Cooperative Multi-Agent Reinforcement Learning Framework for Edge Intelligence-Empowered Traffic Light Control

Haiyong Shi, Bingyi Liu, Enshu Wang, Weizhen Han, *Graduate Student Member, IEEE*, Jinfan Wang, Shihong Cui, and Libing Wu, *Member, IEEE*

***Abstract*—Edge Intelligence (EI) technologies obtain an advance with promotion by Consumer Electronics (CE) and spread to the Intelligent Transportation System (ITS). As part of the edge in ITS, traffic lights suffer from overlooking the importance of cooperation among traffic lights and lack of long sequence scheduling. To address this challenge, we formulate the control problem of multi-intersection traffic lights as a multi-agent Markov game problem. In response, we propose a Cooperative Adaptive Control Method (CACOM), a framework based on multi-agent reinforcement learning. CACOM integrates the mixing network and the options framework. Specifically, the mixing network enables cooperation among intersections, and the options framework provides the ability for intersections to make a long sequence scheduling. Besides, we designed a weight generator for the mixing network based on the traffic conditions at intersections, allowing the agents to adjust their weights adaptively during cooperation. Finally, we build a simulator including two real-world urban road networks for extensive evaluation. In contrast to the best baseline methods, our approach achieves an average waiting time reduction of around 24% and 42% for high-priority vehicles in two scenarios. Moreover, the waiting time for all vehicles is decreased by approximately 15% and 6%, respectively.**

***Index Terms*—Edge intelligence, traffic light control, cooperative multi-agent reinforcement learning, options framework, mixing network.**

Manuscript received 1 January 2024; revised 4 March 2024 and 14 May 2024; accepted 9 June 2024. Date of publication 21 June 2024; date of current version 31 December 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62272357 and Grant 62302326; in part by the Key Research and Development Program of Hubei Province under Grant 2022BAA052; and in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101130003. *(Corresponding authors: Bingyi Liu; Enshu Wang.)*

Haiyong Shi, Bingyi Liu, and Weizhen Han are with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China (e-mail: shihaiyong@whut.edu.cn; byliu@whut.edu.cn; hanweizhen@whut.edu.cn).

Enshu Wang and Libing Wu are with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430070, China (e-mail: wanges17@whu.edu.cn; wu@whu.edu.cn).

Jinfan Wang is with the Institute of Future Networks, Southern University of Science and Technology, Shenzhen 518055, China (e-mail: wangjf@sustech.edu.cn).

Shihong Cui is with the Marketing Department, Tianyijiaotong Technology Company Ltd., Suzhou 215000, China (e-mail: cuishihong202406@163.com).

Digital Object Identifier 10.1109/TCE.2024.3416822

I. Introduction

THE MATURITY of next-generation artificial intelligence and the gradual spread of 5G networks support the evolution of Consumer Electronics (CE) toward cooperation and intelligence. In recent years, CE promotes continuous innovation in edge intelligence technology, and the research community is focusing on scenarios where edge intelligence can be used in Intelligent Transportation Systems (ITS), including self-driving vehicles [1], [2], vehicular edge computing [3], [4], [5], unmanned aerial vehicles [6] and traffic light control [7], [8], etc. Due to the complexity of the interactive scene between individual agents in different edge systems, how to design a cooperative edge system with intelligence is an open problem.

As part of the edge node in the ITS, traffic lights can provide good support for creating a safe and efficient traveling experience. Existing traffic light control schemes on urban road environments can be grouped in several ways, including traditional methods based on rules and mathematical models [15], [16], [17], [18], [21] and methods based on Multi-Agent Reinforcement Learning (MARL) [19], [20], [23], [24], [25], [26]. Although rule-based and mathematical modeling methods for controlling traffic lights on urban roads (e.g., timing control methods and area adaptive control methods) can maintain reasonable traffic order, they lack flexibility and are not capable of autonomous learning. The MARL-based method opens new ideas for traffic light control, but there are still some shortcomings. In these methods, intersections are modeled as agents, and each agent independently explores the intersection traffic conditions and learns the control strategy autonomously, without considering the cooperation between the agents. In fact, vehicles may pass through a number of intersections while traveling in an urban road scene. The purpose of having traffic lights at each intersection is to serve vehicles. Therefore, the relationship among intersections is cooperative in nature, rather than competitive in nature. It is necessary to consider cooperation among intersections.

To achieve this, we propose a cooperative MARL framework. Firstly, the multi-intersection traffic light control problem is formulated as a multi-agent Markov game problem that views each intersection as an agent. Secondly, we propose an improved mixing network to prompt cooperation among agents. More specifically, inspired by QMIX [27], the mixing

1558-4127 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

network is used to generate a shared total state value for all agents. With a total state value sharing among agents, each agent is encouraged to optimize towards the same target. In the mixing network, we design a weight generator, which generates network weights to calculate the total state value based on the contribution at each intersection. Hence, during the training process, the agents can adaptively adjust the cooperation weights according to the contribution at each intersection. Meanwhile, to align the interests of each agent with the collective interests of all agents, we add monotonicity constraints in the process of calculating the weights of the mixing network.

Although a basic control strategy can be achieved through self-learning, this approach faces two fundamental challenges: (1) When the traffic flow coming from a direction is larger than usual, each intersection should keep a satisfactory phase for a while to mitigate the traffic pressure. (2) The long waiting time at the intersection hinders high-priority vehicles from accomplishing traveling tasks in time. To address the above challenges, the decision-making mechanism should maintain a proper phase for a long time for the vehicles near intersections to drive away quickly before high-priority vehicles arrive. Hence, the decision-making mechanism of each intersection should be empowered to have a long-sequence decision-making capability for phase control. To address the challenge of long-sequence scheduling, inspired by DAC [28], we employ the options framework, suitable for challenging traffic scenarios featured with multiple intersections and variable priority vehicles. The options framework is employed to provide long-sequence decision-making capability for intersections. Specifically, there are two layers in the options frameworks, the high-rank layer is equipped with a high-rank policy that can provide each agent with long-sequence decision-making capabilities, and the low-rank layer is equipped with a low-rank policy which is used to specify an appropriate phase.

In this paper, we consider improvements in the traveling efficiency of drivers in intelligent transportation by tackling the challenges caused by long-sequence scheduling for traffic flow and non-cooperative style. To achieve this, we propose a MARL framework, called *cooperative adaptive control with the options framework and the mixing network (CACOM)*, incorporating the options framework and the mixing network.

In summary, the key contributions of this work include the following 3 aspects.

- We propose a cooperative MARL framework that enables multiple intersections to adaptively learn traffic light control strategies from complex traffic flows and optimize the traffic efficiency of high-priority vehicles.
- We design a weight generator to promote cooperation among intersections by a dynamic contribution based on local traffic conditions.
- We build a multiple intersections-based simulator with various priority-level vehicles and conduct extensive experiments to evaluate CACOM. In comparison to the best baselines, CACOM significantly reduces the average waiting time for all vehicles.

II. Related Work

A. Edge Intelligence on Consumer Electronics

The progress of the Internet of Things (IoT) technology has promoted the prosperity of consumer electronics, and the large amount of data generated in the process of consumer electronics applications has promoted the development of edge computing [9]. In terms of vehicle collision detection, Ke et al. [2] proposed a lightweight edge intelligence framework for vehicle event monitoring and logging. Liu et al. [10] considered the synergistic impacts among vehicles and infrastructures in vehicular edge computing and proposed a scheme to balance the inference acceleration and reliability of deep neural networks. Dong et al. [11] proposed a relation-based network with adaptive top-k dependencies employed in consumer electronics for crowd counting. In terms of privacy protection, Li et al. [12] proposed a machine learning-based method in a distributed and resilient style to protect consumer privacy. Huang et al. [13] focused on the application of consumer electronics in agriculture and employed machine learning methods in lightweight consumer electronics assisting agricultural applications for higher quality products. In ITS, Liu and Liu [14] proposed a partition and offloading scheme in a distributed manner to speed up deep neural networks within vehicular edge computing tasks. In addition, the large amount of data generated in the transportation system has created conditions for the deployment of edge intelligence and consumer electronics, such as traffic light control systems.

