IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 11 November 2023, accepted 14 December 2023, date of publication 21 December 2023,
date of current version 4 January 2024.

Digital Object Identifier 10.1109/ACCESS.2023.3345343

RESEARCH ARTICLE

Large-Scale Traffic Signal Control Based on Multi-Agent Q-Learning and Pressure

LIANG QI, (Member, IEEE), YUANZHEN SUN, AND WENJING LUAN, (Member, IEEE)
College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, Shandong 266590, China

Corresponding author: Wenjing Luan (wenjingmengjing@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61903229, and in part by the Natural Science Foundation of Shandong Province under Grant ZR2019BF004.

ABSTRACT Traffic congestion has become an increasingly severe problem and needs to be solved urgently. Reinforcement learning (RL) is an advanced data-driven approach for large-scale adaptive traffic signal control (ATSC) in the complex urban traffic network. The decentralized multi-agent RL (MARL) framework is feasible for multi-intersection scenarios. However, the environment becomes partially observable and nonstationary without effective communication among agents from the perspective of each local agent. Previous studies solve this problem by improving the observation of each local agent and using a spatial discount factor. In this paper, we are motivated by achieving coordination among intersections in the RL settings and better performance in ATSC and solve the problem of traffic signal control at multiple intersections in large-scale networks based on RL techniques. Pressure is used as an indicator to represent the traffic conditions of an intersection considering its neighbor intersections. A new reward function with pressure and vehicle waiting time is designed to achieve coordination among intersections. The above three control policies are innovatively applied to a multi-agent Q-learning method for ATSC at multiple intersections. Experiments are carried out on a large synthetic traffic grid under simulated peak-hour traffic dynamics. Results demonstrate that our approach outperforms the state-of-the-art decentralized MARL algorithms in average queue length and intersection delay.

INDEX TERMS Adaptive traffic signal control, multi-agent reinforcement learning, max pressure control, Q-learning.

I. INTRODUCTION

In recent years, the number of vehicles has risen dramatically, and traffic congestion has become a severe problem in urban areas. In addition to its negative effects on travel efficiency and economic growth, traffic congestion also causes increased safety risks and environmental pollution. Alleviating traffic congestion is a long-standing hot research topic in urban traffic control. As one of the most widely used methods, traffic signal control (TSC) [1] can effectively control traffic, ensure safety at road intersection, and alleviate congestion in urban areas. According to the scope of intersections to be controlled, TSC methods can be divided into ''spot'' control for a single intersection [2], ''line'' control for an arterial road [3], and ''plane'' control for a large-scale network [4]. Due to the complexity and uncertainty of urban traffic, ''plane'' control is the most challenging problem.

Classical TSC methods can be divided into fixed-time control and dynamic control. Fixed-time control [5] performs the signal phase with a fixed duration in a specific sequence. It can adapt to all types of intersections but cannot dynamically adjust signal phases according to the traffic conditions at an intersection. Dynamic control, such as the self-organizing traffic lights [6], maintains or changes the current phase according to the vehicle queue length and waiting time at an intersection. Varaiya [7] introduces the concept of pressure at an intersection, which describes the difference between the queuing vehicles in the incoming lanes and outgoing lanes

The associate editor coordinating the review of this manuscript and approving it for publication was Jiachen Yang.

and represents the imbalance of vehicles among intersections. By calculating the pressure at each phase, this approach selects one with the maximum pressure as the next serving phase. However, these methods mainly focus on a single intersection. Large-scale cooperative control of traffic signals cannot be achieved.

Recently, reinforcement learning (RL) has become an essential technique for TSC [8]. One significant advantage of RL is its ability to fit a parametric model directly, learn from interactions with a complex environment, and achieve optimal control without relying on heuristic assumptions or equations. Neural networks are used to realize these functions [25]. Previous research has shown that RL performs better than traditional traffic control methods in TSC. Wei et al. [9] introduce IntelliLight, a novel controller based on Deep Q-Network (DQN), and conduct experiments on a real-world, and large-scale traffic scenario. They also provide an interpretation of the reward function, which includes multiple components such as queue length and average waiting time. Tan et al. [10] propose a cooperative deep RL framework to tackle traffic congestion reduction problem. It contains multiple regional agents and a centralized global agent. Each regional agent learns its policy and value function in a small region. The centralized global agent then hierarchically aggregates the RL results from different regional agents and forms the final Q-function over the entire large-scale traffic grid. Chen et al. [11] address the challenge of multi-intersection TSC using RL techniques. They overcome scalability, signal coordination, and data feasibility challenges by utilizing a pressure to achieve signal coordination, implementing individual control agents with a thoughtfully designed reward system, and conducting extensive experiments on multiple scenarios.

The most popular decentralized multi-agent RL (MARL) algorithm is independent Q-learning (IQL) [12]. Each agent trains independently and dynamically updates its policy. Since each agent treats the other agents as part of the environment, achieving global convergence is complicated. From the perspective of each agent, the environment is partially observable and nonstationary. To solve this problem, Tesauro [13] directly takes the parameter of Q networks of other agents into account, and the work [14] contains a low dimension of fingerprints, i.e., the exploration rate of the behavior policy and the number of iterations. Some research [26], [27] on event-triggered multi-agent communication utilizes a fully distributed way with intermittent communication. A recent study [4] stabilizes the training process by improving the observation of each agent and using a spatial discount factor. Specifically, the states and policies of the neighbor agents are incorporated into the local agent. A spatial discount factor is introduced to reduce the impact of the remote agent on the local agent. These two control strategies are applied to the independent advantage actor-critic (IA2C) algorithm and achieve great performance in adaptive traffic signal control (ATSC).

Motivated by achieving coordination among intersections in the RL settings and better performance in ATSC, these two control strategies in [4] are applied to the IQL algorithm, which improves the robustness and stationary state of the algorithm. Meanwhile, a well-designed reward function is significant for RL performance. Pressure is used as an indicator to represent an intersection's traffic conditions considering its neighbor intersections. Inspired by [3], a new reward function that includes both the pressure at the intersection and the waiting time of the vehicle is designed to instruct the agent to maximize the traffic throughout and achieve coordination among intersections. We refer to the improved IQL as the multi-agent Q-learning (MAQL). Experiments are carried out in a large synthetic traffic grid with high traffic flow density. Simulation results indicate that the proposed algorithm can achieve optimal performance with the new reward function. Compared with other state-of-the-art decentralized MARL algorithms, our algorithm shows superior performance in average queue length and intersection delay. Ablation experiments show that the new reward function can indeed improve the RL performance in ATSC.

The main contributions of this work are as follows:

1) In the multi-intersection scenario, two control policies in [4] are applied to the IQL algorithm, which stabilizes the training process and improves the robustness and stationary state of the algorithm.

