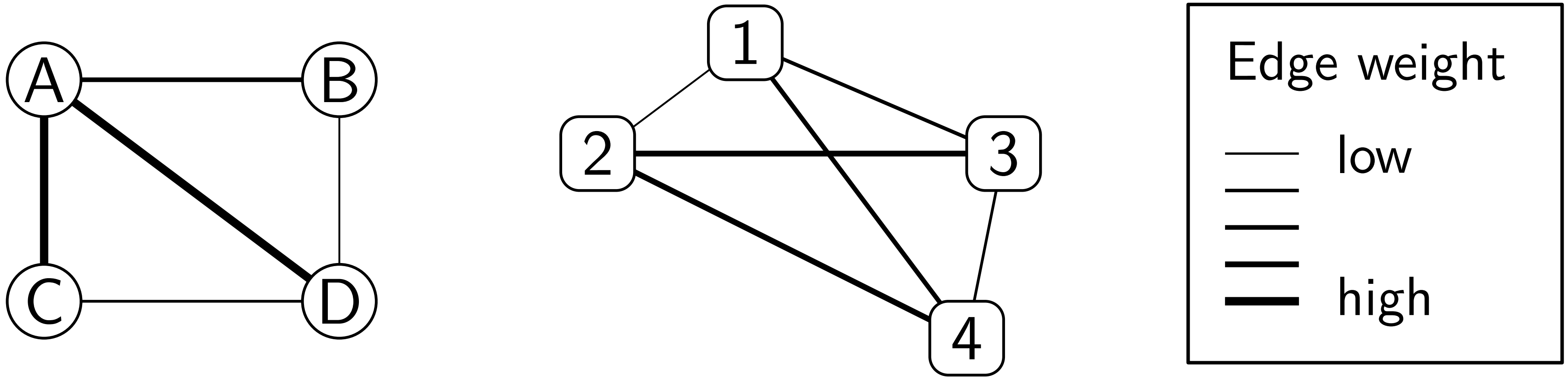
# Using Reinforcement Learning to solve Quadratic Assignment Problems

Proposal Talk
Tim Göttlicher

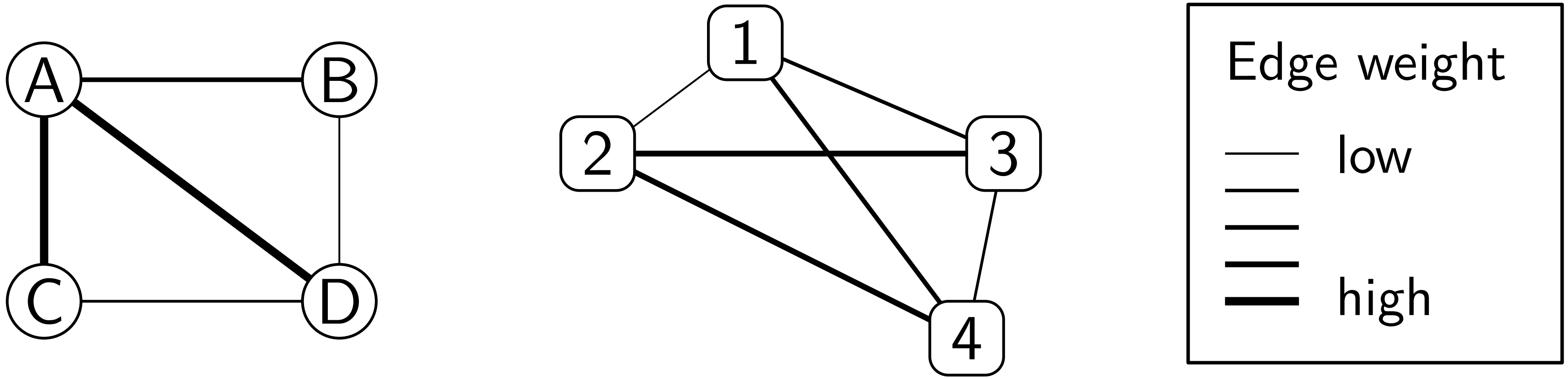
Supervisors
Prof Dr. Verena Wolf
Dr. Andreas Karrenbauer
Joschka Groß