B. Rule-Based Traffic Light Control Methods

With the rapid advancement in traffic light control systems, addressing their unique challenges in ITS has become a focal point of attention. Robert and Wagner [15] developed a traffic light control method that adjusts the green light time based on the vehicle's delay at intersections. Long et al. [16] proposed a self-organizing control model incorporating constraints on road capacity. They achieve a real-time adaptive traffic light duration control and take the minimal delay of road-expecting traffic flow as an optimization goal. Zhang et al. [17] considered pedestrian crossing risk and proposed a macroscopic model-based traffic light control strategy for pedestrian-vehicle mixed-flow scenarios. Wu et al. [18] presented an event-triggered control strategy based on distributed thresholds, enabling asynchronous updating of traffic signals. Besides, focusing on the design of the model, the joint control of connected vehicles and traffic lights has also begun to receive attention from scholars, such as the studies conducted by Dai et al. [19] and Guo et al. [20], etc. However, these methods often rely on local traffic conditions to design rules, and the rules proposed empirically may lack flexibility in dealing with dynamic traffic flows. Moreover, they might struggle to learn autonomously. To address these limitations, this paper adopts a reinforcement learning-based method for traffic light control, aiming to learn an optimal control strategy by autonomously exploring the environment.

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

C. MARL-Based Traffic Light Control Methods

Rule-based methods often fall short of expressing the hidden features of data. Consequently, an increasing number of researchers have turned to building deep reinforcement learning models to learn from traffic flow data and derive optimal control strategies. For instance, Kumar et al. [21] implemented a traffic light scheduling approach utilizing a deep reinforcement learning model. This method uses dynamic information on real-time traffic conditions, such as the quantity of vehicles and their velocities, as inputs to learn an optimal control strategy. Lin et al. [22] proposed a decentralized reinforcement learning method called TeDA-GCRL, which combines temporal differential aware and graph convolution methods. Jamil et al. [23] proposed integrating rewards obtained through various methods into the training process. Each reward has a separate network for learning the Q-value, contributing to the final action through a voting mechanism that interacts with the environment. Wu et al. [24] focused on real-time control of large-scale transportation networks and developed a distributed control framework for mixed urban and highway transportation networks. Liu et al. [25] proposed a traffic light timing system based on a Q-learning algorithm, which can effectively determine the duration of traffic lights by controlling the changes through the switching of time slices. Wang et al. [26] proposed a spatiotemporal MARL framework named STMARL, leveraging spatiotemporal information of dynamic traffic flow. In most of MARL-based approaches, each agent independently explores intersection traffic conditions and learns strategies autonomously, often without considering cooperation among agents.

In multi-agent systems, the exponential growth of action spaces with the quantity of agents poses challenges, making it difficult to learn a joint policy directly from the joint action space. To address this, researchers have explored the Centralized Training and Decentralized Execution (CTDE) algorithmic framework. Notable works in this area include MADDPG [29], COMA [30], MAPPO [31], VDN [32], QMIX [27], and WQMIX [33]. MADDPG, COMA, and MAPPO are all training algorithms based on policy gradients. MADDPG focuses on learning deterministic policies for continuous actions, while COMA and MAPPO learn stochastic policies for discrete actions. In methods that emphasize cooperation among agents, multiple agents often share a joint value function during training. VDN, for instance, sums the Q-values of all agents to create a shared Q-value. This simple summation method cannot distinguish the contributions of individual agents. QMIX and WQMIX compute the shared Q-value using a mixing network. However, QMIX-based methods typically use global observations to generate weights and do not consider the influence of individual local observation states on cooperation. In this paper, a weight generator based on the contribution at each intersection is designed for the mixing network. This design allows agents to adjust weights adaptively according to their contributions at each intersection during cooperation.

III. Problem Modeling

In a real-world transportation system, addressing the control of traffic lights and the scheduling of traffic flow at multiple intersections naturally becomes a sequential decision-making problem. Traditionally, one approach to solving such a problem is to treat the transportation system as a centralized control system, making decisions centrally for all intersections [34]. However, this centralized approach encounters challenges related to the exponential growth of state and action spaces, given the large number of intersections in a typical real-world transportation system. Additionally, to alleviate computational burdens and maintain real-time responsiveness, intersections are often treated as edge nodes within the transportation system. Consequently, we adopt a decentralized MARL framework. The traffic lights control problem is formulated as a cooperative multi-agent Markov game problem.

The specific definitions of the elements in the Markov game are as follows:

- Agent: We treat each intersection in the transportation system as an agent in our CACOM, and use $i \in I \equiv \{1, \dots N\}$ to denote an agent.
- State: At every time step t, the state s_t is composed of four elements, including the present phases of all agents i, the quantity of agents, the location details of all vehicles, and the priority details of all vehicles. The state space S encompasses all potential states.
- Observation: At every time step t, each agent $i \in I$ receives a local observation $\omega_{i,t}$ containing the current phase of agent i and the traffic conditions near the intersection. The observation space Ω encompasses all possible observations.
- Action: Each agent i executes an action $u_{i,t}$ after receiving an observation at each time step t. The action space of the agent i is denoted as U_i and the joint action space of all agents is denoted as $U = U_1 \times \cdots \times U_N$.
- High-rank Policy: At each time step t, each high-rank layer takes a previous option $o_{i,t-1}$ and a current observation $\omega_{i,t}$ as inputs. The high-rank policy $\pi^H_{i,t}$ for agent i selects each option $o_{i,t} \in O$ based on a probability function $\pi^H_{i,t}(o_{i,t}|o_{i,t-1}, \omega_{i,t})$.
- Low-rank Policy: Each low-rank layer takes an observation $\omega_{i,t}$ and an option $o_{i,t}$ as input. Then, the agent take each action $u_{i,t} \in A$ according to a probability function $\pi^L_{i,t}(u_{i,t}|\omega_{i,t}, o_{i,t})$.
- Transition Probability: The transition function takes the state s_t, the joint option $\boldsymbol{o_\tau} = (o_{1,t}, \dots, o_{N,t})$ of all agents, as well as the joint action $\boldsymbol{u_\tau} = (u_{i,t}, \dots, u_{N,t})$ of all agents, the current state s_t transits to the next s_{t+1} with probability $P(s_{t+1}|s_t, \boldsymbol{o_\tau}, \boldsymbol{u_\tau})$.
- Reward: each agent i will receive a team reward $r_{i,t}(s, \boldsymbol{u}) = \frac{\eta}{|M_t|}\sum_i\sum_m \lambda_m \cdot (w_{i,m,t-1} - w_{i,m,t})$ after executing actions at time step t, where $w_{i,m,t}$ denotes the cumulative waiting time of vehicle m in the vicinity of agent i, η is a constant, λ_m represents the traveling priority of vehicle m, and M_t represents all vehicles.

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

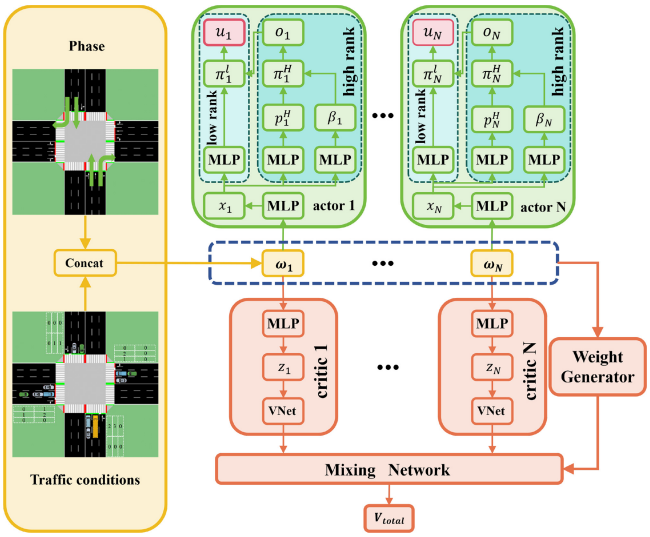

Fig. 1. The overall framework of the proposed method.

In the Markov game, the agent determines a final policy and executes an action based on the observed information. Specifically, the agent takes the current traffic conditions and phase information as observed information and inputs it into the decision model. In the decision model, the high-rank module first determines the high-rank policy based on observed information and generates a set of options. The high-rank module determines an option based on the high-rank policy. Next, the option information and the observed information are input into the low-rank module. The low-rank module determines low-rank policy based on observed information and then outputs phase information to control traffic lights.

IV. Cooperative Multi-Agent Reinforcement Learning Framework

A. Framework Overview

The overall framework of the proposed method is depicted in Fig. 1. Each agent is comprised of an actor module and a critic module, with the actor module incorporating the options framework. The high-rank layer of the options framework encompasses options that enable the agent to make long-sequence decisions, while the low-rank layer facilitates phase switching for the agent. The inputs for an agent include the local observation and the current phase. In the critic module, a VNet module is employed to output a state value for the agent. Subsequently, the values from every critic are input into the mixing network for calculating the total state value. The following two subsections elaborate on the design of the actor module and critic module, as well as the mixing network module.

