Neighborhood Cooperative Multiagent Reinforcement Learning for Adaptive Traffic Signal Control in Epidemic Regions

Chengwei Zhang, Yu Tian, Zhibin Zhang, Wanli Xue, Xiaofei Xie, Tianpei Yang, Xin Ge, and Rong Chen

***Abstract*— Nowadays, multiagent reinforcement learning (MARL) have shared significant advances in the adaptive traffic signal control (ATSC) problems. For most of the researches, agents are all isomorphic, which disregards the situation in which isomerous intersections cooperative together in a real ATSC scenario, especially in epidemic regions where different intersections have quite different levels of importance. To this end, this paper models the ATSC problem as a networked Markov game (NMG), in which agents take into account information, including traffic conditions of it and its connected neighbors. A cooperative MARL framework named neighborhood cooperative hysteretic DQN (NC-HDQN) is proposed. Specifically, for each NC-HDQN agent in the NMG, first, the framework analyses correlation degrees with their connected neighbors and weighs observations and rewards by these correlations. Second, NC-HDQN agents independently optimize their strategies on the weighted information using hysteretic DQN (HDQN), which is designed to learn optimal joint strategies in cooperative multiagent games. Third, a rule-based NC-HDQN method and a Pearson correlation coefficient based NC-HDQN method, *i.e.*, empirical NC-HDQN (ENC-HDQN) and Pearson NC-HDQN (PNC-HDQN), respectively, are designed. The first method maps the correlation degree between two connected agents according to vehicle numbers on roads between the two agents. In contrast, the second method uses the Pearson correlation coefficient to calculate the correlation degree adaptively. Our methods are empirically evaluated in both a synthetic scenario and two real-world traffic scenarios and give better performances in almost every standard test metric for ATSC.**

***Index Terms*— Multi-agent learning, cooperative Markov game, independent reinforcement learning, adaptive traffic signal control.**

Manuscript received 11 August 2021; revised 16 January 2022, 21 March 2022, and 19 April 2022; accepted 22 April 2022. Date of publication 13 May 2022; date of current version 5 December 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61906027 and Grant 61906135 and in part by the China Postdoctoral Science Foundation Funded Project under Grant 2019M661080. The Associate Editor for this article was Y.-J. Zheng. *(Corresponding author: Wanli Xue.)*

Chengwei Zhang, Yu Tian, Xin Ge, and Rong Chen are with the School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China (e-mail: chenvy@dlmu.edu.cn; ty97@dlmu.edu.cn; ge_xin@dlmu.edu.cn; rc@dlmu.edu.cn).

Zhibin Zhang and Wanli Xue are with the School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300355, China (e-mail: tjut-zzb@hotmail.com; xuewanli@email.tjut.edu.cn).

Xiaofei Xie is with the School of Computing and Information Systems, Singapore Management University, Singapore 188065, Singapore (e-mail: xfxie@ntu.edu.sg).

Tianpei Yang is with the College of Intelligence and Computing, Tianjin University, Tianjin 300350, China (e-mail: tpyang@tju.edu.cn).

Digital Object Identifier 10.1109/TITS.2022.3173490

I. Introduction

TRAFFIC congestion has become a common problem in modern society, seriously affecting people's daily lives and social development. The ongoing outbreak of novel coronavirus exerts immense pressure on existing urban transport infrastructure because good traffic conditions are a necessary condition for patients to receive timely treatment. Adaptive traffic signal control (ATSC) [1]–[3] adaptively controls traffic signal cycles and becomes a research hotspot to relieve traffic congestion in recent years.

ATSC aims to reduce traffic jams in the way that learns to control the signal phase adaptively. To alleviate this problem, multiagent reinforcement learning (MARL) learns policies for traffic scenarios with multiple intersections and has shown promising results [4]–[6] in reducing potential congestion. Considering that in an ATSC scenario, each intersection's state observation and reward are dependent on the joint policy of all intersections, and all intersections have consistent optimization objectives, to model the ATSC problem as a cooperative multiagent game is a reasonable solution. Unlike traditional MARL test scenarios, such as robot control and games, limiting agents' numbers, a real-world ATSC scenario may have thousands of intersections. The massive intersections make it impossible to employ the commonly used centralized training distributed execution (CTDE) mechanism [7], [8] in the MARL field to an ATSC scenario. Not surprisingly, most ATSC work learns cooperation policy based on an independent training strategy. Each intersection employs a single RL agent to control traffic signals through local observation and messages from other coordinating intersections [5], [9]–[12].

The previously mentioned methods considered the ATSC problem a global-level multiagent cooperative game [4] or a neighborhood-level [9], [10], [12] multiagent cooperative game among intersections. Intersections in these works treat other intersections equally or with fixed weights according to their distance in graph-based networks. In the ATSC scenarios, information from different intersections at different time may have different levels of importance. The correlation between intersections is affected not only by the network relationship between two intersections but also by the vehicle distribution of the two adjacent intersections. For instance, if the number of vehicles on the road between two intersections is small, there is minimal correlation between them at this state. If we consider the information of other intersections with equal

treatment, it will not only waste computing resources but also affect the learning performance. We will verify this statement in the experimental section, which shows that in a real-world traffic network with few vehicles, the training performances of some well-designed MARL ATSC methods are inferior to those of simple RL algorithms, *e.g.*, independent learning with DQN (IDQN) [13]). Considering that traffic flow distribution may be quite uneven in epidemic areas, traffic flows near hospitals should be significantly increased. In contrast, traffic flows in epidemic communities are far less than usual due to strict traffic control. In this case, the importance of different intersections varies, resulting in different influences between them. Thus, it is essential to consider the correlation between two intersections to learn a cooperative strategy.

This work assumes that the status of neighboring intersections is observable by an intersection and models the ATSC problem as a networked Markov game, where intersections are modeled as connected agents according to the road network. Each agent independently controls an intersection with the intersection's local observations and information obtained from its connected neighbors. Next, we propose a neighborhood-level cooperative MARL framework, named neighborhood cooperative hysteretic DQN (NC-HDQN), which considers the correlation between two intersections. Specifically, for each NC-HDQN agent (intersection) in NMG, first, the framework analyses the correlation degrees between it and its connected neighbors and weighs observations and rewards obtained from environment by these correlation degrees. Based on this, multiple NC-HDQN agents independently interact with environments and cooperatively optimize their strategies by decentralized training using Hysteretic DQN (HDQN) [13], which is a independent learning cooperative MARL algorithm designed to learn the optimal cooperative strategy. Second, we propose two NC-HDQN algorithms based on different correlation calculation methods, *i.e.*, a rule-based method named empirical NC-HDQN (ENC-HDQN) and a Pearson correlation coefficient based method named Pearson NC-HDQN (PNC-HDQN). The first method maps the correlation degree between an agent and its neighbors according to the number of vehicles on roads between two adjacent intersections. In contrast, the second method uses the Pearson correlation coefficient to adaptively calculate the correlation degree of each pair of agents. In order to verify the experimental effect, we experiment our methods in a synthetic scenario and two real-world scenarios. The experimental results shown that our methods outperform state-of-the-art ATSC methods in the most commonly employed metrics, *i.e.*, travel time, queue length, and throughput.

The rest of this paper is structured as follows: Section II introduces related work. Section III introduces necessary notations of ATSC and RL. The proposed networked ATSC model is introduced in Section IV. NC-HDQN and the two NC-HDQN framework-based algorithms are proposed in Section V. Models and algorithms are evaluated in Section VI. The paper is concluded in Section VII.

II. Related Work

Traditional traffic signal control methods [14]–[16] generally have pre-defined rules based on expert experience strategies. These methods cannot adaptively adjust the signal phase according to real-time traffic flow information. In recent years, deep reinforcement learning (RL) has shown remarkable advantages in ATSC problems. Existing RL works in ATSC with multiple intersections can be roughly divided into two classes according to joint-action learning and independent learning.