04.03.2022

# The Quadratic Assignment Problem

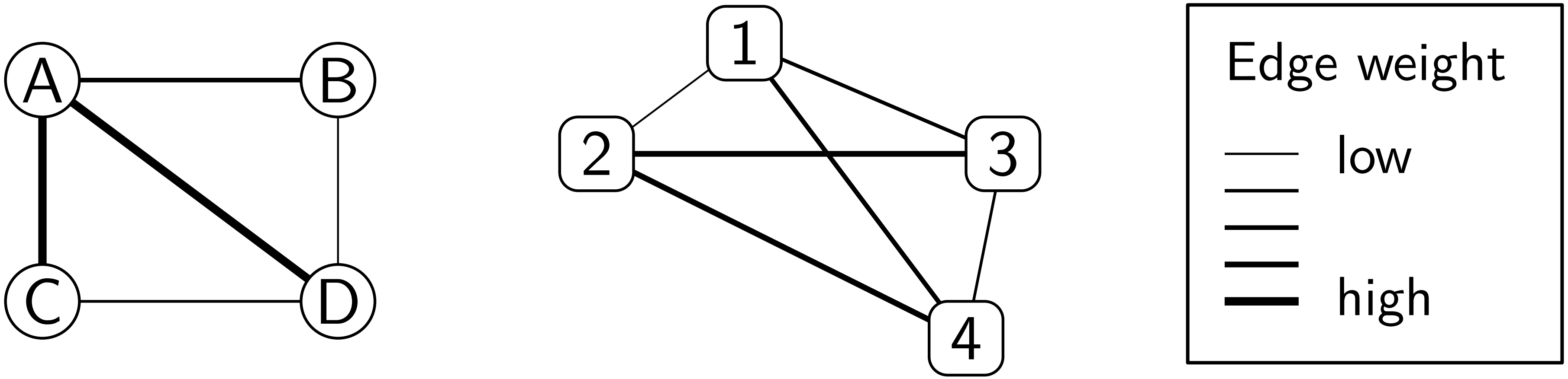


# The Quadratic Assignment Problem




Example: Economics

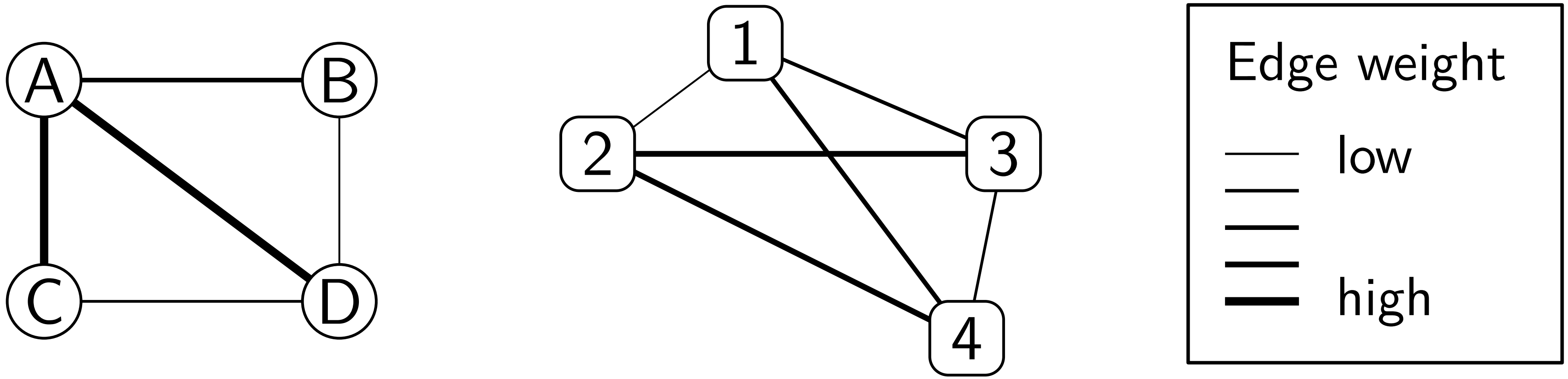
# The Quadratic Assignment Problem




## Example: Economics

Transport volume
between facilities

# The Quadratic Assignment Problem

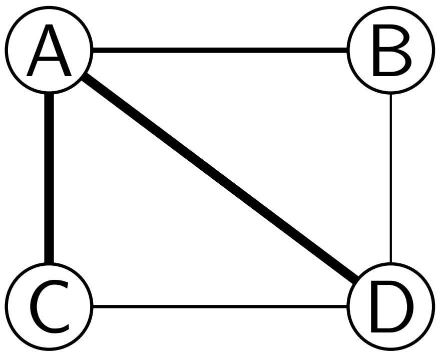


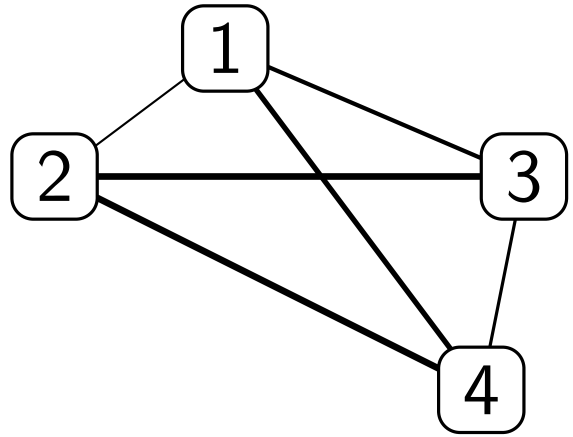

Example: Economics

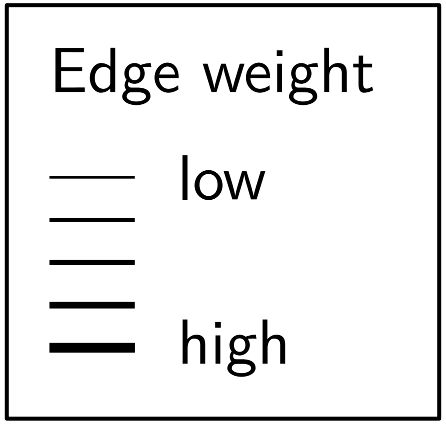
Transport volume
between facilities

Cost per unit
between locations

# The Quadratic Assignment Problem



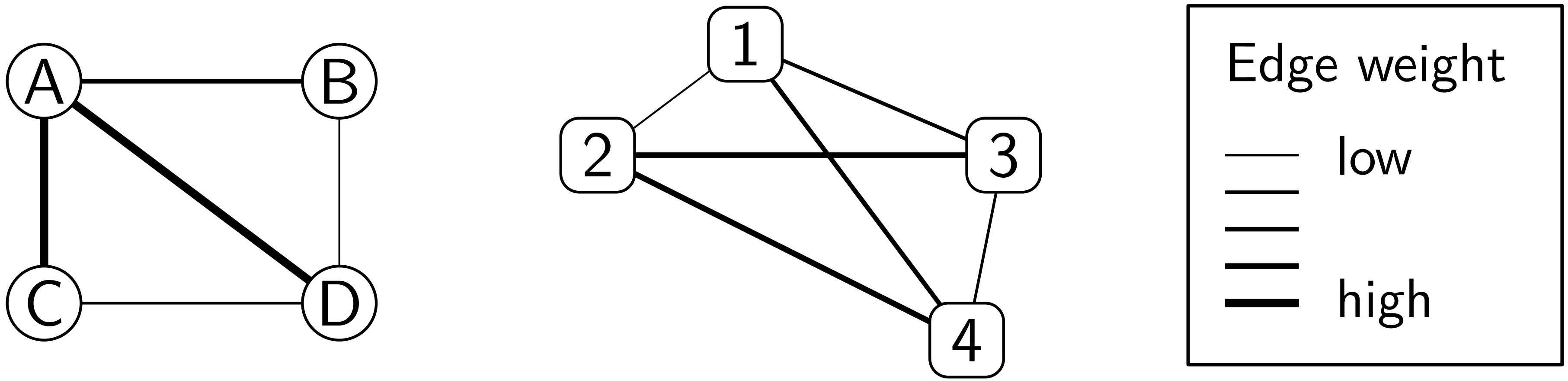

## Example: Economics

Transport volume between facilities

Cost per unit between locations

## Example: Keyboard layout

# The Quadratic Assignment Problem




## Example: Economics

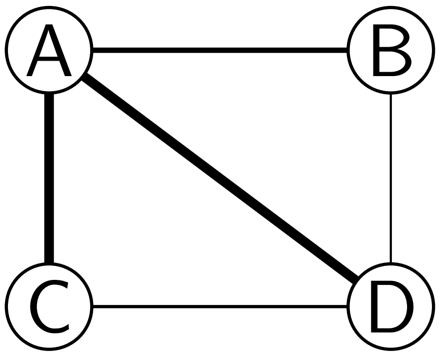
Transport volume between facilities

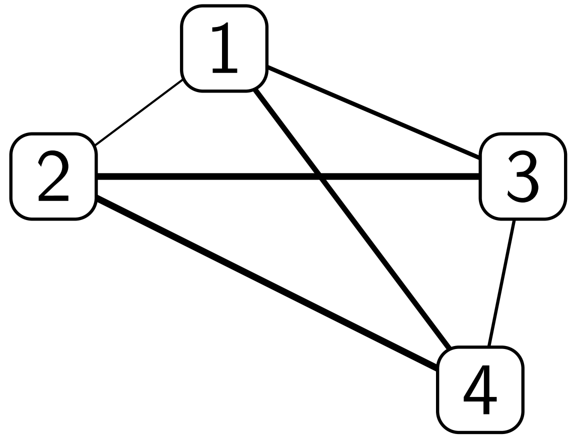
Cost per unit between locations

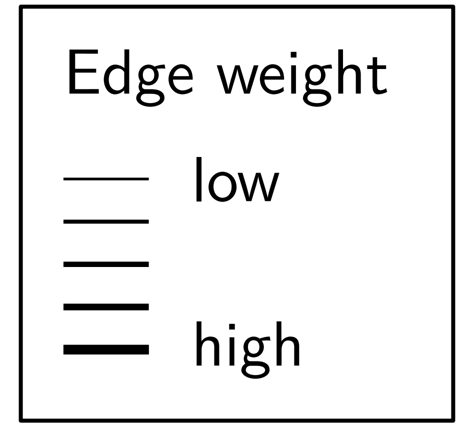
## Example: Keyboard layout

Letter pair frequency

# The Quadratic Assignment Problem




## Example: Economics

Transport volume between facilities

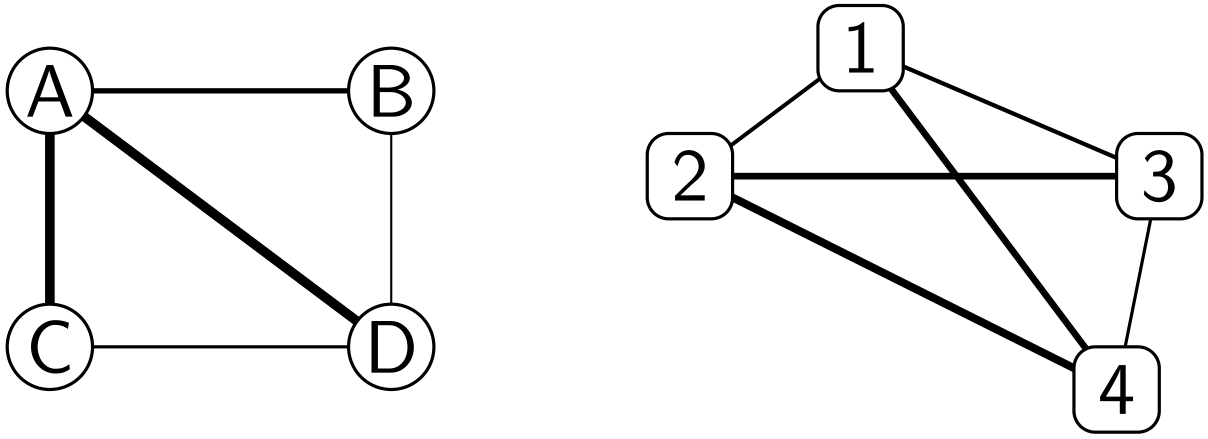
Cost per unit between locations

## Example: Keyboard layout

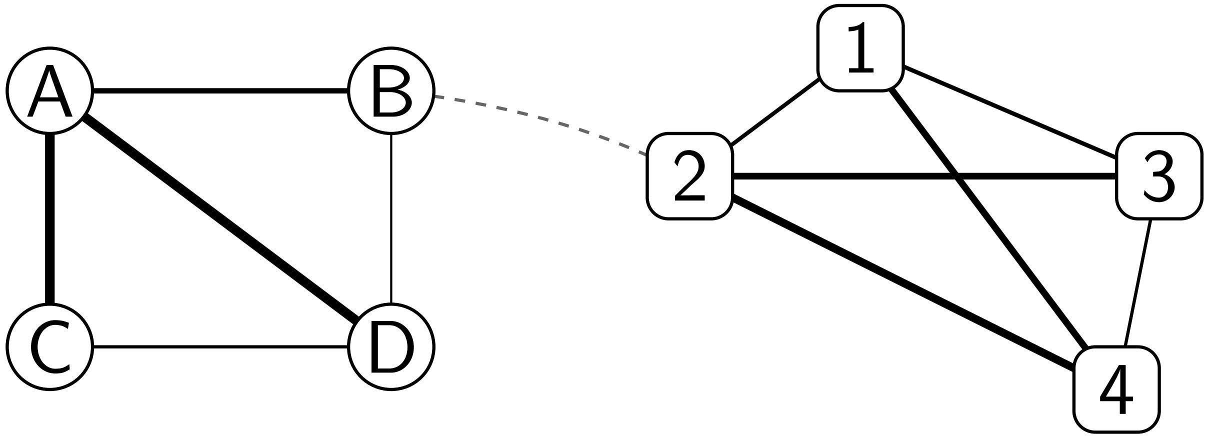
Letter pair frequency

Travel time between keys

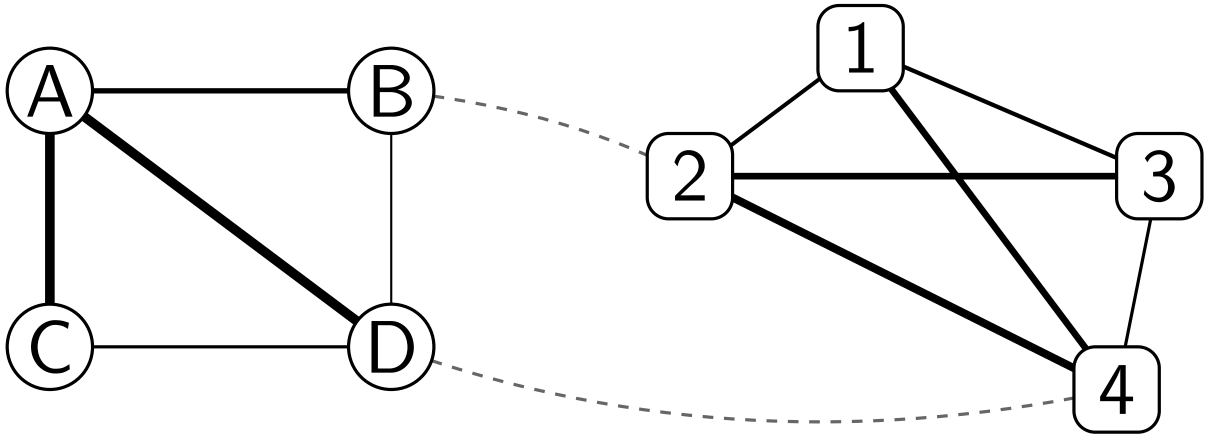
# Computing the cost of an assignment



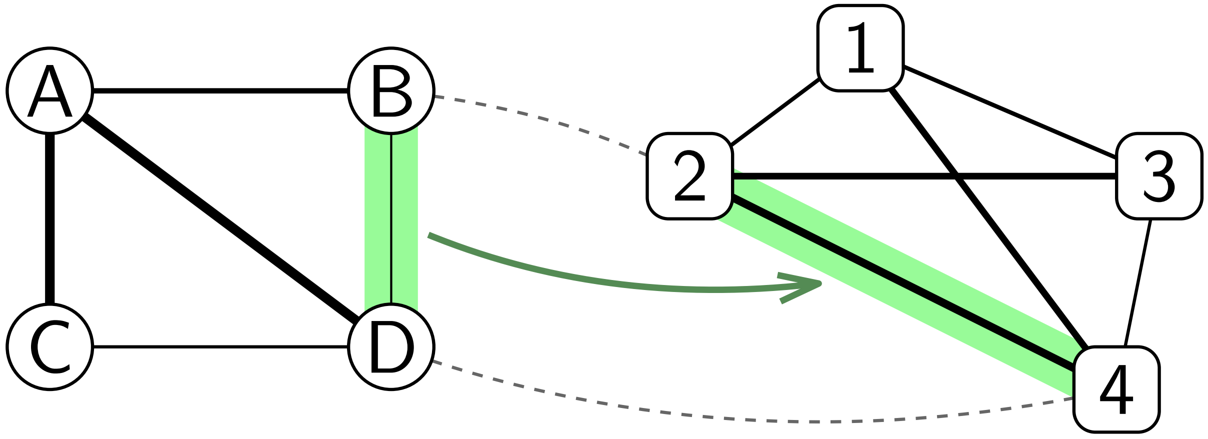
# Computing the cost of an assignment



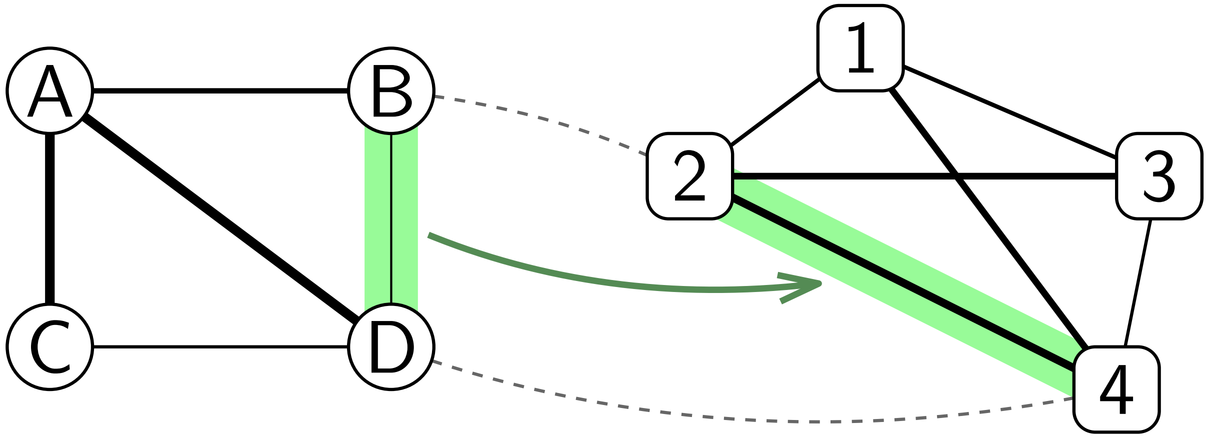
# Computing the cost of an assignment

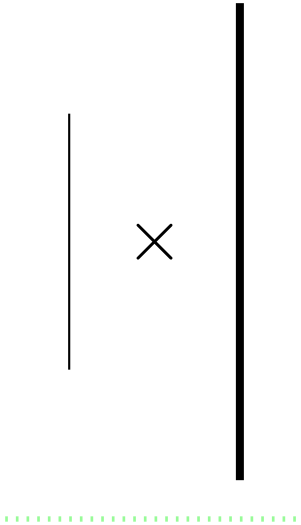


# Computing the cost of an assignment



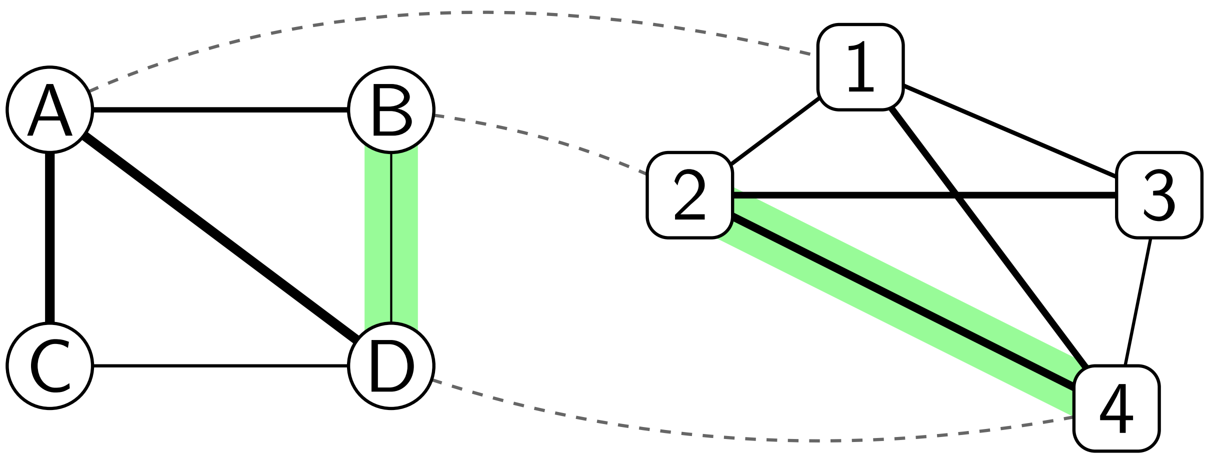
# Computing the cost of an assignment

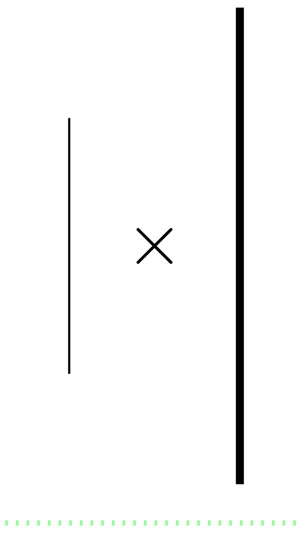


$$1 \cdot 5$$

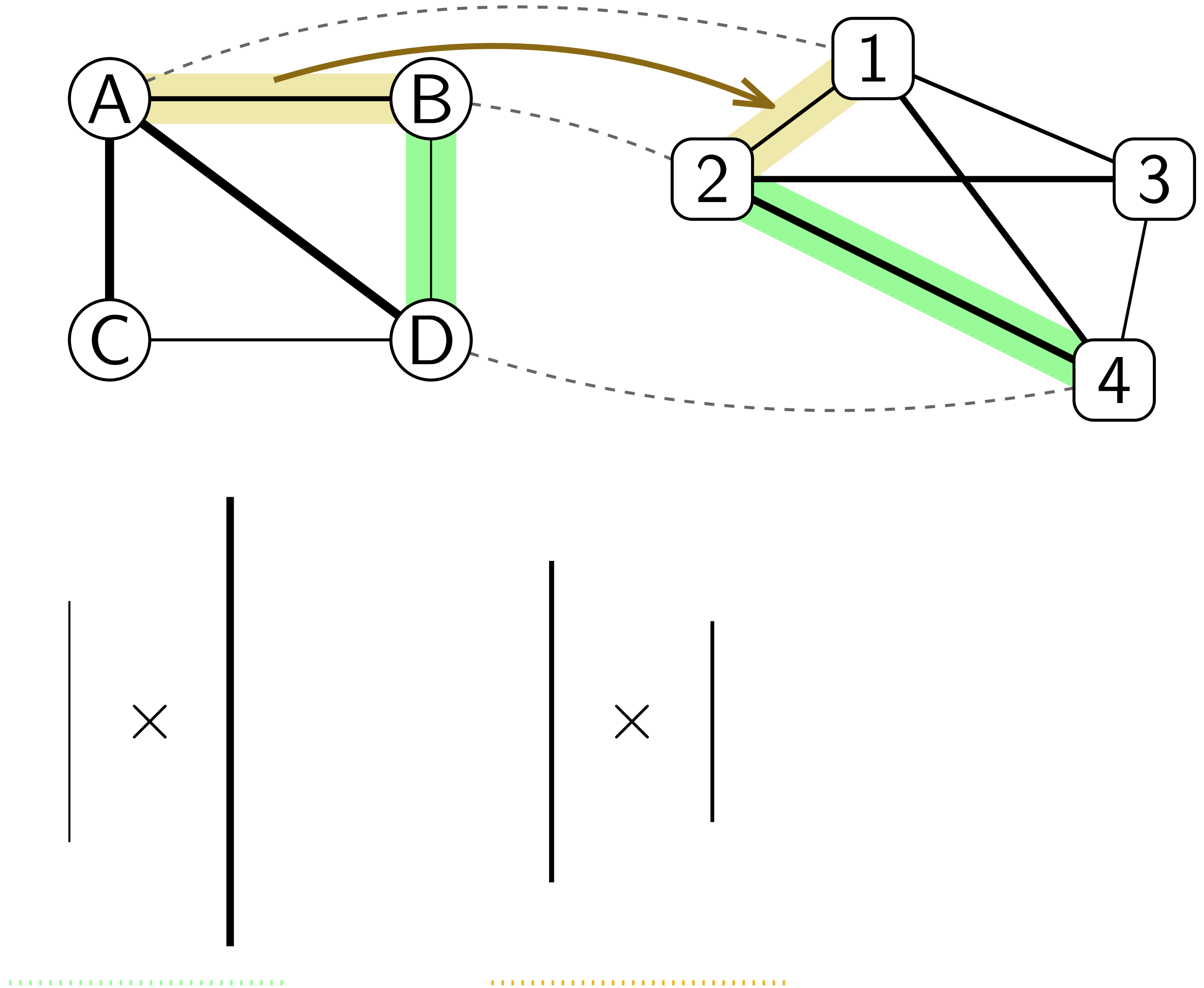
