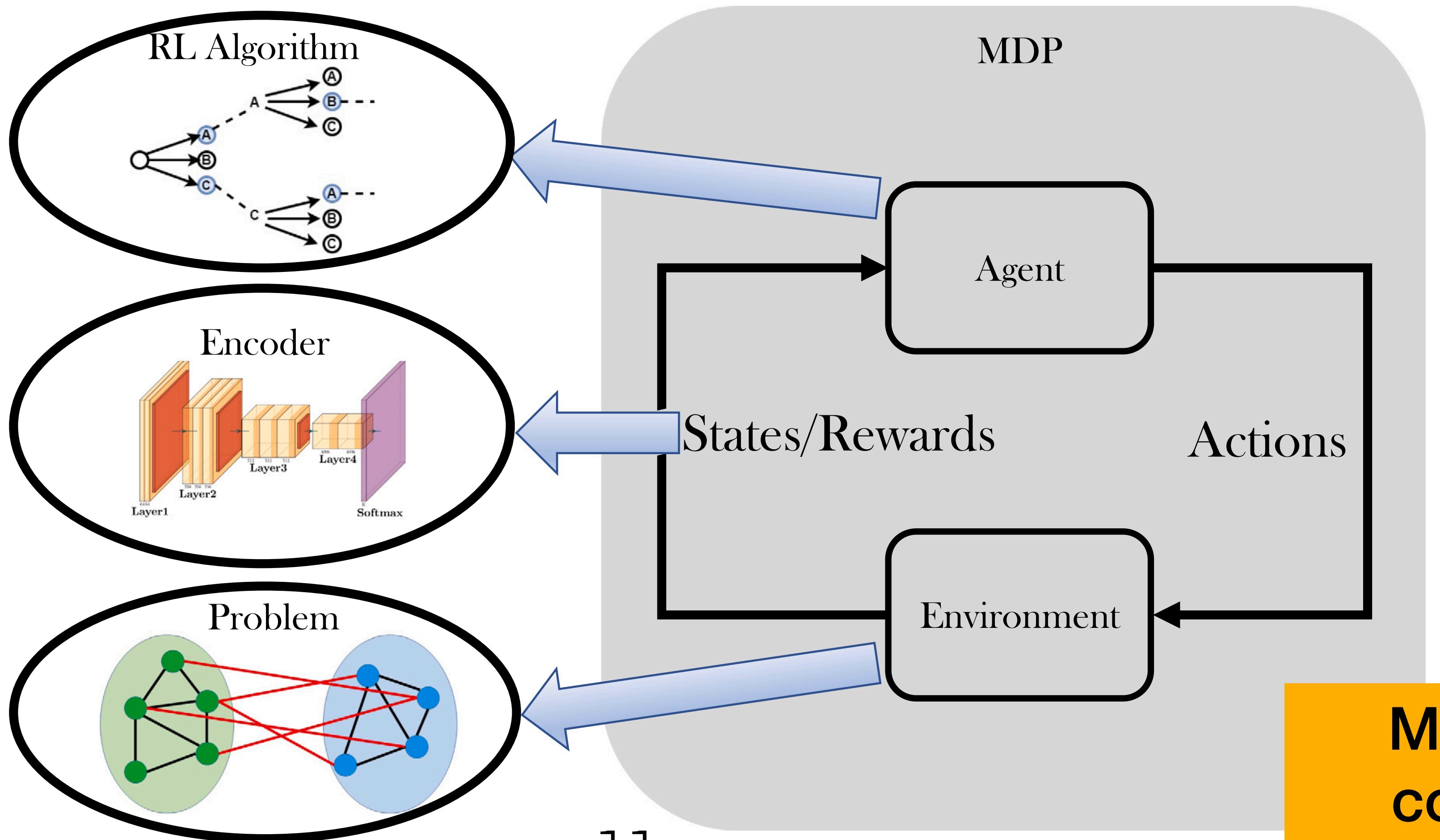
$x^{(i)} \leftarrow x^{(i-1)} + \Delta x$

end for

Gradient Descent	$\phi(\cdot) = -\gamma \nabla f(x^{(i-1)})$
Momentum	$\phi(\cdot) = -\gamma \left(\sum_{j=0}^{i-1} \alpha^{i-1-j} \nabla f(x^{(j)}) \right)$
Learned Algorithm	$\phi(\cdot) =$ Neural Net

Today's foci:

- **Modeling** iterative algorithms or online policies as MDPs
- **Representation** via Feedforward Networks or Graph Neural Networks
- **Evaluation** best practices in the context of combinatorial optimization

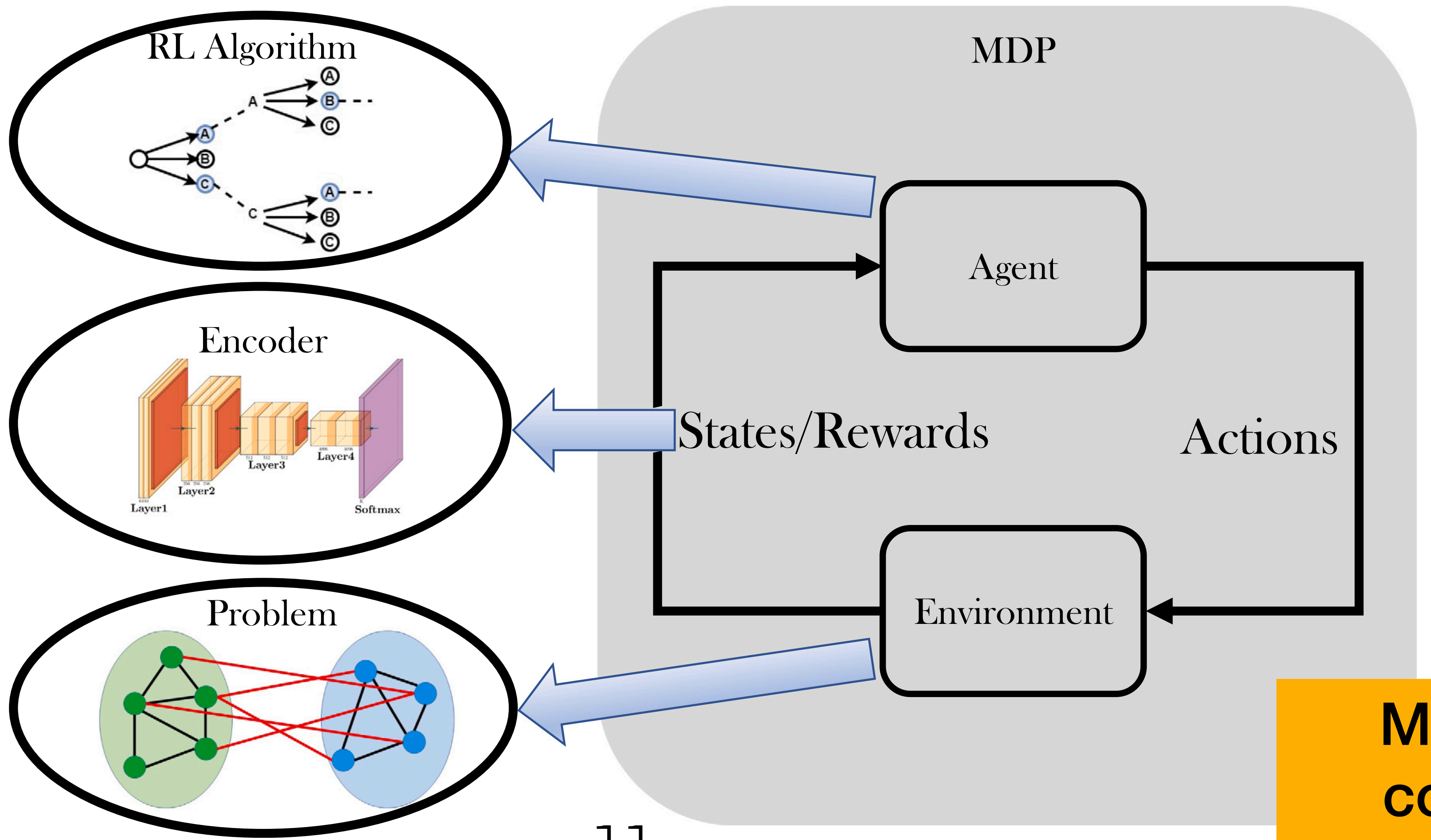


Mazyavkina, Nina, et al. "Reinforcement learning for combinatorial optimization: A survey." Computers & Operations Research (2021): 105400.

Today's foci:

How do we make RL work as a tool for designing combinatorial optimization algorithms?

- **Modeling** iterative algorithms or online policies as MDPs
- **Representation** via Feedforward Networks or Graph Neural Networks
- **Evaluation** best practices in the context of combinatorial optimization



Mazyavkina, Nina, et al. "Reinforcement learning for combinatorial optimization: A survey." Computers & Operations Research (2021): 105400.

Deep Reinforcement Learning for Online Combinatorial Optimization

The Case of Bipartite Matching

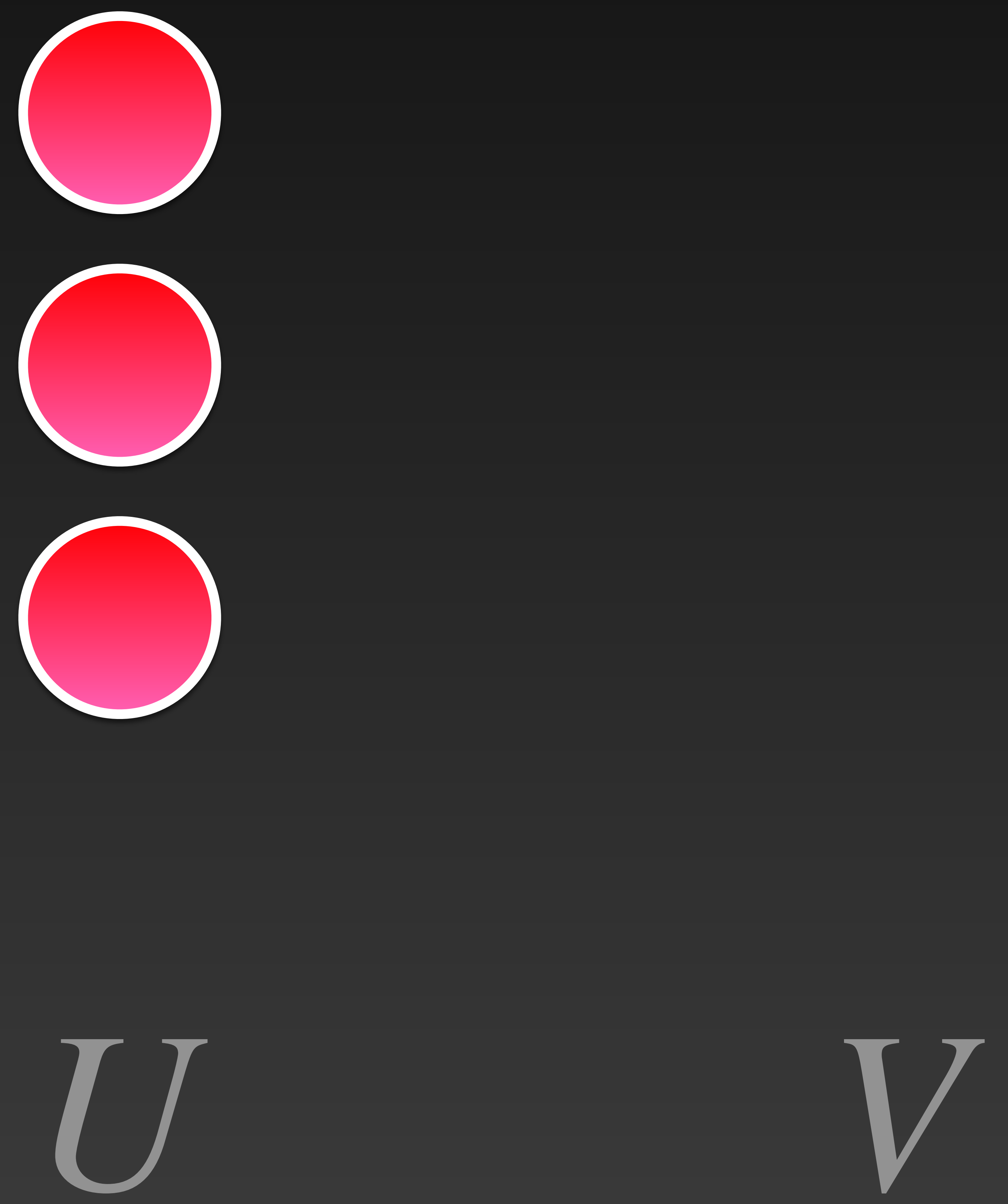
Joint work with Mohammad Ali Alomrani & Reza Morajev (Toronto)

Published in Transactions of Machine Learning Research (TMLR)

arXiv:2109.10380

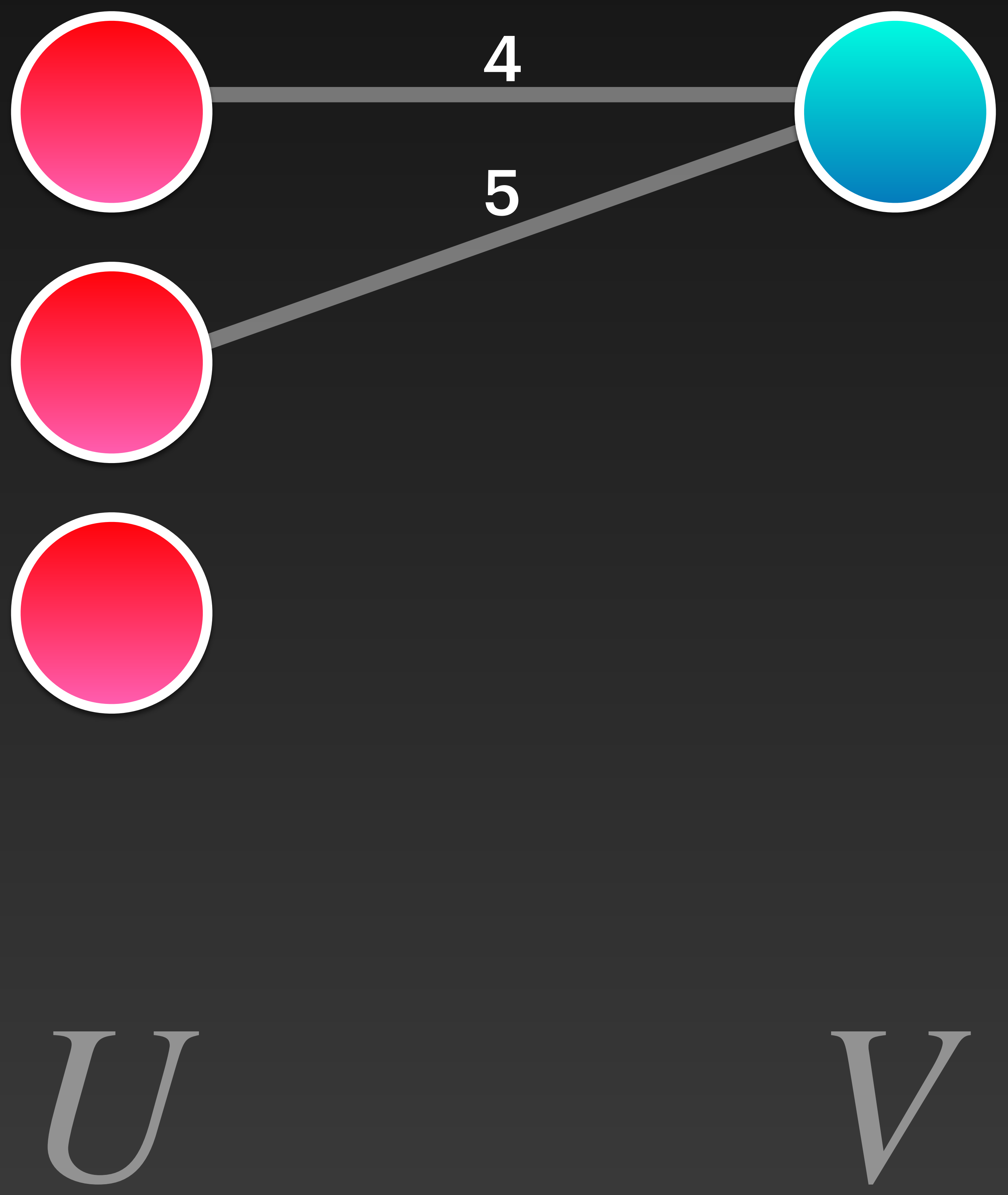
Edge-Weighted Online Bipartite Matching

(E-OBM)

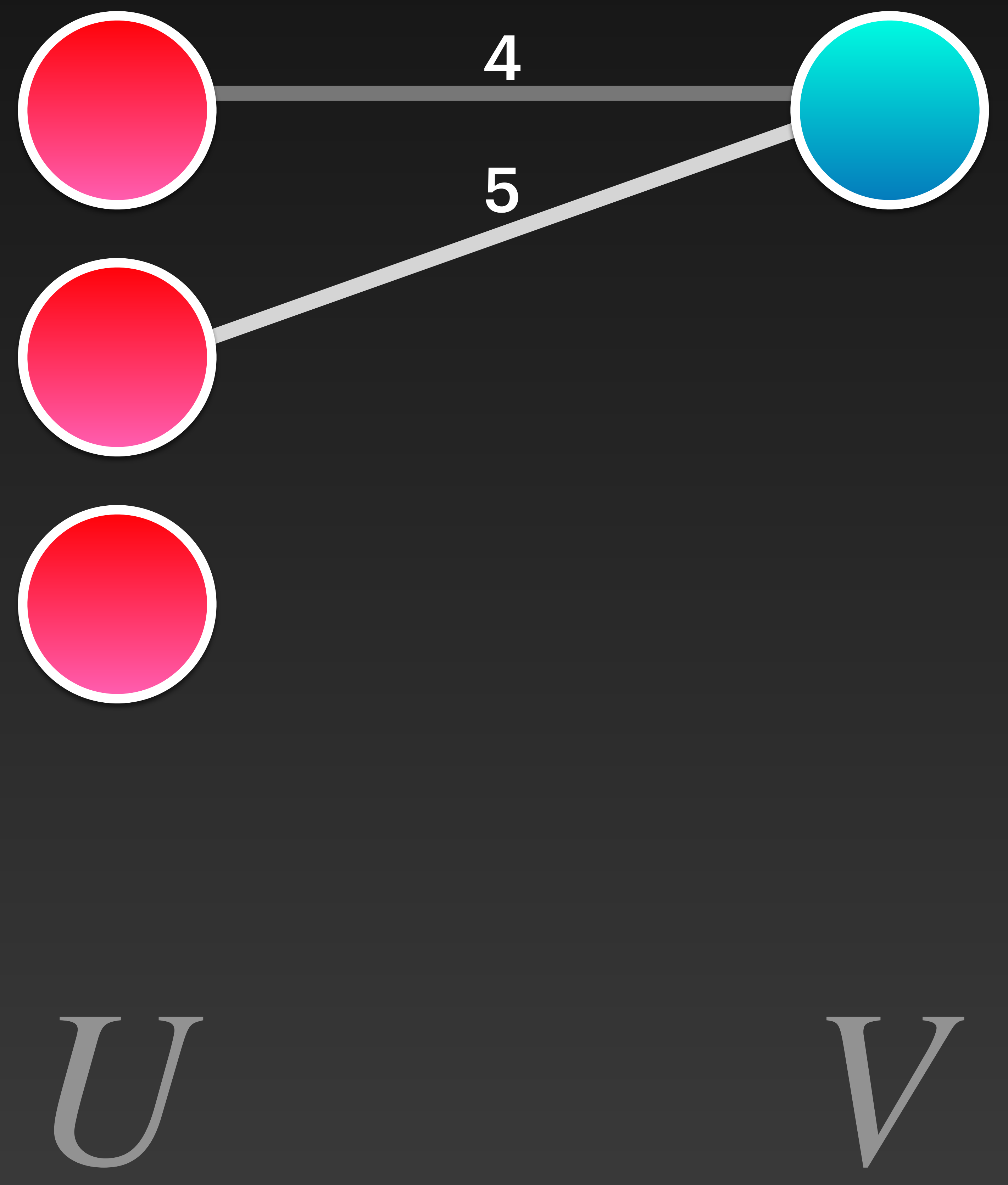


Edge-Weighted Online Bipartite Matching

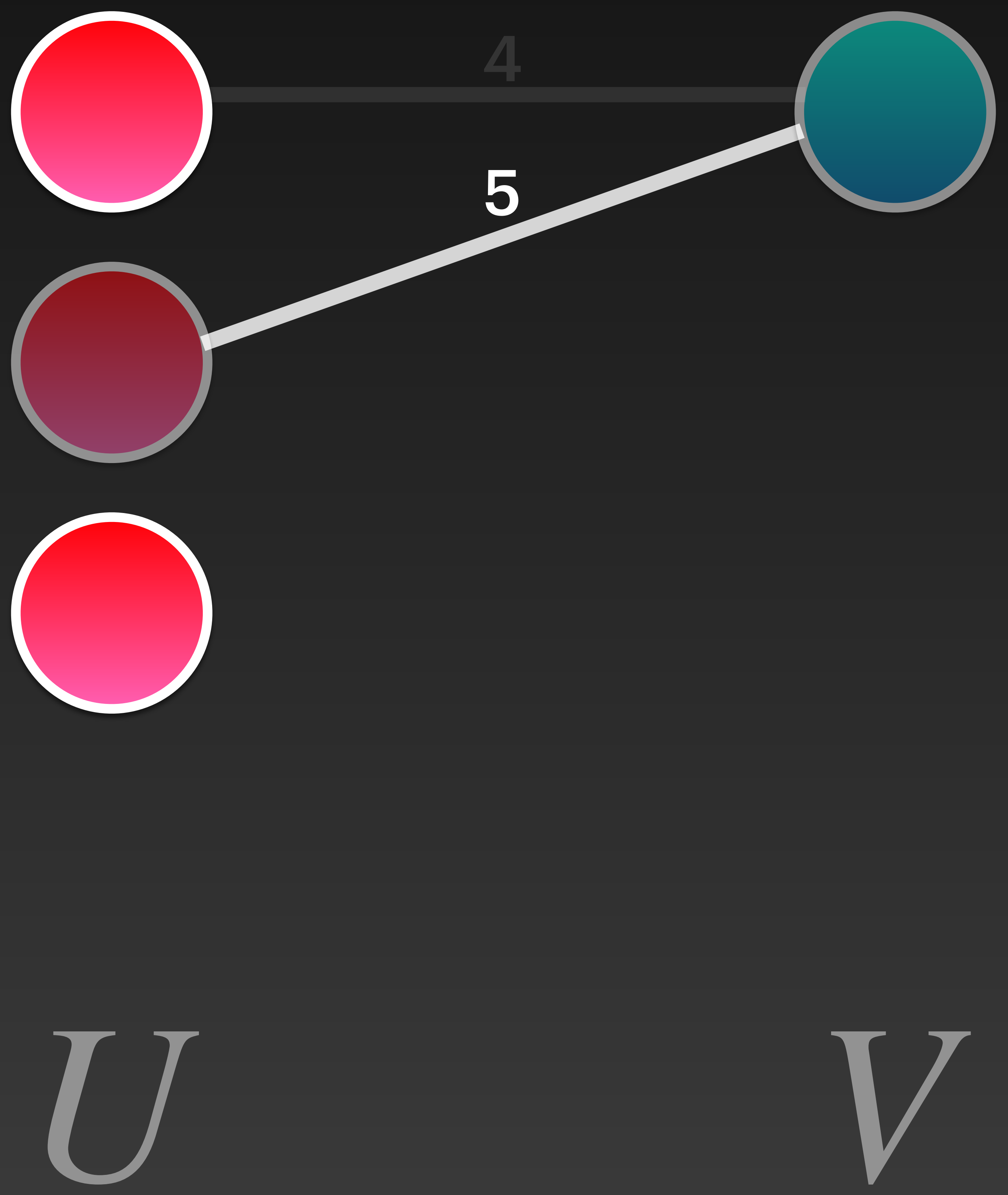
(E-OBM)



Edge-Weighted Online Bipartite Matching
(E-OBM)

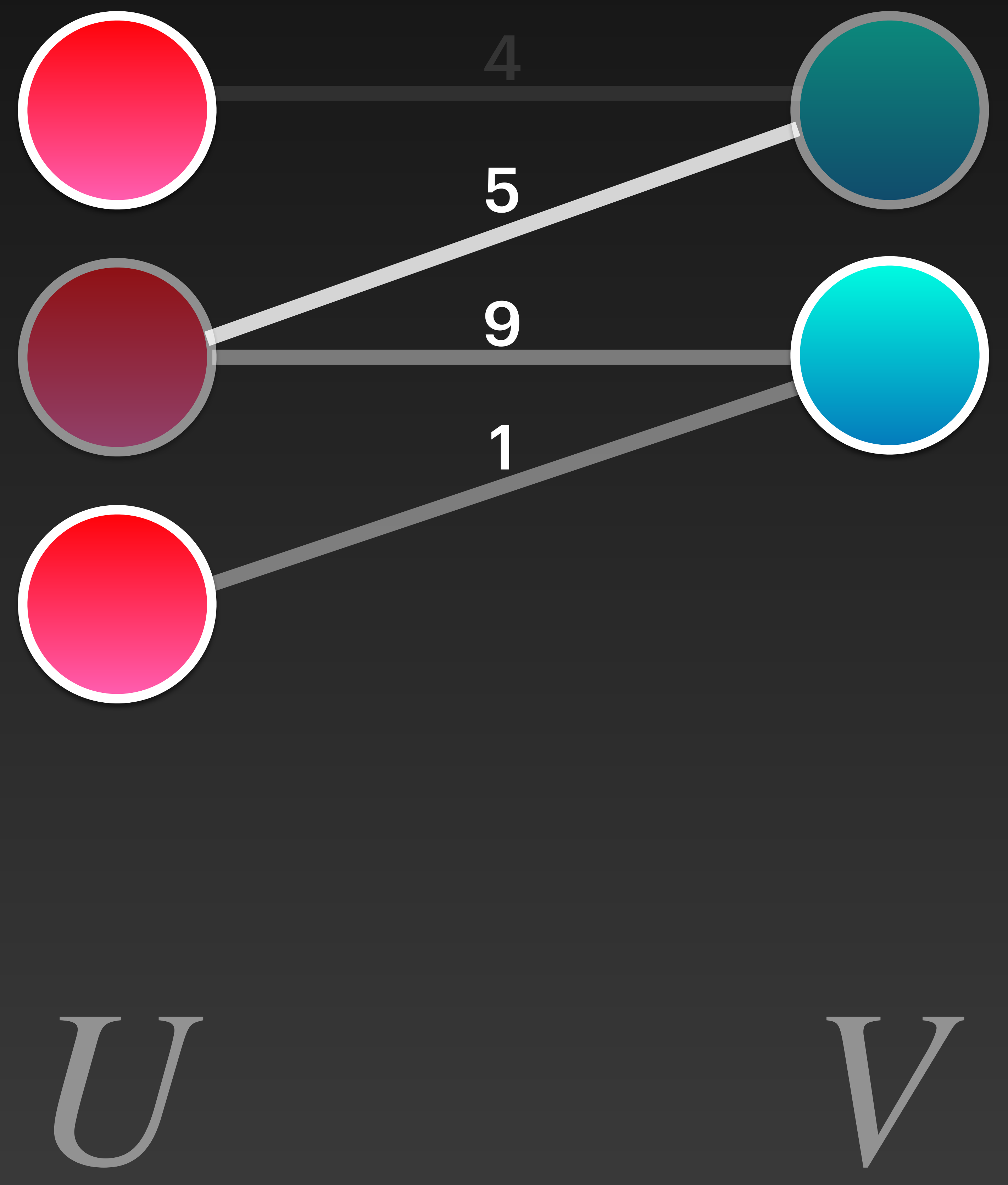


Edge-Weighted Online Bipartite Matching
(E-OBM)



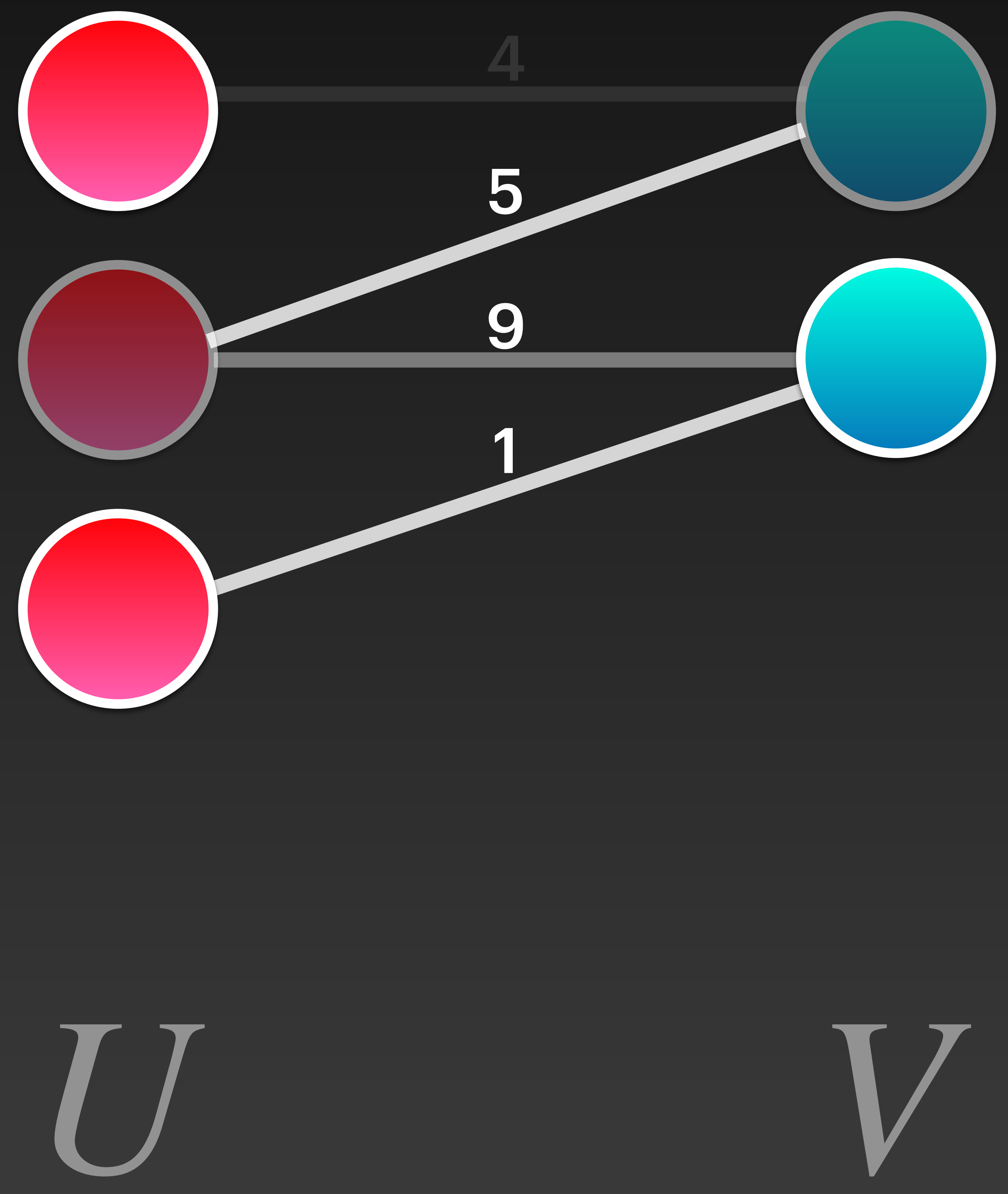
Edge-Weighted Online Bipartite Matching

(E-OBM)



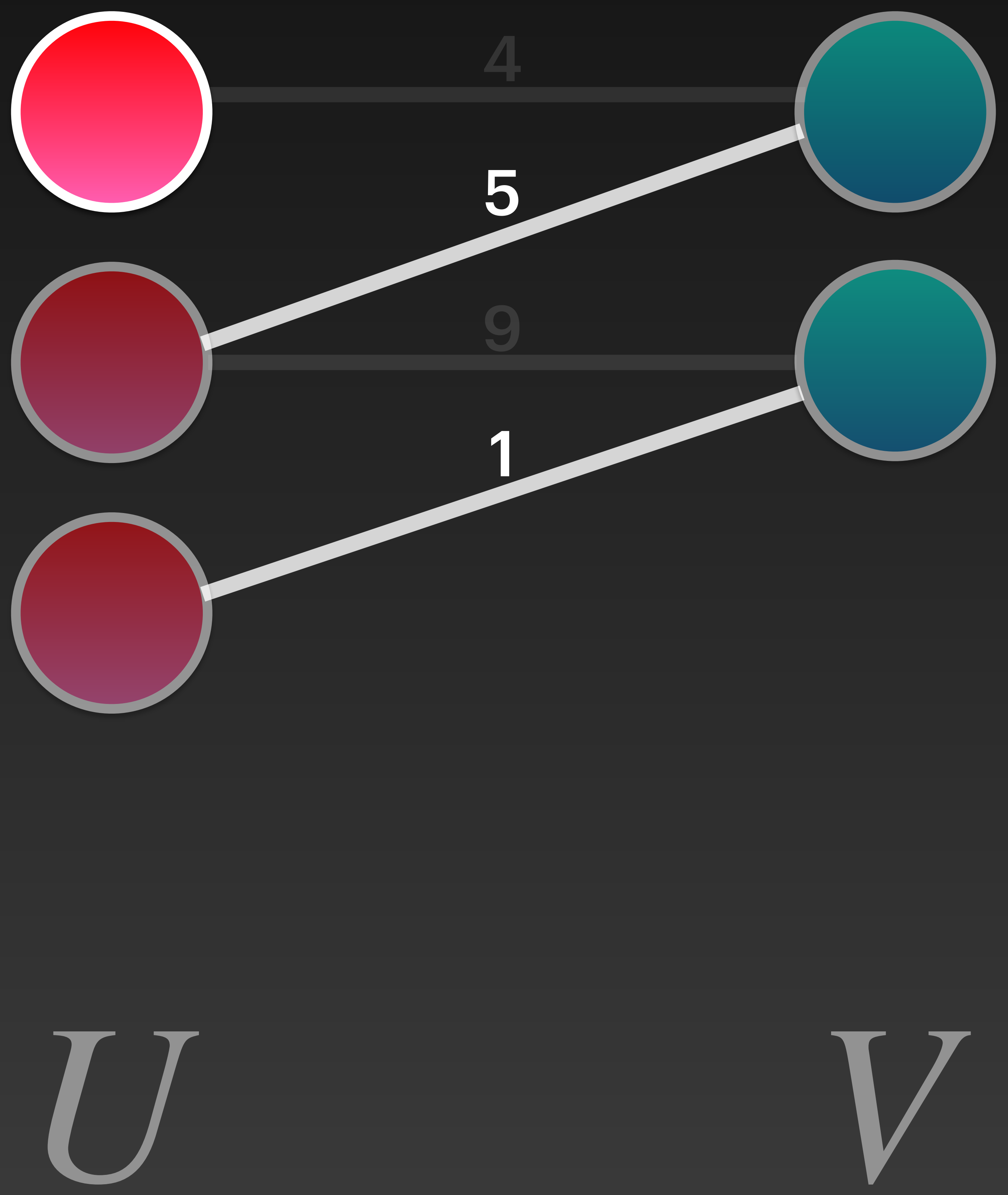
Edge-Weighted Online Bipartite Matching

(E-OBM)

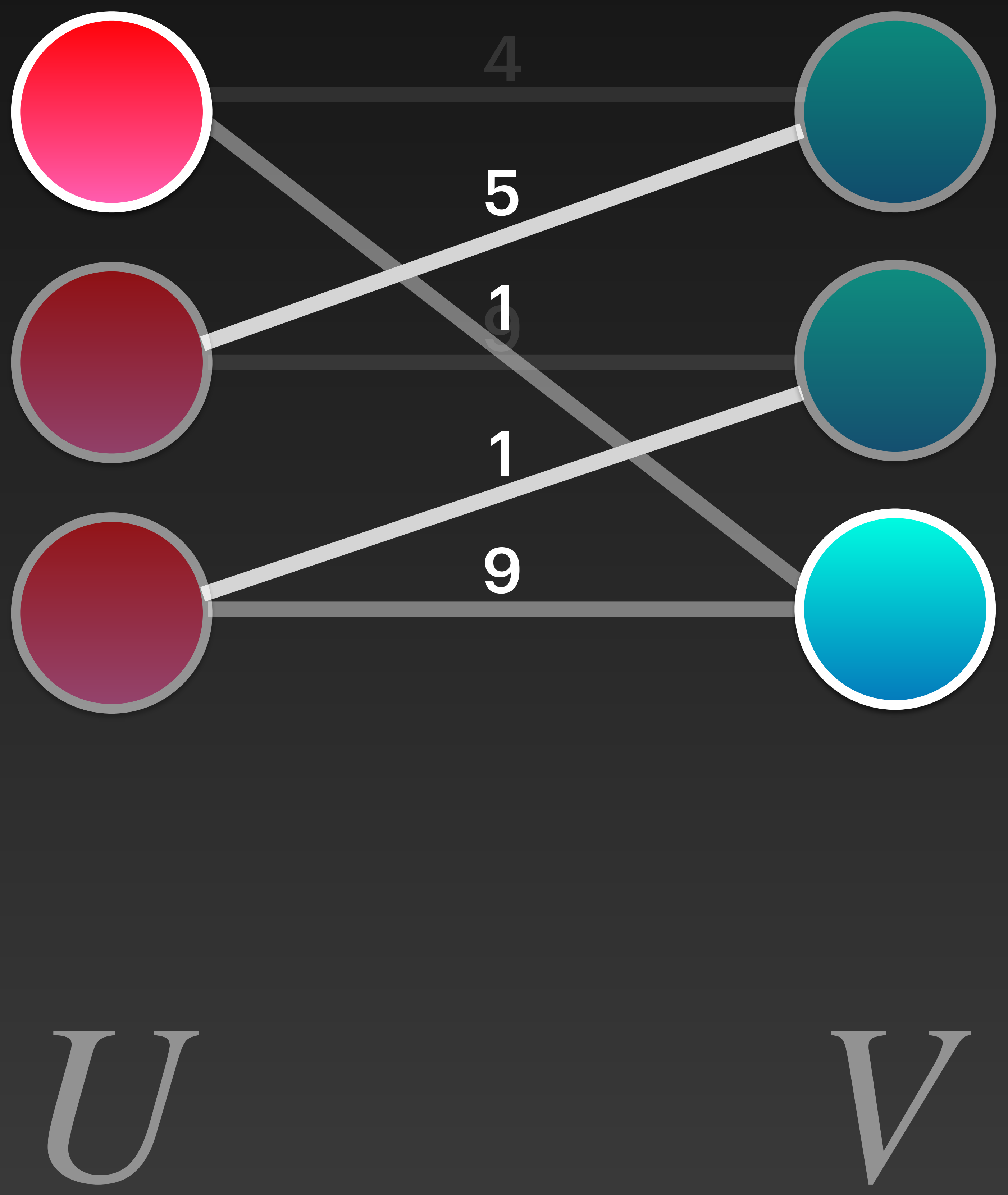


Edge-Weighted Online Bipartite Matching

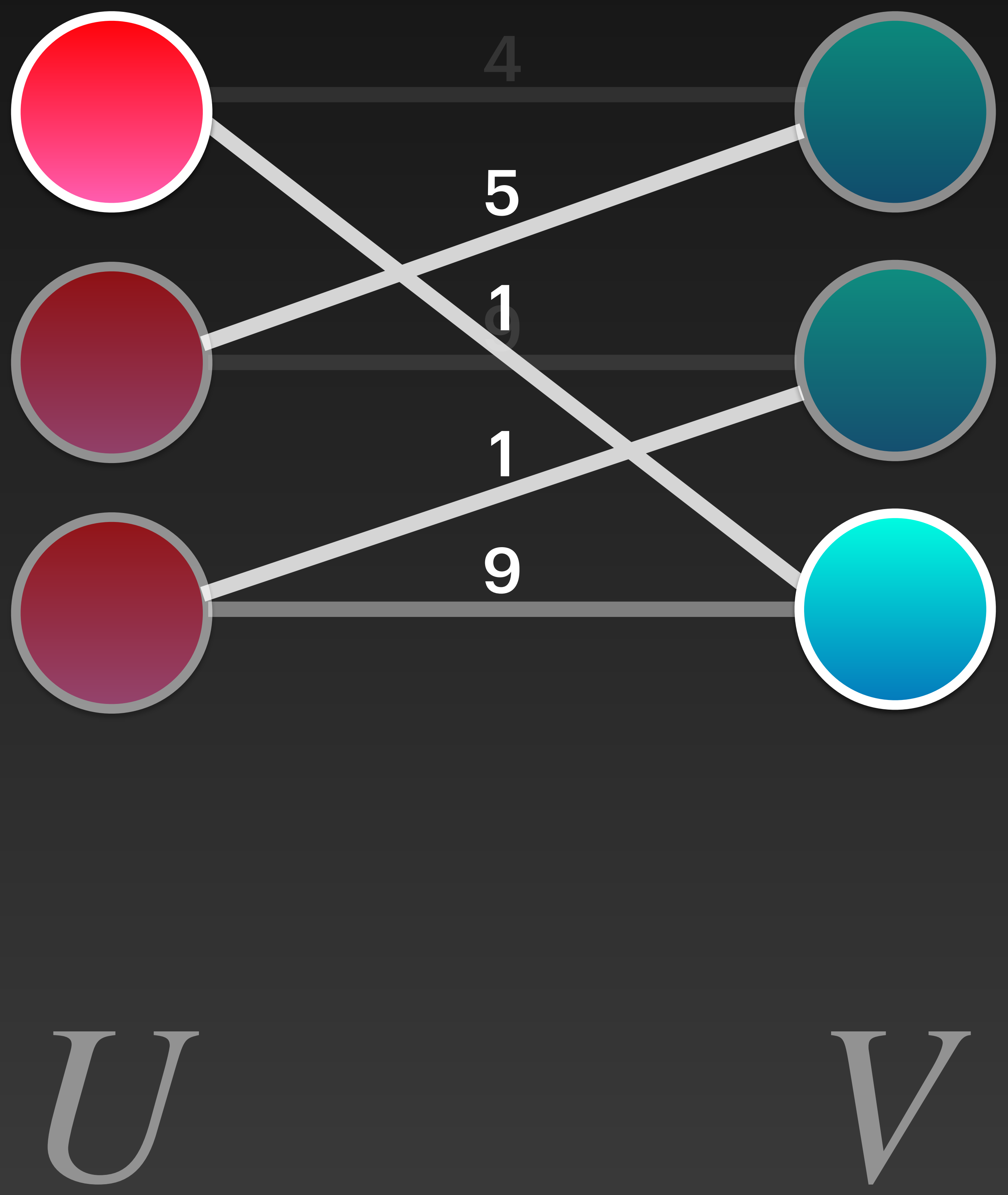
(E-OBM)



Edge-Weighted Online Bipartite Matching
(E-OBM)

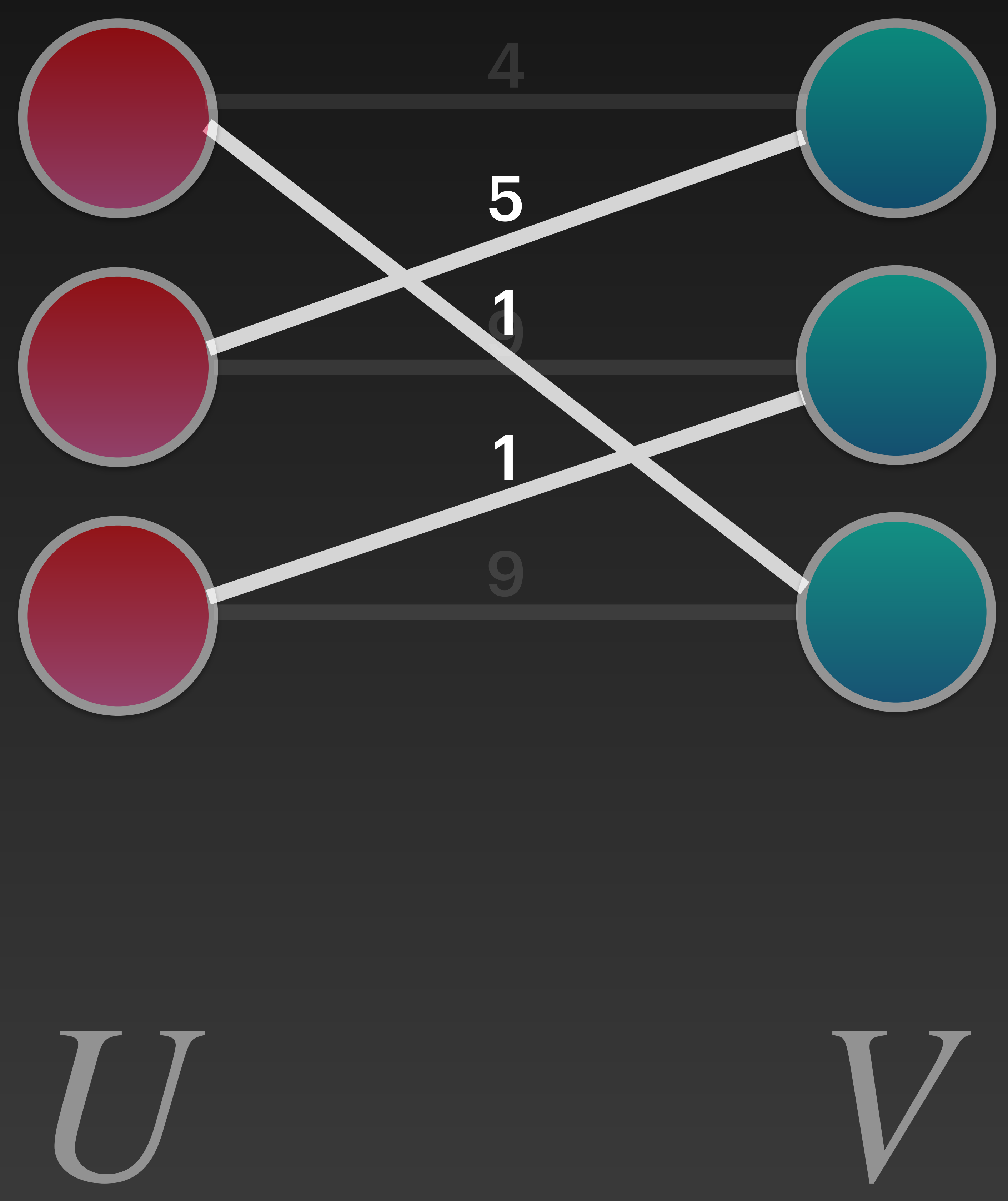


Edge-Weighted Online Bipartite Matching
(E-OBM)



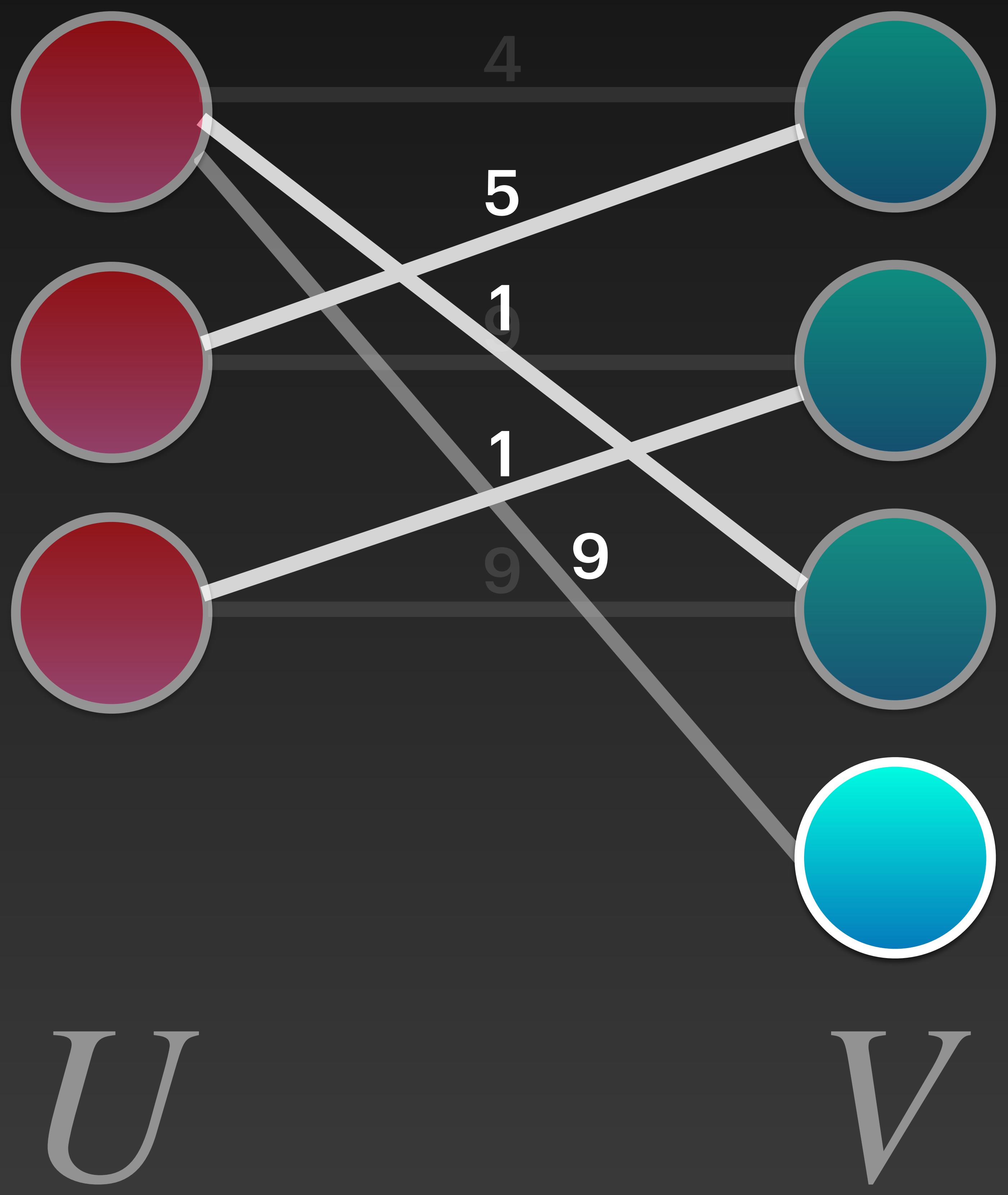
Edge-Weighted Online Bipartite Matching

(E-OBM)



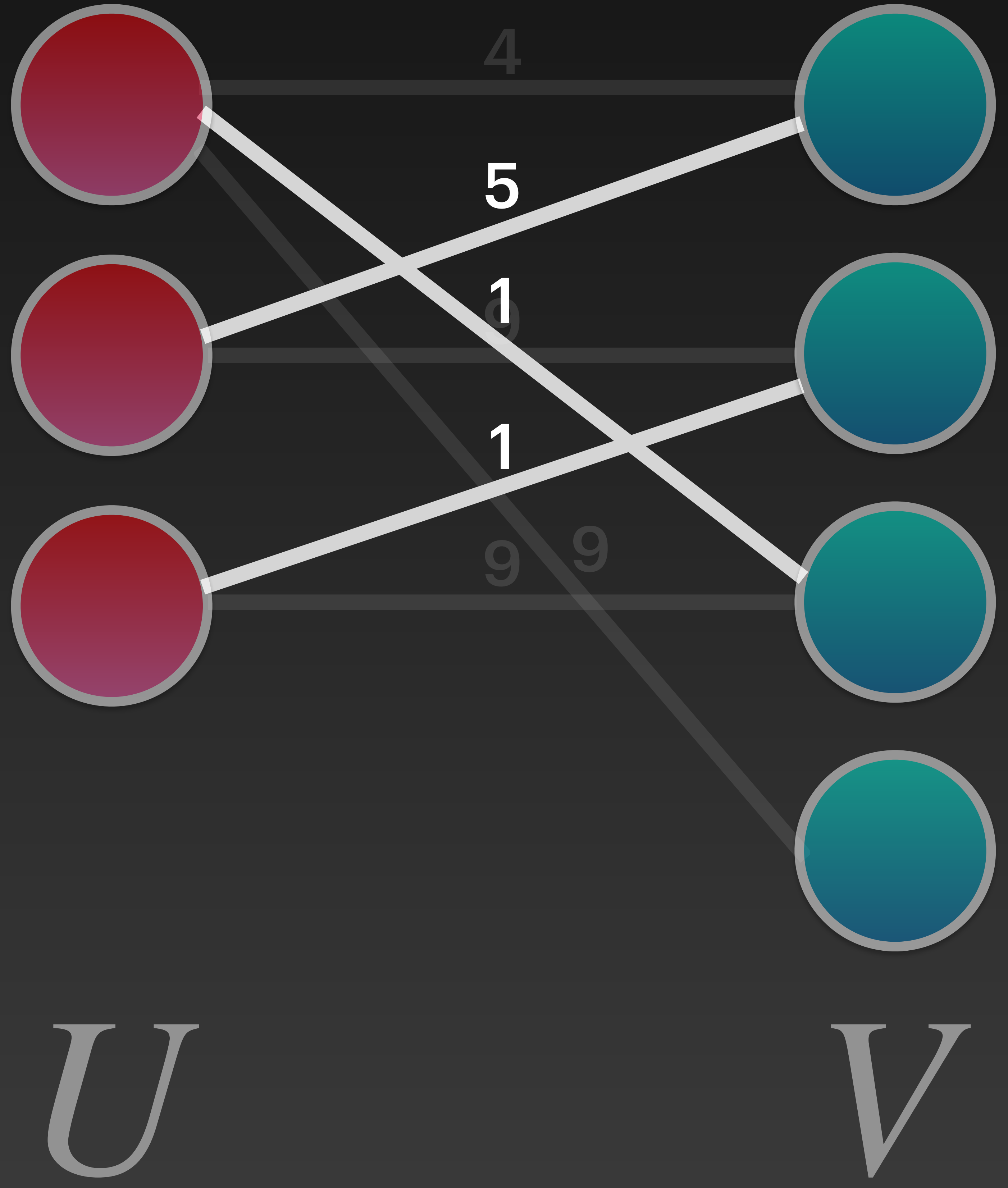
Edge-Weighted Online Bipartite Matching

(E-OBM)



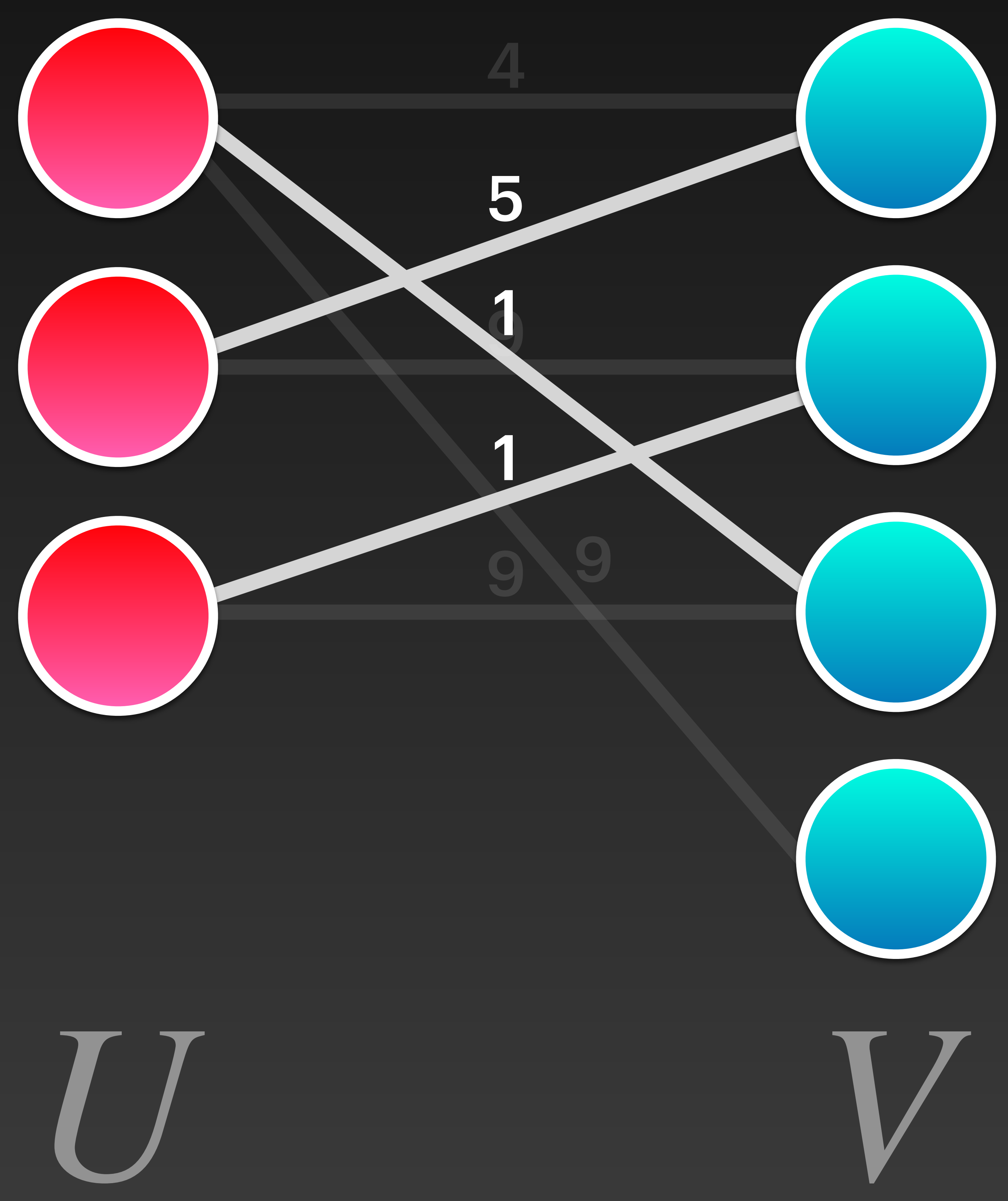
Edge-Weighted Online Bipartite Matching

(E-OBM)



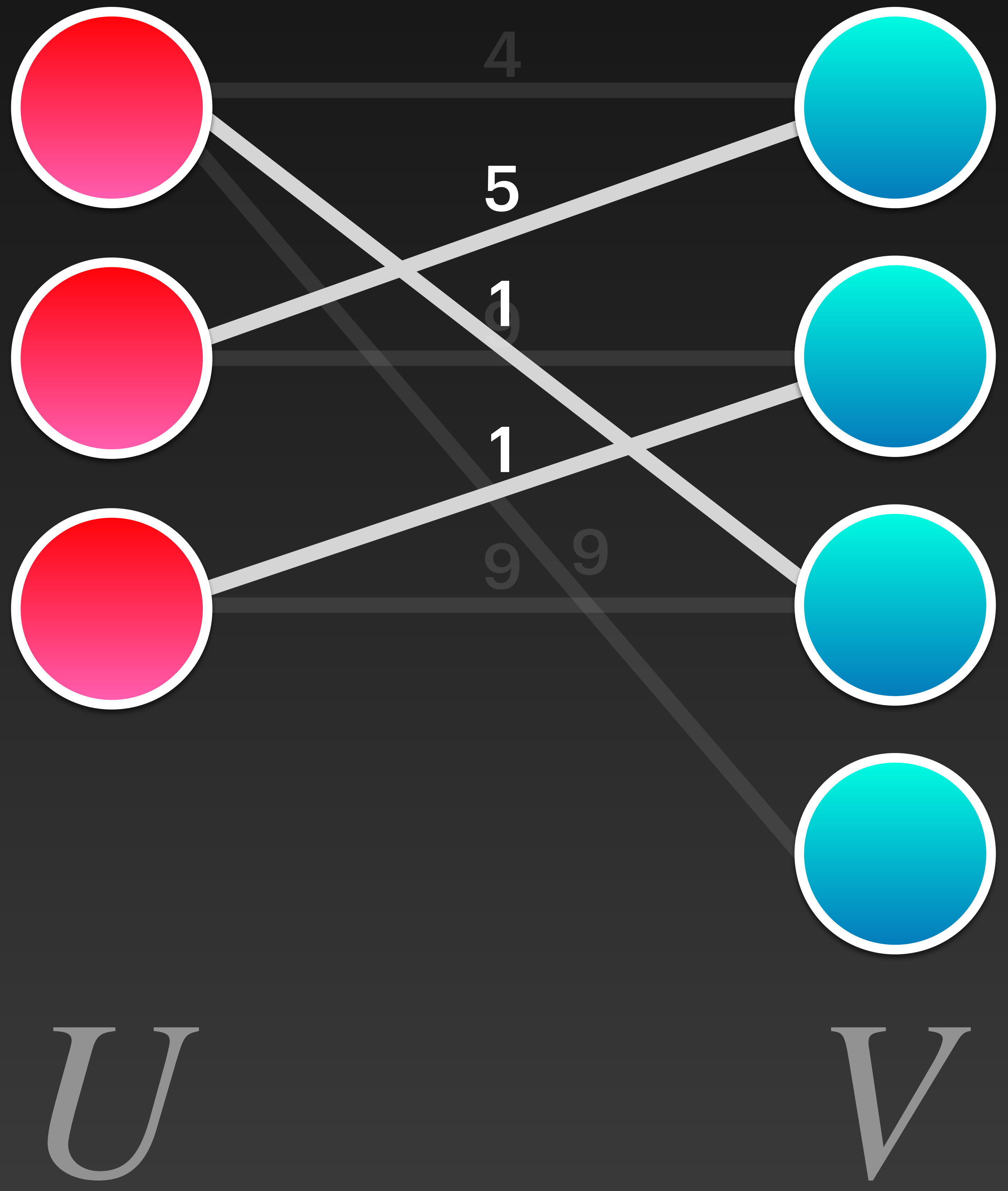
Edge-Weighted Online Bipartite Matching

(E-OBM)

