Resource Provisioning for Mitigating Edge DDoS Attacks in MEC-Enabled SDVN

Yuchuan Deng, Hao Jiang, *Member, IEEE*, Peijing Cai, Tong Wu, Pan Zhou, *Senior Member, IEEE*, Beibei Li, *Member, IEEE*, Hao Lu, *Member, IEEE*, Jing Wu, *Member, IEEE*, Xin Chen, and Kehao Wang, *Member, IEEE*

***Abstract*—Vehicular ad hoc network (VANET) has become an accessible technology for improving road safety and driving experience, the problems of heterogeneity and lack of resources it faces have also attracted widespread attention. With the development of software-defined networking (SDN) and multiaccess edge computing (MEC), a variety of resource allocation strategies in MEC-enabled software-defined networking-based VANET (SDVN) have been proposed to solve these problems. However, we note that few of these work involves the situation where SDVN is under Distributed Denial of Service (DDoS) attacks. Actually, Internet of Things (IoT) devices are extremely easy to be compromised by malicious users, and compromised IoT devices may be used to launch edge DDoS attacks against the MEC servers in MEC-enabled SDVN at any time. In this article, we propose a graph neural network (GNN)-based collaborative deep reinforcement learning (GCDRL) model to generate the resource provisioning and mitigating strategy. The model evaluates the trust value of the vehicles, formulates mitigation of edge DDoS attacks and resource provisioning strategies to ensure that the MEC servers can work normally under edge DDoS attacks. In addition, GNN is adopted in the DRL model to extract the structure feature of the graph composed of MEC servers, and help transfer computing tasks between MEC servers to alleviate the problem of resources imbalance between them. Experimental results show that the method of estimating the vehicular trust value is effective, and our method can make the average throughput of edge nodes more stable and lower down the average delay and the average energy consumption under the edge DDoS attack. Also, a real-world case study is conducted to verify our conclusion.**

***Index Terms*—Deep reinforcement learning, edge Distributed Denial of Service (DDoS) attack, graph neural networks (GNNs), multiaccess edge computing (MEC), software-defined networking-based vehicular ad hoc network (SDVN).**

Manuscript received 11 April 2022; accepted 1 July 2022. Date of publication 25 July 2022; date of current version 21 November 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U19B2004, and in part by the Open Funding Projects of the State Key Laboratory of Communication Content Cognition under Grant 20K05 and Grant A02107. The work of Pan Zhou was supported by the National Natural Science Foundation of China (NSFC) under Grant 61972448. The work of Beibei Li was supported in part by the National Natural Science Foundation of China under Grant 62002248; in part by the Sichuan Youth Science and Technology Innovation Team under Grant 2022JDTD0014; and in part by the Sichuan Science and Technology Program under Grant 2022YFG0193. The work of Kehao Wang was supported by the National Natural Science Foundation of China (NSFC) under Grant 62172313. *(Corresponding author: Jing Wu.)*

Please see the Acknowledgment section of this article for the author affiliations.

Digital Object Identifier 10.1109/JIOT.2022.3189975

I. Introduction

In RECENT years, the concept of the Internet of Things (IoT) has received more and more attention, one of the hotspots is the vehicular ad hoc network (VANET). With the pervasive use of smart devices and advances in the development of wireless access technologies, VANET has become an accessible technology for improving road safety and transportation efficiency [1]. However, the features of VANET also bring new challenges. For example, the heterogeneity of VANET makes it tricky to manage, the high mobility of vehicles makes message propagation in VANET quite challenging, and it is also a challenge to provide the required services with limited resources of vehicles. Therefore, a number of research have been conducted to solve these problems. To deal with the heterogeneity of VANET and to promote its scalability, software-defined networking-based VANET (SDVN) [2] was proposed, which has the characteristics of software-defined networking (SDN), i.e., decoupling of the control plane and data plane [3], [4]. Plus, to cope with the dynamics of VANET communication topology, mobility prediction [5] was proposed, which can greatly reduce the latency of communication in VANET. Moreover, in order to break through the constraint of limited resources, multiaccess edge computing (MEC) architecture [6] has been applied to VANET to provide additional computing and storage resources near vehicles. Furthermore, plenty of research on finding the optimal method to allocate resources in MEC have been conducted [7]–[9].

Nonetheless, we noted that most of the research about resources provisioning for MEC-enabled SDVN ignore potential security risks in the system [10]. In MEC-enabled SDVN, MEC servers provide additional resources near the vehicles and reduce the burden of the backbone network, which greatly reduces transmission latency and network congestion. However, MEC servers are also vulnerable to Distributed Denial-of-Service (DDoS) attacks due to the limitation of resources. So, it is important to improve the robustness of MEC servers and make them work properly even under extreme situations, such as DDoS attacks, as security is one of the top topics of VANET [1]. Actually, the total number of DDoS attacks has increased rapidly in recent years [11], [12], and IoT devices play an important role in it [13]–[16].

Although there are many methods for mitigating DDoS attacks, most of them cannot be directly transplanted to

MEC-enabled SDVN. For centralized servers, one can redirect all the network traffic to a traffic filtering center for cleansing, or simply increase hardware resources of servers to defend against DDoS attacks. However, for distributed MEC servers, one cannot increase the computing and storage resources they carried to defend against DDoS attacks due to the constraints of their sizes and economic costs. It is also impractical to redirect the network traffic from users under the coverage of MEC servers to a traffic filtering center because of latency. Therefore, new solutions must be proposed to defend against DDoS attacks near MEC servers [17]–[19].

For the scenario with strict latency requirements and limited energy supply, trust value evaluation is lightweight and efficient, which is widely adopted in many fields, like privacy protection, resource allocation, and access control. The trust value is the degree of certainty with which the received information is accepted and acted upon [20], it can be calculated according to some related factors, such as the frequency of task requests, past interactions of evaluated users with the evaluation nodes, etc. The abnormal behavior of users can be restricted according to their trust values. Recently, the trust value evaluation of vehicles has been proposed to improve the security of VANET. Some methods [21], [22] prioritize the trust value and perform different strategies on users with different priorities. Inspired by them, we evaluate the trust value of vehicles and mitigate DDoS attacks based on it, which not only prevents edge DDoS attacks launched by malicious vehicles at the source but also optimizes the results of resource allocation.

In this article, we focus on the issue of computing resources provisioning and edge DDoS attack mitigating in MEC-enabled SDVN [18]. Toward securing MEC-enabled SDVN, we evaluate the trust value of all vehicles first, and make offloading decisions of vehicular computing tasks based on that. Then, the computing tasks offloaded to MEC servers may also be transferred between MEC servers to further alleviate the imbalance of resources and service requests brought by edge DDoS attacks.

Specifically, we formulate resource allocation and mitigation in MEC-enabled SDVN under edge DDoS attacks as an optimization problem, and propose a graph neural network (GNN)-based collaborative deep reinforcement learning (GCDRL) algorithm to solve it, which does not require any prior knowledge and will automatically learn the optimal strategy by interacting with the environment. As a matter of fact, considering that the Quality of Experience (QoE) [23] constraint of VANET is stringent, and the energy available for MEC servers and vehicles is limited, the objective of the proposed algorithm is to minimize the latency and energy consumption of all computing offloading and transferring tasks. The method will determine the proportion of computing task of the vehicle is executed locally or offloaded to the edge node. It will also determine whether to transfer an offloaded task to another target MEC server.

In the proposed MEC-enabled SDVN architecture, each edge node is responsible for formulating and executing a local offloading strategy. However, there must be more than one edge node, and an edge node always connects to some other nodes. Therefore, the connection between edge nodes should be considered. Since the relationship between edge nodes is in the form of graph, we use GNNs to model the relationship between edge nodes and obtain the structure feature of them, which is helpful in understanding resource distribution and making transferring decisions between MEC servers. We may use edge nodes and MEC servers alternately in the rest of this article.

Our contributions are listed as follows.

1) We propose a novel MEC-enabled SDVN system architecture featuring computing offloading with edge DDoS attack mitigation. The architecture evaluates the trust value of vehicles and takes it into consideration during resource provisioning, and transfers computing tasks between MEC servers to balance computing resources and requests.
2) We formulate the resource allocation and mitigation in MEC-enabled SDVN as an optimization problem to minimize the latency and energy consumption of all computing offloading and transferring tasks.
3) We propose an integrated algorithm, namely, GCDRL, to obtain a solution to the formulated optimization problem. Among it, the convolution neural network (CNN) and GNN are used to extract vehicular spatial features and structure features of edge nodes, respectively. In addition, long short-term memory (LSTM) is used to extract the temporal features of the entire state space.
4) We conduct experiments to evaluate the performance of the proposed algorithm, and compare it with other methods in terms of throughput, latency, and energy consumption. The experimental results show that GCDRL method reduces the adverse effects brought by edge DDoS attacks effectively.

The remainder of this article is organized as follows. Section II introduces background and related work. The system models and problem formulation are presented in Sections III and IV, respectively. In Section V, we propose a DRL-based resource provisioning algorithm which is combined with the GNN-based feature extraction method. The experiments and result analysis are presented in Section VI. Finally, we conclude this article and discuss future work in Section VII.

II. Background and Related Work

A. Resource Allocation in MEC

MEC provides extra computing and storage resources near end users, and avoids data transmission in the backbone network, which is an effective supplement to cloud computing [24]–[31]. In MEC-enabled VANET, the computing tasks of vehicles can be partly offloaded to MEC servers to alleviate the problem of insufficient resource in vehicles [23]. However, it is also challenging to develop an efficient offloading strategy in VANET with high node mobility, dynamic network topology, and changing vehicle density.

There are mainly two ways to offload computing tasks, one is to offload computing tasks of vehicles to MEC servers, and the other may also offload tasks to neighboring users

with free resources. We mainly focus on the former one. Mao *et al.* [32] proposed an algorithm based on Lyapunov optimization to find the optimal solution for calculating the offloading strategy, and the energy harvesting is taken into account, but the algorithm is only suitable for single-user scenarios. Chen *et al.* [33] proposed a multiuser computing offloading strategy based on game theory by minimizing the latency and energy consumption.

In the aforementioned work, a computing task is either offloaded to a MEC server for computing, or computed locally. However, recent popular offloading methods usually offload part of a task to MEC servers, and compute the other part locally, so as to promote flexibility and efficiency of resources allocation. Min *et al.* [34] used an algorithm based on reinforcement learning to dynamically adjust the offloading strategy according to the power of the device, and the author also used transfer learning to speed up training. You *et al.* [21] formulated the server resource allocation in computing offloading as a convex optimization problem, and then derived an offload priority function, and implemented different offloading strategies according to the priority of users. Wang *et al.* [35] injected dynamic voltage scaling technology into computing offloading, and used a univariate search technique to find the local optimal solution, which can significantly reduce the latency and energy consumption. However, the above methods are all based on the known models and are not suitable for networks with rapidly changing traffic conditions, so the performance of them are not stable in the dynamically changing VANET.

B. Deep Reinforcement Learning in VANET

Reinforcement learning has now been effectively applied to obtain optimal strategies for network entities. Among them, Q-learning is a typical model-free reinforcement learning algorithm, which is different from both supervised learning and unsupervised learning. In Q-learning, the Q-values corresponding to all state and action pairs are stored in a Q-table. However, in complex and large-scale networks, such as VANET, the state space, and action space are usually too large to process, and the reinforcement learning algorithm cannot find the optimal strategy in a reasonable time. Inspired by literature [36], we use DRL to solve the problem of complex network optimization. In DRL algorithm, all Q-values are obtained through adaptive learning by deep neural networks, without the need for Q-tables to save Q-values [37].

In the field of MEC [38], computation offloading has become a research hotspot. They focus on when, where, and how to transfer computing tasks from the end to MEC servers. Tong *et al.* [39] proposed the algorithm using the DRL method to determine whether the task needs to be offloaded and allocates computing resources for the task, but the model is focused on the objects with low mobility and not suitable for the VANET environment that vehicle topology changes frequently. Liu *et al.* [40] modeled the computation offloading problem as a semi-Markov decision process (MDP), and adopted a deep Q-network (DQN) to find the policy of computation offloading and resource allocation. He *et al.* [41] proposed a framework to allocate resources in SDN with DRL technique, but the article lacks consideration of energy-efficiency issues. Compared with the above work, we use DRL to obtain the resource allocation policy in VANET under DDoS attacks.

