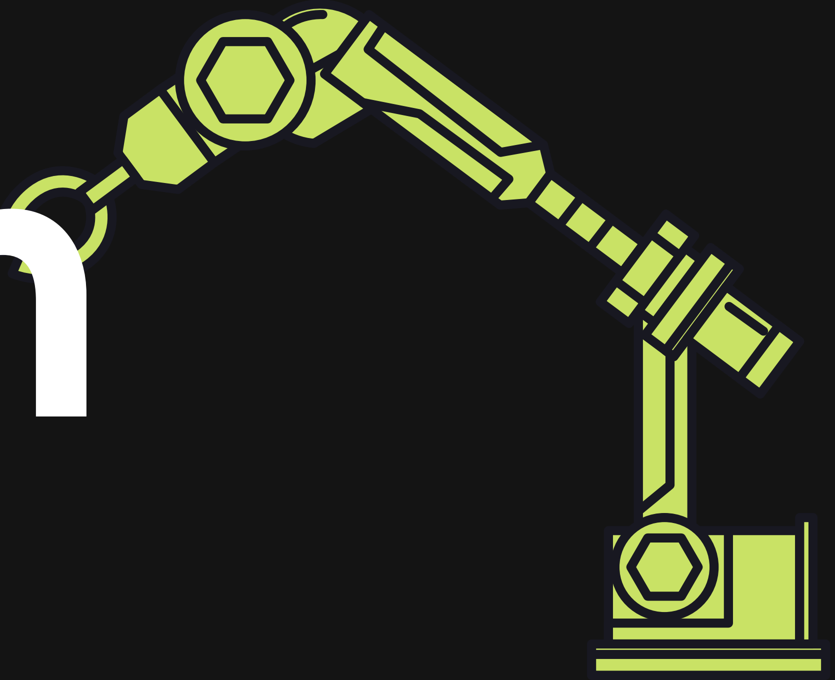
Reinforcement learning project

TSP problem



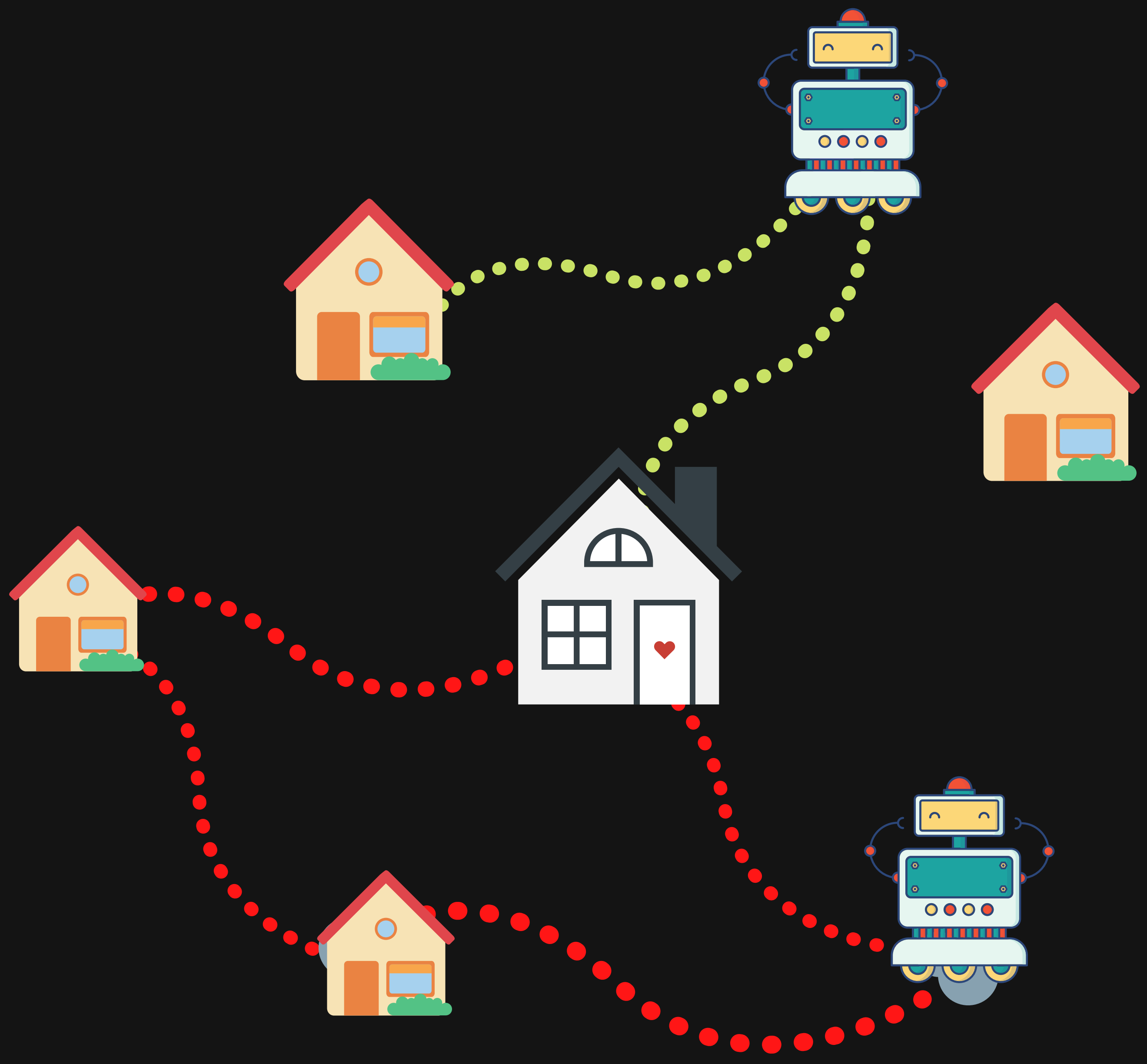
by Nina Konovalova
and Albina Klepach

*TSP -- Travelling Salesman Problem

Plan

- Introduction
- Description of the problem
- Description of the methods
- Results
- Conclusion

