Decentralized Power Allocation for MIMO-NOMA Vehicular Edge Computing Based on Deep Reinforcement Learning

Hongbiao Zhu, Qiong Wu, *Member, IEEE*, Xiao-Jun Wu, *Member, IEEE*, Qiang Fan, Pingyi Fan, *Senior Member, IEEE*, and Jiangzhou Wang, *Fellow, IEEE*

***Abstract*—Vehicular edge computing (VEC) is envisioned as a promising approach to process the explosive computation tasks of vehicular user (VU). In the VEC system, each VU allocates power to process partial tasks through offloading and the remaining tasks through local execution. During the offloading, each VU adopts the multi-input multi-output and non-orthogonal multiple access (MIMO-NOMA) channel to improve the channel spectrum efficiency and capacity. However, the channel condition is uncertain due to the channel interference among VUs caused by the MIMO-NOMA channel and the time-varying path loss caused by the mobility of each VU. In addition, the task arrival of each VU is stochastic in the real world. The stochastic task arrival and uncertain channel condition affect greatly on the power consumption and latency of tasks for each VU. It is critical to design an optimal power allocation scheme considering the stochastic task arrival and channel variation to optimize the long-term reward, including the power consumption and latency in the MIMO-NOMA VEC. Different from the traditional centralized deep reinforcement learning (DRL)-based scheme, this article constructs a decentralized DRL framework to formulate the power allocation optimization problem, where the local observations are selected as the state. The deep deterministic policy gradient (DDPG) algorithm is adopted to learn the optimal power allocation scheme based on the decentralized DRL framework. Simulation results demonstrate that our proposed power allocation scheme outperforms the existing schemes.**

***Index Terms*—Decentralized, deep reinforcement learning (DRL), multi-input multi-output and non-orthogonal multiple access (MIMO-NOMA), power allocation, vehicular edge computing (VEC).**

Manuscript received November 27, 2021; accepted December 20, 2021. Date of publication December 27, 2021; date of current version July 7, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61701197; in part by the Beijing Natural Science Foundation under Grant 4202030; and in part by the 111 Project under Grant B12018. *(Corresponding author: Qiong Wu.)*

Hongbiao Zhu and Qiong Wu are with the School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China (e-mail: hongbiaozhu@stu.jiangnan.edu.cn; qiongwu@jiangnan.edu.cn).

Xiao-Jun Wu is with the School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214122, China (e-mail: wu_xiaojun@jiangnan.edu.cn).

Qiang Fan is with Wistron AiEdge, San Jose, CA 95131 USA (e-mail: qiang_fan@wistron.com).

Pingyi Fan is with the Department of Electronic Engineering, Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: fpy@tsinghua.edu.cn).

Jiangzhou Wang is with the School of Engineering, University of Kent, Canterbury CT2 7NT, U.K. (e-mail: j.z.wang@kent.ac.uk).

Digital Object Identifier 10.1109/JIOT.2021.3138434

I. Introduction

WITH the increasing number of vehicles, the growing demand of computation-intensive applications, such as virtual/augmented reality (VR/AR), image processing, face detection, and recognition, is emerging to satisfy the infotainment experience of vehicular users (VUs) [1]. These applications are realized through collecting a great amount of data by various VU equipments, such as smartphones and wearable devices. Such large amount of data results in intensive computation tasks, which need to be processed in time, thus leading to heavy computation burden for VUs [2], [3]. Vehicular edge computing (VEC) is a promising way to relieve the burden [4], where a VEC server with high computational capability is connected with a base station (BS) to provide VUs with computation resources at the edge [5], [6]. When a VU has some tasks to process, it can either offload the tasks to the VEC server collocated connecting with the BS [7] or execute the tasks locally. For the task offloading, the VU has to consume energy in the data transmission, where the offloading power is defined as the transmission power. In addition, when the VU processes the tasks locally, the local task processing will incur the energy consumption at its central processing unit (CPU). For simplicity, we define the power consumption of task processing at the VU as the local execution power.

During the offloading, the multi-input multi-output and non-orthogonal multiple access (MIMO-NOMA) channel is considered here due to its high channel spectrum efficiency and channel capacity. Specifically, each VU can share the whole spectrum and undivided bandwidth to offload tasks and the BS is equipped with multi-antenna to receive tasks from all VUs simultaneously [8]–[11]. However, the channel condition is time varying due to the channel interference among VUs caused by the MIMO-NOMA channel and the time-varying path loss caused by the mobility of each VU. In addition, the task arrival of each VU is stochastic in practice. The stochastic task arrival and uncertainty of channel condition significantly impact the power consumption and latency of task processing for each VU. For example, a VU would take more time in task offloading when the task arrival rate is increasing and the channel condition is getting deteriorated, which increases the power consumption and latency. In this case, the VU should allocate more local execution power to reduce the power consumption and latency. In the VEC, VU equipment has limited energy and the applications, such as VR/AR and real-time interactive 3-D gaming, should be processed within a limited time; therefore, power consumption and latency are two important performance metrics in task processing [12]–[14]. It is critical to design an optimal power allocation scheme considering the stochastic task arrival and uncertainty of channel condition in the MIMO-NOMA VEC.

Deep reinforcement learning (DRL) is a favorable framework to formulate the similar optimization problem in complex environments [15]. Many existing works have designed the offloading scheme based on the centralized DRL framework in VEC by taking various factors into account, where the BS first collects the global information including all VUs' states to determine the action of each VU, which causes huge overhead and extra latency [16]–[27]. Only a few works focused on decentralized DRL-based offloading schemes, where each VU collects the local observations to select its action; thus, the overhead and latency can be reduced efficiently [28], [29]. However, these works did not consider the channel caused by employing the MIMO-NOMA mode. To the best of our knowledge, no work has considered the stochastic task arrival and the uncertainty of the MIMO-NOMA channel condition in the decentralized DRL-based optimal power allocation scheme in VEC.

In this article, we consider the stochastic task arrival, and the channel condition uncertainty caused by the MIMO-NOMA channel interference and the mobility of VUs, and propose a decentralized DRL-based power allocation scheme to optimize the long-term reward in VEC in terms of power consumption and latency. The main contributions of this article are summarized as follows.

1) We formulate the power allocation optimization problem, where the state, action, and reward function are elaborately defined to enable each VU to learn the optimal power allocation scheme according to the local observations. Then, the deep deterministic policy gradient (DDPG) algorithm is adopted to learn the optimal power allocation decision based on the DRL framework.
2) Extensive experiments are carried out to test the performance of the proposed scheme and show its superiority to other existing polices in terms of power consumption and latency of task processing.

The remainder of this article is organized as follows. Section II reviews the related work. Section III introduces the system model. In Section IV, the decentralized DRL framework is set up to formulate the power allocation problem. Section V presents the DDPG algorithm on how to learn the optimal power allocation scheme based on the DRL framework. Section VI presents the simulation results. Finally, Section VII concludes this article.

II. Related Work

In this section, we first review the related works on the offloading scheme in mobile-edge computing (MEC) considering the MIMO or NOMA channel, and then we review the existing works on the DRL-based offloading scheme in the VEC.

A. *Offloading in MIMO or NOMA MEC*

In recent years, many works have considered the MIMO or NOMA channel while designing the offloading scheme in MEC.

Wang *et al.* [4] employed the NOMA channel in the MEC computation offloading system to minimize the energy consumption of all users where Lagrange dual was adopted to make decisions about task offloading proportion, successive interference cancellation order, offloading power, and local CPU frequencies. Pan *et al.* [30] considered the NOMA channel in MEC for uploading computation tasks and downloading computation result where convex optimization was adopted to minimize the energy consumption by determining offloading task partitions, offloading power, and task time allocation. Huang *et al.* [31] focused on the channel estimation process with pilots in the massive MIMO MEC system to minimize the offloading latency of all users by the optimizing power of pilot transmission and data transmission, as well as the allocation of computing resource. Ding *et al.* [32] studied a multiuser MIMO (MU-MIMO) MEC system to minimize the system cost, the weighted sum of latency, and energy consumption. Feng *et al.* [33] considered the fairness of all users in an MU-MIMO MEC system and to minimize offloading latency through optimizing the distribution of resource, transmission of pilot sequence, and data. However, these works did not consider the scenario of vehicular scenarios.

B. *DRL-Based Offloading in VEC*

Many works have discussed the DRL-based offloading scheme in VEC. Dong *et al.* [19] considered the NOMA channel in VEC where a deep *Q*-network (DQN) was applied to guarantee the delay requirement and minimize the energy consumption. Ke *et al.* [20] designed a three-layer VEC offloading system, including a macro BS, multiple small BSs, and vehicles where DRL is applied to minimize the cost consisting of energy consumption and transmission delay. He *et al.* [21] proposed an offloading scheme considering network, cache, and computation resource in VEC, where DQN was employed to select the optimal offloading decision that maximizes the reward, including the caching state, computation capability, and received signal-to-noise ratio (SNR). Tan and Hu [22] formulated the joint optimal caching and computing allocation problem to minimize the VEC system cost, including communication, computation, and storage under the constraint of server deadline. DQN was employed to solve the optimization problem where the channel was assigned by orthogonal frequency-division multiplexing (OFDM). Ning *et al.* [23] constructed an edge computation and cache model for VEC consisting of macro BS, several RSUs, and VUs, where the tasks of VUs were divided into computing tasks and content tasks. DDPG was employed to obtain the optimal resource allocation in order to maximize the reward of a mobile network operator (MNO), including computing and caching cost, and penalty on Quality of Experience (QoE). Liu *et al.* [24] took stochastic traffic and uncertain communication conditions of VEC into consideration, and adopted the semi-Markov process to formulate an optimization problem to maximize the total

