# ML-BASED COMBINATORIAL OPTIMIZATION FOR SOLVING THE REALISTIC VEHICLE ROUTING PROBLEM

Final Presentation
Student Research Project, Winter Semester 22/23
02.12.2022

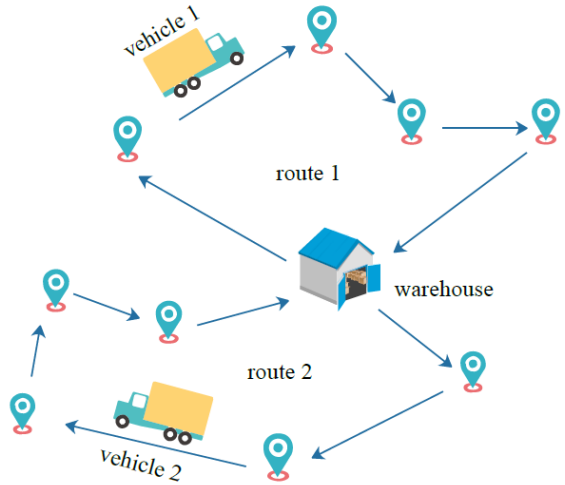
Project Supervisor: Daniela Thyssens

Student team: Ashish Sahu, Ilayda Topuz, Jaishree Janu, Jatin Garg, Sophia Lawal

# AGENDA

- Introduction to Vehicle Routing Problem (VRP)
- Research Workflow
- Methodology
  - Realistic data generation
  - Setup baseline model and training environment
  - Proposed model architecture for time dependent dynamism
  - Experiments with various model configurations
- Experiments
  - Experimental set-up
  - Evaluation of test results vs baseline
  - Training curves
  - Sample tour solutions
- Conclusion and Discussion

# INTRODUCTION TO VRP (OVERVIEW)




[Image Source: Duan et al, 2020]

## Graph Structure

Nodes: Co-ordinates, Demands
Edges: Distance between node-pairs

## PROBLEM STATEMENT

- All vehicles start from the depot node
- Fulfill all N customer demands
- Visit all customers once (no partial fulfilment)
- Final stop for all → Depot
- Time window constraints have not been considered

# RESEARCH WORKFLOW

## Realistic Data Generation

- Extracted co-ordinates from Open Street Map (OSM)
- Generated graph instances by sampling nodes and their respective distances
- Extracted live traffic variations to generate time dependent dynamism in graph edge features

## Baseline implementation

- Setup training components and model architecture in Tianshou & PyTorch frameworks.
- Trained baseline model on the generated realistic datasets (also with edge dynamism)
- Evaluated the results for test data and extract optimal tours, then compare with ours

## Graph Attention with edge feature encoding

- Experimented with different model configurations to encode edge features in graph context
- Finalized the model architecture for feeding edges to the attention-based encoder
- Experimented and evaluated the model to compare with baselines

## Experiments

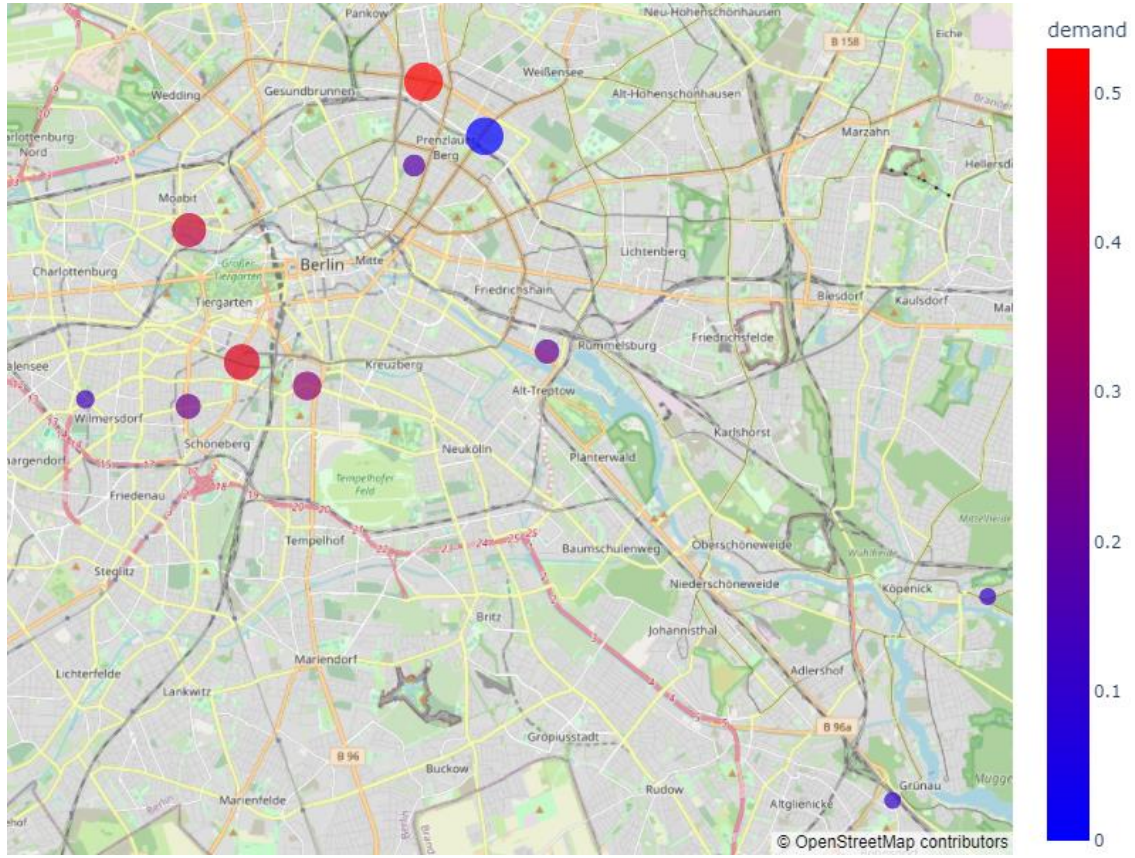
- Evaluation Metrics → Rewards and Loss vs trained epochs
- Performed hyperparameter tuning to optimize model training process
- Performed various experiments with different graph sizes and datasets with dynamic as well as static distance edges.

