Urban Traffic Control in Software Defined Internet of Things via a Multi-Agent Deep Reinforcement Learning Approach

Jiachen Yang, *Member, IEEE*, Jipeng Zhang, and Huihui Wang, *Senior Member, IEEE*

***Abstract*— As the growth of vehicles and the acceleration of urbanization, the urban traffic congestion problem becomes a burning issue in our society. Constructing a software defined Internet of things(SD-IoT) with a proper traffic control scheme is a promising solution for this issue. However, existing traffic control schemes do not make the best of the advances of the multi-agent deep reinforcement learning area. Furthermore, existing traffic congestion solutions based on deep reinforcement learning(DRL) only focus on controlling the signal of traffic lights, while ignore controlling vehicles to cooperate traffic lights. So the effect of urban traffic control is not comprehensive enough. In this article, we propose Modified Proximal Policy Optimization (Modified PPO) algorithm. This algorithm is ideally suited as the traffic control scheme of SD-IoT. We adaptively adjust the clip hyperparameter to limit the bound of the distance between the next policy and the current policy. What's more, based on the collected data of SD-IoT, the proposed algorithm controls traffic lights and vehicles in a global view to advance the performance of urban traffic control. Experimental results under different vehicle numbers show that the proposed method is more competitive and stable than the original algorithm. Our proposed method improves the performance of SD-IoT to relieve traffic congestion.**

***Index Terms*— Urban traffic control, software defined internet of things, multi-agent deep reinforcement learning, modified proximal policy optimization.**

I. Introduction

In THE last decades, urban traffic congestion was more and more severe because of the sharply increased vehicle population. Urban traffic congestion leads to pollution of noise and air, wastes drivers' time and seriously affects our society. To cope with the urban traffic congestion problem, many traffic control approaches are proposed. Traditional solutions include increasing the number of streets and broadening roads. But these ways do not efficiently improve the mobility of traffic networks. With the development of Software Defined Network(SDN) and the Internet of Things(IoT), building an SD-IoT with appropriate flow control plans provides an efficient way to make use of existing traffic networks.

Manuscript received February 20, 2020; revised June 9, 2020; accepted September 8, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61871283, in part by the Foundation of Pre-Research on Equipment of China under Grant 61400010304, and in part by the Major Civil-Military Integration Project, Tianjin, China, under Grant 18ZXJMTG00170. The Associate Editor for this article was Z. Lv. *(Corresponding author: Jipeng Zhang.)*

Jiachen Yang and Jipeng Zhang are with the School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: patriotzjp@tju.edu.cn).

Huihui Wang is with the Department of Engineering, Jacksonville University, Jacksonville, FL 32211 USA.

Digital Object Identifier 10.1109/TITS.2020.3023788

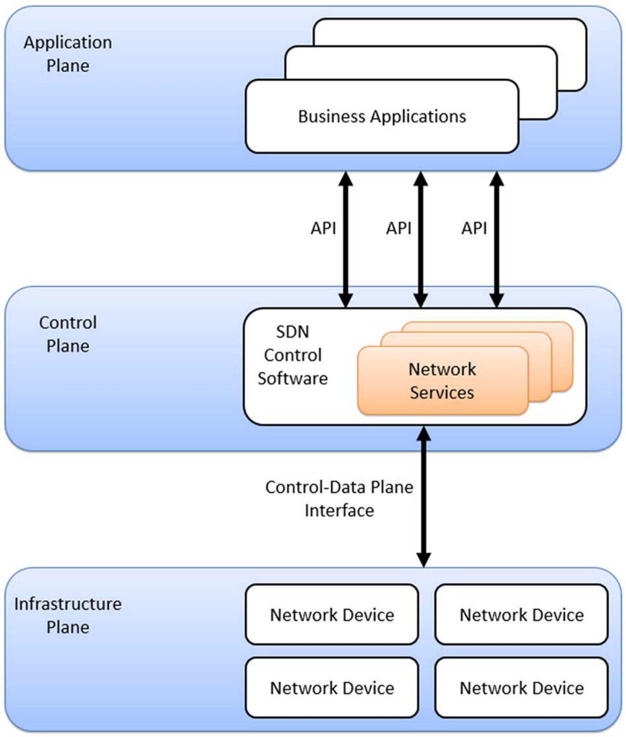


Fig. 1. SDN Architecture.

SD-IoT is the combination of SDN and IoT. SDN is defined and controlled in the form of software programming. Due to the separation of control plane and data plane and the characteristics of open programmability, SDN is considered as a revolution in the field of network, which gives an advanced way for searching new Internet architecture. The architecture of SDN is described in Figure1. IoT refers to the real-time collection of any object requiring observing, contact and communication through diverse information sensors, global positioning system as well as other devices. Through all kinds of possible network access, IoT realizes the connection between objects and people, and realizes the intelligent perception, recognition and management of objects. Misbahuddin *et al.* proposed a solution of intelligent city traffic management based on Internet of things, that is, traffic flow is dynamically conducted by field traffic managers through smartphones, or can be centrally monitored through the Internet [1]. Muñoz *et al.* proposed and evaluated the Internet of things sensing SDN and cloud architecture which deploys the Internet of things traffic monitoring and congestion avoidance technology, to effectively distribute the Internet of things analysis and processing from the core data center to the network edge [2]. However, these methods do not incorporate Deep Reinforcement Learning(DRL) algorithms to improve performance.

Recently, DRL algorithms attract researchers' attention because of the ability to intelligently make decisions. DRL has many advantages, including data fitting ability of Deep Learning(DL) and autonomous decision-making ability of Reinforcement Learning(RL). DL is a sub-field of machine learning, it has a great impact on our society over the past few years. Because deep learning algorithms can accurately extract features and fit almost any functions, it has made great progress in many areas with the development of datasets and computing power, such as object detection [3]–[5], face recognition [6], [7], voice recognition [8], [9], natural language processing [10], [11] and so on. Except for DL, RL is another subset of machine learning. The key to RL is helping agents find the optimal policy and take right actions in a given environment to maximize reward. Q-learning [12], State–Action–Reward–State–Action(SARSA) [13] are classic RL algorithms. They use tabular to store state-action pairs and corresponding rewards. The tabular storing method is time-consuming and takes up a lot of space. To overcome these shortcomings, deep learning and reinforcement learning combine. DRL has advantages of DL and RL and obtained considerable development. Silver *et al.* proposed AlphaGo [14]. AlphaGo won four out of five matches against Sedol, a world-class professional player, proving that AI is advanced enough to beat humanity at one of its most complex abstract strategy games. Hwangbo *et al.* applied Trust Region Policy Optimization (TRPO) algorithm to improve robot's, named ANYmal, performance in the real world. ANYmal can run forward in 1.2m/s and recover from falling [15]. In addition to the work mentioned above, reinforcement learning made a significant breakthrough in many areas, such as video games [16], E-commerce [17], clinical medicine [18] and so on.

In traffic control area, a lot of novel deep reinforcement learning algorithms are proposed. Pol and Oliehoek [19] employed deep Q-learning method to regulate coordinating traffic lights. Juntao *et al.* [20] propose another value-based deep reinforcement learning algorithm to extract useful features from raw inputs and get a better policy for traffic light controlling than traditional methods which depend on hand-craft features. Abdelgawa *et al.* [21] used an RL algorithm to control traffic signals under different network configurations and congestion levels. Jin and Ma proposed an adaptive group-based signal control method based on RL [22]. Although these RL algorithms enjoy some success in reducing waiting time of vehicles, these methods only take traffic lights into account and ignore vehicles. Traffic lights and vehicles are not cooperative. The moderating effect of vehicles in relieving traffic jams is not fully played out. Moreover, they do not apply the multi-agent theory to solving traffic congestion. The lack of applying the multi-agent theory limits the effect of RL algorithms in this area.

In the real world, the multi-agent theory has been applied in many areas. Ryan *et al.* introduced Multi-agent Deep Deterministic Policy Gradient algorithm [23]. This method is superior to traditional RL algorithms in some multi-agent environments. Iqbal and Sha introduced Multi-agent Actor-Critic algorithm [24]. This method is flexible enough to be applied to most multi-agent learning problems. Vinyals *et al.* proposed openfive [25] and deepmind proposed AlphaStar [26]. These algorithms obtain great success in Multiplayer Online Battle Arena (MOBA) games and defeat world-class professional players. However, multi-agent RL algorithms did not play a crucial role in relieving the traffic congestion problem.

In this article, we build a traffic light-vehicle cooperation environment based on the SD-IoT. Traffic light-vehicle cooperation refers to comprehensively adjusting the actions of traffic lights and vehicles in a global view. Traffic lights intelligently modify traffic signal timing plan while vehicles intelligently change speed to solve traffic congestion and to boost the average speed of vehicles. This way leads to higher traffic efficiency and less commuting time. This environment includes traffic lights controlled by DRL algorithms, vehicles controlled by DRL algorithms and vehicles controlled by fixed rules. We can apply DRL algorithms to comprehensively control traffic lights and vehicles. DRL algorithms guide agents, including traffic lights and vehicles controlled by DRL algorithms, find a proper policy to cooperate with others in our multi-agent smart grid system. Therefore, the average speed of all vehicles is increased and the traffic congestion problem is relieved in this environment. What's more, based on the original Proximal Policy Optimization (PPO) algorithm [27], we applied the Modified PPO algorithm to decide the actions of traffic lights and vehicles controlled by RL algorithms. Moreover, to help performance of algorithm jump from local optimal solution, we propose clip hyperparameter exploration mechanism. Experimental results show that the Modified PPO algorithm with clip hyperparameter exploration mechanism outperforms the original PPO algorithm and the Modified PPO algorithm without clip hyperparameter exploration mechanism.