B. Design of Actor Module and Critic Module

As depicted in Fig. 1, each agent comprises an actor module and a critic module. At every time step t, the observation $\omega_{i,t}$ includes the current traffic conditions and the previous action $u_{i,t-1}$ of agent i, which is then input into a Multilayer Perception (MLP) layer for observation representation $x_{i,t}$. Within the high-rank layer, we use an MLP layer to extract a feature vector from $x_{i,t}$, and the feature vector is processed through a softmax layer for $p^H_{i,t}$ which represents a probability distribution. Similarly, we use an MLP layer to extract a feature vector from $x_{i,t}$, then a ReLU activation takes the feature vector as input to determine the termination condition $\beta o_{i,t}$. The high-rank policy $\pi^H_{i,t}$ is defined by $p^H_{i,t}$ as well as $\beta_{o_{i,t-1}}$, as expressed in the following equation (1).

$$\pi^H_{i,t}(o_{i,t}|o_{i,t-1},\omega_{i,t}) = \beta_{o_{i,t-1}} p^H_{i,t}(o_{i,t}|\omega_{i,t}) + (1-\beta_{o_{i,t-1}})\cdot \mathbb{I}_{o_{i,t-1}=o_{i,t}}, \quad (1)$$

where the $\mathbb{I}$ represents an indicator function. It's noteworthy that when the agent i samples the first option, no previous option exists. Hence, $\mathbb{I}_{o_{i,t-1}=o_{i,t}}$ is equivalent to 0. Subsequently, an option $o_{i,t}$ is sampled according to the policy $\pi^H_{i,t}$. Within the low-rank layer, the observation representation $x_{i,t}$ undergoes a set of MLPs, generating probability distributions for all options. During the decision-making process, the actor first takes an option $o_{i,t}$ by the policy $\pi^H_{i,t}$. Then, agent i will specify a probability distribution from $o_{i,t}$ as the low-rank policy. Thereafter, agent i will take an action $u_{i,t}$ according to low-rank policy $\pi^L_{i,t}$. Each agent i's actor regards its observation representation $x_{i,t}$ as input. When agent i interacts with the environment, it will receive an immediate reward $r_{i,t}$ after taking an action $u_{i,t}$.

In the critic module, agent i first uses an MLP layer to extract its observation representation $z_{i,t}$, as is shown in Fig. 1. Then, observation $z_{i,t}$ is fed to a VNet for state value $v^L_{i,t}$ to evaluate the low-rank layer. Referring to the DAC framework [28], the state value $v^L_{i,t}$ can also be used to evaluate the high-rank layer while the state value function is used as the critic. Therefore, the calculation of the state value $v^H_{i,t}$ is expressed in equation (2).

$$v^H_i(o_{i,t},\omega_{i,t+1}) = \sum_{o_{i,t+1}} \pi^H_i(o_{i,t+1}) v^L_i(\omega_{i,t+1}, o_{i,t+1}). \quad (2)$$

C. Mixing Network Module

Inspired by QMIX [27], we propose an improved mixing network and designed a network weight generator based on the contribution at each intersection. Different from the traditional attention mechanism, we add monotonicity constraints to the weight generator to ensure that the changes in individual state values are consistent with the collective, promoting cooperation among agents. The structure of the weight generator is depicted in Fig. 2, At each time step t, the input to the weight generator consists of the observation $\omega_{i,t}$ of all agents i. The observation $\omega_{i,t}$ of all agents are concated together and fed into an MLP layer to get a state representation s_t. Before calculating the contribution value, the state s_t and the observation $\omega_{i,t}$ of the agent are first mapped into a query vector $\boldsymbol{q_s}$, a key vector $\boldsymbol{k_i}$ and a value vector $\boldsymbol{g_i}$, respectively, which are calculated as shown in equation (3), equation (4) and equation (5).

$$\boldsymbol{q_s} = W_q(s_t;\rho_q), \quad (3)$$

$$\boldsymbol{k_i} = W_k(\omega_{i,t};\rho_k), \quad (4)$$

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

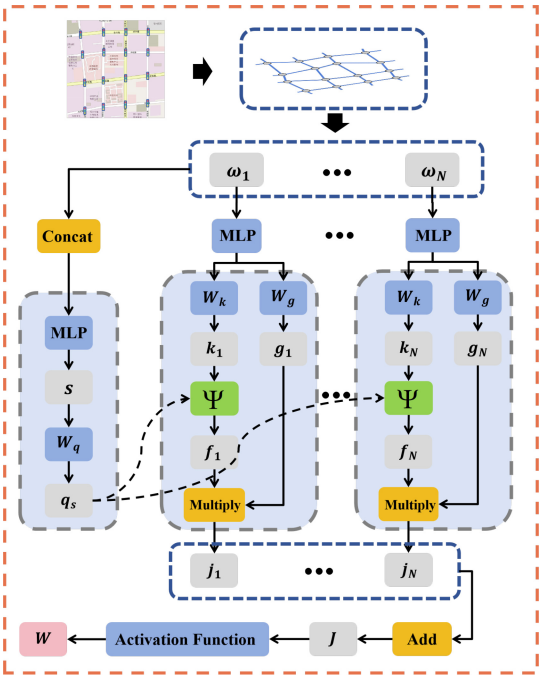

Fig. 2. The weight generator.

$$\boldsymbol{g_i} = W_g(\omega_{i,t}; \rho_g), \tag{5}$$

where W_q is a query generator parameterized by ρ_q, W_k is a key generator parameterized by ρ_k, and W_g is a value generator parameterized by ρ_g. Then, the query vector $\boldsymbol{q_s}$ of the global environment is further sent to the agent i. To determine the weights of the observation, the query vector $\boldsymbol{q_s}$ of the state s_t and the key vector $\boldsymbol{k_i}$ of the observation $\omega_{i,t}$ are fed into the similarity function Ψ to compute the matching scores between the two vectors as shown in equation (6) and equation (7).

$$f_i = \Psi(\boldsymbol{q_s}, \boldsymbol{k_i}), \tag{6}$$

$$\Psi(\boldsymbol{q_s}, \boldsymbol{k_i}) = \frac{\boldsymbol{q_s^T k_i}}{\sqrt{K}}, \tag{7}$$

where Ψ is a similarity function to measure the similarity between two vectors. Then, the value vectors J of all agents are calculated based on the matching scores f_i and the agent i's value vectors $\boldsymbol{g_i}$, as shown in equation (8).

$$J = \sum_{i=1}^{N} f_i \boldsymbol{g_i}, \tag{8}$$

Finally, J is fed into an absolute value activation function to obtain the non-negative weights of the mixing network.

D. Training Algorithms

In this paper, we bring the idea of Proximal Policy Optimization (PPO) [35] to the training process of CACOM. In addition, the specific training process of CACOM is shown in Algorithm 1.

At the end of time step t, the agent i will receive a reward $r_{i,t}$ when it takes an action $u_{i,t}$ according to the low-rank policy $\pi_{i,t}^L$ and interacts with the environment. Meanwhile, the VNet of the agent i takes the observation representation as input and outputs a state value the low-rank critic which is denoted as $V_{i,t}^L = V_{\Phi_i}^L(z_{i,t}; \Phi_i)$, where VNet is parameterized by Φ_i. Then, a replay buffer D is used to store tuple $d_t = (U_t, O_t, \pi_t^L, \pi_t^H, V_t^L, R_t)$, where U_t, O_t, π_t^L, π_t^H, V_t^L, R_t denotes the action set, options, low-rank policies, high-rank policies, low-rank state values, as well as rewards, respectively.

