JAGIELLONIAN UNIVERSITY
IN KRAKÓW

Ph.D. Qualification Project

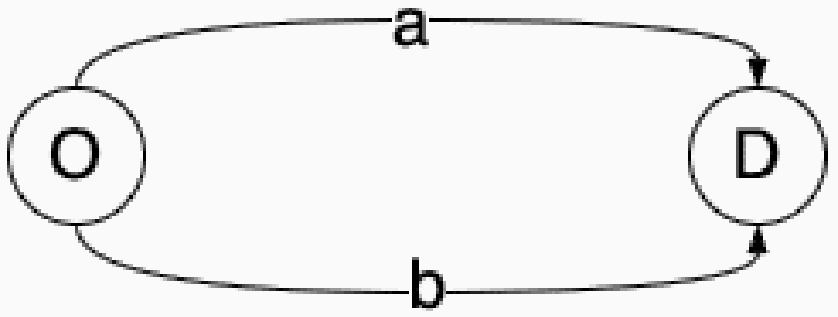
A. Onur Akman

OUTLINE

1. PROBLEM
2. RL-BASED SOLUTIONS
3. DL-BASED APPROACH WITH COMPLEXITIES
4. PROBLEM EXTENSION

PROBLEM

PROBLEM



- A set of Q individual travelers want to reach from their origin O to their destination D.
  - Everyday they choose between the two alternative routes: a or b .
- The cost (travel time) is given with a naive, non-linearly increasing BPR formula:

$$t_a(q_a) = t_a^0(1 + (q_a/Q_a)^2)$$

$t_a(q_a)$ - is the travel time on arc a (or b)

q_a - is the flow (number of vehicles using arc)

t_a^0 - is the free flow speed (with no other vehicles)

Q_a - is the capacity (maximal number of vehices)

Q = 1000 veh/h

t_a^0 = 5 min

t_b^0 = 15 min

Q_a = 500 veh/h

Q_b = 800 veh/h

PROBLEM

System Optimum

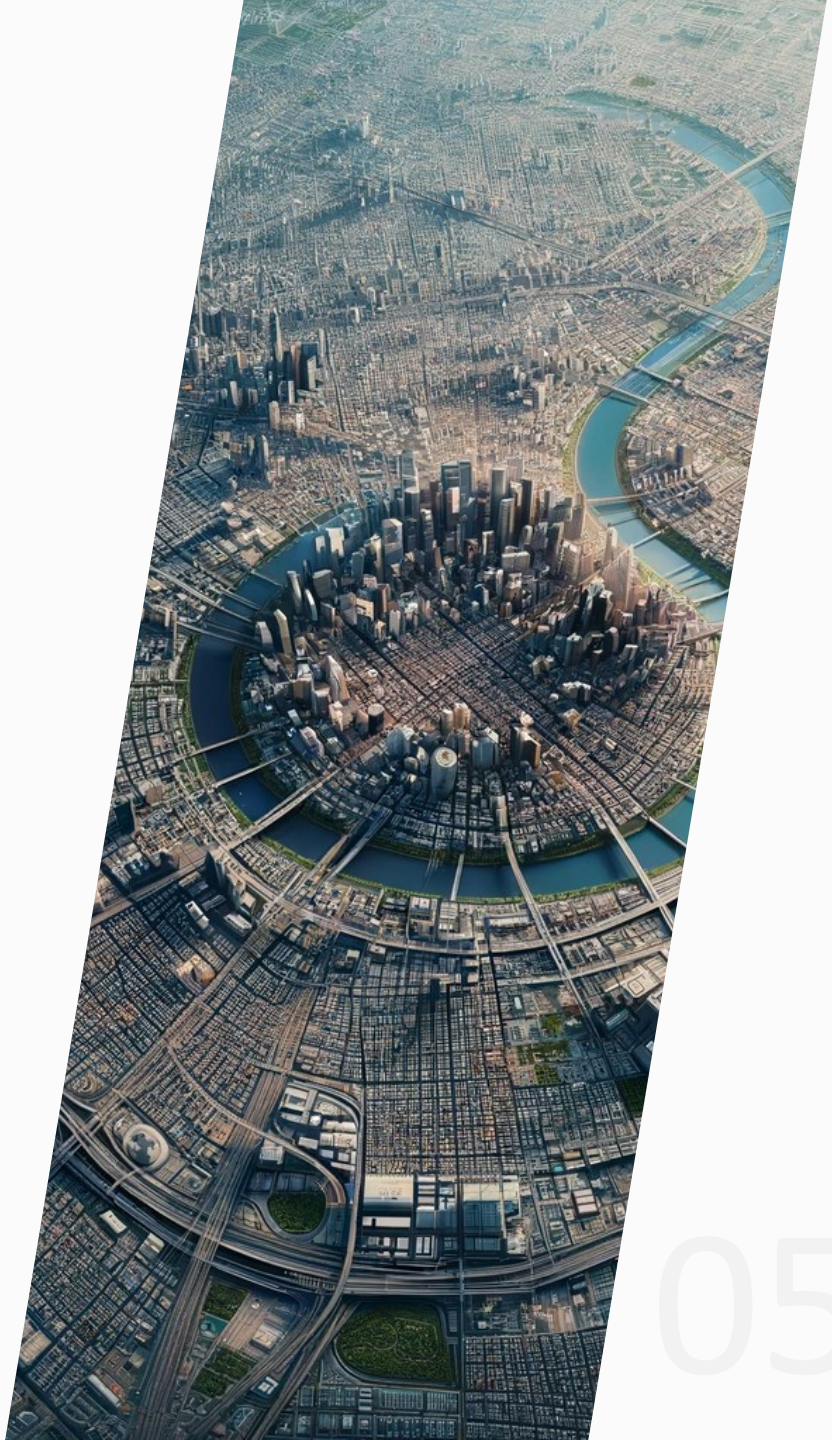
Optimal distribution of travelers between the routes that minimizes the total cost (travel time) across all individuals.

$$t_a(q_a) \cdot q_a + t_b(q_b) \cdot q_b$$
$$\text{s.t. } q_a + q_b = Q \text{ and } q_a, q_b \geq 0$$

User Equilibrium

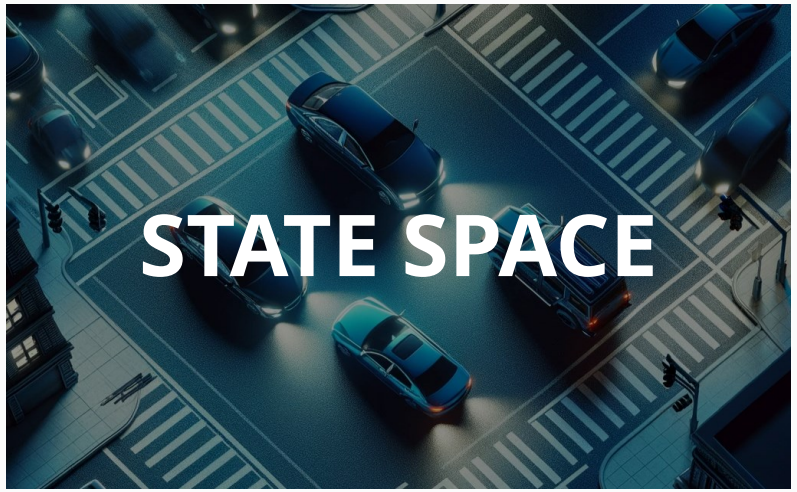
The configuration where every traveler is individually satisfied with their choice, meaning that the travel time on both routes is equal.

$$t_a(q_a) = t_b(q_b)$$




JAGIELLONIAN UNIVERSITY
IN KRAKÓW
RL-BASED
SOLUTIONS
06

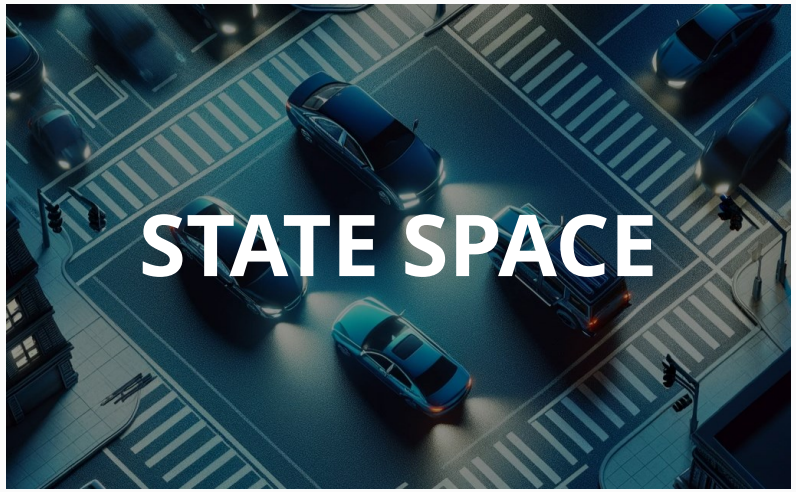
SYSTEM OPTIMUM (CENTRALIZED)



SYSTEM OPTIMUM (CENTRALIZED)

Traffic Environment

- Both roads are initially empty.
- Accepts two actions: assigning the next car in queue to a or b.
- Finalizes the episode when $q_a + q_b = \mathrm{Q}$



Discrete State Space

- Made of tuples: Number of cars in each route (q_a, q_b)
- Transition only in increment: Addition to one of the routes.
- Ideal for a table look-up policy.

SYSTEM OPTIMUM (CENTRALIZED)

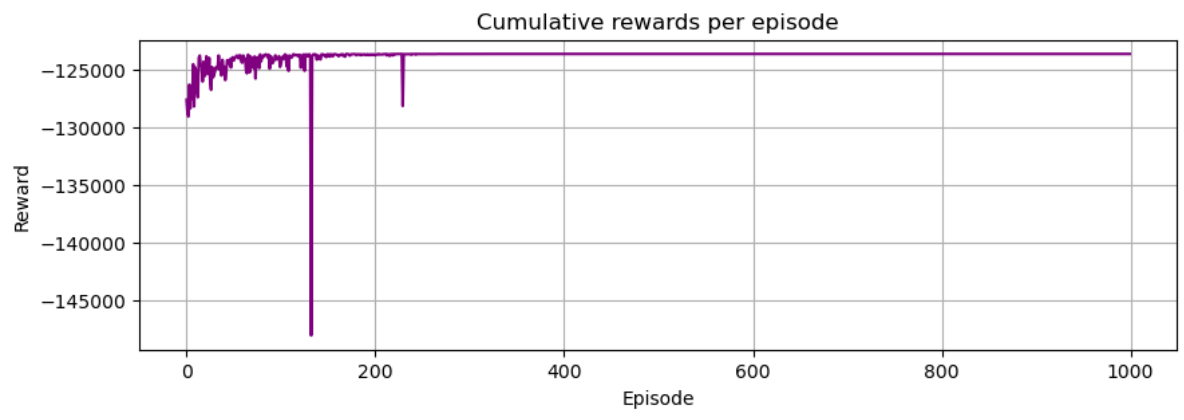
Central Agent

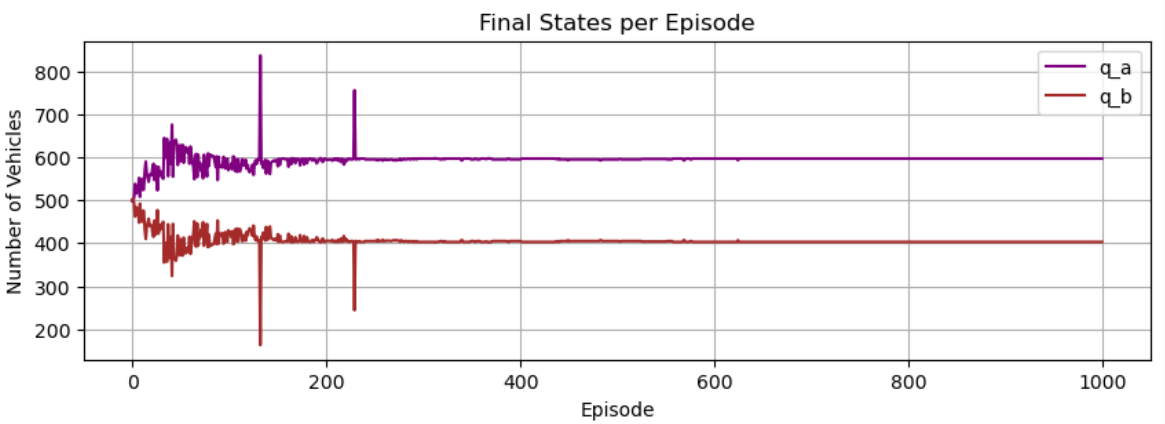
- Policy is a deterministic look-up table.
- Employs Q-Learning Algorithm, learns values of actions.
- A decaying epsilon for exploration.