A. Joint-Action Learning

Joint-action learning evaluates returns of actions of all agent directly to learn the optimal joint strategy of all traffic intersections. These methods may lead to a dimensional disaster in large traffic networks due to the exponential growth of the action space with the number of intersections. To this, Van der Pol and Oliehoek [17] factorized the global Q-function as a linear combination of local subproblems. To ensure that individual agents consider other agents' learning processes, Tan *et al.* [18] regarded the joint Q-function as a weighted sum of local Q-functions by minimizing the difference from the global return. In addition, to facilitate the calculation of joint action probabilistic reasoning, Zhu *et al.* [19] proposed a joint tree in a probability graph model. Although joint action learning is hot research in the field of MARL, such as the well-known centralized training and decentralized execution framework-based methods [8], [20], [21], their training effects are largely affected by the agents' numbers. There are also works attempting to expand the agents' scale, *e.g.*, Wang *et al.* [22] proposed a cooperative double Q-learning (Co-DQL) method for ATSC problem. Co-DQL uses the mean-field approximation [23] to model the interaction among agents, where the joint-action of other agents is averaged as a scalar by the mean-field theory to reduce the scale of agents' action space in a large-scale environment. Compere with joint-action learning, independent methods is more widely used in ATSC.

B. Independent Learning

In the literature on independent learners, each RL agent dispersedly controls an intersection and usually has partial visibility of the environment. The cooperation of multiple independent learning agents is mainly reflected in the communication between agents about their observations and policies [24]. Literature of this kind of works mainly focuses on how to obtain perfect global state features through local information and interaction between intersections. MA2C [12] extends the independent A2C algorithm to multiagent scenarios by considering state information and strategies of neighbors. Wei *et al.* [9] combines max pressure [25] to multiagent RL to get a more intuitive representation of the state and reward. Recently, the popular graph convolutional neural network (GCN) [26] provides a better way to extract state information features for problems with network topology, such as the ATSC problem. Nishi *et al.* [27] models

the influence of neighbors by the fixed adjacency matrix defined in GCN, in which the influences between neighbors are static. Hu *et al.* [28] combines traffic prediction and RL control, in which RL agents control intersections using their real-time observations and short-term future information predicted by GCN. However, those two algorithms did not take into account the influence of different neighbors. To this, Wei *et al.* [4] adds an attention mechanism to the GCN to learn the dynamic influence between the ego agent and its neighbors. Wang *et al.* [29] and Zhao *et al.* [30] combine recurrent neural network for the Spatio-temporal dependency to graph attention neural network for iterative relational reasoning and propose two Spatio-Temporal MARL frameworks, respectively.

In those GCN-based methods, all the agents for different intersections share one learning network, which not only reduces the computational cost but also reduced the diversity of learned strategies. Diversity is also necessary for the ATSC problem, for the optimal strategies of intersections under the same observation in the same network may also be different. To this end, Chen *et al.* [10] specifies the rewards of each intersection to describe the demand for coordination between two neighbors. Zhang *et al.* [31] proposes a neighborhood cooperative Markov game framework, which specified the goal of each intersection as the average accumulated return in the neighborhood and learned cooperation strategies independently according to the 'lenient' principle. Wang *et al.* [32] proposes a decentralized framework based on (A2C) by assigning global control to each local RL agent, where the global information is a concatenation of observations (state and reward information) from neighbor intersections weighted by a graph attention network. Ma and Wu [11] extends MA2C with a feudal hierarchy by splitting the traffic network into a few regions, where each region is controlled by a high-level agent and the low-level agents control the traffic lights in the region. Luan *et al.* [33] proposes a leader-follower paradigm to control intersections by two kinds of heterogeneous agents. However, the distribution of vehicles at different intersections in the ATSC scenario often varies, which may also limit their performance. The attention mechanism based on neural network training may not be sensitive in estimating the correlation between intersections with changes. In this work, we separate the correlation calculation design between intersections from the training of RL agents so that the RL algorithm can deal with dynamic data changes.

III. Basic Notation

This section introduces the basic notations of ATSC and MARL. We list the major notations in in Table I.

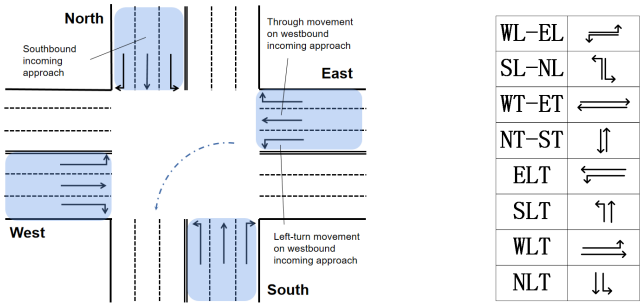
A. ATSC

Traffic network: A traffic network can be defined as a graph $G(\mathcal{N}, \mathcal{E})$, where $i, j \in \mathcal{N}$ and $ij \in \mathcal{E}$ indicate that the two intersections $i, j \in \mathcal{N}$ are physically connected. Similarly, we denote $\mathcal{N}_i = \{j \in \mathcal{N} | (i, j) \in \mathcal{E}\}$ as the set of i's neighbors.

Approach: A roadway meeting at an intersection is referred to as an approach. There are two kinds of approaches: incoming (blue shaded area in Fig. 1(a)) and outgoing approaches.

TABLE I
Notations Table

Notation	Definition
$\mathcal{N}$	Set of intersections
$\mathcal{N}_i$	Set of intersection i's neighbors
$\mathcal{E}$	Edges between adjacent intersections
L_i	Incoming lanes of i
$wait_l$	Number of waiting vehicles alone incoming lane l
$phase_i$	Phase of intersection i
r_i	Individual reward of intersection i
o_i	Observation of agent i
c_{ij}	Coefficient between agent i and j



(a) Intersection (b) Eight phases

Fig. 1. Intersection and signal phases.

Lane: An approach consists of a set of lanes, which can be divided into incoming lanes and outgoing lanes. A traffic movement refers to vehicles moving from an incoming approach to an outgoing approach, which is generally categorized as a though,left turn, and right turn. To avoid conceptual conflict, we also use the lane to represent movement in this article. As shown in Fig. 1(a), an intersection has four incoming approaches: East, South, West, and North, with 12 lanes in total. The incoming lanes at intersection i are defined as $L_i = \bigcup_{j \in \mathcal{N}_i} L_{j,i}$, where $L_{j,i}$ represents all incoming movements originating from all $j \in \mathcal{N}_i$.

Movement signal and phase: A movement signal is defined on the traffic movement, with green (or red) indicating the corresponding movement is allowed (or prohibited). A phase is a combination of nonconflict movement signals. As shown in Fig. 1(b), there are eight phases in total. Here, we employ only four phases as the actions of an intersection: WL-EL, SL-NL, WT-ET, and NT-ST, considering that one traffic light generally controls both going straight and turning right at the same time in real life.

B. Markov Game

The standard multiagent RL model is the so called Markov game (MG), formally defined as a 7-tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \gamma \rangle$. $\mathcal{N}$ is the agent set and $|\mathcal{N}|$ is the set size. $\mathcal{S}$ is the state space. An agent i in M can only observe part of state $s \in \mathcal{S}$, noted by observation o_i. Observations of all agents constitute the joint observation space $\mathcal{O} = \langle \mathcal{O}_1, \ldots, \mathcal{O}_{|\mathcal{N}|} \rangle$. Similarly, $\mathcal{A}_i$ represents the action space of agent i and $\mathcal{A} = \langle \mathcal{A}_1, \ldots, \mathcal{A}_{|\mathcal{N}|} \rangle$ is the joint action space. Transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ maps a state action to a new state which represents the environment changes

caused by agent actions. Reward function $\mathcal{R}_i =: \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ maps a state action to a real value which represents the feedback of agent i caused by agents' joint actions.

Formally, the goal for MARL in an MG is to find a joint policy $\pi = \langle \pi_i, \ldots, \pi_{|\mathcal{N}|} \rangle$ that every agent i can maximize its expected discounted return, *i.e.*, the state-value function $V_i^{\pi}(s) = \mathbb{E}^{\pi}[\sum_{k=0}^{\infty} \gamma^k \mathcal{R}_i(s^{(t)}, \pi(s^{(t)})|s_0 = s]$, where $\pi_i : O_i \rightarrow \Delta(A_i)$ is the individual policy of agent i that maps the observation space of i to a probability distribution of action space, $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The corresponding action-value function is denoted as $Q_i^{\pi}(s^{(t)}, a^{(t)}) = \mathbb{E}^{\pi}[\mathcal{R}_i(s^{(t)} + \gamma V_i^{\pi}(s^{(t+1)}))|s_t]$.