The rest of our paper is organized as follows. Section II describes related work. Section III briefly introduces necessary background knowledge and presents the proposed approach. Results and discussion are arranged in Section IV. At last, we conclude this article in Section V.

II. Related Works

SDN obtained a huge development in the last decade. McKeown *et al.* explained the working principle of Openflow and enumerates several application scenarios of Openflow [28]. As the most mature interface protocol, Openflow protocol has been widely supported by the industry. The Openflow switch specification clearly defines the switch composition and Openflow protocol in the switch. The Openflow switch contains one or more flow tables and a group table, as well as a secure channel interface for communicating with the Openflow controller. In order to avoid the over-expansion of the single flow table mechanism, the multi-level flow table mechanism is introduced in the version after Openflow protocol 1.1. To better control SDN, Gude *et al.* proposed NOX [29], which is a single centralized structure controller. Cai et al introduce Maestro, a kind of parallel controller, to give full play to the multi-core parallel processing ability of the high-performance server and improve the performance in the case of large-scale networks. For many medium-sized networks, a single controller is usually enough to complete the corresponding control functions, which will not have a significant impact on the performance. However, for large-scale networks, it is necessary to expand the number of controllers, that is, to physically allocate controllers throughout the network and maintain the control characteristics of the logical center, so that each switch is close to the control, so as to interact with each other to improve the overall performance of the network. Flat control mode is one of expansion methods. Onix [30] synchronous updates between controllers by keeping the consistency of network information base. HyperFlow [31] communicates through registration and broadcast mechanism and reconfigures the switch managed by the failed controller to the new controller through manual configuration when a controller fails. Another expansion method is hierarchical control. Kandoo [32] asks the nearest local controller to judge whether the message belongs to local information. If it gets a negative answer, it will ask the global controller and send the obtained information to the switch. This way avoids the frequent interaction of the global controller and effectively reduces the traffic load. In addition to the above work, Lantz *et al.* proposed Mininet [33]. Mininet is a system that can quickly build a large-scale network prototype on the limited capabilities of a laptop. P4 [34], which is introduced by Bosshart *et al.*, breaks through the constraints of traditional data plane processing architecture, and enables developers to flexibly define the format of various protocol messages, and fully controls the data plane equipment to process data packets through programming in the control plane. Berde *et al.* proposed Open Network Operating System(ONOS) [35].ONOS focuses on how to use the controller to effectively control the operator level network, and create a highly available, scalable, high-performance, fully open-source controller. To tackle the terrible traffic situation of the metropolitans, more and more researchers pay attention to combining SDN with IoT. Ku *et al.* used SDN to provide networks with resilience and programmability and to add new services and functions to Vehicular Ad Hoc Network [36]. Zheng *et al.* [37] proposed a heterogeneous vehicle network architecture based on software definition, which is compatible with a variety of wireless communication protocols. At the same time, to ensure the real-time requirements of automatic driving vehicle communication, this architecture provides a hierarchical control structure. The main controller grasps the global information and is responsible for the resource allocation task at the macro level, while the secondary controller focuses on the resource allocation strategy of driving vehicles. However, these methods do not fully use DRL algorithms and their effects of ascending urban traffic capacity are limited.

Deep learning, a part of machine learning, is a series of approaches based on deep artificial neural networks. Such methods apply deep networks which consist of multiple layers to extract various levels of feature from the raw input. As the performance of computers evolved, deep learning approaches have been greatly developed. In 2006, Hinton *et al.* [38] propose Deep Belief Network (DBN). This model can be viewed as a combination of simple Restricted Boltzmann Machines (RBM) and can be trained greedily. So it is the first effective deep learning algorithm and starts a new era of deep learning. Because DL methods are capable of extracting features automatically and accurately, they replaced traditional methods which depend on hand-craft features. In areas that are related to images and videos, Convolutional Neural Network (CNN) plays a significant role in promoting development. CNN is a kind of neural network that carries out convolution to get a high-level representation of input images. Krizhevsky *et al.* proposed AlexNet [39]. This algorithm achieved the lowest classification error rate in ImageNet LSVRC competition, nearly 12 percent lower than its nearest rival. GoogLeNet was proposed [40]. GoogLeNet applied convolution kernels of different sizes to process input images respectively, then fused these corresponding outcomes to get a better representation of input images. He *et al.* advanced ResNet [41]. ResNet introduced identity mapping to avoid degradation of classification performance. In ImageNet database, it achieves an extremely low error rate on test set and outperformed human experts. Huang *et al.* introduced DenseNet [42], which connects each layer to every other layer in the same block. This design alleviates the vanishing-gradient problem and uses features more efficiently. EfficientNet [43] is introduced by Tan and Le The authors balanced network depth, width, and resolution to achieve better performance. In areas which are related to speech and natural language, Recurrent Neural Network (RNN) leads the trend. Sutskever *et al.* introduced seq2seq algorithm [44]. This algorithm converted English to French excellently. Vaswani *et al.* proposed Transformer model [45]. Depending on the attention mechanism, Transformer model accomplished WMT 2014 English-to-French translation task and got a higher score than ever before.

Reinforcement learning is another part of machine learning. The goal of reinforcement learning algorithms is taking suitable actions to get maximum scores based on given situations. The basic theory of RL originated from the last century. Sutton [46] introduced Temporal Difference (TD) learning. The proposed method can accurately predict future values by a given signal. Watkins *et al.* proposed Q-learning algorithm [12]. Sutton *et al.* introduced SARSA algorithm [13]. These algorithms lay the roots before the combination with DL.

As DL and RL developed, combining DL and RL becomes an inevitable trend. DL needs the strategy generation capability of RL while RL needs the feature extraction and function approximation capability of DL. Mnih *et al.* [47] applied Deep Q-learning Network(DQN) to play Atari games and provided state-of-the-art in six games. This is the first time to combine DL and RL. Since then, DRL has had significant success in numerous areas. Van Hasselt *et al.* [48] introduced

Double Q-learning algorithm. This method reduces overestimation by using two isolated Q-value estimators. Wang *et al.* [49] proposed Dueling DQN. This architecture splits the outcome into two components: state value and advantage value. Experimental results show that this architecture outperforms DQN and double DQN in Atari domain. Although DQN series get success, they cannot work well when action space is continuous. Lillicrap *et al.* introduced Deep Deterministic Policy Gradient (DDPG) algorithm [50]. DDPG is good at generating policy in continuous action space because it provides a deterministic policy to avoid the curse of dimensionality. Based on DDPG, Barth-Maron et al introduced Deep Distributed Distributional Deterministic Policy Gradients (D4PG) algorithm [51]. D4PG replants distributional RL theory to critic network in DDPG algorithm and achieves best performance in several continuous control problems. Twin Delayed Deep Deterministic policy gradient (TD3) algorithm [52] is another modified variant of DDPG. It is proposed by Fujimoto *et al.* This algorithm uses a pair of critic networks and delays updates of the actor. Due to these changes, this algorithm gets excellent scores in a variety of benchmarking environments. DDPG and its variants have some drawbacks: <1> The generated policy is not robust enough. <2> The architecture is too complex. Haarnoja *et al.* introduced Soft Actor-Critic (SAC) algorithm [53]. This method adjusts the objective function. SAC algorithm maximizes the sum of lifetime reward and entropy rather than only lifetime reward. This change brings improvement in several domains such as robotics. All algorithms above are off-policy algorithms, they are complex in a sense. Because the policy used to generate actions is different from the policy that is improved. To address this issue, on-policy algorithms emerge as times require. On-policy methods evaluate the generated policy's value. Proximal Policy Optimization (PPO) algorithm [27] is one of the best on-policy methods. It is proposed by Schulman *et al.* PPO uses clip function to limit KL divergence between policy before and after an update. Wang *et al.* proposed Trust Region-Guided Proximal Policy Optimization (TRGPPO) algorithm [54]. This algorithm adaptively adjusts the clipping range under the guidance of the trust region. A series of experiments demonstrate the strength of the proposed method. Ye *et al.* introduced Dual-PPO algorithm in the same year [55]. Dual-PPO algorithm effectively guarantees convergence with large and deviated batches. This method helps Tencent Lab to develop AI which won 99.8% 1v1 matches of MOBA game Honor of Kings.