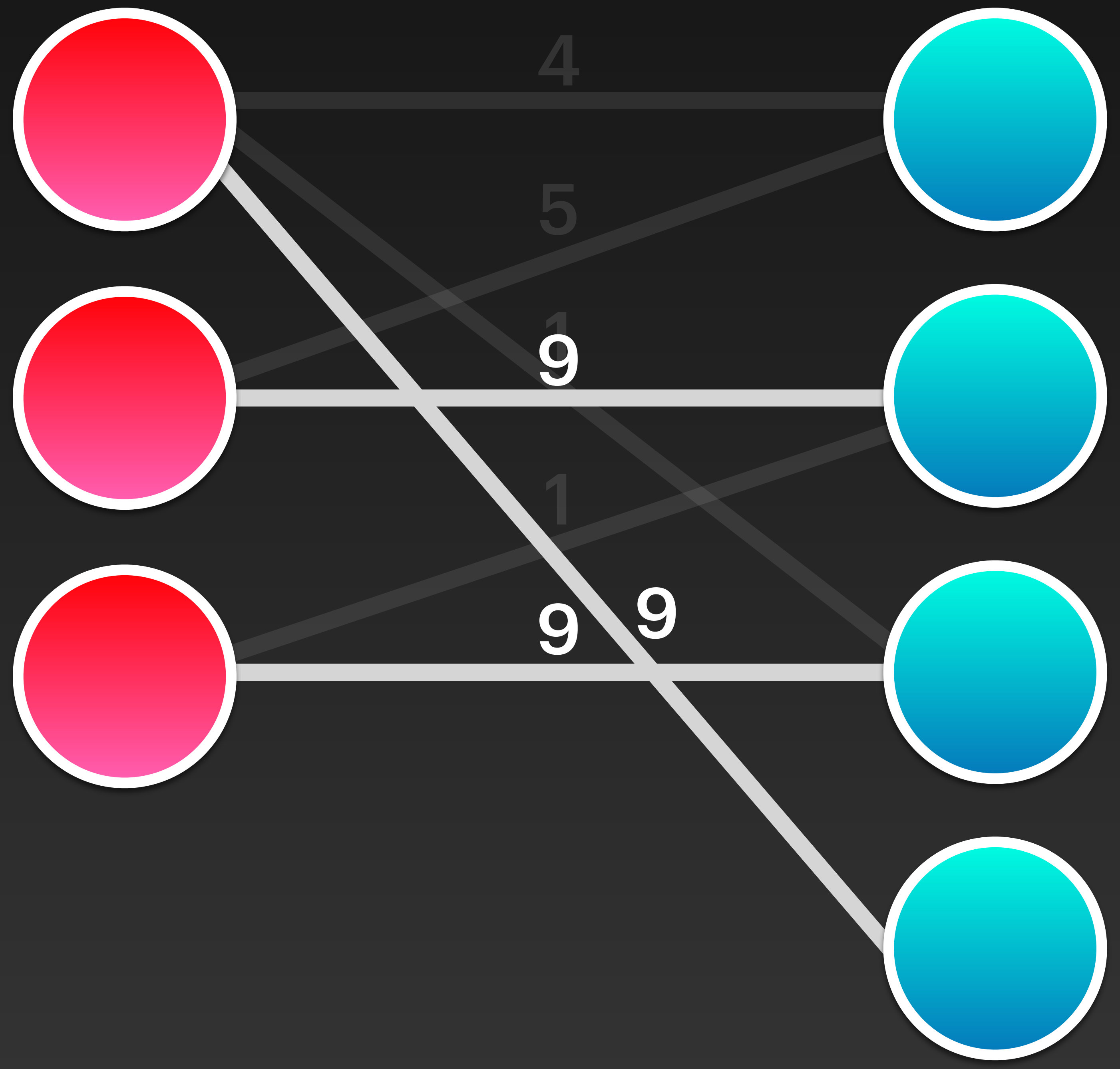


Edge-Weighted Online Bipartite Matching
(E-OBM)

Greedy, Online



Optimum, Offline



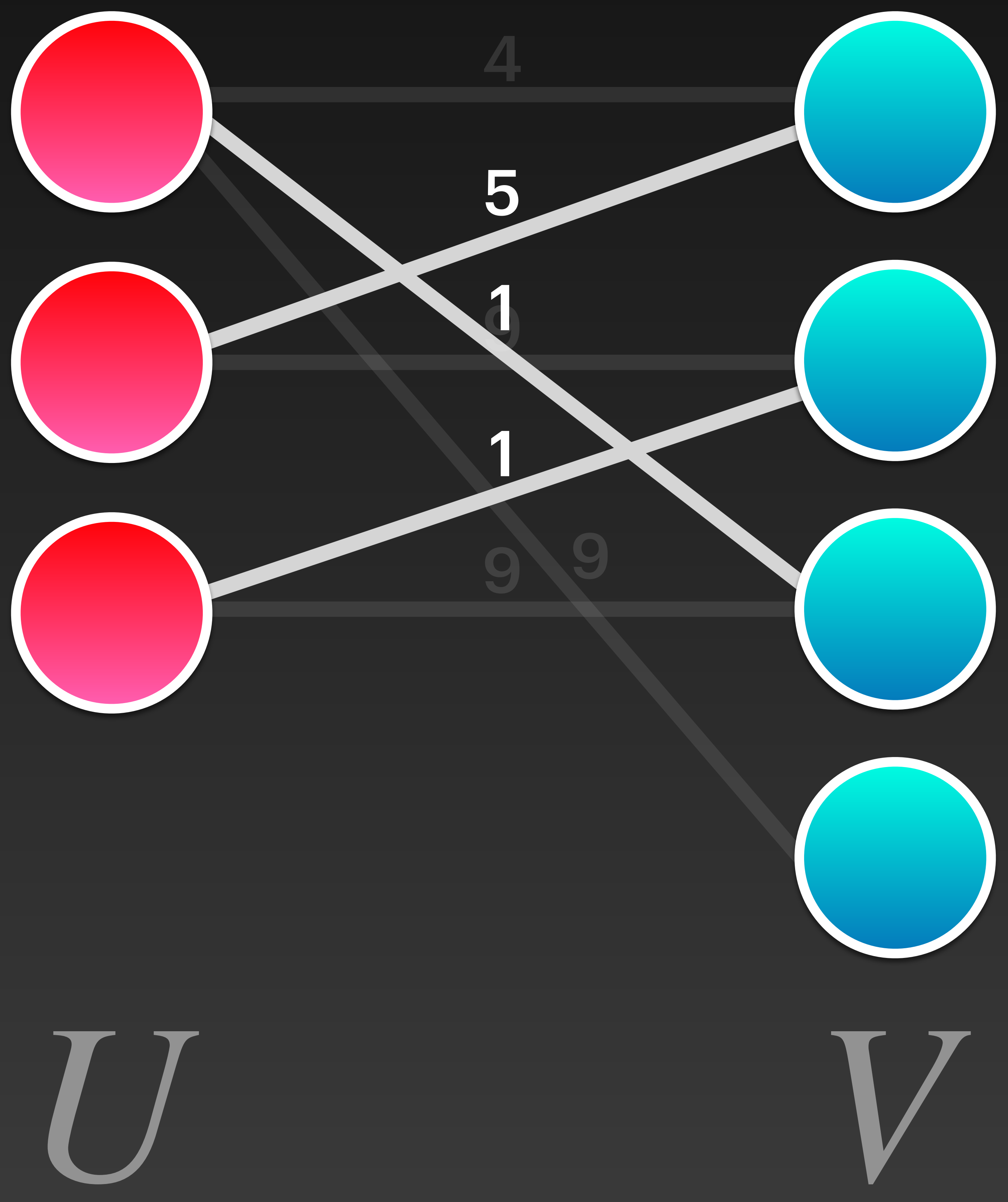
Edge-Weighted Online Bipartite Matching

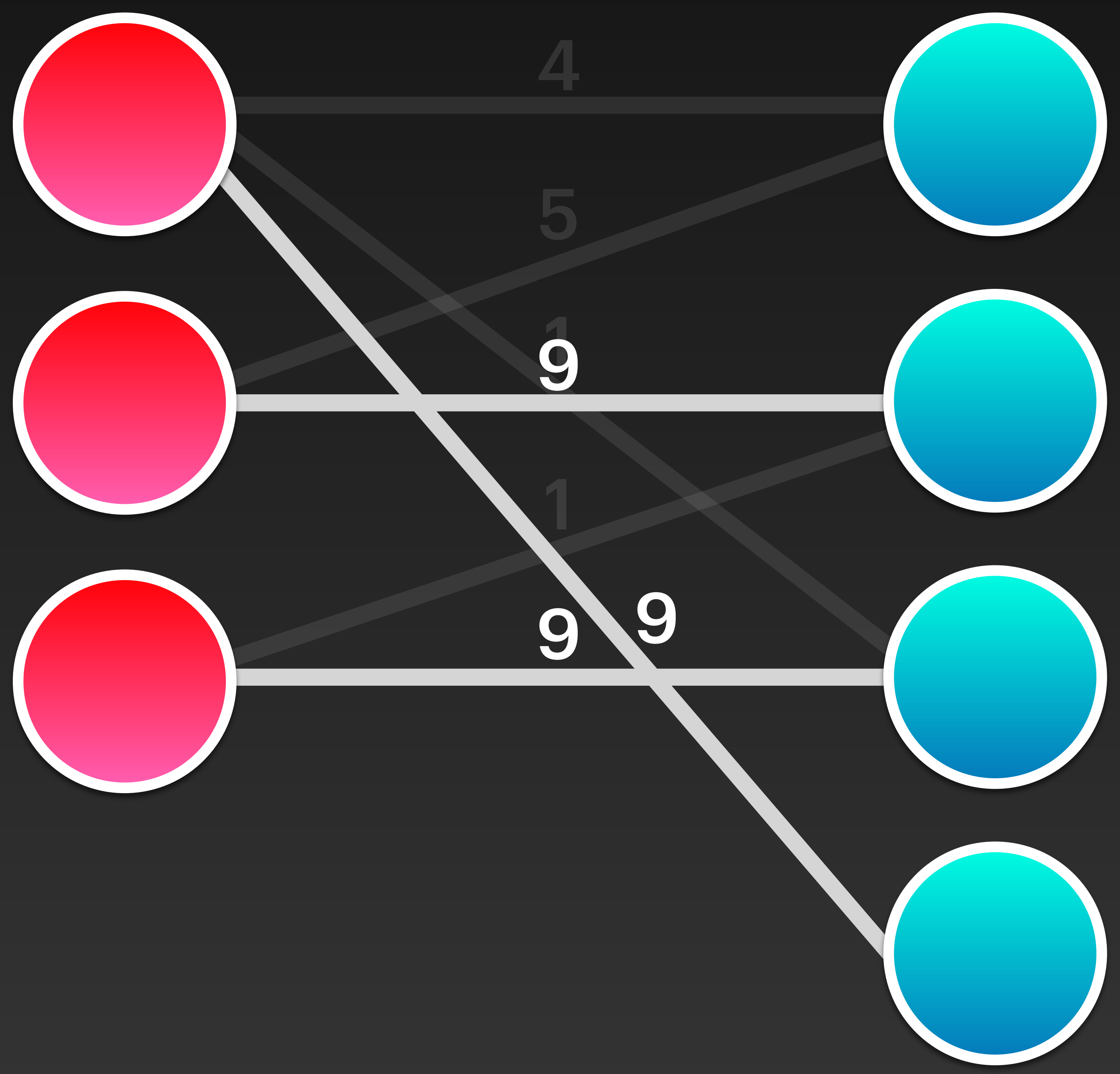
(E-OBM)

Greedy, Online

Optimality Ratio $= \frac{7}{27}$

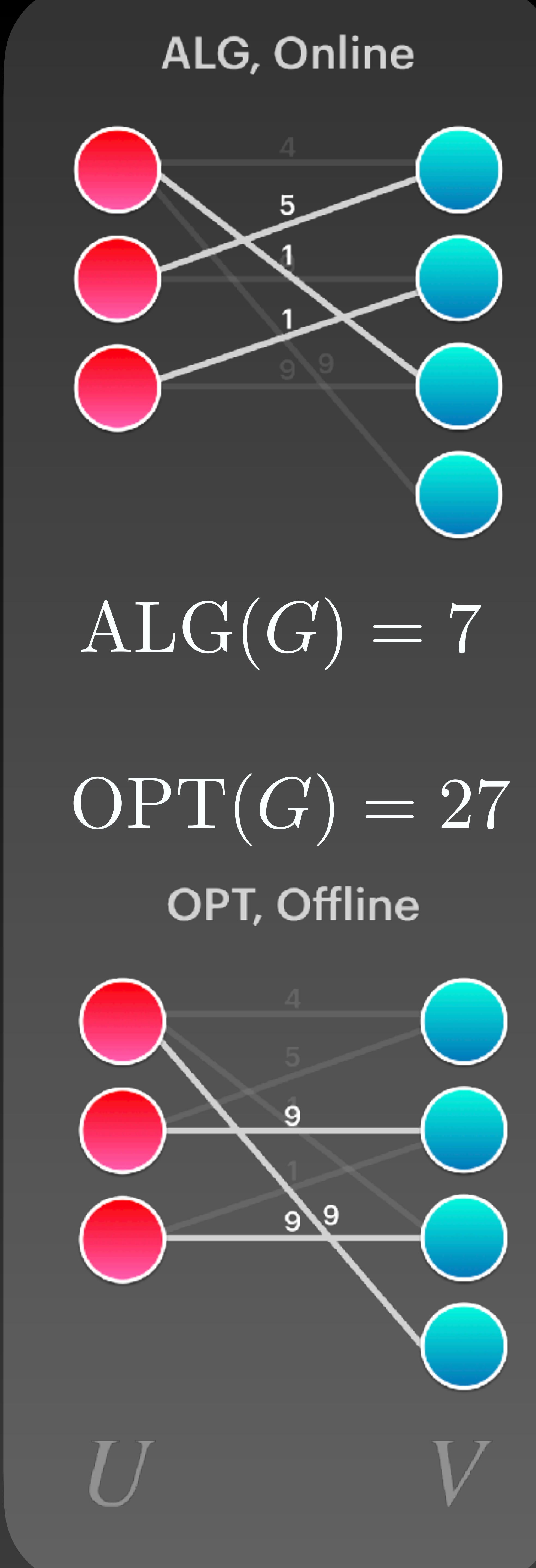
Optimum, Offline





Mehta, Aranyak. "Online Matching and Ad Allocation." Foundations and Trends in Theoretical Computer Science 8.4 (2013): 265-368.

Evaluating Online Algorithms



$\mathcal{D}$: unknown instance-generating distribution

Mehta, Aranyak. "Online Matching and Ad Allocation." Foundations and Trends in Theoretical Computer Science 8.4 (2013): 265-368.

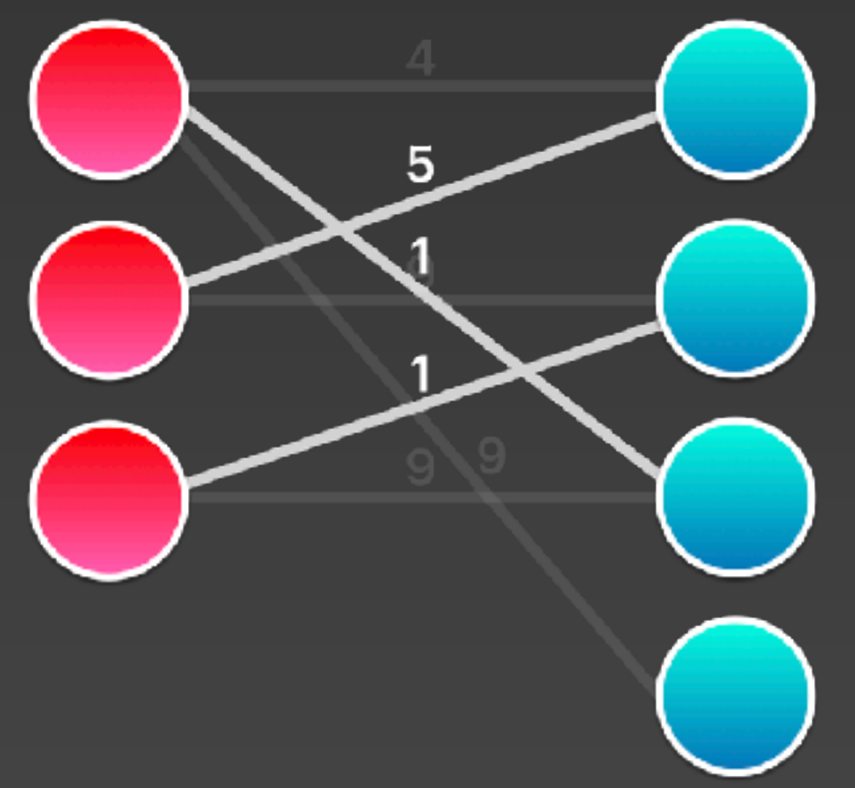
Evaluating Online Algorithms

$$\min_{\substack{G(U,V,E,w)\\ \text{order of } V}} \frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}$$

Competitive Ratio (adversarial)

"Optimality ratio on worst possible instance"

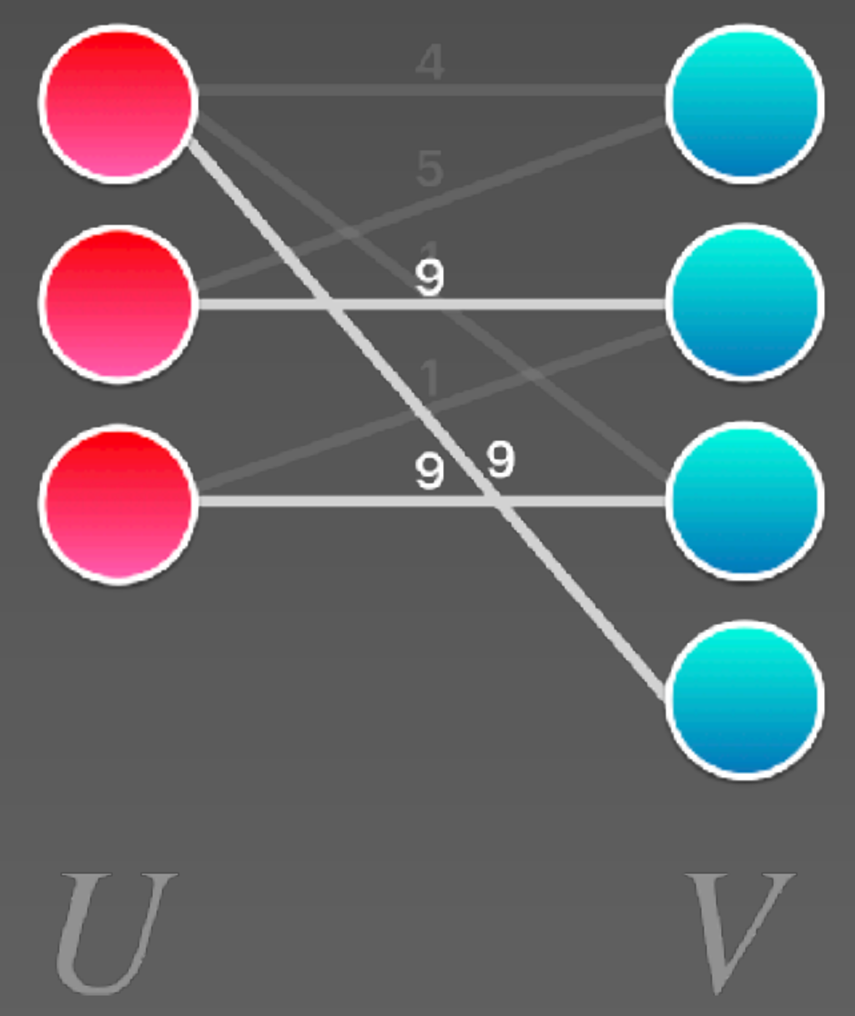
ALG, Online



$$\mathrm{ALG}(G) = 7$$

$$\mathrm{OPT}(G) = 27$$

OPT, Offline



$\mathcal{D}$: unknown instance-generating distribution

Mehta, Aranyak. "Online Matching and Ad Allocation." Foundations and Trends in Theoretical Computer Science 8.4 (2013): 265-368.

Evaluating Online Algorithms

$$\min_{\substack{G(U,V,E,w)\\ \text{order of } V}} \frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}$$

Competitive Ratio (adversarial)

"Optimality ratio on worst possible instance"

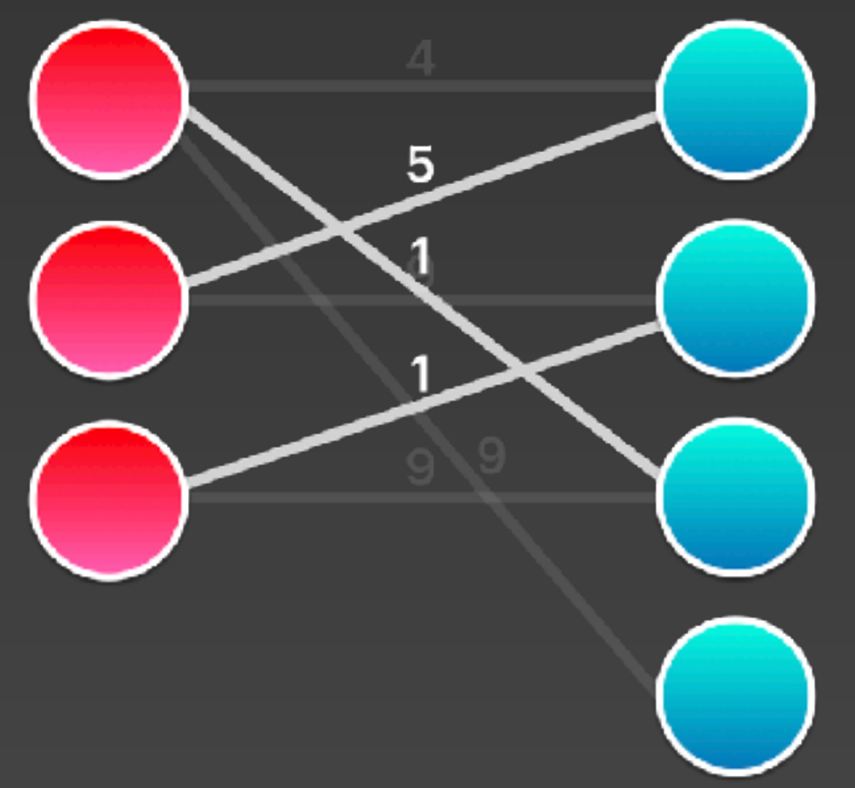
$$\min_{\mathcal{D}} \mathop{\mathbb{E}}_{(G,\ \text{order of } V)\sim\mathcal{D}} \left[\frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}\right]$$

Competitive Ratio (stochastic i.i.d.)

"Expected optimality ratio on worst possible generating distribution"

$\mathcal{D}$: unknown instance-generating distribution

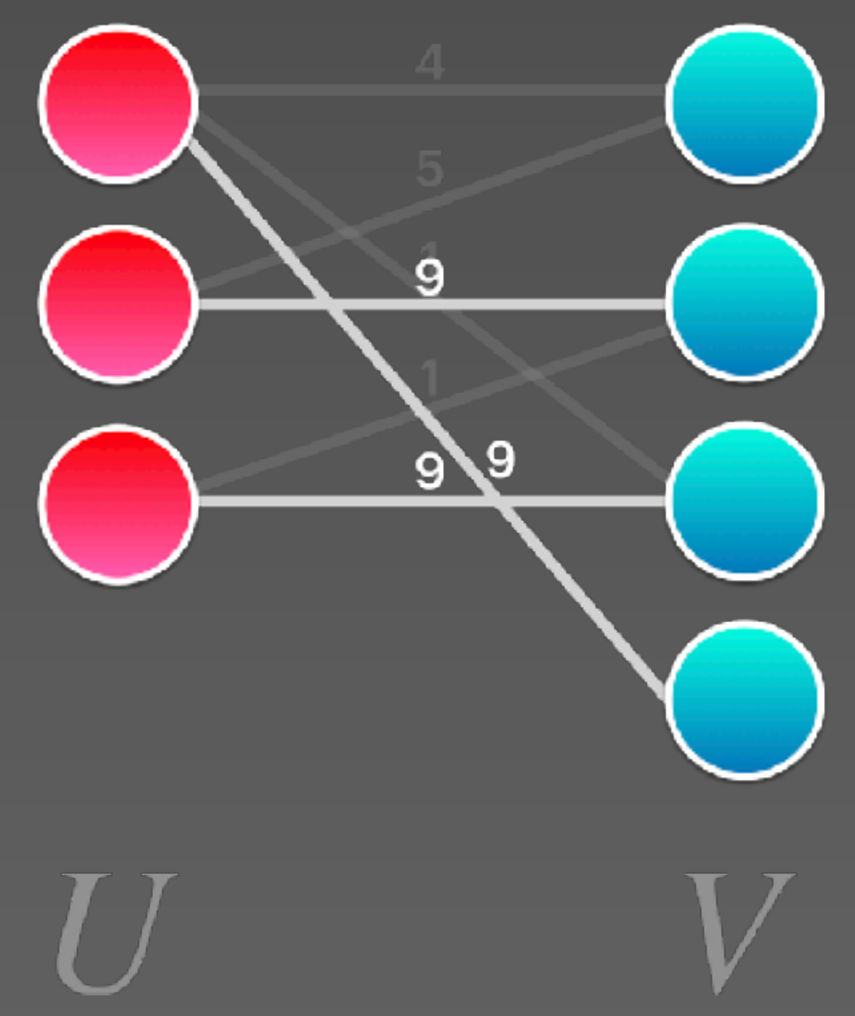
ALG, Online



$\mathrm{ALG}(G) = 7$

$\mathrm{OPT}(G) = 27$

OPT, Offline



Mehta, Aranyak. "Online Matching and Ad Allocation." Foundations and Trends in Theoretical Computer Science 8.4 (2013): 265-368.

Evaluating Online Algorithms

$$\min_{\substack{G(U,V,E,w)\\ \text{order of } V}} \frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}$$

Competitive Ratio (adversarial)

"Optimality ratio on worst possible instance"

$$\min_{\mathcal{D}} \mathop{\mathbb{E}}_{(G,\ \text{order of } V)\sim\mathcal{D}} \left[\frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}\right]$$

Competitive Ratio (stochastic i.i.d.)

"Expected optimality ratio on worst possible generating distribution"

$$\mathop{\mathbb{E}}_{(G,\ \text{order of } V)\sim\mathcal{D}} \left[\frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}\right]$$

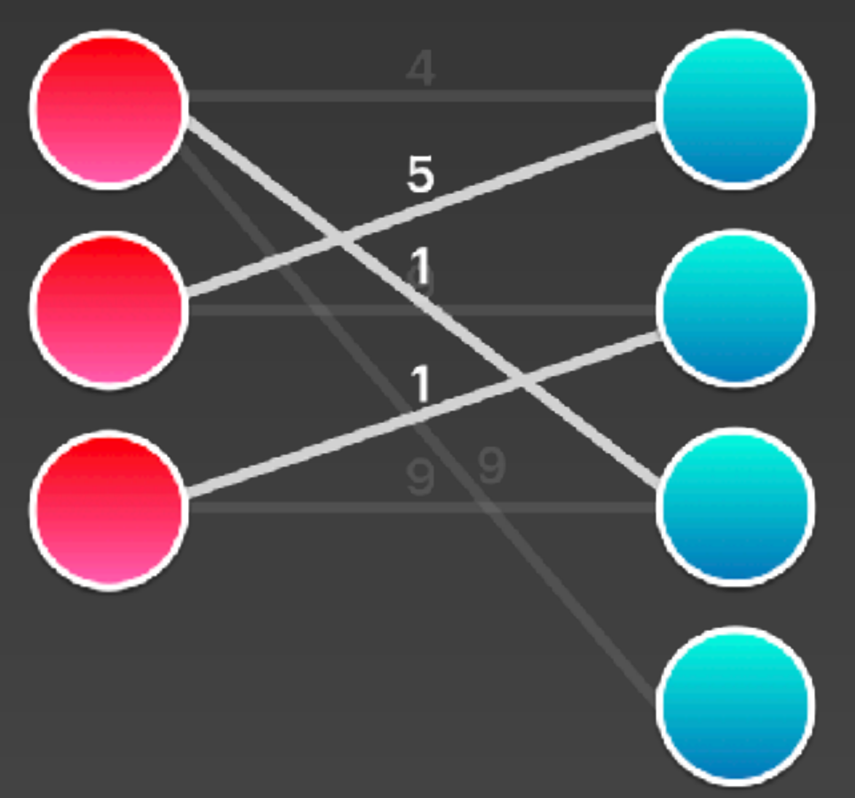
Expected Ratio* (stochastic i.i.d.)

"Expected optimality ratio on a given real dataset or synthetic generating distribution"

$\mathcal{D}$: unknown instance-generating distribution

* Garg, Naveen, et al. "Stochastic analyses for online combinatorial optimization problems." SODA. 2008.

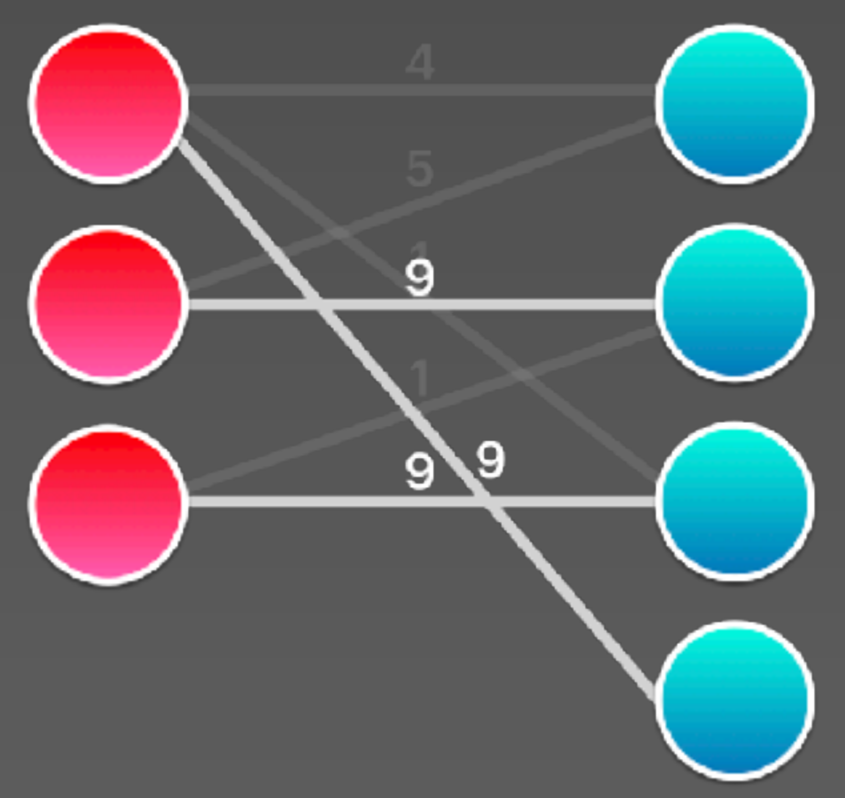
ALG, Online



$\mathrm{ALG}(G) = 7$

$\mathrm{OPT}(G) = 27$

OPT, Offline



U V

Potential for ML

Online vs Offline optimization

The following make ML very suitable for Online Combinatorial Optimization:

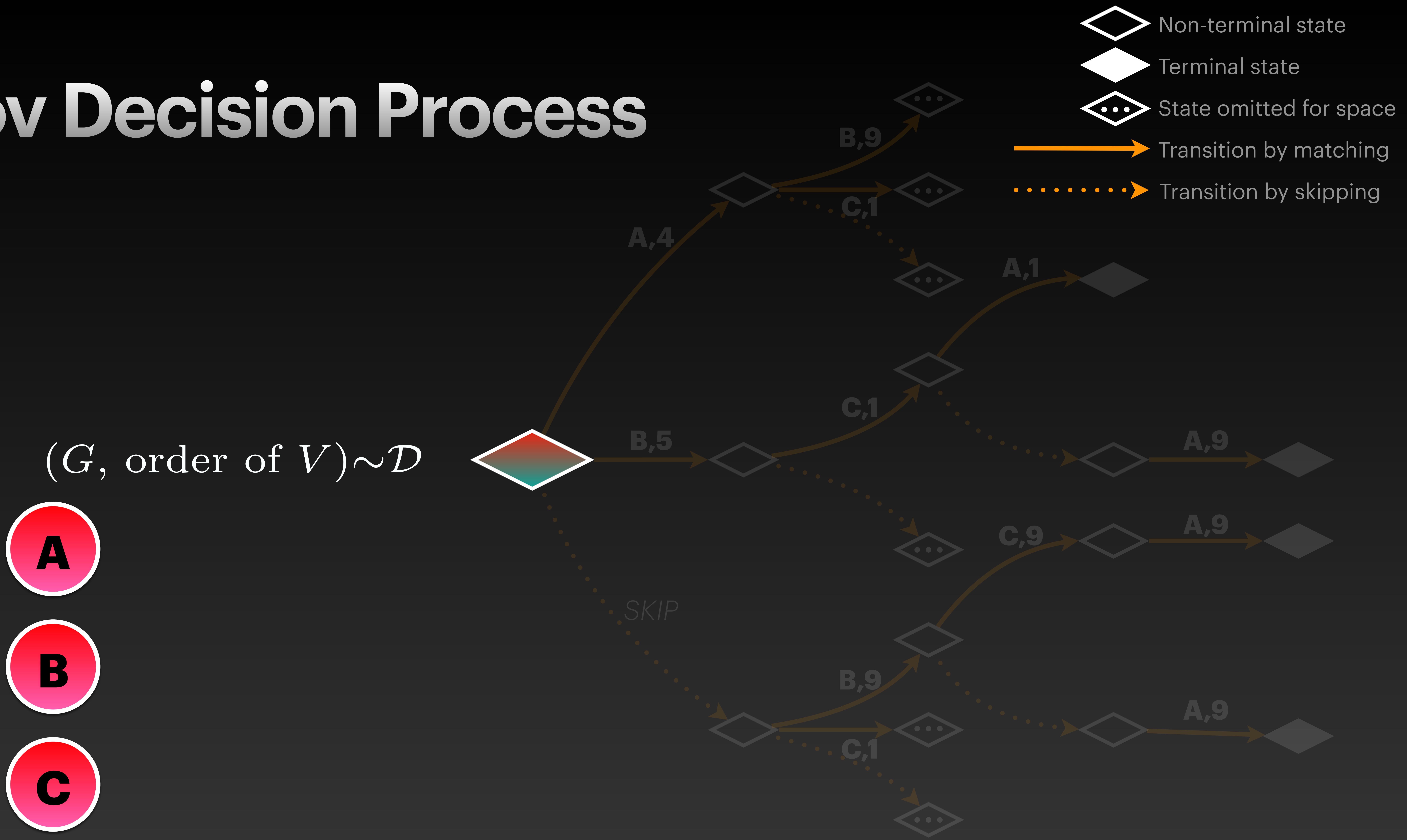
✓ **"Data" more likely to exist** since online means quick/repeated tasks

✓ **Approximation require lots of assumptions** for online problems!

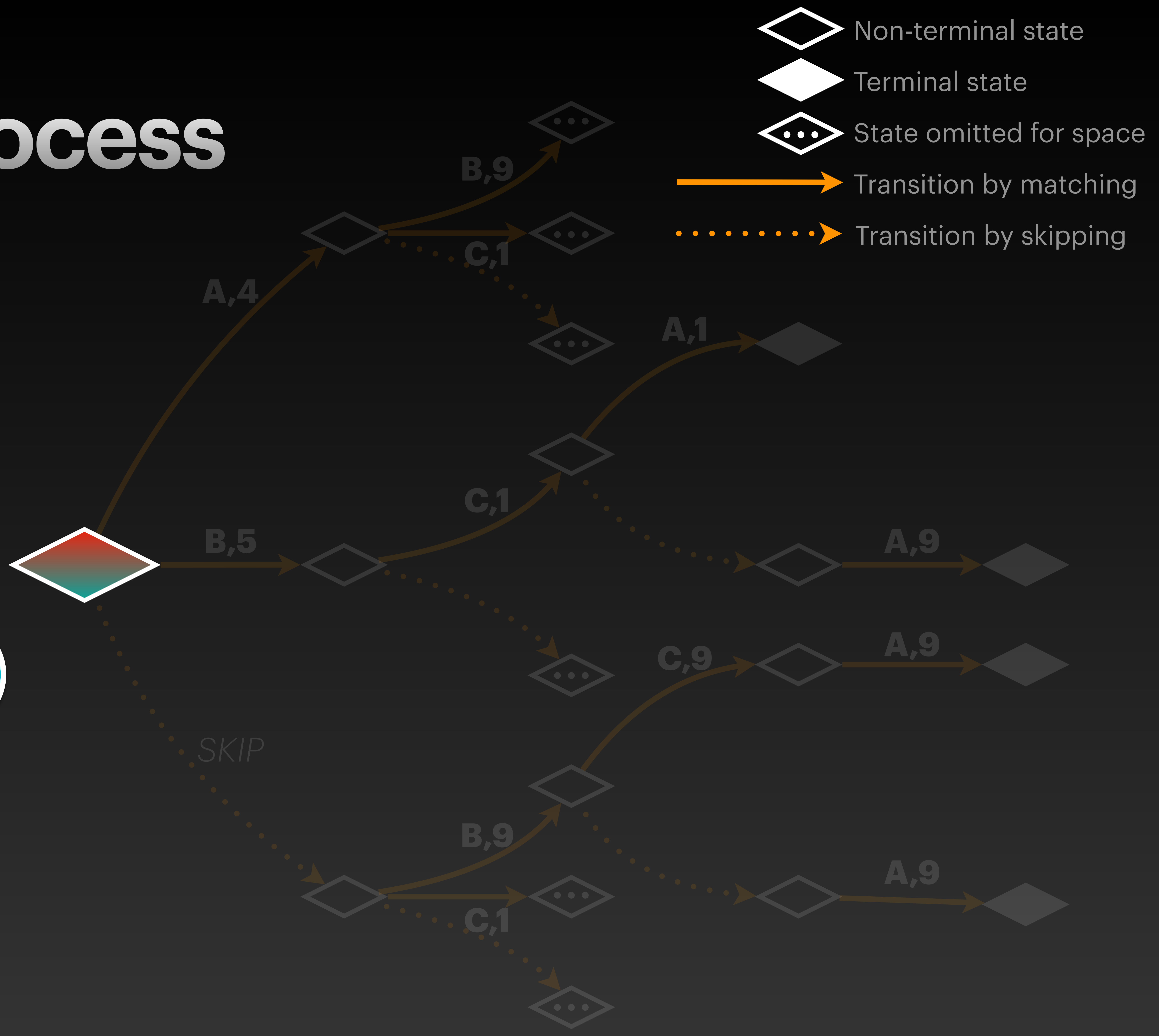
✓ Online optimization **fits nicely with Reinforcement Learning**

Markov Decision Process

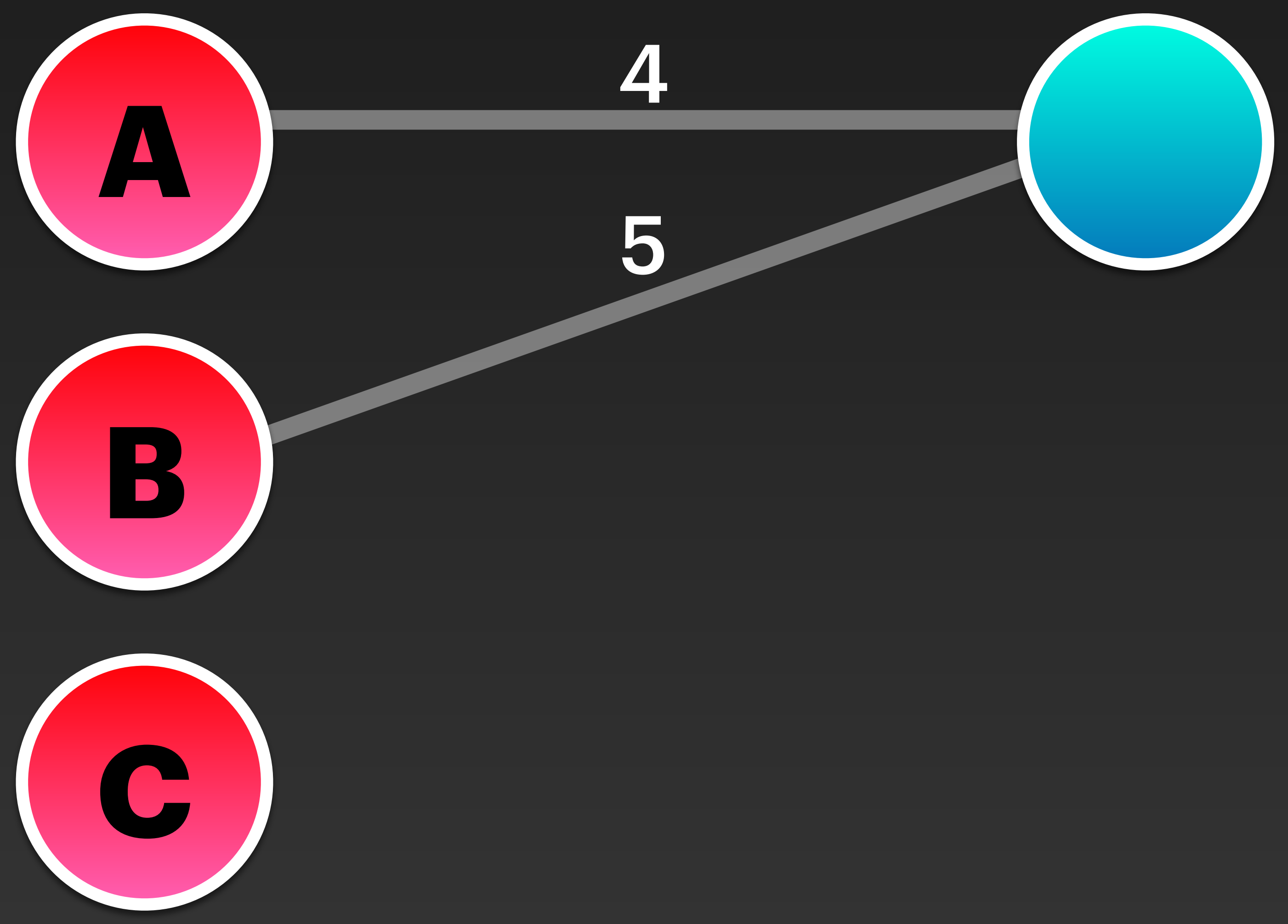
$(G, \text{order of } V) \sim \mathcal{D}$



Markov Decision Process

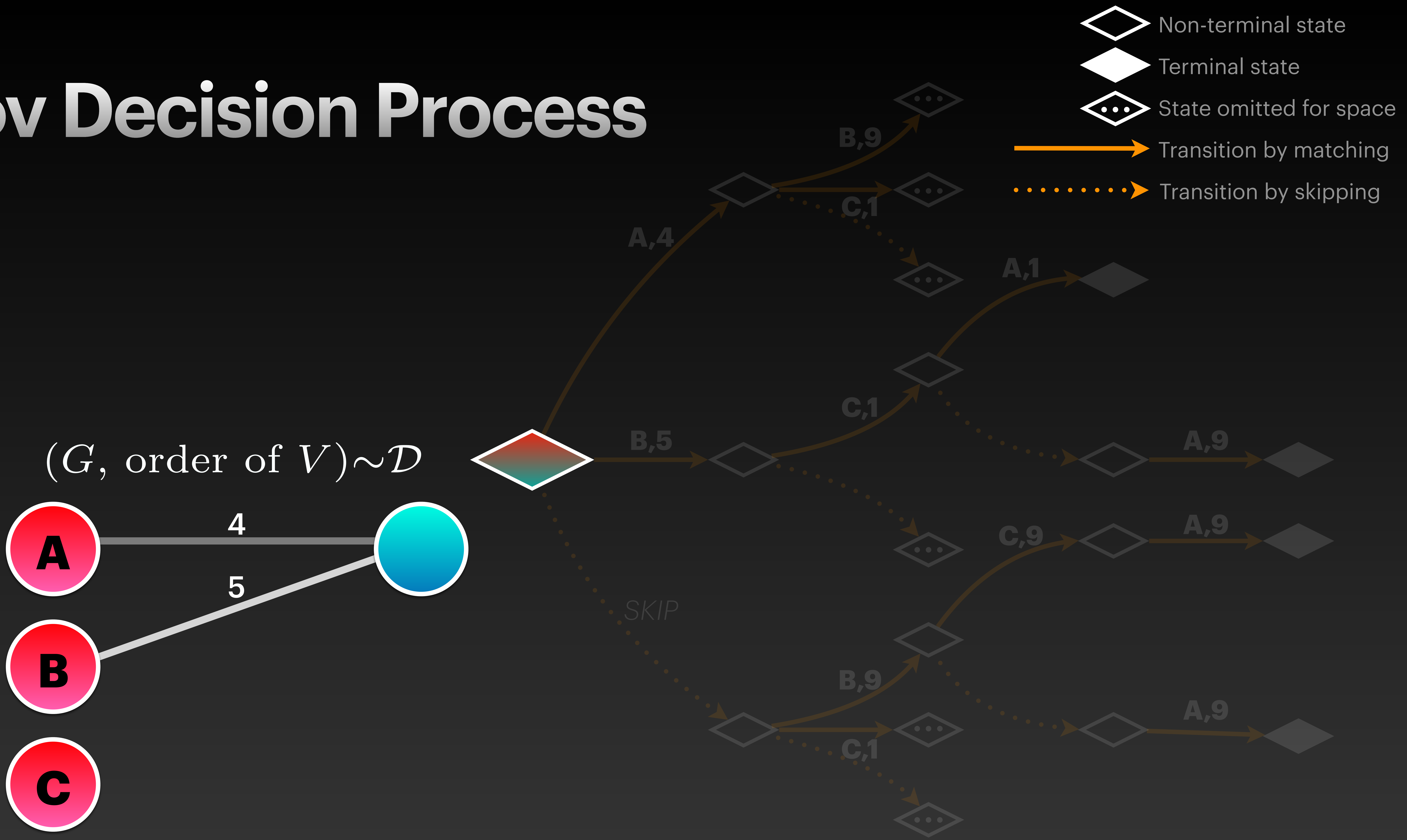


$(G, \text{ order of } V) \sim \mathcal{D}$

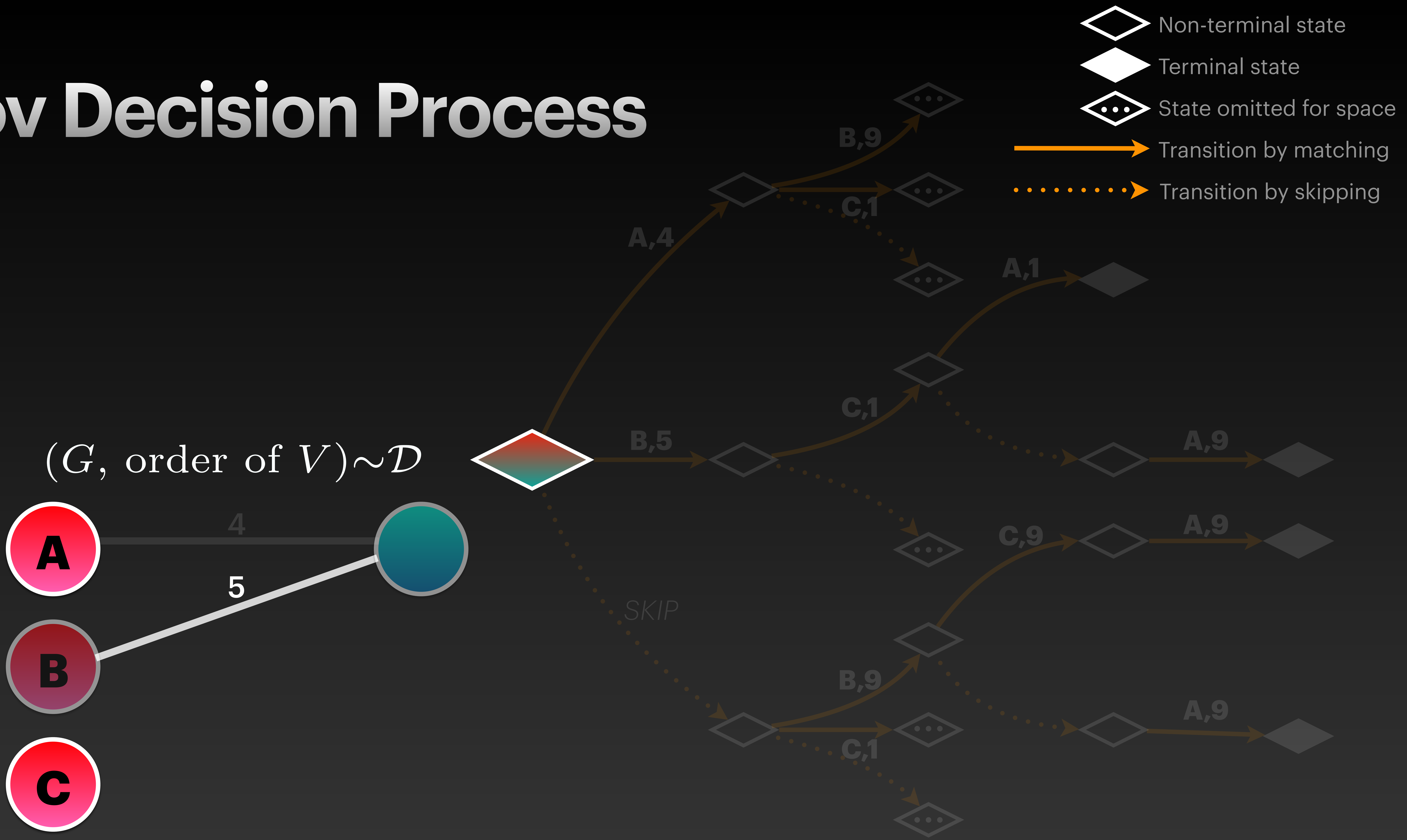


Markov Decision Process

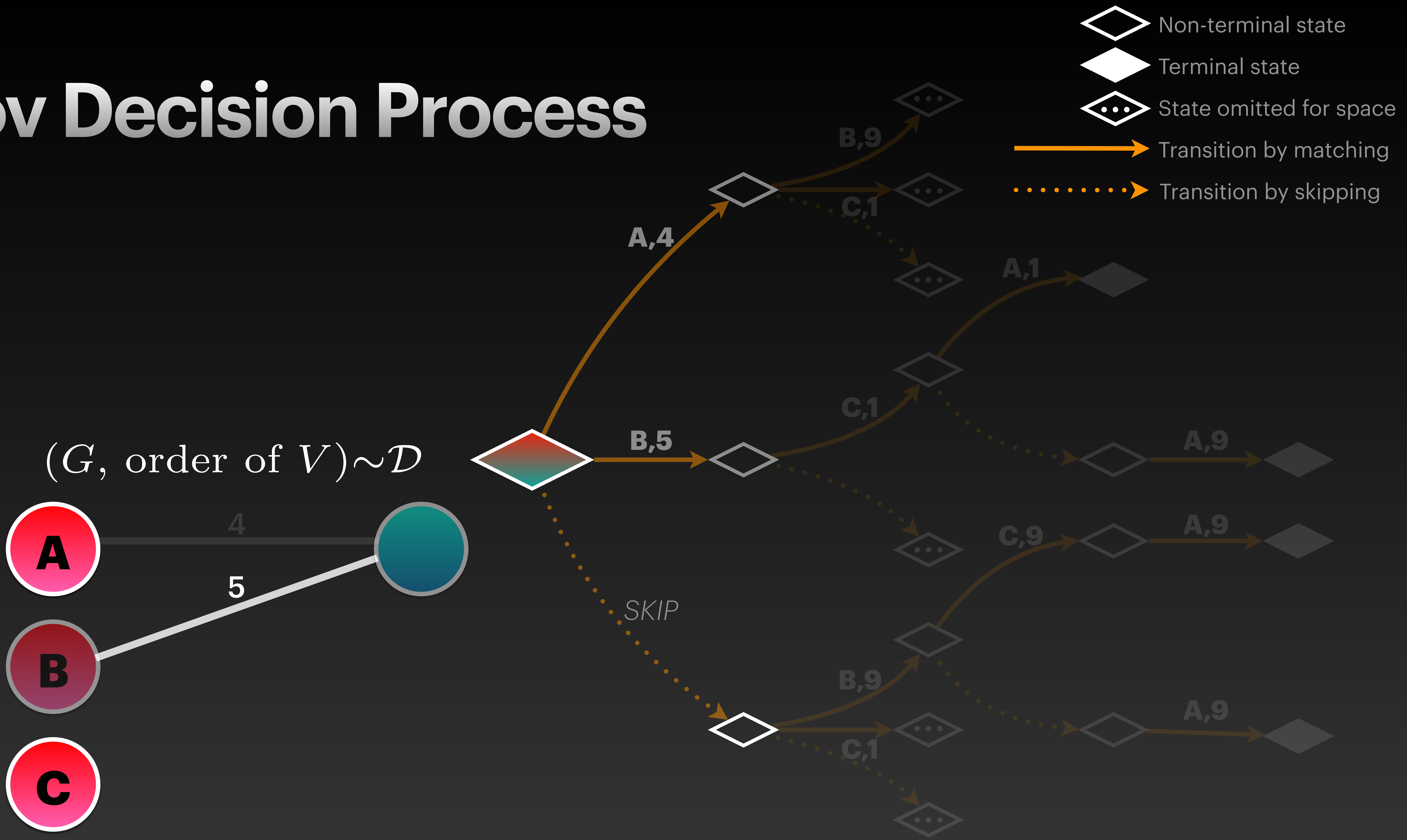
$(G, \text{ order of } V) \sim \mathcal{D}$



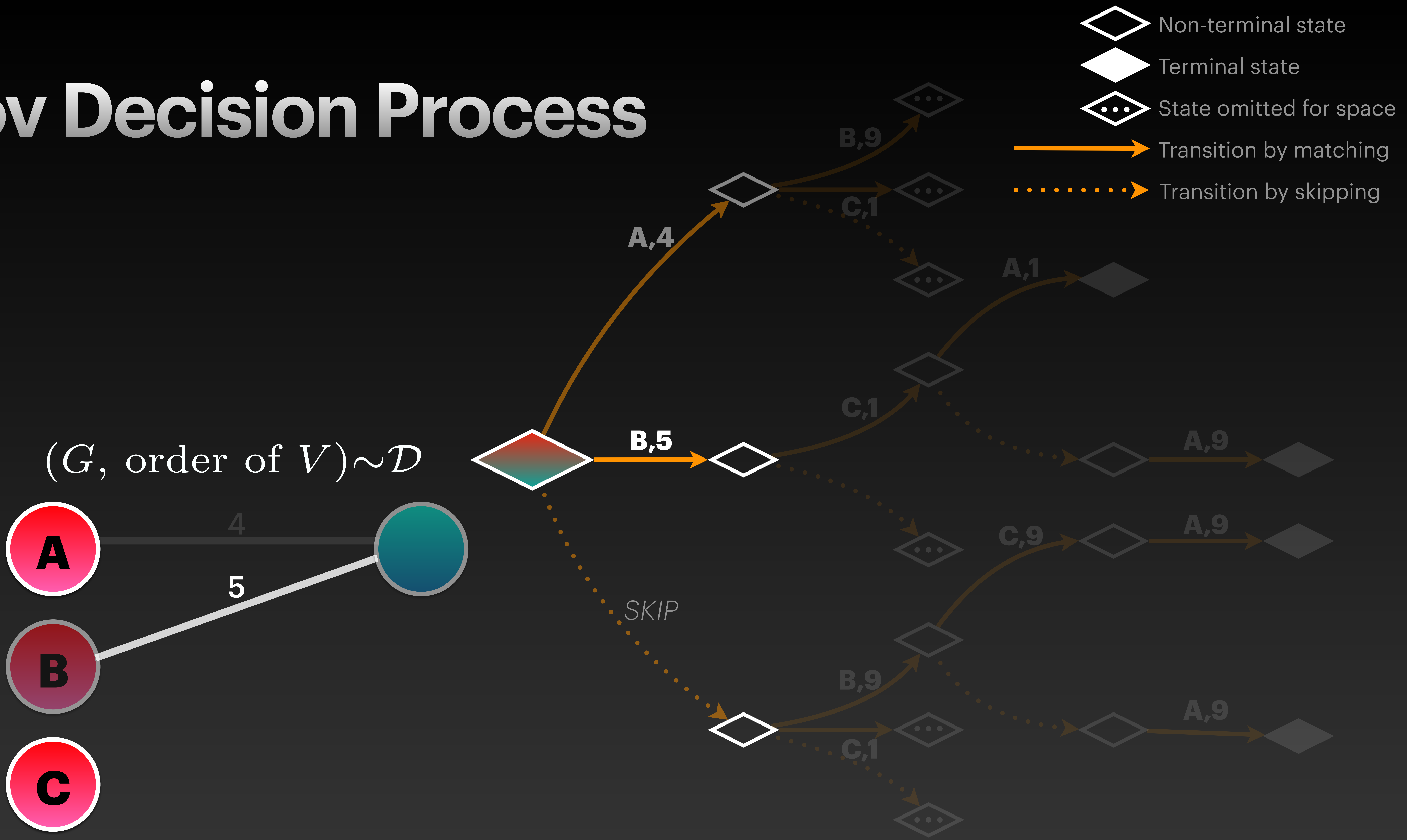
Markov Decision Process



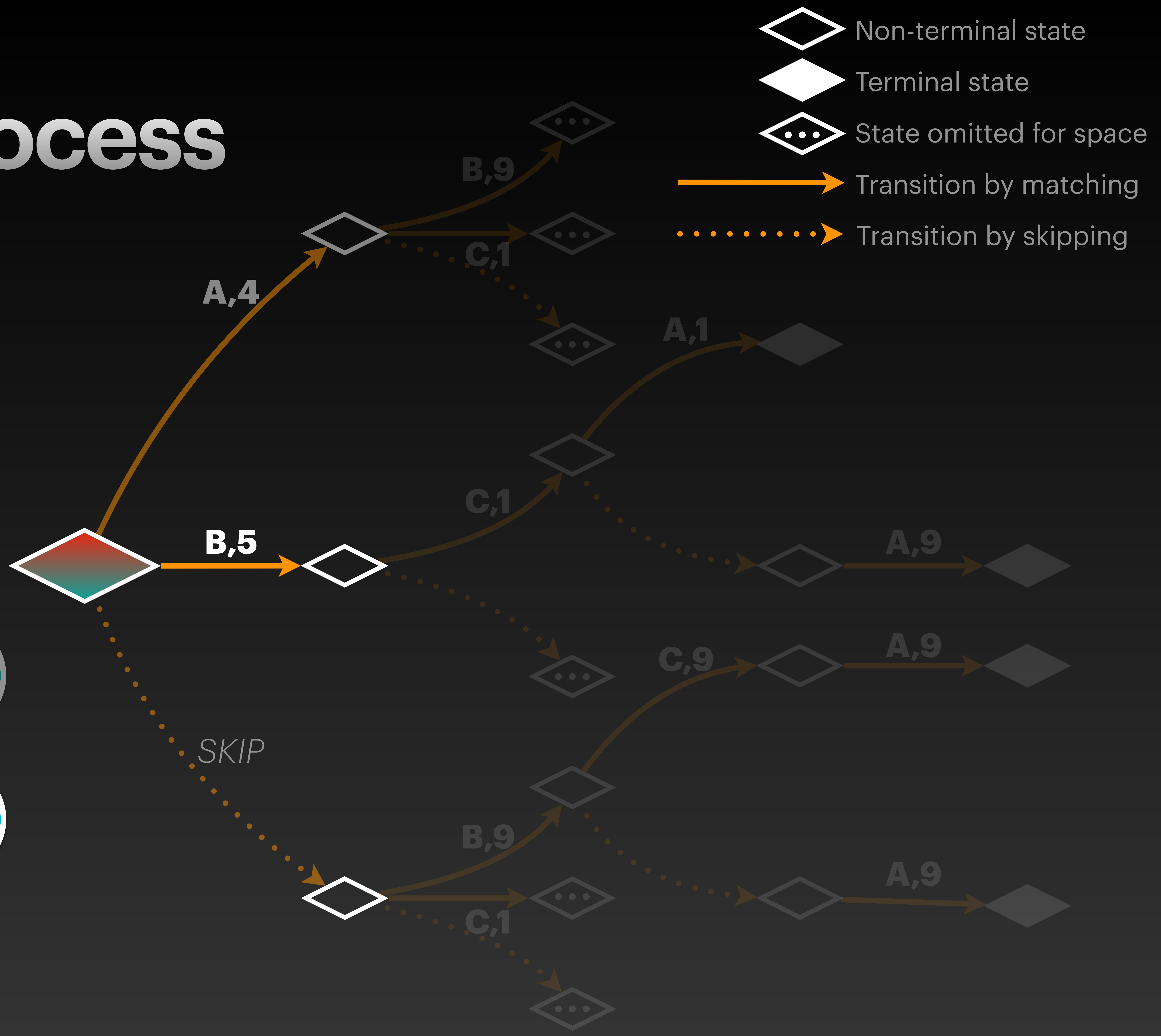
Markov Decision Process



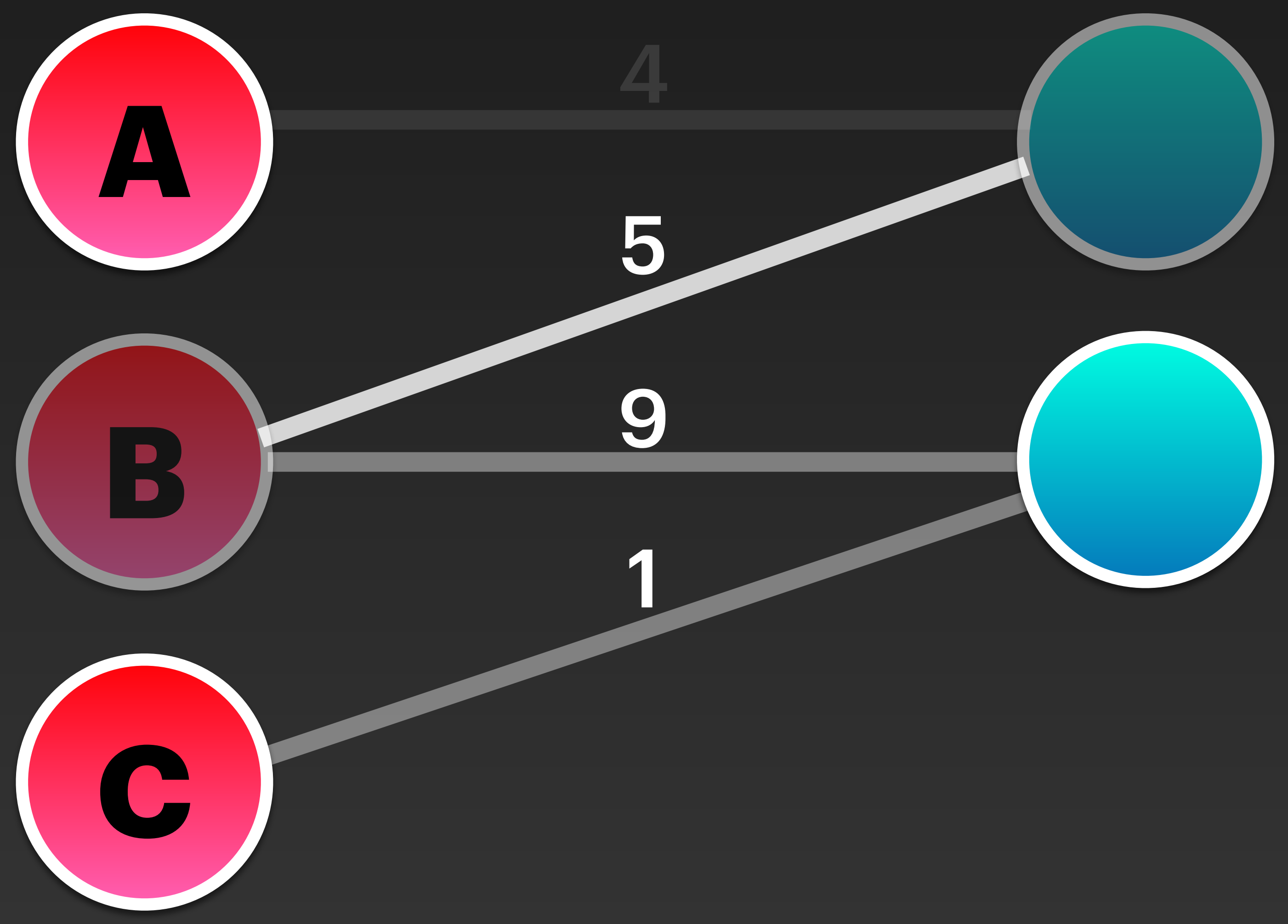
Markov Decision Process



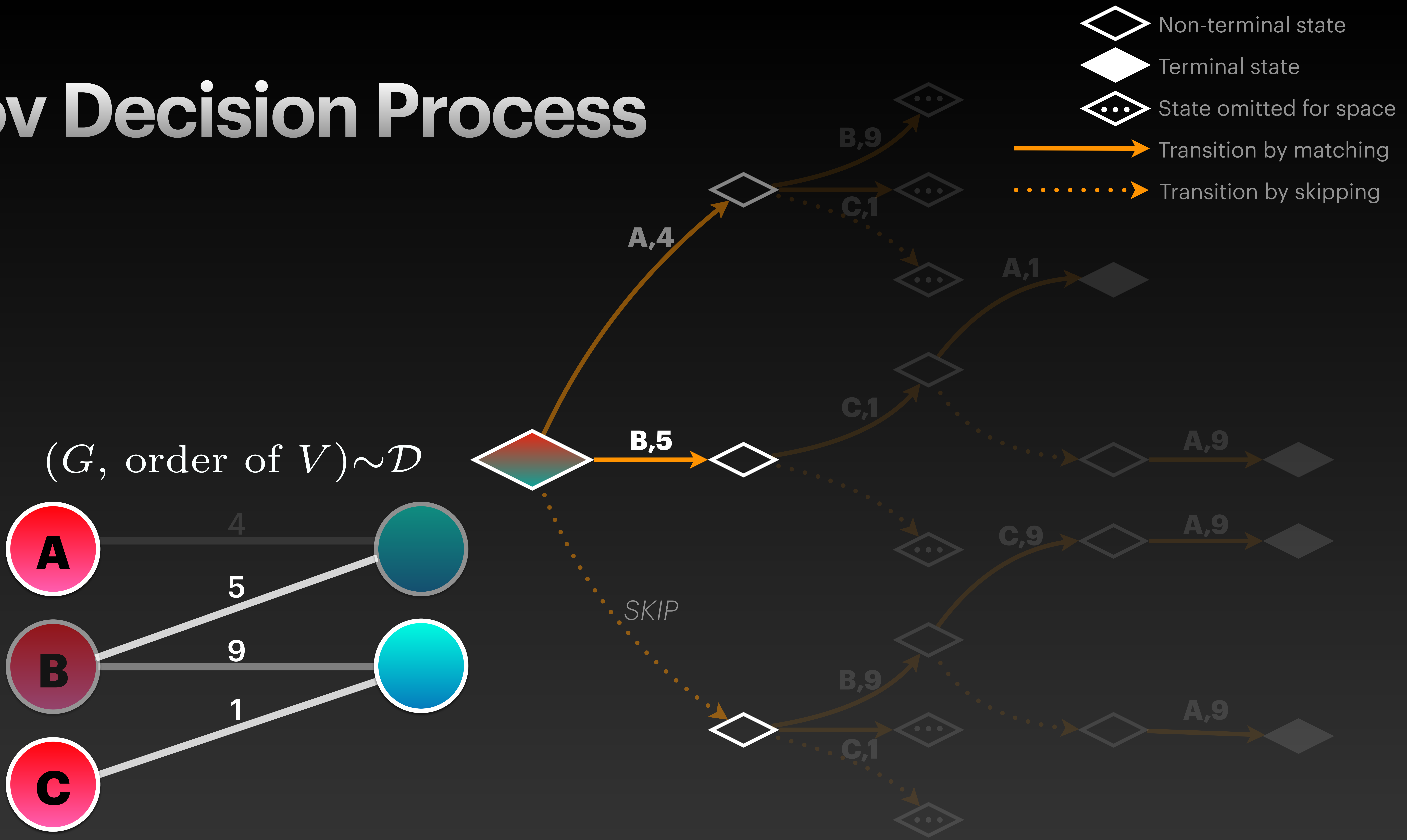
Markov Decision Process



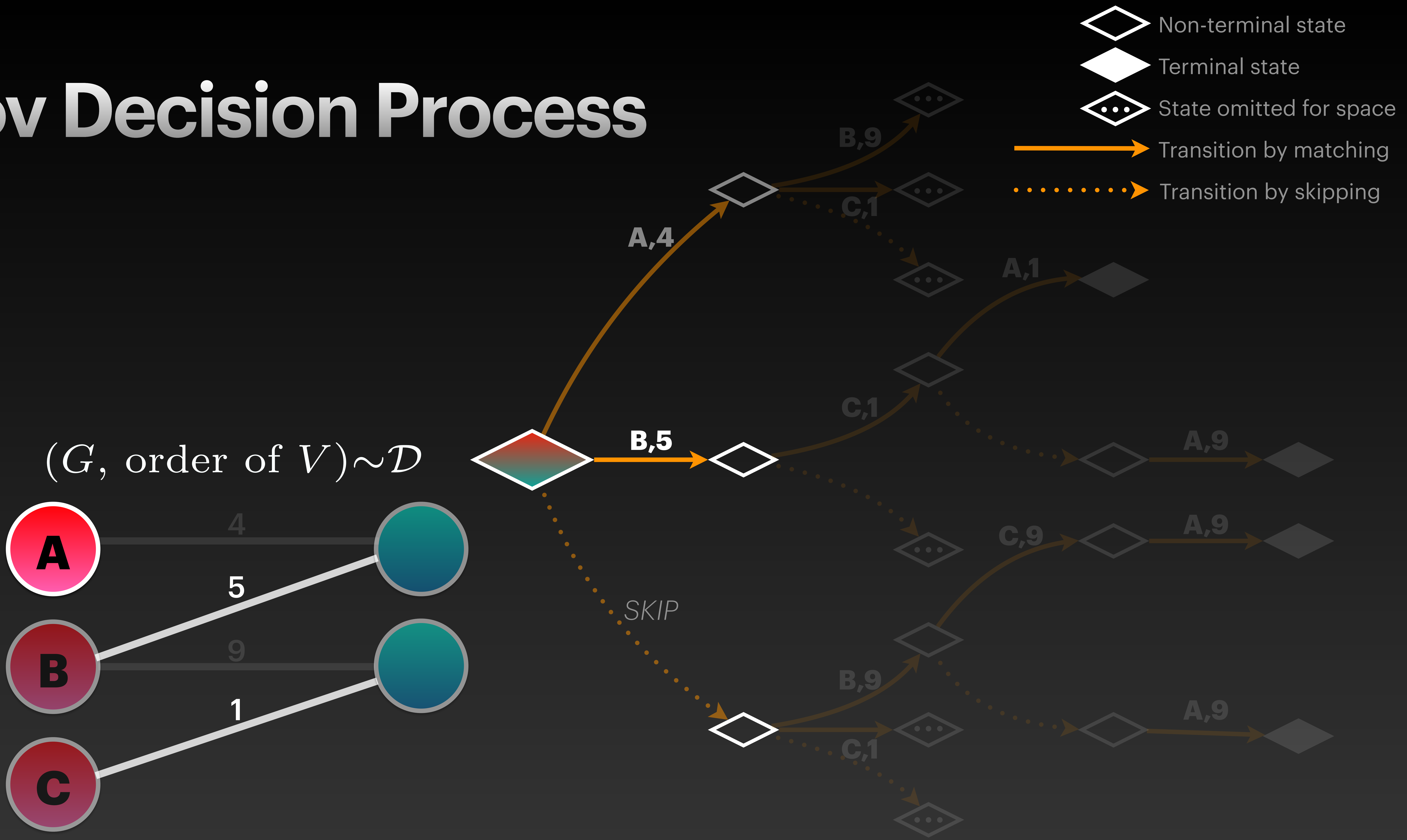
$(G, \text{ order of } V) \sim \mathcal{D}$



