Multi-Agent Deep Reinforcement Learning for Urban Traffic Light Control in Vehicular Networks

Tong Wu, *Student Member, IEEE*, Pan Zhou, *Member, IEEE*, Kai Liu, *Member, IEEE*, Yali Yuan, *Member, IEEE*, Xiumin Wang, *Member, IEEE*, Huawei Huang, *Member, IEEE*, Dapeng Oliver Wu, *Fellow, IEEE*

***Abstract*—As urban traffic condition is diverse and complicated, applying reinforcement learning to reduce traffic congestion becomes one of the hot and promising topics. Especially, how to coordinate the traffic light controllers of multiple intersections is a key challenge for multi-agent reinforcement learning (MARL). Most existing MARL studies are based on traditional Q-learning, but unstable environment leads to poor learning in the complicated and dynamic traffic scenarios. In this paper, we propose a novel multi-agent recurrent deep deterministic policy gradient (MARDDPG) algorithm based on deep deterministic policy gradient (DDPG) algorithm for traffic light control (TLC) in vehiclar networks. Specifically, the centralized learning in each critic network enables each agent to estimate the policies of other agents in the decision-making process and each agent can coordinate with each other, alleviating the problem of poor learning performance caused by environmental instability. The decentralized execution enables each agent to make decisions independently. We share parameters in actor networks to speed up the training process and reduce the memory footprint. The addition of LSTM is beneficial to alleviate the instability of the environment caused by partial observable state. We utilize surveillance cameras and vehicular networks to collect status information for each intersection. Unlike previous work, we have not only considered the vehicle but also considered the pedestrians waiting to pass through the intersection. Moreover, we also set different priorities for buses and ordinary vehicles. The experimental results in a vehicular network show that our method can run stably in various scenarios and coordinate multiple intersections, which significantly reduces vehicle congestion and pedestrian congestion.**

***Index Terms*—Traffic Light Control, Deep Reinforcement Learning, Deep Deterministic Policy Gradient Algorithm, Markov Decision Process, Vehicular Network.**

I. Introduction

Tong Wu, School of Electrical Information and Communication Engineering, Huazhong University of Science & Technology, Wuhan, 430074, China (e-mail: M201971975@hust.edu.cn).

Pan Zhou, Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science & Technology, Wuhan, 430074, China (e-mail: panzhou@hust.edu.cn). (*Corresponding author: Pan Zhou*)

K. Liu is with the College of Computer Science, Chongqing University, Chongqing 400040, China (e-mail: liukai0807@gmail.com).

Y.Yuan is with the institute of computer science, Göttingen University, 37077 Goldschmidtstraße 7, Göttingen, Germany (e-mail: yali.yuan@informatik.uni-goettingen.de).

X. Wang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail:xmwang@ scut.edu.cn).

H. Huang is with the School of Data and Computer Science, Sun Yat-Sen University, 510006 Guangzhou, China (e-mail: huanghw28@mail.sysu.edu.cn).

Dapeng Oliver Wu, Department of Electrical and Computer Engineering, University of Florida, Gainesville, 32611 USA (e-mail: wu@ece.ufl.edu).

WITH the rapid development of the social economy and the improvement of people's material and spiritual life, the city is constantly developing and expanding. And private cars are becoming more and more common, which bring great pressure on traffic problems [1]. Simultaneously, traffic congestion will also bring unnecessary resource loss and environmental pollution, and it will also increase the probability of traffic accidents. The most direct way to alleviate traffic congestion is to control traffic lights. Traditional fixed time traffic light control algorithms [2] become inefficient with respect to alleviating congestion in a highly dynamic traffic environment. The longest queue first traffic light controls algorithm control traffic lights simply according to the length of the queue, and it still cannot solve the congestion at multiple intersections. Manual traffic control is sometimes effective, but it is a waste of manpower and time. Thus, we need computers to be able to learn to make a reasonable decision by interacting with the environment.

In recent years, with the sustainable development of the Internet and wireless communication, the acquisition of various data has become more convenient and faster than before. Rational use of various traffic data is also the key to alleviating traffic congestion. Among them, the Internet of Things (IoT) [3], as a key component of the new generation of information technology, plays an important role in our lives, where vehicular network is a branch of the IoT. Vehicular ad hoc networks (VANETs) [4] offer direct communication between vehicles (V2V) and between vehicles and infrastructure (V2I). And vehicles can send and receive hazard warnings or current traffic information with minimal latency in VANETs. In this regard, we can utilize VANETs to capture real-time vehicle information and road condition information for traffic light control (TLC), similar to [5], [6].

As the recent advance of communication and computational sensing, there are new opportunities for developing intelligent transportation. Many researchers have proposed various adaptive traffic control systems (ATCS). The traffic lights can be reasonably changed according to the current traffic congestion situation, such as SCOOT [7], SCATS [8], OPAC [9], RHODES [10]. These systems have been applied in hundreds of cities around the world. Some researchers have applied neuro-fuzzy [11], neural networks [12] and genetic algorithms [13] to ATSC. As we know, genetic algorithms and neural networks require a lot of computation. Moreover, there exists premature convergence in genetic algorithms. And neuro-fuzzy is not suitable for complex or diverse intersections, as mentioned in [14]. In contrast, the reinforcement learning

algorithm is more concise and efficient for TLC.

In previous works, reinforcement learning [15] has been widely used in robot control and game theory. Traffic light control process can be defined as MDP [16]. Thus many researchers have applied reinforcement learning to TLC. However, the computational complexity of the traditional reinforcement learning system grows exponentially with the increase of the state space. The combination of the fast-developing deep learning and traditional reinforcement learning further solve this problem, i.e., Deep Reinforcement Learning (DRL) [17], [18]. The biggest difference from classic reinforcement learning is that it directly handles the pixel-level super-raw raw image state input instead of abstracting the state into a low-dimensional state. Although the DRL algorithms has achieved good results in single intersection scenarios, there are many problems when they are applied to multiple intersections. In the real world, it will make more sense to directly control multiple intersections with complex traffic dynamics.

In the case of multiple intersections, congestion at different intersections is different and the congestion levels at multiple intersections will affect each other. Each intersection cannot fully sense the traffic conditions of other intersections. This is because each agent can only observe part of the traffic situation, which will lead to a non-stationary environment. Many researchers have begun to apply the multi-agent reinforcement learning (MARL) algorithms [19]–[22] to TLC, so how to coordinate multiple traffic light controllers is a major challenge. However, the centralized algorithms [23], [24] slow down the convergence of the entire system and are not suitable for practical traffic control. Although the decentralized algorithms [25], [26] is robust and has a fast convergence speed, only part of the state of the adjacent intersection is considered in the training and execution process, so that the global optimal policy may not be learned. Moreover, the previous work did not combine the real traffic condition that not only the vehicles want to pass, but also the pedestrians want to pass at each intersection. And the number of passengers on the bus can reach several times that on other vehicles, treating the two as the same kind of vehicle is obviously unfair.

As noticed, when the scale of the traffic network grows, it will inevitably lead to insufficient utilization of the state of the intersection. And the area around local intersections will often have an impact on the traffic control of these junctions. In this paper, we mainly consider and focus on the medium-scale urban traffic light control in a vehicular network. The main contributions of our method are as follow:

1) We propose a novel MARDDPG algorithm to solve traffic congestion at multiple intersections. The centralized learning in each critic network enables each agent to estimate the policies of other agents in the decision-making process. The decentralized execution enables each agent to make decisions independently. Every intersection can compete or collaborate with each other, and ultimately reach the global optimal policy. We also utilize LSTM [27] to capture hidden state information. Moreover, we shares parameter in actor networks to speed up the training process and reduce the memory footprint.
2) Since there are a large number of pedestrians in urban or commercial areas, sometimes traffic lights controlled by the state of the vehicle will cause long waiting time and inconvenience of pedestrians [28]. To tackle this problem, we also take the total waiting time of pedestrians crossing the road into consideration.
3) Different from previous research, in order to make our model more realistic with traffic problems in the real world, we no longer regard all vehicles as the same priority. Buses are used as urban public transport, the number of passengers is more than the number of passengers on ordinary vehicles. Therefore, we give buses a higher priority to pass.
4) In order to make full use of various state information, we have defined more comprehensive states and rewards to improve learning.
5) We collect spatio-temporal traffic information via a vehicular network to better coordinate the cooperation among multiple agents, thereby controlling traffic lights at multiple intersections to obtain an optimal traffic light control policy.

The rest of this paper is organized as follows. In Section II, we discuss the related work. The background on reinforcement learning is presented in Section III. In Section IV, we introduce system model and problem formulation. In Section V, we define the deep reinforcement learning model of vehicular networks. Section VI shows the details of our algorithm for TLC. We evaluate our method in Section VII. Finally, we conclude this paper in Section VIII.

II. RELATED WORK

For complex traffic conditions, reinforcement learning methods can solve traffic congestion problems more effectively. The Q-learning [29] algorithm is a model-free and off-policy reinforcement learning algorithm, and it is very suitable for TCL. However, in Q-learning algorithm [5], all Q-values are stored in a table, the rows and columns of the table are states and actions respectively. When our states and actions are very large, the Q-table will be so large that the memory of computer cannot support such a large Q-table. Therefore, many researchers have applied the Deep Q-learning (DQN) algorithm [30]–[32] to TLC at a single intersection. And the Deep Recurrent Q-learning (DRQN) algorithm [33] has been proposed to solve traffic congestion under partially observable scenario. However, considering only one intersection in real world is definitely not enough. Since multiple intersections will affect each other, many researchers have proposed applying multi-agent DRL algorithm [19]–[22] to TLC, such as [26], [34]–[36].

