# Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks

Yi Liu, Huimin Yu, Shengli Xie, *Fellow, IEEE*, and Yan Zhang, *Senior Member, IEEE*

***Abstract*—Mobile Edge Computing (MEC) is a promising technology to extend the diverse services to the edge of Internet of Things (IoT) system. However, the static edge server deployment may cause "service hole" in IoT networks in which the location and service requests of the User Equipments (UEs) may be dynamically changing. In this paper, we firstly explore a vehicle edge computing network architecture in which the vehicles can act as the mobile edge servers to provide computation services for nearby UEs. Then, we propose as vehicle-assisted offloading scheme for UEs while considering the delay of the computation task. Accordingly, an optimization problem is formulated to maximize the long-term utility of the vehicle edge computing network. Considering the stochastic vehicle traffic, dynamic computation requests and time-varying communication conditions, the problem is further formulated as a semi-Markov process and two reinforcement learning methods: $Q$-learning based method and deep reinforcement learning (DRL) method, are proposed to obtain the optimal policies of computation offloading and resource allocation. Finally, we analyze the effectiveness of the proposed scheme in the vehicular edge computing network by giving numerical results.**

***Index Terms*—Vehicle edge computing, resource allocation, IoT, deep reinforcement learning.**

## I. Introduction

The EVOLVING Internet of Things (IoT) system requires highly scalable infrastructure to adaptively provide proper services for different applications in distinct domains [1]–[3]. The crucial requirements of the IoT is full scalability and interoperability of interconnected devices while providing a high degree of flexibility. Allowing the high integration of advanced communication and computation technologies, such as high density small-cell deployment, massive MIMO, remote cloud service, the IoT network will achieve high magnitudes of connected objects and diverse services by consuming less energy [4]–[7]. However, the long-distance transmission from the static/mobile devices to the base stations or cloud server may cause severe service delay and extra transmission energy cost. Mobile Edge Computing (MEC) is regarded as a key technology and architectural concept to improve the computation offloading efficiency [8]–[11]. With regard to the IoT services scenario, MEC can provide low latency, proximity, high bandwidth and computing agility in the computation process. Specially, for the ultra-dense IoT scenario, the computation offloading and power allocation in MEC can enhance the energy efficiency of the realistic IoT system [12]–[14].

In MEC based IoT network, the devices can offload all/part of the computation tasks to the MEC server which can speed up the processing of the tasks and save energy for devices [15]–[17]. Then, the main technical problem becomes whether/when/how many computation tasks should be offloaded. Numerous literatures are devoted to design the optimal strategy to solve this problem under different performance requirements [18]–[21]. Considering the long-time energy efficiency while using IoT network, an efficient edge computing infrastructure is proposed in [18]. Due to stochastic task arrivals and wireless channels, congested air interface, and prohibitive feedbacks from thousands of IoT devices, authors in [19] generate asymptotically optimal schedules tolerant to out-of-date network knowledge, thereby relieving stringent requirements on feedbacks. The optimal schedule and energy efficient resource allocation policies for MEC are proposed in [20] and [21], respectively.

Since the vehicle is an important type of User Equipment (UE) in IoT system, the vehicular edge computing network structure and related resource allocation methods are studied by many researchers [22]–[27]. C. Wang *et al.*, propose a scalable SDN-enabled MEC architecture that integrates a heterogeneous vehicular network to decrease the overall delay and offload the traffic load from the backbone network [22]. To reduce both the latency and the transmission cost of the computation offloading, a cloud-based MEC offloading framework is proposed for vehicular networks in [23]. In [24], based on the MEC-assisted vehicular offloading mode, a target server selection policy is presented to improve task offloading reliability in the case of vehicular

Manuscript received December 28, 2018; revised April 20, 2019 and July 12, 2019; accepted July 14, 2019. Date of publication August 14, 2019; date of current version November 12, 2019. This work was supported in part by the programs of National Natural Science Foundation of China under Grant 61773126, Grant 61727810, Grant 61701125, Grant 61603099 and Grant 61973087, in part by Pearl River S&T Nova Program of Guangzhou, 201806010176, and in part The European Unions Horizon 2020 research and innovation programme under the Marie Skodowska-Curie Grant agreement 824019. The review of this article was coordinated by Dr. K. Bian. *(Corresponding author: Shengli Xie.)*

Y. Liu is with the School of Automation, Guangdong University of Technology, Key Lab. of Ministry of Education and Guangdong Province Key Lab. of IoT Information Technology, Guangzhou 510006, China (e-mail: yi.liu@gdut.edu.cn).

H. Yu is with the College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China (e-mail: yhm@hunnu.edu.cn).

S. Xie is with the School of Automation, Guangdong University of Technology, and State Key Lab. of Precision Electronic Manufacturing Technology and Equipment, Guangzhou 510006, China (e-mail: shlxie@gdut.edu.cn).

Y. Zhang is with the University of Oslo, Oslo 0315, Norway (e-mail: yanzhang@ieee.org).

Digital Object Identifier 10.1109/TVT.2019.2935450

data transmission failure. Some novel MEC-based vehicular networks are proposed in [25], [26], in which the computation offloading policies are carefully designed according to different scenarios. The reinforcement learning is used for long-term resource provision in vehicular cloud to deal with dynamic demands for the resources and stringent QoS requirements [27]. However, in the existing literatures, the vehicles play the role as users in the MEC network in which the edge servers are statically deployed and may cause "service hole" due to the explosion of service requests of tremendous number of UEs.

The focus of this paper is to design a Vehicle Edge Computing (VEC) network in which the vehicles are able to provide computation services as well as the traditional edge servers. As the traditional edge server, generally connected to road side units, small-cell base stations, etc., has fixed locations, the proposed architecture can extend the computation services range and improve flexibility of the MEC network. Then, we propose an efficient computation offloading scheme for UEs while considering the delay of the computation tasks generated by UEs. Accordingly, we formulate an optimization problem to maximize the total utility of the proposed VEC network.

To solve the problem, the stochastic traffic and communication uncertainty in vehicular communication environment should be carefully addressed. The $Q$-learning method is one of the model-free reinforcement learning methods which are not based on the environment elements are already known [32]. Such feature makes $Q$-learning method suitable for solving the proposed problem in the vehicle edge computing network. The crucial part of $Q$-learning is to accurately and efficiently estimate the $Q$ value, which may lead to the curse of dimensionality as the increasing of state space. Deep reinforcement learning (DRL), which approximates the $Q$-function by using deep neural networks, has more advantageous than Q-learning for providing greater performance and more robust learning [33]–[36]. Hence, the proposed problem is further formulated as a semi-Markov process and two reinforcement learning methods: $Q$-learning based method and deep reinforcement learning (DRL) method are proposed to determine the policies of computation offloading and resource allocation.

In this paper, we propose a VEC network to enhance the flexibility and scalability of the MEC based IoT system with main contributions summarized as follows:

1) We propose a vehicle edge computing network architecture in which the vehicles can provide computation services for UEs as well as the traditional edge server.
2) We propose an efficient offloading scheme for the vehicle edge computing network while considering both delay and limited computation capabilities of vehicles and edge servers. Accordingly, we formulate an optimization problem to maximize the total utility of the vehicle edge computing network.
3) Taking into account of the stochastic traffic and uncertain communication conditions, we reformulated the proposed problem as a semi-Markov process and propose $Q$-learning based reinforcement learning method and DRL method to find the policies of computation offloading and resource allocation.

