Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 6505893, 14 pages
https://doi.org/10.1155/2020/6505893

Research Article

Adaptive Traffic Signal Control Model on Intersections Based on Deep Reinforcement Learning

Duowei Li,[1] Jianping Wu,[1] Ming Xu,[1] Ziheng Wang,[2] and Kezhen Hu[3]

[1]*Department of Civil Engineering, Tsinghua University, Beijing 100084, China*
[2]*Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[3]*China Academy of Information and Communication Technology, Beijing 100804, China*

Correspondence should be addressed to Jianping Wu; jianpingwu@tsinghua.edu.cn

Received 8 November 2019; Revised 11 May 2020; Accepted 17 July 2020; Published 3 August 2020

Academic Editor: Maria Vittoria Corazza

Controlling traffic signals to alleviate increasing traffic pressure is a concept that has received public attention for a long time. However, existing systems and methodologies for controlling traffic signals are insufficient for addressing the problem. To this end, we build a truly adaptive traffic signal control model in a traffic microsimulator, i.e., "Simulation of Urban Mobility" (SUMO), using the technology of modern deep reinforcement learning. The model is proposed based on a deep *Q*-network algorithm that precisely represents the elements associated with the problem: agents, environments, and actions. The real-time state of traffic, including the number of vehicles and the average speed, at one or more intersections is used as an input to the model. To reduce the average waiting time, the agents provide an optimal traffic signal phase and duration that should be implemented in both single-intersection cases and multi-intersection cases. The co-operation between agents enables the model to achieve an improvement in overall performance in a large road network. By testing with data sets pertaining to three different traffic conditions, we prove that the proposed model is better than other methods (e.g., *Q*-learning method, longest queue first method, and Webster fixed timing control method) for all cases. The proposed model reduces both the average waiting time and travel time, and it becomes more advantageous as the traffic environment becomes more complex.

1. Introduction

People's living standards have increased all over the world, leading to an increase in the ownership of private vehicles. While private vehicles have improved people's traveling experience, they have also contributed to traffic congestion, particularly in urban areas. According to data released by China's Ministry of Communications, economic losses caused by static traffic problems account for 20% of the disposable income of urban residents, equivalent to a 5%–8% GDP loss. The residents of 15 large cities in China spend 2.88 billion minutes more than the residents of developed European countries spend to get to work. Further, indirect losses (such as those associated with traffic accidents, social security, and environmental pollution) incurred as a consequence of traffic delays are even more difficult to quantify.

Two types of solutions are commonly employed to address the problems of traffic congestion, travel delays, and vehicle emissions. The first type of solution involves increasing capacity by expanding roads, which can be quite expensive and is too static to address the rapid changes in traffic conditions. The second and more reliable type of solution involves increasing the efficiency of the existing road structure. As an important part of the road network, traffic signal control is one of the most essential steps for improving operation efficiency and traffic safety at intersections [1]. With the rise of connected and automated vehicles (CAVs), many researchers believe that it may introduce great opportunities of reforming the conventional traffic signal operation, i.e., multivehicle cooperative driving around nonsignalized intersections [2]. However, we believe that traffic signal control will be still critical in the near future, where CAVs and traditional vehicles co-exist in a

mixed environment for a long process [3]. Many traffic networks worldwide still use fixed signal timings, i.e., they periodically change the signal in a round-robin manner. Although such a strategy is easy to implement, it does not consider the actual traffic conditions and may result in more congestion. Thus, it is vital to control the traffic signal intelligently and dynamically.

In industrial circles, most existing systems that optimize the specific settings of a traffic controller are based on complex mathematical models. The well-known Split Cycle Offset Optimization Technique (SCOOT, England) [4] and Sydney Coordinated Adaptive Traffic System (SCATS, Australia) [5] are examples of such systems that have improved traffic conditions in many countries. However, they suffer from inefficient handling of emergent traffic conditions, owing to a lack of real-time adaptability and flexibility [6], especially when undesirable human interventions like accidents or important events occur. Even systems that solve dynamic optimization problems in a real-time manner, such as the Real-Time Hierarchical Optimizing Distributed Effective System (RHODES) [7], suffer from exponential complexity that prevents them from being deployed on a large scale [8]. Longest queue first (LQF) is proved to be a robust adaptive algorithm which chooses to let the direction with the highest number of cars be green [9]. However, LQF may be unfair to vehicles waiting in a short queue that cannot accumulate enough length to be scheduled [10].

With the rapid development of artificial intelligence and computer technology, reinforcement learning (RL) has been widely used in academic circles as a method for achieving traffic signal control. Trial-and-error search and delayed reward are the two most important distinguishing features that make RL suitable for traffic signal control. RL can precisely represent the elements associated with the problem: agent (traffic signal controller), environment (state of traffic), and actions (traffic signals) [11].

Balaji et al. [12] proposed a Q-learning based traffic signal control model for optimizing green timing in an urban arterial road network to reduce the total travel time and delay experienced by vehicles. Due to the characteristics of discrete and limited action space in traffic signal control, Q-learning becomes the most common algorithm of RL used in this area. Related works [13–19] using this algorithm all achieve satisfying results. However, with an increase in the complexity of the environment, a computer may run out of memory; further, searching for a certain state from a large Q-table is time-consuming. Fortunately, in machine learning, neural networks are effective for overcoming the aforementioned drawbacks. Wan and Hwang [20] applied deep Q-network (DQN) in 8-phase traffic signal control and efficiently reduced the average system total delay. A DQN algorithm is a type of RL that combines the benefits of Q-learning and neural networks. Previous studies [11, 21–28] achieved good results when applying DQN methods using continuous state representations.

With regard to state space, action space, and reward function, people's choices vary. In general terms, the definitions and representations of state space in existing papers (e.g., total number of queued vehicles [12, 19–21, 27, 29], length of queued vehicles [12], speed of vehicles [11, 18, 23, 27], or traffic flow [15, 30]) can be modified to relay more effective information about the environment, which leads to more accurate judgments about the actions. The action space has been defined as all available signal phases [11, 18, 20, 27, 30, 31], or alternatively, it has been defined to maintain a sequence [22]. As for the definition of a reward function, most studies choose a reduction in the travel time of a vehicle [11, 22, 23], length of a vehicle queue [13, 15], or the time delay in queuing [11, 19, 20, 26, 28, 30]. Others [18] use the increase of throughput as reward, or the difference in queue length in different directions [24, 25, 27].

However, the current research has common problems and still requires improvements. First, most of these works [11, 13, 15, 22–25] focus on improvements at a single intersection and will not be satisfying enough when used in real-life situations. They may not result in an overall performance improvement, as such a policy only focuses on a small range and can cause congestions in upstream and downstream roads. Second, even studies that consider a larger range of networks [13, 14, 18, 19] all use relatively static synthetic data. Traffic conditions often show cyclical changes, and the flow rate also varies in directions in different time periods. The synthetic data used in most of the research studies until now are supposed to be distributed in a uniform manner, implying that the flow rates of all directions are equal. Even if an agent performs well under such traffic conditions, it cannot handle more complex environments, e.g., congestion in the north-south lanes with no vehicles in other directions. Third, the action options are not set in a proper way. A traffic signal usually changes in a round-robin manner, which is set with respect to the principles of transportation, as well as in line with people's habits and fairness guarantee. However, most of the previous research studies [11, 18, 30, 31] randomly choose one phase in each step, regardless of the sequence. This hopping phase design can be confusing for the driver, as the driver cannot prepare for the next phase in advance. Moreover, as the agent always chooses the optimal action, a loss of fairness may occur. For example, a lane with a minimum number of vehicles may never see a green light. In addition, the traffic signal phase setting in some of the previous works [19, 22–26] is too simple to represent a real road environment, which consists only two phases. Fourth, the interval of decision-making is not realistic. For example, studies like [20, 27] choose optimal action every second, which may lead to chaos and even accidents, Because very few drivers can react to such rapid changes. For other works, fixed time interval (e.g., 8 s or 15 s) is chosen without verifying reasonableness. Different traffic conditions may need different intervals, and either too long or too short interval can affect model effects.