The MARL algorithms used to solve traffic congestion at multiple intersections can be divided into three categories. The first category is fully isolated multi-agent reinforcement learning, which assumes that each agent is independent of each other, such as [37]. Only observing the local traffic conditions and controlling the traffic lights at local intersection is simple, but it lacks cooperation among agents. Thus, it is difficult to converge to the optimal joint policy. The second category

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

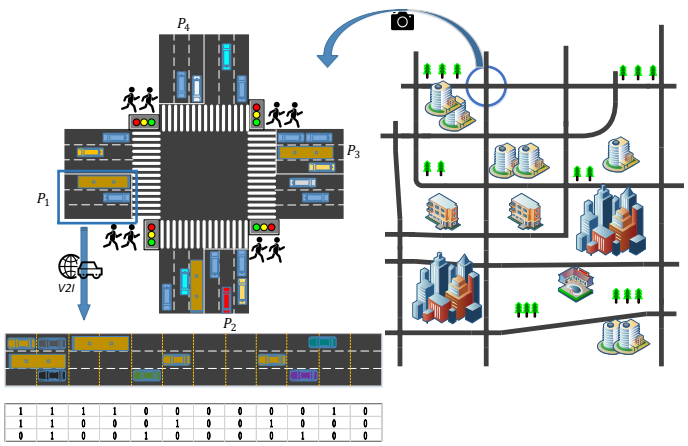


Fig. 1: The upper left part shows a four-way intersection in our model. The lower left part shows the corresponding position matrix on this road.

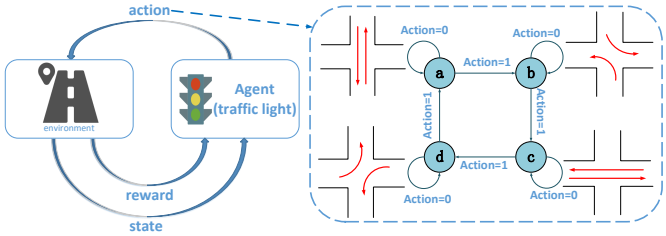


Fig. 2: The left part shows a deep reinforcement learning model for traffic light control in the left part. The right part shows the phase transition diagram.

is multi-agent reinforcement learning with partial state cooperation, such as [25]. This method often takes into account part of the state information of adjacent intersections, and combines the local traffic state and partial state information of neighboring agents to control local traffic lights. It will achieve better result than the first category. Only the partial state of the neighboring agent is considered and the intrinsic relationship between the actions is neglected. And most of the MARL algorithms are based on Q-learning, but the multi-agent Q-learning algorithms do not perform well in multi-agent scenarios. Moreover, each agent still uses the greedy algorithm to select action independently and the final policy may not converge to the balanced joint policy. The third category is multi-agent reinforcement learning of action joints, such as [24], which represents the actions of partial or all agents as a joint vector. Although this method fully considers the state of partial or all agents, which may lead to curse of dimensionality. Thus, this method is rarely used.

The state, reward, and action set in these works mentioned above are different. In general, there are two types of action settings, one is the fixed phase sequence [5], [30] and the other is the variable phase sequence [25], [24]. Although the variable phase sequence is more flexible, that will increase the possibility of traffic accidents caused by the uncertainty of the next phase, thus it is not suitable for the real world. In this paper, we apply advanced multi-agent deep reinforcement learning algorithm to control traffic lights of multiple intersections. Our work takes into account that the number of pedestrians in real traffic is also one of our optimization goals, in which buses should have higher priority than ordinary cars. In addition, we also define new states and rewards.

III. SYSTEM MODEL AND PROBLEM FORMULATION

In this paper, we apply deep reinforcement learning to control traffic lights in a vehicular network, thereby alleviating traffic congestion at multiple intersections. We collect traffic information at intersections via a vehicular network to control the traffic lights at each intersection according to different environments. As shown in Fig. 1, assume that each intersection is a four-way intersection, and each road contains three lanes. The innermost lane is the left-turn lane, the middle is the straight lane, and the outermost is the straight and right-turn lanes. Every vehicle passing through the intersection will be controlled by traffic lights. And there will be a fixed period of yellow light between the green light and red light to ensure traffic safety. The left part of Fig. 2 shows the basic deep reinforcement learning model. We control the traffic flows by adjusting the duration of the red and green lights to minimize traffic congestion, and we get a state and feedback from the environment after each adjustment of the traffic light phase. We use two deep neural networks to estimate the Q-value and select action respectively. Before each selection action, we need to synthesize various aspects of the state information. In the case of multiple intersections, we need to consider the global information to minimize total congestion at all intersections.

To reduce traffic congestion, the problem is how to make use of previous experience and the current intersection environment to make a reasonable decision on the duration of the red or green light. We deploy an agent at each intersection. Therefore, the problem can be defined as a multi-agent problem. We use "R" and "G" to indicate red and green lights respectively, and "E", "S", "W", and "N" to indicate East, South, West, and North respectively. For the situation that the congestion at different intersections is different and there will be interactions among the intersections, the coordination between intersections is important. Therefore, it is our ultimate goal to reduce global traffic congestion by coordinating multiple agents in combination with spatio-temporal relationships. The meaning of each notation is shown in Table I.

IV. DEEP REINFORCEMENT LEARNING MODEL

In order to solve traffic congestion in vehicular networks, we need to establish a model of deep reinforcement learning. We treat each intersection as an agent, and each agent only controls the local traffic light. First, we need to define the single intersection state s_t, agent action a_t and reward r_t at discrete time steps, $t = 0, 1, 2, ...$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

TABLE I: Notations

Notation	Meaning
Env	Environment
S_t	Globe state at time step t
$o_{t,i}$	Observation of agent i at time step t
$a_{t,i}$	Action of agent i at time step t
r_t	Reward at time step t
Δt	Time interval between actions
N	The number of agents
P_i	Position matrix of the i-th road
V_i	Velocity matrix of the i-th road
M	The number of pedestrains
L	State of traffic light
D	Queue length matrix
t_y	Yellow light duration
t_{max}	Maximum duration of each phase
W	Waiting time
d_j	Delay of vehicle j
N_c	Number of vehicles passed the intersection
C	Blink condition of traffic light

A. Intersection State

For a single intersection, we define the state using the location of the vehicle in each lane, the velocity of the vehicle, the phase of the traffic light, the queue length of each lane, and the number of pedestrians waiting to pass. To represent the location of each vehicle at the intersection, we divide each lane into discrete cells of equal length. The setting of the length of the discrete cell is also very important. If the setting is too long, the position information of the vehicle will be inaccurate. If the setting is too short, the state space will be too large. Thus we set the length of the discrete unit to be slightly larger than the average length of the ordinary vehicle. As shown in Fig. 1, we assume that vehicles other than buses are ordinary vehicles. Each cell can only accommodate one ordinary vehicle. One bus occupies two cells. We use the position matrix P to indicate the position of all the vehicles at the intersection. We put the position matrix of the road on the same line together, which is beneficial to exploit the correlation among the roads. Since there are 4 roads at each intersection, the matrix P is given by:

$$P = \begin{bmatrix} P_1 \\ P_3 \\ P_2 \\ P_4 \end{bmatrix}. \tag{1}$$

Matrix P_i represents the position matrix of the vehicles on the i-th road. If a vehicle is present at a cell, the corresponding position of the matrix is set to 1, otherwise, it is 0. Similarly, the velocity matrix V is given by:

$$V = \begin{bmatrix} V_1 \\ V_3 \\ V_2 \\ V_4 \end{bmatrix}. \tag{2}$$

Matrix V_i represents velocity matrix of the vehicles on the i-th road. If a vehicle is present at a cell, the corresponding position of the matrix is set to the velocity of the vehicle, otherwise, it is 0. For the number of pedestrians, we use a vector $M = (M_{SN}, M_{EW})$, where M_{SN} indicates the number of the pedestrians who are going to pass from south to north and from north to south, M_{EW} indicates the number of the pedestrians who are going to pass from west to east and from east to west. As shown in Table II, we use one-hot-coding of length 4 to indicate the traffic light phase L, which will be mentioned in the next subsection. Since there are three lanes in each direction and the direction in which each lane is allowed to pass is different. Thus, the length of the queue cannot be counted by naive averaging. The queue length matrix D can be expressed as:

TABLE II: The state of traffic light.

light				
SN-G	1	0	0	0
SNL-G	0	1	0	0
EW-G	0	0	1	0
EWL-G	0	0	0	1

$$D = \begin{bmatrix} D_{11} & D_{12} & D_{13} \\ D_{21} & D_{22} & D_{23} \\ D_{31} & D_{32} & D_{33} \\ D_{41} & D_{42} & D_{43} \end{bmatrix}. \tag{3}$$

In Equ. (3), D_{ij} represents the queue length on the j-th lane of the i-th road $(1 \leq i \leq 4, 1 \leq j \leq 3)$. We can collect vehicle position information and velocity information for each intersection via a vehicular network. And we can also utilize the snapshot of surveillance cameras deployed at intersections combined with CrowdCount [38] in the computer vision field to get the number of pedestrians. In summary, we define the intersection state as $s_t = (P, V, D, L, M)$ at the discrete time step t.