As DRL algorithms' application scope expands, scenes are more and more eye-catching which include confrontation and cooperation of multiple agents. Deep Multi-agent Reinforcement Learning algorithms are continuously proposed to optimize confrontation and cooperation of multiple agents in given scenes. Lowe *et al.* [23] proposed Multi-agent Actor-Critic algorithm. Every agent studies a centralized critic. And the centralized critic receives actions and observations of all agents. Iqbal and Sha [24] introduced a similar algorithm —— Multi-Actor-Attention-Critic algorithm. This algorithm adopts "centralized critic and multiple actors" mode and replaces DDPG architecture in a centralized critic by SAC architecture. Berner *et al.* [25] proposed the OpenAI Five model which outperformed humans in playing Dota2 MOBA game and defeated professional human teams. Furthermore, Vinyals *et al.* [26] introduced the AlphaStar model which achieved superhuman performance in playing StarCraft II MOBA game and got a place above 99.8% of ranked human players.

Due to great successes in various domains, many scholars tried to make use of these algorithms to control traffic and to relieve traffic congestion. Abdoos *et al.* [56] proposed holonic Q-learning to control the signals. However, this algorithm used RL algorithm without combination with DL algorithm. The proposed algorithm applied some tables to store state space and action space. This method is too simple to accurately present the complexity of a traffic road system. Li *et al.* [57] applied DQN to represent action and state and to generate a signal timing plan. Although this algorithm used DQN, it still has some drawbacks. At first, the environment in that paper is a cross-shape intersection that is naïve for comprehensively describing a traffic system. Moreover, the proposed algorithm controls only traffic lights and ignores vehicles. So, vehicles and traffic lights are not cooperative. Some new ideas emerged in the last years, they were not applied in the traffic control area.

In this article, we propose a simulated environment, based on SD-IoT, that includes several intersections. We comprehensively modify the actions of vehicles and traffic lights simultaneously in a global view to get an optimal solution for relieving traffic congestion. At the same time, we applied multi-agent theory in our work.

III. Method

In this section, we first briefly review some basic concepts and principles of reinforcement learning to better introduce our work. And then we further introduce a significant branch of reinforcement learning —— policy-gradient methods, because we modify the PPO algorithm which belongs to this branch. After the introduction above, we introduce the SD-IoT structure. Finally, we express our environment model as a reinforcement learning task by detailed describing significant RL elements of all agents and propose our Modified PPO algorithm.

A. Background

1) Reinforcement Learning: The core idea of RL algorithms comes from biological mechanisms. Advanced creatures have the ability to change and to optimize their actions for getting better rewards based on the environment. The goal of RL algorithms is making agents have that ability. The relation between elements of RL is shown in Figure 2.

Usually, the problems in which RL algorithms are applied can be viewed as Markov decision processes (MDPs). An MDP can be expressed as a five-tuple $< S, A, T, R, \gamma >$, where S is the state set in a given environment, A is the possible action set in a given environment, T is the transition function which represents the probability of changing from one state to next state, R is the reward function and $\gamma \in [0, 1]$ is the discount factor, which describes different significance of current rewards and future rewards.

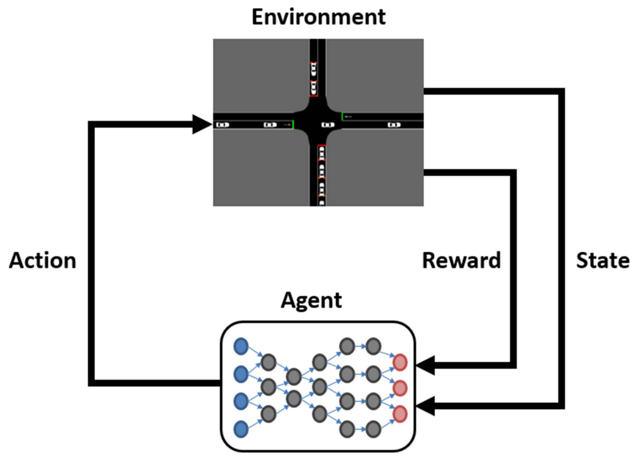


Fig. 2. The relation between basic elements of RL.

In a given environment, every agent receives $s_t \in S$ and chooses an action a_t. After taking the chosen action, the given environment will generate $s_{t+1} \in S$ based on the transition function T. And then, reward r_t is produced by the reward function R and is sent to the agent. If some actions lead to high reward, the possibility of taking these actions will increase, and vice versa. The goal of RL algorithms is maximizing the cumulative discounted reward. The cumulative discounted reward R_t is defined as follows:

$$R_t = E\left[\sum_{k=0}^{T} \gamma^k r_{t+k}\right] \tag{1}$$

where t indicates the current timestep, r_{t+k} is the reward of $(t+k)^{th}$ timestep, T denotes the final timestep, γ is the discount factor.

"Exploit" and "Explore" are two exclusive goals of reinforcement learning algorithms. "Exploit" refers to using current optimal policy to get a cumulative discounted reward as high as possible. "Explore" refers to finding a global optimal policy rather than applying a current optimal policy. $\varepsilon -$ greedy method is proposed to balance "Exploit" and "Explore". Agents use current optimal policy with probability $= (1-\varepsilon)$ and explore other policies with probability $= \varepsilon$.

2) Policy Gradient Methods: Policy gradient methods search policy space to learn policy directly. These methods adjust parameters of policy function and optimize policy by gradient ascent or gradient descent algorithms. They construct an estimator of policy gradient and treat the estimator as a part of gradient ascent or gradient descent algorithms. The popular form of the estimator of policy gradient is as follows:

$$g = \hat{E}\left[\nabla_\theta \log \pi_\theta\left(a_t|s_t\right) A_t\right] \tag{2}$$

where π_θ is a policy which is parameterized by θ, A_t is the estimated advantage function at t^{th} timestep. $\hat{E}[\ldots]$ denotes the empirical average over chosen samples. The estimator g is acquired by taking the derivative of the following objective formula:

$$L^{PG}(\theta) = \hat{E}_t\left[\log \pi_\theta\left(a_t|s_t\right) A_t\right] \tag{3}$$

PPO algorithm introduces the clip function in an objective formula. The objective formula of the PPO algorithm is as follows:

$$L^{PPO}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}\left(r_t(\theta), 1-\beta, 1+\beta\right) A_t\right)\right] \tag{4}$$

where β is a hyperparameter, and $r_t(\theta)$ denotes the probability ration between the old and new policy. $r_t(\theta)$ is represented by the following form:

$$r_t(\theta) = \frac{\pi_\theta\left(a_t|s_t\right)}{\pi_{\theta_{old}}\left(a_t|s_t\right)} \tag{5}$$

The objective formula takes the smaller one between two terms. So actually the outcome of this objective formula is a lower bound of the first term. This program can make objective improvements and guarantee that the distance between the new and old strategies is not too large.

B. SD-IoT Structure

The substructure and superstructure of SD-IoT in this article are shown in Figure 3. In a typical scenario of the SD-IoT substructure, vehicles can communicate with nearby vehicles. These communications consist of V2V (Vehicle-to-Vehicle) connections. Similarly, the communications between vehicles and traffic lights consist of V2L(Vehicle-to-Light) connections. V2I(Vehicle-to-Infrastructure) connections and L2I (Light-to-Infrastructure) connections respectively refer to the communications between vehicles and infrastructures and the communications between vehicles and traffic lights. Vehicles and traffic lights report their state information, such as speed and location, to the infrastructure of intersection where they are. The infrastructure uploads these data to the superstructure of SD-IoT. The main role of the SDN controller is a network monitor. Useful data is uploaded to the function module. The policy generator of function module applied Modified PPO algorithm, which is proposed in the next section, to create proper action of every agent (vehicles and traffic lights controlled by RL algorithms) in SD-IoT. The next step is transmitting information about proper action to the corresponding agent.

C. Environment Model

The environment which is used in this article is based on the "multiagent_traffic_light_grid" example in Flow [58]. Vehicles enter the system from one entrance and get away from another exit. There are several 4-arm intersections in the environment. Half of the vehicles are controlled by RL algorithms and the other half of them are controlled by constant rules. Because we want to simulate the situation that RL vehicles are not universally popularized. But all of the traffic lights are controlled by RL algorithms. Agents in this environment refer to traffic lights and vehicles under RL algorithms' control. The intersection geometry and the environment with 2×3 intersections(2 rows and 3 columns) are respectively shown in Figure 4 and Figure 5.

1) State Space: For every traffic light in this smart grid system, there are two different states as follows:

<1> GRGR: Vehicles may pass the junction in north-south direction and vehicles must stop in the east-west direction.

<2> RGRG: Vehicles may pass the junction in east-west direction and vehicles must stop in the north-south direction.