In particular, a Markov game becomes a cooperative game when learning goals among agents are positively correlated, *i.e.*, $\forall i \neq j \in \mathcal{N}, V_i^{\pi}(s) \propto V_j^{\pi}(s)$. Goal of a cooperative Markov game is also usually defined as the average expected discounted return of a team of agents. Similarly, a networked Markov game (NMG) is a Markov game that contains the adjacency relationship among agents, formally be defined as a 7-tuple $\langle \mathcal{G}(\mathcal{N}, \mathcal{E}), \mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \gamma \rangle$, where $(i, j) \in \mathcal{E}$ represents i and j are adjacent, *i.e.*, there exists information sharing between the two agents. Except that the $\mathcal{N}$ is replaced by a graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$, other elements in tuple are the same define with the Markov game.

C. Deep Q Network and Hysteretic DQN

Deep Q Network (DQN) [34] combines deep neural networks with traditional reinforcement learning by representing the action-value function with deep neural networks. The data used by the neural network is dynamically generated by experiences, $\langle s, a, r, s' \rangle$, generated during the interaction with the environment, where r and s' are reward and next state feedback from the environment, respectively. To improve the experience utilization, DQN saves experience in a replay memory buffer for reuse. Denote θ as parameters of the evaluate network utilized in a DQN agent. Formally, θ in DQN is updated by minimizing the TD loss,

$$L_{\theta}^{DQN} = \sum_{k=1}^{B} \delta_k^2 \tag{1}$$

where $\delta = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a'; \theta^-) - Q(s, a; \theta)$ is the TD error, B is the batch size, θ^- is the target network that is slowly synchronized with θ.

A simple way to independently learn the optimal joint policy for cooperative multiagent games is to generate an optimistic update by employing higher importance to positive experiences, *i.e.*, the experience that can increase Q value, which has remarkable results in many cooperative multiagent tasks [13], [33], [35]–[37]. To this, Hysteretic DQN (HDQN) [13] employs two learning rates, α and $\alpha' = \alpha h$, where $0 < h < 1$ to update neural network θ. Formally,

$$L_{\theta}^{HDQN} = \sum_{k=1}^{B} \bar{\delta}_k^2,$$
$$\bar{\delta}_k = \begin{cases} \delta_k, & \delta_k > 0 \\ h\delta_k, & \delta_k \leq 0 \end{cases} \tag{2}$$

In this research, the multi-agent learning optimal joint policy is trained on two novel algorithms based on the HDQN framework. We will show in the experiment section that the simple hysteretic loss definition (Eq. 2) can also solve the ATSC problem well in a reasonable game design.

IV. Networked Markov Game for the MARL ATSC

The target of this paper is to control all intersections in an ATSC environment to optimize the entire traffic condition in an epidemic region. Considering that multiple intersections in a large region typically affect each other and that different intersections usually have different effects, directly extending a single RL method to a multi-intersection scenario may not be appropriate. This assumption is intuitive because connected neighbors of an intersection have a greater influence on the local traffic congestion compared with farther intersections. Another reason is that traffic conditions and vehicle flow distribution in different areas of the same region often vary, which renders the ATSC a global cooperative model. To this end, we model the ATSC problem as a networked Markov game, $\langle \mathcal{G}(\mathcal{N}, \mathcal{E}), \mathcal{S}, \mathcal{O}, \mathcal{A}, P, R, \gamma \rangle$, where an edge (i, j) in $\mathcal{E}$ indicates that intersections i and j are physically adjacent.

In the NMG model, we assume that local observation information could be transmitted between two adjacent intersections, and the goal of each intersection is to optimize the neighborhood traffic conditions according to this information. The assumption is supported in many ATSC works [5], [11], where information about neighbors can be directly shared to the ego intersection.The effectiveness of message sharing from neighbors has been verified in our previous work [31]. The extended observation $o_i \in \mathcal{O}_i$ of an intersection i is defined by local observation of i and its neighbors,

$$o_i^{(t)} = \{\{phase_j^{(t)}\}_{j \in i \cup \mathcal{N}_i}, \{wait[l]^{(t)}\}_{l \in L_i \cup L_{\mathcal{N}_i}}\} \tag{3}$$

where $phase_j^{(t)}$ represents the phase of agent j at time t. L_i and $L_{\mathcal{N}_i}$ represent the incoming lanes of intersection i and its neighbors, respectively. $wait[l]^{(t)}$ indicates how many vehicles are waiting along lane l.

Note that the goal of the ATSC problem is to optimize the traffic situation of a region, and the number of vehicles waiting near intersections is a reasonable evaluation criterion. The total rewards of all agents observed after all intersections taking their actions at time t can be defined by the average number of waiting vehicles in all incoming lanes,

$$r_{\mathcal{N}}^{(t)} = -\sum_{l \in L_i, i \in \mathcal{N}} \gamma^{t-1} wait[l]^{(t+1)} \tag{4}$$

where $\mathcal{N}$ is the set of intersections. Since our goal is to minimize the number of waiting vehicles, we take a negative value in Function 4. However, this definition is limited by the size of the agent and cannot be observed defectively in the NMG, considering that an agent i in the NMG can only observe information from intersection i and its neighbors. Here, we formally define the individual reward of each intersection,

$$r_i^{(t)} = -\sum_{l \in L_i} wait[l]^{(t+1)} \tag{5}$$

The individual reward can be calculated based on the observation of agent i. Thus i can observe a set of individual rewards,

$$\{\boldsymbol{r}\}_i^{(t)} = \{r_j^{(t)}\}_{j\in\{i\}\cup\mathcal{N}_i} \tag{6}$$

Based on their local observation $o_i^{(t)}$ and reward set $\{\boldsymbol{r}\}_i^{(t)}$, agents in NMG need to establish a joint strategy to optimize the overall return.

V. Neighborhood Cooperative MARL

This section proposes a new multiagent reinforcement learning framework named neighborhood cooperative hysteretic DQN (NC-HDQN) based on the hysteretic DQN architecture [13], where multiple intersections collaborate to identify the optimal joint policy of an entire region. Specifically, each intersection is modeled as an NC-HDQN agent, which training their policies independently based on neighborhood observations and rewards and treats other agents as part of the environment. Unlike centralized learning methods, independent learners in the cooperative MARL literature have to face many pathologies to learn an optimal joint policy [35], [37]. The two most affected pathologies are the nonstationary problem and alter-exploration problem. The nonstationary problem is induced by not satisfying the Markov assumption from the individual perspective caused by changes in other agents' policies. The alter-exploration problem is induced by the exploration-exploitation trade-off because the global exploration probability approaches 1 as the number of agents increases. As a result, multiple agents tend to explore conservatively. The alter-exploration problem is why independent learning methods usually fall into the suboptimal trap. To address these problems, HDQN adjusts the utilization rate of different experiences in the training process to improve the probability of positive updating to obtain a more optimistic result in fully cooperative games, *i.e.*, games where agents share a common goal. Therefore, the first problem to be solved by NC-HDQN is to ensure that the training goals of NC-HDQN agents are consistent under the constraints of NMG.

As defined in the previous section, information observed for an agent i in the NMG contains local traffic information about intersection i and its neighbors'. A simple way is to use the average neighborhood return of all agents as their common goal. However, in the ATSC scenario, different neighbors usually have different influences under the congestion situation of an intersection. If two intersections are highly correlated, *i.e.*, agents' goals are consistent, and the optimal strategy of one intersection is bound to be affected by another intersection. Thus direct averaging returns can make sense. If two intersections are independent, it is not significant to consider the information of other intersections or even causing bad effects. We will validate this argument experimentally in the experiment section. There is also an extreme situation in which the correlation between two agents is negatively correlated, *i.e.*, the goals of the two agents conflict. Observably, direct averaging returns in this situation action can cause negative effects. It should be noted that the situation of negative correlation is scarce in real-world ATSC testing. In addition, due to the different influences of neighbors, the importance of information from different neighbors will vary. In a situation in which two intersections are independent, considering the information of other intersections may not make much sense or even causing negative effects due to redundant information.