B. Agent Action

Each agent selects the action a_t at each time step and receive a new state S_{t+1}. According to traffic rules, vehicles can make a right turn whether it is a red or green light. Thus, we consider that there are only four phases of the traffic light: (a) N to S and S to N, (b) N to E and S to W (turn left), (c) E to W and W to E, (d) E to S and W to N (turn left). Pedestrians waiting to pass from north to south or from south to north can only pass through the intersection at phase (a), and pedestrians waiting to pass from east to west or from west to east can only cross the intersection at phase (c). As shown in the right part of Fig. 2, each phase lasts for a time step and is cycled in the fixed order of (a) (b) (c) (d). When the duration of a phase exceeds t_{max}, the phase of the next time step is forced to switch. Therefore, our action set is $A = \{0, 1\}$. If the agent chooses an action $a_t = 0$, it means that the current traffic light phase is kept unchanged. And the other choice means to switch the current traffic light phase to the next phase.

C. Reward

In order to make each agent to gradually learn to make the most reasonable decision on traffic lights, it is essential to choose the appropriate reward function as feedback of the action, for the reasonable reward has a guiding effect on each decision. The task of each agent is to minimize the congestion at the local intersection by taking appropriate policy. The

total time taken by the vehicle to enter the road until it passes through the intersection (i.e., vehicle travel time) is a significant metric of the traffic light control policy. However, if the total travel time is used as feedback, it may cause delayed reward, which is obviously unreasonable. Thus we can use other metrics instead of the vehicle travel time as the reward. Assume that when the vehicle velocity is less than v_{min}, the vehicle is considered as waiting. We define the following several rewards:

(1) The sum of the queue lengths D_{ij} of all the roads leading to the intersection, i.e., the total number of vehicles waiting for all roads. The reward R_D of queue length can be given by:

$$R_D = \sum_{i=1}^{4} \sum_{j=1}^{3} D_{ij}. \tag{4}$$

(2) The total waiting time for each vehicle on all roads at the intersection. When the driver is waiting, the driver's impatience is not proportional to the waiting time, that is, as the waiting time increases, the driver's impatience index rises. Therefore, in order to ensure the fairness of the vehicle and avoid a small number of vehicles waiting too long, we divide the waiting time of vehicles into three categories, that is, no waiting, waiting for less than 6 time steps and waiting for more than 6 time steps. The number of vehicles at the intersection is N_t, and $W_{j,t}$ indicates the cumulative waiting time of vehicle j at time step t, where $1 < j < N_t$. R_W indicates the reward of vehicle waiting time.

$$W_{j,t} = \begin{cases} W_{j,t-1} + \Delta t, \text{ vehicle speed } < v_{\min} \\ 0, \text{ other} \end{cases}, \tag{5}$$

$$r_{j,t} = \begin{cases} 0, W_{j,t} = 0 \\ -W_{j,t}, \Delta t \le W_{j,t} \le 6\Delta t \\ -2W_{j,t}, W_{j,t} \ge 7\Delta t \end{cases}, \tag{6}$$

$$R_W = \frac{1}{N_t} \sum_{1}^{N_t} r_{j,t}. \tag{7}$$

(3) The system delay represents the difference between the actual travel time of the vehicle and the travel time when the vehicle travels at the maximum velocity allowed. To simplify the expression, we can use the following equation to represent the system delay. d_n represents the delay of vehicle j, v_j represents the velocity of vehicle j, and v_{max} represents the maximum velocity allowed to arrive at the road. The reward R_s of system delay can be given by:

$$R_s = \frac{1}{N_t} \sum_{1}^{N_t} d_j = \frac{1}{N_t} \sum_{1}^{N_t} \left(1 - \frac{v_j}{v_{\max}}\right). \tag{8}$$

(4) The total number N_c of vehicles that passed the intersection in a time step Δt. The reward R_c of the vehicles passing the intersection can be given by:

$$R_c = N_c. \tag{9}$$

(5) We define the blinking condition of the traffic light as C. When $C = 1$, it indicates the current phase has changed. When $C = 0$, it indicates the current phase has not changed. The setting of C is to prevent the flickering of the traffic light, for the frequent flashing traffic lights may reduce vehicle throughput and increase the likelihood of a traffic accident.

(6) The total waiting time of pedestrians waiting to pass in all directions at the intersection. M_t indicates the number of pedestrians waiting to pass at time step t. The method of calculating $r_{k,t}$ is similar to Equ. (5) and Equ. (6), where $1 < k < M_t$. And the reward R_w of pedestrian waiting time can be given by:

$$R_w = \frac{1}{M_t} \sum_{1}^{M_t} r_{k,t}. \tag{10}$$

Different rewards have different importance. In other words, different rewards have distinct effects on each agent's policy. We multiply the above rewards by different weights and add them as our final reward. And we consider that buses and ordinary vehicles have different passing priorities, thus the total reward r is defined by the following equation:

$$r_t = W_1 * R_D + W_2 * R_W + W_3 * R_s + W_4 * R_c + (1+\alpha)(W_2 * R'_W + W_3 * R'_s + W_4 * R'_c) + W_5 * \mathrm{C} + W_6 * R_w, \tag{11}$$

where R_W, R_s, R_c represent the reward of ordinary vehicles, R'_W, R'_s, R'_c indicates the reward of the bus. And α is the weighting factor of the bus reward.

V. DEEP REINFORCEMENT LEARNING ALGORITHM FOR TRAFFIC SIGNAL CONTROL

A. The MARDDPG Algorithm

In this paper, we propose the MARDDPG algorithm to control traffic lights at multiple intersections, inspired by [19], [23]. The MARDDPG algorithm is the extension of the DDPG algorithm on the multi-agent condition. Considering that the traffic flow and people flow are time-continuous, we can utilize this time continuity to capture more potential information and the correlation among various state information. Experiments in [39] show that adding recurrency to the Q-network is beneficial to Q-network to obtain more potential information from the current and previous partial observations, so as to better estimate the Q-value.

Inspired by the DRQN algorithm [39] and RDPG algorithm [40] proposed by previous researchers, we propose a Multi-agent Recurrent Deep Deterministic Policy Gradient Algorithm (MARDDPG) to TLC, our algorithm model is shown in Fig. 3. In our proposed method, no additional exchange of information and the communication channel for each agent. In order to prevent the environment from being non-stationary due to the fact that each agent does not know the policies of other agents during training process, each agent utilizes global states and actions during training process so that the policy of the partner or competitor can be estimated. Therefore, each agent can adjust the local policy according to the estimated policy of other agents to achieve the global optimal policy. In our method, since each agent is distributed in different spatial locations, each agent learns a distinct policy. Therefore, each agent has its own actor network and critic network. During off-line training process,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

each critic network can receive the observation $o_{t,i}$ and action $a_{t,i}$ of other agents at time step t, thus the Q-function can be given by $Q_i^{\emptyset_i}\left(o_{t,1},\ldots,o_{t,N},a_{t,1},\ldots,a_{t,N}\right)$ for all agent $i\in\{1,...,N\}$. This solved the problem that the algorithm is difficult to achieve convergence caused by the non-stationary environment. Suppose there are N agents, and π_i represents the policy of agent i, then:

$$\begin{aligned}P\left(o_{t+1,i}|o_{t,i};Env\right)&=P\left(o_{t+1,i}|o_{t,i};o_{t,1},\ldots,o_{t,N},a_{t,1},\ldots,\right.\\&\left.a_{t,N},\pi_1,\ldots,\pi_N\right)\\&=P\left(o_{t+1,i}|o_{t,i};o_{t,1},\ldots,o_{t,N},a_{t,1},\ldots,\right.\\&\left.a_{t,N}\right)\\&=P\left(o_{t+1,i}|o_{t,i};o_{t,1},\ldots,o_{t,N},a_{t,1},\ldots,\right.\\&\left.a_{t,N},\pi_1',\ldots,\pi_N'\right)\ \text{for any}\ \pi_i\neq\pi_N'.\end{aligned}\tag{12}$$

However, each actor network only selects the action according to the local observation and its own policy $\mu_i^{\theta_i}:O\rightarrow A$:

$$a_{t,i}=\mu_i^{\theta_i}\left(o_{t,i}\right).\tag{13}$$

In this way, each agent can make independent decisions during on-line testing without critic network, namely centralized learning and distributed execution. During the training process, each critic updates the critic network by minimizing the private loss function $\mathcal{L}\left(\emptyset_i\right)$ parameterized by $\emptyset=\{\emptyset_1,\ldots,\emptyset_N\}$:

$$\mathcal{L}\left(\emptyset_i\right)=E_{s_t,a_t}\left[\left(Q_i^{\emptyset_i}\left(S_t,a_t\right)-y_t\right)^2\right],\tag{14}$$

where

$$y_t=r_t+\gamma Q_i^{\emptyset_i}\left(S_{t+1},a_t\right)|_{a_{t,i}=\mu_i^{\theta'_i}\left(o_{t+1,i}\right)},\tag{15}$$

$$\begin{aligned}S_t&=\{o_{t,1},\ldots,o_{t,N}\},\\a_t&=\{a_{t,1},\ldots,a_{t,N}\}.\end{aligned}$$

Each actor updates the actor network by maximizing the future expected cumulative rewards, i.e., updating the actor network target in the direction that maximizes the cumulative reward according to the objective function $J\left(\theta_i\right)$ parameterized by $\theta=\{\theta_i,\ldots,\theta_N\}$:

$$J\left(\theta_i\right)=E_{s_t,a_t}\left[Q_i^{\emptyset_i}\left(s_t,a_t\right)|_{a_{t,i}=\mu_i^{\theta_i}\left(o_i\right)}\right].\tag{16}$$

