Traffic Signal Control Adaptation using Deep Q-Learning with Experience Replay

Presented by: Walid Shaker - Siba Issa

Supervised by: Prof. S. M. Ahsan Kazmi - TA: Mohamed Almadfaa

November 20, 2022

Presentation Outline

Introduction

Introduction

Traffic congestion has always been a major issue that disrupts our lives, make us late for work, and harms the environment due to the amount of fuel consumed. Despite the massive studies conducted on these issues, traffic monitoring, control, and congestion alleviation approaches continue to be popular research topics.

Literature Review

Literature Review

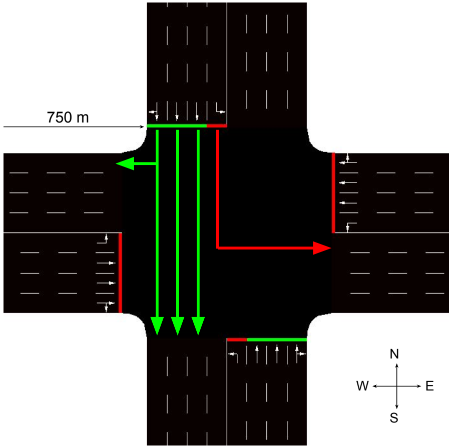
When we searched the literature for solutions to traffic congestion issues, we found two fundamental classifications:

1. Based on the features considered for traffic light control (human-crafted features & machine-crafted features)
2. The used approach (fixed-time & adaptive))

System Modeling and Problem Formalization

System Modeling and Problem Formalization

- Environment
- Traffic light system (agent)

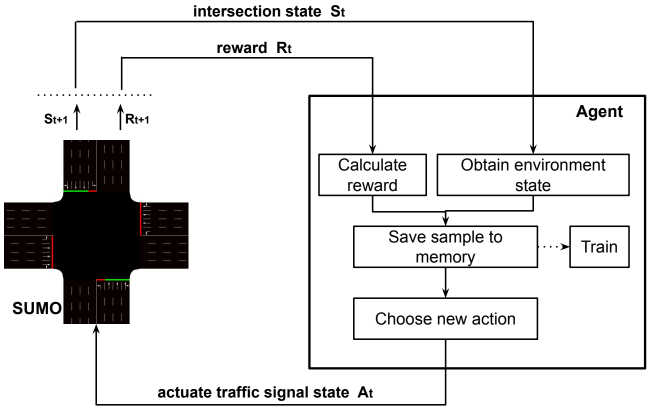


System Modeling and Problem Formalization

We can define the addressed problem as follows: given a state of intersection, and in order to maximize the intersection's traffic efficiency, which traffic light stage the agent has to select from a fixed predefined set of actions.

System Modeling and Problem Formalization

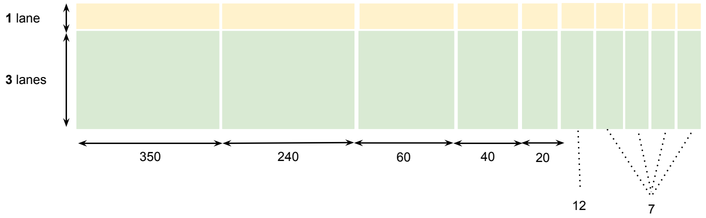
- sumo step
- agentstep
- S_t, A_t, R_t, S_{t+1}