# REALISTIC DATA GENERATION

# GENERATE GRAPH INSTANCES

## Real-time locations from Open Street Map (stores in Berlin)

- For a given address query [city_name, shop_type], coordinates and distances are taken from OSM*
- OSM gives real and asymmetric distances between respective couple of coordinates. Demands are generated stochastically from random Gaussian distributions.
- Efficiently generated large scale realistic graphs → instead of naively sampling with iterator, we created a large pool of co-ordinates and global distance matrix then subsampled graph instances.
- 2000 locations of supermarkets, convenience stores & grocery shops in Berlin was taken for generating global node pool.

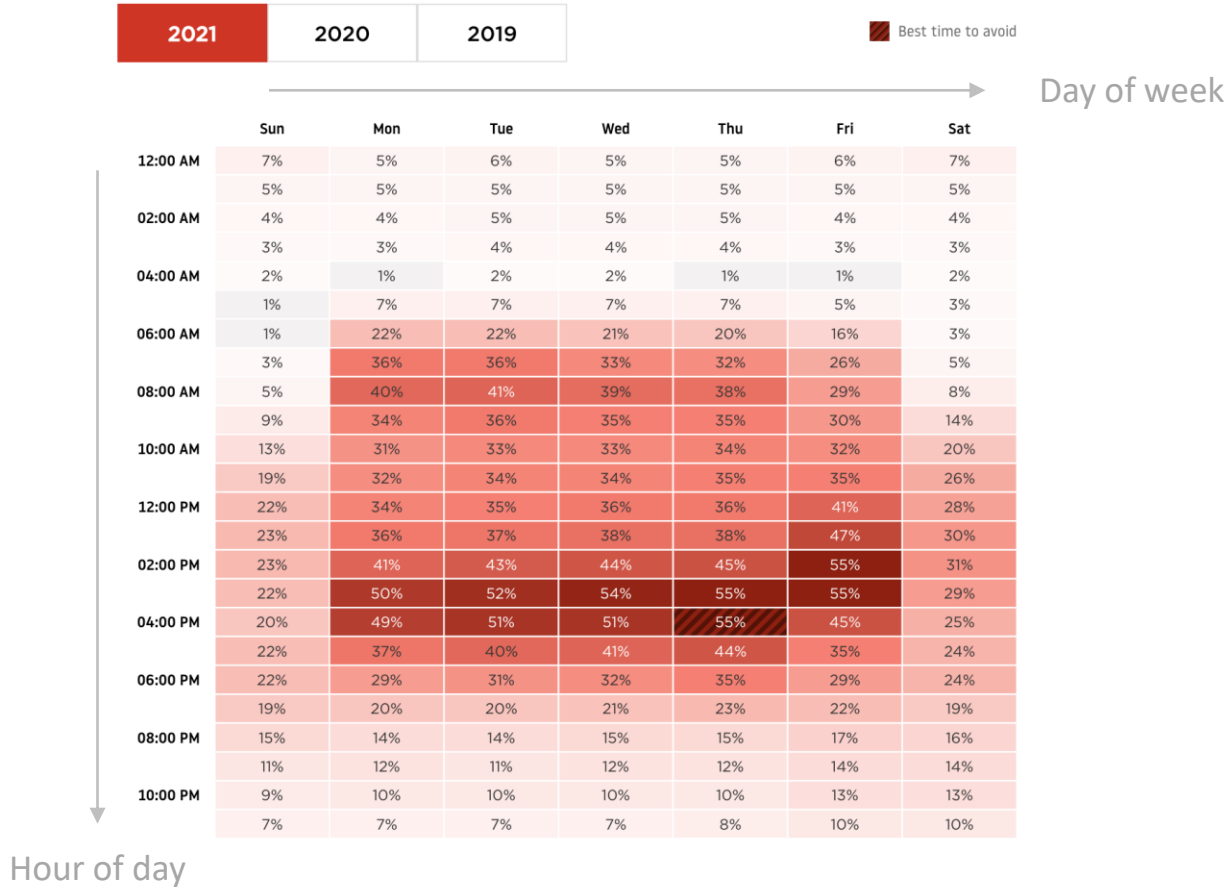



*Accessed Overpass API for OSM node coordinates and distance features. https://python-overpy.readthedocs.io/en/latest/introduction.html

# REALISTIC DATA GENERATION

# DYNAMIC EDGES BY TRAFFIC INDEXING

- For more realistic graph instances, we studied the traffic congestion levels in the city (Berlin) and created timing distortion masks
- These masks are in the form of adjacency matrices, containing delay level information for augmenting graphs with delay-adjusted edge features
- Downloaded hourly traffic variation of past three years and performed aggregations (*AVG.*)
- A 30% congestion level means that on average, travel times were 30% longer than during the static conditions.
  *[This means that a 30-minute trip driven in free-flow condition will take 9 minutes longer when the congestion level is at 30%.]*




| | Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|---|
| 12:00 AM | 7% | 5% | 6% | 5% | 5% | 6% | 7% |
| | 5% | 5% | 5% | 5% | 5% | 5% | 5% |
| 02:00 AM | 4% | 4% | 5% | 5% | 5% | 4% | 4% |
| | 3% | 3% | 4% | 4% | 4% | 3% | 3% |
| 04:00 AM | 2% | 1% | 2% | 2% | 1% | 1% | 2% |
| | 1% | 7% | 7% | 7% | 7% | 5% | 3% |
| 06:00 AM | 1% | 22% | 22% | 21% | 20% | 16% | 3% |
| | 3% | 36% | 36% | 33% | 32% | 26% | 5% |
| 08:00 AM | 5% | 40% | 41% | 39% | 38% | 29% | 8% |
| | 9% | 34% | 36% | 35% | 35% | 30% | 14% |
| 10:00 AM | 13% | 31% | 33% | 33% | 34% | 32% | 20% |
| | 19% | 32% | 34% | 34% | 35% | 35% | 26% |
| 12:00 PM | 22% | 34% | 35% | 36% | 36% | 41% | 28% |
| | 23% | 36% | 37% | 38% | 38% | 47% | 30% |
| 02:00 PM | 23% | 41% | 43% | 44% | 45% | 55% | 31% |
| | 22% | 50% | 52% | 54% | 55% | 55% | 29% |
| 04:00 PM | 20% | 49% | 51% | 51% | 55% | 45% | 25% |
| | 22% | 37% | 40% | 41% | 44% | 35% | 24% |
| 06:00 PM | 22% | 29% | 31% | 32% | 35% | 29% | 24% |
| | 19% | 20% | 20% | 21% | 23% | 22% | 19% |
| 08:00 PM | 15% | 14% | 14% | 15% | 15% | 17% | 16% |
| | 11% | 12% | 11% | 12% | 12% | 14% | 14% |
| 10:00 PM | 9% | 10% | 10% | 10% | 10% | 13% | 13% |
| | 7% | 7% | 7% | 7% | 8% | 10% | 10% |

Data Source: Tomtom traffic index study: https://www.tomtom.com/traffic-index/berlin-traffic#statistics

# MODEL IMPLEMENTATION

# TRAINING ENVIRONMENT SETUP

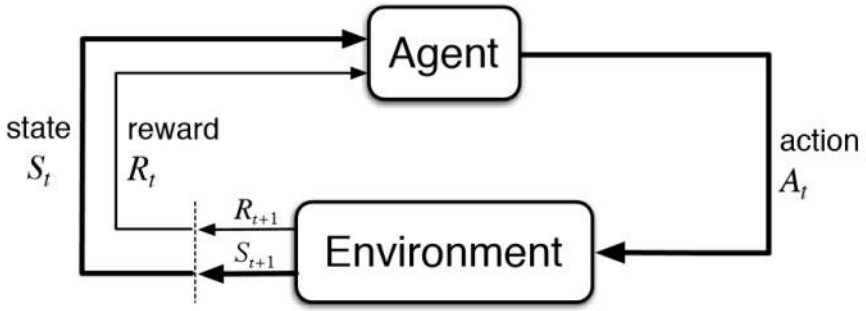
Get actions from the model

What is current state?

$x_t^i$: current position of the vehicle

$o_t$: remaining capacity of the vehicle

$M_t$: action mask: possible actions at current time step




Make environment step as per the selected actions

$r_t^{ij}$: action rewards

Set-up graph environment

$n^i$: node features from the graph

$e_{ij}$: edge: distance between nodes

$d^i$: customer demands at the node

*Tianshou is a reinforcement learning platform based on PyTorch, providing a fast-speed framework and a pythonic API.
Image source: https://tianshou.readthedocs.io/en/master/tutorials/dqn.html

# MODEL IMPLEMENTATION

# BASELINE MODEL ARCHITECTURE

Encoder-Decoder framework for solving VRP

## ENCODER

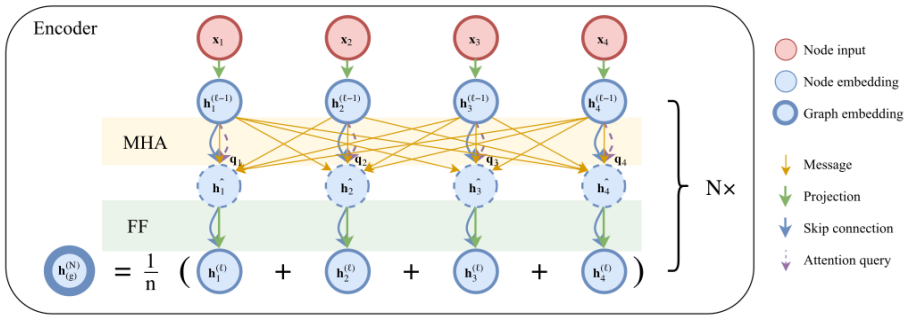



Fig. 2 : Multi-head Attention based encoder inspired from Transformer Architecture

