Network-Scale Traffic Signal Control via Multiagent Reinforcement Learning With Deep Spatiotemporal Attentive Network

Hao Huang, Zhiqun Hu, Zhaoming Lu, and Xiangming Wen, *Senior Member, IEEE*

***Abstract*—The continuous development of intelligent traffic control systems has a profound influence on urban traffic planning and traffic management. Indeed, as big data and artificial intelligence continue to evolve, the traffic control strategy based on deep reinforcement learning (RL) has been proven to be a promising method to improve the efficiency of intersections and save people's travel time. However, the existing algorithms ignore the temporal and spatial characteristics of intersections. In this article, we propose a multiagent RL based on the deep spatiotemporal attentive neural network (MARL-DSTAN) to determine the traffic signal timing in a large-scale road network. In this model, the state information captures the spatial dependency of the entire road network by leveraging the graph convolutional network (GCN) and integrates the information based on the importance of intersections via the attention mechanism. Meanwhile, to accumulate more valuable samples and enhance the learning efficiency, the recurrent neural network (RNN) is introduced in the exploration stage to constrain the action search space instead of fully random exploration. MARL-DSTAN decomposes the large-scale area into multiple base environments, and the agents in each base environment use the idea of "centralized training and decentralized execution" to learn to accelerate the algorithm convergence. The simulation results show that our algorithm significantly outperforms the fixed timing scheme and several other state-of-the-art baseline RL algorithms.**

***Index Terms*—Attention, graph convolutional network (GCN), multiagent reinforcement learning (MARL), recurrent neural network (RNN), traffic signal control (TSC).**

I. Introduction

TRAFFIC congestion has been becoming a daunting problem affecting people's daily lives these days. An immense amount of time on the commute can be wasted due to congestion. In addition, traffic congestion can also cause other negative effects, such as air pollution and economic losses [1]. According to the report of the American Transportation Research Institute, the estimation of the total cost of congestion in the freight sector is as high as $74.1 billion annually [2] and automobile exhaust is the main source of greenhouse gas [3]. To alleviate this issue, traffic signal control (TSC) is an effective method to regulate the traffic flow of road networks and improve the traffic efficiency of intersections.

Early TSC policies typically involve online programmed cycles that dynamically adjust signal timing based on real-time traffic situations estimated from cameras and sensors [4], [5]. The successful products, such as SCATS [6] and SCOOT [7], have been widely deployed in hundreds of cities around the world. However, these control programs take inputs from sensors to detect the existence of vehicles passing the intersection, which causes signal control to lag behind changes in traffic conditions. The control system can only passively adapt to the traffic environment, which hardly adapted to ever-changing dynamics.

Fortunately, as a large amount of traffic data can be obtained in real time by sensors deployed in the road network and data analysis technology continues to develop, TSC can be supported by many advanced methods, such as fuzzy logic and reinforcement learning (RL). Particularly, RL is a powerful tool for solving sequential decision problems like TSC, which uses a trial-and-error mechanism to continuously learn through the interaction between the agent and the environment without a model [8]. Abdulhai *et al.* [9] proposed a case study based on Q-learning to control an isolated two-phase signalized intersection. Shoufeng *et al.* [10] used Q-learning to control traffic to minimize the intersection delay. However, the storage and updating of the Q table limit these attempts to adapt only suitable for small flow systems.

Thanks to the success of deep learning, RL has made tremendous progress. Combining deep neural networks with RL to estimate the value function or action-value function can handle higher-dimensional state and action spaces. Currently, many deep RL methods have been proposed [11]–[16]. Li *et al.* [13] proposed a single intersection control method based on the deep Q network (DQN) algorithm and constructed a two-lane intersection model. DQN guarantees the stability of the algorithm through the target network structure

Manuscript received November 19, 2020; revised March 8, 2021; accepted June 1, 2021. This work was supported in part by the National Natural Science Foundation of China under Grant 61901163 and Grant 61801036; in part by the Beijing Nova Program from Beijing Municipal Science and Technology Commission under Grant Z201100006820123; in part by the Fundamental Research Funds for the Central Universities under Grant 2020RC04; and in part by the Beijing Laboratory of Advanced Information Network. This article was recommended by Associate Editor Y. Shi. *(Corresponding author: Zhiqun Hu.)*

Hao Huang, Zhaoming Lu, and Xiangming Wen are with the Beijing Key Laboratory for Network System Architecture and Convergence, Beijing Laboratory of Advanced Information Networks, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: huanghao@bupt.edu.cn; lzy_0372@163.com; xiangmw@bupt.edu.cn).

Zhiqun Hu is with the School of Computing and Information Engineering, Hubei University, Wuhan 430062, China (e-mail: zhiqunhu520@163.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2021.3087228.

Digital Object Identifier 10.1109/TCYB.2021.3087228

and experience replay mechanism. Wang *et al.* [14] proposed a discrete-time traffic state encoding method to define the traffic state and employed a double dueling DQN to automatically learn useful features from the large-scale raw state data. Compared with DQN, the network structure in [14] combines the features of double Q learning and dueling Q learning, which can obtain a more accurate action-value function (Q value). To further improve the traffic efficiency, Choe *et al.* [15] modified the fully connected network layer of DQN by the LSTM layer to capture the continuous vehicle motion for real-time traffic prediction. Compared with the DQN algorithm, Mousavi *et al.* [16] showed that the policy-based deep RL technology has better stability in traffic control by having smooth convergence and a stable trend after convergence. In [17], the deep deterministic policy gradient (DDPG) was applied to learn the time length of the current phase, where the range is limited to the predefined minimum and maximum phase duration. The works in [17] change the limitation of the DQN algorithm that can only handle discrete action space, such as selecting the next phase from a phase set or deciding whether to change the current phase, which can further improve and stabilize the performance.

Although the neural network has improved the scalability of RL, training a centralized RL agent is still infeasible for large-scale TSC due to the dimensional disaster [18]. As the number of intersections increases, the joint action space will show exponential growth. An alternative way is multiagent RL (MARL) [19] where each traffic intersection is regarded as an agent. In [20], an agent with an independent double dueling DQN model supported with prioritized experience replay is assigned to each intersection, where each agent only considers its local information and ignores the existence of other agents. However, an adaptation of the agent will cause changes in the other agents' behaviors, making it difficult for agents to learn optimal strategies with convergence guarantee [21]. El-Tantawy *et al.* [22] adopted a decentralized mode to design a heuristic neighborhood communication for tabular Q-learning agents, where each message contained the estimated neighbor policies, but does not consider the state information of the entire road network. In [23], when updating the Q-value of each intersection, Kim and Jeong took into account the influence of the Q-value of adjacent intersections. By sharing the Q-value, the collaboration between intersections can be realized and the stability of the algorithm can be enhanced. The idea of Tan *et al.* [24] is similar to that of [23], that is, to find the global optimal Q-value, except that the hierarchical architecture is used when estimating the global optimal Q-value. First, the entire road network is divided into multiple base environments, and each regional agent learns its own RL policy and value function in the base region. Then, the centralized global agent hierarchically aggregates RL achievements from different regional agents and forms the final Q-function over the entire large-scale traffic grid.

However, the above works focus on the shared parameters of the RL framework without considering the temporal and spatial characteristics of the road network, which would reduce the collaborative efficiency of intersections. Zhao *et al.* [25] combined the graph convolutional network (GCN) and the gated recurrent unit (GRU) to dynamically capture the temporal and spatial correlation of the road network, which achieved good results in traffic prediction. Further, Zhou *et al.* [26] proposed reinforced spatial–temporal attention graph neural networks (RSTAG) architecture, which captures dynamic spatial correlation through diffusion network graphs, while temporal dependence is represented by a sequence-to-sequence model with an attention mechanism. In terms of TSC, Devailly *et al.* [27] introduced inductive graph RL based on GCN that adapts to the structure of any road network, to learn detailed representations of traffic controllers and their surroundings. However, they did not capture the time features of the road network. Wang *et al.* [28] controlled each intersection in a distributed way through deep Q learning. To be specific, they constructed the traffic light adjacency graph based on the spatial structure among traffic lights, and then historical traffic records will be integrated with current traffic status via a recurrent neural network (RNN) structure. Colight [29], which considers the temporal and spatial information of neighboring intersections through graph attention network, is an RL-based algorithm for TSC in discrete action space. For Colight, the agent chooses the action from its predefined phase set.