System Modeling and Problem Formalization

Environment State

- Road discretization (1,0)
- 20 cells per road, 80 cells in total

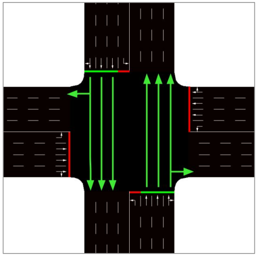


System Modeling and Problem Formalization

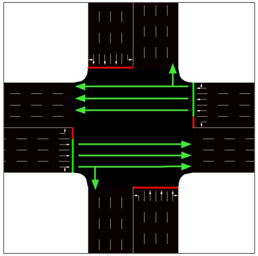
Agent Action Set

$$A = \{NS, EW, NSL, EWL\}$$

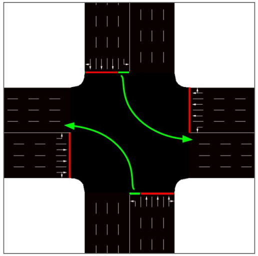
NS



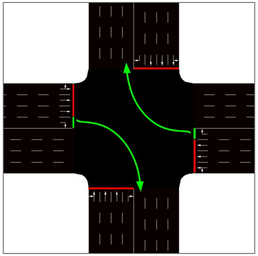
EW



NSL



EWL



System Modeling and Problem Formalization

Agent Action Set

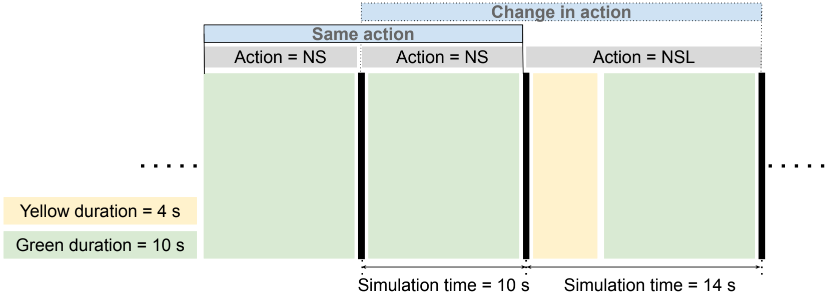


Figure: Possible simulation time of chosen actions.

System Modeling and Problem Formalization

Reward Function

In our problem, our goal is to reduce vehicle waiting time at traffic lights, but we want to cast it as a maximization problem, so our goal is to increase traffic flow through the intersection over time.

Literature reward function:

$$r_t = twt_{t-1} - twt_t \tag{1}$$

$$twt_t = \sum_{veh=1}^{n} wt_{veh,t} \tag{2}$$

System Modeling and Problem Formalization

Reward Function

Aternative reward function

$$r_t = atwt_{t-1} - atwt_t \tag{3}$$

$$atwt_t = \sum_{veh=1}^{n} awt_{veh,t} \tag{4}$$

Where $atwt_t$ represents the accumulated waiting time. By using this reward function we can guarantee that when the vehicle departs but could not cross the intersection, the value of $atwt_t$ does not reset (unlike the value of twt_t), avoiding the misleading case associated with the literature reward function when a long queue appears at the intersection.

System Modeling and Problem Formalization

Agent Goal

The main goal in our case is to increase traffic flow through the intersection by reducing the vehicles waiting time. The agent must choose the appropriate action, which will be in accordance with action policy π^*.

Expected cumulative reward:

$$Q_\pi(s,a) = \mathbb{E}\{R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...|S_t = s, A_t = a, \pi\} \tag{5}$$

$$= \mathbb{E}\{\sum_{k=0}^{\infty} \gamma^k R_{t+k}|S_t = s, A_t = a, \pi\}$$

Such an optimal policy π^* can be given by

$$\pi^* = \underset{\pi}{\operatorname{argmax}} Q_\pi(s,a) \quad for \quad all \quad s \in S, a \in A \tag{6}$$

Deep Q-Learning Algorithm

Deep Q-Learning Algorithm

If the optimal Q-values $Q^*(s, a)$ for all pairs of state-action are known, the agent can find the optimal action policy π^* by basically picking the action a that results in the optimal Q-values $Q^*(s, a)$ for a state s. The optimal Q-values $Q^*(s, a)$ can be calculated from Bellman optimality recursive equation.

$$\begin{aligned} Q^*(s, a) = \mathbb{E}\{R_t + \gamma \max_{a'} Q^*(S_{t+1}, a')|S_t = s, A_t = a\} \\ for \quad all \quad s \in S, a \in A \end{aligned} \tag{7}$$

- large state space
- dynamic environment (transition probability ?)

Deep Q-Learning Algorithm

Why Q-Learning? model-free reinforcement learning approach where there is no need for these transition probabilities. The action value function $Q\left(s_t, a_t\right)$ of picking action a_t at state s_t is defined as follows.

$$Q(S_t, a_t) = Q(S_t, a_t) + \alpha(r_{t+1} + \gamma \cdot \max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)) \tag{8}$$

Another modified version is used to calculate the Q-Learning function as follows.

$$Q(S_t, a_t) = r_{t+1} + \gamma \cdot \max_A Q'(s_{t+1}, a_{t+1}) \tag{9}$$

Due to the large state space, a deep neural network (DNN) is used to approximate the Q-learning function that results in $Q(s, a) \approx Q^*(s, a)$.