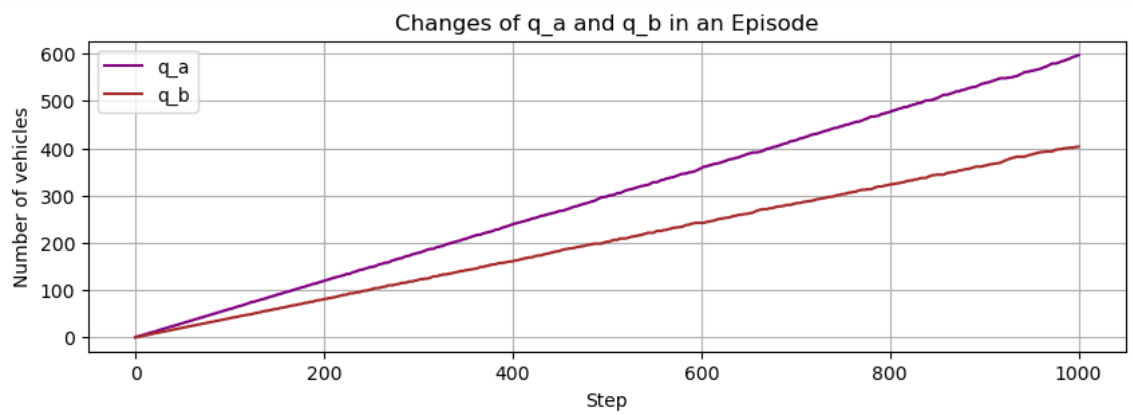
Dense Rewards

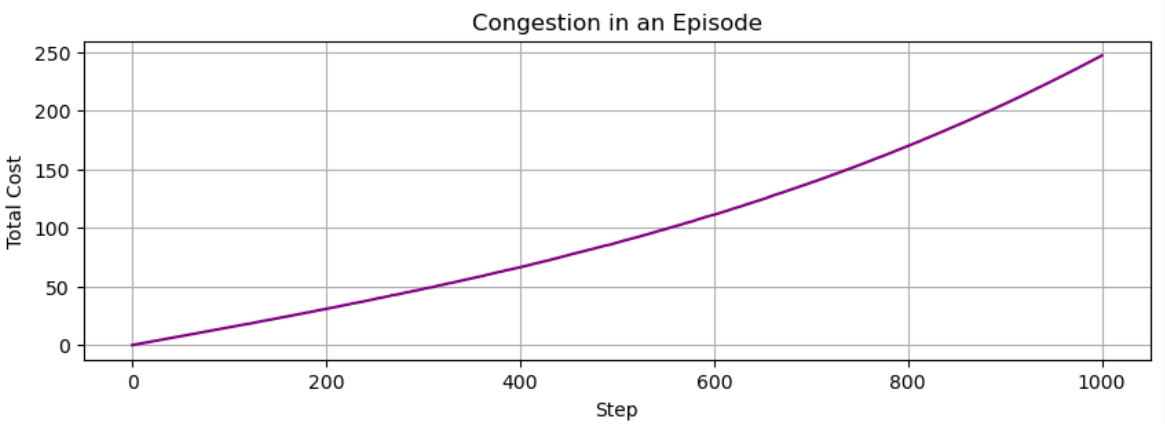
$$curr_Q = q_a + q_b$$
$$k1 = q_a + ((Q - curr_Q) * \frac{q_a}{curr_Q})$$
$$k2 = q_b + ((Q - curr_Q) * \frac{q_b}{curr_Q})$$
$$reward = -(t_a(k1) \cdot k1 + t_b(k2) \cdot k2) * \frac{curr_Q}{Q}$$

SYSTEM OPTIMUM (CENTRALIZED)









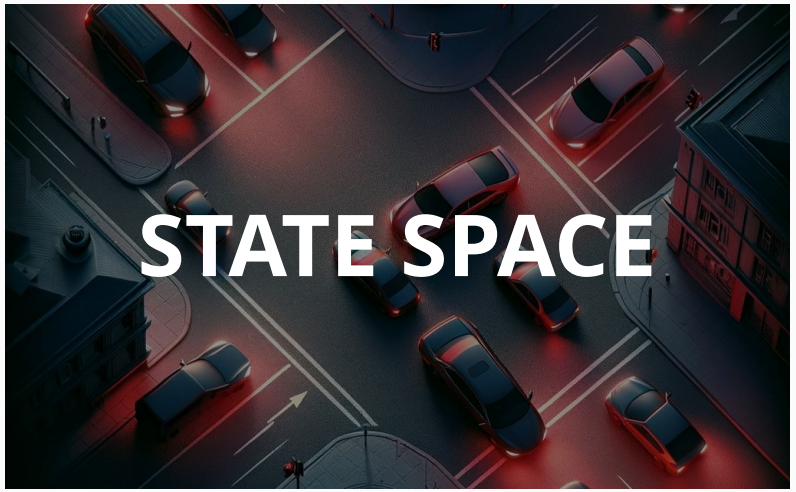
How is it in the multi-agent?

SYSTEM OPTIMUM (MULTI-AGENT)

SYSTEM OPTIMUM (MULTI-AGENT)

Traffic Environment

- Both roads are initially empty.
- Accepts two actions: Choosing a or b.



Discrete State Space

- Made of tuples: Current route, number of vehicles on each route (K, x, y)
- One transition for each agent: Start and after selection
- Ideal for a table look-up policy.

SYSTEM OPTIMUM (MULTI-AGENT)

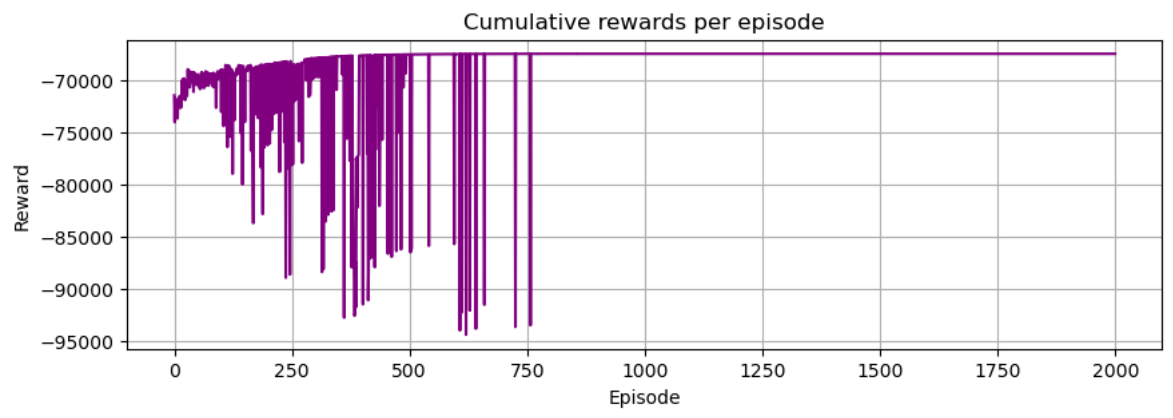
Each Vehicle is an Agent

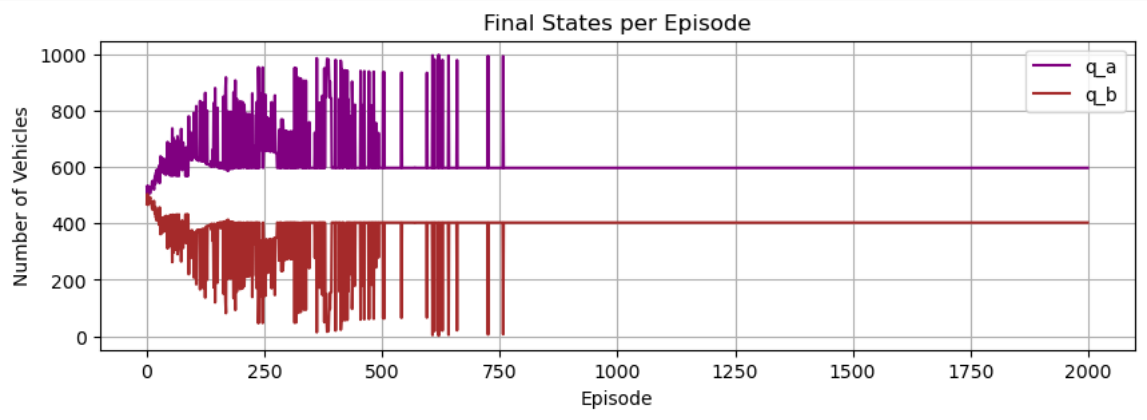
- Policy is a deterministic look-up table.
- Employs Q-Learning Algorithm, learns values of actions.
- A decaying epsilon for exploration.