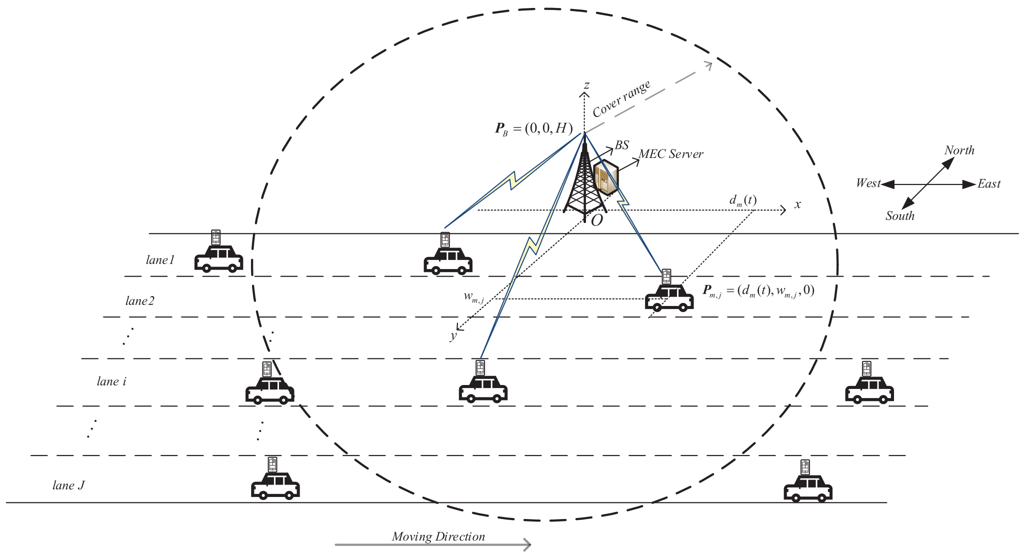

Fig. 1. System model.

network utility of the VEC. The DQN method was employed to obtain the optimal offloading scheme. Ren *et al.* [27] designed a VEC architecture consisting of BSs, RSUs, VUs, and cell software designed network controller where tasks can be migrated among BSs and RSUs. A centralized DRL-based offloading scheme was designed to manage the network resource by making decisions of offloading, migration, and resource allocation. However, these works only focused on the centralized DRL-based offloading scheme.

A few works have also focused on decentralized DRL-based offloading schemes in VEC [28], [29]. Ye *et al.* [28] considered a VEC that is composed of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications where V2I communication reserved orthogonal single-input single-output (SISO) channels. DQN was adopted to select the task transmitting subband and power level for VUs to maximize the reward consisting of system communication capacity and latency. Xu *et al.* [29] considered the similar scenario [28] to maximize the sum rate of every subband communication where DDPG was employed to obtain the optimal policy in continuous action space. However, [28] and [29] did not consider the channel varying caused by the MIMO-NOMA channel interference and the mobility of VU in VEC.

As mentioned above, no work has considered the stochastic task arrival and channel varying in the MIMO-NOMA VEC while designing the decentralized DRL-based power allocation scheme.

III. System Model

The system model is shown as Fig. 1. Consider a VEC system where an N-antenna BS is placed along a one-way J-lane road and a VEC server is attached to the BS. The lanes from near to far according to the vertical distance to the BS are denoted as $1, 2, \ldots, j, \ldots, J$, respectively. M vehicles on different lanes traverse the coverage of BS from left to right with different velocities, where each vehicle carries a computation resource-limited single-antenna VU. The duration time that a VU on lane j stays in the transmission coverage of BS is divided into N_j equal time slots, each of which is a constant τ_0. At each slot, computation-intensive tasks arrive at the first come first service (FCFS) buffer of each VU stochastically following independent and identical distribution (i.i.d.). Meanwhile, each VU allocates the local execution power and offloading power to process the tasks stored in the buffer queue locally or at the VEC server nearby, respectively. Moreover, the channel condition varies due to the interference among VUs' MIMO-NOMA channel and the time-varying path loss caused by the mobility of VUs. During the offloading, each VU first transmits tasks to the BS, and then the BS processes the tasks and adopts the zero-forcing (ZF) technique to detect the received signal and noise of each VU from the received signal of all VUs and further determines the signal-to-interference-plus-noise ratio (SINR) of each VU. Afterward, the BS sends back the computation results as well as the determined SINR to each VU at the next slot. Different from the traditional centralized DRL-based offloading scheme in VEC, in this article, each VU can distributively determine the power allocation based on its local information. Next, we will introduce the computation model, network model, and mobility model to formulate the local information of VU m, such as the buffer length, SINR, and position, respectively. For simplicity, the notations adopted in this article are listed as Table I.

A. Mobility Model

Let $\boldsymbol{P}_{m,j}(t)$ be the position of VU m, which moves on lane j at slot t. A space rectangular coordinate system shown in Fig. 1 is set to illustrate the positions of each VU and the BS, where the origin is the position of the BS, the direction of the x-axis is the moving direction of VUs, i.e., east, the direction of the y-axis is south, and the direction of z-axis is set along the antennas of BS, which is perpendicular to both x-axis and y-axis. Let $d_m(t)$ and $w_{m,j}$ be the distances between VU m and the antennas of BS along x-axis and y-axis at slot

TABLE I
SUMMARY FOR NOTATIONS

Notation	Description	Notation	Description
$a_m(t)$	Task bits of user m arrived at slot t.	$a_{m,t}$	Action space of VU m at slot t.
a_m	Abbreviation of $a_{m,t}$.	a'_m	Abbreviation of $a_{m,t+1}$.
a^i_m	Action of the i-th tuple in mini-batch.	$B_m(t)$	Buffer length of VU m at slot t.
$\mathcal{R}$	Replay buffer.	D	Diameter of BS's coverage.
$d_{m,l}(t)$	Bits of user m processed locally at slot t.	$d_{m,o}(t)$	Bits of user m offload at slot t.
$d_m(t)$	Distance between user m and BS's antennas along the x-axis.	$\boldsymbol{e}(t)$	Error vector of adjacent slots channel vector.
$f_m(t)$	CPU frequence of user m at slot t.	F_{max}	The maximum allowed CPU frequence.
$\boldsymbol{h}^s_m(t)$	Small-scale Rayleigh fading channel gain of VU m at slot t.	$h^p_m(t)$	The large-scale fading coefficient reflects the path-loss of VU m at slot t.
h_r	Reference power gain at distance 1m.	$\boldsymbol{H}(t)$	Channel matrix between BS and every VU at slot t.
H	Height of BS.	I	Size of mini-batch.
K_{max}	The maximum episode in training stage.	i	Index of tuples in the mini-batch.
$J(\mu_m)$	Objective function.	L_m	CPU cycles needed for VU m process one bit.
$L(\zeta^m)$	Loss function.	N_j	Total number of time slots.
$\boldsymbol{n}(t)$	Noise of signal received by BS.	N	Number of antenna.
N_{max}	The maximum number of VUs in the sytem.	$\boldsymbol{P}_{m,j}(t)$	Location of VU m at slot t.
$\boldsymbol{P}_B$	Location of BS.	$p_{m,o}(t)$	Offload power offered by VU m at slot t.
$p_{m,l}(t)$	Local process power offered by VU m at slot t.	$P_{max,o}$	Maximum offload power.
$P_{max,l}$	Maximum local execution power.	$Q^{\mu_{\theta^m}}(s_{m,t},a_{m,t})$	Action-value function of VU m following policy μ_{θ^m}.
$Q^{\zeta^m}(s_{m,t},a_{m,t})$	Action-value function approximated by critic-network.	$Q^{\zeta^{m'}}(s_{m,t},a_{m,t})$	Action-value function of VU m approximated by target critic-network.
$r_{m,t}$	Reward of VU m slot t.	r^i_m	Reward of the i-th tuple in mini-batch.
r_m	Abbreviation of $r_{m,t}$.	$\mathcal{R}$	Replay buffer.
$s_{m,t}$	State space of VU m lot t.	s_m	Abbreviation of $s_{m,t}$.
s'_m	Abbreviation of $s_{m,t+1}$	s^i_m	State of the i-th tuple in mini-batch.
s'^i_m	Next state of the i-th tuple in mini-batch.	T_s	Safety time.
v_j	Velocities of vehicles driven on lane j.	v_m	velocity of vehicular user m.
w_0	The width between the VU driven on the lane 1 and BS's antennas along y-axis.	w	Width of roads.
$w_{m,j}$	Width between VU m driven on lane j and antennas along the y-axis.	W	Bandwidth.
$\boldsymbol{y}(t)$	Signal received by BS.	y^i_m	Target value.
γ	Discounting factor of long-term reward.	$\gamma_m(t)$	SINR of VU m at slot t.
Δ_t	The exploration noise at slot t.	ζ^m	Parameter of critic network.
$\zeta^{m'}$	Parameter of target critic-network.	η	path-loss exponent.
θ^m	Parameter of actor network.	θ^{m*}	Optimized parameter of actor network.
$\theta^{m'}$	Parameter of target actor-network.	κ_m	Effective switched capacitance of user m.
λ_m	Mean rate of tasks arrival for VU m.	μ_{θ^m}	Policy of VU m approximated by actor network.
ρ_m	Normalized channel correlation coefficient of user m between adjacent slots.	σ^2_R	Additive white Gaussian noise variance of the signal received by BS.
τ	Update parameter for target networks.	τ_0	Slot duration.
ω_1,ω_2	Weighted factors of reward.		

t, respectively. Thus, $\boldsymbol{P}_{m,j}(t)$ is denoted as $(d_m(t), w_{m,j}, 0)$ in the space rectangular coordinate system, where $w_{m,j}$ depends on the lane index of VU m, i.e., j, and is calculated as

$$w_{m,j} = (j-1)\cdot w + w_0 \tag{1}$$

here w is the width of a lane and w_0 is the distance between lane 1 and the BS's antennas along y-axis.