2) A new reward function with pressure and vehicle waiting time is proposed in the RL settings to achieve coordination among intersections. Ablation experiments illustrate the effectiveness of introducing pressure into the reward function for large-scale ATSC problems.

3) Experiments on a large-scale synthetic traffic grid demonstrate that our proposed method outperforms other state-of-the-art decentralized MARL algorithms in average queue length and intersection delay.

The organization of the paper is as follows. Section II shows the basic knowledge of RL and the concept of pressure. Section III describes the details of the proposed MAQL algorithm. Section IV introduces the RL settings for ATSC. Experiment results are represented in Section V. Section VI concludes the paper.

II. PRELIMINARIES

A. REINFORCEMENT LEARNING

RL is based on Markov decision process (MDP) and learns to maximize long-term returns. In a fully observable environment, an agent i observes a state $s_t \in \mathcal{S}$ at time t and performs an action $a_t \in \mathcal{A}$ according to a policy function $\pi(a|s)$. Then the next state of the agent is s_{t+1} according to a state transition probability function $p_t(s_{t+1}|s_t, a_t)$, and the environment gives a reward $r_t = r(s_t, a_t, s_{t+1})$. Suppose that an episode has n steps, the agent obtains a return $U_t = \sum_{\tau=t}^{n} \gamma^{\tau-t} r_\tau$ at time t, where $\gamma \in [0, 1)$ is a discount factor. An action-value function, i.e., Q function $Q_\pi(s_t, a_t)$, is the expectations of return U_t for future states and actions at

all times,

$$Q_\pi(s_t, a_t) = E_{S_{t+1}, A_{t+1}, \ldots, S_n, A_n}[U_t | S_t = s_t, A_t = a_t] \quad (1)$$

where S_t and A_t are random variables and denote the state and action of the future moment, respectively. From (2) - (4), the optimal action-value function $Q_*(s_t, a_t)$ generates an optimal policy function π^* to obtain the optimal action a_t at time t. The optimal Bellman equation calculated by (5) is solved iteratively by dynamic programming operation to obtain the optimal action-value function.

$$Q_*(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t) \quad (2)$$

$$\pi^* = \operatorname*{argmax}_\pi Q_\pi(s_t, a_t) \quad (3)$$

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} Q_*(s_t, a) \quad (4)$$

$$Q_*(s_t, a_t) = R_t + \gamma \sum_{S \in \mathcal{S}} p_t(S_{t+1} | s_t, a_t) \max_{A \in \mathcal{A}} Q_*(S_{t+1}, A) \quad (5)$$

In practice, the agent does not know R_t and S_{t+1} and can only obtain the optimal Q function by estimation and Monte Carlo approximation. RL training is based on the experience $e_t = (s_t, a_t, r_t, s_{t+1})$ in a data-driven way.

The Q-learning algorithm is one of the most basic RL methods. Q function is fitted by the model Q_θ with parameters, such as Q table [28], linear regression (LR) [15], and deep neural network (DNN) [16]. According to the behavior policy, the agent collects the training data and stores it in the experience replay buffer in the quadruple form of $e_t = (s_t, a_t, r_t, s_{t+1})$. We use a common behavior policy named ϵ-greedy policy (6). It has the probability of ϵ for uniform sampling an action from $\mathcal{A}$ and probability of 1-ϵ for choosing an action with maximum Q value.

$$a_t = \begin{cases} \operatorname*{argmax}_a Q_\pi(s_t, a), \textit{with probability} 1 - \epsilon \\ \textit{a random action, with probability } \epsilon \end{cases} \quad (6)$$

In the DQN algorithm, a target network is used to mitigate the bias caused by bootstrapping, and the temporal difference (TD) algorithm trains the agent, calculates the TD target $y_t = r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-)$ and TD error $\delta_t = Q(s_t, a_t; \theta) - y_t$. The loss function is

$$\mathcal{L}(\theta) = \frac{1}{2|B|} \sum_{e_t \in B} (Q_\pi(s_t, a_t; \theta) - y_t)^2 \quad (7)$$

where each minibatch $B = \{(s_t, a_t, r_t, s_{t+1})\}$ contains the empirical trajectory of the agent. The return is estimated by $\hat{R}_t = \sum_{\tau=t}^{t_B-1} \gamma^{\tau-t} r_\tau$, where t_B is the last step in minibatches. Other techniques can better estimate the Q function, such as double DQN [17], dueling DQN [18], and prioritized experience replay [19].

B. PRESSURE

The concept of pressure is explained as follows. In an intersection, a traffic movement is defined as the process of a vehicle entering the intersection from an incoming lane to exiting the intersection from an outgoing lane. We use (l, m) to represent a traffic movement, where l represents the incoming lane and m denotes the outgoing lane. The signal phase of a traffic light is composed of a set of allowed traffic movements. We use $\mathcal{P}_i$ to represent the set of all phases of intersection i. For each signal phase $\mathcal{P}$, several traffic movements are allowed. For traffic movement (l, m) in a signal phase, we use $x(l, m)$ to represent the difference between the number of vehicles in lane l and lane m. The pressure of a signal phase is the total sum of the pressure of the allowed traffic movement, i.e., $pressure(\mathcal{P}) = \sum_{(l,m) \in p} x(l, m)$. The pressure of an intersection is denoted as the difference between the queuing vehicles in all incoming lanes and outgoing lanes. The pressure of an intersection [3] is denoted as the absolute value of difference between all the queuing vehicles in the incoming lanes and outgoing lanes. As shown in Fig. 1, yellow vehicles indicate all the queuing vehicles in the incoming lanes, and red vehicles represent all the queuing vehicles in the outgoing lanes. The pressure of the middle intersection is 4.

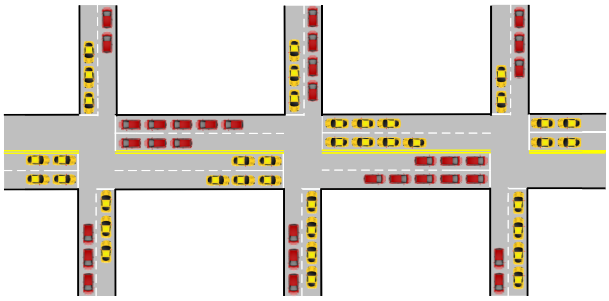

FIGURE 1. Illustration of pressure.

III. MULTI-AGENT Q-LEARNING

In this section, we first introduce the basic concepts and formulas of IQL. Then, the formulaic expression of MAQL is presented. Two control strategies are used to stabilize MAQL training. The first strategy uses the policy for stabilizing IQL in [14], i.e., to inform the local agent of the current policies of other agents. The second one considers the spatial distance of traffic lights and introduces a spatial discount factor to reduce the influence of remote traffic signals on local traffic lights so that local agent can concentrate more on improving local traffic conditions. Therefore, global TSC can converge more stably.

