Deep Reinforcement Learning Based Resource Management for Multi-Access Edge Computing in Vehicular Networks

Haixia Peng, *Student Member, IEEE* and Xuemin Shen, *Fellow, IEEE*

***Abstract*—In this paper, we study joint allocation of the spectrum, computing, and storing resources in a multi-access edge computing (MEC)-based vehicular network. To support different vehicular applications, we consider two typical MEC architectures and formulate multi-dimensional resource optimization problems accordingly, which are usually with high computation complexity and overlong problem-solving time. Thus, we exploit reinforcement learning (RL) to transform the two formulated problems and solve them by leveraging the deep deterministic policy gradient (DDPG) and hierarchical learning architectures. Via off-line training, the network dynamics can be automatically learned and appropriate resource allocation decisions can be rapidly obtained to satisfy the quality-of-service (QoS) requirements of vehicular applications. From simulation results, the proposed resource management schemes can achieve high delay/QoS satisfaction ratios.**

***Index Terms*—Vehicular networks, multi-access edge computing, multi-dimensional resource management, deep reinforcement learning, DDPG**

I. Introduction

With the advances in wireless communications and smart vehicle technologies, vehicular networks have attracted both industry and academia's great attentions in recent years [1]–[3]. For both manually driven and autonomous vehicles, the vehicular network has been widely regarded as a great potential to improve driving safety, traffic mobility, sustainability, and efficiency [1], [4]–[6]. However, with the emerging of various vehicular applications, especially those delay-sensitive and/or computation-intensive ones, salient challenges are confronted by the vehicular network due to limited spectrum resources and limited on-board computing/storing resources [7]. To address these challenges, multi-access edge computing (MEC) has been implemented in vehicular networks recently [8]. By enabling computing and storing capabilities at edge nodes close to the moving vehicles, a vehicle can offload tasks with high computation and sensitive delay requirements to the MEC servers [9], [10]. Thus, issues caused by the heavy demand on spectrum/computing/storing resources and the shortage of vehicles' on-board resources can be solved while short response delay can be guaranteed for various applications [11], [12].

In MEC-based vehicular networks, vehicles are allowed to access the MEC or cloud-computing servers via multiple wireless communication technologies, including dedicated short range communications (DSRC), cellular, Wi-Fi, and White-Fi [12]. For vehicular applications with high demand on computing/storing resources, a vehicle can offload tasks to the MEC server with short response delay by avoiding transmitting data from the MEC server to the cloud-computing server [13].

Despite the advantages of MEC servers, their implementation faces some technical challenges to support vehicular applications. From the network service provider's perspective, we should support the dynamic demand on resources from vehicle user's by the pre-deployed MEC servers. The technical challenges include where and how many MEC servers should be deployed and how the multi-dimensional resources should be allocated among different vehicles [12]. And from the vehicle user perspective, it is critical to make decisions on whether, when, and how many tasks should be offloaded to MEC servers to satisfy the quality-of-service (QoS) requirement.

To address the challenges on implementing MEC-based vehicular networks, many research works have been performed recently, including design of architecture, task offloading scheme, resource management scheme, and so on. For example, the MEC-based hierarchical vehicular network framework, comprised of vehicle level's on-board computing/storing resources and server level's resources (resources placed at the MEC and cloud-computing servers), has been investigated in [10], [12], [14]–[16]. To better manage the spectrum/computing/storing resources among and make task offloading decisions to vehicle users, task offloading and resource management schemes have been proposed in [10], [15]–[17]. Since task offloading and spectrum/computing resource allocation are coupled with each other, the objectives of the most existing works have been achieved by jointly optimizing these two parts with traditional optimization methods [10], [15]. However, only one or two dimensions of resources have been considered in most of the existing schemes, which cannot be directly adopted to support some vehicular applications where high dimensional resources are involved, such as the computing tasks generated by the leading vehicle for platoon/convoy control [7]. Moreover, there are also some works focusing on multi-dimensional resources

Manuscript received October 31, 2019; revised February 12, 2020; accepted February 27, 2020. Date of publication March 6, 2020; date of current version December 30, 2020. This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. Recommended for acceptance by Dr. Chunxiao Jiang. *(Corresponding author: Haixia Peng.)*

The authors are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L3G1, Canada (e-mail: h27peng@uwaterloo.ca; sshen@uwaterloo.ca).