Just like Colight, the action space of the above works [18], [20], [22]–[24], [27], [28] is usually from the perspective of the phase. Algorithms based on Q learning or DQN are only suitable for discrete action space, not for continuous action space, such as using cycle as the signal update frequency to adaptively adjust the green split of each phase. Therefore, it is easy to cause a decrease in control accuracy. In addition, the existing algorithms lack effective restrictions on actions during the exploration stage. Although the works in [28] replace the fully connected layers of the neural network into RNN layers to extract temporal features, it cannot avoid the blindness of random exploration in the early stage of RL. A series of bad trials may cause serious traffic congestion, thereby destroying the entire transportation system. As a result, it is difficult to accumulate valuable sample data in a short period of time. Finally, there is no targeted learning of the spatial position relationship and state information of the intersections, and there is a lack of effective interaction of the state of the entire road network.

In this article, we propose an MARL model based on deep spatiotemporal attentive neural network (MARL-DSTAN) for TSC to solve the limitations of the existing methods. The proposed MARL-DSTAN algorithm conducts a targeted exploration based on RNN, which enhances the efficiency of the agent. Meanwhile, spatial correlations of the road network are modeled through the GCN with an attention mechanism that achieves cooperation among intersections and learns the dynamics of the influence of intersections.

The main contributions are listed as follows.

1) We propose an MARL-DSTAN algorithm to achieve centralized training and decentralized execution by introducing the spatial and temporal information of the traffic network, which shows superior performance improvement in traffic efficiency and enhances the stability against methods without this mechanism in Section IV.

2) The MARL-DSTAN algorithm adopts continuous action space and supports signal timing with the cycle as the update frequency. We can optimize the cycle length and green split, which brings flexibility for the TSC.
3) The spatial dependency module of the proposed MARL-DSTAN algorithm aggregates the spatial state information of the overall road network by using GCN and attention mechanism, which can better coordinate the cooperation among multiagents.
4) The temporal dependency module helps MARL-DSTAN predict the short-term traffic flow from the RNN to reduce unnecessary exploration and improve learning efficiency.

The remainder of this article is organized as follows. Section II presents the preliminaries of our model. The architecture and methodology of MARL-DSTAN are introduced in Section III. In Section IV, the effectiveness and performance of the proposed algorithm are verified by numerical results. Finally, Section V concludes this article.

II. Preliminaries

A. Reinforcement Learning

RL defines the framework that the agent makes its decisions as a function of a state signal from the environment to maximize a numerical reward signal, where the RL is a Markov decision process (MDP) if the state signal has the Markov property [8]. In a fully observable MDP, the learner (i.e., the agent) observes the state s of the environment at each time t, and executes an action a according to the policy π, which is a policy mapping sequences to actions (or distributions over actions). Then, the environment transfers to the state s' according to the state transition probability $p(\cdot|s, a)$, and returns an immediate step reward r to the agent. The process is repeated until it selects the action that maximizes the future reward. The RL algorithms are mainly divided into value-based algorithms and actor–critic architecture-based algorithms. For value-based algorithms, DQN [11], Q-learning [30], and dueling DQN [31] are usually applied to handle discrete and low-dimensional action spaces. The actor–critic architecture-based algorithms, including DDPG [32], DPPO [33], and A3C [34], can learn policies in high-dimensional, continuous action spaces. Among the above algorithms, for example, DQN and DDPG are both off-policy algorithms, because the learning is from the data of the target policy. More important, the above two types of algorithms are inseparable from the calculation of the Q function, which estimates the quality of the policy, denoted by $Q(s, a)$. For MDPs, $Q(s, a)$ can be defined as

$$Q(s,a) = \mathbb{E}\left[\sum_{k=0}^{+\infty} \gamma^k r_{t+k} | s_t = s, a_t = a\right] \tag{1}$$

where $\gamma \in [0, 1]$ is the discount factor, indicating the importance of future returns relative to current returns. The Q function represents the expected cumulative discounted reward in state s and action a. Basically, the Q value for a given state and action can be updated using the Bellman equation as

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma Q(s', a') - Q(s,a)\right] \tag{2}$$

where α is the learning rate, s' represents the state of the next step, and a' represents the next action.

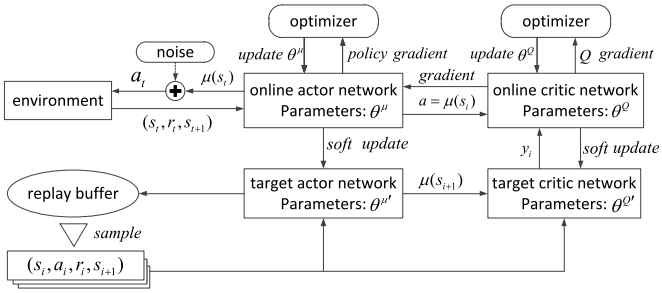


Fig. 1. Architecture of the DDPG algorithm.

Unlike the DQN algorithm, DDPG has two main networks: 1) the actor network and 2) the critic network. The actor network is used to generate a deterministic policy to choose actions, and the critic network is used to simulate the real Q-table using neural networks without the curse of dimensionality. Besides, both the actor network and the critic network contain two subnets: 1) the online network and 2) the target network, one for the target network to be eventually learned and improved, and the other for calculating the TD error when estimating the value function. The subnets have the same structure. The brief structure of DDPG is shown in Fig. 1. All parameters in a specific network are denoted as θ. When the deterministic policy μ is generated from a randomly initialized stochastic policy π, with the approximative Q-table parameterized by θ^Q, the critic network loss can be defined according to (2), as shown below, which can be expressed as

$$L\left(\theta^Q\right) = \mathbb{E}\left[\left(r + \gamma Q'(s', a') - Q(s,a)\right)^2\right]. \tag{3}$$

The actor network updates the policy with the aid of the critic network, where the policy's updating gradient can be written as follows:

$$\nabla_{\theta^\mu} J \approx \mathbb{E}[\nabla_{\theta^\mu} \mu(s) \nabla_a Q(s,a)] \tag{4}$$

where θ^μ can be considered as the variables of the online actor network. DDPG uses a soft update method to update the target network μ' and Q'. In addition, the exploration degree of the action is increased by adding noise, so the action can be expressed as $a_t = \mu_t(s_t|\theta^\mu) + \mathcal{N}_t$.

In addition, in some cooperation or competition scenarios, there are multiple agents interacting with a common environment, called MARL. The most naive way to solve the MARL problem is to treat each agent independently so that the rest of the agents are considered part of the environment, such as IDQN [20]. However, since the actions of each agent will affect the environment, the environment becomes very unstable, which will make the algorithm difficult to converge in the end. One of the effective approaches to deal with this issue is using centralized training, decentralized execution architecture in multiagent DDPG (MADDPG) [35]. By training a fully observable critic network, the states and actions of all

agents can be considered, so even if the strategies of other agents change, the environment is relatively stable. Therefore, in the MADDPG algorithm, the Q function expression can be updated as

$$Q(s, a) \leftarrow Q(s, a_1, a_2, \ldots, a_N) \tag{5}$$

where $s = [o_1, o_2, \ldots, o_N]$, o_i represents the observation of the ith agent, and N is the number of agents. Note that MADDPG requires all the local observations in the critic network, it faces the curse of dimensionality as the number of agents increases.

B. Graph Convolutional Network

The traditional convolutional neural network (CNN) [36] uses the translation invariance of the data to extract information through the convolution kernel, and performs well in processing Euclidean spatial data, such as images and regular grids. However, CNN has limitations in dealing with interpersonal networks and transportation networks with complex topologies. Therefore, it is basically impossible to characterize spatial dependence.

Fortunately, the GCN [37] expressed as $G = (V, E)$ can well extract the topological structure of non-Euclidean spatial data, where $V = \{v_1, v_2, \ldots, v_N\}$ is a set of nodes, E is a set of edges and N is the number of nodes in the network. Each node v_i has its own feature vector x_i, and the feature information of all nodes constitutes a feature matrix $X \in R^{N\times P}$, where P represents the number of node attribute features. The connection relationship between nodes is given by the adjacency matrix $A \in R^{N\times N}$. Specifically, if the nodes v_i and v_j have a direct connection [i.e., there is an edge $(v_i, v_j) \in E$], then $A_{i,j} = 1$. Otherwise, $A_{i,j} = 0$. Based on the adjacency matrix, other nodes adjacent to node i can form its neighborhood set, denoted as $\mathcal{T}(v_i)$.