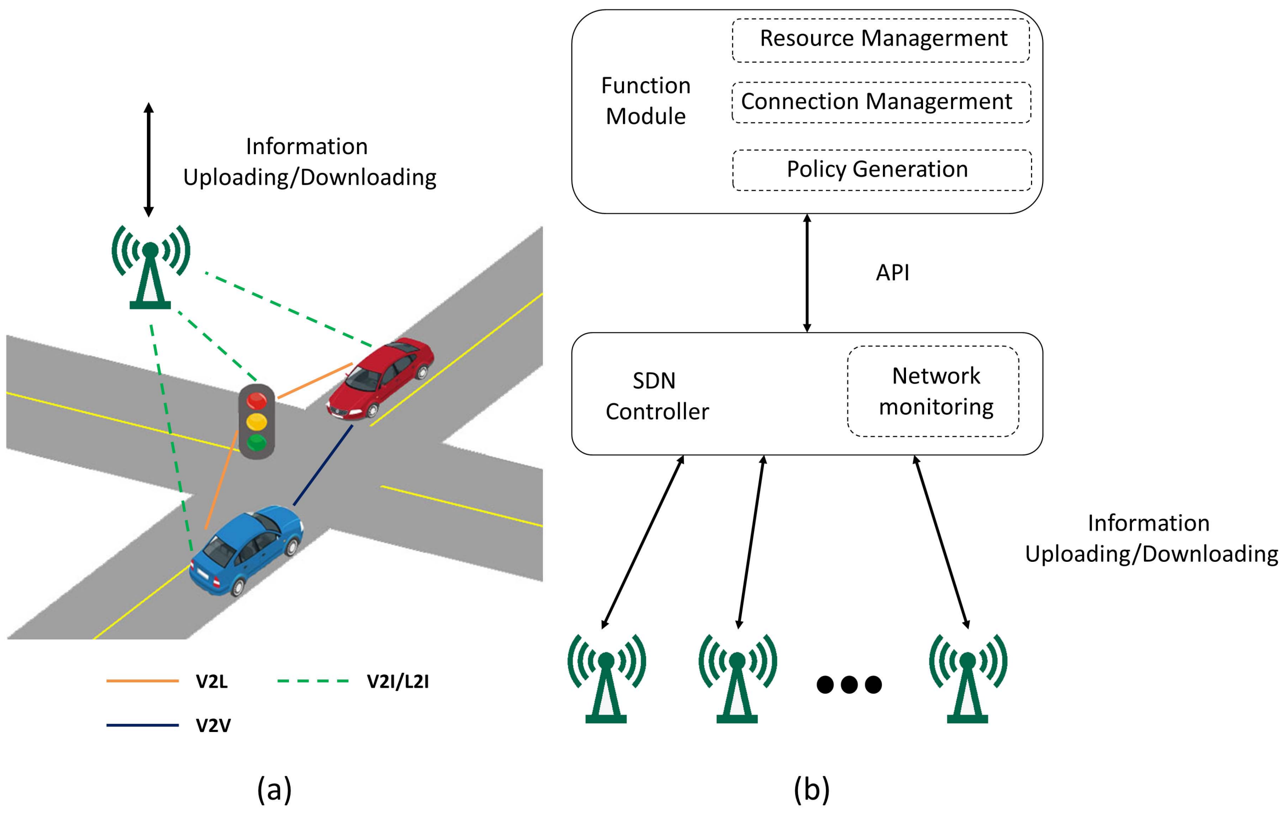


Fig. 3. (a) The substructure of SD-IoT (b) The superstructure of SD-IoT.

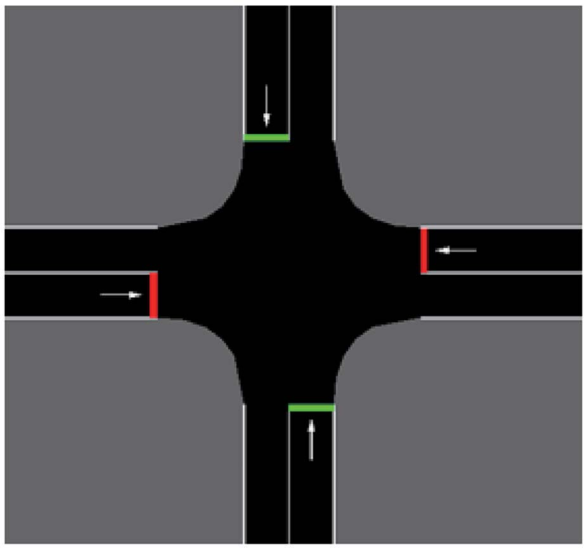


Fig. 4. The intersection geometry in the smart grid system.

State space of every vehicle is more complex than the one of traffic lights, it includes some attributes:

<1> velocity;

<2> distance between the vehicle and next intersection;

<3> the id of the edge of it is traveling on.

Every traffic light and vehicle, including the one which is not controlled by RL algorithms, upload their state information to the SD-IoT superstructure.

2) Action Space: For traffic lights in the smart grid system, there are two actions as follows:

<1> Converting to GRGR;

<2> Converting to RGRG.

The actions of vehicles in the smart grid system, there are two actions as follows:

<1> Acceleration;

<2> Deceleration.

Function module delivers action messages to every agent to guide these agents for relieving traffic congestion.

3) Reward: In our smart grid system, we punish the large delay. The formula for reward is the negative average delay for all vehicles in the system.

$$reward = -avg\_del \tag{6}$$

$$avg\_del = \frac{\sum_{i=1}^{n} del_i}{n} \tag{7}$$

$$del_i = \frac{v_{top} - v_i}{v_{top}} \times timestep \tag{8}$$

where n is the total number of vehicles in system, v_{top} refers to the maximum speed allowed in the system, v_i is the average speed of i^{th} vehicle and timestep is the number of steps after the current episode begins.

Because the reward of every agent takes average delay for all vehicles in the system into consideration, every agent has "team spirit". In other words, they need to cooperate to get high rewards. As a result, all traffic lights and vehicles based RL are cooperative for increasing the speed of vehicles.

Every agent needs to upload their reward information. Function module needs this to understand the effectiveness of traffic control.

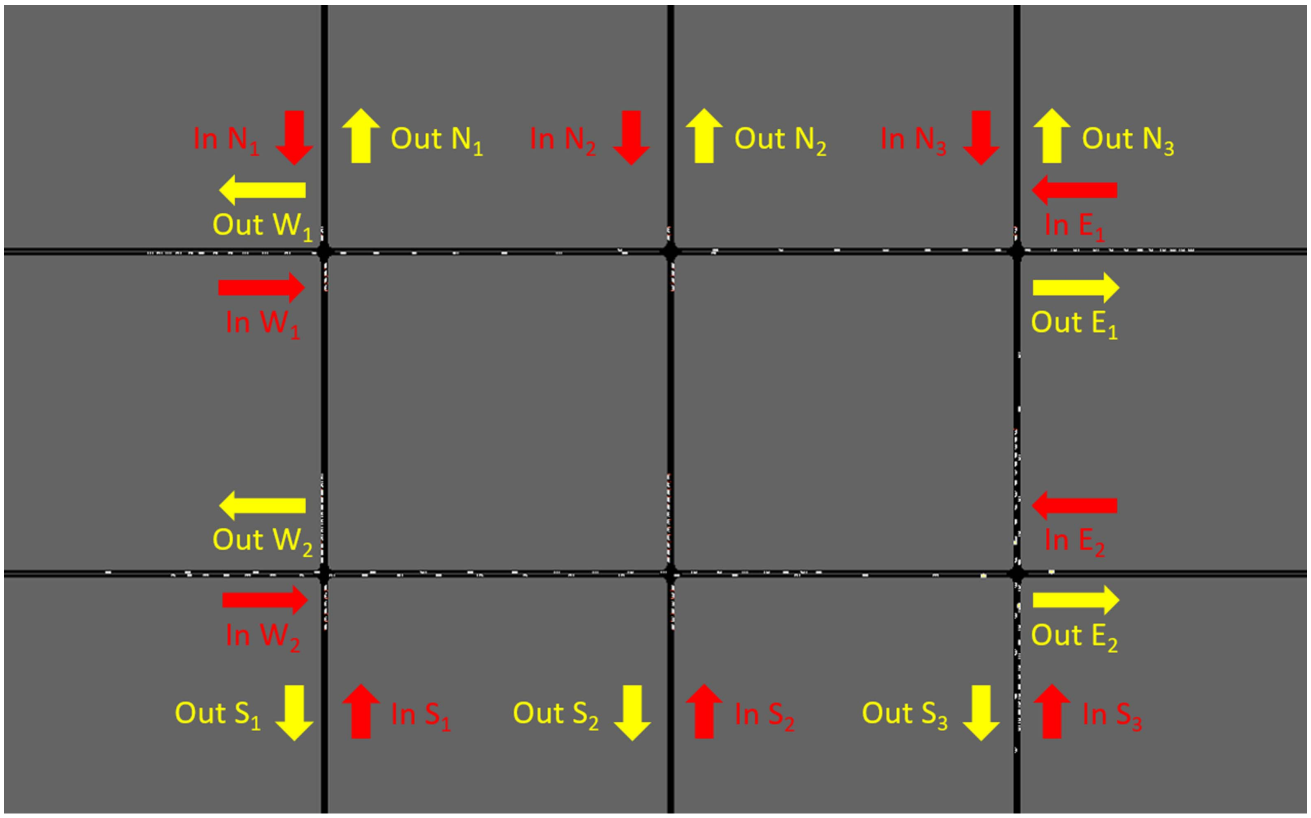


Fig. 5. The environment of the smart grid for the six intersections.

D. Modified PPO Algorithm

The original PPO algorithm uses a constant β as a clip hyperparameter. With the transition of key attributes of agents and the increase of the training epoch number, this setting is not adaptive enough to guide agents to find the optimal policy. In this section, we will introduce our Modified PPO algorithm, which takes key attributes of vehicles in this smart grid and the number of iterations into account. What's more, our Modified PPO algorithm adopts a mechanism that is similar to the "ε-greedy" mechanism in the exploration-exploitation balance. The aim of adopts the mechanism is reducing the possibility of being stuck in local optimal.

