Reinforcement learning in urban mobility

Anastasia Psarou

2023-08-01

Problem

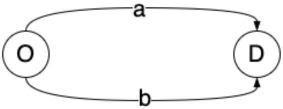

Figure: Environment.

Objective:

- Individual travelers want to reach their destination D from their origin O.

Routes:

- a.
- b.

Cost function:

- $t_a(q_a) = t_a^0 \left(1 + \left(\frac{q_a}{Q_a}\right)^2\right)$

Flow constraints:

- $q_a + q_b = 1000$
- $q_a, q_b > 0$

Parameterization

Given the following parameterization:

- Total flow capacity $Q = 1000$ veh/h.
- Free flow speed on route a is $t_a^0 = 5$ min.
- Free flow speed on route b is $t_b^0 = 15$ min.
- Capacity on route a is $Q_a = 500$ veh/h.
- Capacity on route b is $Q_b = 800$ veh/h.

User equilibrium

User equilibrium:

- All routes chosen by travelers have equal and minimal travel time.
- Each traveler seeks to minimize their own travel time.
- No traveler can unilaterally improve travel time by switching routes.

User equilibrium equation:

$$t_a(q_a) = t_b(q_b) \tag{1}$$

User equilibrium

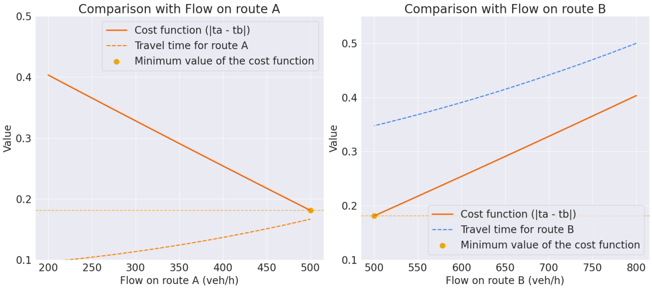

Figure: User equilibrium illustration

Explanation:

- The orange straight line represents the reward/cost function associated with route flows.
- Dotted lines show travel times for routes A and B, corresponding to their respective flows.

Centralized solution: user equilibrium

Figure: Convergence of reward function.

Centralized solution:

- Central controller trains and makes decisions for all agents.

Decision-making Process:

- Systematically evaluates all combinations of q_a and q_b.
- Calculates reward: $|t_a(q_a) - t_b(q_b)|$.
- Updates policy based on state values.

Centralized solution: user equilibrium

Figure: Convergence of reward function.

Training and convergence:

- The centralized solution employs an ε-greedy, on-policy algorithm closely related to SARSA.
- Algorithm iterates over episodes to determine the best actions.

Time complexity:

- $O(Q_a)$ per episode.

Decentralized solution: user equilibrium

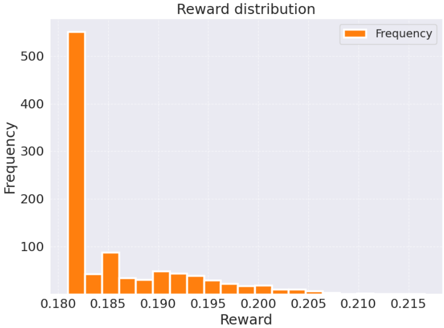

Figure: Decentralized solution plot

MARL (multi-agent reinforcement learning) solution:

- Each agent selects an action randomly, ensuring constraint satisfaction (Qa < qa, Qb < qb).

Training and reward:

- Episode-based training with a specified number of episodes.
- Achieves a minimum reward of 0.181 after 1000 episodes.

Decentralized complexity:

- O(Q) per episode.

Comparison of centralized and decentralized approaches

- Learning speed: Decentralized approach learns faster due to parallel learning and independent decision-making.
- Optimization: Centralized approach performs better as it can leverage global traffic information for efficient allocation.

	q_a	q_b	**Minimum Reward**
Best Case	500	500	0.181
Centralised	495	505	0.185
Decentralised	500	500	0.181

Table: Comparison of the results of user equilibrium for 10 episodes.

System optimum

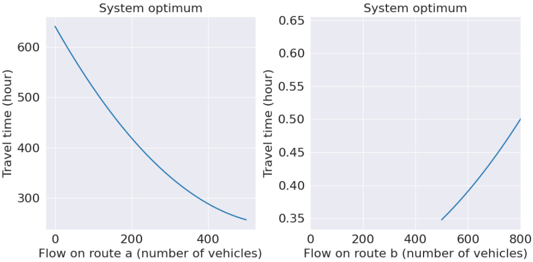

Figure: System optimum illustration.

System optimum:

- Minimizes the total system travel time.

System optimum equation:

$$t_a(q_a) \cdot q_a + t_b(q_b) \cdot q_b \tag{2}$$

Centralized solution: system optimum

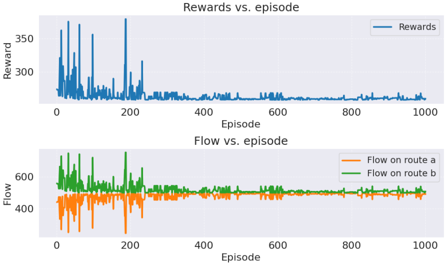

Figure: Convergence of reward function

Implementation:

- Central controller trains and makes decisions for all agents.

Reward convergence:

- Reward gradually decreases during training, indicating system performance improvement.
- Distribution of qa and qb values evolves, reflecting agent's decision-making.

Centralized solution: system optimum

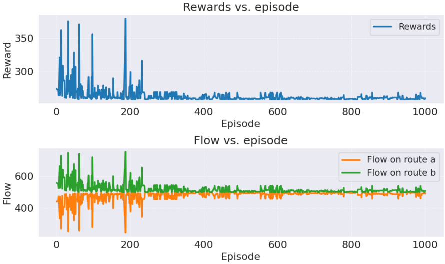

Figure: Convergence of reward function (system optimum)

Choose action function:

- Selects random action with probability epsilon, otherwise exploits current policy.
- Ensures action a and action b adhere to a minimum value of 1 and sum equals Q.

Time complexity:

- $O(Q_a)$ per episode.

Decentralized solution: system optimum

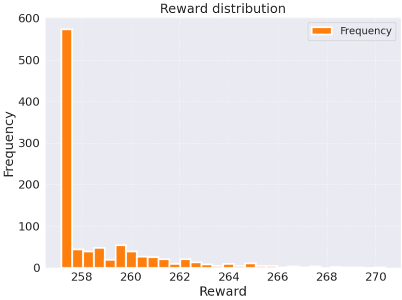

Figure: Frequency distribution of reward values

MARL (multi-agent reinforcement learning) solution:

- Each agent selects an action randomly, ensuring constraint satisfaction (Qa $<$ qa, Qb $<$ qb).

Reward computation:

- Computation of qa and qb values for minimum attainable reward.

complexity:

- O(Q) complexity per episode.
- O(episodes * Q) total complexity.