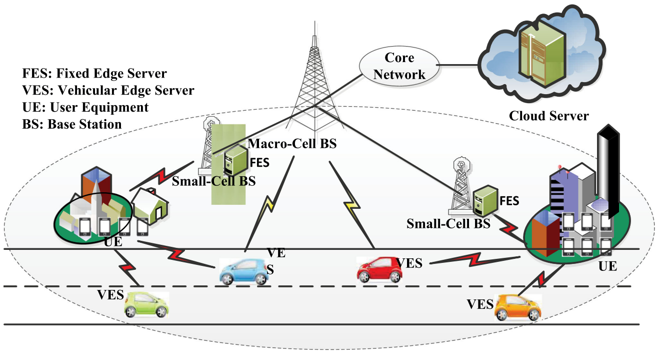


Fig. 1. System model of vehicle-assisted MEC Network.

The remainder of this paper is organized as follows. In Section II, the system model of VEC network is introduced. The efficient computation offloading scheme and the utility maximization problem are described in Section III. The reinforcement learning solutions: $Q$-learning method and DRL method are described in Section IV. Section V presents the numerical results of the proposed methods. Finally, the paper is concluded in Section IV.

## II. System Model

In this section, the system model of the VEC network is presented. We first propose the VEC network infrastructure, then we describe the communication model and computation model in details.

### A. Vehicle Edge Computing Network Infrastructure

*1) Network Infrastructure:* We consider an urban environment of one macro-cell with a macro base station (BS) and $N$ small cells, each of which with a small-cell BS, as shown in Fig. 1. The macro cell is connected to the cloud server through the core network of the cellular communications. One fact is that the small-cell BS and connected edge server may not satisfy the communication and computation demands burst of all UEs in MEC network. However, the huge number of vehicles in urban environment can lead to frequent encounter with UEs and provide computation services for them. Hence, to extend the range of computation services, the edge computing servers are classified as two types: fixed edge server (FES) and vehicular edge server (VES). FES is placed in the small-cell BSs, with fixed location, which is connected to the MBS in wired manner. Meanwhile, any vehicles can be the candidates of the VES, which can connect with UEs within the valid communication range by employing IEEE 802.11p or LTE protocol.

*2) Vehicle Edge Computing Operation Model:* In our model, the FES and VES are managed by a VEC operator which can purchase physical resources (e.g., spectrum and backhaul) from realistic entities to provide **payable** services for UEs. If the UEs decide to offload the computation task, VEC operator will choose VES or FES to execute the task. In such case, we denote $d_i \in \{0, 1\}, \forall i$ as the computation offloading decision of UE $i$. $v_i \in \{0, 1\}, \forall i$ and $o_i \in \{0, 1\}, \forall i$ are defined to denote the decision that the computation tasks are offloading to VES and FES, respectively. Specifically, $d_i = 1$ if UE $i$ decide to compute its task locally and $d_i = 0$ if the UE chooses to offload the computation task to the VEC operator; $v_i = 1$ if UE $i$ offload computation task to VES, otherwise $v_i = 0$; $o_i = 1$ if UE $i$ offload computation task to FES, otherwise $o_i = 0$. So, we have $\mathbf{d} = \{d_i\}_{i\in\mathcal{I}}$, $\mathbf{v} = \{v_i\}_{i\in\mathcal{I}}$ and $\mathbf{o} = \{o_i\}_{i\in\mathcal{I}}$ as the offloading decision profiles of UEs, where $I$ is the number of UEs with computation requests in each small cell.

In the proposed vehicle edge computing model, the service range of the VES can be extended more close to the UEs. However, the stochastic traffic leads to the random number of VES candidates in each time instant. More complicated, the VES candidates may have different incentive desirability to become a formal VES for providing computation services. Hence, to evaluate the number of available VESs for a specific UE, we model the transition of the number of available VESs for one UE as a Markov chain. Let the number of available VES is the random variable $K$ and $K \in \Phi = \{1, 2, \ldots, V\}$, where $V$ is the total number of vehicles in the network. Then, the state transition probability matrix of the number of available VESs for UE $i$ is defined as:

$$\Theta_{VES}(t) = [\omega_{x_i y_i}(t)]_{V\times V} \tag{1}$$

where $\omega_{x_i y_i}(t) = \mathbf{Pr}(K(t) = y_i \mid K(t) = x_i)$ and $x_i, y_i \in \Phi$.

### B. Communication Model

*1) UEs to VESs:* We consider the UEs communicate with VESs using one-hop ad hoc network structure. As the interference from the vehicles reusing the same channel with the transmitting vehicle, the spectrum efficiency for the communication link between the $k$th VES and the UE $i$ is

$$e_{i,k}^{VES} = \log_2\left(1 + \frac{p_i g_{i,k}}{\sum_{j=1}^{I}\sum_{l=1,l\neq k}^{K} p_j g_{j,l} + \sigma}\right) \tag{2}$$

where $p_i$ denotes the transmission power of UE $i$, $g_{i,k}$, $g_{j,l}$ are the channel gain of wireless propagation channel between UE $i$ and VES $k$, UE $j$ and VES $l$, respectively. $\sigma$ is the power of white Gaussian noise.

We denote $\eta_{i,k} \in [0, 1]$ as the percentage of the spectrum allocated to the $i$th UE by the $k$th VES. Then, the data rate of UE $i$ served by VES, $R_{VES,i}$ is calculated as

$$R_{VES,i}^{k} = \eta_{i,k} B e_{i,k}^{VES} \tag{3}$$

where $B$ is the bandwidth of available spectrum between UEs and VESs.

*2) UEs to FESs:* The UEs also can offload computation tasks to the FESs through small-cell BS by 5G network. In this paper, we consider UEs use orthogonal spectrum in each small-cell, hence no interference is caused among UEs in one small cell. However, the interference is from the neighboring small-cells. Hence, the spectrum efficiency for the communication link between the $n$th FES and the UE $i$ is

$$e_{i,n}^{FES} = \log_2\left(1 + \frac{p_i g_{i,n}^{F}}{\sum_{j=1}^{I}\sum_{m=1,m\neq n}^{K} p_j g_{j,m}^{F} + \sigma^{F}}\right) \tag{4}$$

where $p_i$ denotes the transmission power of UE $i$, $g_{i,n}^{F}$, $g_{j,m}^{F}$ are the channel gain of wireless propagation channel between UE $i$ and FES $n$, UE $j$ and FES $m$, respectively. $\sigma^{F}$ is the power of additive white Gaussian noise in the FES communication case. We denote $\theta_{i,n} \in [0, 1]$ as the percentage of spectrum allocated to the $i$th UE by the $n$th FES. Therefore, the data rate of UE $i$ served by FES $n$, $R_{FES,i}$ is calculated as

$$R_{FES,i}^{n} = \theta_{i,n} B_0 e_{i,n}^{FES}, \tag{5}$$

where $B_0$ is the bandwidth of available spectrum between UEs and FESs.

### C. Computation Model

In the computation offloading model, the computation task can be locally computed by the UEs or executed by VES/FES via computation offloading. We denote the computation task from UE $i$ as $S_i \triangleq (H_i, Z_i, T_i^{\max})$. Here, $H_i$ denotes the size of computation data, $Z_i$ stands for the amount of the computation resources required to finish the task $S_i$, and $T_i^{\max}$ is the maximum latency of the task. We next discuss the delay which is defined as the computation execution time of both local MEC and VEC computing methods.