Introduction



Combinatorial optimization

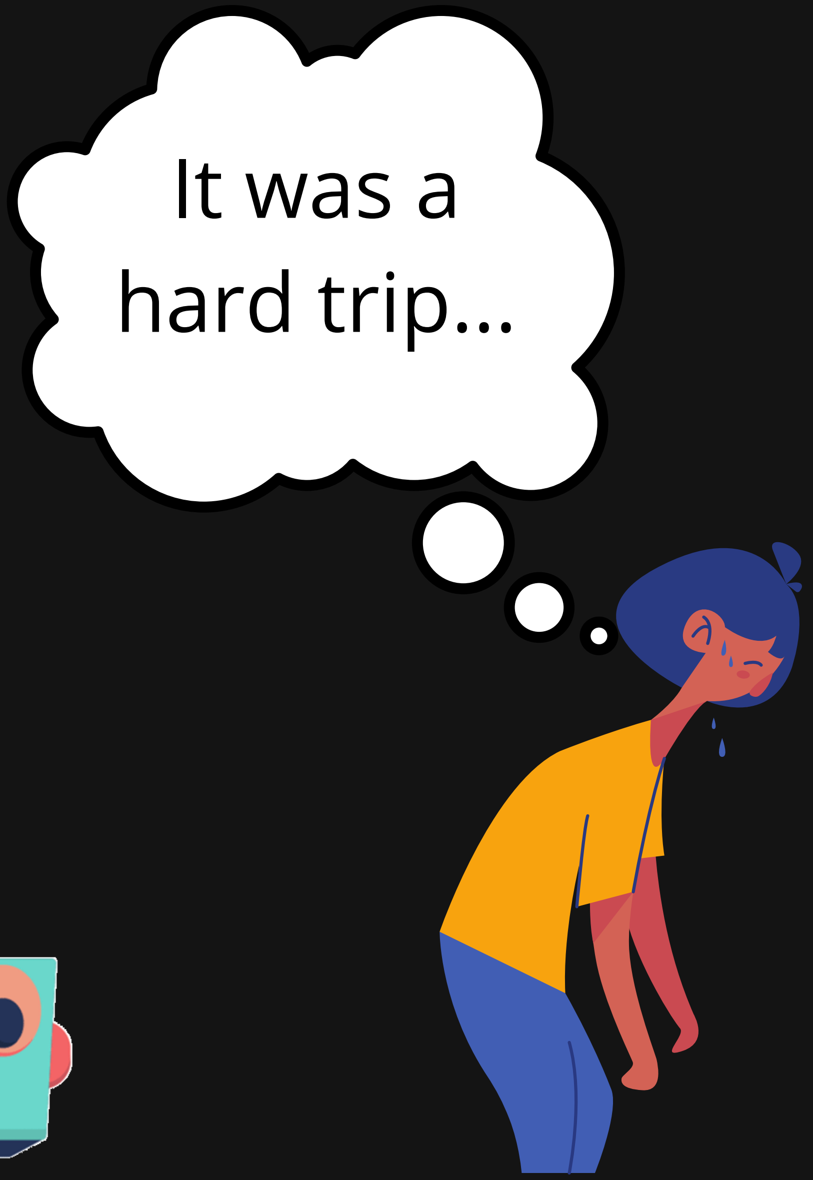
VPR

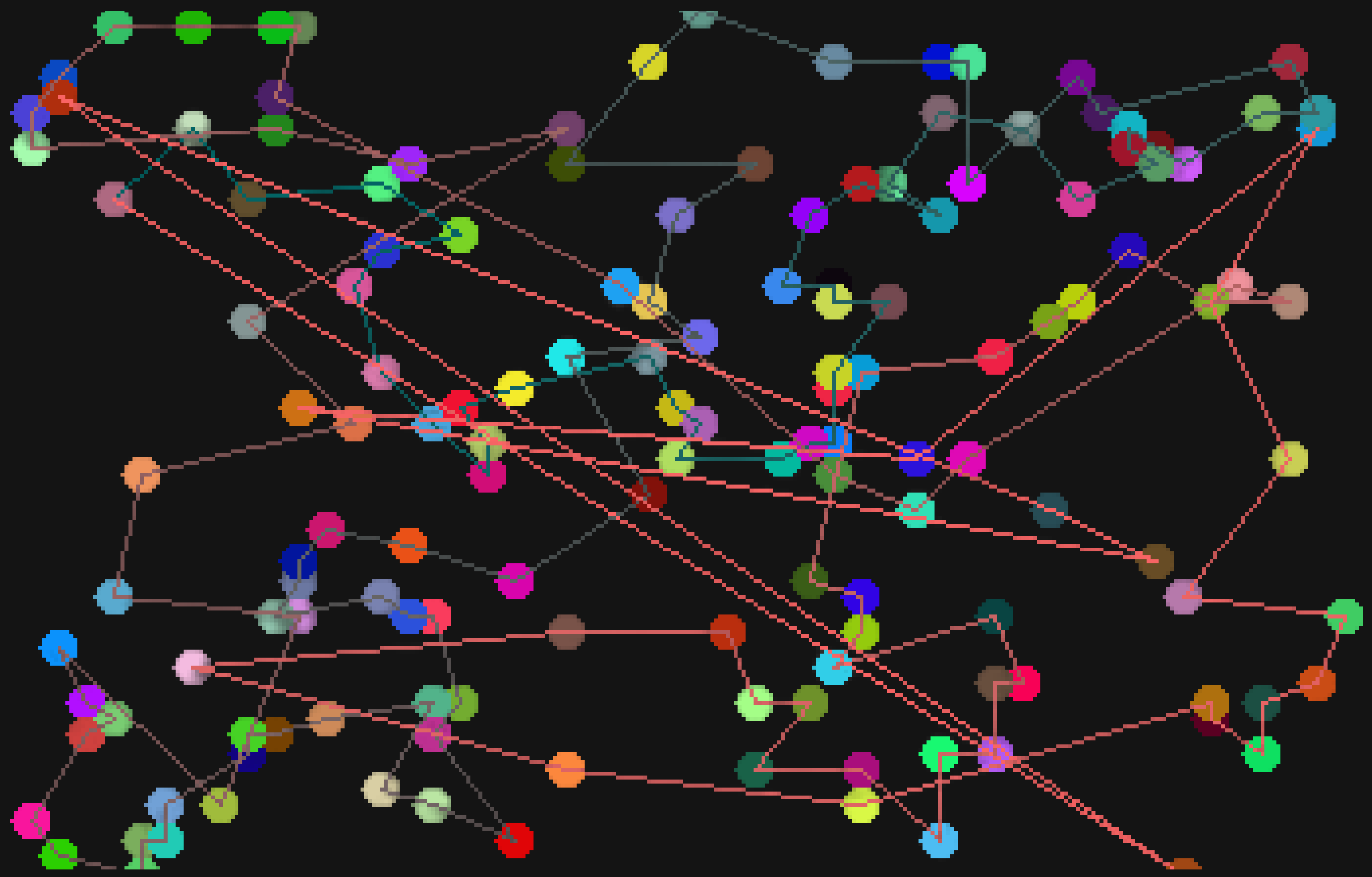
TSP

OP

Description of the problem

Travelling Salesman Problem (TSP)



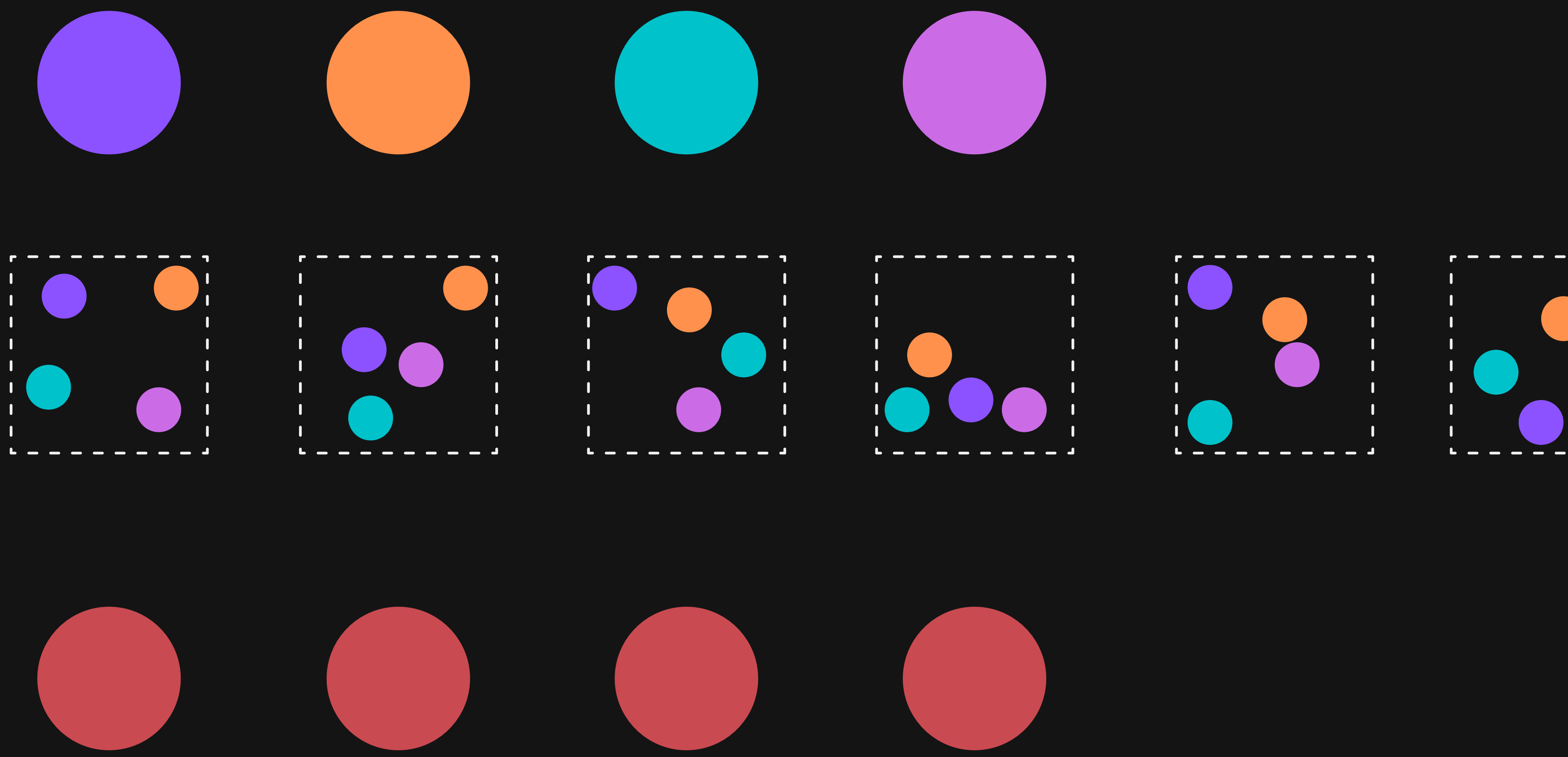


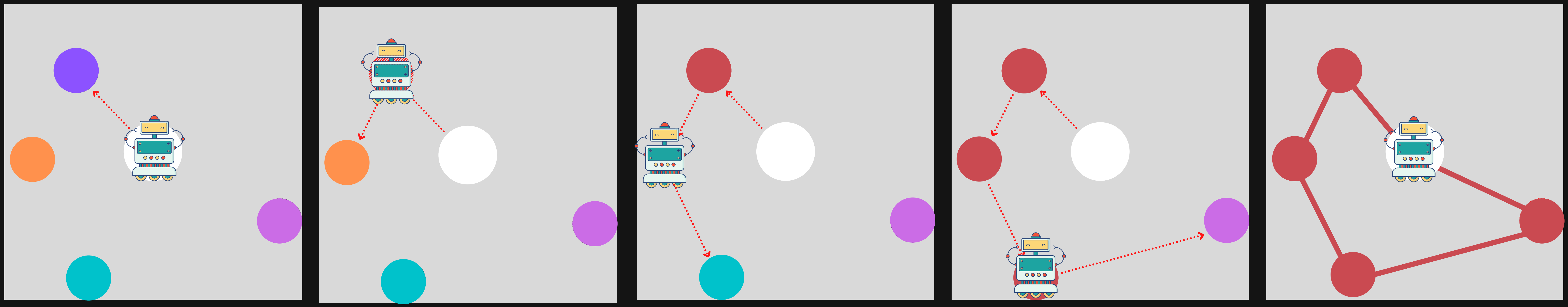
Description of the methods

Markov Decision Process (MDP)

naive approach requires many samples

terminating condition when all destinations are visited





Description of the methods

Sequence of n cities in 2D space $s = \{\mathbf{x}_i\}_{i=1}^{n}$, where each $\mathbf{x}_i \in \mathrm{R}^2$

No depot:

$$L(\pi|s) = ||\mathbf{x}_{pi(n)} - \mathbf{x}_{pi(1)}||_2 + \sum_{i=1}^{n-1} ||\mathbf{x}_{pi(i)} - \mathbf{x}_{pi(i+1)}||_2$$

Depot:

$$L(\pi|s) = ||\mathbf{x}_{pi(n)} - \mathbf{x}_{depot}||_2 + ||\mathbf{x}_{pi(1)} - \mathbf{x}_{depot}||_2 + \sum_{i=1}^{n-1} ||\mathbf{x}_{pi(i)} - \mathbf{x}_{pi(i+1)}||_2$$

Chain rule (to factorize the probability of a tour):

$$p(\pi|s) = \prod_{n=1}^{n} p(\pi(i)|\pi(< i), s)$$

Description of the methods

Out training objective is expected tour length:

$$J(\Theta|s) = \mathbb{E}_{\pi \sim p_\Theta(\cdot|s)} L(\pi|s)$$

The gradient of expected tour length (with REINFORCE algorithm):

$$\nabla_\Theta J(\Theta|s) = \mathbb{E}_{\pi \sim p_\Theta(\cdot|s)} \left[(L(\pi|s) - b(s)) \nabla_\Theta \log p_\Theta(\pi|s) \right]$$

$$\nabla_\Theta J(\Theta) \approx \frac{1}{B} \sum_{i=1}^{B} (L(\pi|s_i) - b(s_i)) \nabla_\Theta \log p_\Theta(\pi_i|s_i)$$

