Large-Scale Traffic Signal Control Using a Novel Multiagent Reinforcement Learning

Xiaoqiang Wang, Liangjun Ke, *Member, IEEE*, Zhimin Qiao, and Xinghua Chai

***Abstract*—Finding the optimal signal timing strategy is a difficult task for the problem of large-scale traffic signal control (TSC). Multiagent reinforcement learning (MARL) is a promising method to solve this problem. However, there is still room for improvement in extending to large-scale problems and modeling the behaviors of other agents for each individual agent. In this article, a new MARL, called cooperative double *Q*-learning (Co-DQL), is proposed, which has several prominent features. It uses a highly scalable independent double *Q*-learning method based on double estimators and the upper confidence bound (UCB) policy, which can eliminate the over-estimation problem existing in traditional independent *Q*-learning while ensuring exploration. It uses mean-field approximation to model the interaction among agents, thereby making agents learn a better cooperative strategy. In order to improve the stability and robustness of the learning process, we introduce a new reward allocation mechanism and a local state sharing method. In addition, we analyze the convergence properties of the proposed algorithm. Co-DQL is applied to TSC and tested on various traffic flow scenarios of TSC simulators. The results show that Co-DQL outperforms the state-of-the-art decentralized MARL algorithms in terms of multiple traffic metrics.**

***Index Terms*—Double estimators, mean-field approximation, multiagent reinforcement learning (MARL), traffic signal control (TSC).**

I. Introduction

TRAFFIC congestion is becoming a great puzzling problem in urban areas, mainly due to the difficulty of effective utilization of limited road resources (e.g., road width). By regulating the traffic flow of the road network, the traffic signal control (TSC) at intersections plays an important role in utilizing the road resources and helping to reduce traffic congestion [1].

Many researchers have devoted efforts to TSC, with the aim of minimizing the average waiting time in the entire traffic system and maximizing social welfare [2]. When traffic signals are large scale, the traditional control methods, such as pretimed [3] and actuated control systems [4], may fail to deal with the dynamic of the traffic conditions or lack the ability to foresee traffic flow. Intelligent computing methods (such as genetic algorithm [5]; swarm intelligence [6]; and neuro-fuzzy networks [7], [8]), however, in many cases, suffer from a slow convergence rate. Reinforcement learning (RL) [9] is a promising adaptive decision-making method in many fields. It has been applied to cope with TSC [10], [11]. Not only does it make real-time decisions according to traffic flow but also predicts future traffic flow. Especially, in recent years, RL has made tremendous progress that significantly attributes to the success of deep learning [12]. By using the deep neural network to approximate the value function or action-value function (such as DQN [13] and deep deterministic policy gradient (DDPG) [14]), RL can be adapted to the problems with large-scale state space or action space.

As for TSC with multiple signalized intersections, a straightforward idea is centralized, in which TSC is considered as a single-agent learning problem [5], [15]. However, centralized approaches often need to collect all traffic data in the network as the global state [16], which may lead to high latency and failure rate. In addition, as the number of intersections increases, the joint state space and action space of the agent will increase exponentially to a large extent, which incurs the curse of dimension. Consequently, a centralized method often requires a very heavy computational and communication burden.

An alternative way is multiagent RL (MARL), in which each signalized intersection is regarded as an agent. The challenge of an MARL approach is how to respond to the dynamic interaction between each signal agent and the environment, which significantly affects the adaptive decision making of other signals [17]. Moreover, most of the current MARL methods are only studied on very limited-size traffic network problems [18], [19]. However, in urban traffic systems, it is often necessary to consider all the signals in a global coordination manner. In [20] and [21], each signal is regarded as an independent agent for training. Although this class of approaches can easily be extended to large-scale scenarios, they directly ignore the actions of other agents in the road network system and implicitly suggest that the environment

Manuscript received August 10, 2019; revised December 24, 2019, March 25, 2020, and June 28, 2020; accepted August 5, 2020. Date of publication September 3, 2020; date of current version December 22, 2020. This work was supported by the National Natural Science Foundation of China under Grant 61973244 and Grant 61573277. This article was recommended by Associate Editor X. Qu. *(Corresponding author: Liangjun Ke.)*

Xiaoqiang Wang is with the State Key Laboratory for Manufacturing Systems Engineering, School of Automation Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China, and also with the CETC Key Laboratory of Aerospace Information Applications, Shijiazhuang 050081, China (e-mail: wangxq5127@stu.xjtu.edu.cn).

Liangjun Ke and Zhimin Qiao are with the State Key Laboratory for Manufacturing Systems Engineering, School of Automation Science and Engineering, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: keljxjtu@xjtu.edu.cn; qiao.miracle@gmail.com).

Xinghua Chai is with the CETC Key Laboratory of Aerospace Information Applications, Shijiazhuang 050081, China (e-mail: cetc54008@yeah.net).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2020.3015811

is static. This makes it difficult for agents to learn favorable strategies with convergence guarantee. In [22], a max-plus method is proposed to deal with the large-scale TSC problem, but this approach requires additional computation during execution. Multiagent A2C (MA2C) [23] is developed from IA2C which is scalable and belongs to a decentralized MARL algorithm, but it may be uneasy to determine the appropriate attenuation factor to weaken the state and reward information from other agents.

In this work, we present a decentralized and scalable MARL method which is called after cooperative double Q-learning (Co-DQL) and apply it to TSC. The new approach adopts a highly scalable independent double Q-learning (IDQL) method, with the aim of avoiding the problem of overestimation suffered from traditional independent Q-learning (IQL) [24]. In the meantime, it can ensure exploration by using the upper confidence bound (UCB) [25] rule. In order to make agents learn a better cooperative strategy for large-scale problems, it employs a mean-field theory [26], which has been studied in [27]. It approximately treats the interactions within the population of agents as the interaction between a single agent and a virtual agent averaged by other individuals, which potentially transmits the action information among all agents in the environment. Furthermore, we introduce a new reward allocation mechanism and a local state sharing method to make the learning process of agents more stable and robust. To theoretically support the effectiveness of the proposed algorithm, we provide the convergence proof for the proposed algorithm under some mild conditions. A numerical experiment is performed on various traffic flow scenarios of TSC simulators. The empirical results show that the proposed method outperforms several state-of-the-art decentralized MARL algorithms in terms of multiple traffic metrics.

The remainder of this article is organized into six sections. Section II describes the background on RL. Section III presents the proposed method and analyzes the convergence properties. Section IV introduces the application of Co-DQL to the TSC problem. Section V describes the setup and conditions of the experiments in detail and makes a comparative analysis and discussion on the experimental results. Section VI summarizes this article.

II. Background on Reinforcement Learning

A. Single-Agent RL

Q-learning is one of the most popular RL methods and it solves sequential decision-making problems by learning estimates for the optimal value of each action. The optimal value can be expressed as $Q^*(s,a) = \max_\pi Q^\pi(s,a)$. However, it is not easy to learn the values of all the actions in all states when the state space or action space is larger. In this case, we can learn a parameterized action-value function $Q(s,a;\boldsymbol{\theta})$. When taking action a_t in state s_t and observing the immediate reward r_{t+1} and resulting state s_{t+1}, the standard Q-learning updates the parameters as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\Big(Y_t^Q - Q(s_t,a_t;\boldsymbol{\theta}_t)\Big)\nabla_{\boldsymbol{\theta}_t} Q(s_t,a_t;\boldsymbol{\theta}_t) \tag{1}$$

where t is the time step, α is the learning rate, and the target Y_t^Q is defined as

$$Y_t^Q \equiv r_{t+1} + \gamma \max_a Q(s_{t+1},a;\boldsymbol{\theta}_t) \tag{2}$$

where the constant $\gamma \in [0,1)$ is the discount factor that trades off the importance of immediate and later rewards. After updating gradually, it can converge to an optimal action-value function.

Note that Q-learning approximates the value of the next state by maximizing over the estimated action values in the corresponding state, namely, $\max_a Q_t(s_{t+1},a;\boldsymbol{\theta}_t)$ and it is an estimate of $E\{\max_a Q_t(s_{t+1},a;\boldsymbol{\theta}_t)\}$, which, in turn, is used to approximate $\max_a E\{Q_t(s_{t+1},a;\boldsymbol{\theta}_t)\}$. This method of approximating the maximum expected value has a positive deviation [24], [28], [29], which leads to overestimation of the optimal value and may damage the performance.

B. Multiagent RL

The single-agent RL is based on the Markov decision process (MDP) theory, while for MARL, it mainly stems from Markov game [30], which generalizes the MDP and was proposed as the standard framework for MARL [31].

We can use a tuple to formalize Markov game, namely, $(N, \boldsymbol{S}, \boldsymbol{A}_{1,2,\ldots,N}, r_{1,2,\ldots,N}, p)$, where N is the number of agents in the game system, $\boldsymbol{S} = \{\boldsymbol{s}_1,\ldots,\boldsymbol{s}_n\}$ is a finite set of system states, n is the number of states in the system, $\boldsymbol{A}_k$ is the action set of agent $k \in \{1,\ldots,N\}$; $r_k : \boldsymbol{S}\times\boldsymbol{A}_1\times\cdots\times\boldsymbol{A}_N\times\boldsymbol{S}\to\mathbb{R}$ is the reward function of agent k, determining the immediate reward, and $p : \boldsymbol{S}\times\boldsymbol{A}_1\times\cdots\times\boldsymbol{A}_N \to \mu(\boldsymbol{S})$ is the transition function. Each agent has its own strategy and chooses actions according to its strategy. Under the joint strategy $\boldsymbol{\pi} \triangleq (\pi_1,\ldots,\pi_N)$, at each time step, the system state is transferred by taking the joint action $\boldsymbol{a} = (a_1,\ldots,a_N)$ selected according to the joint strategy and each agent receives the immediate reward as the consequence of taking the joint action. To measure the performance of a strategy, either the future discounted reward or the average reward over time can be used, depending on the policies of other agents. This results in the following definition for the expected discounted reward for agent k under a joint policy $\boldsymbol{\pi}$ and initial state $\boldsymbol{s}(0) = \boldsymbol{s} \in \boldsymbol{S}$:

$$V_k^{\boldsymbol{\pi}}(s) = E^{\boldsymbol{\pi}}\left\{\sum_{t=0}^{\infty}\gamma^t r_k(t+1)|\boldsymbol{s}(0)=\boldsymbol{s}\right\} \tag{3}$$

while the average reward for agent k under this joint policy is defined as

$$J_k^{\boldsymbol{\pi}}(s) = \lim_{T\to\infty}\frac{1}{T}E^{\boldsymbol{\pi}}\left\{\sum_{t=0}^{T} r_k(t+1)|\boldsymbol{s}(0)=\boldsymbol{s}\right\}. \tag{4}$$

On the basis of (3) (the most used form), the action-value function $Q_k^{\boldsymbol{\pi}} : \boldsymbol{S}\times\boldsymbol{A}_1\times\cdots\times\boldsymbol{A}_N\to\mathbb{R}$ of agent k under the joint strategy $\boldsymbol{\pi}$ can be written as follows according to the Bellman equation:

$$Q_k^{\boldsymbol{\pi}}(s,\boldsymbol{a}) = r_k(s,\boldsymbol{a}) + \gamma E_{s'\sim p}\big[V_k^{\boldsymbol{\pi}}(s')\big] \tag{5}$$

Algorithm 1: General Multiagent Q-Learning Framework

Input: Initial Q value of all state-action pairs for each agent k

Output: Convergent Q value for each agent k

Initialize $Q_k(s, \boldsymbol{a}) = 0 \quad \forall s, \boldsymbol{a}, k$;
while *not termination condition* **do**
  for *all agents* k **do**
    select action a_k
  execute joint action $\boldsymbol{a} = (a_1, \ldots a_N)$;
  observe new state s', rewards r_k;
  for *all agents* k **do**
    $Q_k(s, \boldsymbol{a}) = (1-\alpha)Q_k(s, \boldsymbol{a}) + \alpha\big[r_k(s, \boldsymbol{a}) + \gamma V_k(s')\big]$

where $V_k^{\boldsymbol{\pi}}(s) = \mathbb{E}_{\boldsymbol{a}\sim\boldsymbol{\pi}}[Q_k^{\boldsymbol{\pi}}(s, \boldsymbol{a})]$ and s' is the system state at the next time step. The commonly used MARL methods are generally based on Q-learning. The general multiagent Q-learning framework is shown in Algorithm 1.