Takes raw node embeddings and generate fixed size graph context

## DECODER

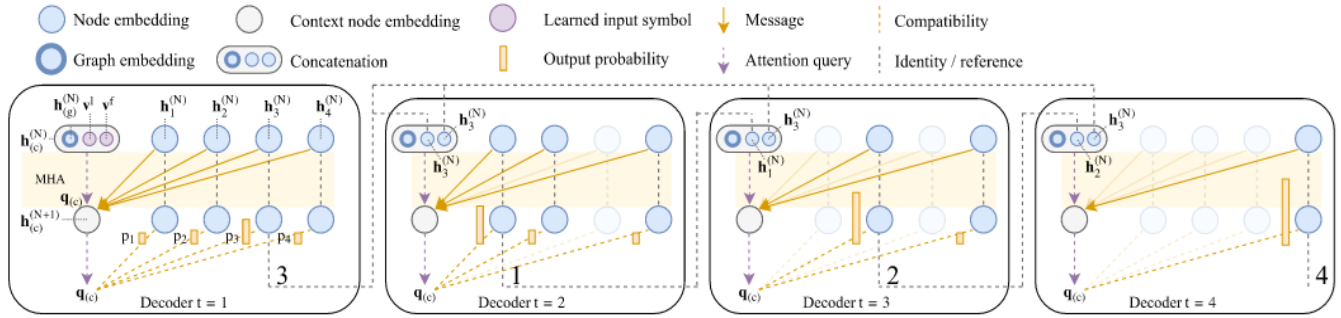


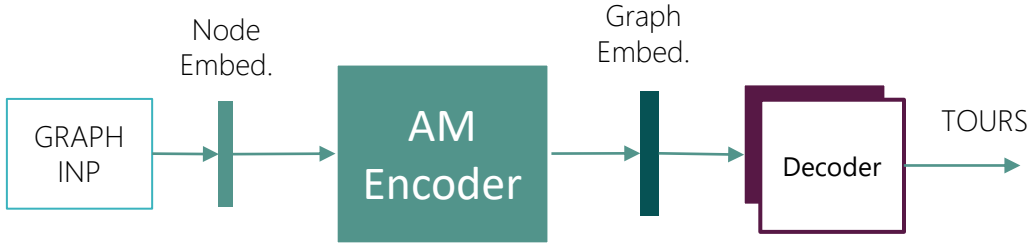

Fig. 3 : Attention based decoder

Takes graph context embeddings and generate tours autoregressively

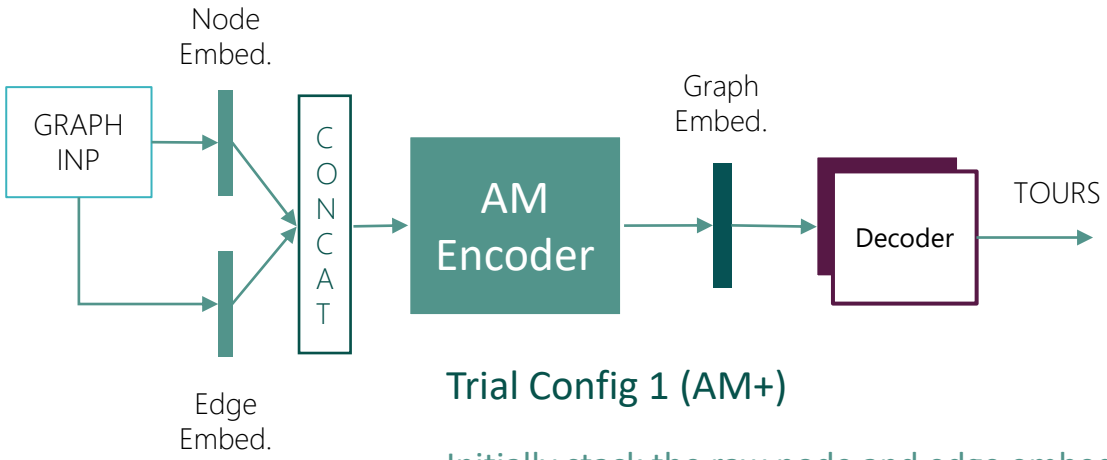
[Inspired from: Velickovic et al, 2018; Duan et al, 2020; Kool et al, 2019; Li et al, 2019 GAT: Graph Attention Network]

# MODEL IMPLEMENTATION

# EXTENDING BASELINE WITH EDGE CONTEXT

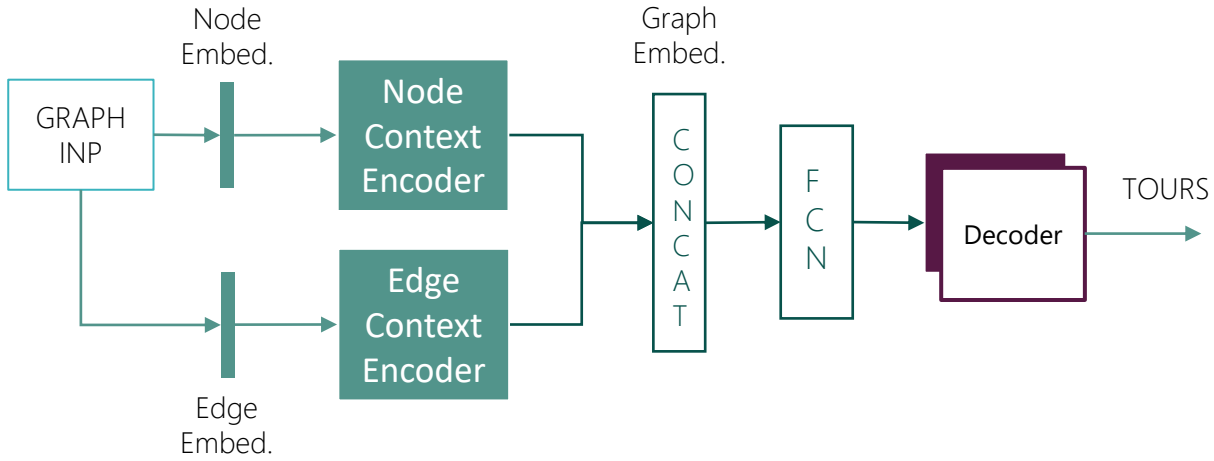



Baseline (AM)




Trial Config 1 (AM+)

Initially stack the raw node and edge embeddings,
then process through graph encoder