Similar to [34], the position of VU m is approximately constant within each time slot due to the sufficiently small value of τ_0. Since VU m moves on lane j with a constant velocity v_j, $d_m(t)$ is updated as

$$d_m(t) = d_m(t-1) + v_j\tau_0, d_m(t) \in \left[-\frac{D}{2}, \frac{D}{2}\right] \tag{2}$$

where D is the coverage of the BS and $d_m(1) = -(D/2)$. VU m communicates with the BS once it enters the coverage of the BS and calculates $d_m(t)$ at each slot t according to (2). Therefore, $d_m(t)$ is a local observation of VU m at slot t to reflect the mobility of VU m.

Note that according to the 4-second rule [35], the maximum number of VUs on lane j can be calculated as $\lfloor D/(v_j \cdot T_s)\rfloor$, where T_s is the safety time, i.e., 4 s. Thus, the maximum number of VUs in the sytem can be calculated as

$$N_{\max} = \sum_{j=1}^{J} \left\lfloor D/\left(v_j \cdot T_s\right)\right\rfloor. \tag{3}$$

B. Network Model

The channel matrix at slot t can be expressed as $\boldsymbol{H}(t) = [\boldsymbol{h}_1(t), \ldots, \boldsymbol{h}_m(t), \ldots, \boldsymbol{h}_M(t)] \in \mathbb{C}^{N\times M}$, where $\boldsymbol{h}_m(t) \in \mathbb{C}^{N\times 1}$ is the channel vector between VU m and BS. In the MIMO-NOMA channel, the signal received by BS at slot t is the signal transmitted from all VUs, which can be expressed as

$$\boldsymbol{y}(t) = \sum_{m\in\mathcal{M}} \boldsymbol{h}_m(t)\sqrt{p_{m,o}(t)}s_m(t) + \boldsymbol{n}(t)$$
$$p_{m,o}(t) \in [0, P_{\max,o}] \tag{4}$$

where $p_{m,o}(t)$ is the offloading power of VU m at t, $P_{\max,o}$ is the maximum offloading power, $s_m(t)$ is the complex data symbol with unit variance, and $\boldsymbol{n}(t)$ is the vector of additive white Gaussian noise (AWGN) with variance σ_R^2, i.e., $\boldsymbol{n}(t) \sim \mathcal{CN}(\mathbf{0}, \sigma_R^2 \boldsymbol{I}_N)$, and $\boldsymbol{I}_N$ is an $N \times N$ identity vector. Furthermore, $\boldsymbol{h}_m(t)$ is an integrated one of the stochastic small-scale fading channel gain $\boldsymbol{h}_m^s(t)$ and the large-scale fading coefficient $h_m^p(t)$, which reflects the path loss of VU m [17], i.e.,

$$\boldsymbol{h}_m(t) = \boldsymbol{h}_m^s(t)\sqrt{h_m^p(t)}. \tag{5}$$

In (5), $h_m^p(t)$ characterizes the mobility of VU m and is calculated as

$$h_m^p(t) = \frac{h_r}{\left\|\boldsymbol{P}_{m,j}(t) - \boldsymbol{P}_B\right\|^{\eta}} \tag{6}$$

where η is the path-loss exponent, h_r is the channel power gain at 1-m distance, $\boldsymbol{P}_{m,j}(t) = (d_m(t), w_{m,j}, 0)$ is the position of VU m at slot t, and $\boldsymbol{P}_B$ is the position of antennas of BS. Note that $w_{m,j}$ is calculated according to (1) and $d_m(t)$ is calculated according to (2). Let H be the height of antennas of BS; thus, $\boldsymbol{P}_B = (0, 0, H)$.

In addition, the following autoregressive (AR) model is adopted to formulate the relationship between $\boldsymbol{h}_m^s(t)$ and $\boldsymbol{h}_m^s(t-1)$ [36], i.e.:

$$\boldsymbol{h}_m^s(t) = \rho_m \boldsymbol{h}_m^s(t-1) + \sqrt{1-\rho_m^2}\,\boldsymbol{e}(t) \tag{7}$$

where ρ_m is the normalized channel correlation coefficient between the consecutive slots, and $\boldsymbol{e}(t)$ is the error vector, which obeys the complex Gaussian distribution and is correlated with $\boldsymbol{h}_m^s(t)$.

According to Jake's fading spectrum, $\rho_m = J_0(2\pi f_d^m \tau_0)$, where $J_0(\cdot)$ is the zeroth-order Bessel function of the first kind and f_d^m is the Doppler frequency of VU m [37], which can be calculated as

$$f_d^m = \frac{v_m}{\Lambda}\cos\Theta \tag{8}$$

where Λ is the wavelength, and Θ is the angle between moving direction, i.e., $\boldsymbol{x}_0 = (1, 0, 0)$, and uplink communication direction, i.e., $\boldsymbol{P}_B - \boldsymbol{P}_{m,j}(t)$. Thus, $\cos\Theta$ can be calculated as

$$\cos\Theta = \frac{\boldsymbol{x}_0 \cdot \left(\boldsymbol{P}_B - \boldsymbol{P}_{m,j}(t)\right)}{\left\|\boldsymbol{P}_B - \boldsymbol{P}_{m,j}(t)\right\|}. \tag{9}$$

Then, the BS adopts the pseudoinverse of $\boldsymbol{H}(t)$, which is denoted as $\boldsymbol{H}^{\dagger}(t)$, as the ZF detector to detect the received signal of VU m from $\boldsymbol{y}(t)$. According to [36], $\boldsymbol{H}^{\dagger}(t)$ is calculated as

$$\boldsymbol{H}^{\dagger}(t) = \left(\boldsymbol{H}^H(t)\boldsymbol{H}(t)\right)^{-1}\boldsymbol{H}^H(t) \tag{10}$$

where $\boldsymbol{H}^H(t)$ is the conjugate transpose of $\boldsymbol{H}(t)$.

Specifically, letting $\boldsymbol{g}_m^H(t)$ be the mth row of $\boldsymbol{H}^{\dagger}(t)$, by multiplying $\boldsymbol{y}(t)$ with $\boldsymbol{g}_m^H(t)$, we can obtain the following equation according to (4):

$$\boldsymbol{g}_m^H(t)\boldsymbol{y}(t) = \boldsymbol{g}_m^H(t)\sum_{m\in\mathcal{M}}\boldsymbol{h}_m(t)\sqrt{p_{m,o}(t)}s_m(t) + \boldsymbol{g}_m^H(t)\boldsymbol{n}(t). \tag{11}$$

Since $\boldsymbol{g}_m^H(t)$ is the mth row of $\boldsymbol{H}^{\dagger}(t)$, according to (10), we have

$$\boldsymbol{g}_m^H(t)\boldsymbol{h}_i(t) = \delta_{m,i}(t) = \begin{cases} 1, & i = m \\ 0, & i \neq m. \end{cases} \tag{12}$$

Substituting (12) into (11), we have

$$\boldsymbol{g}_m^H(t)\boldsymbol{y}(t) = \sqrt{p_{m,o}(t)}s_m(t) + \boldsymbol{g}_m^H(t)\boldsymbol{n}(t) \tag{13}$$

where $\sqrt{p_{m,o}(t)}s_m(t)$ is the signal received by BS from VU m, $\sqrt{p_{m,o}(t)}$ is the power of the received signal, and $\boldsymbol{g}_m^H(t)\boldsymbol{n}(t)$ is the noise of VU m received by BS. Since the power of $\boldsymbol{n}(t)$ is σ_R^2, the noise power is calculated as $\|\boldsymbol{g}_m^H(t)\|^2\sigma_R^2$. Thus, the SINR of VU m can be calculated as

$$\gamma_m(t) = \frac{p_{m,o}(t)}{\left\|\boldsymbol{g}_m^H(t)\right\|^2\sigma_R^2}. \tag{14}$$

The BS can detect the SINR of VU m at slot t, i.e., $\gamma_m(t)$, according to (4)–(14) and transmit $\gamma_m(t)$ to VU m at next slot. In this case, VU m receives $\gamma_m(t-1)$ at slot t and thus, $\gamma_m(t-1)$ is also a local observation of VU m at slot t to reflect the channel variation.

C. Computation Model

For VU m, the buffer length of VU m at slot t is denoted as $B_m(t)$ and the relationship between $B_m(t)$ and $B_m(t-1)$ is expressed as

$$B_m(t) = \left[B_m(t-1) - (d_{m,o}(t-1) + d_{m,l}(t-1))\right]^+ + a_m(t-1) \tag{15}$$

where $[\cdot]^+ = \max(0, \cdot)$, $a_m(t-1)$ is the amount of the tasks arriving at the buffer queue of VU m at slot $t-1$, and $d_{m,l}(t-1)$ and $d_{m,o}(t-1)$ are the amount of the tasks processed by local execution and offloaded to the BS at slot $t-1$, respectively. Therefore, the amount of tasks departing from the buffer at slot $t-1$ becomes $d_{m,l}(t-1) + d_{m,o}(t-1)$, which should not exceed $B_m(t-1)$. We will also explain how $d_{m,l}(t-1)$ and $d_{m,o}(t-1)$ are determined as follows.

1) Local Execution: Let L_m be the computation intensity of tasks (i.e., the number of CPU cycles required to processed one bit data) and $f_m(t-1)$ be the CPU frequency of VU m at slot $t-1$. The task size that can be processed by local execution at slot $t-1$ is calculated as

$$d_{m,l}(t-1) = \tau_0 f_m(t-1)/L_m. \tag{16}$$

Letting $p_{m,l}(t-1)$ be the local execution power at slot $t-1$, $f_m(t-1)$ is calculated as

$$f_m(t-1) = \sqrt[3]{p_{m,l}(t-1)/\kappa}$$
$$p_{m,l} \in [0, P_{\max,l}], f_m(t-1) \in [0, F_{\max}] \tag{17}$$

where κ is the effective switched capacitance, $P_{\max,l}$ is the maximum local execution power, and $F_{\max}$ is the maximum CPU frequency. According to (17), $F_{\max}$ can be calculated as $F_{\max} = \sqrt[3]{P_{\max,l}/\kappa}$.