Algorithm 1 Training Algorithm of CACOM

initialize learning rate α, discount factor γ, explore probability ϵ, episodes Ep, simulation steps T, replay buffer B, model parameters θ, ϕ, φ, φ^-;
foreach episode from 1 to Ep
  foreach step from 1 to T
    each agent i samples $o_{i,t}$ using $\pi_{i,t}^H$;
    each agent i samples $u_{i,t}$ using $\pi_{i,t}^L$;
    each agent i calculates the state value $v_{i,t}^L$;
    each agent i executes an action $u_{i,t}$;
    each agent i receives a reward $r_{i,t}$;
    store $(u_{i,t}, o_{i,t}, \pi_{i,t}^L, v_{i,t}^L, r_{i,t})$ to D;
  calculate $L^{V_{total}}$;
  foreach agent i
    sample a random batch of experiences from D;
    calculate $v_{i,t}^H$ using equation (2);
    update the low-rank critic using equation (9) and equation (15);
    update the low-rank actor using equation (11) and equation (15);
    update the high-rank critic using equation (12) and equation (15);
    update the high-rank actor using equation (14) and equation (15);
    update the Mixing Network using equation (15);
  $\varphi^- \leftarrow \varphi$;

In the process of updating model parameters, we initiate the optimization of the low-rank policy π_t^L, maintaining the parameters of the high-rank policy π_t^H as well as the parameters of termination function β_{o_i} fixed. We use a loss function to minimize the gap between the target value and the predicted value of agent i's low-rank critic in the low-rank layer as expressed in equation (9).

$$L(\phi_i) = \mathbb{E}_{d_{i,t} \sim D}\left[\left(r_{i,t} + \gamma V_{i,t+1}^L - V_{i,t}^L\right)^2\right], \tag{9}$$

where the critic network is parameterized by ϕ_i. In this paper, an importance sampling method is used to enhance the data efficiency. Consequently, an objective function designed for conservative policy iteration (CPI) is expressed in equation (10).

$$\begin{aligned} L^{CPI}(\theta_i) &= \mathbb{E}\left[\frac{\pi_{i,t}^L(u_{i,t}|\omega_{i,t}, o_{i,t})}{\pi_{i,t}^{L,old}(u_{i,t}|\omega_{i,t}, o_{i,t})} A_{i,t}^L\right] \\ &= \mathbb{E}\left[\mu_{i,t}^L(\theta_i) A_{i,t}^L\right], \end{aligned} \tag{10}$$

where θ_i and $\pi_{i,t}^{L,old}$ denotes the learning parameters of actor-critic network and agent i's old policy parameterized by θ_i^{old} respectively, within the low-rank layer. An advantage function $A_{i,t}^L = r_{i,t} + \gamma V_{i,t+1}^L - V_{i,t}^L$ is defined in the low-rank layer.

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

Subsequently, to mitigate the problem of too large an update of the policy parameters in $L^{CPI}(\theta_i)$, we employ a clipped objective function L^{CPI} which is expressed in equation (11).

$$L^{CLIP(\theta_i)} = \mathbb{E}\Big[\min\Big(clip\big(\mu_{i,t}^{L}(\theta_i), 1-\sigma, 1+\sigma\big)A_{i,t}^{L}, \mu_{i,t}^{L}(\theta_i)A_{i,t}^{L}\Big)\Big], \quad (11)$$

where σ represents a parameter that serves as the clipping ratio.

Then, the parameters of high-rank policy π_i^H and the parameters of termination function β_{o_i} are implicitly optimized while the parameters of low-rank policies π_i^L are fixed. Within the high-rank layer, we use a loss function to minimize the gap between the target value and the predicted value of agent i's high-rank critic, which is expressed in the following equation (12).

$$L(\phi_i) = \mathbb{E}_{d_{i,t}\sim D}\Big[\big(r_{i,t} + \gamma V_{i,t+1}^{H} - V_{i,t}^{H}\big)^2\Big], \quad (12)$$

where $V_{i,t}^{H}$ represents the state value, calculated as $V_{i,t}^{H} = \sum_{\omega_{i,t}} \pi_{i,t}^{H} V_{i,t}^{L}$. Same as the low-rank layer, a CPI objective function is used to train the high-rank layer, which is expressed in equation (13).

$$\begin{aligned} L^{CPI}(\theta_i) &= \mathbb{E}\left[\frac{\pi_{i,t}^{H}\big(u_{i,t}|\omega_{i,t}, o_{i,t}\big)}{\pi_{i,t}^{H,old}\big(u_{i,t}|\omega_{i,t}, o_{i,t}\big)} A_{i,t}^{H}\right] \\ &= \mathbb{E}\big[\big(\mu_{i,t}^{H}(\theta_i)A_{i,t}^{H}\big)\big], \end{aligned} \quad (13)$$

where θ_i and $\pi_{i,t}^{H,old}$ represents the learning parameters of actor network and agent i's old policy parameterized by $\pi_{i,t}^{H,old}$ respectively, within the high-rank layer. An advantage function $A_{i,t}^{H} = r_{i,t} + \gamma V_{i,t+1}^{H} - V_{i,t}^{H}$ is defined within the high-rank layer. Likewise, we employ a clipped objective function L^{CLIP} to the training process of high-rank actor for agent i as formulated in equation (14).

$$L^{CLIP(\theta_i)} = \mathbb{E}\Big[\min\Big(clip\big(\mu_{i,t}^{H}(\theta_i), 1-\sigma, 1+\sigma\big)A_{i,t}^{H}, \mu_{i,t}^{H}(\theta_i)A_{i,t}^{H}\Big)\Big], \quad (14)$$

where σ represents a parameter that serves as the clipping ratio. Subsequently, the calculation of the multi-agent cooperation loss is performed as illustrated in equation (15).

$$L^{V_{total}}(\varphi) = \frac{1}{T}\sum_{t=1}^{T}\Big(y_t^{total} - V_{total}(V, s_t; \varphi)\Big)^2, \quad (15)$$

$$y_t^{total} = r + \gamma V_{total}\big(V', s_t'; \varphi^-\big), \quad (16)$$

where T, s_t, s_t', V, V', φ, φ^-, $V_{total}(\cdot)$ denote simulation steps, the current state, the next state, the set of the agent's current state value, the set of the agent's next state value, the mixing network parameters, the target network parameters of the mixing network, and the mixing network, respectively.

V. Experimental Results and Analysis

The simulation experiments in this paper are conducted on a computer running an Ubuntu system. The system specifications include a Supermicro X13SAE-F motherboard, 128GB of memory, and a 13th Gen Intel(R) Core(TM) i9-13900K CPU with a maximum frequency of 5.8GHz. The GPU employed is an NVIDIA GeForce RTX 3090 with 24GB of video memory. The experiment utilizes software tools such as VS Code, miniconda3, and SUMO-1.8.0. The model, implemented in this paper, is realized using the PyTorch deep learning framework, with Python version 3.7 and PyTorch version 1.13.1.

(a) Daxing District, Beijing, China. (b) Suzhou Industrial Park, Suzhou, China.

Fig. 3. Simulated intersections diagrams.

A. Simulation Experiment Setup

We establish a multi-intersection simulation environment using SUMO (Simulation of Urban Mobility) [36]. This simulation environment replicates the arrangement of intersections in both Suzhou Industrial Park and a segment of Daxing District in Beijing. The regional map utilized in the simulation is illustrated in Fig. 3. Specifically, Fig. 3(a) illustrates the road network map for a region in Daxing District, Beijing, while Fig. 3(b) illustrates the road network map for a region in Suzhou Industrial Park. Notably, the intersections in the road network of Fig. 3(a) are more densely distributed. In this study, we employ 16 intersections to simulate traffic flow conditions, and the locations of the signal lights in Fig. 2 correspond to the intersections used in the simulation environment. The information integrated into the simulation environment encompasses vehicle arrival times and vehicle priorities. Notably, although we evaluate in typical four-way intersection scenarios, the proposed method is not constrained by any intersection type. By customizing the actions for different intersection types, we can apply them to heterogeneous scenarios including multiple types of intersections.

In this study, the simulation environment comprises 16 intersections, and the signal light at each intersection operates through four phases to regulate traffic flow. Each phase, denoting a combination of red and green lights, has a duration of ten seconds. The visual representation of each phase in the real environment is illustrated in Fig. 4. Among these phases, Phase 1 and Phase 4 govern the vehicle's straight and right turn movements in the east-west and north-south directions, respectively. Conversely, Phase 2 and Phase 3 regulate the east-west and north-south directions for the vehicle's left turn movements, respectively.