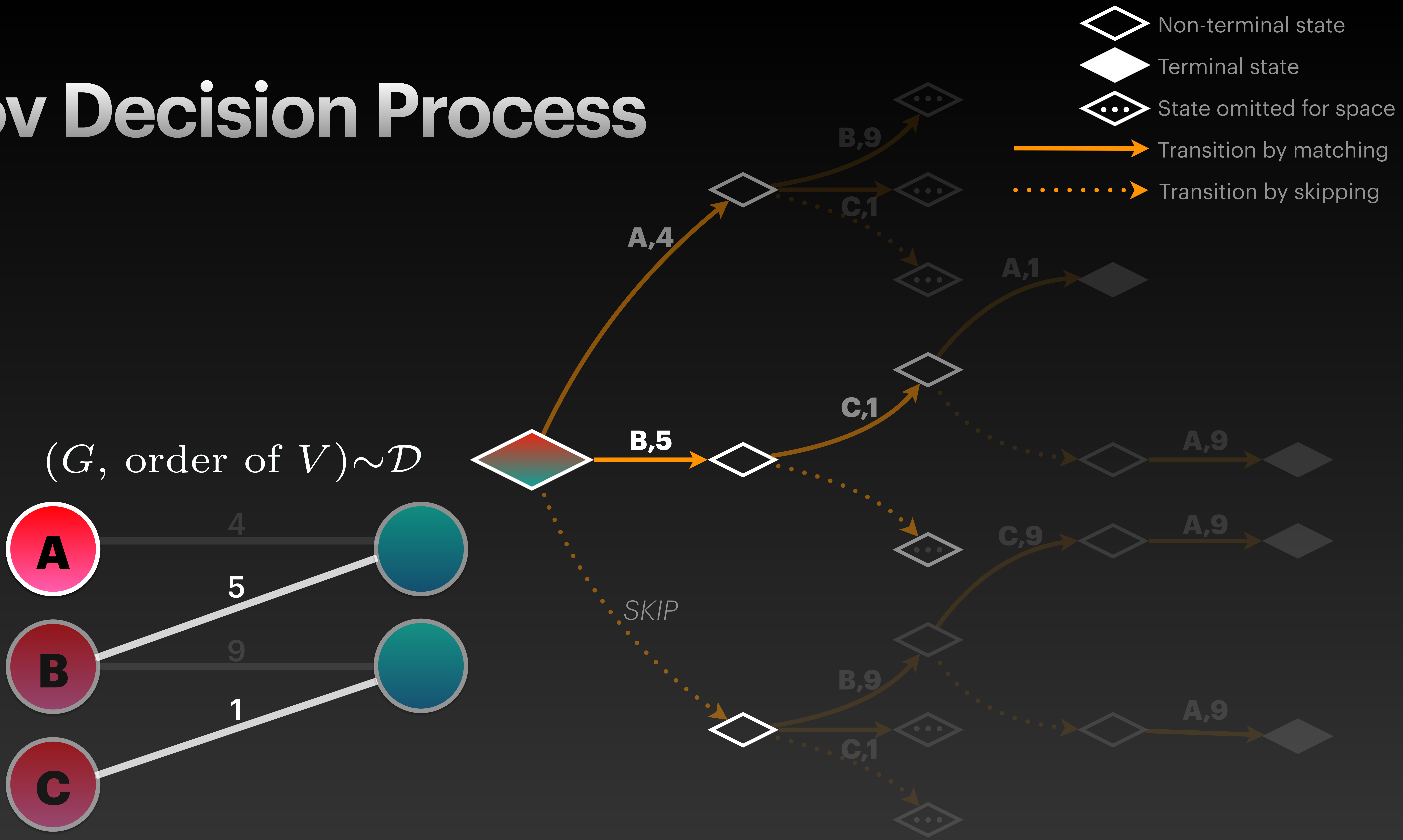
Markov Decision Process



Markov Decision Process

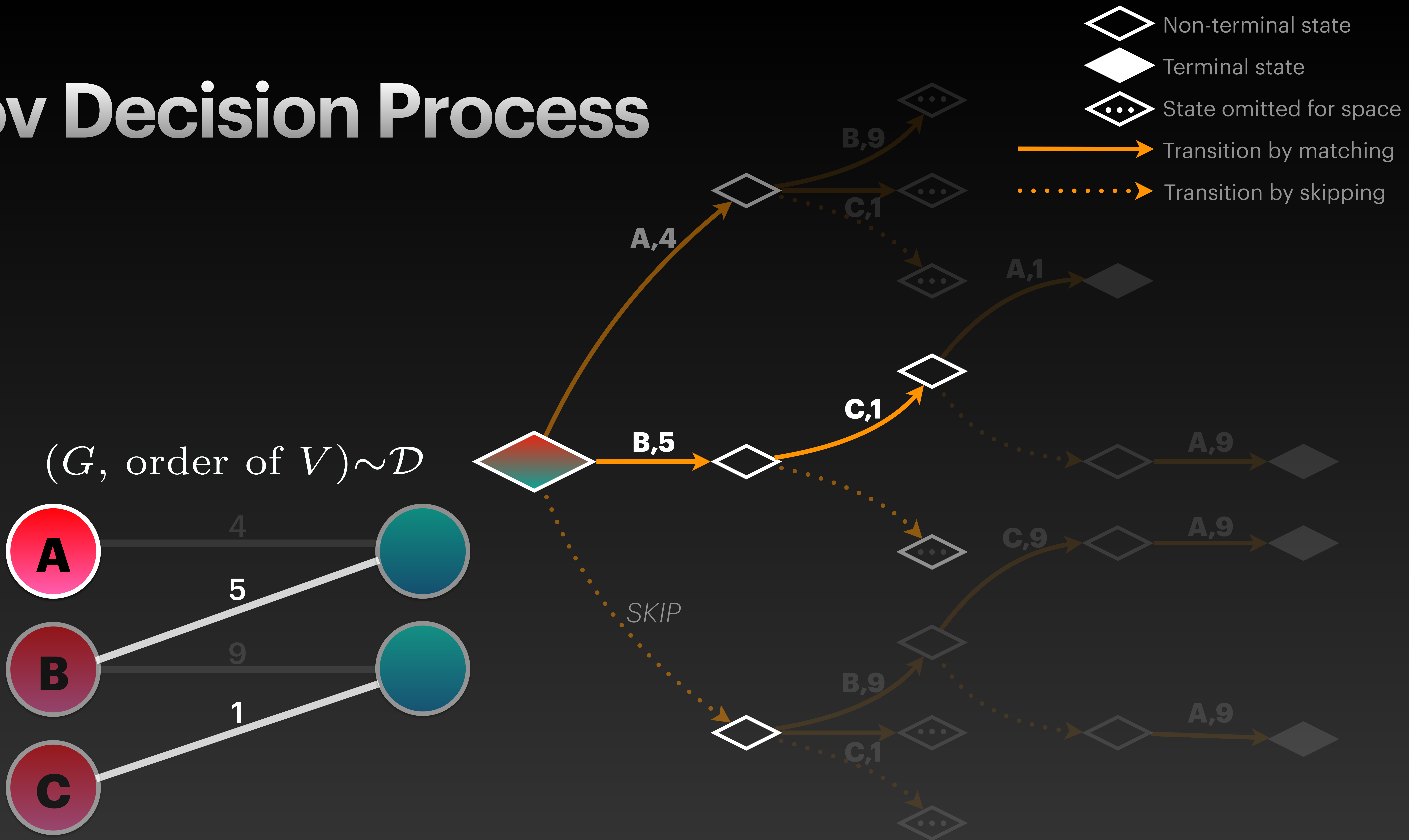


Markov Decision Process

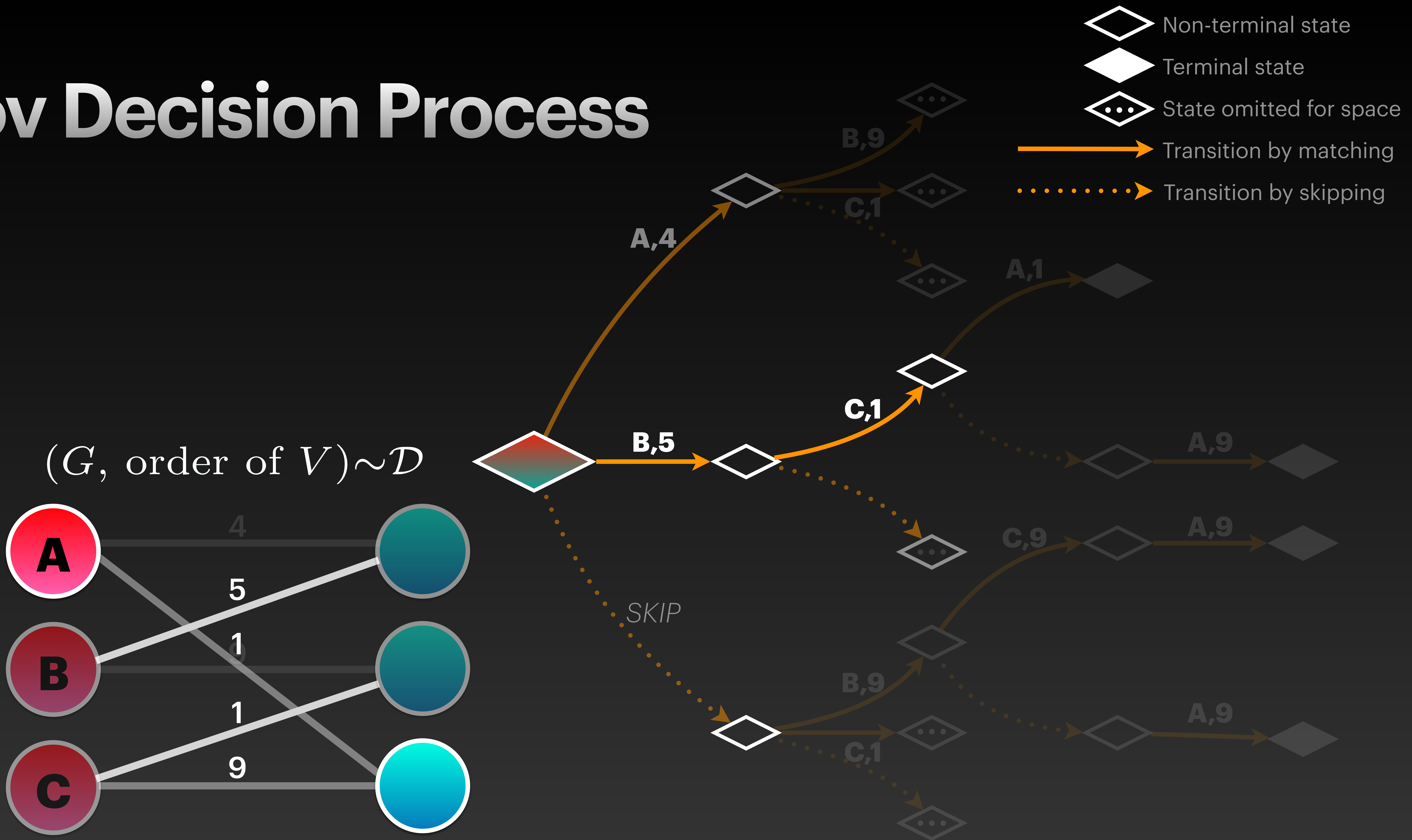


Markov Decision Process

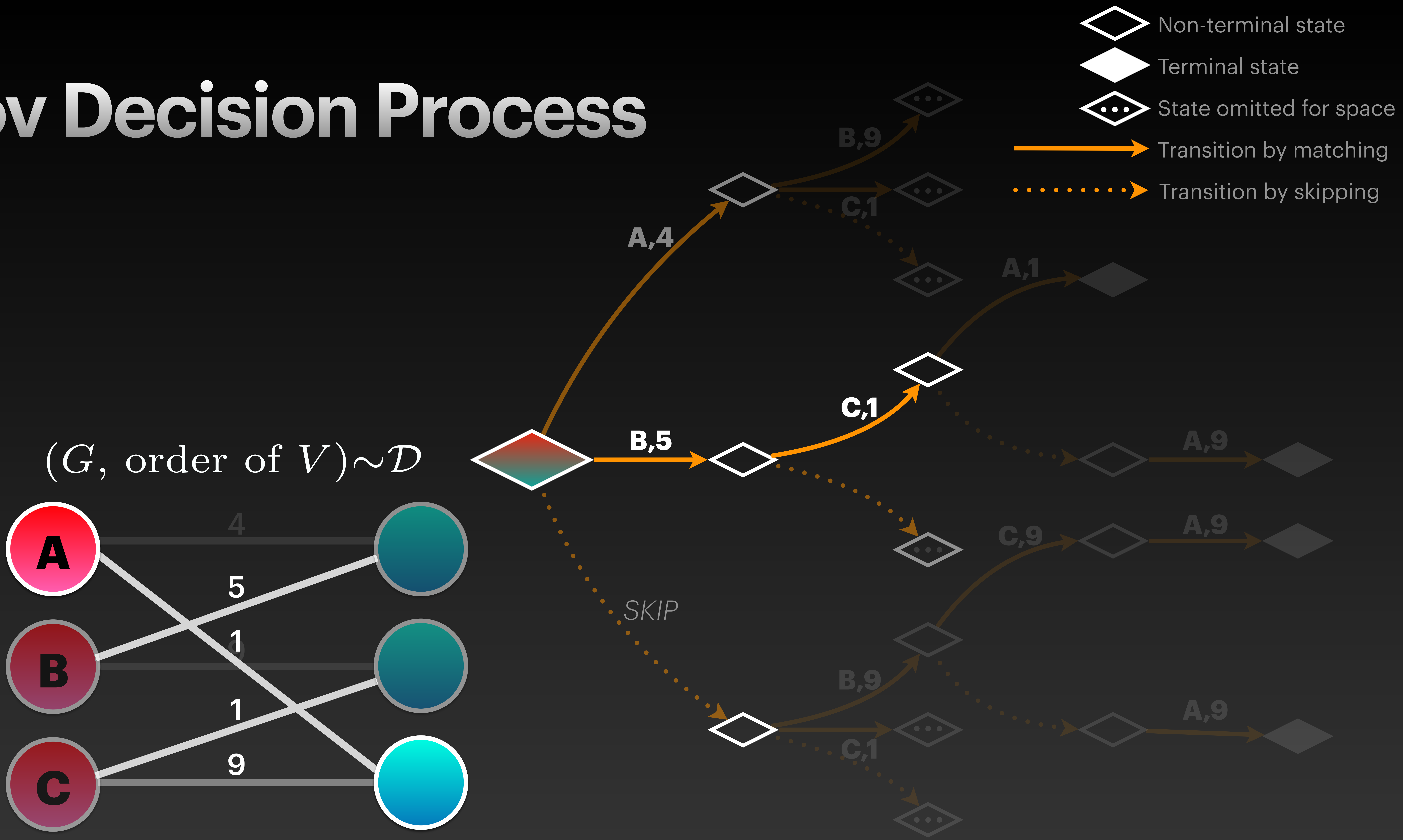
$(G, \text{ order of } V) \sim \mathcal{D}$



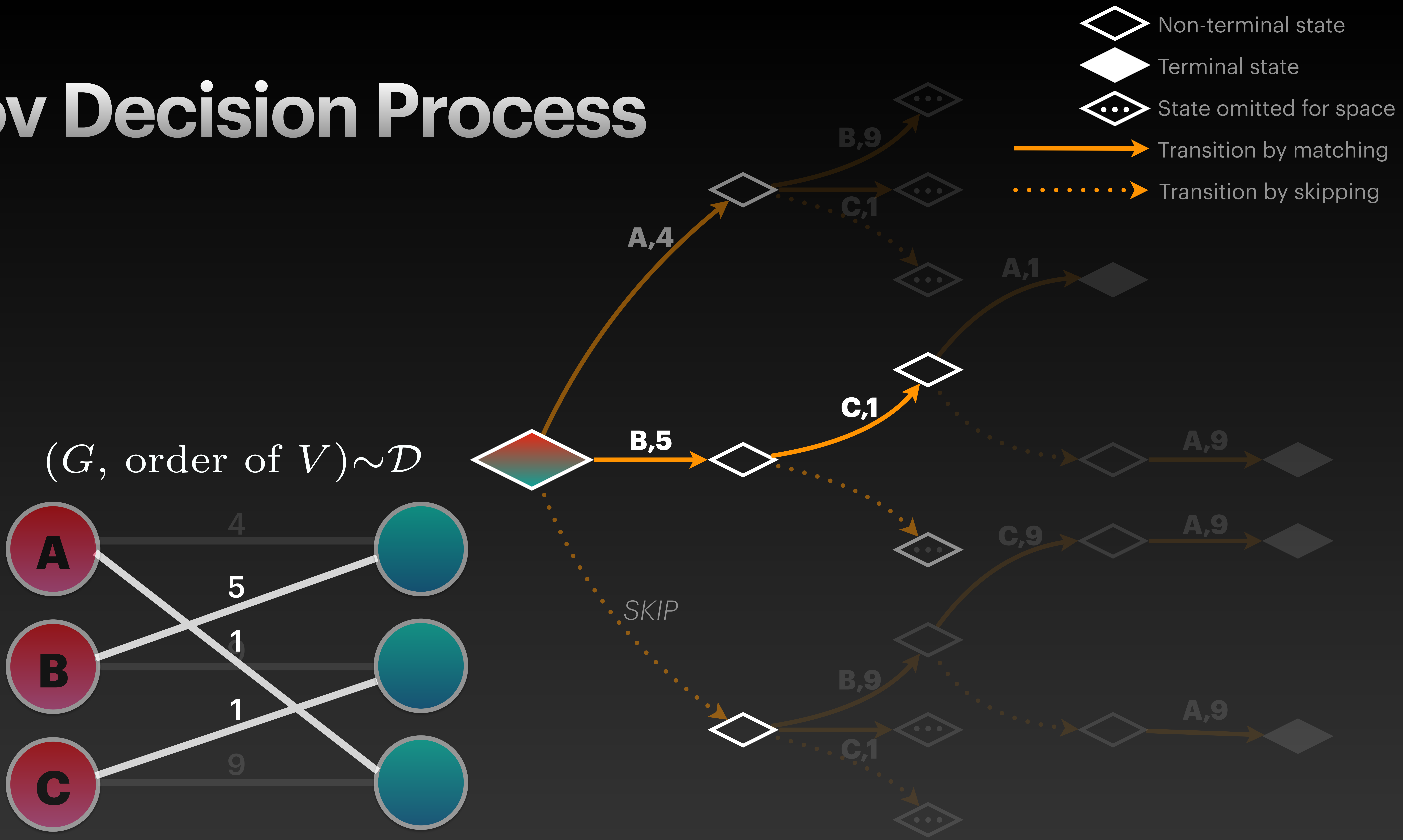
Markov Decision Process



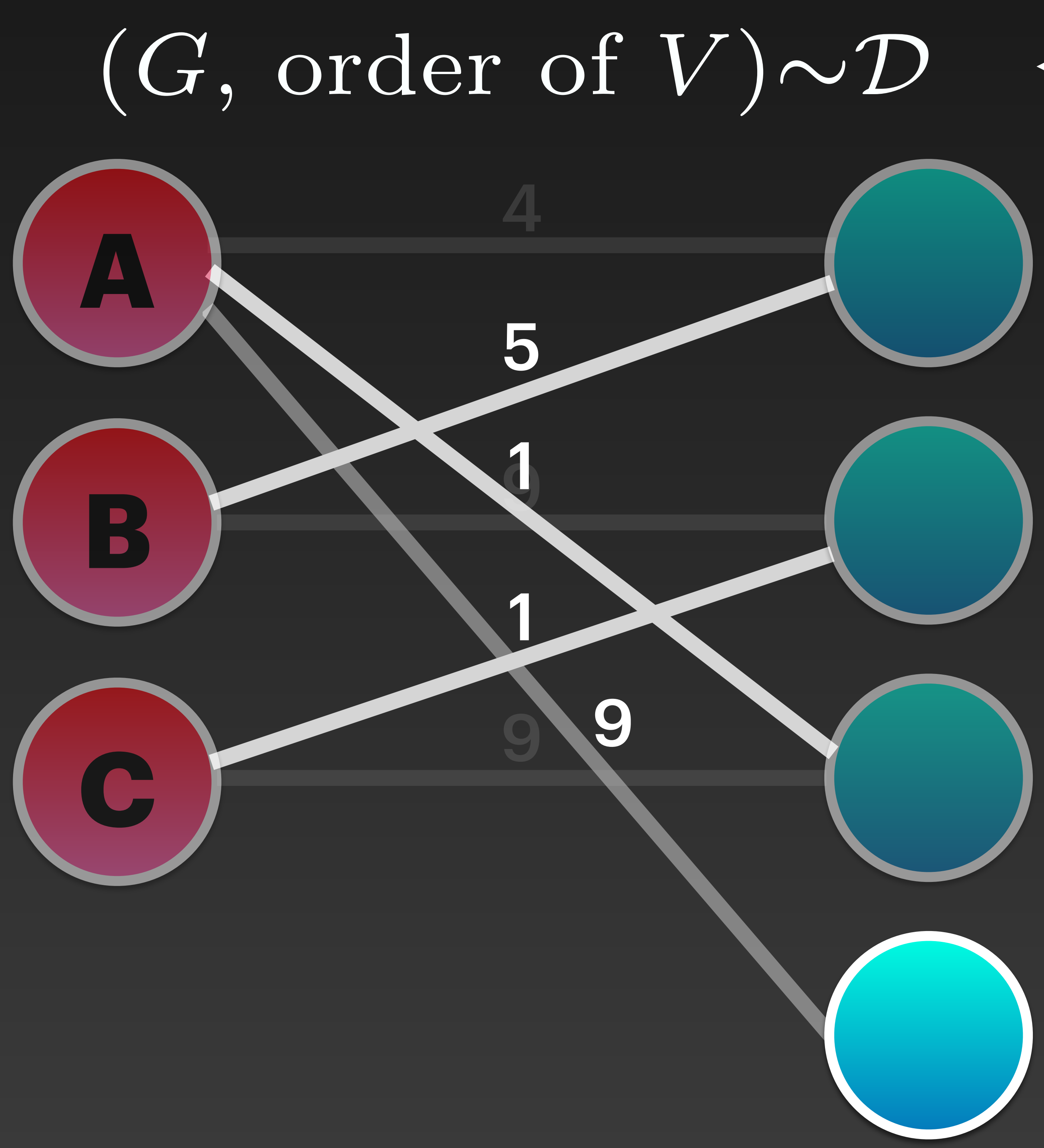
Markov Decision Process

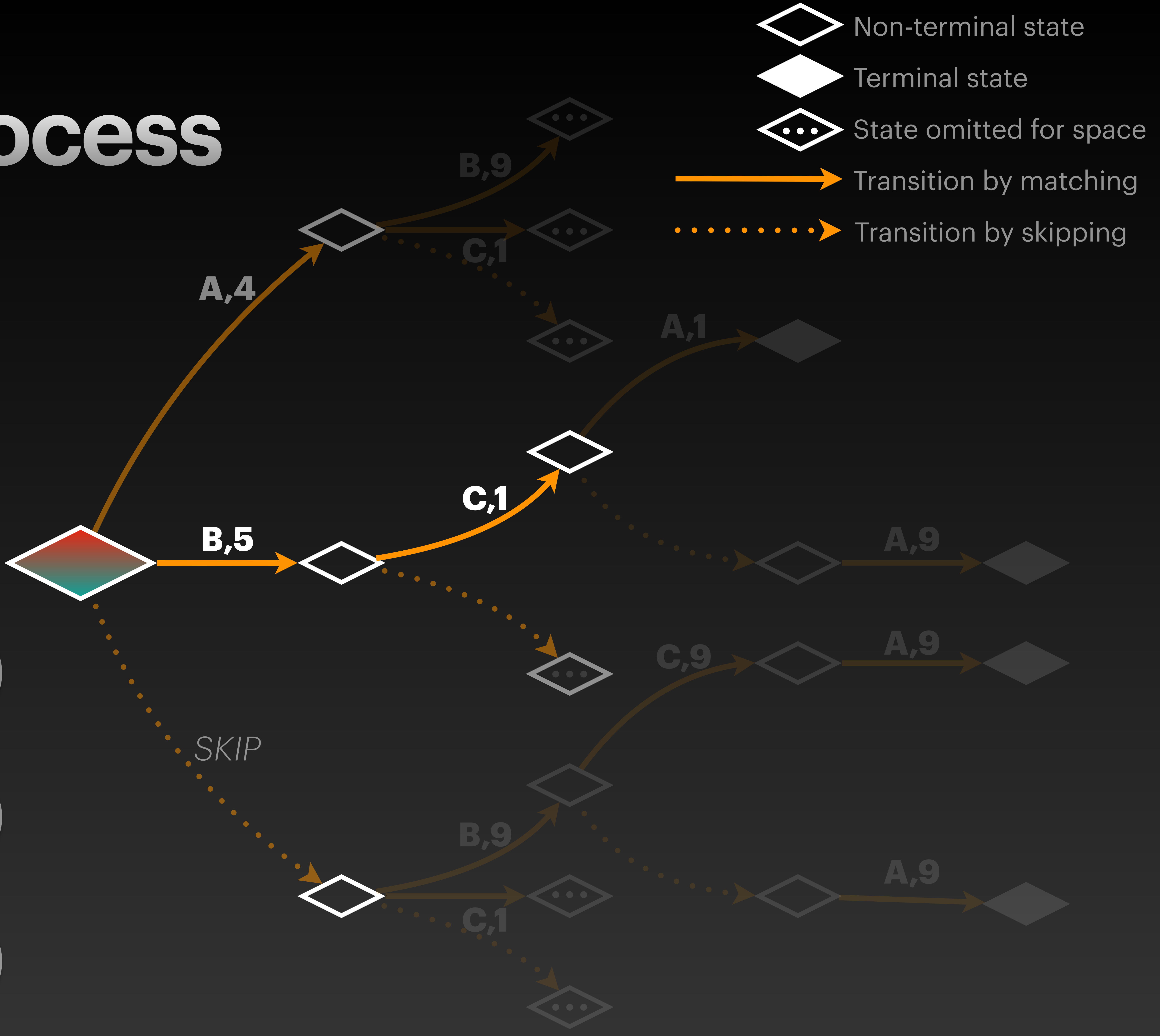


Markov Decision Process

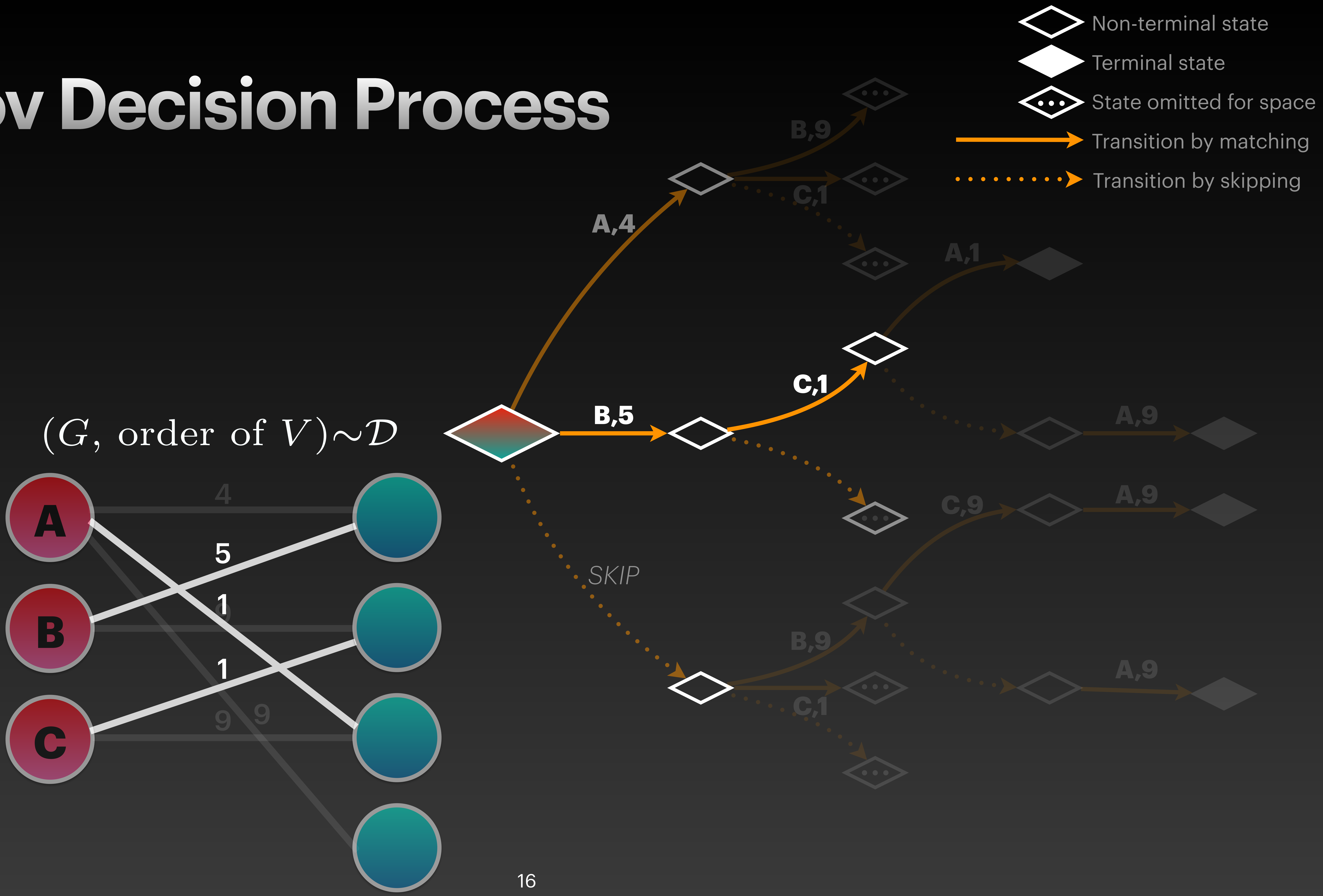


Markov Decision Process

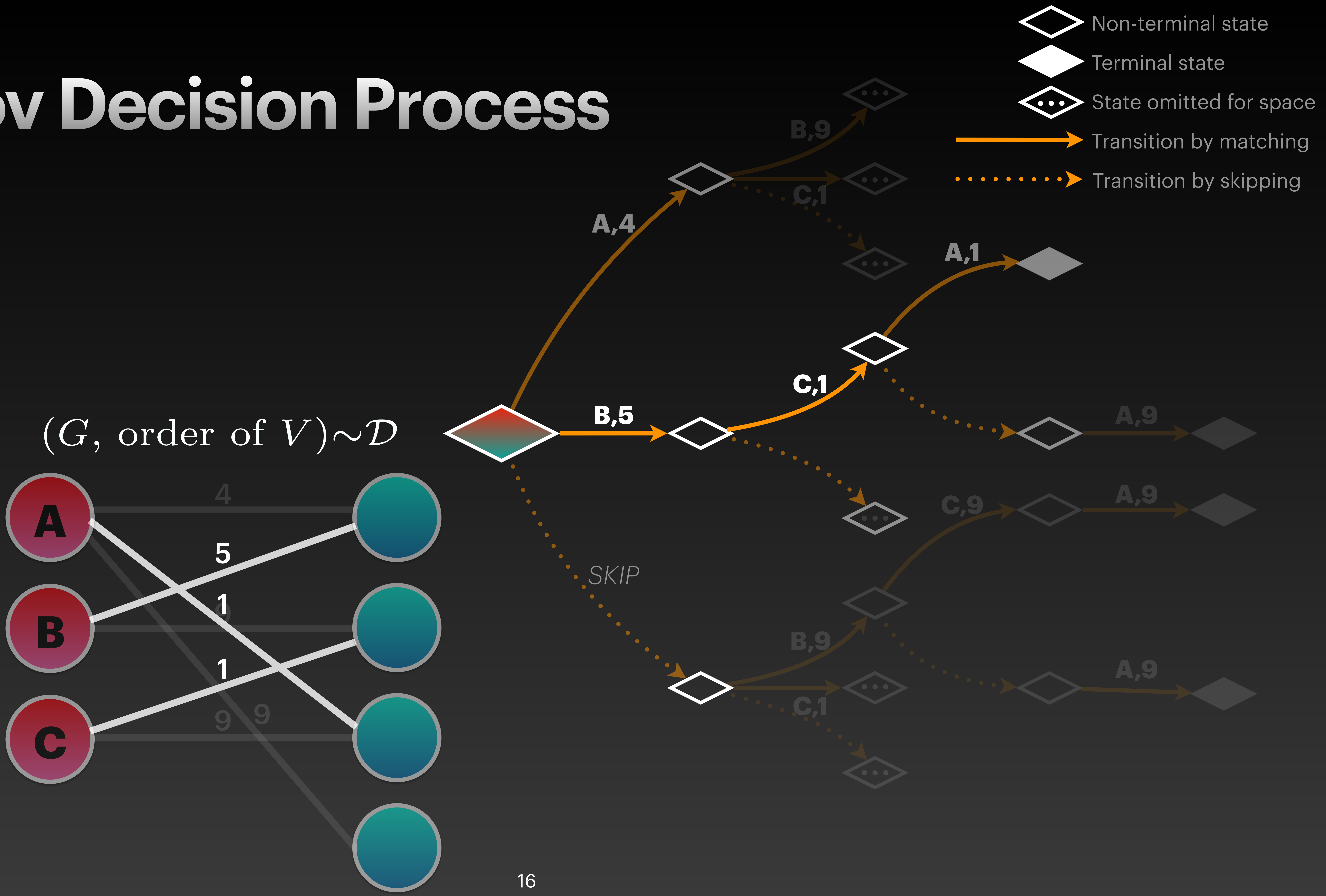




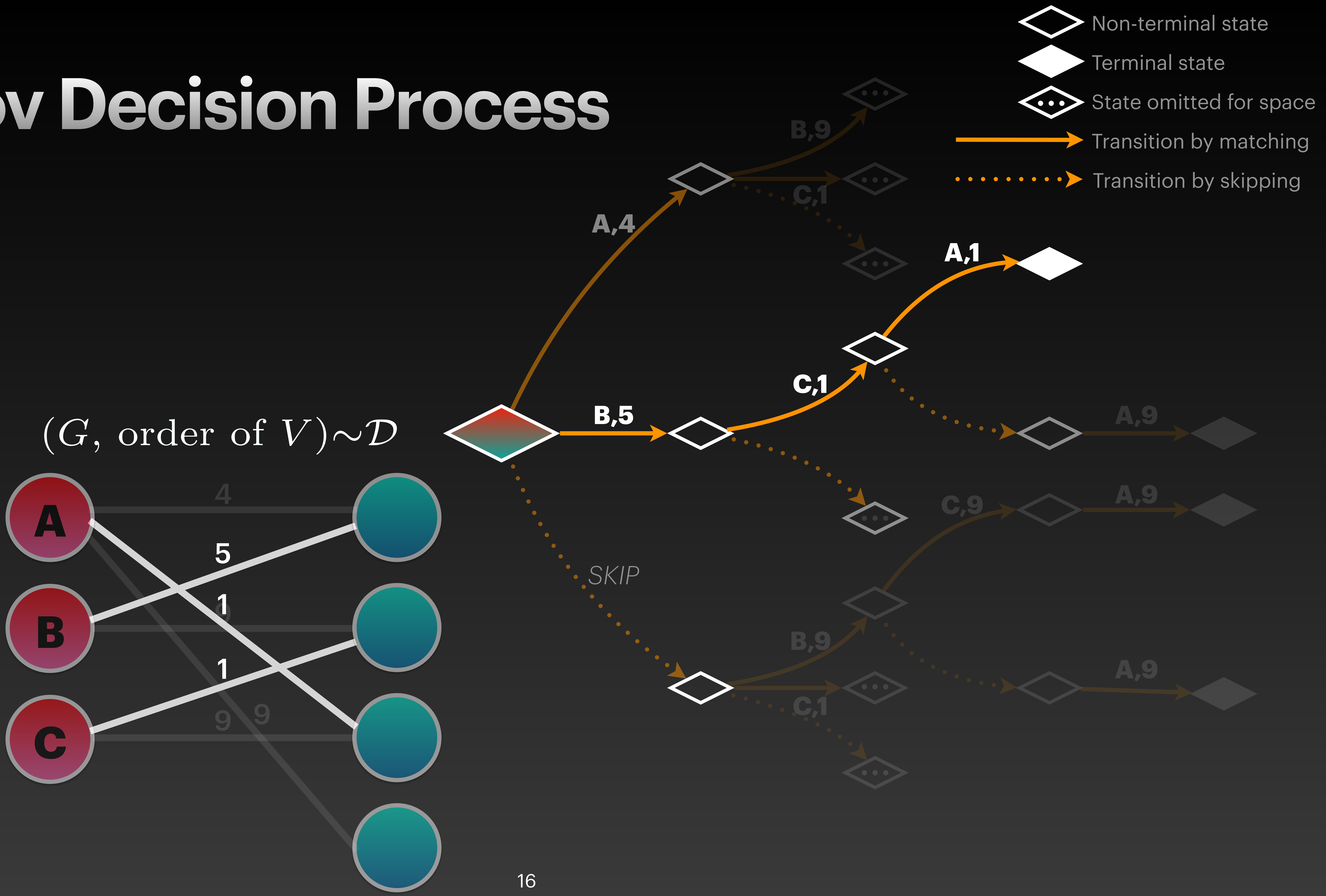
Markov Decision Process



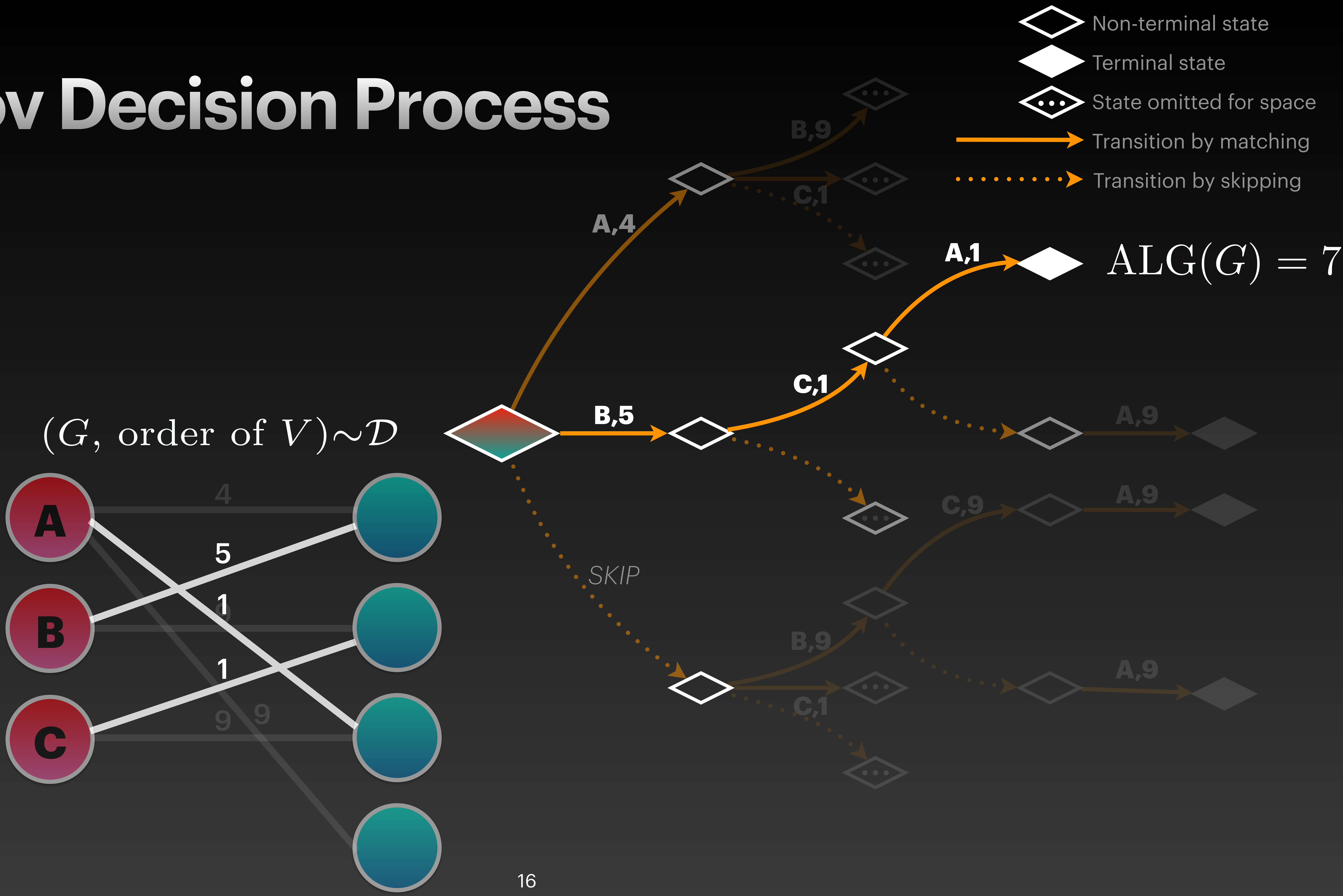
Markov Decision Process



Markov Decision Process

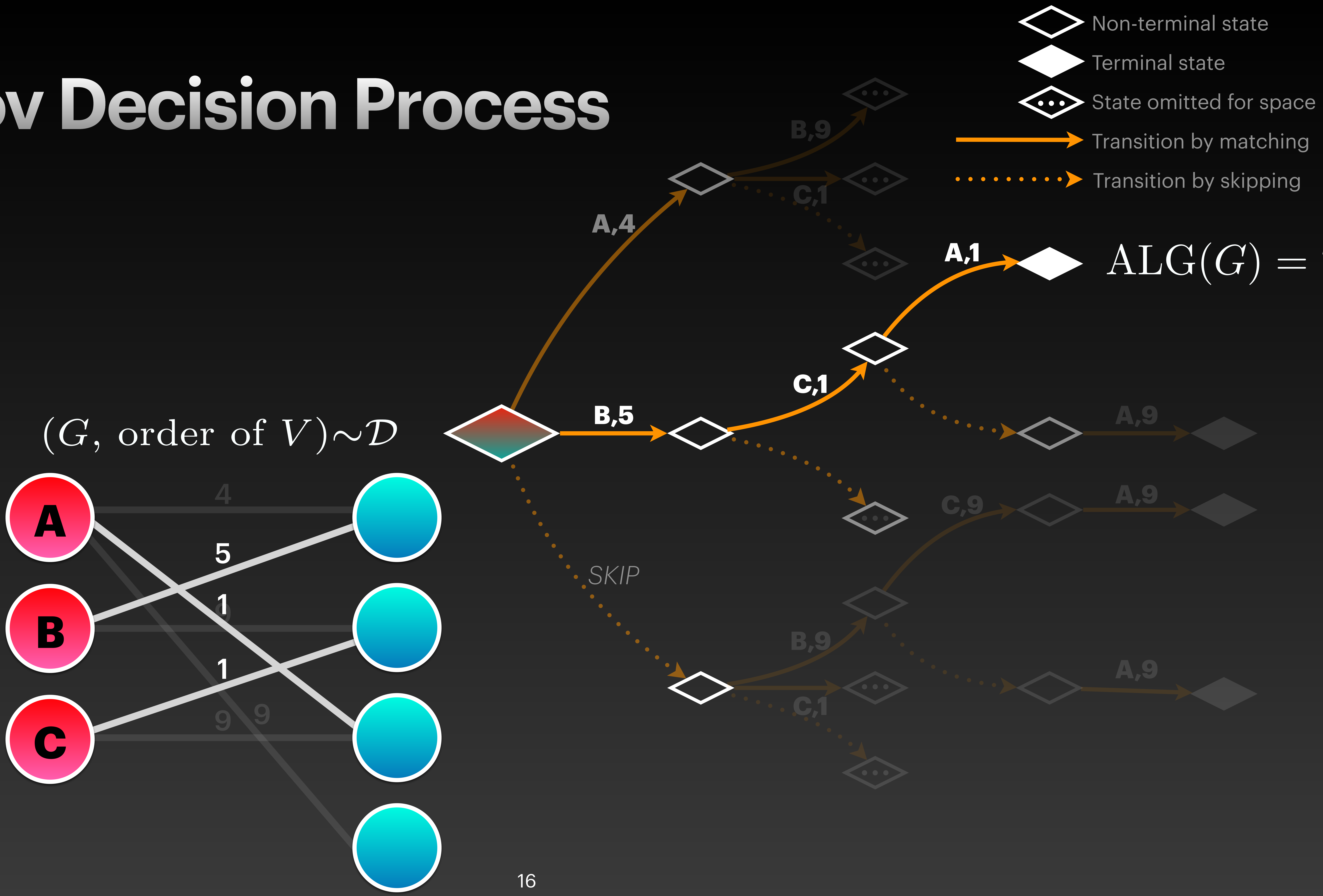


Markov Decision Process



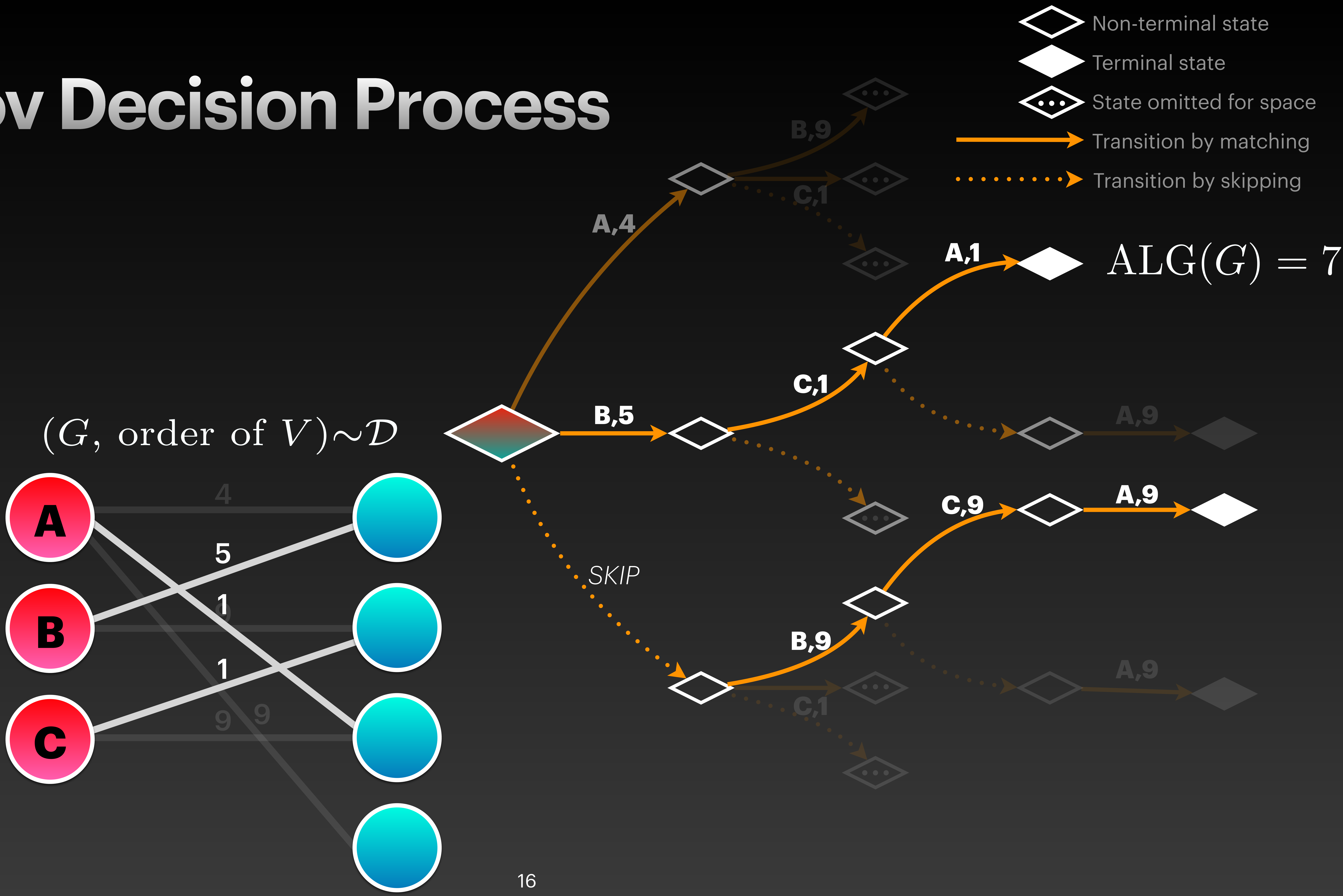
Markov Decision Process

$$(G,\ \text{order of } V) \sim \mathcal{D}$$

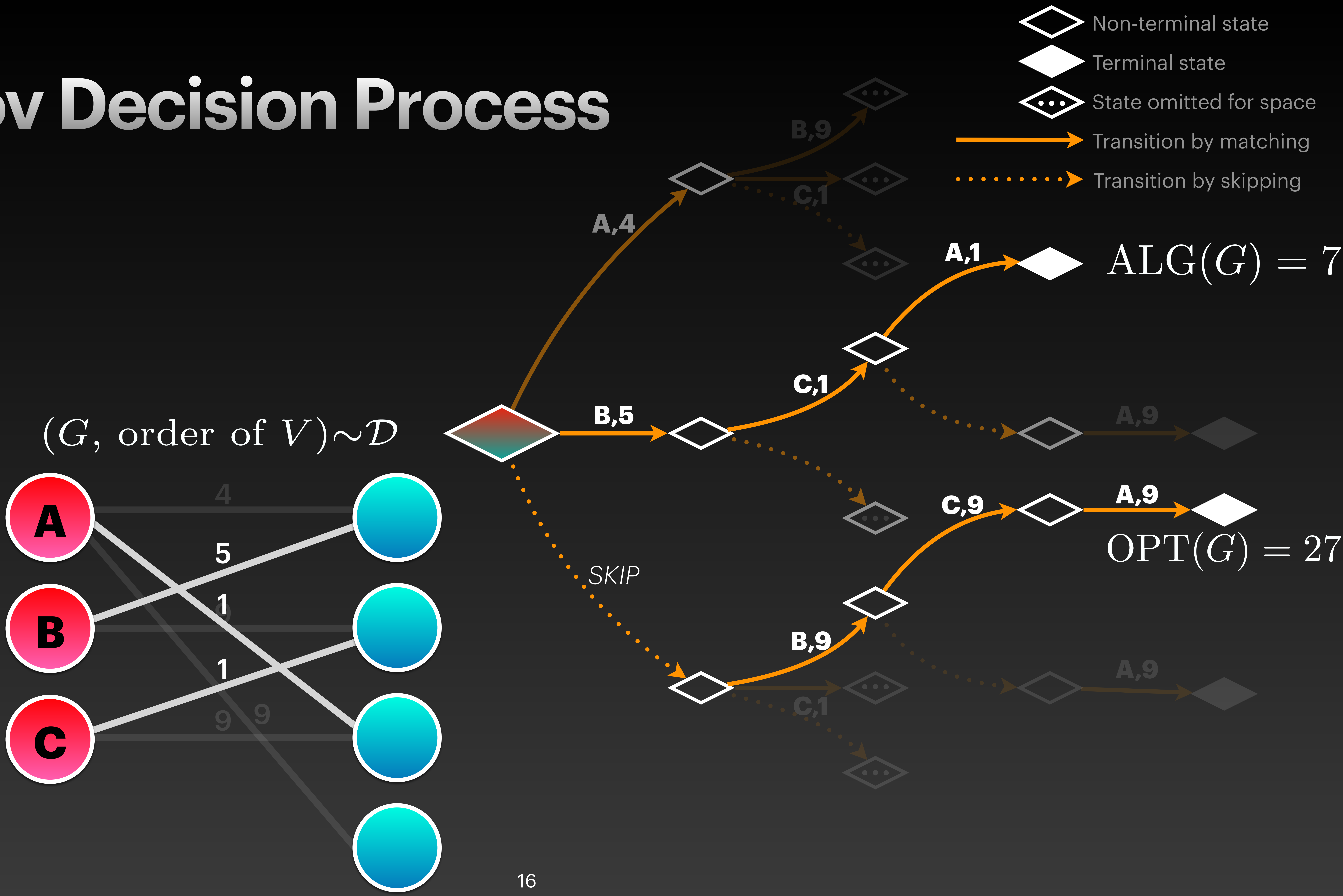


$$\mathrm{ALG}(G) = 7$$

Markov Decision Process

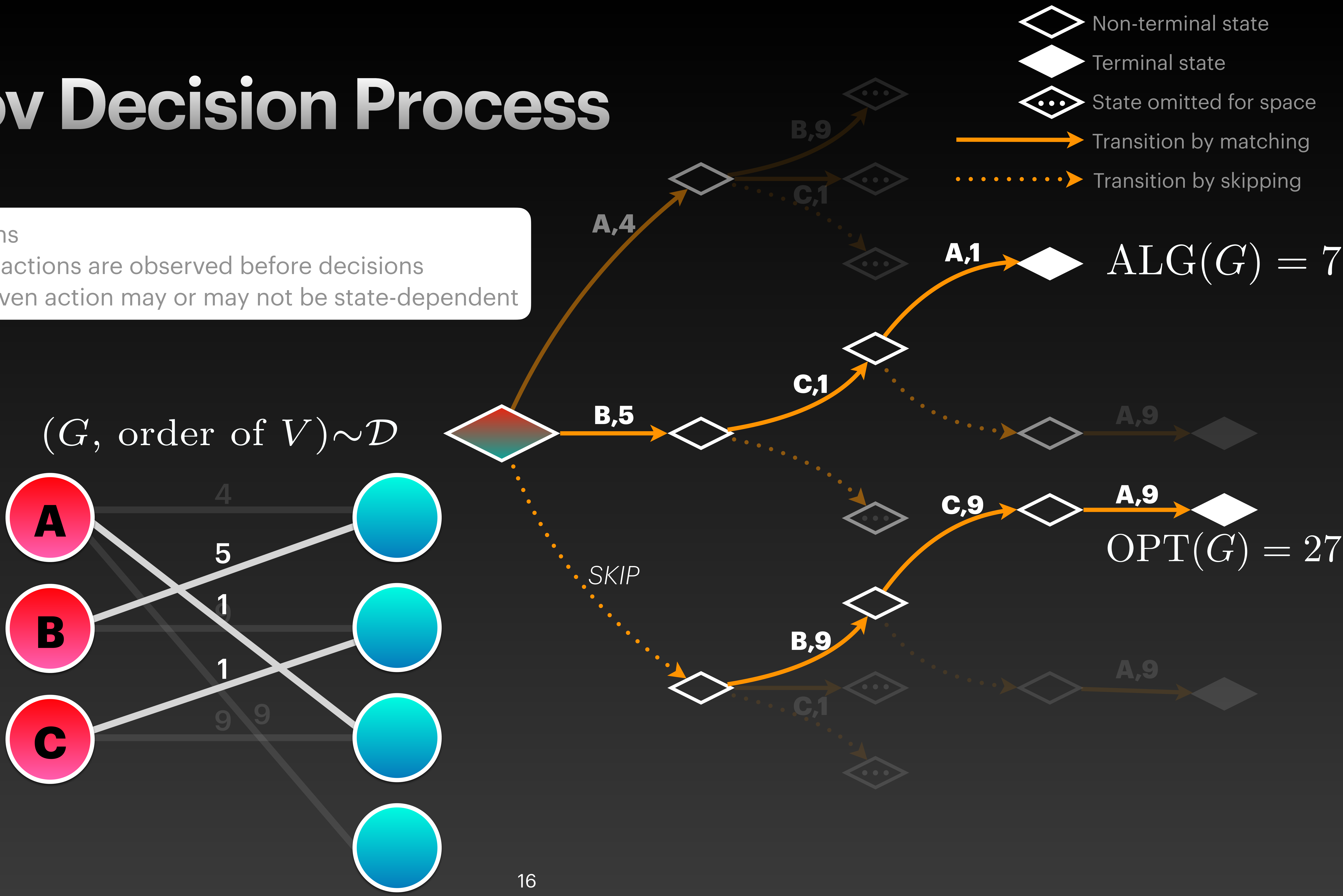


Markov Decision Process

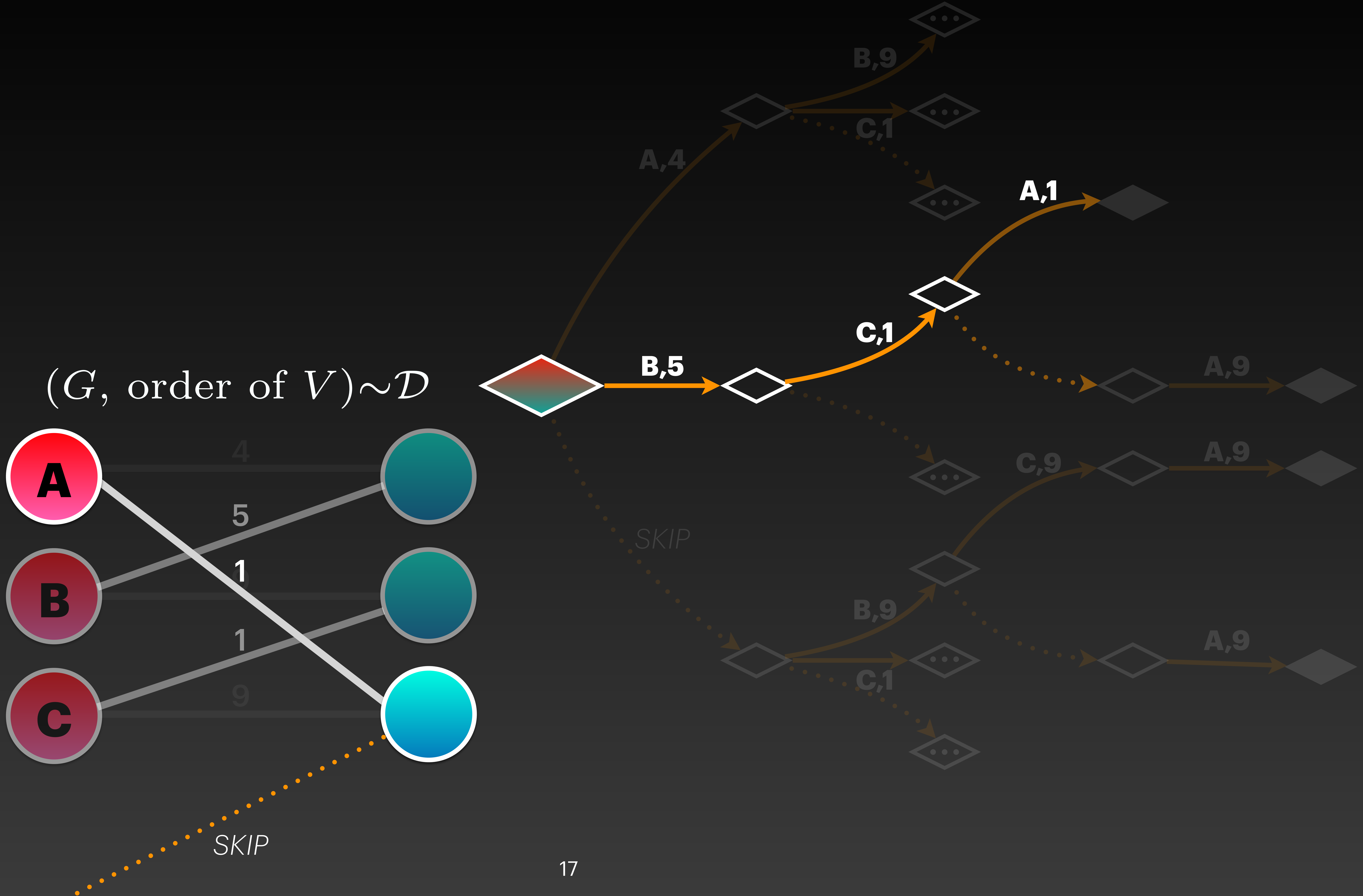


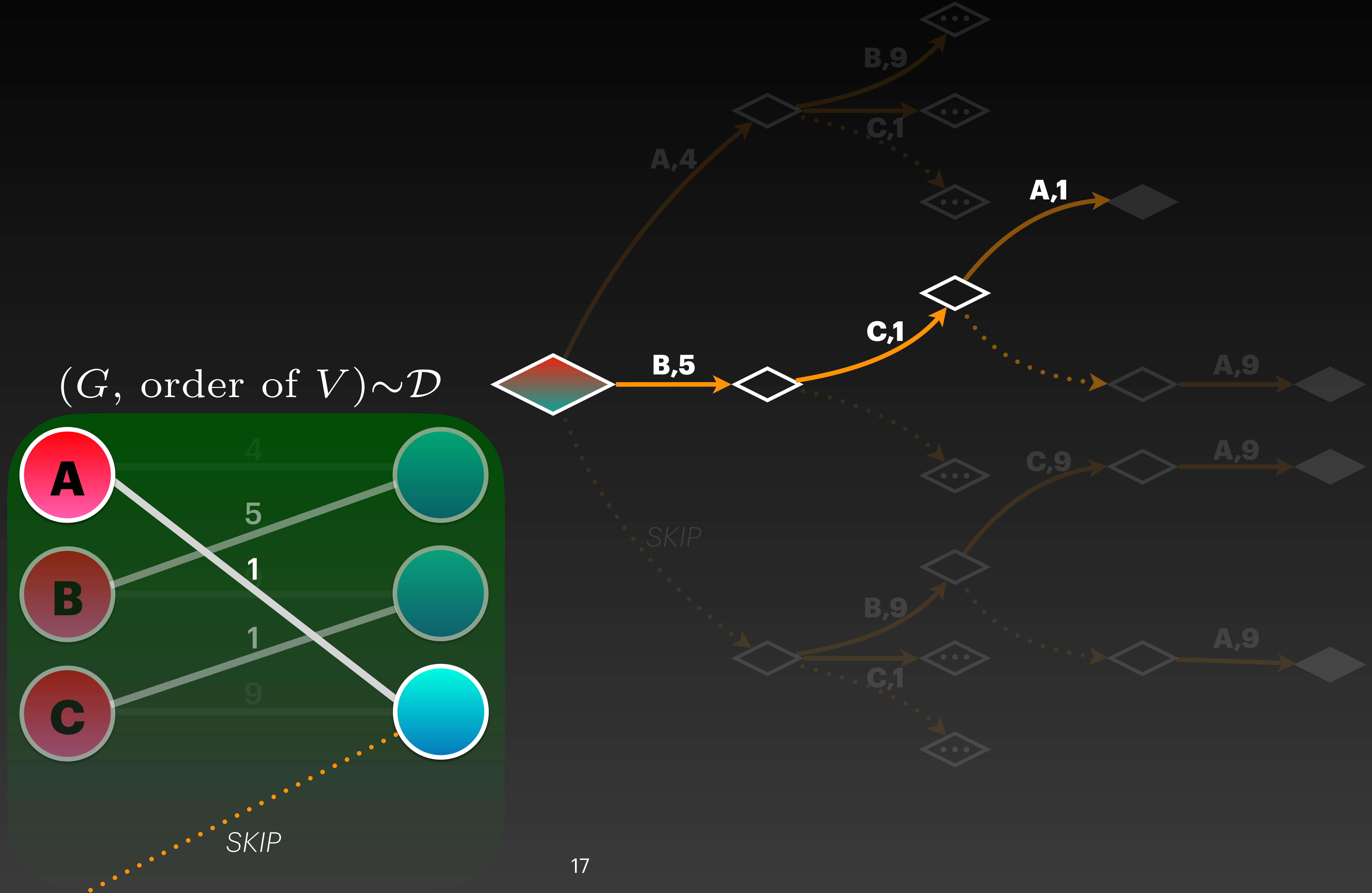
Markov Decision Process

- Deterministic transitions
- Per-step rewards of all actions are observed before decisions
- Per-step reward of a given action may or may not be state-dependent