C. Security Issues of SDVN

The combination of SDN and VANET overcomes some limitations of VANET, but SDVN still faces some security threats in terms of availability, confidentiality, and data integrity. Attacks that SDVN may suffer include replay attack, sybil [42], [43], sinkhole, malware injection, forgery, and DDoS. Among the attacks, the DDoS attack is one of the most severe attacks that threaten the network availability of SDVN, which is launched by multiple malicious nodes in a distributed way, and will consume a lot of network bandwidth and other resources of the victim host [44].

Many researchers focus on detecting DDoS attacks by analyzing network traffic. For example, Cao *et al.* [45] proposed a DDoS detection and mitigation method based on the spatial–temporal graph convolutional network, which can accurately detect the path that the DDoS attack flows pass through. Kolandaisamy *et al.* [46] proposed a multivariate stream analysis approach to detect and mitigate DDoS attacks, aiming at the problems of low accuracy and high computational overhead of existing DDoS detection techniques. However, these methods cannot be directly applied to highly dynamic VANET with unknown environmental conditions. De Biasi *et al.* [47] proposed a defense mechanism to detect the DDoS flooding attack by time-series analysis of network flow and mitigate the attack by creating a flow tree to find the source of spoofed packets. It focuses on attacks on vehicles without considering the safety of infrastructure, such as roadside units (RSUs) and the SDN controller, while the single-center structure is prone to the single point of failure. There are also some methods to prevent DDoS attacks through resource allocation. He *et al.* [18] modeled edge DDoS mitigation as a constraint optimization problem, and proposed an optimal approach and a novel game-theoretical approach for mitigating the attack. Liu *et al.* [48] modeled the strategic resource allocation problem between the attacker and the defender as a Bayesian Q-learning game, and proposed a greedy Q-learning algorithm to dependably allocate resources against DDoS attacks. However, the method proposed by He and Liu cannot prevent the attack at the source, and the resource consumption for mitigating the DDoS attack is still high.

Toward securing SDVN, many researchers tried to establish a trust model in SDVN. Li *et al.* [49] proposed a trust management algorithm to constrain and standardize the behavior of vehicles, and used blockchain to implement the data security of vehicles. Zhao *et al.* [22] associated a resource allocation problem with the trust value of vehicles. After calculating the trust value of vehicles, more resources would be assigned to vehicles with high credibility when SDN services are provided. The simulation results indicated that the trust management architecture can improve security in SDVN and balance the network load. Compared with the previous work, our research

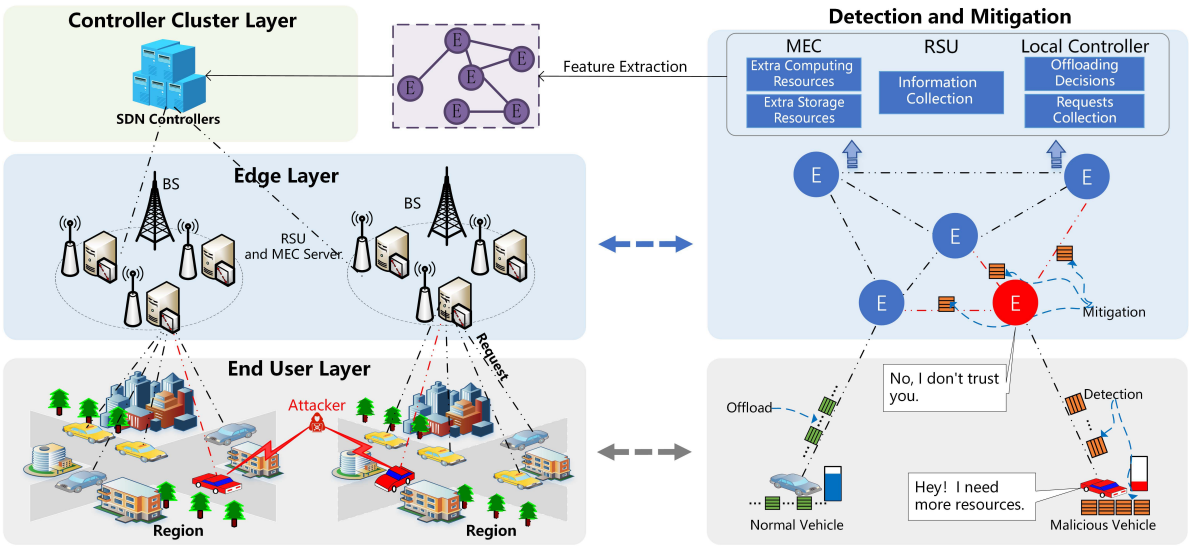


Fig. 1. Illustration of the MEC-enabled SDVN architecture.

not only assesses trust value of vehicles to prevent attack but also performs the mitigation algorithm to allocate computing resources reasonably.

III. System Model

A. Network Model

Our system framework is based on a MEC-enabled SDVN, which is composed of three layers, namely, controller cluster layer, edge layer, and end user layer, as shown in Fig. 1.

Among them, the controller cluster is located in the cloud, and is used to collect global network information, such as a network topology. The controller cluster collects the state of each MEC server from local controllers located in the edge layer, extracts the connection relationship and other features of edge nodes by analyzing the information it collects, and finally delivers the results to local controllers.

The edge layer is composed of edge nodes, including base stations (BSs), local controllers, RSUs, and MEC servers. In this work, the local controller, RSU, and MEC server are integrated, and we may use integrated edge node or integrated MEC server to refer it. In the edge layer, regions are divided according to geographical locations. In each region, there are one BS and multiple integrated edge nodes to provide services for users within this region. RSUs are responsible for collecting road information, including location, direction, speed of vehicles, etc. Local controllers are responsible for resource provisioning and task scheduling. MEC servers provide extra computing and storage resources for vehicles. When computing resources on the end user layer are not sufficient, the local controller collects requests and makes task offloading decisions. In the proposed collaborative DRL algorithm, each integrated edge node is regarded as an *agent*, which coordinates with other agents to make the optimal resource provisioning decision.

The end-user layer is composed of IoT devices such as vehicles. Users in the end-user layer may have various needs. End users perform computation locally. While local resources are not sufficient, end users will send requests to the edge node. In particular, when a vehicle is detected as a malicious vehicle, its trust value decreases, and tasks are offloaded to the edge nodes less frequently.

B. Threat Model

There are two sources of DDoS attacks on edge servers: 1) the Internet and 2) local IoT devices (e.g., vehicles). We mainly discuss DDoS attacks launched by malicious vehicles, i.e., edge DDoS attacks [18]. We assume that the DDoS attack is launched by one or more malicious vehicles within the coverage of edge nodes. The devices launching the edge DDoS attack are with legitimate IDs which means they can pass the identification and authentication of the system. In addition, malicious vehicles will hide themselves by forging IP addresses and other methods, so it is not easy to identify them by IP addresses. When launching an attack, malicious vehicles send a large number of fake service requests to run out the resources of MEC servers, so that they cannot respond to legitimate service requests within the maximum tolerable time.

Now, we formally represent the process of edge DDoS attacks. Let $\mathcal{S}_e = \{s_1, s_2, \ldots, s_M\}$ denote the set of MEC servers, where M is the number of them. Suppose the size of all tasks that s_j can process at time t theoretically, i.e., computing capacity of s_j, is $X_{s_j}^{c,t}$, and there are n vehicles connected to it which can be denoted as $V^t = \{v_1, v_2, \ldots, v_n\}$. The set of offloading requests from V^t to s_j at time t is $\mathcal{O}^t = \{O_1^t, O_2^t, \ldots, O_n^t\}$. When the computing offloading requests of vehicles at time t are much greater than the computing capacity available on s_j, that is

$$\sum_{v_i \in V^t} O_i^t >> X_{s_j}^{c,t} \tag{1}$$

then, we think that the MEC server s_j is suffering an edge DDoS attack. In this case, a mass of tasks that cannot be

processed in time will enter the task buffer and wait in line. In particular, it is possible that a subset of the vehicles $V^{\text{sub},t} = \{v_1, v_2, \ldots, v_m\}$ which contains the least vehicles but has a significantly larger computing offloading requests than other vehicles, that is

$$\sum_{v_i \in V^{\text{sub},t}} O_i^t >> \sum_{v_i \in (V^t - V^{\text{sub},t})} O_i^t \tag{2}$$

in this case, the vehicles in V^{sub} are regarded as *malicious vehicles*.

C. Computation Model

Vehicles in SDVN provide various applications for drivers and passengers, such as autonomous driving, navigating, and path planning, which require a lot of computing resources, and the resources required may not be satisfied by vehicles themselves. In this case, vehicles would like to offload some computing tasks to the MEC servers. In this section, we formally represent the latency and energy consumption brought by computing.

Assuming that time is slotted, and for computing task O_i^t, the maximum tolerable delay is τ^p. There are two options for O_i^t in our computing model, processing it locally or offloading it to a MEC server for processing. We use $u_i^t \in [0, 1]$ to represent the offloading rate of O_i^t, i.e., the proportion of computing task O_i^t to be offloaded to the MEC server s_j. The details of local computing and offloading computing are described as follows.

1) Local Computing Model: If we decide to process task O_i^t locally, it will use the computation resource of the vehicle. The latency and energy consumption for the local computing are presented as follows.

Latency: For a vehicle, assuming that the number of CPU cycles required by computing 1-byte data is χ_c, and the clock speed, i.e., computation capability in CPU cycles per second, of CPU is f_v, the time required for computing task O_i^t is given by

$$L_i^{l,t} = \chi_c \frac{\left(O_i^t\left(1 - u_i^t\right)\right)}{f_v}. \tag{3}$$

Energy Consumption: Assuming δ_v is the energy consumption efficiency coefficient of the vehicle, of which the value depends on the CPU architecture. Then, the energy consumption per CPU cycle is $\delta_v (f_v)^2$ [50], [51]. So, the total energy consumption is expressed by

$$E_i^{l,t} = \delta_v (f_v)^2 \chi_c \left(O_i^t\left(1 - u_i^t\right)\right) = \delta_v (f_v)^3 L_i^{l,t}. \tag{4}$$

2) Offloading Computing Model: In this case, part of the task O_i^t is offloaded to the MEC server s_j. Edge node s_j provides computing services for the vehicles under its coverage, and it is very vulnerable to edge DDoS attacks. Therefore, the latency and energy consumption of computation offloading tasks need to be handled carefully.

Latency: The latency of offloading computing $L_i^{o,t}$ comes from three parts, namely, uploading, queuing, and computing, and it can be expressed as

$$L_i^{o,t} = L_i^{u,t} + L_i^{q,t} + L_i^{c,t} \tag{5}$$

where $L_i^{u,t}$, $L_i^{q,t}$, and $L_i^{c,t}$ represent uploading latency, queuing latency, and task computing latency of task O_i^t, respectively. The details of obtaining them are given as follows.

1) *Uploading Latency:* We first discuss the SNR before calculating the uploading latency. Each vehicle is connected with an edge node through a wireless channel, we use $\mathcal{Y}^t = \{y_1^t, y_2^t, \ldots, y_n^t\}$ to represent the set of wireless channels that vehicles used to connect with s_j. For the channel y_i at time t which is working normally, the SNR of it is

$$\text{SNR}_i^t = \frac{P_v h_i^t}{\sigma} \tag{6}$$

where P_v is the transmission power of the vehicle, and h_i^t and σ are the channel power gain and noise power of the channel y_i at time t, respectively. We use r_i^t to denote the maximum transmission rate of channel y_i at time t. According to the Shannon formula, r_i^t can be calculated as

$$r_i^t = w_i^t \log_2\left(1 + \text{SNR}_i^t\right) \tag{7}$$

where w_i^t is the transmission bandwidth of the upload channel. Thus, we can express the uploading latency for task O_i^t as

$$L_i^{u,t} = \frac{\left(O_i^t u_i^t\right)}{r_i^t}. \tag{8}$$

2) *Queuing Latency:* Let b_q denote the total number of CPU cycles that task O_i^t waits in the task buffer, we can get the queuing latency

$$L_i^{q,t} = \frac{b_q}{f_e} \tag{9}$$

where f_e is the clock speed of the MEC server. Obviously, when an edge node suffers an edge DDoS attack, the queuing latency of offloading tasks will greatly increase, which will affect the total latency of computation offloading tasks.