MARL enables each agent to learn the optimal strategy to maximize its cumulative reward. However, the value function of each agent is related to the joint strategy $\boldsymbol{\pi}$ of all agents, so it is, in general, impossible for all players in a game to maximize their payoff simultaneously. For MARL, an important solution concept is *Nash equilibrium*. Given these opponent strategies, the *best response* of agent k to a vector of opponent strategies is defined as the strategy π_k^* that achieves the maximum expected reward, which is given as follows:

$$E\{r_k|\pi_1, \ldots, \pi_k, \ldots, \pi_N\} \leq E\big\{r_k|\pi_1, \ldots, \pi_k^*, \ldots, \pi_N\big\} \quad \forall \pi_k. \tag{6}$$

Then, the *Nash equilibrium* is represented by a joint strategy $\boldsymbol{\pi}^* \triangleq (\pi_1^*, \ldots, \pi_N^*)$ in which each agent acts with the *best response* π_k^* to others and all other agents follow the joint policy $\boldsymbol{\pi}_{-k}^*$ of all agents except k, where the joint policy $\boldsymbol{\pi}_{-k}^* \triangleq (\pi_1^*, \ldots, \pi_{k-1}^*, \pi_{k+1}^*, \ldots, \pi_N^*)$. In this case, as long as all other agents keep their policies unchanged, no agent can benefit by changing its policy. Many MARL algorithms reviewed strive to converge to *Nash equilibrium*. In addition, the Q-function will eventually converge to the *Nash Q-value* $\boldsymbol{Q}^* = (Q_1^*, \ldots, Q_N^*)$ received in a *Nash equilibrium* of the game.

III. Description of the Proposed Method

Co-DQL is developed from a new algorithm, called the IDQL method, which is also first proposed in this article. In the following, we first present the IDQL method, then introduce Co-DQL, and finally, we analyze its convergence properties.

A. Independent Double Q-Learning Method

Most MARL methods are based on Q-learning. However, as described in Section II-A, traditional RL methods cause the problem of overestimation, which to some extent harms the performance of RL methods. In [24], a double Q-learning algorithm is proposed, which uses double estimators instead of $\max_a Q_t(s_{t+1}, a)$ to approximate $\max_a E\{Q_t(s_{t+1}, a)\}$, which is helpful to avoid the problem of overestimation in standard Q-learning.

Inspired by IQL [21], we develop an IDQL method based on the UCB rule. For each agent k, it is associated with two different action-value functions, each of which is updated with a value from the other action-value function for the next state. More specifically, suppose that the two action-value functions are $Q_k^{\mathfrak{a}}$ and $Q_k^{\mathfrak{b}}$, and one of them is randomly selected for updating each time. The updating process of the action-value function $Q_k^{\mathfrak{a}}$ is as follows. First, the maximal valued action a_k^* in the next state s' is selected according to the action-value function $Q_k^{\mathfrak{a}}$, namely, $a_k^* = \text{argmax}_a\, Q_k^{\mathfrak{a}}(s', a)$. Then, we use the value $Q_k^{\mathfrak{b}}(s', a_k^*)$ to update $Q_k^{\mathfrak{a}}$

$$Q_k^{\mathfrak{a}}(s, a) \leftarrow Q_k^{\mathfrak{a}}(s, a) + \alpha\Big(r_k + \gamma Q_k^{\mathfrak{b}}\big(s', a_k^*\big) - Q_k^{\mathfrak{a}}(s, a)\Big) \tag{7}$$

instead of using the value $Q_k^{\mathfrak{a}}(s', a_k^*) = \max_a Q_k^{\mathfrak{a}}(s', a)$ to update $Q_k^{\mathfrak{a}}$ in IQL. The updating of $Q_k^{\mathfrak{b}}$ is similar to this.

Here, two multilayer neural networks are used to fit the two Q-functions, which are expressed as $Q_k^{\mathfrak{a}}(s, a; \boldsymbol{\theta}_t)$ and $Q_k^{\mathfrak{b}}(s, a; \boldsymbol{\theta}_t')$, respectively. Usually, the latter is called the target Q-function (or target network). The update mode is similar to the one of deep double Q-learning [28] and the target value $Y_{k,t} \equiv r_{k,t+1} + \gamma Q_k^{\mathfrak{a}}(s_{t+1}, \text{argmax}_a\, Q_k^{\mathfrak{a}}(s_{t+1}, a; \boldsymbol{\theta}_t), \boldsymbol{\theta}_t')$. In order to make the target network update smoother, we adopt the soft target update [14] instead of copying the network weights directly [13]

$$\boldsymbol{\theta}' \longleftarrow \tau\boldsymbol{\theta} + (1-\tau)\boldsymbol{\theta}' \tag{8}$$

where $\tau \ll 1$. The soft update method makes the weights of the target Q-function change slowly, so does the target values. Compared with the direct copy of the weights, the soft update method can enhance the learning stability [13].

To balance exploration and exploitation, the UCB exploration strategy is used to select an action to be performed by the agent k

$$a_k = \underset{c\in\boldsymbol{A}_k}{\text{argmax}}\left\{Q_k(s_k, c) + \sqrt{\frac{\ln R_{s_k}}{R_{s_k,c}}}\right\} \tag{9}$$

where R_{s_k} denotes the number of times state s_k has been visited and $R_{s_k,c}$ denotes the number of times action c has been chosen in this state until now. If action c has been chosen rarely in some states, then the second term will dominate the first term and action c will be explored. As learning progresses, the first term dominates the second term and the UCB strategy ultimately becomes a greedy one. Although the ϵ-greedy strategy is easier to implement for problems with larger state space, we prefer the UCB strategy if possible, since in the preliminary test, we observe that the UCB strategy is slightly better than the ϵ-greedy strategy [32]. From the perspective of exploration mechanism, the exploratory action selection for UCB is based on both the learned Q-values and the number of times an action has been chosen in the past, hence it tends to be more inclined to explore those actions that are rarely explored.

In this method, agent k just regards other agents as a part of the environment. Therefore, this method ignores the dynamic resulting from the actions of the other agents and the convergence is not guaranteed. In order to learn better cooperative

strategies and make the learning process more stable and robust, we introduce Co-DQL, which uses mean-field approximation, a new reward allocation mechanism, and local state sharing method.

B. Cooperative Double Q-Learning Method

With the number of agents increasing, the dimension of joint action $\boldsymbol{a}$ increases exponentially, so when the number of agents is relatively large, it is often not feasible to directly calculate the joint action function $Q_k(s, \boldsymbol{a})$ for each agent k. Mean-field approximation is first proposed in [27] to deal with the problem. Its core idea is that the interactions within the population of agents are approximated by those between an agent and the average of its neighboring agents.[1] Specifically, a very natural approach is to decompose the joint action-value function as follows:

$$Q_k(s_k, \boldsymbol{a}) = E_{l\sim d}[Q_k(s_k, a_k, a_l)] \tag{10}$$

where d is the uniform distribution on the index set $\mathcal{N}(k)$, which is the set of the neighboring agents of agent k and the size of the index set is $N_k = |\mathcal{N}(k)|$. Suppose that each agent has C discrete actions $\{1, 2, \ldots, C\}$. Then, the action a_k of agent k can be coded using one hot, namely, $a_k \triangleq [a_{k,1}, a_{k,2}, \ldots, a_{k,C}]$, where each component corresponds to a possible action, and obviously at any time only one component is one and the others are zero. Hence, the mean action $\overline{a}_k$ can be expressed as $\overline{a}_k \triangleq [\overline{a}_{k,1}, \overline{a}_{k,2}, \ldots, \overline{a}_{k,C}]$, where each component $\overline{a}_{k,i} = E_{l\sim d}[a_{l,i}]$ for $i \in \{1, 2, \ldots, C\}$, simply recorded as $\overline{a}_k = E_{l\sim d}[a_l]$. Intuitively, $\overline{a}_k$ can be seen as the empirical distribution of the actions taken by the neighbors of agent k [27]. Naturally, there is the following relationship between the one-hot coding action a_l of agent l and the mean action:

$$a_l = \overline{a}_k + \delta_{l,k} \tag{11}$$

where $\delta_{l,k}$ is a small fluctuation. Under the premise of twice differentiable, using the Taylor expansion theory, the mean-field approximation is expressed by the following formulation on the basis of (10):

$$\begin{aligned} Q_k(s_k, \boldsymbol{a}) &= E_{l\sim d}[Q_k(s_k, a_k, a_l)] \\ &= E_{l\sim d}\Big[Q_k(s_k, a_k, \overline{a}_k) + \nabla Q_k(s_k, a_k, \overline{a}_k)\cdot\delta_{l,k} \\ &\qquad + \frac{1}{2}\delta_{l,k}\cdot\nabla^2 Q_k\big(s_k, a_k, \xi_{l,k}\big)\cdot\delta_{l,k}\Big] \\ &= Q_k(s_k, a_k, \overline{a}_k) + \nabla Q_k(s_k, a_k, \overline{a}_k)\cdot E_{l\sim d}[\delta_{l,k}] \\ &\quad + \frac{1}{2}E_{l\sim d}\Big[\delta_{l,k}\cdot\nabla^2 Q_k(s_k, a_k, \xi_{l,k})\cdot\delta_{l,k}\Big] \\ &= Q_k(s_k, a_k, \overline{a}_k) + \frac{1}{2}E_{l\sim d}[R_k(a_l)] \\ &\approx Q_k(s_k, a_k, \overline{a}_k) \end{aligned} \tag{12}$$

where $E_{l\sim d}[\delta_{l,k}] = 0$ is easily known from (11), and $R_k(a_l) \triangleq \delta_{l,k}\cdot\nabla^2 Q_k(s_k, a_k, \xi_{l,k})\cdot\delta_{l,k}$ denotes the Taylor polynomial's remainder with $\xi_{l,k} = \overline{a}_k + \epsilon_{l,k}\cdot\delta_{l,k}$ and $\epsilon_{l,k} \in [0, 1]$ [27]. Under some mild conditions, it can be proved that $R_k(a_l)$ is a random variable close to zero and can be omitted [27]. For large-scale TSC, this way of implicit modeling the behavior of other agents has great advantages, which makes the input dimension of each agent k's Q-function drastically reduce, and the joint action dimension decreases from C^{N_k} to constant C^2. It is worth noting that we only need to pay attention to the actions of the current time step, rather than the historical behavior of the neighbors. This is mainly due to the fact that the traffic state dynamics is Markovian, which will be further discussed in Section IV-A.