In order to effectively utilize the timing information, we extend the MADDPG algorithm [19] into the MARDDPG algorithm, which adds the LSTM to the critic network and the actor network. h_t^{critic} and h_t^{actor} are the historical information in the critic network and the actor network, respectively. Each agent can select the action according to the previous state information: $a_{t,i}=\mu_i^{\theta_i}\left(h_{t,i}\right)$, and the Q-function becomes $Q_i^{\emptyset_i}\left(h_{t,1}^{critic},\ldots,h_{t,N}^{critic},a_t\right)$. Similarly, the loss function $\mathcal{L}\left(\emptyset_i\right)$ in the critic network becomes:

$$\mathcal{L}\left(\emptyset_i\right)=E_{h_t,a_t}\left[\left(Q_i^{\emptyset_i}\left(h_{t,1}^{critic},\ldots,h_{t,N}^{criric},a_t\right)-y_{t,i}\right)^2\right],\tag{17}$$

where

$$\begin{aligned}y_{t,i}=r_{t,i}+\gamma Q_i^{\emptyset_i'}&\left(h_{t+1,1}^{critic},\ldots,h_{t+1,N}^{critic},\mu_1^{\theta_1'}\left(h_{t+1,1}^{actor}\right),\ldots,\right.\\&\left.\mu_N^{\theta_N'}\left(h_{t+1,N}^{actor}\right)\right).\end{aligned}\tag{18}$$

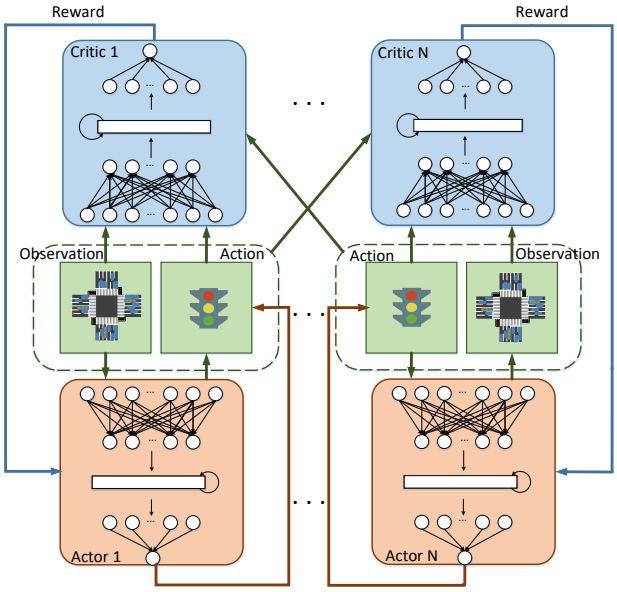


Fig. 3: MARDDPG-TLC algorithm of traffic light control for N intersections.

The objective function $J\left(\theta_i\right)$ and its gradient in the actor network become:

$$\begin{aligned}J\left(\theta_i\right)=E_{h_t,a_t}\left[Q_i^{\emptyset_i}\right.&\left(h_{t,1}^{critic},\ldots,h_{t,N}^{critic},\mu_1^{\theta_1}\left(h_{t,1}^{actor}\right),\ldots,\right.\\&\left.\left.\mu_N^{\theta_N}\left(h_{t,N}^{actor}\right)\right)\right|_{a_i=\mu_i^{\theta_i}\left(h_{t,i}^{actor}\right)}\Bigg].\end{aligned}\tag{19}$$

During the training process, we store the experience trajectory in the replay buffer. At each training step, a minibatch of episodes is randomly sampled from the replay buffer instead of the expectation in the above equation to update the critic networks and actor networks simultaneously. $\emptyset'$ and θ' are parameters in the target-critic network and target-actor network, respectively, we use "soft updates" to update target networks, like [41]. Then, our proposed MARDDPG algorithm of traffic light control for N intersections can be summarized as Algorithm 1. The detailed steps of the MARDDPG-TLC algorithm are as follows:

(1) The actor networks and critic networks initialize the parameterized policy $\mu_i^{\theta_i}\left(h_{t,i}^{actor}\right)$ and the parameterized value function approximation $Q_i^{\emptyset_i}\left(h_{t,1}^{critic},\ldots,h_{t,N}^{critic},a_{t,1},\ldots,a_{t,N}\right)$ respectively, all target networks are initialized with random weights θ_i' and $\emptyset_i'$ and the replay buffer is also initialized.

(2) In each episode, each agent selects action according to current policy $\mu_i^{\theta_i}\left(h_{t,i}^{actor}\right)$ and $a_{t,i}=\mu_i^{\theta_i}\left(h_{t,i}^{actor}\right)+N_t$, where N_t indicates exploration noise. Each agent receives feedback from the environment and new observation o_{t+1} after executing its own action. Then, the hidden layers of the LSTM in critic networks and actor networks are updated according to their respective rules.

(3) Store the current experience $\{o_{1,i},a_{1,i},r_{1,i},o_{2,i},a_{2,i},r_{2,i},\ldots\}$ in the replay buffer D as the data set for training actor networks and critic networks.

(4) For agent i, sample a random minibatch of S episodes $\left\{o_{1,i}^{j}, a_{1,i}^{j}, r_{1,i}^{j}, \ldots\right\}$ for all $i \in \{1, \ldots, N\}$ from replay buffer D to train actor networks and critic networks

(5) For each agent, from $t = T$ down to $t = 1$, for any sample $j \in S$, the critic estimates the Q-value aproximation $Q_i^{\emptyset_i}\left(h_{t,1}^{critic}, \ldots, h_{t,N}^{critic}, a_{t,1}, \ldots, a_{t,N}\right)$, and calculates the TD error and updates its parameters $\emptyset_i$ by minimizing the average loss function over the minibatch. Similarly, the actor calculates the policy gradient using the loss function, and the parameters θ_i of actor network are updated using the averaged gradient over the minibatch.

(6) Update the parameters $\emptyset_i'$ and θ_i' of the target networks using current $\emptyset_i'$ and θ_i', τ $(0 < \tau < 1)$ represents the rate at which the target network is updated according to the primary network.

B. Network Structure

The network structure of the actor network is shown in the Fig. 4, the input of the actor network is the local agent's observation $o_{t,i} = (P, V, D, L, M)$. Since P, V, D, and L are matrices of different dimensions or vectors of different lengths. The five metrics should first be processed differently to extract features. First, we combine the position matrix and velocity matrix as a two-channel image, and the image is fed to a stacked sub-network. The first convolution layer of the sub-network contains 32 filters, where the size of each filter is 4×4 applied with strides $(2, 2)$. The second convolution layer of the sub-network contains 64 filters, where the size of each filter is 2×2 applied with strides $(2, 2)$. Similarly, the queue length matrix D is fed to a convolution layer containing 8 filters, where the size of each filter is 2×2 applied with strides $(1, 1)$. We use two simple fully connected layers to encode the pedestrian vector M and traffic light phase vector L into a 8-dimensional vector respectively. Then, we concatenate the output of the above network into a vector. At last, we send this composite vector into the LSTM with 64 hidden units, and the output of the LSTM is fed to a fully connected layer with 16 output and a fully connected layer with two outputs. Finally, the last layer uses softmax as the activation function. Similarly, the network structure of the critic network is similar to the network structure of the actor network, but the input of the critic network includes not only the global states of all intersections but also the global actions of all agents. In the critic network, actions of all agents are concatenated with other feature vectors that the convolutional layers and the fully connected layers output as input to the LSTM. And the last layer is replaced by a fully connected layer with one output.

C. Parameter Sharing in Actor Networks

Each agent has a unique actor network and critic network, and there are multiple agents in our model. This will result in a lot of parameters that need to be trained in the entire model. In order to reduce the number of trainable parameters in the training process, speed up training process and reduce the memory footprint, we propose to share parameters [21] in the actor network or critic network of different agents. In general, sharing the parameters of all layers in the network will result in the same policy for each agent. The location and spatial relationship of each agent is different, thus the policy of each agent should not be identical. In the actor network and critic network, the lower layers are typically used to extract the features of the input state, while the higher layers are used to select the action and output the estimated Q-value.

Algorithm 1 Algorithm 1 MARDDPG-TLC for N intersections.