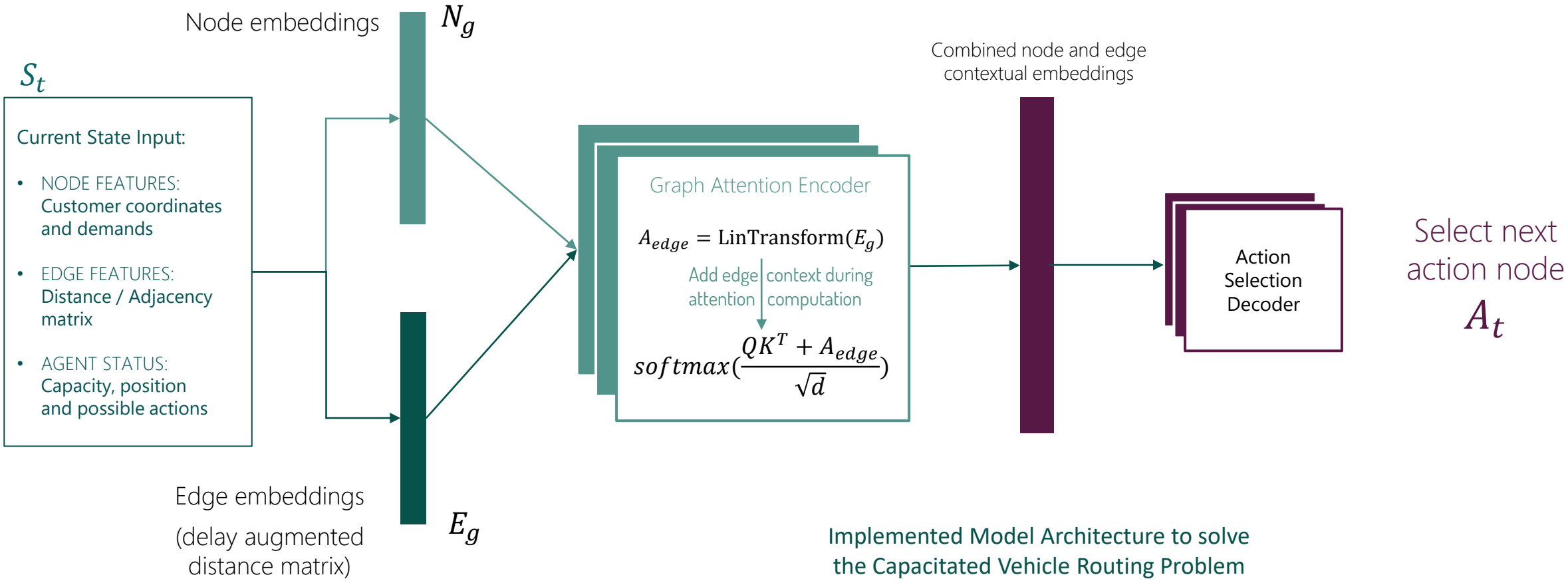

Trial Config 2 (AM++)

Generate node and edge contexts,
then stack and bring to same embedding space with FCN

[Inspired from: Velickovic et al, 2018; Duan et al, 2020; Kool et al, 2019; Li et al, 2019
GAT: Graph Attention Network]

# MODEL IMPLEMENTATION

# ENCODING EDGES FOR GRAPH ATTENTION




Implemented Model Architecture to solve the Capacitated Vehicle Routing Problem with dynamic and realistic edge features

# EXPERIMENT PROCESS

# DATASETS, GRAPH SIZES AND PARAMETER TUNING

- Took 2000 locations of stores in Berlin (from Open Street Map), determined all distance pairs to create a global distance matrix
- One dataset with time-indexed edged features *(RealData-D)* while other with static node distances *(RealData-S)*
- Each graph instance was augmented with 24 delay levels, representing congestion levels throughout the day
- Different graph sizes: 10, 20, 30 and 50
- Training size: 10,000; Test size: 5,000. Also generated large scale datasets with 50k, 100k & 500k instances.
- Grid search for tuning hyperparameters (learning rate, weight decay, etc.)

# EXPERIMENT PROCESS

# EVALUATION OF RESULTS

| Dataset | Realitistic Data (static edges) | | | Realitistic Data (time-indexed edges) | | |
|---|---|---|---|---|---|---|
| Graph size | Attention Model (AM) | Edge+ AM (Ours) | Gap (vs base) | Attention Model (AM) | Edge+ AM (Ours) | |
| 10 | 81.576 | 81.114 | -0.56% | 106.312 | 105.924 | -0.36% |
| 20 | 170.821 | 150.645 | -11.81% | 201.274 | 200.606 | -0.33% |
| 30 | 252.196 | 254.413 | +0.88% | 309.903 | 311.544 | +0.53% |

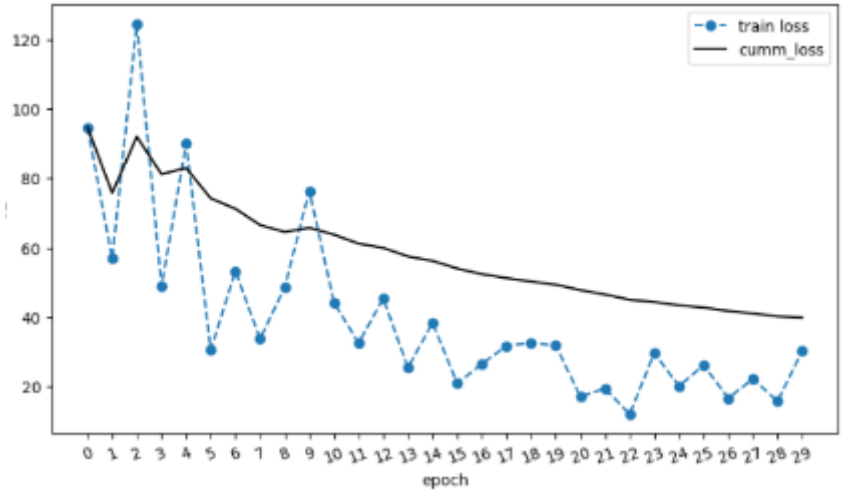
## Best Test Reward values (tour length in km)

Total data size: 20k static, 24k dynamic graphs
Test / Train ratio: 50%
Epochs trained: 30