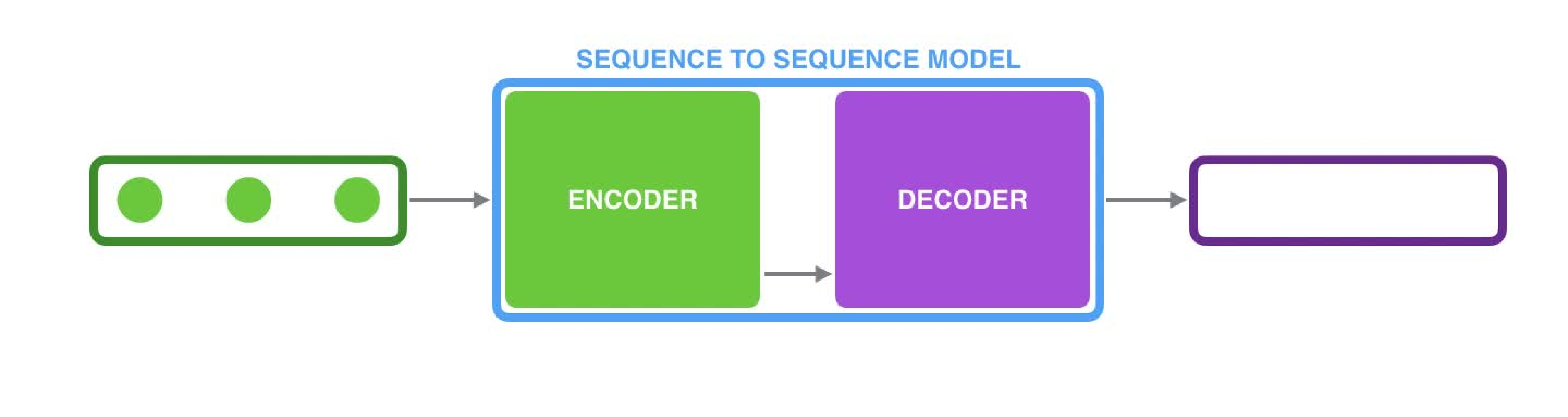
The objective for critic ($b_{\Theta_v}(s)$ is a prediction):

$$\mathbb{L}(\Theta_v) = \frac{1}{B} \sum_{i=1}^{B} ||b_{\Theta_v}(s_i) - L(\pi_i)||_2^2$$

Algorithm 1 Actor-critic training

procedure TRAIN(training set S, number of training steps T, batch size B)
    Initialize pointer network params θ
    Initialize critic network params θ_v
    for $t = 1$ to T **do**
        $s_i \sim$ SAMPLEINPUT(S) for $i \in \{1, \ldots, B\}$
        $\pi_i \sim$ SAMPLESOLUTION($p_\theta(.|s_i)$) for $i \in \{1, \ldots, B\}$
        $b_i \leftarrow b_{\theta_v}(s_i)$ for $i \in \{1, \ldots, B\}$
        $g_\theta \leftarrow \frac{1}{B} \sum_{i=1}^{B} (L(\pi_i|s_i) - b_i) \nabla_\theta \log p_\theta(\pi_i|s_i)$
        $\mathcal{L}_v \leftarrow \frac{1}{B} \sum_{i=1}^{B} ||b_i - L(\pi_i)||_2^2$
        $\theta \leftarrow$ ADAM(θ, g_θ)
        $\theta_v \leftarrow$ ADAM($\theta_v, \nabla_{\theta_v} \mathcal{L}_v$)
    end for
    return θ
end procedure

Seq2Seq



Pointer Network

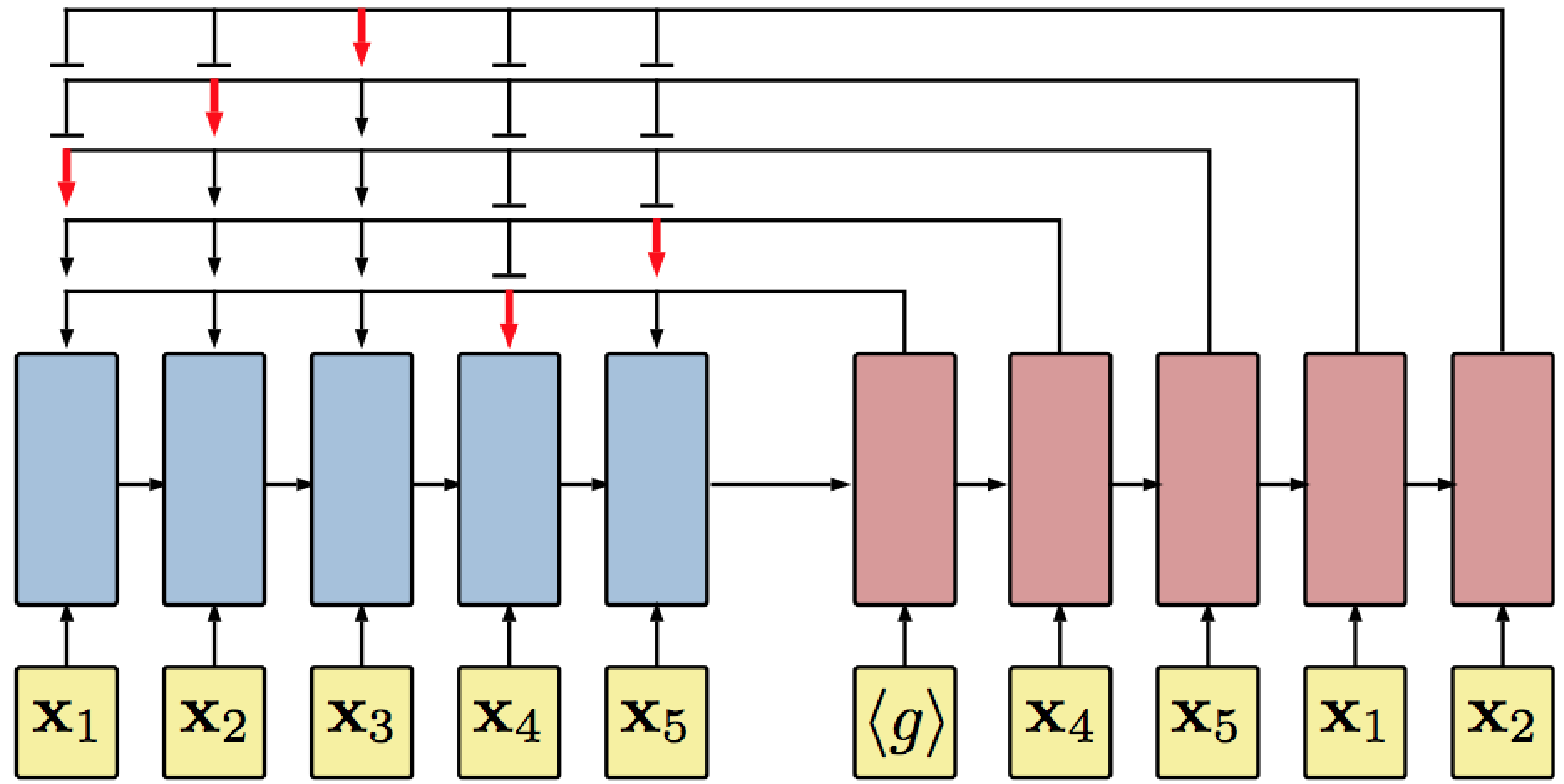


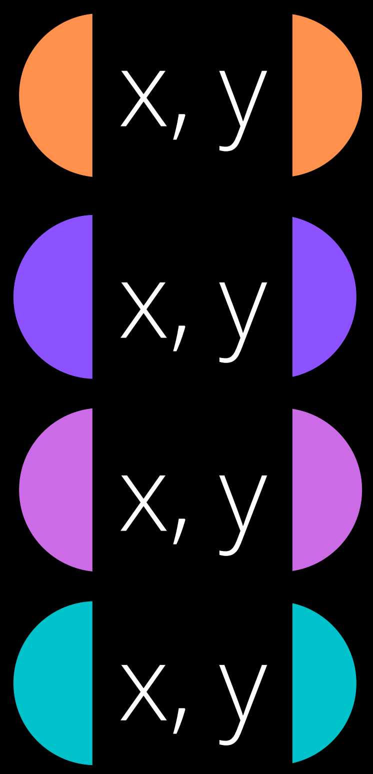
Figure 1: A pointer network architecture introduced by (Vinyals et al., 2015b).

Graph embeddings

Graph represenations:

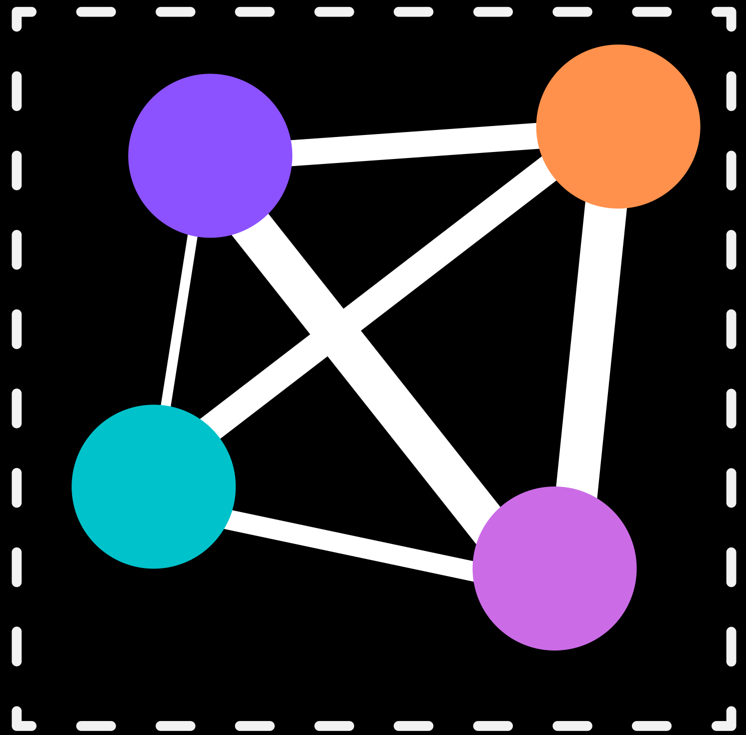
list of coordinates:

- Linear layer
- Simple node encoding $v_{node} \cdot v_{rand}$

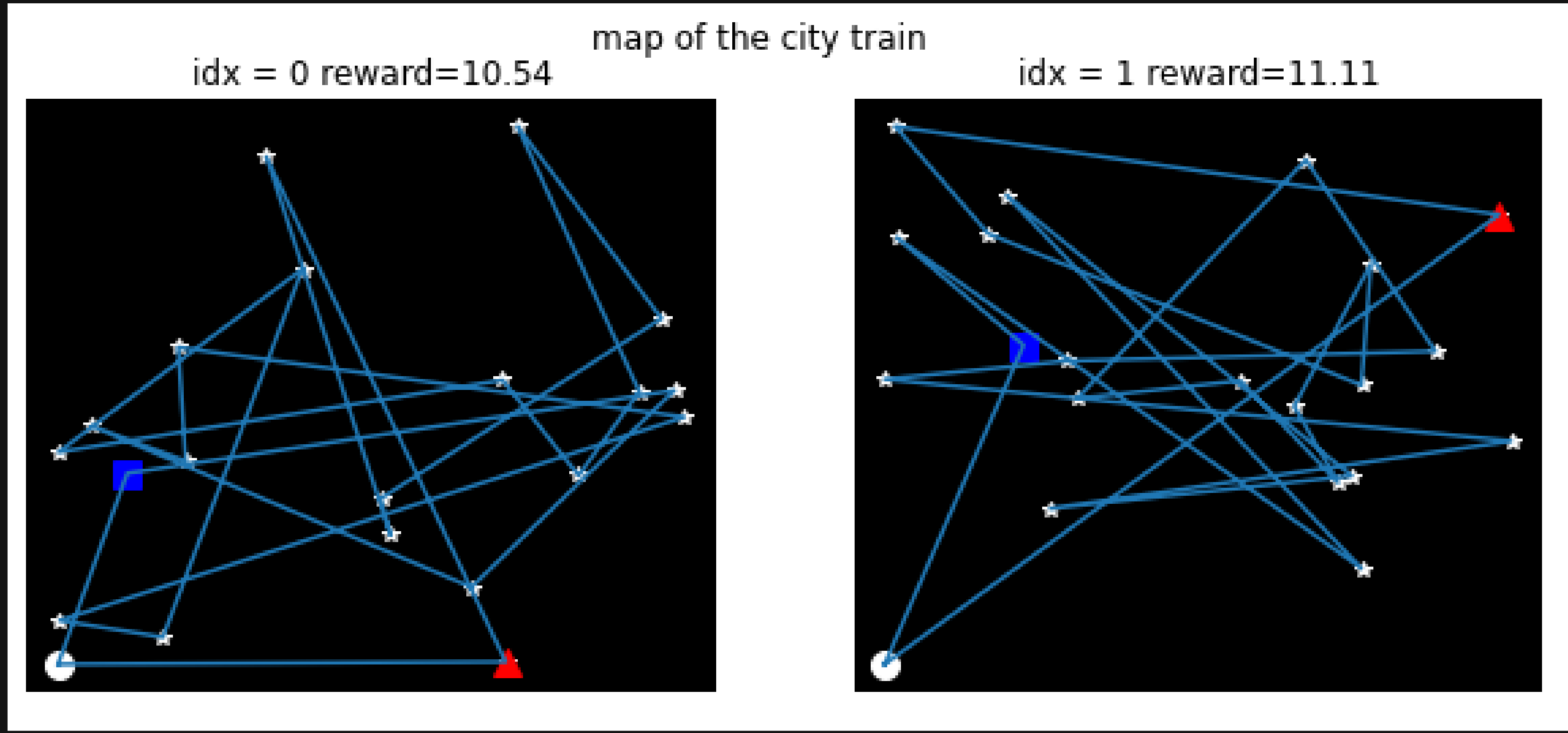


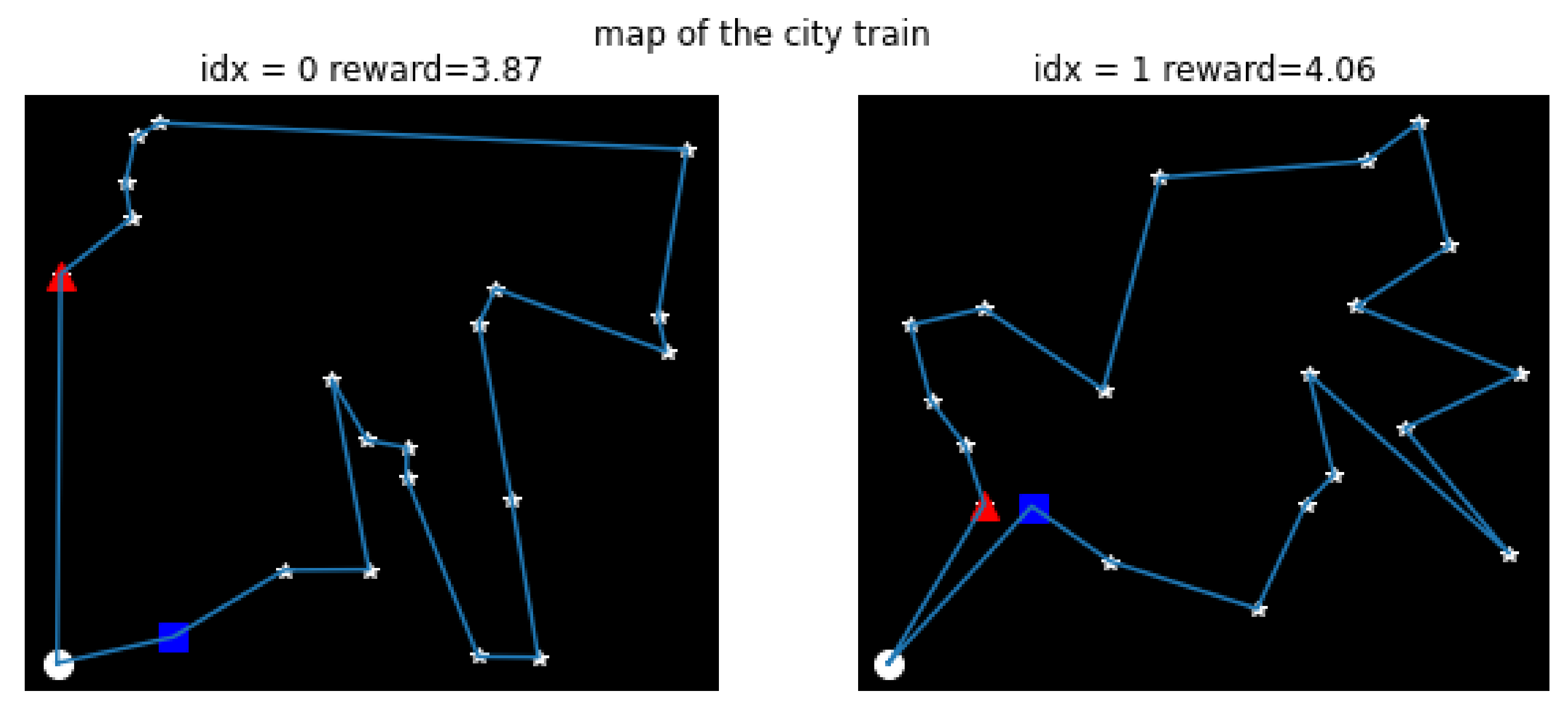
weighted graph (distance matrix):