This study proposes a truly adaptive traffic signal control agent, using DQN technology in the traffic microsimulator "Simulation of Urban Mobility" (SUMO). The function of the agent is defined as follows: given the state of traffic at one or more intersections, the agents will provide an optimal traffic signal phase and duration that should be implemented. Based on the above analysis of the previous studies, our approach offers several important contributions:

(1) Multiagent model that controls a large road network: Both single-agent case and multiagent case are demonstrated in this work. In particular, four agents that represent four adjacent intersections are trained at the same time, so as to achieve the effects of collaborative work and maximize the efficiency of the entire network.

(2) Global state and information sharing between agents: In a multi-intersection case, each agent can not only observe global traffic situation, but also obtain the current action of other agents. That is used to achieve cooperation between the agents.

(3) Action options that match the actual situation: The traffic signal in our approach contains four complete phases, while the action space only contains two options: change to the next phase or maintain the current phase. The agent must change to the next phase if the current phase has been maintained for three rounds. The action options in our approach match actual situations, and habituation and fairness are simultaneously guaranteed.

(4) Optimal action time interval for different traffic conditions: Experiments have been carried out to find the suitable interval under various conditions.

(5) Good model performance under various traffic conditions: Three different traffic conditions are tested in a simulation containing uniform and nonuniform distributions, sudden changes in traffic directions, and even more complex environments.

The rest of this paper is organized as follows. Section 2 describes related knowledge on the RL and DQN methods. Section 3 defines the general framework of the system, including the agent, state space, action apace, and rewards. The experiment results are presented in Section 4, and Section 5 provides concluding statements on our work.

2. Introduction to Reinforcement Learning and Deep *Q*-Network (DQN)

Inspired by behaviourist psychology, RL is concerned with how software agents should take actions in an environment so as to maximize expected benefits [31]. Unlike most machine learning methods, learners in RL are not told what action to take, but by trying to find out which behaviour produces the highest return [32]. In the most interesting and challenging cases, actions can not only affect direct rewards, but also affect the subsequent situation and all subsequent rewards. The framework of RL is shown in Figure 1. An agent is composed of three modules: state sensor I, learning machine L, and an action selector P. State sensor I maps an environmental state s to an agent internal perception i; action selector P selects an action a to act on the environment W according to the current strategy; learning machine L updates the agent's strategy based on the reward value r and the internal perception i; and finally, environment W facilitates a change to a new states' under action a. The basic principle of RL is that if a certain action of the agent leads to a positive environmental reward (strengthened signal), then the tendency of the agent to produce this action will strengthen. Conversely, if it leads to a negative reward, the tendency of the agent to produce this action will weaken [33].

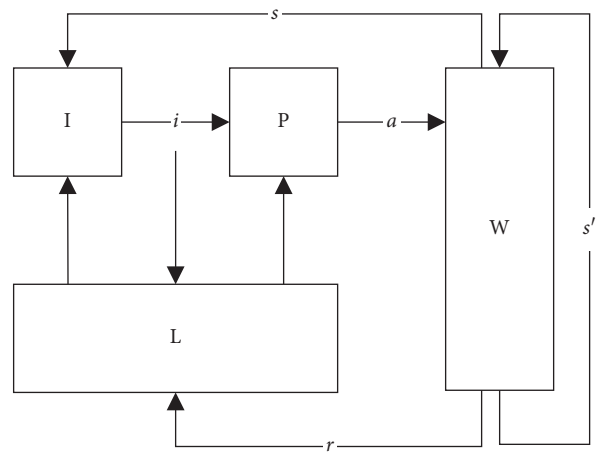


Figure 1: Framework of reinforcement learning.

Q-learning (Watkins and Dayan) [34] is a form of model-free, value-based, and off-policy reinforcement learning. It works by learning an action-value function that ultimately gives the expected utility of a given action a in a given state s, following optimal tactics. The policy π is the rule that the agent follows when choosing an action, given the state it is in [35]. When learning this action-value function, the optimal strategy can be constructed by selecting the action with the highest value in each state. The core of the algorithm is a simple value iteration update, as shown in equation (1), using the weighted average of the old value and the new information. The learning rate $\alpha(0 \le \alpha \le 1)$ determines to what extent the newly acquired information overrides old information, whereas the discount factor $\gamma(0 \le \gamma \le 1)$ determines the importance of future rewards [36].

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\left[r + \gamma \max_{a'} Q_t\left(s', a'\right) - Q_t(s,a)\right]. \tag{1}$$

In *Q*-learning, a *Q*-table is used to store each state and a corresponding *Q*-value owned by each action in this state. However, as discussed above, maintaining a *Q*-table is quite expensive when the environment becomes very complex. DQN, which combines the benefits of *Q*-learning and convolutional neural networks (CNNs), can overcome this problem very well. Receiving states and actions as input, the neural network can analyse and return the *Q*-value of each action [37], so that there is no need to record the *Q*-value in a table. A CNN is a class of deep, feed-forward artificial neural networks which has been successfully employed to analyse visual imagery. Since the state space in our model includes several large matrixes that can be regarded as pictures, CNNs are the best choice since they behave well in extracting spatial features from images so as to fully understand the

spatial characteristics around the intersection. A CNN consists of an input and an output layer, multiple convolutional layers, as well as optional hidden layers such as pooling layers, fully connected layers, and normalization layers. Figure 2 is a demonstration of how these layers can be combined to build a CNN according to the requirement. Convolutional layers apply a convolution operation to the input and pass the result to the next layer, so as to achieve feature extraction [38].

DQN modifies standard Q-learning in two ways, to make it suitable for training large neural networks without diverging. First, we use a technique known as experience replay, in which we store the agent's experiences at each time step $e_t = (s_t, a_t, r_t, s_{t+1})$ in a data set $D_t = \{e_1, \ldots, e_t\}$ pooled over many episodes (where the end of an episode occurs when a terminal state is reached) into a replay memory. During the inner loop of the algorithm, we apply Q-learning updates or mini-batch updates to samples of experience $(s, a, r, s') \sim U(D)$ drawn at random from the pool of stored samples. The second modification to Q-learning is aimed at further improving the stability of neural networks. It uses a separate network for generating the targets y_j in the Q-learning update. More precisely, after every C update, we clone the network Q to obtain a target network $\widehat{Q}$ and use $\widehat{Q}$ for generating the Q-learning targets y_j for the following C updates to Q. This modification makes the algorithm more stable as compared to standard online Q-learning [39].

3. Approach

Our truly adaptive traffic signal control system is divided into three modules: a signal control core algorithm, an interaction and control module, and a simulation module. The flowchart of information transfer between them is shown in Figure 3. First, the interaction and control module feeds the current environment state to the core algorithm. Second, the core algorithm passes the optimal action back according to ϵ-greedy strategy. Third, the interaction and control module changes the traffic signal, and the results are passed to the simulation module to be displayed in the SUMO GUI. Fourth, the interaction and control module calculates the rewards and passes them to the core algorithm. Fifth, the core algorithm learns and updates the policy according to the rewards received.