For partially observable Markov traffic scenarios, each agent k can obtain its own reward r_k and local observation s_k at each time step. The goal of MARL in a cooperative situation is to maximize the global benefits or minimize the regrets.[2] However, there may be the so-called credit assignment problem [33] in MARL, so each agent often does not directly regard the global reward as its reward. Instead, we set each agent to maintain its own reward. In addition, if each agent only considers its own immediate reward, then the agent may become selfish, which may be harmful to cooperation. Based on the above considerations, we propose to allocate each agent's reward according to the following formulation:

$$\hat{r}_k = r_k + \alpha\cdot\sum_{i\in\mathcal{N}(k)} r_i \tag{13}$$

where $\alpha \in [0, 1]$ is a discount factor that can be flexibly used to balance selfishness and cooperation. If α is set to 0, then each signal agent only considers the immediate reward of its own intersection, greedily maximizing the throughput of its own intersection, which may damage the global reward of the road network; if α is set to 1, this means that each agent may obtain the global reward and suffers from the credit assignment problem as described earlier. Specifically, we make $0 < \alpha < 1$. The idea behind is as follows. For each signal agent k, despite the action selection may be not always beneficial to the neighboring agents, the reallocated reward received after an action depends on its own immediate reward and the immediate reward of the neighboring agents. Once the immediate rewards of the neighboring agents are low, the second term of (13) will take a small value which means the action taken by signal agent k may be not so great for the neighboring agents. While higher immediate rewards of the neighboring agents will encourage signal agent k and accordingly the second term of (13) will take a larger value. This reward allocation mechanism in (13) in turn affects the action selection of agent k, with the aim of maximizing the global reward of the road network. The reward allocation mechanism is similar to the one mentioned in [23], but we do not strictly limit the distance between agent k and the neighboring agents.

The local state sharing method is described below. For agent k, the average of the local state of its neighboring agents is taken as the additional input of agent k's action-value

[1]The neighborhood size is a user-specific parameter. It can take a value from [1, N], where N is the total number of agents.

[2]In this article, regrets refer to the waiting time of vehicles, the length of queues, etc.

function. Hence, the state of agent k can be represented as

$$\hat{s}_k = \left\langle s_k, \frac{1}{N_k} \sum_{i \in \mathcal{N}(k)} s_i \right\rangle \tag{14}$$

where $\hat{s}_k$ represents agent k's joint state. This method implicitly shares state information among agents, and if the dimension of local state is assumed to be $|s|$, its joint dimension is constant $|s|^2$, which is independent of the number of agents.

Based on the above introduction, the Co-DQL algorithm is proposed. Compared with the centralized control method [16], [34], this algorithm reduces the joint input dimension of the action-value function from $C^{N_k} \cdot |s|^{N_k}$ to $C^2 \cdot |s|^2$ at the cost of a small amount of communication and calculation [27], which avoids the curse of dimension in large-scale problems. The pseudocode of Co-DQL is given in Algorithm 2. In this algorithm, multilayer perceptions parameterized by ϕ and ϕ_- are used to represent the two action-value functions of each agent. Co-DQL works as follows.

Step 0 (Initialize): For each $k = 1, \ldots, N$, initialize neural-network parameters ϕ_k, $\phi_{-,k}$ and initialize mean action $\overline{a}_k$ for agent k.

Step 1 (Check the Termination Condition): If a problem-specific stopping condition is met, stop and save the training neural-network model.

Step 2 (Select Action): For each $k = 1, \ldots, N$, according to the current observation $\hat{s}_k$ of agent k, select action a_k under the UCB policy.

Step 3 (Execute Action): For each $k = 1, \ldots, N$, agent k executes action a_k (all agents execute action synchronously), gets immediate reward r_k, and next state observation s'_k.

Step 4 (Obtain Samples): For each $k = 1, \ldots, N$, compute the mean action $\overline{a}_k$, reward $\hat{r}_k$ after reallocation, and next local state $\hat{s}'_k$ after sharing.

Step 5 (Store Samples in Buffer): For each $k = 1, \ldots, N$, store the results of step 4 as a tuple sample $\langle \hat{\boldsymbol{s}}, \boldsymbol{a}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{s}}', \overline{\boldsymbol{a}} \rangle$ in replay buffer $\mathcal{D}_k$; if the number of samples stored in the $\mathcal{D}_k$ is less than the minimum number of samples required for training, go to step 1; otherwise, the next step is executed sequentially.

Step 6 (Compute Sample Target Values): For each $k = 1, \ldots, N$, M samples are randomly extracted from $\mathcal{D}_k$ and the target value $Y_k^{\text{Co-DQL}}$ is calculated according to the sample data.

Step 7 (Update Neural-Network Parameters): For each $k = 1, \ldots, N$, the gradient of the parameter ϕ_k is obtained from the loss function, and ϕ_k is updated according to the learning rate, then $\phi_{-,k}$ is softly updated with update rate τ. Go to step 1.

For most RL algorithms, the termination condition is generally set to be that the number of episodes experienced by agents reaches the preset number. The preset number of episodes is usually selected according to the training situation of the algorithm in the given problem.

Algorithm 2: Co-DQL

Input: Initial parameters ϕ and mean action $\overline{a}$ for all agents

Output: Parameters ϕ for all agents

Initialize $Q_k^{\mathfrak{a}}(\cdot|\phi)$, $Q_k^{\mathfrak{b}}(\cdot|\phi_-)$ and $\overline{a}_k$ for all $k \in \{1, \ldots, N\}$

while *not termination condition* **do**

  For each agent k , select action a_k using the UCB exploration strategy from (9)

  Take the joint action $\boldsymbol{a} = (a_1, \ldots a_N)$ and observe the reward $\boldsymbol{r} = (r_1, \ldots, r_N)$ and the next observations $\boldsymbol{s}' = (s'_1, \ldots, s'_N)$

  Compute $\overline{\boldsymbol{a}}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{s}}$ and $\hat{\boldsymbol{s}}'$

  Store $\langle \hat{\boldsymbol{s}}, \boldsymbol{a}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{s}}', \overline{\boldsymbol{a}} \rangle$ in replay buffer $\mathcal{D}$

  for $k = 1$ *to* N **do**

    Sample M experiences $\langle \hat{\boldsymbol{s}}, \boldsymbol{a}, \hat{\boldsymbol{r}}, \hat{\boldsymbol{s}}', \overline{\boldsymbol{a}} \rangle$ from $\mathcal{D}$

    Compute target value $Y_k^{\text{Co-DQL}}$ by (16)

    Update the Q network by minimizing the loss

$$\mathcal{L}(\phi_k) = \frac{1}{M} \sum \left(Q_k^{\mathfrak{a}}(\hat{s}_k, a_k, \overline{a}_k; \phi) - Y_k^{\text{Co-DQL}} \right)^2$$

  Update the parameters of the target network for each agent k with updating rate τ

$$\phi_k^- \leftarrow \tau \phi_k + (1 - \tau) \phi_{,k}$$

The action-value function $Q_k^{\mathfrak{a}}(\cdot|\phi)$ (parameterized by ϕ) is trained by minimizing the loss

$$\ell(\phi_k) = \left(Q_k^{\mathfrak{a}}(\hat{s}_k, a_k, \overline{a}_k; \phi) - Y_k^{\text{Co-DQL}} \right)^2 \tag{15}$$

where $Y_k^{\text{Co-DQL}}$ is the target value of agent k and is calculated by the following formulation:

$$Y_k^{\text{Co-DQL}} = \hat{r}_k + \gamma Q_k^{\mathfrak{b}}\left(\hat{s}'_k, \text{argmax}_{a_k} Q_k^{\mathfrak{a}}(\hat{s}'_k, a_k, \overline{a}_k; \phi), \overline{a}'_k; \phi_-\right). \tag{16}$$

In Co-DQL, the mean-field approximation makes every independent agent learn the awareness of collaboration with the others. Moreover, the reward allocation mechanism and the local state sharing method of agents improve the stability and robustness of the training process compared with the independent agent learning method.

In order to theoretically support the effectiveness of our proposed Co-DQL algorithm, we provide the convergence proof under some assumptions in the next section.

C. Convergence Analysis

In the previous literature, the convergence of mean-field Q-learning under the set of tabular Q-functions and the convergence of when Q-function is represented by other function approximators have been proved [35], [27]. Under similar constraints, we develop the convergence proof of Co-DQL, which is the mean-field RL with double estimators.

Assuming that there are only a limited number of state–action pairs, for each agent k, we can write updating rules

of two functions $Q_k^{\mathfrak{a}}$ and $Q_k^{\mathfrak{b}}$ of agent k according to Sections III-A and III-B

$$\begin{aligned} Q_k^{\mathfrak{a}}(s,a_k,\overline{a}_k) &\leftarrow (1-\alpha)Q_k^{\mathfrak{a}}(s,a_k,\overline{a}_k) \\ &\quad + \alpha\Big(r+\gamma Q_k^{\mathfrak{b}}\big(s',\mathfrak{a}_k^*,\overline{a}_k\big)\Big) \\ Q_k^{\mathfrak{b}}(s,a_k,\overline{a}_k) &\leftarrow (1-\alpha)Q_k^{\mathfrak{b}}(s,a_k,\overline{a}_k) \\ &\quad + \alpha\big(r+\gamma Q_k^{\mathfrak{a}}\big(s',\mathfrak{b}_k^*,\overline{a}_k\big)\big) \end{aligned} \tag{17}$$

where $\mathfrak{a}_k^* = \operatorname{argmax}_{a_k} Q_k^{\mathfrak{a}}(s',a_k,\overline{a}_k)$ and $\mathfrak{b}_k^* = \operatorname{argmax}_{a_k} Q_k^{\mathfrak{b}}(s',a_k,\overline{a}_k)$. At any update time step, either of the two of (17) is updated. Our goal is to prove that both $\boldsymbol{Q}^{\mathfrak{a}} = (Q_1^{\mathfrak{a}},\ldots,Q_N^{\mathfrak{a}})$ and $\boldsymbol{Q}^{\mathfrak{b}} = (Q_1^{\mathfrak{b}},\ldots,Q_N^{\mathfrak{b}})$ converge to Nash Q-values. Our proof follows the convergence proof framework of single-agent double Q-learning [24], and we use the following assumptions and lemma.

Assumption 1: Each action-value pair is visited infinitely often, and the reward is bounded by some constant K.

Assumption 2: Agent's policy is greedy in the limit with infinite exploration (GLIE). In the case with the Boltzmann policy, the policy becomes greedy with respect to the Q-function in the limit as the temperature decays asymptotically to zero.