2) Offloading: In the offloading mode, the computation resources of VEC server are sufficient; thus, the latency that the VEC server processes the tasks is negligible. In addition, the size of computation result is usually very small; thus, the feedback delay can also be ignored. Therefore, the delay of task transmission is the duration of a slot τ_0. In this case, the amount of tasks processed by offloading at slot $t-1$ can be calculated according to the Shannon theory, i.e.,

$$d_{m,o}(t-1) = \tau_0 W \log_2(1+\gamma_m(t-1)) \tag{18}$$

where W is the bandwidth and $\gamma_m(t-1)$ is the SINR of VU m at slot $t-1$.

Since VU m receives its SINR information $\gamma_m(t-1)$ from the BS at slot t, it allocates $p_{m,l}(t-1)$ and observes $a_m(t-1)$ at slot $t-1$. In this case, VU m can calculate $B_m(t)$ at slot t according to (15)–(18) given L_m, κ, $P_{\max}$, τ_0, and W and thus, $B_m(t)$ is another local observation of VU m at slot t to reflect the stochastic task arrival and the uncertain channel condition.

IV. Problem Formulation

In the system, statistics task arrival and uncertain channel condition are all unknown to each VU; thus, we adopt the DRL framework, which includes state, action, policy, and reward to formulate the power allocation problem in the VEC [15]. Specifically, for each VU at each slot t, VU observes the current local state s_t and makes action a_t based on s_t according to policy μ, i.e., the function that generates the action based on the state at each slot. Then, VU receives a reward r_t and observes the state at the next slot s_{t+1}, which is transited from the current state s_t. Next, the state s_t, action a_t, and reward r_t of VU at slot t will be defined, respectively.

A. State

Different from the traditional centralized DRL-based offloading scheme in VEC, each VU observes its local state to determine the power allocation in this article. Since the power consumption and delay are impacted by the stochastic task arrival and uncertain channel condition caused by the MIMO-NOMA channel interference and mobility of VUs, the local state should be selected to reflect the stochastic task arrival and uncertain channel condition as well as the mobility of the VU.

In the system model, the distance between VU m, and the antennas of the BS along x-axis at slot t, i.e., $d_m(t)$, determines the position of VU m at slot t, which reflects the mobility of VU m. In addition, according to (14), the SINR of VU m at slot t, i.e., $\gamma_m(t-1)$, depends on $\boldsymbol{g}_m^H(t-1)$ that is related with the channel vector $\boldsymbol{h}_m(t-1)$, and thus, $\gamma_m(t-1)$ can reflect the uncertain channel condition at slot t. Moreover, according to (15)–(18), the buffer length of VU m at slot t, i.e., $B_m(t)$, is a function of $a_m(t-1)$ and $\gamma_m(t-1)$, where $a_m(t-1)$ reflects the stochastic task arrival and $\gamma_m(t-1)$ reflects the uncertain channel condition. Therefore, $B_m(t)$ reflects both stochastic task arrival and uncertain channel condition. As shown in the system model, $B_m(t)$, $\gamma_m(t-1)$, and $d_m(t)$ are all the local observations of VU m at slot t; therefore, the state of VU m at slot t can be defined as

$$s_{m,t} = \left[B_m(t), \gamma_m(t-1), d_m(t)\right] \tag{19}$$

where $\gamma_m(t-1)$ depends on $\boldsymbol{h}_m(t-1)$ and $B_m(t)$ depends on $\gamma_m(t-1)$ and $a_m(t-1)$. Since $a_m(t-1)$ and $\boldsymbol{h}_m(t-1)$ are random values within continuous space, thus the state space of VU is continuous.

B. Action

Each VU m allocates the local execution power and offloading power based on the local observed state $s_{m,t}$; thus, the local execution and offloading power are defined as the action of VU m at slot t, i.e.,

$$a_{m,t} = \left[p_{m,o}(t), p_{m,l}(t)\right]. \tag{20}$$

Note that similar to [38], we consider the fine-grained computation applications; thus, VU m allocates the local execution and offloading power within continuous spaces in $[0, P_{m,l}]$ and $[0, P_{m,o}]$ to process the tasks, respectively. In this case, the action space of VU m is continuous.

C. Reward Function

In this article, VU m aims to improve the network performance in terms of the power consumption and delay. As described in the computation model, the latency of task processing at the VEC server is negligible and the feedback delay during the offloading is also ignored at each slot. In this case, the delay of task transmission is a constant, i.e., the duration of a slot. Thus, the delay consumed by VU m is impacted by the buffer delay that is proportional to the average buffer length according to Little's Theorem [39]. Therefore, the reward function of VU m at slot t is defined as

$$r_{m,t} = -\left[\omega_1\left(p_{m,o}(t) + p_{m,l}(t)\right) + \omega_2 B_m(t)\right] \tag{21}$$

where ω_1 and ω_2 are the nonnegative weighted factors.

The expected long-term discounted reward of VU m is calculated as

$$J(\mu_m) := \mathbb{E}_{\mu_m}\left[\sum_{t=1}^{N_j} \gamma^{t-1} r_{m,t}\right] \tag{22}$$

where $\gamma \in [0, 1]$ is the discounting factor and N_j is the upper limit of slot index when VU m moves on lane j. In this article, we aim to find the optimal policy μ_m^* to maximize the expected long-term discounted reward of VU m.

Note that the network condition may be changed after each slot due to the dynamic VEC. At the beginning of each slot, each VU first observes its local state to acquire the changed network condition, and then makes actions based on its own local observation. Therefore, the reliability of our proposed scheme can be guaranteed under the dynamic VEC network.

V. Solution

In this section, we first describe the training stage to obtain the optimal policy, and then introduce the testing stage to test the performance under the optimal policy.

Algorithm 1: Training Stage for the DDPG-Based Framework

```
   Input: $\gamma$, $\tau$, $\theta^m$, $\zeta^m$
   Output: optimized $\theta^{m*}$, $\zeta^{m*}$
1  Randomly initialize the $\theta^m$, $\zeta^m$;
2  Initialize target networks by $\zeta^{m'} \leftarrow \zeta^m$, $\theta^{m'} \leftarrow \theta^m$;
3  Initialize replay experience buffer $\mathcal{R}$;
4  for episode from 1 to $K_{max}$ do
5      Reset simulation parameters for the VEC system model;
6      Receive initial observation state $s_1$;
7      for time slot t from 1 to $N_j$ do
8          Generate the power for local process and computation offloading according to the current policy and exploration noise $a_m = \mu_{\theta^m}(s_m|\theta^m) + \Delta_t$ ;
9          Execute action $a_m$, observe reward $r_m$ and new state $s'_m$ from the system model;
10         Store transition $(s_m, a_m, r_m, s'_m)$ in $\mathcal{R}$;
11         if number of tuples in $\mathcal{R}$ is larger than $I$ then
12             Randomly sample a mini-batch of $I$ transitions tuples from $\mathcal{R}$;
13             Update the critic network by minimizing the loss function according to Eq. (25);
14             Update the actor network according to Eq. (26);
15             Update target networks according to Eqs. (27) and (28).
```

A. Training Stage

Since the state and action spaces are continuous and the DDPG algorithm is suitable to solve the DRL-based problem under the continuous state and action space, therefore, we utilize the DDPG algorithm to obtain the optimal policy in the training stage.

The DDPG algorithm is based on the actor–critic architecture. The actor is applied for policy improvement, and the critic is applied for policy evaluation. The DDPG algorithm adopts a deep neural network (DNN) on actor and critic to efficiently approximate and evaluate the policy, respectively, thus forming the corresponding actor network and critic network. The actor network is used to approximate the policy μ_m, where the approximated policy is denoted as μ_{θ^m}, and the output of the actor network is the action based on the policy μ_{θ^m} and observed state. In the DDPG algorithm, the optimal policy is obtained through iterative policy improvement and evaluation. Moreover, the DDPG algorithm adopts the target networks, including target actor network and target critic network, to guarantee the stability of the algorithm. The architecture of the target actor network and target critic network is the same with the actor network and critic network, respectively. The pseudocode of the proposed algorithm is described in Algorithm 1. Let θ^m and ζ^m be the parameters of the actor network and critic network, respectively, and $\theta^{m'}$ and $\zeta^{m'}$ be the parameters of the target actor network and target critic network, respectively, and Δ_t be the noise for action exploration at slot t. For ease of understanding, we further introduce the DDPG algorithm in detail as follows.

First, θ^m and ζ^m are initialized randomly, while $\theta^{m'}$ and $\zeta^{m'}$ are initialized as θ^m and ζ^m, respectively. A replay buffer $\mathcal{R}$ with sufficient space is constructed to cache transition at each slot (lines 1–3).

Then, the algorithm is executed for $K_{\max}$ episodes. In the first episode, the position of VU m $(d_m(1), w_{m,j}, 0)$ is reset as the position that it enters the coverage area of the BS, i.e., $d_m(1)$ is set as $-(D/2)$, and $B_m(1)$ is initialized as half of the buffer size. Then, $\boldsymbol{h_m}^s(0)$ is initialized randomly, and $\boldsymbol{g}_m(0)$ is calculated according to (10) based on $\boldsymbol{h_m}^s(0)$, given the $\boldsymbol{g}_m(0)$ the initial SINR $\gamma_m(0)$ is calculated according to (14). Thus, VU m can observe the state at slot 1, i.e., $s_{m,1} = [B_m(1), \gamma_m(0), d_m(1)]$ (lines 4–6).

