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

Walid Shaker
Innopolis University
Innopolis, Russia
w.shaker@innopolis.university

Siba Issa
Innopolis University
Innopolis, Russia
s.issa@innopolis.ru

***Abstract*—Traffic congestion is a major area of concern that grab the researchers attention for many years.Traffic congestion can be effectively controlled by using traffic signals that adapt to real-world traffic scenarios. In this paper, we discuss using deep Q-learning to control traffic lights at four-way intersections. To enhance the model performance during the training, experience replay method is coupled with the deep Q-Learning. Simulation is conducted with SUMO (Simulation of Urban MObility) simulator in which real-world scenarios are examined. The obtained results are evaluated using two metrics such as the queue length and accumulative waiting time for all vehicles at the intersection. The results show that the agent managed to optimize the traffic flow by reducing vehicles' waiting time.**

***Index Terms*—reinforcement learning, deep Q-learning, traffic light control, traffic optimization, experience replay, traffic system simulation.**

I. Introduction

Throughout history, traffic congestion has been a major inconvenience, 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. In fact, the causes of traffic congestion can be summarized by two major factors; increasing traffic and inefficient traffic flow due to the increased number of vehicles. As a result, the importance of traffic control has grown and the control of the traffic light system has become one of the most substantial topics.

In this paper, we will focus on a concise and straightforward case that is nevertheless critical to the field of study: controlling a traffic light signal that directs traffic flow in four directions at a single intersection. we will model the studied case using an autonomous agent who is aware of the traffic around him and learns from previous experiences. For this research, the simulation is performed by using the SUMO tool which is a simulator of Urban Mobility [1]. In this simulation environment, the traffic light control agent can explore and analyze overall traffic flow as well as the rewards it receives for its actions. This is done through an external program.

The paper is organized as follows. As a starting point (section II), we will review related work on deep reinforcement learning and its limitations, Following that, we will formulate our problem and describe the intersection model, in addition to defining the reinforcement learning components: environment state, agent action set, reward function, and our agent goal in section III. After that we will introduce in section IV the deep Q-learning algorithm and its architecture. Therefore, we can move to section V and section VI to highlight the simulation setup and verify the algorithm by evaluating the results. Lastly, we will end the paper with our conclusion and our plans for the future in section VII.

II. Literature Review

When we searched the literature for solutions to traffic congestion issues, we found two fundamental classifications: one is based on the features considered for traffic light control (human-crafted and machine-crafted features), and the other is based on the approach utilized (fixed-time and adaptive approach).

According to an urban traffic signal control survey [2], controlling traffic lights based on human-crafted features does not allow for any optimal control. It relies on raw traffic data, such as vehicle location and speed, but it also ignores a lot of substantial information because it believes that the average vehicle delay is entirely determined by historical traffic data (Queue length) and does not process traffic in real time (it does not take into consideration vehicles that have not yet reached the waiting queue but are on their way). To extract all of the useful features for traffic regulation, machine-crafted features was adopted instead.

In [3], authors presented a deep reinforcement learning [4] algorithm based on machine-crafted features to control traffic lights in real time and learn the optimal traffic control policy. They used deep convolutional neural networks to obtain the required features (location, speed, and state) and then obtain the optimal decision to control the traffic light. However, the algorithm they used to make the decision was unstable and divergent, which is a well-known issue in the field of deep reinforcement learning. To solve this problem, they developed their algorithm based on experience replay and target network as proposed in [5].

As for controlling traffic lights using the fixed time approach, it is common to use, where the timing of the traffic light at the intersection is determined in advance and is improved based on pre-recorded traffic data (offline), and thus if this control system is used on real-life scenarios, we will not get satisfactory results due to its incompatibility with the dynamic movement requirements. Basically, every traffic signal phase has a fixed duration that is independent of the volume of traffic at the time, which can result in pointless delays during the green phase [6].

As a result, the dynamic approach to the control of traffic light signals has been demonstrated to be a practical technique to reduce traffic congestion, as it implies adjusting the traffic light in real time based on traffic demand, making it an effective and more appropriate method for real implementation [7], [8]. To adaptively control traffic lights, the authors in [7], [9] proposed using reinforcement learning - to solve the control problem - by modeling it as a Markov decision process [4].