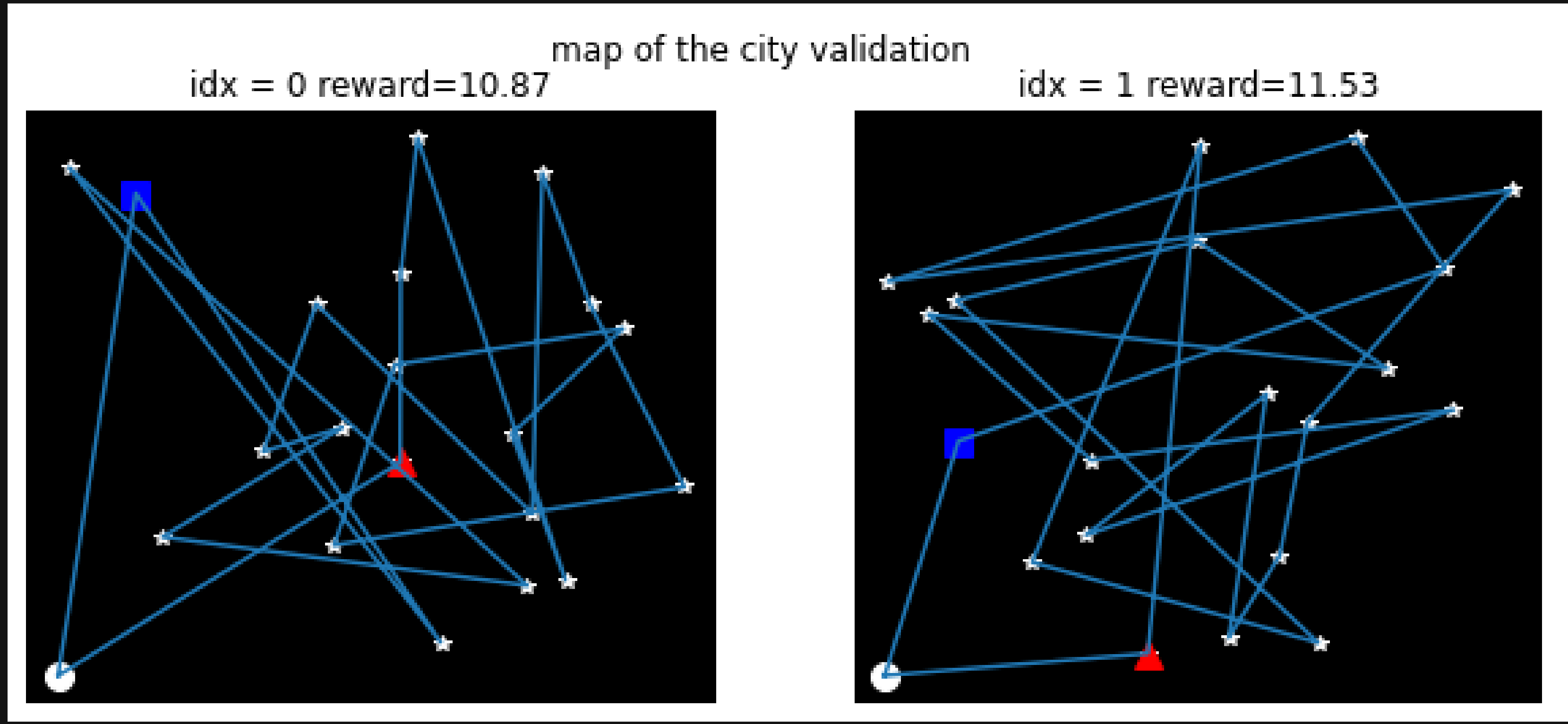
- Node2Vec
- DeepWalk

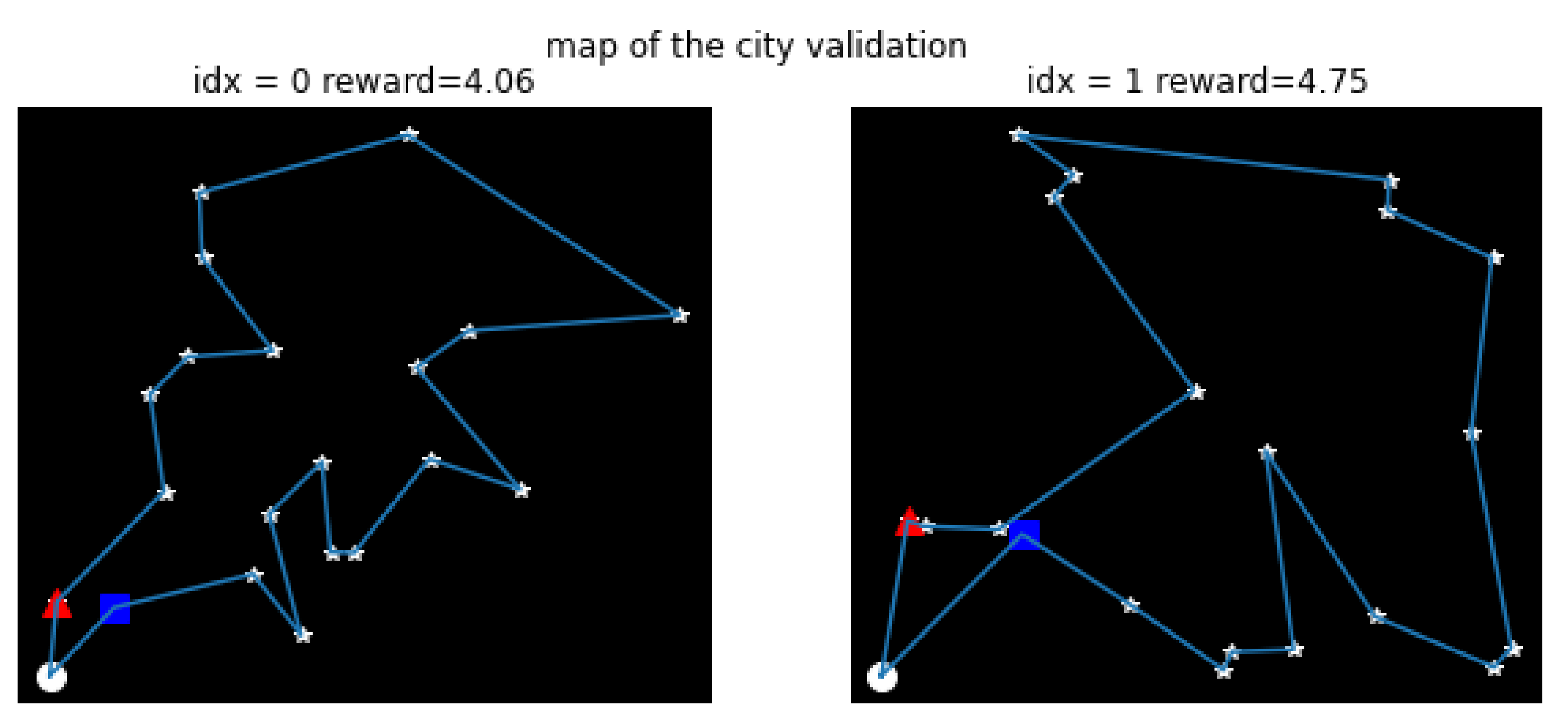


Results

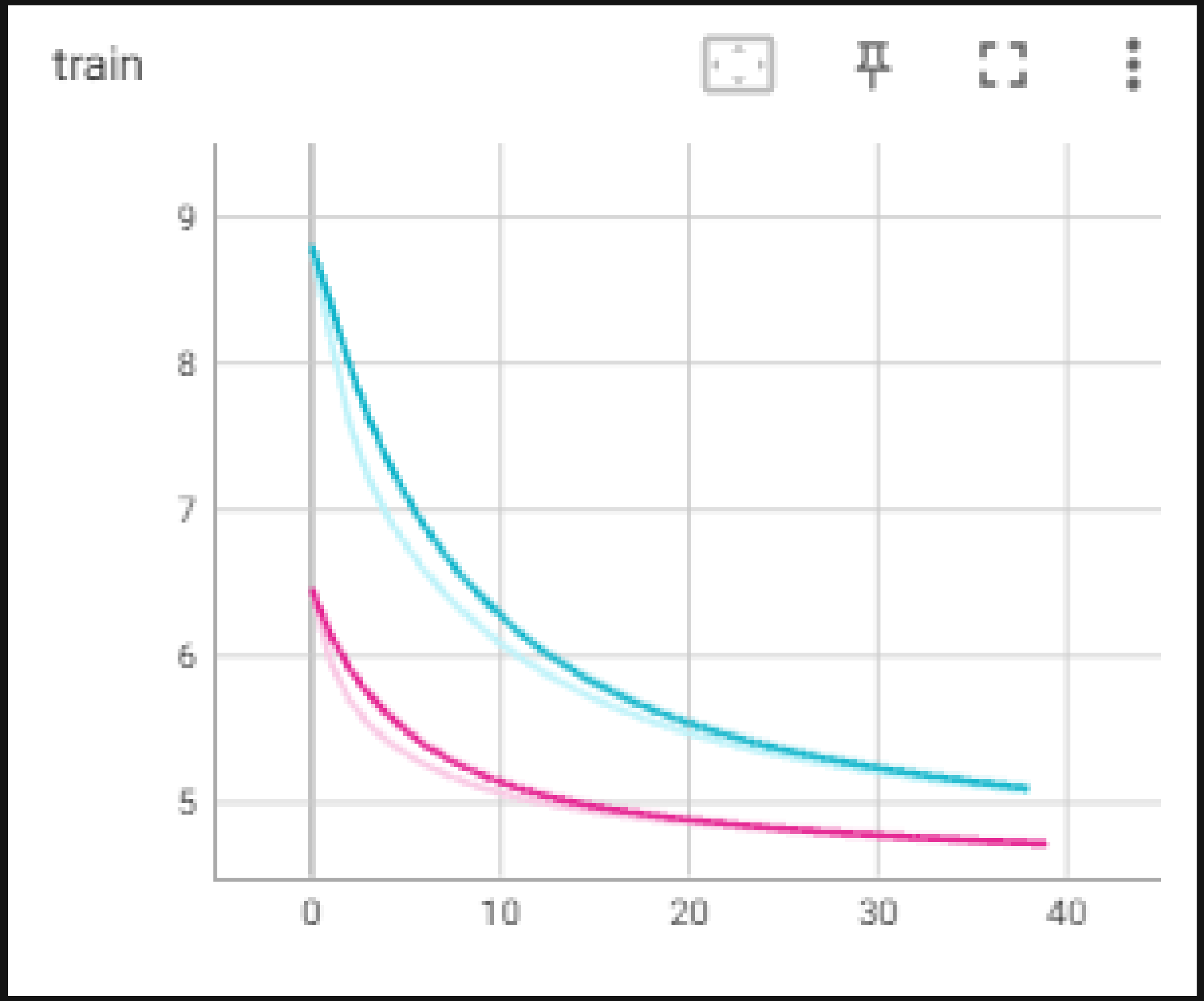

map of the city train
idx = 0 reward=10.54
idx = 1 reward=11.11

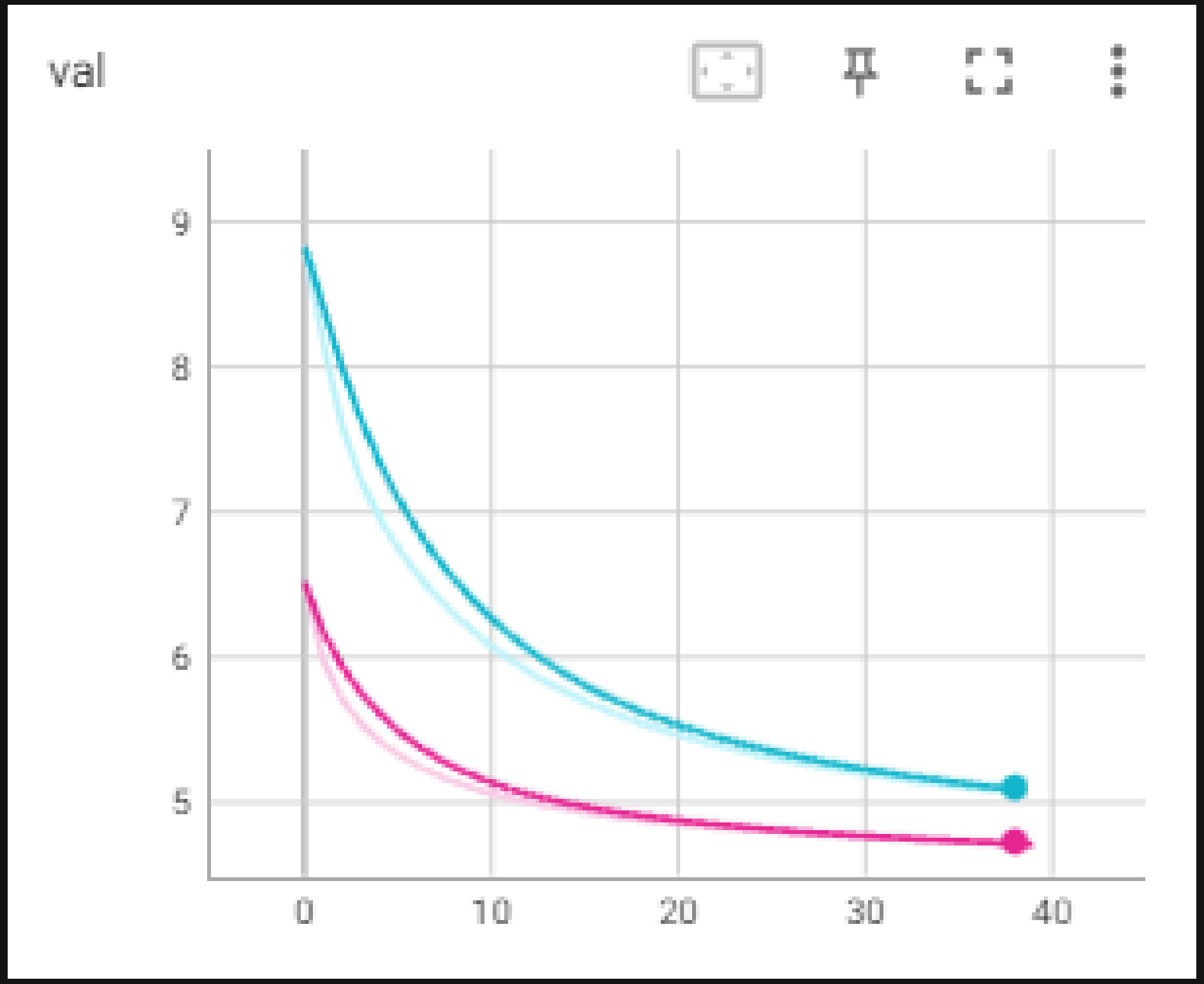

map of the city train
idx = 0 reward=3.87