*1) Local Computing:* For the local computing method, each UE calculate the computation task $S_i$ by using the computation resource of UE $i$, which is denoted as $C_{local,i}$. Here, we consider different UEs have distinct computational capabilities. The computation time $T_i$ of task $S_i$ executed by UE $i$ is expressed as

$$T_i^{local} = \frac{Z_i}{C_{local,i}}. \tag{6}$$

*2) VES Computing:* For the VES computing method, the $i$th UE can offload the computation task $S_i$ to the VES through wireless connection between users and vehicles. Then, the VES will compute the task for the UEs. Hence, the time cost of executing the task includes two parts: communication time and computation time. The communication time costs are depending on the size of computation input data $H_i$ and the data rate of UE $i$ served by VES $R_{VES,i}$, hence we have

$$T_{i,comm}^{VES} = \frac{H_i}{R_{VES,i}^{k}}. \tag{7}$$

Let $C_{VES,i}$ denote the computation resource of the VES assigned to UE $i$. Then, the computation time costs for task $S_i$ can be calculated as

$$T_{i,comp}^{VES} = \frac{Z_i}{C_{VES,i}^{k}}. \tag{8}$$

Therefore, the total execution time of the task of UE $i$ served by VES is given by

$$T_i^{VES} = T_{i,comm}^{VES} + T_{i,comp}^{VES}. \tag{9}$$

*3) FES Computing:* For the FES computing method, the $i$th UE offloads its computation task $S_i$ via small-cell BS to its associated FES. Then, the FES will execute the computation task for UE $i$. Similar to the VES computing, the time costs for executing the computation task of UE $i$ consists of communication time and computation time. The communication time is calculated as

$$T_{i,comm}^{FES} = \frac{H_i}{R_{FES,i}^n} \tag{10}$$

where $R_{FES,i}^n$ denote the data rate of UE $i$ served by FES $n$. Let $C_{FES,i}$ denote the computation capability (i.e., CPU cycles per second) of the FES allocated to UE $i$. Then, the computation time of the FES on task $S_i$ is given as

$$T_{i,comp}^{FES} = \frac{Z_i}{C_{FES,i}^n}. \tag{11}$$

Therefore, the total computation time of the task of UE $i$ served by FES is given by

$$T_i^{FES} = T_{i,comm}^{FES} + T_{i,comp}^{FES}. \tag{12}$$

## III. Vehicle Edge Computing Offloading and Resource Allocation

### A. Vehicle-Assisted Computing Offloading Scheme

Based on the proposed system model, the vehicles-assisted computing offloading scheme can be described as follows: At time instant $t$, upon UE $i$ generates a computation task, it transmits a notification message to the VEC operator. By knowing the information of the computation task $S_i$, the VEC operator will calculate the utilities of executing the task by local computing, VES computing and FES computing, respectively. By solving the optimization problem of maximizing the utility, which will be descried in Section IV in details, the VEC operator will obtain the decision of using which computing methods as well as the communication and computing resources allocation policies. Then, the operator executes the computation task and sends the results to the UE when computing finished. If the task cannot be finished by the selected methods during given time period $T_i^{\max}$, the VEC operator will send a failure beacon to the UE $i$ and start a new round of computing process.

### B. VEC Network Utility

The utility of the VEC network is consisted by two parts: the communication utility and the computation utility.

*1) Communication Utility:* The VEC operator charges UE $i$ for transmitting computation task to VES or FES, and the unit price of transmitting is defined as $a_i(t)$ per bps at time slot $t$, respectively. Meanwhile, the VEC operator rents spectrum and backhaul from wireless network in both VES and FES cases. The unit price of leasing spectrum from FES is defined as $\beta_{FES,i}$ per Hz, while the unit price of leasing spectrum from VES to UE $i$ is defined as $\beta_{VES,i}$ per bps. Hence, the communication utility of VEC operator for assigning radio resources to UE $i$ at time instant $t$ is

$$\begin{aligned} F_{comm,i}(t) = a_i(t)H_i - o_i\beta_{FES,i}R_{FES,i}^n(t) \\ - v_i\beta_{VES,i}R_{VES,i}^k(t). \end{aligned} \tag{13}$$

*2) Computation Utility:* Next, we will discuss the utility of VEC operator for providing computation services to UEs. The unit price that the VEC operator charge UEs for computation task $S_i$ at time slot $t$ is $b_i(t)$. Meanwhile, the unit price of the computation resource that VEC operator rents from FES and VES are $\xi_{FES,i}\xi_{VES,i}$, respectively. Let $C_{FES,i}$ and $C_{VES,i}$ denote the computation resource of FES and VES assigned to mobile UE $i$ at time instant $t$, respectively. We have

$$\begin{aligned} F_{comp,i}(t) &= b_i(t)Z_i - o_i\xi_{FES,i}C_{FES,i}^n(t) - v_i\xi_{VES,i}C_{VES,i}^k(t) \\ &= b_iZ_i - o_i\rho_{i,n}\xi_{FES,i}C_{FES}^n(t) \\ &\quad - v_i\varphi_{i,k}\xi_{VES,i}C_{VES}^k(t) \end{aligned} \tag{14}$$

where $\rho_{i,n}$ and $\varphi_{i,k}$ denote the percentage of FES $n$ and VES $k$ computation resource allocated to UE $i$, respectively. Therefore, the total utility of the VEC operator serve UE $i$ at time instant $t$ can be calculated as

$$F_i(t) = F_{comm,i}(t) + F_{comp,i}(t). \tag{15}$$

Finally, we give the utility function of VEC operator as follows

$$\begin{aligned} F(t) &= \sum_{i\in\mathcal{I}} d_iF_i(t) \\ &= \sum_{i\in\mathcal{I}} d_i(F_{comm,i}(t) + F_{comp,i}(t)). \end{aligned} \tag{16}$$

### C. Utility Maximization Problem Formulation

The main objective of this paper is to maximize the VEC network utility by deciding offloading and resource allocation policies. Then, we have the optimization problem as follows

$$\begin{aligned} (\mathbf{P1}) \quad & \max_{\substack{d_i,v_i,o_i, \\ \eta_{i,k},\theta_{i,n},\rho_{i,n},\varphi_{i,k},\forall i,k}} \lim_{t\to\infty} \frac{1}{t}\sum_{t=1}^{T} F(t) \\ \text{s.t.} \quad & C1: \sum_{i\in\mathcal{I}} d_i + v_i + o_i = 1, \\ & C2: \sum_{i\in\mathcal{I}} R_{VES,i}(t) \le R_k^{VES}, \forall t \\ & \qquad \sum_{i\in\mathcal{I}} R_{FES,i}(t) \le R_n^{FES}, \forall t, \\ & C3: \sum_{l=1,l\neq k}^{K} \sum_{i\in\mathcal{I}} p_{i,l}g_{i,l} \le \Lambda^V, \\ & C4: \sum_{m=1,m\neq n}^{N} \sum_{i\in\mathcal{I}} p_{i,m}^F g_{i,m}^F \le \Lambda^F, \\ & C5: T_i^{local}, T_i^{VES}, T_i^{FES} \le T_i^{\max}, \forall i \\ & C6: \sum_{i\in\mathcal{I}} v_i\varphi_{i,k} \le 1, \quad \sum_{i\in\mathcal{I}} o_i\rho_{i,n} \le 1. \end{aligned} \tag{17}$$

In the set of constraints, constraint (C1) guarantees that each UE chooses local computing, VES or FES to execute the computation task. constraint (C2) guarantees that the spectrum used by all UEs for offloading cannot exceed the total available spectrum of VES and FES, respectively. Constraint (C3) ensures that the interference on VES $k$ caused by other VESs below the threshold $\Lambda^V$. Similar constraints for FES $n$ is showed in (C4). Constraints (C5) makes sure the execution time of local, VES and FES computing methods will not exceed the maximum latency of the task. Constraints (C6) guarantee that the computation resource allocated to all UEs less than the total amount of computation resource provided by the VES and FES, respectively.

Problem (P1) can be solved to find results of offloading decision variables $d_i, v_i, o_i, \forall i$ and resource allocation variables $\eta_{i,k}$,$\theta_{i,n}$, $\rho_{i,n}$, $\varphi_{i,k}$. However, for the fact that the decision variables are binary variable, the formulated problem (P1) is not convex. Moreover, we consider the realistic scenarios, where the interaction between UEs and vehicles, the communication channel conditions, the VESs' and FESs' computation abilities are all dynamically changing. In such case, the operator should collect large amount of system states, and make the global decision on offloading operation and resources allocations for every UE in the network according to the systems current state. In this paper, we propose the reinforcement learning methods to solve this problem.