A variety of factors [10] influence applying reinforcement learning for controlling the traffic. The agent needs only a simple information from its environment in which it learns, with the reward serving as a metric for system performance. If the agent is properly trained, it will show a good performance in a plenty examples (traffic accidents, weather conditions). Furthermore, reinforcment learning agents can learn in the absence of supervision or prior knowledge of their surroundings. As a result, in the context of the RL framework, any approach to designing any problem must define the state representation, possible actions, and reward function.

III. System Modelling and Problem Formulation

This study will examine the potential to improve traffic flow through a traffic light-controlled intersection using the reinforcement learning framework. The environment used is a 4-road intersection as shown in Fig. 1. Traffic lights at this intersection control how cars move: green lights indicate that cars can pass through the intersection, yellow lights indicate that cars can move cautiously and be ready to stop, and red lights indicate that cars must stop. For instance, Fig. 1, the green light is on for North South traffic situation.

The traffic light (our agent) has a goal to maximize a special metric to evaluate its efficiency. Based on our assumption we can formulate our problem as following: given a state of intersection, and in order to maximize the efficiency; the agent has to select from a specified set of actions. In short, to formulate the traffic light signal control problem based on reinforcement learning, system state, actions, the reward function, and the agent learning approach should be defined.

The formulation of the problem and the general workflow of the agent can be shown in Fig. 2, where the agent interacts with the environment in discrete time steps, $t = 0, 1, 2, \cdots$. It should be highlighted that steps in the SUMO simulation represent time passing, which we will call ”steps”. The agent only functions at specific time steps, which we will refer to as ”agentstep”. As a result, the agent begins to acquire the current intersection state S_t of the environment after many simulation steps. The action A_t is selected and this action transitions the intersection state to the next state S_{t+1} as vehicles move under the actuated traffic signal. Furthermore, the agent uses some indicators of the current traffic situation to determine the reward R_t for the previous action actuated. Moreover, a

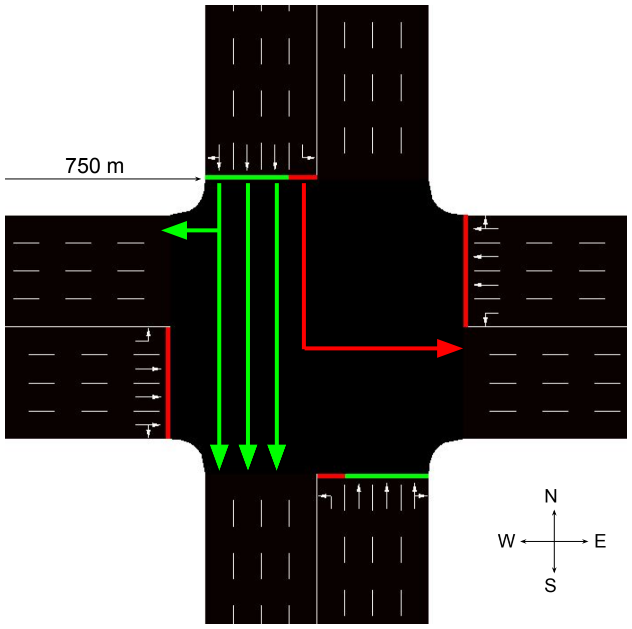


Fig. 1. A four-road intersection environment.

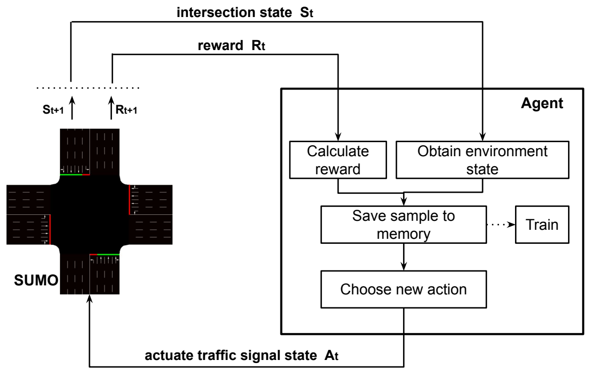


Fig. 2. Reinforcement learning agent for traffic signal control.

memory is used to store the sample of data comprising all of the details of the most recent simulation steps, which is then afterwards retrieved for a training session. Hence, the sequence of the agent’s interaction with the environment can be described as $\cdots, S_t, A_t, R_t, S_{t+1}, \cdots$. The following step is to specify the intersection state S_t, agent action A_t, and reward R_t.