3.1. Agent Design. The three most essential parts of the agent are the state space S, action space A, and reward R.

3.1.1. State Space. The definitions and representations of the state space are very important, as the accuracy of judgments depends on the effectiveness of the information received about the environment. Thus, the system has very high requirements for the detector. Besides the two most common methods for acquiring traffic data, loop, and video detectors, CAVs can be utilized as "mobile detectors" to overcome those problems in the near future. CAVs can provide real-time vehicle location, speed, acceleration, and other vehicle information [40]. To take advantage of the CNN, the environment is processed as four pictures in our model: a map of vehicle locations, a map of the vehicle speed, a map of the trained intersection signal phase, and a map of the rest signal phase. It is worth noting that the map of the rest signal phase is specifically for multi-intersection case, which separates the signal of the intersection that the agent controls from the signal of other intersections. A representation of this process is shown in Figure 4, with triangles representing vehicles traveling on the road and the red line on the rightmost representing the right traffic signal in Figure 4(a). Notice that vehicles are supposed to have standard length, and the dotted lines in Figure 4(a) shows how the picture is divided into grids that is long in standard vehicle length and wide in lane width. Figure 4(b) shows the presence or absence of a vehicle in each location, and their corresponding speeds (m/s) are shown in Figure 4(c). The vehicles across two grids are presented in the grid to which its centre point belongs to. In Figure 5, the map of signal phase is processed as follows: the traffic lanes with green signal are set to 1, and others with red signal is set to 0. Considering information sharing between all agents in a multi-intersection case, a global traffic situation is used to achieve co-operation between the agents. The four input pictures are processed in the same way, only the size of the picture is larger. These settings ensure that the environment is accurately and sufficiently represented and that the state space is not too complex.

3.1.2. Action Space. In consideration of people's driving habits, a signal should be changed in a round-robin manner: NSG $\longrightarrow$ NSLG $\longrightarrow$ EWG $\longrightarrow$ EWLG (Figure 6). The action is defined as $a = 1$: change the signal to the next phase; and $a = 0$: maintain the current phase. A decision is made every 15 s, and according to the simulation results, the action time interval Δt has a negligible influence on performance as long as it is between 8 and 25 s (mentioned in details in section 4). No phase is allowed to be maintained for more than three rounds, and a yellow light is added for 3 s whenever a phase change occurs.

3.1.3. Reward. In each time step, all of the vehicles in the network are iterated. As shown in equation (2), if the speed v_i of vehicle i is below 2 m/s, then it is regarded as low-speed driving or waiting, and its waiting time W_i adds one. Once its speed reaches 2 m/s, W_i resets.

$$W_i(t) = \begin{cases} W_i(t-1) + 1, & v_i(t) < 2, \\ 0, & v_i(t) \geq 2. \end{cases} \tag{2}$$

The reward is calculated by equation (3) so as to make it inversely proportional to the average waiting time of each vehicle, which satisfies a target of RL, i.e., maximizing the reward. As Figure 7 shows, the reward r_i decreases faster as W_i increases. When W_i reaches a threshold value W_m, r_i will become negative, indicating that vehicle i has waited too long and green signal should be scheduled. Constant c is a parameter to control the upper bound of r_i. To test the performance more comprehensively, the average travel time

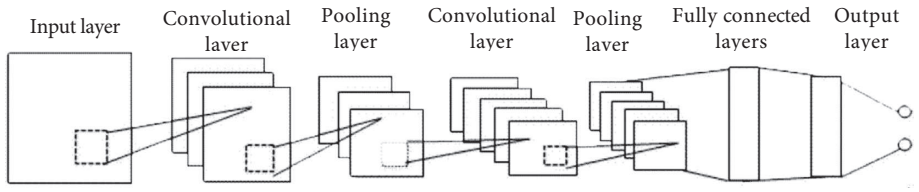


Figure 2: Architecture of CNNs.

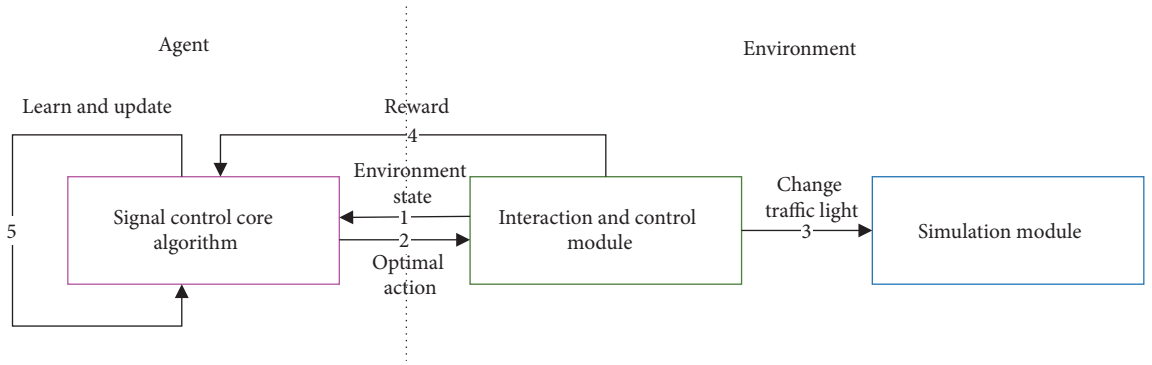
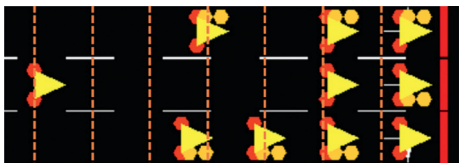


Figure 3: Flowchart of information transfer between modules.



(a)

0	0	0	1	0	1	1
1	0	0	0	0	1	1
0	0	1	0	1	1	1

(b)

0	0	0	12	0	0	0
22	0	0	0	0	0	0
0	0	5	0	2	0	0

(c)

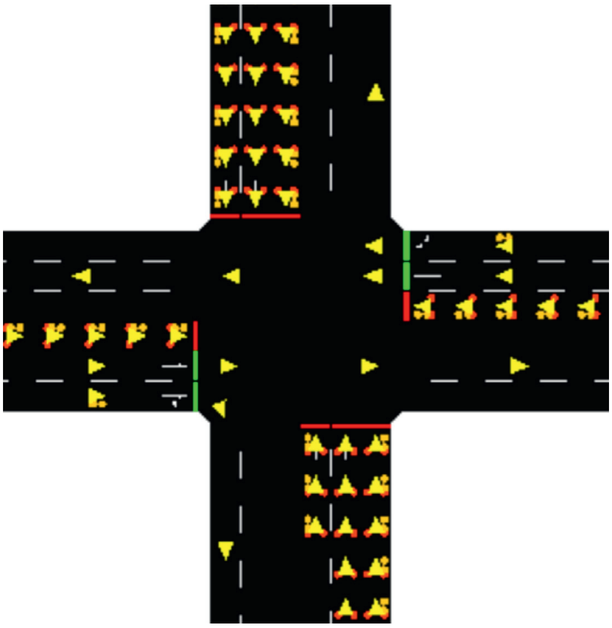
Figure 4: Example of processing maps of vehicle position and speed. (a) Simulation interface. (b) Vehicle locations. (c) Vehicle speed.