# Computing the cost of an assignment



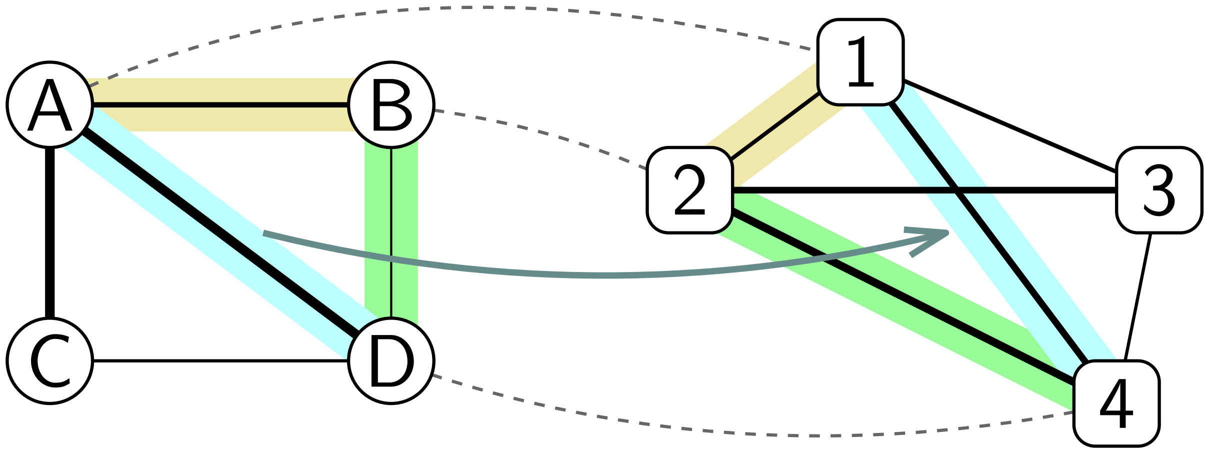
$$1 \quad \cdot \quad 5$$

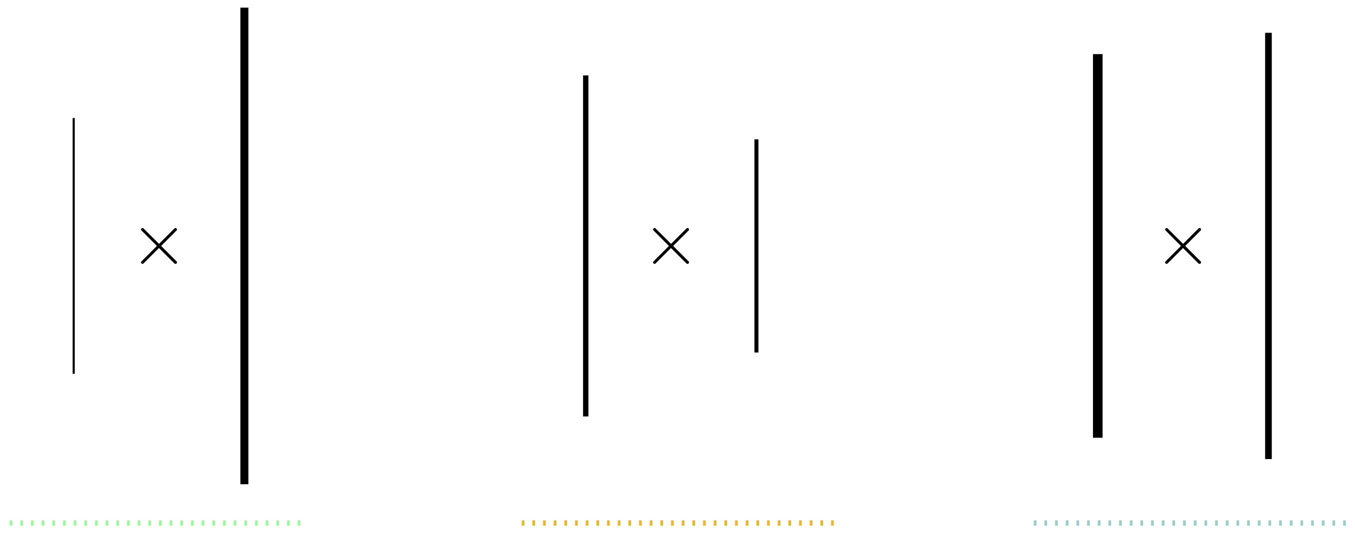
# Computing the cost of an assignment




$$1 \cdot 5 \quad + \quad 2 \cdot 1.5$$

# Computing the cost of an assignment



$$1 \cdot 5 \quad + \quad 2 \cdot 1.5 \quad + \quad 5 \cdot 4$$

# Mathematical formulation

$$\sum_{i,j} a_{i,j} b_{f(i),f(j)}$$

# Mathematical formulation

$$\sum_{i,j} a_{i,j} b_{f(i),f(j)}$$

edges in left graph

# Mathematical formulation

$$\sum_{i,j} a_{i,j} b_{f(i),f(j)}$$

edges in left graph

assigned edge in right graph

# Mathematical formulation

edge weights

$$\sum_{i,j} a_{i,j} b_{f(i),f(j)}$$

edges in left graph

assigned edge in right graph

# Mathematical formulation

edge weights

$$\sum_{i,j} a_{i,j} b_{f(i),f(j)}$$

edges in left graph

assigned edge in right graph

**Goal**

Find assignment $f$ that minimizes cost

# QAP is hard

# QAP is hard *NP-hard

# QAP is hard *NP-hard

## Unsolved problems in QAPLIB

| |
|---|
| **Thonemann and Bölte (1994)** |
| Tho40 (n = 40) |
| Tho150 (n = 150) |

| |
|---|
| **Wilhelm and Ward (1987)** |
| Wil50 (n = 50) |
| Wil100 (n = 100) |

| |
|---|
| **Taillard (1991, 1995)** |
| Tai35a (n = 35) |
| Tai40a (n = 40) |

...

# QAP is hard *NP-hard

Unsolved problems in QAPLIB

| |
|---|
| **Thonemann and Bölte (1994)** |
| Tho40 (n = 40) |
| Tho150 (n = 150) |
| **Wilhelm and Ward (1987)** |
| Wil50 (n = 50) |
| Wil100 (n = 100) |
| **Taillard (1991, 1995)** |
| Tai35a (n = 35) |
| Tai40a (n = 40) |

...

**Goal**

Find good assignment within time constraints

# QAP is hard *NP-hard

Unsolved problems in QAPLIB

| |
|---|
| **Thonemann and Bölte (1994)** |
| Tho40 (n = 40) |
| Tho150 (n = 150) |
| **Wilhelm and Ward (1987)** |
| Wil50 (n = 50) |
| Wil100 (n = 100) |
| **Taillard (1991, 1995)** |
| Tai35a (n = 35) |
| Tai40a (n = 40) |

...