Afterward, the algorithm is executed iteratively from slot 1 to slot N_j. Given the input $s_{m,1}$, the output of the actor network is $\mu_{\theta^m}(s_{m,1}|\theta^m)$. As a noise Δ_1 is generated randomly and VU m sets the action $a_{m,1}$ as $\mu_{\theta^m}(s_{m,1}|\theta^m) + \Delta_1$, thus the offloading power $p_{m,o}(1)$ and local execution power $p_{m,l}(1)$ are determined. Then, VU m allocates the offloading power and local execution power to process the task, while achieving the reward $r_{m,1}$ according to (16). Then, BS adopts the ZF technology to determine the SINR $\gamma_m(1)$. Specifically, BS collects the channel vector of each vehicle, calculates $\boldsymbol{g}_m(1)$ according to (10), and then determines the initial SINR $\gamma_m(1)$ according to (14) under the obtained $\boldsymbol{g}_m(1)$. Afterward, VU m observes the next state $s_{m,2} = [B_m(2), \gamma_m(1), d_m(2)]$. Specifically, VU m calculates $B_m(2)$ according to (11), where $d_{m,l}(1)$ is calculated based on (16)–(17) under $p_{m,l}(1)$ and $d_{m,o}(1)$ is calculated according to (18) under $p_{m,o}(1)$. In addition, VU m receives its SINR $\gamma_m(1)$ from BS. Moreover, VU m calculates $d_m(2)$ according to (2) given the position $d_m(1)$. Then, the tuple $(s_{m,1}, a_{m,1}, r_{m,1}, s_{m,2})$ is stored in the replay buffer. When the number of tuples stored in the replay buffer is less than I, VU m inputs the next state into the actor network and begins the next iteration (lines 7–10).

When the number of the stored tuples is larger than I, the parameters of actor network, critic network, and target networks, i.e., θ^m, ζ^m, $\theta^{m'}$, and $\zeta^{m'}$, are updated literately to maximize $J(\mu_{\theta^m})$. The parameters of the actor network θ^m is updated with the policy gradient, i.e., updating θ^m toward the direction of the gradient of $J(\mu_{\theta^m})$, which is denoted as $\nabla_{\theta^m} J(\mu_{\theta^m})$. Let $Q^{\mu_{\theta^m}}(s_{m,t}, a_{m,t})$ be the action-value function of VU m following policy μ_{θ^m} under $s_{m,t}$ and $a_{m,t}$, which stands for the expected discounted long-term reward of VU m from slot t, i.e.:

$$Q^{\mu_{\theta^m}}\left(s_{m,t}, a_{m,t}\right) := \mathbb{E}_{\mu_{\theta^m}}\left[\sum_{k=t}^{N_j} \gamma^{k-t} r_{m,k}\right]. \tag{23}$$

Silver *et al.* [40] proved that solving $\nabla_{\theta^m} J(\mu_{\theta^m})$ can be substituted by solving the gradient of $Q^{\mu_{\theta^m}}(s_{m,t}, a_{m,t})$, which is denoted as $\nabla_{\theta^m} Q^{\mu_{\theta^m}}(s_{m,t}, a_{m,t})$. However, $Q^{\mu_{\theta^m}}(s_{m,t}, a_{m,t})$ in (20) cannot be calculated by the Bellman equation due to the continuous action space [41]. To address this issue, the critic network adopts DNN parameterized by ζ^m to approximate the action-value function $Q^{\mu_{\theta^m}}(s_{m,t}, a_{m,t})$, and the action-value function approximated by the critic network is denoted as $Q^{\zeta^m}(s_{m,t}, a_{m,t})$.

The iteration in slot t ($t = 1, 2, \ldots, N_j$) to update θ^m, ζ^m, $\theta^{m'}$, and $\zeta^{m'}$ is described as follows when the number of the stored tuples is larger than I. For simplicity, $r_{m,t}$, $s_{m,t}$, $a_{m,t}$, $s_{m,t+1}$, and $a_{m,t+1}$ are expressed as r_m, s_m, a_m, s'_m, and a'_m, respectively. VU m first uniformly samples I tuples from replay buffer to form a minibatch. Let $(s^i_m, a^i_m, r^i_m, s'^i_m)$ ($i = 1, 2, \ldots, I$) be the ith tuple in the minibatch. Then, VU m inputs each tuple into the target actor network, target critic network, and critic network. For tuple i, VU m first inputs s'^i_m into the target actor network and outputs the action $a'^i_m = \mu_{\theta^{m'}}(s'^i_m|\theta^{m'})$, and then VU m inputs s'^i_m and $a'_{m,i}$ into the target critic network and outputs the action-value function $Q^{\zeta^{m'}}(s'^i_m, a'^i_m)$. After that, VU m calculates the target value as

$$y^i_m = r^i_m + \gamma Q^{\zeta^{m'}}\left(s'^i_m, a'^i_m\right)|_{a'^i_m=\mu_{\theta^{m'}}\left(s'^i_m|\theta^{m'}\right)}. \tag{24}$$

Then, the loss function can be calculated as

$$L(\zeta^m) = \frac{1}{I}\sum_{i=1}^{I}\left[y^i_m - Q^{\zeta^m}(s^i_m, a^i_m)\right]^2 \tag{25}$$

and the critic network updates its parameters using $\nabla_{\zeta^m}L(\zeta^m)$ to minimize the loss function through gradient descending [42] (lines 11–13).

Similarly, the actor network updates its parameters using $\nabla_{\theta^m}J(\mu_{\theta^m})$ to maximize $J(\mu_{\theta^m})$ through gradient ascending [42], where $\nabla_{\theta^m}J(\mu_{\theta^m})$ is calculated by the action-value function, which is approximated by the critic network, i.e., (line 14)

$$\begin{aligned}&\nabla_{\theta^m}J(\mu_{\theta^m})\\ &\approx \frac{1}{I}\sum_{i=1}^{I}\nabla_{\theta^m}Q^{\zeta^m}\left(s^i_m, a^\mu_m\right)|_{a^\mu_m=\mu_{\theta^m}(s^i_m|\theta^m)}\\ &= \frac{1}{I}\sum_{i=1}^{I}\nabla_{a^\mu_m}Q^{\zeta^m}\left(s^i_m, a^\mu_m\right)|_{a^\mu_m=\mu_{\theta^m}(s^i_m|\theta^m)}\\ &\quad\cdot\nabla_{\theta^m}\mu_{\theta^m}\left(s^i_m|\theta^m\right)\end{aligned} \tag{26}$$

here the chain rule is applied since that $a^\mu_m = \mu_{\theta^m}(s^i_m|\theta^m)$ is the input of Q^{ζ^m}.

At the end of slot t, VU m updates the parameters of the target actor network and target critic network as

$$\zeta^{m'} \leftarrow \tau\zeta^m + (1-\tau)\zeta^{m'} \tag{27}$$

$$\theta^{m'} \leftarrow \tau\theta^m + (1-\tau)\theta^{m'} \tag{28}$$

where τ is a constant satisfying $\tau \ll 1$ (line 15).

Finally, VU m inputs s'_m into the actor network and begins the iteration in next slot. The episode is finished when the number of iterations reaches N_j. Then, VU m will initialize $B_m(1)$, $\gamma_m(0)$, and $d_m(1)$ and start the next episode. The algorithm will finally terminate when the number of episodes reaches $K_{\max}$, which means that the training stage is finished. The flow diagram of the DDPG algorithm is shown in Fig. 2.

Algorithm 2: Testing Stage for the DDPG-Based Framework

```
1 for episode from 1 to $K'_{max}$ do
2     Reset simulation parameters for the VEC system model;
3     Receive initial observation state $s_1$;
4     for time slot t from 1 to $N_j$ do
5         Generate the power for local process and computation offloading according to the optimal policy $a_m = \mu_{\theta^m}(s_m|\theta^{m*})$ ;
6         Execute action $a_m$, observe reward $r_m$ and new state $s'_m$ from the system model.
```

B. Testing Stage

The testing stage omits the critic network, target actor network, and target critic network in the training stage and employs the optimal policy with optimized parameter θ^{m*} to test the performance. The pseudocode of the testing stage is shown in Algorithm 2.

C. Complexity Analysis

In this section, we analyze the complexity of the DDPG algorithm. Let G_A and G_C be the computational complexity of computing gradients for the actor network and critic network, respectively, and U_A and U_C be the computation complexity of updating parameters for the actor network and critic network, respectively. Since the architecture of the target actor network and target critic network is the same as the actor network and critic network, the complexity of updating parameters for target networks is the same as actor network and critic network. The complexity of the DDPG algorithm is affected by the number of slots for training. For each slot for training, the actor network and critic network compute gradients and update parameters, while the target networks update parameters without computing gradients. Thus, the complexity of the DDPG algorithm in a slot for training is calculated as $O(G_A+G_C+2U_A+2U_C)$. In addition, the training and updating parameters process will not be activated until the tuples stored in replay buffer are larger than I, and the algorithm loops for Kmax episodes and each episode includes N_j slots for training; thus, the complexity of the DDPG algorithm is calculated as $O((K_{\max}\cdot N_j - I)(G_A + G_C + 2U_A + 2U_C))$.

VI. Simulation Results and Analysis

In this section, we conduct simulation experiments[1] to verify the effectiveness of the optimal power allocation scheme, i.e., the optimal policy, in the training and testing stage, respectively. The simulation tool is Python 3.6. The scenario is described in the system model. In the simulation experiments, both actor network and critic network are the four-layer fully connected DNN with two hidden layers, which are equipped with 400 and 300 neurons, respectively. The Adam

[1]The source code has been released at: https://github.com/qiongwu86/VEC_DRL_Doppler.git.