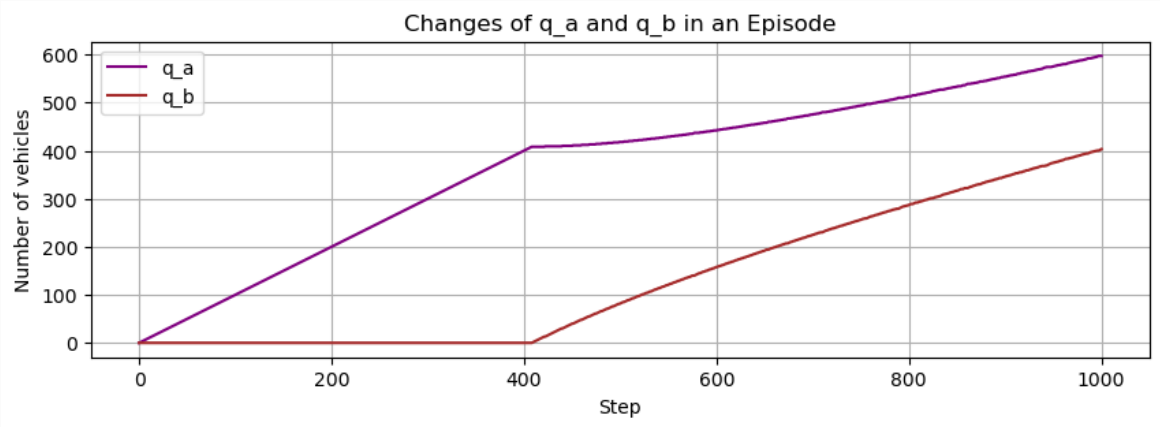
Dense Rewards

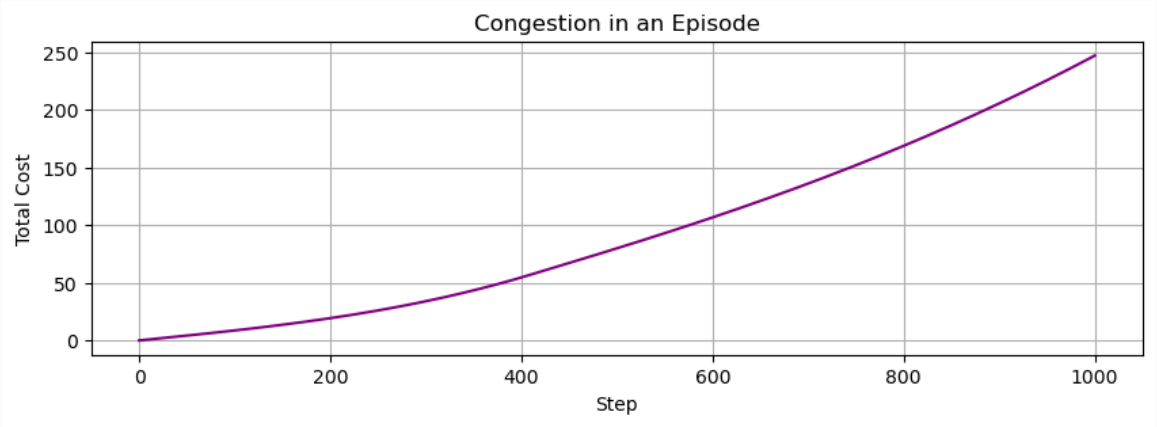
- Current cost (negative), scaled by the ratio of current number of vehicles and the total number of vehicles.
- Last vehicles are penalized more drastically.

SYSTEM OPTIMUM (MULTI-AGENT)



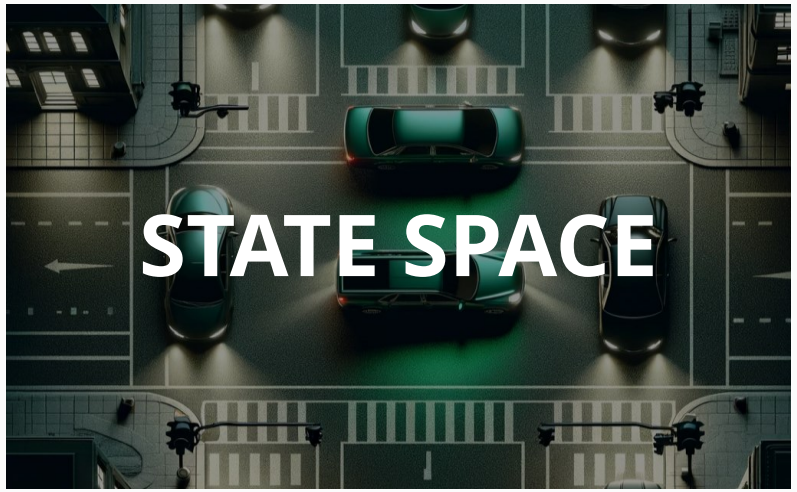




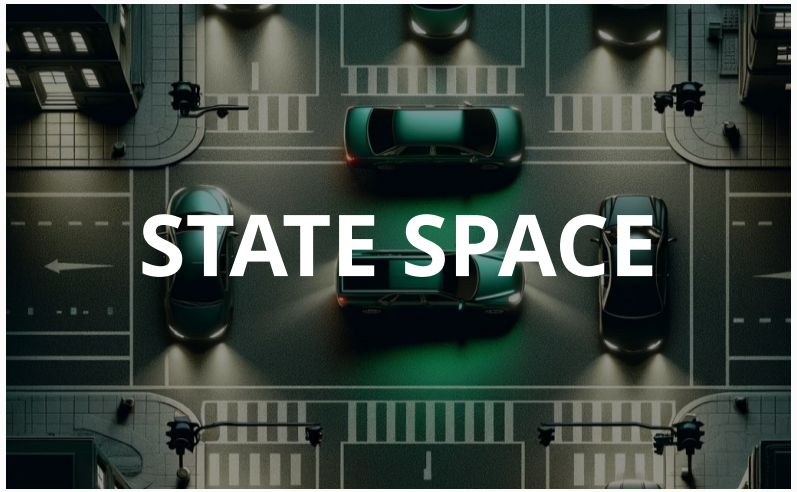


How was it in the central agent?

USER EQUILIBRIUM (CENTRALIZED)



USER EQUILIBRIUM (CENTRALIZED)



Traffic Environment

- Same as the SO version.
- Both roads are initially empty.
- Accepts two actions: assigning the next car in queue to a or b.
- Finalizes the episode when $q_a + q_b = \mathrm{Q}$

Discrete State Space

- Same as the SO version.
- Made of tuples: Number of cars in each route (q_a, q_b)
- Transition only in increment: Addition to one of the routes.

USER EQUILIBRIUM (CENTRALIZED)

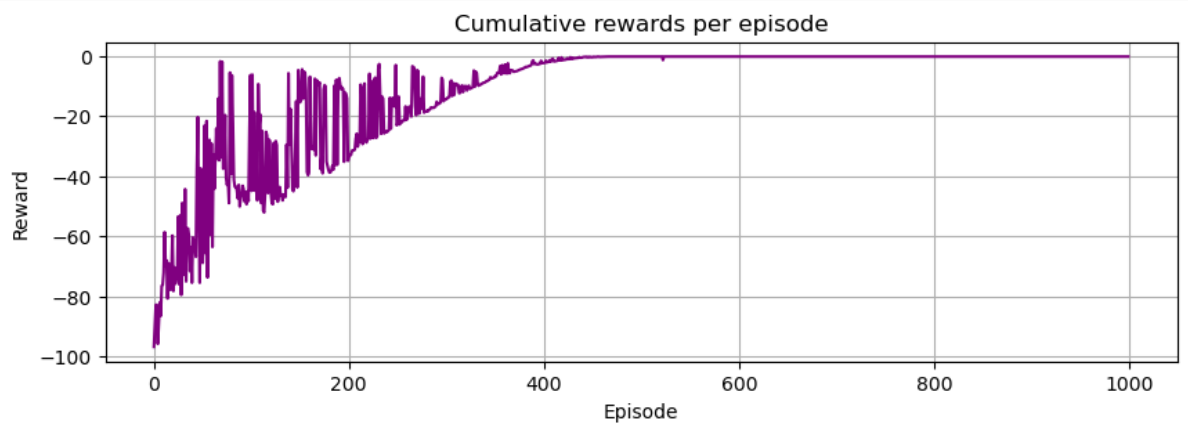
Central Agent

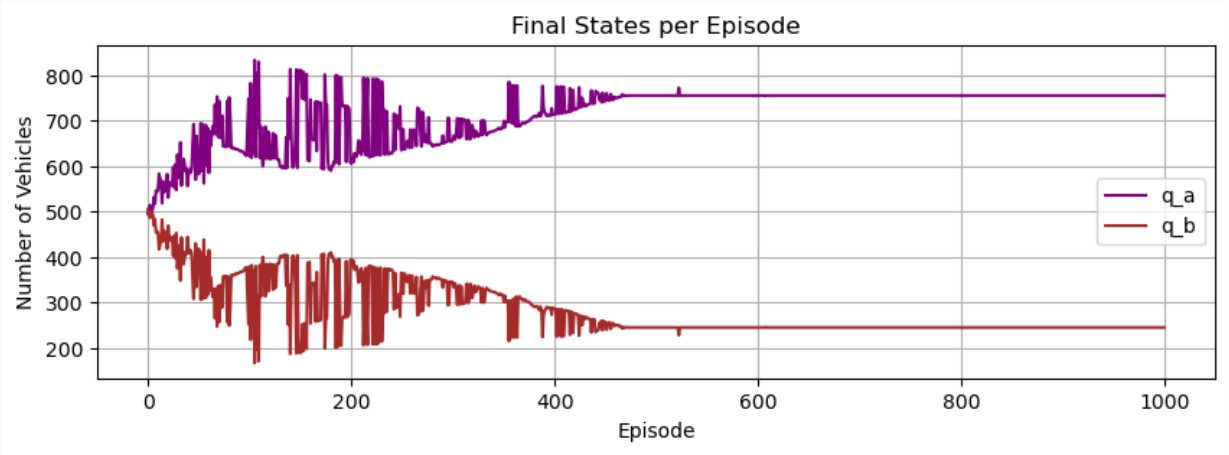
- Same as the SO version.
- Policy is a deterministic look-up table.
- Employs Q-Learning Algorithm, learns values of actions.
- A decaying epsilon for exploration.

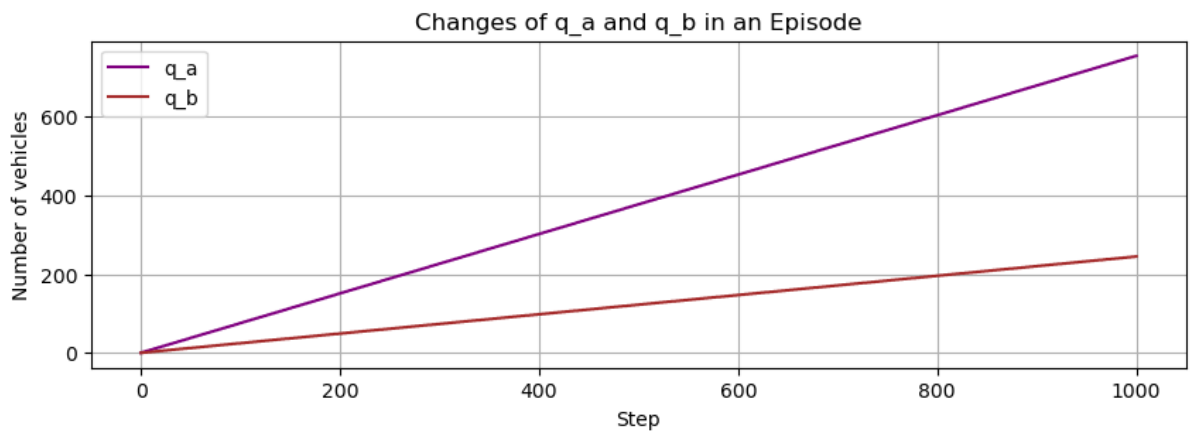
Dense Rewards

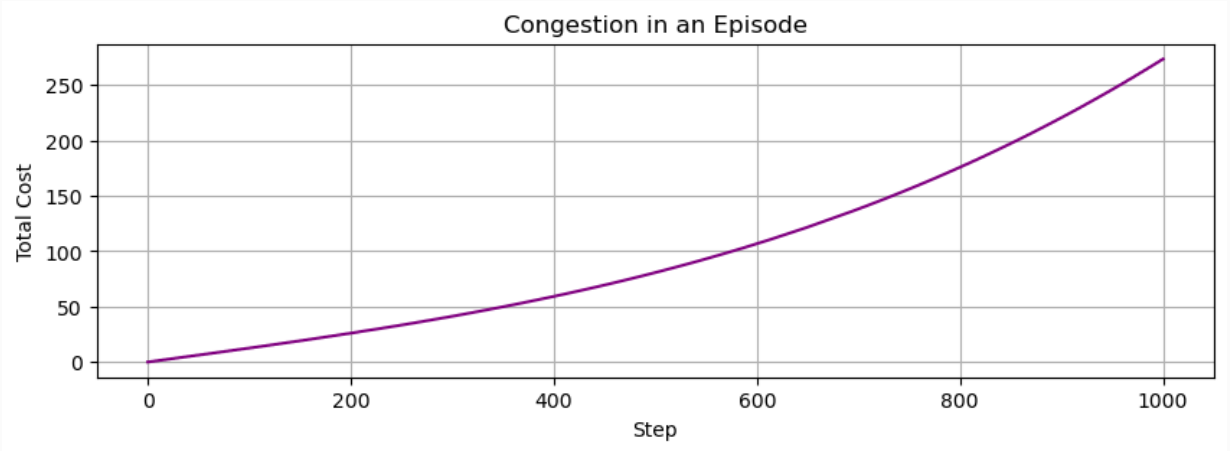
$$curr_Q = q_a + q_b$$
$$k1 = q_a + ((Q - curr_Q) * \frac{q_a}{curr_Q})$$
$$k2 = q_b + ((Q - curr_Q) * \frac{q_b}{curr_Q})$$
$$reward = -abs(t_a(k1) - t_b(k2)) * \frac{curr_Q}{Q}$$

USER EQUILIBRIUM (CENTRALIZED)









How is it in the multi-agent?