## IV. Reinforcement Learning-Based Solution

In this section, we firstly define the state space, action space and reward function for the proposed problem. Then, we formulate the utility maximization problem as a Markov decision process and $Q$-learning based reinforcement learning method to find the results of the problem. Furthermore, to avoid the curse of dimensionality, we propose a DRL method which use a deep neural network (DNN) to estimate the action-value function of $Q$-learning.

### A. State, Action, and Reward Definition

There are three critical factors should be identified in the reinforcement learning method, namely state, action and reward, which will be described as follow.

*1) State Space:* The state of the available VES $k \in \{1, \ldots, K\}$, the available FES $n \in \{1, \ldots, N\}$ for UE $i$ in time slot $t \in \{0, 1, \ldots, T-1\}$ is determined by the realization of the states $R_{VES,i}(t), C_{VES,i}(t), R_{FES,i}(t), C_{FES,i}(t)$. Hence, the state vector can be presented as

$$\begin{aligned}\mathbf{x}_i(t) = [&K_i(t), R^1_{VES,i}(t), R^2_{VES,i}(t), \ldots, R^{K_i(t)}_{VES,i},\\ &C^1_{VES,i}(t), C^2_{VES,i}(t), \ldots, C^{K_i(t)}_{VES,i},\\ &R^1_{FES,i}(t), R^2_{FES,i}(t), \ldots, R^N_{FES,i}(t),\\ &C^1_{FES,i}(t), C^2_{FES,i}(t), \ldots, C^N_{FES,i}(t)]. \end{aligned} \tag{18}$$

*2) Action Space:* In the proposed VEC network, the VEC operator has to decide the computation task offloading strategy for selecting local computing, VES computing or FES computing methods. Meanwhile, the percentage of the communication and computation resources of VES and FES that are allocated to UE $i$ at time slot $t$ also should be determined. Accordingly, the current action vector $\mathbf{u}_i(t)$ can be described as follows

$$\begin{aligned}\mathbf{u}_i(t) = [&d_i(t), v_i(t), o_i(t),\\ &\eta_{i,1}(t), \eta_{i,2}(t), \ldots, \eta_{i,K}(t),\\ &\theta_{i,1}(t), \theta_{i,2}(t), \ldots, \theta_{i,N}(t),\\ &\rho_{i,1}(t), \rho_{i,2}(t), \ldots, \rho_{i,K}(t),\\ &\varphi_{i,1}(t), \varphi_{i,2}(t), \ldots, \varphi_{i,N}(t)]. \end{aligned} \tag{19}$$

*3) Reward Function:* The reward function is the objective function of problem (P1), which denote the VEC network utility. The gain of the reward is determined by the system action. Hence, the immediate network reward function is the VEC operator's utility from a certain UE $i$ at time slot $t$, i.e., $F_i(t)$ in (15). The agent obtains $F_i(t)$ in state $\mathbf{x}_i(t)$ when action $\mathbf{u}_i(t)$ is performed in time slot $t$. To maximize the long-term utility of VEC operator, the cumulative utility is given as

$$\bar{F}_i = \max E\left[\sum_{t=0}^{T-1} F_i(t)\right]. \tag{20}$$

### B. Q-Learning Based Solution

*1) Markovian Decision Process Formulation:* The proposed vehicle-assisted computing offloading process described is approximated as a Markovian decision process. In a Markovian decision problem, the state at stage $t$ is denoted as $\mathbf{x}(t)$, and the action can be denoted as $\mathbf{u}(t)$, which can be obtained from $U_{\mathbf{x}}(t)$ of admissible actions. When the VEC operator executes action $\mathbf{u}(t) \in U_{\mathbf{x}}(t)$, we can use $\mathbf{y}$ to represent the VEC network's state at the next stage. Then, the state-transition probability is given as $p_{\mathbf{xy}}(\mathbf{u})$. The recursive equation can be written as

$$\Omega_t(\mathbf{x}(t)) = \min_{\mathbf{u}(t)\in U_{\mathbf{x}}(t)} \left\{ F(\mathbf{x}(t),\mathbf{u}(t)) + \sum_{\mathbf{y}\in \mathbf{X}_t} p_{\mathbf{xy}} \Omega_{t+1}(\mathbf{y}(t+1)) \right\}$$
$$\text{s.t. constraints in (C1)} - \text{(C6)}. \tag{21}$$

In this paper, we use $Q$-learning method to solve Markovian decision problems. The $Q$-learning method estimates the optimal $Q$ values of the state and admissible action pairs. More specifically, we define the $Q$ value of state $\mathbf{x}$ and action $\mathbf{u}$ pair as the action cost $Q^{\Omega}(\mathbf{x}, \mathbf{u})$ evaluated by $\Omega$ [32], as follows

$$Q^{\Omega}(\mathbf{x}, \mathbf{u}) = F(\mathbf{x}, \mathbf{u}) + \sum_{\mathbf{y}\in\mathbf{X}} p_{\mathbf{xy}}(\mathbf{u})\Omega(\mathbf{y}). \tag{22}$$

Let $Q^*(\mathbf{x}, \mathbf{u})$ denote the optimal $Q$ value of state $\mathbf{x}$ and action $\mathbf{u} \in U_{\mathbf{x}}$, the optimal evaluation function $\Omega^*$ for $\mathbf{x}$ is

$$\begin{aligned}\Omega^*(\mathbf{x}) &= \min_{\mathbf{u}\in U_{\mathbf{x}}} Q^*(\mathbf{x}, \mathbf{u})\\ &= \min_{\mathbf{u}\in U_{\mathbf{x}}} \left[ F(\mathbf{x}, \mathbf{u}) + \sum_{\mathbf{y}\in\mathbf{X}} p_{\mathbf{xy}}(\mathbf{u})\Omega^*(\mathbf{y}) \right] \end{aligned} \tag{23}$$

Let $Q_t(x, u)$ denote the estimate of $Q^*(\mathbf{x}, \mathbf{u})$ at stage $t$. According to (23), the estimate of $\Omega^*(\mathbf{x})$ at stage $t$ can be

**Algorithm 1:** $Q$-Learning Based Algorithm.

```
Initialize $Q(\mathbf{x}, \mathbf{u})$;
For any stage $t$, VEC operator acquires state vector $\mathbf{x}(t)\ \forall t$;
The RL agent at VEC operator computes the $Q$ value $Q_t(\mathbf{x}, \mathbf{u})$ according to (22);
The agent computes $\Omega_t(\mathbf{x})$ and renew $Q_{t+1}(\mathbf{x}, \mathbf{u})$ according to (24) and (25), respectively ;
The agent obtains a optimal policy $U_{\mathbf{x}^*}(t)^*$ at stage $t$ that is greedy with respect to $\Omega_t$;
Terminate when reach expected state $\mathbf{x}_{terminal}$.
```

written as,

$$\Omega_t(\mathbf{x}) = \min_{\mathbf{u} \in U(\mathbf{x})} Q_t(\mathbf{x}, \mathbf{u}). \tag{24}$$