The GCN extends the convolution operation from traditional data to graph data. The core idea of GCN is to learn function mapping. Through this mapping, the node v_i in the graph can aggregate its own features x_i and its neighbor features $x_j (j \in \mathcal{T}(v_i))$ to generate a new representation of node v_i. Given an adjacency matrix A and the feature matrix X, the GCN model can be expressed as [37]

$$f(X, A) = \sigma\left(D^{-1}\widetilde{A}XW^G\right) \tag{6}$$

where $\widetilde{A} = A + I_N$ is the matrix with added self-connections, I_N is the identity matrix, and D is the degree matrix of $\widetilde{A}$. Through this transformation, it is possible to prevent the node from losing its feature information and realize the normalization of the feature representation. Besides, $W^G \in R^{P\times T}$ represents the weight matrix from input to output, and T is the number of output units, and $\sigma(\cdot)$ represents the sigmoid function for a nonlinear model. GCN can effectively extract graph structure data information, which provides a new idea for state fusion in MARL.

C. Attention

In real life, we often have different levels of attention to different information in the environment. And the importance of different information is also changing over time. To describe the dynamics of the influences, the attention mechanism is introduced into the neural network. To learn the importance of information from node v_j in determining the policy for node v_i, we define the attention coefficient α_{ij} with the following operations [38]:

$$e_{ij} = x_i \cdot W^Q \cdot \left(x_j \cdot W^K\right)^{\mathrm{T}} \tag{7}$$

$$\alpha_{ij} = \exp\left(e_{ij}\right) / \sum_{k=1}^{N} \exp(e_{ik}) \tag{8}$$

where $x_i, x_j \in R^{1\times T}$ is the original feature vector of node i, j in the sequence, and W^Q and W^K $(\in R^{T\times U})$ are feature transformation matrices for nodes v_i and v_j, respectively. N is the total number of nodes that node i needs to pay attention to. $(\cdot)^{\mathrm{T}}$ represents the transpose operation of the matrix. The attention scores of different nodes on node i indicate their importance to each other, which can be calculated by (7). Here, node j is taken as an example. The higher the score, the greater the degree of association between nodes. By normalizing the attention score through the softmax function, we can obtain the attention coefficient between nodes, as shown in (8).

To model the overall influence of neighborhoods of the node v_i, the feature vector of several nodes is combined with their respective importance, which can be given by

$$z_j = x_j \cdot W^V \tag{9}$$

$$x_i^{\text{new}} = \sum_{k=1}^{N} \alpha_{ik} \cdot z_k \tag{10}$$

where $W^V \in R^{T\times U}$ is a feature transformation matrix. Equation (9) is used to perform feature transformation of nodes, and (10) is used to perform weighting on the influence of different nodes. Through the above formulas, node i can effectively aggregate the information of other nodes according to the degree of importance. Finally, we denote the new feature representation of node i as x_i^{new}.

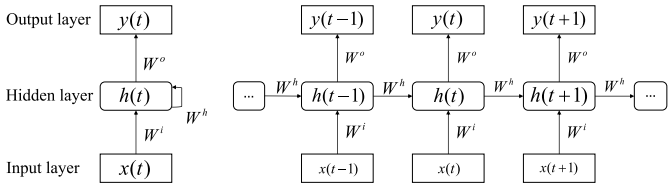


Fig. 2. Basic structure of RNN.

D. Recurrent Neural Network

GCN and attention mechanism can extract spatial features of data, while the RNN [39] can effectively process time-series data. The biggest advantage of RNN is that it has dynamic memory. The output h at each time point is used as the input of the next time point, which can make full use of the feature information of the previous time. The network structure of RNN is shown in Fig. 2. Let $x_t = (x_1, x_2, \ldots, x_T)$ be the feature sequence at time point t. The hidden layer sequence h

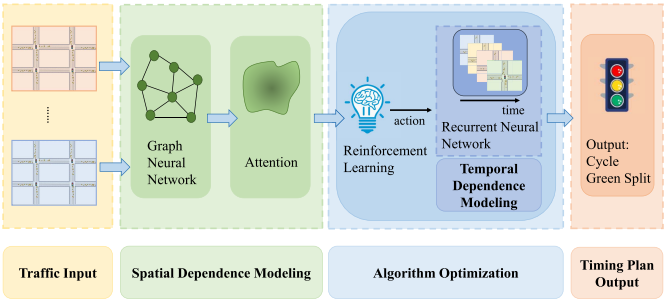


Fig. 3. Architecture of our proposed MARL-DSTAN.

and the final output y can be defined as follows:

$$h_t = f\left(W^i \cdot x_t + W^h \cdot h_{t-1}\right) \tag{11}$$

$$y_t = f\left(W^o \cdot h_t\right) \tag{12}$$

where W^i is the weight matrix from the input layer to the hidden layer, W^h is the weight matrix from the hidden layer to the hidden layer, and W^o is the weight matrix from the hidden layer to the output layer.

III. MARL-DSTAN: Architecture and Methodology

In this section, we introduce the specific details of our proposed MARL-DSTAN model.

A. Overview of MARL-DSTAN

The purpose of the proposed MARL-DSTAN model is to adaptively adjust TSC based on real-time traffic conditions, to reduce traffic congestion and improve the efficiency of intersections. Based on the hierarchical idea, the large-scale road network is divided into multiple regions called base environments, where each base environment is regarded as a traffic control unit. As shown in Fig. 3, our MARL-DSTAN model consists of two main components: 1) spatial dependence module and 2) algorithm optimization module.

For the spatial dependence module, we first introduce the GCN defined in (6) to aggregate neighboring information of each intersection to obtain a new road network feature matrix. Subsequently, to consider the importance of between intersections, we associate a spatial attention coefficient defined in (8) for each intersection in the road network, and then a new feature vector for the specified intersection can be obtained, which combines the entire road network state information.

As for the algorithm optimization module, it is mainly composed of the algorithm decision part and the temporal dependence part. The algorithm decision part determines the timing strategy by means of RL according to the road network state aggregated by the spatial dependence module. And the temporal dependence part uses (12) to predict the flow of each phase to limit the action space, avoiding the intersection agent blindly accumulating experience in the initial exploration stage.

B. Spatial Dependence Modeling

The urban traffic road network is intricate, and the traffic flows of different roads are closely related and affect each other. For example, the trend of downstream traffic flow is positively correlated with that of upstream traffic flow. The spatial dependence module that combines GCN and attention mechanism can perceive the distribution of traffic flow in the road space, so it can enable the agent to make a more efficient timing strategy.

Considering that the agents in the base environment are closer together and have stronger correlations with each other, we use two spatial modules to aggregate information. One aggregates base environmental information, and the other aggregates global road network information. The adjacency matrices of the base environment and the global network are denoted as $A_{\text{base}} \in R^{N_b \times N_b}$ and $A_{\text{global}} \in R^{N_g \times N_g}$, respectively, where N_b represents the number of agents in the base environment and N_g represents the number of agents in the global network. In the MARL-DSTAN model, these two matrices are shared by each agent. The input feature matrix $X \in R^{N_g \times P}$ represents the current traffic state of the global network, from which the feature representation $X_{\text{base}} \in R^{N_b \times P}$ of each base environment can also be extracted. After passing through the GCN, we can obtain new feature matrices X^{new} and $X^{\text{new}}_{\text{base}}$ that incorporate spatial information, in which each row still corresponds to the state vector of the corresponding intersection. Through the attention module, we perform targeted fusion of matrix information to obtain the new feature vectors x_i^{base} and x_i^{global} of agent i. They are concatenated as the final spatial state extraction form and used as input to the fully connected layer, as shown in

$$x_i^{\text{final}} = \text{concat}\left(x_i^{\text{base}}, x_i^{\text{global}}\right). \tag{13}$$

C. Temporal Dependence Modeling

The dilemma that arises in RL is the tradeoff between exploration and exploitation. However, the related issue does not been encountered in supervised learning. Motivated by supervised learning, we predict the traffic flow based on RNN to guide the direction of exploration. During the random exploration, the action space of the agent is limited according to the predicted traffic flow, which can reduce the random search space and avoid completely blind exploration. Meanwhile, the efficiency of experience sampling can be improved, which can enhance learning efficiency.