To solve the above mentioned problems, NC-HDQN weights observations and rewards according to correlations between two adjacent intersections. Denote $c_{ij} \in [-1, 1]$ as the degree of correlation between intersections i and j, where a positive value of correlation suggests that two intersections have a positive correlation, i.e., the value of i increases with an increase in j's value, and a negative value indicates that two intersections have a negative correlation, *i.e.*, the value of i decreases with an increase in j's value. Furthermore, the higher the absolute correlation degree is, the stronger the correlation between two agents. Based on this finding, NC-HDQN refines the observation of i at time t by

$$\tilde{o}_i^{(t)} = \{phase_j^{(t)}, \{|c_{ij}^{(t)}|wait\,[l]^{(t)}\}_{l\in L_j}\}_{j\in\{i\}\cup\mathcal{N}_i} \tag{7}$$

where $c_{ii}^{(t)} = 1$. Function 7 refines the observation of intersection i by reducing the weight of weakly correlated neighbors based on the number of waiting vehicles in incoming lanes between the two intersections. Similarly, we define i's reward as the weighted average of waiting vehicle numbers in neighborhood regions, *i.e.*, in the set $\{i\} \cup \mathcal{N}_i$, weighted by correlation $c_{ij}^{(t)}$.

$$\tilde{r}_i^{(t)} = \frac{\sum\limits_{j\in\{i\}\cup N_i} c_{ij}^{(t)} r_j^{(t)}}{\sum\limits_{j\in\{i\}\cup N_i} c_{ij}^{(t)}} = -\frac{\sum\limits_{j\in\{i\}\cup N_i}\sum\limits_{l\in L_j} c_{ij}^{(t)} wait[l]^{(t+1)}}{\sum\limits_{j\in\{i\}\cup N_i} c_{ij}^{(t)}} \tag{8}$$

With the correlation-based reward function, each agent aims to maximize the returns of its and its positively correlated agents and minimize the returns of its opponents. Each agent may slightly sacrifice its benefit if the team's benefit is increased. In addition, the correlation-based reward function incorporates only the individual rewards of neighborhood regions to reduce the communication cost and improve the speed of policy learning.

Figure 2 shows the overall architecture of the proposed NC-HDQN framework. Obviously, the calculation of correlation c_{ij} is a crucial step in the NC-HDQN framework. In the following two subsections, we introduce a rule-based NC-HDQN method and a Pearson correlation coefficient-based NC-HDQN method. The first method maps the degree of correlation between an agent and neighbors in a given observation according to the number of vehicles between two adjacent intersections. The map between the number of vehicles to the correlation degree needs an empirical setting, so we name it empirical NC-HDQN (ENC-HDQN). The second method relaxes this restriction, which uses the Pearson correlation coefficient [38] to adaptively calculate correlation degrees, so we name it Pearson NC-HDQN (PNC-HDQN).

A. ENC-HDQN

Assumption 1: The correlation degree of two adjacent intersections is positively correlated with the number of vehicles between the two intersections.

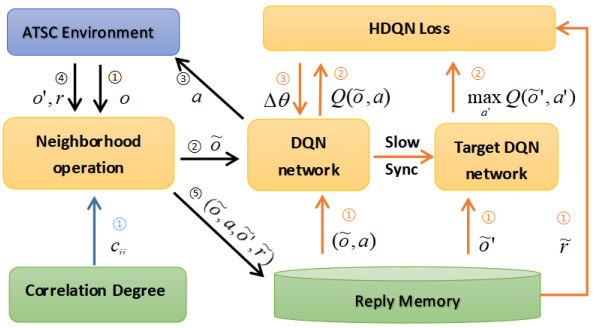


Fig. 2. The architecture of a NC-HDQN agent. NC-HDQN is built on the standard DQN architecture by adding a hysteretic learning strategy (**top right**) to learn cooperative joint strategy, as well as a neighborhood operator **(middle left)** to refine observations and rewards used to train neural networks of DQN or choice actions when interacting with ATSC environments. Actions are selected using the ε-greedy exploration strategy, where the greedy part chooses actions with maximal $Q(\tilde{o}, a)$. Orange arrows and black arrows represent the control flow of the neural network and the interaction flow with the environment, respectively. Correlation degrees between the agent and its neighbors are calculated by a correlation operator (**bottom left**).

ENC-HDQN is built on the above assumption. Intuitively, if there are many vehicles from intersection j to i, then the continuous transportation of vehicles from intersection j will directly affect the reward of intersection i. Otherwise, if there are many vehicles from i to j and j prohibits traffic in this direction, then traffic of i in direction $i \rightarrow j$ will continue to be blocked even if i is allowed to pass. In these two situations, the correlation between the two intersections is high. In contrast, if there are few vehicles between i and j, the correlation between the two intersections is not as strong because it will take a certain amount of time for vehicles to start at an intersection and then to wait at another intersection. In this situation, it is not suitable to take into account nonessential information from connected neighbors with equal treatment. The correlation degree in ENC-HDQN between i and j at t is defined by

$$c_{ij}^{(t)} = \begin{cases} 0 & \sum_{l \in L_{i\rightarrow j} \cup L_{j\rightarrow i}} wait[l]^{(t)} < \frac{\xi}{3} \\ 0.5 & \frac{\xi}{3} < \sum_{l \in L_{i\rightarrow j} \cup L_{j\rightarrow i}} wait[l]^{(t)} < \frac{2\xi}{3} \\ 1 & \frac{2\xi}{3} < \sum_{l \in L_{i\rightarrow j} \cup L_{j\rightarrow i}} wait[l]^{(t)} \end{cases} \tag{9}$$

where ξ is a constant to measure the upper limit vehicles of a line. $\sum_{l \in L_{i\rightarrow j} \cup L_{j\rightarrow i}} wait[l]^{(t)}$indicates how many vehicles are waiting between intersection i and j, which can be obtained directly in $o_i^{(t)}$. $c_{ij}^{(t)}$ divides the correlation between i and j into three categories according to the number of waiting vehicles in incoming lines between i and j.

Algorithm 1 shows details of an ENC-HDQN agent. Similar to DQN, ENC-HDQN has two randomly initialized neural networks, *i.e.*, a Q evaluation network θ_i and a Q target network $\tilde{\theta}_i$. The episode length is T. Agents choose their actions by their neighborhood observations according to the ε-greedy strategy at each step(line 5), where ε gradually decreases from 1 to 0 as the number of episodes increases. After all agents perform their selected actions, each agent gets its and its neighbors' local observations (line 6). Based on these new observations, the ENC-HDQN agent calculates

Algorithm 1 ENC-HDQN for Agent i

1: Initialize Q network θ_i and target Q network $\tilde{\theta}_i$;
Initialize replay memory D_i.
2: **repeat**
3: **while** $t \leq T$ **do**
4: Observes $o_i^{(t)}$, and calculate $\tilde{o}_i^{(t)}$ by Eq.7;
5: Randomly select an action $a_i^{(t)}$ with probability ε; otherwise select $a_i^{(t)} = \arg\max_a Q_i(\tilde{o}_i^{(t)}, a; \theta)$;
6: Observe next observation $o_i^{(t+1)}$ after all agents implement actions;
7: Calculate c_{ij}^t for all $j \in \mathcal{N}_i$ by Eq.9.;
8: Calculate $\tilde{o}_i^{(t+1)}$ and $\tilde{r}_i^{(t)}$ by Eq.7 and Eq.8;
9: Store transition $(\tilde{o}_i^{(t)}, a_i^{(t)}, \tilde{r}_i^{(t)}, \tilde{o}_i^{(t+1)})$ in D_i;
10: Update θ_i by Eq.2;
11: Update $\tilde{\theta}_i$ by $\tilde{\theta}_i \leftarrow \theta_i$ every N_θ steps.
12: **end while**
13: **until** stop condition reached

correlations $c_{ij}^{(t)}$ between it and all its neighbors $j \in \mathcal{N}_i$ at time t (Line 7). The neighborhood reward and new neighborhood observation are calculated according to Eq. 8 and Eq. 7, respectively. The experience of this step is stored in the replay memory of the agent (line 9). The training of the networks follows the normal DRL algorithms.(Lines 10-11).

B. PNC-HDQN

ENC-HDQN uses parameter ξ to describe the degree of traffic congestion between two intersections, related to the length of the line, speed, and vehicles numbers. Its value is affected by the environment, and prior knowledge is needed to design parameters reasonably. In addition, the relationship between two agents can be a positive correlation, negative correlation, and independence for a game. This relationship is influenced by agents' rewards when they execute their actions at each state, as each agent seeks to maximize their cumulative reward. As defined in Eq. 9, correlations between two intersections are always positive. Although in various experimental tests, we didn't observe negative correlations between any two connected intersections during any interaction step and ENC-HDQN also achieved good results, the definition in Eq. 9 is still limited. Here, we propose PNC-HDQN, which calculates the correlation of two agents by the Pearson coefficient of synchronous reward trajectories, to adaptively calculate the correlation coefficient in the interval $[-1, 1]$. The Pearson correlation coefficient is the most commonly employed statistical estimator to measure the correlation between two variables. For two data arrays $X = x_i, 1 \leq i \leq n$ and $Y = y_i, 1 \leq i \leq n$, the Pearson correlation coefficient is:

$$p(X, Y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \tag{10}$$

where $\bar{x}$ and $\bar{y}$ denote the mean of X and Y, respectively.