3) *Computing Latency:* Similar to (3), the time required for a MEC server to compute the task can be calculated as

$$L_i^{c,t} = \frac{\left(O_i^t u_i^t\right)\chi_c}{f_e}. \tag{10}$$

For an offloading computing task, the size of output is much smaller than the size of input, and the transmission power of RSUs is stronger than vehicles, so the time to send the result of computing task O_i^t to the vehicle can be ignored.

Energy Consumption: Similar to latency, the total energy consumption also comes from the aforementioned three parts, which can be expressed as

$$E_i^{o,t} = P_v L_i^{u,t} + P_c L_i^{q,t} + \delta_e (f_e)^3 L_i^{c,t} \tag{11}$$

where P_v is the transmission power of vehicle, P_c is the static circuit power of the MEC server, and δ_e is the energy consumption efficiency coefficient of the MEC server.

D. Mitigation Model

When a MEC server suffers edge DDoS attacks or other situations where a short-term surge of service requests appears, the computing and storage resources of the victim MEC server may not be enough to handle all the service requests. As a result, the latency and energy consumption of the victim server will increase dramatically, and the network services it provides may even be unavailable. In order to reduce the adverse effects caused by edge DDoS attacks, we propose a mitigation strategy when provisioning resources. Our mitigation model is shown on the right side of Fig. 1. On the one hand, we evaluate the trust value to rate each vehicle based on the behavior of it to prevent edge DDoS attacks at the source. Vehicles with the low trust value will be at a disadvantage when provisioning MEC computing resources which will be elaborated in Section IV. On the other hand, the edge node being attacked may transfer some requests to other edge nodes with idle resources to further alleviate the imbalance of resources between MEC servers. The details of trust value evaluation and transferring are presented as follows.

1) Estimation of Vehicular Trust Value: When assessing the trust value of a vehicle, we mainly consider the frequency of offloading requests initiated by the vehicle, and the computing resources of the MEC server consumed by the vehicle in the past time λ. For vehicle v_i under the coverage s_j, the trust value of it at time t is

$$\kappa_i^t = e^{-\left(\beta_1 \kappa_i^{f,t} + \beta_2 \kappa_i^{c,t}\right)} \tag{12}$$

where κ_i^t is the trust value of v_i at time t, and $\kappa_i^{f,t}$ and $\kappa_i^{c,t}$ are related to the frequency of offloading requests and consumption of computing resources in the past time λ, respectively. β_1 and β_2 are two weighted factors, which satisfy $\beta_1, \beta_2 \geq 0$ and $\beta_1 + \beta_2 = 1$. The trust value κ_i^t is in the range of 0–1. The closer κ_i^t is to 0, the more likely that v_i is a malicious vehicle. The details of calculating $\kappa_i^{f,t}$ and $\kappa_i^{c,t}$ are given as follows.

1) *Frequency of Offloading Requests:* One way for malicious vehicles to launch an edge DDoS attack is sending a large number of offloading requests in a short time. So, we consider offloading request frequency as one of the key points to estimate the vehicular trust value. Obviously, the higher the offloading request frequency of v_i, the larger the likelihood of v_i being malicious, and the lower the trust value of it should be. Assuming that the total number of offloading requests sent by v_i in time interval λ is N_i^t. $\kappa_i^{f,t}$ is calculated as

$$\kappa_i^{f,t} = 1 - \zeta^{N_i^t/\lambda} \tag{13}$$

where $\zeta \in (0, 1)$ is the frequency factor.

2) *Consumption of Computing Resources:* In order to exhaust the resources of a MEC server and make it deny legitimate service requests, malicious vehicles are likely to consume far more MEC resources than legitimate vehicles, and they may also ask for computing resources beyond the capacity of a MEC server. Therefore, we consider the MEC resource consumptional rate of a vehicle when estimating the vehicular trust value, and try to avoid the situation where the resources of a MEC server are consumed too much for one vehicle. The trust value component related to resources consumption rate of v_i in time interval λ is

$$\kappa_i^{c,t} = \frac{O_i^{\lambda} \chi_c}{f_e \lambda} \tag{14}$$

where O_i^{λ} is the total computing offloading tasks requested by v_i in time interval λ.

2) Load Balancing Between MEC Servers: Tasks transferring between MEC servers may further alleviate the imbalance of resources, and decrease the latency and energy consumption brought by queuing. However, transfer will also bring extra latency and energy consumption. Therefore, transferring is meaningful only if the latency and energy consumption brought by it are lower than queuing.

For the task O_i^t, the transferring latency of it is denoted by $L_i^{m,t} (L_i^{m,t} < \tau^m)$ where τ^m is the maximum tolerable latency of transferring a task, and the energy consumption of transferring task O_i^t is denoted by $E_i^{m,t}$. If task O_i^t is not transferred, then both the transferring latency and energy consumption of it are 0. Otherwise, the latency and energy consumption of O_i^t are related to the distance it is transferred, which is represented by the number of hops $h_{(j,k)}$ between the source s_j and destination s_k. Also, it is related to the change of queuing. The transferring latency can be expressed as

$$L_i^{m,t} = \frac{u_i^t \cdot O_i^t \cdot h_{(j,k)}}{r_j^t} + \frac{b_{q,k} - b_{q,j}}{f_e} \tag{15}$$

where r_j^t is the transmission rate between MEC servers, $b_{q,j}$ is the task buffer size of source, and $b_{q,k}$ is the task buffer size of destination. The energy consumption of transferring can be expressed as

$$E_i^{m,t} = P_e \cdot \frac{u_i^t \cdot O_i^t \cdot h_{(j,k)}}{r_j^t} + P_c \cdot \frac{b_{q,k} - b_{q,j}}{f_e} \tag{16}$$

where P_e is the transmit power of MEC servers.

It is worth noting that the movement of vehicles may cause handover between RSUs, which may also cause transferring of tasks between different MEC servers, but it is beyond the scope of this work, so we do not discuss it here. Important notations presented in this article are summarized in Table I

IV. Problem Formulation of Resource Provisioning Under Edge DDoS Attack

We formulate the computation offloading and the edge DDoS attack mitigating as a joint optimization problem. The decision-making process is essentially an MDP [52]. We define the MDP as a five-tuple $\{S, a_i\ (i \in S), q, \gamma, r\}$, where S represents the state space, a_i represents the action, and q represents the transition probability between two different states. γ represents the discount factor. The greater the value of γ is, the larger the importance of future reward. Plus, r represents the corresponding reward. Then, a DRL algorithm is adopted to find the optimal policy. As shown in Fig. 2, each integrated edge node represents an agent in the DRL model and chooses the offloading and mitigating decision a_t according to the state S_t of the current environment at each time

TABLE I
NOTATIONS

Notation	Description
$\mathcal{S}_e$	The set of MEC servers.
V^t	The set of vehicles connected to MEC server s_j at time t.
$\mathcal{Y}^t$	The set of wireless channels connecting s_j and vehicles at time t.
O^t	The set of offloading computing requests to be offloaded to s_j at time t.
u_i^t	Offloading rate of task O_i^t.
$L_i^{l,t}$	Latency of computing task O_i^t locally.
$E_i^{l,t}$	Energy consumption of computing task O_i^t locally.
$L_i^{o,t}$	Latency of offloading task O_i^t.
$E_i^{o,t}$	Energy consumption of offloading task O_i^t.
$L_i^{m,t}$	Latency of transferring task O_i^t to another MEC server.
$E_i^{m,t}$	Energy consumption of transferring task O_i^t to another MEC server.
κ_i^t	Trust value of vehicle v_i at time t.
τ^p / τ^m	The maximum tolerable latency for a computing task / transferring task.
S^t	The state space at time slot t.
a^t	The action space at time slot t.
r_t	The reward at time slot t.
V^t	The features of vehicles at time slot t.
G_S^t	The undirected graph of edge nodes at time slot t.

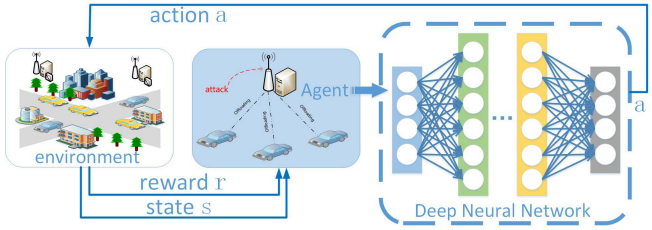


Fig. 2. Deep reinforcement learning model for resource provisioning under the edge DDoS attack.

step. Then, the corresponding feedback, i.e., reward r_t, will be given by the environment. This process will be repeated to maximize the cumulative rewards, that is, the Q-value, which can be expressed as

$$Q(\mathrm{S},\mathrm{a}) = \mathrm{E}\Big[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | S_t = S, a_t = a\Big] \quad (17)$$

and the agents will find the optimal policy eventually by continuously trial and error like this.

Our objective is to find the optimal policy of offloading and mitigating when the network is under edge DDoS attacks, that is, minimizing the latency and energy consumption of all tasks at time t while satisfying the constraints of maximum tolerable latency and bandwidth, which can be expressed as

$$\begin{aligned}
\min_{A} \quad & \sum_{i=1}^{N}\left(\frac{1}{\alpha_i^t}\left(L_i^{l,t}+E_i^{l,t}\right)+\alpha_i^t\left(L_i^{o,t}+E_i^{o,t}\right)+L_i^{m,t}+E_i^{m,t}\right) \\
\text{s.t.} \quad & C1: \max\left(L_i^{o,t}, L_i^{l,t}\right) \le \tau^p, \quad \text{if} \quad \kappa_i > \epsilon \\
& C2: L_i^{m,t} \le \tau^m \quad \forall i,t \\
& C3: w_i^t \le w_v \quad \forall i,t \\
& C4: r_j^t \le r_e \quad \forall j,t \\
& C5: \alpha_i^t \ge 1 \quad \forall i,t. \qquad (18)
\end{aligned}$$

In (18), A denotes the decision of offloading and mitigating. α_i^t is a coefficient related to the trust value of v_i, with which the model will be punished if it offloads computing tasks of vehicles with low trust value to the MEC server. $C1$ defines the maximum latency τ^p that can be tolerated by computation offloading tasks of v_i whose trust value is higher than a small value ϵ. We do not limit the computing latency of vehicles with a trust value lower than ϵ, which means the tasks of vehicles with bad behavior may not be completed in time. $C2$ defines the maximum latency τ^m that can be tolerated by transferring tasks. $C3$ represents the offloading bandwidth w_i^t that does not exceed the maximum offloading bandwidth w_v of v_i. $C4$ represents the transferring rate r_j^t that does not exceed the maximum transmission rate r_e of edge nodes.

We adopt a deep reinforcement learning model to solve this problem. However, before discussing the DRL model in detail, we first define the state space, action space, and reward function from the perspective of an integrated edge node, i.e., an agent in the DRL model.