USER EQUILIBRIUM (MULTI-AGENT)

USER EQUILIBRIUM (MULTI-AGENT)

Traffic Environment

- Same as the SO version.
- Both roads are initially empty.
- Accepts two actions: Choosing a or b.

Discrete State Space

- Same as the SO version.
- Made of tuples: Current route, number of vehicles on each route (K, x, y)
- One transition for each agent: Start and after selection

USER EQUILIBRIUM (MULTI-AGENT)

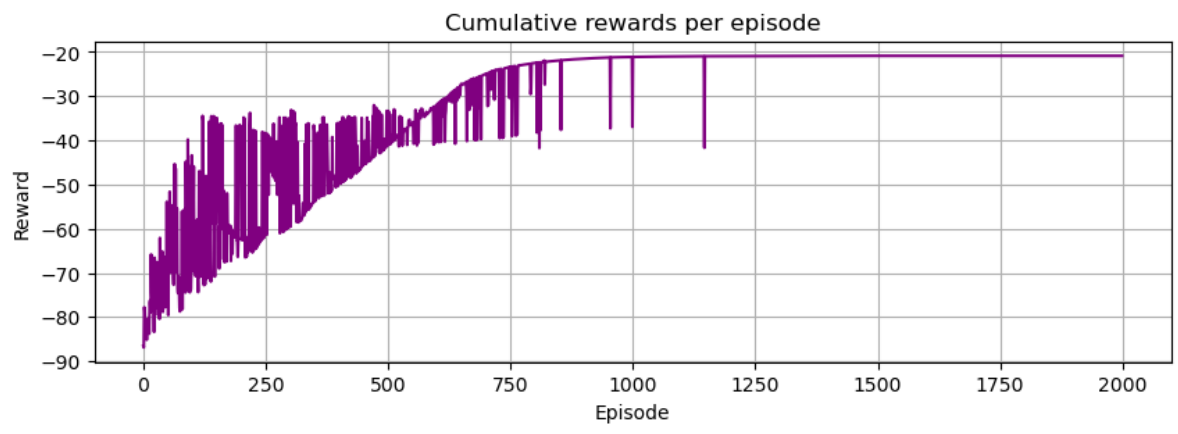
Each Vehicle is an Agent

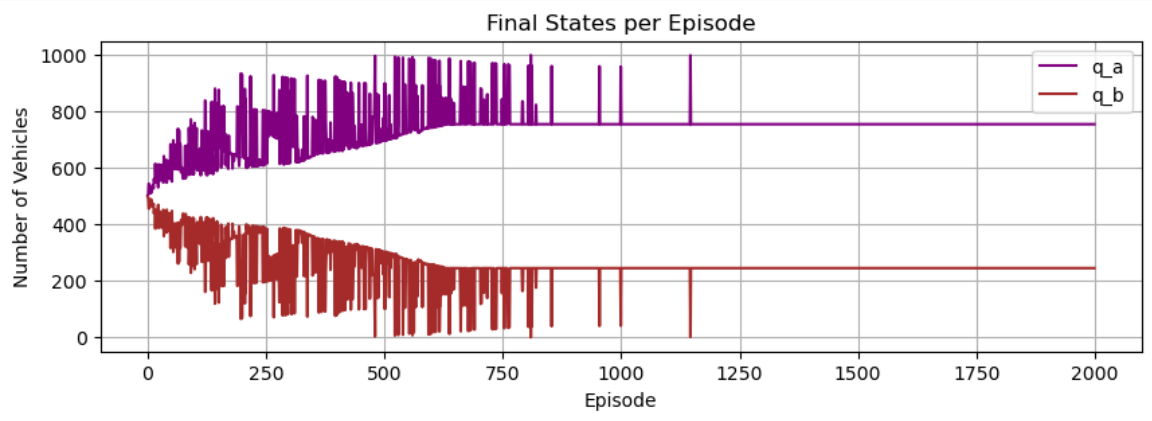
- Same as the SO version.
- Policy is a deterministic look-up table.
- Employs Q-Learning Algorithm, learns values of actions.
- A decaying epsilon for exploration.

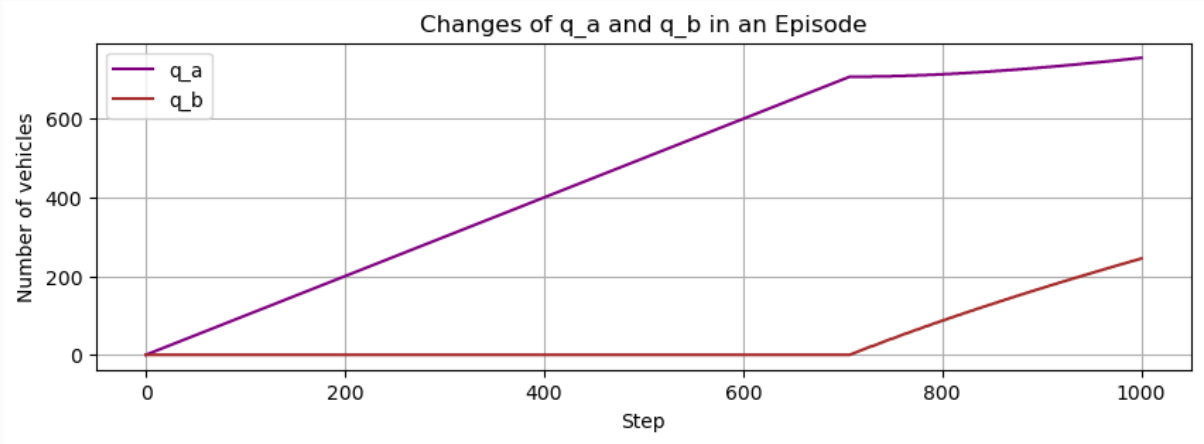
Dense Rewards

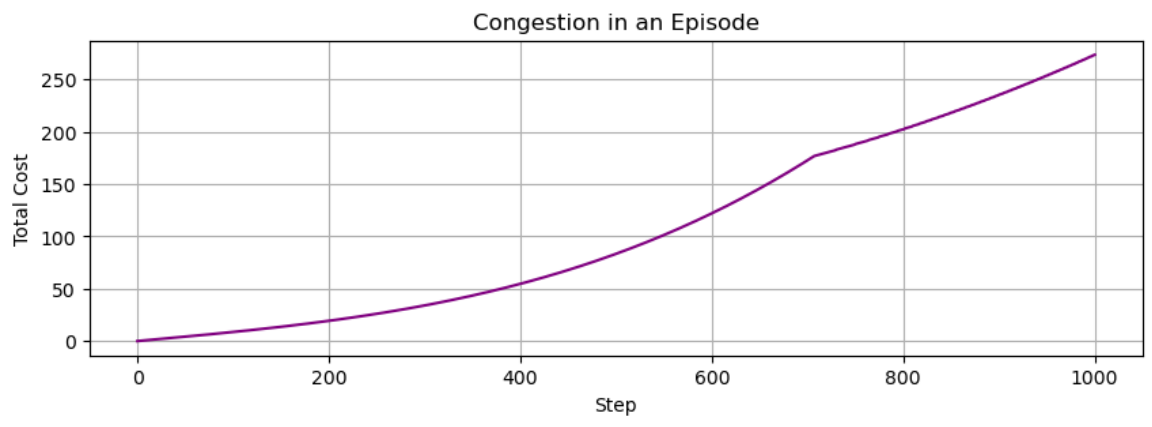
- Current cost difference (negative), scaled by the ratio of current number of vehicles and the total number of vehicles.
- Last vehicles are penalized more drastically.

USER EQUILIBRIUM (MULTI-AGENT)







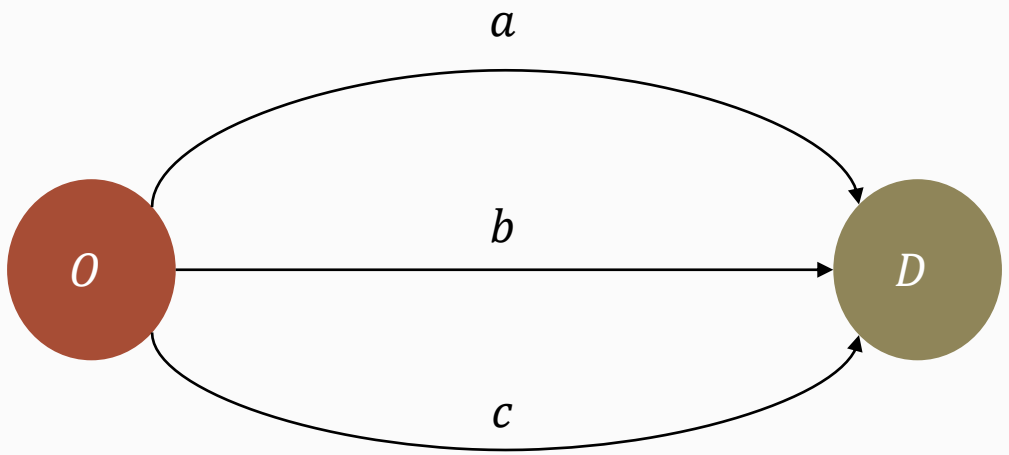


How was it in the central agent?

DL-BASED APPROACH WITH COMPLEXITIES

NEW PROBLEM

- A set of Q individual travelers want to reach from their origin O to their destination D.
- Everyday they choose between the **three** alternative routes: a, b, or c.
- The cost (travel time) for each route is given with the same BPR formula as before.
- Occasionally, using a $randomness$ parameter, **congestions are increased randomly**.