Algorithm 2 PNC-HDQN for Agent i

Initialize Q network θ_i and target Q network $\tilde{\theta}_i$;
Initialize correlation coefficient $c_{ij} = 1$ for all $j \in \mathcal{N}_i$;
Initialize replay memory $D_i = \emptyset$ and $D_i^\tau = \emptyset$.
repeat
  while $t \leq T$ **do**
    Initialize observations $o_i^{(t)}$, and calculate $\tilde{o}_i^{(t)}$ by Eq.7;
    Randomly select an action $a_i^{(t)}$ with probability ε; otherwise select $a_i^{(t)} = \arg\max_a Q_i(\tilde{o}_i^{(t)}, a; \theta)$;
    Observe next observations $o_i^{(t+1)}$ after all agents implement actions;
    Calculate $\tilde{r}_i^{(t)}$ and $\tilde{o}_i^{(t+1)}$ by Eq.8 and Eq.7;
    Store transition $(\tilde{o}_i^{(t)}, a_i^{(t)}, \tilde{r}_i^{(t)}, \tilde{o}_i^{(t+1)})$ in D_i;
    Update θ_i by Eq.2;
    Update $\tilde{\theta}_i$ by $\tilde{\theta}_i \leftarrow \theta_i$ every N_θ steps.
    Store rewards vector $\{\boldsymbol{r}\}_i^{(t)}$ in D_i^τ;
    Every N_l steps update c_{ij} for all $j \in \mathcal{N}_i$ using D_i^τ by Eq.11, and then reset $D_i^\tau = \emptyset$.
  end while
until stop condition reached

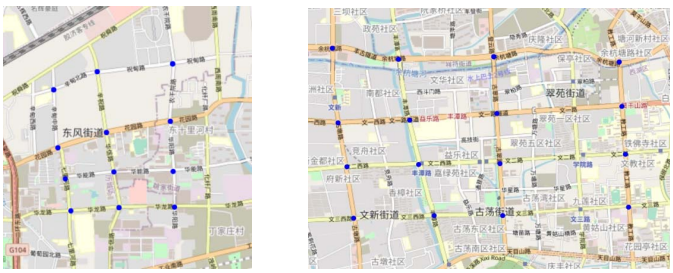
Alg.2 shows the learning process of PNC-HDQN, which is the same as Alg.1, except that c_{ij} is updated periodically, while ENC-HDQN calculates separately for each experience. Specifically, apart from experience replay memory D_i, the PNC-HDQN agent i also maintains a temporary memory D_i^τ of size N_l to store recent individual rewards of it and those of its neighbors. This design meets the requirements of the NAMG model defined in the previous sections. Initially, the correlation coefficient c_{ij} between agent i and each of its neighbors j is set to 1. When the memory D_i^τ becomes full, PNC-HDQN agent i updates c_{ij} for all $j \in \mathcal{N}_i$ as follows:

$$c_{ij} = \frac{\sum_{\{\boldsymbol{r}\}_i \in D_i^\tau} (r_i - \bar{r}_i)(r_j - \bar{r}_j)}{\sqrt{\sum_{\{\boldsymbol{r}\}_i \in D_i^\tau} (r_i - \bar{r}_i)^2} \sqrt{\sum_{\{\boldsymbol{r}\}_i \in D_i^\tau} (r_j - \bar{r}_j)^2}} \quad (11)$$

where $\{\boldsymbol{r}\}_i = \{r_j\}_{j \in \{i\} \cup \mathcal{N}_i}$ is the reward set obtained by agent i (Eq.6), $\bar{r}_i$ and $\bar{r}_j$ are the means of all r_i and r_j stored in D_i^τ, respectively. Using the Pearson correlation coefficient method, agents can identify neighbors' roles, *i.e.*, strongly correlated teammates (or opponents) or weakly correlated teammates (or opponents).

C. Complexity Analysis

Independent learning has natural advantages in algorithm complexity because the information of other agents is not considered. In this work, both ENC-HDQN and PNC-HDQN are built based on HDQN. Compared with HDQN, PNC-HDQN redefined rewards and states with the same dimension for each experience that was stored in experience replay memory, so the space complexity of PNC-HDQN also didn't increase significantly. For ENC-HDQN, experience is stored without adding anything new. Thus, there is almost no increase in space complexity. In the case of the time complexity, both ENC-HDQN and PNC-HDQN train their Q networks by minimizing the loss defined in Eq. 1, where an error judgment is added before the calculating error of each experience. In addition, the two algorithms have a neighborhood operator and correlation operator to redesign the observation and reward values of experiences stored in the ERM, which is independent of the training of neural networks. Considering that the time complexity of training a neural network is much larger than that of the two operators, its time complexity does not increase significantly compared to HDQN.



(a) Jinan network (b) Hangzhou network

Fig. 3. The two real traffic networks.

VI. Experiment

We apply our methods using a commonly employed traffic simulator, *i.e.*, Cityflow [39], in two real-world traffic networks provided by Jinan and Hangzhou city and a synthetic traffic grid. First, we tested ENC-HDQN, PNC-HDQN and three SOTA MARL methods (MA2C [12], Colight [4] and PressLight [9]) and a traditional traffic method (Fixedtime [14] in the MA2C paper for completeness. We also test the dynamics of the correlation degree of intersections during the training process using PNC-HDQN to verify the issues we mentioned in the previous section. Second, we perform ablation experiments to show that both the neighborhood mechanism and HDQN loss in NC-HDQN are helpful in improving the effectiveness of ATSC. Finally, we test the generalization performance of PNC-HDQN and compare ENC-HDQN and PNC-HDQN by performing robustness experiments of important parameters.

A. Scenarios Setting

We employ three publicly available traffic scenarios, *i.e.*, two real-world traffic scenarios, as shown in Figure 3, and a 4×4 synthetic traffic grid scenario based on Cityflow, to evaluate our methods. All the three scenarios simulate peak-hour traffic, and the action spaces of intersections in each scenario are the same. Considering that traffic signals cannot be changed too frequently in reality, the traffic signals at each intersection are synchronously controlled at 10 seconds. In summary, one step of an RL agent corresponds to 10 seconds in the traffic scenarios, which brings about a fixed length (360) to an episode in the Markov game.

In the three traffic scenarios, vehicle data of traffic flow includes vehicle IDs, entering and leaving locations, arrival time, and their driving routes. In the real-world scenarios, we obtain the vehicle data from the camera data of each intersection in the corresponding traffic environments. In contrast, vehicle data in the synthetic scenario is generated randomly,

TABLE II
SCENARIOS STATISTICS

Scenarios	Intersections	Arrival num(vehicles/300s)			
		Mean	Std	Max	Min
Jinan	12	524.58	98.53	672	256
Hangzhou	16	248.58	40.45	333	212
Synthetic	16	935.92	17.47	960	896

TABLE III
PARAMETER TABLE

Component	Hyper-parameter	Value
DQN network	Learning rate α	0.001
	Discount γ	0.99
	ERM size	200000
	Network optimizer	Adam
	Activation function	Relu
ϵ-greedy	Initial and final ϵ	1.0 and 0.001
	Decay rate d_ϵ	1.0/20000
HDQN	Discount rate h	0.5
ENC-HDQN	ξ in Eq. 9	300 (synthetic) 200 (Jinan, Hangzhou)
PNC-HDQN	Size of D_i^τ	90

where the vehicle number is generated by a Gaussian distribution and the driving route of each vehicle is generated randomly by the categorical distribution $Cat(0.6, 0.1, 0.3)$ with the three probabilities corresponding to go straight, turn left or right when arrives at each intersection, respectively. Thus the synthetic scenario can simulate the uneven traffic flow distribution in epidemic environments. We summarize the detailed vehicle and intersection statistics of the three scenarios in Table II. The table shows the fewest intersections in the Jinan scenario, and the traffic flow of the synthetic scenario is the highest. The synthetic scenario has the most intersections and the highest traffic flow. The higher the traffic flow and intersection number are, the higher the performance requirements of the algorithm.