A. State Space

State is a representation of the current environment. We define the state space of MEC server s_j at time t as $S^t = \{V^t, G_S^t\}$. Since we mainly focus on the latency and energy consumption of resource provisioning, we need to focus on the current distance between vehicles and edge servers and the possible distance in the near future, as well as the processing capabilities of both vehicles and edge servers. Thus, the features of vehicles in the coverage of s_j at time t is $V^t = \{X_{lc}^t, X_{sp}^t, X_{dr}^t, \mathcal{O}^t, \mathcal{Y}^t, C^{v,t}, \kappa^t\}$, which includes location $X_{lc}^t \in \mathbb{R}^{N\times 2}$, speed $X_{sp}^t \in \mathbb{R}^N$, direction $X_{dr}^t \in \mathbb{R}^N$, size of tasks to be processed $\mathcal{O}^t \in \mathbb{R}^N$, state of the wireless channel between vehicles and the edge node $\mathcal{Y}^t \in \mathbb{R}^N$, computing capability of vehicles $C^{v,t} \in \mathbb{R}^N$, and the trust value of vehicles $\kappa^t \in \mathbb{R}^N$ etc. $G_S^t = \{S^t, \mathcal{E}_S^t, A_S^t, \mathbf{X}_S^t\}$ denotes the undirected graph of edge nodes, where S^t is the set of all edge nodes which are nodes in graph G_S^t, $\mathcal{E}_S^t$ is the set of connections between edge nodes, i.e., edges in graph G_S^t, $A_S^t \in \mathbb{R}^{M\times M}$ is the adjacency matrix of all edge nodes at time t, and $\mathbf{X}_S^t = \{\mathcal{O}^{e,t}, C^{e,t}\}$ is the features of all edge servers, including the size of tasks to be processed of each server $\mathcal{O}^{e,t} \in \mathbb{R}^M$, and the computing capability of MEC servers $C^{e,t} \in \mathbb{R}^M$.

B. Action Space

The action space of MEC server s_j at time t can be expressed as $a^t = \{a^{p,t}, a^{m,t}\}$, where $a^{p,t} \in \mathbb{R}^N$ is offloading decision vector, which denotes the offloading rate of all the vehicular computing tasks under the coverage of s_j, and $a_i^{p,t} = 0$ means the computing task of vehicle v_i will be executed locally. $a^{m,t} \in \mathbb{R}^{N\times M}$ is the transferring decision matrix, which maps tasks to be transferred of s_j at time t and corresponding target MEC servers.

C. Reward Function

The definition of reward function is one of the key issues of reinforcement learning, because reward has a guiding effect on the learning of agents. As our objective is to minimize

the latency and energy consumption of all tasks, we use latency and energy consumption as reward. In addition, we assign different weights to the latency and energy consumption of different offloading tasks according to the corresponding vehicular trust value, so as to reduce the offloading of computing tasks coming from vehicles with a low trust value. The reward consists of offloading rewards and mitigating rewards, so the reward can be expressed as

$$r_t = \sum_{i=1}^{N}\Bigg(\frac{1}{\alpha_i^t}\Big(\eta_1 L_i^{l,t} + \eta_2 E_i^{l,t}\Big) + \alpha_i^t\big(\eta_3 L_i^{o,t} + \eta_4 E_i^{o,t}\big) + \eta_5 L_i^{m,t} + \eta_6 E_i^{m,t}\Bigg) \tag{19}$$

$$\alpha_i^t = \max\left(\frac{1}{2\times\kappa_i^t},\ 1\right). \tag{20}$$

α_i^t is a coefficient related to the trust value κ_i^t, with which the model will be punished if it offloads computing tasks of vehicles with low trust value to the MEC server because the offloading latency and energy consumption are amplified, and it will be more inclined to make vehicles with low trust value to compute locally due to the shrink coefficient. It is worth noting that only vehicles with a trust value less than 0.5 will be affected. η_i $(i = 1, 2, 3, 4, 5, 6)$ are coefficients which determine the weight of corresponding part in the reward, and η_1–η_4 are negative which means that the less the energy consumption and latency, the higher the reward given by the environment.

V. Collaborative DRL-Based Resource Provisioning and Mitigating Algorithm

This section mainly introduces the proposed cooperative DRL algorithm for resource provisioning and mitigating edge DDoS attacks. First, we introduce feature extraction of the edge nodes based on GNN and the spatial feature extraction of vehicles based on CNN. Then, with the characteristics of each key information in state space, the principle and specific implementation of the proposed GCDRL algorithm are discussed. Finally, the neural network structure involved is described.

A. GNN-Based Feature Extraction

In order to ensure the efficiency of resource provisioning, we have to obtain the features of edge servers according to state space. Neural network-based methods can extract features effectively from a large amount of data which are in the form of a sequence or grid. However, the interaction between edge nodes is related to their connection, which constitutes a graph. As we treat each edge node as an *agent* and they cooperate with each other to make the optimal decision, we use GNN to extract structural features of the edge node graph to promote collaboration, as shown in Fig. 3. Because compared with the conventional neural network, GNN is more suitable for extracting features from non-Euclidean data like graph [53]–[55].

Therefore, we use GraphSAGE-based GNN [54] to model the graph of edge nodes. For GNN, each vertex aggregates information of itself and its neighbors at first, and then update

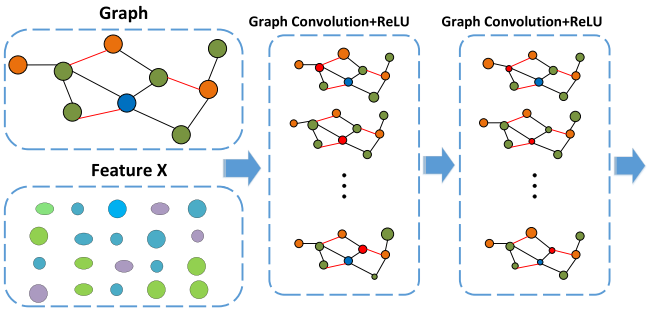


Fig. 3. GNN and its input.

the representation of itself. In this way, each vertex can propagate its own information to neighboring vertices and each vertex can also obtain the information of neighboring vertices. Furthermore, a vertex can obtain information about the vertices that are not directly adjacent to it by iterating the process of aggregate and update. For a GNN model, the aggregate process at the kth layer can be expressed as

$$\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k\Big(\Big\{\mathbf{h}_u^{k-1} \quad \forall u \in \mathcal{N}(v)\Big\}\Big) \tag{21}$$

where $\mathbf{h}_{\mathcal{N}(v)}^k$ is the intermediate representation of aggregation, $\mathcal{N}(v)^k$ is the neighbor of vertex v, $\mathbf{h}_u^{k-1}$ is the representation of vertex u at the k-1th layer, and $\text{AGGREGATE}_k(\cdot)$ is the aggregation function which is MEAN in this work. The update process at the kth layer can be expressed as

$$\mathbf{h}_v^k \leftarrow \sigma\Big(\mathbf{W}^k \cdot \text{CONCAT}\Big(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k\Big)\Big) \tag{22}$$

where $\mathbf{h}_v^{k-1}$ is the representation of vertex v at the k-1th layer, W^k is the learnable weight matrix, $\sigma(\cdot)$ is the activation function, and $\mathbf{h}_v^k$ is the representation of vertex v at the kth layer.

B. GNN-Based Collaborative and Defensive Deep Reinforcement Learning Algorithm

The actor–critic algorithm [56] is a policy-based deep reinforcement learning algorithm. Each agent in the DRL model has an actor network and a critic network. Since our resource provisioning policy is the decision of the offloading rate and mitigating target, the action space contains continuous and discrete actions. Similar to the policy gradient algorithm (PG), the actor network can select appropriate actions in the continuous action space. But the PG is based on round updates which is inefficient. Also, it can achieve single-step updates by using a value-based algorithm as the critic network in the actor–critic algorithm. Specifically, the actor network can be regarded as the executor of the action, and the critic network can be regarded as the referee who scores each action. The executor will choose the action of the next time slot according to the comment given by the referee, and the referee will score the selected action. By modifying the policy like this iteratively, the agent will eventually learn the optimal policy. However, the actor–critic algorithm faces the problem of slow convergence. Fortunately, the deep deterministic policy gradient (DDPG) algorithm [57], which is an improved actor–critic

algorithm, can solve this problem. The network structure of DDPG is similar to that of DQN.

Since the task transfer decision in joint decision making involves transferring tasks to other servers, the entire edge node graph is shared by all agents. Compared with the cooperative strategy of sharing the state of all agents, this way of sharing partial states to achieve coordination and cooperation between multiple agents will not cause dimensional disasters. It is helpful for each agent to make a more appropriate task transfer strategy. Since each agent is located in a different location and environment, they will learn different strategies for their environment, and the partially shared state will also help each agent learn more stably in an unstable environment. Therefore, we call the algorithm proposed in this article the GCDRL Algorithm.

We parameterize the online actor network as θ^{μ} which maps state to action, while the online critic network is parameterized as φ^{c} and it estimates the Q-value. For parameters of the target network, we express them as $\hat{\theta}^{\mu}$ and $\hat{\varphi}^{c}$. The target network is used to calculate the target value, and its update frequency is lower than that of the online network to reduce the correlation of data.

We know that $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$, where $\mathcal{N}_t$ represents random noise, used to balance exploration and exploitation. For the actor, we use function

$$\begin{aligned} J(\theta^{\mu}) &= \mathbb{E}\left[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots\right] \\ &= \int_S \rho^{\mu}(s) Q\left(s, \theta^{\mu}(s)\right) ds \\ &= \mathbb{E}_{s\sim\rho^{\mu}}\left[Q(s, \theta^{\mu}(s)\right] \end{aligned} \tag{23}$$

to measure the performance of the actor network. ρ^{μ} is the probability distribution function of the optimal behavior strategy μ. $Q(s, \theta^{\mu}(s)$ represents the expected return value obtained by using the μ strategy to select the action in the state s. With DDPG, the strategy gradient is

$$\nabla_{\theta^{\mu}} J = \mathbb{E}_{s\sim\rho^{\mu}}\left[\nabla_a Q\left(s, a|\varphi^{c}\right)|_{s=s_i, a=\theta^{\mu}(s_i)} \nabla_{\theta^{\mu}} \mu\left(s|\theta^{\mu}\right)|_{s_i}\right] \tag{24}$$

where the update of actor network parameters depends on the critic network. The critic network instructs the actor network to update the parameters in a direction that maximizes the Q-value, that is, the action a that can make the Q-value larger is selected, so that the policy $\theta^{\mu}(s_i)$ that the actor network learned will be closer and closer to the optimal policy. In the same way, if a vehicle is a malicious attacker, then the vehicle has a high probability of having a lower trust value. According to the definition of the reward function, it can be seen that if the current vehicle trust value is low, the reward can only be maximized by avoiding the task generated by the vehicle to be offloaded to the edge node. Then, the algorithm will be forced to try not to deal with the malicious requests of attackers at the edge node, so as to achieve the effect of defense.

The critic network updates the parameters based on minimizing the loss function which is defined as

$$L\left(s, a|\varphi^{c}\right) = \left(Q\left(s, a|\varphi^{c}\right) - \left(r + \gamma \hat{Q}\left(s', a'|\hat{\varphi}^{c}\right)\right)\right)^2 \tag{25}$$

where s' is the state after s by performing action a, and a' is the action which takes at state s', i.e., $a' = \hat{\mu}(s'|\hat{\theta}^{\mu})$.

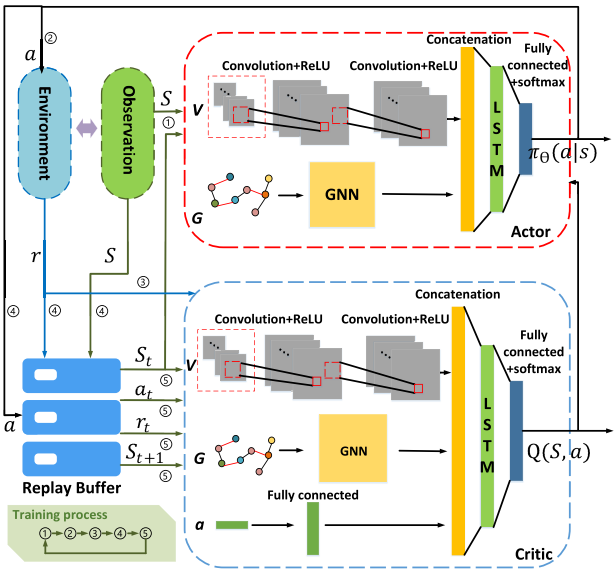


Fig. 4. DDPG algorithm for resource provisioning and defense under the edge DDoS attack.