Deep Q-Learning Algorithm

Deep Neural Network Architecture

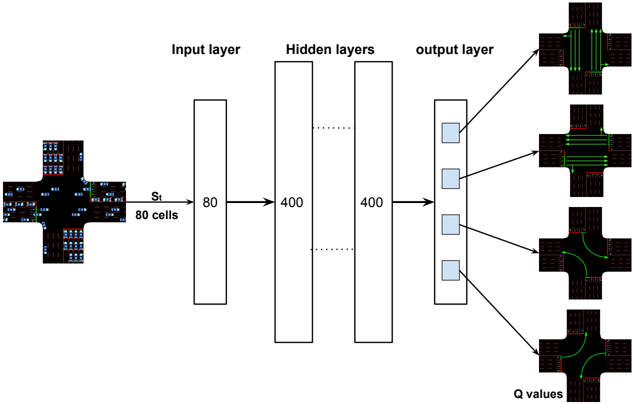
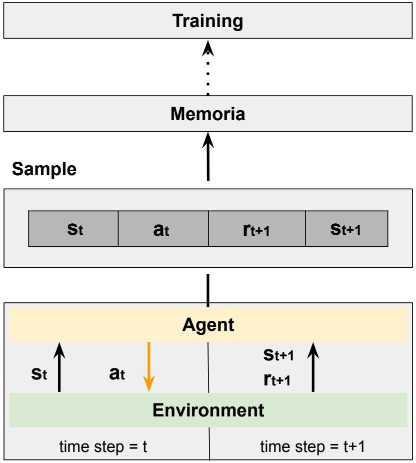


Figure: Deep neural network architecture.

Deep Q-Learning Algorithm

Experience Replay

- Improve model performance
- randomized batch of samples
- eliminate correlations
- $E_t = \{S_t, A_t, R_{t+1}, S_{t+1}\}$
- $M = \{E_t, E_{t+1}, E_{t+2}, \cdots\}$
- Memory size $= 5000$
- Batch size $= 100$

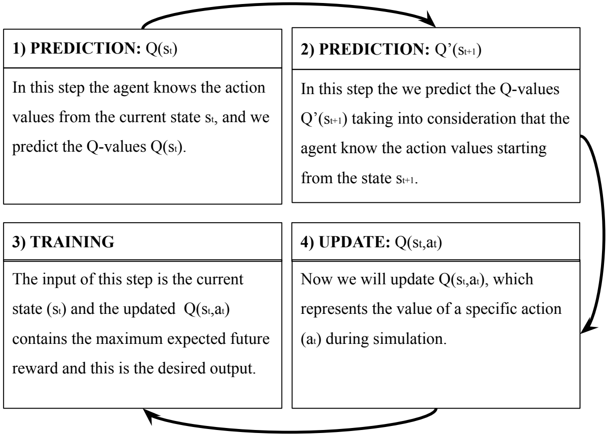


Deep Q-Learning Algorithm

Training Process

During the update of $Q(s, a)$, the greedy method is adopted to choose the action that maximizes the predicted action value function $Q'(s_{t+1}, a_{t+1})$.

While acting, the $\epsilon - greedy$ policy, which performs the exploration with probability ϵ and chooses the exploitation with probability $1 - \epsilon$.
$\epsilon_n = 1 - \frac{n}{N}$



1) PREDICTION: Q(s$_t$)

In this step the agent knows the action values from the current state s$_t$, and we predict the Q-values Q(s$_t$).

2) PREDICTION: Q'(s$_{t+1}$)

In this step the we predict the Q-values Q'(s$_{t+1}$) taking into consideration that the agent know the action values starting from the state s$_{t+1}$.

3) TRAINING

The input of this step is the current state (s$_t$) and the updated Q(s$_t$,a$_t$) contains the maximum expected future reward and this is the desired output.

4) UPDATE: Q(s$_t$,a$_t$)

Now we will update Q(s$_t$,a$_t$), which represents the value of a specific action (a$_t$) during simulation.

Simulation Settings

Simulation Settings

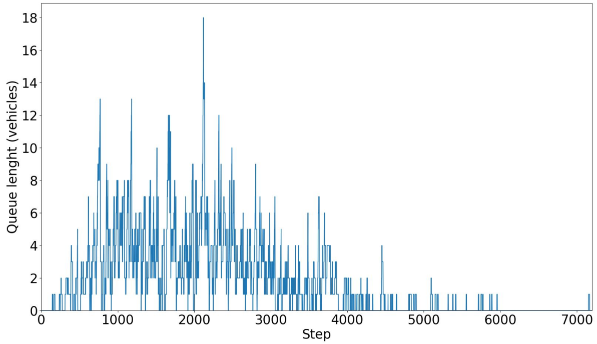
Training Setup

1. **Environment:** SUMO simulator provides a time frequency of 1 second per step.
2. **Training Time:** Duration for each episode to 2 hours $2 \times 60 \times 60 = 7200$ steps.
3. **Episode numbers:** 100 episode of 2 hours each, which is more than eight days of continuous traffic.
4. **High-graphics laptop**
5. **Training time** 3.5 hours.