We choose the following three metrics to evaluate our model: (a) travel time m_t: average of vehicles' travel time, which is the most frequently used metric applied to evaluate ATSC methods; (b) queue length m_q: average of vehicle length in lanes of intersections, and (c) throughput m_{th}: number of vehicles arriving at destination. Formally,

$$\begin{aligned} m_t &= \frac{1}{|\mathcal{V}_{in}|} \sum_{v \in \mathcal{V}_{in}} (t_v^{out} - t_v^{in}) \\ m_q &= \frac{1}{|\mathcal{N}|T} \sum_{t \le T} \sum_{i \in \mathcal{N}} \sum_{l \in L_i} \frac{wait^{(t)}[l]}{|L_i|} \\ m_{th} &= |\mathcal{V}| \end{aligned} \tag{12}$$

t_v^{in} and t_v^{out} are vehicle v enters and leaves times, respectively. $\mathcal{V}$ is the vehicles that reaches their destinations. $\mathcal{V}_{in}$ represents vehicles that enter the traffic scenario.

Here, we employ the architecture of DQN [34] as a basis of the algorithms' architecture. HDQN, ENC-HDQN, and PNC-HDQN have the same Q networks, including two fully connected layers, each of which has 100 hidden nodes, and one output node is for action in the output layer. The hyper-parameters used in these algorithms are also the same. We summarize parameters in Table III. For fairness, we ran source codes of Fixedtime, Colight, PressLight, and MA2C algorithms released in a public benchmark.[1]

B. Performance Comparison

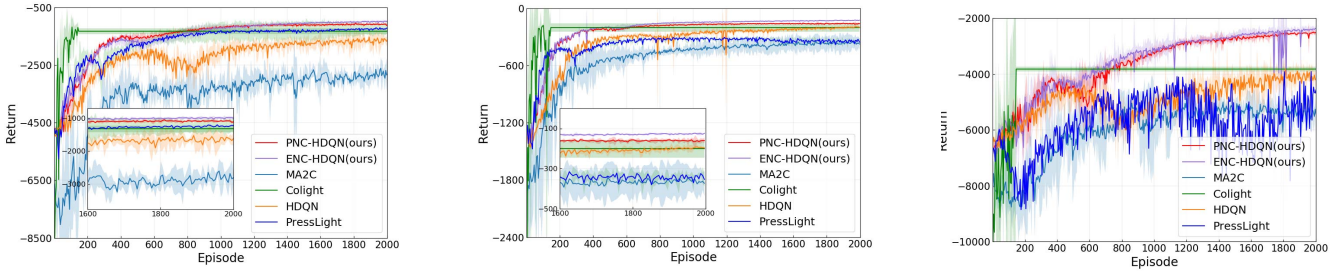
Figure 4 refers to the learning dynamics of the MARL methods mentioned in the three traffic scenarios. The lines and shadows around the curves represent the average cumulative returns and error range of these algorithms on learning episodes. As shown in Figure 4, we can observe that ENC-HDQN performs best among these methods in terms of learning effectiveness, but it is slightly unstable in the synthetic environment. In all settings, PNC-HDQN achieves stable and nearly the highest returns. Colight's convergence rate is faster than other methods and obtains almost the same results as PNC-HDQN in the Jinan and Hangzhou scenarios. In addition, the result of HDQN is similar to that of Colight in the Hangzhou scenario and slightly lags behind it in the other two scenarios. To compare the execution performance, Table IV shows the final statistical data results of the trained MARL policies and the Fixedtime method, which indicates that our algorithm gains the best results in all scenarios. In summary, our methods can reasonably regulate the relationships among various intersections with different traffic levels.

To explain experimental results, we show the dynamics of the correlation degree of intersections during the training process using PNC-HDQN in the two real-world scenarios, as shown in Figure 5(a) and (b). The lines in the figure show the dynamics of the Pearson coefficients between an agent and its four neighboring agents (four adjacent intersections in the: 1 south, 2 east, 3 north, and 4 west directions) and one non-neighboring agent of each scenario. We note that the correlations with neighbors are significantly higher than those with non-neighbors, which shows that our hypothesis in the NC-HDQN is meaningful. In addition, the overall correlation of Jinan is higher than that of Hangzhou, mainly because of the low density of vehicles in Hangzhou. In view of the low correlation of intersections in the Hangzhou environment, we use the basic RL method DQN to make a comparison. Figure 5(c) shows the final returns of Colight, HDQN, MA2C, Presslight, and two DQN settings, *i.e.*, intersections are independently controlled by a DQN agent with neighborhood observations and rewards (NC-IDQN) in which the coefficient between two neighboring intersections are fixed to 1, or a DQN agent with individual observations and rewards (IDQN). The result shows that in the case of low correlation, learning independently with local information is enough to achieve good results, while using global observation and reward definitions, such as MA2C, Colight and PressLight are counterproductive. This finding validates the conclusion in Section V that considering the information of other agents when the two agents exhibit a low correlation is not beneficial.

C. Validation and Robustness Analysis

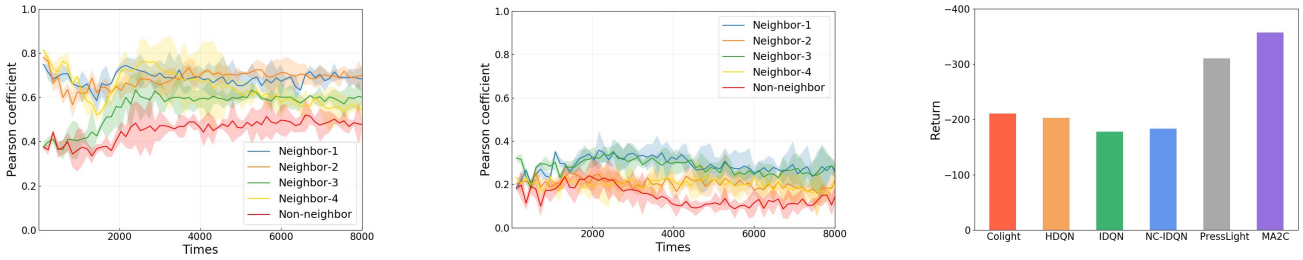
To verify the rationality of the proposed NC-HDQN framework, we need to explain the following three questions:

[1] https://traffic-signal-control.github.io/



(a) Jinan Scenario (b) Hangzhou Scenario (c) Synthetic Scenario

Fig. 4. Performance comparison. Learning dynamics of RL methods (PNC-HDQN '–', ENC-HDQN '–', MA2C '–', Colight '–', HDQN '–',PressLight '–') in the three traffic scenarios.



(a) Jinan (b) Hangzhou (c) Average returns in Hangzhou scenario

Fig. 5. Dynamics of Pearson coefficients during the learning process of independent DQN of an intersection with its four neighbors and a nonneighbor in the (a) Jinan scenario and (b) Hangzhou scenario. (c) Performances of state-of-the-art RL methods in the Hangzhou scenario.