The target network is a copy of the online network, and the network structures of them are exactly the same. We denote them as $\hat{\mu}(s|\hat{\theta}^{\mu})$ and $\hat{Q}(s, a|\hat{\varphi}^{c})$. In (25), we can see that the target network $\hat{Q}$ is used to compute the value, because using online network Q to update the parameter directly will cause destructive feedback and divergence, while using the target network can guarantee stability and convergence.

For the parameter updating of the online network and the target network, we make the target network slowly copy the parameters of the online network, which can be expressed as

$$\begin{cases} \hat{\theta}^{\mu} = \tau\theta^{\mu} + (1-\tau)\hat{\theta}^{\mu} \\ \hat{\varphi}^{c} = \tau\varphi^{c} + (1-\tau)\hat{\varphi}^{c} \end{cases} \tag{26}$$

where τ $(0 < \tau < 1)$ represents the rate that the target network is updated according to an online network. Algorithm 1 describes the process of the GNN-based collaborative and defensive DRL algorithm for resource provisioning and mitigating.

C. Network Structure

Our DDPG-based resource provisioning model is shown in Fig. 4. It mainly includes the replay buffer, actor network, and critic network. Each agent of our model contains an actor network and a critic network, which have similar network structures. The actor network selects actions $\mu(s|\theta^{\mu})$ based on the environment observation, and the critic network obtains trajectory data $\{s_t, a_t, r_t, s_{t+1}\}$ from the replay buffer for training and generates the scores $Q(s, a|\varphi^{c})$ of actions selected by the actor network. For the actor network, its input S_t includes the vehicle information and edge node information. It first extracts the features of the edge node graph with GNN. For the position information, the trust value of vehicles and other vehicle information, CNN is adopted to extract the spatial features of vehicles to obtain potential spatial associations, then the intermediate results are concatenated as the input of LSTM to

Algorithm 1 GNN-Based Collaborative and Defensive DRL Algorithm for Resource Provisioning and Mitigating

Input: $S_0 = \left\{V^0, G_S^0\right\}$, replay buffer $\mathcal{D}$

Output: $Q(s,a|\varphi^c)$, $\mu(s|\theta^\mu)$, $\hat{Q}(s,a|\hat{\varphi}^c)$, $\hat{\mu}\left(s|\hat{\theta}^\mu\right)$

for *episode* $= 1$ *to* M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  while $t < T$ *and* $S_t \neq$ *terminal* **do**
    Local controllers collect V^t
    $\theta^\mu, s_t \longrightarrow a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$
    Executes $a_t = \left\{a^{p,t}, a^{m,t}\right\} \rightarrow \quad r_t, s_{t+1}$
    $t = t + 1$
    Store $\{s_t, a_t, r_t, s_{t+1}\}$ in $\mathcal{D}$
    Sample a random minibatch of S transitions $\{s_i, a_i, r_i, s_{i+1}\}$ from D
    Update critic network by minimizing $L(s,a|\varphi^c) = \frac{1}{S}\sum_{i=1}^{S}\left(Q(s_i,a_i|\varphi^c) - \left(r_i + \gamma\hat{Q}(s_{i+1},a_{i+1}|\hat{\varphi}^c)\right)\right)^2$
    Update actor network by $\nabla\theta^\mu J = \frac{1}{S}\sum_{i=1}^{S}\nabla Q(s,a|\varphi^c)|_{s=s_i,a=\mu(s_i)}\nabla\theta^\mu\mu(s|\theta^\mu)|_{s_i}$
    Update the target networks by $\begin{cases}\hat{\theta}^\mu = \tau\theta^\mu + (1-\tau)\hat{\theta}^\mu \\ \hat{\varphi}^c = \tau\varphi^c + (1-\tau)\hat{\varphi}^c\end{cases}$
  end while
end for

extract the temporal feature, and the output is passed to a fully connected layer, finally, an action a_t is obtained. Similarly, the structure of the critic network is similar to that of the actor network. However, the input of the critic network has not only state S_t but also action a_t. In the critic network, we use a fully connected layer to extract features of action information, and the structure of other parts of the critic network is basically the same as that of the actor network.

VI. Experiments and Results

A. Experimental Settings

We use simulation of urban mobility (SUMO) to simulate the traffic, and Network Simulator (version 2) (NS2) to simulate the communication of vehicles and edge nodes. In addition, we use Python 3.7 and TensorFlow 1.14.0 to construct the resource provisioning and mitigating model.

For vehicles, the calculation intensity of computing χ_c is set to 800 CPU cycles per byte (cpb), the clock speed f_v is set to 1.5 GHz, the energy consumption coefficient δ_v is set to 10^{-27}, the transmission power P_v is set to 0.1 W, channel power gain is $h_i^t = d_{i,j}^{-\gamma}$, where $d_{i,j}$ is the distance of the vehicle v_i and the edge node s_j, and the path-loss factor γ is set to 4 [58], the Gaussian white noise of the communication between vehicles and edge nodes σ is set to -120 dBm, the transmission bandwidth w_v is set to 1 Mb/s. The maximum tolerable latency τ^p of computing is 80 ms. For edge nodes, the clock speed f_e is set to 5.0 GHz, the static circuit power P_c is set to 0.05 W, the energy consumption coefficient δ_e is set to 10^{-28} [59], the transmission rate r_j^t is set to 50 Mb/s, the transmission power P_e is set to 3 W, and the maximum tolerable latency τ^m of transferring is 0. In addition, the coverage radius of an edge node is set to 300 m. For trust value estimation, the time interval λ is set to 10 s, the frequency factor ζ is set to 0.8, the threshold of trust value ϵ is set to 0.3, and β_1 and β_2 are set to 0.3 and 0.7, respectively. We assume that the experimental scene is a 1 km×1 km square area. Without special settings, there are totally eight integrated edge nodes and 200 vehicles in the area. For a normal vehicle, the probability that it generates a computing offloading request at a time slot is 0.3, and the size of a computing task is uniformly distributed in the range of 1–100 kB, and the time slot is set to 0.1 s. In the training process, the edge DDoS attack is randomly launched, while in the test, an attack lasts for 60 s is launched every 5 min. The main parameters are collected in Table II.

TABLE II
Experimental Parameters

Parameters	Value	Parameters	Value
χ_c	800 cpb	f_v	1.5 GHz
δ_v	10^{-27}	P_v	0.1 W
γ	4	w_v	1 Mb/s
σ	-120 dBm	f_e	5.0 GHz
τ^p	80 ms	δ_e	10^{-28}
P_c	0.05 W	P_e	3 W
r_j^t	50 Mb/s	λ	10 s
ζ	0.8	ϵ	0.3
β_1	0.3	β_2	0.7

We regard the multiple features of vehicle as multiple channels of the input, and the convolution layer contains 32 and 64 filters, respectively, and the size of each filter is 4×4 and 2×2, and the stride is (2,2). In addition, the LSTM layer contains 512 hidden units, and the fully connected layer contains 256 neural units. The learning rates of critic and actor networks are set to 0.01 and 0.001, respectively. The discount factor γ is set to 0.9, the greedy coefficient is set to 0.1, the size of replay buffer $\mathcal{D}$ is set to 1024, and the batch size is set to 64.

B. Evaluation Indicators

1) *Average Reward:* The average reward for each edge node to complete the offloading and mitigating tasks in each episode during the training process.
2) *Average Energy Consumption:* Average energy consumed by the entire system to complete the offloading and mitigating tasks in each time slot.
3) *Average Latency:* Average latency required by each edge node to complete the offloading and mitigating task in each time slot.
4) *Average Throughput:* The average size of offloading computing tasks processed by each edge node in each time slot.

C. Compared Methods

1) Baseline Method:

1) *Random Policy:* The actions are randomly selected at each time slot with random policy.
2) *Greedy Algorithm:* The best action of action space is selected at each time slot with greedy algorithm.
3) *Deep Q-Learning Algorithm [60]:* The DQN algorithm is used to select an action based on the Q-value at

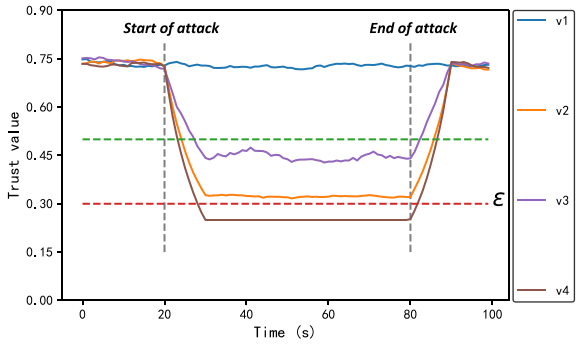


Fig. 5. Results of trust value estimation.

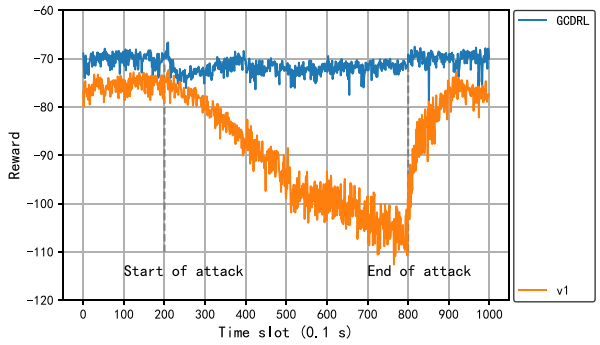


Fig. 6. Defense effect of the attack in scenario 4.

each time slot. We discretize the offloading rate to $\{0, 0.1, 0.2, \ldots, 1\}$ to fit the operation of DQN.

4) *PG [61]:* It selects an action and directly back-propagates by observing information. It uses the reward to directly increase and decrease the probability of selecting an action.

2) Variants of Our Method:

1) *Without Mitigating (v1):* No mitigating strategy is adopted, that is, no DDoS attack is dealt with.
2) *Without LSTM Layer (v2):* The LSTM layer the in actor and critic networks is replaced by a fully connected layer.
3) *Without GNN (v3):* Instead of using GNN to capture the structure feature of edge nodes, only CNN is adopted.

D. Performance Analysis

1) Simulation of Trust Value Estimation: We simulate four different behaviors of vehicles and evaluate the corresponding trust value of them. Specifically, the behaviors of vehicles are as follows.

1) *Normal:* The vehicle generates a computing offloading request with a probability of 0.3 at each time slot, and the task size is uniformly distributed between 1 and 100 kB.
2) *Request Offloading in High-Frequency:* The vehicle randomly generates one or two task requests at each time slot, and the task size is uniformly distributed between 1 and 100 kB.
3) *Request Offloading With Large Task:* The vehicle generates an offloading request with a probability of 0.3 at each time slot, and the task size is fixed to 200 kB.
4) *Request in High-Frequency With Large Task:* The vehicle always generates one task with a size fixed to 100 kB at each time slot.

We perform the simulation 20 times and record the average values. Each simulation lasts for 100 s, all four vehicles behave normally in 0–20 s and 81–100 s. In 21–80 s, however, vehicle #1 behaves normally, vehicle #2 requests offloading in high-frequency, vehicle #3 requests offloading with large task, and vehicle #4 requests offloading in high frequency with a large task. The experimental results are shown in Fig. 5.

Obviously, the trust values of all #2, #3, and #4 drop rapidly below 0.5 after about 10s after the attack started, which means that the behavior of allocating MEC computing resources to them will be punished. Among them, the trust value of #4 drops the most, because it generates computing offloading requests at a high frequency and the requested tasks are very large, which means that it trends to consume more MEC resources. Finally, the trust values of #2–#4 returned to normal levels quickly in about 10 s after the end of the attack.

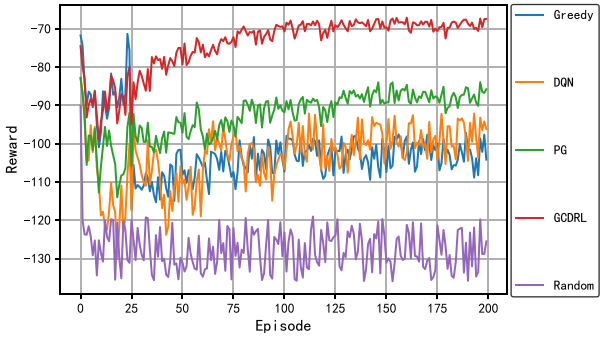


Fig. 7. Reward comparison during the training process (an episode contains about 30 steps).