(from departure to arrival) and average speed of all vehicles is also output as an indicator.

$$r_i = c - c\left(\frac{W_i}{W_m}\right)^2. \tag{3}$$

3.2. Signal Control Algorithm Using DQN. The process using a DQN for optimal signal control (signal control core algorithm) is given in Algorithm 1 At each step t, the agent stores the observed environment experience $e_t = (s_t, a_t, r_t, s_{t+1})$ in the replay memory pool D. If D with finite capacity N is full, old experiences will be replaced by new ones. For the decision-making process, the agent chooses the action following a ε_greedy strategy. Because in the initial stage, Exploration (random exploration of the environment) is often better than Exploitation (fixed behavioral model choosing the action with highest value), so a parameter ε is imported to control the level of greediness (i.e., random action with probability ε and optimal action with probability $1 - \varepsilon$). As the training time increases, ε will gradually increase until equals to 1. Before the training process begins, the agent will observe without training for n steps until the replay memory reaches a certain size to guarantee a diverse interaction sample for the training. Once the training process begins, input data set is drawn randomly from the memory pool D. As mentioned in section 2, the corresponding target y_j in line 21 is generated by a separate Target_net with parameter $\widehat{Q}$. After collecting training data, network parameters θ is updated by perform a stochastic gradient descent step, where the loss function (Mean Squared Error) defined as equation (4) is minimized by Adam optimization algorithm [41]. For every fixed C steps, the Target_net updates its parameter $\widehat{Q}$ to Q.

$$L_j = \left(y_j - Q\left(s_j, a_j\right)\right)^2. \tag{4}$$

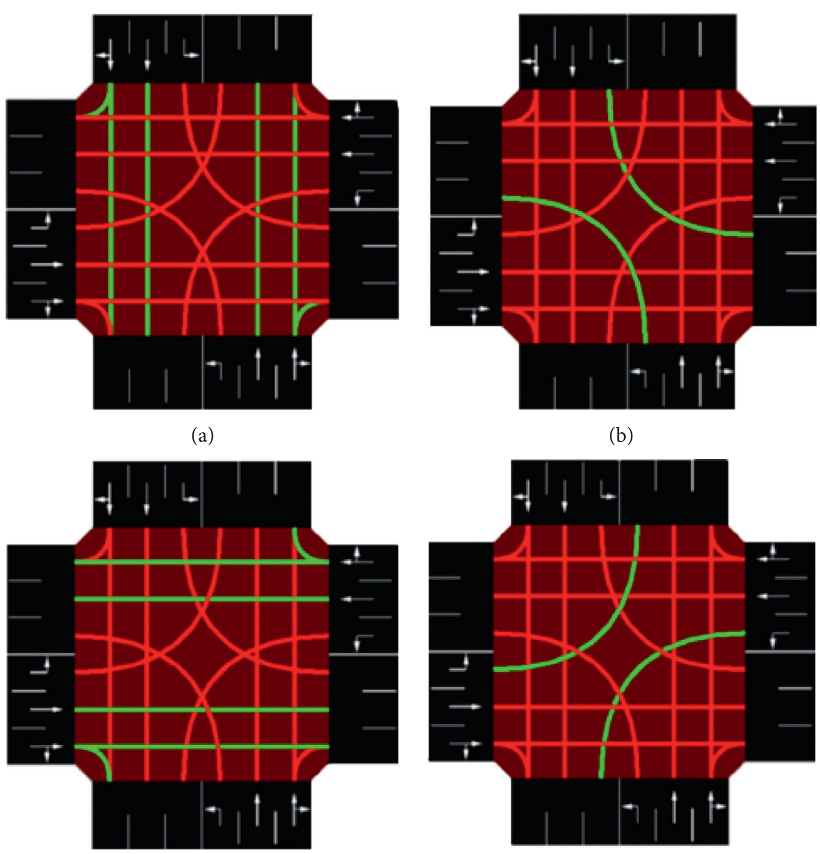


(a)

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

(b)

Figure 5: Example of processing map of signal phase. (a) Simulation interface. (b) Map of signal phase.



(a) (b) (c) (d)

Figure 6: Sequence of traffic signal phases. (a) NSG. (b) NSLG. (c) EWG. (d) EWLG.

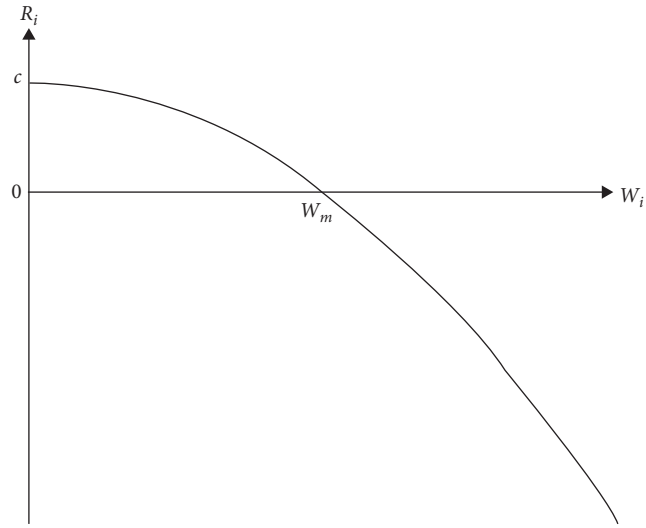


Figure 7: Schematic diagram of the function of reward as a function of waiting time.

(1) Definition
(2) D: = replay memory pool
(3) N: = maximum number of experiences in D
(4) Q: = action-value function in Eval_net
(5) $\widehat{Q}$: = action-value function in Target_net
(6) M: = maximum number of episode
(7) T: = maximum number of iteration in each episode
(8) Initialization
(9) $D \longleftarrow$ Initial replay memory to capacity N
(10) $Q \longleftarrow$ Initial evaluate action-value function with random weights θ
(11) $\widehat{Q} \longleftarrow$ Initial target action-value function with random weights $\theta^{-} = \theta$
(12) For episode = 1, M do
(13) Observe n steps before decision-making
(14) Initialize environment state s_1
(15) For $t = 1, T$ do
(16) With probability ε select a random action a_t
(17) Otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
(18) Execute action a_t in SUMO and observe reward r_t and environment state s_{t+1}
(19) Store experience $e_t = (s_t, a_t, r_t, s_{t+1})$ in D
(20) Sample random *batch_size* experiences $e_j = (s_j, a_j, r_j, s_{j+1})$ from D
(21) Set $y_j = \begin{cases} r_j, & \text{if episode terminates at step } j+1, \\ r_j + \gamma \max_{a'} \widehat{Q}(s_{j+1}, a'; \theta^{-}), & \text{otherwise.} \end{cases}$
(22) Updating network parameters θ by perform a gradient decent step on $(y_j - Q(s_j, a_j; \theta))^2$
(23) Every C steps reset $\widehat{Q} = Q$
(24) Set $s_t = s_{t+1}$
(25) End for
(26) End for

Algorithm 1: DQN with experience replay

In the multiagent case, each agent is trained individually, which means they keep their own neural network parameters.

3.3. Network Structure. As mentioned in Section 2, two separate neural networks are introduced in this model. Target_net is used to predict the Q_target value, and it does not update the parameters in time. Eval_net is used to predict Q_eval, and it has the latest neural network parameters. These two neural networks have completely identical structures, but they contain different parameters.