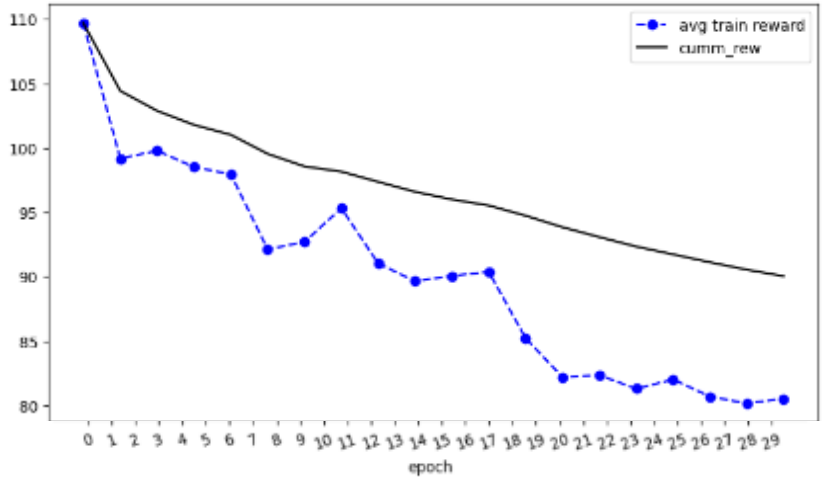
# EXPERIMENT PROCESS

# TRAINING CURVES AND REWARDS

## Training loss vs Epochs




## Collected train rewards vs Epochs

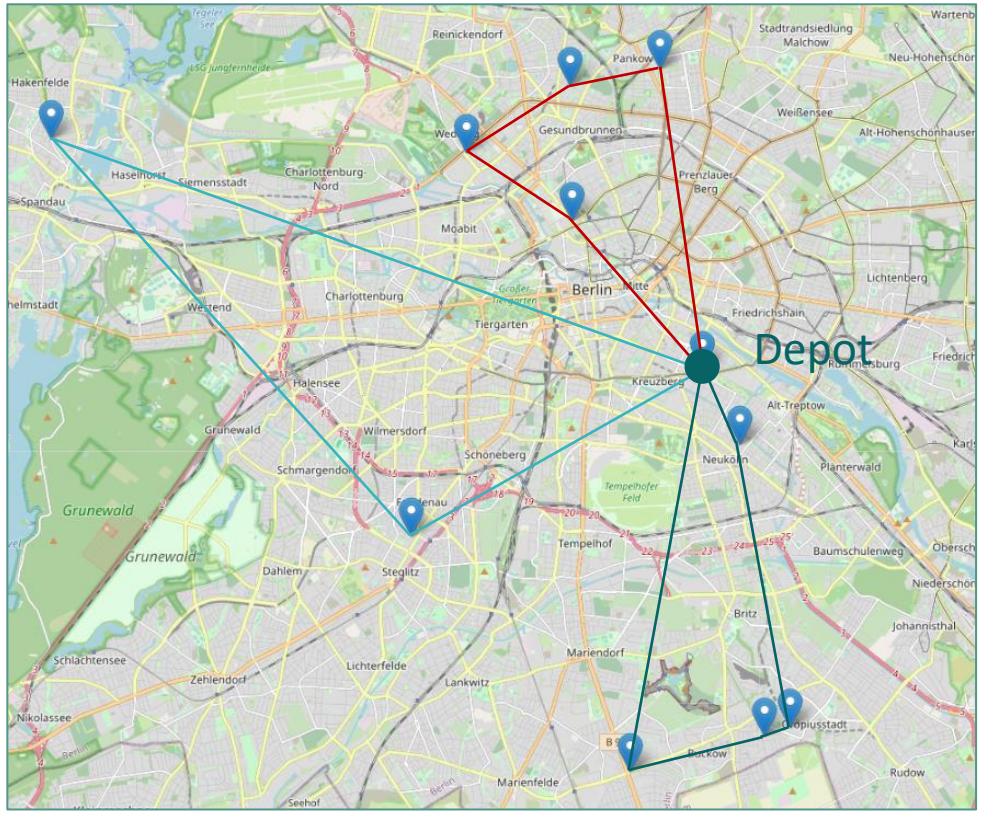



Training setting: Realistic static edge feature data
Total data size: 20k static, 24k dynamic graphs
Test / Train ratio: 50%
Epochs trained: 30