A. Environment State

The state S_t is a model of the environment at a specific agentstep t. We need to give the agent enough information abot its environment and how the vehicle are distributed on each lane in order to make the agent learn how to optimize the traffic flow. In literature, the road lanes were discretized to set of cells and expressed as a vector that marking a cell with a 1 if a vehicle is present there and a 0 otherwise [11].

In [3], a new vector represent the velocity of the vehicles is combined with position vector. A third vector is added by [12] that represents the traffic light phase.

The selected representation's aim, which was inspired by the discrete traffic state encoding (DTSE) [12], is to inform the agent at agentstep t about the location of cars inside the environment. More specifically, the adopted state design is irregular in shape and contains only spatial data about the cars placed within the environment. The discretization of the continuous environment is done using irregular cells. The incoming lanes on each road of the intersection are technically divided into small areas that can determine whether or not a vehicle is present.

Fig. 3 shows the state representation for an intersection's road proposed by [6], where there are 20 cells between the start of the route and the stop line at the intersection. Half of the cells are assigned along the left-only lane, while the other half is spread out over the other lanes. Each cell's size varies according to its distance from the end line; the bigger the area, the more distance - from the lane - it covers. Each cell's length must be carefully chosen. The smallest areas, which are also the nearest to the end line, are bigger than the car's size.

Since there are 80 cells total in the 4-road intersection, The presence or absence of vehicles in each cell will be described by the agent observation of the environment state.

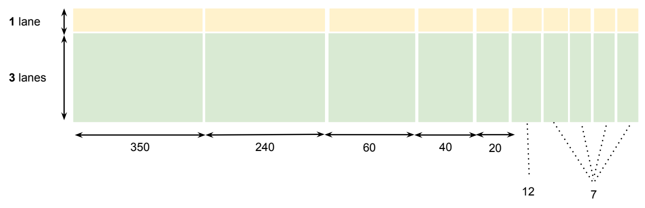


Fig. 3. State representation design with cells length.

B. Agent Action Set

Because the agent is a traffic light system, so doing an action means activating the green stage of a set of lanes for period of time. We defined he green duration as a 10 seconds, while the yellow duration is defined as 4 seconds. The actions represents all of the possible choices that the agent can take. (1).

$$A = \{NS, EW, NSL, EWL\} \tag{1}$$

The following is a description of every action in set (1).

- North-South (NS): Vehicles on the north and south roads are allowed to go straight ahead or make a right turn during the green period.
- East-West (EW): Vehicles on the east and west roads are allowed to go straight ahead or make a right turn during the green period.
- North-South Left (NSL): During the green period, vehicles in the specified sides are permitted to turn left.
- East-West Left (EWL): During the green period, vehicles on the east and west roads are permitted to turn left.

The four potential actions are represented graphically in Fig. 4.

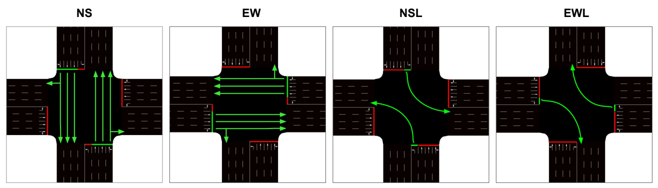


Fig. 4. The graphical representation of the four potential actions.

We should mention that if the state of the traffic light did not change, then the yellow sign wont be activated. But if we have a situation where the action is changed, then a yellow phase will start with 4 seconds simulation time and we will have a total simulation time between a two actions is 14 seconds (yellow phase in addition to the green phase). This process outline is depicted in Fig. 5, where first choosing the same action (NS), then translating from Action (NS) to Action (NSL).

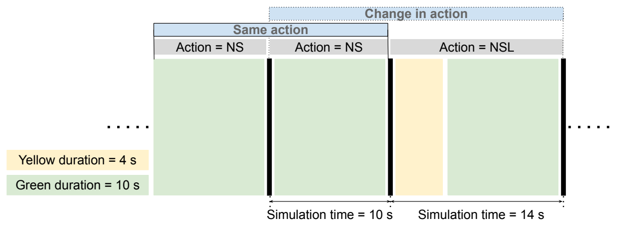


Fig. 5. Possible simulation time of chosen actions.