Each neural network receives four pictures (301×301) as input in multi-intersection case, and after processing the picture through six layers (four convolutional layers and two fully connected layers), they output a list (2×1) representing the value of each action. The structure of the entire network, including the processing method in each layer and the picture size

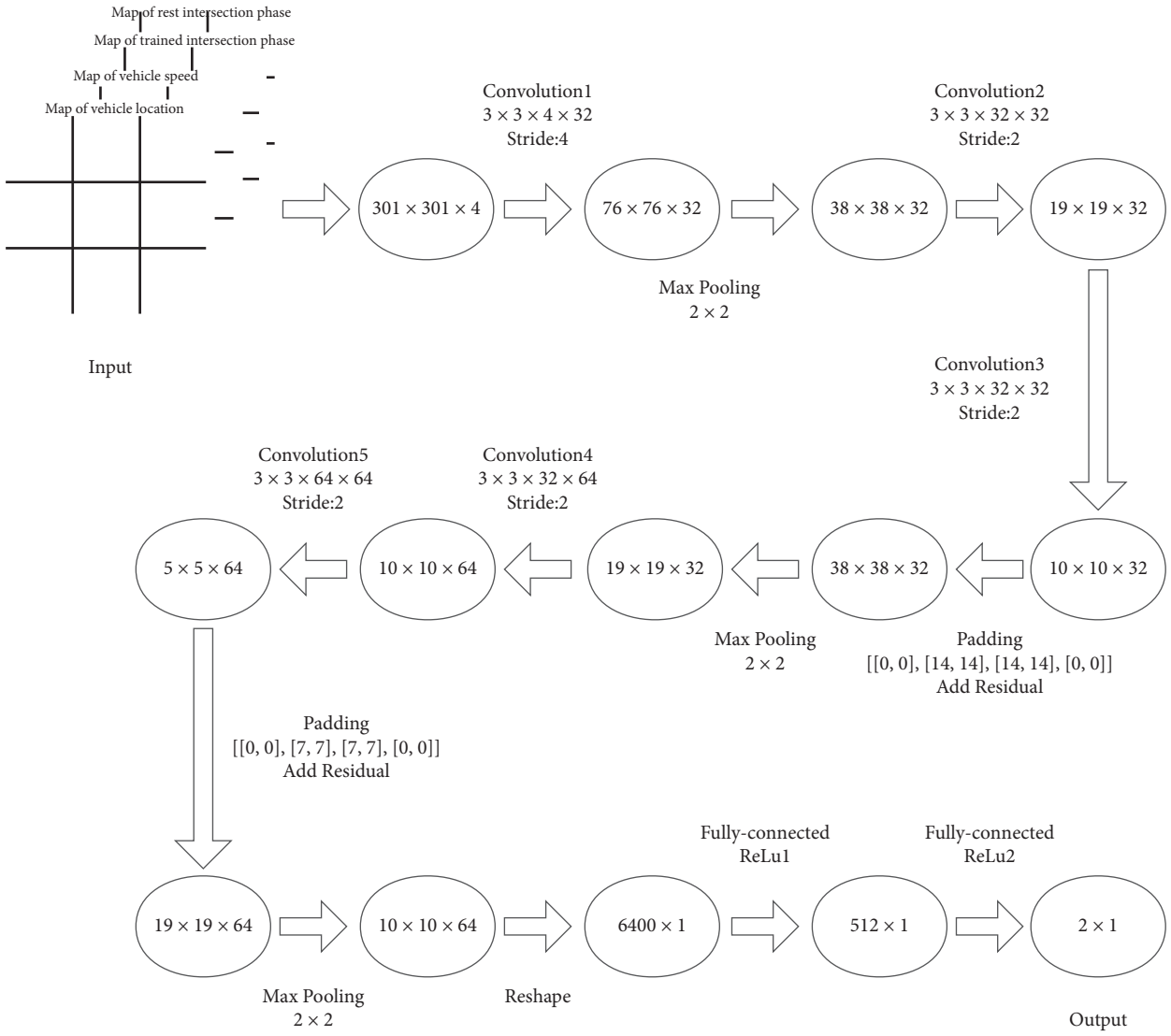


Figure 8: Structure of the neural network in multiagent case.

before and after each layer, is shown in Figure 8. The network structure of single-intersection case is not presented here.

4. Experiment and Results

In this section, 6 simulation tests are performed to show the performance of the system, including three different traffic conditions under the single-intersection case and the multi-intersection case, respectively.

4.1. Experiment Settings. SUMO is a free and open traffic simulation suite, available since 2001, that allows intermodal traffic systems, including road vehicles, public transport, and pedestrians, to be modelled [42]. The "Traffic Control Interface" (TraCI) is an interface of SUMO that provides access to a running road traffic simulation, retrieves values of simulated objects, and manipulates their behaviour "on-line".

The simulation network environments of the single-intersection case and multi-intersection case are shown in Figure 9, where the numbers within the parenthesis is the coordination of each node with meters as unit. Each intersection is connected with four road segments (Figure 6), consisting of a left-only lane, a straight-only lane, and a straight-right lane.

4.2. Parameter Settings. The parameter settings of our method are listed in Table 1.

4.3. Data Settings. As discussed in Section 1, three data sets are designed to cover a variety of traffic environments. The three data sets for the single-intersection case pertain to three different traffic conditions: No. 1–evenly distributed steady traffic; No. 2–sudden change in traffic direction; and No. 3–unevenly distributed steady traffic. The data sets are shown in Table 2. The data sets for the multi-intersection case are similar to those shown in Table 2 and are not listed here.

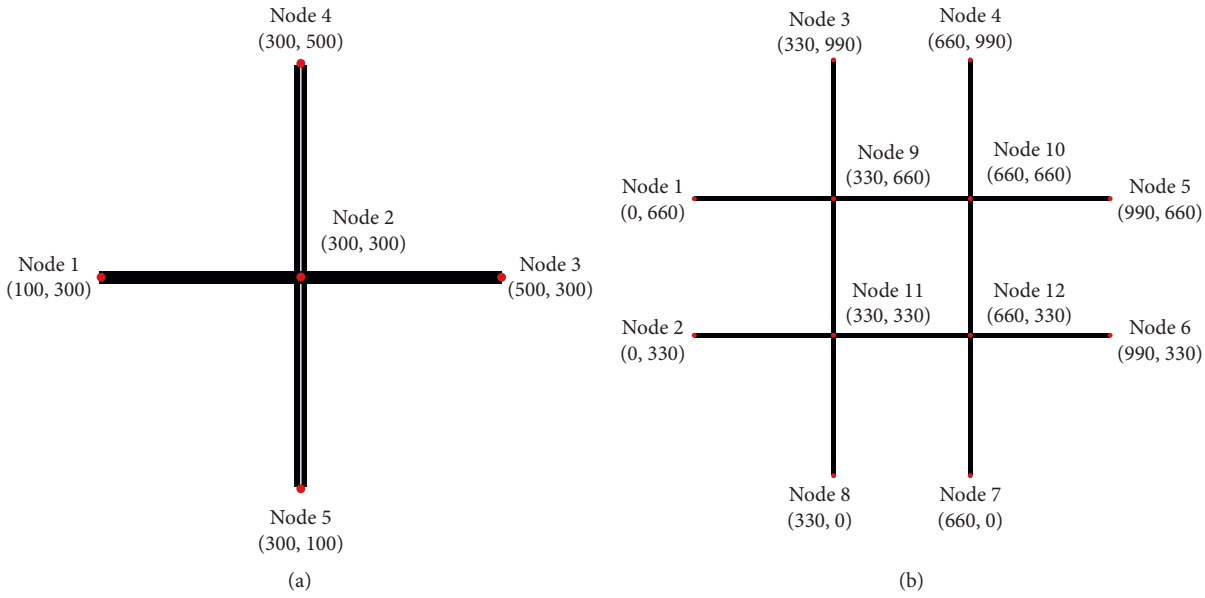


Figure 9: Coordination of simulated traffic network: (a) single-intersection case; (b) four-intersection case.