Deep Q-Learning Algorithm

Experience Replay

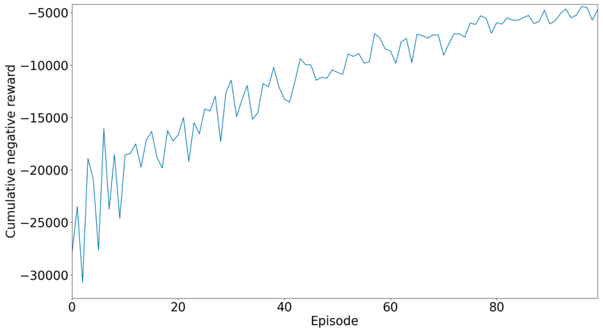
- High traffic scenario: 4000 cars reach the intersection.
- Low traffic scenario: 600 cars arrive at the intersection.
- North-South (NS) traffic scenario: 2000 cars arriving at the intersection.
- East-West (EW) West traffic scenario: 2000 cars arriving at the intersection.



Simulation Results

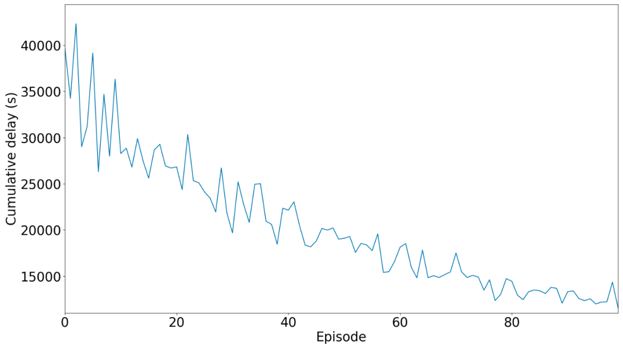
Simulation Results

The agent performance is examined using the training's reward trend followed by some metrics such as the average queue length (number of vehicles) and the cumulative delay time for all vehicles at the intersection.



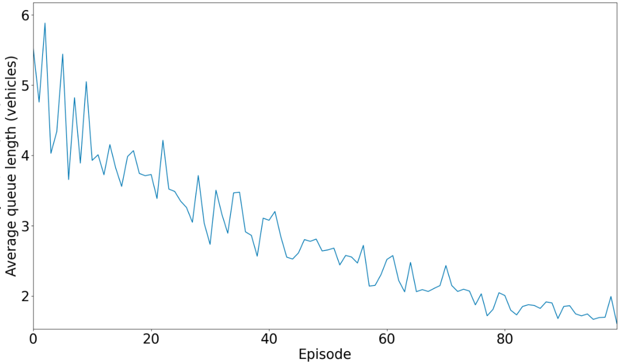
Simulation Results

The total accumulative waiting time (delay) was measured during the training using the alternative reward function which accumulates each waiting time for the vehicle until it passes the intersection.



Simulation Results

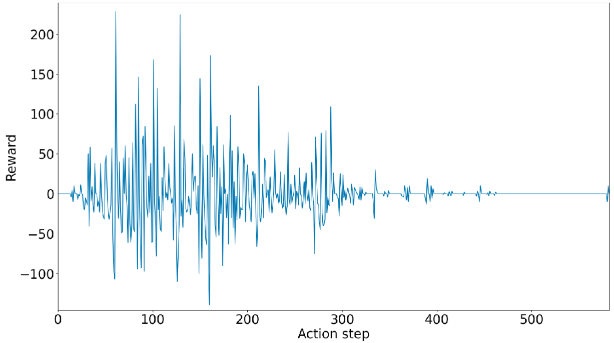
The corresponding metric for the cumulative waiting time is the queue length, which indicates the average number of cars queued per step in each episode.



Simulation Results

Model testing is performed and the associated reward to the actions taken at the agentstep is recorded.

SUMO results



Introduction

Conclusion

1. Experience replay is coupled with the Q- learning to eliminate the correlation in the observation sequence; consequently, improve the model performance.
2. **SUMO** environment was generated to train and test the traffic light system (agent).
3. Two metrics have been employed to evaluate the agent behaviour:
    3.1 Average queue length (numbers of vehicles)
    3.2 Accumulative delay time for all vehicles at intersection.
4. The results demonstrated that the proposed algorithm leads to a fair traffic control policy that **maximize** the **traffic flow** and **reduce** the **accumulative waiting time**.
5. **Future work** aims to improve the learning approach to accelerate convergence and avoid the occurrence of negative rewards.