Assumption 3: For each stage game $[Q_t^1(s),\ldots,Q_t^N(s)]$ at time t and in state s in training, for all $t,s,j\in\{1,\ldots,N\}$, the Nash equilibrium $\boldsymbol{\pi}_* = [\pi_*^1,\ldots,\pi_*^N]$ is recognized either as: 1) the global optimum or 2) a saddle point expressed as:

1) $\mathbb{E}_{\pi_*}[Q_t^j(s)] \geq \mathbb{E}_{\pi}[Q_t^j(s)] \quad \forall\pi\in\Omega(\prod_k \mathcal{A}^k)$;
2) $\mathbb{E}_{\pi_*}[Q_t^j(s)] \geq \mathbb{E}_{\pi^j}\mathbb{E}_{\pi_*^{-j}}[Q_t^j(s)] \quad \forall\pi^j\in\Omega(\mathcal{A}^j)$ and $\mathbb{E}_{\pi_*}[Q_t^j(s)] \leq \mathbb{E}_{\pi_*^j}\mathbb{E}_{\pi^{-j}}[Q_t^j(s)] \quad \forall\pi^{-j}\in\Omega(\prod_{k\neq j}\mathcal{A}^k)$.

Lemma 1: The random process $\{\Delta_t\}$ defined in $\mathbb{R}$ as

$$\Delta_{t+1}(x) = (1-\alpha_t(x))\Delta_t(x) + \alpha_t(x)F_t(x)$$

converges to zero with probability 1 (w.p.1) when:

1) $0\leq\alpha_t(x)\leq 1, \sum_t \alpha_t(x)=\infty, \sum_t \alpha_t^2(x)<\infty$;
2) $x\in\mathscr{X}$, the set of possible states, and $|\mathscr{X}|<\infty$;
3) $\|\mathbb{E}[F_t(x)|\Im_t]\|_W \leq \gamma\|\Delta_t\|_W + c_t$, where $\gamma\in[0,1)$ and c_t converges to zero w.p.1;
4) $\operatorname{var}[F_t(x)|\Im_t]\leq K(1+\|\Delta_t\|_W^2)$ with constant $K>0$.

Here, $\mathscr{F}_t$ denotes the filtration of an increasing sequence of σ-fields including the history of processes; $\alpha_t,\Delta_t,F_t\in\mathscr{F}_t$ and $\|\cdot\|_W$ is a weighted maximum norm [30].

Proof: Similar to the proof of [36, Th. 1] and [37, Corollary 5]. ∎

Our theorem and proof sketches are as follows.

Theorem 1: In a finite-state stochastic game, if Assumptions 1–3 and Lemma 1's first and second conditions are met, then both $\boldsymbol{Q}^{\mathfrak{a}}$ and $\boldsymbol{Q}^{\mathfrak{b}}$ as updated by the rule of Algorithm 2 in (17) will converge to the Nash Q-value $\boldsymbol{Q}^* = (Q_1^*,\ldots,Q_N^*)$ with probability one.

Proof: We need to show that the third and fourth conditions of Lemma 1 hold so that we can apply it to prove Theorem 1. Obviously, the updates of functions $\boldsymbol{Q}^{\mathfrak{a}}$ and $\boldsymbol{Q}^{\mathfrak{b}}$ are symmetrical, so as long as one of them is proved to converge, the other must converge. By subtracting two sides of (17) by $\boldsymbol{Q}^*$, and then the following formula can be obtained by comparing with the equation in Lemma 1:

$$\boldsymbol{\Delta}_t(s,\boldsymbol{a}) = \boldsymbol{Q}_t^{\mathfrak{a}}(s,\boldsymbol{a}) - \boldsymbol{Q}_*(s,\boldsymbol{a})$$

$$\boldsymbol{F}_t(s_t,\boldsymbol{a}_t) = \boldsymbol{r}_t + \gamma\boldsymbol{Q}_t^{\mathfrak{b}}\big(s_{t+1},\mathfrak{a}^*\big) - \boldsymbol{Q}_*(s_t,\boldsymbol{a}_t) \tag{18}$$

where $\mathfrak{a}^* = \operatorname{argmax}_{\boldsymbol{a}} \boldsymbol{Q}^{\mathfrak{a}}(s_{t+1},\boldsymbol{a}_t,\overline{\boldsymbol{a}}_t)$. Let $\Im_t = \{\boldsymbol{Q}_0^{\mathfrak{a}},\boldsymbol{Q}_0^{\mathfrak{b}},s_0,\boldsymbol{a}_0,\alpha_0,\boldsymbol{r}_1,s_1,\ldots,s_t,\boldsymbol{a}_t\}$ denote the σ-fields generated by all random variables in the history of the stochastic game up to time t. Note that $\boldsymbol{Q}_t^{\mathfrak{a}}$ and $\boldsymbol{Q}_t^{\mathfrak{b}}$ are two random variables derived from the historical trajectory up to time t. Given the fact that all $\boldsymbol{Q}_\tau^{\mathfrak{a}}$ and $\boldsymbol{Q}_\tau^{\mathfrak{b}}$ with $\tau<t$ are $\mathscr{F}_t$-measurable, both $\boldsymbol{\Delta}_t$ and $\boldsymbol{F}_t$ are, therefore, also $\mathscr{F}_t$-measurable.

Since the reward is bounded by some constant K in Assumption 1, then $\operatorname{Var}[\boldsymbol{r}_t]<\infty$, the fourth condition in the lemma holds.

Next, we show that the third condition of the lemma holds. We can rewrite (18) as follows:

$$\boldsymbol{F}_t(s_t,\boldsymbol{a}_t) = \boldsymbol{F}_t^Q(s_t,\boldsymbol{a}_t) + \gamma\Big(\boldsymbol{Q}_t^{\mathfrak{b}}\big(s_{t+1},\mathfrak{a}^*\big) - \boldsymbol{Q}_t^{\mathfrak{a}}\big(s_{t+1},\mathfrak{a}^*\big)\Big) \tag{19}$$

where $\boldsymbol{F}_t^Q = \boldsymbol{r}_t + \gamma\boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{a}^*) - \boldsymbol{Q}^*(s_t,\boldsymbol{a}_t)$ is the value of $\boldsymbol{F}_t$ if normal MF-Q would be under consideration. In [17], $\|\mathbb{E}[\boldsymbol{F}_t^Q|\Im_t]\|_W \leq \gamma\|\Delta_t\|_W$ has been proved, so in order to meet the third condition, we identify $c_t = \gamma(\boldsymbol{Q}_t^{\mathfrak{b}}(s_{t+1},\mathfrak{a}^*) - \boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{a}^*))$ and it is sufficient to show that $\boldsymbol{\Delta}_t^{\mathfrak{ba}} = \boldsymbol{Q}_t^{\mathfrak{b}} - \boldsymbol{Q}_t^{\mathfrak{a}}$ converges to zero. The update of $\boldsymbol{\Delta}_t^{\mathfrak{ba}}$ depends on whether $\boldsymbol{Q}^{\mathfrak{b}}$ or $\boldsymbol{Q}^{\mathfrak{a}}$ is updated, so

$$\begin{aligned} \boldsymbol{\Delta}_{t+1}^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t) &= \boldsymbol{\Delta}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t) + \alpha_t\boldsymbol{F}_t^{\mathfrak{b}}(s_t,\boldsymbol{a}_t), \text{ or} \\ \boldsymbol{\Delta}_{t+1}^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t) &= \boldsymbol{\Delta}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t) - \alpha_t\boldsymbol{F}_t^{\mathfrak{b}}(s_t,\boldsymbol{a}_t) \end{aligned} \tag{20}$$

where $\boldsymbol{F}_t^{\mathfrak{a}}(s_t,\boldsymbol{a}_t) = \boldsymbol{r}_t + \gamma\boldsymbol{Q}_t^{\mathfrak{b}}(s_{t+1},\mathfrak{a}^*) - \boldsymbol{Q}_t^{\mathfrak{a}}(s_t,\boldsymbol{a}_t)$ and $\boldsymbol{F}_t^{\mathfrak{b}}(s_t,\boldsymbol{a}_t) = \boldsymbol{r}_t + \gamma\boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{b}^*) - \boldsymbol{Q}_t^{\mathfrak{b}}(s_t,\boldsymbol{a}_t)$. We define $\xi_t^{\mathfrak{ba}} = (1/2)\alpha_t$, then

$$\begin{aligned} &\mathbb{E}\Big[\boldsymbol{\Delta}_{t+1}^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t)|\Im_t\Big] \\ &\quad= \boldsymbol{\Delta}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t) + \mathbb{E}\Big[\alpha_t\boldsymbol{F}_t^{\mathfrak{b}}(s_t,\boldsymbol{a}_t) - \alpha_t\boldsymbol{F}_t^{\mathfrak{a}}(s_t,\boldsymbol{a}_t)|\Im_t\Big] \\ &\quad= \boldsymbol{\Delta}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t) + \mathbb{E}\Big[\alpha_t\gamma\Big(\boldsymbol{Q}_t^{\mathfrak{a}}\big(s_{t+1},\mathfrak{b}^*\big) - \boldsymbol{Q}_t^{\mathfrak{b}}\big(s_{t+1},\mathfrak{a}^*\big)\Big) \\ &\qquad\qquad - \alpha_t\Big(\boldsymbol{Q}_t^{\mathfrak{b}}(s_t,\boldsymbol{a}_t) - \boldsymbol{Q}_t^{\mathfrak{a}}(s_t,\boldsymbol{a}_t)\Big)|\Im_t\Big] \\ &\quad= \Big(1-\xi_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t)\Big)\boldsymbol{\Delta}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t) \\ &\qquad+ \xi_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t)\mathbb{E}\Big[\boldsymbol{F}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t)|\Im_t\Big] \end{aligned}$$

where $\mathbb{E}[\boldsymbol{F}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t)|\Im_t] = \gamma\mathbb{E}[\boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{a}^*) - \boldsymbol{Q}_t^{\mathfrak{b}}(s_{t+1},\mathfrak{a}^*)|\Im_t]$. At each time step, one of the following two cases must hold.