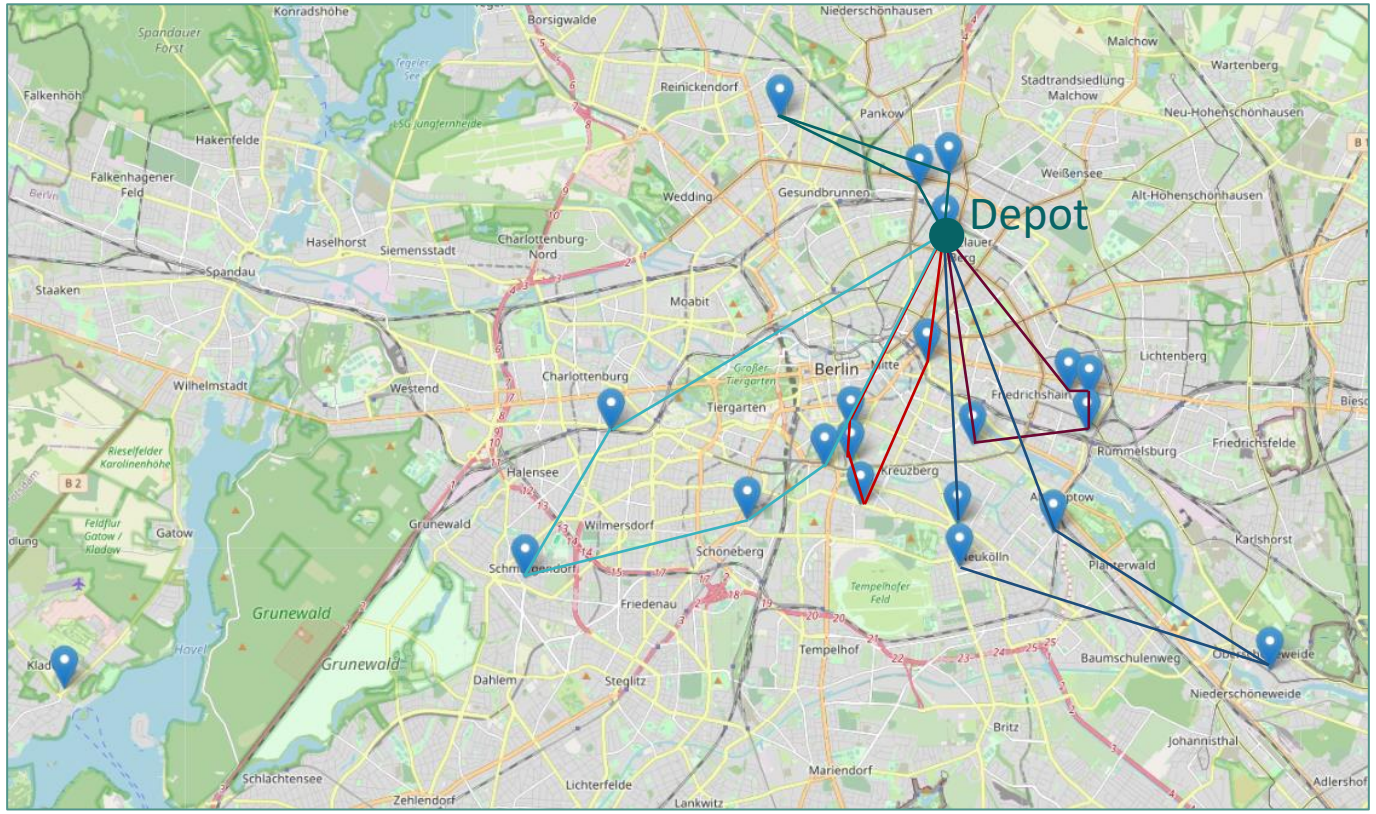
# EXPERIMENT PROCESS

# SAMPLE SOLUTION REALISTIC TOURS

N = 10; Length = 82.754 Km




N = 20; Length = 202.916 Km



# CONCLUSION AND FUTURE SCOPE

- In this project, we have solved the Capacitated Vehicle Routing Problem (CVRP) by Deep Reinforcement Learning on realistically generated datasets.
- For both the datasets, our approach achieves better performance for small and medium graph sizes (in the given training setting). For N=30, the method is performing at par with the Attention Model baseline
- We have generated large scale realistic datasets for CVRP problems. Our method is scalable and can be applied to different cities to adapt routing solutions as per the traffic congestion in the city/state.
- As a future step, we will scale up the model training with large scale data (500k, 100k, etc.) and will try to cope with OOM limitations with computational tricks. Furthermore, we can evaluate our realistic generation method on various ML-based solutions – can it act as a benchmark generation method for the community?

# REFERENCES


[1] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural Combinatorial Optimization with Reinforcement Learning. CoRR, abs/1611.09940, 2016. arXiv: 1611.09940.

[2] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. European Journal of Operational Research, 290(2):405–421, April 2021.

[3] Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, Jiangwen Wei, Xiaodong Zhang, and Yinghui Xu. Efficiently Solving the Practical Vehicle Routing Problem: A Novel Joint Learning Approach. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20, pages 3054–3063, New York, NY, USA, August 2020. Association for Computing Machinery.

[4] Wouter Kool, Herke van Hoof, and Max Welling. Attention, Learn to Solve Routing Problems! Feb 2019.

[5] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Seungjai Min, and Youngjune Gwon. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. Advances in Neural Information Processing Systems, pages 21188–21198, 2020. arXiv: 2010.16011.

[6] Jingwen Li, Yining Ma, Ruize Gao, Zhiguang Cao, Andrew Lim, Wen Song, and Jie Zhang. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Transactions on Cybernetics, pages 1–14, 2021.

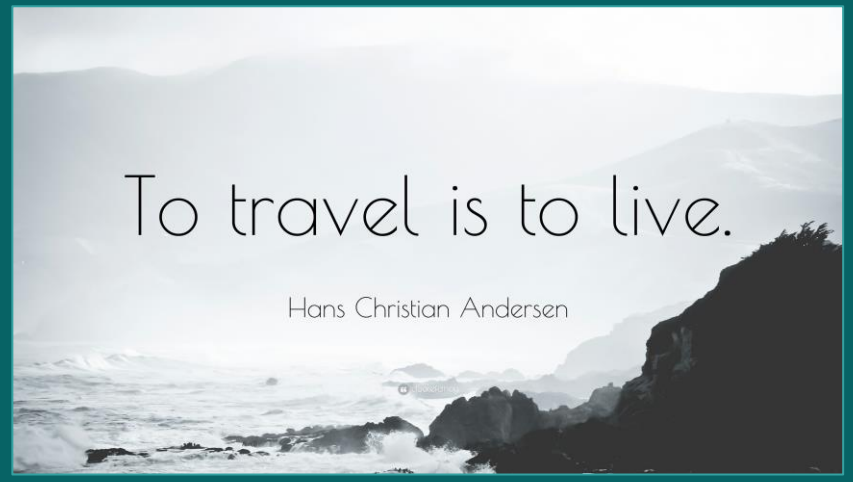
[7] Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to Iteratively Solve Routing Problems with Dual-Aspect Collaborative Transformer. May 2021.

[8] Daniela Thyssens, Jonas Falkner, and Lars Schmidt-Thieme. Supervised permutation invariant networks for solving the cvrp with bounded fleet size. 2022.

[9] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. In Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015.

[10] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. & Bengio, Y. (2017). Graph Attention Networks. 6th International Conference on Learning Representations.

# THANK YOU FOR YOUR ATTENTION!




Let's Discuss..