> **Goal**
> Find good assignment within time constraints

$\rightarrow$ Learn heuristics with RL

# Reinforcement learning approach

Necessary for (deep) RL:

1. suitable environment
2. policy function

# Reinforcement learning approach

Necessary for (deep) RL:

1. suitable environment
2. policy function

Deterministic MDP as an environment:

State
Initial state
Actions
Reward
Next state

# Reinforcement learning approach

Necessary for (deep) RL:

1. suitable environment
2. policy function

Deterministic MDP as an environment:

| | |
|---|---|
| State | Graph representation of the problem |
| Initial state | |
| Actions | |
| Reward | |
| Next state | |

# Reinforcement learning approach

Necessary for (deep) RL:

1. suitable environment
2. policy function

Deterministic MDP as an environment:

| | |
|---|---|
| State | Graph representation of the problem |
| Initial state | Unassigned QAP from a training set |
| Actions | |
| Reward | |
| Next state | |

# Reinforcement learning approach

Necessary for (deep) RL:

1. suitable environment
2. policy function

Deterministic MDP as an environment:

| | |
|---|---|
| State | Graph representation of the problem |
| Initial state | Unassigned QAP from a training set |
| Actions | Pairs of nodes |
| Reward | |
| Next state | |

# Reinforcement learning approach

Necessary for (deep) RL:

1. suitable environment
2. policy function

Deterministic MDP as an environment:

| | |
|---|---|
| State | Graph representation of the problem |
| Initial state | Unassigned QAP from a training set |
| Actions | Pairs of nodes |
| Reward | Negative cost of assignment |
| Next state | |

# Reinforcement learning approach

Necessary for (deep) RL:

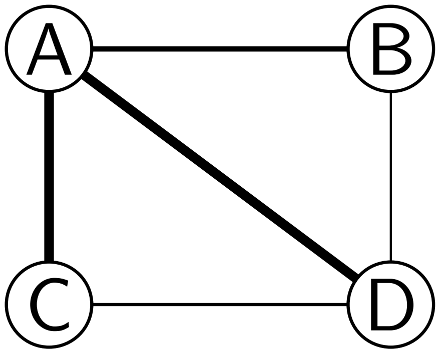
1. suitable environment
2. policy function

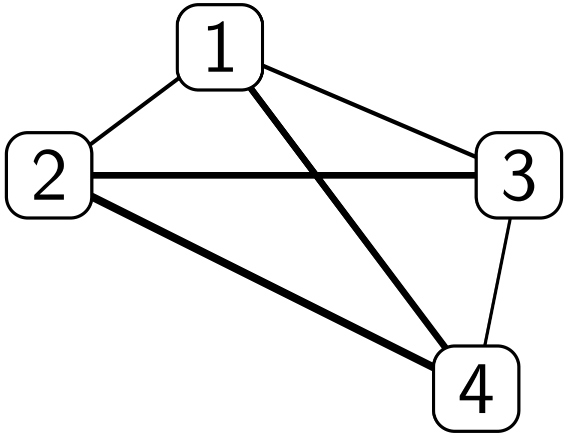
Deterministic MDP as an environment:

| | |
|---|---|
| State | Graph representation of the problem |
| Initial state | Unassigned QAP from a training set |
| Actions | Pairs of nodes |
| Reward | Negative cost of assignment |
| Next state | Remaining problem after assignment |

# Representing partial assignments

Options:

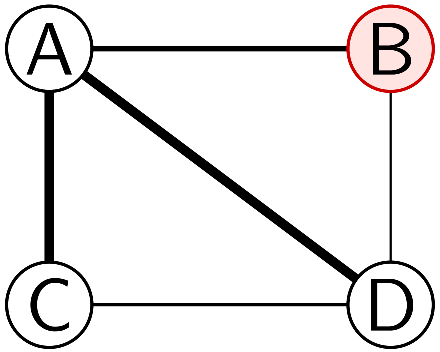


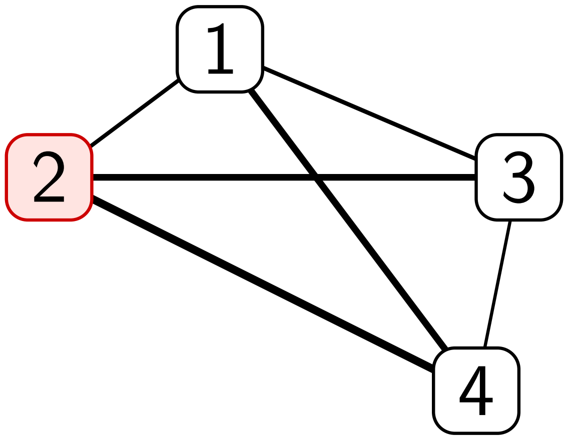



# Representing partial assignments

Options:

- Add a binary feature to the node

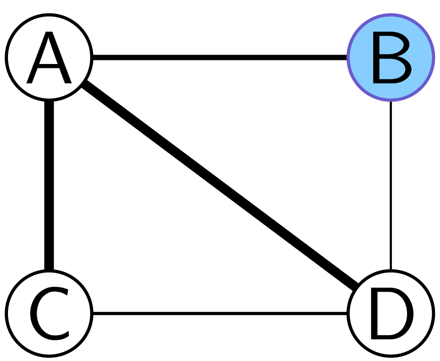


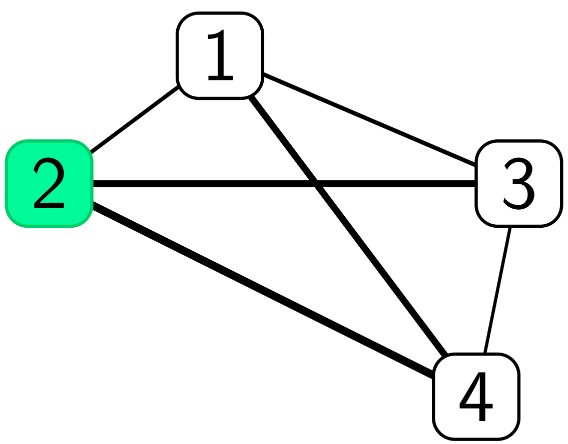



# Representing partial assignments

Options:

- Add a binary feature to the node
- Use special network to encode nodes

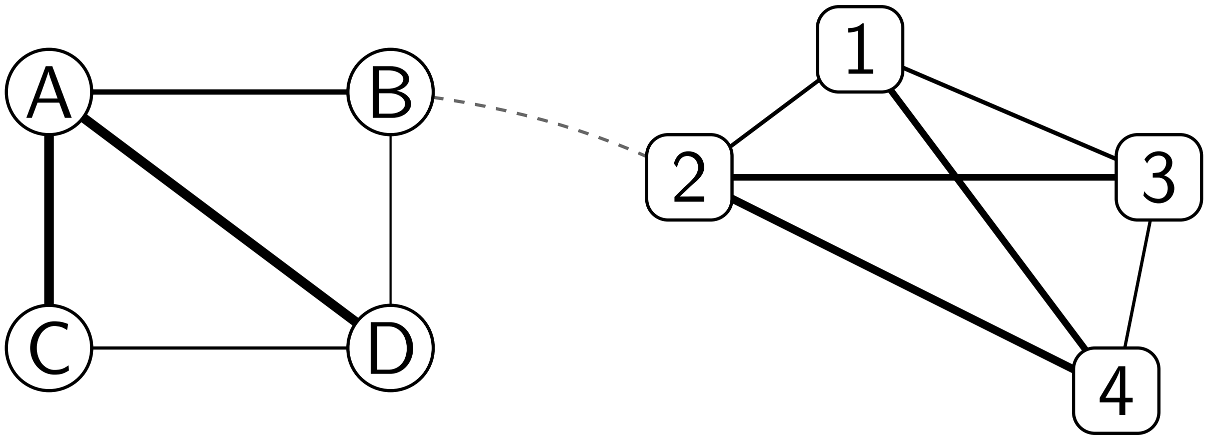


# Representing partial assignments

Options:

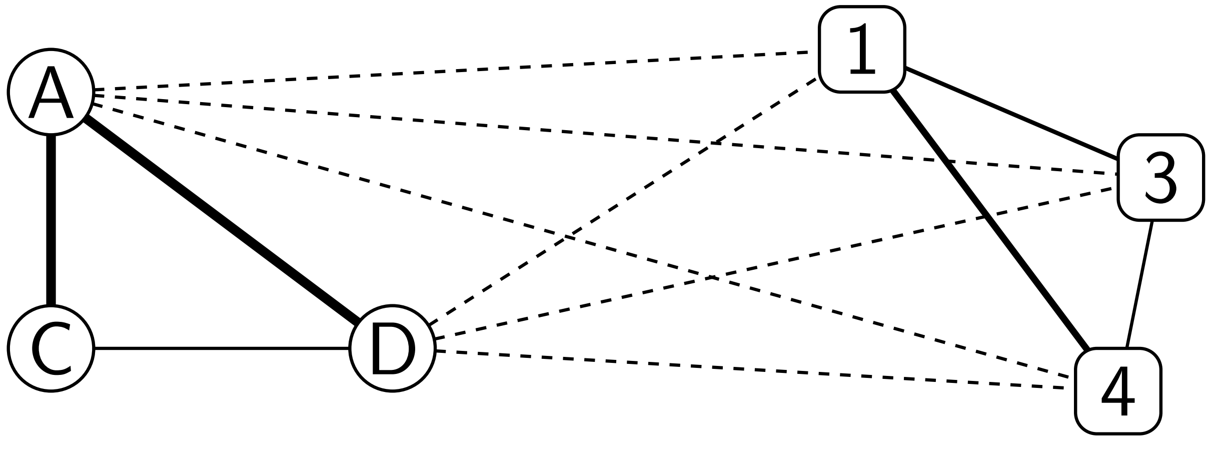
- Add a binary feature to the node
- Use special network to encode nodes
- Add a new edge between the graphs



# Representing partial assignments

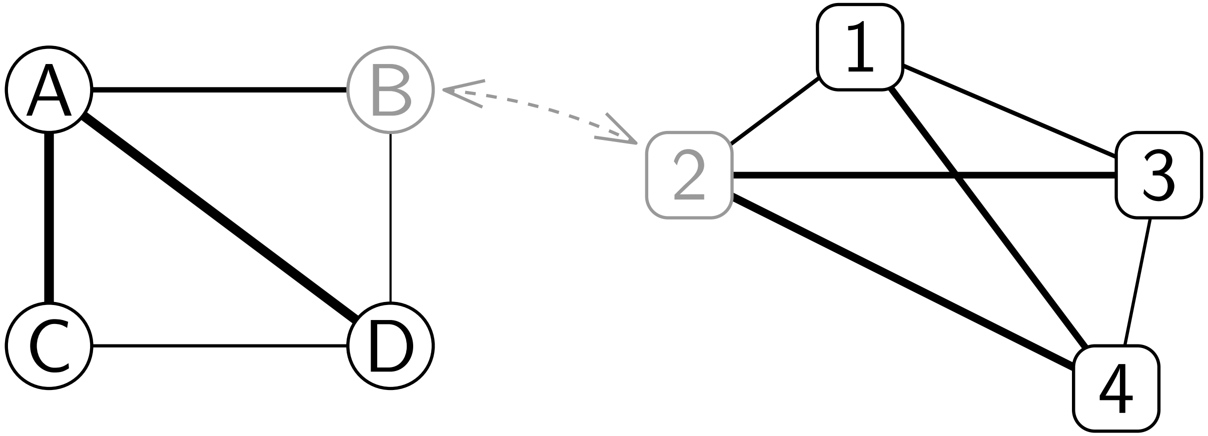
Options:

- Add a binary feature to the node
- Use special network to encode nodes
- Add a new edge between the graphs
- Compute an equivalent subproblem for the assignment



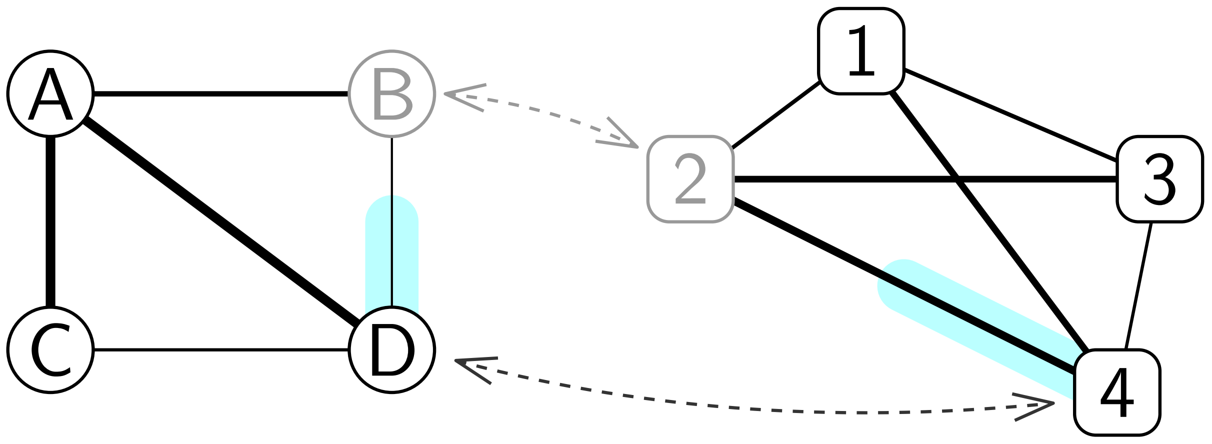
# Reduced subproblem representation

Can we remove the assigned nodes from the graph?