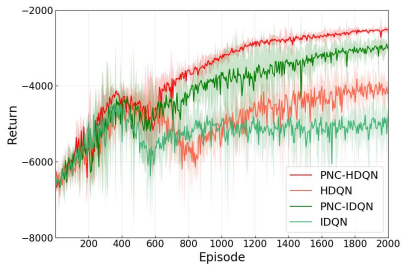
TABLE IV

EXECUTION PERFORMANCE COMPARISON (THE BEST VALUE: 'RED', THE SECOND VALUE: 'BLUE')

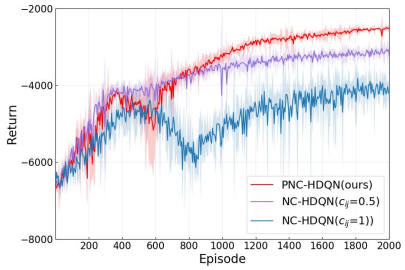
Dataset	Metrics	Fixedtime	Colight	MA2C	PressLight	HDQN	ENC-HDQN	PNC-HDQN
Jinan	m_t	811.92	290.90±3.08	417.34±27.70	281.92±6.63	309.47±20.6827	263.43±1.24	272.49±1.04
	m_q	7.80	1.21±0.09	2.75±0.48	1.14±0.08	1.46±0.24	0.92±0.01	1.04±0.10
	m_{th}	3475	5747.00±53.77	5197.33±160.58	5795.33±22.34	5723.67±59.26	5847.00±29.80	5810.00±17.66
Hangzhou	m_t	705.31	280.10±17.93	321.94±13.42	297.15±2.17	284.28±5.01	268.82±0.43	276.06±1.28
	m_q	2.22	0.18±0.01	0.35±0.08	0.35±0.01	0.19±0.01	0.13±0.01	0.16±0.02
	m_{th}	2000	2774.00±32.45	2723.00±52.83	2280±2.82	2781.67±4.10	2796.67±3.40	2784.00±4.32
Synthetic	m_t	869.88	356.52±50.25	497.19±13.23	534.31±94.42	369.68±13.74	262.49±3.10	272.17±4.96
	m_q	9.75	3.61±1.95	5.05±0.02	5.87±1.67	3.71±0.18	2.20±0.04	2.33±0.07
	m_{th}	5022	9386.00±171.54	7733.00±146.66	5571.5±234.5	9246.66±128.74	10266.79±45.80	10192.33±34.88

(1) whether the two mechanisms, *i.e.*the neighborhood cooperative mechanism and HDQN loss, in the NC-HDQN framework, are effective; (2) whether it is necessary to adaptively learn the correlation coefficient; and (3) how much is the performance of the algorithm affected by the value of the super parameter in the base algorithm HDQN. In this subsection, we performed two sets of ablation experiments to explain questions (1) and (2) and a parameter robustness experiment to explain question (3). The previous subsection shows that algorithms with simple settings can achieve almost optimal effects in the Hangzhou environment. Here, we only show the learning dynamics of each experiment in the scenario with the highest vehicle density, *i.e.*, the synthetic scenario.

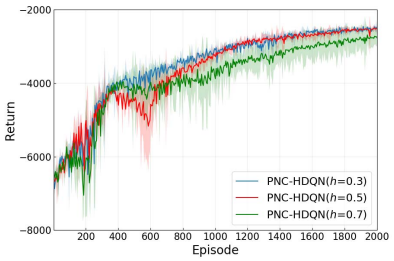
The first ablation experiment tested the framework in two settings, *i.e.*, with or without the NC mechanism, with HDQN and IDQN. Figure 6(a) shows the training curves of the four settings for the synthetic scenario. As seen from the figure, methods' performances with the NC mechanism in the two base RL methods (HDQN and IDQN) are better than those without the NC mechanism. The two experiments using HDQN are also better than those using IDQN. The second ablation experiment tested the necessity of using adaptively learned correlation coefficients. Figure 6(b) shows the learning comparison of PNC-HDQN with two fixed correlation settings on the synthetic scenario, where in NC-HDQN($c_{ij} = 0.5$) and NC-HDQN($c_{ij} = 1$), the weights of neighbors in Eq. 7 and 8 are set to 0.5 and 1, respectively. The experimental results verify the effectiveness of the adaptive mechanism. Figure 6(c) shows the learning curves of PNC-HDQN under different h employed in HDQN loss (Eq. 2) to show the robustness of the parameter. Since h does not belong to the hyperparameters of the NC mechanism, its robustness reflects the conclusion that the performance advantage of our proposed algorithm is not attributed to parameter tuning in HDQN. For completeness, we present the travel time metrics m_t over trained policies in all three scenarios in Table V. The experimental results in Table V and Figure 6 verify the validation of both the NC mechanism and the HDQN loss in NC-HDQN.



(a) W or w/o NC mechanism and HDQN loss



(b) W or w/o adaptive correlation mechanism



(c) Parameter robustness in HDQN (h)

Fig. 6. Validation of NC-HDQN. (a) Learning curves of DQN, PNC-DQN, HDQN and PNC-HDQN in the synthetic scenario; (b) compares PNC-HDQN with two fixed correlation settings, *i.e.*$c_{ij} = 0.5$ and $c_{ij} = 1$, for all contiguous intersection pairs $(i, j; i \neq j)$ in the synthetic scenario. (a) and (b) are two ablation experiments to show the effectiveness of the neighborhood cooperative mechanism in the NC-HDQN framework. (c) Learning curves of PNC-HDQN with different h in the Jinan scenario to verify the robustness of parameters not belonging to the neighborhood cooperative mechanism.

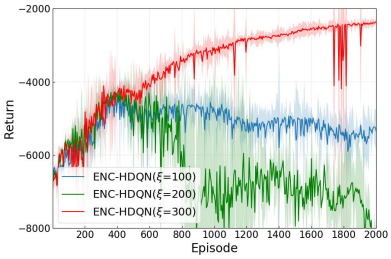


Fig. 7. Learning curves of ENC-HDQN with different ξ in the synthetic scenario.

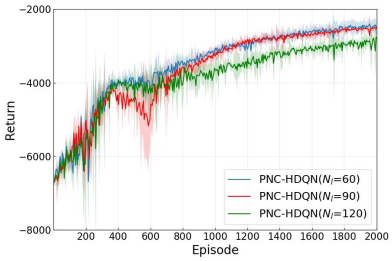


Fig. 8. Learning curves of PNC-HDQN with different N_l in the synthetic scenario.

TABLE V
ABLATION COMPARISON IN ALL SCENARIOS WITH METRIC TRAVEL TIME

	Jinan	Hangzhou	synthesis
IDQN	327.02	280.52	427.43
HDQN	309.47	284.28	369.68
PNC-IDQN	279.80	282.57	319.03
NC-HDQN($c_{ij} = 0.5$)	298.53	274.41	319.56
NC-HDQN($c_{ij} = 1$)	309.47	284.28	369.68
PNC-HDQN	272.49	276.06	272.17

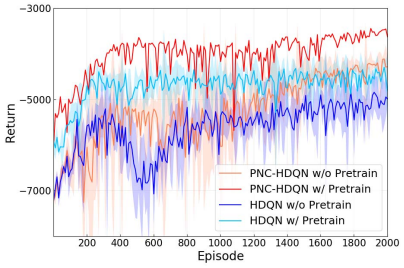


Fig. 9. Learning curves of PNC-HDQN and HDQN (w or w/o pre-trained model).

D. Compare PNC-HDQN With ENC-HDQN

As shown in Subsection VI-B, the ENC-HDQN based on expert experience slightly outperforms PHC-HDQN, which adaptively calculates the correlation coefficient between two intersections in metrics terms in almost all scenarios. Compared with ENC-HDQN, PNC-HDQN has the advantage of more relaxed algorithm conditions. PNC-HDQN does not need to know the vehicle capacity of intersections in a traffic network, which is established as a preset parameter (ξ) in ENC-HDQN. In this subsection, we tested the robustness of ENC-HDQN with different ξ values. Considering that PNC-HDQN uses a temporary memory D_i^τ with size N_l to calculate the correlation coefficient, we experiment PNC-HDQN with different N_l as a comparison. The results are shown in Figure 7 and 8. We note that ENC-HDQN truly needs an appropriate ξ to obtain stable and reasonable results. Since PNC-HDQN does not require additional information, it is better than ENC-HDQN in algorithm adaptability.

E. Generalization Verification of PNC-HDQN

As analyzed in previous sections, PNC-HDQN calculates the correlation between intersections using reward trajectories directly, and RL models of each intersection are trained independently by experiences modified according to the correlation degrees. The advantage of this is that when the traffic flow changes, the correlation between intersections can be obtained quickly so that the algorithm has better generalization performance. In this section, we test the generalization performance of PNC-HDQN. Specifically, we increased the vehicle density through an intersection (the intersection in Row 2 and column 2) in the synthetic environment by 30% to simulate congestion, and applied the pre-trained agents in the original scenario to the new scenario. Figure 9 shows the learning curves of PNC-HDQN and HDQN in settings that with or without the pre-training model. We can see that PNC-HDQN has better initial performance in the new environment and can quickly learn better strategies.