Initialize:

critic network $Q_i^{\emptyset_i}\left(h_{t,1}^{critic}, \ldots, h_{t,N}^{critic}, a_{t,1}, \ldots, a_{t,N}\right)$ and $\mu_i^{\theta_i}\left(h_{t,i}^{actor}\right)$ for all agent $i \in \{1, \ldots, N\}$ of traffic light control

target networks $Q_i^{\emptyset_i'}$ and $\mu_i^{\theta_i'}$

replay buffer D

for *episode* $= 1$ *to* M **do**

  Initialize empty history $h_{0,i}^{critic}, h_{0,i}^{actor}$

  Initialize observation o_0 of the local intersections and $t = 1$

  Initialize a random process $\mathcal{N}$ for action exploration

  while $t < T$ *and* $o_t \neq$ *terminal* **do**

    For each agent i of traffic light control, select action $a_{t,i} = \mu_i^{\theta_i}\left(h_{t,i}^{actor}\right) + \mathcal{N}_t$ according to current policy and exploration noise

    Execute action $a_t = (a_{t,1}, \ldots, a_{t,N})$ and receive reward r_t and new observation o_{t+1} of the local intersections

    $h_{t+1}^{\text{critic}} = \text{LSTM}\left(h_t^{\text{critic}}, a_t, o_{t+1}\right)$

    $h_{t+1}^{actor} = L\,\text{STM}\left(h_t^{actor}, o_{t+1}\right)$

    $t = t + 1$

  end while

  store episode $\{o_{1,i}, a_{1,i}, r_{1,i}, o_{2,i}, a_{2,i}, r_{2,i}, \ldots\}$ for all $i \in \{1, \ldots, N\}$

  for agent $i = 1$ *to* N **do**

    sample a random minibatch of S episodes $\left\{o_{1,i}^{j}, a_{1,i}^{j}, r_{1,i}^{j}, \ldots\right\}$ for all $i \in \{1, \ldots, N\}$ from replay buffer D

    for $t = T$ *down to 1* **do**

      set $y_{t,i}^{j} = r_{t,i}^{j} + \gamma Q_i^{\emptyset_i'}\left(h_{t+1,1}^{critic,j}, \ldots, h_{t+1,N}^{critic,j}, \mu_1^{\theta_1'}\left(h_{t+1,1}^{actor,j}\right), \ldots, \mu_N^{\theta_N'}\left(h_{t+1,N}^{actor,j}\right)\right)$

      update critic by minimizing the loss:

      $\mathcal{L}\left(\emptyset_i\right) = \frac{1}{s}\sum_j \left(Q_i^{\emptyset_i}\left(h_{t,1}^{\text{critic},j}, \ldots, h_{t,N}^{\text{critic},j}, a_{t,1}^{j}, \ldots, a_{t,N}^{j}\right) - y_{t,i}^{j}{}^2\right.$

      update actor using the sampled policy gradient:

      $\nabla_{\theta_i} J\left(\theta_i\right) \approx \frac{1}{S}\sum_j \nabla_{\theta_i}\mu_i^{\theta_i}\left(h_{t,i}^{actor,j}\right)\nabla_{a_i} Q_i^{\emptyset_i}\left(h_{t,1}^{critic,j}, h_{t,N}^{critic,j}, \mu_1^{\theta_1}\left(h_{t,1}^{actor,j}\right), \ldots, \mu_N^{\theta_N}\left(h_{t,N}^{actor,j}\right)\right)$

    end for

  end for

  update the target networks:

  $\emptyset_i' \leftarrow \tau\emptyset_i + (1-\tau)\emptyset_i'$

  $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$

end for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

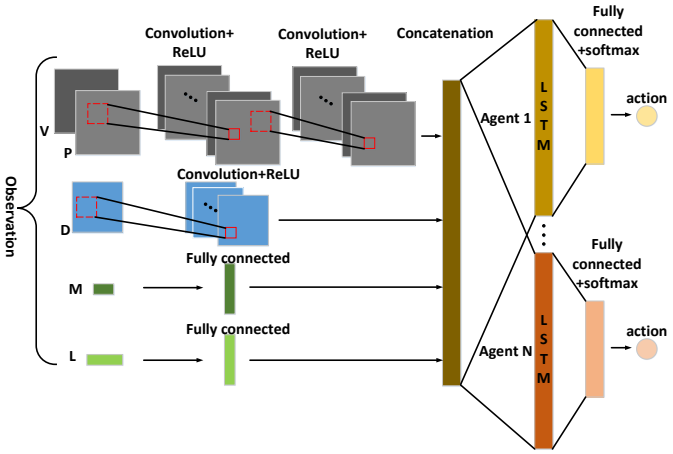


Fig. 4: The architecture of actor networks to select actions.

Therefore, the best option is to share the sub-networks of the actor networks or critic networks that is used to extract the features of each agent's state, so that each agent can do the same processing for input observation and can also develop a unique policy. We also need to consider whether to share parameters between all agents for both the actor network and critic network networks or just one of them. In the study [21], the experimental results show that in the case of sharing only the critic networks, it is more likely that all agents have similar policies. This is because critics has a significant guiding effect on actors. Therefore, we only share some layers between the actor networks in the experiment, as shown in the Fig. 4.

D. Optimal Policy

During training process, it is impossible for each agent to traverse all state spaces in a limited training episode, which may result in the agent not learning the optimal policy. Similar to the situation in the DQN algorithm, the agent only experiences limited intersection states. The agent may not be able to accurately estimate the Q-value of the unexperienced state. Therefore, we face a trade-off problem of exploration and exploitation, that is, use the best policy we have learned before or exploring the new and possibly better policy. We adopt a simple method to balance the exploration and exploitation. When selecting the action according to the current policy, we introduce a random noise $\mathcal{N}$ according to $a_{t,i} = \mu_i^{\theta_i}\left(h_{t,i}^{actor}\right) + \mathcal{N}_t$ to select action, which can increase the coverage of learning and explore more potential optimal policy. In this paper, we use ADAptive Moment estimation (Adam) [42] to optimize the critic networks and actor networks. The Adam algorithm has the advantages of both the AdaGrad and RMSProp algorithms. Adam not only calculates the adaptive parameter learning rate based on the first-order moment mean like the RMSProp algorithm, but also makes full use of the second-order moment mean of the gradient.

VI. EXPERIMENTAL RESULTS

In this section, we evaluate our method through simulation experiments and compare it with previous work. The simulation results verify the efficiency of our proposed method.

TABLE III: Model parameter setting.

Parameter	Value
v_{min}	0.2 m/s
v_{max}	16.67 m/s
Δt	5 seconds
t_y	3 seconds
t_{max}	90 seconds
γ	0.99
Sample size	64
Memory length	20000
Pre-training steps t_p	2000
Target network update rate τ	0.001
Learning rate	0.0001
Simulation times per episode	300 seconds
Optimizer	Adam

TABLE IV: Reward coefficients.

W_1	W_2	W_3	W_4	W_5	W_6	α
-0.25	0.2	-1	1	-4	-0.5	-0.5

A. Experiment and Parameter Setting

The evaluation is on a simulation platform conducted in SUMO [43] (Simulation of Urban Mobility). SUMO is a time-discrete, spatially continuous microscopic traffic simulation software that supports dynamic routing based on the right-side driving rules of road intersections. It also provides a visual graphical interface and supports multiple grid format inputs and various road network designs. Moreover, it can control the traffic lights at each intersection according to any given policy.

The shorter the distance between two intersections is, the stronger the correlation between them will be in the traffic network. Our experiments are carried out in two experimental scenarios, that is, a traffic network with two intersections and six (2×3) intersections with three lanes in each direction of the intersection. Each traffic light contains only four phases and cycles in a fixed phase sequence. Assume that each intersection scenario is a $480m \times 480m$ area. The length of the ordinary vehicle is 4 meters, the length of the bus is 8 meters and the minimum distance between vehicles is 1.5 meters. Thus we set the cell length c to 6 meters, thus the dimensions of the position matrix P_i and the velocity matrix V_i are both 3×40. The straight-going vehicles arrival rate for each road is 0.2 vehicles per second, and the left-turning and right-turning vehicles arrival rate both are 0.05 vehicles per second. This uneven traffic flow is close with the case in the real world where the flow rate of all through traffic and the turning traffic are not the same. The ratio between bus arrival rate and ordinary vehicle arrival rate is 1:10. And the destination of each vehicle on each lane is randomly set according to the discrete uniform distribution. The pedestrian arrival rate is set to 0.3 persons per second. The weights of various parameters and rewards are shown in Table III and Table IV.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

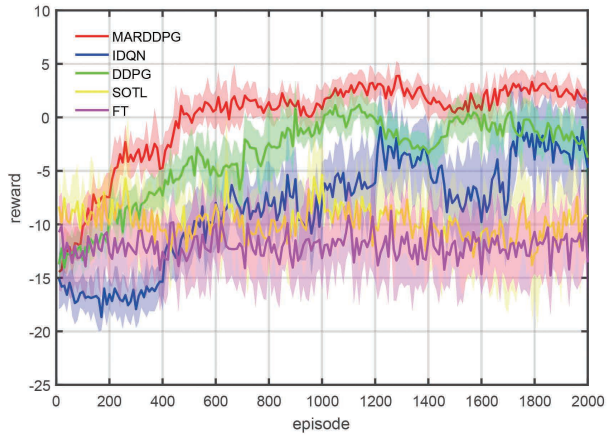


Fig. 5: The reward during all the training episodes in scenario 1.

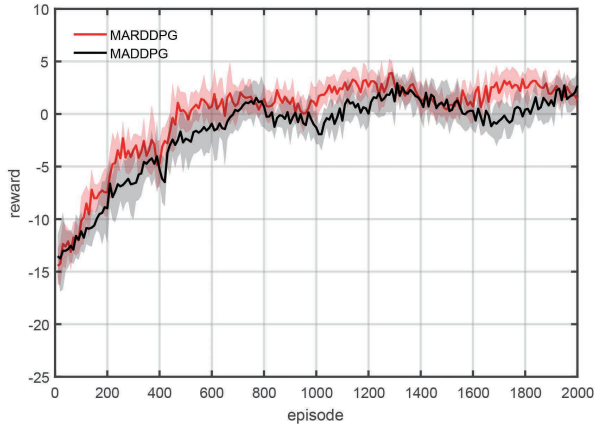


Fig. 6: The rewards of the MARDDPG and MADDPG algorithms during all the training episodes in scenario 1.

B. Compared Methods

To test the effectiveness of our model, we compare our method with the following baseline methods. All baselines use the same states, actions and rewards we defined above. And we assume that all vehicles are equipped with VANET technology.

1) Fixed-time Traffic Light Control (FT): Traffic lights are cycled in a fixed phase sequence [2]. The duration of each phase is set first and does not change with traffic flow. It is the one of the most widely used traffic control policy in real world.