*2) Q-Learning Based Algorithm:* A $Q$-learning based algorithm is proposed to solve (P1) problem as shown in Algorithm 1. For each stage $t$, the VEC operator updates the $Q$-values of a certain number of $(\mathbf{x}, \mathbf{u})$ pairs and leaves other admissible $(\mathbf{x}, \mathbf{u})$ pairs with unchanged $Q$-values. Let $W_t^Q \subseteq \{(\mathbf{x}, \mathbf{u}) \mid \mathbf{x} \in \mathbf{X}, \mathbf{u} \in U_{\mathbf{x}}\}$ denote the subset of state-action $(\mathbf{x}, \mathbf{u})$ pairs whose $Q$-values are updated at stage $t$. For each state-action pair in $W_t^Q$, the updating of the $Q$-value has two approaches: the $Q$-value is decided by its old value, and the $Q$-value is replaced by a backed-up value. To get compatibility of these two approaches, a learning rate parameter, which is denoted as $\alpha_t, 0 < \alpha_t < 1$, is used to update the $Q$-value of $(\mathbf{x}, \mathbf{u})$ at stage $t$. Then, $Q_{t+1}$ is updated as follows: if $(\mathbf{x}, \mathbf{u}) \in W_t^Q$,

$$Q_{t+1}(\mathbf{x}, \mathbf{u}) = (1 - \alpha_t) Q_t(\mathbf{x}, \mathbf{u}) + \alpha_t [F(\mathbf{x}, \mathbf{u}) + \Omega_t(\mathbf{x})], \tag{25}$$

where $\Omega_t(\mathbf{x})$ can be obtained by (24). The $Q$-values of other state-action $(\mathbf{x}, \mathbf{u})$ pairs remains unchanged, i.e,

$$Q_{t+1}(\mathbf{x}, \mathbf{u}) = Q_t(\mathbf{x}, \mathbf{u}),$$

where $(\mathbf{x}, \mathbf{u}) \nsubseteq W_t^Q$. We can record all possible $Q$-values of the admissible state-action pairs $(\mathbf{x}, \mathbf{u})$ at each stage $t$. Meanwhile, if the learning rate $\alpha_t$ descend over stage $t$, the sequence $Q_t(\mathbf{x}, \mathbf{u})$ achieved by the $Q$-learning method will converge to $Q^*(\mathbf{x}, \mathbf{u})$ with probability one as $t \rightarrow \infty$.

### C. DRL-Based Vehicle Edge Computing

In the above subsection, although the $Q$-learning based solution can solve the problem (P1) with random variables, the effectiveness of the solution will be reduced when the high-dimensional state and action spaces should be dealt with in actual complicated problems. To handle this problem, we propose a DRL-based method which uses DNN to estimate $Q(x, u)$ instead of calculating $Q$ value for each state-action pair in $Q$-learning based method.

*1) DRL Method:* In DRL-based method, the multi-layer deep convolutional networks is used to replace ordinary neural networks to extract high-level features from raw input data. Two neural networks are employed in DRL method. One is the main neural network which is used to represent $Q$ function, denoted as $Q(\mathbf{x}, \mathbf{u}; \theta)$, where $\theta$ stands for the weights of the main neural network. Another is the target neural network, which generates the target $Q$-value $y'_j$ as

$$y'_j = F_j + \varepsilon \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}'; \theta'_j) \tag{26}$$

where $\theta'$ is the parameter of the target network. For the sake of training stabilization, $\theta'$ has fixed value during the calculation of $y'_j$ and is updated after some training steps. Then, we train the deep $Q$-function according to the target $Q$-value by minimizing the loss function, denoted as $Loss(\theta)$, which is given by

$$Loss(\theta) = \frac{1}{U} \sum_j (y'_j - Q(\mathbf{x}, \mathbf{u}; \theta_j))^2. \tag{27}$$

Note that the weight of target neural network $\theta'_j$ is updated using lately weight $\theta_j$.

Moreover, to get rid of the divergence or oscillation due to the strong correlations between consecutive transitions in $Q$-function approximation, the DRL method use experience replay, which stores the experience transition tuple $(\mathbf{x}(t), \mathbf{u}_t, F_t, \mathbf{x}_{t+1})$ at time $t$ into a replay memory $D = \{D_1, \ldots, D_t\}$. Then, the DRL agent randomly samples mini-batch data from the experience memory to train the parameters of DNN. This allows repeated usage of experience data to increase sample efficiency, and can speed up the convergence of learning algorithms.

*2) Model-Assisted DRL Framework:* We note that the experience transition data in the tuple $(\mathbf{x}(t), \mathbf{u}_t, F_t, \mathbf{x}_{t+1})$ have different level of complexities due to different time slot $t$. For example, the data rate between VESs and UEs $R_{VES}$ during daytime are more complex than that at night since less vehicles are appearing in such period. But for the club streets the story is totally different. This implies the complexity of transition data is time varying and has distinct complexity even for the same location. To further improve the efficiency of the DRL method, in this paper, we propose a model-assisted DRL framework in which the DRL agent adaptively selects the training data according to their learning complexities instead of choosing data randomly from experience memory.

The main idea of the model-assisted DRL can be described as follows. At the beginning of the learning phase, the agent selects a transition data from the replay memory and calculate the complexity value of the data. If the complexity value is below the pre-set threshold $\lambda$, the selected data is partially replaced by the calculated data obtained by the model. More specifically, the replaced part of the data is obtained by the communication model as shown in (3) (5). Otherwise, the selected data is used for training. Then, the agent increases the proportion of complex transitions along with the training process. To sort the transitions data in a meaningful order, a complexity function is defined as follows

*Definition 1:* (Complexity Function): Assume a function $CM(\varpi) \rightarrow \mathbb{R}$ denotes the complexity of the sample $\varpi_i$ from a sample set. For any given two samples $\varpi_i$ and $\varpi_j$, if $CM(\varpi_i) < CM(\varpi_j)$, we say that sample $\varpi_j$ is harder to learn than sample $\varpi_i$.

The mathematical formula of complexity function can be obtained from [37]. With the complexity function, the DRL

TABLE I
THE SIMULATION PARAMETERS

| | | |
|---|---|---|
| $B$, $B_0$ | 5, 10 MHz | The spectrum bandwidth between UEs and VES, FES |
| $P_i$ | 100 mwatts | The transmitting power of UE $i$ |
| $\sigma^2$ | -100 dBm | The Background noise |
| $H_i$ | 1 MB | The data size of computation task for UE $i$ |
| $Z_i$ | 2500 Megacycles | The number of CPU cycles of computation task for UE $i$ |
| $C_{local,i}$ | 0.5GHz | The computation capability of UE $i$ |
| $C_{VES}$ | 20GHz | The computation capability of VES |
| $C_{FES}$ | 100GHz | The computation capability of FES |
| $T_i^{max}$ | 10ms | The maximal time of task execution |

**Algorithm 2:** DRL-Based Algorithm for Vehicle Edge Computing Network.

```
OFFLINE:
Preprocess the VEC network with random policy for enough long time;
Obtain and store the state transition profiles and the corresponding estimated $Q(\mathbf{x}, \mathbf{u})$ into the experience memory $D$;
Train the DNN with input state-action pairs $(\mathbf{x}, \mathbf{u})$ and outcomes $Q(\mathbf{x}, \mathbf{u})$;
ONLINE:
For each episode
    Obtain the original observation $s_1$, and pre-process $s_1$ as the state $x_1$ ;
    For each step $t$
        With probability $\varepsilon$ select random action $\mathbf{u}_t$;
        Otherwise $\mathbf{u}_t = \arg\max_{\mathbf{u}} Q(\mathbf{x}_t, \mathbf{u}_t; \theta)$;
        Execute action $\mathbf{u}_t$ in the VEC network, achieve the reward $F_t$ and observation $s_{t+1}$;
        Process $s_{t+1}$ to the next state $\mathbf{x}_{t+1}$;
        Store the experience $(\mathbf{x}_t, \mathbf{u}_t, F_t, \mathbf{x}_{t+1})$ into the experience memory $D$;
        Calculate the complexity function CM and obtain a batch of $U$ samples $(\mathbf{x}_j, \mathbf{u}_j, F_j, \mathbf{x}_{j+1})$ from the experience memory;
        Calculate the target $Q$-value $y'_j$ from the target DNN as
            $y'_j = F_j + \varepsilon \max_{\mathbf{u}'} Q(\mathbf{x}', \mathbf{u}'; \theta'_j)$
        Update the weight of the main DNN $\theta$ by minimizing
            $Loss(\theta) = \frac{1}{U} \sum_j (y'_j - Q(\mathbf{x}, \mathbf{u}; \theta_j))^2$
        Perform a gradient descent step on $Loss(\theta)$ with respect to the parameter $\theta$ and update $\theta'_j$ by using lately weight $\theta_j$;
    End For
End For
```

agent generates or selects appropriate transitions data based on the complexity level. After that, the DNN parameters are updated using the stochastic gradient descent method. The description of the proposed DRL-based algorithm for vehicle edge computing are given in the next subsection.