Route A: Capacity = 20, Free-flow travel time = 5

Route B: Capacity = 25, Free-flow travel time = 8

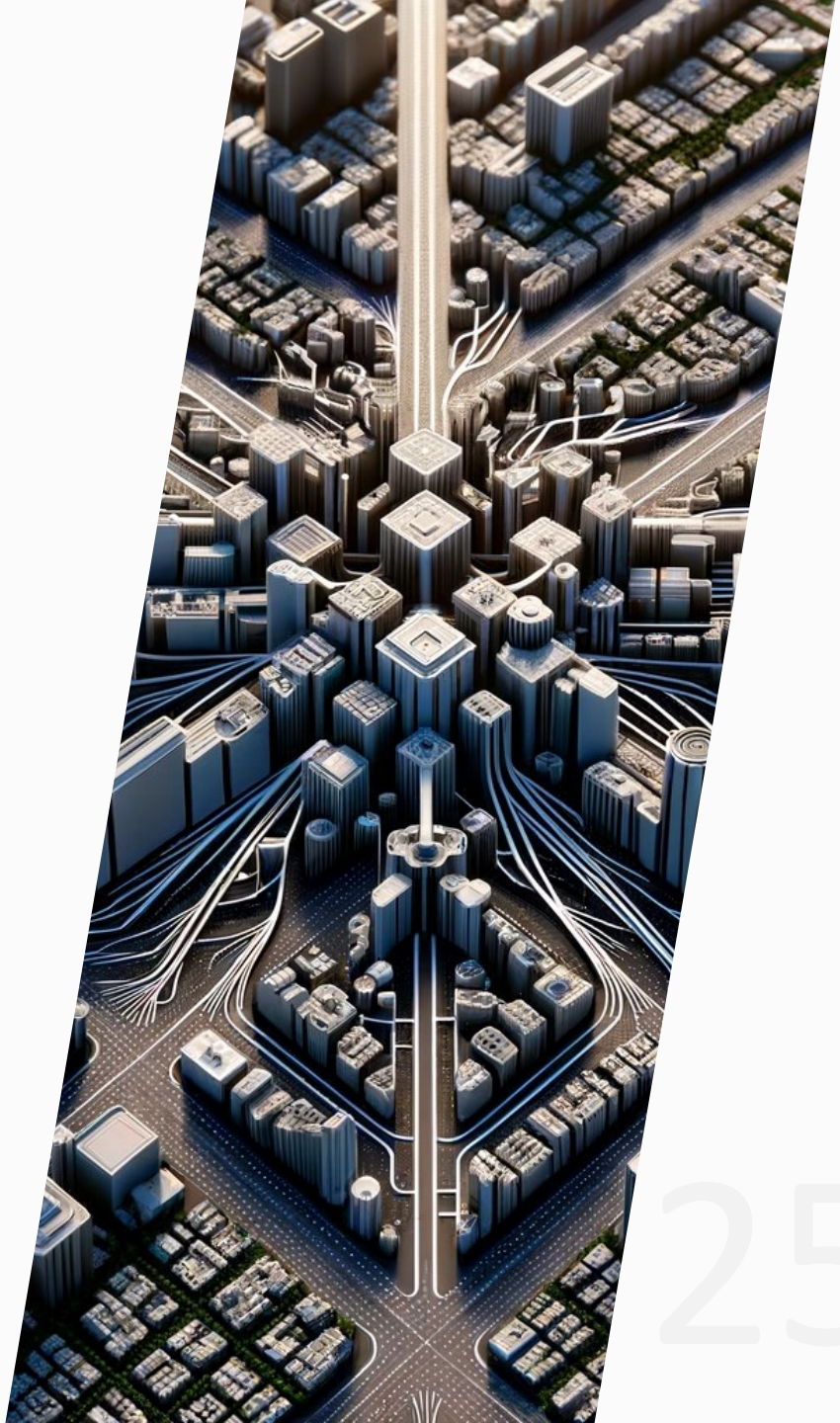
Route C: Capacity = 40, Free-flow travel time = 11

Traffic Environment

- Returns states as tuples: Number of cars in each route, and randomness (q_a, q_b, q_c, R)
- Accepts 3 sorts of actions: 0, 1, 2
- Reward: (Negative) Travel time on the selected route.
- Jackpot when travel times are equalized. (Disabled)

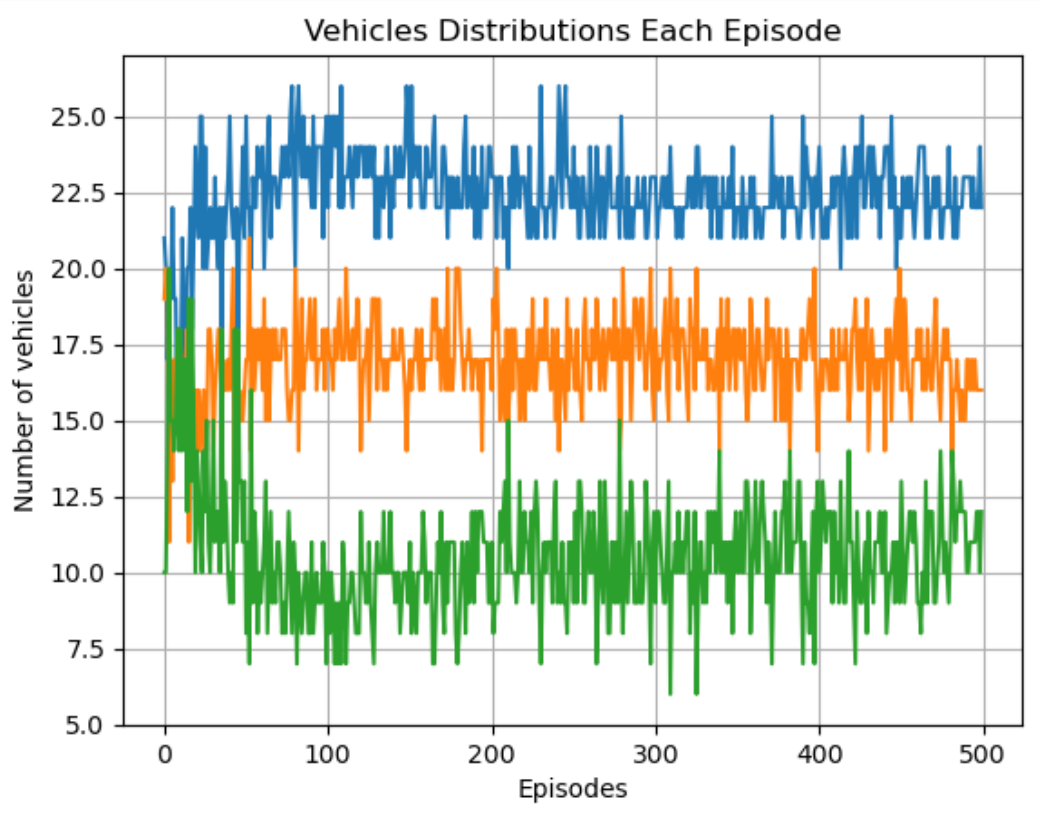
Central Agent

- For this implementation, the number of vehicles is set to 50.
- Learning employs a **Deep-Q-Network** with **Experience Replay**. Approximates a Q-Table.
- Actions are predicted by the DQN, which is a feedforward neural network with 2 hidden layers and 235 trainable parameters.

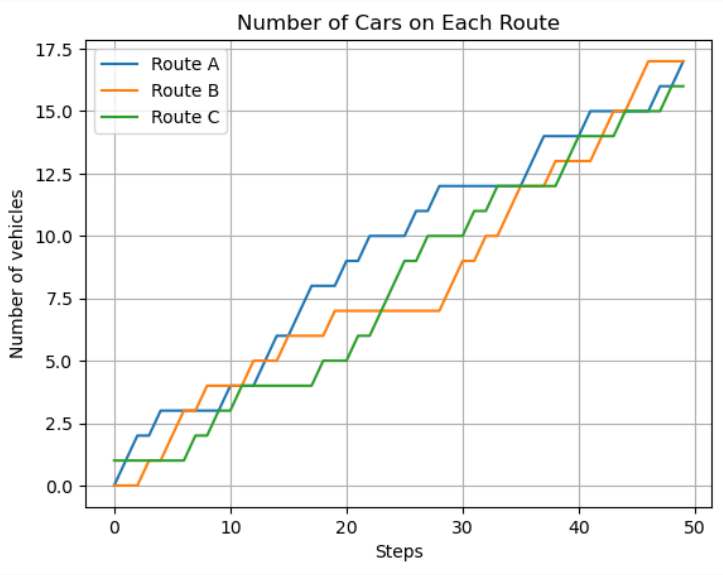


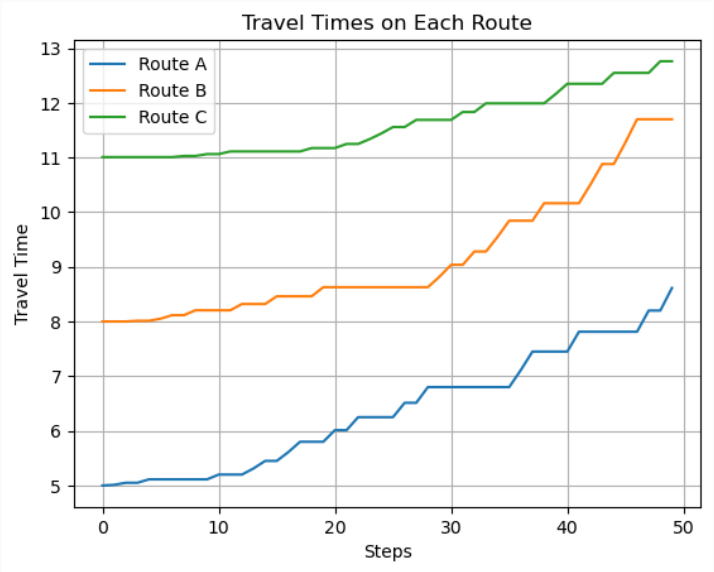
Training


Rewards Achieved Throughout Training
Episode reward
−440
−445
−450
−455
−460
−465
−470
−475
0
100
200
300
400
500
Episodes

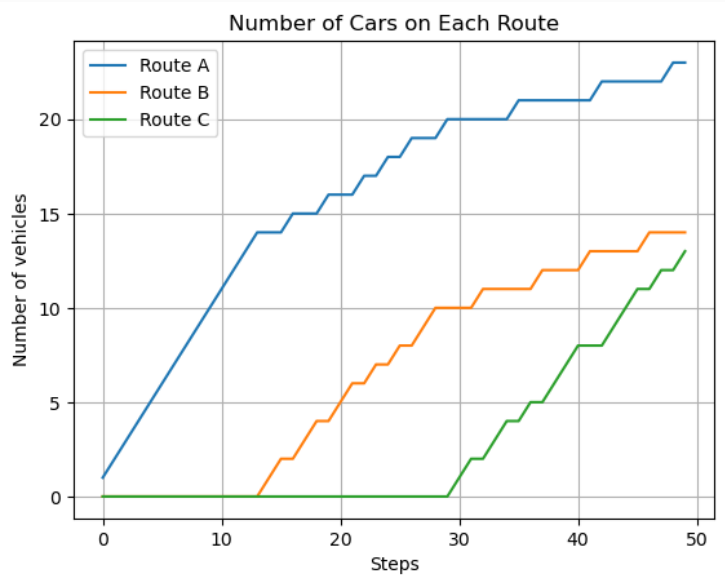

Vehicles Distributions Each Episode
Number of vehicles
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
0
100
200
300
400
500
Episodes