The input of the RNN module is the phase traffic flow of the previous T_n ($T_n = 5$) cycles at each intersection. The output is the phase traffic flow of the current cycle. Combining (11) and (12), the traffic flow f_t at the current cycle t can be given by

$$f_t = \text{RNN}\left(\left[f_{t-1}, f_{t-2}, \ldots, f_{t-T_n}\right]\right). \tag{14}$$

It is worth noting that the accuracy of our prediction module does not need to be too high. This is because the output of the final timing plan is still determined by the algorithm decision module. According to the prediction results, we divide the traffic flow into three levels: 1) low; 2) medium; and

3) high. For different levels, the agent explores in different action spaces.

D. MARL-DSTAN Algorithm

This section describes the implementation details of MARL-DSTAN for TSC. Specifically, we define the state, action, reward, the MARL-DSTAN network structures, and the MARL-DSTAN training update strategy.

1) State: The state is a quantitative representation of the road network environment observed by each intersection agent. Due to the advantage of the deep neural network in learning representation of data, the state space should reflect the characteristics of the road network as completely as possible. In the road environment, the queue length can reflect the congestion situation in space, and the traffic flow can indicate the traffic situation in time. Therefore, we define the observation of intersection i as

$$o_{i,t} \triangleq \{l_1, \ldots, l_n, q_1, \ldots, q_n\} \tag{15}$$

where l represents the queue length, q represents the road flow, and n is the number of entrance lanes at the intersection. In our experiment, an induction loop will be placed on each road at the upstream of the intersection to count the traffic flow, which will be placed 500 m away from the intersection. It should be noted that q represents the statistical flow of the last cycle, and the unit is veh/s. The state of the entire road network includes the observation of all agents, which is given by the following matrix:

$$s_t = \begin{bmatrix} o_{1,t} \\ \vdots \\ o_{N,t} \end{bmatrix}. \tag{16}$$

In this way, state s reflects the entire environmental information. Then, we use the state s to represent the feature matrix X of the road network, which is taken as the input of the GCN.

2) Action: After the intersection agent has observed the state of the environment, it must choose an action (i.e., the timing scheme) to respond to the state. Common DQN-based algorithms usually use the phase as the update frequency to select action [18], [20], [22]–[24], [27], [28]. This discretized action space will bring quantization errors and reduce the utilization of timing. To overcome this weakness, we use the timing cycle as the update frequency. Specifically, the action space of agent i is expressed as follows:

$$a_{i,t} \triangleq \{c, d_1, \ldots, d_m\} \tag{17}$$

where c represents the duration factor that determines the next cycle length. In order to avoid the cycle being too long or too short, we limit the cycle time to the range of $[c_{\min}T, c_{\max}T]$, where T is the length of the reference cycle. $d_1, \ldots, d_m$ are the scale factors of the duration of each phase in the next cycle, and m is the number of phases. Here, we set $c_{\min} = 0.6$, $c_{\max} = 1.0$, and $T = 100$. This kind of action space design has stronger traffic flow predictability and can help vehicles better plan their routes. In addition, it also avoids the frequent decision making of the agent, and can directly complete the algorithm deployment on most signal controllers.

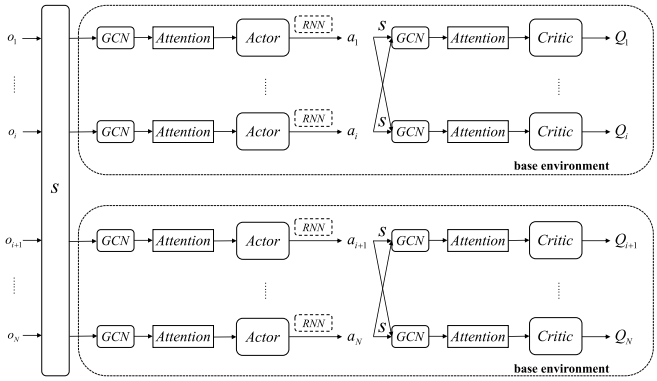


Fig. 4. Flowchart of the MARL-DSTAN algorithm.

3) Reward: The definition of reward guides the direction of agent learning and finally evaluates the quality of the model. For traffic control optimization problems, the goal is to reduce the average delay of vehicles and improve the traffic efficiency of the intersection. To this end, we define the reward as a function of the vehicle waiting time. The shorter the waiting time, the higher the reward. Meanwhile, the fairness is also considered to avoid a small number of vehicles waiting too long. Then, the reward function of the ith agent is defined as follows:

$$r_{i,t} \triangleq \frac{1}{N_T} \sum_{n=1}^{N_T} \eta \left[1 - \left(\frac{k_n}{C} \right)^{\rho} \right] \tag{18}$$

where N_T represents the number of waiting vehicles at intersection i, k_n represents the waiting time of the nth car, C represents tolerable waiting time, and η and ρ are constants. Here, we set $C = 20$, $\eta = 0.15$, and $\rho = 2$ [40].

4) Algorithm Architecture: In this article, we propose the MADDPG algorithm [35] as the basis of TSC in the base environment, which is the extension of the DDPG algorithm on the multiagent condition. The MADDPG algorithm combines the advantage of the policy-based and value approximation method in RL, which trains a Q-value function (critic) while simultaneously updates the policy parameters (actor) in the direction suggested by the Q-value function. Besides, the MADDPG algorithm adopts a centralized training and decentralized execution framework. Each agent maintains a local centralized critic network, which accepts observations and actions of all agents as a joint state–action pair for learning.

Although in the MARL-DSTAN model, the critic network only considers the action information of each agent in the base environment, the state information integrates the global network information through the space module. Therefore, the critic network of each agent fits a global value function of the whole road network, and each agent can adjust the local policy according to the estimated policy of other agents to achieve the global optimal policy. In this way, the environmental nonstationarity caused by the actions of other agents can be reduced.

The basic idea of MADDPG is to use a deterministic strategy μ_i to select the current action for each agent, which can be expressed as $a_{i,t} = \mu_{i,t}(s_{i,t}|\theta_i^{\mu})$, where $s_{i,t} =$

$(o_{1,t}, o_{2,t}, \ldots, o_{N,t})$, θ_i^μ is the actor network parameter that generates the deterministic action. To fit the Q-value function through the critic neural network, we parameterize it to $Q_i(s, a|\theta_i^Q)$, where a is the action set of all agents in the base environment. The main difference between MADDPG and DDPG is whether the actions of other agents are considered in the critic network update. Therefore, based on (4), the policy gradient equation of MADDPG can be rewritten as

$$\nabla_{\theta_i^\mu} J \approx \mathbb{E}\Big[\nabla_{\theta_i^\mu} \mu_i(s_i) \nabla_{a_i} Q_i(s, a_1, \ldots, a_N)\Big]. \tag{19}$$

The algorithm structure is shown in Fig. 4. The GCN and Attention modules represent spatial information fusion of the input state, and the RNN module is used in the exploration stage of RL to avoid blind exploration and generate bad actions.

5) Algorithm Update: We now present the training process of MARL-DSTAN. To ensure algorithm convergency, the model is trained by using experience replay and the target network. Then, each agent includes actor network μ_i, critic network Q_i, and the corresponding target networks μ_i' and Q_i'. Each agent updates its actor parameter θ_i^μ by maximizing the future expected cumulative reward according to the objective function $J(\theta_i^\mu) = \mathbb{E}[Q_i(s, a)]$. The actor parameters are updated by applying the chain rule to the objective function $J(\theta_i^\mu)$. Based on (19), it can be rewritten as

$$\nabla_{\theta_i^\mu} J \approx \mathbb{E}\Big[\nabla_{\theta_i^\mu} \mu_i(s_i) \nabla_{a_i} Q_i\big(s, \mu_1(s_1), \ldots, \mu_{N_b}\big(s_{N_b}\big)\big)\Big]. \tag{20}$$

Accordingly, the critic network is regarded as the supervision signal to evaluate the actor network. By introducing a function approximators with parameter θ_i^Q, each agent updates its critic network by minimizing the loss function $L(\theta_i^Q)$ based on (3), which can be given by

$$L\Big(\theta_i^Q\Big) = \mathbb{E}\Big[(y_i - Q_i(s, a))^2\Big] \tag{21}$$

where

$$\begin{aligned} y_i &= r_i + \gamma Q_i'(s', a') \\ s &= \big\{o_1, o_2, \ldots, o_{N_g}\big\} \\ a &= \big\{a_1, a_2, \ldots, a_{N_b}\big\}. \end{aligned} \tag{22}$$

In addition, to achieve smooth learning and easy convergence, a soft update strategy is used when updating the target network, which can be given by

$$\theta_i^{\mu'} \leftarrow \tau\theta_i^\mu + (1-\tau)\theta_i^{\mu'} \tag{23}$$

$$\theta_i^{Q'} \leftarrow \tau\theta_i^Q + (1-\tau)\theta_i^{Q'}. \tag{24}$$

The pseudocode of the MARL-DSTAN algorithm for TSC is presented in Algorithm 1. The detailed steps of the MARL-DSTAN algorithm are as follows.