The specific parameter configurations for the simulation environment are outlined in Table I. Each round of training encompasses 50 simulation steps. To ensure traffic safety, a 3-second yellow light is incorporated after each phase, allowing the remaining vehicles from the previous phase to clear

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

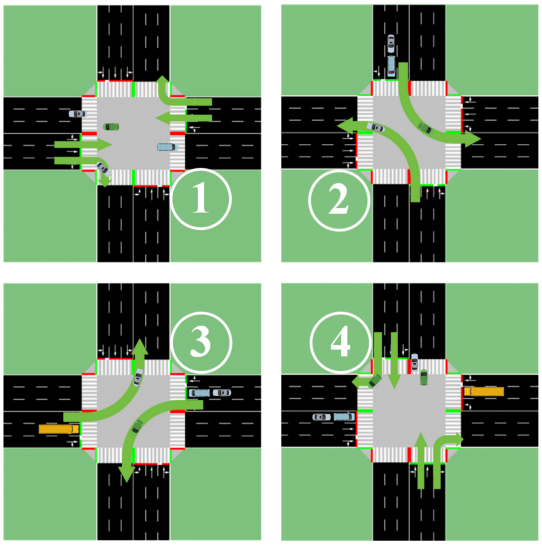

Fig. 4. Four phase diagrams.

TABLE I
SIMULATION PARAMETERS

Parameters	values
Actor learning rate	$1e^{-4}$
Critic learning rate	$5e^{-4}$
Mixing Network learning rate	$7e^{-4}$
simulation steps	50
Phase Duration	10s
Yellow time duration	3s
Number of phases	4
Number of intersections	16
Length for a traffic condition cell	10m
Vehicle arrival rate	1.2 vehicles/s
Arrival rate for high priority vehicles	0.12 vehicles/s
The max proportion for high-priority vehicles	10%

the intersection area. Information for each lane is composed of multiple cells, each recording the priority of internal vehicles. Collectively, these cells constitute the intersection vehicle priority matrix, serving as a local observation. The vehicle arrival rates at intersections are set to 1.2 vehicles per second, with a corresponding high-priority vehicle arrival rate of 0.12 vehicles per second. In the simulation scenario, the proportion of high-priority vehicles is random with a maximum not exceeding 10% at each time step, reference to [8], [37]. Besides, with scheduling by each intersection, traffic flow in each intersection will be impacted by intersections with implicit relations. Hence, each agent's traffic condition is complex and highly dynamic.

B. Baselines and Evaluation Metrics

1) *Baselines*: The proposed method in this paper will be compared with several baseline methods, including 2 classical cooperative reinforcement learning methods.

- Traffic light control method based on QMIX: The QMIX framework was applied in [27] within a multi-agent interactive environment. To be more specific, within this cooperative framework, each agent utilizes its local observation as an input and the global state serves as input for the mixing network. Throughout our experiments, we adhere to the original parameters for both the critic and the actor.
- Traffic light control method based on WQMIX: The WQMIX framework is an improved method based on QMIX and has been used in [33]. More concretely, the WQMIX framework modified the weighting function to fit better the curve of the function that generates the total value. Likewise, we use the original parameters for both the critic and the actor.

2) *Evaluation Metrics*: In our experiments, we employ waiting time as the primary evaluation metric for the proposed method. Specifically, we calculate the average waiting time (WT) of vehicles to assess the overall performance of the proposed method in the global environment. Additionally, we consider the average waiting time of high-priority vehicles (HWT) near the intersection to evaluate the efficiency of the proposed method when high-priority vehicles travel or commute in this environment. Our rationale is based on the expectation that vehicles with higher priority should experience shorter waiting times. Consequently, we propose the concept of weighted waiting time (WWT) for high-priority vehicles. The WWT as well as the HWT serve as reference metrics for each other. The above performance metrics are all measured in seconds. In this paper, waiting time refers to the cumulative waiting time of vehicles at each intersection for a single vehicle from origin to destination in the simulation scenario. Furthermore, we utilize the ratio of HWT to WT to evaluate the impact of high-priority vehicles on the waiting times of all vehicles.

C. Performance Evaluation on the Industrial Park of Suzhou

The method proposed in this paper is compared with the baseline method QMIX and its improved version WQMIX in a simulation experiment based on a regional road network in Suzhou Industrial Park. The evaluation results of the proposed method and the baseline methods are summarized in Table II. From the results presented in Table II, it is evident that the QMIX method exhibits the lowest performance, with a ratio of the HWT to the WT (RHWT) at 1.08. With the QMIX method, the average traffic efficiency of all vehicles is higher than that of high-priority vehicles at intersections. WQMIX achieves a substantial reduction in the waiting time for both HWT and WT, resulting in an RHWT of 0.94. Comparatively, our proposed method outperforms both QMIX and WQMIX in terms of both HWT and WT, yielding an RHWT value of 0.84. In comparison to the WQMIX method, our approach reduces the duration of HWT by approximately 24% and the duration of WT by around 15%. These results indicate that the combination of our designed mixing network and the options framework significantly has better performance in this scenario.

To analyze the impact of the quantity of options on the CACOM, we conduct experiments with varying numbers of options. The experimental results are depicted in Fig. 5, where

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

TABLE II
PERFORMANCE COMPARISON ON INDUSTRIAL PARK OF SUZHOU

Method	HWT	WT	WWT	RHWT
QMIX	234.86	215.52	1220.07	1.08
WQMIX	105.17	111.17	603.61	0.94
CACOM	79.92	94.31	493.84	0.84

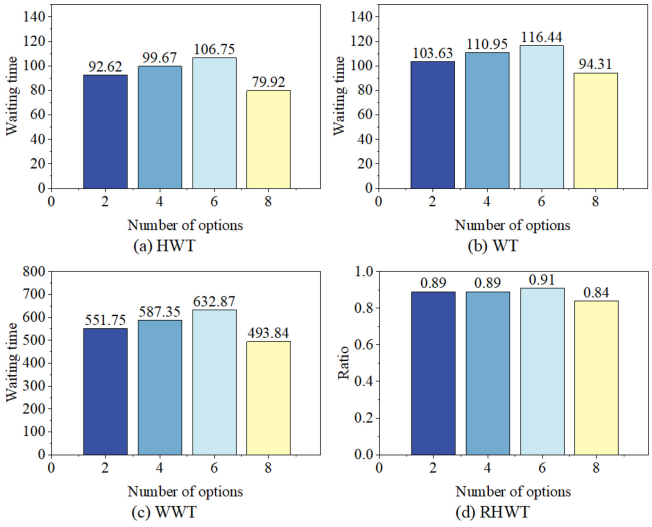

Fig. 5. Evaluation results of different options on the Industrial Park of Suzhou.

TABLE III
PERFORMANCE COMPARISON ON DAXING DISTRICT OF BEIJING

Method	HWT	WT	WWT	RHWT
QMIX	282.47	194.98	390.03	1.44
WQMIX	216.54	99.03	198.73	2.18
CACOM	124.25	92.17	184.26	1.34

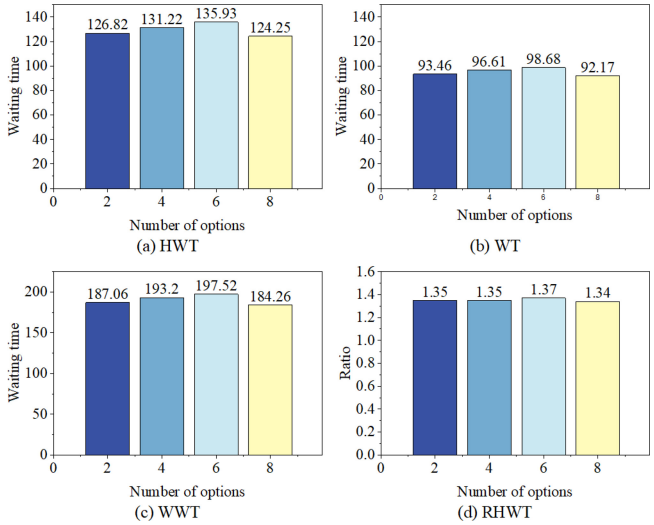

Fig. 6. Evaluation results of different options on Daxing District of Beijing.

Fig. 5(a) illustrates the effect of different numbers of options on the performance of HWT, Fig. 5(b) shows the impact on WT, Fig. 5(c) demonstrates the effect on WWT, and Fig. 5(d) displays the variation of RHWT with the quantity of options. In Fig. 5, results for a low number of options are excluded because the performance loss due to too few options is not informative for studying the effect of the quantity of options, especially in continuous scenarios where this effect is predictable. Therefore, the initial number of options is set to 2. Upon inspection of Fig. 5, it becomes evident that an increase in the quantity of options does not consistently lead to improved traffic regulation effectiveness. Notably, when the quantity of options increases from 4 to 6, the average waiting time of vehicles increases. However, as the quantity of options continues to rise, a significant enhancement in the regulation effect on traffic flow is observed when the quantity of options is 8, resulting in a further reduction in RHWT. Consequently, the default number of options in this paper for this scenario is 8.