# Reduced subproblem representation

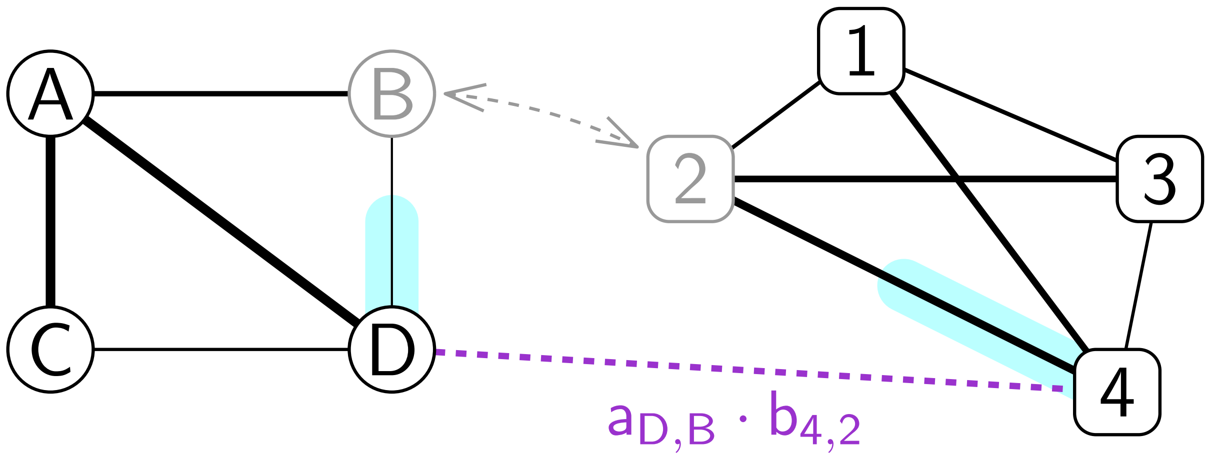
Can we remove the assigned nodes from the graph?




Assignment cost based on half assigned edges

# Reduced subproblem representation

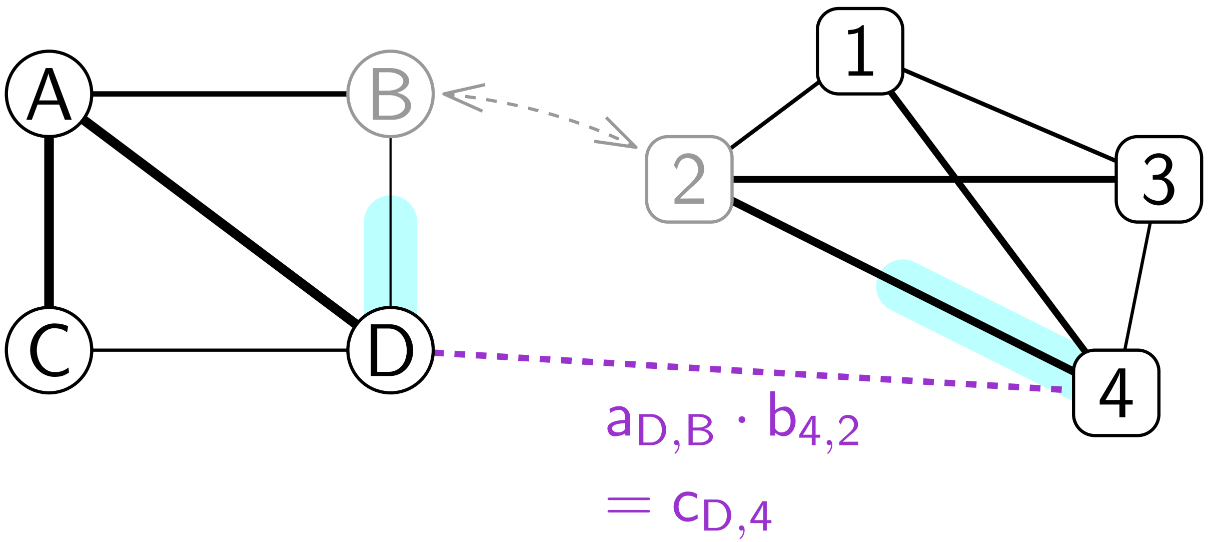
Can we remove the assigned nodes from the graph?




Assignment cost based on half assigned edges

# Reduced subproblem representation

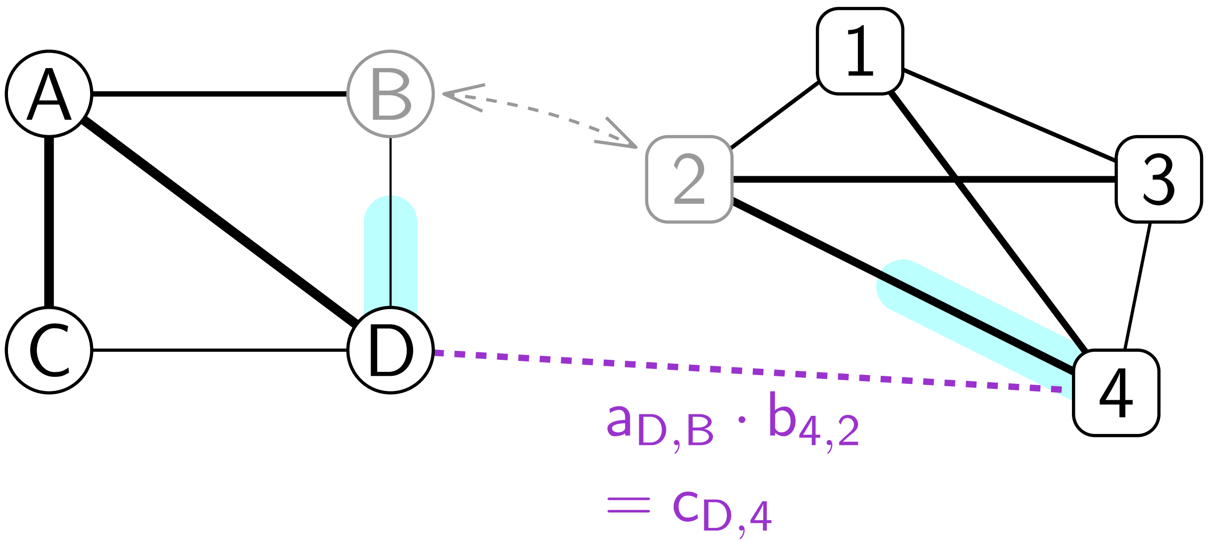
Can we remove the assigned nodes from the graph?




Assignment cost based on half assigned edges

# Reduced subproblem representation

Can we remove the assigned nodes from the graph?

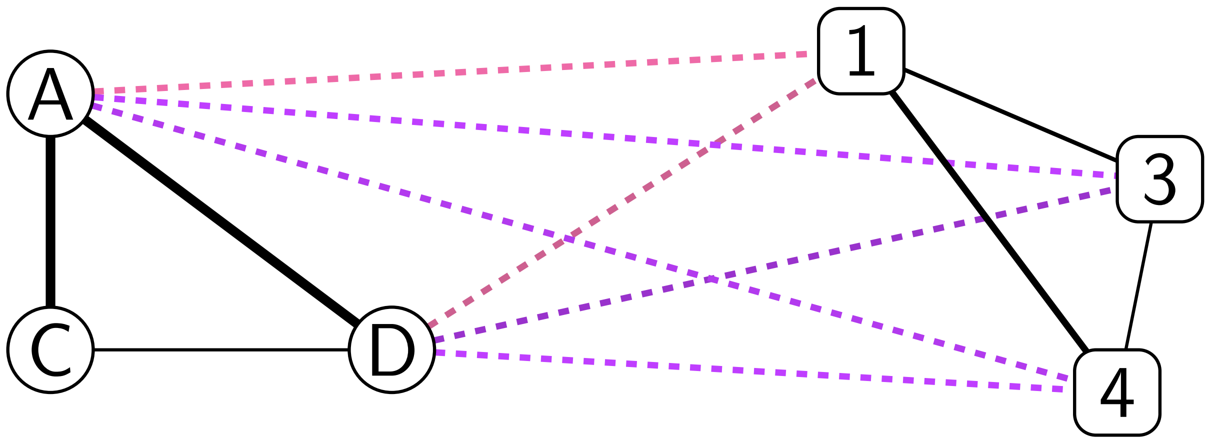



Assignment cost based on half assigned edges

$\rightarrow$ Encode in new edge type between graphs

# Reduced subproblem representation

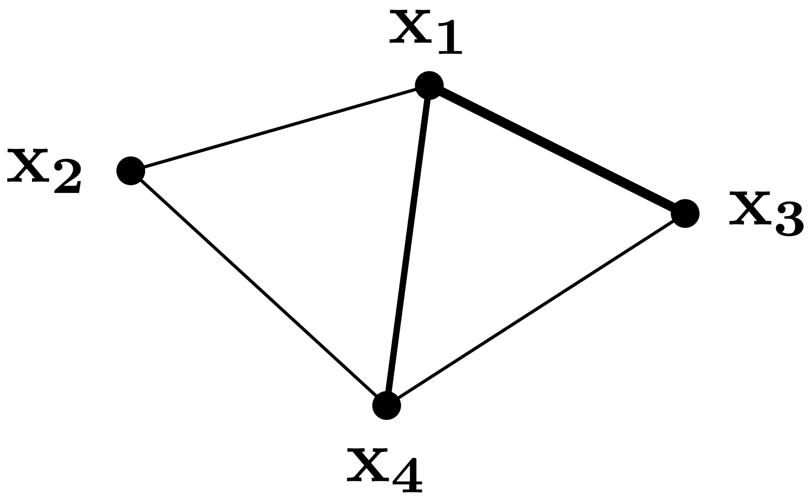
Can we remove the assigned nodes from the graph?