2) Defensive Effect Against Edge DDoS Attacks: For the fourth scenario mentioned above, which is the most common attack scenario, we conduct a 60-s attack on the GCDRL algorithm and the variant v1 to test its defense effects. The test result is shown in Fig. 6. It can be seen that before the attack, the reward of GCDRL is higher than that of variant v1, and the rewards of these two algorithms are in a stable state. When the attack starts at the 200th time slot, the reward of variant v1 drops sharply and the variance of the reward value also increases significantly. However, the reward of GCDRL only fluctuates and decreases slightly in the 200 to 300 time slots, and then the reward value tends to be stable. Finally, the attack ends at the 800th time slot, and variant v1 begins to recover to the previous reward value slowly. At the same time, the reward value of the GCDRL algorithm has also been slightly improved, and the change is not obvious. Compared with variant v1, it can be seen that the reward of the GCDRL algorithm remains in a stable state during the process of edge DDoS attacks. That is to say, the mitigation strategy in the GCDRL algorithm can effectively defense and alleviate edge DDoS attacks.

3) Reward of Different Models: Fig. 7 shows the change in the average reward of each algorithm in training. The reward

TABLE III
COMPREHENSIVE EVALUATION OF DIFFERENT METHODS

Algorithm	Average Reward	Average latency (ms)	Average energy consumption (mJ)	Convergence time (step)
Random	-120.78 ± 7.99	96.32	58.32	–
Greedy	-105.90 ± 5.86	88.98	55.92	3000
DQN	-99.67 ± 8.01	78.45	50.45	6000
PG	-89.91 ± 4.91	72.55	51.54	4200
GCDRL	**-71.94** ± 4.26	**51.00**	**41.45**	**3800**

TABLE IV
COMPREHENSIVE EVALUATION OF THE VARIANTS

Algorithm	Average Reward	Average latency (ms)	Average energy consumption (mJ)	Convergence time (step)
v1	-87.72 ± 6.71	76.70	42.71	4900
v2	-75.19 ± 5.01	67.90	47.90	3900
v3	-81.89 ± 7.21	58.56	48.57	4000
GCDRL	**-71.94** ± 4.26	**51.00**	**41.45**	**3800**

of the random algorithm has been jittering irregularly, and although the Greedy algorithm can converge faster, the final reward is not good, this is because it converges to a poorly performing local optimal policy. Compared with common DQN and PG algorithms, our algorithm can converge faster, and the final reward is also higher. As shown in Table III, the average reward of our algorithm after convergence is higher by 40.4%, 32.1%, 27.8%, and 20.0% than other algorithms, respectively. In comparison, the main disadvantage of the PG algorithm is that it converges too slowly and the training is longer. The DQN is not stable enough, and the jitter amplitude is large. Part of the reason is that when we use the DQN algorithm, we need to discretize the offloading rate. It can also be seen from Table III that the latency and energy consumption of our algorithm are smaller than the PG algorithm.

In order to prove the effectiveness of the mitigating strategy, LSTM layer, and GNN module, we compare the experimental results of our method with v1–v3, respectively. As shown in Table IV, it is clear that without mitigating, the final reward is as low as −87.72, sometimes, even lower than the PG algorithm, which proves that the mitigating strategy plays a key role in mitigating edge DDoS attacks. Then, without LSTM, the average energy consumption and latency increase 15.6% and 33.1% separately. Because temporal features of the state space cannot be effectively captured without LSTM. Similarly, without GNN, the energy consumption and latency increased 17.2% and 14.8%. As without GNN, the structure feature of the edge node graph cannot be captured which is a loss of useful information in training.

4) Scalability of Different Models: We expand and reduce the scale of the network proportionally, and conduct experiments on networks with different numbers of edge nodes. Other parameters in Section VI-A remain unchanged. Fig. 8 shows the relationship between the average reward of different algorithms and the number of edge nodes.

For the GCDRL algorithm, when the number of edge nodes is larger, there are more choices of target edge servers available for task transfer, so its reward will increase slowly as the number of edge nodes increases, but when the number of edge nodes is greater than 12, the global edge node graph is also bigger, causing slight regress in learning effect and

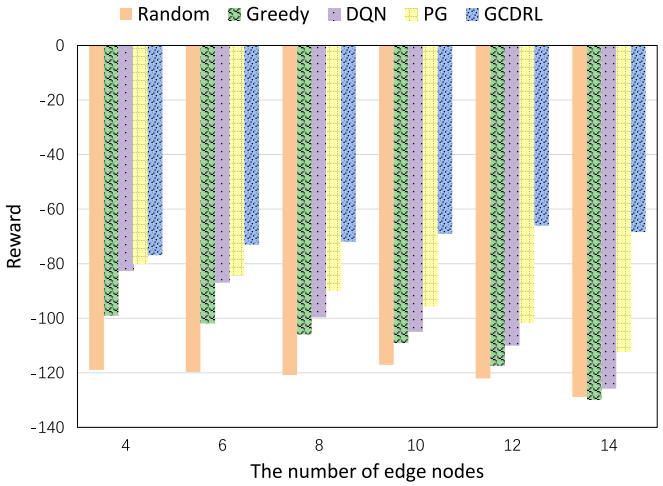


Fig. 8. Comparison of reward with a different number of edge nodes.

reward. For other baseline algorithms, as the number of edge nodes increases, the uncertainty of the environment becomes higher. Because there is neither state sharing nor cooperation between agents, the learning effect is unstable, which leads to the decrease in reward as the number of edge nodes increases. In general, as the number of edge nodes increases, the result of the GCDRL algorithm is more stable and its reward is higher than other baseline algorithms.

We also evaluated our method in scenarios with different numbers of vehicles while the number of edge nodes remains at 8. As shown in Fig. 9, when the number of vehicles is less than 250, the reward of our algorithm will remain relatively stable. However, when the number of vehicles exceeds 300, the reward of our algorithm drops, because the edge nodes can no longer handle so many offloading tasks, even if the mitigating strategy is adopted. For other baseline methods, as the number of vehicles increases, edge nodes cannot handle the increasing tasks, so the reward of these baseline methods decreases accordingly. This proves that our method can mitigate edge DDoS attacks to a certain extent compared with the baseline methods, but the mitigation effect against edge DDoS attacks is not unlimited. Therefore, we need to explore the stability of our method under different attack intensities.

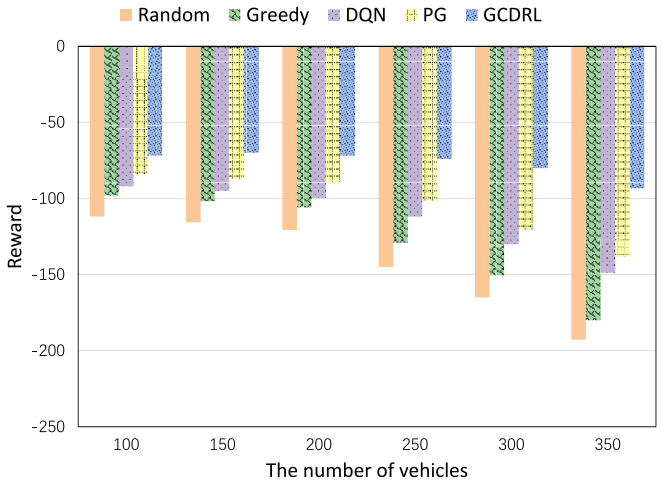


Fig. 9. Comparison of reward with a different number of vehicles.

E. Security Analysis

We conduct security analysis for two attack scenarios, including different numbers of malicious vehicles and different attack duration. Under each scenario, we compare GCDRL with other methods and analyze the mitigation effect of GCDRL. Meanwhile, we analyze the stability of GCDRL under different attack intensities.

1) Security Analysis Under Different Numbers of Malicious Vehicles: We compare the impact of different numbers of malicious vehicles on average throughput, average latency, and average energy consumption. We assume each malicious vehicle generates 1 task with size fixed to 100 kB at each time slot, and the attack lasts for 600 time slots.

The results of average throughput are shown in Fig. 10(a). With the increase of malicious vehicles, the throughput of GCDRL maintains at a steady state around 2.0 Mb per time slot. The throughput of other methods increases first. When the resources of the victim edge nodes are exhausted, the throughput will be maintained in the upper limit around 5.0 Mb per time slot under edge nodes computing capability. However, with our method, most of the tasks of malicious vehicles will compute locally or be discarded because the trust value of malicious vehicles is very low. When the number of malicious vehicles is 60, the throughput using our method is 2.12 Mb per time slot, while the throughput corresponding to other methods are 4.94, 4.95, 4.96, 4.96, and 4.90 Mb per time slot, respectively. At this point, the edge nodes are working properly only if the system adopts our method, otherwise, the edge nodes will be fully loaded or cause a denial of service in severe cases.

As shown in Fig. 10(b), on the one hand, with the increase of malicious vehicles, the average latency of other methods increases. It is worth noting that the average latency of random policy and greedy policy is significantly higher than other models, which are 54.83 and 50.43 ms, respectively, even when there is no attack. It shows that the strategies adopted by these models are far from the optimal resource provisioning policy. On the other hand, due to the use of mitigation strategies, when the number of malicious vehicles increases, the average latency of our method maintains at a steady state around 50 ms. When the number of malicious vehicles is 60, the average latency of the system using our method is 52.46 ms, while the average latency corresponding to other methods are 282.50, 287.50, 334.16, 357.50, and 278.33 ms, respectively.

The results of average energy consumption are shown in Fig. 10(c). As the number of malicious vehicles increases, the energy consumption of all methods gradually increases, because malicious vehicles send a large number of malicious requests, and the edge nodes need to continuously compute these useless tasks if the system adopts other methods while mitigation strategy of GCDRL requires additional energy consumption to deal with resources scarcity. When there is no attack, the average energy consumption of all methods is 29.4, 30.5, 32.2, 36.9, 37.6, and 29.9 mJ, respectively. It can be seen that the energy consumption of our method is always lower than that of the other models, because models with mitigation will refuse to offload most of the computing tasks of vehicles with low trust values to the MEC server. But on the other hand, as the number of malicious vehicles increases, our model has to transfer tasks between edge nodes more and more frequently, which obviously consumes additional energy.

2) Security Analysis Under Different Attack Duration: We compare the impact of average throughput, average latency, and average energy consumption. We assume there are totally 40 malicious vehicles, each of which generates 1 task with size fixed to 100 kB at each time slot and the total duration of the attack is 1500 time slots.

As shown in Fig.11(a), as the duration of attack increases, the throughput corresponding to other methods first rises during the first 100 time slots of the attack. After 100 time slots, other methods almost reach the upper limit under edge nodes computing capability. In GCDRL, attacks from malicious vehicles will not result in a surge in throughput because of the mitigation strategy, the throughput will maintain at a steady state around 2.1 Mb per time slot. When the attack lasts for 1500 time slots, the throughput of all models are 2.06, 4.74, 4.94, 4.83, 4.96, and 4.68 Mb per time slot, respectively.

The results of average latency are shown in Fig. 11(b). The average latency of other methods without the mitigation strategy increases with the attack duration, and the computing capacity of edge nodes reaches the upper limit after the 100th time slot. On the one hand, tasks cannot be transferred. On the other hand, the attacking vehicles are still continuously sending offload task requests. The server cannot handle so many tasks, resulting in the increasing queuing latency. Among them, the performance of random policy and greedy policy is the worst. The average latency of them increases from 54.89 and 50.43 ms to 280.55 and 275.69 ms, respectively, when the attack lasts for 1500 time slots. The difference is that the average latency of our model is stable, which is around 50 ms, while the average latency of other models continuously grows. Although the process of mitigating will introduce extra latency, the latency caused by queuing is reduced at the same time.