Table 1: Parameter settings for the model.

Parameter	Value	Meaning
Learning rate α	0.9	Extent to which new information is covered by old information.
Discount factor γ	0.9	Importance of future rewards.
ε	0.1	90% of the time the agent chooses the optimal strategy, while 10% of the time randomly explores.
replay_memory size N	1000	Maximum size of the memory pool.
batch_size	32	Size of memory that we extract from the pool for learning each time.
Constant c in reward function	0.15	The upper bound of reward.
W_m in reward function	60	Threshold value of W_i when reward becomes negative.
Update interval C	200	Frequency with which the parameters of the target_net updates.
Observe step n	100	Number of steps to observe before training process.
Training time	40000	Number of steps the agent trains.
Episode number M	200	Maximum number of episodes.
Iteration number T	200	Maximum number of iterations in each episode.

4.4. Control and Compared Methods. Four methods are compared and used for performing control experiments:

(1) Webster fixed signal timing control: Four traffic signal phases are periodically changed in a round-robin manner, where the duration of phases is designed using the most common Webster method [43]. Taking SL No. 3 in the single-intersection case as an example, the duration of each of the phases are 15 s, 33 s, 8 s, and 17 s, respectively.

(2) Longest Queue First (LQF) method: In every fixed step (same as our method.), this method always chooses the direction with the highest number of cars be green, which means the sequence of phases can be disrupted.

(3) Q-learning method: This method uses a Q-table to store each state and a corresponding Q-value owned by each action in this state. The state space is presented as a tuple with 25 elements (average speed, vehicle number for 12 lanes, and a current traffic phase). The other settings are the same as in our method.

(4) DQN method (without information sharing and global environment observation): This method is a base version of our proposed multiagent model. It regards the multi-intersection environment as several single and independent intersections, which means each agent can only observe the state of its corresponding intersection, without knowing the global state.

4.5. Performance Analysis

4.5.1. Single-Intersection Case. The performances of our method and the other compared methods under three traffic conditions are shown in Table 3 the values of DQN and Q-learning are those after training). It can be concluded that under a simple environment like a single intersection, both the Q-learning model and our model exhibit improvements. Although the Webster method performs fine under SL No. 1 (because a round-robin manner is suitable under evenly distributed steady traffic), it performs badly under the other

Table 2: Configurations of simulation data in single-intersection case.

SL no.	Traffic conditions	Directions	Number of vehicles	Time interval (s)	Flow rate (veh/h)
1	Evenly distributed steady traffic	South–North	500	0–3600	500
		North–South	500	0–3600	500
		West–East	500	0–3600	500
		East–West	500	0–3600	500
		North–East	500	0–3600	500
		East–South	500	0–3600	500
		South–West	500	0–3600	500
		West–North	500	0–3600	500
2	Simply direction changing traffic	South–North	400	0–1800	800
		North–South	400	0–1800	800
		West–East	400	1800–3600	800
		East–West	400	1800–3600	800
		North–East	400	0–1800	800
		East–South	400	1800–3600	800
		South–West	400	0–1800	800
		West–North	400	1800–3600	800
3	Unevenly distributed steady traffic	South–North	500	0–3600	500
		North–South	500	0–3600	500
		West–East	250	0–3600	250
		East–West	250	0–3600	250
		North–East	500	0–3600	500
		East–South	250	0–3600	250
		South–West	500	0–3600	500
		West–North	250	0–3600	250

Table 3: Performance in single-intersection case. Travel time: the lower the better; other measures: the higher the better.

SL no.	Method	Reward	Average travel time (s)	Average speed (m/s)
1	DQN (ours)	1.93	222.51	2.29
	Q-learning	1.90	228.42	2.26
	LQF	1.85	230.66	2.24
	Webster	1.66	240.96	1.84
2	DQN (ours)	5.02	89.31	4.83
	Q-learning	4.68	97.55	4.52
	LQF	0.08	170.93	2.63
	Webster	2.42	166.61	2.24
3	DQN (ours)	3.96	113.57	3.51
	Q-learning	3.54	118.98	3.37
	LQF	2.39	154.85	2.43
	Webster	2.54	163.89	2.33

two conditions. As for the LQF method, it performs smartly in SL No.1, but fails when there is a short queue that cannot accumulate enough length to be scheduled as mentioned in section 1. For example, when the vehicle flow direction suddenly changes in SL No.2, the small number of vehicles accumulated in North-South direction will never meet the green signal. And that also leads to the low value of reward according to equation (3). Table 4 indicates the improvements of the measures in our model relative to those of the *Q*-learning, LQF, and Webster methods. It can be seen that our model performs the best under all three traffic conditions, particularly by reducing travel time by 46.4% and by increasing the average speed to >100% under SL No. 2 as compared to the Webster method. It is worth noting that in SL No. 2, by observing the action records, our model can adjust the phase durations quickly when sudden changes occur. Figure 10 shows the episode rewards in 200 training episodes (40000 steps) under the three simulations. Our model converges within 90 episodes, and then remains steady afterwards.

4.5.2. Multi-Intersection Case. As Table 5 shows, the performances differ in the multi-intersection case. Our model is more efficient than the Webster method under all conditions, but the *Q*-learning model does not show a considerable improvement in this case. The failure of *Q*-learning is evident. When the state space becomes too complex, the number of rows in the *Q*-table will exponentially increase. For example, in 40000 steps, the number of rows in the *Q*-table is more than 20000. This means that the agent takes longer to randomly select an action under a new state than

Table 4: Improvements in measures of our model relative to those of other methods for single-intersection case.

SL no.	Method	Reward (%)	Average travel time (%)	Average speed (%)
1	Q-learning	1.6	−2.6	1.3
	LQF	4.3	−3.5	2.2
	Webster	16.3	−7.7	24.5
2	Q-learning	7.2	−8.4	6.9
	LQF	>100	−47.8	83.7
	Webster	>100	−46.4	>100
3	Q-learning	11.8	−4.5	4.2
	LQF	65.7	−26.6	44.4
	Webster	55.9	−30.7	50.6

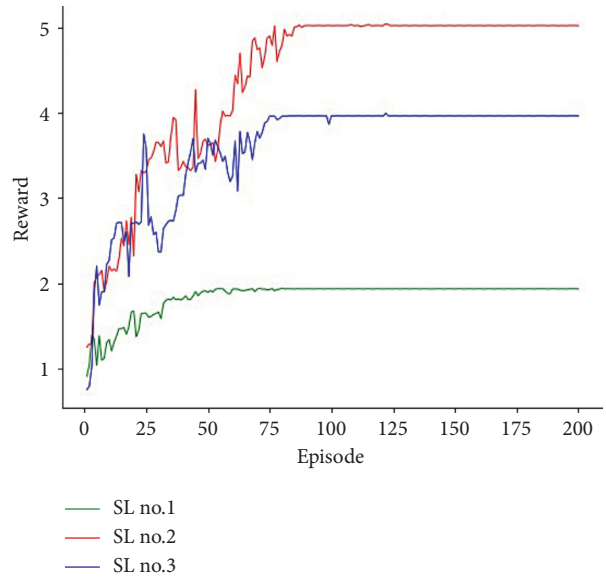


Figure 10: Episode rewards of our model in training process in single-intersection case.

select the best action according to the policy in an existing state. Thus, our model has more value for use in reality, as the environment in reality could be more complex. LQF still fails totally under SL No.2 and No.3. The base version of DQN model is always the second best method, but still performs poor compared to our method when the environment becomes more complex. That is because the state space and rewards of our proposed method are all global. The agent can finally learn a policy that gains the best overall performance, rather than only improving the traffic condition of its own intersection. That proves the importance of information sharing and global environment observation, which guarantees overall optimization. As shown in Table 6, our model still achieves the best performance under SL No. 2, where the travel time is reduced by 35.1% and the average speed is increased by 63.7% as compared to the travel time and average speed achieved when using the Webster method. Figure 11 shows the episode rewards in 200 training episodes (40000 steps) under the three simulations. Due to the complexity of the environment, the convergence speed is lower compared with single-intersection case. All three simulations converge and perform steady after 170 episodes.