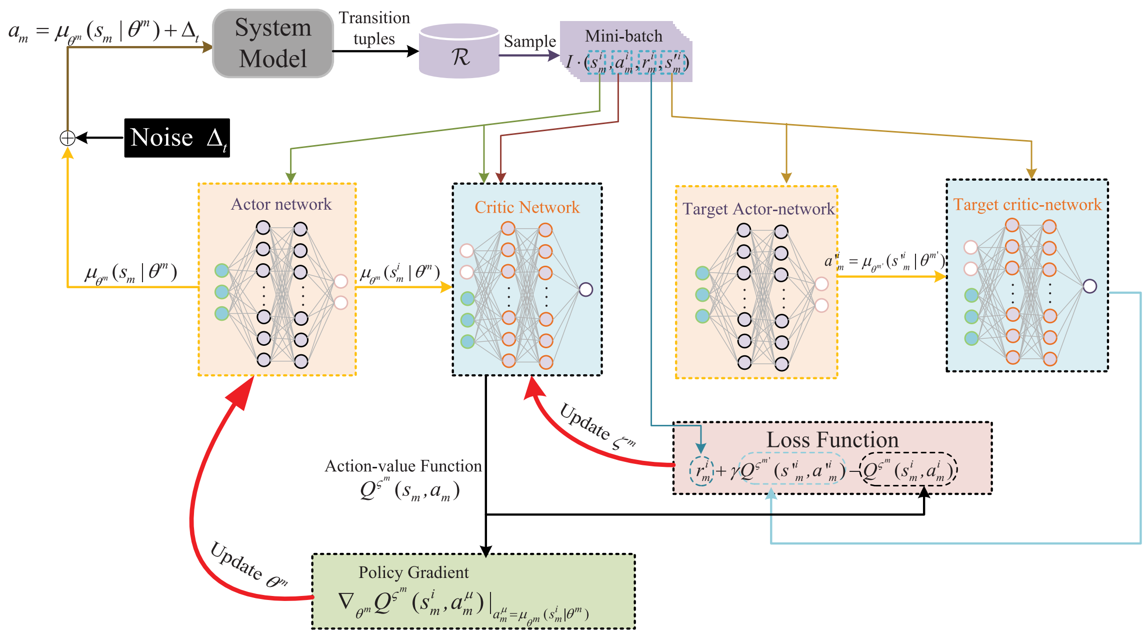
$a_m = \mu_{\theta^m}(s_m \mid \theta^m) + \Delta_t$
System Model
Transition tuples
$\mathcal{R}$
Sample
Mini-batch
$I \cdot (s^i_m, a^i_m, r^i_m, s'^i_m)$
Noise Δ_t
Actor network
Critic Network
Target Actor-network
Target critic-network
$\mu_{\theta^m}(s_m \mid \theta^m)$
$\mu_{\theta^m}(s^i_m \mid \theta^m)$
$a'^i_m = \mu_{\theta^{m'}}(s'^i_m \mid \theta^{m'})$
Update ς^m
Loss Function
$r^i_m + \gamma Q^{\varsigma^{m'}}(s'^i_m, a'^i_m) - Q^{\varsigma^m}(s^i_m, a^i_m)$
Action-value Function
$Q^{\varsigma^m}(s_m, a_m)$
Update θ^m
Policy Gradient
$\nabla_{\theta^m} Q^{\varsigma^m}(s^i_m, a^\mu_m)|_{a^\mu_m = \mu_{\theta^m}(s^i_m|\theta^m)}$

Fig. 2. Flow diagram of DDPG.

TABLE II
VALUES OF THE PARAMETERS IN THE EXPERIMENTS

Parameters of System Model			
Parameter	**Value**	**Parameter**	**Value**
σ_R^2	10^{-9} W	h_r	-30 dB
Λ	7 m	W	1 MHz
τ_0	20 ms	κ	10^{-28}
v_1	20 m/s	v_2	25 m/s
v_3	30 m/s	w	5 m
L_m	500 cycles/bit	λ_m	3 Mbps
H	10 m	N	4
D	500 m	$P_{max,l}$	1 W
$P_{max,o}$	1 W	F_{max}	2.15 GHz
J	3	w_0	5 m
T_s	4 s	N_{max}	15

Parameters of DDPG			
Parameter	**Value**	**Parameter**	**Value**
ω_1	0.9	ω_2	0.1
γ	0.99	τ	0.001
K_{max}	2000	I	64
K'_{max}	10	$\|\mathcal{R}\|$	2.5×10^5

optimization method [43] is adopted to update the parameters of the critic network and actor network with learning rate as 10^{-3} and 10^{-4}, respectively.

The noise Δ_t for exploration follows the Ornstein–Uhlenbeck (OU) process [44] with the decay rate and variation as 0.15 and 0.02, respectively. The size of experience replay buffer is $|\mathcal{R}|$. The task arrivals at each slot follow the Poisson distribution with mean task arrival rate λ_m. The maximum local process power $P_{m,l}$ is calculated according to (17) given the maximum allowable CPU frequency $F_{\max}$. The small scale fading of VU m is initialized as $\boldsymbol{h}^s_m(0) \sim \mathcal{CN}(\mathbf{0}, \boldsymbol{I}_N)$. The target VU, i.e., VU m, moves on lane 2 with velocity v_2. Three other VUs on each of three lanes drive into the coverage of BS when $d_m(t) = 0$. The remaining parameters and algorithm parameters are shown in Table II.

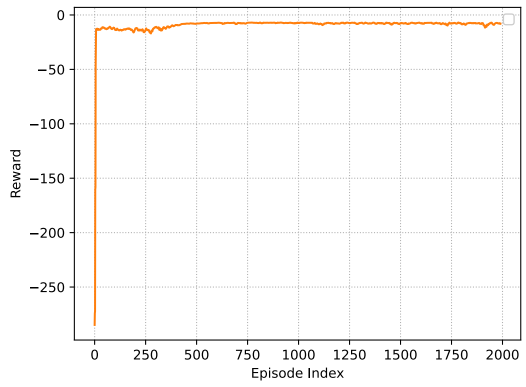

Fig. 3. Learning curve.

A. Training Stage

Fig. 3 shows the learning curve of the training process, which reflects the average reward in each slot under different episodes. It can be seen that the average reward rises rapidly from episode 0 to episode 10, and then the uptrend of the curve slows down from episode 10 to 600, which reflects that VU m is learning the policy efficiently toward the optimal reward. Then, the reward turns to be stable with little jitter, because the policy is adjusted slightly due to the exploration noise to prevent the policy from converging to local optimal value.

B. Testing Stage

In the testing stage, VU m adopts learned policy in the training stage to test the performance. Figs. 4–5 compare the testing performance, including the power consumption, buffer length, and reward under the optimal policy with that under greedy local execution first (GD-Local) and greedy offload first (GD-Offload) policy, where the performance value is obtained through averaging the results obtained in 100 episodes. GD-Local policy and GD-Offload policy are introduced as follows.

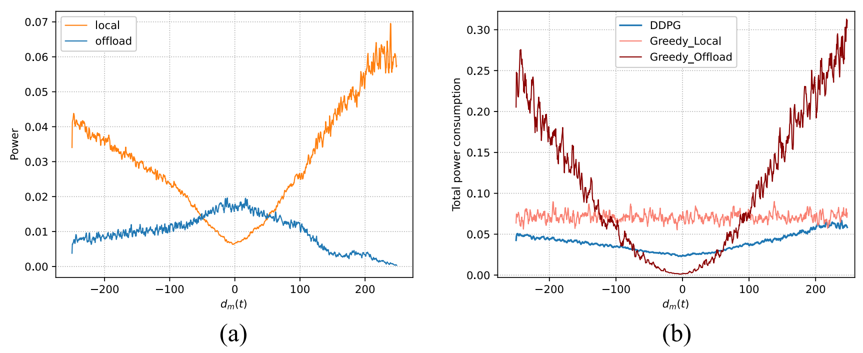

Fig. 4. Power. (a) Optimal power allocation; (b) Total power consumption.

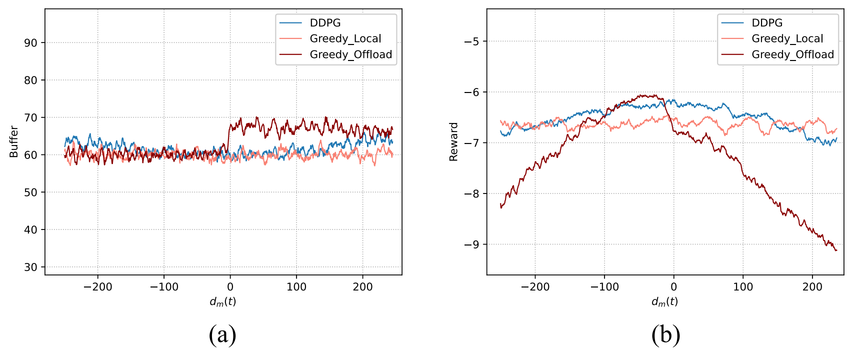

Fig. 5. Performance. (a) Buffer length. (b) Reward.

1) *GD-Local Policy:* VU m first adopts the maximum local execution power to process tasks at each slot through local execution, while the remaining tasks are processed through offloading.
2) *GD-Offload Policy:* VU m first adopts the maximum offloading power to process tasks through offloading at each slot, while the remaining tasks are processed through local execution.