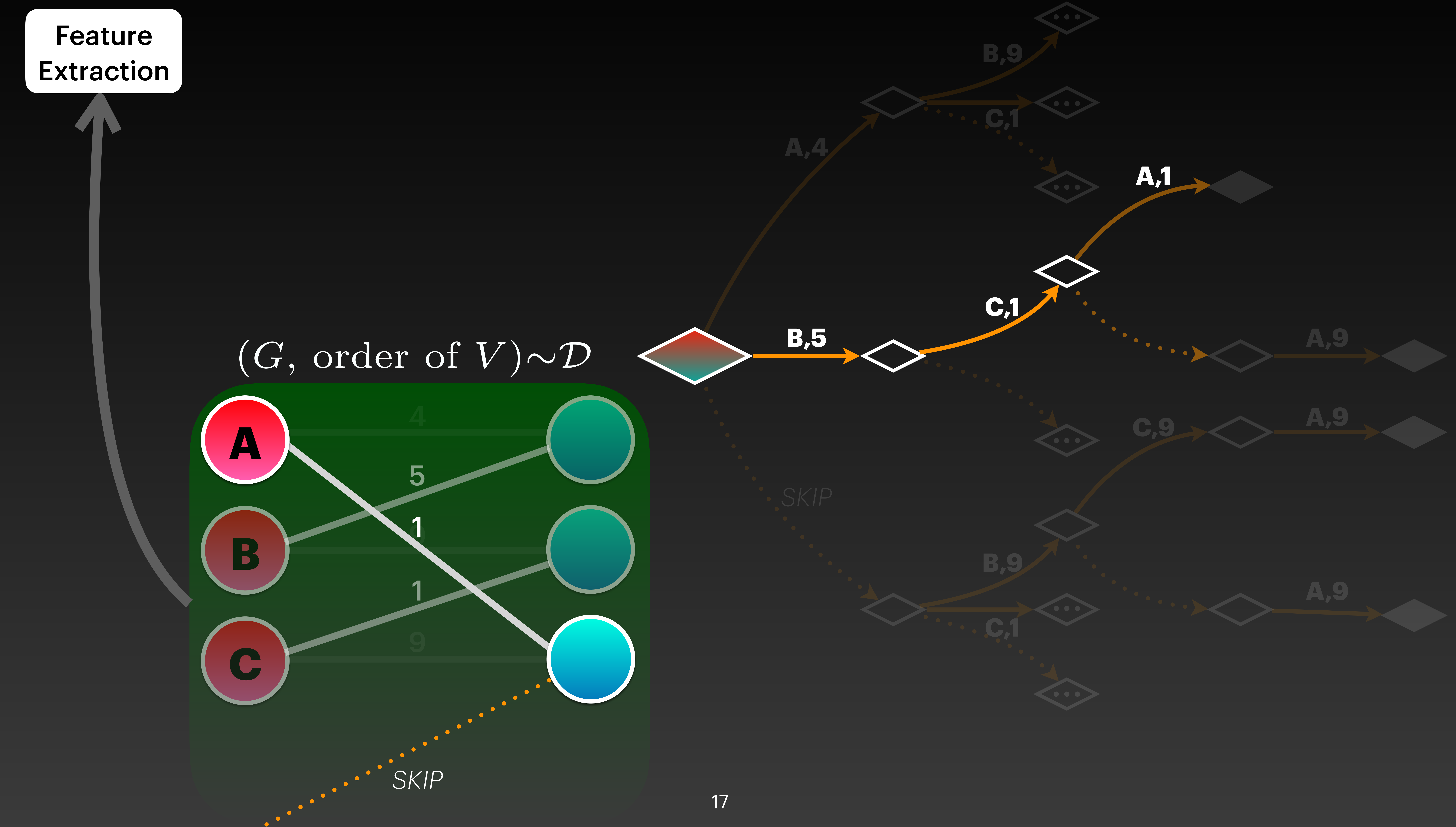


$(G, \text{ order of } V) \sim \mathcal{D}$

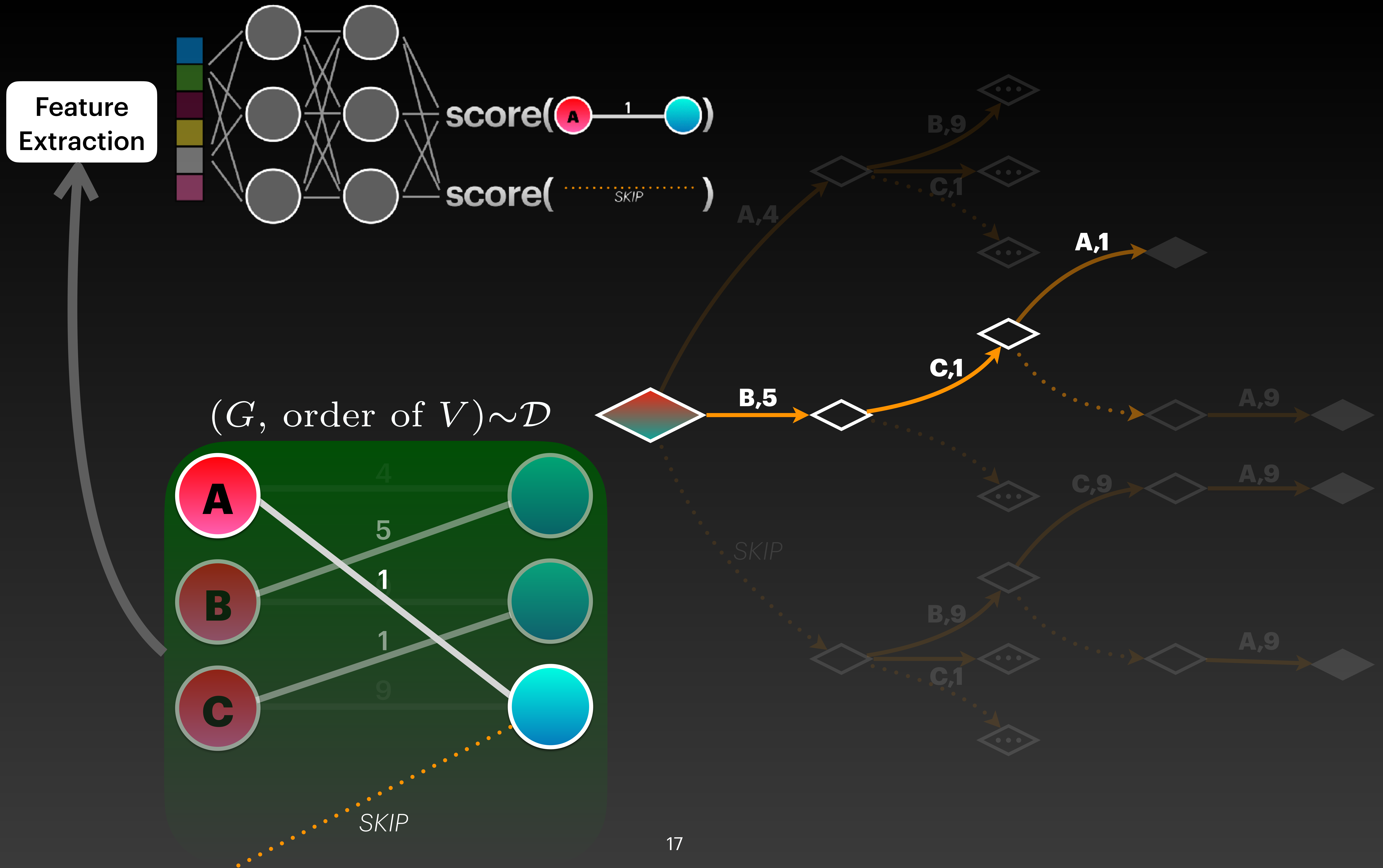




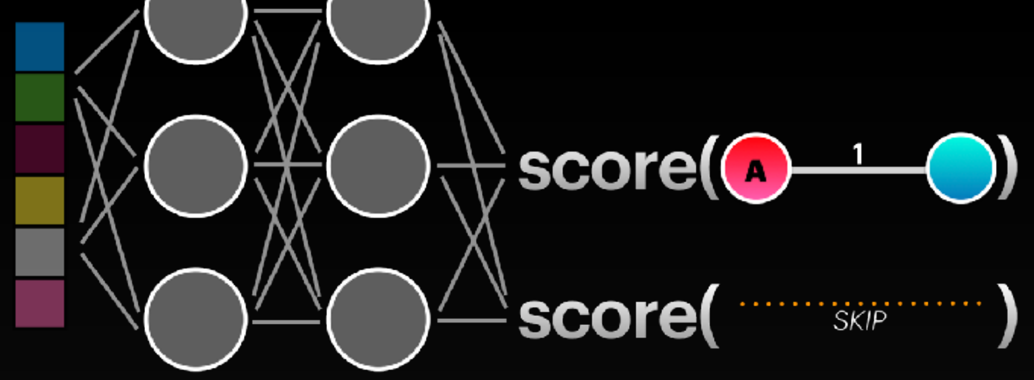
$(G,\ \text{order of } V) \sim \mathcal{D}$
A
B
C
4
5
1
1
9
SKIP
A,4
B,5
B,9
C,1
A,1
C,1
A,9
C,9
A,9
SKIP
B,9
C,1
A,9



Feature Extraction
(G, order of V)∼D
A
B
C
4
5
1
1
9
SKIP
B,5
C,1
A,1
A,4
B,9
C,1
A,9
C,9
A,9
SKIP
B,9
C,1
A,9

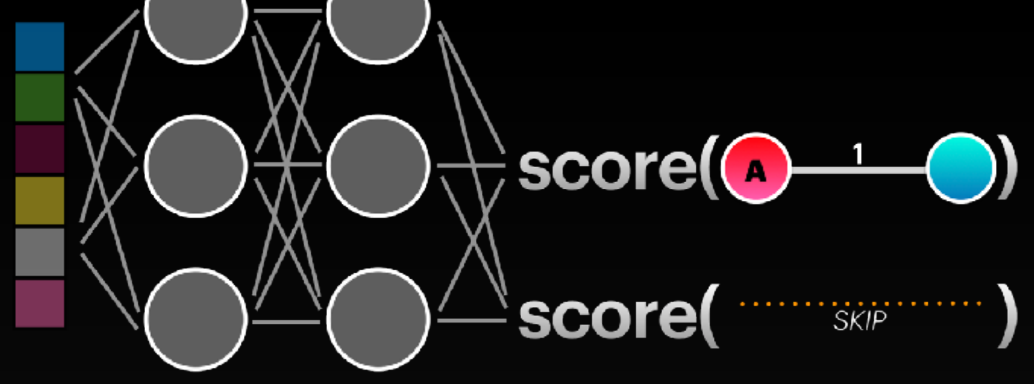

Feature Extraction
score(A 1)
score(SKIP)
$(G, \text{ order of } V) \sim \mathcal{D}$
A
B
C
4
5
1
1
9
SKIP
B,5
C,1
A,1
A,4
B,9
C,1
A,9
C,9
A,9
SKIP
B,9
C,1
A,9

Desirable Properties of a Model

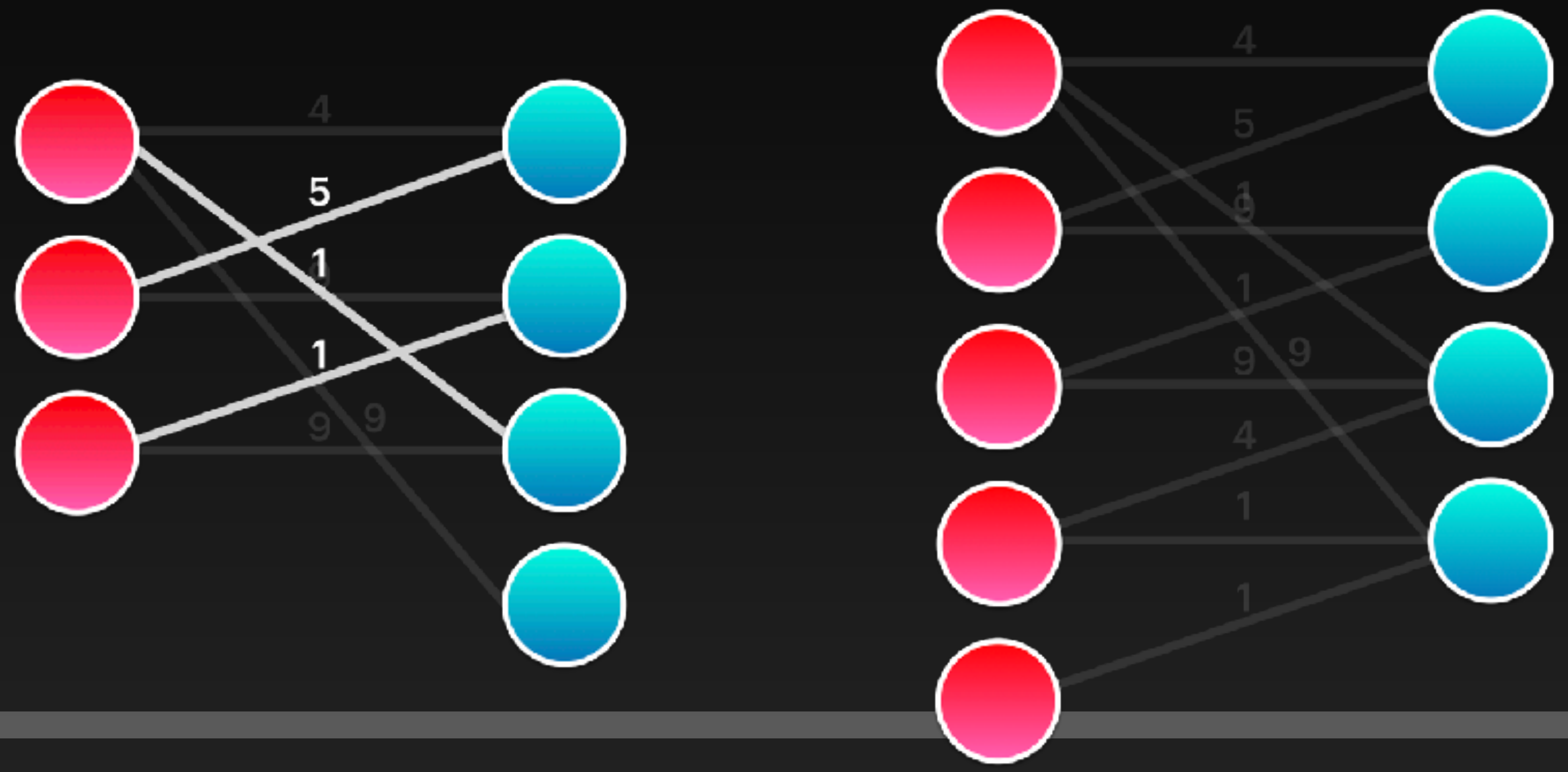


Sources: movielens.org, vecteezy.com

Desirable Properties of a Model

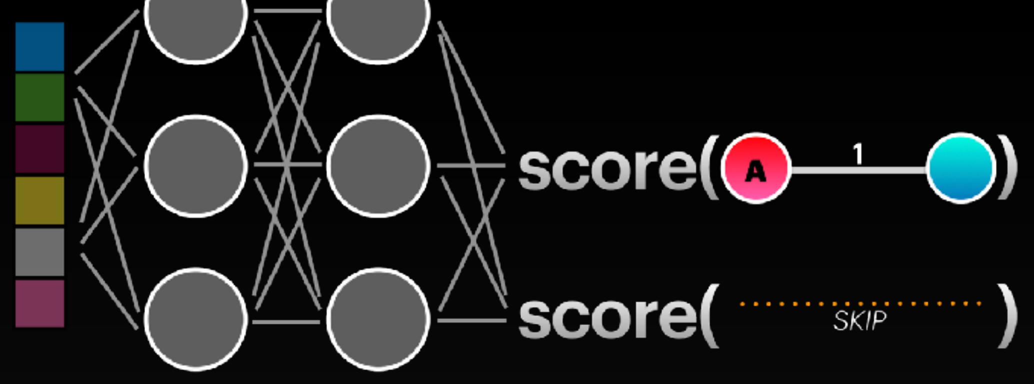


1 Works for different Graph Sizes



Sources: movielens.org, vecteezy.com

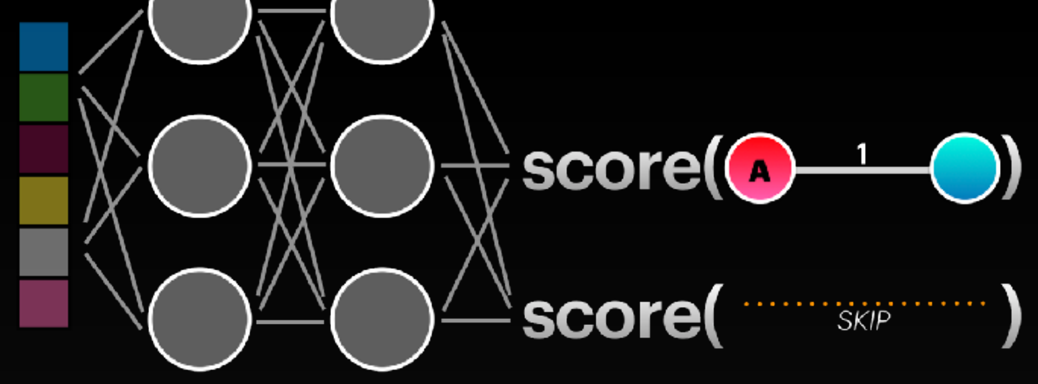
Desirable Properties of a Model



1 Works for different Graph Sizes

Sources: movielens.org, vecteezy.com

Desirable Properties of a Model



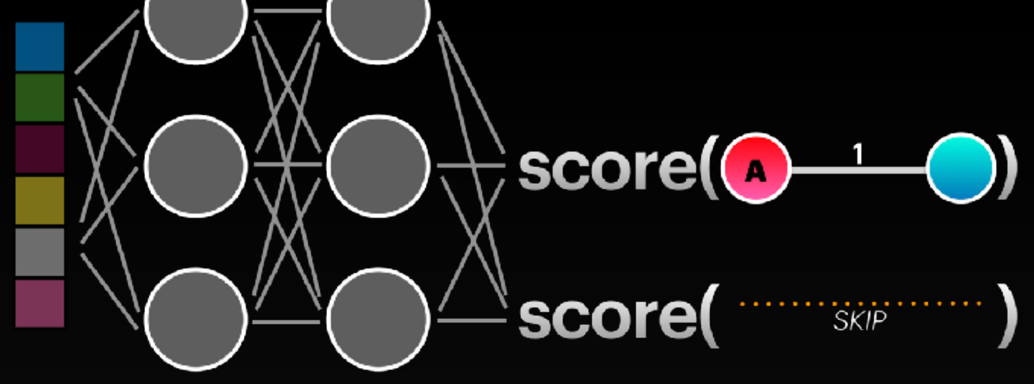
1 Works for different Graph Sizes

Permutation-equivariant 2

Zaheer, Manzil, et al. "Deep Sets." NeurIPS (2017).

Sources: movielens.org, vecteezy.com

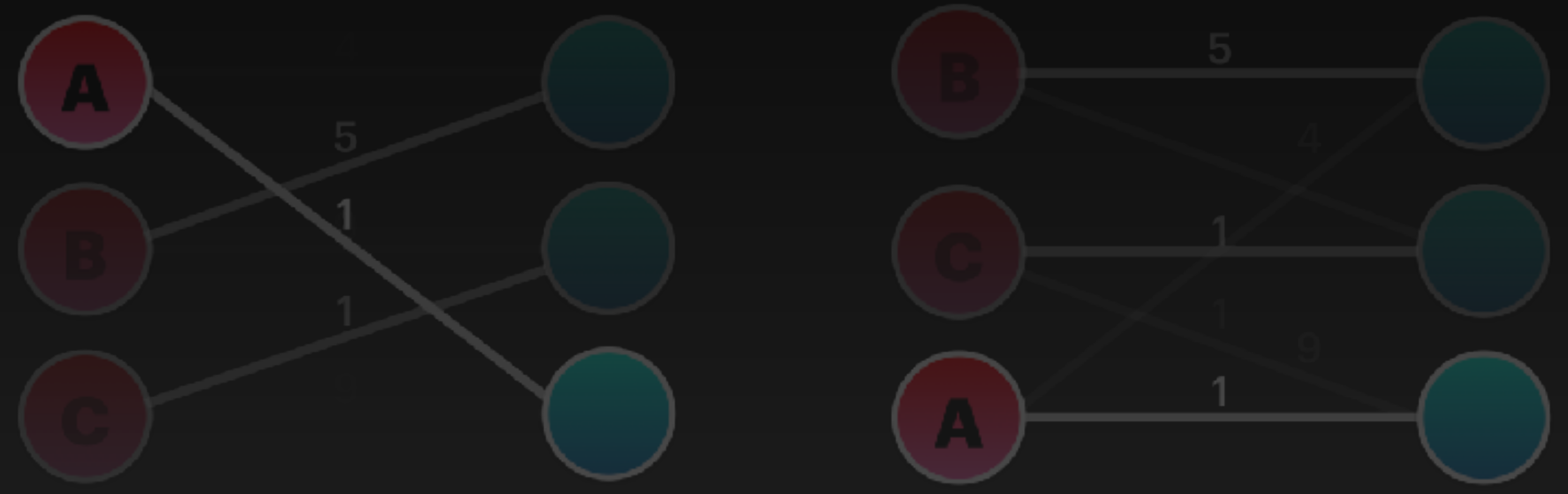
Desirable Properties of a Model



1 Works for different Graph Sizes

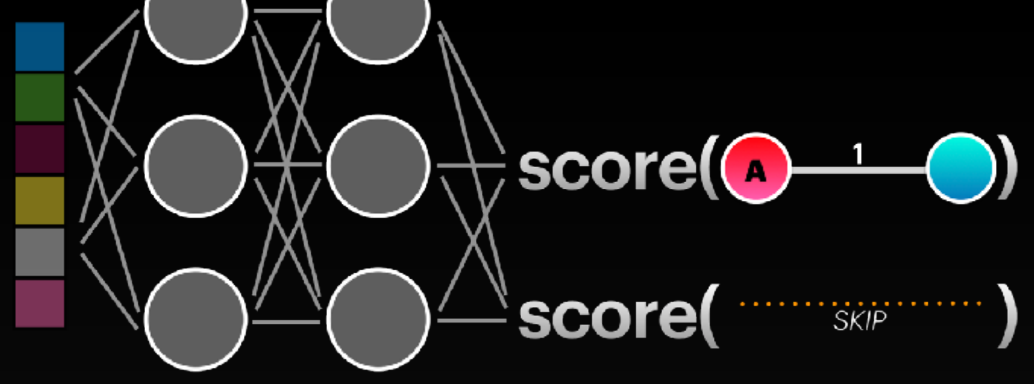
Permutation-equivariant 2

Zaheer, Manzil, et al. "Deep Sets." NeurIPS (2017).



Sources: movielens.org, vecteezy.com

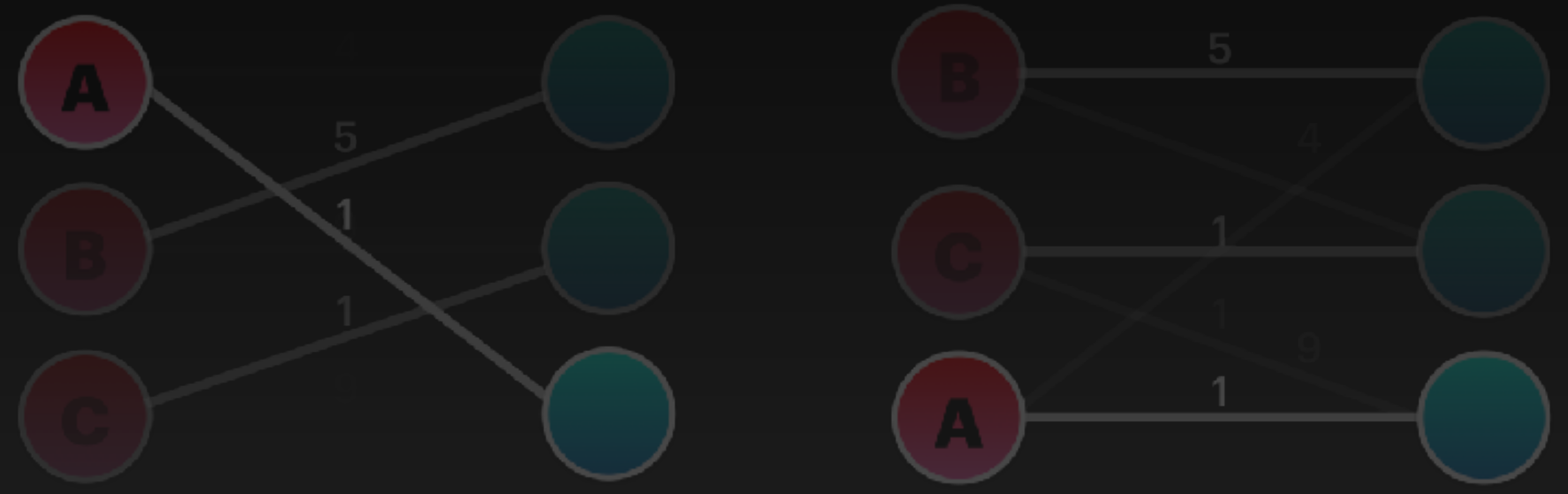
Desirable Properties of a Model



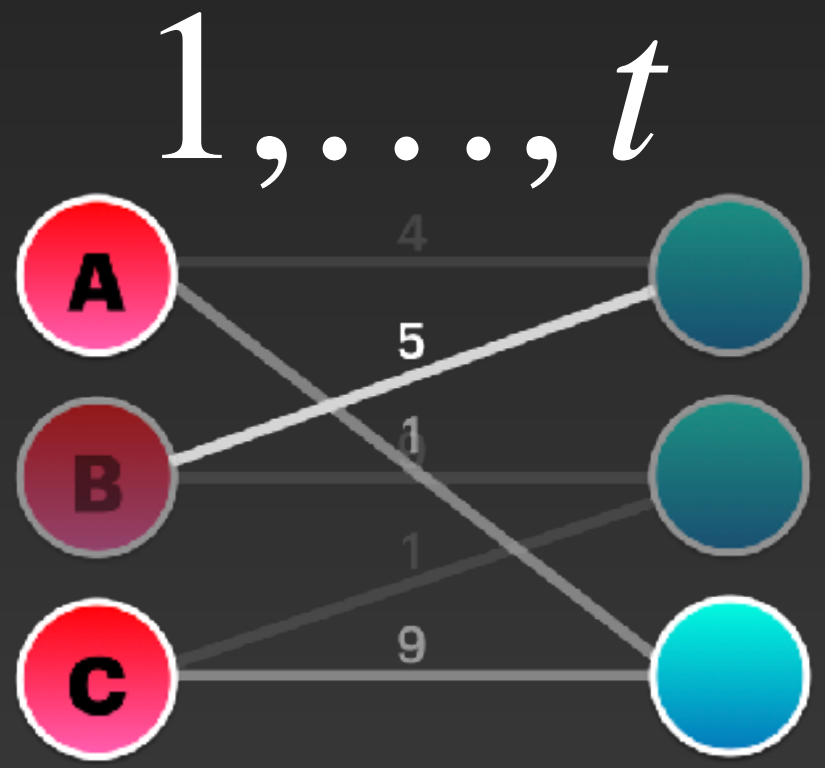
1 Works for different Graph Sizes

Permutation-equivariant 2

Zaheer, Manzil, et al. "Deep Sets." NeurIPS (2017).



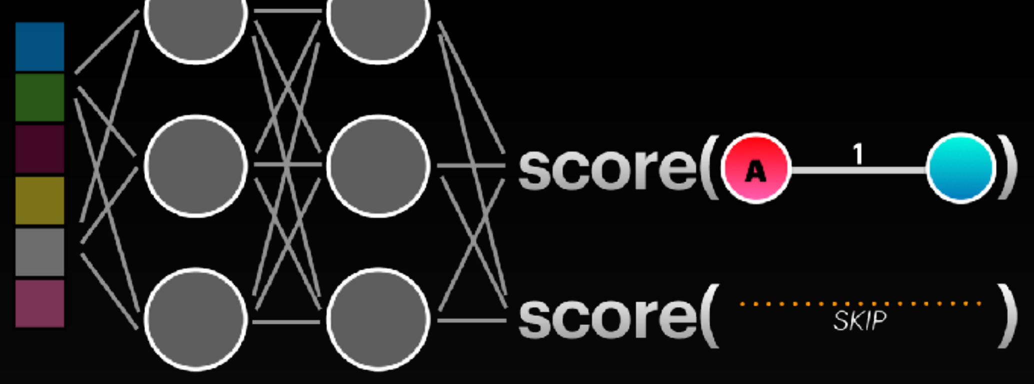
3 Accounting for History



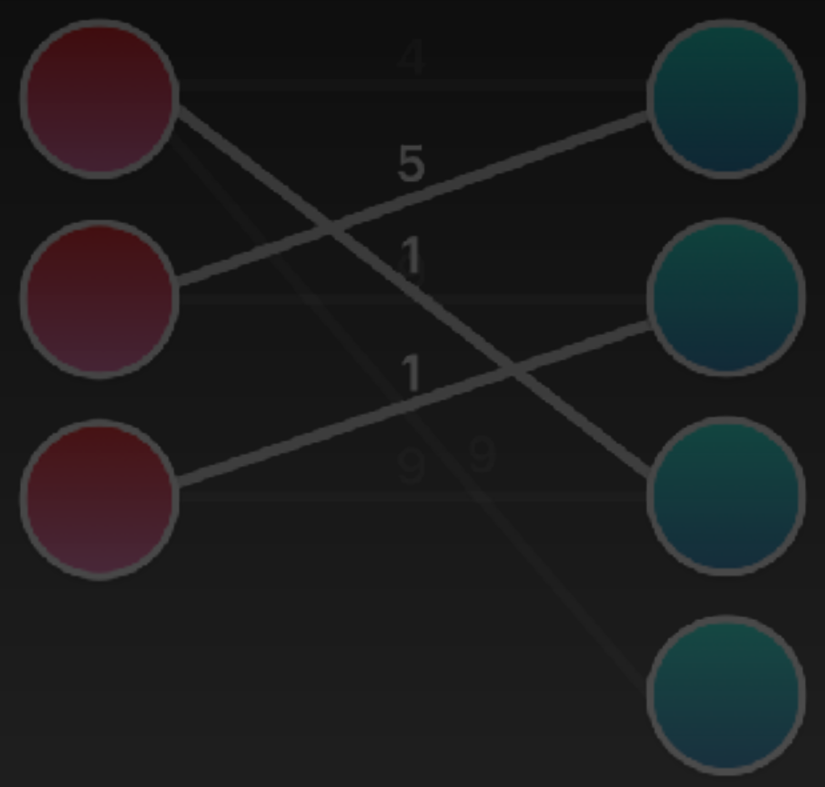
- Avg./Variance/Min/Max of weights in current matching
- How many skips?
- Average U-node degree, ...

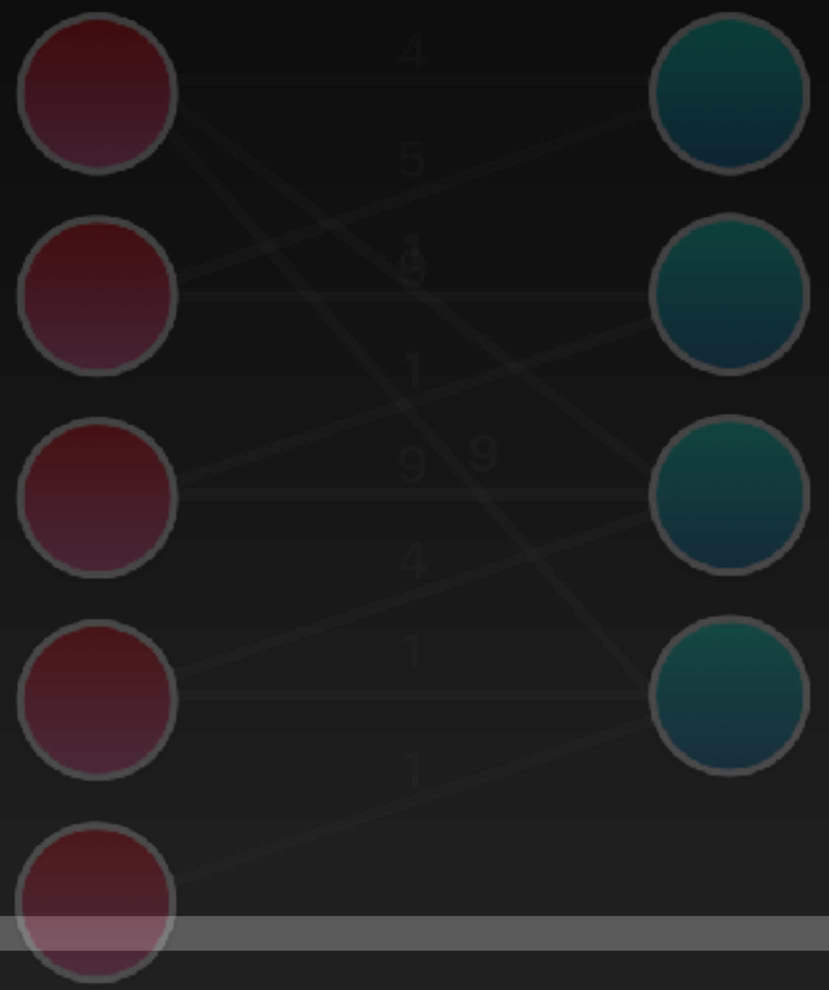
Sources: movielens.org, vecteezy.com

Desirable Properties of a Model



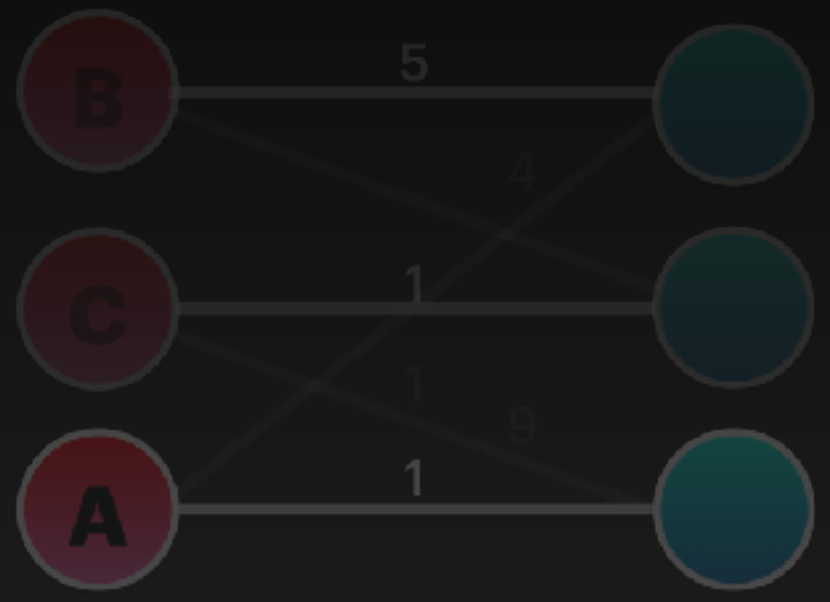
1 Works for different Graph Sizes





Permutation-equivariant 2

Zaheer, Manzil, et al. "Deep Sets." NeurIPS (2017).



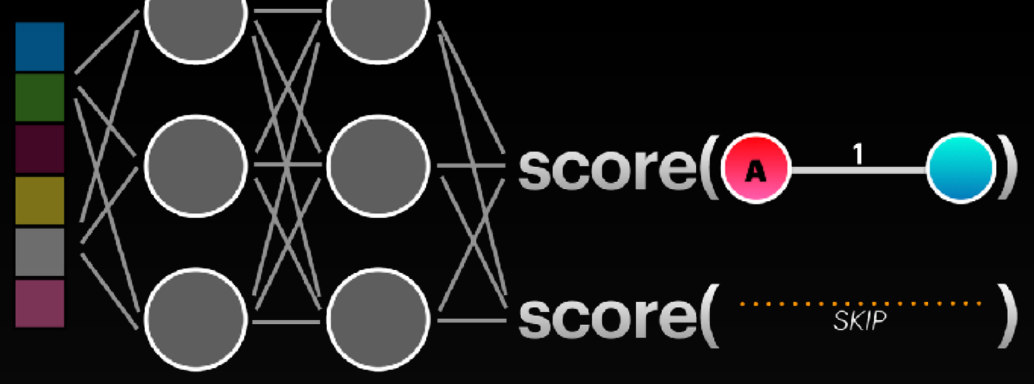
3 Accounting for History

$1, \ldots, t$

- Avg./Variance/Min/Max of weights in current matching
- How many skips?
- Average U-node degree, ...

Sources: movielens.org, vecteezy.com

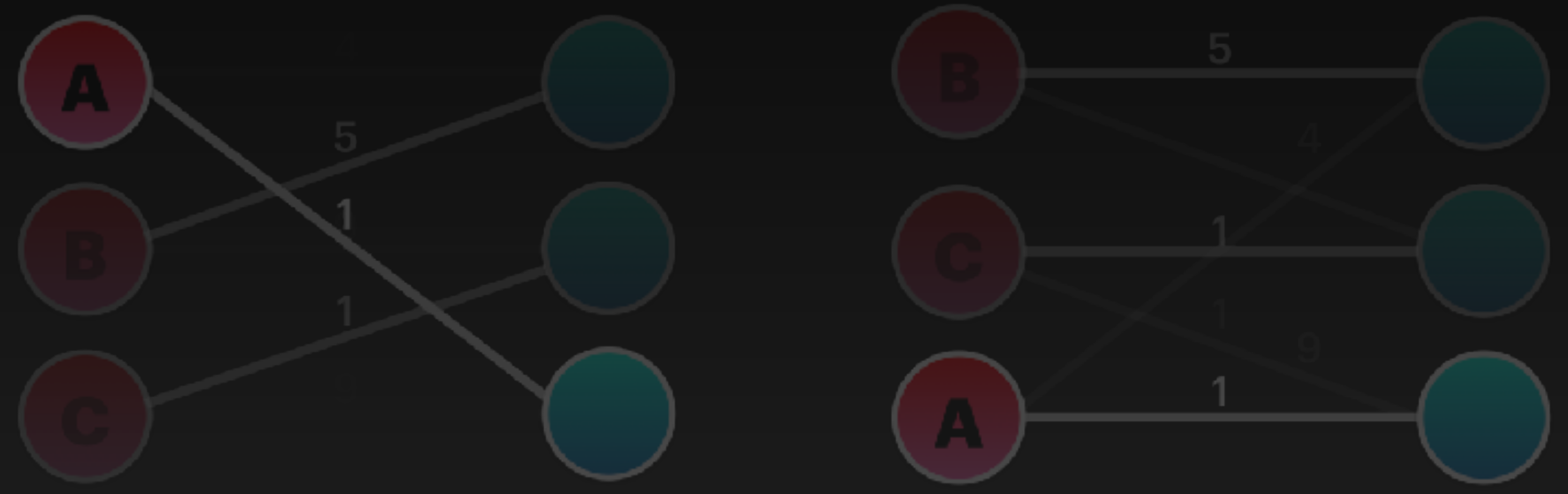
Desirable Properties of a Model



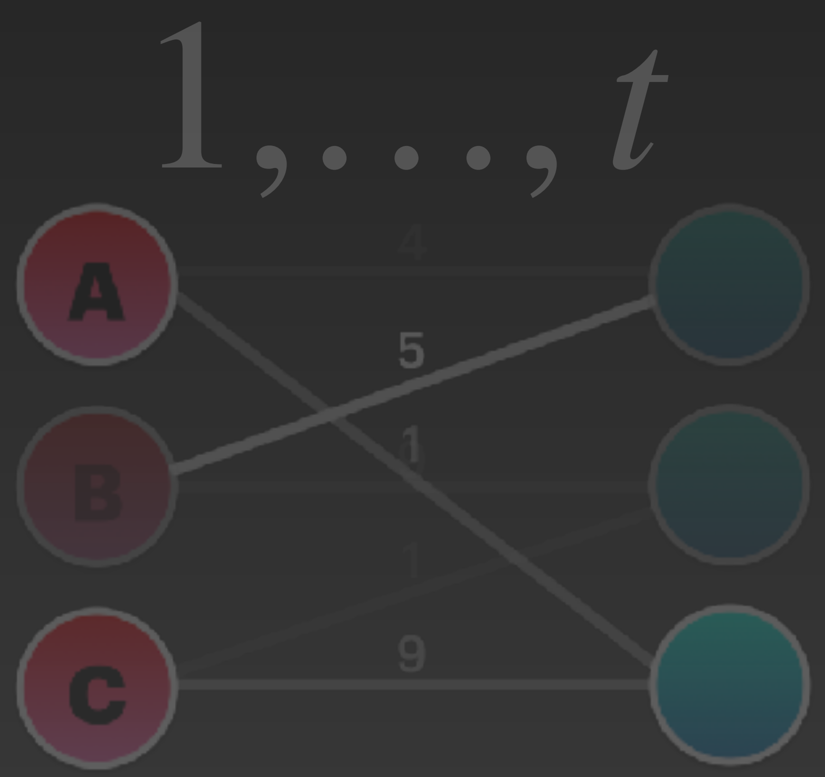
1 Works for different Graph Sizes

Permutation-equivariant 2

Zaheer, Manzil, et al. "Deep Sets." NeurIPS (2017).

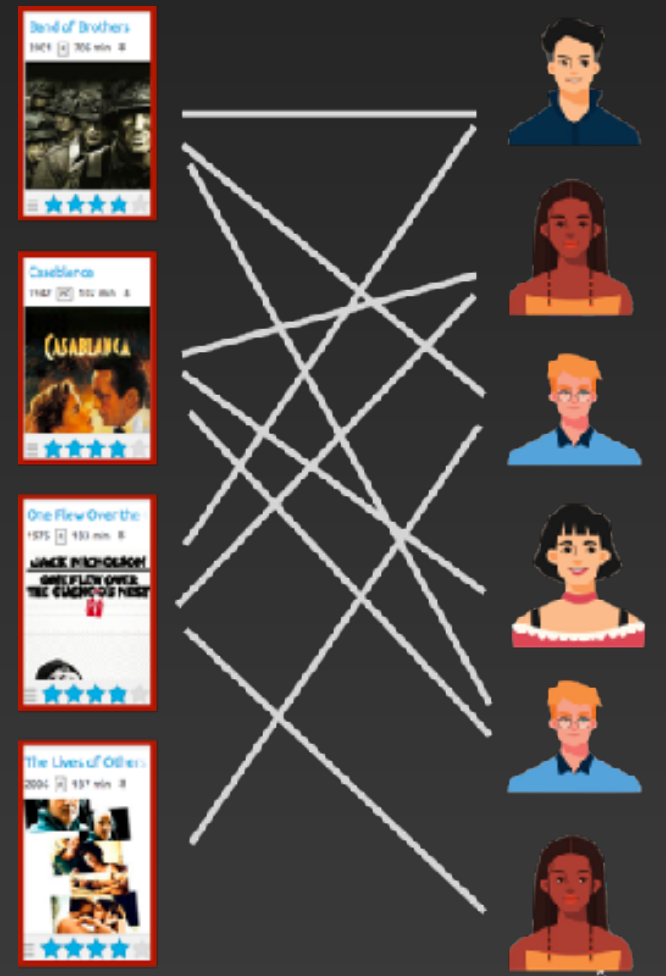


3 Accounting for History



- Avg./Variance/Min/Max of weights in current matching
- How many skips?
- Average U-node degree, ...

Accounting for Node Features 4



Sources: movielens.org, vecteezy.com

ff
a basic feed-forward model

ff

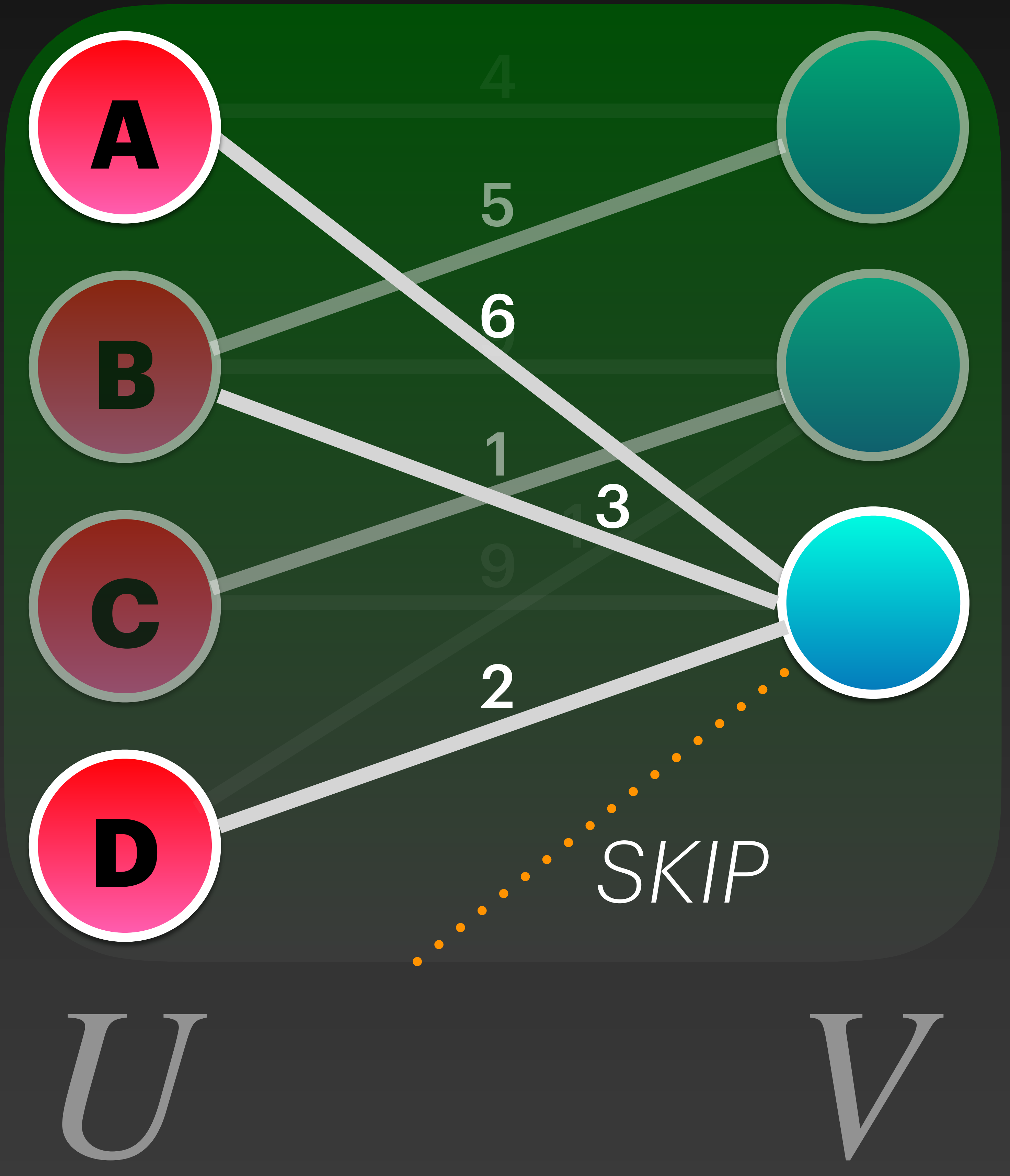
a basic feed-forward model

- Number of learnable parameters depends on |U| —> **assumes fixed graph size**
- **Not equivariant to permutations** in |U|

inv-ff

parameter sharing as key design principle

State t



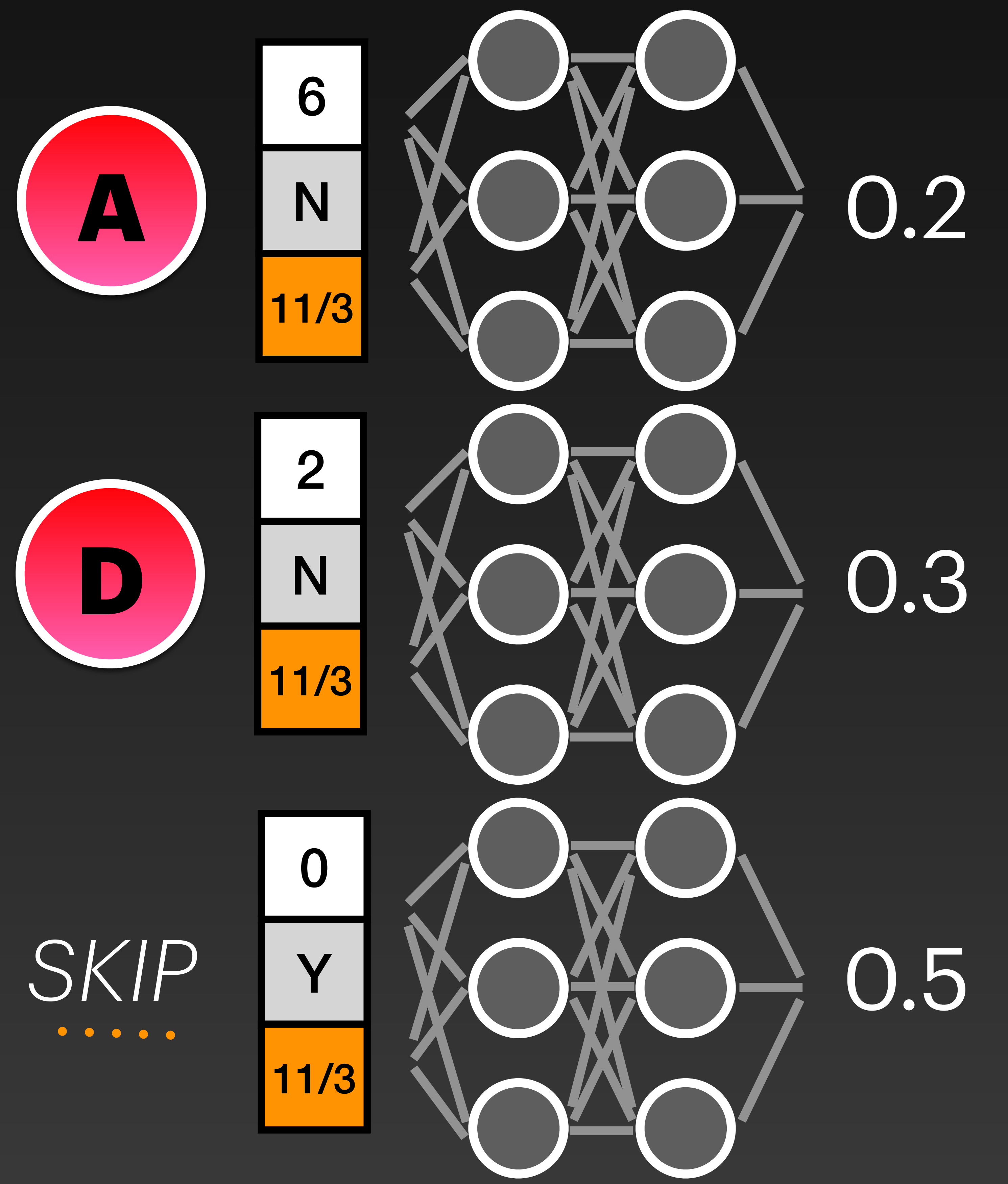
Features

6 Edge weight

Y SKIP action or not

11/3 Average weight of incoming edges

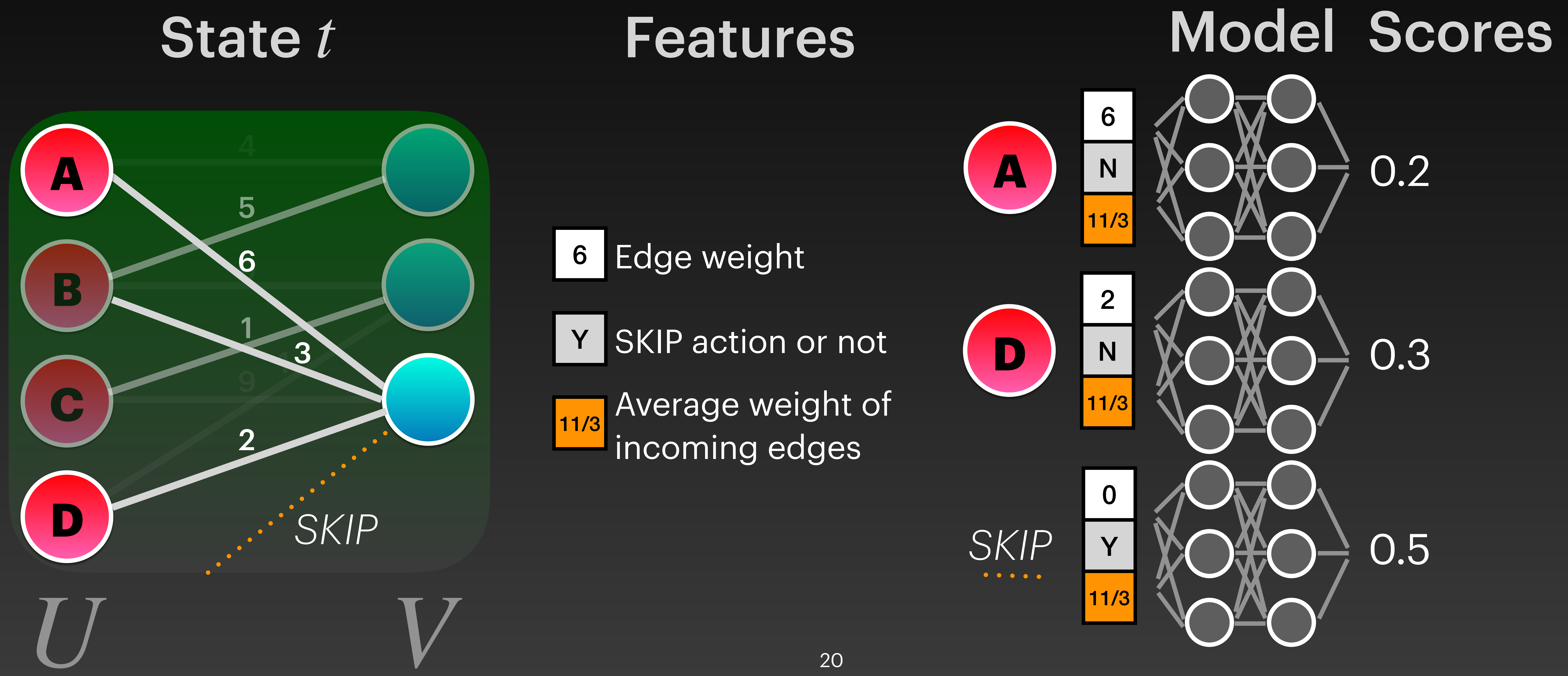
Model Scores



inv-ff

parameter sharing as key design principle

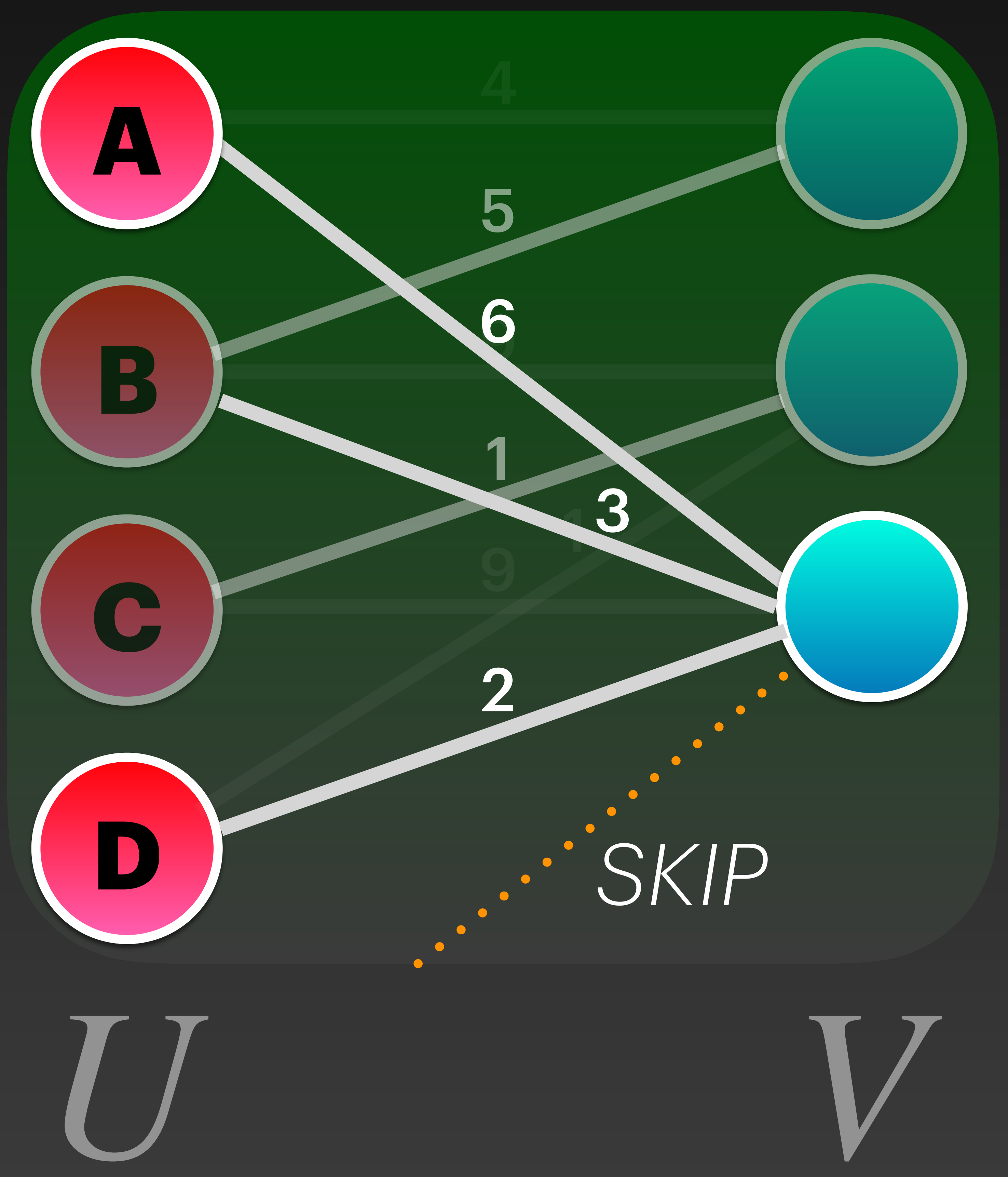
- **Same feed-forward network** is used in parallel for all valid actions (edges or “skip”)
- **Works with any graph size** (i.e., |U|)
- **Permutation-equivariant**



inv-ff-hist

capturing history

State t



Features

6 Edge weight

Y SKIP action or not

11/3 Average weight of incoming edges

Average/variance of weights per U node up to t

...