Case 1: $\mathbb{E}[\boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{b}^*)|\Im_t] \geq \mathbb{E}[\boldsymbol{Q}_t^{\mathfrak{b}}(s_{t+1},\mathfrak{a}^*)|\Im_t]$. We have $\boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{a}^*) = \max\boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\boldsymbol{a}) \geq \boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{b}^*)$, therefore

$$\begin{aligned} |\mathbb{E}\Big[\boldsymbol{F}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t)|\Im_t\Big]| &= \gamma\mathbb{E}\Big[\boldsymbol{Q}_t^{\mathfrak{a}}\big(s_{t+1},\mathfrak{b}^*\big) - \boldsymbol{Q}_t^{\mathfrak{b}}\big(s_{t+1},\mathfrak{a}^*\big)|\Im_t\Big] \\ &\leq \gamma\mathbb{E}\Big[\boldsymbol{Q}_t^{\mathfrak{a}}\big(s_{t+1},\mathfrak{a}^*\big) - \boldsymbol{Q}_t^{\mathfrak{b}}\big(s_{t+1},\mathfrak{a}^*\big)|\Im_t\Big] \\ &\leq \Big\|\boldsymbol{\Delta}_t^{\mathfrak{ba}}\Big\|. \end{aligned}$$

Case 2: $\mathbb{E}[\boldsymbol{Q}_t^{\mathfrak{a}}(s_{t+1},\mathfrak{b}^*)|\Im_t] < \mathbb{E}[\boldsymbol{Q}_t^{\mathfrak{b}}(s_{t+1},\mathfrak{a}^*)|\Im_t]$. We have $\mathbb{E}[\boldsymbol{Q}_t^{\mathfrak{b}}(s_{t+1},\mathfrak{b}^*)|\Im_t] \geq \mathbb{E}[\boldsymbol{Q}_t^{\mathfrak{b}}(s_{t+1},\mathfrak{a}^*)|\Im_t]$. Then

$$\Big|\mathbb{E}\Big[\boldsymbol{F}_t^{\mathfrak{ba}}(s_t,\boldsymbol{a}_t)|\Im_t\Big]\Big| = \gamma\mathbb{E}\Big[\boldsymbol{Q}_t^{\mathfrak{b}}\big(s_{t+1},\boldsymbol{e}^*\big) - \boldsymbol{Q}_t^{\mathfrak{a}}\big(s_{t+1},\mathfrak{b}^*\big)|\Im_t\Big]$$

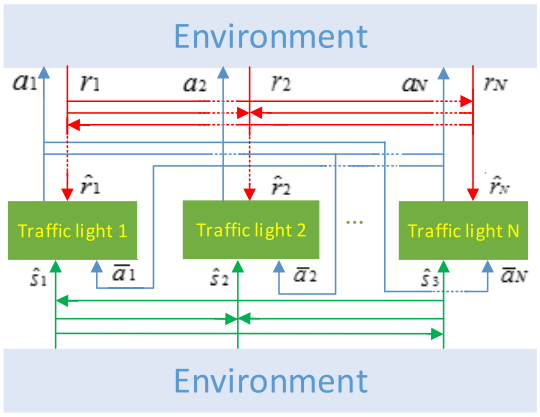

Fig. 1. Architecture diagram of Co-DQL for TSC. For each $k = 1, \ldots, N$, $\hat{s}_k$ denotes the local state information after sharing, $\bar{a}_k$ represents the mean action information, a_k means the action will be executed, and r_k and $\hat{r}_k$ represent the immediate reward before and after reallocation, respectively. The red arrow, blue arrow, and green arrow represent the transfer of reward information, action information, and state information, respectively.

$$\leq \gamma \mathbb{E}\left[\boldsymbol{Q}_t^{\mathfrak{b}}\left(s_{t+1}, \mathfrak{b}^*\right) - \boldsymbol{Q}_t^{\mathfrak{a}}\left(s_{t+1}, \mathfrak{b}^*\right) | \Im_t\right]$$
$$\leq \left\| \boldsymbol{\Delta}_t^{\mathfrak{b}\mathfrak{a}} \right\|.$$

Hence, no matter which of the above cases are hold, we can obtain the satisfactory result, that is, $|\mathbb{E}[\boldsymbol{F}_t^{\mathfrak{b}\mathfrak{a}}(s_t, \boldsymbol{a}_t)|\Im_t]| \leq \|\boldsymbol{\Delta}_t^{\mathfrak{b}\mathfrak{a}}\|$. Then, we can apply Lemma 1 and obtain the convergence of $\boldsymbol{\Delta}_t^{\mathfrak{b}\mathfrak{a}}$ to 0, the third condition is thus hold. Finally, with all conditions are satisfied, Theorem 1 is proved. ■

IV. Application of Co-DQL to TSC

This section first uses MDP notations to represent the key elements of the TSC problem so that MARL can be used in TSC. To facilitate the training and evaluation of the MARL model applied to TSC problem, we also introduce the TSC simulators.

A. Description of TSC Based on MDP Notations

Although we model the entire traffic network in a decentralized way as a multiagent structure, the global state of the entire traffic system is still Markov, namely, the next state only depends on the current state

$$s_{t+1} = f(s_t, \boldsymbol{a}_t) \tag{21}$$

where s_t and s_{t+1} denote the state of traffic system at time step t and $t+1$, and $\boldsymbol{a}_t$ denotes the joint action of the traffic system at time step t. Therefore, it can be modeled using the framework of MARL described in Section II-B.

When to cope with the TSC problem, there are many different MDP settings. Their differences lie in the definition of action space, state space, reward function, etc. [11], [23], [38], [39], [40], [41]. Here, we focus on the following two kinds of MDP settings. Note that it may be potential to extend our method to other kinds of settings. Since the source code of our method is open,[3] interested readers can try to test or extend it to deal with other kinds of MDP settings.

[3]https://github.com/Brucewangxq/larger_real_net

1) Simplified MDP Setting for the TSC Problem: Suppose a road network has N signalized intersections, that is, N signal agents. The action of signal agent k at time step t can be written as $a_{k,t}$, and its local observation or state is $s_{k,t}$. We set the signal agent's actions at each intersection has only two possible cases {0, 1}: 1) green traffic lights for incoming traffic in the north and south directions and 2) red traffic lights in the east and west directions at the same time, or contrary to that, so the action space is $\{0, 1\}^N$. The local state, which is the observation vector $s_{k,t}$, is the waiting queue density (or queue length) on all the one-way lanes (or edges) connected to the intersection k: $s_{k,t} = [q[kn], q[ks], q[kw], q[ke]]$, where $q[kn]$, $q[ks]$, $q[kw]$, and $q[ke]$ represent the waiting queue density in four directions related to intersection k, respectively, and they are the lanes of vehicles driving in the direction of intersection k. The value space of each of them can be expressed as $\{0, 1, 2, \ldots, \max_q\}$, where $\max_q$ is the maximum capacity of vehicles on a lane between every two intersections. For the peripheral signal agent of the system, if there is no road connected to it in a certain direction, the number of vehicles in that direction is always zero. For simplicity, it is assumed that a normally traveling vehicle has the same speed and can start or stop immediately.

For any signal agent k, the reward at time step t can be calculated by the number of vehicles waiting on all lanes toward the intersection, that is

$$r_{k,t} = -\sum_{j \in \{n,s,w,e\}} \left| q_t[kj] \right| \tag{22}$$

where $q_t[kj]$ is the number of vehicles that have zero speed on lane kj leading to intersection k. To avoid changing traffic signal too frequently, the action can be taken every Δt time steps, that is, a Markov state transition occurs only once every Δt time steps. Then, from the T-th to $T+1$th state transition, the signal agent obtains the sum of the rewards in Δt time steps, that is

$$\boldsymbol{R}_T = \sum_{t=(T-1)\Delta t}^{T\Delta t - 1} \boldsymbol{r}_t(s_t, \boldsymbol{a}_t). \tag{23}$$

Our goal is to minimize the total waiting time of vehicles in the traffic network

$$\max_{\boldsymbol{\pi}} J = \mathbb{E}\left[\sum_{T=1}^{T_{\max}} \gamma^{T-1} \left(\frac{1}{\Delta t} \sum_{t=(T-1)\Delta t}^{T\Delta t - 1} \boldsymbol{r}_t(s_t, \boldsymbol{a}_t) \right)\right] \tag{24}$$

where $T_{\max}$ denotes the total number of state transitions, and the joint action $\boldsymbol{a}$ changes every Δt time steps.

In this simplified situation, all other agents are treated as the neighboring agents of each agent. Fig. 1 shows how to apply Co-DQL to TSC. The input information of each agent includes the shared local state information and the mean action information calculated from actions of the neighboring agents in the previous time step. Each agent receives a reallocated reward after performing an action.

2) More Realistic MDP Setting for TSC Problem: In the literature of RL for TSC, there are several standard action definitions, such as phase duration [38]; phase switch [39], [40]; and phase itself [11], [23], [41]. Here, we follow the last

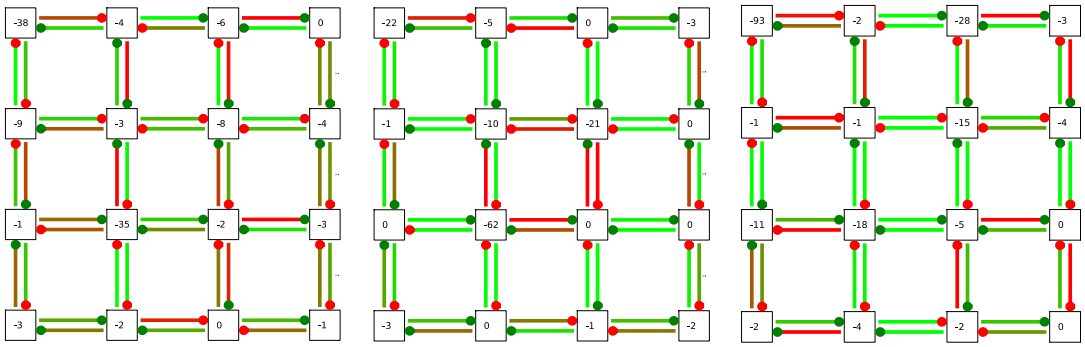

Fig. 2. Illustration of the grid traffic signal system simulator. (a) Global random traffic flow. (b) Double-ring traffic flow. (c) Four-ring traffic flow. Each rectangle represents a signalized intersection and each two adjacent intersections are connected by two one-way lanes. The color of each lane implies the level of congestion and the number in the rectangle represents the immediate reward for the intersection.

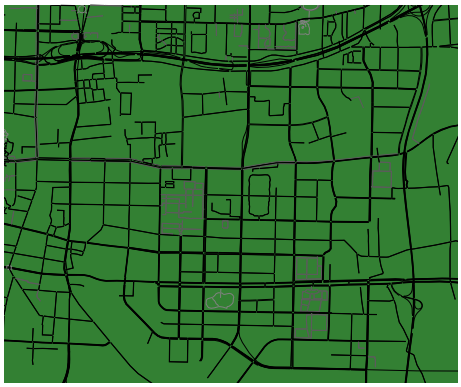

Fig. 3. Overall view of the realistic road network with asymmetric geometry.

definition and predefine a set of feasible phases for each signal agent. Specifically, we adopt the definition of feasible phases in [23], which defines five feasible phases for each signal agent, including east-west straight, east-west left-turn, and three straight and left-turn for east, west, and north-south. These five feasible phases constitute the action space, each phase corresponds to an action. Each signal agent selects one of them to implement for a duration of Δt at each Markov time step. In addition, a yellow time $t_y < \Delta t$ is enforced after each phase switch to ensure safety.

After comprehensively understanding a variety of commonly used state definitions [23], [38], [41], we tend to follow the one in [23] and define local state as

$$s_{k,t} = \left\{\text{wait}_{k,t}[\text{lane}],\ \ \text{wave}_{k,t}[\text{lane}]\right\} \tag{25}$$

where lane is each incoming lane of intersection k. wait measures the cumulative delay [s] of the first vehicle and *wave* measures the total number [veh] of approaching vehicles along each incoming lane. In our experiment, we use `laneAreaDetector` in simulation of urban mobility (SUMO) [42], [43] to obtain the state information, and in practice, the state information can be obtained by near-intersection induction-loop detectors as described in [23].

Similar to the definition of reward in the simplified TSC problem mentioned earlier, we also further consider the cumulative delay of the first car as a regularizer

$$r_{k,t} = -\sum_{\text{lane}} \left| q_{k,t+\Delta t}[\text{lane}] + \beta \cdot \text{wait}_{k,t+\Delta t}[\text{lane}] \right| \tag{26}$$

where β is the regularization rate and typically chosen to approximately scale different reward terms into the same range. Note that the rewards are only measured at time $t+\Delta t$. Compared to other reward definitions, such as wave [38] and appropriateness of green time [44], the reward we defined emphasizes traffic congestion and travel delay, and it is directly correlated to state and action [23].