Fig. 4(a) and (b) shows the test results of power allocation and power comparison under three policies. Fig. 4(a) compares the local execution power with offloading power under the optimal policy. It is seen that when $d_m(t) < 0$, the local execution power decreases obviously and the offloading power increases slowly. After that, the local execution power increases obviously and the offloading power decreases slowly. It is because that according to (5), the channel condition is impacted by path loss. When VU m is getting close to the BS, the path loss is decreasing, thus leading to the better channel condition. Therefore, VU m will consume more offloading power and less local execution power to process more tasks when it gets close to the BS. On the contrary, VU m will consume less offloading power and more local execution power to process more tasks when it gets far away from the BS. The local execution power increases sharply after $d_m(t) > 0$. This is because that three other vehicles drive into the BS's coverage area when $d_m(t) = 0$, which imposes interference on VU m and incurs the deteriorated channel condition. In this case, VU m consumes more local execution power and less offloading power to process more tasks. Fig. 4(b) shows the total power consumption under three policies. It can be seen that similar to the local execution power under the optimal policy in Fig. 4(a), the total power under the optimal policy and GD-Offload decreases when $d_m(t) < 0$, and increases when $d_m(t) > 0$. This is because that the total power of the optimal policy is composed of both the local execution power and offloading power at each slot in Fig. 4(a), where the local execution power overweighs offloading power. For the GD-Offload policy, the offloading power always keeps the maximum value; thus, VU m will process more tasks through offloading when it gets close to the BS, which results in less local execution power consumption in the GD-Offload policy. In contrast, it consumes more local execution power when the VU gets far away from the BS. Moreover, it can be seen that the total power almost does not change at each distance in the GD-Local policy. This is because that the local execution power always keeps the maximum power, while the offloading power is much smaller than the local execution power, which can be ignored in the total power under the GD-Local policy.

Fig. 5(a) and (b) compares the testing performances at each distance, including buffer length and reward under three policies. Fig. 5(a) shows the buffer length under three policies. The buffer length of GD-Offload is increased when $d_m(t) = 0$. This is because that more tasks cached in the buffer owing to the deteriorated channel condition. Moreover, the buffer length of the optimal policy also fluctuates around the mean of tasks arrival, which means that VU m can process the tasks in time without increasing the buffer length when other VUs drive into the coverage of BS. Fig. 5(b) compares the rewards of VU m under the three policies. It can be seen that the reward of the

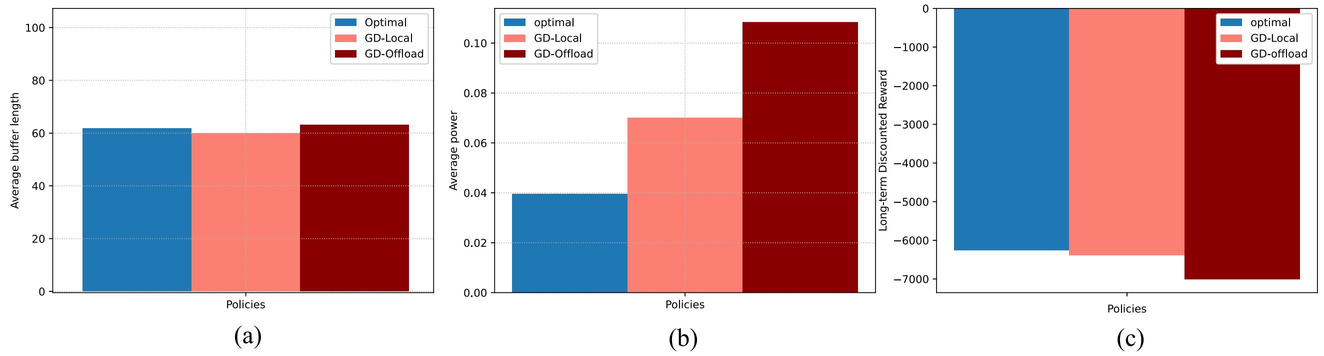

Fig. 6. (a) Average buffer length. (b) Average power consumption. (c) Long-term discounted reward.

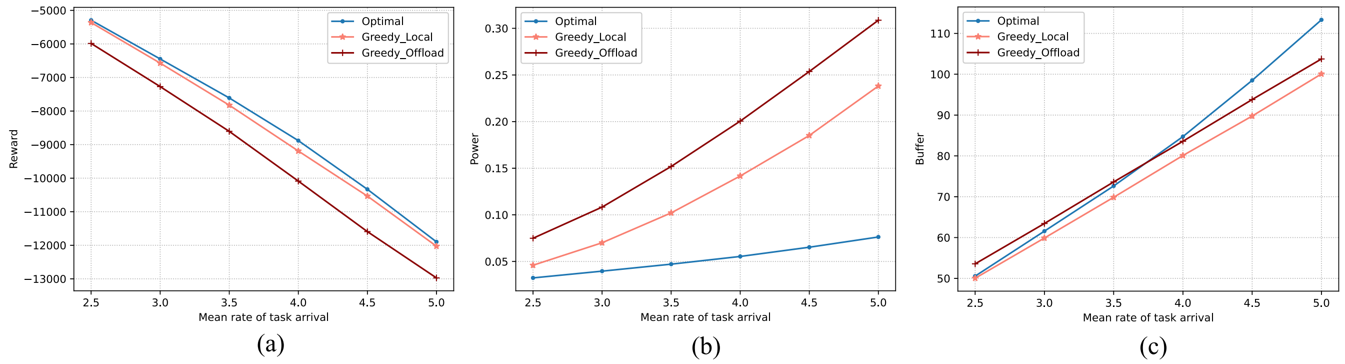

Fig. 7. Performance versus task arrival. (a) Long-term discounted reward. (b) Power consumption. (c) Buffer length.

optimal policy is usually larger than that of other two policies owing to the adaptive power allocation.

Fig. 6(a) shows the average buffer length under three policies, where the average buffer length is obtained by averaging over the all slots in Fig. 5(a). It can be seen that the average buffer lengths under three policies are nearly equal to the mean of tasks arrival rate in each slot and do not change significantly. Fig. 6(b) shows the average power consumption under three policies, where the average power consumption are obtained by averaging over the all slots in Fig. 4(a). It can be seen that compared with GD-Local, the average power consumption of the optimal policy is reduced by 44.2%. Compared with GD-Offload, the average power consumption of the optimal policy is reduced by 63.0%.

Fig. 6(c) compares the long-term discounted reward under the three policies. As one can see, the optimal policy always has a higher long-term discounted reward than other policies. This is because the optimal policy can adaptively adjust power allocation to maximize the long-term discounted reward.

Fig. 7(a)–(c) illustrates the long-term discounted reward, power consumption, and buffer length of the three policies under different task arrivals, respectively. It can be seen that the long-term discounted rewards of the three policies decrease as the task arrival rate increases. As seen, increasing task arrival will lead to more power consumption and longer buffer length, thus degrading the reward according to (21). It also can be seen that the optimal policy outperforms GD-Local and GD-Offload policies in terms of power consumption and long-term discounted reward, but it has a slightly higher buffer length than other policies. This is because the objective of the optimal policy is to maximize the long-term discounted reward by making a tradeoff between power the consumption and buffer length, which may lead to the compromise for buffer length.

VII. Conclusion

In this article, we considered the stochastic task arrival and uncertain channel condition caused by both the MIMO-NOMA channel interference and VU mobility in VEC, and proposed a decentralized power allocation scheme based on the DRL to maximize the long-term reward, including the power consumption and delay. We first formulated the system model and then constructed a DRL framework where the state is defined as the local observations. The DDPG algorithm has been adopted to learn the optimal policy. Extensive simulations have demonstrated the optimal policy outperforms the other existing policies. According to the theoretical analysis and simulation results, the conclusions can be made as follows.

1) Since the channel condition is impacted by the mobility of VU, more offloading power and less local execution power are consumed to process more tasks when the VU gets close to the BS. On the contrary, less offloading power and more local execution power are consumed when it gets far away from the BS.
2) In order to maximize the long-term discounted reward, the optimal policy emphasizes the power consumption and compromises the buffer length as compared to the

GD-Local and GD-Offload policies given different tasks arrival rates.

For future work, we will consider the data freshness to design the offloading scheme in VEC.

References

[1] A. Bonadio, F. Chiti, and R. Fantacci, "Performance analysis of an edge computing SaaS system for mobile users," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2049–2057, Feb. 2020.

[2] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay-sensitive task offloading in the 802.11p-based vehicular fog computing systems," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 773–785, Jan. 2020.

[3] Q. Wu, H. Ge, H. Liu, Q. Fan, Z. Li, and Z. Wang, "A task offloading scheme in vehicular fog and cloud computing system," *IEEE Access*, vol. 8, pp. 1173–1184, 2020.

[4] J. Wang, C. Jiang, K. Zhang, T. Q. S. Quek, Y. Ren, and L. Hanzo, "Vehicular sensing networks in a smart city: Principles, technologies and applications," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 122–132, Feb. 2018.

[5] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[6] X. Hou *et al.*, "Reliable computation offloading for edge-computing-enabled software-defined IoV," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7097–7111, Aug. 2020.

[7] Q. Wu, S. Xia, P. Fan, F. Qiang, and Z. Li, "Velocity-adaptive V2I fair-access scheme based on IEEE 802.11 DCF for platooning vehicles," *Sensors*, vol. 18, no. 12, p. 4198, 2018.

[8] X. Liu, T. Huang, N. Shlezinger, Y. Liu, J. Zhou, and Y. C. Eldar, "Joint transmit beamforming for multiuser MIMO communications and MIMO radar," *IEEE Trans. Signal Process.*, vol. 68, pp. 3929–3944, Jun. 2020.

[9] L. P. Qian, B. Shi, Y. Wu, B. Sun, and D. H. K. Tsang, "Noma-enabled mobile edge computing for Internet of Things via joint communication and computation resource allocations," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 718–733, Jan. 2020.

[10] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE Trans. Wireless Commun.*, vol. 9, no. 11, pp. 3590–3600, Nov. 2010.