As the number of iteration increases, the policy of agents is improved but the likelihood of suffering performance degradation surges. As a result, we tightened the restriction between the new policy and current policy by reducing hyperparameter β to avoid performance degradation. In our multi-agent smart grid system, we pay attention to the average speed of each vehicle and the training epoch number. We incorporate these factors into the progress of calculating clip hyperparameter. The average speed of each vehicle is a crucial characteristic because it, to a large extent, represents the traffic congestion level of our multi-agent smart grid system. Obviously, the faster the average speed of each vehicle, the more likely our multi-agent smart grid system effectively relieve traffic jam. The training epoch number shows the progress of training models, updating parameters and improving policies. To make our algorithm is suited for the need for training models and driving up the performance of the proposed algorithm, it is necessary to consider the above factors. In our Modified PPO algorithm, clip hyperparameter β_{Mod} is the sum of two elements. The first element is cp_a, which is defined in Formula (9).

$$cp_a = base \times \frac{v_{top} - v_{aver}}{v_{top}} \tag{9}$$

where $base \in [0.1, 0.2, 0.3]$ is a constant hyperparameter. This element describes the gap between the upper limit of speed and the average speed of each vehicle. The goal of a_{cp} is restricting the distance between the new policy and current policy by introducing increasing v_{aver}. The policy is better, v_{aver} is faster, and the value of a_{cp} becomes smaller.

The second element is cp_b, which is defined in Formula (10).

$$cp_b = base \times (1 + 0.002 \times iter)^{-0.75} \tag{10}$$

where $base \in [0.1, 0.2, 0.3]$ is a constant hyperparameter, same as the one in Formula (9). The goal of b_{cp} is restricting the distance between the new policy and current policy by introducing the iteration number. As the iteration number increases, the value of b_{cp} becomes smaller.

However, blindly reducing clip hyperparameter β_{Mod} may lead to a serious consequence. When clip hyperparameter β_{Mod} is too small, the distance between the new policy and current policy close to zero. This situation makes the algorithm converge to a locally optimal solution and keep the algorithm far from the global optimal solution. So we introduce a mechanism that is similar to the ε-greedy strategy in an exploration-exploitation balance problem. This mechanism loosens the restriction between the new policy and current policy with a small probability and helps the policy of

Algorithm 1 Modified PPO

```
1: Initialize policy parameters $\theta$ and value function parameters $\phi$;
2: Initialize iter counter $i \leftarrow 0$
3: while $i \leq 5000$ do
4:     Initial episode counter $e \leftarrow 0$
5:     while $e \leq 400$ do
6:         Initial timestep counter $t \leftarrow 0$
7:         while $t \leq 1000$ do
8:             Run current policy $\pi_\theta$,then acquire reward $r_t$ and next state $s_{t+1}$
9:             $t \leftarrow t+1$
10:        end while
11:        $R = V(s;\phi)$
12:        for $j \in [1,\ldots,400]$ do
13:            $A = R - V(s_j;\phi)$
14:            Compute clip hypermeter $\beta_{Mod}$ by Formula (9)-(11)
15:            Compute objective function by Formula (12)
16:            Update the policy by $\theta^{j+1} = \arg\max_\theta L^{\text{Mod\_PPO}}(\theta)$
17:            Fit value function by regression on mean-squared error $\phi = \arg\min_\theta A^2$
18:        end for
19:        $e \leftarrow e+1$
20:    end while
21:    $i \leftarrow i+1$
22: end while
```

agents to jump from local optimum. The formula of calculating hyperparameter β_{Mod} with clip hyperparameter exploration is as follows:

$$\beta_{Mod} = \begin{cases} base & \text{with } \varepsilon \text{ probability} \\ cp_a + cp_b & \text{otherwise} \end{cases} \tag{11}$$

Besides $base \in [0.1, 0.2, 0.3,]$ another constant hyperparameter is $\varepsilon = 0.1$, this hyperparameter presents the possibility of exploration. υ_{top} represents the upper limit of speed, in our paper the value of υ_{top} is set to 35. υ_{aver} is the average speed.

The objective function of Modified PPO algorithm is as follows:

$$L^{Mod\_PPO}(\theta) = \hat{E}_t \left[\min\left(r_t(\theta)A_t, \operatorname{clip}\left(r_t(\theta), 1-\beta_{Mod}, 1+\beta_{Mod}\right)A_t\right)\right] \tag{12}$$

The pseudo-code for the proposed algorithm is presented by Algorithm 1.

IV. Experimental Studies

We present experimental settings and parameter settings in this part. And then we show the results and corresponding discussion.

A. Experimental Settings

We apply Flow [58] to complete our experiments. Flow is a well-known framework that affords a computational scheme of managing operations and deep RL algorithms in the traffic simulation area. In our experiments, the number of epoch is 5000. In each epoch, 400 episodes are simulated and each episode consists of 1000 time steps. The discount factor $\gamma = 0.999$. We use generalized advantage estimator(GAE) [59] when computing advantage function. The parameter λ is set to 0.97. The upper bound of speed is 35, the upper bound of acceleration is 2.6 while the one of deceleration is 4.5. Vehicles enter our system at 15 speed. Vehicles do not turn left or right, the only possible direction is going straight. All vehicles which achieve entrance of the system will enter the system from another entrance.

B. Results and Discussions

With the reward function that we noted above, we compare the performance of the original PPO algorithm and the Modified PPO algorithm without and with clip hyperparameter exploration in solving the traffic congestion problem in our environment. Figure 4 depicts the average reward of following methods when the number of vehicles in this smart grid equals 50 and 100:

<1> The original PPO algorithm

<2> The Modified PPO algorithm without clip hyperparameter exploration;

<3> The Modified PPO algorithm with clip hyperparameter exploration.

Three columns of subgraphs of Figure 6 respectively describe the outcome of these methods with base $\in$ [0.1,0.2,0.3]. PPO_base0.1 refers to the original PPO algorithm with hyperparameter $\beta = 0.1$.Mod_PPO_base0.1_w/o means Modified PPO algorithm without clip hyperparameter exploration and the base is 0.1. Mod_PPO_base0.1_w/ is Modified PPO algorithm with clip hyperparameter exploration and the base is 0.1. We use similar abbreviations when hyperparameter β of the original PPO algorithm and the base of the Modified PPO algorithm are 0.2 and 0.3. As shown in Figure 6, the Modified PPO algorithm with clip hyperparameter exploration outperforms other methods with a big margin, especially when the number of vehicles is 50. The performance of Modified PPO algorithm with clip hyperparameter exploration is far ahead of other methods after the number of epoch is over one thousand. When the number of vehicles is 100, the Modified PPO algorithm with clip hyperparameter exploration is still the method with the best performance. It should be noted that the original PPO algorithm achieves maximal average reward value in a short time when β hyperparameter is 0.3, but it is not stable enough and suffers serious performance declines after shortly getting good results. This phenomenon reflects the drawback of the original PPO algorithm. In sharp contrast, there is no violent fluctuation in the average reward curve of other methods. The maximum values of all plots in two rows of Figure 6 are respectively reported in Table I and Table II.

To explore the influence of training iterations on experimental results, we increased the number of training iterations. Figure 7 shows the outcome of these methods when base is 0.1 and the number of training iterations is 8000. As described in Figure 7, the mean episode rewards of all curves fluctuate in a relatively stable range, when the number

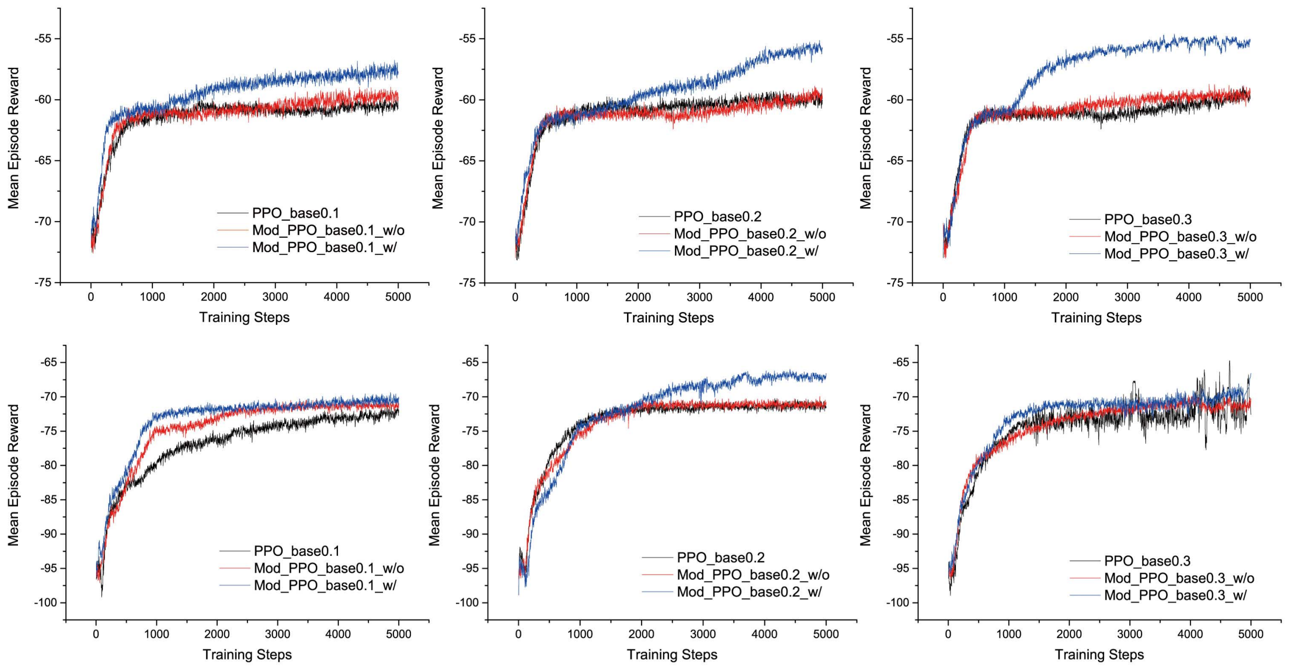


Fig. 6. Top Row: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when the number of vehicles in smart grid equals to 50 and base $\in$ [0.1, 0.2, 0.3]. Bottom Row: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when the number of vehicles in smart grid equals to 100 and base $\in$ [0.1, 0.2, 0.3].