A. INDEPENDENT Q-LEARNING

In a multi-agent network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, agents i and j are neighbor if there is an edge connecting them. The set of all neighbors of agent i is denoted by $\mathcal{N}_i$, and a local region is denoted by $\mathcal{V}_i = \mathcal{N}_i \cup i$. The distance $d(i, j)$ between i and j is calculated by the sum of edges on the shortest path between i and j. As shown in Fig. 2, the neighborhood of agent i is $\mathcal{N}_i = \{a, j, x\}$ and the local region is $\mathcal{V}_i = \{a, i, j, x\}$.

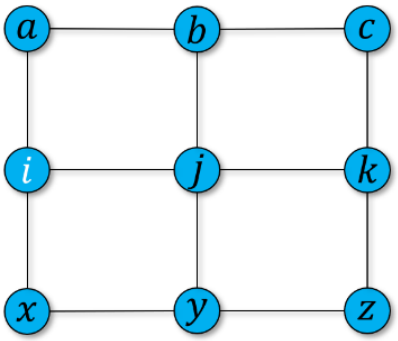

FIGURE 2. Neighbor node diagram.

In IQL, each local agent independently learns its own Q function and policy function. Assuming that agents share global rewards and full states, IQL updates the local Q function by estimating the TD target of the local agent and optimizing the loss function. TD target and loss function are as follows.

$$\bar{y}_{t,i} = \hat{R}_t + \gamma^{t_B - t} Q\left(s_{t_B}, a_{t_B,i}; \theta_i^-\right) \tag{8}$$

$$\bar{\mathcal{L}}\left(\theta_i\right) = \frac{1}{2\left|B\right|} \sum_{e_t \in B} \left(Q\left(s_t, a_{t,i}; \theta_i\right) - \bar{y}_{t,i}\right)^2 \tag{9}$$

where $\bar{y}_{t,i}$ denotes the TD target of agent i at time t and $a_{t_B,i}$ represents the action of agent i at time t_B.

In the real-time ATSC system, global information sharing is not feasible due to the high communication delay and computational complexity. Therefore, communication among agents is limited to each local region. Specifically, we consider the states of neighborhood and take the states of the whole local region as DNN input. We can obtain the TD target and loss function on the condition of local region states.

$$\hat{y}_{t,i} = \hat{R}_t + \gamma^{t_B - t} Q\left(s_{t_B,\mathcal{V}_i}, a_{t_B,i}; \theta_i^-\right) \tag{10}$$

$$\hat{\mathcal{L}}\left(\theta_i\right) = \frac{1}{2\left|B\right|} \sum_{e_t \in B} \left(Q\left(s_{t,\mathcal{V}_i}, a_{t,i}; \theta_i\right) - \hat{y}_{t,i}\right)^2 \tag{11}$$

The agents are suffered from partially observations of the environment after the global state is replaced by the local region state. At the same time, the training updating of the agent still has the problem of nonstationary state, and the updating Q function is also affected by other agent policies.

B. MULTI-AGENT Q-LEARNING

In IQL, independent learning and training of agents lead to difficult convergence and a nonstationary state. Therefore, two improved methods are applied to solve the above problems. The first method is to inform the local agent of the states and the policies of the neighbor agents to improve the observation of the local agent. The second method considers the spatial distance of intersections and introduces a spatial discount factor to reduce the influence of the states and rewards of other agents on local agent. For IQL, the behavior policies of other agents are added to the training of the local agent. The work [16] directly takes the parameter of Q networks of other agents into account, and the work [13] contains a low dimension of fingerprints, i.e., ϵ-greedy behavior policy of the exploration rate and a training iteration count. Here, we include Q values of neighbor agents, and transform them into probability values by mapping functions, representing the probability of the action choice of an agent, and then include the policy functions of other agents. Sigmoid function [8] is used to achieve the transformation by (12).

$$p\left(a_{t,i} | s_{t,\mathcal{V}_i}\right) = \frac{\text{sigmoid}\left(Q\left(s_{t,\mathcal{V}_i}, a_{t,i}\right)\right)}{\sum_{a_{t,i} \in \mathcal{A}_i} \text{sigmoid}\left(Q\left(s_{t,\mathcal{V}_i}, a_{t,i}\right)\right)} \tag{12}$$

where $\mathcal{A}_i$ denotes the action space of agent i. Compared with the direct use of Q values, such transformation can make the local agent learn the policy information of the neighbor agents better. It is worth noting that as the training progresses, the agent gradually increases the probability of good actions and decreases the probability of bad actions. Since communication among agents is limited in the local region, we add the latest policies of the neighbor agents into the input of DNN. The policy function of the local agent is calculated by (13). It takes the local region state at time t and the policy functions of the neighbor agents at time $t-1$ as the input of the neural network.

$$\pi_{t,i} = \pi_{\theta_i^-}\left(\cdot \left| s_{t,\mathcal{V}_i}, \pi_{t-1,\mathcal{N}_i}\right.\right) \tag{13}$$

In an ATSC system, the latest policies of the neighbor agents are notified to the local agent in real-time, and the behaviors of the neighbor agents are obtained to implement the cooperative control of multi-intersection traffic signals. Traffic conditions at local intersection change very little in a short period, and traffic dynamics are Markov chains, only related to the state and policy of the previous moment.

From the global perspective, a global reward is defined as the sum of rewards for all local agents, i.e., $r_t = \sum_{i \in \mathcal{V}} r_{t,i}$, which is reasonable for large-scale ATSC. Although the local agent already knows the states and policies of the neighbor agents, the global reward is still tricky to fit. Therefore, a spatial discount factor α is introduced to alter the global reward for each local agent i,

$$\tilde{r}_{t,i} = \sum_{d=0}^{D_i} \left(\sum_{j \in \mathcal{V} | d(i,j) = d} \alpha^d r_{t,j} \right) \tag{14}$$

where D_i represents the maximum distance between agent i and other agents on the network. For the spatial distance of intersections in the road network, the influence of other agents on local agent are reduced in a spatial order. In contrast to utilizing the identical global reward among agents, the spatially discounted global reward is more adaptive in ATSC, which is a trade-off between completely independent control and fully collaborative control among agents. Similarly, we use the spatial discount factor α on the states of the neighbor agents.

$$\tilde{s}_{t,\mathcal{V}_i} = \left[s_{t,i}, \alpha\left(s_{t,j}\right)_{j \in \mathcal{N}_i}\right] \tag{15}$$