It should be noted that in a real-world application, such a presented agent can be easily replaced with a traffic control system.

C. Reward Function

The agent must evaluate and comprehend the chosen action's efficacy in the most recent state to improve the way of choosing the future actions. The reward function, like vehicle delays, queue lengths, and waiting times, is a performance efficiency indicator [6]. The cumulative waiting time of each vehicle, which leads to a cumulative reward function is the most commonly used approach in the traffic congestion problem, as adopted in [12] and [6]. In literature a weighted sum of multiple indicators combined together as suggested in [13]. To summarize, reward is the agent's way of evaluating the action he has taken, allowing him to improve his model of choices later on.

We can write the reward function however we want. In our problem, our goal is to reduce vehicle waiting time at traffic lights, but we want to cast it as a maximization problem, As a result, our goal is to gradually increase traffic flow through the intersection. To accomplish this in a realistic manner, the reward obtained from calculating the mean of the delay and travel time. Therefore, the agent must be aware of whether the decision he takes makes him close or far away from the desired goal (increasing traffic flow through the intersection).

The reward function varies depending on the performance measures that are considered. So, we will present two different reward function: Literature reward function and Alternative reward function.

1) Literature reward function: This function is named after studies that addressed the same problem, and uses the total waiting time as a metric as formulated in Eq. 2 where $twtt$ is the total waiting time in agentsetep t and n is the number of vehicles in the queue.

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

Hence, the literature reward function can be formulated as follows.

$$r_t = twt_{t-1} - twt_t \tag{3}$$

From Eq. 3, a negative reward is associated with a bad action that increases the number of vehicles in queues in the current agentstep t relative to the condition in the prior agentstep $t-1$, leading to wait for more time than in the previous agentstep. [6].

2) Alternative reward function: This function differs from the previous function in the metric as it uses the *accumulated waiting time* defined in Eq. (4)

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

Thus, the alternative reward function can be defined as follows.

$$r_t = atwt_{t-1} - atwt_t \tag{5}$$

$atwtt$ is the total amount of waiting time. By using this reward function, we can ensure that when the vehicle departs but is unable move forward (after intersection), $atwtt$ doesn't reset, to avoid the case when a long queue appears at the intersection, as well described in [6].

D. Agent Goal

The main goal in our case is to increase traffic flow through the intersection by reducing the vehicles waiting time. The agent observation of the intersection state, the agent must choose the appropriate action, which will be in accordance with action policy π^*. Each time step will have a reward for each action $(R_t, R_{t+1}, R_{t+2}, \cdots)$. We must emphasize a critical point here: when formulating our problem, we must define the agent's goal with respect to time: Do we need to achieve this goal one time? More formally, the agent should find an optimal policy π^* that maximizes the to be predicated cumulative reward is described by the action-value function in Eq. 6.

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}\{R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...|S_t = s, A_t = a, \pi\} \\ &= \mathbb{E}\{\sum_{k=0}^{\infty} \gamma^k R_{t+k} | S_t = s, A_t = a, \pi\} \end{aligned} \tag{6}$$

where γ is a discount factor, $0 \leq \gamma \leq 1$, so when $\gamma = 0$ indicates the immediate reward only, and when γ reaches 1 ithe agent is more foresighted.

Such an optimal policy π^* can be given by

$$\pi^* = \underset{\pi}{\text{argmax}} Q_\pi(s,a) \quad for \quad all \quad s \in S, a \in A \tag{7}$$

IV. Deep Q-Learning Algorithm

We introduce here the deep Q-Learning algorithm, which consists of deep neural networks and Q-Learning [14]. To achieve the agent goal (finding the optimal action policy π^*), the optimal Q-values $Q^*(s,a)$ are required to be obtained. As 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 [4] as follows.

$$\begin{aligned} Q^*(s,a) = \mathbb{E}\{R_t + \gamma \max_{a'} Q^*(S_{t+1}, a') | S_t = s, A_t = a\} \\ for \quad all \quad s \in S, a \in A \end{aligned} \tag{8}$$