B. Description of the Simulation Platform

1) Simplified TSC Simulator: The simulation platform used in Section V-B is a grid TSC system based on OpenAI-gym [45]. There are three different scenarios in the experiment: 1) global random traffic flow; 2) double-ring traffic flow; and 3) four-ring traffic flow, which correspond to the three subfigures of Fig. 2, respectively.

Each rectangle denotes a signalized intersection and the number in the rectangle represents the immediate reward for the intersection. Every two adjacent intersections are connected by two one-way lanes. The color of each lane in the picture ranges from green to red, which vaguely means the number of vehicles waiting (at zero speed) on the lane, that is, the level of congestion. Green means unimpeded and red indicates serious congestion. During the operation of the simulator, a certain number of vehicles will be generated at each time step and scattered randomly in the road network. Every newly generated vehicle will have a randomly generated route according to a certain rule, and the vehicle will follow the route and finally the vehicle will be removed from the road network when it reaches the destination.

Among these three scenarios, the one difference is that the rules of generating a driving route of a vehicle, which results in different levels of congestion at different intersections. This can simulate the real information of the traffic flow between the main and secondary roads in the city. In the actual traffic

TABLE I
PARAMETER SETTINGS FOR SIMULATOR

Parameter Type	Value [*unit of measure*]
Normal driving time between two nodes	5 [t]
Initial vehicles in simulator	100 [*veh*]
New vehicles added	5;4;3 [*veh/t*]
Shortest route length	2 [n]
Longest route length	20 [n]
Signal agent action time interval	4 [t]
Initial random seed number	10

t means discrete time step, *veh* is the abbreviation of vehicle, n denotes node, i.e. intersection. 5;4;3 [*veh/t*] means that the number of new vehicles added per time step is optional and can be set to 5,4 or 3 as needed.

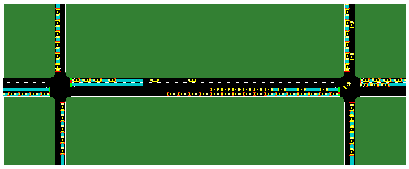

Fig. 4. Local view of two adjacent intersections of the realistic road network.

network, serious congestion does often occurs only in certain specific sections. The other difference is that the number of new vehicles added per time step is various, which can be used to simulate different levels of traffic congestion.

The primary parameters of the simulator are listed in Table I. The normal driving time between two intersections, that is, the distance between two intersections, indicates that normal driving vehicles need five time steps from one intersection to an adjacent intersection. The initial number (note that it is not the number after resetting the simulator when training model) of vehicles in a simulator is used to obtain random seeds. The shortest route length is 2, which means that the shortest distance that a vehicle generated in the simulator can travel is two intersections. The longest route length is 20, which means that the longest distance that a vehicle generated in the simulator can travel is 20 intersections. The action time interval of the signal agent is 4, which means that a signal agent must keep at least four time steps before it can change one action.

2) More Realistic TSC Simulator: We take the road network in some areas of Xi'an as the prototype of the real road network to design a TSC simulator based on SUMO, which has 49 signalized intersections on the real road network. Figs. 3 and 4 show the overall road network view and a local view of two adjacent intersections, respectively. The cars driving on the road network have the following properties: the length is 5 m, the acceleration is 5 m/s, and the deceleration is 10 m/s. As for the setting of signal agents' action time interval Δt, as discussed in [23], if Δt is too long, the signal agent will not be adaptive enough, if Δt is too short, the agent decision will not be delivered on time due to computational cost and communication latency, and it may be unsafe since the action is switched too frequently. Some recent works suggested $\Delta t = 10$ s, $t_y = 5$ s [38], $\Delta t = 5$ s, and $t_y = 2$ s [23]. We adopt the latter setting in the simulator to ensure that each signal agent is more adaptive.

In order to evaluate the robustness and optimality of algorithms in a challenging TSC scenario, we design intensive, stochastic, time-variant traffic flows to simulate the peak-hour traffic, instead of fixed congestion levels in the simplified TSC simulator. The simulation time of each episode is 60 min and we set up four traffic flow groups. Specifically, four traffic flow groups are generated as multiples of "unit" flows 1100, 660, 920, and 552 veh/h. The first two traffic flows are simulated during the first 40 min, as $[0.4, 0.7, 0.9, 1.0, 0.75, 0.5, 0.25]$ unit flows with 5-min intervals while the last two traffic flows are generated during a shifted time window from 15 to 55 min, as $[0.3, 0.8, 0.9, 1.0, 0.8, 0.6, 0.2]$ unit flows with 5-min intervals.

V. NUMERICAL EXPERIMENTS AND DISCUSSION

A. Implementation Details of Algorithms

In order to analyze the performance of the proposed algorithm, we compared it with several popular RL methods in the same traffic scenarios. Details of the implementation of Co-DQL and the other methods are described as follows.

Co-DQL: The procedure described in Section III-B is implemented. The multilayer fully connected neural network is used to approximate the Q-function of each agent. We use the ReLU-activation between hidden layers, and transform the final output of Q-network with it. All agents share the same Q-network, the shared Q-network takes an agent embedding as input and computes Q-value for each candidate action. We also feed in the action approximation $\overline{a}_k$ and sharing joint state $\hat{s}_k$. We use the Adam optimizer with a learning rate of 0.0001. The discounted factor γ is set to 0.95, the mini-batch size is 1024, and the reward allocation factor α is set to $1/n$, where n represents the number of neighbor agents. The size of the replay buffer is 5×10^5 and $\tau = 0.01$ for updating the target networks. The network parameters will be updated once episode samples are added to the replay buffer.

MA2C: The state-of-the-art MARL (decentralized) algorithms for large-scale TSC. The hyperparameters of the algorithm in the experiment are basically consistent with the original one [23].

IQL: It has almost the same hyperparameters settings as Co-DQL. The network architecture is identical to Co-DQL, except a mean action and sharing joint state are not fed as an addition input to the Q-network.

IDQL: The parameter setting of this method is almost the same as that of IQL. The main difference is that double estimators are used when calculating the target value.

DDPG: This is an off-policy algorithm too. It consists of two parts: 1) actor and 2) critic. Each agent is trained with the DDPG algorithm and we share the critic among all agents in each experiment and all of the actors are kept separate. It uses Adam optimizer with a learning rate of 0.001 and 0.0001 for critics and actors, respectively. The settings of other parameters are the same as those of Co-DQL.

It is noteworthy that all the hyperparameter settings of all algorithms may affect the performance of the algorithm to a certain extent.

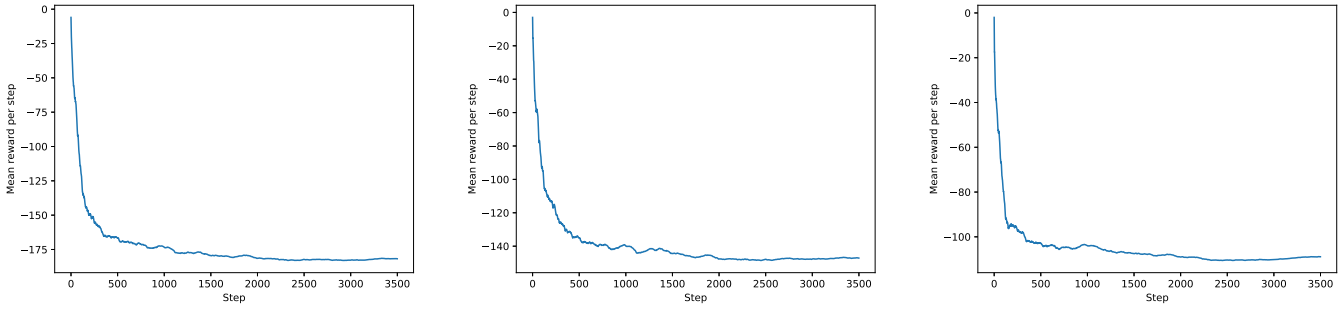

Fig. 5. Illustration of mean reward change curve of signal agents using random strategy in various traffic flows scenarios. (a) Global random traffic flow scenario, (b) double-ring traffic flow scenario, and (c) four-ring traffic flow scenario. At the beginning, there are fewer vehicles running in the simulator. As vehicles are added to the simulator at each time step, there are more and more vehicles in the road network, and the mean reward of signal agents is getting smaller. As the vehicle arriving at the destination will be removed from the simulator, the level of congestion will reach a stable range. We intercept a certain number of simulator states after stabilization as the selectable initial state of the simulator when training and evaluating MARL models.

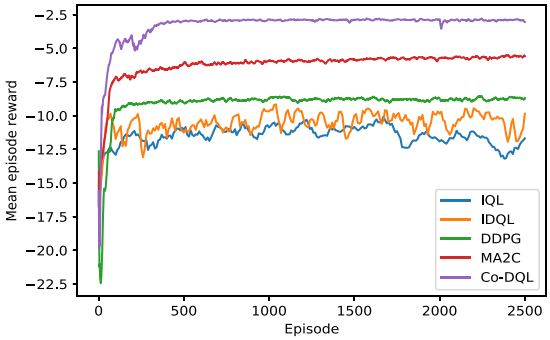

Fig. 6. Reward curve of signal agent during training in the global random traffic flow scenario.

TABLE II
MODEL PERFORMANCE IN GLOBAL RANDOM TRAFFIC FLOW SCENARIO

Method	Average Delay Time [t]	Mean Episode Reward
IQL	148.500($\pm$8.963)	$-$11.602($\pm$0.700)
IDQL	131.854($\pm$7.534)	$-$10.301($\pm$0.589)
DDPG	111.057($\pm$0.606)	$-$8.676($\pm$0.047)
MA2C	71.553($\pm$0.5812)	$-$5.590($\pm$0.045)
Co-DQL	36.981($\pm$0.509)	$-$2.889($\pm$0.040)

t means discrete time step.

B. Experiments in the Simplified TSC Simulator

By training and evaluating the proposed method in different traffic scenarios, we can demonstrate that the proposed method is promising. Next, we will analyze the performance of the algorithms in three scenarios.

1) Global Random Traffic Flow: As shown in Fig. 5(a), under the condition that signal agents adopt a random strategy, the mean reward reaches stable after about 2000 time steps, which means that the traffic flow of the simulator reaches a stable state too. In order to ensure the diversity of training samples and avoid overfitting some traffic flow states as far as possible, we record ten discrete simulator states (i.e., vehicle position, driving status, and signal status) after 2000 time steps as random seeds and it will be used to train and evaluate these methods. In the global random traffic flow, we set the number of new vehicles added at each time step to 5, which corresponds to a high level of traffic congestion.

Result Analysis: We run 2500 episodes for training all five models, and regularly save the trained models. The mean reward curve of signal agents is shown in Fig. 6. It can be seen from the figure that IQL suffers the lowest training performance. Although IDQL is just slightly better than IQL, the results tend to indicate that overestimation of the action-value function will damage the performance of signal control and that using double estimators can improve the performance to a certain extent. Interestingly, the performance of DDPG is better than that of IDQL, it may be due to the advantages of the actor-critic structure. Although MA2C and Co-DQL both have more robust learning ability, Co-DQL greatly outperforms all the other methods. Co-DQL uses mean-field approximation to directly model the strategies of other agents, thus it can learn good cooperative strategies and maximize the total reward of the road network.