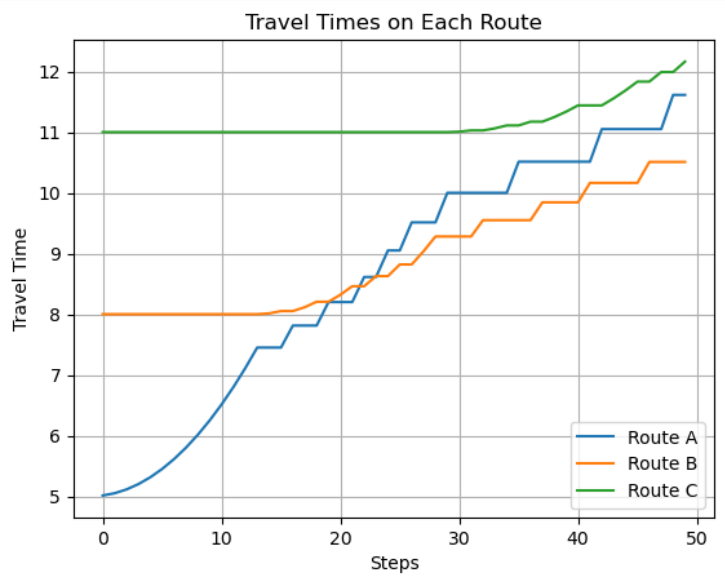
Random Agent



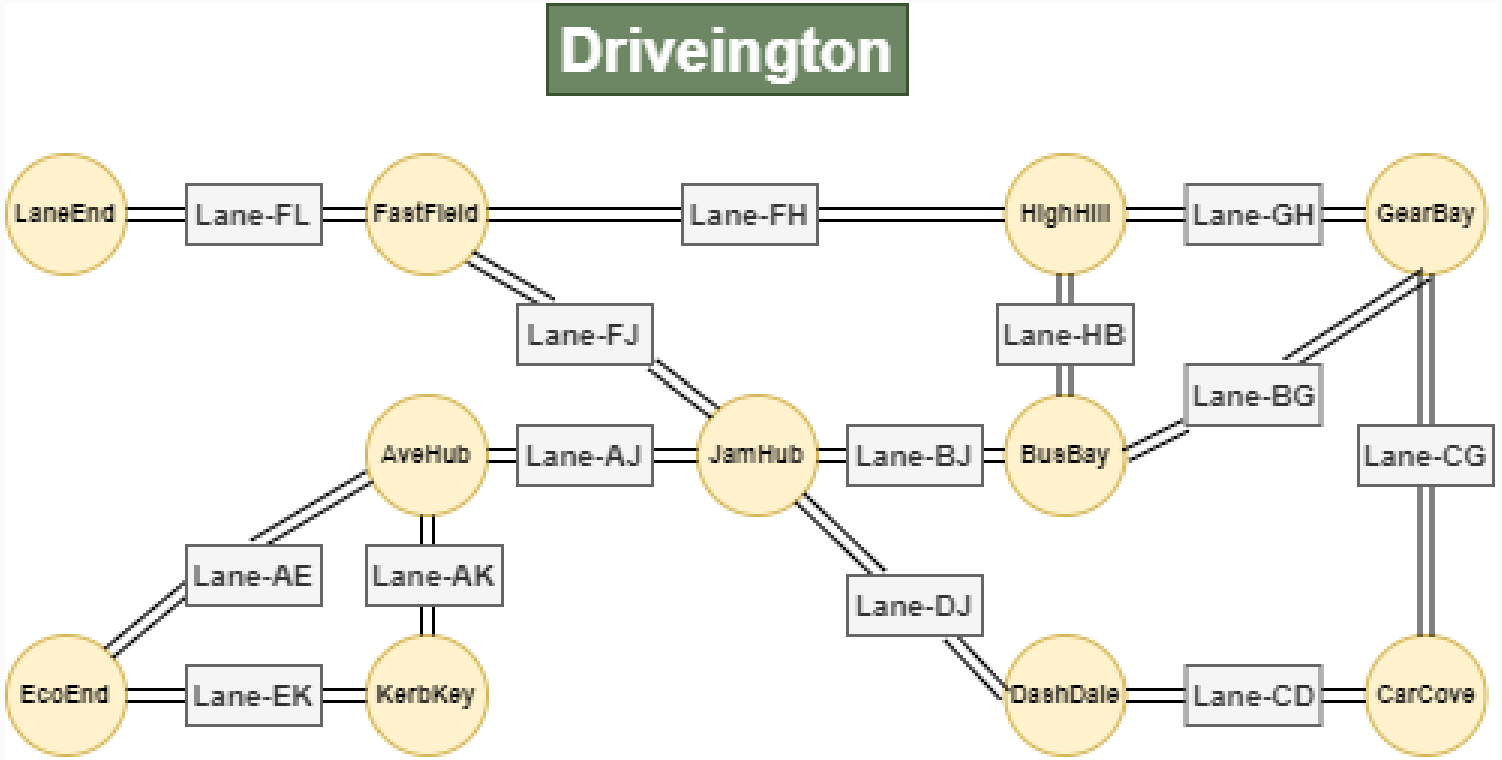


Trained Agent





PROBLEM EXTENSION




```
sect_names = ["AveHub","BusBay","CarCove","DashDale","EcoEnd","FastField","GearBay","HighHill","JamHub","KerbKey","LaneEnd"]
# How likely these sections should be origins and destinations?
sect_sampling_coeff = [(1, 9), (1, 9), (9, 1), (1, 1), (9, 1), (1, 1), (9, 1), (1, 1), (0, 10), (5, 5), (10, 0)]

# Road origins and destinations, free flow durations and capacities
roads = [["LaneEnd", "FastField", 2/60, 300], ["FastField", "HighHill", 8/60, 800], ["HighHill", "GearBay", 6/60, 500],\
         ["FastField", "JamHub", 13/60, 400], ["HighHill", "BusBay", 10/60, 500], ["GearBay", "BusBay", 15/60, 600], \
         ["GearBay", "CarCove", 25/60, 1000], ["AveHub", "JamHub", 3/60, 250], ["JamHub", "BusBay", 4/60, 250], \
         ["AveHub", "EcoEnd", 15/60, 600], ["AveHub", "KerbKey", 7/60, 300], ["JamHub", "DashDale", 9/60, 450], \
         ["EcoEnd", "KerbKey", 4/60, 250], ["DashDale", "CarCove", 3/60, 300]]
```

The **City** Has...

Sections

The city consists of sections with different properties.

Roads

City is equipped with two-way roads, with different lengths and capacities, connecting different sections.

Vehicles

Vehicles, in other words agents, start their travels at the same time from different origins, to reach to different sections.

Randomness

The city has a randomness factor, ignoring agent decisions for one turn, reflecting the random nature of cities in real life. (Disabled due complexity)

Every **Section** Has...

A Name and ID

Each section has a unique name and identifiers, corresponding to their initials.

A Set of Neighbors

Each section has neighbors connected by a two-way road.

An Origin Sampling Coefficient

While assigning routes for drivers, each section has their coefficients, reflecting how likely they are to be the origin.

A Destination Sampling Coefficient

Each section has different probabilities for being the destination of a given route. City center is a more popular destination than outer sections.

Every **Vehicle** Has...

A Vehicle Type

Each vehicle has a type, either a car or a bus. This type reflects its influence on the congestion.

An Origin & Destination Pair

Each vehicle starts their travel from a section, aiming to reach to another one as fast as possible.

Randomness

Each vehicle has its own randomness factor, reflects its probability of changing its destination in the mid-travel. (Disabled due complexity)

Location

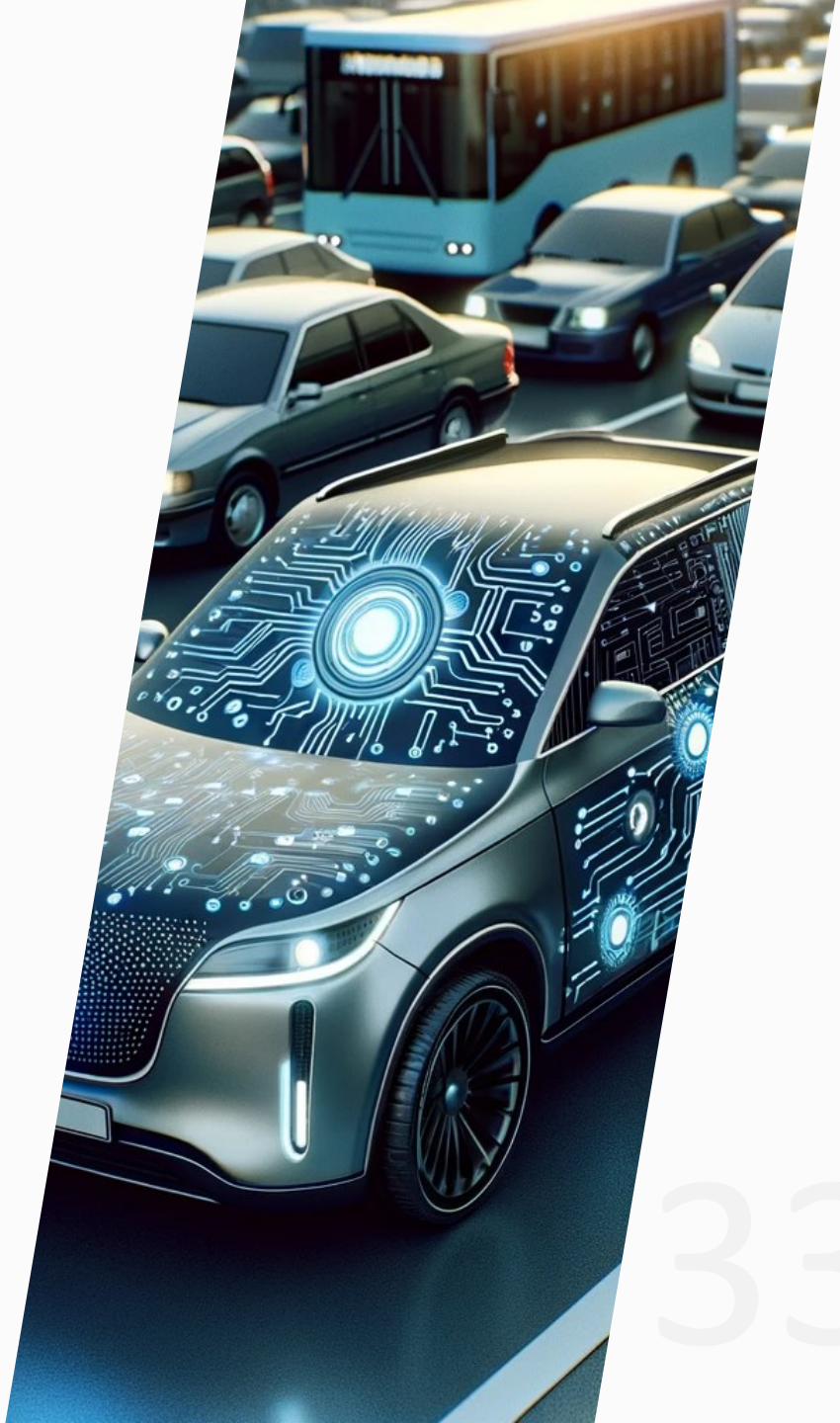
In each step of an episode, each vehicle is located in a section.