If the total number of states is finite and we are aware of every aspect of the underlying system model, such as the transition probabilities of intersecting states and related expected reward, we can theoretically solve the Bellman optimality equation 8 to obtain the optimal Q-values $Q^*(s,a)$. Since the traffic environment is dynamically changing at the intersection, it is impossible to find the transition probabilities for the large intersection states caused by complex traffic scenarios. Therefore, Q-Learning has been widely utilized to overcome these issues, as it is a model-free [15] reinforcement learning approach where there is no need for these transition probabilities. The action value function $Q(s_t, a_t)$ of picking action a_t at state s_t is defined as follows.

$$Q(S_t, a_t) = Q(S_t, a_t) + \alpha(r_{t+1} + \gamma \cdot \max_A Q(s_{t+1}, a_t) - Q(s_t, a_t)) \tag{9}$$

This equation updates the Q-value by learning rate α that discounts the updated quantity. The temporal link between doing the action a_t and getting the resultant reward is highlighted by the subscript $t+1$. So, the reward for moving from state s_t in to state s_{t+1} taking an action a_t is represented by r_{t+1}. The expression max_A denotes that the most valuable action is chosen of all those available in the state s_{t+1}.

Authors in [6] used a modified version of Eq. 10 to calculate the Q-Learning function as follows.

$$Q(S_t, a_t) = r_{t+1} + \gamma \cdot \max_A Q'(s_{t+1}, a_{t+1}) \qquad (10)$$

where the Q-value for the action a_{t+1} in the state s_{t+1}, or the state that comes after taking an action a_t in the state s_t, is referred to as $Q'(s_{t+1}, a_{t+1})$.

It is frequently unfeasible to find and save each state-action pair in a reinforcement learning application due to the large state space. As a result, a deep neural network (DNN) is used to approximate the Q-learning function that results in $Q(s,a) \approx Q^*(s,a)$.

A. Deep Neural Network Architecture

In this study, a fully connected DNN is used as graphically presented in Fig. 6. The input layer has 80 neurons which represents the total number of cells at the intersection state. The input layer is then followed by 5 hidden layers, 400 neurons each, and activated using (ReLU) function [16]. The output layer consists of four neurons corresponding to the 4 possible action values with a linear activation function.

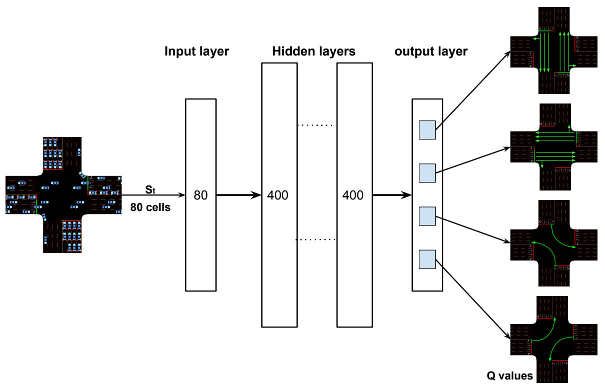


Fig. 6. Deep neural network architecture.

To enhance the agent's performance and learning effectiveness during the training, experience replay [17] technique is implemented. Instead of giving the agent the data it learns throughout the simulation right away, this method involves giving it the information it needs to learn in the form of a batch of randomly selected samples (commonly called Online Learning). Since the environment's state s_{t+1} is the next state of s_t and the agent will be learnt more in S_{t+1}, so the correlation might reduce the agent's capacity to learn, this approach is used to eliminate correlations in the observation sequence. The scheme of data collection task is depicted in Fig. 7.

As shown in the above scheme, each sample (E) can be formed by the set (11).

$$E_t = \{S_t, A_t, R_{t+1}, S_{t+1}\} \qquad (11)$$

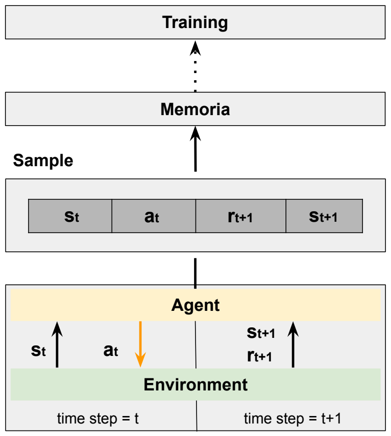


Fig. 7. Data collection task.