1) We treat each intersection as an agent and each agent is in a base environment. The actor network and critic network of each agent are randomly initialized with $\mu_i(s|\theta_i^\mu)$ and $Q_i(s, a|\theta_i^Q)$, respectively. The target networks μ_i' and Q_i' and replay buffer are also initialized.

Algorithm 1 MARL-DSTAN Algorithm Framework for TSC—Pseudocode for Each Agent

Initialization:

Randomly initialize actor and critic neural networks $\mu_i(s|\theta_i^\mu)$ and $Q_i(s, a|\theta_i^Q)$ for each intersection agent.

Initialize corresponding target networks μ_i' and Q_i' with weights $\theta_i^{\mu'} \leftarrow \theta_i^\mu$,$\theta_i^{Q'} \leftarrow \theta_i^Q$.

Initialize replay buffer and a random process $\mathcal{N}$ for action exploration.

Algorithm:

for episode=1 to M **do**
  Initialize observation o_0 of the local intersection
  while $t < T$ and $o_t \neq$ terminal **do**
    $s_{i,t} = \{o_{1,t}, o_{2,t}, ..., o_{N,t}\}$
    $s_{i,t}^{base} = GCN(s_{i,t}^{base})$, $s_{i,t}^{global} = GCN(s_{i,t}^{global})$
    $s_{i,t}^{base} = ATTN(s_{i,t}^{base})$, $s_{i,t}^{global} = ATTN(s_{i,t}^{global})$
    $s_{i,t} = concat(s_{i,t}^{base}, s_{i,t}^{global})$
    Select action $a_{i,t} = \mu_{i,t}(s_{i,t}|\theta_i^\mu) + \mathcal{N}_t$ according to current policy and exploration noise
    Clip action according to the RNN module: $a_{i,t} = clip(a_{i,t})$
    Execute action $a_{i,t}$ and receive reward $r_{i,t}$ and new observation o_{t+1} of the local intersection
    $t = t + 1$
    Store $(s_t, a_{base,t}, r_t, s_{t+1})$ in replay buffer
  end while
  Sample a random minibatch of L samples (s_l, a_l, r_l, s_{l+1}) from replay buffer
  Set $y_l = r_l + \gamma Q_i'(s_{l+1,}a_{1,l+1}, ..., a_{N_b,l+1})$
  Update critic by minimizing the loss: $loss = \frac{1}{L}\sum_l (y_l - Q_i(s_l, a_l))^2$
  Update actor using the sampled policy gradient: $\nabla_{\theta_i^\mu} J \approx \mathbb{E}[\nabla_{\theta_i^\mu} \mu_i(s_i) \nabla_{a_i} Q_i(s, \mu_1(s_1), ..., \mu_{N_b}(s_{N_b}))]$
  Update the target networks for intersection agent i: $\theta_i^{\mu'} \leftarrow \tau\theta_i^\mu + (1-\tau)\theta_i^{\mu'}$, $\theta_i^{Q'} \leftarrow \tau\theta_i^Q + (1-\tau)\theta_i^{Q'}$
end for

2) In the initial random exploration stage, the agent chooses action a_i according to the current policy $\mu_i(s_i) = \mu_i(s_i|\theta_i^\mu) + \mathcal{N}$, where $\mathcal{N}$ indicates Gaussian exploration noise. To avoid completely blind exploration in large action space, the RNN module is introduced to clip the action space based on predicted traffic. During the exploration, the relationship between the action space and phase traffic flow is shown in Table I. As the agent begins to learn, the degree of exploration gradually decreases and eventually tends to balance exploration and exploitation.
3) Store the current experience $(s_t, a_{\text{base},t}, r_t, s_{t+1})$ in the replay buffer for each agent as the dataset for training actor networks and critic networks, where s_t is the observation set of the global network agent, $a_{\text{base},t}$ is the action set of all agents in the base environment where the current intersection is located, and r_t is the reward of the current intersection.

TABLE I
RELATIONSHIP BETWEEN ACTION SPACE AND PHASE TRAFFIC FLOW

Traffic Flow(Veh/h)	Traffic Level	Action Space
≤ 100	low	[0,0.3]
100-400	medium	[0.3,0.7]
≥ 400	high	[0.7,1.0]

4) For each agent i, sample a random minibatch of L samples according to batch normalization [41]. And then, based on the corresponding update rules, network parameter learning is performed on the actor network and the critic network. Finally, the target network is updated based on the soft update method.

IV. SIMULATION AND PERFORMANCE EVALUATION

A. Experimental Setup

The proposed MARL-DSTAN algorithm for TSC is evaluated in Simulation of Urban Mobility (SUMO), which is widely used in TSC research.

1) Road Network Setting: We use the synthetic road network to define the network in SUMO. The experiment is conducted on 4×4 road network scenario, that is, the entire road network contains a total of 16 intersections, and each intersection is set to be a four-way intersection with six lanes with opposite directions of travel. The rightmost lane only allows right-turn traffic, the middle lane only allows through traffic, and the left inner lane only allows left-turn traffic. It should be noted that the entire road network is divided into four base environments, each of which consists of 2×2 intersections. The simulation environment is shown in Fig. 5. Four-phase signal control is used at each intersection, as shown in Fig. 6. The end of each phase is followed by 3-s yellow light to clean the intersection. Vehicles can always turn right when there is no conflicting traffic. The length between the two intersections is 500 m and the road speed limit is 40 km/h.

2) Traffic Flow Setting: To depict the diversity of training samples and consider the real-time traffic condition, we adopt a random strategy to generate the traffic flow on the different road segments. The traffic characteristics of a day are divided into three types, including: 1) high peak; 2) flat peak; and 3) low peak. The average vehicle generation probabilities in different periods of the entire road network are set to 1.48, 1.19, and 0.74 veh/s, respectively.

3) Training Network Parameters Setting: We conduct 500 episodes of training for each RL algorithm. The size of the replay buffer is 10 000, the size of the minibatch is 32, the discount factor γ is 0.95, and the soft update coefficient is 0.99. The learning rate of the actor network and the critic network is set to 1×10^{-4} and 5×10^{-5}, respectively. The network structure of the actor network is shown in Table II. The critic network has the same network structure as the actor network, except that the input dimension is changed to $16 \times 16 + 4 \times 5$ and the output dimension is 1.

4) Compared Methods: We compare our MARL-DSTAN with the following baseline algorithms, including traditional methods and machine learning-based approaches.

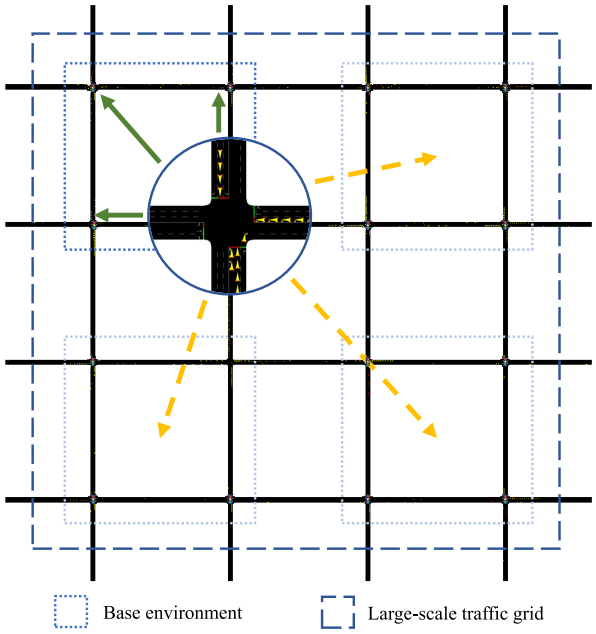


Fig. 5. Simulation environment. The entire road network consists of 4×4 intersections, which are divided into four 2×2 base environments.