States and Rewards

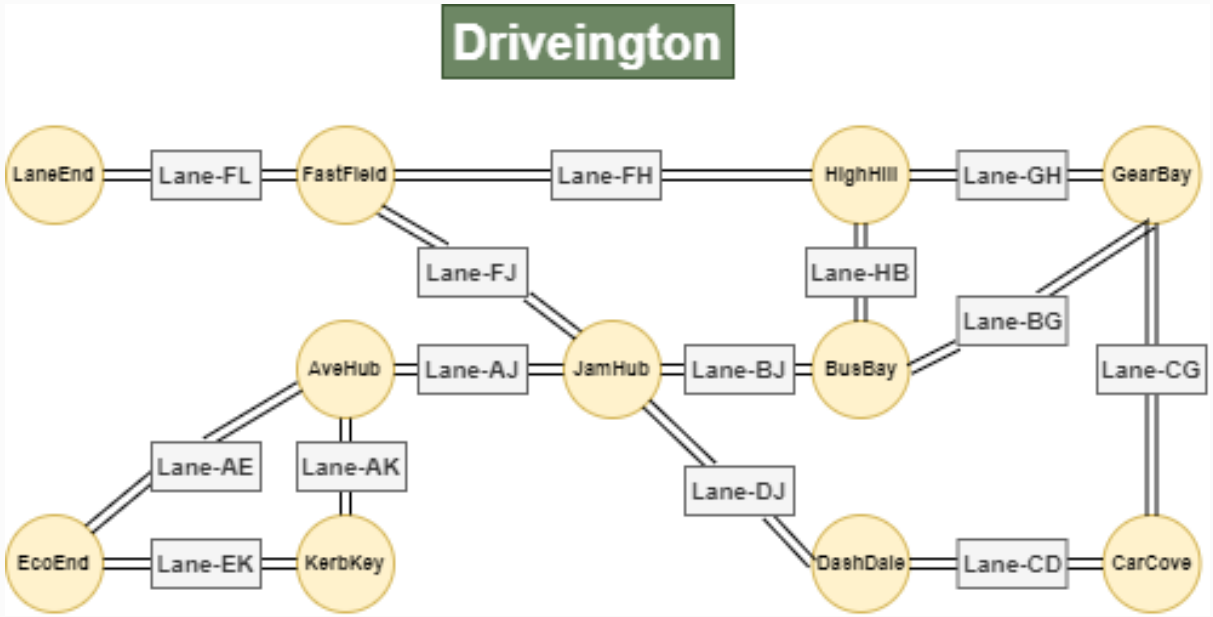
- The states in each step are tuples in the form **(origin, destination, a dictionary of possible roads and the number of vehicles using it)**
- Agents know the number of vehicles on roads, not congestions, so that they learn road characteristics.
- **Reward**: $-t_x$ if the travel continues, 0 if already arrived.

Agents

- 500 vehicles, ~%5 are busses.
- Deterministic policies, based on Q-function.
- Q-Learning based learning.

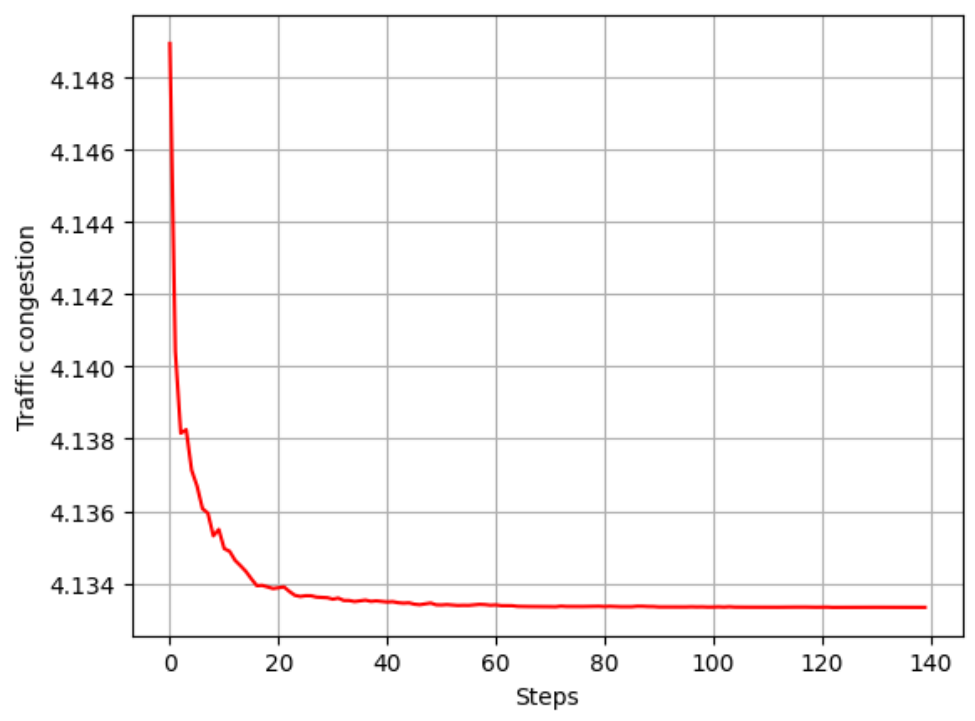


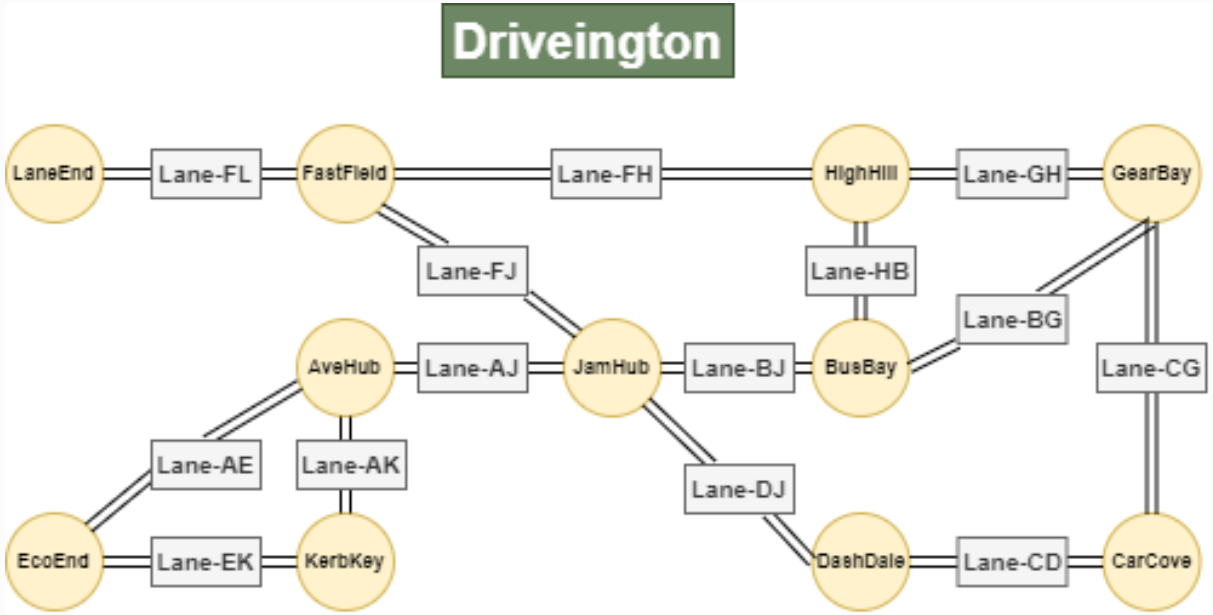
Population




```
car is now at KerbKey. It is going from KerbKey to AveHub
car is now at GearBay. It is going from GearBay to BusBay
car is now at LaneEnd. It is going from LaneEnd to BusBay
car is now at HighHill. It is going from HighHill to KerbKey
car is now at AveHub. It is going from AveHub to JamHub
car is now at LaneEnd. It is going from LaneEnd to FastField
car is now at LaneEnd. It is going from LaneEnd to JamHub
car is now at EcoEnd. It is going from EcoEnd to JamHub
car is now at GearBay. It is going from GearBay to BusBay
car is now at CarCove. It is going from CarCove to AveHub
```