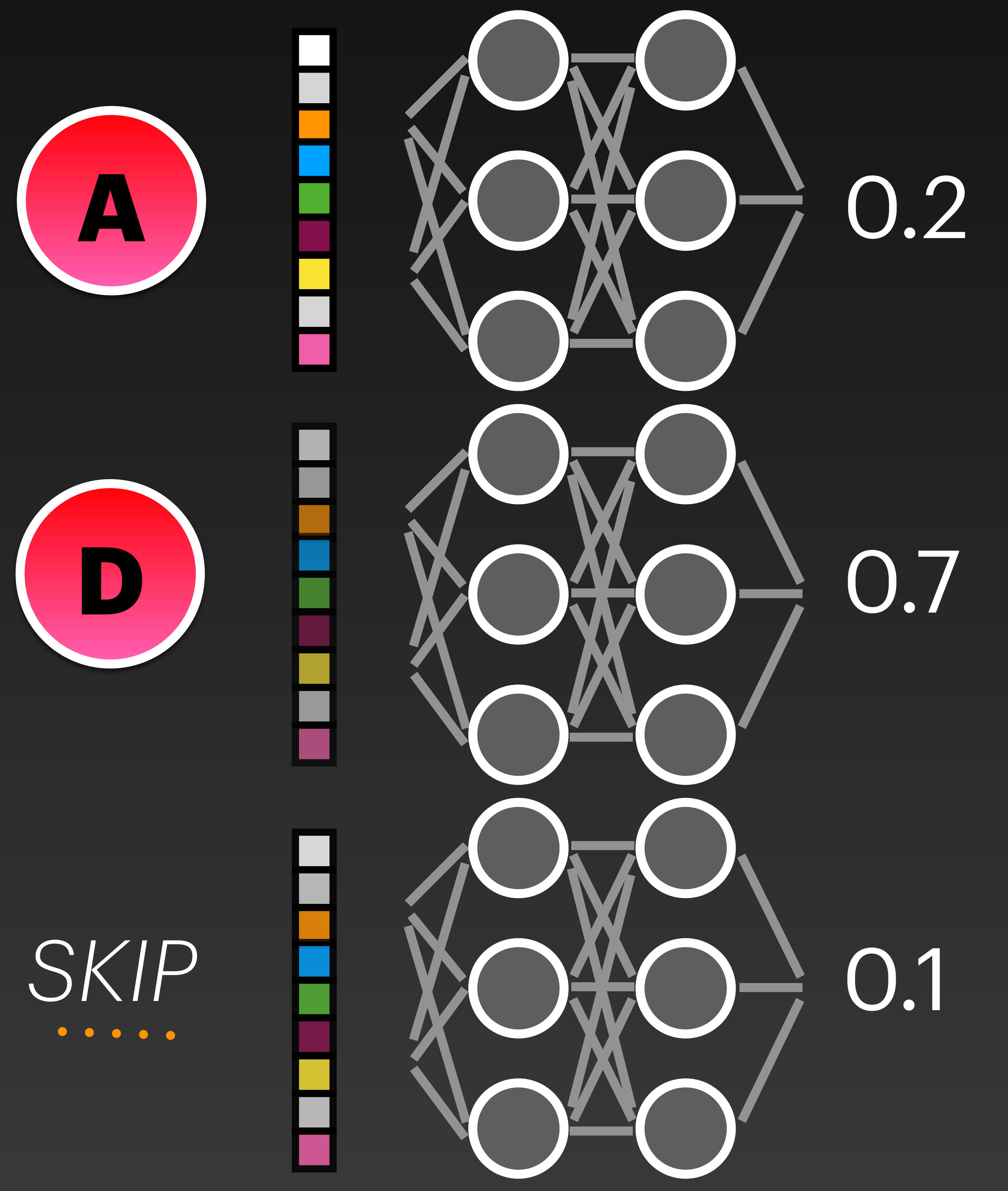
Average degree of U nodes up to t

...

Min/Max/Avg/Var of weights in current matching

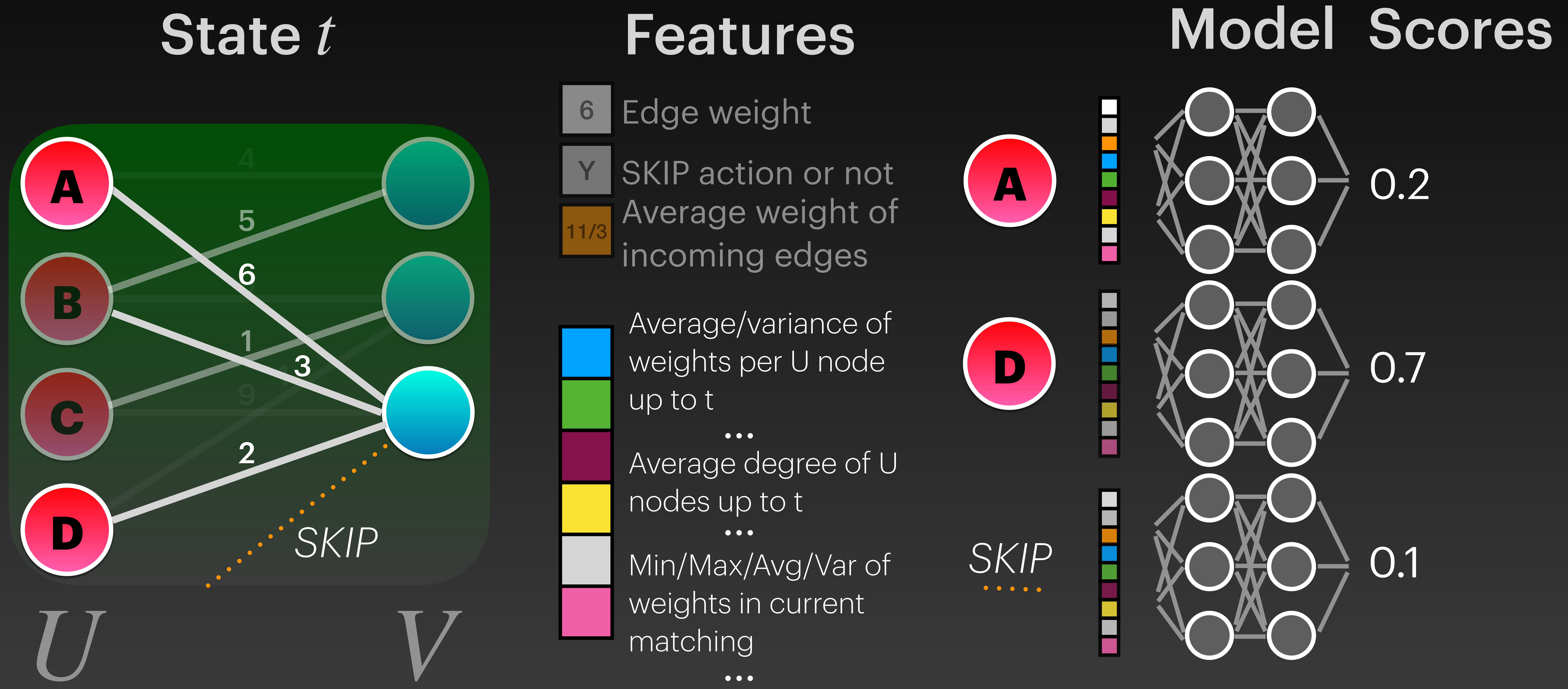
...

Model Scores



inv-ff-hist
capturing history

- **History features** at the levels of the Graph, current matching, and incoming V node
- Graph+Matching features are the same, serve as **additional context** for all actions



Graph Neural Nets

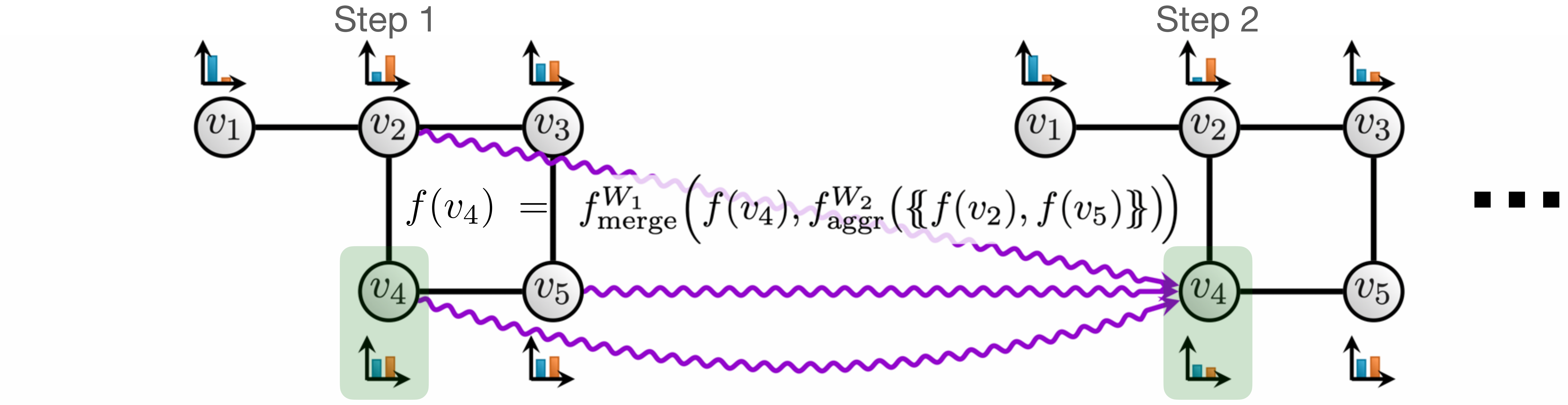


Figure 3: Illustration of the neighborhood aggregation step of a GNN around node v_4.

Combinatorial optimization and reasoning with graph neural networks.
Q. Cappart, D. Chételat, E.B. Khalil, A. Lodi, C. Morris, P. Veličković.
https://arxiv.org/abs/2102.09544

Graph Neural Nets

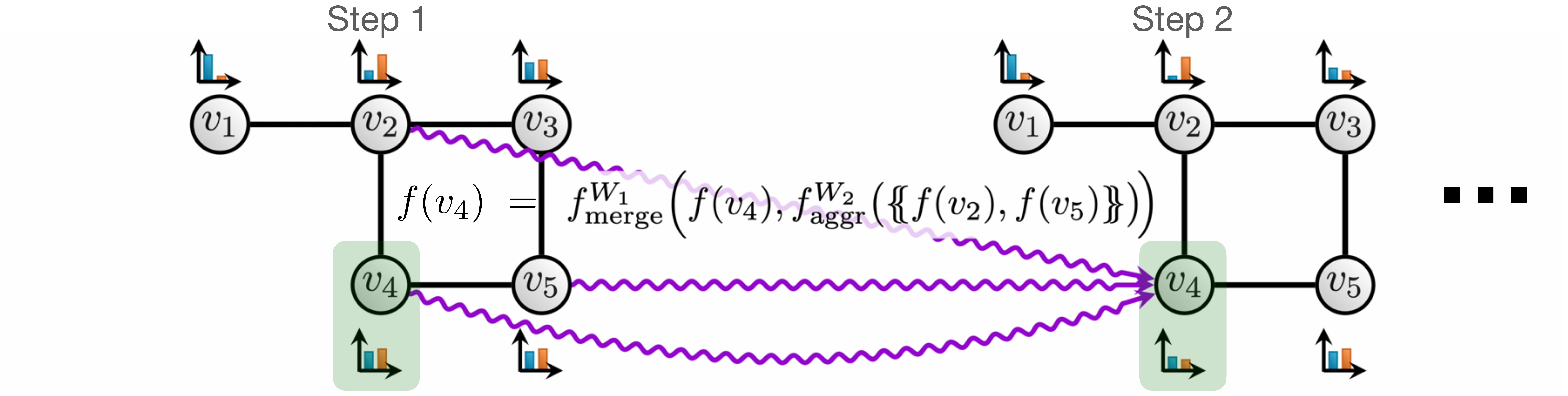


Figure 3: Illustration of the neighborhood aggregation step of a GNN around node v_4.

$$f^{(t)}(v) = \sigma\Big(f^{(t-1)}(v) \cdot W_1 \; + \sum_{w \in N(v)} f^{(t-1)}(w) \cdot W_2 \;\Big)$$

Combinatorial optimization and reasoning with graph neural networks.
Q. Cappart, D. Chételat, E.B. Khalil, A. Lodi, C. Morris, P. Veličković.
https://arxiv.org/abs/2102.09544

Graph Neural Nets

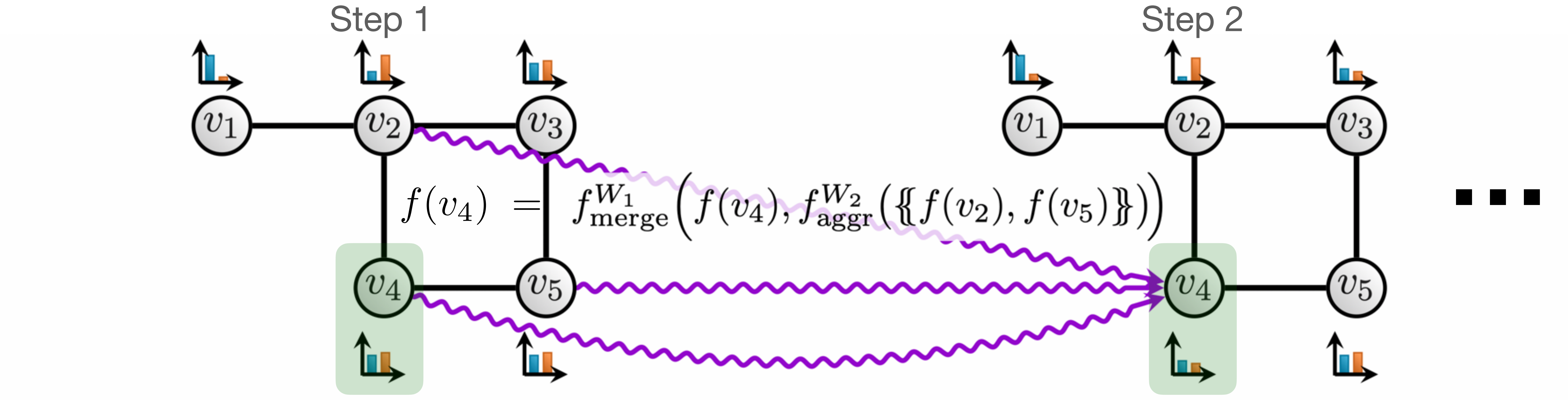


Figure 3: Illustration of the neighborhood aggregation step of a GNN around node v_4.

- **Invariant/Equivariant** to node permutations
- Model parameters (W_1, W_2) are shared
  —> **applies to graphs of arbitrary size**
- **Expressive** local/global features are learned through non-linear layers

$$f^{(t)}(v) = \sigma\left(f^{(t-1)}(v) \cdot W_1 + \sum_{w \in N(v)} f^{(t-1)}(w) \cdot W_2\right)$$

Combinatorial optimization and reasoning with graph neural networks.
Q. Cappart, D. Chételat, E.B. Khalil, A. Lodi, C. Morris, P. Veličković.
https://arxiv.org/abs/2102.09544

gnn-hist

capturing graph structure

gnn-hist

capturing graph structure

- Use MPNN (Gilmer et al., 2017) to encode graph; MLP decodes each candidate edge
- **Most flexible** model but **harder to train**
- Naturally captures **node dependencies**

Experimental Setup

1

2

3

4

Experimental Setup

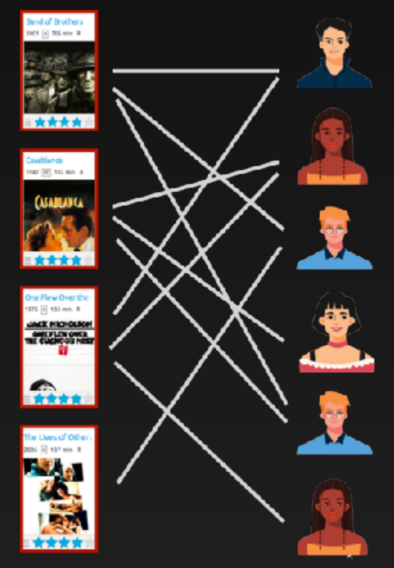
1

Two Matching Problems

E-OBM: Edge-Weighted OBM

OSBM: Online Submodular Bipartite Matching

Dickerson et al., AAAI-19



2

3

4

Experimental Setup

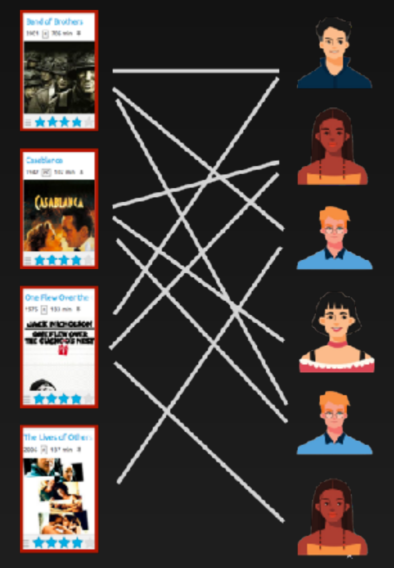
1

Two Matching Problems

E-OBM: Edge-Weighted OBM

OSBM: Online Submodular Bipartite Matching

Dickerson et al., AAAI-19



Two Synthetic Datasets

2

ER: Erdos-Renyi graphs, varying |U|, |V|, sparsity levels

E-OBM

BA: Barabasi-Albert graphs, varying |U|, |V|, weight distributions

3

4

Experimental Setup

1

Two Matching Problems

E-OBM: Edge-Weighted OBM

OSBM: Online Submodular Bipartite Matching

Dickerson et al., AAAI-19

2

Two Synthetic Datasets

ER: Erdos-Renyi graphs, varying |U|, |V|, sparsity levels

BA: Barabasi-Albert graphs, varying |U|, |V|, weight distributions

E-OBM

3

Two Real Datasets

gMission: crowdsourcing, workers (U) to tasks (V) **E-OBM**

Chen et al., VLDB-14

MovieLens: movie (U) to user (V), weights favour diverse genre recs. **OSBM**

Features: Movie genres, User age/sex/occupation

Dickerson et al., AAAI-19

4

Experimental Setup

1

Two Matching Problems

E-OBM: Edge-Weighted OBM

OSBM: Online Submodular Bipartite Matching

Dickerson et al., AAAI-19

Two Synthetic Datasets

2

ER: Erdos-Renyi graphs, varying |U|, |V|, sparsity levels

BA: Barabasi-Albert graphs, varying |U|, |V|, weight distributions

E-OBM

3

Two Real Datasets

gMission: crowdsourcing, workers (U) to tasks (V)

E-OBM

Chen et al., VLDB-14

MovieLens: movie (U) to user (V), weights favour diverse genre recs.

OSBM

Features: Movie genres, User age/sex/occupation

Dickerson et al., AAAI-19

Other Details

4

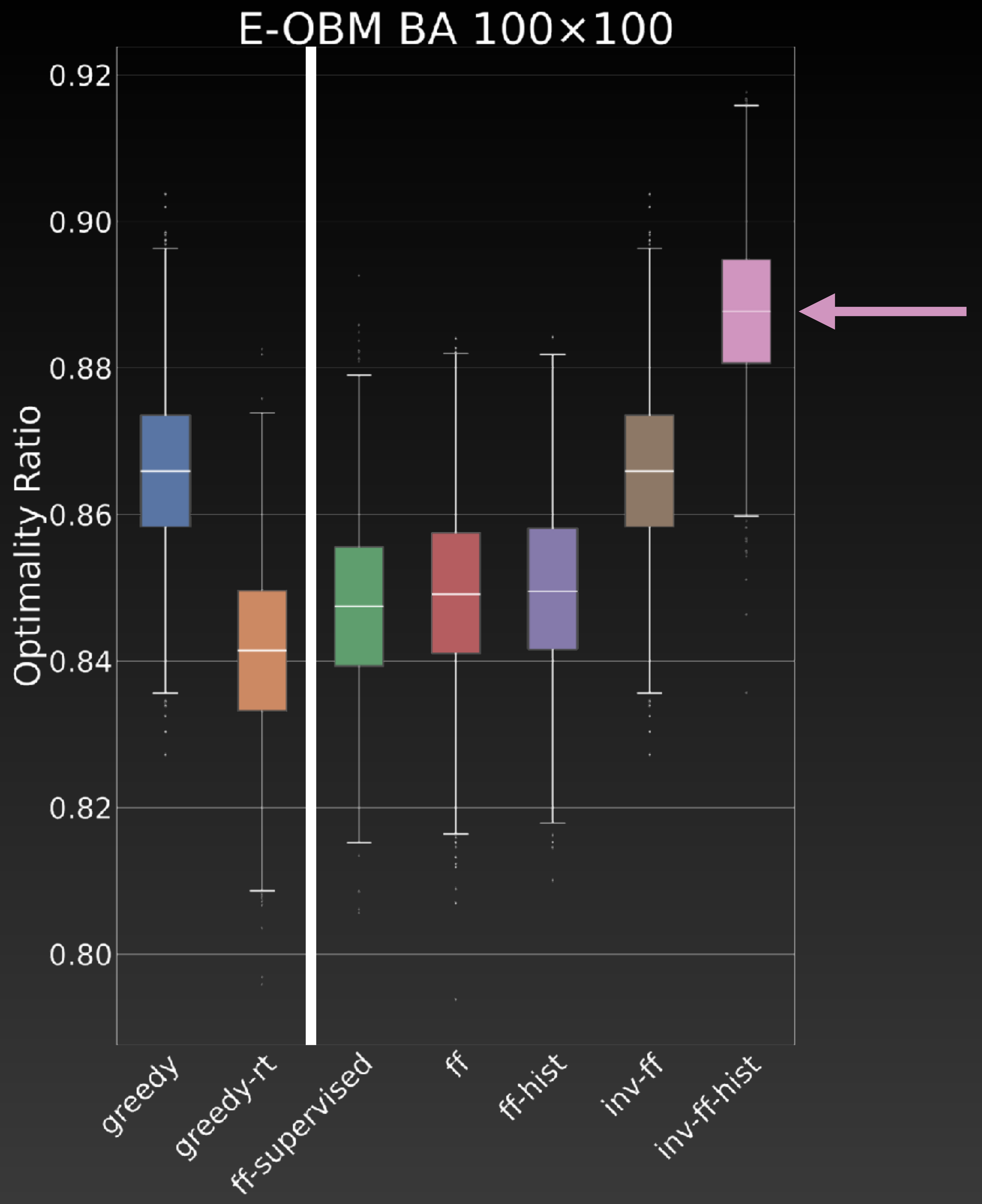
Policy Gradient (REINFORCE) for RL training

Train/Validation/Test split

$$\mathop{\mathbb{E}}_{(G,\ \text{order of}\ V)\sim\mathcal{D}}\left[\frac{\mathrm{ALG}(G)}{\mathrm{OPT}(G)}\right]$$

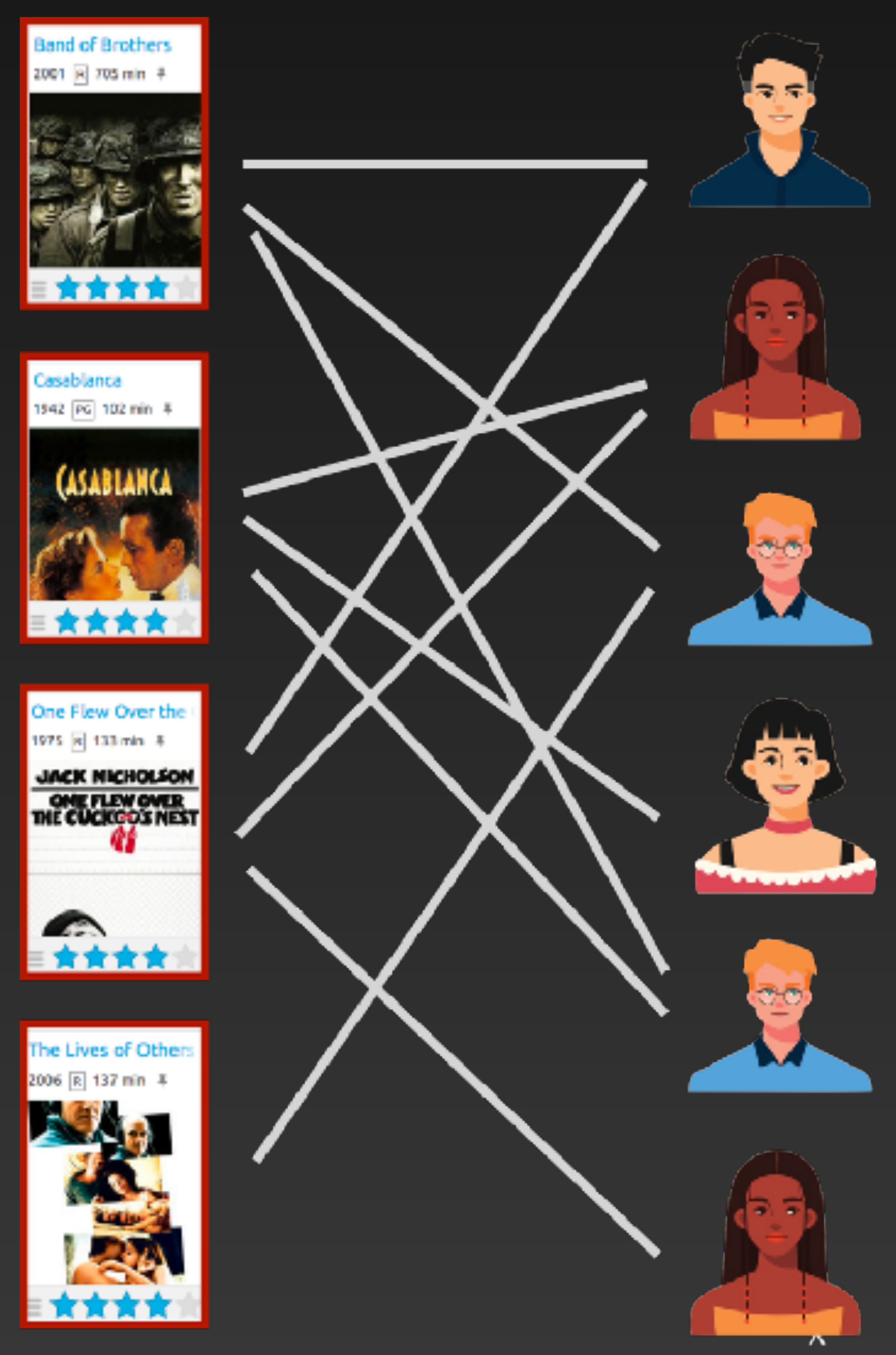
Distributions of the Optimality Ratios for E-OBM on BA graphs

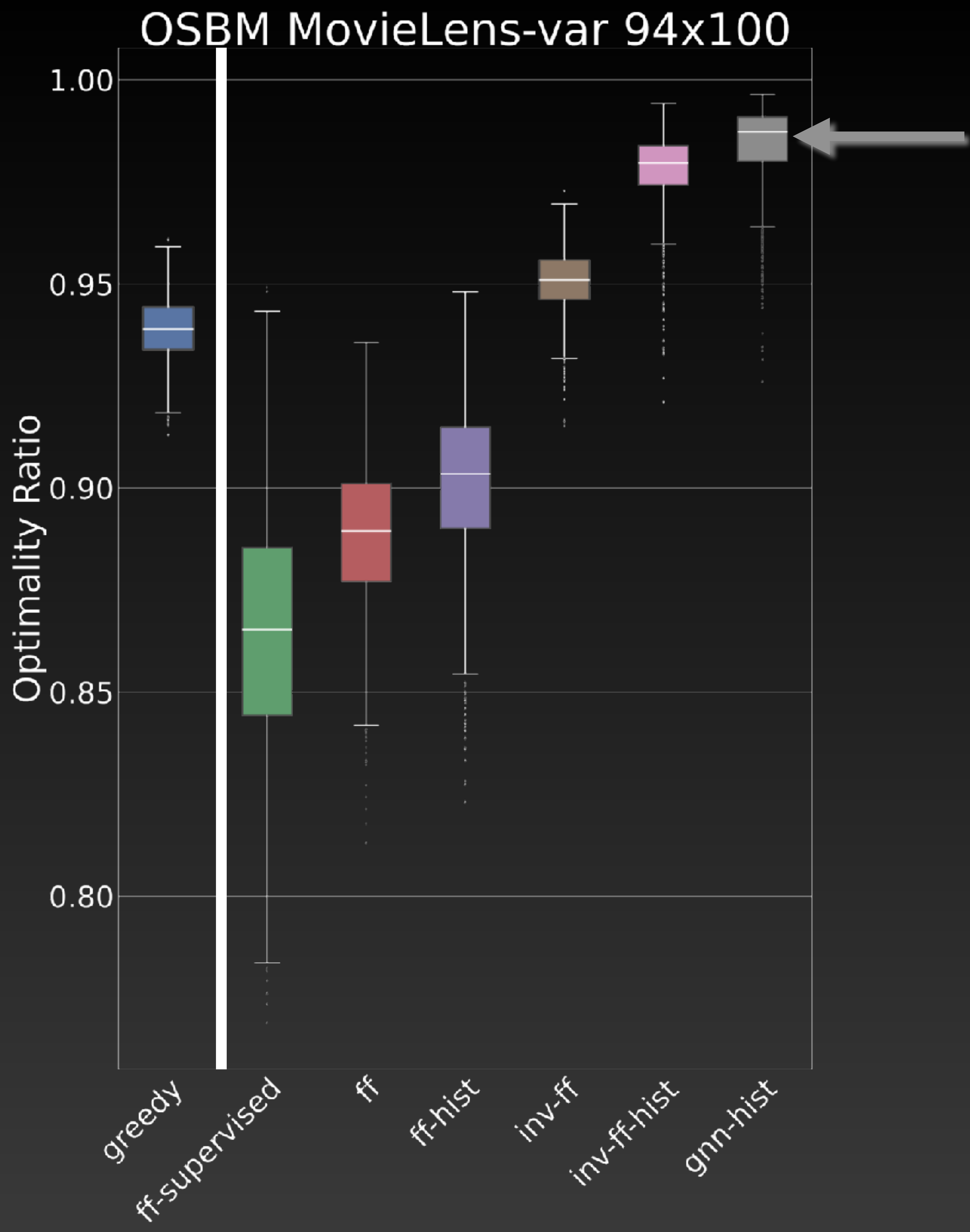
Higher is better



Distributions of the Optimality Ratios for OSBM on MovieLens graphs

Higher is better





Takeaways #1

Published in Transactions on Machine Learning Research (10/2022)

Deep Policies for Online Bipartite Matching: A Reinforcement Learning Approach

Mohammad Ali Alomrani *mohammad.alomrani@mail.utoronto.ca*
Department of Electrical & Computer Engineering
University of Toronto

Reza Moravej *mreza.moravej@mail.utoronto.ca*
Department of Mechanical & Industrial Engineering
University of Toronto

Elias B. Khalil *khalil@mie.utoronto.ca*
Department of Mechanical & Industrial Engineering
SCALE AI Research Chair in Data-Driven Algorithms for Modern Supply Chains
University of Toronto

Reviewed on OpenReview: https://openreview.net/forum?id=mbwm7NdkpO

arXiv:2109.10380

code linked in paper

- **<u>Online</u> combinatorial optimization** is very amenable to RL.
- Careful combination of **feature engineering + MLP-based architecture** design seems to go a long way!
- **GNN provides some edge when the data is rich and complex.**

Learning Combinatorial Optimization over Graphs

Our first attempt at RL for algorithm design

Joint work with Hanjun Dai (co-first author), Yuyu Zhang, Bistra Dilkina, Le Song

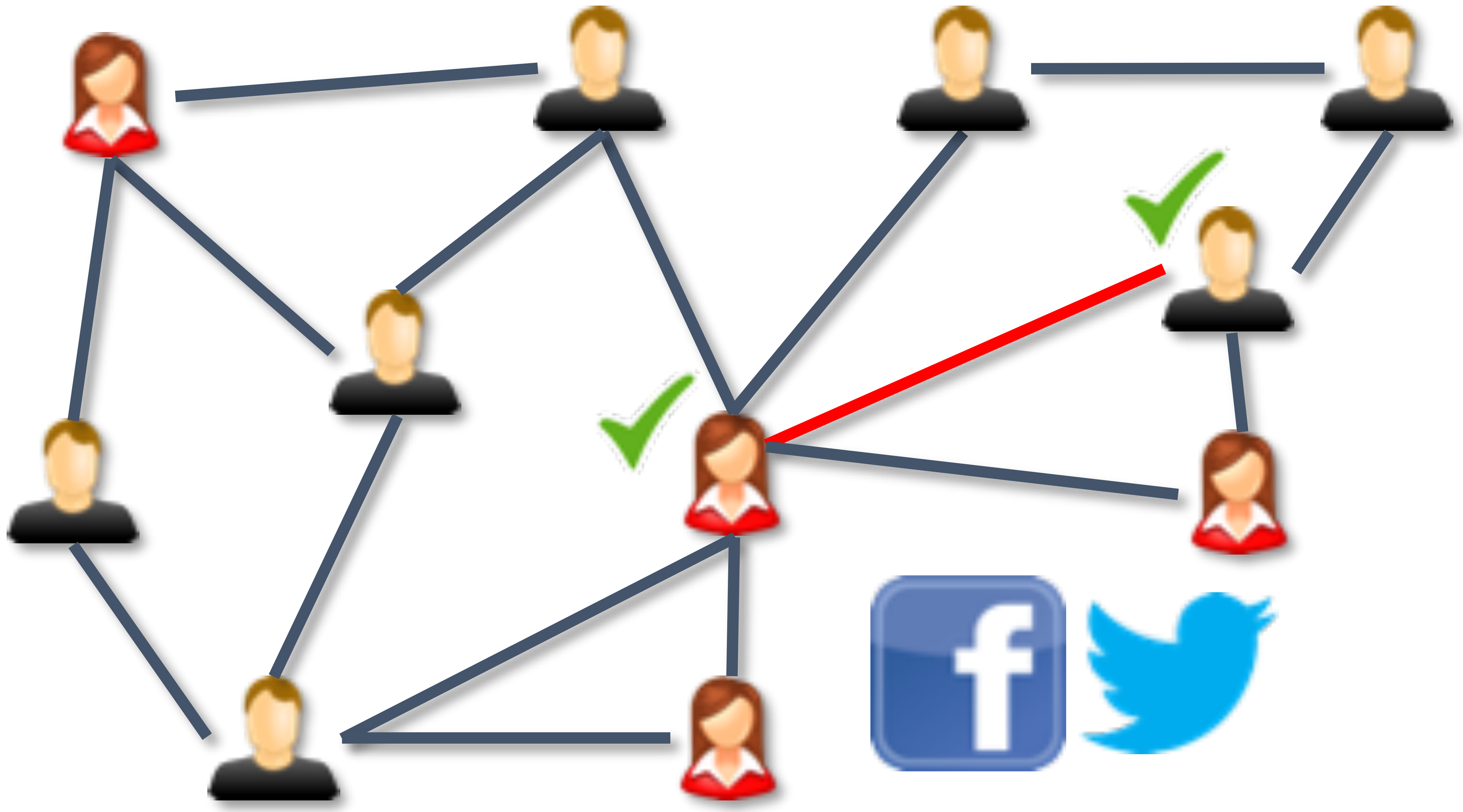
NeurIPS 2017

arXiv:2109.10380

Greedy Graph Optimization

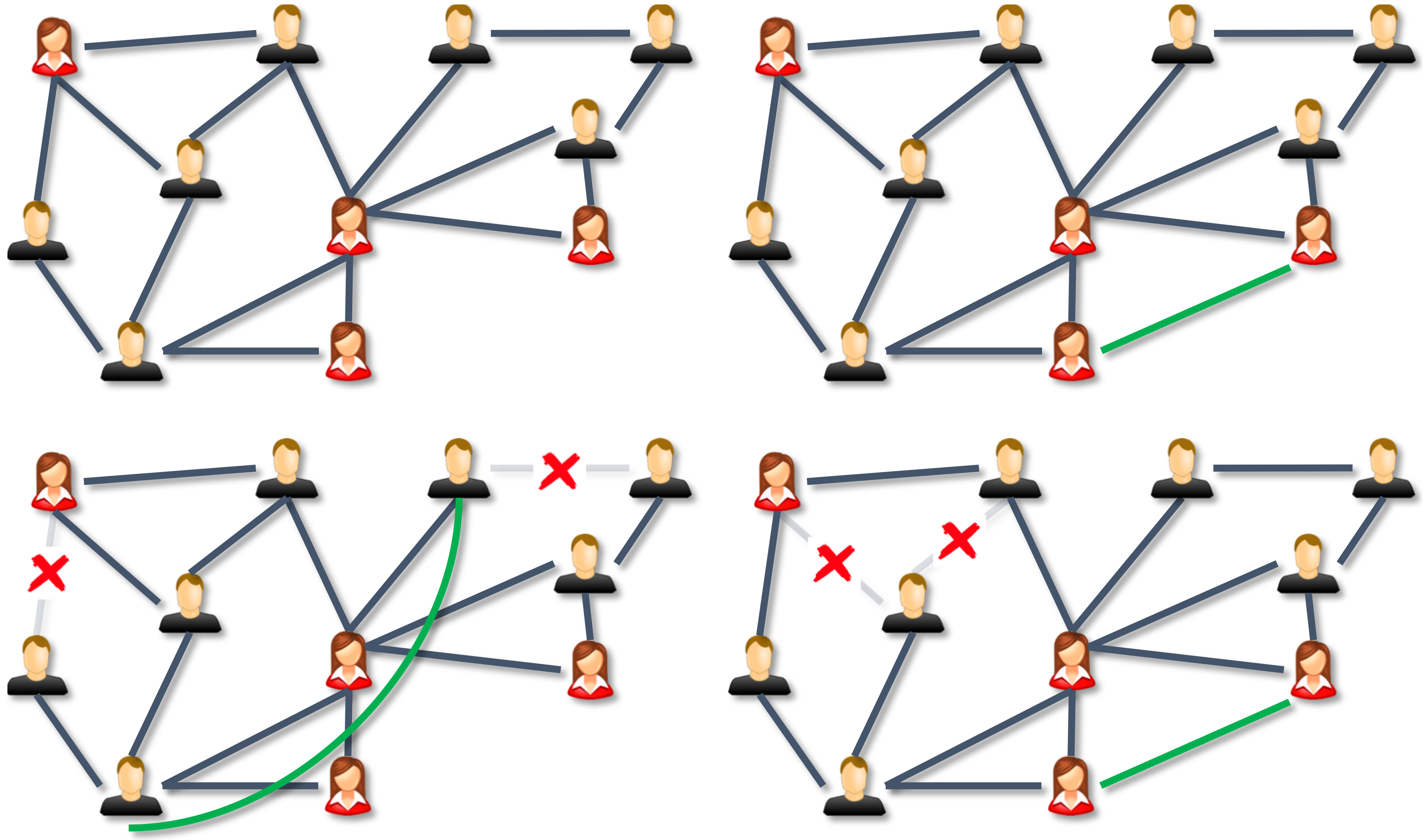
Minimum Vertex Cover
Find smallest vertex subset such that each edge is covered

2-Approximation:
Greedily add vertices of edge with **max degree sum**



Problem Statement

Given a **graph optimization problem G** and a **distribution $\mathcal{D}$** of problem instances, can we **learn better greedy heuristics** that generalize to unseen instances from $\mathcal{D}$?



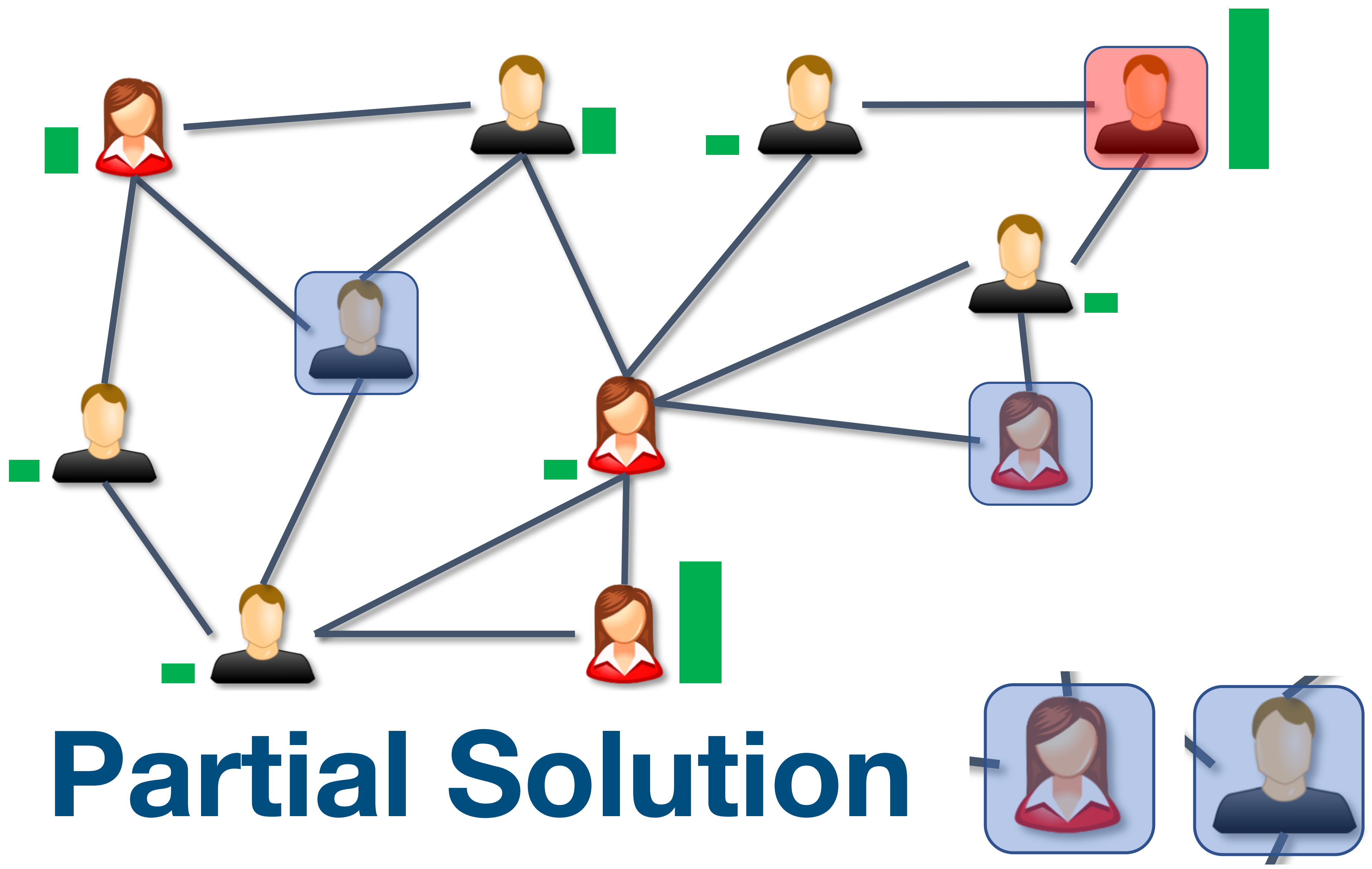
$\sim \mathcal{D}$

Challenge #1: How to Learn

Possible approach: **Supervised learning**

- Given a partial solution, predict next vertex to add to solution
- **Data**: collect (partial solution, next vertex) pairs

  features **label**
- **Task**: multi-class classification
- [Vinyals, et al., NIPS 2015]: a smarter approach with recurrent neural networks



Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." *NIPS*. 2015.

Pointer Networks

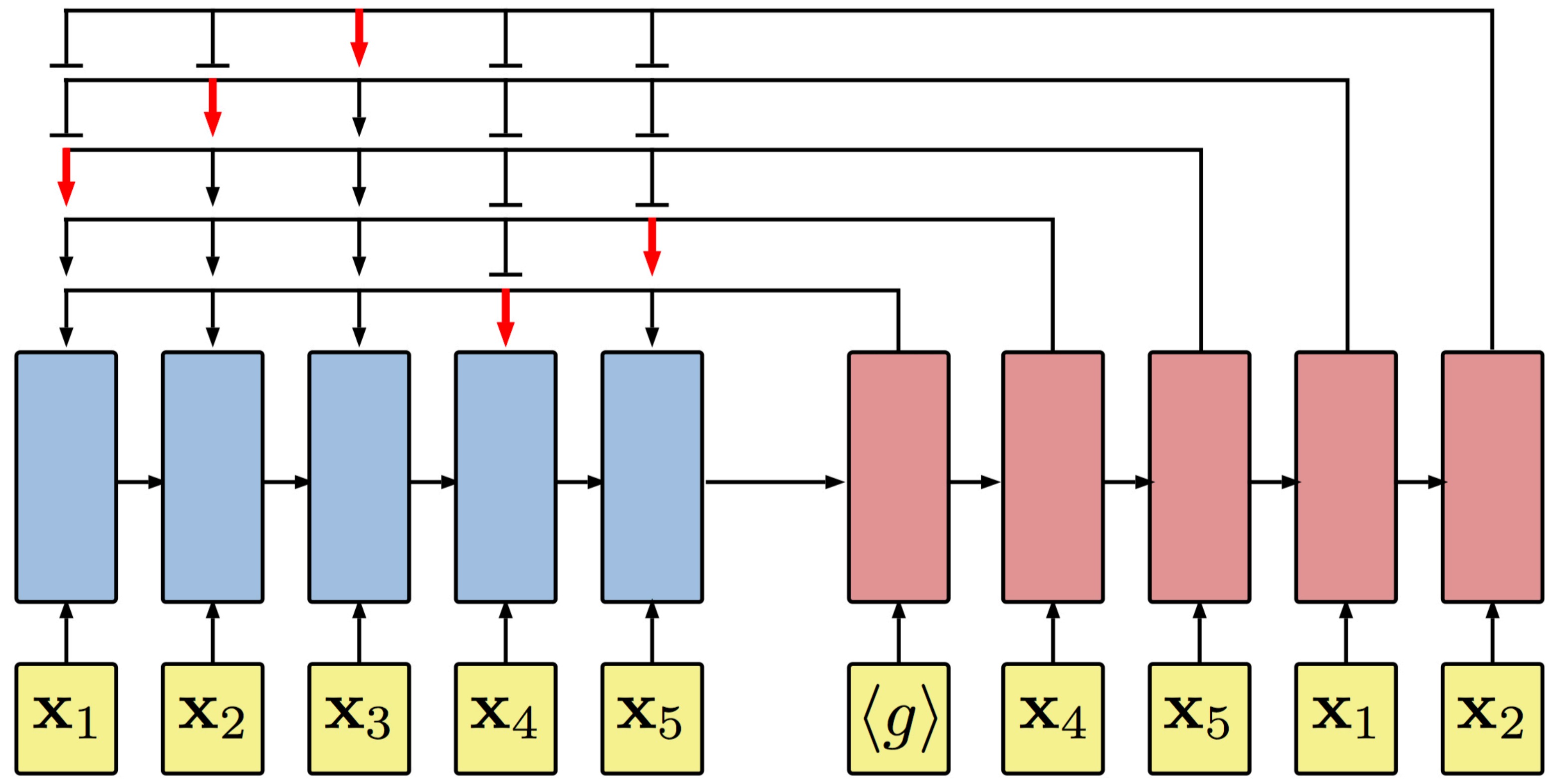
Challenge #1: How to Learn

- For an instance P, desired output is C^P
- Supervised Learning with Pointer-Networks:

$$\theta^* = \arg\max_{\theta} \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \log \underbrace{p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta)}$$

Probability of outputting C^P given:

- Instance P
- Parameters θ



Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." *NIPS*. 2015.

Pointer Networks

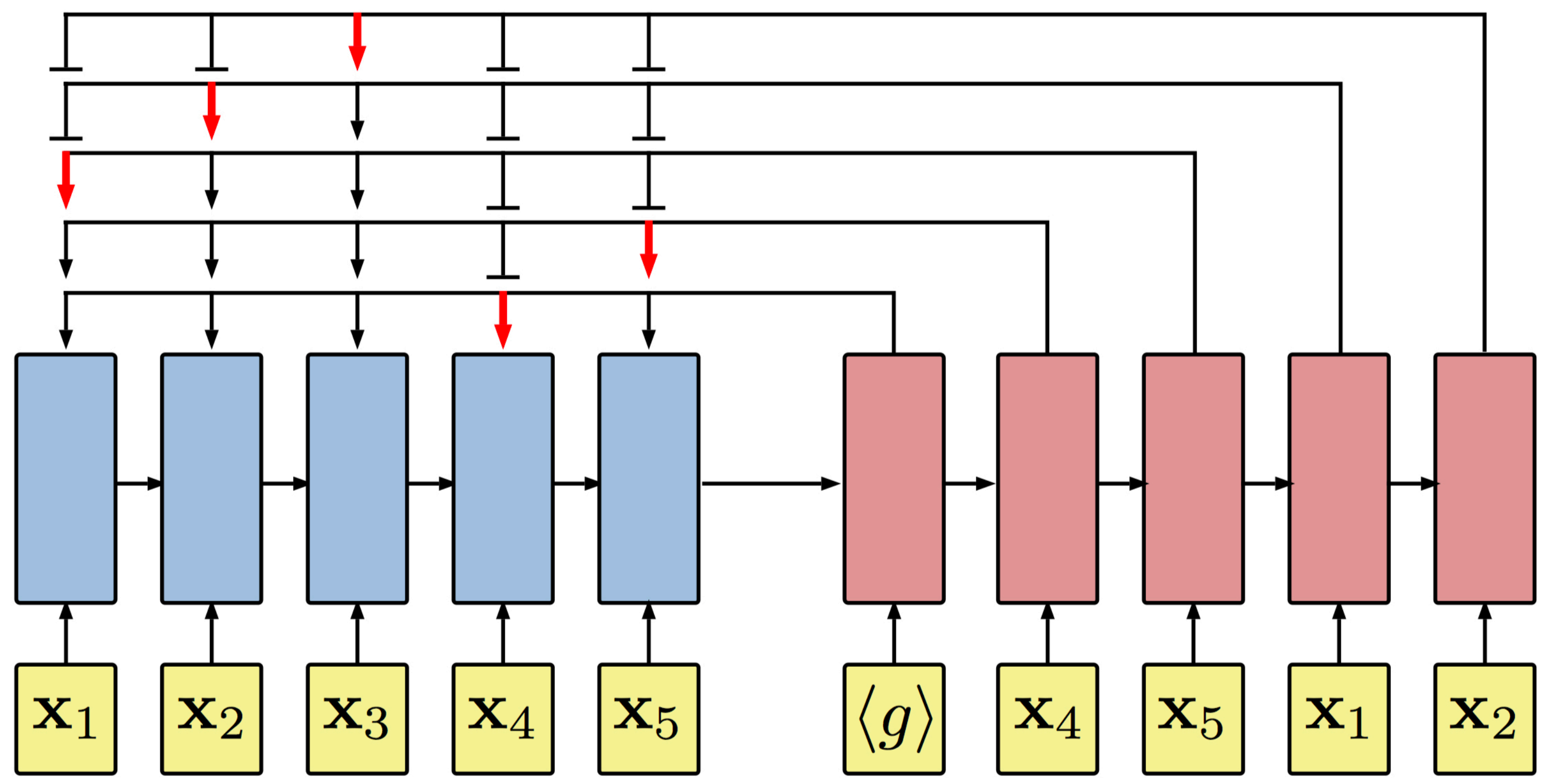
Challenge #1: How to Learn

- For an instance P, desired output is C^P
- Supervised Learning with Pointer-Networks:

$$\theta^* = \arg\max_{\theta} \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \underbrace{\log p(\mathcal{C}^{\mathcal{P}}|\mathcal{P};\theta)}$$

Probability of outputting C^P given:

- Instance P
- Parameters θ



PROBLEM

Supervised learning → Need to compute good/optimal solutions to NP-Hard problems in order to learn!!