VII. CONCLUSION

This paper modeled the ATSC problem as a Networked Markov game, in which each intersection can observe

information from both it and its connected neighbors. Based on the NMG, we proposed a neighborhood learning framework to preprocess data from neighbors according to the correlation between two intersections. Next, we proposed two decentralized learning multiagent cooperative reinforcement algorithms based on HDQN, named ENC-HDQN and PNC-HDQN, to pursue the optimal joint policy in a multi-intersection environment. We compared our method with SOTA methods in two real-world and one synthetic traffic scenario. Sufficient experimental results verified the effectiveness of NC-HDQN and the two NC-HDQN based algorithms.

In the experimental results, we find an interesting phenomenon in Table IV. Comparing the vehicles' average travel time in the three networks, we find that the optimal results are all about 270, even in the synthetic scenario where vehicle density is the most significant. This finding shows that the vehicle densities in Jinan and Hangzhou have not reached saturation and can accommodate more vehicles. There are many reasons for traffic jams in the real-world environment. One is that the signal control strategy needs to be improved. The more practical reason is the emergencies in the real-world traffic network, such as traffic accidents, vehicles that do not obey the traffic rules, etc. For future work, we will focus on this issue by designing simulation environments with emergencies and control algorithms with high robustness that can deal with uncontrolled environmental conditions.

REFERENCES

[1] H. Wei, G. Zheng, V. Gayah, and Z. Li, "Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation," *ACM SIGKDD Explor. Newslett.*, vol. 22, no. 2, pp. 12–18, Jan. 2021.

[2] Q. Guo, L. Li, and X. J. Ban, "Urban traffic signal control with connected and automated vehicles: A survey," *Transp. Res. C, Emerg. Technol.*, vol. 101, pp. 313–334, Apr. 2019.

[3] L. N. Alegre, T. Ziemke, and A. L. C. Bazzan, "Using reinforcement learning to control traffic signals in a real-world scenario: An approach based on linear function approximation," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 29, 2021, doi: 10.1109/TITS.2021.3091014.

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

[5] T. Chu, S. Chinchali, and S. Katti, "Multi-agent reinforcement learning for networked system control," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–17.

[6] J. Yang, J. Zhang, and H. Wang, "Urban traffic control in software defined Internet of Things via a multi-agent deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3742–3754, Jun. 2021.

[7] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[8] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97, 2019, pp. 5887–5896.

[9] H. Wei *et al.*, "PressLight: Learning max pressure control to coordinate traffic signals in arterial network," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1290–1298.

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

[11] J. Ma and F. Wu, "Feudal multi-agent deep reinforcement learning for traffic signal control," in *Proc. 19th Int. Conf. Auto. Agents Multiagent Syst. (AAMAS)*, 2020, pp. 816–824.

[12] 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.

[13] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 2681–2690.

[14] N. H. Gartner, S. F. Assmann, F. Lasaga, and D. L. Hous, "Multiband—A variable-bandwidth arterial progression scheme," *Transp. Res. Rec.*, no. 1287, pp. 212–222, 1990.

[15] S.-B. Cools, C. Gershenson, and B. D'Hooghe, "Self-organizing traffic lights: A realistic simulation," 2006, *arXiv:nlin/061004*.

[16] J. Y. K. Luk, "Two traffic-responsive area traffic control methods: Scat and scoot," *Traffic Eng. control*, vol. 25, no. 1, pp. 14–22, 1984.

[17] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1–8.

[18] 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.

[19] F. Zhu, H. M. A. Aziz, X. Qian, and S. V. Ukkusuri, "A junction-tree based learning algorithm to optimize network wide traffic control: A coordinated multi-agent framework," *Transp. Res. C, Emerg. Technol.*, vol. 58, pp. 487–501, Sep. 2015.

[20] T. Rashid, M. Samvelyan, C. S. Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4292–4301.

[21] J. Wang, Z. Ren, T. Liu, Y. Yu, and C. Zhang, "Qplex: Duplex dueling multi-agent q-learning," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–27.

[22] X. Wang, L. Ke, Z. Qiao, and X. Chai, "Large-scale traffic signal control using a novel multiagent reinforcement learning," *IEEE Trans. Cybern.*, vol. 51, no. 1, pp. 174–187, Jan. 2021.

[23] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, J. Dy and A. Krause, Eds., Stockholm, Sweden, Jul. 2018, pp. 5567–5576.

[24] S. Sukhbaatar and R. Fergus, "Learning multiagent communication with backpropagation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 2244–2252.

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

[26] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2020.

[27] T. Nishi, K. Otaki, K. Hayakawa, and T. Yoshimura, "Traffic signal control based on reinforcement learning with graph convolutional neural nets," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 877–883.

[28] X. Hu, C. Zhao, and G. Wang, "A traffic light dynamic control algorithm with deep reinforcement learning based on GNN prediction," 2020, *arXiv:2009.14627*.

[29] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong, "STMARL: A spatio-temporal multi-agent reinforcement learning approach for cooperative traffic light control," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2228–2242, Jun. 2020.

[30] 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. 2019.

[31] C. Zhang, S. Jin, W. Xue, X. Xie, S. Chen, and R. Chen, "Independent reinforcement learning for weakly cooperative multiagent traffic control problem," *IEEE Trans. Veh. Technol.*, vol. 70, no. 8, pp. 7426–7436, Aug. 2021.

[32] M. Wang, L. Wu, J. Li, and L. He, "Traffic signal control with reinforcement learning based on region-aware cooperative strategy," *IEEE Trans. Intell. Transp. Syst.*, early access, Mar. 10, 2021, doi: 10.1109/TITS.2021.3062072.

[33] L. Luan *et al.*, "Marl for traffic signal control in scenarios with different intersection importance," in *Distributed Artificial Intelligence*, J. Chen, J. Lang, C. Amato, and D. Zhao, Eds. Cham, Switzerland: Springer, 2022, pp. 93–106.

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

[35] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," *Knowl. Eng. Rev.*, vol. 27, no. 1, pp. 1–31, Feb. 2012.

[36] G. Palmer, K. Tuyls, D. Bloembergen, and R. Savani, "Lenient multi-agent deep reinforcement learning," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 443–451.
[37] G. Palmer, R. Savani, and K. Tuyls, "Negative update intervals in deep multi-agent reinforcement learning," in *Proc. 18th Int. Conf. Auton. Agents MultiAgent Syst.*, 2019, pp. 2681–2690.
[38] Y. Zhang, Q. Yang, D. An, and C. Zhang, "Coordination between individual agents in multi-agent reinforcement learning," in *Proc. 35th AAAI Conf. Artif. Intell. (AAAI)*. Palo Alto, CA, USA: AAAI Press, 2021, pp. 11387–11394.
[39] H. Zhang *et al.*, "CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario," in *Proc. World Wide Web Conf.*, May 2019, pp. 3620–3624.

Chengwei Zhang received the Ph.D. degree in computer software and theory from Tianjin University in 2018. He is currently a Lecturer with the College of Information Science and Technology, Dalian Maritime University. His current research interests include reinforcement learning and multi-agent reinforcement learning.

Yu Tian is currently pursuing the M.Sc. degree in software engineering with Dalian Maritime University. Her current research interests include multi-agent reinforcement learning and traffic signal control.

Zhibin Zhang is currently pursuing the M.Sc. degree in computer technology with the Tianjin University of Technology. His research interests include visual object tracking and deep learning.

Wanli Xue received the Ph.D. degree in technology of computer application from Tianjin University in 2018. Currently, he is a Lecturer with the School of Computer Science and Engineering, Tianjin University of Technology. His research interests include visual tracking and machine learning.

Xiaofei Xie received the B.E., M.E., and Ph.D. degrees from Tianjin University. He is currently an Assistant Professor with Singapore Management University, Singapore. His research mainly focuses on program analysis, traditional software testing, and quality assurance analysis of artificial intelligence. He was a recipient of the two ACM SIGSOFT Distinguished Paper Awards in FSE'16 and ASE'19.

Tianpei Yang received the Ph.D. degree in software engineering from Tianjin University in 2021. Her research interests include artificial intelligence, multi-agent systems, and deep reinforcement learning.

Xin Ge received the Ph.D. degree in computer system architecture from Northeastern University, China, in 2011. His research interests include complex systems, including real world networks topological analysis and data driven modeling of dynamic processes, such as viral spreading, opinion interactions, and behavior contagion.

Rong Chen received the Ph.D. degree in computer software and theory from Jilin University, China, in 2000. He is currently a Professor with the College of Information Science and Technology, Dalian Maritime University. His research interests include software diagnosis, collective intelligence, activity recognition, Internet, and mobile computing.