*3) DRL-Based Algorithm for Vehicle Edge Computing:* As shown in Algorithm 2, the DRL-based algorithm consists of two phases: offline training phase and online deep $Q$-learning phase.

In the offline training phase, the DNN is constructed to derive the correlation between each state-action pair $(\mathbf{x}, \mathbf{u})$ of the system under control and its value function $Q(\mathbf{x}, \mathbf{u})$. Specifically, we firstly preprocess the computing modes (local, VES or FES) selection and resource allocation of VEC network with random policy for enough long time. Then, we obtain and store the state transition profiles and the corresponding estimated $Q(\mathbf{x}, \mathbf{u})$ into an experience memory $D$ with capacity $C_D$. At last, the DNN is pre-trained with input pairs $(\mathbf{x}, \mathbf{u})$ and outcomes $Q(\mathbf{x}, \mathbf{u})$.

In the online deep $Q$-learning phase, the action selection and $Q$ value update is achieved by deep $Q$-learning method. Specifically, in each episode, the DRL agent firstly initiate the observation of the vehicular network state $\mathbf{x}_1$. Then, the $\varepsilon$-greedy is utilized to select the execution action $\mathbf{u}_k$. With $\varepsilon$ probability, an action from the action set is randomly selected. Otherwise, the action is chosen with the highest estimated $Q$ value which is derived from the main DNN with the input of state-action pair $(\mathbf{x}_t, \mathbf{u}_t)$. As obtaining the reward value $F_t$ and next state $\mathbf{x}_{t+1}$ from the vehicle edge computing network, the state transition $(\mathbf{x}_t, \mathbf{u}_t, F_t, \mathbf{x}_{t+1})$ will be stored into the experience memory $D$. To train the main DNN, $U$ samples of the transition data are selected following the proposed model-assisted DRL method. After calculating the target $Q$-value $y'_j$, the DRL agent updates parameters $\theta$ of the main DNN by using stochastic gradient descent method on the loss function $Loss(\theta)$.

## V. NUMERICAL RESULTS

To illustrate the performance of the proposed methods, we consider a 120 × 120 km$^2$ area which includes 5-50 randomly deployed small cells. At each time slot, there are 4-10 UEs with computation task and 10 vehicles running at constant speed in each small cell. Other evaluation parameters are listed in Table I [28], [29].

The performance of the proposed $Q$-learning based method and DRL-based method is compared with other two methods: VES and FES methods. In VES method, the UEs decide to execute the computation tasks locally or only by VES computing. In FES method, UEs execute the computation tasks locally or offload their tasks to the FES server and the entire computational resource is equally allocated to each UE. In both VES

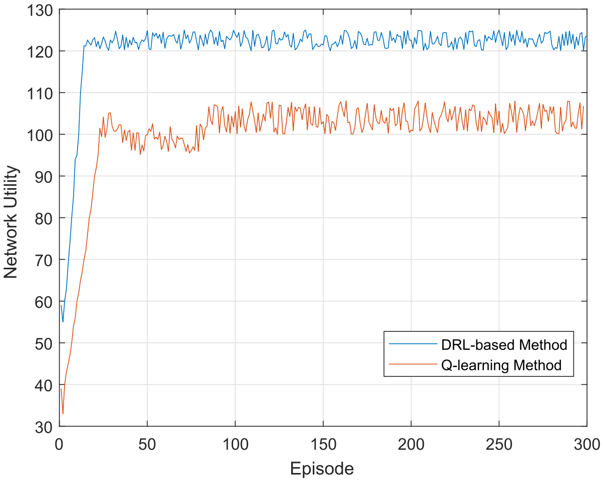


Fig. 2. Convergence of the proposed DRL based method and Q-learning based method.

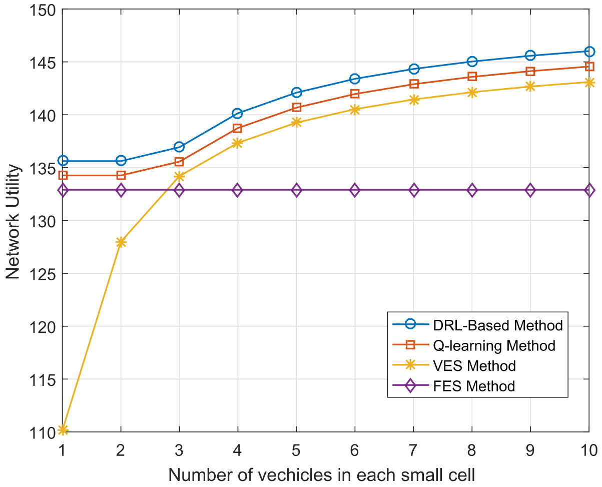


Fig. 4. Network utility comparison in terms of the number of vehicles in each small cell.

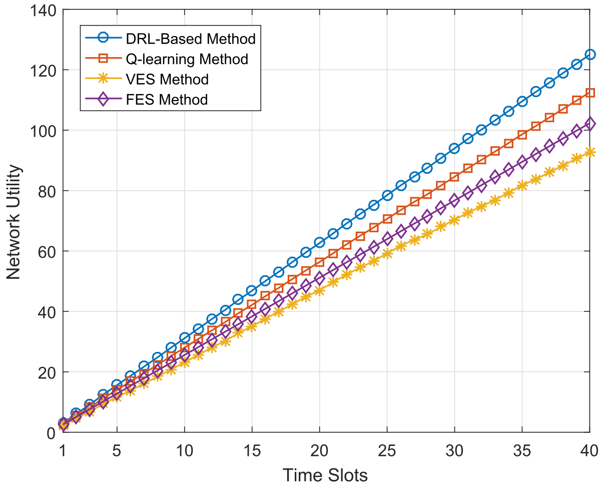


Fig. 3. Network utility of the VEC network.

and FES methods, the optimization problem is formulated to obtain the optimal offloading decisions and resource allocation policies [28]. Here, we set the capacity of experience memory $C_D = 500$, and the capacity of mini batch $U = 32$. The reward discount $\mu = 0.9$ and $\varepsilon = 0.5$.

Fig. 2 shows the convergence performance of the proposed DRL based method and $Q$-learning based method. It is observed that the network utility increases as the number of the episodes increase until it attains a relatively stable value in both of the proposed DRL-based method and the $Q$-learning based method. It is shown that the proposed methods are convergent. We can also observe that the network utility of the Q-learning method is less than that of the DRL-based method, which will be explained in the next subsection.

### A. *Utility Comparison*

Fig. 3 shows the comparison of the network utility of the proposed DRL-based method, $Q$-learning method, the VES and FES methods. The DRL-based method achieves the highest utility among all the methods. This is because the DRL-based method uses the optimal offloading policies and resource allocation for vehicle edge network at each time slot. In contrast, the VES and FES methods only use VES or FES as the offloading server and employ the greedy strategies which may not be able to reach the optimal utility under different system states. Meanwhile, we can also observer that the performance of the $Q$-learning method is less than that of the DRL-based method which can handle the high dimensional computation well.