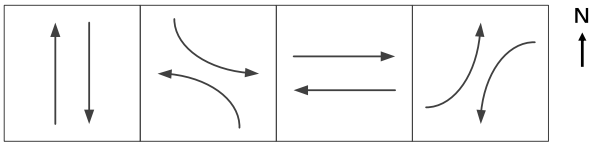


Fig. 6. Diagram of the four phases present in the single intersection.

TABLE II
PARAMETERS OF ACTOR NEURAL NETWORK FOR EACH AGENT

Name	Number	Size	Activation Function
Input Layer	1	16×16	NA
GCN Layer (W^G)	2	16×8,16×8	sigmoid
ATTN Layer (W^Q, W^K, W^V)	2	8×8,8×8	softmax
Fully Connected Layer	2	256,256	ReLU,ReLU6
Output Layer	1	5	sigmoid

1) *Fixed Time:* The fixed timing scheme is a naive method where the duration of each phase is preset based on past experience without any change.
   We use the same fixed timing scheme at each intersection, and each phase duration is set to 20 s.
2) *Longest-Queue-First:* Each intersection agent selects the phase with the corresponding longest queue to decongest local traffic. Update the phase every 20 s.
3) *Center-DDPG:* A centralized RL method for multi-intersection signal control with joint-action modeling. All intersections in the entire road network are controlled by one agent to optimize the global timing plan.

4) *Independent-DDPG:* Each intersection is controlled by one DDPG agent. The cooperation is achieved by sharing state information, but each agent updates its own networks independently.
5) *Temporal-DDPG:* Based on Independent-DDPG, agents limit the action space in the random exploration stage by using the RNN module to predict the traffic flow.
6) *Spatial-DDPG:* Based on Independent-DDPG, agents aggregate road network state information through space modules (GCN and Attention).
7) *Colight:* An algorithm for TSC in a discrete action space through a graph attentional network. For details, refer to [29].

B. Experimental Results

In this section, we investigate the performance of our proposed MARL-DSTAN on halting number, waiting time, and road speed, and compare them with state-of-the-art methods. First, test the performance of the model in a base environment, that is, a 2×2-intersection scenario, and then extend it to a 4×4-intersection scenario to verify the scalability of the model.

1) 2×2-Intersection Scenario: In Fig. 7, we compare MARL-DSTAN's reward at each episode to the corresponding learning curves for the other four RL methods. We can see that the performance of MARL-DSTAN is better than any of the baselines by a large margin, both in the initial exploration stage and in the final convergence stage. From Fig. 7, we discover that the reward of Center-DDPG and Independent-DDPG models start with extremely huge fluctuation and approach to the optimal performance after a long training time. The reason is that the Center-DDPG model is a single-agent central control algorithm with a large state and action dimension, which requires solving the optimization problem over a large joint action space and faces a scalability issue. At the same time, due to the lack of coordination between agents, it is difficult for Independent-DDPG to converge to the global optimal solution. Furthermore, both Temporal-DDPG and Spatial-DDPG perform well on learning performance. But the latter has a slower convergence speed, especially at the beginning of training. The reason is that the RNN module helps the agent to accumulate more valuable experiences during the training process. Compared with Temporal-DDPG and Spatial-DDPG, MARL-DSTAN not only learns the information of the surrounding intersections but also predicts the traffic flow to speed up the acquisition of optimal policy.

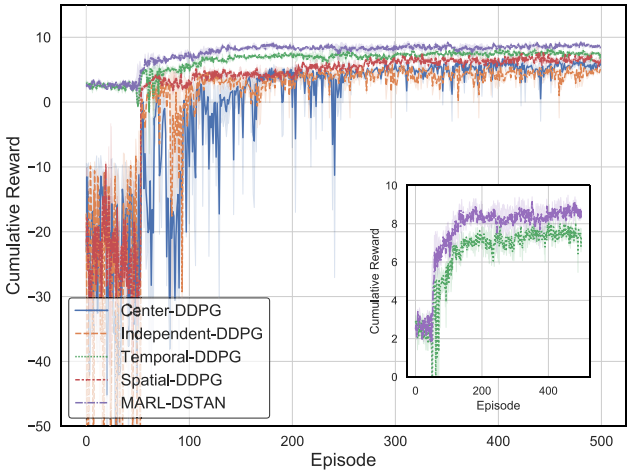


Fig. 7. Reward of different algorithms during each episode of training in the 2×2-intersection scenario.

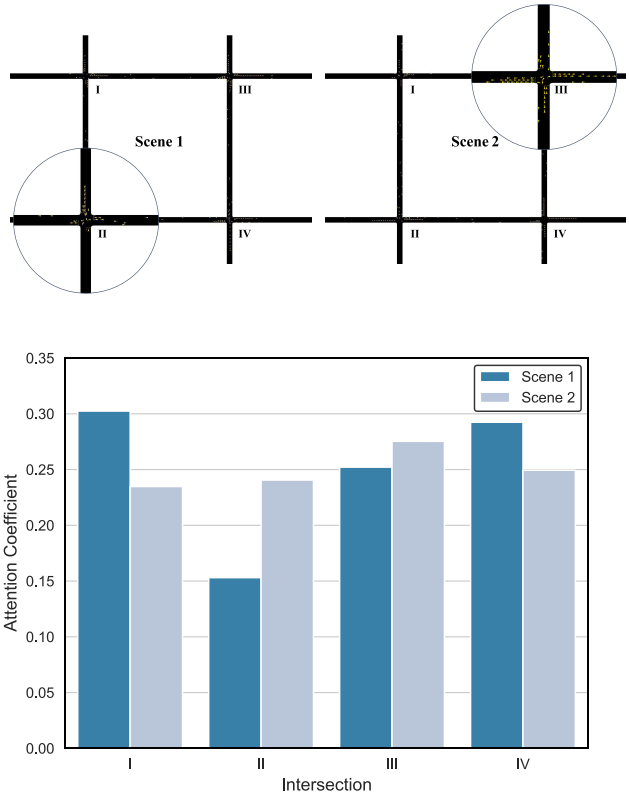


Fig. 8. Attention coefficient of intersection III to different intersections in different scenarios.

Fig. 8 takes intersection III as an example to show the relationship between the attention coefficients of intersection III and other intersections in different scenarios. In scene 1, the traffic flow at different intersections is significantly different, where intersection I and IV have many vehicles and intersection II has fewer. In scene 2, the traffic flow at different intersections is evenly distributed with a small difference. As shown in Fig. 8, intersection III gives high attention to intersections I and IV in scene 1, while intersection II is the opposite. Such disparity of attention coefficient agrees with our expectations, that intersections I and IV are close to intersection III and have more influence on intersection III. On the other hand, in scene 2, the attention coefficient at each intersection is basically the same due to the small difference at each intersection. Then, intersection III gives more attention to itself.

We define the halting number as the total number of halting vehicles at each intersection every second, where a speed of less than 0.1 m/s is considered a halt. The average waiting time indicates the sum of the waiting time of all vehicles within 1 s at the intersection. As shown in Table III, our proposed MARL-DSTAN outperforms the traditional method and other

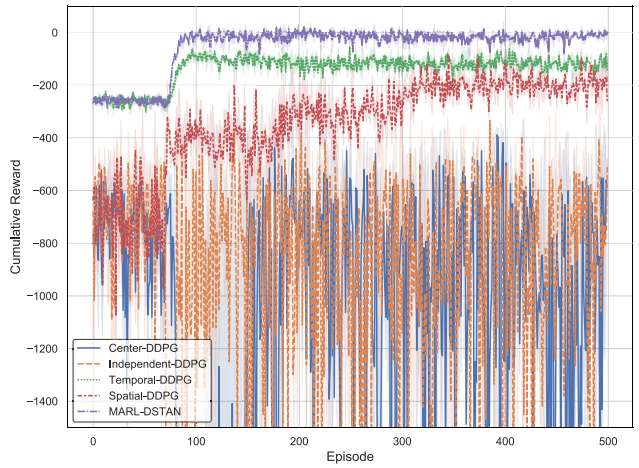


Fig. 9. Reward of different algorithms during each episode of training in the 4×4-intersection scenario.