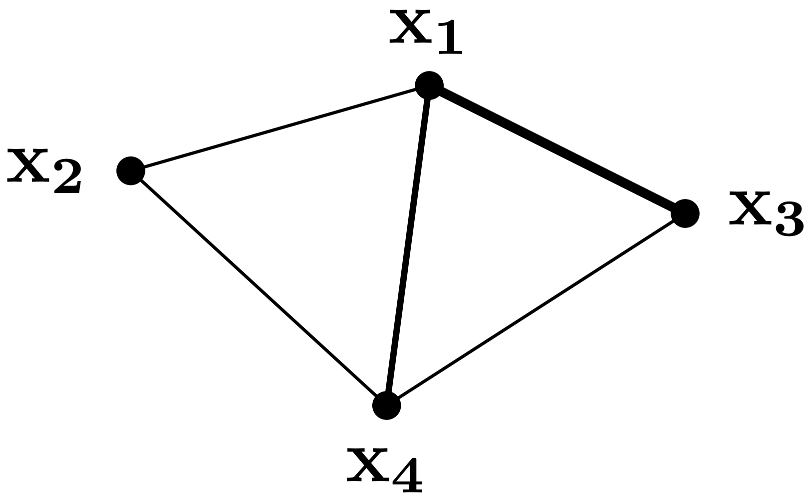

Assignment cost based on half assigned edges

$\rightarrow$ Encode in new edge type between graphs

# Graph neural networks



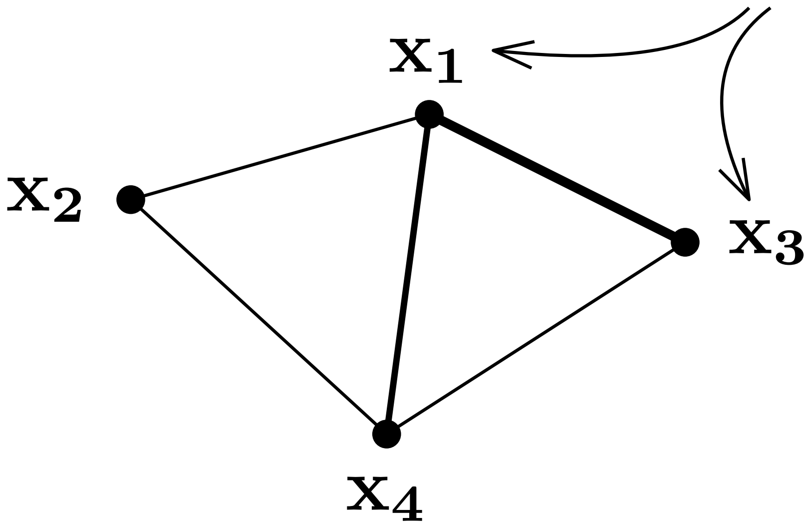
# Graph neural networks



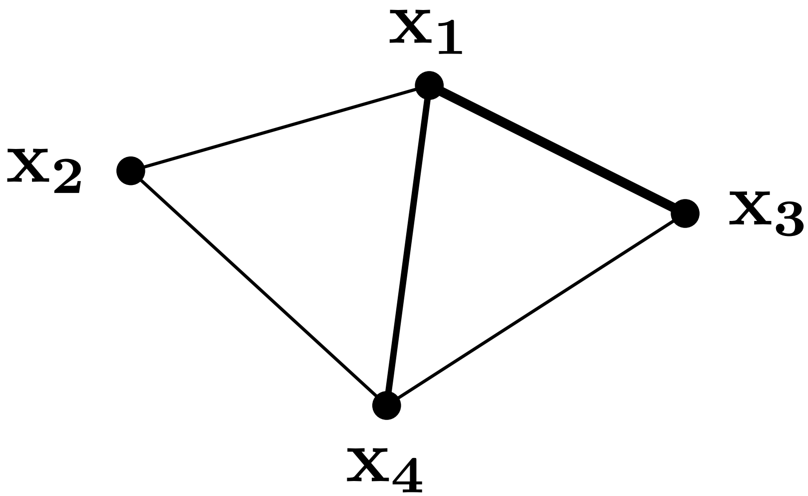

How can we create a general function $f_\theta(\mathcal{G})$?