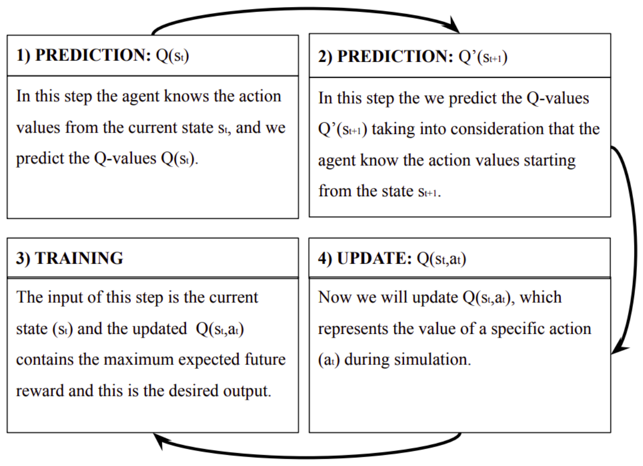
The batch is drawn from a data structure known as memory (M), which holds each sample collected throughout the training phase.

$$M = \{E_t, E_{t+1}, E_{t+2}, \cdots\} \qquad (12)$$

It is defined by a memory size and a batch size. In this model, the memory size was set to 5000 which represents the maximum number of samples that memory can hold. The memory size, which is set at 50000 samples, indicates how many samples the memory can hold. The batch size is set to 100 and it indicates the quantity of samples that are brought from memory during one training occurrence.

B. Deep Neural Network Process

The training process is performed iteratively for each training instance using the extracted data from the batch samples. Fig. 8 shows the training procedures that are carried out from the perspective of each sample.



1) PREDICTION: Q(st)

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

2) PREDICTION: Q'(st+1)

In this step the we predict the Q-values Q'(st+1) taking into consideration that the agent know the action values starting from the state st+1.

3) TRAINING

The input of this step is the current state (st) and the updated Q(st,at) contains the maximum expected future reward and this is the desired output.

4) UPDATE: Q(st,at)

Now we will update Q(st,at), which represents the value of a specific action (at) during simulation.

Fig. 8. Training process scheme.

It should be noted that during the update of $Q(s,a)$, the greedy method is adopted to choose the action that maximizes the predicted action value function $Q'(s_{t+1}, a_{t+1})$ as described in Eq. 10. However, this is not always the case when performing. We use the $\epsilon - greedy$ policy, described in Eq. 13, which performs the exploration with probability ϵ and chooses the exploitation with probability $1 - \epsilon$.

$$\epsilon_n = 1 - \frac{n}{N} \tag{13}$$

where n:current episode N:total number of episodes. Initially, ϵ is set to 1, indicating that the agent only explores. But as the training progresses, ϵ gradually decreases, allowing the agent to exploit what it has learned more and more exclusively.

Once the DNN has approximated the Q-Learning function, the optimal policy π^* is obtained, which selects the highest action value given the current state to maximize traffic efficiency.

V. Simulation Setting

In this section, we will demonstrate the setup for the SUMO simulation. In addition, the traffic route generation approach will be presented.

A. Training Setup

As we are working with a simulator, it is important to start by calculating the training time. We set the duration for each episode to 2 hours,and SUMO's default time frequency is one second. Therefore, N is 7200. To project this on a real life scenario, we set 100 episodes of 2 hours each, which is more than eight days of nonstop traffic, and by using a high-graphics laptop we managed to finish the entire training by 3.5 hours.

B. Traffic Route Generation

Traffic generation is a very critical point in our problem and to maintain a high level of realism, we used a Weibull distribution (shape = 2) to as it was proposed by [6]. The distribution is represented as a histogram, with the number of vehicles generated on the Y-axis and the steps of one episode on the X-axis. The Weibull distribution, as discussed in [6] where cars number increases early on, representing the busiest hour in the day and this simulates a real-life situations. The number of arriving cars then gradually decreases, as shown in Fig. 9. Furthermore, the dimensions and performance of each generated vehicle are the same.

To get an adaptive agent, we should include different scenarios as follows.