The training time and space usage of our method for the whole 200 episodes is listed in Table 7. In addition, our experiment platform is a personal computer with Core (TM) M-5Y71 CPU @ 1.20 GHz 1.40 GHz/RAM: 8.00 GB. Python 3.6 and Tensorflow 1.0.0 are used to realized the models.

4.6. Influence of Action Time Interval $\Delta\mathbf{t}$. The action time interval $\Delta\mathbf{t}$ is another important parameter to the model. It should be kept within reasonable limits, and either too long or too short can affect model effects. We study the performance of our model using different values of $\Delta\mathbf{t}$ under SL No.1 in single-agent case. The result is shown in Figure 12, where 10 sets of value are taken nonequidistantly between 3 s and 40 s. The travel time is satisfactory (lower than 230 s) when $\Delta\mathbf{t}$ is in the range of [8, 27] and reaches the minimum value at 15 s. It is out of reality when $\Delta\mathbf{t}$ is below 8 s, because very few vehicles can pass through in such a short interval since people need time to react and start the vehicle. Also, the model will fail if $\Delta\mathbf{t}$ is too long. Once applied practically, the more frequent the decision-making, the higher the operating expenses (e.g., cost to switch light and observe the

Table 5: Performance in multi-intersection case. Travel time: the lower the better; other measures: the higher the better.

SL no.	Method	Reward	Average travel time (s)	Average speed (m/s)
1	DQN (ours)	2.54	438.26	2.49
	DQN (base)	2.42	486.95	2.21
	Q-learning	1.49	752.17	1.28
	LQF	2.37	496.80	1.93
	Webster	2.05	528.13	1.88
2	DQN (ours)	2.74	418.11	2.57
	DQN (base)	2.51	498.69	2.16
	Q-learning	1.68	701.82	1.38
	LQF	−0.02	816.29	1.18
	Webster	1.71	644.27	1.57
3	DQN (ours)	2.78	391.01	2.64
	DQN (base)	2.24	573.16	1.83
	Q-learning	1.71	681.12	1.44
	LQF	1.28	827.84	1.15
	Webster	1.75	588.22	1.71

Table 6: Improvements in measures of our model relative to those of other methods for multi-intersection case.

SL no.	Method	Reward (%)	Average travel time (%)	Average speed (%)
1	DQN (base)	5.0	−10.0	12.7
	Q-learning	70.5	−41.7	94.5
	LQF	7.2	−11.8	29.0
	Webster	23.9	−17.0	32.4
2	DQN (base)	9.2	−16.2	19.0
	Q-learning	63.1	−40.4	86.2
	LQF	>100	−48.8	>100
	Webster	60.2	−35.1	63.7
3	DQN (base)	24.1	−31.8	44.3
	Q-learning	62.6	−42.6	83.3
	LQF	>100	−52.8	>100
	Webster	58.9	−33.5	54.4

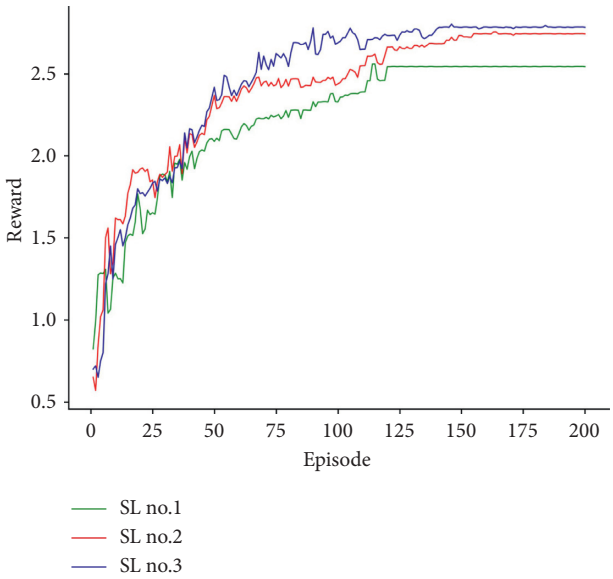


Figure 11: Episode rewards of our model in training process in multi-intersection case.

Table 7: Training time and space usage of DQN (ours) method.

	Training time (h)	Space usage (MB)
Single-intersection case	6	51
Multi-intersection case	25	500

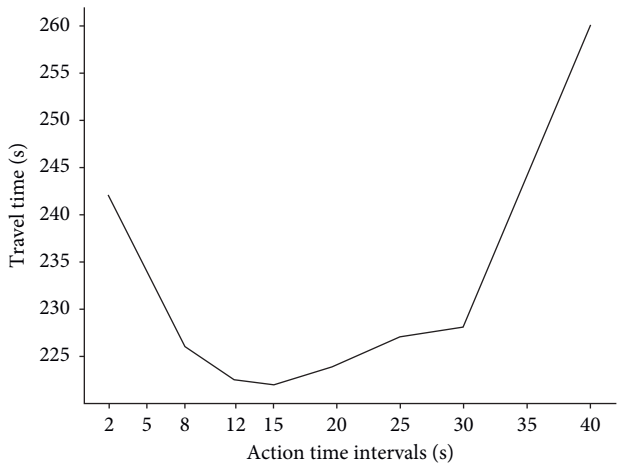


Figure 12: Travel time under different action time interval **t**.

environment). According to the analysis above, $\Delta\mathbf{t}$ is set as 15 s in our system. However, what must be acknowledged is that the influence of $\Delta\mathbf{t}$ varies under different traffic conditions. Due to time constraints, influence under other sets of simulation is not studied here.

5. Conclusions

In this paper, an intelligent and adaptive traffic signal control model based on a deep RL method is proposed. Using the advantages of DQN, agents learn how to determine an optimal signal phase and duration in reaction to a specific environment, in order to reduce waiting time and travel time and increase vehicle speed. The multiagent model observes global state and achieves information sharing between agents, so as to improve overall performance in a large road network. Various traffic conditions are considered to make our model suitable for all kinds of scenarios. Simulation results prove that our model performs better than three existing popular methods, *Q*-learning, LQF and Webster methods, and another base version of DQN method under all cases. The more complex the environment, the better the performance of our model.

Our study proves the reliability and efficiency in using RL for traffic signal control. With regard to future work, we acknowledge that this project is not perfect and that there are still many aspects that can be improved upon and researched. First, the experiment can be extended to use more complicated real map information. Second, real-world data and even real-world experiments should be carried out to further validate the performance of our method. Lastly, strengthen communication and co-operation between agents in the multi-intersection case may lead to better overall performance.