Algorithm 1 Multi-Agent Q-Learning for ATSC

Input: α, γ, T, $|B|$, η_θ.
Output: $\{\theta_i\}_{i\in\mathcal{V}}$.

initialize s_0, π_0, $t \leftarrow 0$, $k \leftarrow 0$, $B = \emptyset$;
while stop condition is not reached **do**
  for $i \in \mathcal{V}$ **do**
    perform $a_{t,i}$ from $\pi_{t,i}$;
    receive $\tilde{r}_{t,i}$ and $\tilde{s}_{t,i}$;
  end for
  $B \leftarrow B \cup \left\{\left(\tilde{s}_{t,i}, a_{t,i}, \tilde{r}_{t,i}, \tilde{s}_{t+1,i}, \pi_{t,i}\right)\right\}_{i\in\mathcal{V}}$;
  $t \leftarrow t+1$, $k \leftarrow k+1$;
  if $t = T$ **then**
    initialize s_0, π_0, $t \leftarrow 0$;
  end if
  if $k = |B|$ **then**
    for $i \in \mathcal{V}$ **do**
      estimate $\hat{R}_{\tau,i}, \forall\tau \in B$;
      estimate $\tilde{R}_{\tau,i}, \forall\tau \in B$;
      calculate $\tilde{y}_{t,i}$ and $\tilde{\mathcal{L}}(\theta_i)$;
      update θ_i with $\eta_\theta \cdot \nabla\tilde{\mathcal{L}}(\theta_i)$;
    end for
    $B \leftarrow \emptyset$, $k \leftarrow 0$;
  end if
end while

(15) indicates that the state consists of the state of the local agent and the spatially discounted states of the neighbor agents. Given the above discounted global reward, we have $\tilde{R}_{t,i} = \sum_{\tau=t}^{t_B-1} \gamma^{\tau-t}\tilde{r}_{t,i}$. Consequently, TD target and loss function under the local region are as follows.

$$\tilde{y}_{t,i} = \tilde{R}_{t,i} + \gamma^{t_B-t} Q\left(\tilde{s}_{t_B,\mathcal{V}_i}, a_{t_B,i}; \theta_i^-, \pi_{t-1,\mathcal{N}_i}\right) \tag{16}$$

$$\tilde{\mathcal{L}}(\theta_i) = \frac{1}{2|B|}\sum_{e_t\in B}\left(Q\left(\tilde{s}_{t,\mathcal{V}_i}, a_{t,i}; \theta_i, \pi_{t-1,\mathcal{N}_i}\right) - \tilde{y}_{t,i}\right)^2 \tag{17}$$

In this way, the training update is more stable: first, the policies of the neighbor agents are added and input into the Q function, improving the fitting ability; the second is the use of spatial discounted reward, which is more consistent with the observation of the local region.

Algorithm 1 presents the MAQL algorithm. α is the spatial discount factor, γ is the discount factor, T is the planning horizon per episode, $|B|$ is the minibatch size, and η_θ is the learning rate for Q network. First, each local agent performs the action from the current policy, and the environment feedbacks the reward to the local agent. The experience is then collected and stored in the minibatch until there are enough samples for the minibatch to update. We reset the state and policy to restart a new episode if the episode is finished during the training process. Next, each Q network is updated by applying the minibatch gradient. Gradient optimizers are used to optimize the algorithm. At last, the learning procedure is terminated until the stop condition is reached.

IV. MAQL FOR TRAFFIC SIGNAL CONTROL

This section describes the implementation details of the MAQL algorithm for ATSC based on the Simulation of Urban MObility (SUMO [23]). Specifically, we define action, state, reward, neural network architecture, standardization methods, and training hyperparameter of the algorithm.

In the simulated environment over a period of T_s seconds, Δt denotes the time an agent interacts with the environment. Consequently, the environment is simulated for Δt seconds after each MDP steps [4]. For safety considerations, a yellow time $t_y < \Delta t$ is added after each traffic signal changes. In this paper, Δt= 5s and t_y= 2s, which leads to a planning horizon $T = T_s/\Delta t$ steps.

A. ACTION

There are many action settings, such as the duration of the current signal [20], the next phase of the traffic light [4], and holding or changing the current phase [22]. We choose the second set and define each local action as a possible phase or a combination of the red and green phases at the intersection. This setting is more flexible and suitable for ATSC. According to the current intersection traffic, an agent performs the best phase from all possible phase set $\mathcal{A}_i$.

B. STATE

The setting of a state is significant to describe the traffic condition at an intersection. We define the local state as follows:

$$s_{t,i} = \{\texttt{wait}_t[l_i], \texttt{wave}_t[l_i]\}_{l_i\in L_i} \tag{18}$$

where l_i is an incoming lane of intersection i and L_i denotes all the incoming lanes of intersection i. `wait` describes the waiting time for the first vehicle in the lane, and `wave` represents the total number of vehicles in each incoming lane within 50m of the intersection. A *laneAreaDetector* in SUMO is used to capture state information, which ensures the real-time ATSC.

C. REWARD

The definition of reward is vital in RL performance. The goal of an agent is to maximize long-term returns. Combined with the definition of reward in [3] and [4], a new reward function is defined as

$$r_{t,i} = -\sum_{l_i\in L_i}\left(\texttt{pressure}_{t+\Delta t}[l_i] + a\cdot\texttt{wait}_{t+\Delta t}[l_i]\right) \tag{19}$$

where a [veh/s] is a trade-off coefficient, and `pressure` is the evaluation metric based on the vehicle queue length and can be simply measured. The reward is post-decision, and both `pressure` and `wait` are calculated at time $t+\Delta t$. This kind of reward setting is closely related to state and action, which can directly reflect the degree of traffic congestion and vehicle delay time at the intersection.