As shown in Fig. 11(c), as the duration of the attack increases, the average energy consumption of the baseline methods gradually increases. Because the victim edge nodes

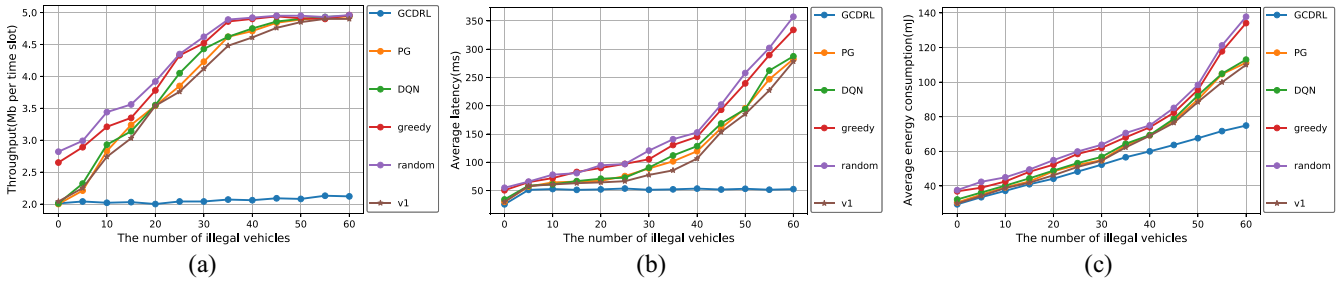


Fig. 10. Relationship between the number of malicious vehicles and indicators. (a) Throughput. (b) Average latency. (c) Average energy consumption.

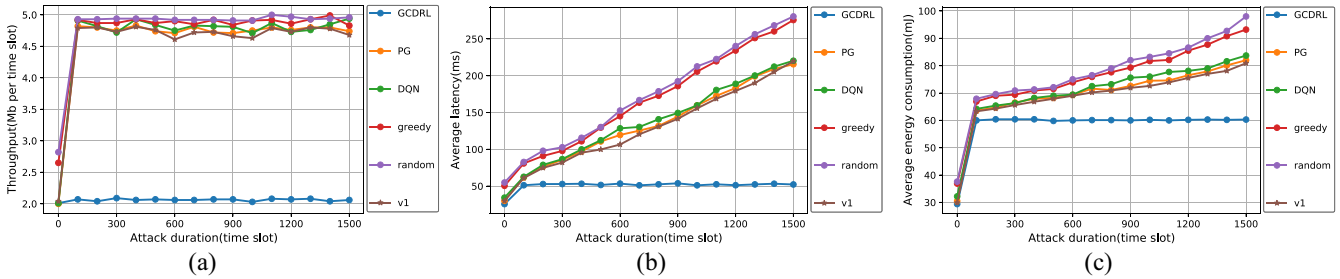


Fig. 11. Relationship between the attack durations and indicators. (a) Throughput. (b) Average latency. (c) Average energy consumption.

need to continuously process the useless service requests sent by malicious vehicles, which consumes a lot of energy. The longer the attack duration, the more useless requests are received; thus, the more energy is consumed. It can be seen that the energy consumption of our method is always lower than that of the baseline models and maintains around 60 mJ. When the duration of the attack is 600 time slots, the average energy consumption corresponding to our method and the compared methods are 60.0, 69.3, 69.4, 73.9, 75.0, and 69.0 mJ, respectively. After 600 time slots, the energy consumption of the random policy and greedy policy increases faster than the baseline methods. The average energy consumption of these two methods is up to 98.0 and 93.2 mJ, respectively, at 1500 time slots.

F. Case Study in the Real World

In order to verify the effectiveness of our method in the real scenario, we build a small VANET with eight unmanned vehicles, three PCs, and a Wi-Fi router. Each vehicle is equipped with a Raspberry Pi 4B, which is equipped with a 1.5-GHz quad-core 64-bit ARM Cortex-A72 chip, 1GB LPDDR4 SDRAM, and an onboard dual-band 802.11ac wireless network card. We use PCs as the MEC servers, which is equipped with Intel Core i7-7700HQ CPU (four physical cores, each core running at 2.8–3.8 GHz) and 16-GB RAM. In the experiment, each vehicle drives along a preset route, as shown in Fig. 12. The PCs, which are regarded as edge servers, communicate with the vehicles through the Wi-Fi router. The vehicles will generate computation offloading requests, and the algorithm will match these requests with the best execution strategy. We deploy a computation offloading method on both the vehicles and the PCs to support edge computing. Specifically, we set three small cars as malicious vehicles and five cars as normal users. A malicious vehicle will generate a

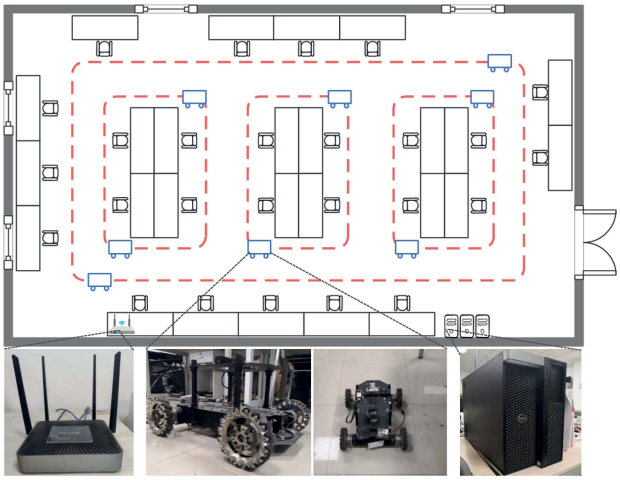


Fig. 12. Scenario in a real-world case.

computing offloading request at each time slot and the size of a task is 300 kB. The probability of normal cars generating a computing offloading request at a time slot is 0.4, and the size of a task is uniformly distributed in the range of 200–300 kB. We evaluate the performance, i.e., average throughput, average latency, and average energy consumption of each edge node, of different methods under different attack duration.

The results are shown in Fig. 13. It can be seen that our method outperforms the other methods. The throughput of all methods will be maintained in a stable state. The throughput of baseline methods almost reaches the upper limit around 3.0 Mb per time slot under edge nodes computing capability. However, the throughput of our method stabilizes at around 1.0 Mb per time slot because of the mitigation strategy. When the attack lasts for 3600 time slots, the throughput of all methods are 2.99, 2.95, 2.93, 2.80, and 0.98 Mb per

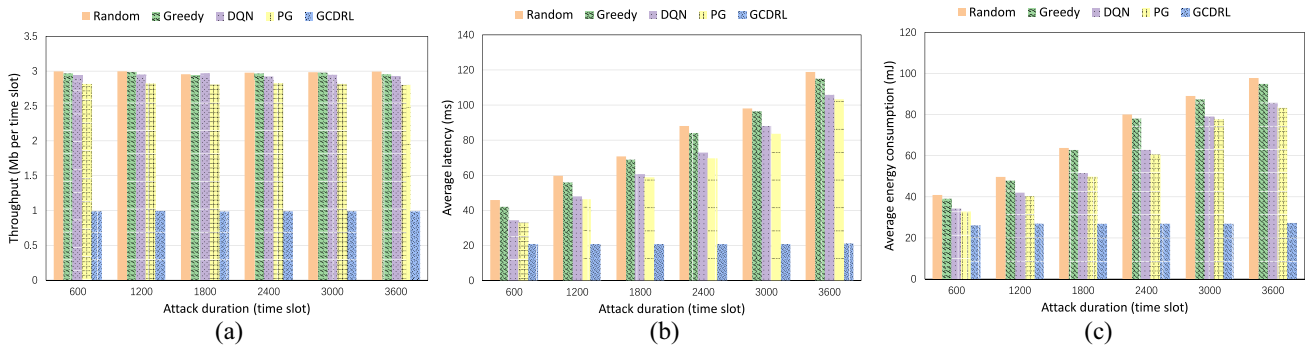


Fig. 13. Results of the real-world case study. (a) Throughput. (b) Average latency. (c) Average energy consumption.

time slot, respectively. As for the average latency, the difference of the average latency between the baseline and our method gradually increase from 600 time slots to 3600 time slots. When the attack lasts for 3600 time slots, the average latency of all methods are 118.83, 114.98, 105.83, 103.3, and 21.9 ms, respectively. The trend of average energy consumption is similar to the trend of average latency. The average energy consumption of baseline methods increases over time because the victim edge nodes need to continuously process the useless service requests sent by malicious vehicles. The energy consumption of our method is always lower than that of other methods and maintains around 27 mJ. In general, our algorithm is more stable when the network is under edge DDoS attacks in a real scenario.

VII. Conclusion

MEC-enabled SDVN is vulnerable to DDoS attacks. To cope with it, we proposed a collaborative DRL-based method to jointly allocate computing resources and mitigate the edge DDoS attack in MEC-enabled SDVN. Specifically, we first evaluated the trust value of all vehicles and make offloading decisions of vehicular computing tasks based on that. Then, the computing tasks offloaded to MEC servers may also be transferred between MEC servers, which can further alleviate the imbalance of resources and service requests. Experimental results showed that our algorithm is effective in mitigating edge DDoS attacks, and it outperforms the other algorithms in performance indicators, such as latency, energy consumption, and server throughput. In particular, in order to verify the effectiveness of our method in a real scenario, we built a small VANET with eight unmanned vehicles, this experiment once again proves the efficiency of the GCDRL algorithm. In the future, we will consider offloading computing tasks to other vehicles with idle resources, then the mobility of vehicles and the dynamics of communication must be considered more comprehensively, and the corresponding DDoS attack should also be considered.

Acknowledgment

Yuchuan Deng, Hao Jiang, Peijing Cai, and Jing Wu are with the Electronic Information School, Wuhan University, Wuhan 430072, Hubei, China (e-mail: dengyc@whu.edu.cn; jh@whu.edu.cn; caipeijing@whu.edu.cn; wujing@whu.edu.cn).

Tong Wu is with the School of Electrical Information and Communication Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: m201971975@hust.edu.cn).

Pan Zhou is with the Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China (e-mail: panzhou@hust.edu.cn).

Beibei Li is with the School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China (e-mail: libeibei@scu.edu.cn).

Hao Lu is with the Information Technology Office, Air Force Early Warning Academy, Wuhan 430019, China (e-mail: haolu@foxmail.com).

Xin Chen is with Air Force Early Warning Academy, Wuhan 430015, Hubei, China (e-mail: cxin917@126.com).

Kehao Wang is with the Hubei Key Laboratory of Broadband Wireless Communication and Sensor Networks, Wuhan University of Technology, Wuhan 430070, China (e-mail: kehao.wang@whut.edu.cn).

References

[1] W. B. Jaballah, M. Conti, and C. Lal, "Security and design requirements for software-defined VANETs," *Comput. Netw.*, vol. 169, Mar. 2020, Art. no. 107099.

[2] 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. Mediterr. Ad Hoc Netw. Workshop (MED-HOC-NET)*, 2014, pp. 103–110.

[3] D. Mu, P. Zhou, Q. Li, R. Li, and J. Xu, "Privacy-preserving MEC-enabled contextual online learning via SDN for service selection in IoT," in *Proc. 16th Int. Conf. Mobile Ad-Hoc Smart Syst. (IEEE MASS)*, 2019, pp. 1–9.

[4] T. Wu *et al.*, "Joint traffic control and multi-channel reassignment for core backbone network in SDN-IoT: A multi-agent deep reinforcement learning approach," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 231–245, Jan.–Mar. 2021.

[5] Y. Tang, N. Cheng, W. Wu, M. Wang, Y. Dai, and X. Shen, "Delay-minimization routing for heterogeneous VANETs with machine learning based mobility prediction," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3967–3979, Apr. 2019.

[6] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 94–100, Jul. 2017.

[7] Y. He, D. Zhai, R. Zhang, J. Du, G. S. Aujla, and H. Cao, "A mobile edge computing framework for task offloading and resource allocation in UAV-assisted VANETs," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–6.

[8] B. Liu, C. Liu, and M. Peng, "Resource allocation for energy-efficient MEC in NOMA-enabled massive IoT networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 4, pp. 1015–1027, Apr. 2021.