Data Availability

All data and program files included in this study are available upon request to the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization under Grant U1709212, and "Research on frontiers of intelligent transport system" funded by China Association for Science and Technology and National Natural Science Foundation of China under Grant U1509205.

References

[1] Z. H. Li, Q. Cao, Y. H. Hua, and Z. Rui, "Signal cooperative control with traffic supply and demand on a single intersection," *IEEE Access*, vol. 6, pp. 54407–54416, 2018.

[2] Y. Meng, L. Li, F.-Y. Wang, K. Li, and Z. Li, "Analysis of cooperative driving strategies for nonsignalized intersections," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 2900–2911, 2017.

[3] Q. Guo, L. Li, and X. Ban, "Urban traffic signal control with connected and automated vehicles: A survey," *Transportation Research Part C: Emerging Technologies*, vol. 101, pp. 313–334, 2019.

[4] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, "The SCOOT on-line traffic signal optimisation technique," *Traffic Engineering and Control*, vol. 23, no. 4, pp. 190–192, 1982.

[5] A. G. Sims and K. W. Dobinson, "The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits," *IEEE Transactions on Vehicular Technology*, vol. 29, no. 2, pp. 130–137, 1980.

[6] S. Chiu, "Adaptive traffic signal control using fuzzy logic," in *Proceedings of the Intelligent Vehicles 92 Symposium*, pp. 98–107, Detroit, MI, USA, July 1992.

[7] P. Mirchandani and L. Head, "A real-time traffic signal control system: Architecture, algorithms, and analysisfic signal control system: Architecture, algorithms, and analysis," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 6, pp. 415–432, 2001.

[8] Y. Dujardin, D. Vanderpooten, and F. Boillot, "A multi-objective interactive system for adaptive traffic controlfic control," *European Journal of Operational Research*, vol. 244, no. 2, pp. 601–610, 2015.

[9] R. Wunderlich, C. Liu, I. Elhanany, and T. Urbanik, "A novel signal-scheduling algorithm with quality-of-service provisioning for an isolated intersection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 3, pp. 536–547, 2008.

[10] J. Wu, D. Ghosal, M. Zhang, and C.-N. Chuah, "Delay-based traffic signal control for throughput optimality and fairness at an isolated intersection," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 896–909, 2017.

[11] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," 2016, http://arxiv.org/abs/1611.01142.

[12] P. G. Balaji, X. German, and D. Srinivasan, "Urban traffic signal control using reinforcement learning agents," *IET Intelligent Transport Systems*, vol. 4, no. 3, pp. 177–188, 2010.

[13] Y. K. Chin, N. Bolong, A. Kiring, and S. S. Yang, "Q-learning based traffic optimization in management of signal timing plan," *International Journal of Simulation: Systems, Science and Technology*, vol. 12, no. 3, pp. 29–35, 2011.

[14] A. Salkham, R. Cunningham, A. Garg, and V. Cahill, "A collaborative reinforcement learning approach to urban traffic control optimization," in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, vol. 2, pp. 560–566, Washington, DC, USA, 2008.

[15] R. Marsetič, D. Šemrov, and M. Žura, "Road artery traffic light optimization with use of the reinforcement learning," *PROMET–Traffic&Transportation*, vol. 26, no. 2, pp. 101–108, 2014.

[16] C. Ozan, O. Baskan, S. Haldenbilen, and H. Ceylan, "A modified reinforcement learning algorithm for solving coordinated signalized networks," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 40–55, 2015.

[17] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Holonic multi-agent system for traffic signals control," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1575–1587, 2013.

[18] M. Stevens and C. Yeh, "Reinforcement learning for traffic optimization," 2016, https://www.stanford.edu.

[19] A. Daeichian and A. Haghani, "Fuzzy Q-learning-based multi-agent system for intelligent traffic control by a game theory approach," *Arabian Journal for Science and Engineering*, vol. 43, no. 6, pp. 3241–3247, 2018.

[20] C. H. Wan and M. C. Hwang, "Adaptive traffic signal control methods based on deep reinforcement learning," *It's for Everyone's Mobility*, pp. 195–209, Springer, Berlin, Germany, 2019.

[21] M. A. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pp. 1151–1158, Stanford, CA, USA, June 2000.

[22] H. Wei, G. Zheng, H. Yao, and A. Li, "Intellilight," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2496–2505, London, UK, August 2018.

[23] J. Gao, Y. She, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," 2017, http://arxiv.org/abs/1705.02755.

[24] L. Li, Y. Lv, and F. Y. Wang, "Traffic signal timing via deep reinforcement learning," *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 3, pp. 247–254, 2016.

[25] S. S. Mousavi, M. Schukat, and E. Howley, "Traffic light control using deep policy-gradient and value-function-based reinforcement learning," *IET Intelligent Transport Systems*, vol. 11, no. 7, pp. 417–423, Aug. 2017.

[26] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, Barcelona, Spain, 2016.

[27] H. W. Ge, Y. M. Song, C. G. Wu, J. K. Ren, and G. Z. Tan, "Cooperative deep Q-learning with Q-value transfer for multi-intersection signal control," *IEEE Access*, vol. 7, pp. 40797–40809, 2019.

[28] X. Y. Liang, X. H. Du, G. L. Wang, and H. Zhu, "A deep Q learning network for traffic lights' Cycle control in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1243–1253, 2019.

[29] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *Journal of Transportation Engineering*, vol. 129, no. 3, pp. 278–285, 2003.

[30] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, "Reinforcement learning-based multi-agent system for network traffic signal control," *IET Intelligent Transport Systems*, vol. 4, no. 2, pp. 128–135, 2010.

[31] R. Lu, S. H. Hong, and X. Zhang, "A dynamic pricing demand response algorithm for smart grid: reinforcement learning approach," *Applied Energy*, vol. 220, pp. 220–230, 2018.

[32] B. Abdulhai and L. Kattan, "Reinforcement learning: Introduction to theory and potential for transport applications," *Canadian Journal of Civil Engineering*, vol. 30, no. 6, pp. 981–991, 2003.

[33] Y. Gao, S. F. Chen, and X. Lu, "Research on reinforcement learning technology: A review," *Acta Automatica Sinica*, vol. 30, no. 1, pp. 86–100, 2004.

[34] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[35] H. Shi, Z. Lin, K.-S. Hwang, S. Yang, and J. Chen, "An adaptive strategy selection method with reinforcement learning for robotic soccer games," *IEEE Access*, vol. 6, pp. 8376–8386, 2018.

[36] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, MIT press, Cambridge, MA, USA, 2018.

[37] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[38] P. Sermanet, D. Eigen, X. Zhang, and M. Mathieu, "Overfeat: Integrated recognition, localization and detection using convolutional networks," 2013, http://arxiv.org/abs/1312.6229.

[39] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[40] K. Gao, F. R. Han, P. P. Dong, N. X. Xiong, and R. H. Du, "Connected vehicle as a mobile sensor for real time queue length at signalized intersections," *Sensors*, vol. 19, no. 9, p. 2059, 2019.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, http://arxiv.org/abs/1412.6980.

[42] A. K. Ama and S. Kumar, "Traffic simulation of vehicular cloud network using sumo," *International Journal of Soft Computing and Engineering*, vol. 6, no. 1, pp. 378–383, 2016.

[43] F. V. Webster, "Traffic signal settings," in *Proceedings of the TRB's Conference*, Washington, DC, USA, 1958.