D. Performance Evaluation on Daxing District of Beijing

Table III provides a comparison with baselines in a simulation experiment based on a regional road network in Daxing District, Beijing. While the QMIX method is close to CACOM in terms of RHWT, there remains a significant gap in the two metrics of HWT for high-priority vehicles and WT for all vehicles. Consequently, at intersections controlled by the QMIX method, the average efficiency of high-priority vehicles, as well as the average efficiency of all vehicles, is lower than that achieved by the other methods. WQMIX also exhibits notable improvements. Compared with QMIX, WQMIX achieves a reduction of approximately 23% in the duration of HWT and a reduction of about 49% in the duration of WT. Our proposed method outperforms both QMIX and WQMIX in terms of both HWT and WT, yielding an RHWT value of 1.34. In comparison to the WQMIX method, our approach reduces the duration of HWT by approximately 42% and the duration of WT by about 6%. Compared to the results of the Suzhou Industrial Park scenario, the duration of HWT is reduced even further in the Daxing District scenario. This result suggests that CACOM can also exhibit good performance in denser intersection scenarios.

We investigate the effect of the quantity of options on CACOM in the Beijing Daxing District scenario. Experiments are done with a different number of options respectively and the results are shown in Fig. 6. Among them, Fig. 6(a) displays the effect of a different number of options on the performance of HWT, Fig. 6(b) displays the effect of a different number of options on the performance of WT, Fig. 6(c) displays the effect of a different number of options on the WWT, and Fig. 6(d) displays the variation of RHWT with the quantity of options. As with the previously mentioned test method, the initial number of options is set to 2. It can be visualized from Fig. 6 that when the quantity of options increases from 4 to 6, the average waiting time increases instead. As we continue to increase the quantity of options to 8, the effect of CACOM on the regulation of traffic flow appears to be significantly improved, and the RHWT is further reduced. Combining the results of the two experimental scenarios, this paper's default number of options is 8.

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

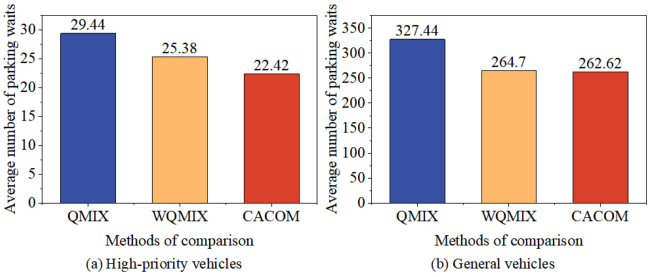

Fig. 7. The average number of vehicles stop at intersections in the Industrial Park of Suzhou.

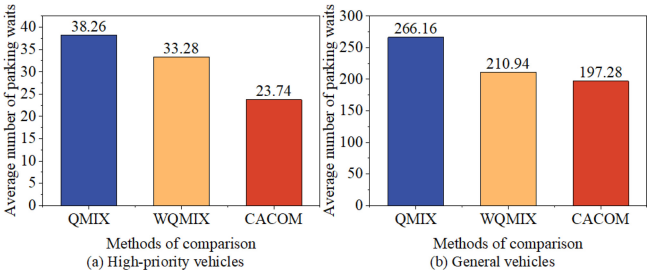

Fig. 8. The average number of vehicles stop at intersections in the Daxing District of Beijing.

E. Case Study

In Fig. 7(a) and Fig. 7(b), the average number of stops at intersections per time step in the simulation environment for high-priority vehicles and normal vehicles in the Suzhou Industrial Park scenario is illustrated, respectively. During the process of road travel, vehicle deceleration or stopping can negatively impact overall passing efficiency. Therefore, minimizing the quantity of vehicle stops at intersections is crucial to ensure efficient traffic flow. To assess the impact of different methods on the quantity of vehicles waiting to stop at intersections, we analyzed both the baseline method and the proposed method. In this comparison, we focus on the average number of stops at intersections for high-priority vehicles and normal vehicles, as depicted in Fig. 7(a). Notably, our method outperforms both the QMIX method and the WQMIX method. As indicated by the bar chart in joint Fig. 7(b), our proposed method achieves a reduction in the quantity of high-priority vehicles stopping and waiting at intersections without compromising the efficiency of general vehicles when compared to the baseline method.

Under the same parameters of the simulation environment, denser intersections result in more vehicles stopping and waiting at the intersection. Fig. 8(a) illustrates that the difference between the average number of high-priority vehicles stopping and waiting at intersections and the baseline method is more pronounced for our proposed method in the scenario of Daxing District, Beijing. This contrast with the quantity of high-priority vehicles stopping and waiting at intersections does not show a significant gap in Fig. 7(a) when compared with the WQMIX method. This suggests that the dense area of the intersection better reflects the effectiveness of the multi-agent cooperation mechanism. In terms of general vehicle traffic efficiency, Fig. 7(b) indicates that our proposed method does not exhibit a significant gap compared to the WQMIX method under sparse intersection conditions. However, as shown in the bar chart in Fig. 8(b), the advantage of our proposed method over the WQMIX method expands. This indicates that in the case of dense intersections, our proposed CACOM method outperforms the WQMIX method. Therefore, the mixing network we design can better integrate the effects of the intersection environment when constraining multi-intersection cooperation.

As is shown in Fig. 9, there is a visualized example of a high-priority vehicle traveling through an intersection. In time step t_1, before a high-priority vehicle approaches the intersection, the phase is switched so that ordinary vehicles around the intersection can travel away. In time step t_2, when a phase duration is finished, a 3-second yellow light status follows. In time step t_3, The next phase is switched to a status that vehicles in the lane of the high-priority vehicle can travel away. Hence, the intersection can achieve a continuous scheduling in time to a certain direction of the road. Fig. 10 illustrates learned policies in a simulation round, in which we use four color blocks to signify Phase 1, Phase 2, Phase 3, and Phase 4, respectively. The continuous color blocks represent that the proposed method with a dual-structure framework can execute long-sequence scheduling.

VI. Discussion

In this section, we first discuss the convergence and complexity of CACOM. Then, we discuss how would the parameters of CACOM be updated and managed across the distributed system.

A. Convergence

During the training process, we consider that the proposed CACOM framework achieves convergence when the evaluation metrics fluctuate around a certain value. In this paper, we consider that the proposed CACOM framework achieves convergence during training when the average waiting time remains small and does not decrease after 1000 episodes. The convergence curve of the proposed method for the average waiting time in two multi-intersection urban road scenes is shown in Fig. 11. Among them, the orange curve represents the average waiting time of high-priority vehicles, the blue curve represents the average waiting time of all vehicles, and the red area represents the overlapping part. From Fig. 11, it can be seen that in dense intersection scenes, the average waiting time of high-priority vehicles is slightly longer than the average waiting time of all vehicles. In sparse intersection scenarios, the waiting time of high-priority vehicles is smaller than the average traffic efficiency of all vehicles. In addition, recent references [38] have demonstrated that the dual structure algorithm framework used in this paper can converge to a minimum value.

B. Complexity

In the actor module, the observation is first input into an MLP for the observation representation, with a complexity of $O(ND_\omega D_x)$, where D_x represents the local observation

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

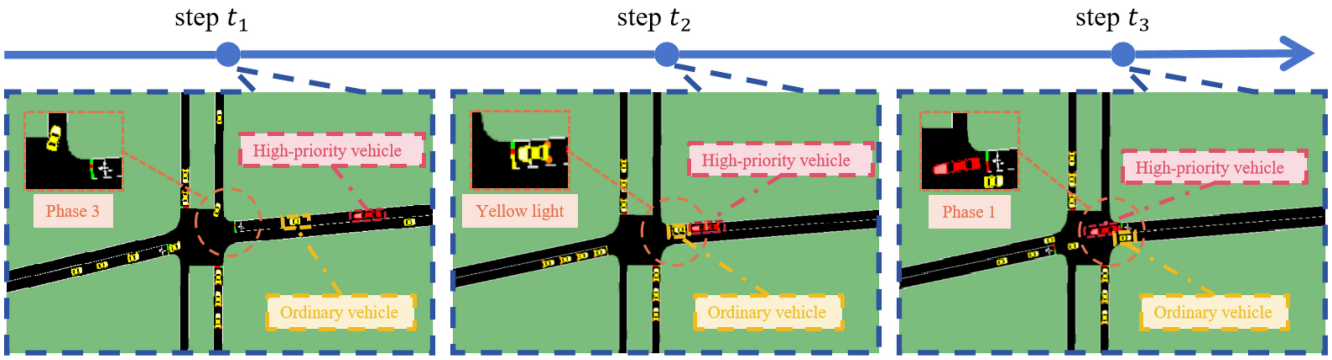

Fig. 9. A visualized example of high-priority vehicle travel through an intersection.