For each algorithm, the best model obtained in the training process is used to test in this scenario. We evaluate all of them over 100 episodes. Table II shows the results of evaluation. The average delay time is calculated from the total delay time of vehicles in the road network during an episode. The standard deviation is given in parentheses after the mean value. Co-DQL greatly reduces the average delay time compared with the other methods. The test results are basically consistent with the trained model performance, which shows the validity of our trained model.

2) Double-Ring Traffic Flow: Fig. 5(b) shows the mean reward curve of agents using random strategies in the double-ring traffic flow scene. Similarly, ten simulator states are selected as seeds. In this scenario, we set the number of new vehicles added to the network at each time step to 4, which corresponds to a medium level of traffic congestion. The other parameters of the simulator are the same as those in Section V-B1.

Result Analysis: Similarly, we train all the models in this scenario and save the model with the best training performance. The mean reward curve is shown in Fig. 7. As expected, the training performance of the Co-DQL method still outperforms all the other methods. In addition, mainly

TABLE III
MODEL PERFORMANCE IN DOUBLE-RING TRAFFIC FLOW SCENARIO

Method	Average Delay Time [t]	Mean Episode Reward
IQL	$89.838(\pm 5.645)$	$-5.615(\pm 0.353)$
IDQL	$83.921(\pm 2.273)$	$-5.245(\pm 0.142)$
DDPG	$86.581(\pm 1.182)$	$-5.411(\pm 0.074)$
MA2C	$58.857(\pm 0.779)$	$-3.679(\pm 0.049)$
Co-DQL	$26.046(\pm 0.751)$	$-1.628(\pm 0.047)$

t means discrete time step.

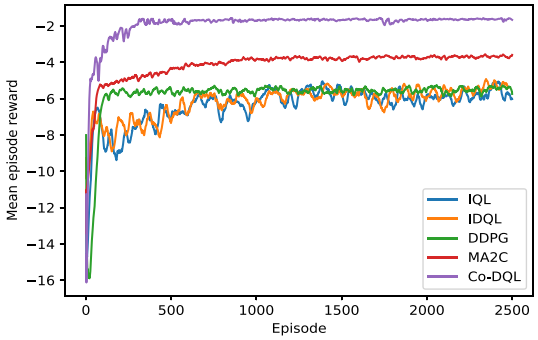

Fig. 7. Reward curve of signal agent during training in the double-ring traffic flow scenario.

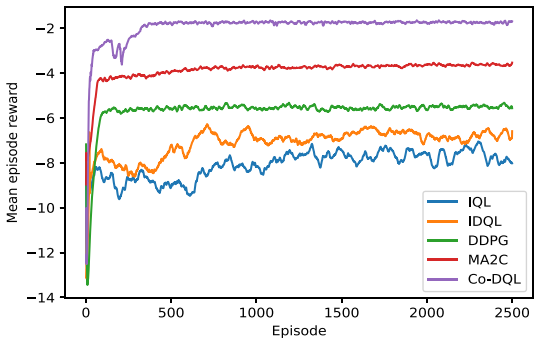

Fig. 8. Reward curve of signal agent during training in the four-ring traffic flow scenario.

TABLE IV
MODEL PERFORMANCE IN FOUR-RING TRAFFIC FLOW SCENARIO

Method	Average Delay Time [t]	Mean Episode Reward
IQL	$168.526(\pm 2.673)$	$-7.900(\pm 0.125)$
IDQL	$143.986(\pm 3.761)$	$-6.749(\pm 0.176)$
DDPG	$116.823(\pm 1.610)$	$-5.476(\pm 0.075)$
MA2C	$77.633(\pm 0.660)$	$-3.639(\pm 0.031)$
Co-DQL	$37.174(\pm 0.937)$	$-1.743(\pm 0.044)$

t means discrete time step.

due to the information transfer among agents, MA2C can obtain better training results in contrast to the independent agent methods, that is, IQL and IDQL. However, although the convergence rates of DDPG, IQL, and IDQL are different, the final training results are basically similar. This may be because the problem of double-ring traffic flow is relatively simple, so these three methods can achieve relatively consistent results. In this scenario, the evaluation results are shown in Table III. Co-DQL can obtain shorter average delay time and smaller standard deviations than other methods.

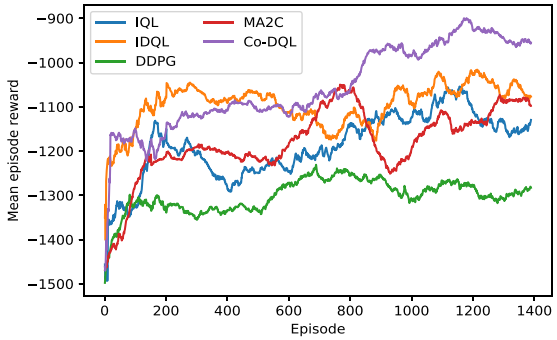

Fig. 9. Reward curve of signal agent during training on the real road network with asymmetric geometry.

3) Four-Ring Traffic Flow: Select seeds for the four-ring traffic flow according to the curve of Fig. 5(c). In order to simulate traffic conditions with a low level of traffic congestion, we set the number of new vehicles added to the road network at each time step to 3. The other parameters of the simulator are set in the same way as other scenarios.

Result Analysis: The training curve in this scenario is shown in Fig. 8, and the test results are shown in Table IV. In this scenario, the training performance of IDQL is significantly better than that of IQL without double estimators. The learning process of Co-DQL and MA2C is relatively stable and the standard deviation in the evaluation process is smaller than that of IQL, IDQL, and DDPG, this may be due to that they share information among agents. But ultimately, Co-DQL achieves the shortest average delay time by means of mean-field approximation for opponent modeling and local information sharing.

C. Experiment in the More Realistic TSC Simulator

Experiment Settings: Experiment with the simulator setup described in Section IV-B2. Regarding MDP setting, the regularization rate β in reward is set to 0.2veh/s, and the regularization factors of *wave*, *wait*, and reward are 5 veh, 100 s, and 2000 veh. Here, we train all MARL models around 1400 episodes given episode horizon $T = 720$ steps, then evaluate the trained models over ten episodes.

Result Analysis: The mean episode reward curve during the training in this scenario is shown in Fig. 9. In this challenging scenario, DDPG suffers from the worst training performance, which may be due to the time-varying traffic flow leading to a large variance of critics, so it cannot effectively guide the learning of actors. Surprisingly, although the training performance of MA2C is much better than that of DDPG, it has no obvious advantage over IQL and IDQL. This may be due to MA2C is more sensitive to the number of agents, and the setting of many hyperparameters involved is also a big challenge. As expected, Co-DQL achieves the best training performance.

In this more realistic simulator, we have the opportunity to consider more traffic metrics than in the simplified one. Table V shows the evaluation results using ten different random seeds, in which avg. vehicle speed is calculated by dividing the total distance traveled by the driving time, avg.

TABLE V
MODEL PERFORMANCE IN REAL ROAD NETWORK WITH ASYMMETRIC GEOMETRY

Metrics	IQL	IDQL	DDPG	MA2C	Co-DQL
Mean Episode Reward	$-1160.52(\pm 190.62)$	$-1076.34(\pm 193.53)$	$-1296.68(\pm 140.87)$	$-1108.52(\pm 83.41)$	$-930.38(\pm 87.45)$
Avg. Vehicle Speed [m/s]	$4.33(\pm 0.49)$	$4.53(\pm 0.45)$	$3.81(\pm 0.35)$	$4.65(\pm 0.23)$	$5.35(\pm 0.26)$
Avg. Intersection Delay [s/veh]	$28.52(\pm 5.55)$	$27.17(\pm 5.42)$	$33.01(\pm 4.50)$	$27.98(\pm 2.57)$	$20.31(\pm 2.55)$
Avg. Queue Length [veh]	$10.03(\pm 2.05)$	$10.01(\pm 2.12)$	$12.53(\pm 1.87)$	$9.80(\pm 1.21)$	$7.51(\pm 1.29)$
Trip Delay[s]	$278.38(\pm 35.35)$	$254.20(\pm 46.04)$	$311.34(\pm 30.01)$	$253.23(\pm 14.01)$	$177.73(\pm 16.70)$
Trip Arrived Rate	$0.74(\pm 0.08)$	$0.80(\pm 0.07)$	$0.57(\pm 0.05)$	$0.79(\pm 0.03)$	$0.91(\pm 0.03)$

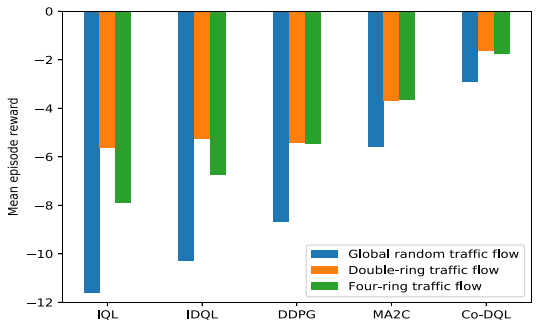

Fig. 10. Mean episode reward comparison for testing the corresponding model in different traffic flow scenarios of simplified TSC.

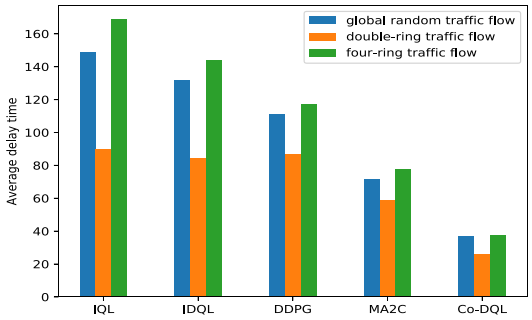

Fig. 11. Mean delay time comparison for testing the corresponding model in different traffic flow scenarios of simplified TSC.

intersection delay is calculated by dividing the total delay time of each intersection by the total number of vehicles at the intersection, and avg. queue length is calculated by the queue length of each time period, and trip delay refers to the total delay time of vehicles in the driving process, and trip arrived rate is calculated by dividing the number of vehicles that have arrived at the destination before the end of the simulation by the total number of vehicles. The comparison results in terms of all measures are relatively consistent.

According to the results, overestimation makes a difference in the performance between IQL and IDQL, and the use of double estimators in IDQL always has a slight advantage over IQL according to most of the measurements. Compared with IQL, IDQL, and DDPG, Co-DQL and MA2C show more robust test performance (less standard deviation), which shows that information sharing among agents brings benefits to cooperation among agents, and Co-DQL achieves the best average performance with respect to multiple measures, which shows the advantage of the mean field approximation in agent behavior modeling.