Reinforcement Learning as Alternative

A Reinforcement Learning Approach to Job-shop Scheduling

Wei Zhang
Department of Computer Science
Oregon State Unjversity
Corvalhs, Oregon 97331-3202
U S A

Thomas G Diettench
Department of Computer Science
Oregon State University
Corvalhs, Oregon 97 511-3202
U S A

IJCAI
1995

- **Optimization problem:** scheduling jobs under resource and precedence constraints
- **Domain:** NASA space shuttle processing
- **Key Idea:** learn to construct schedules by trial-and-error over a set of instances

RL with Deep Neural Nets

NEURAL COMBINATORIAL OPTIMIZATION WITH REINFORCEMENT LEARNING

ArXiv 2016

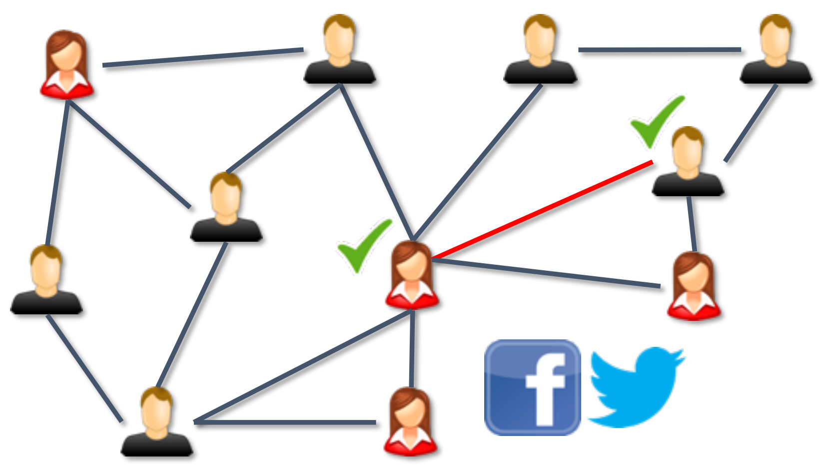
Irwan Bello*, Hieu Pham*, Quoc V. Le, Mohammad Norouzi, Samy Bengio
Google Brain
{ibello,hyhieu,qvl,mnorouzi,bengio}@google.com

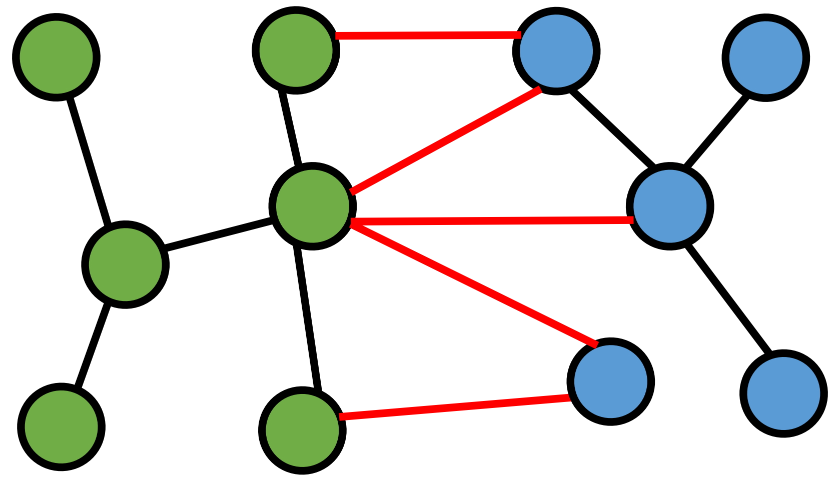
- **Key contribution:** using Pointer-Networks as model for Q-function
- **Improved TSP results:** can train on larger instances
- **Drawback:** may not fully exploit graph structure for sparse graphs (more details in a few slides)

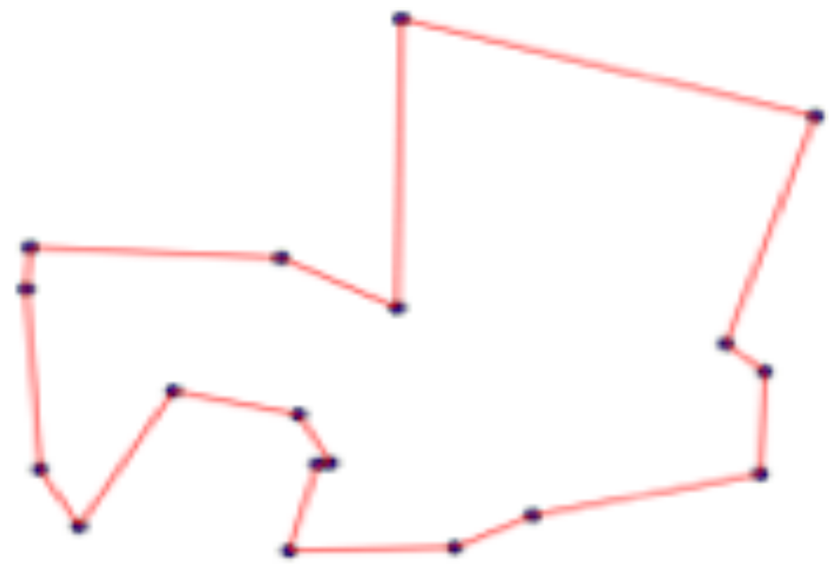
Learning Greedy Heuristics

Given: graph problem, family of graphs
Learn: a **scoring function** to **guide** a **greedy** algorithm

Problem	Minimum Vertex Cover	Maximum Cut	Traveling Salesman Problem
Domain	Social network snapshots	Spin glass models	Package delivery
Greedy operation	Insert nodes into cover	Insert nodes into subset	Insert nodes into sub-tour





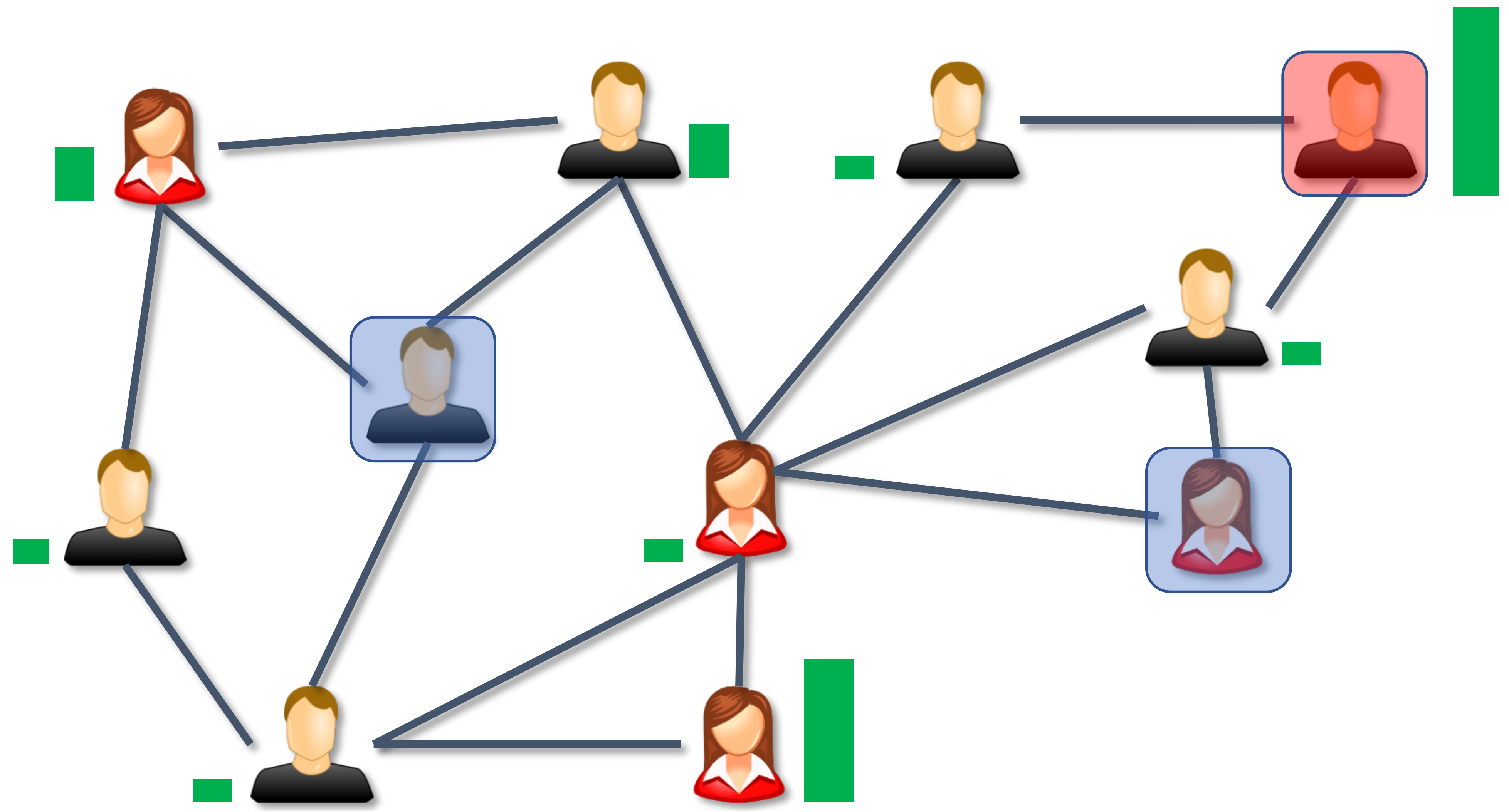


Reinforcement Learning

Greedy Algorithm		Reinforcement Learning
Partial solution	≡	State
Scoring function	≡	Q-function
Select best node	≡	Greedy Policy

Repeat until all edges are covered:

1. Compute node scores
2. Select best node w.r.t. score
3. Add best node to partial sol.

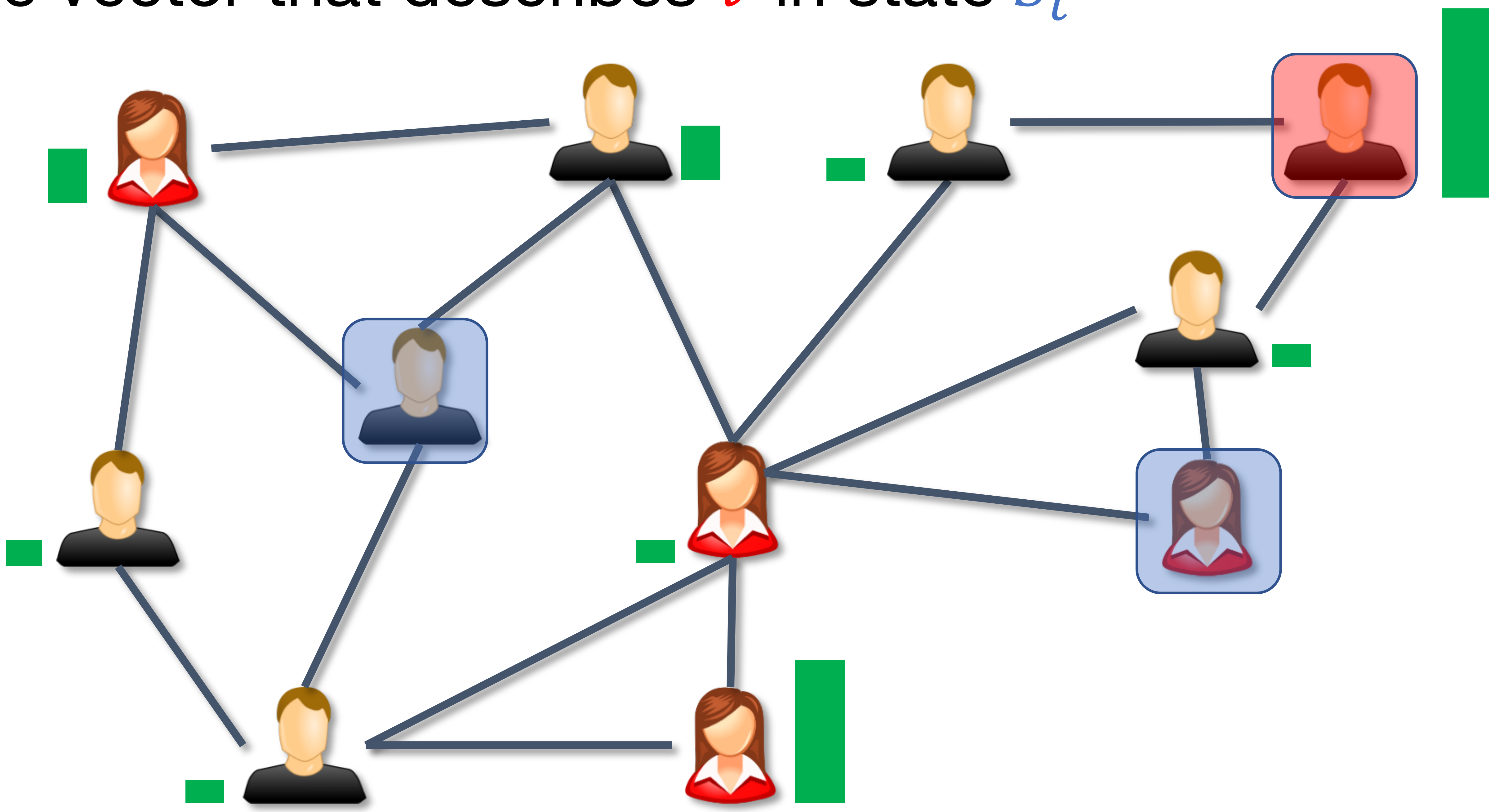


Partial Solution 

Challenge #2:

Representing Nodes

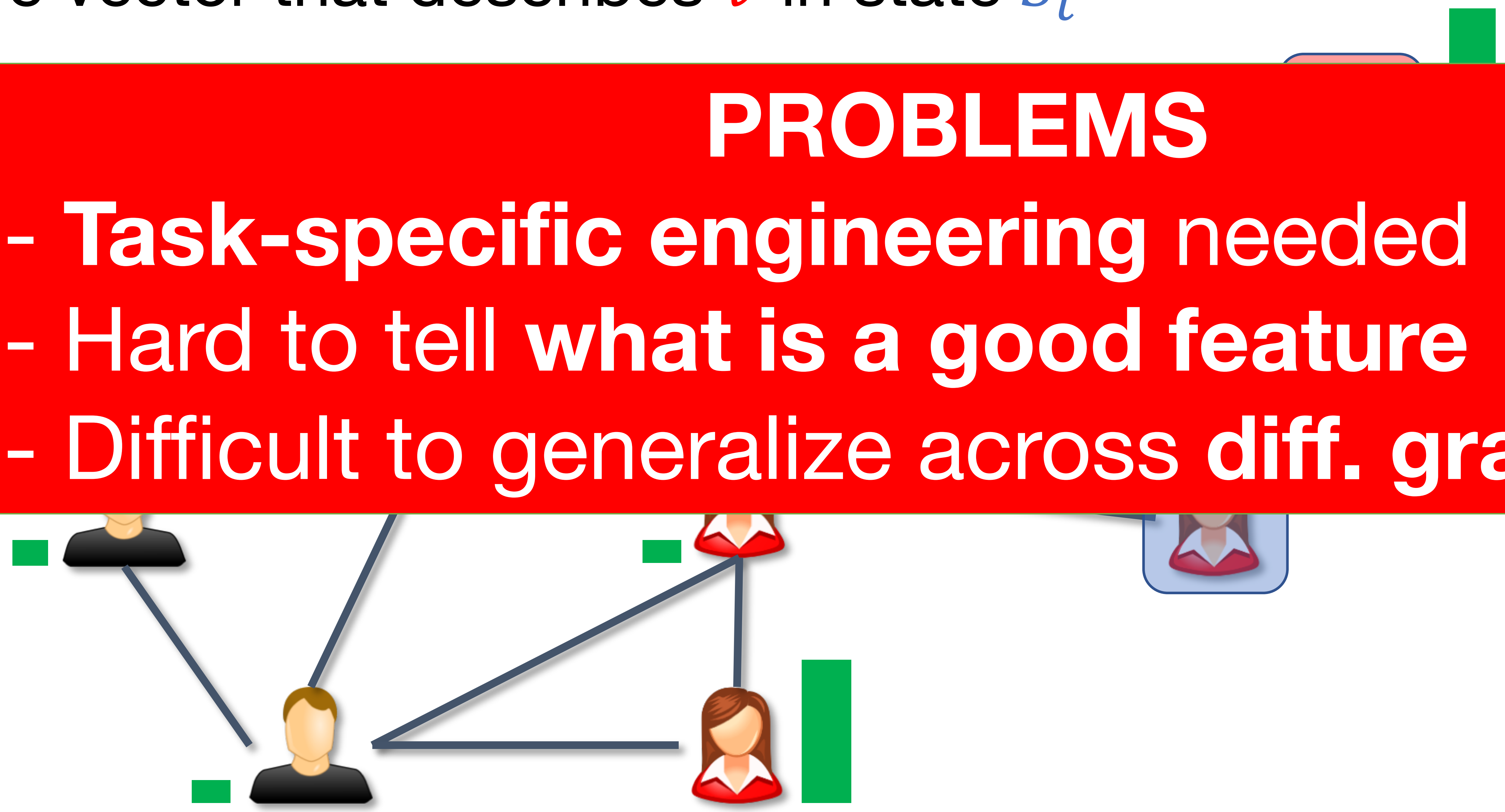
- **Action value function:** $\hat{Q}(S_t, v; \Theta)$
  - Estimate of goodness of vertex v in state S_t
- **Representation of v**
  - A feature vector that describes v in state S_t



Challenge #2:

Representing Nodes

- **Action value function:** $\hat{Q}(S_t, v; \Theta)$
  - Estimate of goodness of vertex v in state S_t
- **Representation of v**
  - A feature vector that describes v in state S_t



PROBLEMS

1- **Task-specific engineering** needed

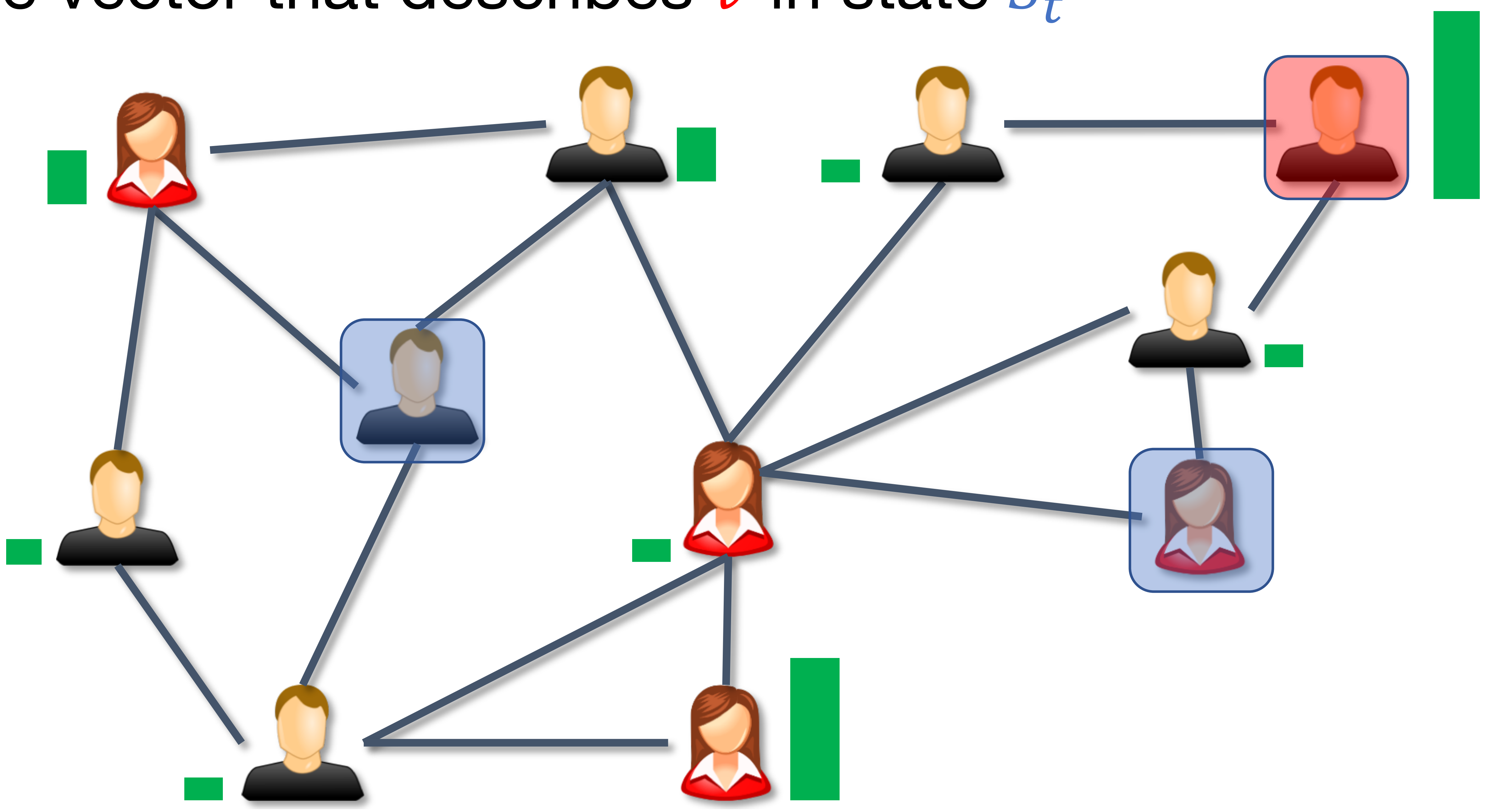
2- Hard to tell **what is a good feature**

3- Difficult to generalize across **diff. graph sizes**

Challenge #2:

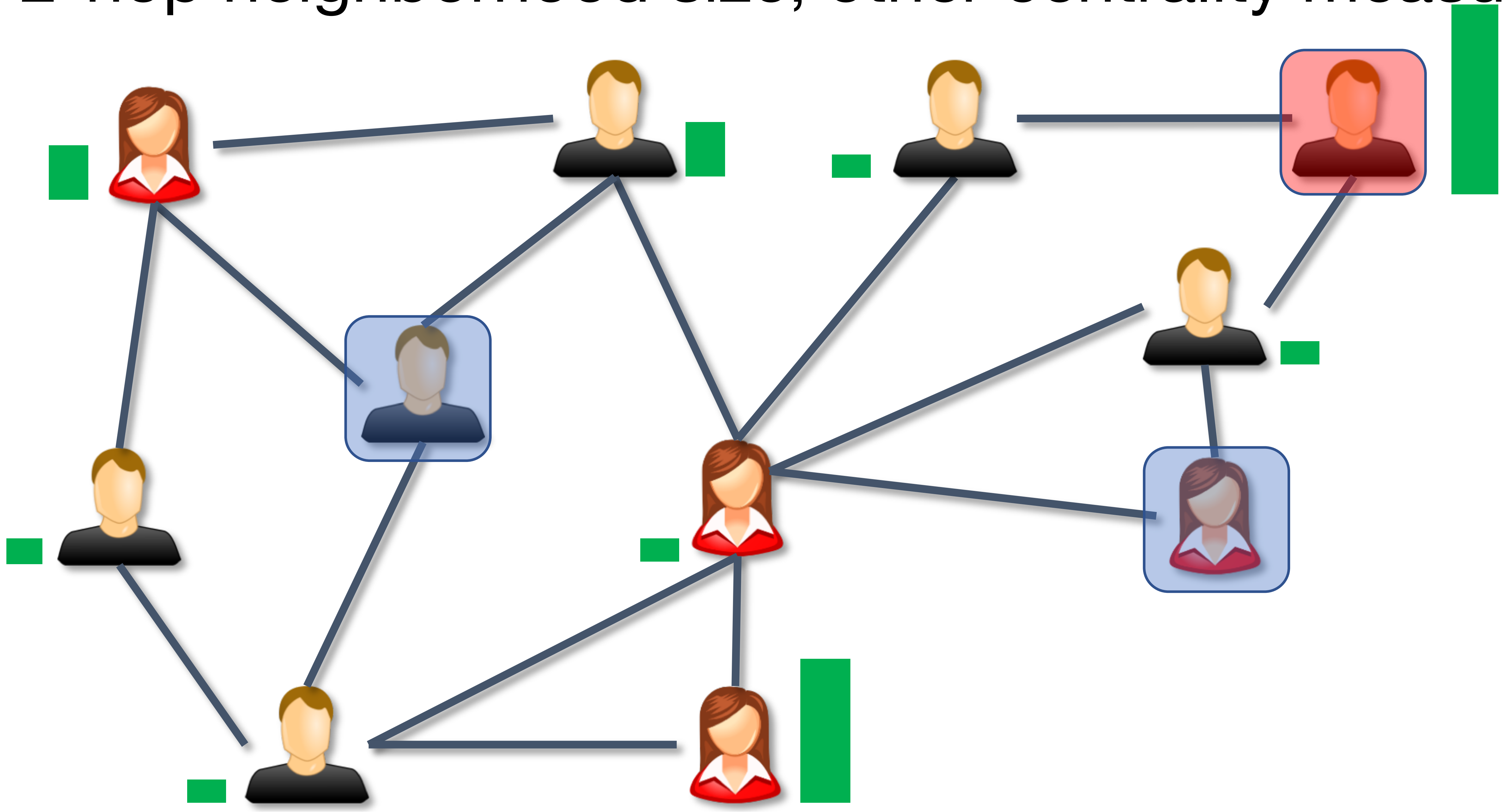
Representing Nodes

- **Action value function:** $\hat{Q}(S_t, v; \Theta)$
  - Estimate of goodness of vertex v in state S_t
- **Representation of** v
  - A feature vector that describes v in state S_t



Challenge #2:
Representing Nodes

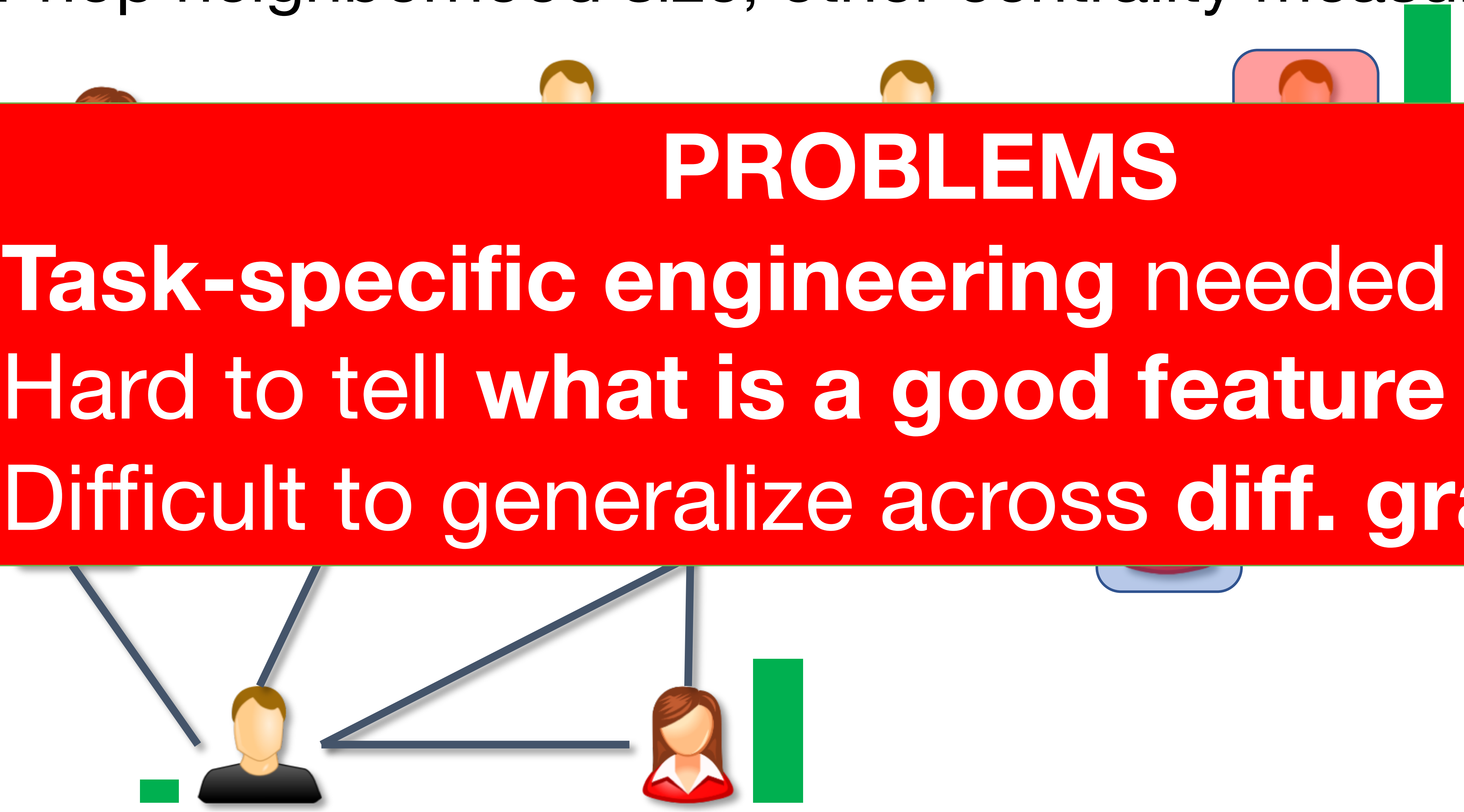
- **Action value function:** $\hat{Q}(S_t, v; \Theta)$
  - Estimate of goodness of vertex v in state S_t
- **Representation of v: Feature engineering**
  - Degree, 2-hop neighborhood size, other centrality measures…



Challenge #2:

Representing Nodes

- **Action value function:** $\hat{Q}(S_t, v; \Theta)$
  - Estimate of goodness of vertex v in state S_t
- **Representation of v: Feature engineering**
  - Degree, 2-hop neighborhood size, other centrality measures…



PROBLEMS

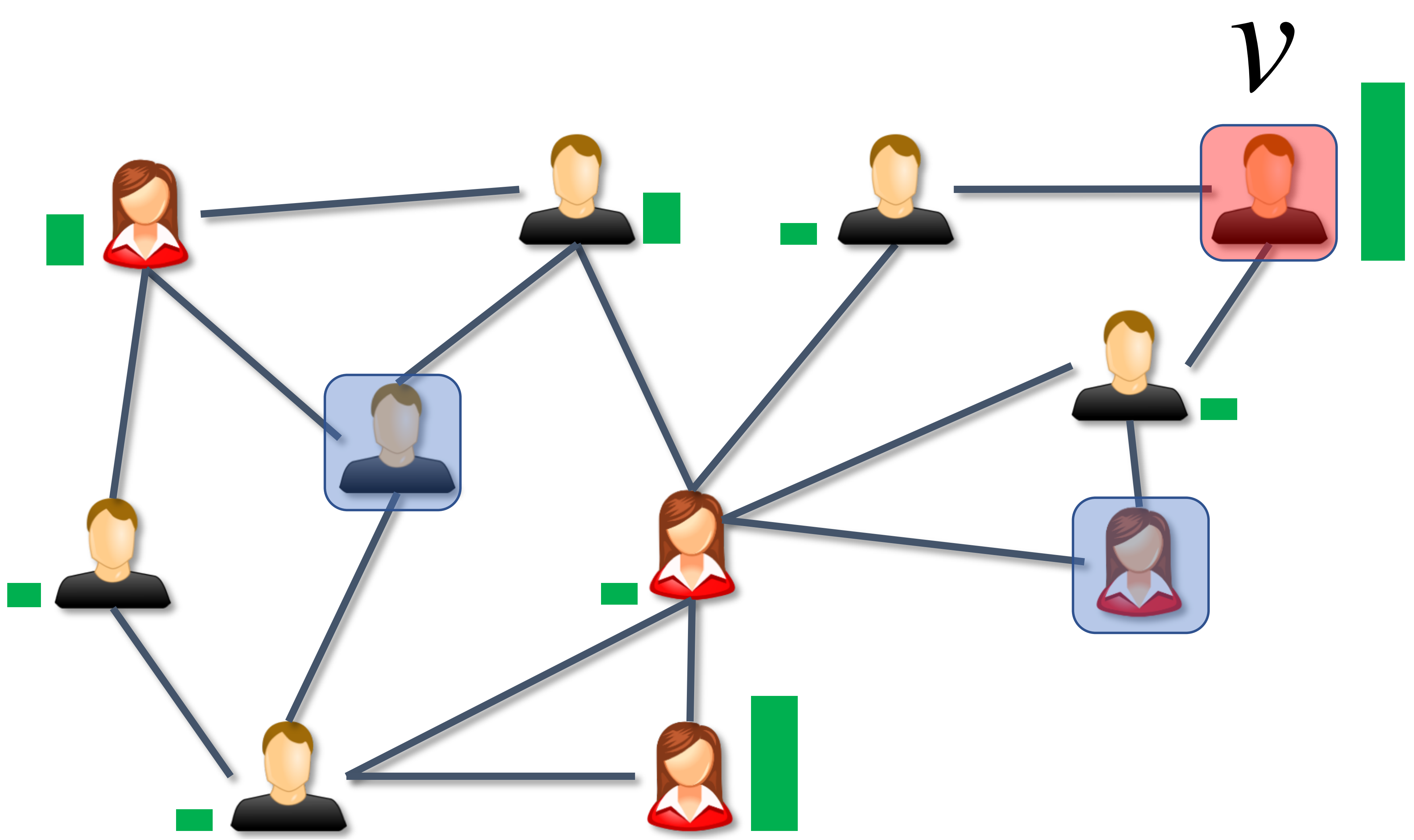
1- **Task-specific engineering** needed

2- Hard to tell **what is a good feature**

3- Difficult to generalize across **diff. graph sizes**

Learning Node Features

Scoring Function: Need to represent node with a **feature vector** first



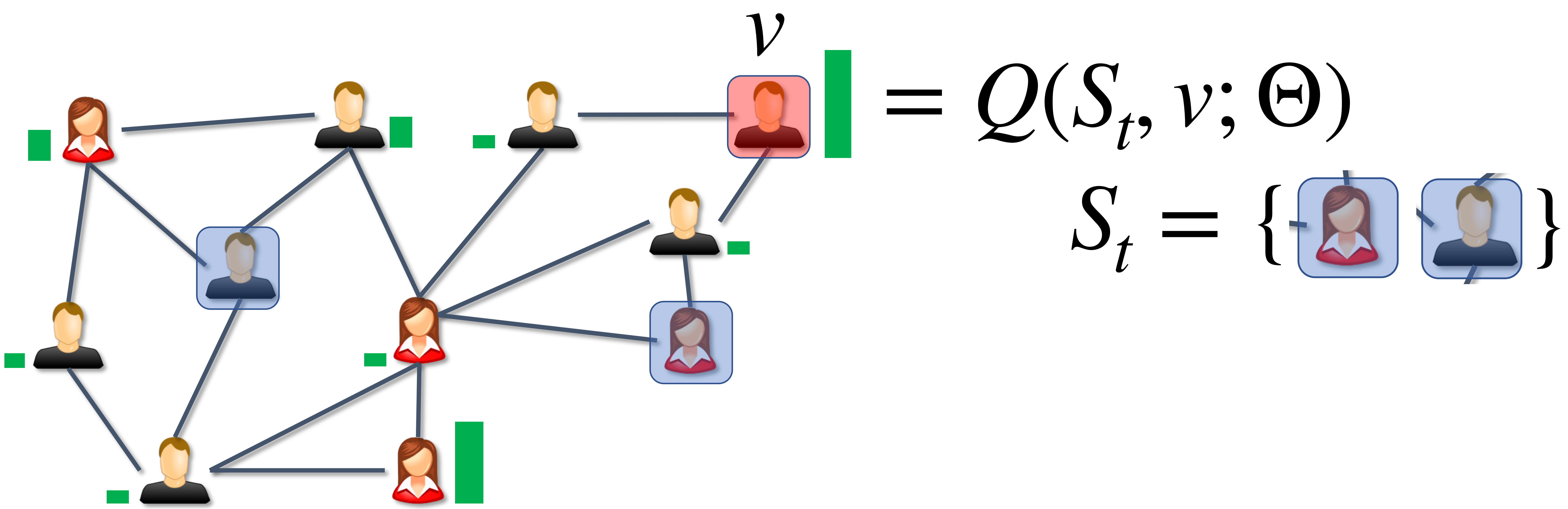
$$= Q(S_t, v; \Theta)$$

$S_t = \{$  $\}$

Learning Node Features

Scoring Function: Need to represent node with a **feature vector** first

Problem: Not clear what good node features are!

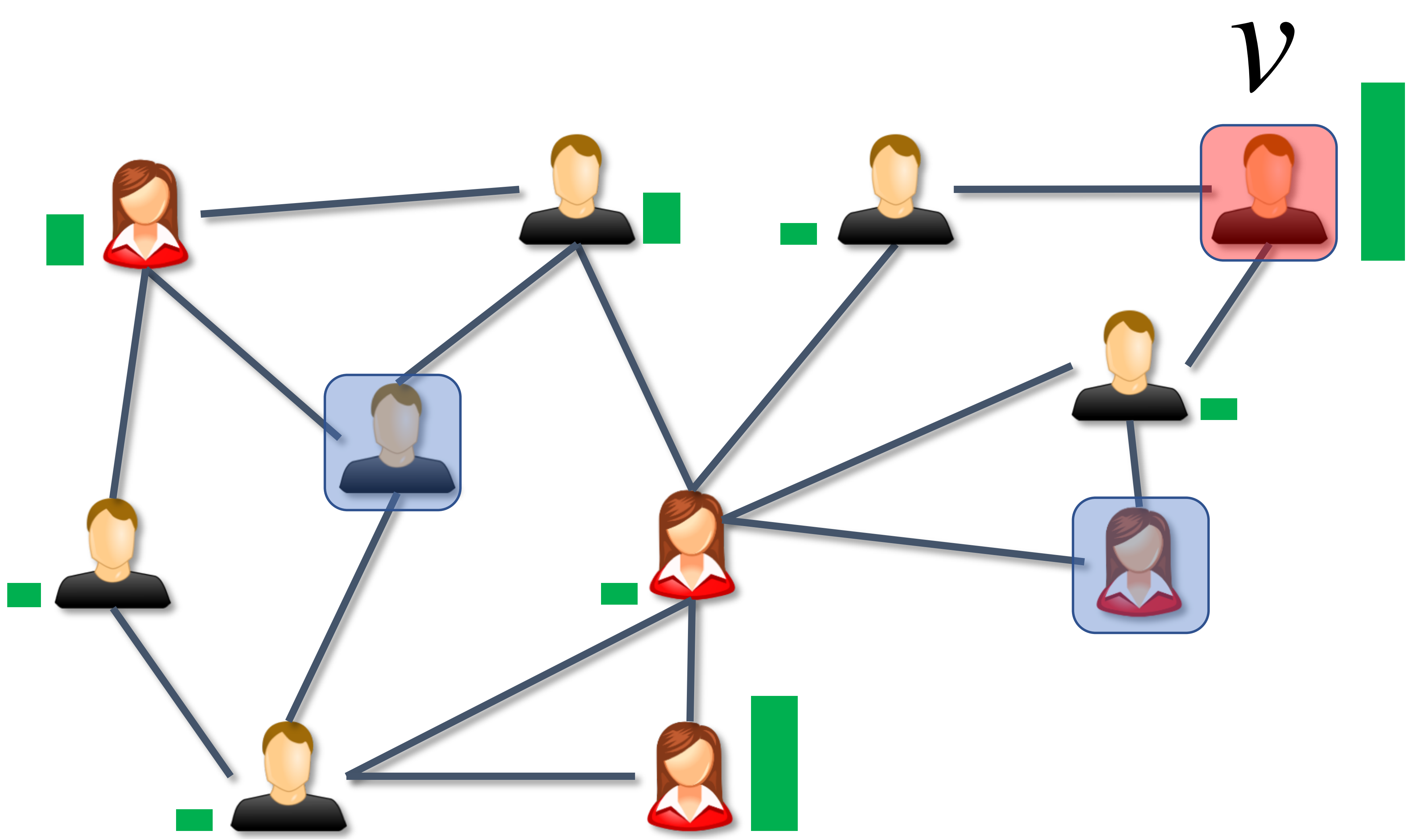


Learning Node Features

Scoring Function: Need to represent node with a **feature vector** first

Problem: Not clear what good node features are!

Solution: Parametrize a **Graph Neural Network** with parameters Θ



$$= Q(S_t, v; \Theta)$$

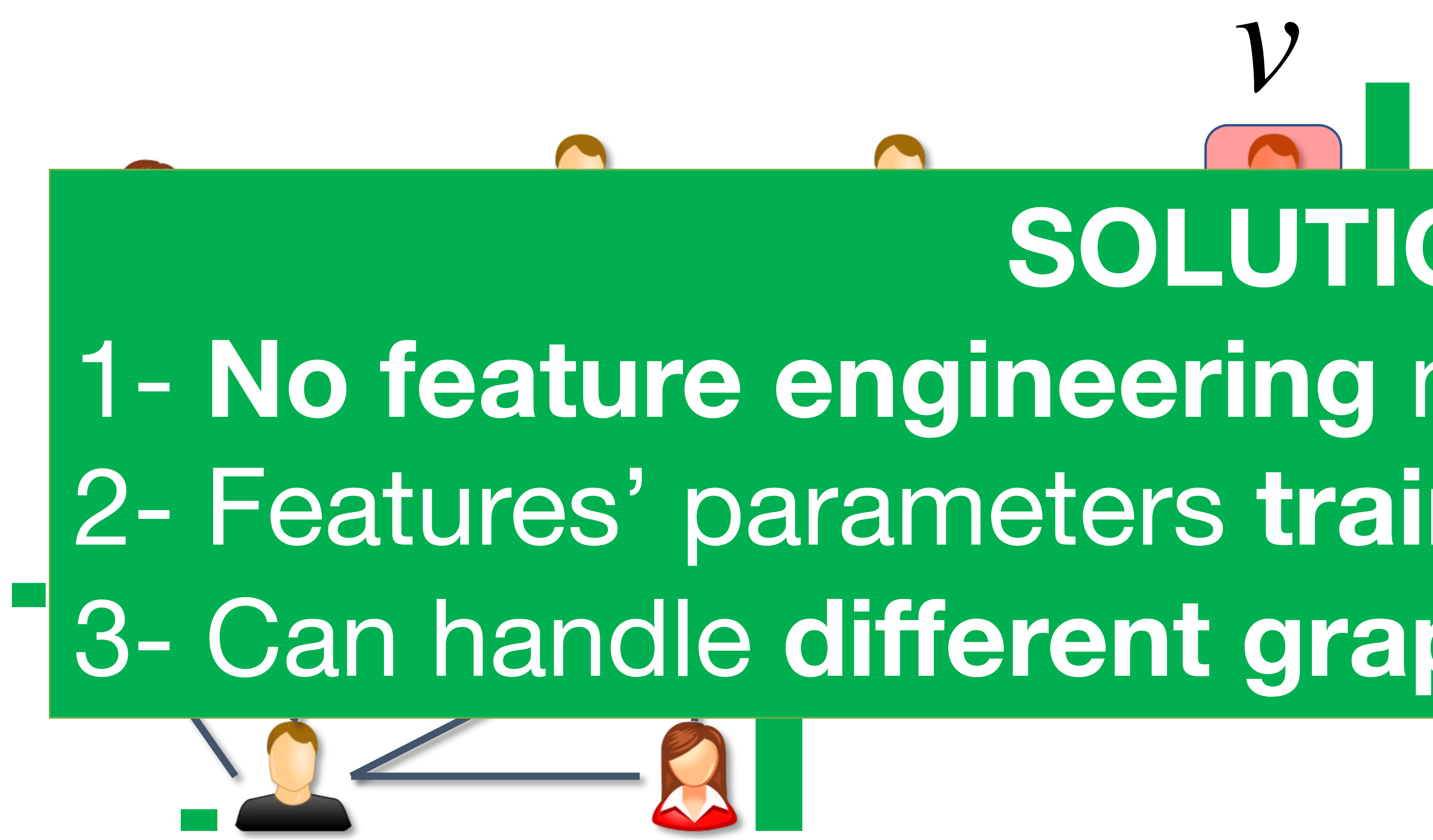
$$S_t = \{ \quad \}$$

Learning Node Features

Scoring Function: Need to represent node with a **feature vector** first

Problem: Not clear what good node features are!

Solution: Parametrize a **Graph Neural Network** with parameters Θ



$= Q(S, v; \Theta)$

SOLUTION

1- **No feature engineering** needed

2- Features' parameters **trained to be good**

3- Can handle **different graph sizes**

Graph Neural Nets in a Nutshell

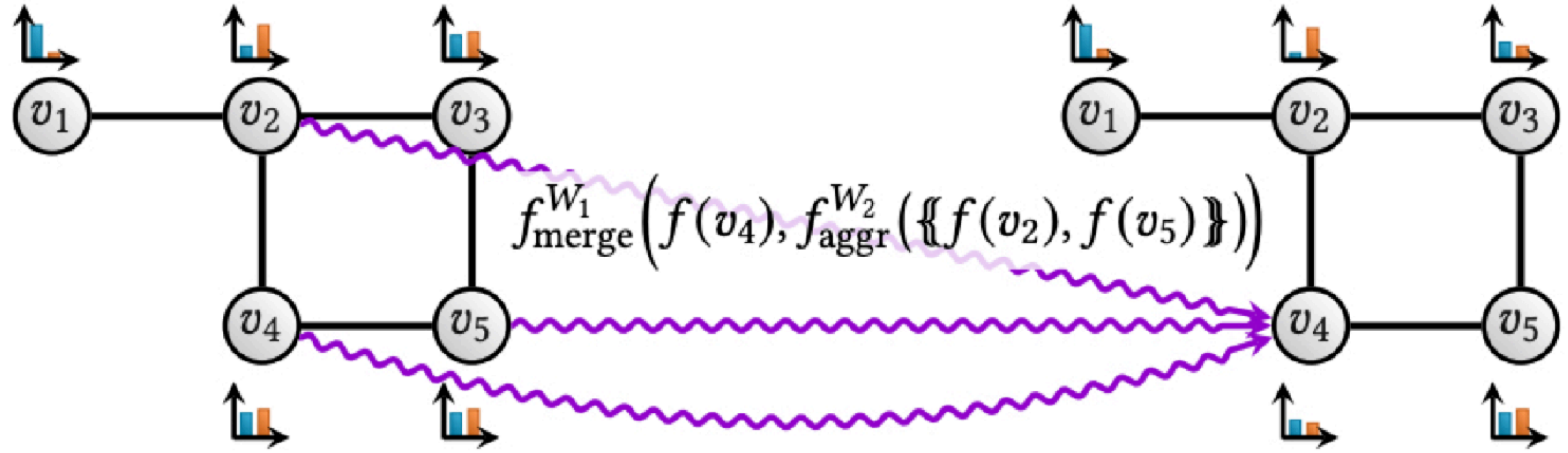


Figure 3: Illustration of the neighborhood aggregation step of a GNN around node v_4.

Combinatorial optimization and reasoning with graph neural networks.
Q. Cappart, D. Chételat, E.B. Khalil, A. Lodi, C. Morris, P. Veličković. **arXiv:2102.09544 2021.**

Reinforcement Learning Algorithm

Algorithm 1 **Q-learning for the Greedy Algorithm**

```
Initialize experience replay memory M to capacity N
for episode e = 1 to L do
    Draw graph G from distribution D
    Initialize the state to empty S_1 = ()
    for step t = 1 to T do
        v_t = { random node v ∈ S̄_t,                              w.p. ε
              { argmax_{v∈S̄_t} Q̂(h(S_t), v; Θ),   otherwise
        Add v_t to partial solution: S_{t+1} := (S_t, v_t)
        if t ≥ n then
            Add tuple (S_{t−n}, v_{t−n}, R_{t−n,t}, S_t) to M
            Sample random batch from B ~iid. M
            Update Θ by SGD over (6) for B
        end if
    end for
end for
return Θ
```

Θ: model parameters

Depend on vertex features

Sample graph instance

Explore or

Exploit according to current policy

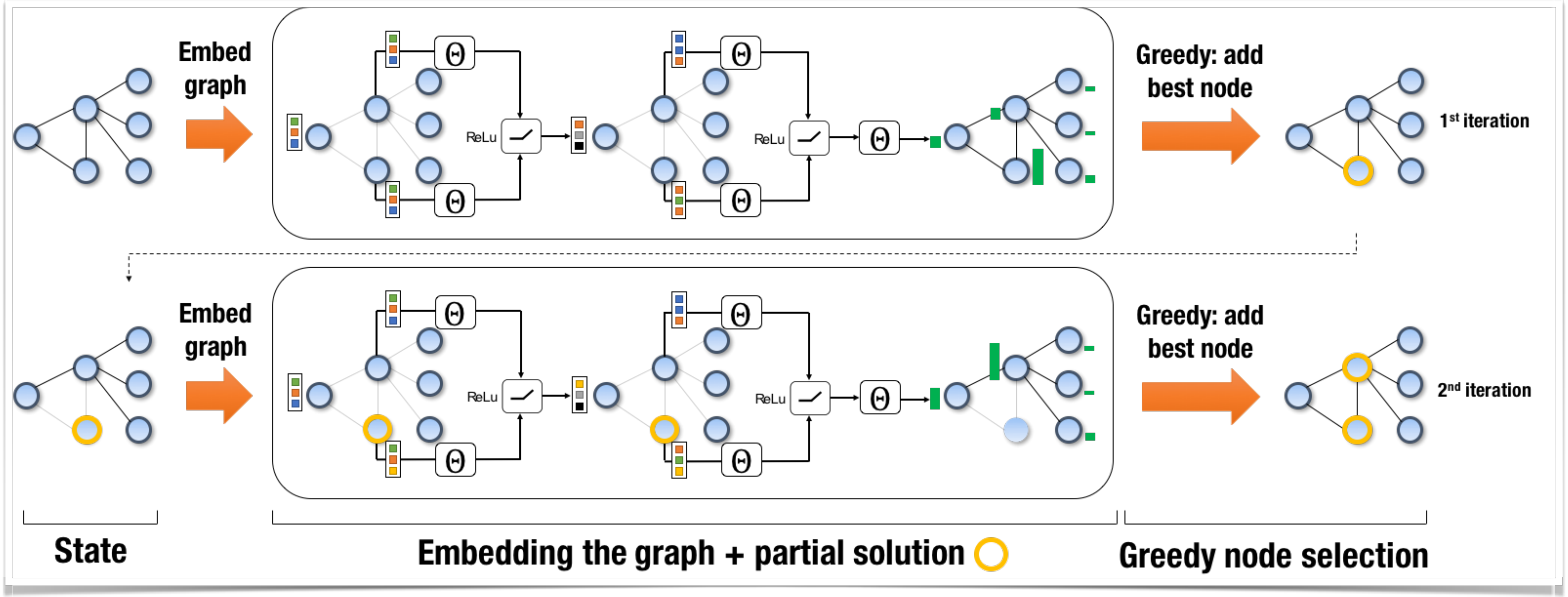
Update state

Optimize model parameters

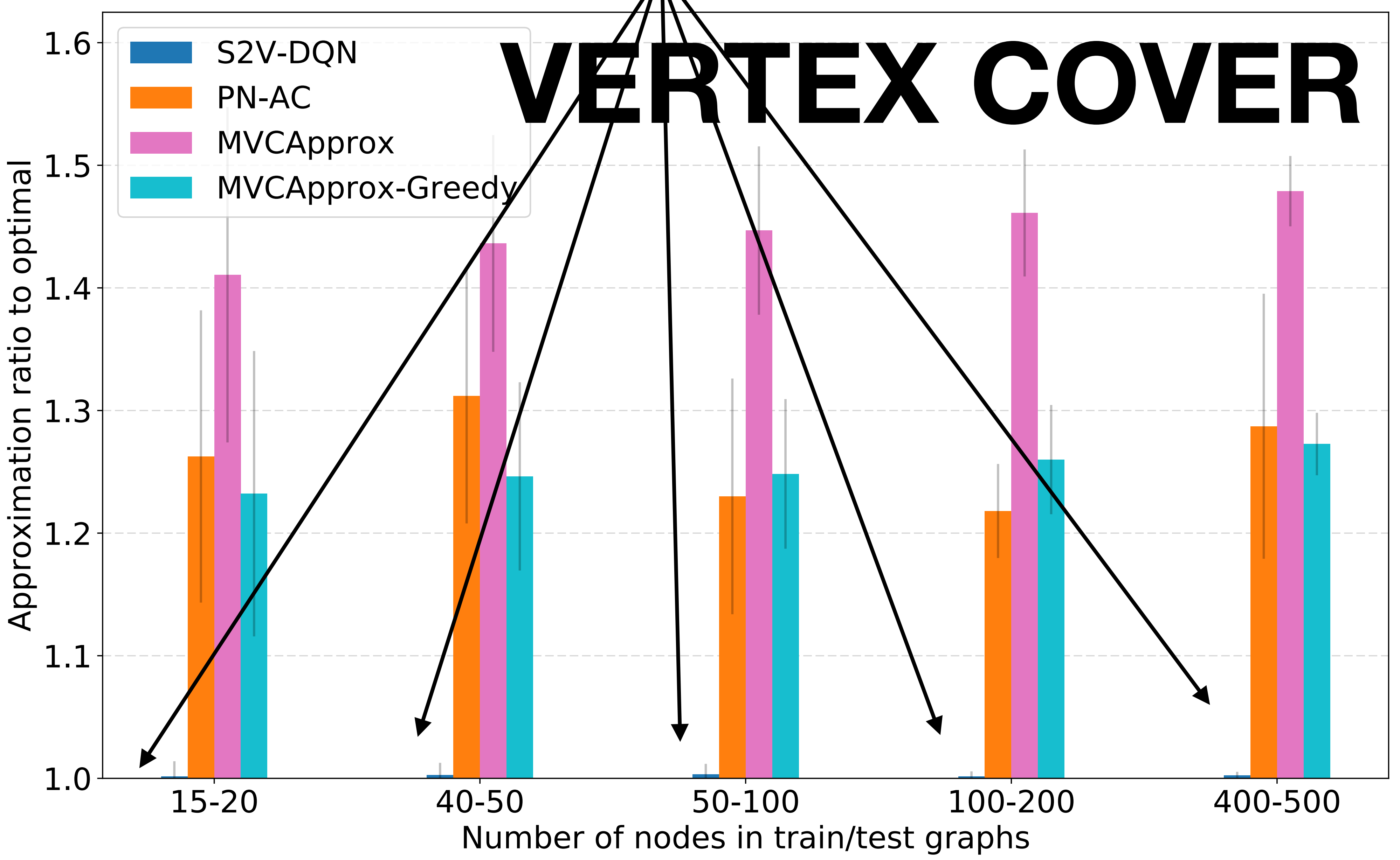
$$(y - \widehat{Q}(h(S_t), v_t; \Theta))^2$$

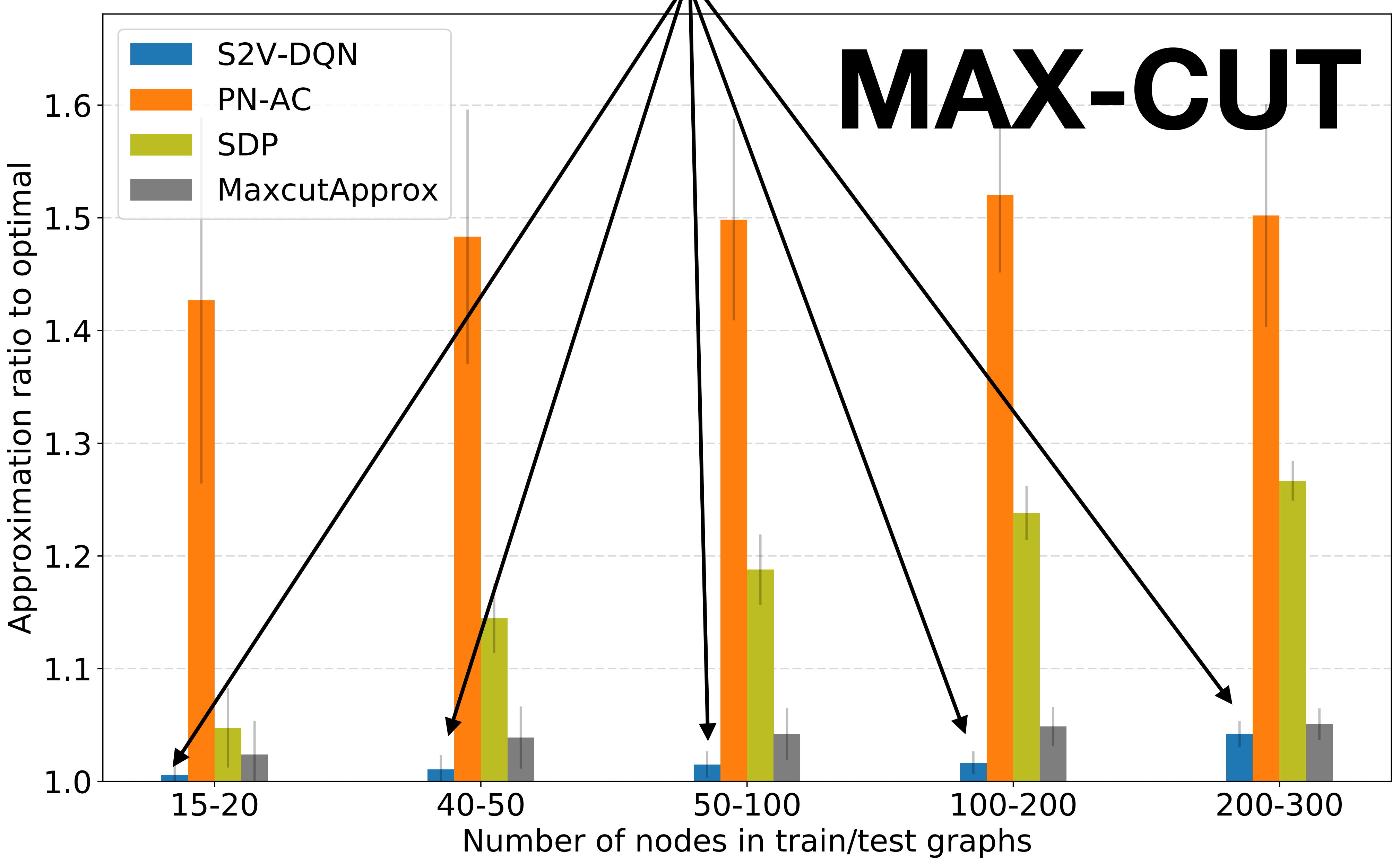
$$y = \gamma \max_{v'} \widehat{Q}(h(S_{t+1}), v'; \Theta) + r(S_t, v_t)$$

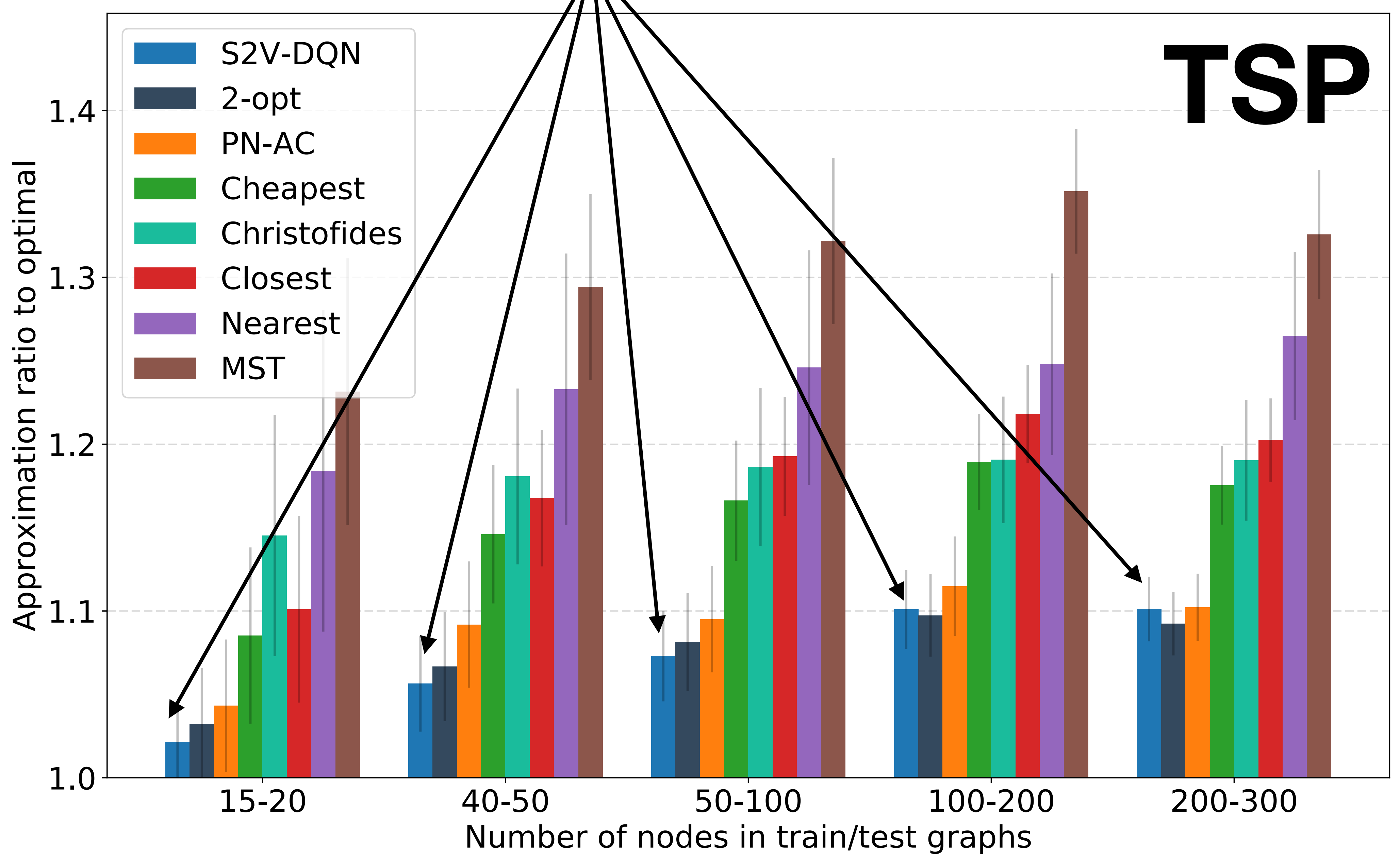
Overall Framework



Learning Greedy in Practice







Approximation Ratio

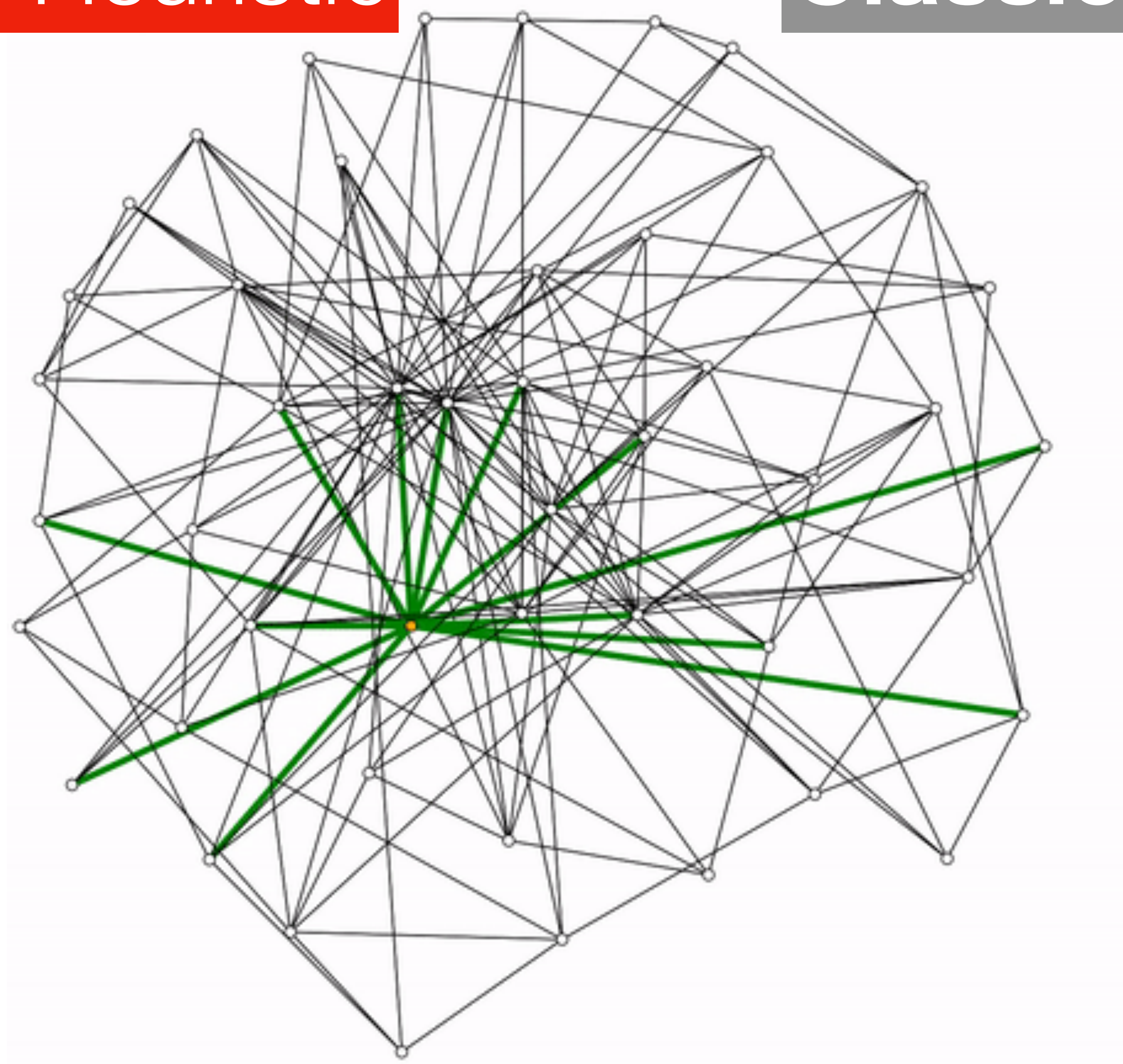
Code:

https://github.com/Hanjun-Dai/graph_comb_opt

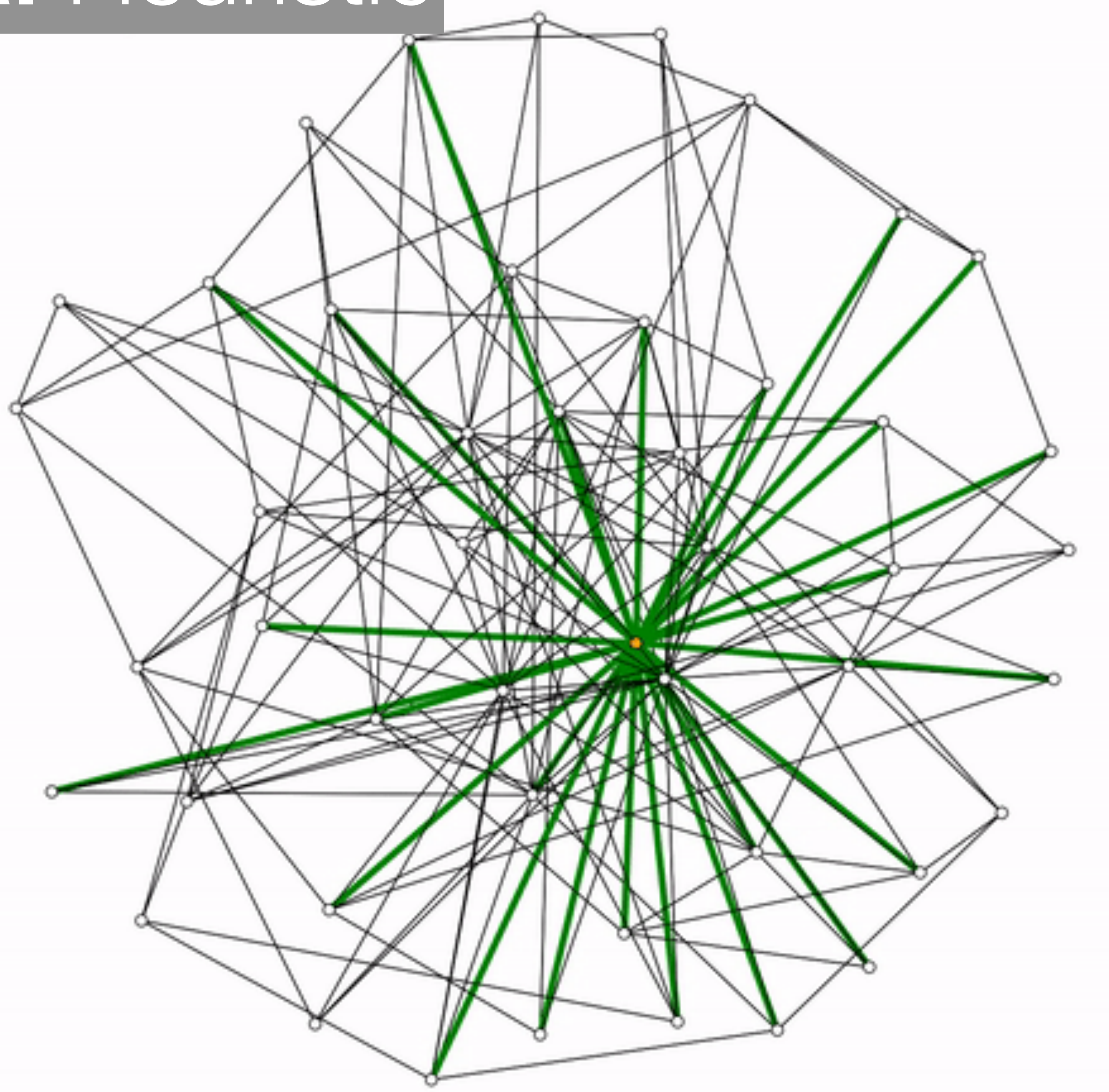
Data-Driven Algorithm Design

automatically **discovers** **novel** search **strategies**

Learned Heuristic



Classical Heuristic



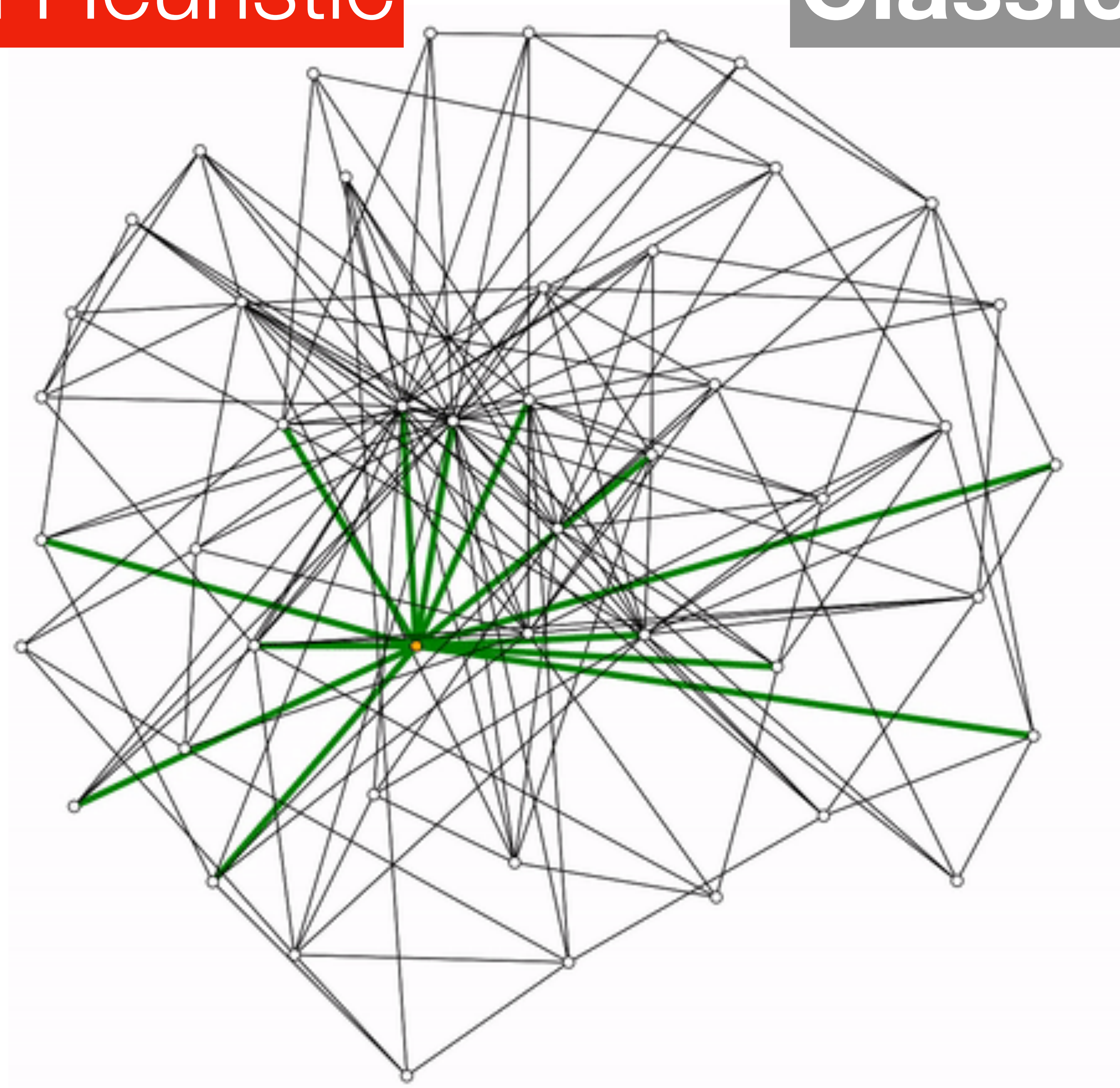
Minimum Vertex Cover

Find **smallest vertex subset** such that each **edge is covered**

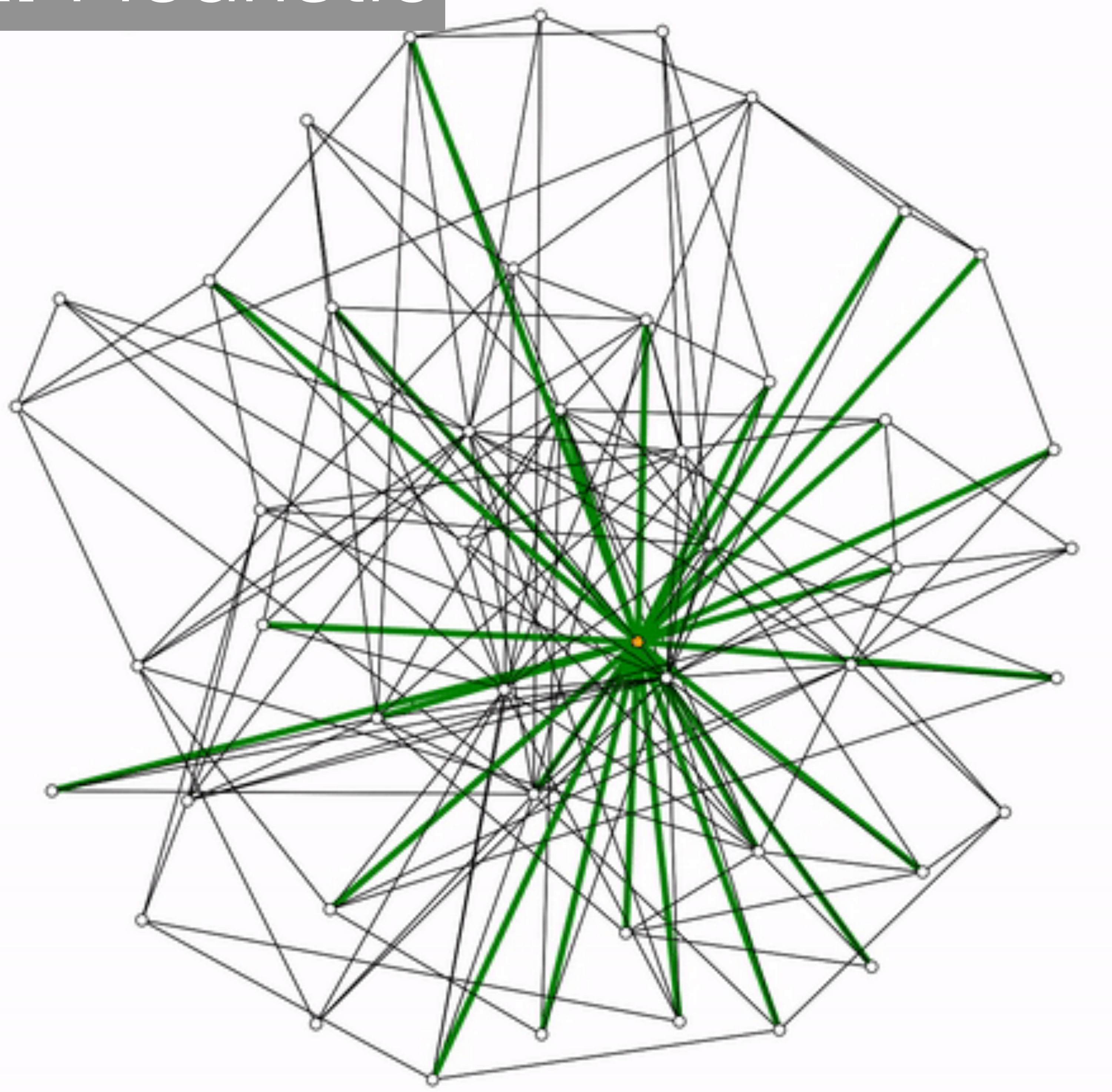
Data-Driven Algorithm Design

automatically **discovers** **novel** search **strategies**

Learned Heuristic



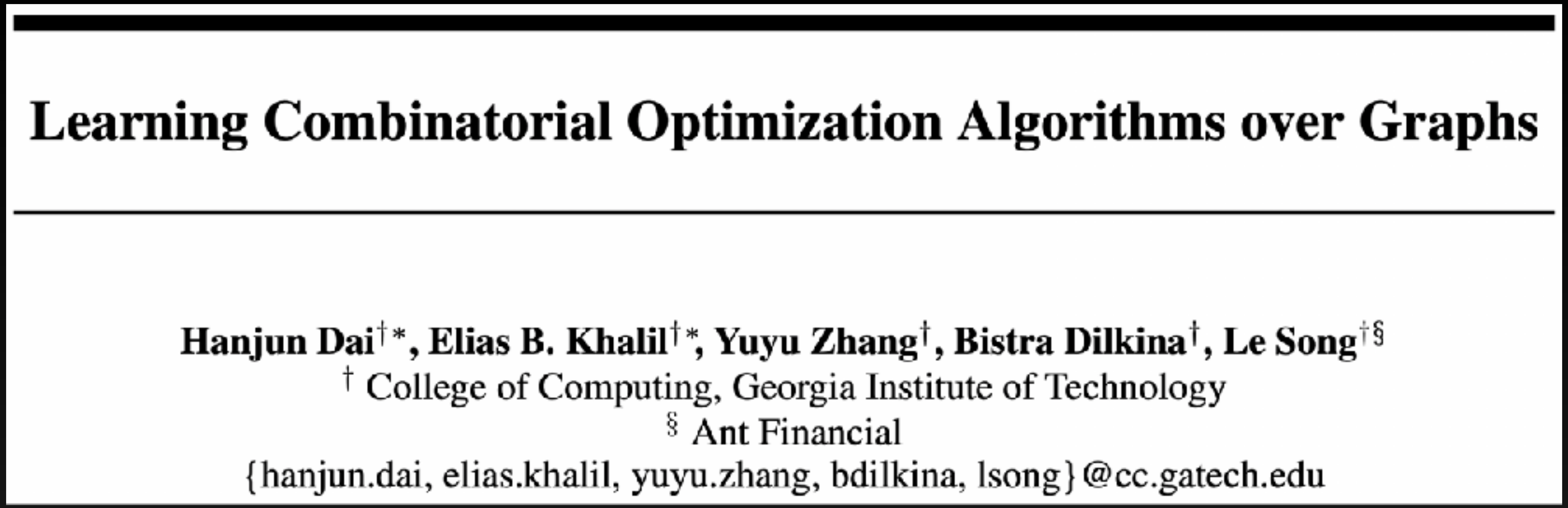
Classical Heuristic



Minimum Vertex Cover

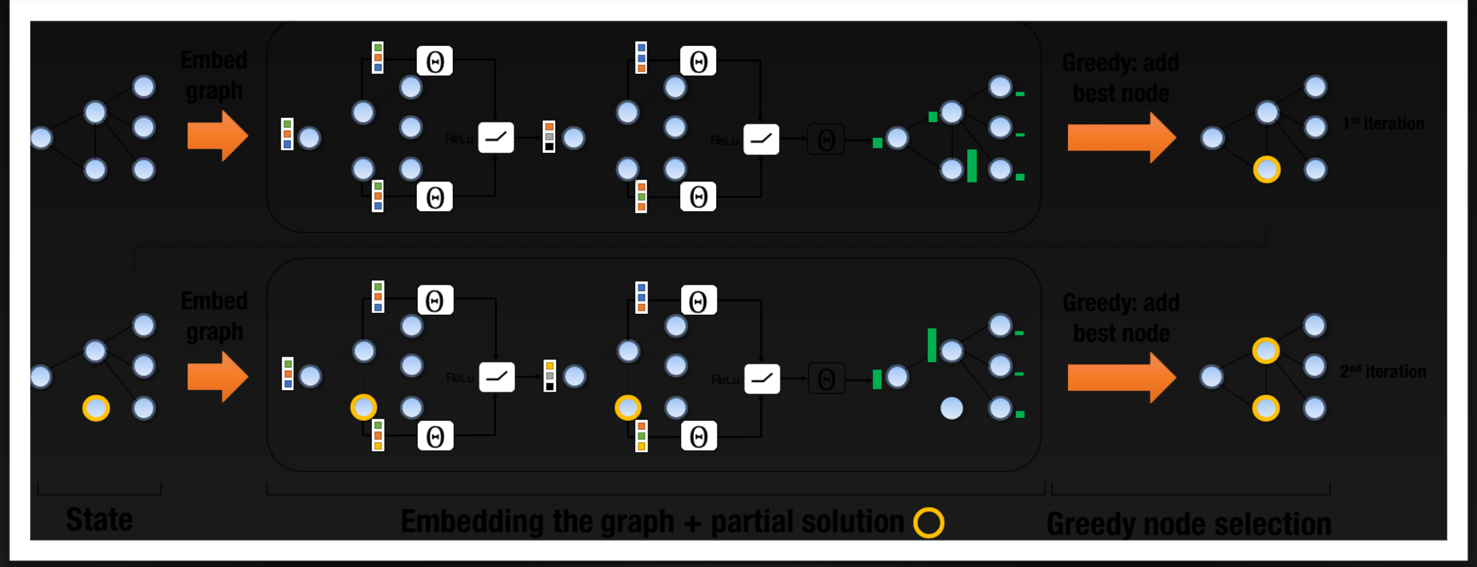
Find **smallest vertex subset** such that each **edge is covered**

Takeaways #2



Learning Combinatorial Optimization Algorithms over Graphs

Hanjun Dai[†*], Elias B. Khalil[†*], Yuyu Zhang[†], Bistra Dilkina[†], Le Song[†§]
[†] College of Computing, Georgia Institute of Technology
[§] Ant Financial
{hanjun.dai, elias.khalil, yuyu.zhang, bdilkina, lsong}@cc.gatech.edu



arXiv:1704.01665

code at https://github.com/Hanjun-Dai/graph_comb_opt

- **DRL tailors greedy search** to your instances.
- **Learn features jointly with greedy policy**.
- **Human priors** encoded via (greedy) meta-algorithm.
- Interesting, **novel strategies emerge**!

A Deep Reinforcement Learning Framework For Column Generation

or accelerating the solution of exponential LPs with RL

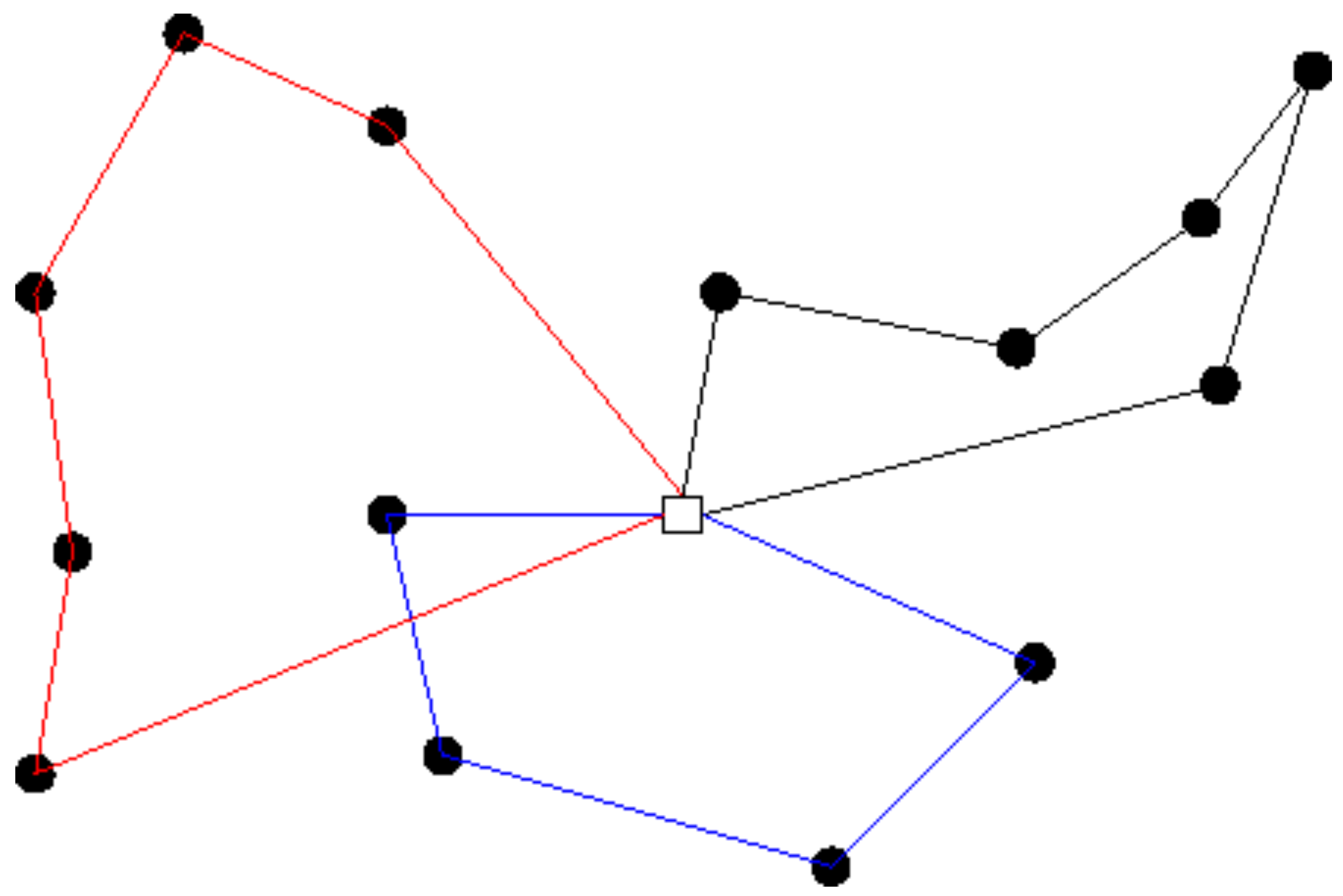
Joint work with students Cheng Chi, Juyoung Wang, Zoha Sherkat-Masoumi at Toronto — MIE, and Amine Aboussalah at NYU

NeurIPS 2022

arXiv:2206.02568

Complicated real-world optimization problems

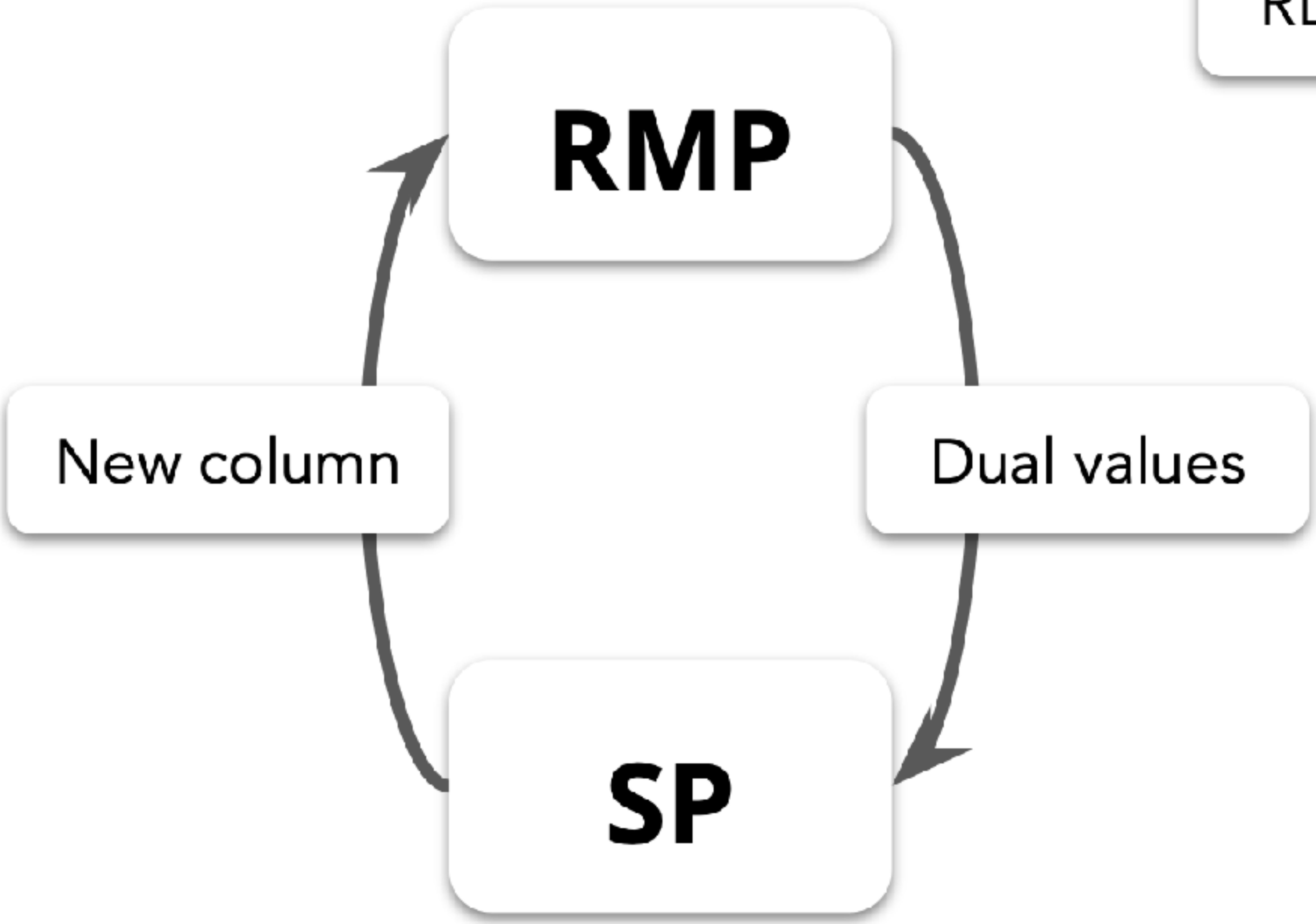
- ➔ Sometimes useful/natural to model with an **exponential** number of decision variables
- ➔ + Hard, combinatorial constraints (many infeasible solutions)
- ➔ **In this work**: Linear Programs with exponentially many variables
- ➔ Foundation for Integer Linear Programming with many variables



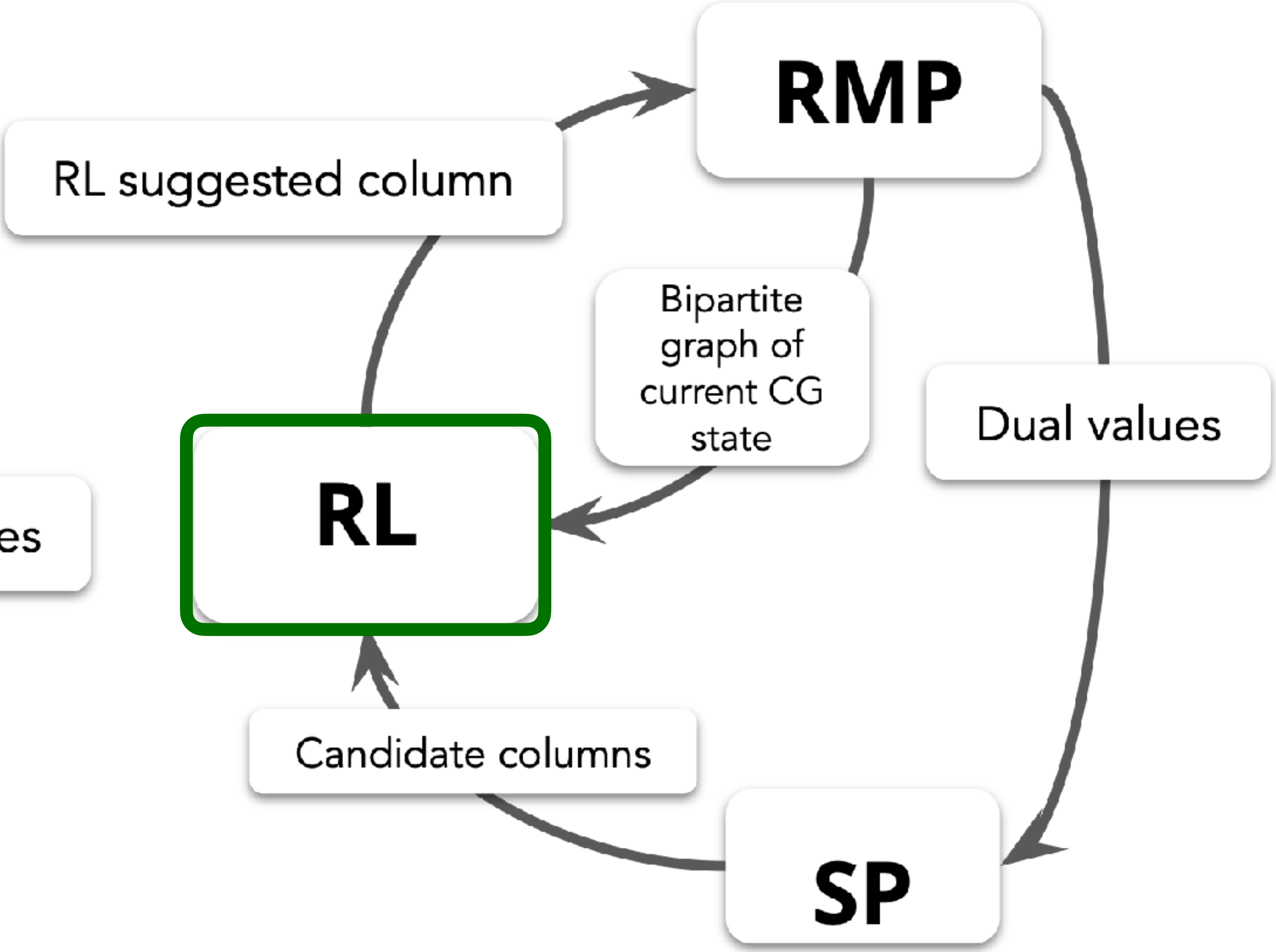
Vehicle Routing Problem (VRP)

- Decision variable ⇔ one route through a subset of locations —> exponentially many!
- Search for a (binary) decision vector in a way that satisfy all the constraints, minimizes linear objective


RMP
New column
Dual values
SP
RMP
RL suggested column
Bipartite graph of current CG state
Dual values
RL
Candidate columns
SP



Column Generation (CG)



RLCG

Complicated real-world optimization problems

- ➔ Sometimes useful/natural to model with an **exponential** number of decision variables
- ➔ + Hard, combinatorial constraints (many infeasible solutions)
- ➔ **In this work**: Linear Programs with exponentially many variables
- ➔ Foundation for Integer Linear Programming with many variables

Complicated real-world optimization problems

- ➔ Sometimes useful/natural to model with an **exponential** number of decision variables
- ➔ + Hard, combinatorial constraints (many infeasible solutions)
- ➔ **In this work**: Linear Programs with exponentially many variables
- ➔ Foundation for Integer Linear Programming with many variables

RLCG

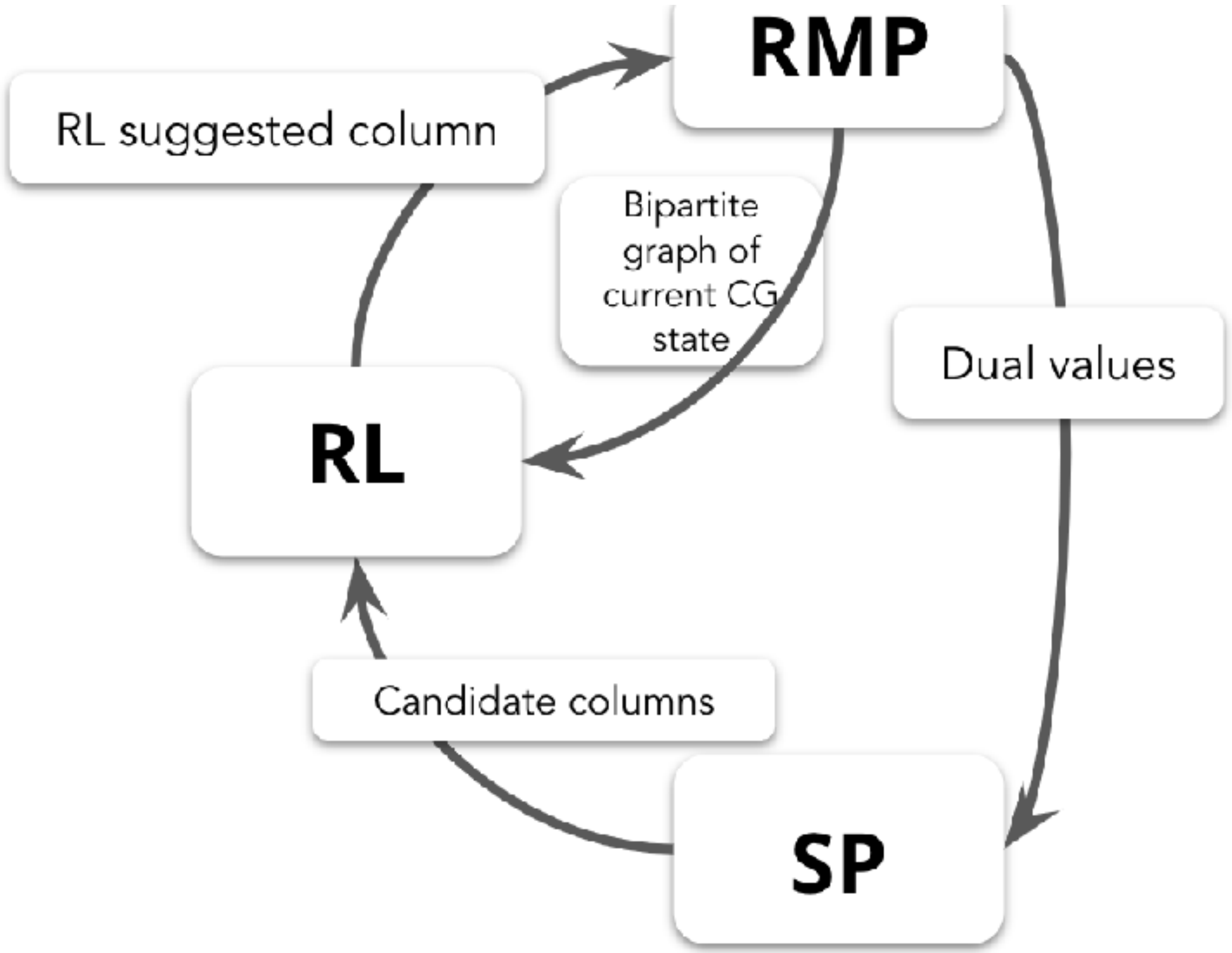
RL

Minimizing CG solving iterations

Column Generation

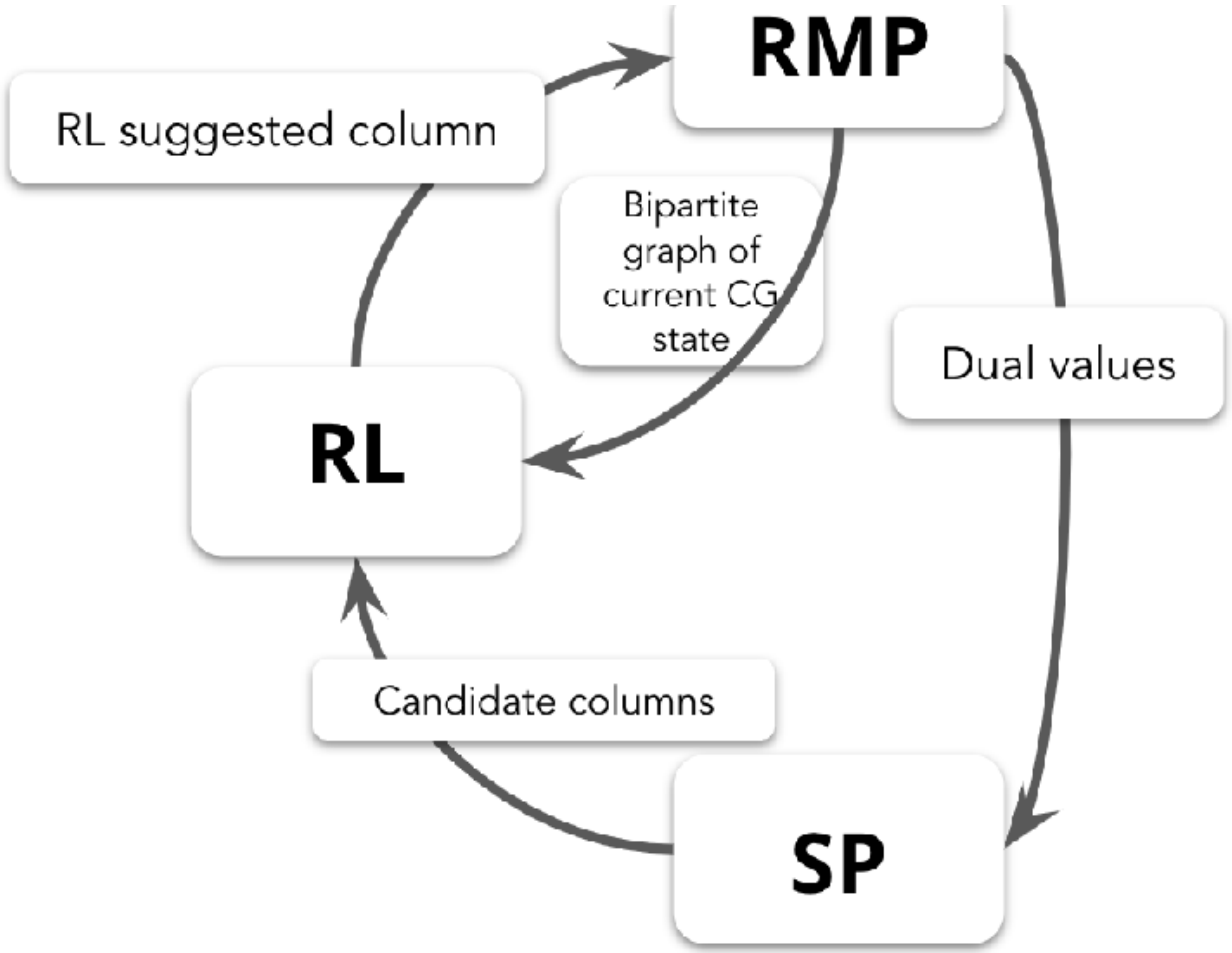
- Handle exponentially many decision variables
- Handle feasibility in hard constraints

RLCG framework



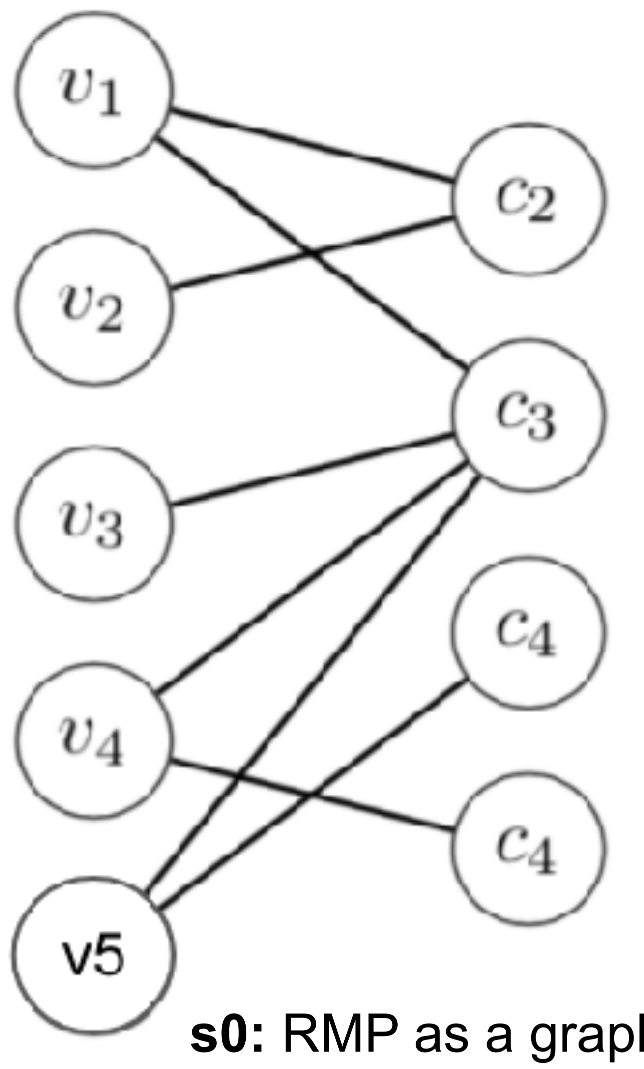
RLCG framework

At each CG iteration:

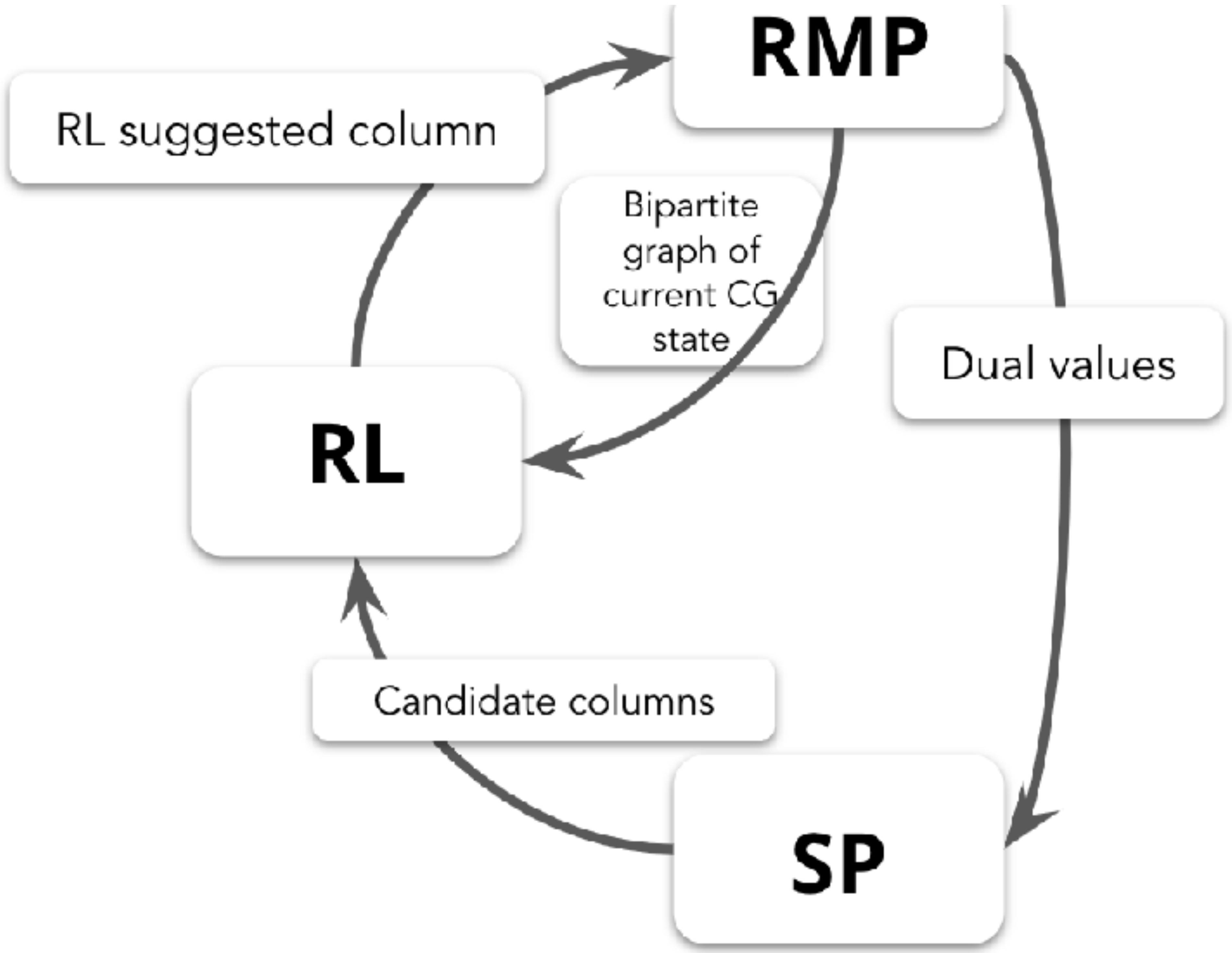


RLCG framework

At each CG iteration:



s0: RMP as a graph



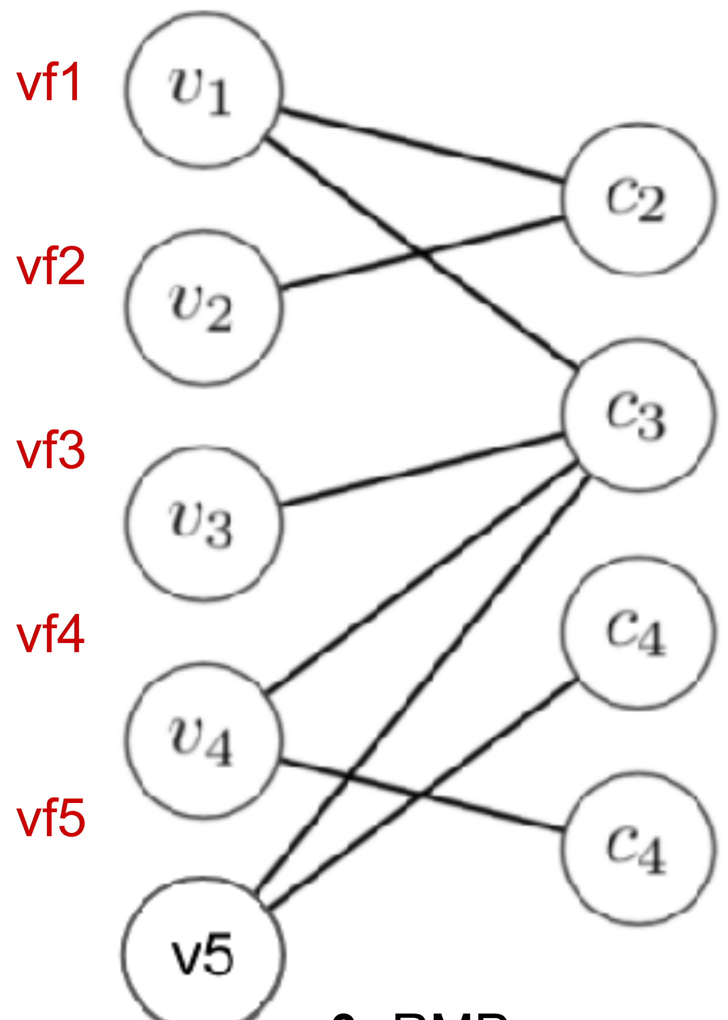
RLCG framework

At each CG iteration:

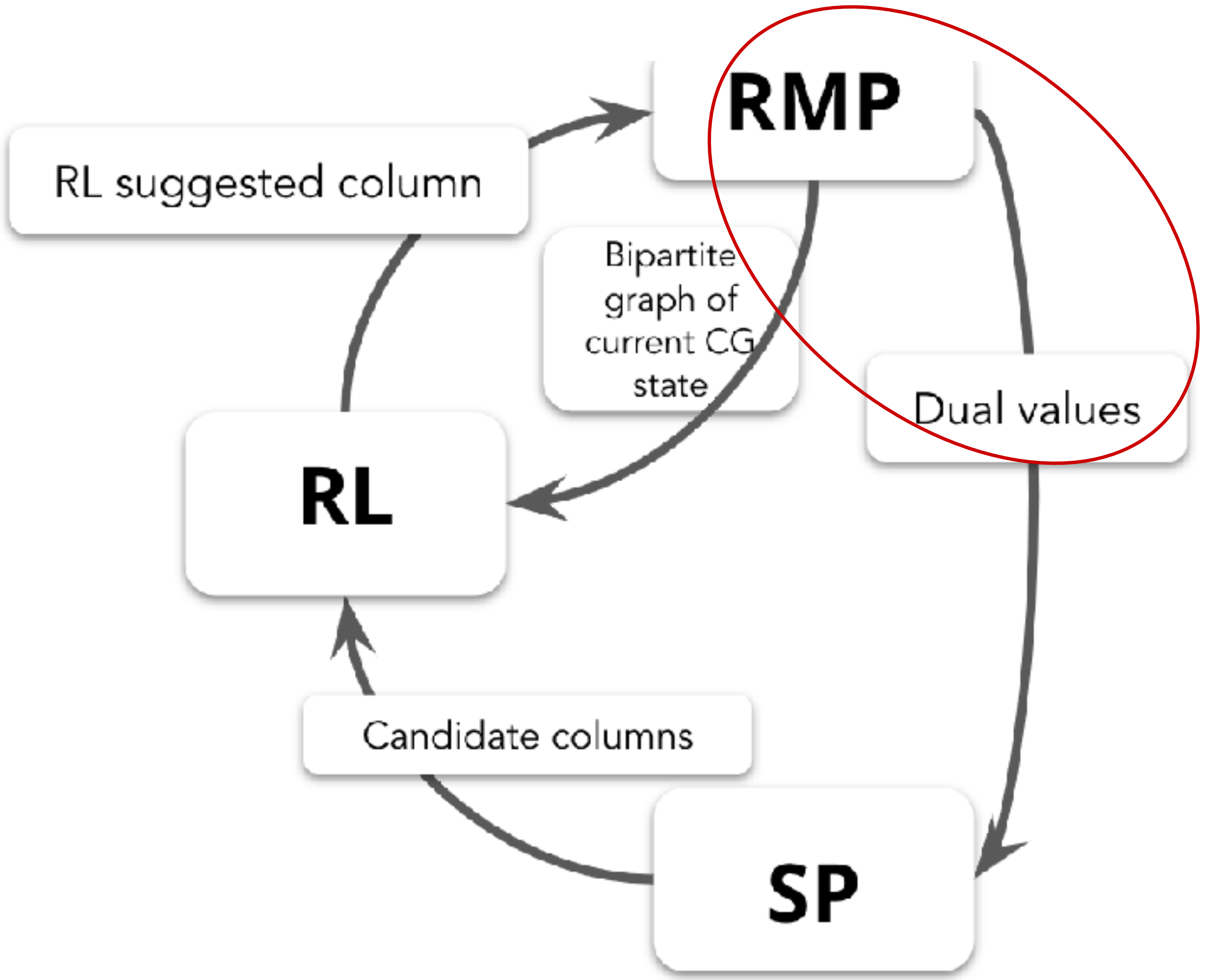
1.Solve this Restricted Master Problem (RMP)

get variable nodes features

get dual values



s0: RMP as a graph



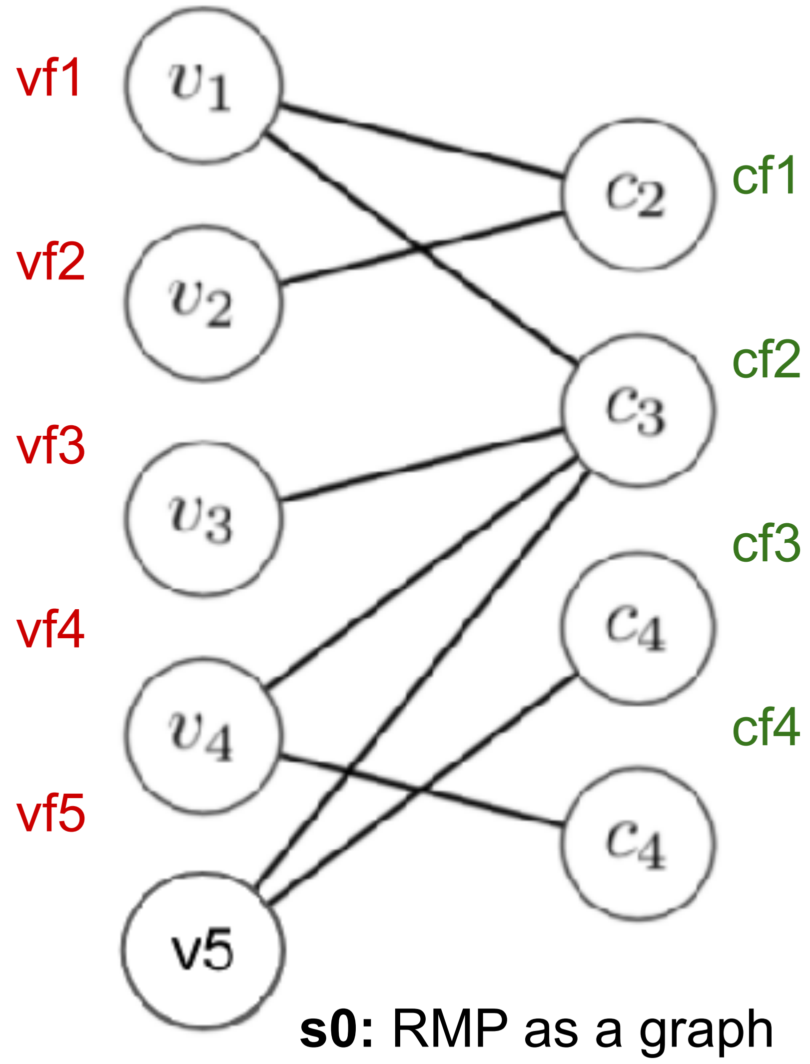
RLCG framework

At each CG iteration:

1.Solve this Restricted Master Problem (RMP)

get variable nodes features

get dual values

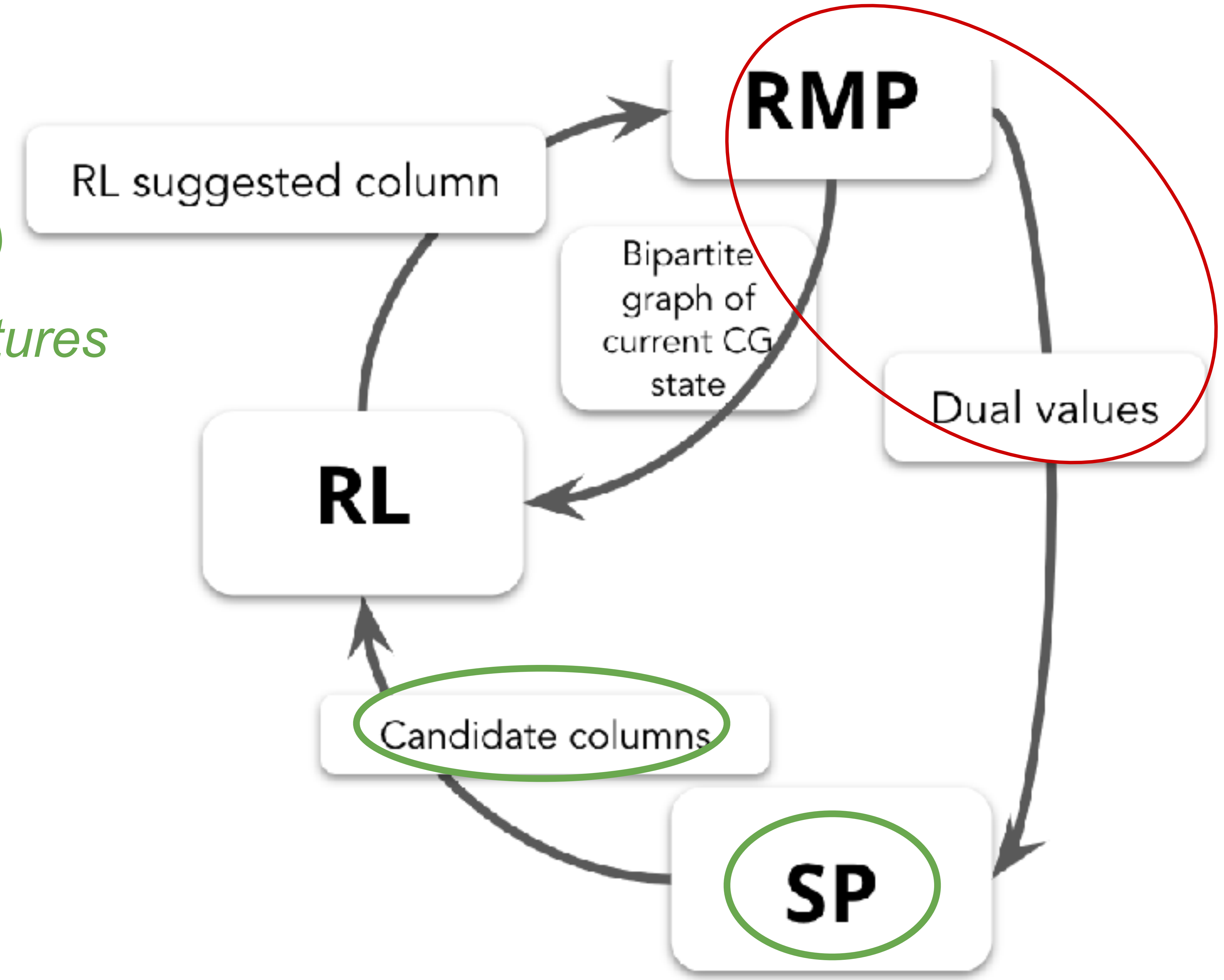


s0: RMP as a graph

2. Solve Subproblem (SP)

get constraint nodes features

get candidate variables (actions)



RLCG framework

At each CG iteration:

1.Solve this Restricted Master Problem (RMP)

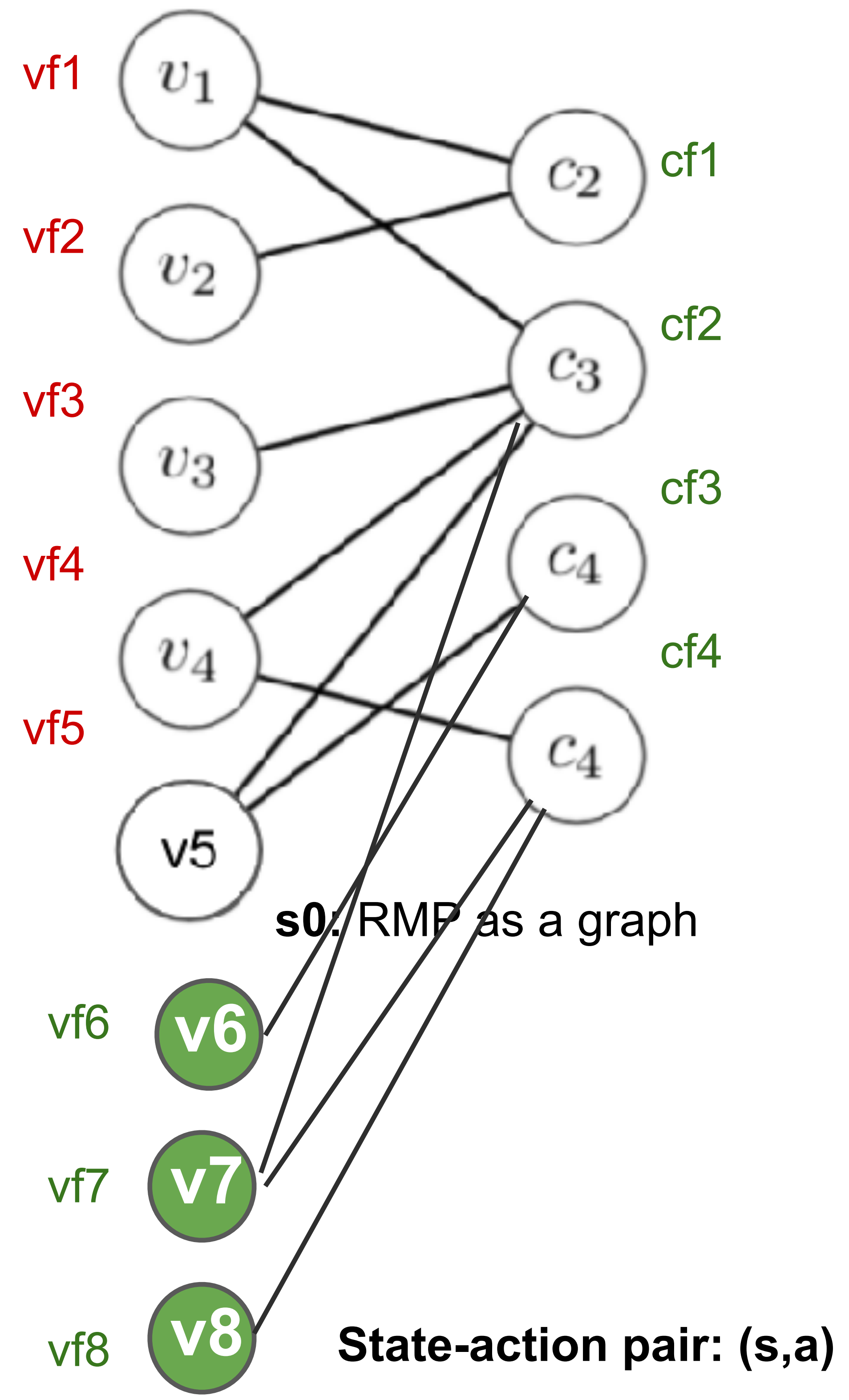
get variable nodes features

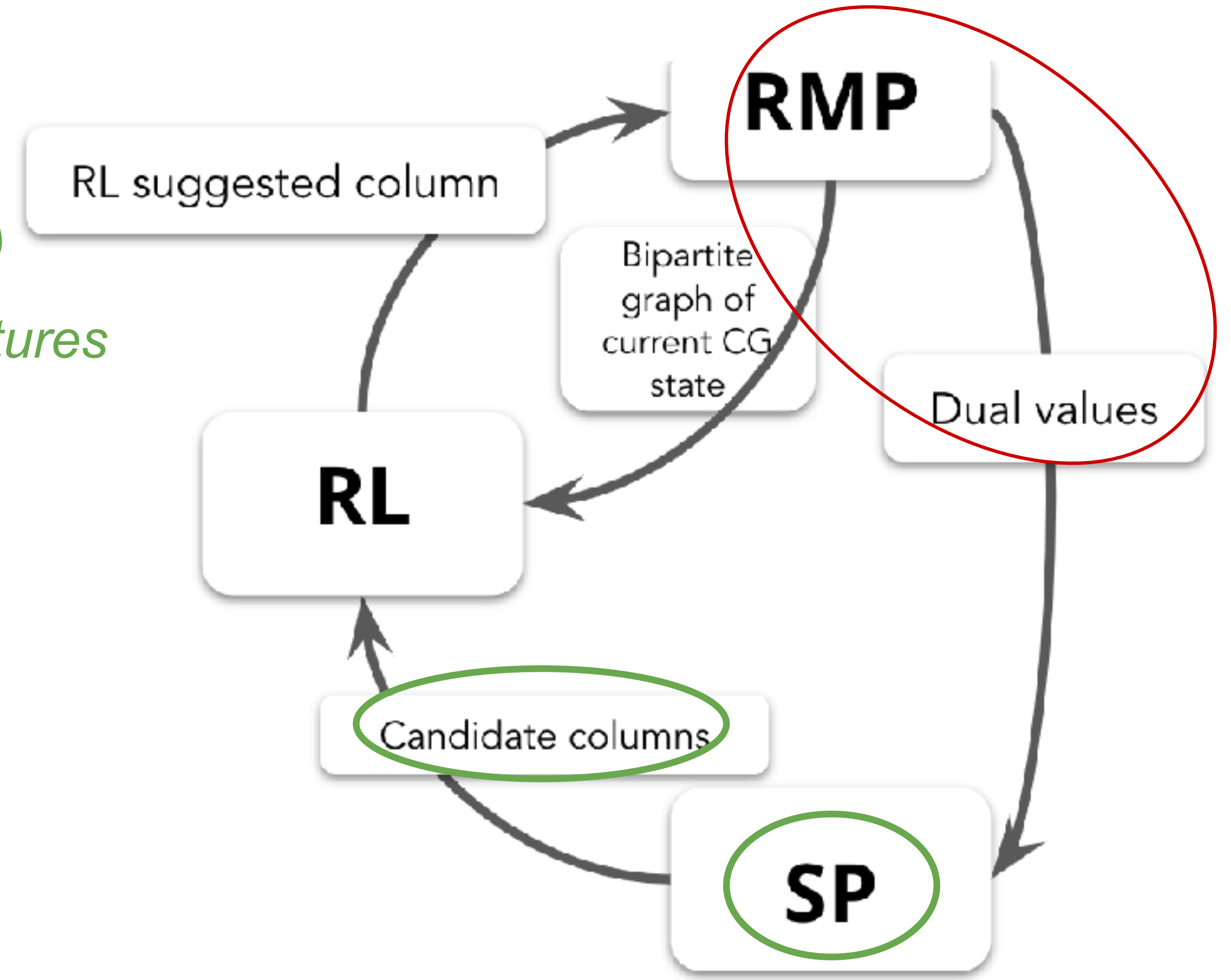
get dual values

2. Solve Subproblem (SP)

get constraint nodes features

get candidate variables (actions)





RLCG framework

At each CG iteration:

1.Solve this Restricted Master Problem (RMP)

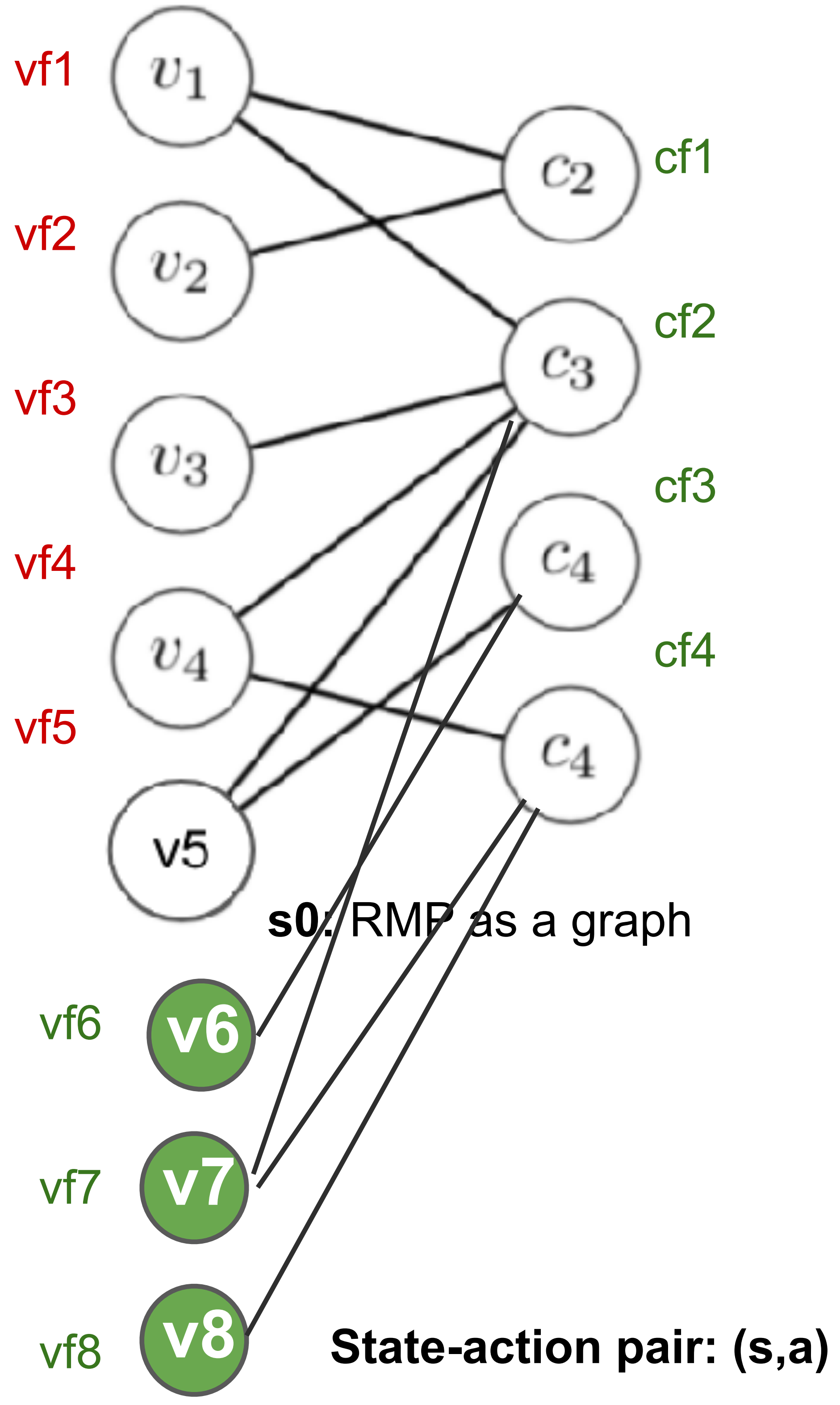
get variable nodes features

get dual values

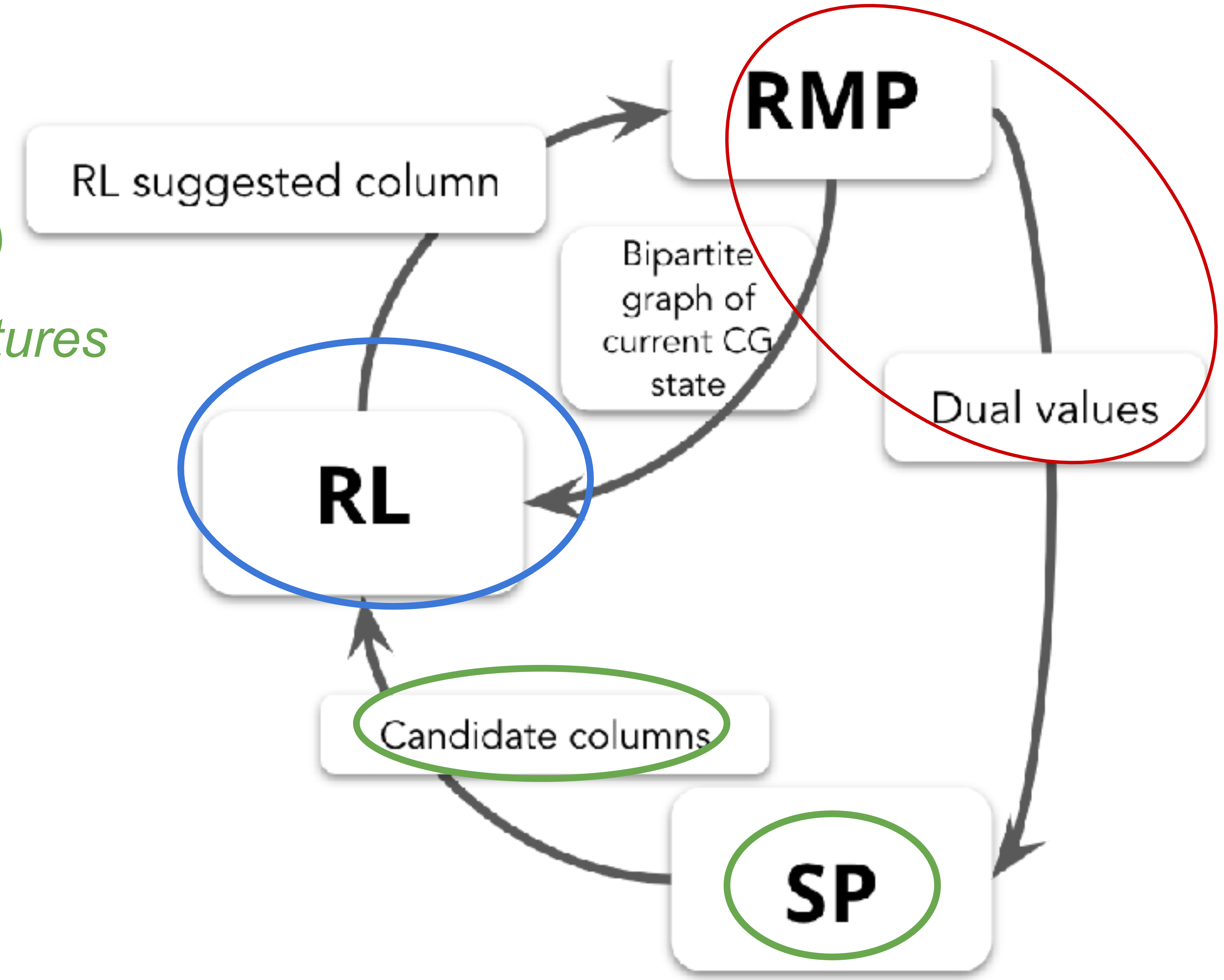
2. Solve Subproblem (SP)

get constraint nodes features

get candidate variables (actions)