In Fig. 4, we also compare the network utility in terms of the number of vehicles in each small cell. It is interesting to see that the utility obtained by VES method increases at first, then, when the number of vehicles reaches about 5, the acceleration of the increasing utility goes down. The reason is that when more and more VES is used for the offloading, the VEC operator could gain more from allocating spectrum and computation resources to more UEs. However, when there are too many VESs in the system, all those transmissions between the VESs and UEs will cause severe interference to each other during the computation tasks offloading process, so the operator will automatically decline some of the offloading requests generated by the UEs.

Still, we can observe that the proposed DRL-based method has the highest utility among other solutions. That is, the DRL-based method always makes better choice of computation strategy (local, VES or FES) according to dynamic network environment. The $Q$-learning method, thanks to the proposed computation selection and resource allocation strategy, has better utility than that of the VES and FES methods but less than that of the DRL-based method. For the FES method, the achieved utility is keeping a fixed value as the increasing number of vehicles. Since the FES method only uses FES and local computation to execute the task, the increasing number of vehicles will not affect its performance.

Fig. 5 shows the effects of the computation offloading charging price ($b_i$). The network utility of all methods increase with the increase of offloading charging price due to the fact that

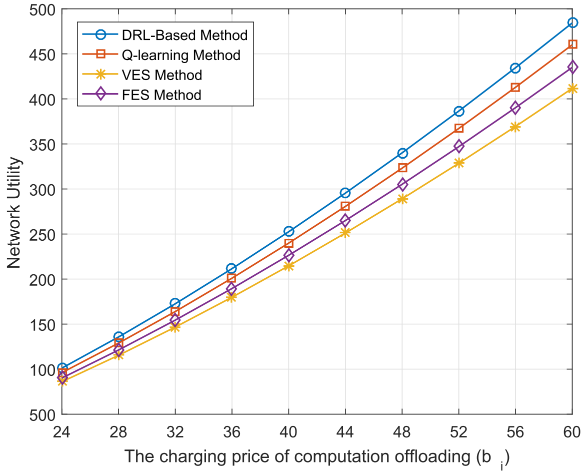


Fig. 5. Network utility comparison in terms of different computation offloading charging price.

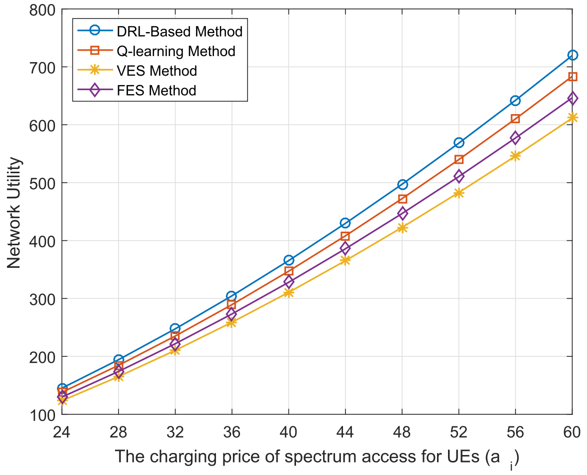


Fig. 6. Network utility comparison in terms of different spectrum access charging price.

a higher price will increase the charging fee for computation offloading, which induces a higher gain of VEC server, and thus the network utility increases. Similar to the above analysis, the DRL-based method and $Q$-learning method with proposed offloading and resource allocation strategy achieve higher utility than that in the VES and FES methods. The utility gap between the FES and VES is not wide, this reveals that the offloading charging price has the nearly same affect to the VES and FES methods.

In Fig. 6, we shows effects of the spectrum access charging price ($a_i$) for each UE. It is obvious that the network utility of all methods increase with the increase of spectrum access price which increases the gain of VEC server, and thus increases network utility. We can observe that the network utility in VES case are the lowest which indicates that the spectrum access price has much effects to the VES method. The reason is that the communications between the moving VES and UEs are easily affected by dynamic changing of radio environment, which results in more communication requests are declined and cause the utility reducing.

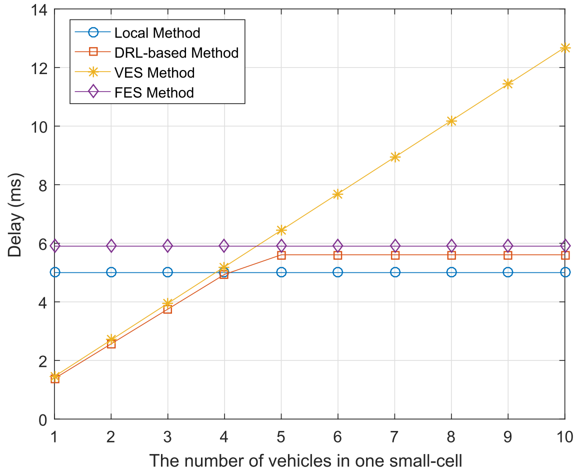


Fig. 7. Delay comparison in terms of the number of vehicles in one small cell.

## B. Delay Comparison

In this subsection, we define the delay as the execution time of a computation task which includes the computation time and offloading time for both VES and FES methods. For local computing method, the delay is the local execution time of the task. Since the DRL-based method and the $Q$-learning based method use the same offloading and resource allocation strategy, we just evaluate the delay of the DRL-based method in this case.

Fig. 7 shows the comparison of delay among the proposed DRL-based method, local method, VES and FES method. For FES method, the delay has constant value since the increasing number of vehicles will not affect the delay performance of FESs. In the local method, the UEs execute the computation task by themselves. Hence, the delay of this method is lower than that of FES method.

We also observe that the delay of the VES method increases with the increasing number of vehicles in each cell. This is because the increment of vehicles implies more VESs will join the offloading process, which may cause more severe interference among VESs and cause higher service delay. In addition, the VES method has lower delay than that of the FES and local methods when the number of the vehicles is small, which indicates that the VES method can achieve lower delay as the number of the vehicles not exceed 5 in each small cell. Note that the proposed DRL-based method has lower delay than that of both VES and FES methods. That is, the DRL-based method can intelligently find the most appropriate offloading policies under different scenarios.

Fig. 8 shows the delay comparison in terms of the computation resources (the number of CPU cycles) required for a computation task. It can be seen that delay of the local method increases as the required computation resources increases, which indicates that the local method is suitable for the computation tasks only require little computation resources. Compared to the local method, the delay generated by other three methods, i.e.,

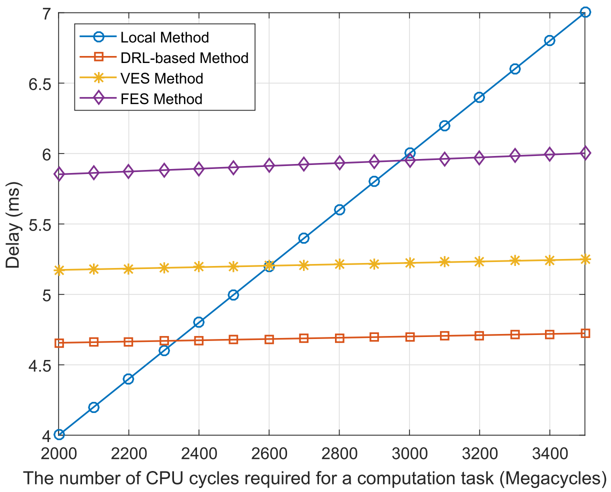


Fig. 8. Delay comparison in terms of the computation resources required for a computation task.

DRL-based method, VES method and FES method, increases slowly as the increasing of the required computation resources for the computation task. That is, these three methods can employ VES or FES to offload the computation and drastically save the computation time. Still, with the offloading strategy selection and resource allocation, the DRL-based method achieves lower delay than that of the VES and FES methods.