# Graph neural networks

encode structure in node embeddings



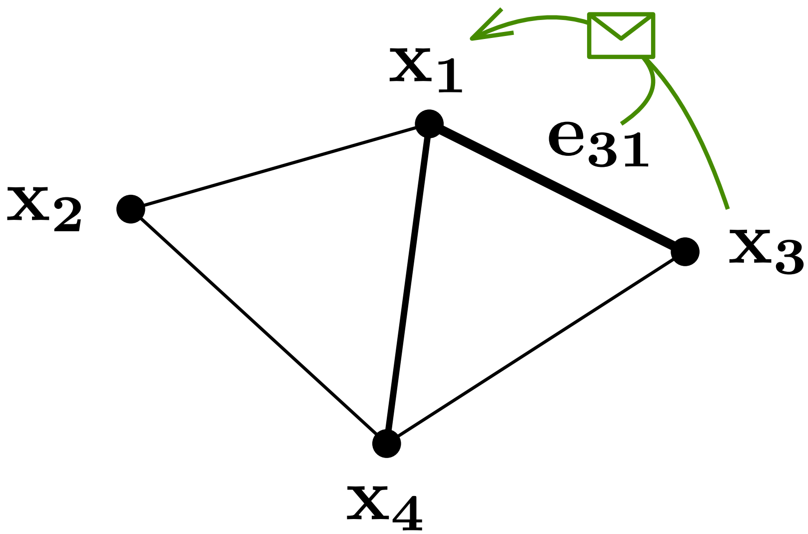
# Graph neural networks




1. Create messages from neighbor edges

$$_{j \in \mathcal{N}(i)} \phi\left(\mathbf{x}_j, \mathbf{e}_{ji}\right)$$

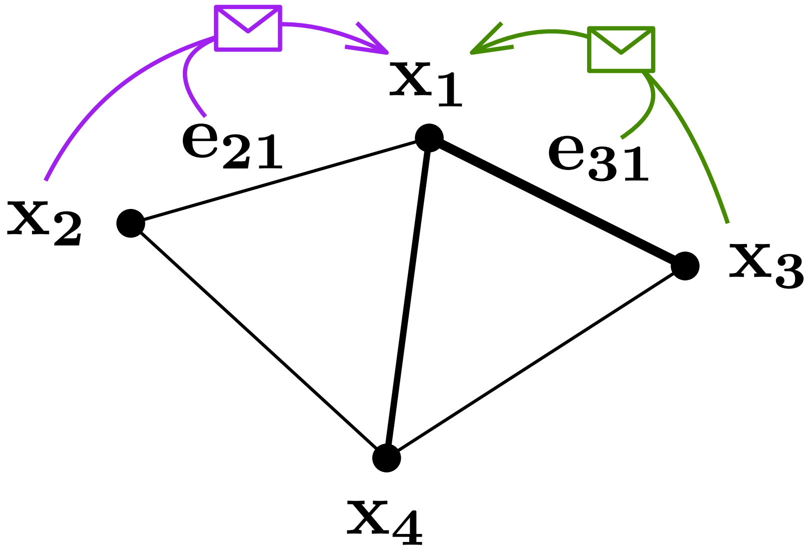
# Graph neural networks




1. Create messages from neighbor edges

$$_{j \in \mathcal{N}(i)} \phi(\mathbf{x}_j, \mathbf{e}_{ji})$$

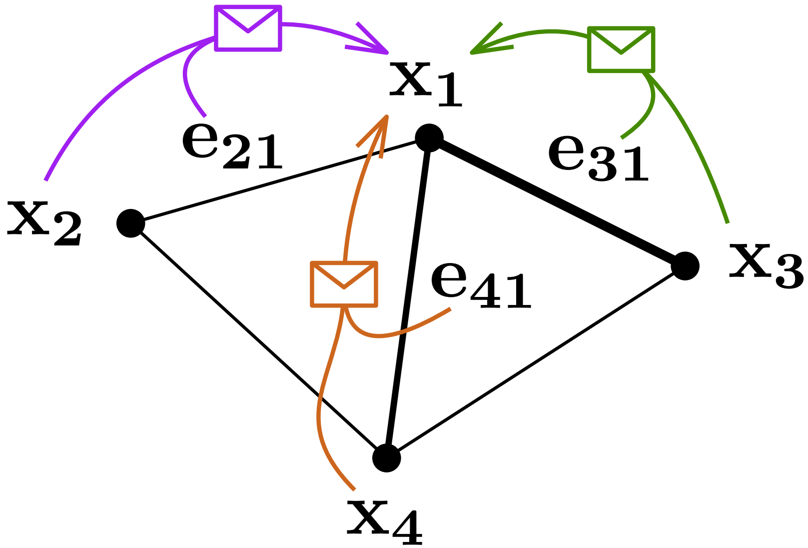
# Graph neural networks




1. Create messages from neighbor edges

$$_{j\in\mathcal{N}(i)}\ \phi(\mathbf{x}_j, \mathbf{e}_{ji})$$

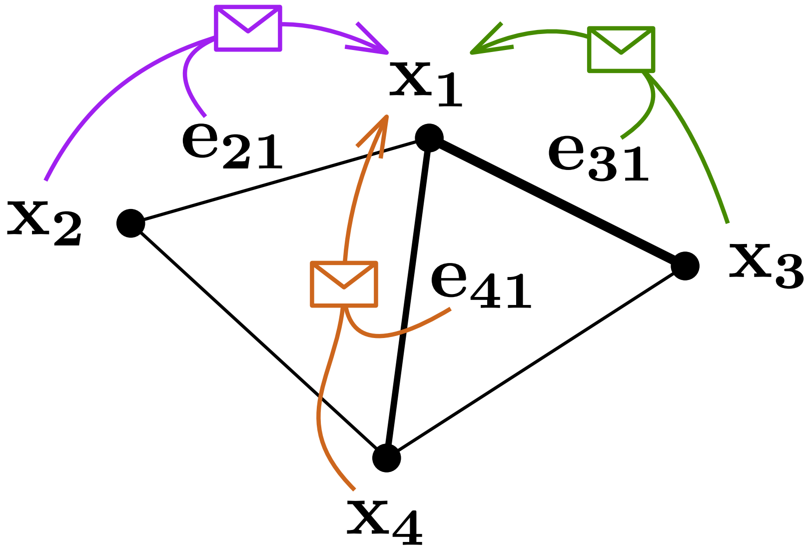
# Graph neural networks




1. Create messages from neighbor edges

$$ {}_{j\in\mathcal{N}(i)}\ \phi(\mathbf{x}_j\,,\mathbf{e}_{ji}) $$

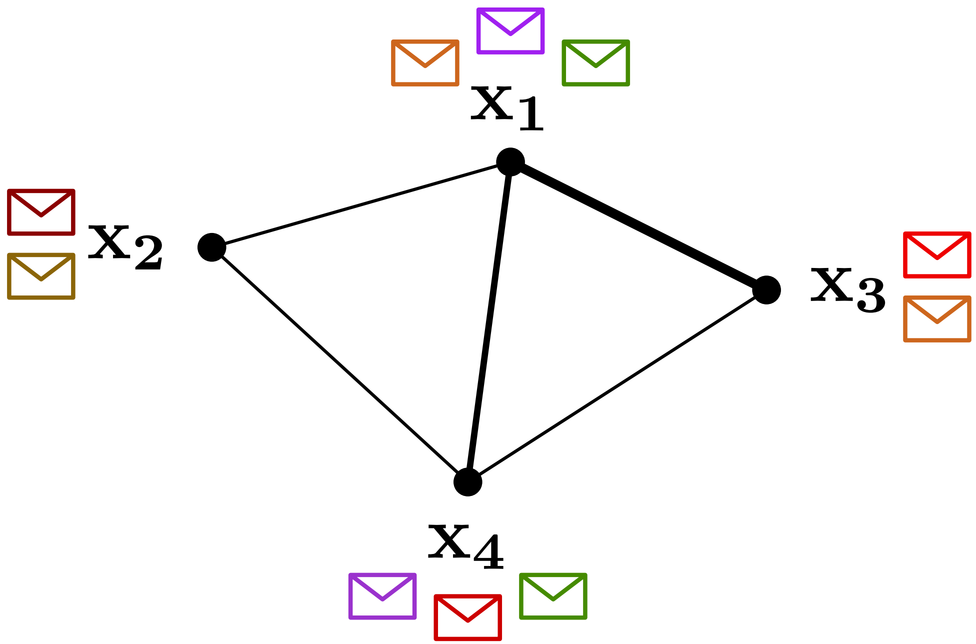
# Graph neural networks




1. Create messages from neighbor edges
2. Aggregate messages

$$\sum_{j \in \mathcal{N}(i)} \phi(\mathbf{x}_j, \mathbf{e}_{ji})$$

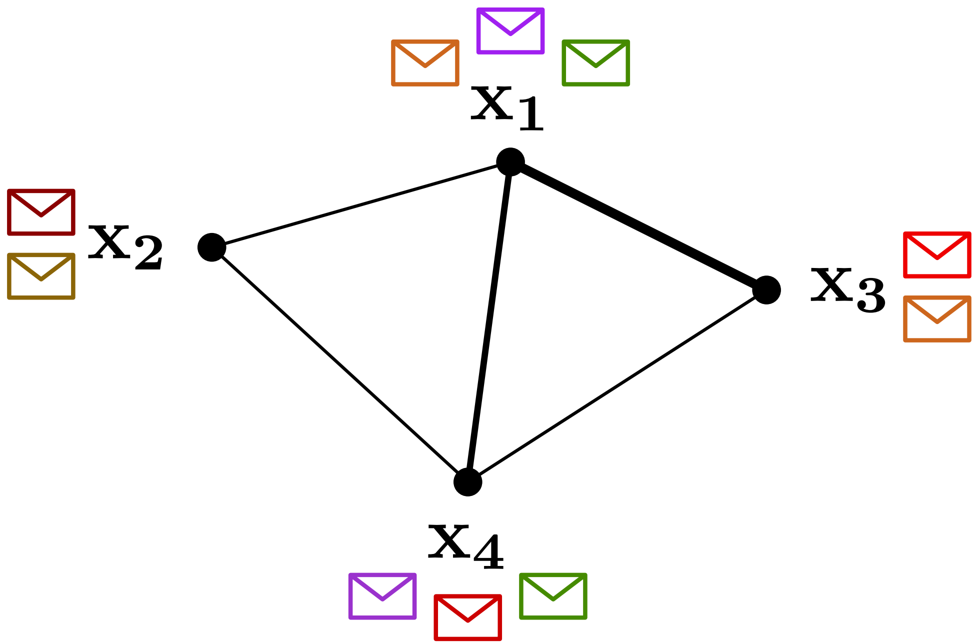
# Graph neural networks




1. Create messages from neighbor edges
2. Aggregate messages

$$\sum_{j \in \mathcal{N}(i)} \phi(\mathbf{x}_j, \mathbf{e}_{ji})$$

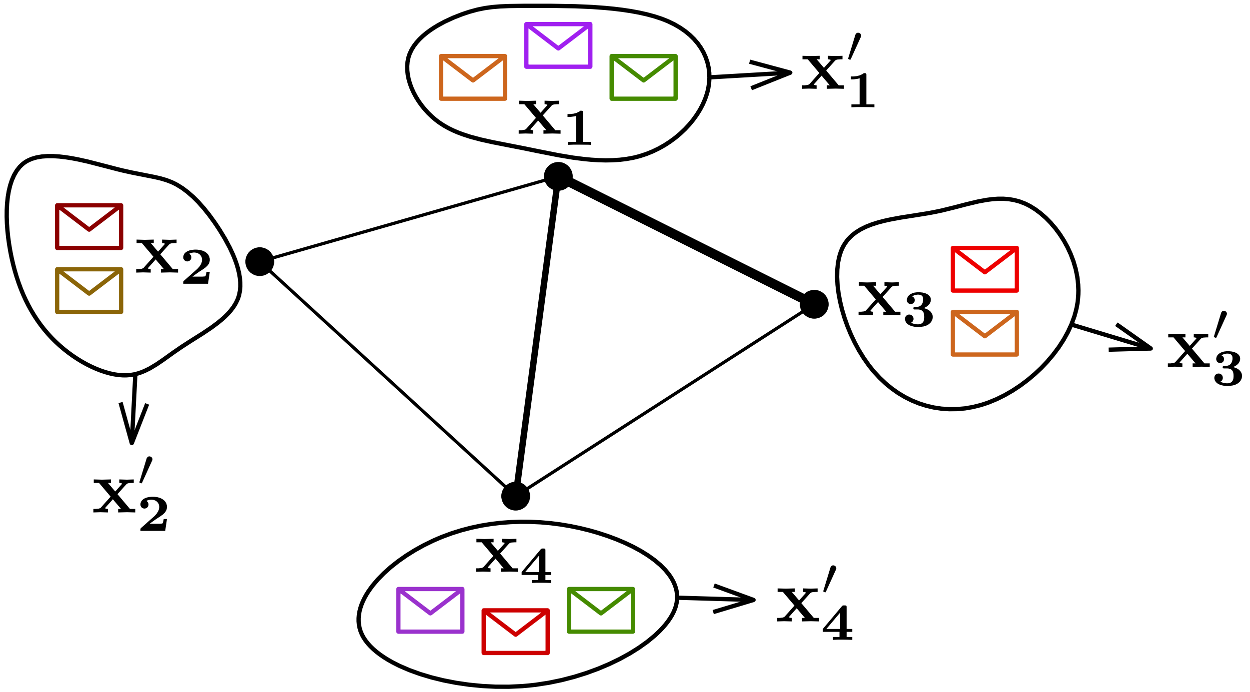
# Graph neural networks




1. Create messages from neighbor edges
2. Aggregate messages
3. Apply transformation

$$\mathbf{x}_i' = \psi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}(i)} \phi(\mathbf{x}_j, \mathbf{e}_{ji})\right)$$

# Graph neural networks




1. Create messages from neighbor edges
2. Aggregate messages
3. Apply transformation

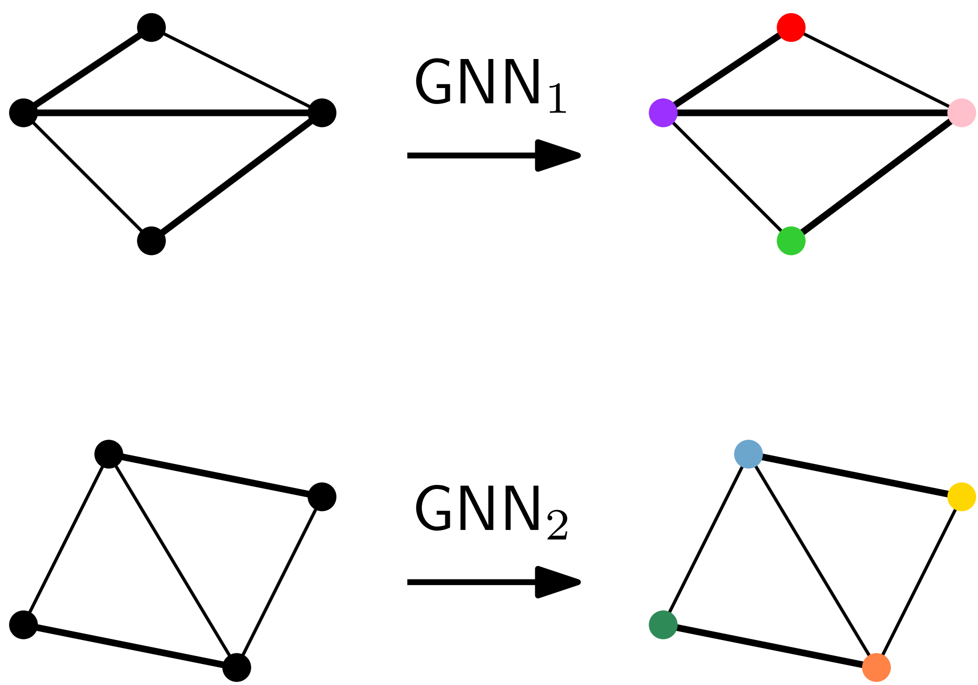
$$\mathbf{x}'_i = \psi\left(\mathbf{x}_i, \sum_{j \in \mathcal{N}(i)} \phi(\mathbf{x}_j, \mathbf{e}_{ji})\right)$$

# Q-network architecture

Predicts the best achievable value after taking an action

# Q-network architecture

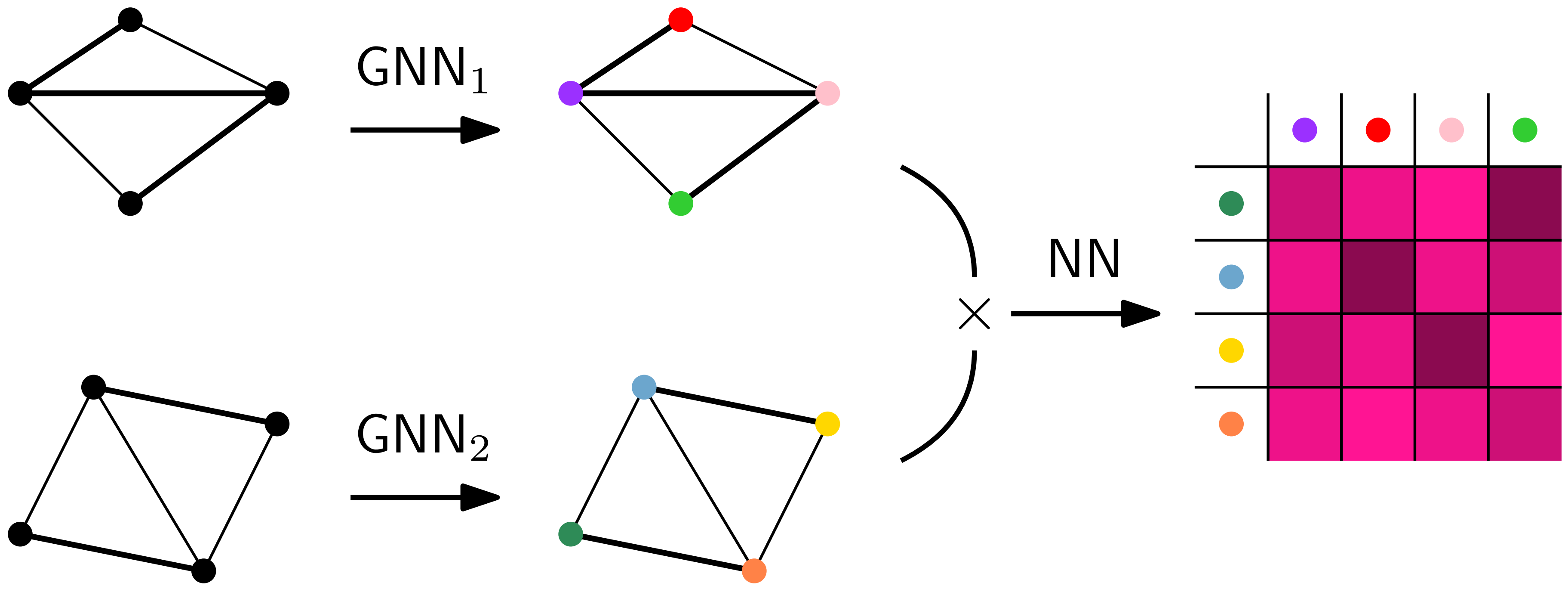
Predicts the best achievable value after taking an action