[11] B. Di, L. Song, Y. Li, and Z. Han, "V2X meets NOMA: Non-orthogonal multiple access for 5G-enabled vehicular networks," *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 14–21, Dec. 2017.

[12] Q. Wu, H. Ge, P. Fan, J. Wang, Q. Fan, and Z. Li, "Time-dependent performance analysis of the 802.11p-based platooning communications under disturbance," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15760–15773, Dec. 2020.

[13] J. Zheng and Q. Wu, "Performance modeling and analysis of the IEEE 802.11p EDCA mechanism for VANET," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2673–2687, Apr. 2016.

[14] Q. Wu, Z. Wan, Q. Fan, P. Fan, and J. Wang, "Velocity-adaptive access scheme for MEC-assisted platooning networks: Access fairness via data freshness," *IEEE Internet Things J.*, early access, Aug. 9, 2021, doi: 10.1109/JIOT.2021.3103325.

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

[16] W. Zhan, C. Luo, J. Wang, G. Min, and H. Duan, "Deep reinforcement learning-based computation offloading in vehicular edge computing," in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, 2019, pp. 1–6.

[17] W. Zhan *et al.*, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[18] H. Wang, H. Ke, G. Liu, and W. Sun, "Computation migration and resource allocation in heterogeneous vehicular networks: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 171140–171153, 2020.

[19] P. Dong, Z. Ning, R. Ma, X. Wang, X. Hu, and B. Hu, "Noma-based energy-efficient task scheduling in vehicular edge computing networks: A self-imitation learning-based approach," *China Commun.*, vol. 17, no. 11, pp. 1–11, Nov. 2020.

[20] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, "Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.

[21] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.

[22] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.

[23] Z. Ning *et al.*, "Joint computing and caching in 5G-envisioned internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5201–5212, Aug. 2021.

[24] 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.

[25] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9637–9650, Oct. 2020.

[26] G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Jan. 2020.

[27] Y. Ren, X. Yu, X. Chen, S. Guo, and X.-S. Qiu, "Vehicular network edge intelligent management: A deep deterministic policy gradient approach for service offloading decision," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, 2020, pp. 905–910.

[28] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3163–3173, Apr. 2019.

[29] Y.-H. Xu, C.-C. Yang, M. Hua, and W. Zhou, "Deep deterministic policy gradient (DDPG)-based resource allocation scheme for NOMA vehicular communications," *IEEE Access*, vol. 8, pp. 18797–18807, 2020.

[30] Y. Pan, M. Chen, Z. Yang, N. Huang, and M. Shikh-Bahaei, "Energy-efficient NOMA-based mobile edge computing offloading," *IEEE Commun. Lett.*, vol. 23, no. 2, pp. 310–313, Feb. 2019.

[31] T. Huang *et al.*, "Joint pilot and data transmission power control and computing resource allocation for the massive MIMO based MEC network," in *Proc. IEEE 19th Int. Conf. Commun. Technol. (ICCT)*, 2019, pp. 860–865.

[32] C. Ding, J.-B. Wang, H. Zhang, M. Lin, and J. Wang, "Joint MU-MIMO precoding and resource allocation for mobile-edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1639–1654, Mar. 2021.

[33] W. Feng, J. Zheng, and W. Jiang, "Joint pilot and data transmission power control and computing resource allocation algorithm for massive MIMO-MEC networks," *IEEE Access*, vol. 8, pp. 80801–80811, 2020.

[34] Y. Jang, J. Na, S. Jeong, and J. Kang, "Energy-efficient task offloading for vehicular edge computing: Joint optimization of offloading and bit allocation," in *Proc. IEEE 91st Veh. Technol. Conf. (VTC2020-Spring)*, 2020, pp. 1–5.

[35] S. J. W. Tang, K. Y. Ng, B. H. Khoo, and J. Parkkinen, "Real-time lane detection and rear-end collision warning system on a mobile computing platform," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, vol. 2, 2015, pp. 563–568.

[36] H. Q. Ngo, E. G. Larsson, and T. L. Marzetta, "Energy and spectral efficiency of very large multiuser MIMO systems," *IEEE Trans. Commun.*, vol. 61, no. 4, pp. 1436–1449, Apr. 2013.

[37] M. Abramowitz and I. A. Stegun, "Handbook of mathematical functions: With formulas, graphs and mathematical tables," *Amer. J. Phys.*, vol. 56, no. 10, pp. 958–962, 1988.

[38] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

[39] C. King, "Fundamentals of wireless communications," in *Proc. IEEE-IAS/PCA Cement Ind. Tech. Conf.*, 2014, pp. 1–7.

[40] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 387–395.

[41] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, p. 1054, Sep. 1998.

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

[43] D. P. Kingma and J. Ba, "ADAM: A method for stochastic optimization," 2015, *arXiv:1412.6980.*

[44] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the brownian motion," *Revista Latinoamericana De Microbiología*, vol. 15, no. 1, pp. 29–35, 1973.

Hongbiao Zhu received the B.S. degree from Jiangnan University, Wuxi, China, in 2016, where he is currently pursuing the M.S. degree.

His current research interests include VEC, MIMO-NOMA, DRL, and data freshness.

Qiong Wu (Member, IEEE) received the Ph.D. degree in information and communication engineering from the National Mobile Communications Research Laboratory, Southeast University, Nanjing, China, in 2016.

From 2018 to 2020, he was a Postdoctoral Researcher with the Department of Electronic Engineering, Tsinghua University, Beijing, China. He is currently an Associate Professor with the School of Internet of Things Engineering, Jiangnan University, Wuxi, China. His current research interest focuses autonomous driving communication technology.

Xiao-Jun Wu (Member, IEEE) received the B.S. degree in mathematics from Nanjing Normal University, Nanjing, China, in 1991, and the M.S. and Ph.D. degrees in pattern recognition and intelligent system from Nanjing University of Science and Technology, Nanjing, in 1996 and 2002, respectively.

From 1996 to 2006, he taught with the School of Electronics and Information, Jiangsu University of Science and Technology, Zhenjiang, China, where he was an exceptionally promoted Professor. He joined the School of Information Engineering (currently renamed as School of Artificial Intelligence and Computer Science), Jiangnan University, Wuxi, China, in 2006, where he is a Professor. He was a Visiting Postdoctoral Researcher with the Centre for Vision, Speech, and Signal Processing, University of Surrey, Guildford, U.K., from 2003 to 2004, under the supervision of Prof. J. Kittler. He has published more than 300 papers in his fields of research. His current research interests are pattern recognition, computer vision, fuzzy systems, neural networks, and intelligent systems.

Prof. Wu won the most Outstanding Postgraduate Award by Nanjing University of Science and Technology. He has won different awards, including international award, national award and provincial award for his research achievements. He was a Fellow of United Nations University, International Institute for Software Technology from 1999 to 2000.

Qiang Fan received the M.S. degree in electrical engineering from the Yunnan University of Nationalities, Kunming, China, in 2013, and the Ph.D. degree in electrical and computer engineering from the New Jersey Institute of Technology, Newark, NJ, USA, in 2019.

He was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA, USA. He currently becomes a Senior Wireless Software Engineer with Wistron Aiedge Inc., San Jose, CA, USA. His current research interests include wireless communications and networking, mobile-edge computing, machine learning, and drone-assisted networking.

Dr. Fan has served as a Reviewer for over 120 journal submissions, such as IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, and IEEE TRANSACTIONS ON COMMUNICATIONS.

Pingyi Fan (Senior Member, IEEE) received the B.S. degree from the Department of Mathematics, Hebei University, Baoding, China, in 1985, the M.S. degree from the Department of Mathematics, Nankai University, Tianjin, China, in 1990, and the Ph.D. degree from the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 1994.

He is currently a Professor with the Department of EE, Tsinghua University. From August 1997 to March 1998, he visited Hong Kong University of Science and Technology, Hong Kong, as a Research Associate. From May 1998 to October 1999, he visited the University of Delaware, Newark, DE, USA, as a Research Fellow. In March 2005, he visited NICT, Tokyo, Japan, as a Visiting Professor. From June 2005 to May 2019, he visited Hong Kong University of Science and Technology for many times and from July 2011 to September 2011, he is a Visiting Professor with the Institute of Network Coding, Chinese University of Hong Kong, Hong Kong. His main research interests include B5G technology in wireless communications, machine learning and AI in wireless communications, information theory in big data analysis, network coding, and network information theory.

Prof. Fan has received some academic awards, including the IEEE ICC20, TAOS20, Globecom14, the WCNC08 Best Paper Awards, the ACM IWCMC10 Best Paper Award, and the IEEE ComSoc Excellent Editor Award for IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS in 2009. He has attended to organize many international conferences, including as the TPC Co-Chair of Chinacom 2020 and IEEE WCNIS 2010 and a TPC Member of IEEE ICC, Globecom, WCNC, VTC, and Infocom. He has served as an Editor of IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, *International Journal of Ad Hoc and Ubiquitous Computing* (Inderscience), *Journal of Wireless Communication and Mobile Computing* (Wiley), and *Electronics and Open Journal of Mathematical Sciences* (MDPI). He is also a reviewer of more than 30 international Journals, including 24 IEEE Journals and 8 EURASIP Journals. He is an Oversea Member of IEICE.

Jiangzhou Wang (Fellow, IEEE) has been a Professor with the University of Kent, Canterbury, U.K., since 2005. His research interest is in the area of mobile communications.

Prof. Wang is a Fellow of the Royal Academy of Engineering, U.K. and IET. He was the Technical Program Chair of the 2019 IEEE International Conference on Communications (ICC2019), Shanghai, the Executive Chair of the IEEE ICC2015, London, and the Technical Program Chair of the IEEE WCNC2013.