2) Self-Organizing Traffic Light Control (SOTL) [44]: The states of traffic lights change directly according to the elapsed time and the number of vehicles in queues.

3) Independent Deep Q-learning (IDQN) [26]: Each intersection is controlled by an agent, and the DQN algorithm is applied to find the optimal traffic light control policy. Specifically, there is no exchange of information between each agent and the decision is made independently according to the greedy algorithm.

4) Deep Deterministic Policy Gradient (DDPG) [23]: All intersections are controlled by centralized traffic light controller and the DDPG algorithm is used to find the optimal traffic light control policy.

In addition, in order to verify the effectiveness of the MARDDPG algorithm that incorporates the LSTM, we also consider a variant of our model without the LSTM, the algorithm uses the same network structure, i.e., Multi-agent Deep Deterministic Policy Gradient (MADDPG) algorithm. Similarly, we also tested the method of removing parameter sharing and optimal policy to verify the effectiveness of parameter sharing and optimal policy. It is worth noting that the SOTL algorithm relies on processed traffic information such as the elapsed time and the number of vehicles in queues, while the DRL algorithms can take raw traffic information as input.

C. Evaluation Metric

We use the average reward over time and all intersections as our reward. This reward is the weighted sum of each metric.

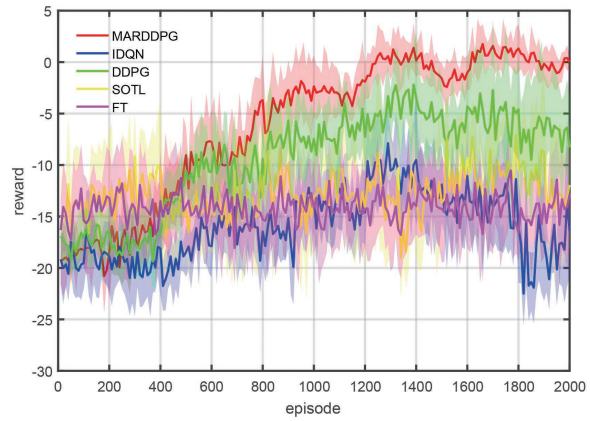


Fig. 7: The reward during all the training episodes in scenario 2.

A smaller reward means there are worse performance. In addition, for a more comprehensive evaluation of the proposed algorithm, we also use the average vehicle queue length, average travel time, average delay and average number of pedestrians during the last 100 episodes as our evaluation metrics. The average vehicle queue length refers to the average sum of the vehicle queue lengths of all approaching lanes over time and all intersections. The average travel time is the average total time it takes a vehicle to pass through each intersection. The average delay is the average sum of the delays of all approaching lanes over time and all intersections. The average number of pedestrians refers to the average sum of the number of pedestrians waiting to pass at the intersection over time and all intersections. The smaller the values of the three metrics indicate the better the performance of the algorithm.

D. Results in Scenario 1

1) Comparision with baseline: The simulation results are shown in Fig. 5. we down-sampled reward data collected from 2000 episode and drew 200 points in each training result figure. The solid line indicates the mean of reward, and shallowed area indicates the standard deviation. In the scenario

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

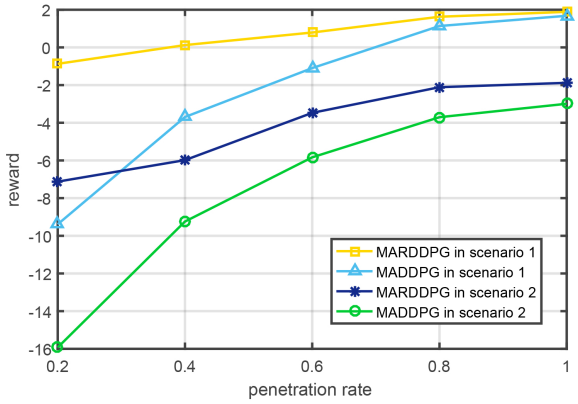


Fig. 8: The average reward under different penetration rates.

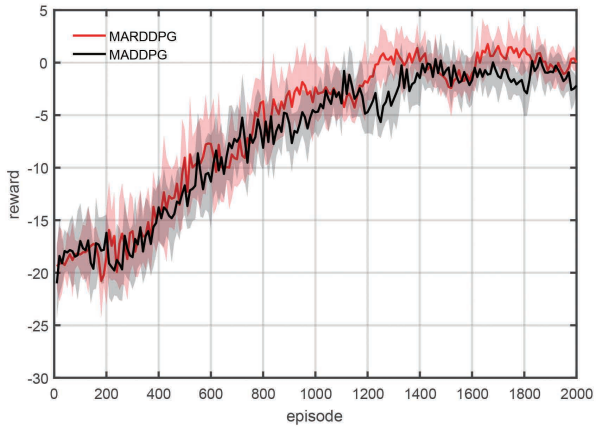


Fig. 9: The rewards of the MARDDPG and MADDPG algorithms during all the training episodes in scenario 2.

with only two intersections, We can see from the figure that the reward of the FT algorithm fluctuates around -12, and the reward does not increase with the increase of training episodes. Similarly, the reward of the SOTL algorithm fluctuates around -10, and the standard deviations of the rewards are greater. This is because the FT and SOTL algorithm cannot control the traffic lights in real time through continuous learning from environment.

The reward of the IDQN algorithm has an upward trend during the first 1200 episodes, but the fluctuations are large. After 1200 cycles, the reward shows a sharp downward trend, and the reward for each episode is very unstable. That is because the traffic lights at each intersection in the IDQN algorithm are only controlled according to the local traffic conditions, so it is difficult to ensure that all intersections can learn the optimal policy. The curve of the DDPG algorithm rises slowly before 1100 episodes, and the final reward fluctuates within (-4,1). This is because the centralized traffic light controller in the DDPG algorithm considers the traffic conditions of multiple intersections simultaneously to update the joint policy. Thus the algorithm can slowly converge. We can see that the MARDDPG algorithm can learn the fastest among the five algorithms, and the final reward is stable around 2. That means our algorithm reaches the best policy and faster than others. Therefore, the MARDDPG algorithm performs significantly better than the other four algorithms in reducing traffic congestion.

2) Comparision with variant of our proposed method: The comparison of the results of the MARDDPG and MADDPG algorithms are shown in Fig. 6. It can be seen from the figure that the general trend of the two curves is the same. When the training episodes over 600 times, the rewards of the two algorithms become more stable than previous episodes. However, the final reward of the MARDDPG algorithm is more stable and slightly higher than the MADDPG algorithm. This indicates that the adding LSTM enables agent to learn a better policy. The superscripts 1 and 2 represent scenario 1 and 2 respectively in Table VII. It can be seen from Table VII that after removing the parameter sharing, the total convergence time is increased by 500 episodes, and the average reward after convergence is also lower than the method without removing the parameter sharing. This shows that the use of parameter sharing can significantly accelerate the convergence time of our method. After removing the optimal strategy, the average reward is significantly reduced, which indicates that the optimal strategy may not be found. This is because the method after removing the optimal policy loses explorability and it is difficult to find the optimal policy.

Comprehensive evaluation metrics are shown in Table V. We can see from the datas in the table that the DDPG, MADDPG and MARDDPG algorithms performed well in the scenario 1 during the testing phase, for they are based on the framework of the actor-critic algorithm. In the MARDDPG algorithm, the average vehicle queue length is reduced by 22.08% and 5.79% of the average vehicle queue length of the DDPG and MADDPG algorithms, respectively. And the average travel time is reduced by 10.81% of the average travel time of DDPG. This proves that under our method, the total time it takes for a vehicle to cross an intersection is significantly reduced. The average pedestrian number is reduced by 1.55% and 1.15% of the average pedestrian number of the DDPG and MADDPG algorithms respectively. Moreover, the standard deviation is also the smallest.

3) Comparision with varying penetration rates: In the real-world environment, not all vehicles are equipped with VANET technology (wireless transceivers), and some agents may also fail during operation, so the traffic light controllers at each intersection may not be able to detect partial vehicle information. Thus, we evaluate our methods under different penetration rates. Penetration rate refers to the proportion of vehicles that are VANET enabled. The experimental results are shown in Fig. 8. Both algorithms performed well under the higher penetration rates in scenario 1, but the MADDPG algorithm performed significantly worse than the MARDDPG algorithm under the lower penetration rates. This is because the information of some vehicles is not available, making the environment become POMDP [45]. Since the traffic states have time continuity, historical information can be remembered in LSTM. Thus the MARDDPG algorithm has better robustness than the MADDPG algorithm under the low penetration rate in scenario 1.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

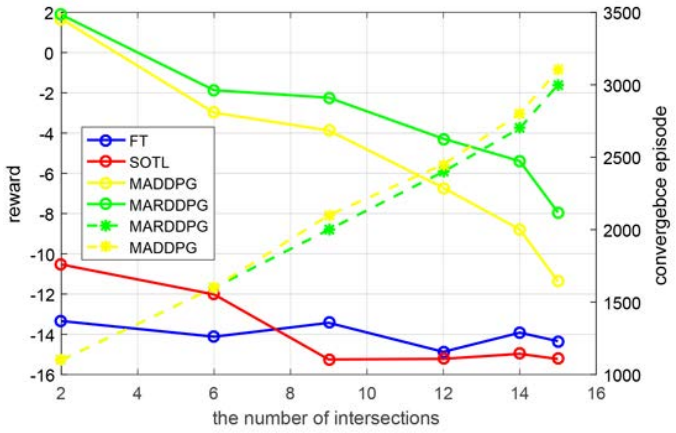


Fig. 10: The performance of our method at different numbers of intersections.