# PROBLEM STATEMENT AND RESEARCH SCOPE

# APPENDIX A I

## GRAPH STRUCTURE

- One depot and multiple customer locations (N) with demands + co-ordinates
- Heterogeneously capacitated: Fleet of vehicles with different capacities (cars, trucks, etc.)
- Directed asymmetric graph: Distances between node-pairs are asymmetric, $d_{AB} \neq d_{BA}$
- Dynamic distances: Distances can change when vehicle in transit due to traffic/weather (TBD*)

## LEARNING METHOD

- Dual objective: Minimize total distance travelled by each vehicle + number of vehicles used
- Construction approach: Generate policy regressively, at each step predicting next node distribution
- DeepRL methodology: Encoder-Decoder framework, determine near optimal solution

## ASSUMPTIONS

- All vehicles start & end at depot node, and each customer is visited exactly once
- Vehicle has sufficient capacity to fulfill all customer demands within one tour-path
- Demands are fixed, all the vehicles have same speed and no time window constraints

# RELATED WORK AND THEIR LIMITATIONS

# APPENDIX A2

## Pointer Networks

- RNN + Attention based methods to generate optimal tour policy with sequence-to-sequence learning approach
- Supervised Learning setting requires carefully generated ground-truth labels for good model predictions, only for TSP
- **Actor-Critic Algorithm**: PN without supervised setting, uses training instance tour costs as Monte-Carlo estimates for stochastic policy

[Bello et al, 2016; Vinyals et al. 2015]

## Graph Neural Networks

- Graph Convolutions (GCN) and Graph Attention (GAT) based encoders can be used to learn node context embeddings at different time steps. Attention based decoders used to compute distribution over policies
  [Duan et al, 2020; Kool et al, 2019]
- For HCVRP (heterogeneous capacity), a 2-level decoder was proposed: first one selects a vehicle, and the other predicts next node to visit for that vehicle
  [Li et al, 2019]
- These methods use either Euclidean distance between stochastically generated nodes or homogeneous fleet for learning, failing to model complex real-world scenario

# PROBLEM APPROACH
# TRAINING OBJECTIVE AND ALGORITHM

## LOSS FUNCTION

- Minimize total length of tours: sum of distances between selected nodes
- Learning objective function:
  - $m_{\pi_t \pi_{t+1}}$ is travel distance between current time position and selected next node

$$\min \sum_{t=1}^{T-1} m_{\pi_t \pi_{t+1}}$$

## TRAINING ALGORITHM

- REINFORCE – Policy Gradient Algorithm (with baseline)
- Gradient estimator for parameter update:

$$\nabla \mathcal{L}(\boldsymbol{\theta}|s) = \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)} \left[ (L(\boldsymbol{\pi}) - b(s)) \nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s) \right]$$

# PROBLEM APPROACH

# HOW TO PERFORM A STEP IN ENVIRONMENT

## Transition

Given the current state, transit to next node with highest probability

Transition from previous state $S_t$ to the next state $S_{t+1}$ based on the performed action $A_t$:

Vehicle capacity update: $o_{t+1} = o_t - d_t^i$

Action mask update: $M_{t+1} = mask(M_t, A_t)$

Partial route update: $\pi_{t+1} = [\pi_t, A_t]$

Current position update: $x_{t+1}^i = A_t$

## Reward

The agent will be trained to maximize reward

Reward is defined as the negative of the tour length, which corresponds to the distance between nodes (from edge feature matrix).

*Tianshou is a reinforcement learning platform based on PyTorch, providing a fast-speed framework and a pythonic API.

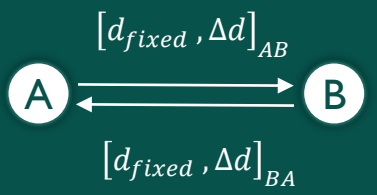
# RESEARCH IDEA

## DYNAMIC DISTANCES BETWEEN NODES

Distance travelled by a vehicle can change due to weather/construction, and so the tour length

Can we enable the model to handle this dynamism in edge features during the training time?

$d^w = d\ (\pm\Delta d)$ There can be delays due to weather, traffic condition, etc. any time


$d^a = d\ (\pm\Delta d)$ There might be some alternate routes with slightly deviated distance/time

How to create this dynamism in graph, model and data generation?

# REALISTIC DATA GENERATION

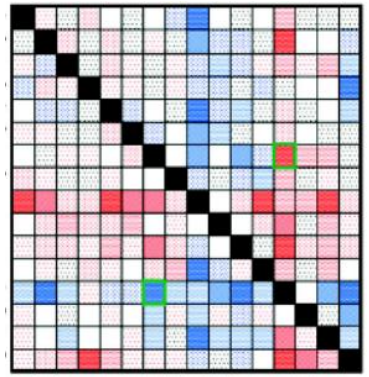
# DYNAMIC DISTANCING BETWEEN NODES

There can be delays due to weather, traffic condition, etc. any time

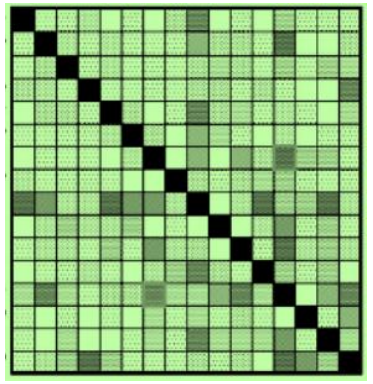
$d^w = d\ (\pm\Delta d)$


$d^a = d\ (\pm\Delta d)$

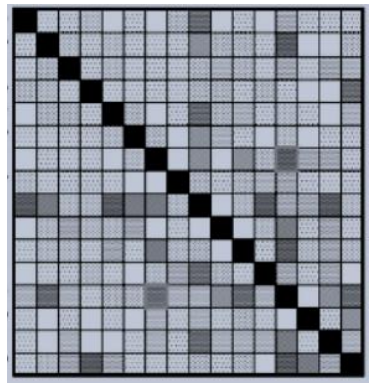
There might be some alternate routes with slightly deviated distance/time



×



×



## Edge features

Static distances between the nodes

Obtained from Open Street Map

## Timing distortions

Possible +- changes in distance/time

Noise distribution by domain knowledge

## Noise Weights

Learned model weights for dynamism

Fine-tune timing distortions by weights