D. NETWORK ARCHITECTURE

In a large-scale ATSC problem, traffic flow is a complex spatiotemporal data. The MDP of global TSC causes the nonstationary state if each agent only captures the state currently. DQN uses experience replay buffers to store historical

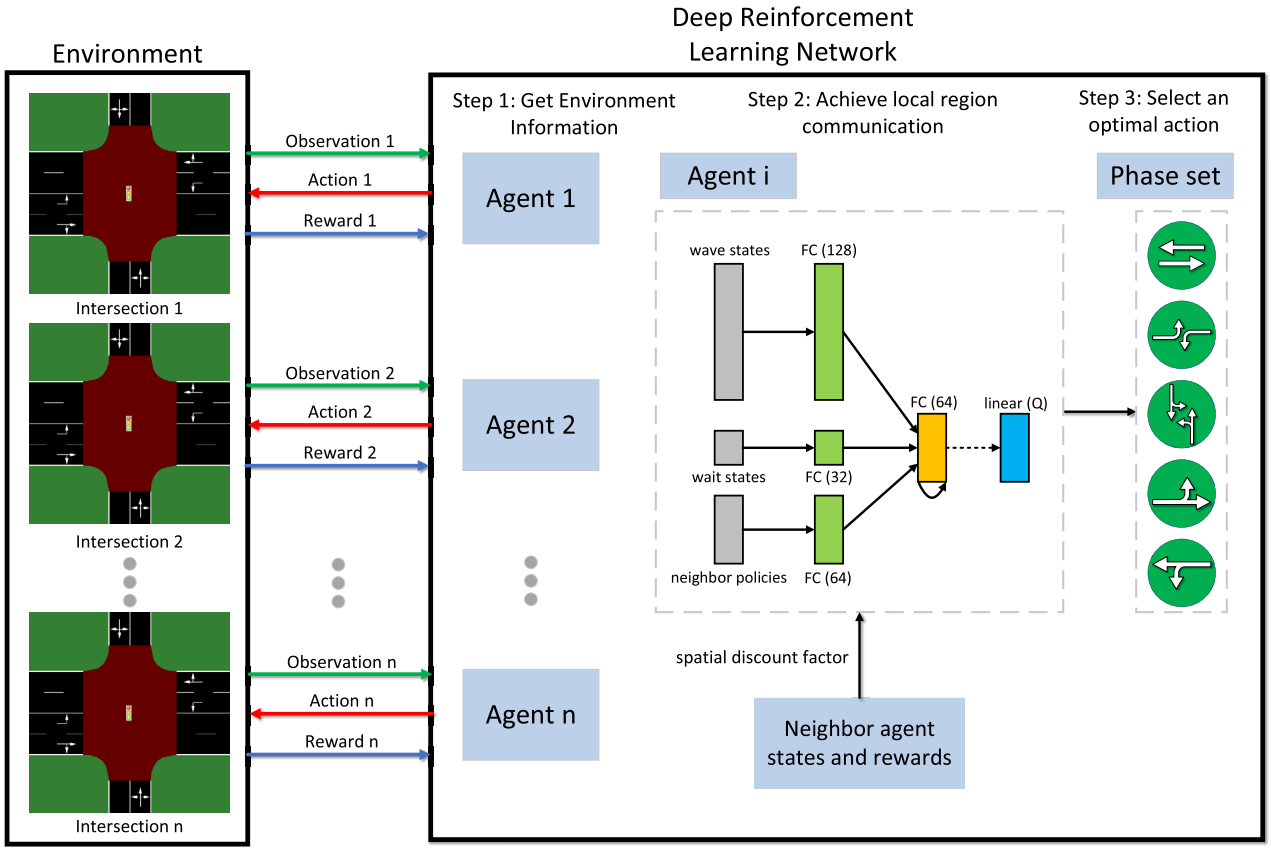

FIGURE 3. Deep reinforcement learning framework diagram.

experience data and utilizes the data to improve algorithm performance. Fig. 3 shows the architecture of deep reinforcement learning. Each intersection in the environment is treated as an agent to independently obtain the state of intersection. Then, we implement communication among agents under the local region. In addition to the wave and wait for states of the local agent itself, the neighbor policies are also added to the state of local agent. The state and reward of the neighbor agents processed by the spatial discount factor are input into the training process of the local agent. All of these enable local region communication. The local agent obtains the Q value of each action and selects an optimal action as the next phase of the intersection according to the behavior policy. Fig. 4 shows the DNN architecture of MAQL, where wave states, wait states, and neighbor policies are first handled by independent fully connected (FC) layers. Then, all hidden layer units are consolidated and input into a FC layer. A linear function is used to process the output layer. In the training process of DNN, we use advanced orthogonal initialization and RMSprop as the gradient optimizer. All standard states are clipped to [0, 2] to prevent gradient explosion, with an upper limit of 40 for each gradient. The standardization of the reward is set to clip to [-2, 2], which is used to stabilize minibatch updates. The MAQL algorithm is based on a decentralized framework and communication with neighbor agents. Only its state and policy and that of its neighbors need

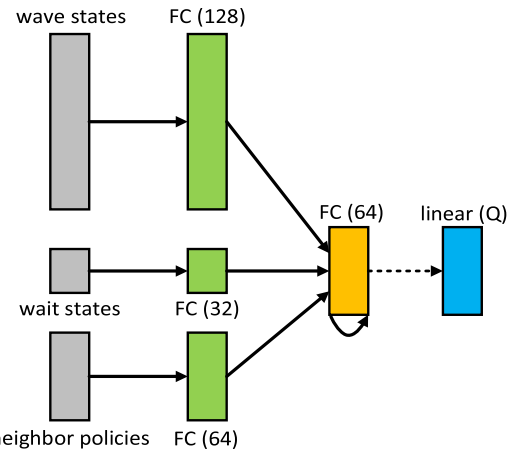

FIGURE 4. DNN architectures of MAQL in ATSC. Hidden layer size is denoted in parenthesis.

to be considered for each local agent. The MAQL algorithm is simply scalable to a new environment.

V. NUMERICAL EXPERIMENTS

ATSC based on MAQL is simulated in a 5×5 synthetic traffic grid. In this section, we intend to develop a challenging simulation environment to conduct fair comparison experiments among the state-of-the-art algorithms.

A. GENERAL SETTINGS

To demonstrate the effectiveness and robustness of the MAQL algorithm, we compare it with several advanced controllers. IQL-DNN [22] is an algorithm that uses DNN to fit the Q function, and its network architecture is consistent with the MAQL. IQL-LR [22] is an IQL algorithm based on LR. IA2C algorithm [4] extends the idea of the IQL to the A2C algorithm. MA2C algorithm [4] is a stable approach based on the IA2C, and its performance is greatly improved compared with the IA2C. MA2C applies the long-short term memory layer as the last hidden layer to extract features from different data types. Greedy is a fully decentralized algorithm that chooses the phase related to the maximum overall green wave on total incoming lanes. Each controller has identical settings in terms of action, state, and interaction with the environment.

For IQL-DNN and IQL-LR, Q function is fitted differently. IQL-DNN uses DNN to fit the Q function, while IQL-LR uses LR to fit the Q function. IA2C is an independent advantage actor-critic algorithm and trains each agent independently using the A2C algorithm. The difference between IQL and IA2C is that the agent is trained using a different algorithm. MA2C is based on IA2C and introduces two control strategies to the training process. MAQL is based on IQL and applies communication among neighbor agents, spatial discount factor and a new reward function with pressure and vehicle waiting time, significantly improving the evaluation performance in ATSC.