Encode graph structure

# Q-network architecture

Predicts the best achievable value after taking an action

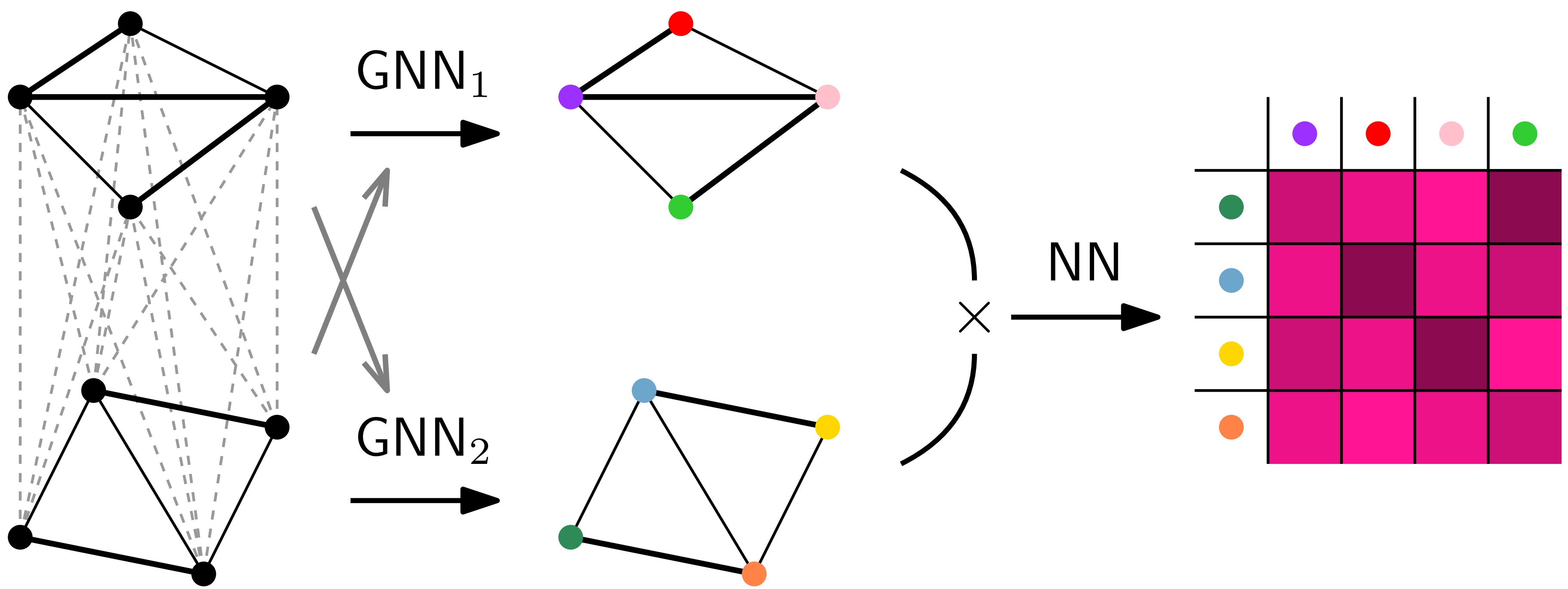



Encode graph structure

Predict pair values

# Q-network architecture

Predicts the best achievable value after taking an action




Encode graph structure

Predict pair values

# Experiment setup

# Experiment setup

**Problem generator**

Undirected 8 node graphs with random weights

# Experiment setup

**Problem generator**

Undirected 8 node graphs with random weights

$8! = 40320$ possible assignments

# Experiment setup

**Problem generator**

Undirected 8 node graphs with random weights

$8! = 40320$ possible assignments

**Training sets**

1. Single QAP during training
2. Random QAP in every episode

# Experiment setup

**Problem generator**
Undirected 8 node graphs with random weights

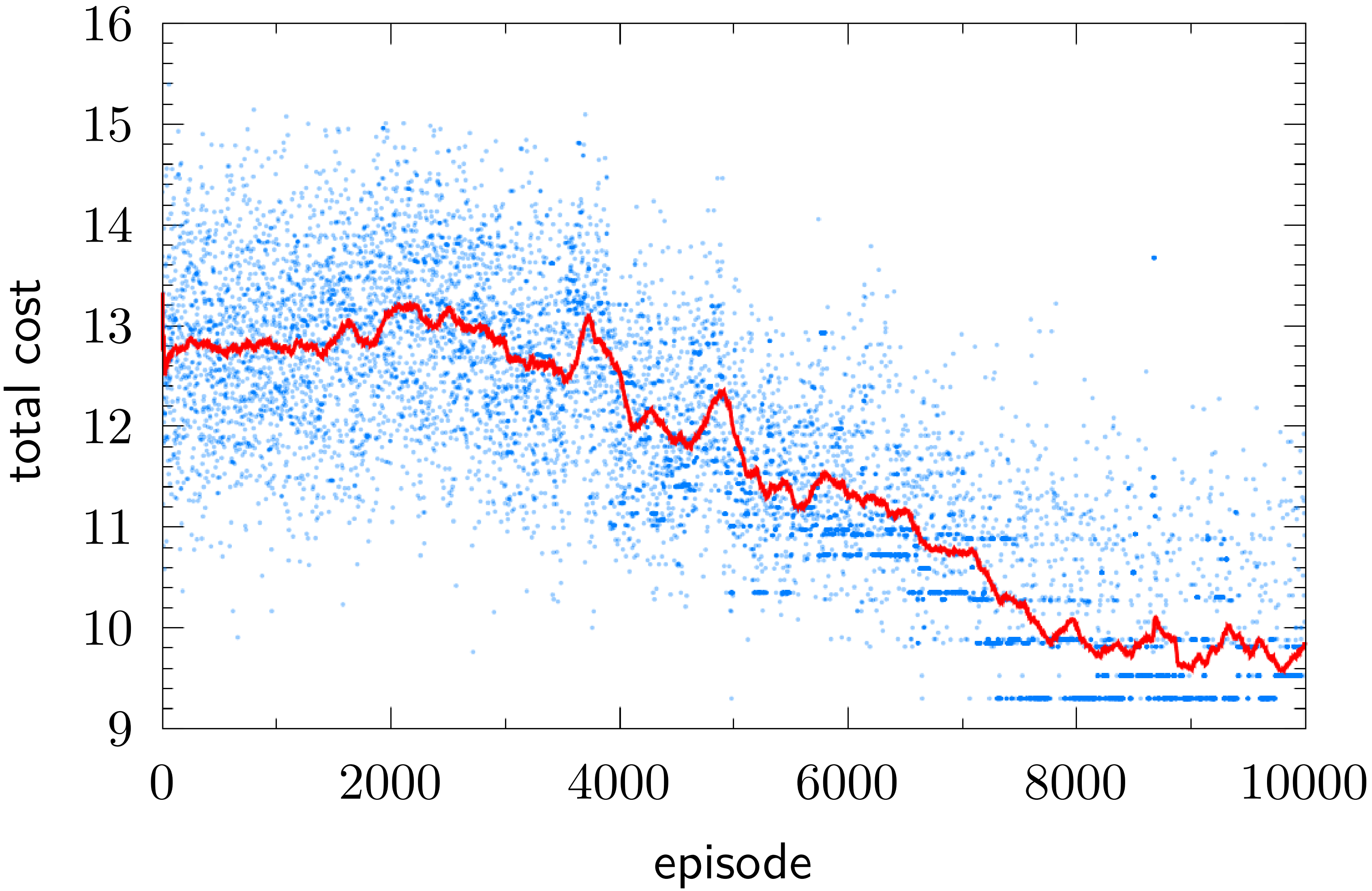
$$8! = 40320 \text{ possible assignments}$$

**Training sets**
1. Single QAP during training
2. Random QAP in every episode

**Agent**
DQN with GNN based Q-network

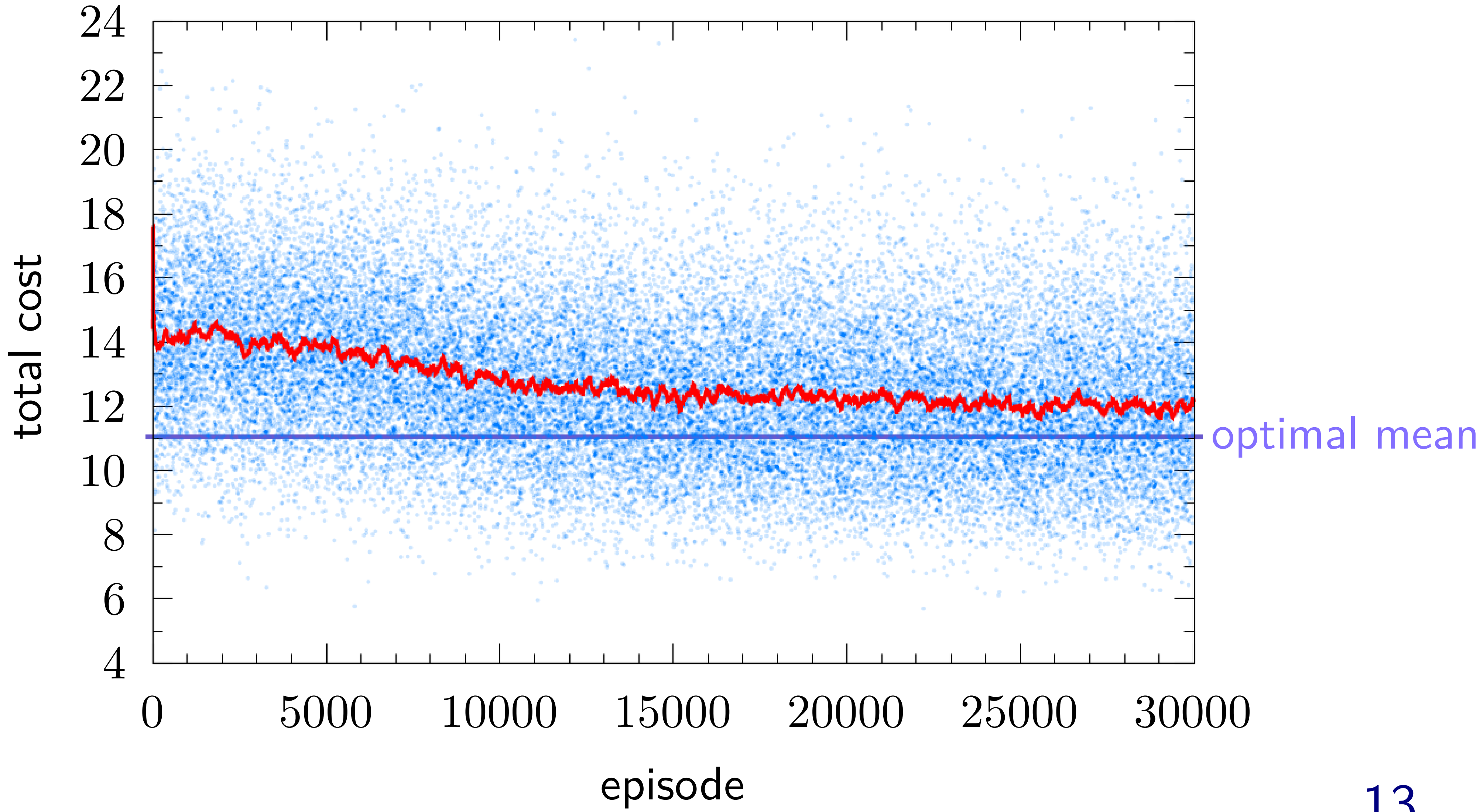
# Experiment on single small problem

The agent is able to find the optimal solution



# Experiment on distribution of small problems

The agent can approach previously unseen instances

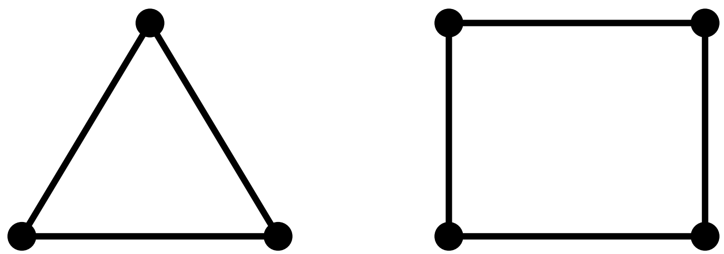


# Limitations of graph neural networks

- Message passing GNNs cannot distinguish some graphs

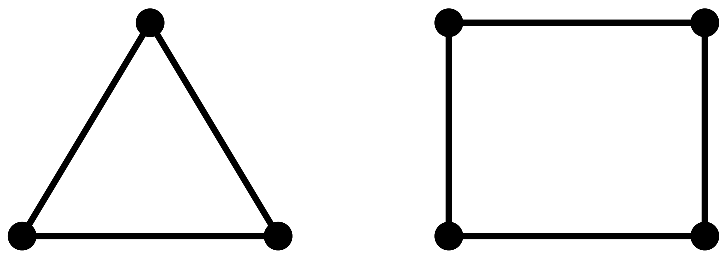
# Limitations of graph neural networks

- Message passing GNNs cannot distinguish some graphs



# Limitations of graph neural networks

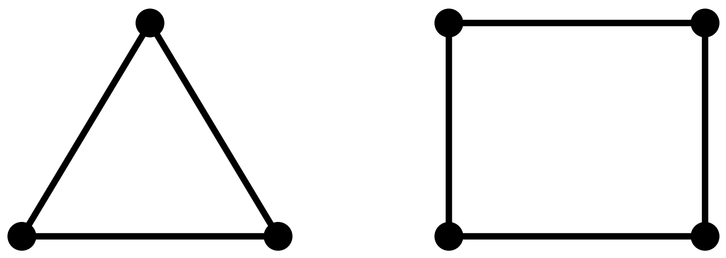
- Message passing GNNs cannot distinguish some graphs



$\rightarrow$ A GNN will not be able to solve some QAPs

# Limitations of graph neural networks

- Message passing GNNs cannot distinguish some graphs



$\rightarrow$ A GNN will not be able to solve some QAPs

- Embeddings become too similar with more layers (Oversmoothing)

# Possible remedies for limitations of GNNs

- More expressive GNNs
  (higher-order GNNs, special node features, ...)

# Possible remedies for limitations of GNNs

- More expressive GNNs
  (higher-order GNNs, special node features, . . . )
- Normalization layers to force distance between nodes
  (PairNorm)

# Possible remedies for limitations of GNNs

- More expressive GNNs
  (higher-order GNNs, special node features, . . . )
- Normalization layers to force distance between nodes
  (PairNorm)
- Local search
  (tree search, revocable actions, . . . )

# Open questions

- What is the impact of state representation on performance?
- Which patterns does the GNN need to be able to recognize?
- What heuristics can the agent learn?
- How important is exploration in this task?

# References


[1] E. L. Lawler, “The quadratic assignment problem,” *Management Science*, vol. 9, pp. 586–599, 1963.

[2] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?,” *arXiv preprint arXiv:1810.00826*, 2018.

[3] E. Loiola, N. Abreu, P. Boaventura-Netto, P. Hahn, and T. Querido, “A survey of the quadratic assignment problem,” *European Journal of Operational Research*, vol. 176, pp. 657–690, 01 2007.

[4] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna, “Revised note on learning algorithms for quadratic assignment with graph neural networks,” 2018.

[5] C. Liu, R. Wang, Z. Jiang, J. Yan, L. Huang, and P. Lu, “Revocable deep reinforcement learning with affinity regularization for outlier-robust graph matching,” 2021.

[6] T. C. Koopmans and M. Beckmann, “Assignment problems and the location of economic activities,” *Econometrica*, vol. 25, no. 1, pp. 53–76, 1957.

[7] R. Sato, M. Yamada, and H. Kashima, “Random features strengthen graph neural networks,” in *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341, SIAM, 2021.

[8] R. Sato, “A survey on the expressive power of graph neural networks,” *ArXiv*, vol. abs/2003.04078, 2020.