[9] F. Fang, Y. Xu, Z. Ding, C. Shen, M. Peng, and G. K. Karagiannidis, "Optimal resource allocation for delay minimization in NOMA-MEC networks," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7867–7881, Dec. 2020.

[10] L. Hou, L. Lei, K. Zheng, and X. Wang, "A Q -learning-based proactive caching strategy for non-safety related services in vehicular networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4512–4520, Jun. 2019.

[11] "Cyber Threats & Trends: Securing Your Network Pandemic-Style." 2021. [Online]. Available: https://www.cio.com/resources/214249/cyber-threats-trends-securing-your-network-pandemic-style

[12] "DDoS in the Time of COVID-19." 2021. [Online]. Available: https://www.imperva.com/resources/resource-library/reports/ddos-in-the-time-of-covid-19/

[13] M. Antonakakis *et al.*, "Understanding the mirai botnet," in *Proc. 26th USENIX Security Symp. (USENIX Security)*, 2017, pp. 1093–1110.

[14] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, "A survey of IoT-enabled cyberattacks: Assessing attack paths to critical infrastructures and services," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3453–3495, 4th Quart., 2018.

[15] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, "Towards IoT-DDoS prevention using edge computing," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–8.

[16] X. Chen, L. Xiao, W. Feng, N. Ge, and X. Wang, "DDoS defense for IoT: A Stackelberg game model-enabled collaborative framework," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9659–9674, Jun. 2022.

[17] X. Tan, H. Li, L. Wang, and Z. Xu, "Global orchestration of cooperative defense against DDoS attacks for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2019, pp. 1–6.

[18] Q. He *et al.*, "A game-theoretical approach for mitigating edge DDoS attack," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2333–2348, Aug. 2022.

[19] D. Gong *et al.*, "Practical verifiable in-network filtering for DDoS defense," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2019, pp. 1161–1174.

[20] R. Hussain, J. Lee, and S. Zeadally, "Trust in VANET: A survey of current solutions and future research opportunities," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2553–2571, May 2021.

[21] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[22] N. Zhao, H. Wu, and X. Zhao, "Consortium blockchain-based secure software defined vehicular network," *Mobile Netw. Appl.*, vol. 25, no. 1, pp. 314–327, 2020.

[23] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020.

[24] X. Wang, J. Wang, X. Zhang, X. Chen, and P. Zhou, "Joint task offloading and payment determination for mobile edge computing: A stable matching based approach," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 12148–12161, Oct. 2020.

[25] Z. Cao, P. Zhou, R. Li, S. Huang, and D. O. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, Jul. 2020.

[26] Z. Hu *et al.*, "Joint offloading and charge cost minimization in mobile edge computing," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 205–216, 2020.

[27] L. Chen, C. Shen, P. Zhou, and J. Xu, "Collaborative service placement for edge computing in dense small cell networks," *IEEE Trans. Mobile Comput.*, vol. 20, no. 2, pp. 377–390, Feb. 2021.

[28] P. Zhou, W. Chen, S. Ji, H. Jiang, L. Yu, and D. Wu, "Privacy-preserving online task allocation in edge-computing-enabled massive crowdsensing," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7773–7787, Oct. 2019.

[29] P. Zhou, K. Wang, J. Xu, and D. Wu, "Differentially-private and trustworthy online social multimedia big data retrieval in edge computing," *IEEE Trans. Multimedia*, vol. 21, no. 3, pp. 539–554, Mar. 2019.

[30] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2018, pp. 207–215.

[31] Z. Xu, L. Zhou, S. C.-K. Chau, W. Liang, Q. Xia, and P. Zhou, "Collaborate or separate? distributed service caching in mobile edge clouds," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2020, pp. 112–121.

[32] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[33] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[34] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1930–1941, Feb. 2019.

[35] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

[36] Y. He, F. R. Yu, N. Zhao, H. Yin, and A. Boukerche, "Deep reinforcement learning (DRL)-based resource management in software-defined and virtualized vehicular ad hoc networks," in *Proc. 6th ACM Symp. Develop. Anal. Intell. Veh. Netw. Appl.*, New York, NY, USA, 2017, pp. 47–54. [Online]. Available: https://doi.org/10.1145/3132340.3132355

[37] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[38] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[39] Z. Tong, X. Deng, F. Ye, S. Basodi, X. Xiao, and Y. Pan, "Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment," *Inf. Sci.*, vol. 537, pp. 116–131, Oct. 2020.

[40] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.

[41] Y. He, F. R. Yu, N. Zhao, V. C. M. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.

[42] G. Guette and B. Ducourthial, "On the sybil attack detection in VANET," in *Proc. IEEE Int. Conf. Mobile Adhoc Sens. Syst.*, 2007, pp. 1–6.

[43] B. Yu, C.-Z. Xu, and B. Xiao, "Detecting sybil attacks in VANETs," *J. Parallel Distrib. Comput.*, vol. 73, no. 6, pp. 746–756, 2013.

[44] A. K. Malhi, S. Batra, and H. S. Pannu, "Security of vehicular ad-hoc networks: A comprehensive survey," *Comput. Security*, vol. 89, Feb. 2020, Art. no. 101664. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404818312872

[45] Y. Cao, H. Jiang, Y. Deng, J. Wu, P. Zhou, and W. Luo, "Detecting and mitigating DDoS attacks in SDN using spatial-temporal graph convolutional network," *IEEE Trans. Dependable Secure Comput.*, early access, Sep. 1, 2021, doi: 10.1109/TDSC.2021.3108782.

[46] R. Kolandaisamy *et al.*, "A multivariant stream analysis approach to detect and mitigate DDoS attacks in vehicular ad hoc networks," *Wireless Commun. Mobile Comput.*, vol. 2018, no. 20, pp. 1–13, May 2018. [Online]. Available: https://doi.org/10.1155/2018/2874509

[47] G. de Biasi, L. F. Vieira, and A. A. Loureiro, "Sentinel: Defense mechanism against DDoS flooding attack in software defined vehicular network," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2018, pp. 1–6.

[48] J. Liu, X. Wang, S. Shen, G. Yue, S. Yu, and M. Li, "A Bayesian Q-learning game for dependable task offloading against DDoS attacks in sensor edge cloud," *IEEE Internet Things J.*, vol. 8, no. 9, pp. 7546–7561, May 2021.

[49] B. Li, R. Liang, D. Zhu, W. Chen, and Q. Lin, "Blockchain-based trust management model for location privacy preserving in VANET," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3765–3775, Jun. 2021.

[50] S. Bi and Y. J. Zhang, "Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading," *IEEE Trans. Wireless Commun.*, vol. 17, no. 6, pp. 4177–4190, Jun. 2018.

[51] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 1927–1941, Sep. 2018.

[52] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: Wiley, 2014.

[53] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2016, pp. 3844–3852.

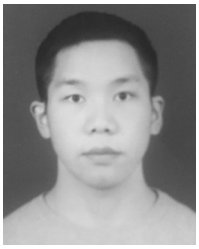
[54] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2017, pp. 1024–1034.

[55] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–17. [Online]. Available: https://openreview.net/forum?id=ryGs6iA5Km

[56] A. Tampuu *et al.*, "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, 2017, Art. no. e0172395.

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

[58] W. Chen *et al.*, "Cooperative and distributed computation offloading for blockchain-empowered industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8433–8446, Oct. 2019.
[59] F. Khoramnejad and M. Erol-Kantarci, "On joint offloading and resource allocation: A double deep Q-network approach," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 4, pp. 1126–1141, Dec. 2021.
[60] P. Yang, L. Li, W. Liang, H. Zhang, and Z. Ding, "Latency optimization for multi-user NOMA-MEC offloading using reinforcement learning," in *Proc. 28th Wireless Opt. Commun. Conf. (WOCC)*, 2019, pp. 1–5.
[61] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. NIPs*, vol. 99, 1999, pp. 1057–1063.



Yuchuan Deng received the bachelor's degree in computer science and technology from Sichuan University, Chengdu, China, in 2019. He is currently pursuing the master's degree in information and communication engineering with Wuhan University, Wuhan, China.

His research interests include software-defined network, system security, and machine learning.

Hao Jiang (Member, IEEE) received the B.Eng. degree in communication engineering and the M.Eng. and Ph.D. degrees in communication and information systems from Wuhan University, Wuhan, China, in 1999, 2001, and 2004, respectively.

He undertook his postdoctoral research work with LIMOS, Clermont-Ferrand, France, from 2004 to 2005. He was a Visiting Professor with the University of Calgary, Calgary, AB, Canada, ISIMA, Aubière, France, and Blaise Pascal University, Clermont-Ferrand. He is currently a Professor with Wuhan University. He has authored over 60 papers in different journals and conferences. His research interests include mobile ad hoc networks and mobile.

Peijing Cai received the bachelor's degree in cyberspace security from Nanchang University, Nanchang, China, in 2020. She is currently pursuing the master's degree in information and communication engineering with Wuhan University, Wuhan, China.

Her research interests include software-defined network, network security, and routing algorithm.

Tong Wu is currently pursuing the Ph.D. degree with the School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, China, working with Prof. P. Zhou.

Her research interests include deep learning, deep reinforcement learning, and MEC.

Pan Zhou (Senior Member, IEEE) received the Ph.D. degree from the School of Electrical and Computer Engineering, The Georgia Institute of Technology, Atlanta, GA, USA, in 2011.

He is currently a Full Professor and a Ph.D. Advisor with Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. His current research interests include security and privacy, big data analytics, machine learning, and information networks.

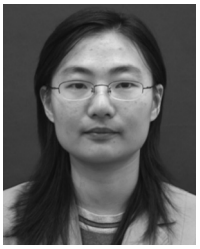
Prof. Zhou received the Rising Star in Science and Technology of HUST in 2017, and the Best Scientific Paper Award in the 25th International Conference on Pattern Recognition (ICPR 2020). He is currently an Associate Editor of the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

Beibei Li (Member, IEEE) received the B.E. degree (awarded Outstanding Graduate) in communication engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2014, and the Ph.D. degree (awarded Full Research Scholarship) from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2019.

He is currently an Associate Professor (Doctoral Advisor) with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China. He was invited as a Visiting Researcher with the Faculty of Computer Science, University of New Brunswick, Saint John, NB, Canada, from March to August 2018, and also the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, from February to April 2019. His current research interests include several areas in security and privacy issues on cyber–physical systems, such as smart grids, industrial control systems, and IoT, with a focus on intrusion detection techniques, artificial intelligence, and applied cryptography.

Hao Lu (Member, IEEE) received the bachelor's degree from Guilin University of Electronics Technology, Guilin, China, in 1995, and the master's degree from the School of Computer Science, National University of Defense Technology, Changsha, China, in 2005.

He is currently an Associate Professor with the Air Force Early Warning Academy, Wuhan, China. His main research interests are computer network and information security, virtualization and cloud computing, and applications of artificial intelligence and big data.



Jing Wu (Member, IEEE) received the B.Eng. degree in communication engineering and the Ph.D. degree in communication and information systems from Wuhan University, Wuhan, China, in 2002 and 2007, respectively.

She undertook her postdoctoral research work with LIMOS, Clermont-Ferrand, France, from 2004 to 2005. She is currently an Associate Professor with Wuhan University. Her research interests include wireless communication networks, network simulation, and intelligence data processing.

Xin Chen received the Ph.D. degree from the National University of Defense Technology, Changsha, China, in 2010.

He is currently an Associate Professor with the Air Force Early Warning Academy, Wuhan, China. His current research fields include deep learning-based image recognition and big data mining technology applications.

Kehao Wang (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in communication and information system from Wuhan University of Technology, Wuhan, China, in 2003 and 2006, respectively, and the Ph.D. degree from the Department of Computer Science, University of Paris-Sud XI, Orsay, France, in 2012.

He was a Postdoctoral Fellow with Hong Kong Polytechnic University, Hong Kong, from February 2013 to August 2013. He joined the School of Information Engineering, Wuhan University of Technology, in 2013, where he is currently a Professor. He has been a Visiting Scholar with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA, since December 2015. His research interests include stochastic optimization, operation research, scheduling, wireless network communications, and embedded operating system.