3. Input a state action pair to RL agent



RLCG framework

At each CG iteration:

1.Solve this Restricted Master Problem (RMP)

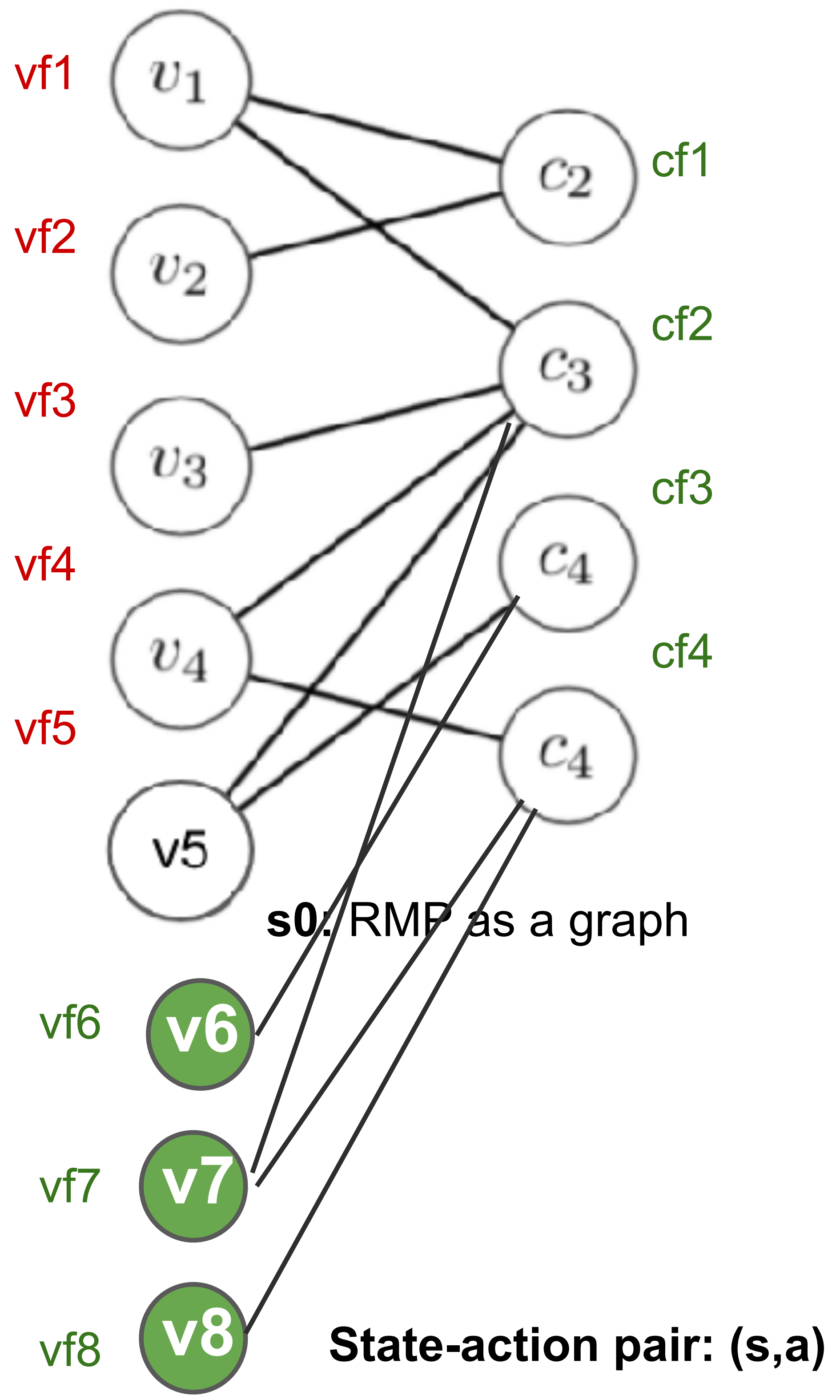
get variable nodes features

get dual values

2. Solve Subproblem (SP)

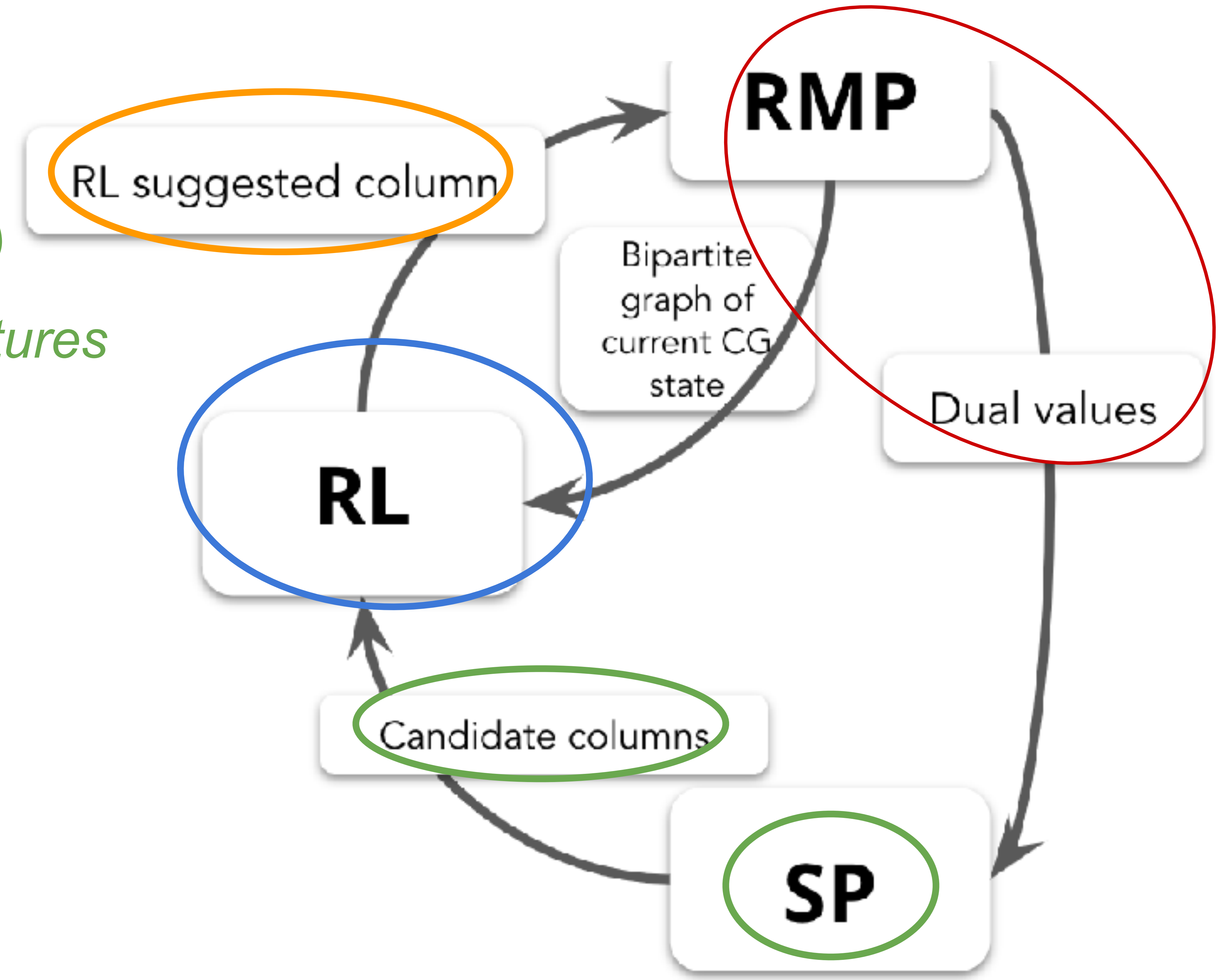
get constraint nodes features

get candidate variables (actions)



3. Input a state action pair to RL agent

4. Evaluate Q(s,a) to select and execute a*



RLCG framework

At each CG iteration:

1.Solve this Restricted Master Problem (RMP)

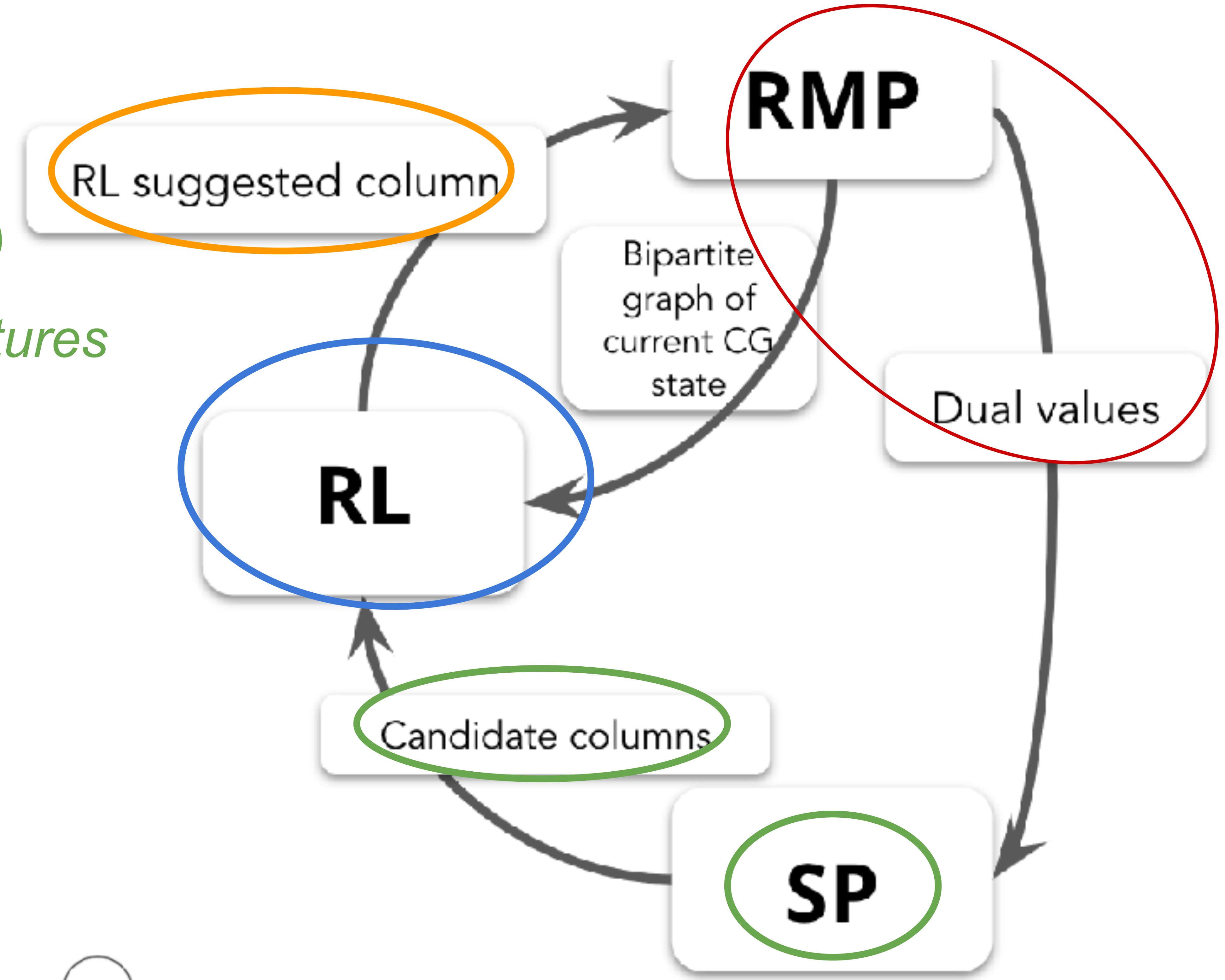
get variable nodes features

get dual values

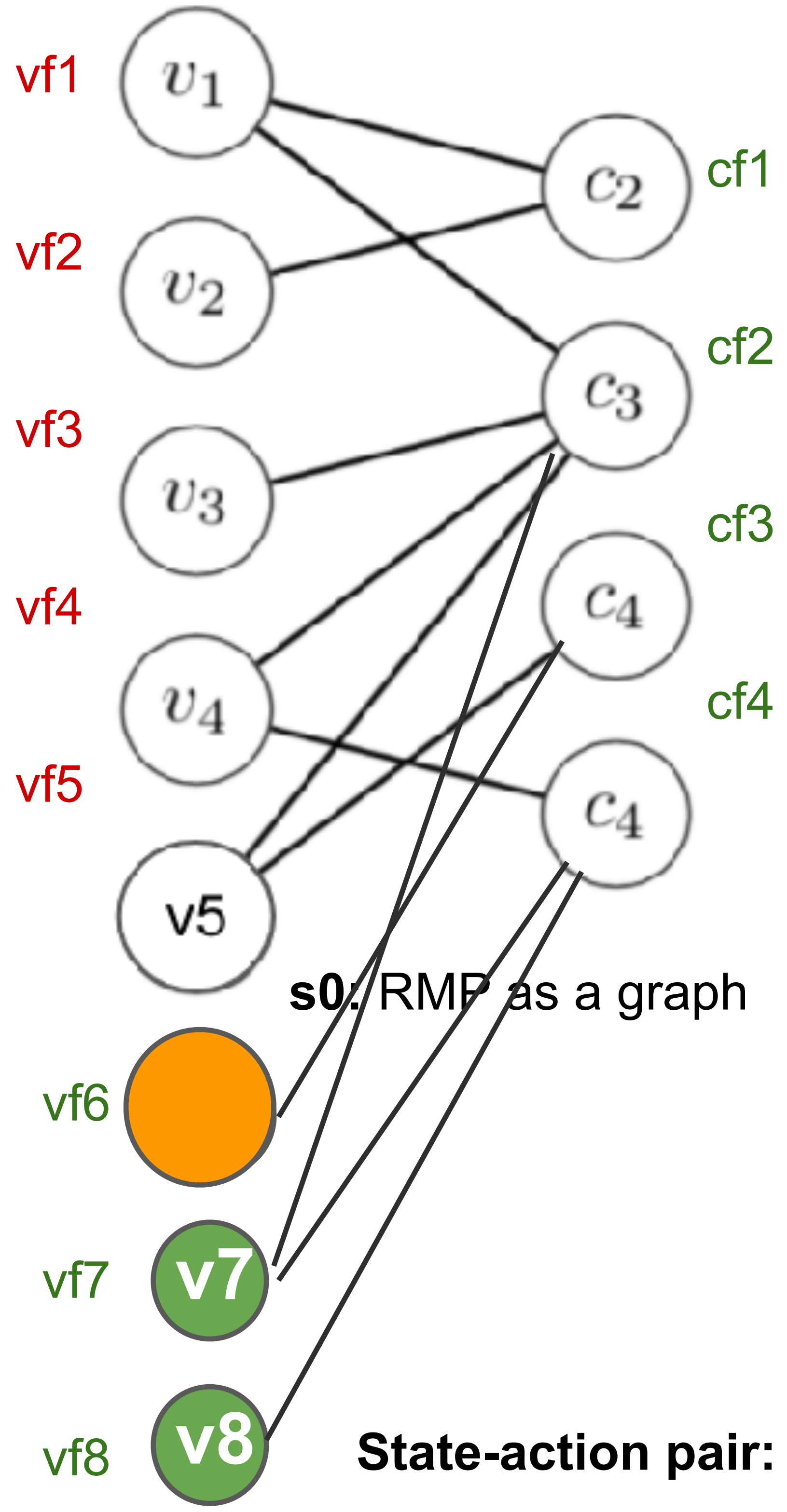
2. Solve Subproblem (SP)

get constraint nodes features

get candidate variables (actions)



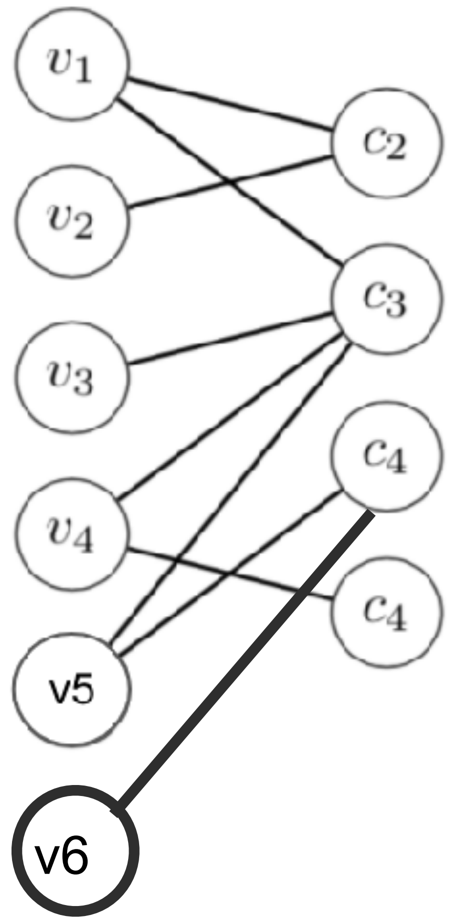
3. Input a state action pair to RL agent



s0: RMP as a graph

State-action pair: (s,a)

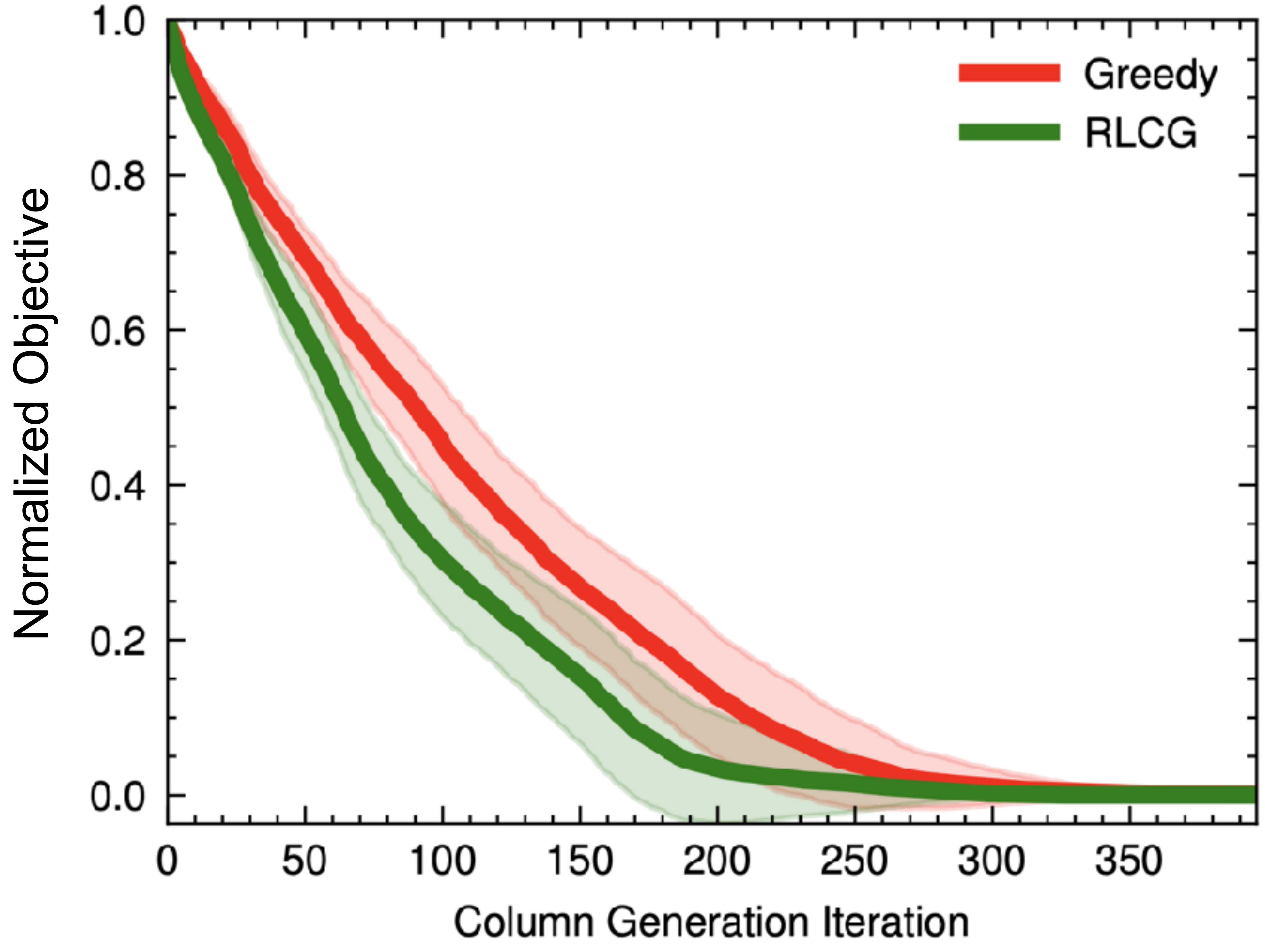
4. Evaluate Q(s,a) to select and execute a*



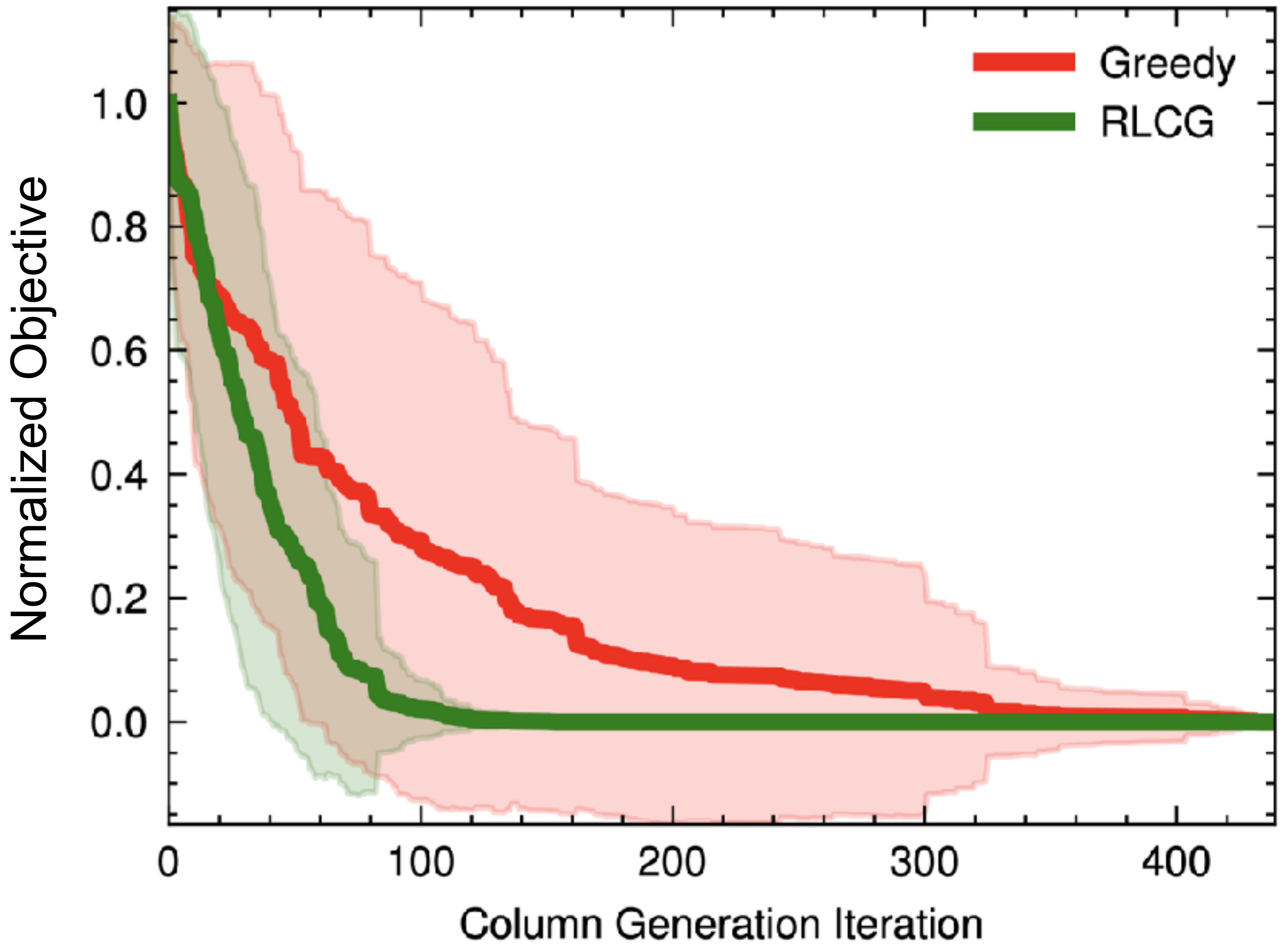
s1: RMP as a graph

Better performance & wide applicability

Cutting Stock Problem



Vehicle Routing Problem with Time Windows



Finding Backdoors to Integer Programs

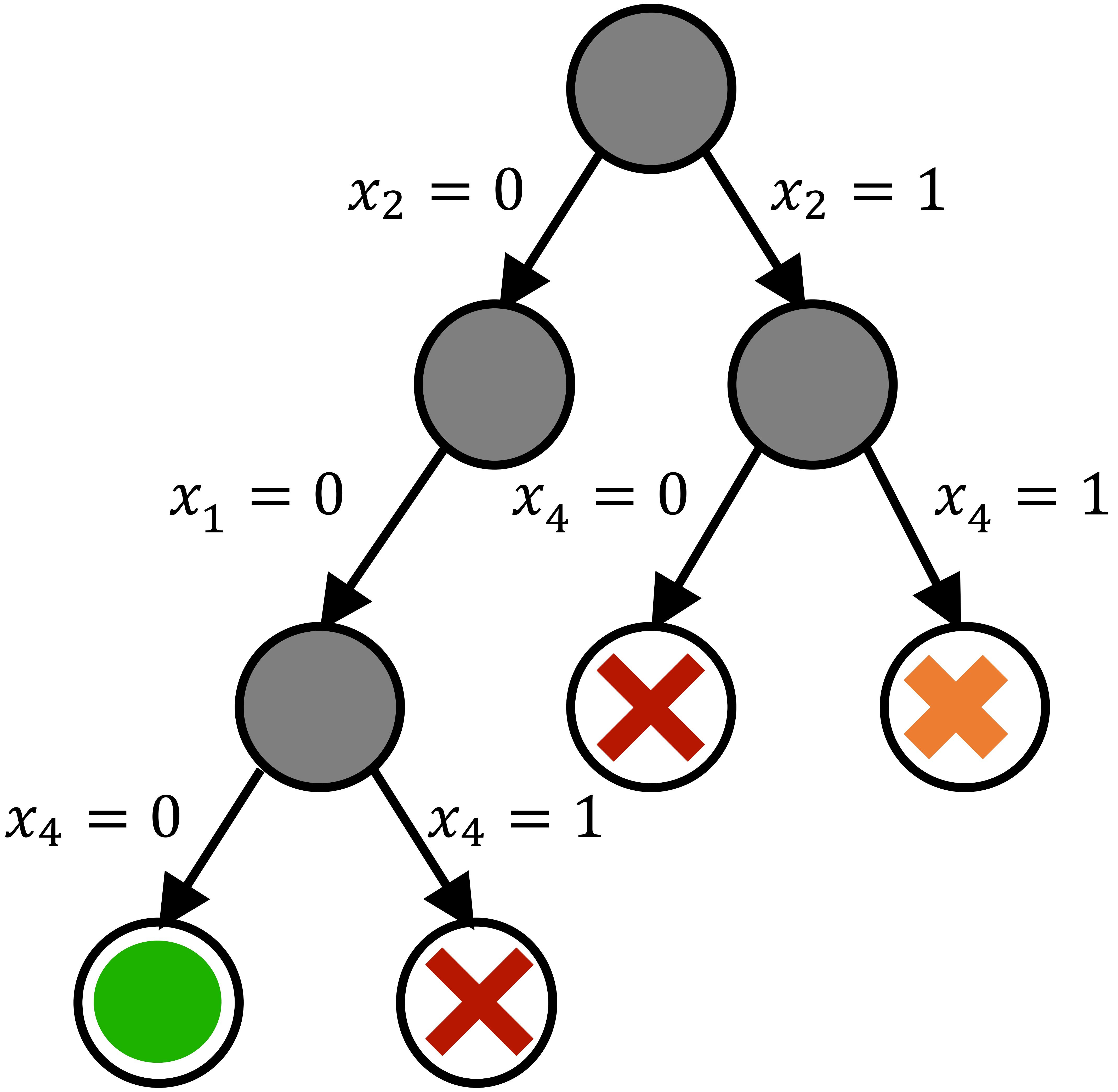
A Monte Carlo Tree Search Framework

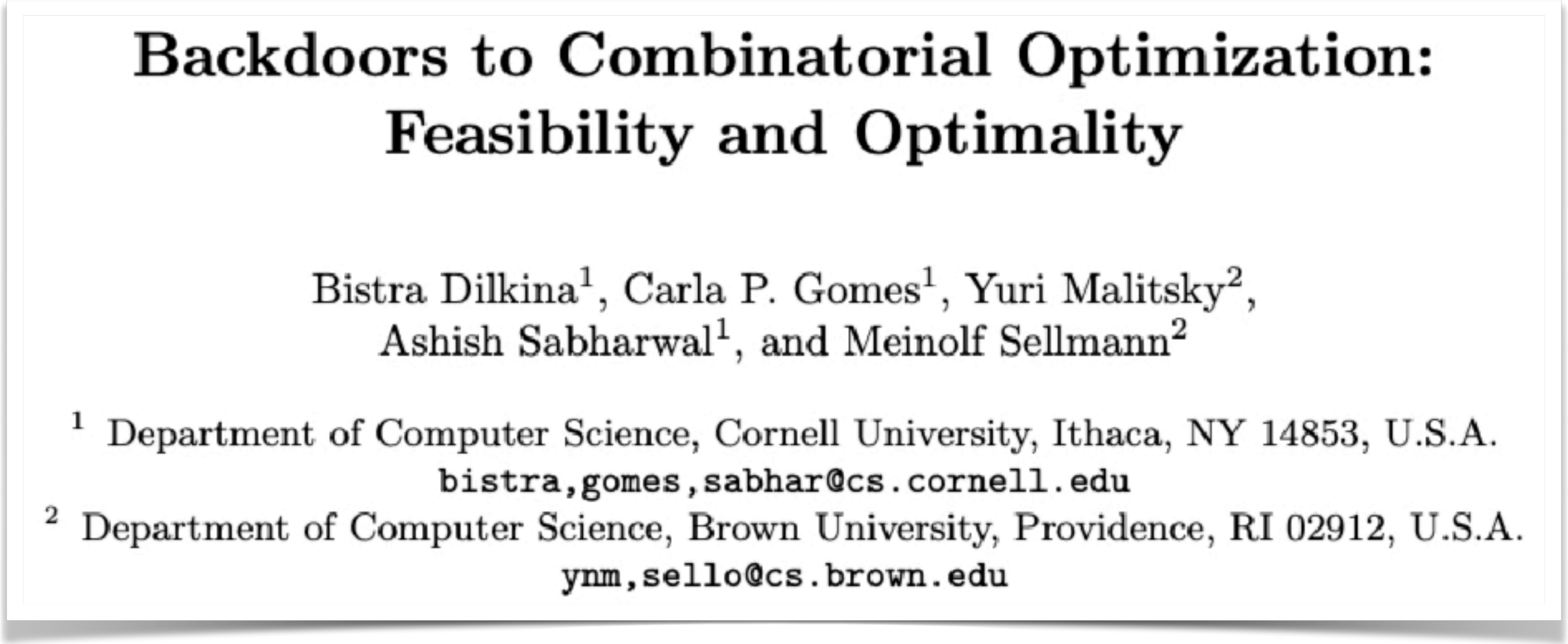
Joint work with Pashootan Vaezipoor (Toronto — CS), Bistra Dilkina (USC-CS)

AAAI 2022

arXiv:2110.08423

$$\min_x c^T x \textbf{ s.t. } Ax \leq b, x \in \{0,1\}^n, n \gg 3$$



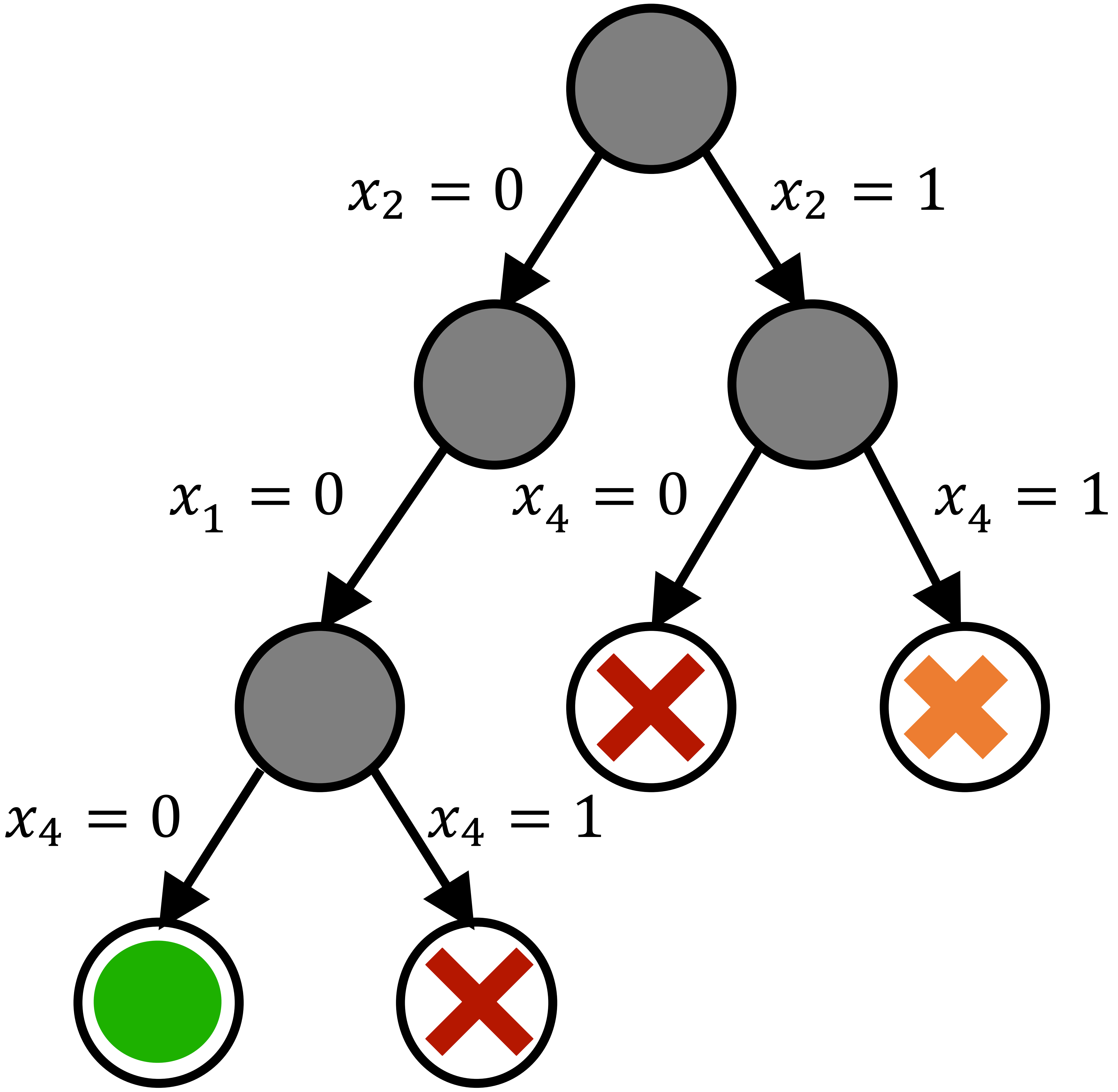

Backdoors to Combinatorial Optimization: Feasibility and Optimality

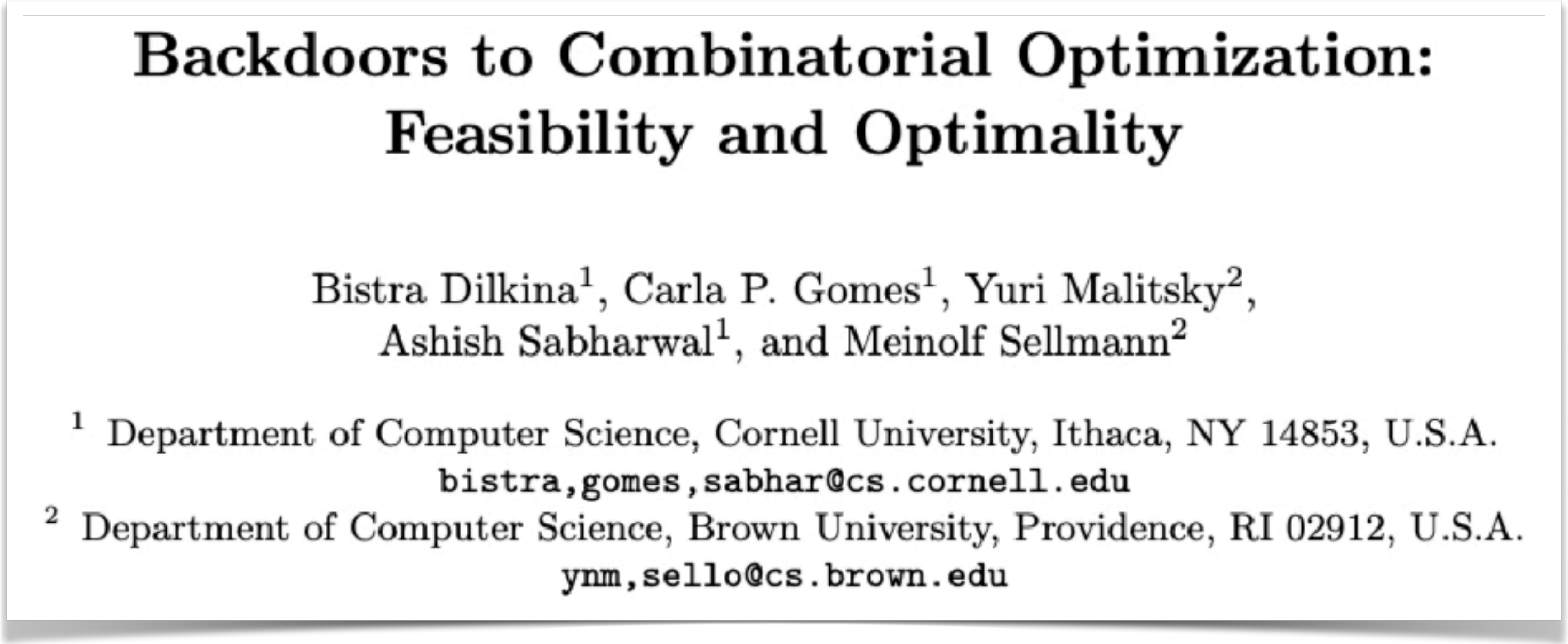
Bistra Dilkina[1], Carla P. Gomes[1], Yuri Malitsky[2], Ashish Sabharwal[1], and Meinolf Sellmann[2]

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853, U.S.A.
`bistra,gomes,sabhar@cs.cornell.edu`
[2] Department of Computer Science, Brown University, Providence, RI 02912, U.S.A.
`ynm,sello@cs.brown.edu`

Dilkina et al., CPAIOR, 2009

$$\min_x c^T x \textbf{ s.t. } Ax \leq b, x \in \{0,1\}^n, \boxed{n \gg 3}$$





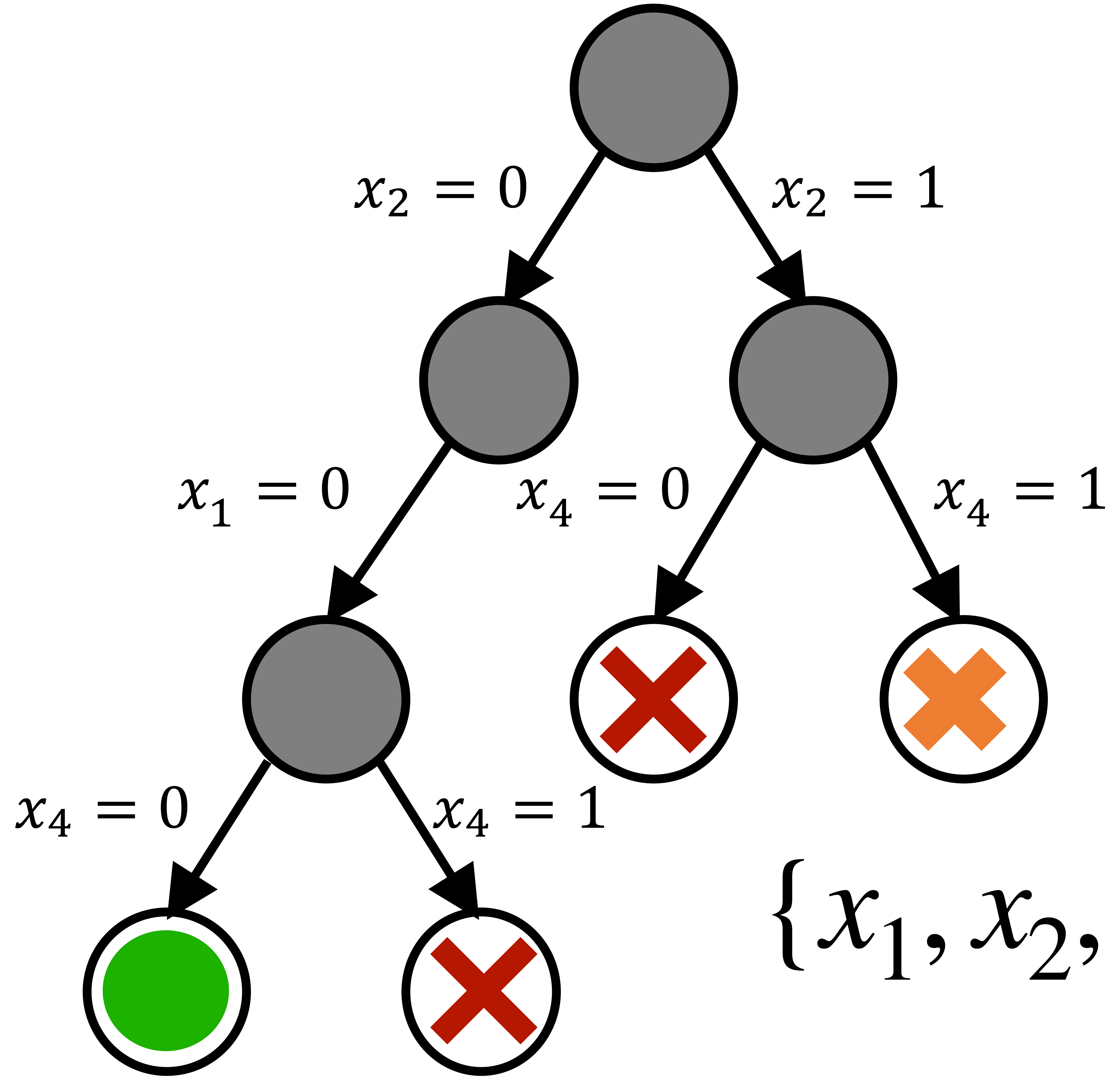
Backdoors to Combinatorial Optimization: Feasibility and Optimality

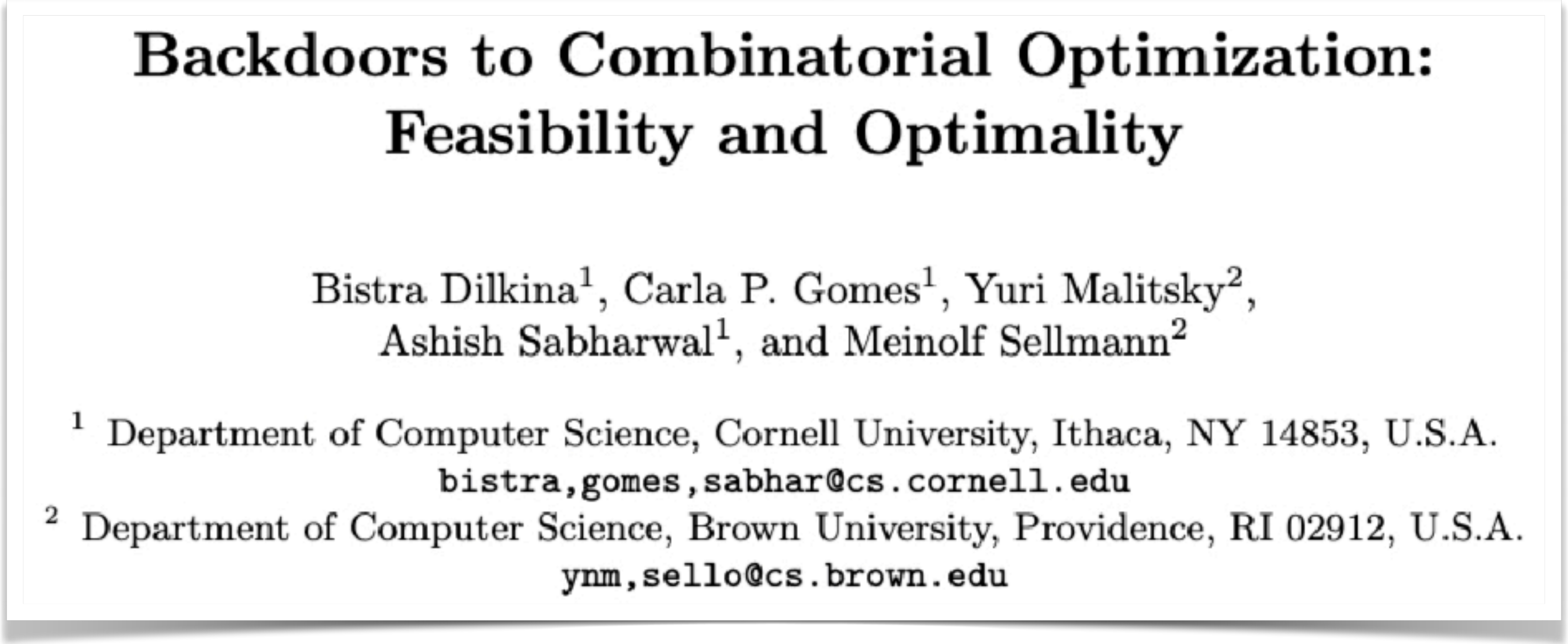
Bistra Dilkina[1], Carla P. Gomes[1], Yuri Malitsky[2], Ashish Sabharwal[1], and Meinolf Sellmann[2]

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853, U.S.A.
bistra,gomes,sabhar@cs.cornell.edu
[2] Department of Computer Science, Brown University, Providence, RI 02912, U.S.A.
ynm,sello@cs.brown.edu

Dilkina et al., CPAIOR, 2009

$$\min_x c^T x \ \textbf{s.t.}\ Ax \leq b, x \in \{0,1\}^n, \boxed{n \gg 3}$$





Backdoors to Combinatorial Optimization: Feasibility and Optimality

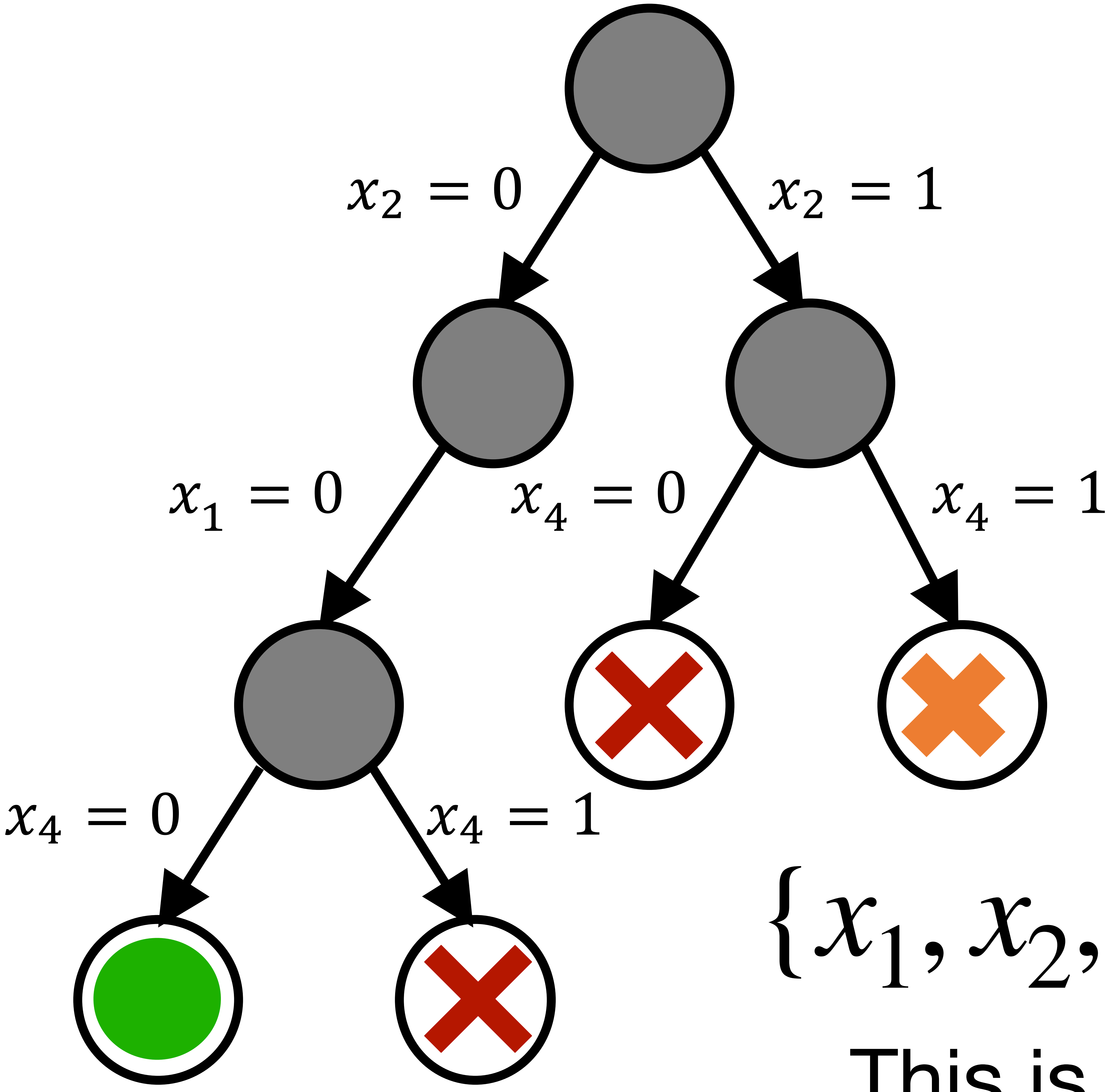
Bistra Dilkina[1], Carla P. Gomes[1], Yuri Malitsky[2], Ashish Sabharwal[1], and Meinolf Sellmann[2]

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853, U.S.A.
bistra,gomes,sabhar@cs.cornell.edu
[2] Department of Computer Science, Brown University, Providence, RI 02912, U.S.A.
ynm,sello@cs.brown.edu

Dilkina et al., CPAIOR, 2009

$\{x_1, x_2, x_4\}$ **is a Backdoor!**

$$\min_{x} c^T x \textbf{ s.t. } Ax \leq b, x \in \{0,1\}^n, \boxed{n \gg 3}$$



Backdoors to Combinatorial Optimization: Feasibility and Optimality

Bistra Dilkina[1], Carla P. Gomes[1], Yuri Malitsky[2], Ashish Sabharwal[1], and Meinolf Sellmann[2]

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853, U.S.A.
bistra,gomes,sabhar@cs.cornell.edu
[2] Department of Computer Science, Brown University, Providence, RI 02912, U.S.A.
ynm,sello@cs.brown.edu

Dilkina et al., CPAIOR, 2009

$\{x_1, x_2, x_4\}$ is a **Backdoor!**

This is a "**certificate tree**"

BamCTS: Backdoor Monte Carlo Tree Search

- **Selection (UCT)**

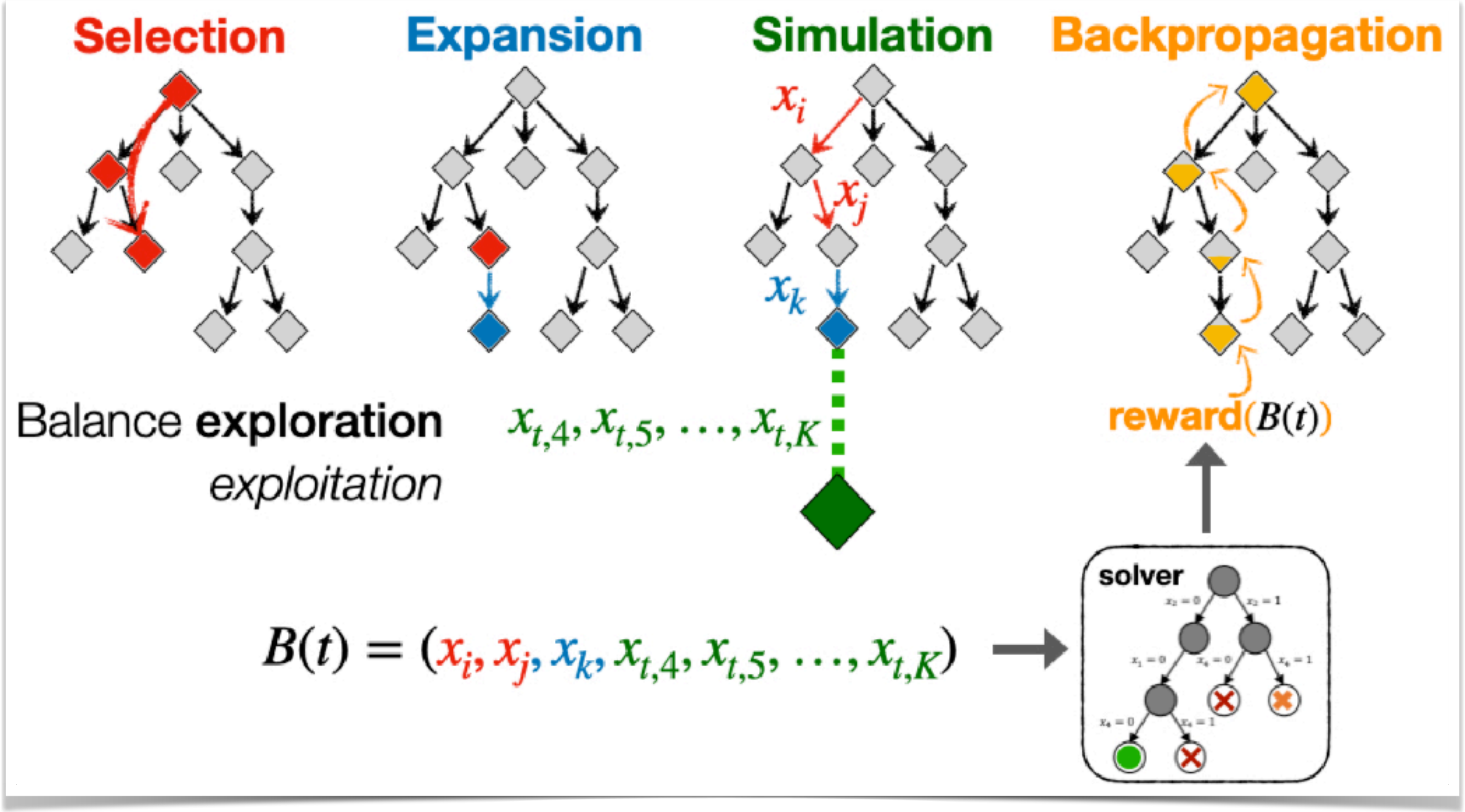
$$\mathrm{SCORE}(S, S') = (1 - \alpha_{\mathrm{PC}})\mathrm{UCT}_{\mathrm{score}}(S, S') + \alpha_{\mathrm{PC}}\hat{\mathrm{PC}}_i$$

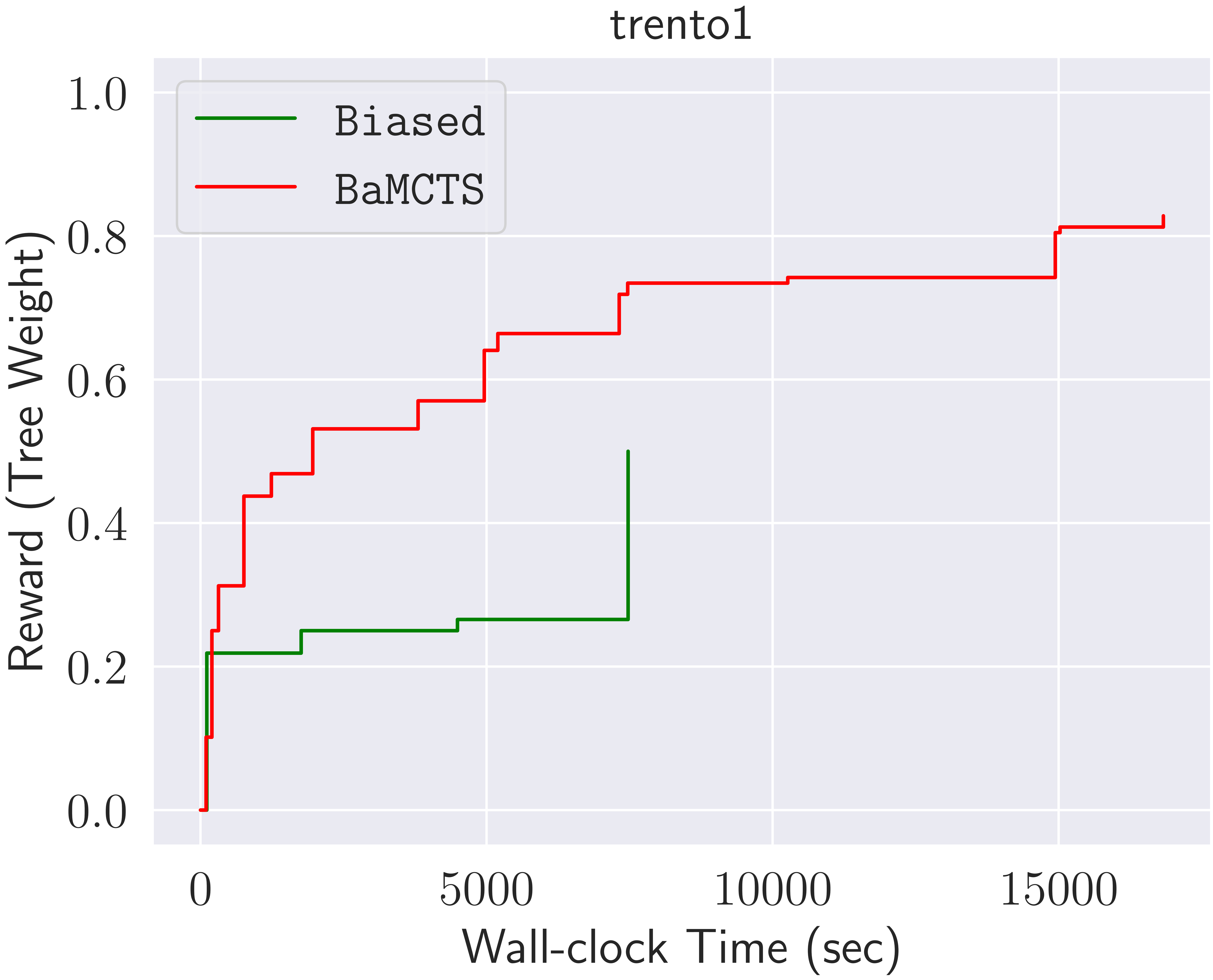
- **Expansion**: use "**progressive widening**" to get around huge branching factor by selectively expanding; results in more diving.

- **Simulation**: random sampling

- **Backpropagation**: $\boxed{\text{max or sum backup}}$

Hyperparameter





better

Evolution of best rewards during backdoor search

BaMCTS generally finds <u>better</u> solutions <u>faster</u> than (biased) **sampling**

Does branching on backdoors improve MIP solving?

Experimental Setup: 115/142 instances for which BaMCTS found backdoor with non-zero reward.

Solver: CPLEX 12.10, single-threaded, 1-hour time limit, no "dynamic search", with primal heuristics.

Takeaways: 700-1100 fewer B&B nodes, slight reduction in time, smaller gaps for hard instances

	seed 1 (47, 56, 4)		seed 2 (45, 61, 7)		seed 3 (46, 60, 6)	
	CPX-def	CPX-BaMCTS	CPX-def	CPX-BaMCTS	CPX-def	CPX-BaMCTS
# of nodes (solved by both)	6902.74	**6097.70**	6173.84	**5030.13**	7744.98	**7001.99**
total time (solved by both)	179.53	**175.97**	156.61	**138.34**	169.41	**156.83**
optimality gap (not solved by either)	23/56	**33/56**	24/61	**37/61**	20/60	**40/60**

Concluding Remarks

Reinforcement Learning is an effective tool for designing algorithms for combinatorial optimization in many of its flavours!

1. RL for Online Combinatorial Optimization (TMLR-22)
2. RL for Offline Graph Optimization (NeurIPS-17)
3. MCTS for Integer Programming (without learning) (AAAI-22)
4. RL for Large-Scale Linear Programming (NeurIPS-22)