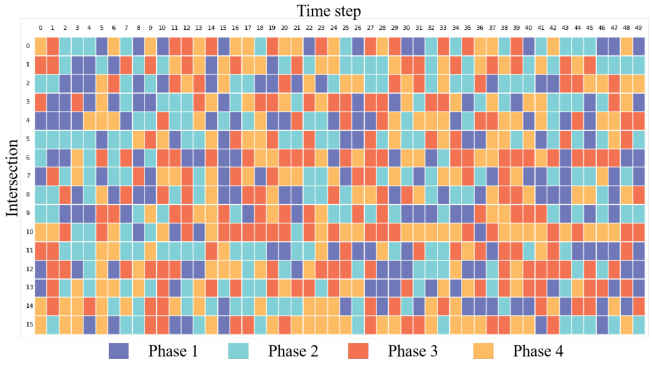

Fig. 10. Illustration of learned policies in a simulation round.

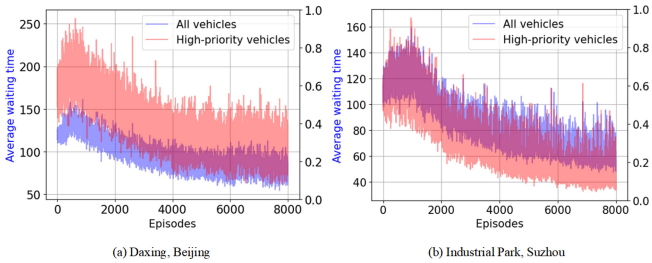

Fig. 11. The average waiting time during the training process.

dimension, D_x represents the output dimension of the MLP, N represents the number of agents. Next, calculate the options and termination conditions, with a complexity of $O(ND_xD_o)$ and $O(ND_x)$, where D_o represents the number of options. Since each option provides a policy set for the high-rank module to select, the complexity of obtaining the final policy and determining the final action is $O(ND_o|U|)$, where $|U|$ represents the action space. In the critic module, the local observation is first input into an MLP for the observation representation, with a complexity of $O(ND_\omega D_s)$. Next, the observation representation is fed into VNet for the state value, with a complexity of $O(ND_sD_v)$. Among them, D_s, and D_v represent the output dimension of the MLP and VNet layer, respectively. In the mixing network module, the weight generator's complexity includes calculating the query vector for the state, key vectors, and value vectors for observations. The complexity of calculating the query vector is $O(D_sD_q)$, where D_s represents the dimension of the state, D_q represents the dimension of the query vector. The complexity of calculating key vectors and value vectors are $O(ND_\omega D_k)$ and $O(ND_\omega D_g)$, where D_k and D_g represent the dimensions of the key vector and value vector. The complexity of calculating network weights is $O(ND_qD_k + ND_g)$. In summary, the complexity of the actor module O_a, the complexity of critical module O_c, and the complexity of mixing network module O_m are expressed by equations (17), (18), and (19), respectively.

$$O_a = O(ND_\omega D_x) + O(ND_xD_o) + O(ND_x) + O(ND_o|U|), \tag{17}$$

$$O_c = O(ND_\omega D_s) + O(ND_sD_v), \tag{18}$$

$$O_m = O\big(D_sD_q\big) + O(ND_\omega D_k) + O\big(ND_\omega D_g\big) + O\big(ND_qD_k + ND_g\big). \tag{19}$$

C. Implementation in the Distributed System

This paper adopts a multi-agent reinforcement learning framework with centralized training and decentralized execution. In this framework, agents cooperate to learn during training and act independently during execution. During centralized training, agents share value functions and rewards with the central controller or commentator. This central component can evaluate collective actions based on data collected from various agents. Then, the central controller guides the agent to learn better cooperation strategies based on the evaluation. In the process of decentralized execution, agents act independently without the assistance of a central controller. Each agent relies on its local observation and learning strategies to make decisions. This requires agents to make autonomous decisions and adapt to constantly changing environments or scenarios.

Therefore, to implement multi-agent systems in a distributed manner in the real world, updating and managing parameters is a key step in ensuring the effective operation of the system. The concept of the distributed system is shown in Fig. 12. Firstly, we deploy a communicable edge server at each intersection responsible for data collection and model deployment. Technically, we need to design a distributed parameter update mechanism where each edge server of the intersection is responsible for updating its local parameters and coordinating its update process by regularly exchanging information with other intersections. This mechanism ensures that the parameter updates of the entire system are synchronized and consistent. Secondly, to manage these parameters,

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

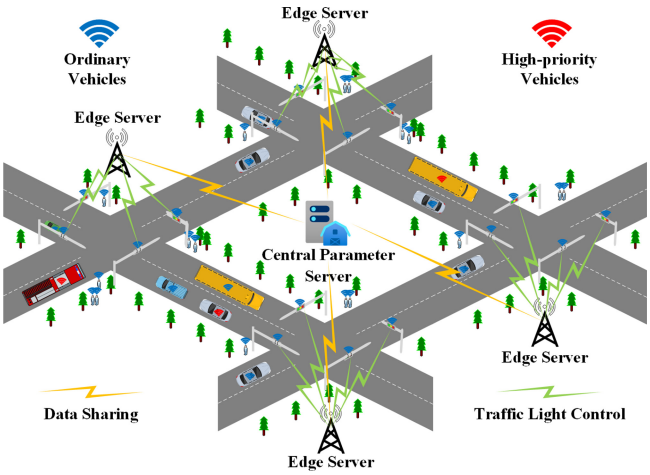

Fig. 12. The distributed traffic light control system.

we need a central parameter server responsible for storing and synchronizing the parameters of all intersections. After updating the local parameters, each edge server of the intersection will send the updated parameters to the central parameter server for storage and synchronization. In this way, even if one edge server fails or goes offline, other intersections can still obtain the latest parameters from the central parameter server, ensuring the robustness and reliability of the system.

VII. Conclusion

In this paper, we have proposed a cooperative adaptive traffic light control approach CACOM which integrates an options framework and a mixing network. Firstly, the traffic light control problem in multi-intersection scenarios has been formulated as a multi-agent Markov game problem and the intersection has been modeled as an agent. To solve the problem, a cooperative MARL framework incorporating the options framework and the mixing network has been designed. The options framework equips the intersections with long sequential decision-making capability, and the mixing network realizes the cooperation among the intersections. In addition, a weight generator based on the contribution of a single agent to a global objective has been designed for the mixing network, so that the agent can take into account the influence of the traffic condition of each intersection when cooperating. Experimental results have shown that our proposed method outperforms the compared baseline methods. It reduces about 24% and 15% in HWT and WT respectively over the best baseline method in the Suzhou Industrial Park scenario, and about 42% and 6% in HWT and WT over the best baseline method in the Beijing Daxing District scenario.

References

[1] B. Liu, W. Han, E. Wang, S. Xiong, C. Qiao, and J. Wang, "An efficient message dissemination scheme for cooperative drivings via cooperative hierarchical attention reinforcement learning," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 5527–5542, May 2024.

[2] R. Ke et al., "Lightweight edge intelligence empowered near-crash detection towards real-time vehicle event logging," *IEEE Trans. Intell. Veh.*, vol. 8, no. 4, pp. 2737–2747, Apr. 2023.

[3] P. Dai et al., "Meta reinforcement learning for multi-task offloading in vehicular edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 3, pp. 2123–2138, Mar. 2024.

[4] K. Liu et al., "Fog computing empowered data dissemination in software defined heterogeneous VANETs," *IEEE Trans. Mobile Comput.*, vol. 20, no. 11, pp. 3181–3193, Nov. 2021.

[5] P. Dai, M. Wu, K. Li, X. Wu, and Y. Ding, "Joint optimization for quality selection and resource allocation of live video streaming in Internet of Vehicles," *IEEE Trans. Services Comput.*, early access, Jan. 1, 2024, doi: 10.1109/TSC.2023.3349051.

[6] F. Song et al., "Evolutionary multi-objective reinforcement learning based trajectory control and task offloading in UAV-assisted mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 12, pp. 7387–7405, Dec. 2023.

[7] T. Gong, L. Zhu, F. R. Yu, and T. Tang, "Edge intelligence in intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 9, pp. 8919–8944, Sep. 2023.