Random Agents – 1 Episode

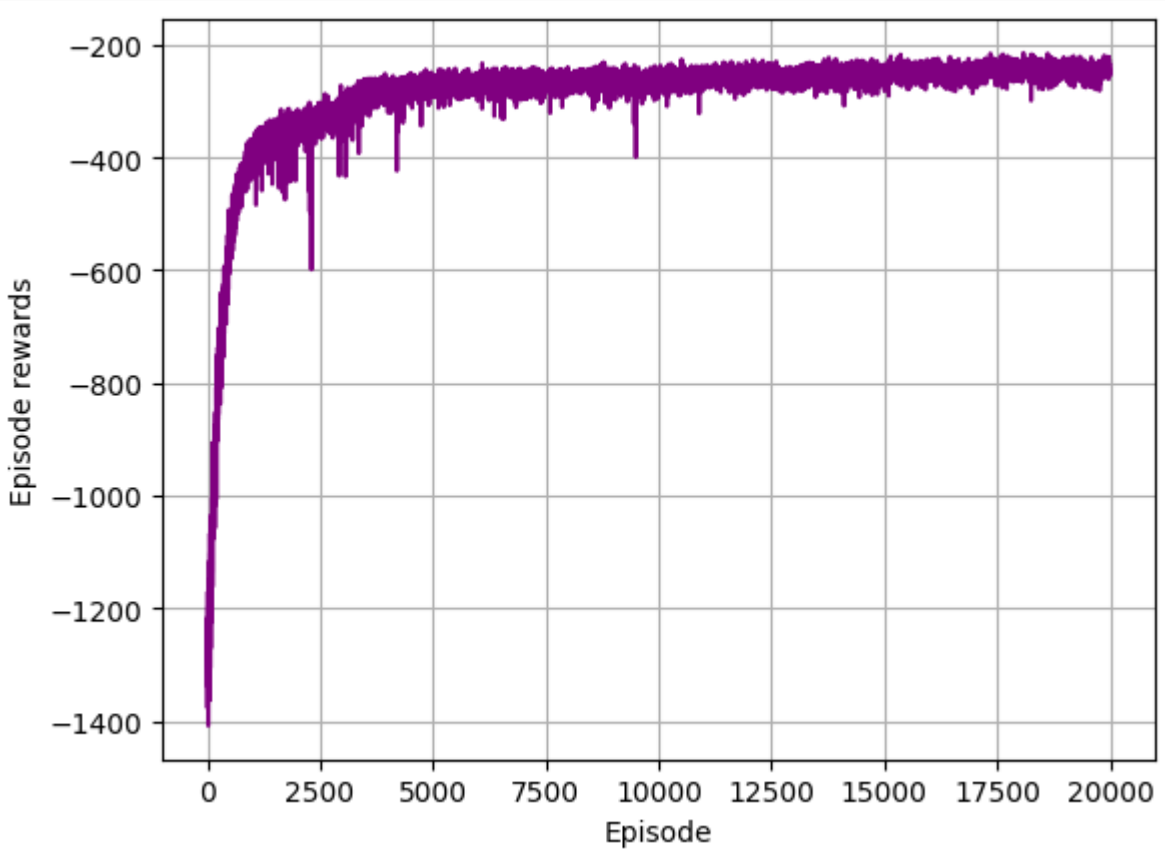


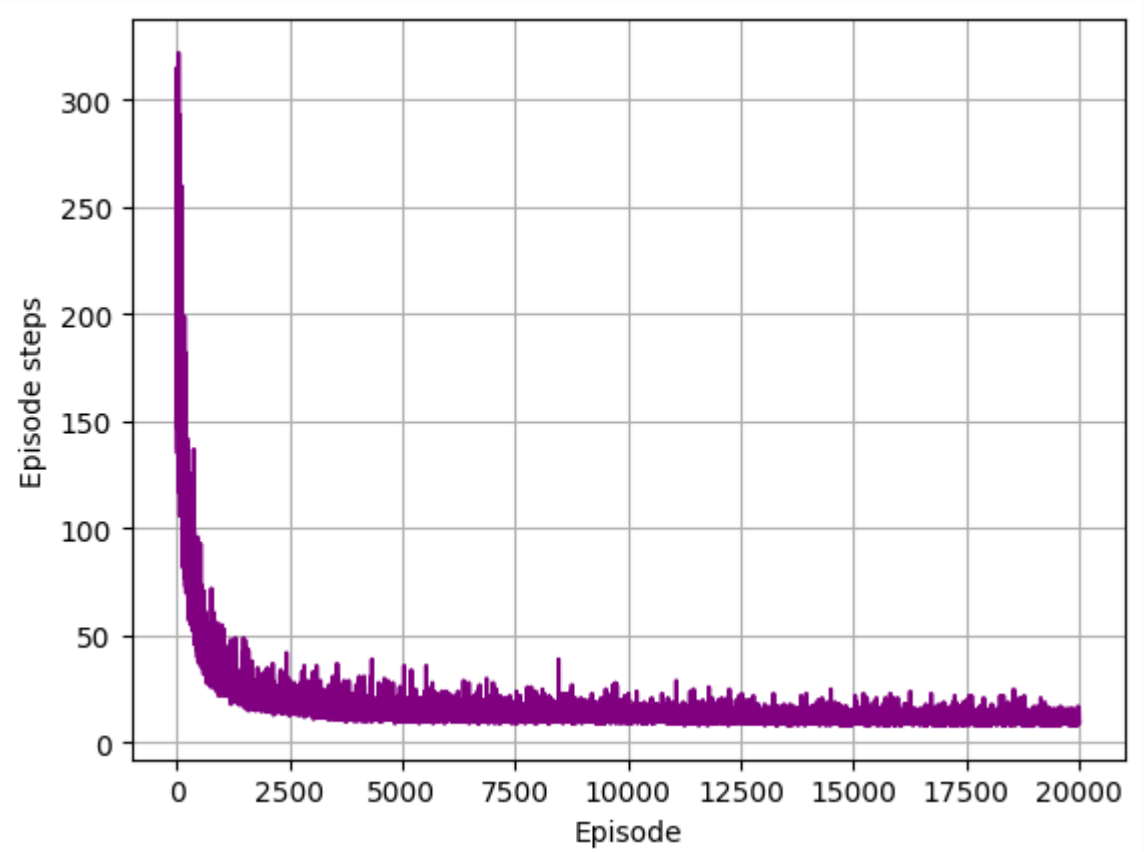


See Trained

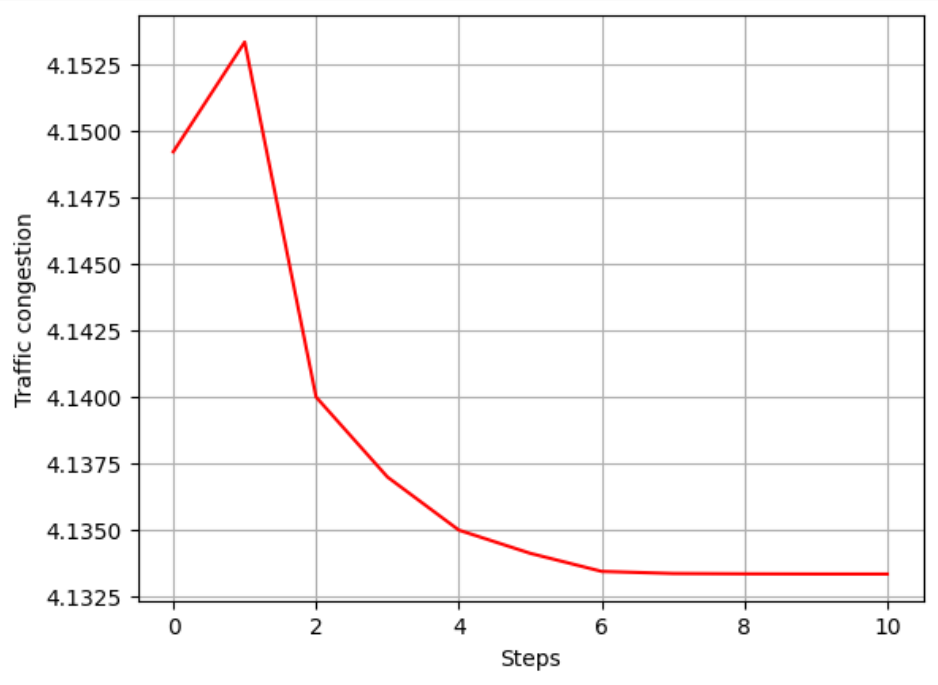
```
car #0 moving from KerbKey to AveHub followed (first 10):  ['KerbKey', 'EcoEnd', 'KerbKey', 'EcoEnd', 'KerbKey', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub']
car #1 moving from GearBay to BusBay followed (first 10):  ['GearBay', 'CarCove', 'GearBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay']
car #2 moving from LaneEnd to BusBay followed (first 10):  ['LaneEnd', 'FastField', 'LaneEnd', 'FastField', 'HighHill', 'GearBay', 'HighHill', 'GearBay', 'BusBay', 'BusBay']
car #3 moving from HighHill to KerbKey followed (first 10):  ['HighHill', 'FastField', 'HighHill', 'BusBay', 'GearBay', 'HighHill', 'FastField', 'LaneEnd', 'FastField', 'JamHub']
car #4 moving from AveHub to JamHub followed (first 10):  ['AveHub', 'KerbKey', 'AveHub', 'KerbKey', 'EcoEnd', 'KerbKey', 'EcoEnd', 'AveHub', 'JamHub', 'JamHub']
car #5 moving from LaneEnd to FastField followed (first 10):  ['LaneEnd', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField']
car #6 moving from LaneEnd to JamHub followed (first 10):  ['LaneEnd', 'FastField', 'LaneEnd', 'FastField', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub']
car #7 moving from EcoEnd to JamHub followed (first 10):  ['EcoEnd', 'AveHub', 'KerbKey', 'EcoEnd', 'AveHub', 'EcoEnd', 'AveHub', 'JamHub', 'JamHub', 'JamHub']
car #8 moving from GearBay to BusBay followed (first 10):  ['GearBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay']
car #9 moving from CarCove to AveHub followed (first 10):  ['CarCove', 'GearBay', 'CarCove', 'GearBay', 'HighHill', 'FastField', 'LaneEnd', 'FastField', 'JamHub', 'DashDale']
```

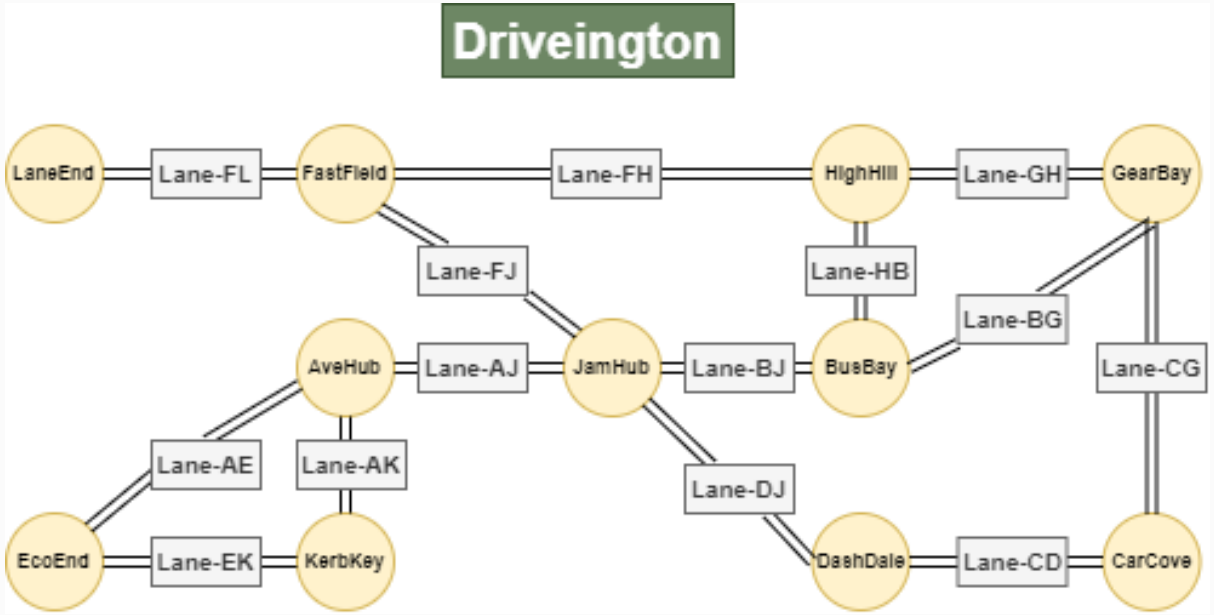
Training


Episode rewards
−200
−400
−600
−800
−1000
−1200
−1400
0
2500
5000
7500
10000
12500
15000
17500
20000
Episode


Episode steps
300
250
200
150
100
50
0
0
2500
5000
7500
10000
12500
15000
17500
20000
Episode

Trained Agents – 1 Episode





See Random

```
car #0 moving from KerbKey to AveHub followed (first 10):  ['KerbKey', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub']
car #1 moving from GearBay to BusBay followed (first 10):  ['GearBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay']
car #2 moving from LaneEnd to BusBay followed (first 10):  ['LaneEnd', 'FastField', 'JamHub', 'AveHub', 'JamHub', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay']
car #3 moving from HighHill to KerbKey followed (first 10):  ['HighHill', 'FastField', 'LaneEnd', 'FastField', 'JamHub', 'AveHub', 'KerbKey', 'KerbKey', 'KerbKey', 'KerbKey', 'KerbKey', 'KerbKey']
car #4 moving from AveHub to JamHub followed (first 10):  ['AveHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub']
car #5 moving from LaneEnd to FastField followed (first 10):  ['LaneEnd', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField', 'FastField']
car #6 moving from LaneEnd to JamHub followed (first 10):  ['LaneEnd', 'FastField', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub']
car #7 moving from EcoEnd to JamHub followed (first 10):  ['EcoEnd', 'AveHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub', 'JamHub']
car #8 moving from GearBay to BusBay followed (first 10):  ['GearBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay', 'BusBay']
car #9 moving from CarCove to AveHub followed (first 10):  ['CarCove', 'DashDale', 'CarCove', 'GearBay', 'CarCove', 'GearBay', 'BusBay', 'JamHub', 'AveHub', 'AveHub', 'AveHub', 'AveHub']
```

JAGIELLONIAN UNIVERSITY
IN KRAKÓW

Thank You

for your attention