4) Comparision under peak hours: In this part, we evaluate our method by comparing the performance under high pedestrian rate. The peak hours mean that the vehicle arrive rate and pedestrian rate are increased. Therefore, we set the pedestrian arrival rate to 0.5 persons per second. And the straight-going vehicles arrival rate for each road is set to 0.3 vehicles per second, and the left-turning and right-turning vehicles arrival rate both are set to 0.1 vehicles per second. The other experimental parameters remain unchanged. The experimental result is shown in Fig. 11. It can be seen from the figure that the average reward of our algorithm is significantly higher than the other five algorithms. The average vehicle queue length, average number of pedestrians and average delay are at least 50%, 18% and 48% less than the other four algorithms respectively. That means that our algorithm can still show great results during peak hours in scenario 1.

E. Results in Scenario 2

1) Comparision with baseline: In scenario 2, the simulation results are shown in Fig. 7. The reward of the FT and SOTL algorithms fluctuate around -14 and -13 respectively, and the standard deviation is larger than that in scenario 1. The IDQN algorithm does not learn anything, for the changing environment causes the IDQN algorithm to not converge. The DDPG algorithm may converge after 2000 episodes, but the convergence speed is slow. Considering the limited computational cost, it was unnecessary to let it run indefinitely. Our MARDDPG algorithm gradually stabilizes after several fluctuations, which means that our algorithm finds the optimal policy after continuous trial and error. In addition, the reward of our algorithm is smaller than that in scenario 1 and the speed of convergence is slower, which indicates the optimal policy becomes harder to be learnt than scenario 1. However, our algorithm also performs significantly better than the other four algorithms across all episodes.

2) Comparision with variant of our algorithm: The comparison of the results of the MARDDPG and MADDPG algorithms is shown in Fig. 9. Starting from the 900th episode, the advantages of the MARDDPG algorithm are apparent. The reward of the MARDDPG algorithm starts to be higher and more stable than the MADDPG algorithm This is because the MARDDPG algorithm has memory characteristics and the useful historical information can be remembered. Therefore, in the traffic network of six intersections, the addition of the LSTM is beneficial to capture potential information. As shown in Table VII, the convergence time of our method is significantly shorter than the method without the parameter sharing, and the average reward of our method is significantly longer than the method without the parameter sharing or the optimal policy. Therefore, the addition of the parameter sharing and optimal policy can improve the experimental results.

As can be seen from Table VI, the average queue length, average delay, average travel time, and average number of pedestrians of the MARDDPG algorithm are lower than the corresponding values of the MADDPG algorithm by 4.68%, 9.57%, 3.46%, and 19.66% respectively. Since the traffic condition is more complicated, the standard deviations of these six algorithms are significantly larger than that in scenario 1, while it is also within the acceptance range. It is worth noting that the average reward for the MARDDPG algorithm is also lower than the average reward in scenario 1. Therefore, we also need to consider the impact of the number of intersections on the simulation results. Since each critic network takes the global state as input during training process, when there are more intersections, the state space will increase dramatically, which may lead to curse of dimensionality. Thus the MARDDPG algorithm may not be suitable for large-scale traffic network. Fortunately, for the medium-scale traffic network in the experiment, our algorithm can still handle the traffic congestion problem significantly.

3) Comparision with varying penetration rates: As shown in Fig. 8, the trend of fold line in scenario 2 are similar to that in scenario 1. As the penetration rate decreases, the MARDDPG algorithm is significantly more stable than the MADDPG algorithm. This means that the addition of LSTM allows our algorithm to run more stably in uncertain environments.

4) Comparision under peak hours: We use the same vehicle arrival rate and pedestrian arrival rate as that in scenario 1 under peak hours. The experimental results are shown in Fig. 11. Although the average reward of our algorithm is slightly lower than that of scenario 1, it still outperforms the other five algorithms. The average vehicle queue length is at least 50% less than the other algorithms. The average delay and average number of pedestrians are at least 23% and 50% less than the other algorithms respectively. This indicates that our algorithm can still stably learn an excellent policy under complex traffic conditions.

F. Scalability analysis

The performance of our method at two intersections and six intersections has been analyzed in the two subsections above. It can be seen from the results that our method shows excellent performance in both scenarios, but it is slightly worse in scenario 2. Therefore, we speculate that the effectiveness of our method may be affected by the scale of the traffic

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVT.2020.2997896, IEEE Transactions on Vehicular Technology

TABLE V: Comprehensive evaluation in scenario 1.

Algorithm	Reward	Queue length	Delay	No. of pedestrians	Travel time
FT	-13.34 ± 5.43	12.23 ± 8.15	3.09 ± 1.16	6.86 ± 5.83	46.12 ± 4.83
SOTL	-10.53 ± 8.75	10.27 ± 12.19	4.14 ± 1.19	6.25 ± 6.12	42.49 ± 7.23
IDQN	-4.84 ± 6.26	7.98 ± 7.21	3.57 ± 1.01	3.16 ± 4.93	49.63 ± 9.34
DDPG	-3.03 ± 4.74	5.36 ± 5.98	3.19 ± 1.01	3.19 ± 4.00	45.88 ± 8.47
MADDPG	1.67 ± 3.29	4.09 ± 5.76	**2.47** ± 0.89	1.17 ± 3.93	**40.79** ± 6.89
MARDDPG	**1.88** ± 3.19	**3.04** ± 5.19	2.59 ± 0.89	**1.13** ± 3.88	40.92 ± 5.16

TABLE VI: Comprehensive evaluation in scenario 2.

Algorithm	Reward	Queue length	Delay	No. of pedestrians	Travel time
FT	-14.12 ± 6.19	13.23 ± 9.13	3.80 ± 1.35	8.37 ± 6.09	50.35 ± 7.46
SOTL	-12.02 ± 8.97	11.76 ± 10.05	4.61 ± 1.21	7.10 ± 7.27	59.72 ± 12.68
IDQN	-16.06 ± 9.78	15.26 ± 12.89	4.02 ± 1.33	9.77 ± 9.56	68.56 ± 16.13
DDPG	-8.29 ± 7.63	9.89 ± 8.84	3.67 ± 1.38	5.24 ± 6.31	54.04 ± 10.76
MADDPG	-2.99 ± 4.84	6.93 ± 6.89	3.62 ± 1.12	2.92 ± 4.81	48.24 ± 12.55
MARDDPG	**-1.87** ± 3.82	**4.32** ± 5.87	**3.27** ± 1.20	**1.82** ± 4.29	**46.57** ± 9.27

TABLE VII: Comparision with variant of our proposed method.

Method	Reward[1]	Convergence episode[1]	Reward[2]	Convergence episode[2]
Ours	1.88	1100	-1.87	1600
Without parameter sharing	1.14	1600	-3.02	>2000
Without optimal policy	-4.79	1300	-13.25	1900

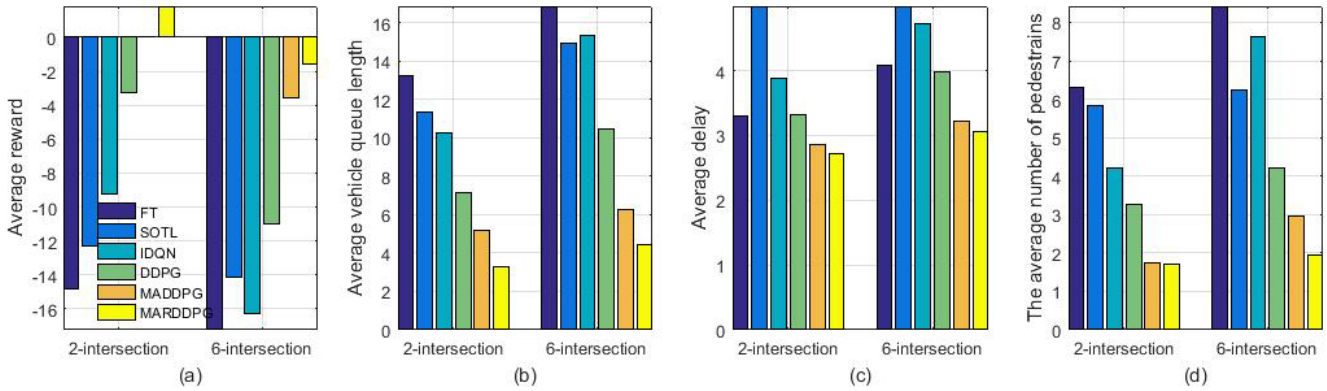


Fig. 11: The comparison of each evaluation metric under peak hours.

network. We performed the same tests on 9 intersections, 12 intersections, and 14 intersections, 15 intersections, and 16 intersections using the same environmental configuration as before, that is, the traffic flow is the same as that in scenario 1 and scenario 2 with the penetration rate of 0.8.

As shown in Fig. 10, we show the time it takes for our method to converge in different scenarios and the average reward after convergence. The dashed line in the figure indicates the number of episodes required for the method to converge, and the solid line indicates the average reward after convergence. The polyline of the IDQN and DDPG algorithms is not shown in the figure because the IDQN and DDPG algorithms cannot converge in scenarios where the number of intersections is greater than or equal to 6. It is experimentally obtained that our method cannot converge in the scenario of 16 intersections, so we only draw the scenario where the number of intersections is less than 16. Our method can converge in scenarios with less than 16 intersections, and the reward after convergence is still higher than the other two algorithms. This proves that our method has a good effect on the medium-scale (less than 16) traffic network. However, with the increase of traffic scale (more than 15), our method may not be able to handle. This is because as the scale of the traffic network increases, the state space and behavior space of each agent input will be larger, and the learning time will be longer. Moreover, when there are too many intersections, a single agent will consider the state of all other intersections. However, the state of intersections far from the local intersections does not actually have a greater impact on the policy of the local intersections, so too much useless information will lead to lower learning efficiency for each agent. All in all, our algorithm is more suitable for medium-scale (less than 16) traffic networks.