All MARL algorithms are trained in 1million steps, which contains nearly 1400 episodes under episode horizon $T = 720$ steps. Then all controllers are evaluated over ten episodes. Random seeds can generate a set of random numbers and play an essential role in the training and evaluation. For MDP, we set $\gamma = 0.99$ and $\alpha = 0.9$. For MAQL, we set the learning rate $\eta_\theta = 1\mathrm{e}-4$, the minibatch size $|B| = 120$, and the replay buffer size 1000. Due to the partial observability of environment, the size of replay buffer is set relatively small. The behavior policy ϵ-greedy is used and ϵ linearly decays from 1.0 to 0.01 during the training process. For IA2C and MA2C, the parameter setting is the same as [4]. For MAQL, the parameters are consistent with the IQL for a fair comparison.

B. SYNTHETIC TRAFFIC GRID

The experimental scenario is a large 5×5 synthetic traffic grid, as shown in Fig. 5, an arterial street consisting of two lanes with a speed limit of 20m/s, and a one-lane street with a speed limit of 11m/s. The action space at each intersection includes five potential phases: east-west straight phase, east-west left turn phase, east straight and left turn phase, west straight and left turn phase, and north-south straight and left turn phase. Right turning is allowed by default. Obviously, the centralized RL framework is infeasible due to the size of the joint action space is 5^{25}. We set up four time-dependent traffic flow groups as shown in Fig. 6 [4]. At the outset, the three main traffic flow F_1 generate (x_{10}, x_4), (x_{11}, x_5) and (x_{12}, x_6)

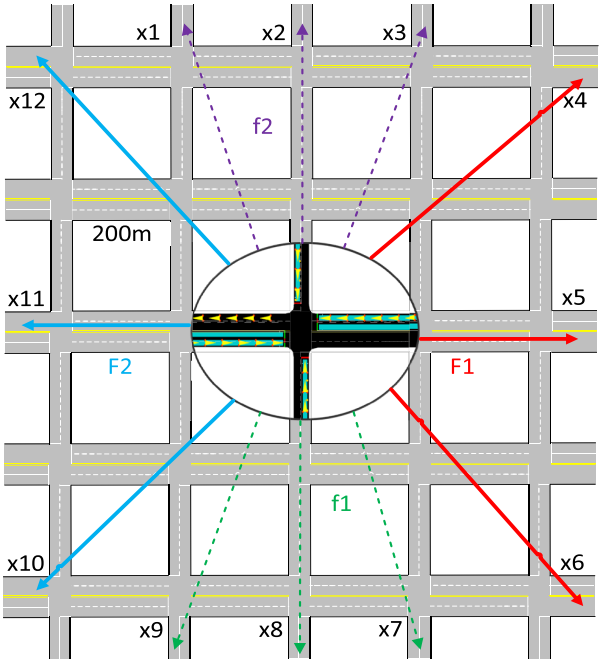

FIGURE 5. Synthetic traffic grid with 25 intersections and time-dependent traffic flow.

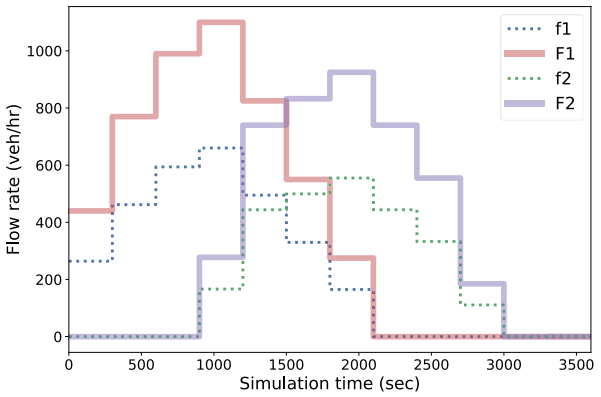

FIGURE 6. Time-variant traffic flow in synthetic traffic grid.

in an origination-destination pair. Meanwhile, three minor traffic flow f_1 generate (x_1, x_7), (x_2, x_8) and (x_3, x_9). After 15 minutes, traffic flow F_1 and f_1 start to decrease, and traffic flow in the opposite direction F_2 and f_2, i.e., the origination and the destination switch, begin to generate traffic flow. It is worth noting that traffic flow only represents a high level of traffic demand, while the route of each vehicle is randomly generated. Taking into account the setting of MDP, we set the reward factor at 0.2veh/s, and the standardization factor of the wave, wait, and reward at 5veh, 100s, and 3000veh respectively.

$$\bar{R} = \frac{1}{T}\sum_{t=0}^{T-1}\left(\sum_{i\in\mathcal{V}} r_{t,i}\right) \tag{20}$$

Here, we use five evaluation metrics to comprehensively evaluate the performance of various MARL algorithms on large-scale ATSC.

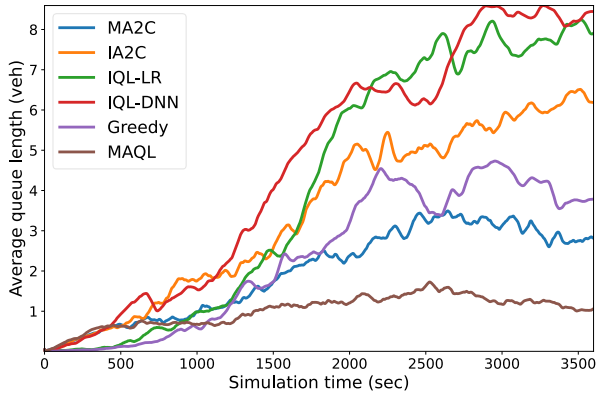

FIGURE 7. Average queue length in synthetic traffic grid.

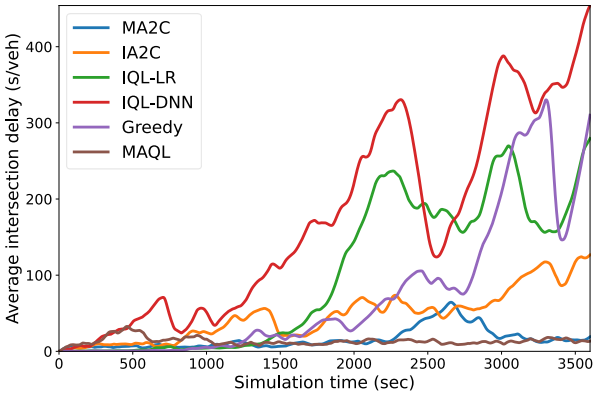

FIGURE 8. Average intersection delay in synthetic traffic grid.

Average queue length: Average queue length is the average number of stopped vehicles in each lane and is often used to measure the performance of traffic signals. A vehicle with a speed of less than 0.1m/s is considered a stopped vehicle.