TABLE I
THE MAXIMUM VALUE OF PLOTS IN THE TOP ROW OF FIGURE 6

base value \ Method	PPO	Mod_PPO_w/o	Mod_PPO_w/
0.1	-59.89	-58.96	-56.76
0.2	-59.22	-60.22	-56.06
0.3	-59.41	-59.16	-55.34

TABLE II
THE MAXIMUM VALUE OF PLOTS IN THE BOTTOM ROW OF FIGURE 6

base value \ Method	PPO	Mod_PPO_w/o	Mod_PPO_w/
0.1	-71.20	-70.17	-69.38
0.2	-70.22	-69.94	-65.99
0.3	-64.71	-69.45	-66.58

of training iterations is between 5000 and 8000. And compared to outputs when the number of training iterations is 5000, those with 8000 training iterations are not significantly different. This situation also happens when base is 0.2 and 0.3. The number of timesteps is very large. Results with 5000 training iterations are achieved after $5000 \times 400 \times 1000 = 2 \times 10^9$. Such a huge amount of computation has made the result reach the bottleneck. So increasing the number of training iterations is not the key to improving performance.

To investigate the effect of ε on the experimental results, we extend the range of ε from 0.1 to [0.1,0.2,0.3]. Three columns of subgraphs of Figure 8 respectively describe the outcome of these methods when base is in [0.1,0.2,0.3] and ε is in [0.1,0.2,0.3]. When ε equals to 0.1, curves are relatively stable and finally achieve better mean episode rewards. In contrast, other curves fluctuate violently and the results are poor, even if some of them get better results at the beginning of training. Such results indicate that 0.1 is the best value of ε. 0.2 and 0.3 are so large, leading to abnormal fluctuations of curves.

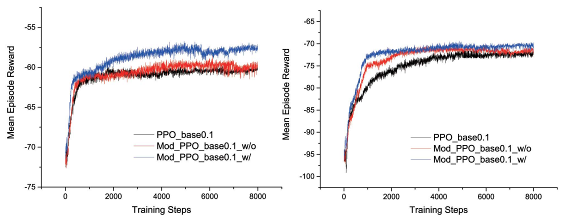


Fig. 7. Left: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when the number of vehicles in smart grid is 50 and base is 0.1. Right: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when the number of vehicles in smart grid is 100 and base is 0.1.

Another factor to be studied is the parameter decay strategy of cp_b. We apply three decay strategies, including exponential decay, linear decay and polynomial decay, to make results more inclusive. The formulas of these decay strategies are as follows:

$$cp_{b1} = base \times (1 + 0.002 \times iter)^{-0.75} \tag{13}$$

$$cp_{b2} = base \times [(1 - endcpb) \times (1 - iter/5000) + endcpb)] \tag{14}$$

$$cp_{b3} = base \times [((1 - endcpb) \times (1 - iter/5000)^2 + endcpb)] \tag{15}$$

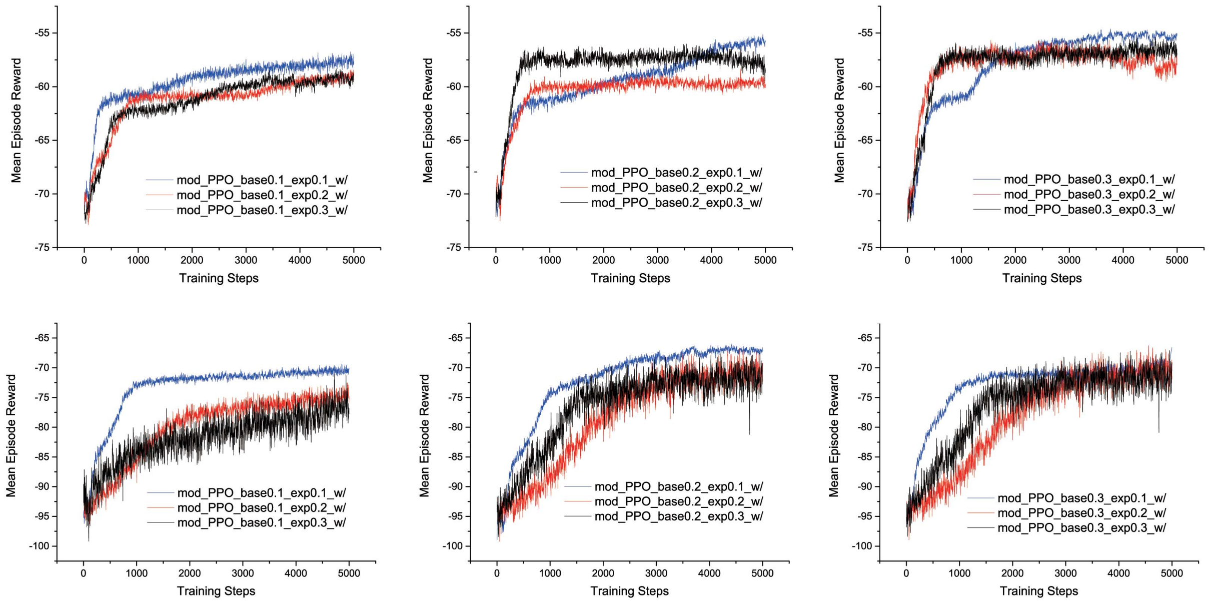


Fig. 8. Top Row: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when the number of vehicles in smart grid equals to 50, base $\in [0.1, 0.2, 0.3]$ and $\varepsilon \in [0.1, 0.2, 0.3]$. Bottom Row: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when the number of vehicles in smart grid equals to 100, base $\in [0.1, 0.2, 0.3]$ and $\varepsilon \in [0.1, 0.2, 0.3]$.

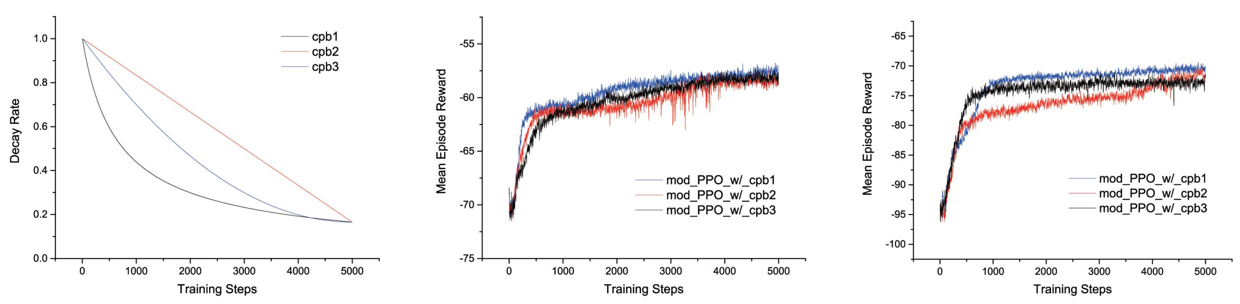


Fig. 9. Left: Decay rate of three decay strategies. Middle: The average reward of Modified PPO with clip hyperparameter exploration and different decay strategies when vehicle number is 50. Right: The average reward of Modified PPO with clip hyperparameter exploration and different decay strategies when vehicle number is 100.

To increase the comparability of these strategies, endcpb is set to $(1+0.002\times 5000)^{-0.75} = 0.1656$. The decay curves of these strategies are shown in the left of Figure 9. The outcome of these decay strategies when base is 0.1, ε is 0.1 and vehicle number is 50 is illustrated in the middle of Figure 9. And similar outcome but vehicle number is 100 is illustrated in the right of Figure 9. The results show that exponential decay strategy is superior to the other two strategies. But the mean episode rewards of exponential decay are not much higher than those of the other two strategies. So from such results, we can get a conclusion that parameter decay strategy of cp_b does not have a great impact on the performance of the algorithm.