- High traffic scenario: four thousands cars reach the intersection.
- North-South (NS) traffic scenario: The traffic in this scenario is neither high nor low; it is in the middle, with two thousands cars arriving at the intersection.
- East-West (EW) West traffic scenario: The traffic in this scenario is neither high nor low; it is in the middle, with two thousands cars arriving at the intersection.
- Low traffic scenario: six hundred cars arrive at the intersection.

Each episode will represent different scenario, and scenarios will iterate throughout the training.

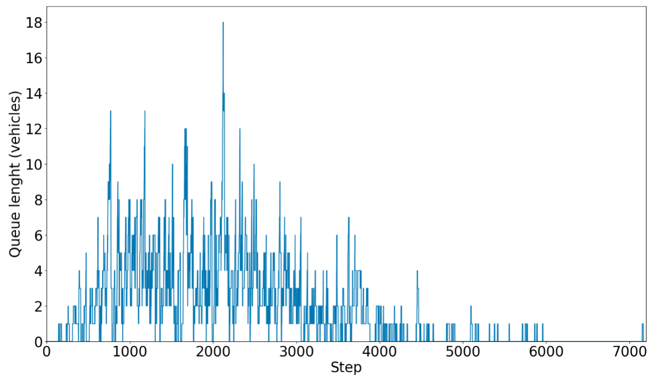


Fig. 9. Weibull distribution for traffic generation on a single episode.

VI. Simulation Results

The agent performance is examined using the training's reward trend followed by some metrics. These metrics evaluate the behavior of the traffic light system (agent) behaviour. The metrics used are the average queue length (number of vehicles) and the cumulative delay time for all vehicles at the intersection. Fig. 10 illustrates how agent's learning progressed throughout training in terms of cumulative negative reward. As can be shown, the agent has acquired a sufficiently correct policy.

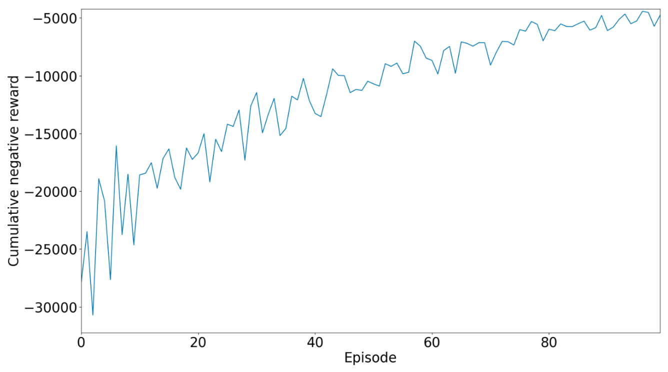


Fig. 10. Cumulative negative reward during the training per episode.

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. This function has proven its efficiency especially in long queues as it does not reset vehicles' waiting times by any simple movement through the queue. Fig. 11 shows the accumulative delay time during the training episodes. At the end of training, the agent managed to reduce the waiting time by utilizing the learned policy which is obtained by an appropriate approximation of the Q-values.

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. Fig. 12 represents the average queue length (vehicles) through the training episodes.

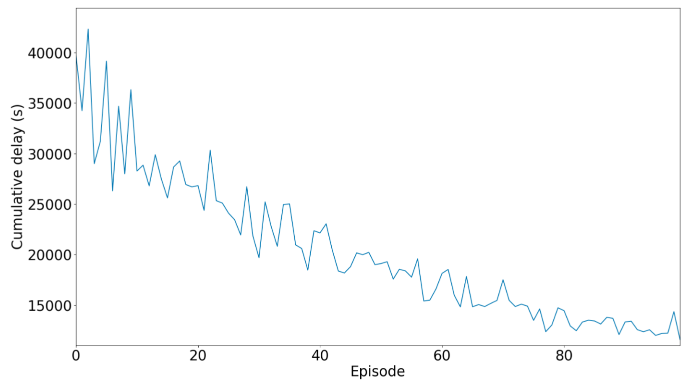


Fig. 11. Cumulative delay during the training per episode.