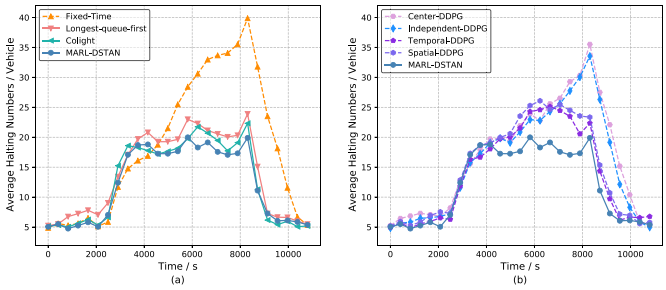


Fig. 10. Comparison of halting number over different algorithms in the 4×4-intersection scenario.

RL methods in average halting number, average waiting time, as well as average speed. Specifically, the performance of the average halting number is improved up to 36.18% over the fixed-time scheme, and the average waiting time is saved by 46.69%.

2) 4×4-Intersection Scenario: In this section, we give an analysis of the simulation results of 4×4 intersections. As is shown in Table IV and the convergence curve in Fig. 9, MARL-DSTAN performs consistently better than other RL methods as the number of intersections increases. In addition, it can be more clearly seen that the convergence speed of Spatial-DDPG is slower, and its reward value fluctuates more sharply. While other algorithms, such as Center-DDPG and Independent-DDPG, are hard to learn the optimal policy as the network scale increases. It should be noted that the following data are analyzed based on the optimal results of each algorithm.

Fig. 10 presents the performance of the halting number during the test period. The simulation time of 10 800 s is divided into a period of time every 400 s for statistical analysis. As can be seen in Fig. 10, each algorithm shows good performance during the low peak period. This is due to the small number of vehicles, and the performance is less affected by the timing strategy. Because of the lack of adaptability and predictability, the fixed timing scheme cannot adjust the timing strategy in time according to the traffic changes during the high peak. This can easily lead to congestion at the intersection, and then the

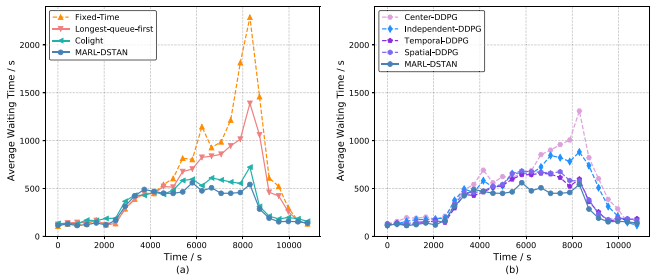


Fig. 11. Comparison of waiting time over different algorithms in the 4×4-intersection scenario.

chain reaction will lead to a rapid decline in the performance of the entire area. Longest-queue-first can give priority to the phase with the longest queue, so it can take care of most vehicles, and the halting number indicator reaches 13.93 vehicles. As the scale of intersections expands, the disadvantages of other RL-based algorithms have emerged, especially the Center-DDPG and Independent-DDPG algorithms. The curse of dimensionality makes it difficult for them to learn effective knowledge. Meanwhile, the results are fluctuating and unable to reach the optimal solution. Similarly, Temporal-DDPG lacks the learning ability between intersections and between regions, so it is also difficult to converge to the best. Spatial-DDPG shows similar performance to Temporal-DDPG due to the lack of valuable samples. Colight shows the performance second only to MARL-DSTAN in the halting number, but the discrete phase selection method reduces the performance of the algorithm to a certain extent. In addition, this method requires the agent to make decisions in a shorter period, resulting in a waste of computing resources. As shown in Table IV, our proposed algorithm reduces the average halting number to 12.17 vehicles, which is a 33.79% reduction compared to the fixed timing scheme.

In Fig. 11, the performance of waiting time is calculated under different TSC methods. From this figure, we can see that the performance of waiting time is consistent with the halting number. The Longest-queue-first algorithm does not perform well in terms of waiting time. The reason is that the algorithm easily causes some short queues to be delayed and cannot be released, which may cause the waiting time of some vehicles to be too long, so the algorithm lacks fairness considerations. Our MARL-DSTAN method outperforms the other seven strategies, especially during the peak hours (i.e., 4000–8000 s). As shown in Table IV, our MARL-DSTAN model reduces the waiting time at each intersection to about 314.70 s, where the average reduction is about 49.51% for the fixed timing scheme and 12.63% for the Colight. The average speed is calculated based on four time periods, each of which is 2700 s. As shown in Fig. 12, MARL-DSTAN also shows the best performance. The performance improvements are attributed to the benefits of capturing the spatiotemporal dependency of the entire road network.

Finally, we give the attention coefficient graph under the 4×4 road network, as shown in Fig. 13. The graph takes the intersection of the third row and the third column as an example for analysis and counts the average value of the attention

TABLE III
COMPARISON OF VARIOUS INDICATORS OF 2×2 INTERSECTIONS BASED ON FIXED TIMING WITH DIFFERENT ALGORITHMS

	Average Halting Number / veh		Average Waiting Time / s		Average Speed / (km/h)	
Fixed Time	15.34	——	404.33	——	15.54	——
Longest-queue-first	11.85	-22.75%	352.54	-12.81%	16.13	+3.80%
Center-DDPG	11.35	-26.01%	284.23	-29.70%	16.27	+4.70%
Independent-DDPG	11.47	-25.23%	292.24	-27.72%	16.15	+3.93%
Temporal-DDPG	10.78	-29.73%	263.13	-34.92%	16.45	+5.86%
Spatial-DDPG	11.15	-27.31%	277.85	-31.28%	16.34	+5.15%
Colight	10.08	-34.29%	236.87	-41.42%	16.61	+6.89%
MARL-DSTAN	**9.79**	**-36.18%**	**215.54**	**-46.69%**	**16.85**	**+8.43%**

TABLE IV
COMPARISON OF VARIOUS INDICATORS OF 4×4 INTERSECTIONS BASED ON FIXED TIMING WITH DIFFERENT ALGORITHMS

	Average Halting Number / veh		Average Waiting Time / s		Average Speed / (km/h)	
Fixed Time	18.38	——	623.34	——	19.92	——
Longest-queue-first	13.93	-24.21%	494.00	-20.75%	20.23	+1.56%
Center-DDPG	16.86	-8.27%	514.71	-17.43%	19.94	+0.10%
Independent-DDPG	15.94	-13.28%	450.17	-27.78%	20.38	+2.31%
Temporal-DDPG	14.04	-23.61%	365.63	-41.34%	20.67	+3.77%
Spatial-DDPG	14.82	-19.37%	381.93	-38.73%	20.45	+2.66%
Colight	12.67	-31.07%	360.21	-42.21%	20.76	+4.22%
MARL-DSTAN	**12.17**	**-33.79%**	**314.70**	**-49.51%**	**21.06**	**+5.72%**

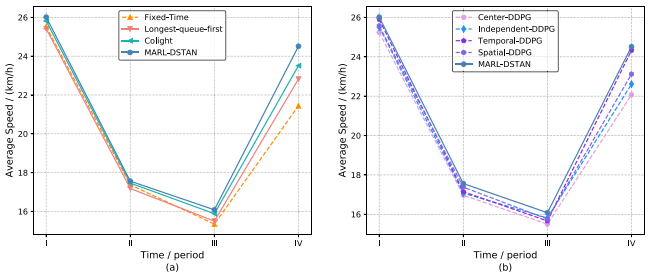


Fig. 12. Comparison of average speed over different algorithms in the 4×4-intersection scenario.