Average intersection delay: Average intersection delay is the average waiting time for all vehicles in the entire traffic network at the intersection.

Average vehicle speed: The average vehicle speed is the average speed of all vehicles in the whole traffic network.

Trip completion flow: Trip completion flow is the average number of vehicles completing trips per second across the entire traffic network, i.e., the average number of vehicles per second that have reached their destination.

Trip delay: Trip delay is the average waiting time for all vehicles in the entire road network.

It needs to be emphasized that average intersection delay differ from trip delay. The average intersection delay describes the average waiting time at the intersection, while the trip delay measures the average waiting time of vehicles during travel.

Evaluation results: In (20), the average reward per evaluation episode $\bar{R}$ are -221.69, -412.06, -898.32, -1152.01, -1763.55 and -2850.74, for MAQL, MA2C, IA2C, Greedy, IQL-LR, and IQL-DNN. Apparently, MAQL outperforms other approaches for the given objective. As shown in Fig. 7, the average queue length of the traffic network is plotted at each simulation step, where the lane shows the average across evaluation episodes. IQL-DNN, IQL-LR, and IA2C may not learn a stable strategy for solving the traffic congestion in the end. On the contrary, the Greedy policy performs well in reducing vehicle queue length due to maximizing the traffic flow at each step. MA2C is a steadier strategy that further reduces traffic congestion. MAQL generally maintains the lowest queue length and has acceptable robustness and a steady state.

The curve of average intersection delay of the network with simulation time is plotted in Fig. 8. IQL-DNN, IQL-LR, and Greedy fail to perform well in the traffic congestion. IA2C maintains a low growth trend in traffic delays. MA2C keeps the traffic delay at a low level. At last, MAQL outperforms other controllers in average traffic delay at intersections even under the peak-hour traffic dynamics.

Table 1 uses several significant metrics to evaluate ATSC performance. These metrics are calculated and integrated during each evaluation episode. Apparently, MAQL outperforms other controllers in reward, average queue length, and average intersection delay. Regarding other metrics, Greedy performs better than other MARL controllers. This may be caused by the low-density traffic flow in the first 15 minutes. Since Greedy does not consider the upstream and downstream intersections, its performance variance is high under the high-density traffic flow.

C. ABLATION EXPERIMENTS

We test the performance of different RL-based methods in the presence and absence of pressure. It is worth noting that since MAQL method already uses pressure in the reward function, we use queue length instead of pressure design in the reward function. As can be seen from Table 2, pressure design can significantly improve the performance of RL model in terms of average queue length and intersection delay.

Fig. 9 illustrates the performance of several RL-based methods in the synthetic traffic grid scenario with or without pressure on the average queue length. The upper left corner of Fig. 9 respectively shows that the first four RL methods add pressure, and the MAQL algorithm removes pressure. The comparison between Fig. 7 and Fig. 9 shows that MA2C, IA2C, IQL-DNN and IQL-LR algorithms reduce the average queue length after adding pressure. The performance of the MAQL algorithm on the average queue length increases after removing pressure.

Fig. 10 plots the performance of several RL-based methods in the synthetic traffic grid scenario with or without pressure on the average intersection delay. As can be seen from the comparison between Fig. 8 and Fig. 10, MA2C, IA2C, IQL-DNN, and IQL-LR algorithms reduce the average intersection delay after introducing pressure. After the pressure is removed, the performance of the MAQL algorithm on average intersection delay decreases. Results of ablation experiments further demonstrate that pressure

TABLE 1. ATSC performance in synthetic traffic grid best values are in bold.

Metrics	Greedy	MAQL (Our Method)	MA2C [4]	IA2C [4]	IQL-DNN [22]	IQL-LR [22]
reward	-1152.01	**-221.69**	-412.06	-898.32	-2850.74	-1763.55
avg. queue length [veh]	2.41	**1.02**	2.00	3.52	4.75	4.21
avg. intersection delay [s/veh]	76.61	**13.54**	16.19	47.91	174.72	102.05
avg. vehicle speed [m/s]	**3.31**	2.94	2.38	1.68	1.54	2.72
trip completion flow [veh/s]	**0.79**	0.61	0.66	0.41	0.20	0.52
trip delay [s]	**196**	214	329	482	356	225

TABLE 2. Performance of different RL-based methods with and without pressure in synthetic traffic grid best values are in bold.

Model	Average Queue Length	Average Intersection Delay
IQL-DNN	4.75	174.72
IQL-DNN + pressure	**4.12**	**136.84**
IQL-LR	4.21	102.05
IQL-LR + pressure	**4.04**	**79.46**
IA2C	3.52	47.91
IA2C + pressure	**3.37**	**39.67**
MA2C	2.00	16.19
MA2C + pressure	**1.87**	**14.14**
MAQL - pressure	1.99	27.71
MAQL	**1.02**	**13.54**

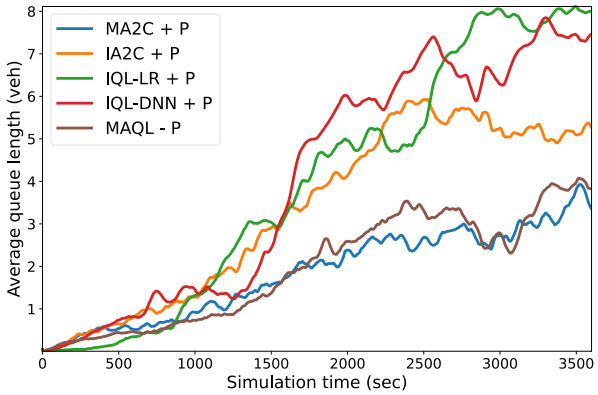

FIGURE 9. Average queue length of the conducted ablation experiments: Effect of the presence and absence of pressure in synthetic traffic grid.

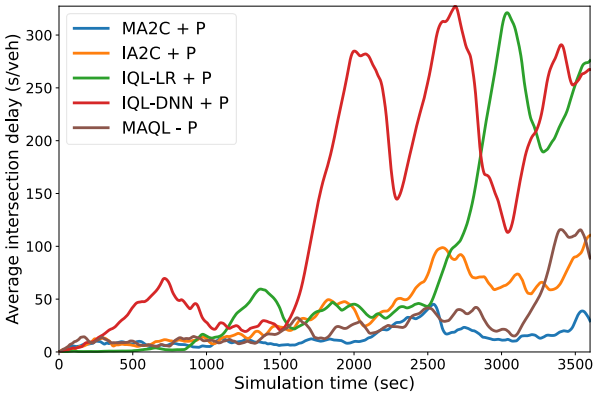

FIGURE 10. Average queue length of the conducted ablation experiments: Effect of the presence and absence of pressure in synthetic traffic grid.