idx = 1 reward=4.06


map of the city validation
idx = 0 reward=10.87
idx = 1 reward=11.53


map of the city validation
idx = 0 reward=4.06
idx = 1 reward=4.75

Results





Attention:

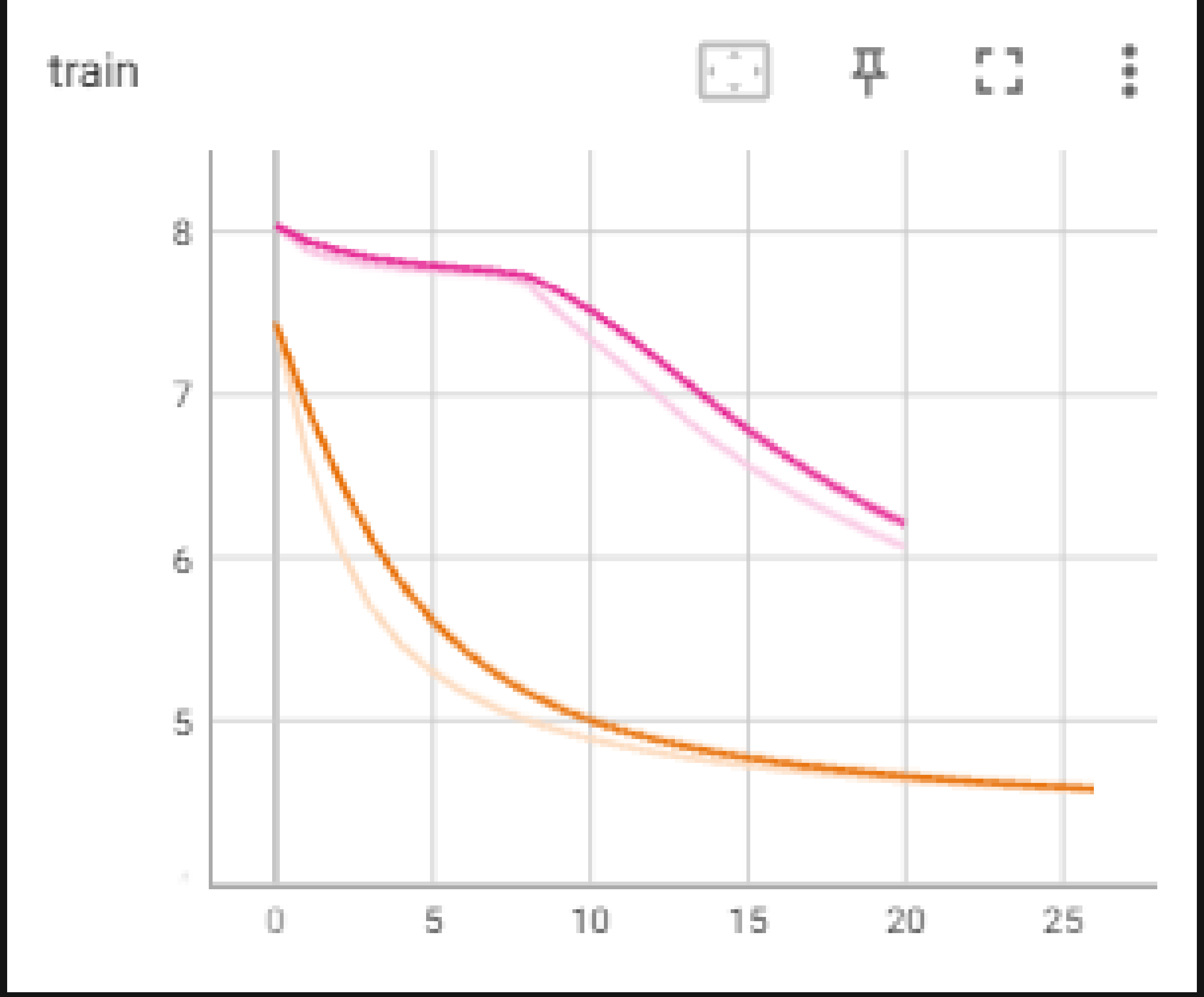
<u>Dot</u>

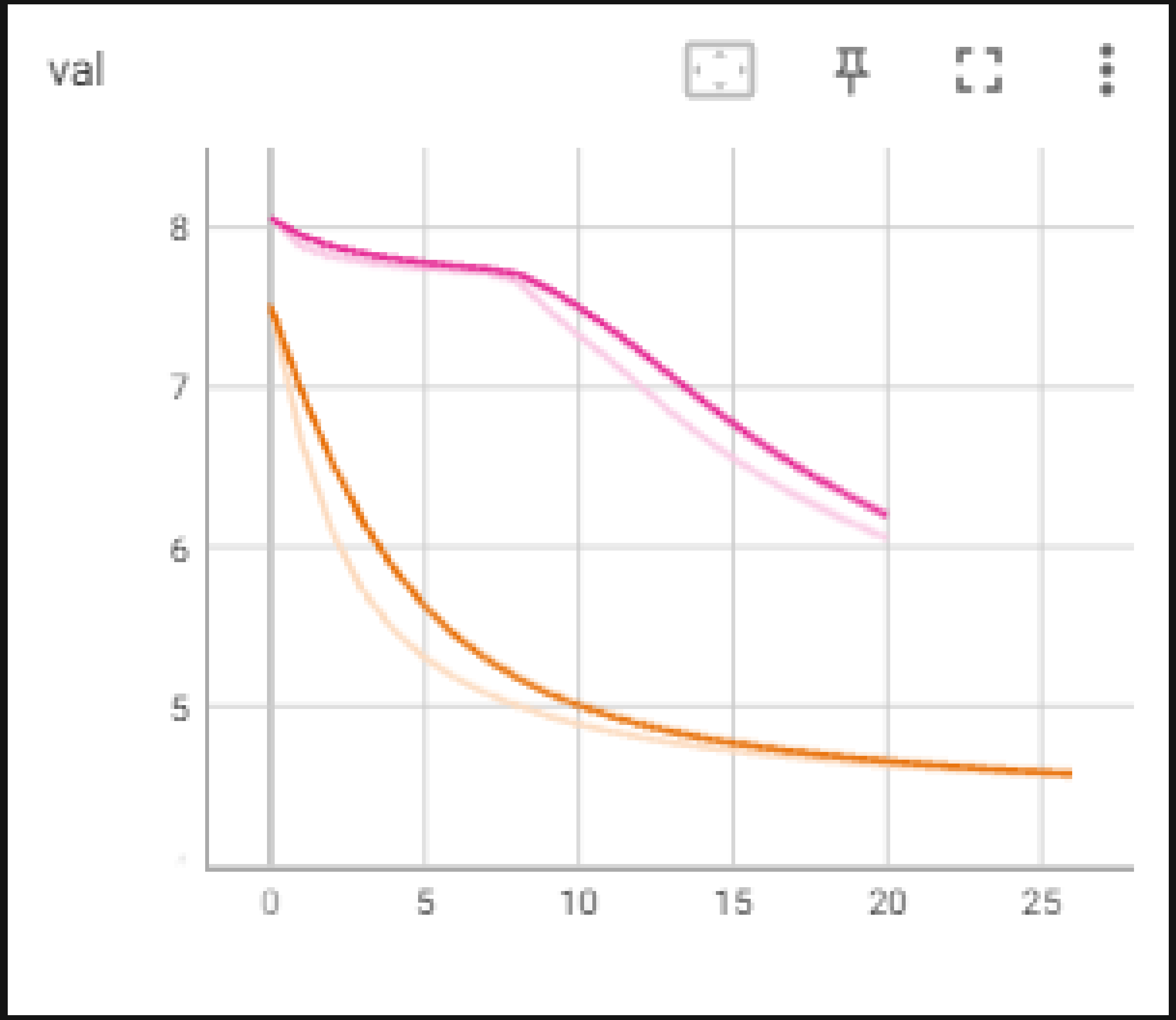
Embeddings:

- **Linear GE**
- **Simple GE**

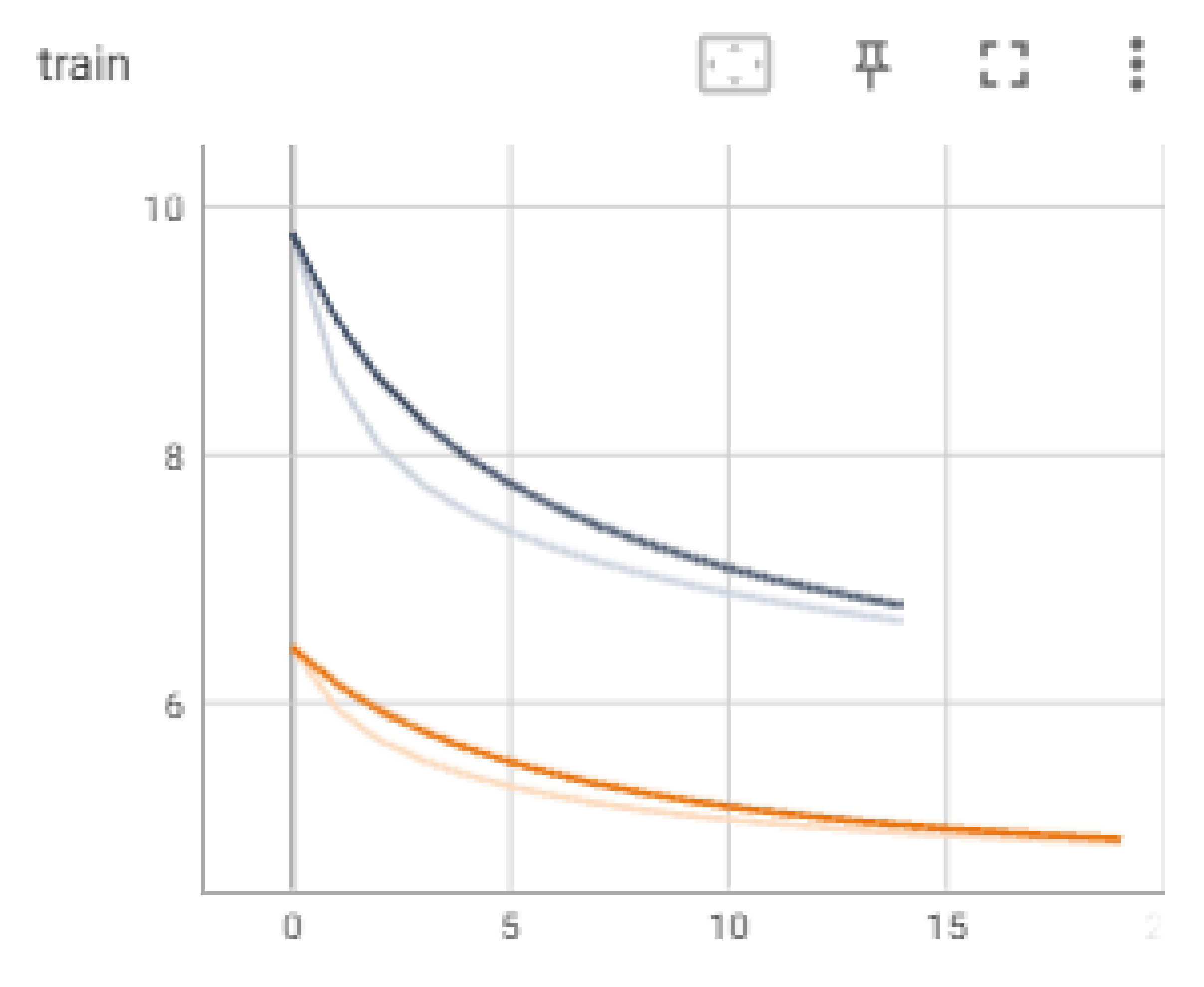
Results

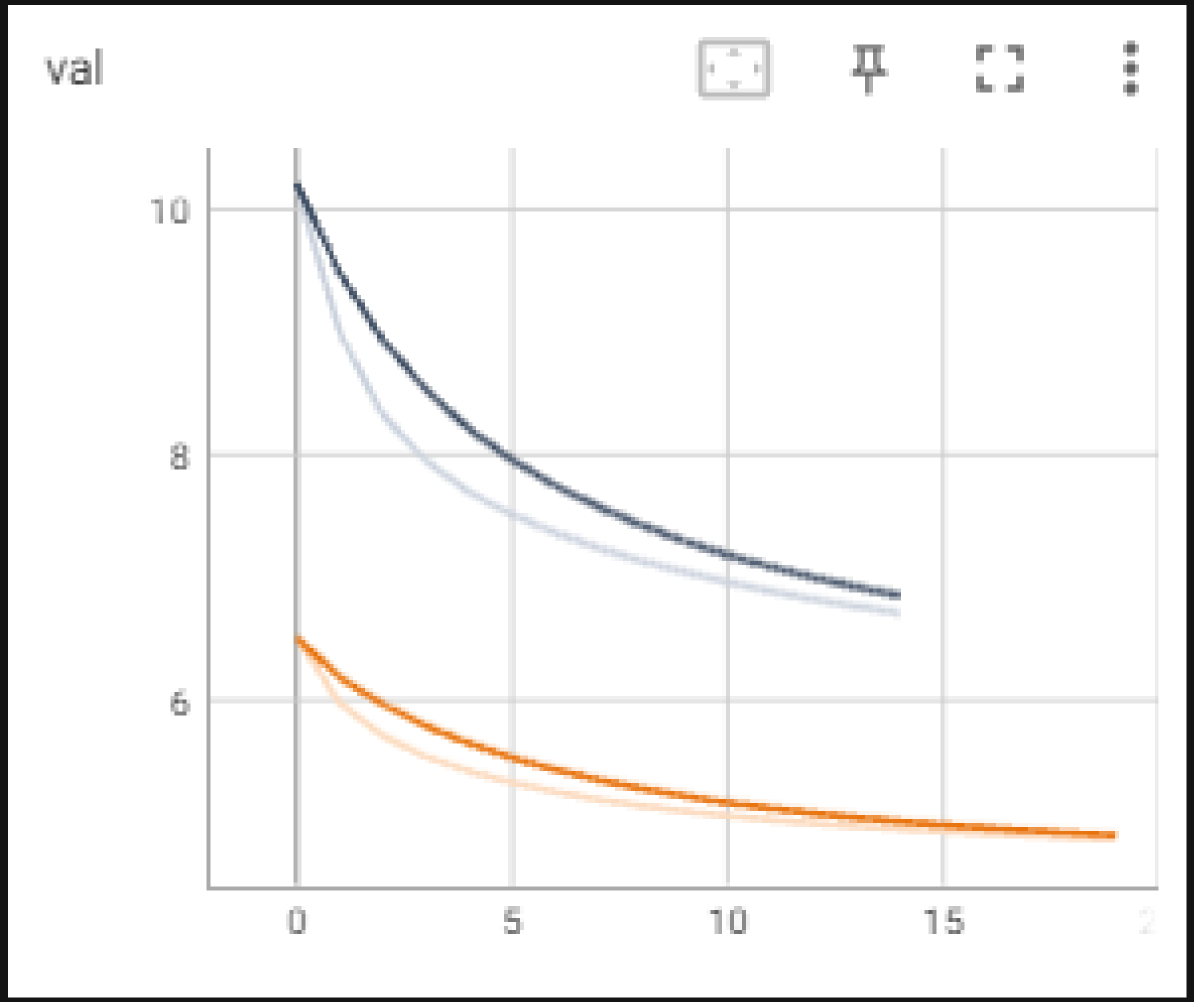
Attention:
Bahdanau
Embeddings:
- **Linear GE**
- **Simple GE**