Another significant hyperparameter is base. To study the effect of this parameter on the result, we extend the range of base. Figure 10 illustrates mean episode rewards of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration with base $\in$ [0.1,0.2,0.3,0.4,0.5] when vehicle number is 50 and 100 respectively. The result shows that the performance of the algorithm declines when the value of base hyperparameter is too large. In [27], the value of base is fixed at 0.2. So the reasonable range of this parameter is near 0.2.

Based on the previous discussion, we have added two new combinations of the number of vehicles and the number of intersections. These two combinations are as follows:

(1) 50 vehicles and 9 intersections(3 rows and 3 columns);

(2) 100 vehicles and 9 intersections(3 rows and 3 columns);

Results of the first additional combination are described in the top row of Figure 11 and those of another one are described in the bottom row. The number of training iterations is 5000, ε is 0.1, decay strategy is exponential decay and base is in [0.1,0.2,0.3]. The results show that the increase in the number of vehicles and the number of intersections makes traffic congestion more serious, such a phenomenon fits the actual traffic. The Modified PPO algorithm with clip hyperparameter exploration is still the one with the highest mean episode reward.

The experimental results suggest that our proposed method has led to promising results. Our proposed method surpasses the original PPO. And clip hyperparameter exploration

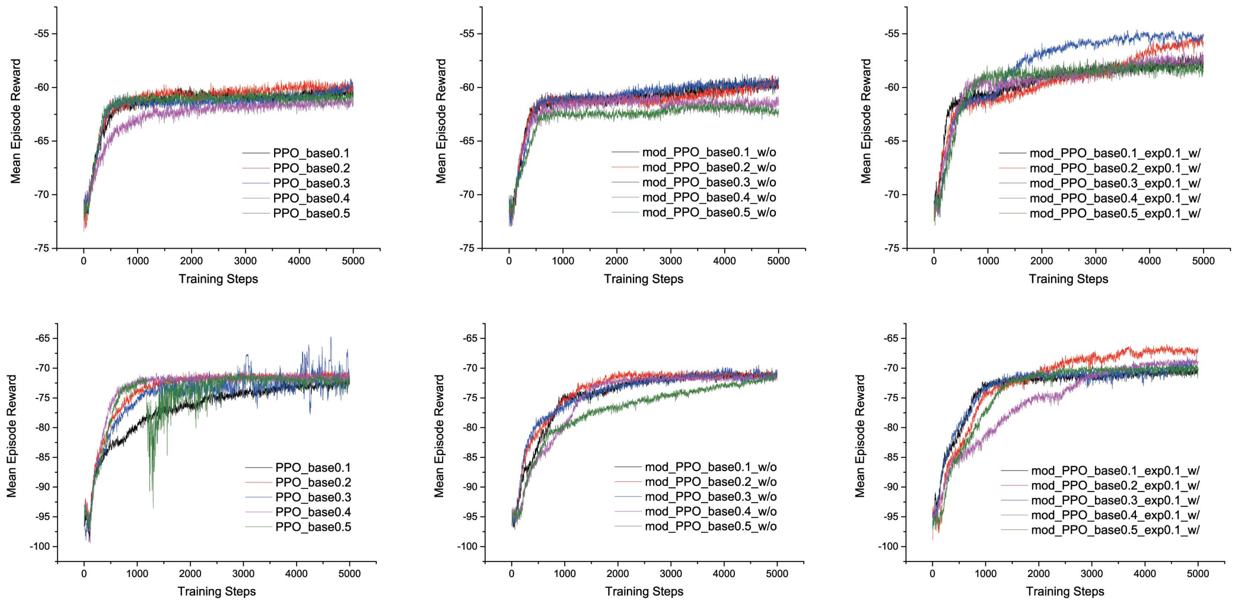


Fig. 10. (a) When vehicle number is 50 and base range is [0.1,0.2,0.3,0.4,0.5], the average reward of original PPO. (b) When vehicle number is 50 and base range is [0.1,0.2,0.3,0.4,0.5], the average reward of Modified PPO without clip hyperparameter exploration. (c) When vehicle number is 50 and base range is [0.1,0.2,0.3,0.4,0.5], the average reward of Modified PPO with clip hyperparameter exploration. (d) When vehicle number is 100 and base range is [0.1,0.2,0.3,0.4,0.5], the average reward of original PPO. (e) When vehicle number is 100 and base range is [0.1,0.2,0.3,0.4,0.5], the average reward of Modified PPO without clip hyperparameter exploration. (f) When vehicle number is 100 and base range is [0.1,0.2,0.3,0.4,0.5], the average reward of Modified PPO with clip hyperparameter exploration.

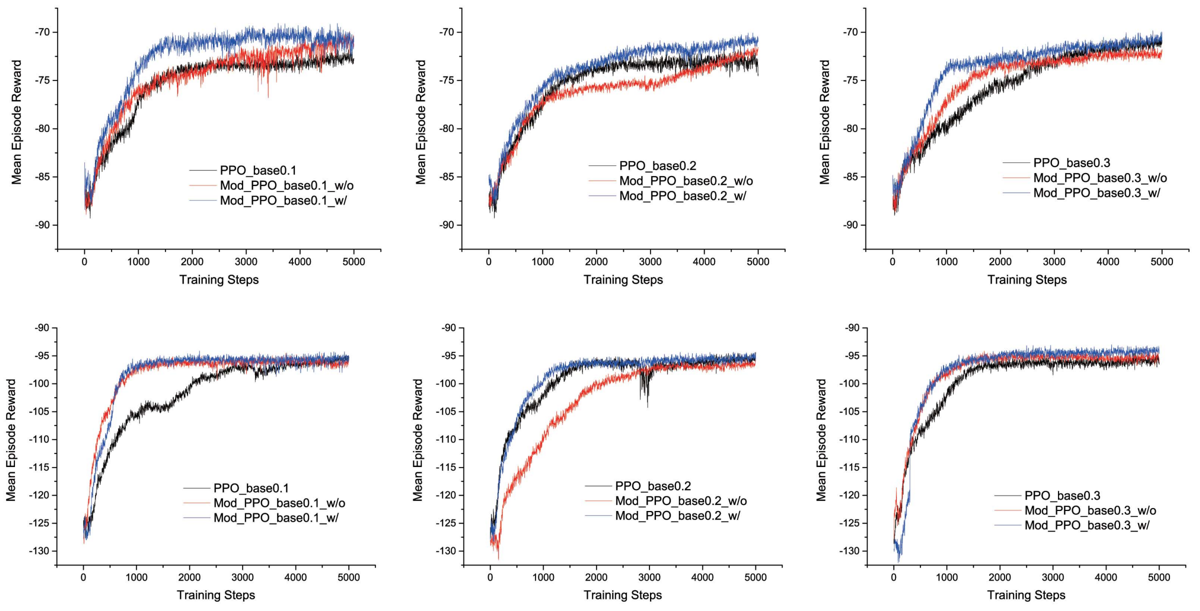


Fig. 11. Top row: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when base is in [0.1,0.2,0.3], vehicle number is 50 and intersection number is 9. Bottom row: The average reward of original PPO, Modified PPO without clip hyperparameter exploration and with clip hyperparameter exploration when base is in [0.1,0.2,0.3], vehicle number is 100 and intersection number is 9.

mechanism is crucial for improving the performance of figuring out traffic congestion. The original PPO, which uses constant clip hyperparameter, are not capable of adaptively following the situation and get a negative outcome. Although Modified PPO without clip hyperparameter exploration adjusts clip hyperparameter as the situation changes, the method is easily stuck in local optimum and also achieves a negative outcome. Modified PPO with clip hyperparameter exploration not only follows the situation but also has the capacity to breaking away from local optimum. This is the reason why Modified PPO with clip hyperparameter exploration achieves the best performance.

Furthermore, the multi-agent smart grid system is very significant and meaningful. Compared to a simulated grid system with only one intersection and a few vehicles, our multi-agent smart grid system with numerous vehicles is closer to the real traffic situation. In this system, vehicles and traffic lights assist each other in a comprehensive viewpoint. On one hand, vehicles are capable of adaptively adjusting their speed to shorten the time near intersections and to reach intersections at the right moment. On the other hand, traffic lights intelligent change the state to reduce the probability of vehicles waiting for a red light.

V. Conclusion

In this article, we proposed the Modified PPO algorithm, which is appropriate as the traffic control scheme of SD-IoT, to help reduce traffic jams and applied it to a multi-agent smart grid system which includes multiple intersections and dozens of vehicles. Our proposed algorithm tightens the limitation of the distance between the new policy and current policy by contemplating the speed of vehicles and the iteration number. This move takes key indicators and model training progress into account, which builds the general trends that β hyperparameter gradually decreases and makes model training progress more stable. At the same time, our approach loosens the limitation with a small probability. Therefore, our approach avoids performance degradation and gets rid of the local optimum. This move makes it more possible that our proposed algorithm finds better policy to relieve traffic congestion in a complex multi-agent smart grid system.