TABLE 3. Performance of main metrics with different spatial discount factor value.

α	$\bar{R}$	Average Queue Length	Average Intersection Delay
0	-888.59	2.72	68.33
0.1	-602.31	2.19	47.01
0.2	-717.61	2.56	53.28
0.3	-409.28	1.86	30.07
0.4	-483.87	2.31	31.78
0.5	-561.10	2.17	43.09
0.6	-242.49	1.24	16.38
0.7	-431.32	1.66	35.94
0.8	-232.13	1.06	15.89
0.9	**-221.69**	**1.02**	**13.54**
1.0	-263.40	1.24	17.95

can significantly improve the performance of RL-based algorithms in the ATSC scenario by introducing pressure into the reward function of RL.

We test the performance of the MAQL algorithm on ATSC under eleven groups of α value with other conditions being constant as shown in Table 3. It shows that the MAQL algorithm obtains the highest reward and the best performance in terms of average queue length and intersection delay when $\alpha = 0.9$. It is worth noting that the MAQL algorithm ignores the states and rewards of the neighbor agents when $\alpha = 0$, and the agents are in a fully independent control state. The state and reward weights of the local agent and the neighbor agents are the same when $\alpha = 1$, and the agents are in a completely cooperative control state. The introduction of spatial discount factor α is a compromise of these two control approaches.

VI. CONCLUSION

In this paper, a decentralized multi-agent Q-learning framework with local region communication among agents is proposed for ATSC. Specifically, the states and policies of neighbor agents are notified to improve the observability of each local agent. The spatial discount factor is used to scale down the influence of other agents, which can reduce the difficulty of learning procedure. Motivated by achieving coordination among intersections in the RL settings and

better performance in ATSC, these two control strategies are applied to the MAQL algorithm. A new reward function with pressure and vehicle waiting time is designed to achieve coordination among intersections. Experiments conducted on a large synthetic traffic grid demonstrate the optimality, robustness and effectiveness of MAQL compared to the state-of-the-art decentralized MARL algorithms. Ablation experiments show that the reward function with pressure and vehicle waiting time can indeed improve the RL performance in ATSC. Future work will study ingenious designs for coordination among agents to improve performance. We will also verify the feasibility and scalability of MAQL in a real-world scenario.

REFERENCES

[1] A. Haydari and Y. Yilmaz, "Deep reinforcement learning for intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 11–32, Jan. 2022.

[2] H. Wei, G. Zheng, V. Gayah, and Z. Li, "A survey on traffic signal control methods," 2019, *arXiv:1904.08117*.

[3] H. Wei, C. Chen, G. Zheng, K. Wu, V. Gayah, K. Xu, and Z. Li, "PressLight: Learning max pressure control to coordinate traffic signals in arterial network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1290–1298.

[4] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 1086–1095, Mar. 2020.

[5] P. Koonce and L. Rodegerdts, "Traffic signal timing manual," Fed. Highway Admin., Washington, DC, USA, Tech. Rep. FHWA-HOP-08-024, 2008.

[6] S. B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," in *Advances in Applied Self-Organizing Systems*. London, U.K.: Springer, 2013, pp. 45–55.

[7] P. Varaiya, "Max pressure control of a network of signalized intersections," *Transp. Res. C, Emerg. Technol.*, vol. 36, pp. 177–195, Nov. 2013.

[8] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, p. 1054, Sep. 1998.

[9] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 2496–2505.

[10] T. Tan, F. Bao, Y. Deng, A. Jin, Q. Dai, and J. Wang, "Cooperative deep reinforcement learning for large-scale traffic grid signal control," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2687–2700, Jun. 2020.

[11] C. Chen, "Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, 2020, pp. 3414–3421.

[12] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. Int. Conf. Mach. Learn.*, 1993, pp. 330–337.

[13] G. Tesauro, "Extending Q-learning to general adaptive multi-agent systems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 871–878.

[14] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 1146–1155.

[15] C. Szepesvári, "Algorithms for reinforcement learning," *Synthesis lectures Artif. Intell. Mach. Learn.*, vol. 4, no. 1, pp. 1–98, Jul. 2010.

[16] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[17] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI. Conf. Art. Intell.*, vol. 30, no. 1, Mar. 2016, pp. 2094–2100.

[18] Z. Wang, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*.

[20] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transp. Res. C, Emerg. Technol.*, vol. 85, pp. 732–752, Dec. 2017.

[21] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2017, *arXiv:1706.02275*.

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

[23] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2575–2582.

[24] S. Bouktif, A. Cheniki, A. Ouni, and H. El-Sayed, "Deep reinforcement learning for traffic signal control with consistent state and reward design approach," *Knowl.-Based Syst.*, vol. 267, May 2023, Art. no. 110440.

[25] Z. Yao, X. Liang, G.-P. Jiang, and J. Yao, "Model-based reinforcement learning control of electrohydraulic position servo systems," *IEEE/ASME Trans. Mechatronics*, vol. 28, no. 3, pp. 1446–1455, Jun. 2023.

[26] Q. Hou and J. Dong, "Distributed dynamic event-triggered consensus control for multiagent systems with guaranteed L_2 performance and positive inter-event times," *IEEE Trans. Autom. Sci. Eng.*, early access, Dec. 29, 2022, doi: 10.1109/TASE.2022.3231845.

[27] Q. Hou and J. Dong, "Cooperative fault-tolerant output regulation of linear heterogeneous multiagent systems via an adaptive dynamic event-triggered mechanism," *IEEE Trans. Cybern.*, vol. 53, no. 8, pp. 5299–5310, Aug. 2023.

[28] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

LIANG QI (Member, IEEE) received the Ph.D. degree in computer software and theory from Tongji University, Shanghai, China, in 2017. From 2015 to 2017, he was a Visiting Student with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. He is currently an Associate Professor with the College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, China. He has published more than 100 papers in journals and conference proceedings. His current research interests include Petri nets, optimization, machine learning, and intelligent transportation.

YUANZHEN SUN received the B.S. degree from the Shandong University of Science and Technology, Qingdao, China, in 2020, where he is currently pursuing the M.S. degree with the College of Computer Science and Engineering. His current research interests include reinforcement learning, traffic signal control, and intelligent transportation systems.

WENJING LUAN (Member, IEEE) received the Ph.D. degree in computer software and theory from Tongji University, Shanghai, China, in 2018. From May 2017 to July 2017, she was a Visiting Student with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. She is currently a Lecturer of computer science and technology with the Shandong University of Science and Technology, Qingdao, China. She has published more than 30 papers in journals and conference proceedings. Her current research interests include machine learning, recommender systems, and intelligent transportation systems.