D. Discussion

First, we discuss the performance of different algorithms in three traffic flow scenarios with the simplified MDP setting. As seen from Fig. 10 (blue bar), all methods have a smaller mean episode reward in the global random traffic flow scenario than in the other scenarios, which is due to the highest level of traffic congestion and the largest traffic volume in this scenario. According to Fig. 11 (green bar), although the mean episode reward level of each evaluation model in the four-ring traffic flow scenario is moderate, the number of vehicles in this scenario is small, which may lead to greater average vehicle delay. Although the traffic volume of double-ring traffic flow scenario is larger than that of four-ring traffic flow scenario, the evaluation results in the former scenario (orange bar) are even slightly better than the latter (green bar), regardless of the mean episode reward of the agent or the average waiting time of the vehicle. The analysis shows that the double-ring traffic flow scenario just needs the cooperation between two groups of agents, namely, the cooperation of signal agents in the inner and outer loop, while the four-ring traffic flow scenario needs the collaboration among four groups, so the cooperation task of signal agents in the latter may be more complex.

Experimental results on multiple scenarios show that the performance of the algorithm with a double estimator is always better than that without double estimator. Compared with the simplified situation, in the more realistic case, MA2C does not achieve the desired performance. Co-DQL can still obtain more training reward and better evaluation performance than the state-of-the-art decentralized MARL algorithms. In addition, we also conducted an experiment on a 7×7 grid road network simulator, the setting and results about the experiment are shown in the supplementary material. One can notice that Co-DQL can achieve the best results.

In the society of RL, a hot topic is how to use it in reality. Because the uncertainty brought by the exploration behavior of the RL model in the training process is a potential safety hazard for the application of TSC in practice, the training stage of our model is completed in a TSC simulator in a similar way as most RL models [15], [16], [23], and the model deployed in reality is generally the model trained in the simulator. Although there is a gap between the simulator and the real environment, simulation to reality (sim2real) [46], as a branch of RL, has been widely studied in order to bridge the gap.

VI. CONCLUSION

When to design an MARL algorithm, a critical challenge is how to make the agents efficiently cooperate, and one of

the breaches of realizing is properly estimating the Q values and sharing local information among agents. Along this line of thought, this article developed Co-DQL, which takes advantage of some important ideas studied in the literature. In more detail, Co-DQL employs an IDQL method based on double estimators and the UCB exploration, which can eliminate overestimation of traditional IQL while ensuring exploration. It adopts mean-field approximation to model the interaction among agents so that agents can learn a better cooperative strategy. In addition, we presented a reward allocation mechanism and a local state sharing method. Based on the characteristics of TSC, we gave the details of the algorithmic elements. To validate the performance of the proposed algorithm, we tested Co-DQL on various traffic flow scenarios of TSC simulators. Compared with several state-of-the-art MARL algorithms (i.e., IQL, IDQL, DDPG, and MA2C), Co-DQL can achieve promising results.

In the future, we hope to further test Co-DQL on the real city road network, and we will consider other approaches on large-scale MARL such as hierarchical architecture [41], [47]. In addition, note that the local optimization of an agent's reward (throughput) may reduce the neighboring agents' rewards in a nonlinear way. Such a nonlinearity is typical in traffic flow. Using the linear weighted function with a constant α may not fully capture the nonlinear throughput relationship between neighboring intersections. Also, each agent's reward will appear multiple times, depending on the number of connected neighboring intersections. For instance, an intersection with five legs will receive more weights than a three-leg intersection that may cause a biased optimal solution. Hence, it may be interesting to further study the reward allocation mechanism.

So far, a great number of methods have been proposed for TSC, such as max pressure [48] and cell transmission model [49]. It may be interesting to comprehensively compare these methods. Furthermore, parameters heavily affect the performance of an algorithm, it is interesting to study how to automatically adjust them so as to achieve a promising quality. Finally, it may be interesting to study our method on the other MDP settings for the TSC problem.

REFERENCES

[1] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Comput. Surveys*, vol. 50, no. 3, p. 34, 2017.

[2] Q. Wu and J. Guo, "Optimal bidding strategies in electricity markets using reinforcement learning," *Elect. Power Compon. Syst.*, vol. 32, no. 2, pp. 175–192, 2004.

[3] B. Yin, M. Dridi, and A. E. Moudni, "Traffic network microsimulation model and control algorithm based on approximate dynamic programming," *IET Intell. Transp. Syst.*, vol. 10, no. 3, pp. 186–196, 2016.

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

[5] H. Ceylan and M. G. Bell, "Traffic signal timing optimisation based on genetic algorithm approach, including drivers routing," *Transport. Res. B Methodol.*, vol. 38, no. 4, pp. 329–342, 2004.

[6] J. García-Nieto, E. Alba, and A. C. Olivera, "Swarm intelligence for traffic light scheduling: Application to real urban areas," *Eng. Appl. Artif. Intell.*, vol. 25, no. 2, pp. 274–283, 2012.

[7] J. Qiao, N. Yang, and J. Gao, "Two-stage fuzzy logic controller for signalized intersection," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 41, no. 1, pp. 178–184, Jan. 2011.

[8] 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, Sep. 2006.

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[10] M. Wiering, J. van Veenen, J. Vreeken, and A. Koopman, "Intelligent traffic light control," Inst. Inf. Comput. Sci., Utrecht University, Utrecht, The Netherlands, Rep. UU-CS-TR-2004-029, 2004.

[11] L. A. 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, Jun. 2011.

[12] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[13] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[14] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015. [Online]. Available: arXiv:1509.02971.

[15] 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. Disc. Data Min.*, 2018, pp. 2496–2505.

[16] N. Casas, "Deep deterministic policy gradient for urban traffic light control," 2017. [Online]. Available: arXiv:1703.09035.

[17] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *Proc. 15th Nat./10th Conf. Artif. Intell./Innovative Appl. Artif. Intell. (AAAI/IAAI)*, Madison, WI, USA, 1998, pp. 746–752.

[18] S. Shamshirband, "A distributed approach for coordination between traffic lights based on game theory," *Int. Arab J. Inf. Technol.*, vol. 9, no. 2, pp. 148–153, 2012.

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

[20] M. Abdoos, N. Mozayani, and A. L. Bazzan, "Traffic light control in non-stationary environments based on multi agent Q-learning," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, 2011, pp. 1580–1585.

[21] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 330–337.

[22] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis, "Multiagent reinforcement learning for urban traffic control using coordination graphs," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, 2008, pp. 656–671.

[23] 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.

[24] H. V. Hasselt, "Double Q-learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2010, pp. 2613–2621.

[25] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, nos. 2–3, pp. 235–256, 2002.

[26] H. E. Stanley, *Phase Transitions and Critical Phenomena*. Oxford, U.K.: Clarendon Press, 1971.

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

[28] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 222–227.

[29] J. E. Smith and R. L. Winkler, "The optimizer's curse: Skepticism and postdecision surprise in decision analysis," *Manag. Sci.*, vol. 52, no. 3, pp. 311–322, 2006.

[30] L. S. Shapley, "Stochastic games," *Proc. Nat. Acad. Sci. USA*, vol. 39, no. 10, pp. 1095–1100, 1953.

[31] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. Mach. Learn.*, 1994, pp. 157–163.

[32] K. Prabuchandran, H. K. AN, and S. Bhatnagar, "Multi-agent reinforcement learning for traffic signal control," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, 2014, pp. 2529–2534.

[33] A. K. Agogino and K. Tumer, "Analyzing and visualizing multiagent rewards in dynamic and stochastic domains," *Auton. Agents Multiagent Syst.*, vol. 17, no. 2, pp. 320–338, 2008.

[34] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.
[35] M. Li *et al.*, "Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning," in *Proc. ACM World Wide Web Conf.*, 2019, pp. 983–994.
[36] T. Jaakkola, M. I. Jordan, and S. P. Singh, "Convergence of stochastic iterative dynamic programming algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 703–710.
[37] C. Szepesvári and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural Comput.*, vol. 11, no. 8, pp. 2017–2060, 1999.
[38] 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," *Transport. Res. C Emerg. Technol.*, vol. 85, pp. 732–752, Dec. 2017.
[39] 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, Sep. 2013.
[40] J. Jin and X. Ma, "Adaptive group-based signal control by reinforcement learning," *Transport. Res. Proc.*, vol. 10, pp. 207–216, Sep. 2015.
[41] 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.
[42] T. Chu, S. Qu, and J. Wang, "Large-scale traffic grid signal control with regional reinforcement learning," in *Proc. IEEE Amer. Control Conf. (ACC)*, 2016, pp. 815–820.
[43] L. Codeca and J. Härri, "Monaco sumo traffic (most) scenario: A 3D mobility scenario for cooperative its," in *Proc. User Conf. Simul. Auton. Intermodal Transp. Syst. (SUMO)*, 2018, pp. 43–55.
[44] K. T. K. Teo, K. B. Yeo, Y. K. Chin, H. S. E. Chuo, and M. K. Tan, "Agent-based traffic flow optimization at multiple signalized intersections," in *Proc. IEEE 8th Asia Model. Symp.*, 2014, pp. 21–26.
[45] G. Brockman *et al.*, "OpenAI gym," 2016. [Online]. Available: arXiv:1606.01540.
[46] F. Sadeghi, A. Toshev, E. Jang, and S. Levine, "Sim2real viewpoint invariant visual servoing by recurrent control," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4691–4699.
[47] A. S. Vezhnevets *et al.*, "Feudal networks for hierarchical reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 3540–3549.
[48] P. Varaiya, "Max pressure control of a network of signalized intersections," *Transport. Res. C Emerg. Technol.*, vol. 36, pp. 177–195, Nov. 2013.
[49] S. Timotheou, C. G. Panayiotou, and M. M. Polycarpou, "Distributed traffic signal control using the cell transmission model via the alternating direction method of multipliers," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 919–933, Dec. 2014.

Xiaoqiang Wang received the B.E. degree in automation from Chongqing University, Chongqing, China, in 2018. He is currently pursuing the Ph.D. degree in control science and engineering with Xi'an Jiaotong University, Xi'an, China.

He is with the State Key Laboratory for Manufacturing Systems Engineering, Xi'an, and also with the CETC Key Laboratory of Aerospace Information Applications, Shijiazhuang, China. His current research interests include multiagent reinforcement learning and causal discovery.

Liangjun Ke (Member, IEEE) received the B.Sc. and M.Sc. degrees in mathematics from Wuhan University, Wuhan, China, in 1998 and 2001, respectively, and the Ph.D. degree in systems engineering from Xi'an Jiaotong University, Xi'an, China, in 2008.

He is currently a Full Professor with Xi'an Jiaotong University. His main research interests include optimization theory and applications, especially multiobjective optimization, evolutionary computation, and reinforcement learning.

Zhimin Qiao received the B.E. degree in automation from the North University of China, Taiyuan, China, in 2012, and the M.S. degree in control engineering from Northeastern University, Shenyang, China, in 2015. He is currently pursuing the Ph.D. degree in control science and engineering with Xi'an Jiaotong University, Xi'an, China.

His active research interests include connected and automated vehicles, intelligent transportation systems, and evolutionary computation.

Xinghua Chai received the B.S. degree from Hohai University, Nanjing, China, in 2008, the M.S. degree from Beijing Information Science and Technology University, Beijing, China, in 2013, and the Ph.D. degree from Beihang University, Beijing, in 2017.

He is working as a Senior Engineer with the 54th Research Institute of China Electronics Technology Group Corporation, Shijiazhuang, China. His research directions are machine vision, computer vision, and multiagent systems.