In future work, we plan to adjust the reward function to improve the policy of agents more intelligent. We design to consider more factors, such as the distance between vehicles and intersections, to let the reward function more inclusive and nearer to the real world. Besides, to make the proposed algorithm more suitable for SD-IoT, the possibility of adding other useful mechanisms to the Modified PPO algorithm will be explored. What's more, we will extend the multi-agent smart grid system. Specific measures include further increasing the number of intersections and vehicles to suit the wave of urbanization and expanding the state space and action space of vehicles and traffic lights to simulate more complicated situations. In addition to the above plans, we prepare to add intrinsic reward terms to the objective formula to make the algorithm explore in sparse reward task better. What's more, to further increase adaptivity of proposed algorithm, we plan to consider more factors during the process of calculating hyperparameters. At last, introducing energy function is a good way to make policy function fit multimodal function better.

References

[1] S. Misbahuddin, J. A. Zubairi, A. Saggaf, J. Basuni, S. A-Wadany, and A. Al-Sofi, "IoT based dynamic road traffic management for smart cities," in *Proc. 12th Int. Conf. High-Capacity Opt. Netw. Enabling/Emerg. Technol. (HONET)*, Dec. 2015, pp. 1–5.

[2] R. Munoz *et al.*, "IoT-aware multi-layer transport SDN and cloud architecture for traffic congestion avoidance through dynamic distribution of IoT analytics," in *Proc. Eur. Conf. Opt. Commun. (ECOC)*, Sep. 2017, pp. 1–3.

[3] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[4] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring R-CNN," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6409–6418.

[5] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 734–750.

[6] S. Chen, Y. Liu, X. Gao, and Z. Han, "Mobilefacenets: Efficient CNNs for accurate real-time face verification on mobile devices," in *Proc. Chin. Conf. Biometric Recognit.*, 2018, pp. 428–438.

[7] D. Sáez Trigueros, L. Meng, and M. Hartnett, "Enhancing convolutional neural networks for face recognition with occlusion maps and batch triplet loss," *Image Vis. Comput.*, vol. 79, pp. 99–108, Nov. 2018.

[8] Z. Yuan, Z. Lyu, J. Li, and X. Zhou, "An improved hybrid CTC-attention model for speech recognition," 2018, *arXiv:1810.12020*. [Online]. Available: http://arxiv.org/abs/1810.12020

[9] Z. Xiao, Z. Ou, W. Chu, and H. Lin, "Hybrid CTC-attention based end-to-end speech recognition using subword units," in *Proc. 11th Int. Symp. Chin. Spoken Lang. Process. (ISCSLP)*, Nov. 2018, pp. 146–150.

[10] B. Sun, L. Yang, P. Dong, W. Zhang, J. Dong, and C. Young, "Super characters: A conversion from sentiment classification to image classification," 2018, *arXiv:1810.07653*. [Online]. Available: http://arxiv.org/abs/1810.07653

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*. [Online]. Available: http://arxiv.org/abs/1810.04805

[12] C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, 1989.

[13] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Proc. Adv. Neural Inf. Process. Syst.*, 1996, pp. 1038–1044.

[14] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[15] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Sci. Robot.*, vol. 4, no. 26, p. 5872, 2019.

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

[17] F.-L. Li *et al.*, "AliMe assist: An intelligent assistant for creating an innovative E-commerce experience," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 2495–2498.

[18] S. M. Shortreed, E. Laber, D. J. Lizotte, T. S. Stroup, J. Pineau, and S. A. Murphy, "Informing sequential clinical decision-making through reinforcement learning: An empirical study," *Mach. Learn.*, vol. 84, nos. 1–2, pp. 109–136, Jul. 2011.

[19] E. Van der Pol and F. A. Oliehoek, "Coordinated deep reinforcement learners for traffic light control," in *Proc. Learn., Inference Control Multi-Agent Syst.*, 2016, pp. 1–4.

[20] J. Gao, Y. Shen, J. Liu, M. Ito, and N. Shiratori, "Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network," 2017, *arXiv:1705.02755*. [Online]. Available: http://arxiv.org/abs/1705.02755

[21] H. Abdelgawad, B. Abdulhai, S. El-Tantawy, A. Hadayeghi, and B. Zvaniga, "Assessment of self-learning adaptive traffic signal control on congested urban areas: Independent versus coordinated perspectives," *Can. J. Civil Eng.*, vol. 42, no. 6, pp. 353–366, Jun. 2015.

[22] J. Jin and X. Ma, "Adaptive group-based signal control by reinforcement learning," *Transp. Res. Procedia*, vol. 10, pp. 207–216, Dec. 2015.

[23] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2017, pp. 6379–6390.

[24] S. Iqbal and F. Sha, "Actor-Attention-Critic for multi-agent reinforcement learning," 2018, *arXiv:1810.02912*. [Online]. Available: http://arxiv.org/abs/1810.02912

[25] OpenAI *et al.*, "Dota 2 with large scale deep reinforcement learning," 2019, *arXiv:1912.06680*. [Online]. Available: http://arxiv.org/abs/1912.06680

[26] O. Vinyals *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347

[28] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[29] N. Gude *et al.*, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.

[30] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *Proc. OSDI*, vol. 10, 2010, pp. 1–6.

[31] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proc. Internet Netw. Manage. Conf. Res. Enterprise Netw.*, 2010, pp. 1–4.

[32] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proc. 1st workshop Hot topics Softw. defined Netw.*, 2012, pp. 19–24.

[33] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, 2010, pp. 1–6.

[34] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87–95, Jul. 2014.

[35] P. Berde *et al.*, "Onos: Towards an open, distributed sdn os," in *Proc. 3rd workshop Hot topics Softw. defined Netw.*, 2014, pp. 1–6.

[36] I. Ku, Y. Lu, M. Gerla, R. L. Gomes, F. Ongaro, and E. Cerqueira, "Towards software-defined VANET: Architecture and services," in *Proc. 13th Annu. Medit. Ad Hoc Netw. Workshop (MED-HOC-NET)*, Jun. 2014, pp. 103–110.

[37] K. Zheng, L. Hou, H. Meng, Q. Zheng, N. Lu, and L. Lei, "Soft-defined heterogeneous vehicular network: Architecture and challenges," *IEEE Netw.*, vol. 30, no. 4, pp. 72–80, Jul./Aug. 2016.

[38] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

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

[40] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[42] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[43] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*. [Online]. Available: http://arxiv.org/abs/1905.11946

[44] I. Sutskever, O. Vinyals, and Q. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

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

[46] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, 1988.

[47] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602

[48] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1–13.

[49] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," 2015, *arXiv:1511.06581*. [Online]. Available: http://arxiv.org/abs/1511.06581

[50] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: http://arxiv.org/abs/1509.02971

[51] G. Barth-Maron *et al.*, "Distributed distributional deterministic policy gradients," 2018, *arXiv:1804.08617*. [Online]. Available: http://arxiv.org/abs/1804.08617

[52] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, *arXiv:1802.09477*. [Online]. Available: https://arxiv.org/abs/1802.09477

[53] T. Haarnoja *et al.*, "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*. [Online]. Available: http://arxiv.org/abs/1812.05905

[54] Y. Wang, H. He, X. Tan, and Y. Gan, "Trust region-guided proximal policy optimization," 2019, *arXiv:1901.10314*. [Online]. Available: http://arxiv.org/abs/1901.10314

[55] D. Ye *et al.*, "Mastering complex control in MOBA games with deep reinforcement learning," 2019, *arXiv:1912.09729*. [Online]. Available: http://arxiv.org/abs/1912.09729

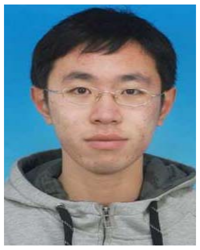
[56] M. Abdoos, N. Mozayani, and A. L. C. Bazzan, "Holonic multi-agent system for traffic signals control," *Eng. Appl. Artif. Intell.*, vol. 26, nos. 5–6, pp. 1575–1587, 2013.

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

[58] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M Bayen, "Flow: A modular learning framework for autonomy in traffic," 2017, *arXiv:1710.05465*. [Online]. Available: http://arxiv.org/abs/1710.05465

[59] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*. [Online]. Available: http://arxiv.org/abs/1506.02438

Jiachen Yang (Member, IEEE) received the M.S. and Ph.D. degrees in communication and information engineering from Tianjin University, Tianjin, China, in 2005 and 2009, respectively. He is currently a Professor with Tianjin University. He was also a Visiting Scholar with the Department of Computer Science, School of Science, Loughborough University, U.K., and the Department of Electrical, Computer, Software, and Systems Engineering, Embry-Riddle Aeronautical University, USA. His research interests include image quality evaluation stereo vision research, pattern recognition, and virtual reality.



Jipeng Zhang received the B.S. degree in communication and information from Tianjin University, Tianjin, China, in 2015, and the M.S degree in communication and information engineering from Beijing Normal University, Being, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Information and Communication Engineering, Tianjin University. His research interests include reinforcement learning and game analysis.

Huihui Wang (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Virginia, Charlottesville, VA, USA, in 2013. In 2011, she was an Engineering Intern with Qualcomm, Inc. In 2013, she joined the Department of Engineering, Jacksonville University, Jacksonville, FL, USA, where she is also an Assistant Professor and the Founding Chair with the Department of Engineering. Her research interests include cyber-physical systems, the Internet of Things, healthcare and medical engineering based on smart materials, robotics, haptics based on smart materials/structures, ionic polymer metallic composites, and MEMS.