VII. Conclusion

In this paper, we propose the MARDDPG algorithm to reduce traffic congestion at multiple intersections. We utilize various road information based on vehicular networks' data collection to change the phase of multiple traffic lights in real time. The traffic light controller at each intersection is not isolated, for it can observe the global state during the training process. Each traffic light controller can estimate the traffic light control policies of other intersections when making decisions, thereby adjusting the local traffic light control policy to make optimal decisions. Moreover, in our TLC system, we also take pedestrian and bus into consideration, which make the whole system more humanized. Scalability analysis shows that our method is more suitable for medium-scale traffic networks. It can be concluded from the simulation results on SUMO that our algorithm can be applied to the TLC at multiple intersections with complicated road conditions, and can achieve good results even in a partially observable environment.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61972448, 61872150, 61872049 and 61902445. It was also supported in part by Guangdong Basic and Applied Basic Research Foundation under Grant No.2020A1515011209. It was also supported by Fundamental Research Funds for the Central Universities of China (19lgpy222), and Natural Science Foundation of Guangdong Province of China (2019A1515011798).

References

[1] L. Mussone, S. Grant-Muller, and J. Laird, "Sensitivity analysis of traffic congestion costs in a network under a charging policy," *Case Studies on Transport Policy*, vol. 3, no. 1, pp. 44–54, 2015.

[2] F. V. Webster, "Traffic signal settings," Tech. Rep., 1958.

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[4] H. Hartenstein and L. Laberteaux, "A tutorial survey on vehicular ad hoc networks," *IEEE Commun. Mag.*, vol. 46, no. 6, pp. 164–171, 2008.

[5] X. Liang, X. Du, G. Wang, and H. Zhu, "Deep reinforcement learning for traffic light control in vehicular networks," 2018.

[6] W. Liu, G. Qin, Y. He, and F. Jiang, "Distributed cooperative reinforcement learning-based traffic signal control that integrates v2x networks' dynamic clustering," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8667–8681, 2017.

[7] P. Hunt, D. Robertson, R. Bretherton, and M. C. Royle, "The scoot on-line traffic signal optimisation technique," *Traffic Engineering & Control*, vol. 23, no. 4, 1982.

[8] P. Lowrie, "The sydney cooridinated adaptive traffic (scat) system-principles, methodology, algorithm," in *Proc. of International Conference on Road Traffic Signaling*. IEEE, 1982, pp. 67–70.

[9] N. H. Gartner, *OPAC: A demand-responsive strategy for traffic signal control*, 1983, no. 906.

[10] S. Sen and K. L. Head, "Controlled optimization of phases at an intersection," *Transportation science*, vol. 31, no. 1, pp. 5–17, 1997.

[11] E. Bingham, "Reinforcement learning in neurofuzzy traffic signal control," *European Journal of Operational Research*, vol. 131, no. 2, pp. 232–241, 2001.

[12] D. Srinivasan, M. C. Choy, and R. L. Cheu, "Neural networks for real-time traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 261–272, 2006.

[13] J. J. Sánchez-Medina, M. J. Galán-Moreno, and E. Rubio-Royo, "Traffic signal optimization in "la almozara" district in saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 132–141, 2010.

[14] Z. Liu, "A survey of intelligence methods in urban traffic signal control," *IJCSNS International Journal of Computer Science and Network Security*, vol. 7, no. 7, pp. 105–112, 2007.

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

[16] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[17] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[18] 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, p. 529, 2015.

[19] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, 2017, pp. 6379–6390.

[20] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[21] M. J. Hausknecht, "Cooperation and communication in multiagent deep reinforcement learning," Ph.D. dissertation, 2016.

[22] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," *arXiv preprint arXiv:1802.05438*, 2018.

[23] N. Casas, "Deep deterministic policy gradient for urban traffic light control," *arXiv preprint arXiv:1703.09035*, 2017.

[24] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1140–1150, 2013.

[25] K. Prabuchandran, H. K. AN, and S. Bhatnagar, "Multi-agent reinforcement learning for traffic signal control," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 2529–2534.

[26] M. Liu, J. Deng, M. XU, X. Zhang, and W. Wang, "Cooperative deep reinforcement learning for traffic signal control," 2017.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] Y. Liu, L. Liu, and W.-P. Chen, "Intelligent traffic light control using distributed multi-agent q learning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–8.

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

[30] 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*. ACM, 2018, pp. 2496–2505.

[31] L. Prashanth and S. Bhatnagar, "Reinforcement learning with function approximation for traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 412–421, 2011.

[32] W. Genders and S. Razavi, "Using a deep reinforcement learning agent for traffic signal control," *arXiv preprint arXiv:1611.01142*, 2016.

[33] J. Zeng, J. Hu, and Y. Zhang, "Adaptive traffic signal control with deep recurrent q-learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1215–1220.

[34] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

[35] M. Schutera, N. Goby, S. Smolarek, and M. Reischl, "Distributed traffic light control at uncoupled intersections with real-world topology by deep reinforcement learning," *arXiv preprint arXiv:1811.11233*, 2018.

[36] X.-Y. Liu, Z. Ding, S. Borst, and A. Walid, "Deep reinforcement learning for intelligent transportation systems," *arXiv preprint arXiv:1812.00979*, 2018.

[37] M. Wiering, "Multi-agent reinforcement learning for traffic light control," in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, 2000, pp. 1151–1158.

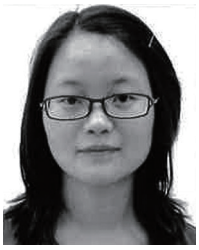
[38] D. B. Sam, S. Surya, and R. V. Babu, "Switching convolutional neural network for crowd counting," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 4031–4039.
[39] M. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," in *2015 AAAI Fall Symposium Series*, 2015.
[40] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," *arXiv preprint arXiv:1512.04455*, 2015.
[41] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[43] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner, "Sumo (simulation of urban mobility)-an open-source traffic simulation," in *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)*, 2002, pp. 183–187.
[44] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in applied self-organizing systems*. Springer, 2013, pp. 45–55.
[45] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine learning proceedings 1994*. Elsevier, 1994, pp. 157–163.

Tong Wu is currently working toward the master degree at the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China, working with Prof. P. Zhou. Her research interests include deep learning, deep reinforcement learning, and mobile edge computing.

Pan Zhou (S'07–M'14) is currently an associate professor with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Wuhan, P.R. China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology (Georgia Tech) in 2011, Atlanta, USA. He received his B.S. degree in the Advanced Class of HUST, and a M.S. degree in the Department of Electronics and Information Engineering from HUST, Wuhan, China, in 2006 and 2008, respectively. He held honorary degree in his bachelor and merit research award of HUST in his master study. He was a senior technical member at Oracle Inc., America, during 2011 to 2013, and worked on Hadoop and distributed storage system for big data analytics at Oracle Cloud Platform. He received the "Rising Star in Science and Technology of HUST" in 2017. His current research interest includes: security and privacy, big data analytics and machine learning, and information networks.

Kai Liu (S'07–M'12) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2011. He is currently an Assistant Professor with the College of Computer Science, Chongqing University, Chongqing, China. From 2011 to 2014, he was a Visiting Scholar with the Computer Science Department, University of Virginia, Charlottesville, VA, USA, and a Post-Doctoral Fellow with Singapore Nanyang Technological University, Singapore, and the City University of Hong Kong, and Hong Kong Baptist University, Hong Kong. His current research interests include Internet of Vehicles, mobile computing, and pervasive computing.

Yali Yuan received the M.Sc. degree from the University of Lanzhou, Lanzhou, China in 2015, and the Ph.D. degree from the University of Göttingen, Germany in 2018, where she is currently working as a Postdoctoral Fellow. Apart from the individual research work, she is also responsible of preparing project applications, mentoring Ph.D. candidates and bachelor/master students, teaching, conducting seminars and handling industrial partners. Her research interests include various topics related to wireless networks and security.



Xiumin Wang received the B.S. from Anhui Normal University, Shanghai, China, in 2006, and a joint Ph.D. degree from the University of Science and Technology of China (USTC), Hefei Shi, China, and the City University of Hong Kong (Cityu), Kowloon, Hong Kong, in 2011. She did her Postdoctoral at Singapore University of Technology and Design from 2011 to 2012. She is an Associate Professor in the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China. Her research interests include mobile computing, wireless networks, and algorithm design and optimization.

Huawei Huang (M'16) received his Ph.D in Computer Science and Engineering from the University of Aizu, Japan. He is currently an associate professor at School of Data and Computer Science, Sun Yat-Sun University, China. His research interests mainly include SDN/NFV, edge computing, and blockchain system optimization. He received the best paper award of TrustCom-2016. He used to be a visiting scholar at the Hong Kong Polytechnic University (2017-2018), a post-doctoral research fellow of JSPS (2016-2018), an assistant professor at Kyoto University, Japan (2018-2019). He is a member of the IEEE.

Dapeng Wu (S'98-M'04-SM06-F'13) received Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003. He is a professor at the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, and information and network security.