Results



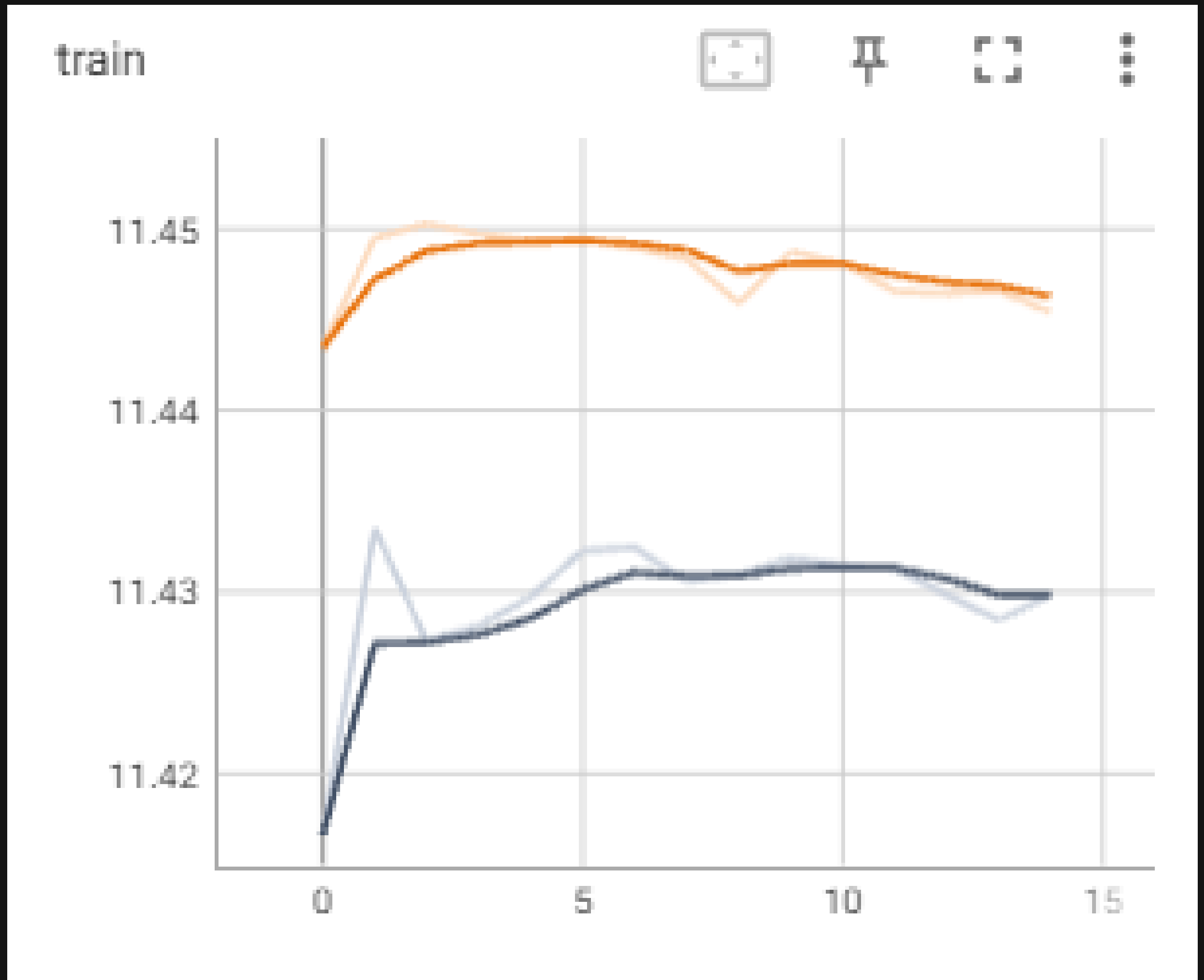


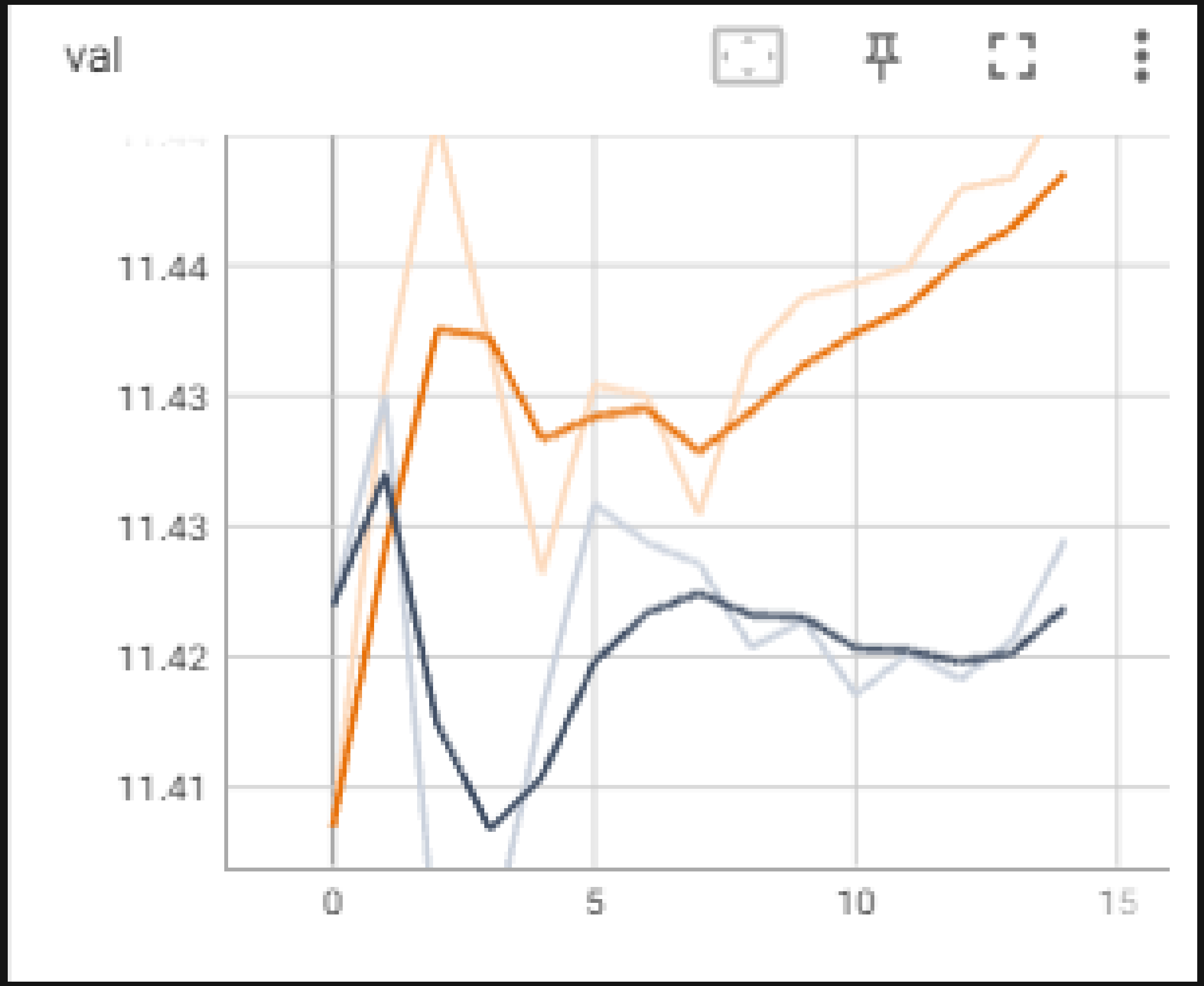
Linear embeddings:

N = 200_000

N = 400_000

Results





Embeddings:

- Node2Vec
- DeepWalk

GitHub repo

Nina-Konovalova/TCP-RL-Skoltech_project

readme.md

RL-Travelling Salesman Problem

Our project is based on Reinforcement Learning (RL) for solving Travelling Salesman Problem (TSP). Our code and experiments around the paper https://arxiv.org/abs/1802.04240.

We consider solving TCP solving with RL based on Pointer Network.

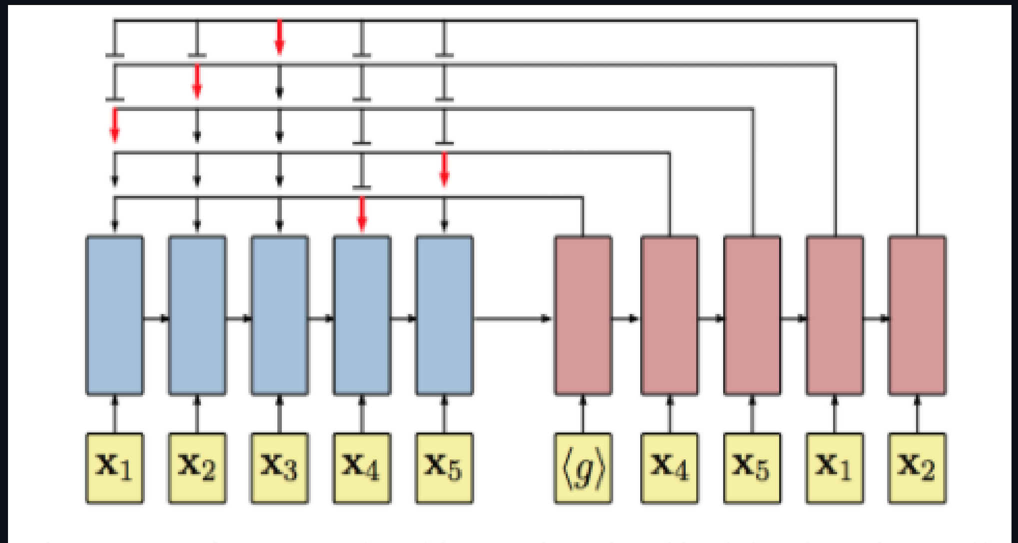
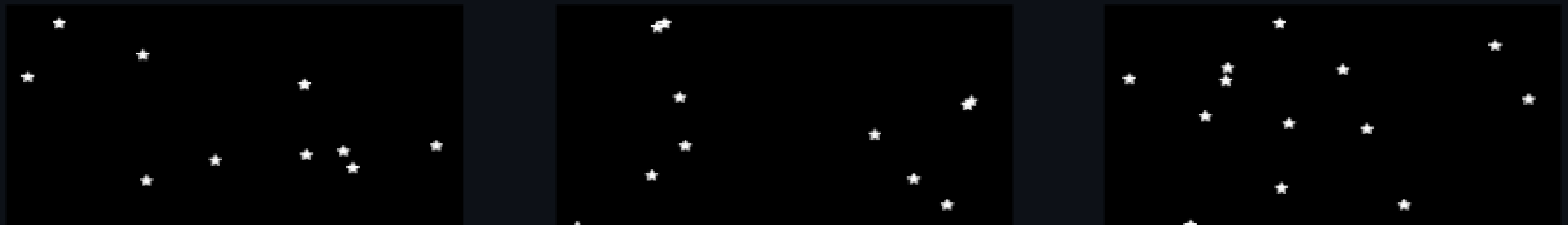


Figure 1: A pointer network architecture introduced by (Vinyals et al., 2015b).

As the dataset 20 uniform distributed points from 0 to 1 for each coordinates were used.

map of the city



Documentation.md

Different inference commands

model

`-e` or `--epochs` - number of epochs. Default: **30**;

`-embedding` or `--embedding_size` - size of embeddings. Default: **128**;

`--embedding_type` - type of embeddings. Default: **simple**. Other possible options: **linear**, **other**;

`-b` or `--batch_size` - batch size. Default: **1024**;

dataset

`-train_size` - size of train dataset for linear and simple embeddings. Default: **100_000**;

`--val_size` - size of val dataset for linear and simple embeddings. Default: **1_000**;

`--path_train` - path for other saved train embedding. Default: **OtherNode2Vec_train.csv**;

`--path_val` - path for other saved val embeddings. Default: **OtherNode2Vec_val.csv**.

Conclusion

TSP problem: Pointer Network (Attention) + model-free policy-based optimization (REINFORSE)

Linear Node Embeddings work better with both types of Attention

Showed the effect of the size of the Network on the results

Thank you for your attention!!!