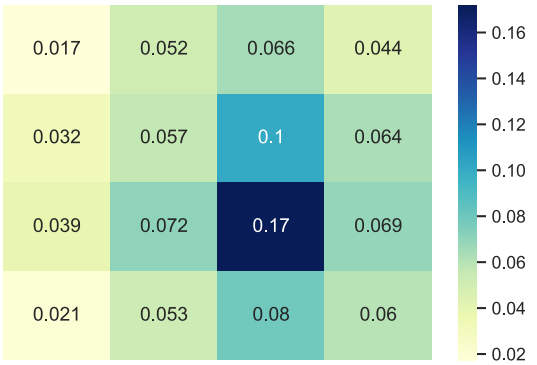


Fig. 13. Attention coefficient graph in the 4×4-intersection scenario. (Take the intersection agent in the third row and third column as an example.)

coefficients of other intersections in the whole process. It is clear that the attention of the intersection to itself will be higher than that of other intersections. Besides, as the distance keeps getting farther, the attention coefficient of the intersection shows a downward trend.

V. CONCLUSION

In this article, we propose an MARL-DSTAN model that considers the temporal and spatial characteristics of the road network to reduce congestion. Considering the issue of dimensional disasters caused by the large scale of the road network, we adopt the hierarchical idea to divide the whole network into multiple base environments. The spatial dependency module combines GCN and attention mechanism to extract the state information of the basic environment and the entire road network in a targeted manner. The temporal dependence module restricts the action space by predicting the flow, thereby improving learning efficiency and accelerating algorithm convergence. In addition, in the basic environment, we introduce the idea of "centralized training and decentralized execution" to improve the stability of agent learning. The simulation results show that the proposed MARL-DSTAN model is better than the fixed timing scheme and other RL methods. MARL-DSTAN can be well adapted to the network-scale road network, with significant improvement in the halting number, waiting time, and average speed.

REFERENCES

[1] P. R. Anciaes, P. J. Metcalfe, and C. Heywood, "Social impacts of road traffic: Perceptions and priorities of local residents," *Impact Assess. Project Appraisal*, vol. 35, no. 2, pp. 172–183, 2017.

[2] *Traffic Congestion Costs U.S. Cities Billions Of Dollars Every Year*. Accessed: May 10, 2020. [Online]. Available: https://www.forbes.com/sites/niallmccarthy/2020/03/10/traffic-congestion-costs-us-cities-billions-of-dollars-every-year-infographic/?sh=2aa28a2b4ff8

[3] C2ES. *Decarbonizing U.S. Transportation*. Accessed: Jul. 2018. [Online]. Available: https://www.c2es.org/document/decarbonizing-u-s-transportation/

[4] Y. Wang, X. Yang, H. Liang, and Y. Liu, "A review of the self-adaptive traffic signal control system based on future traffic environment," *J. Adv. Transp.*, vol. 2018, Jun. 2018, Art. no. 1096123.

[5] S. Yang, B. Yang, H.-S. Wong, and Z. Kang, "Cooperative traffic signal control using multi-step return and off-policy asynchronous advantage actor–critic graph algorithm," *Knowl. Based Syst.*, vol. 183, Nov. 2019, Art. no. 104855.

[6] P. R. Lowrie, *SCATS, Sydney Co-Ordinated Adaptive Traffic System: A Traffic Responsive Method of Controlling Urban Traffic*. Darlinghurst, NSW, Australia: Roads Traffic Authority NSW, 1990.

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

[8] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998.

[9] B. Abdulhai, R. Pringle, and G. J. Karakoulas, "Reinforcement learning for true adaptive traffic signal control," *J. Transp. Eng.*, vol. 129, no. 3, pp. 278–285, 2003.

[10] L. Shoufeng, L. Ximin, and D. Shiqiang, "*Q*-learning for adaptive traffic signal control based on delay minimization strategy," in *Proc. IEEE Int. Conf. Netw. Sens. Control*, 2008, pp. 687–691.

[11] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013. [Online]. Available: arXiv:1312.5602.

[12] J. Skach, B. Kiumarsi, F. L. Lewis, and O. Straka, "Actor–critic off-policy learning for optimal control of multiple-model discrete-time systems," *IEEE Trans. Cybern.*, vol. 48, no. 1, pp. 29–40, Jan. 2018.

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

[14] S. Wang, X. Xie, K Huang, J. Zeng, and Z. Cai, "Deep reinforcement learning-based traffic signal control using high-resolution event-based data," *Entropy*, vol. 21, no. 8, p. 744, 2019.

[15] C.-J. Choe, S. Baek, B. Woon, and S.-H. Kong, "Deep *Q* learning with LSTM for traffic light control," in *Proc. IEEE 24th Asia–Pac. Conf. Commun. (APCC)*, 2018, pp. 331–336.

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

[17] W. Genders, "Deep reinforcement learning adaptive traffic signal control," Ph.D. dissertation, McMaster Univ., Hamilton, ON, Canada, 2018.

[18] 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.

[19] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi, "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3826–3839, Sep. 2020.

[20] J. A. Calvo and I. Dusparic, "Heterogeneous multi-agent deep reinforcement learning for traffic lights control," in *Proc. AICS*, 2018, pp. 2–13.

[21] N. Suematsu and A. Hayashi, "A multiagent reinforcement learning algorithm using extended optimal response," in *Proc. 1st Int. Joint Conf. Auton. Agents Multiagent Syst.*, 2002, pp. 370–377.

[22] 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.

[23] D. Kim and O. Jeong, "Cooperative traffic signal control with traffic flow prediction in multi-intersection," *Sensors*, vol. 20, no. 1, p. 137, 2020.

[24] 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.

[25] L. Zhao *et al.*, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

[26] F. Zhou, Q. Yang, K. Zhang, G. Trajcevski, T. Zhong, and A. Khokhar, "Reinforced spatio-temporal attentive graph neural networks for traffic forecasting," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6414–6428, Jul. 2020.

[27] F. X. Devailly, D. Larocque, and L. Charlin, "IG-RL: Inductive graph reinforcement learning for massive-scale traffic signal control," 2020. [Online]. Available: arXiv:2003.05738.

[28] Y. Wang *et al.*, "STMARL: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control," 2019. [Online]. Available: arXiv:1908.10577.

[29] H. Wei *et al.*, "Colight: Learning network-level cooperation for traffic signal control," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manag.*, 2019, pp. 1913–1922.

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

[31] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.

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

[33] N. Heess *et al.*, "Emergence of locomotion behaviours in rich environments," 2017. [Online]. Available: arXiv:1707.02286.

[34] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[35] R. Lowe *et al.*, "Multi-agent actor–critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

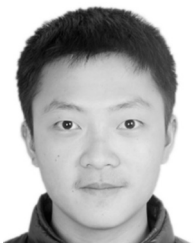
[37] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. [Online]. Available: arXiv:1609.02907.

[38] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[39] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[40] M. Xu, J. Wu, L. Huang, R. Zhou, T. Wang, and D. Hu, "Network-wide traffic signal control based on the discovery of critical nodes and deep reinforcement learning," *J. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 1–10, 2020.

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn. (PMLR)*, 2015, pp. 448–456.



Hao Huang received the B.E. degree in communication engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2019. He is currently pursuing the Ph.D. degree in communication engineering with the Beijing University of Posts and Telecommunications, Beijing, China.

His current research interests include reinforcement learning and intelligent transportation.

Zhiqun Hu received the B.E. degree in communication engineering from the Hubei University of Technology, Wuhan, China, in 2012, and the Ph.D. degree in information and communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2018.

She has been with Hubei University, Wuhan, since 2018, where she is currently an Assistant Professor of Computer Science and Information Engineering. Her main research interests include V2X, reinforcement learning, and intelligent transportation.

Zhaoming Lu received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012.

He is currently an Associate Professor with the School of Information and Communication Engineering, Beijing University of Posts and Telecommunications. His research includes open wireless networks, QoE management in wireless networks, software-defined wireless networks, cross-layer design for mobile video applications, and network-assisted autonomous driving.

Xiangming Wen (Senior Member, IEEE) received the B.E., M.S., and Ph.D. degrees in electrical engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 1982, 1992, and 2002, respectively.

He is currently a Professor with the Communication Network Center, BUPT, where he is also the Director of the Beijing Key Laboratory of Network System Architecture and Convergence. He is the Principle Investigator of more than 18 projects, including the National Key Project of Hi-Tech Research and Development Program of China (863 Program) and the National Natural Science Foundation of China. He is the Vice Director of the Organization Committee of the China Telecommunication Association. In the last five years, he has authored more than 100 published papers. His current research is focused on broadband mobile communication theory, multimedia communications, and information processing.