## VI. Conclusion

In this paper, we proposed a vehicle edge computing network architecture in which the vehicles (VES) can be the edge computing server to provide computation services as well as FES for the UEs. As the VES and FES coexist in the same network, according to practical environment, the offloading policies and the resource allocation scheme for the VES or FES were proposed. Meanwhile, to find the optimal offloading and resource allocation policies, the semi-Markov process is used to model the relationship between the policies and the network environments. With the help of RL-based algorithm, we obtained the results which showed that the proposed scheme and the DRL-based solution can achieve better performance than that of the pure VES or FES methods.

## References

[1] S. Andreev *et al.*, "Understanding the IoT connectivity landscape: A contemporary M2M radio technology roadmap," *IEEE Commun. Mag.*, vol. 53, no. 9, pp. 32–40, Sep. 2015.

[2] Z. Li, Z. Yang, and S. Xie, "Computing resource trading for edge-cloud-assisted Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 15, no. 6, pp. 3661–3669, Jun. 2019.

[3] R. Buyya and A. V. Dastjerdi, *Internet of Things: Principles and Paradigms*. Amsterdam, The Netherlands: Elsevier, 2016.

[4] Y. Liu *et al.*, "Design of a scalable hybrid MAC protocol for heterogeneous M2M networks," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 99–111, Feb. 2014.

[5] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Inform.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.

[6] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for Internet of Vehicles," *IEEE Wireless Commun. Mag.*, vol. 26, no. 3, pp. 12–18, Jun. 2019.

[7] Z. Li, Z. Yang, S. Xie, W. Chen, and K. Liu, "Credit-based payments for fast computing resource trading in edge-assisted Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6606–6617, Aug. 2019.

[8] ETSI, "Mobile-edge computing: Introductory technical white paper," ETSI White Paper, Sep. 2014.

[9] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Mobile edge computing and networking for green and low-latency Internet of Things," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 39–45, May 2018.

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

[11] K. Zhang, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Cooperative content caching in 5G networks with mobile edge computing," *IEEE Wireless Commun. Mag.*, vol. 25, no. 3, pp. 80–87, Jun. 2018.

[12] H. Guo, J. Liu, and J. Zhang, "Computation offloading for multi-access mobile edge computing in ultra-dense networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 14–19, Aug. 2018.

[13] N. Abbas, H. Hajj, S. Sharafeddine, and Z. Dawy, "Traffic offloading with channel allocation in cache-enabled ultra-dense wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8723–8737, Sep. 2018.

[14] H. Guo, J. Zhang, J. Liu, and H. Zhang, "Energy-aware computation offloading and transmit power allocation in ultra-dense IoT networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4317–4329, Jun. 2019.

[15] Z. Cui *et al.*, "Joint optimization of energy consumption and latency in mobile edge computing for Internet of Things," *IEEE Internet Things*, vol. 6, no. 3, pp. 4791–4803, Jun. 2019.

[16] H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," *IEEE Netw.*, vol. 32, no. 1, pp. 96–101, Jan./Feb. 2018.

[17] H. Guo, J. Liu, J. Zhang, W. Sun, and N. Kato, "Mobile-edge computation offloading for ultra-dense IoT networks," *IEEE Internet Things*, vol. 5, no. 6, pp. 4977–4988, Dec. 2018.

[18] Y. Liu, C. Yang, L. Jiang, S. Xie, and Y. Zhang, "Intelligent edge computing for IoT-based energy management in smart cities," *IEEE Netw.*, vol. 33, no. 2, pp. 111–117, Mar. 2019.

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

[20] X. Lyn *et al.*, "Optimal schedule of mobile edge computing for Internet of Things using partial information," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2606–2615, Mar. 2017.

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

[22] C. Wang, Y. Li, D. Jin, and S. Chen, "On the serviceability of mobile vehicular cloudlets in a large-scale urban environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2960–2970, Oct. 2016.

[23] J. Liu *et al.*, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Commun. Mag.*, vol. 35, no. 11, pp. 94–100, Jul. 2017.

[24] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7635–7647, Oct. 2019.

[25] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.

[26] H. Guo and J. Liu, "Collaborative computation offloading for multi-access edge computing over fiber-wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[27] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Reinforcement learning for resource provisioning in the vehicular cloud," *IEEE Wireless Commun.*, vol. 24, no. 4, pp. 128–135, Aug. 2016.

[28] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[29] 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.

[30] K. Zhang *et al.*, "Edge intelligence and blockchain empowered 5G beyond for industrial Internet of Things," *IEEE Netw. Mag.*, to be published.

[31] W. Zhong, K. Xie, Y. Liu, C. Yang, and S. Xie, "Topology-aware vehicle to grid energy trading for active distribution systems," *IEEE Trans. Smart Grid*, vol. 10, no. 2, pp. 2137–2147, Mar. 2019.

[32] A. C. Barto *et al.*, "Learning to act using real-time dynamic programming," Dept. Comput. Inf. Sci., Univ. Massachusetts, Amherst, MA, USA, Tech. Rep. 93-02, 1993.

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

[34] H. Li, T. Wei, A. Ren, Q. Zhu, and Y. Wang, "Deep reinforcement learning: Framework, applications, and embedded implementations: Invited paper," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2017, pp. 847–854.

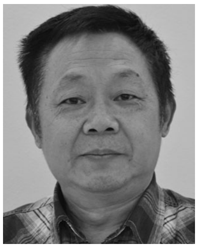
[35] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5G beyond," *IEEE Netw. Mag.*, vol. 33, no. 3, pp. 10–17, May/Jun. 2019.

[36] Y. Xu, J. Yao, H. Jacobsen, and H. Guan, "Cost-efficient negotiation over multiple resources with reinforcement learning," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service*, 2017, pp. 1–6.

[37] L. Jiang *et al.*, "Self-paced curriculum learning," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2694–2700.

**Yi Liu** received the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2011. After that, he joined the Singapore University of Technology and Design (SUTD) as a Postdoctoral Researcher. In 2014, he worked in the Institute of Intelligent Information Processing, Guangdong University of Technology (GDUT), where he is now a Full Professor. His research interests include wireless communication networks, cooperative communications, smart grids and intelligent edge computing.

**Huimin Yu** received the B.E. degree in electronic engineering from Hunan Normal University, Changsha, China, in 2003, and the Ph.D. degree in electromagnetic fields and microwave technology from the Institute of Electronics, Chinese Academy of Sciences, Beijing, China, in 2009. He is currently an Associate Professor with the Department of Communication Engineering, Hunan Normal University. His research interests include wireless communication and ultra wideband system design.



**Shengli Xie** (M'01–SM'02–F'19) received the M.S. degree in mathematics from Central China Normal University, Wuhan, China, in 1992, and the Ph.D. degree in automatic control from the South China University of Technology, Guangzhou, China, in 1997. He was a Vice Dean with the School of Electronics and Information Engineering, South China University of Technology, China, from 2006 to 2010. He is currently the Director of both the Institute of Intelligent Information Processing (LI2P) and Guangdong key Laboratory of Information Technology for the Internet of Things, and also a Professor with the School of Automation, Guangdong University of Technology, Guangzhou, China. He has authored or coauthored four monographs and more than 100 scientific papers published in journals and conference proceedings, and was granted more than 30 patents. His research interests broadly include statistical signal processing and wireless communications, with an emphasis on blind signal processing and Internet of Things.

**Yan Zhang** (SM'10) is a Full Professor at the University of Oslo. He is an Editor of several IEEE publications, including IEEE COMMUNICATIONS MAGAZINE, IEEE NETWORK, IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING, IEEE COMMUNICATIONS SURVEYS & TUTORIALS, and IEEE INTERNET OF THINGS. His current research interests include next-generation wireless networks leading to 5G and cyber physical systems. He is an IEEE VTS Distinguished Lecturer and a Fellow of IET. He received the award "Highly Cited Researcher' (Web of Science top 1% most cited worldwide) according to Clarivate Analytics.