[8] B. Liu et al., "Multi-agent attention double actor-critic framework for intelligent traffic light control in urban scenarios with hybrid traffic," *IEEE Trans. Mobile Comput.*, vol. 23, no. 1, pp. 660–672, Jan. 2024.

[9] J. -H. Syu, J. C. -W. Lin, G. Srivastava, and K. Yu, "A comprehensive survey on artificial intelligence empowered edge computing on consumer electronics," *IEEE Trans. Consum. Electron.*, vol. 69, no. 4, pp. 1023–1034, Nov. 2023.

[10] K. Liu et al., "Accelerating DNN Inference with reliability guarantee in vehicular edge computing," *IEEE/ACM Trans. Netw.*, vol. 31, no. 6, pp. 3238–3253, Dec. 2023.

[11] L. Dong, H. Zhang, K. Yang, D. Zhou, J. Shi, and J. Ma, "Crowd counting by using top-k relations: A mixed ground-truth CNN framework," *IEEE Trans. Consum. Electron.*, vol. 68, no. 3, pp. 307–316, Aug. 2022.

[12] Y. Li et al., "Next-generation consumer electronics data auditing scheme toward cloud–edge distributed and resilient machine learning," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 2244–2256, Feb. 2024.

[13] K. Huang et al., "A novel accurate insecticidal counting method based on solar insecticidal lamp using machine learning," *IEEE Trans. Consum. Electron.*, vol. 69, no. 4, pp. 1045–1054, Nov. 2023.

[14] C. Liu and K. Liu, "Toward reliable DNN-based task partitioning and offloading in vehicular edge computing," *IEEE Trans. Consum. Electron.*, vol. 70, no. 1, pp. 3349–3360, Feb. 2024.

[15] O. Robert and P. Wagner, "Delay-time actuated traffic signal control for an isolated intersection," presented at Proc. 90th Annu. Meeting Transp. Res. Board (TRB), 2011.

[16] G. Long, A. Wang, and T. Jiang, "Traffic signal self-organizing control with road capacity constraints," in *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 18502–18511, Oct. 2022.

[17] Y. Zhang, Y. Zhang, and R. Su, "Pedestrian-safety-aware traffic light control strategy for urban traffic congestion alleviation," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 178–193, Jan. 2021.

[18] N. Wu, D. Li, Y. Xi, and B. de Schutter, "Distributed event-triggered model predictive control for urban traffic lights," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 4975–4985, Aug. 2021.

[19] R. Dai, C. Ding, X. Wu, B. Yu, and G. Lu, "Coupling control of traffic signal and entry lane at isolated intersections under the mixed-autonomy traffic environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 10, pp. 10628–10642, Oct. 2023.

[20] J. Guo, L. Cheng, and S. Wang, "CoTV: Cooperative control for traffic light signals and connected autonomous vehicles using deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 10, pp. 10501–10512, Oct. 2023.

[21] N. Kumar, S. Mittal, V. Garg, and N. Kumar, "Deep reinforcement learning-based traffic light scheduling framework for SDN-enabled smart transportation system," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2411–2421, Mar. 2022.

[22] W.-Y. Lin et al., "Temporal difference-aware graph convolutional reinforcement learning for multi-intersection traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 1, pp. 327–337, Jan. 2024.

[23] A. R. M. Jamil, K. K. Ganguly, and N. Nower, "Adaptive traffic signal control system using composite reward architecture based deep reinforcement learning," *IET Intell. Transp. Syst.*, vol. 14, no. 14, pp. 2023–2041, 2021.

[24] N. Wu, D. Li, and Y. Xi, "Distributed integrated control of a mixed traffic network with urban and freeway networks," *IEEE Trans. Control Syst. Technol.*, vol. 30, no. 1, pp. 57–70, Jan. 2022.

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.

[25] J. Liu, S. Qin, Y. Luo, Y. Wang, and S. Yang, "Intelligent traffic light control by exploring strategies in an optimised space of deep Q-learning," *IEEE Trans. Veh. Technol.*, vol. 71, no. 6, pp. 5960–5970, Jun. 2022.
[26] 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. 2022.
[27] T. Rashid, M. Samvelyan, C. S. D. Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *J. Mach. Learn. Res.*, vol. 21, no. 178, pp. 7234–7248, 2020.
[28] S. Zhang and S. Whiteson, "DAC: The double actor-critic architecture for learning options," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 2012–2022.
[29] R. Lowe, Y. Wu, A. Tamar, et al., "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.
[30] J. N. Foerster, G. Farquhar, T. Afouras, et al., "Counterfactual multi-agent policy gradients," in *Proc. 32nd AAAI Conf. Artif. Intell. 13th Innovat. Appl. Artif. Intell. Conf. 8th AAAI Symp. Educ. Adv. Artif. Intell.*, 2018, pp. 2974–2982.
[31] C. Yu, A. Velu, and E. Vinitsky, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Proc. Neural Inf. Process. Syst.*, 2022, pp. 24611–24624.
[32] P. Sunehag et al., "Value-Decomposition networks for cooperative multi-agent learning based on team reward," in *Proc. 17th Int. Conf. Auton. Agents MultiAgent Syst.*, 2018, pp. 2085–2087.
[33] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proc. Neural Inf. Process. Syst.*, 2020, pp. 10199–10210.
[34] C. Wang, Y. Xu, J. Zhang, and B. Ran, "Integrated traffic control for freeway recurrent bottleneck based on deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15522–15535, Sep. 2022.
[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv.1707.06347*.
[36] P. A. Lopez et al., "Microscopic traffic simulation using SUMO," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, 2018, pp. 2575–2582.
[37] K. P. Hardinda, Y. Sei, Y. Tahara, and A. Ohsuga, "Adaptation plan policy in traffic routing for priority vehicle," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, 2020, pp. 113–118.
[38] M. Holzleitner et al. "Convergence proof for actor-critic methods applied to PPO and RUDDER," in *Transactions on Large-Scale Data-and Knowledge-Centered Systems XLVIII*. Berlin, Germany: Springer, 2021.

Haiyong Shi received the B.Eng. degree from Jilin Normal University, Siping, Jilin, China, in 2019, and the M.Eng. degree from Changzhou University, Changzhou, Jiangsu, China, in 2023. He is currently pursuing the Ph.D. degree with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology. His research interests include reinforcement learning and intelligent connected vehicles.

Bingyi Liu received the Ph.D. degree from the Department of Computer Science, Wuhan University, Wuhan, China, in 2017, and the Department of Computer Science, City University of Hong Kong, Hong Kong, in 2018. He was a Visiting Scholar with the University at Buffalo, The State University of New York, Buffalo, NY, USA, from 2016 to 2017. He is currently a Professor with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan. His research interests include Internet of Vehicles, edge computing, autonomous vehicles, and intelligent transportation systems.

Enshu Wang received the Ph.D. degree from the Department of Computer Science and Engineering, University at Buffalo, State University of New York in 2023. He is currently an Associate Professor with the School of Cyber Science and Engineering, Wuhan University. His research interests include reinforcement learning, Internet of Vehicles, and intelligent transportation systems.

Weizhen Han (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees from the Wuhan University of Technology, Wuhan, China, in 2019 and 2022, respectively, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Artificial Intelligence. His research interests include reinforcement learning and vehicular ad-hoc networks.

Jinfan Wang received the B.E. degree in software engineering from East China Jiaotong University in 2012, the M.Sc. degree in computer science from the City University of Hong Kong in 2013, and the Ph.D. degree from the Department of Computer Science, City University of Hong Kong. He is currently an Assistant Professor with the Institute of Future Networks, Southern University of Science and Technology. His research interests include computer networking, cloud computing, software-defined networks, and blockchain.

Shihong Cui is currently pursuing the Ph.D. degree in computer science and artificial intelligence with the Wuhan University of Technology. He is a General Manager with the Marketing Department, Tianyijiaotong Technology Company Ltd. His research interests include C-V2X, 5G networks, intelligent connected vehicles, and smart transportation.

Libing Wu (Member, IEEE) received the B.S. and M.S. degrees in computer science from Central China Normal University, Wuhan, China, in 1994 and 2001, respectively, and the Ph.D. degree in computer science from Wuhan University, Wuhan, in 2006. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, and also with the Shenzhen Research Institute, Wuhan University, Shenzhen, China. His research interests include network security, Internet of Things, and machine learning.

Authorized licensed use limited to: VIT University. Downloaded on January 22,2025 at 21:13:22 UTC from IEEE Xplore. Restrictions apply.