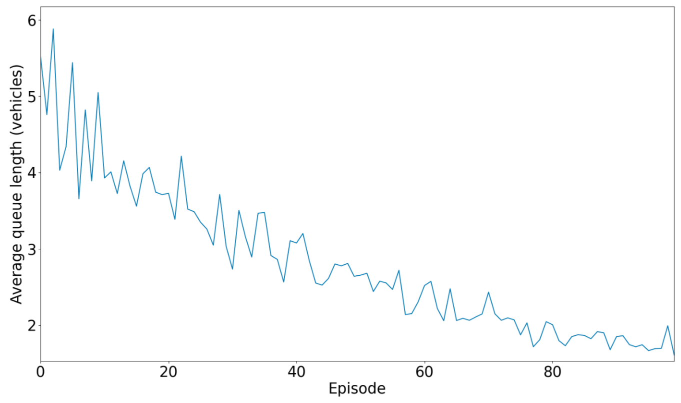


Fig. 12. Average queue length during the training per episode.

Model testing is performed and the associated reward to the actions taken at the agentstep is recorded as depcited in Fig. 13. It is noticeable that there are some negative rewards at some agentstep which indicates that the agent has chosen a bad action. However, the agent still perform well as the number of the good actions dominates the one of the bad actions which can be interpreted by the the reward values as shown in Fig. 13.

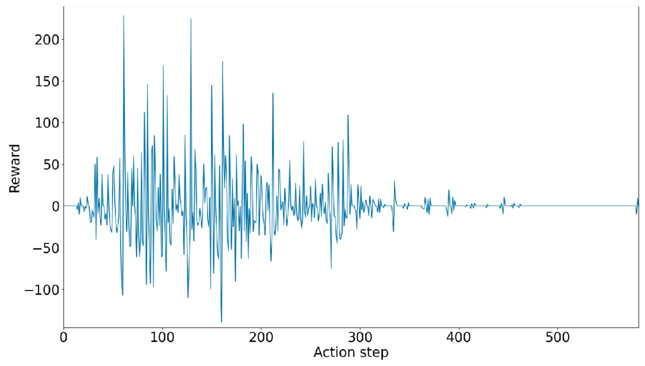


Fig. 13. Average queue length during the training per episode.

VII. Conclusion

The paper presented a deep Q-Learning algorithm to overcome the traffic congestion problem. The proposed algorithm is based on a deep neural network that approximates the Q-learning function. Experience replay is coupled with Q-learning to eliminate the correlation in the observation sequence; consequently, improve the model performance. SUMO environment was generated to train and test the traffic light system (agent). Two metrics have been employed to evaluate the agent behaviour: the average queue length (numbers of vehicles), and the accumulative delay time for all vehicles at intersection. The results demonstrated that the proposed algorithm leads to a fair traffic control policy that maximizes traffic flow and reduces accumulative waiting time.

Future work aims to improve the learning approach to accelerate convergence and avoid the occurrence of negative rewards.

References

[1] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "Sumo–simulation of urban mobility: an overview," in *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[2] D. Zhao, Y. Dai, and Z. Zhang, "Computational intelligence in urban traffic signal control: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 4, pp. 485–494, 2011.

[3] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," *arXiv preprint arXiv:1705.02755*, 2017.

[4] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] A. Vidali, L. Crociani, G. Vizzari, and S. Bandini, "A deep reinforcement learning approach to adaptive traffic lights management." in *WOA*, 2019, pp. 42–50.

[7] P. Mannion, J. Duggan, and E. Howley, "An experimental review of reinforcement learning algorithms for adaptive traffic signal control," *Autonomic road transport support systems*, pp. 47–66, 2016.

[8] A. A. Zaidi, B. Kulcsár, and H. Wymeersch, "Back-pressure traffic signal control with fixed and adaptive routing for urban vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2134–2143, 2016.

[9] L. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 412–421, 2010.

[10] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–38, 2017.

[11] W. Genders and S. Razavi, "Evaluating reinforcement learning state representations for adaptive traffic signal control," *Procedia computer science*, vol. 130, pp. 26–33, 2018.

[12] ——, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[13] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2496–2505.

[14] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[15] C. Watkins, "Learning form delayed rewards," *Ph. D. thesis, King's College, University of Cambridge*, 1989.

[16] B. Van Roy *et al.*, “An analysis of temporal-difference learning with function approximation,” *Automatic Control, IEEE Transactions on*, vol. 42, no. 5, pp. 674–690, 1997.

[17] S. Zhang and R. S. Sutton, “A deeper look at experience replay,” *arXiv preprint arXiv:1712.01275*, 2017.