Digital Object Identifier 10.1109/TNSE.2020.2978856

management in the scenarios with low mobility users [18], [19]. For MEC-based vehicular networks, the computational complexity of multi-dimensional resource management problem increases due to the high vehicle mobility and time-varying demand on resources, therefore increasing the time consumption on the resource management scheme itself. Thus, it is infeasible to adopt the pure optimization approach-based schemes to achieve multi-dimensional resource management in MEC-based vehicular networks, especially for the scenarios with delay-sensitive applications. How to design practical and QoS-oriented multi-dimensional resource management schemes for the MEC-based vehicular networks still needs efforts.

As is known, artificial intelligence (AI) technology, especially reinforcement learning (RL), can be exploited to solve resource management problems quickly [20]–[23]. Q-learning [16], [24], deep Q-learning [25]–[27], actor-critic [18], [28], and other deep RL algorithms have been widely exploited for resource management in wireless communication networks. Inspired by the existing works and considering the dynamic vehicular network environment caused by high vehicle mobility and heterogeneous applications, we investigate how to exploit deep RL to jointly manage the spectrum, computing, and storing resources to support delay-sensitive applications in the MEC-based vehicular network [12] in this paper. Specifically, the main contributions of this work can be summarized as follows,

1) According to the location of the MEC server, two typical multi-dimensional resource management frameworks are proposed with placing the MEC server at a macro-cell base station (MBS) and an edge node (EN), respectively.
2) Leveraging optimization theory, optimization problems are formulated to maximize the number of offloaded tasks with satisfied QoS requirements and constrained total amounts of available spectrum, computing, and storing resources.
3) To rapidly solve the formulated problems and obtain optimal spectrum slicing among base stations (BSs) and optimal spectrum/computing/storing allocation among vehicles, the formulated optimization problems are transformed with deep RL.
4) A deep deterministic policy gradient (DDPG)-based algorithm is proposed to solve the transformed RL problems. As the complexity of the transformed RL problems increases with the sizes of environment state and action, a hierarchical DDPG (HDDPG)-based algorithm is developed by combining the DDPG and the hierarchical learning architecture.

The rest of this paper is organized as follows. First, the system model is presented in Section II, including the MEC-based vehicular network architecture, spectrum management frameworks, communication model, and computing/storing resource allocation models. In Section III, we formulate two optimization problems under two typical MEC architectures to jointly manage the multi-dimensional resources and transform them with deep RL. Then, DDPG- and HDDPG-based algorithms are proposed in Section IV to solve the transformed RL problems. Section V presents extensive simulation results to illustrate the performance of the proposed algorithms. Finally, we provide our concluding remarks in Section VI.

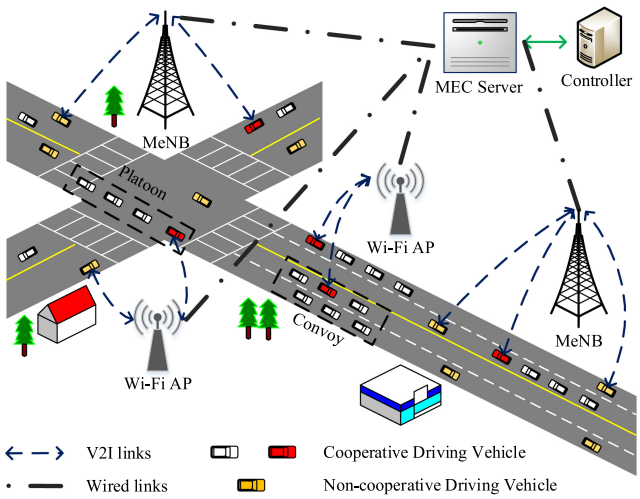


Fig. 1. An illustration of MEC-based vehicular network model.

II. System Model

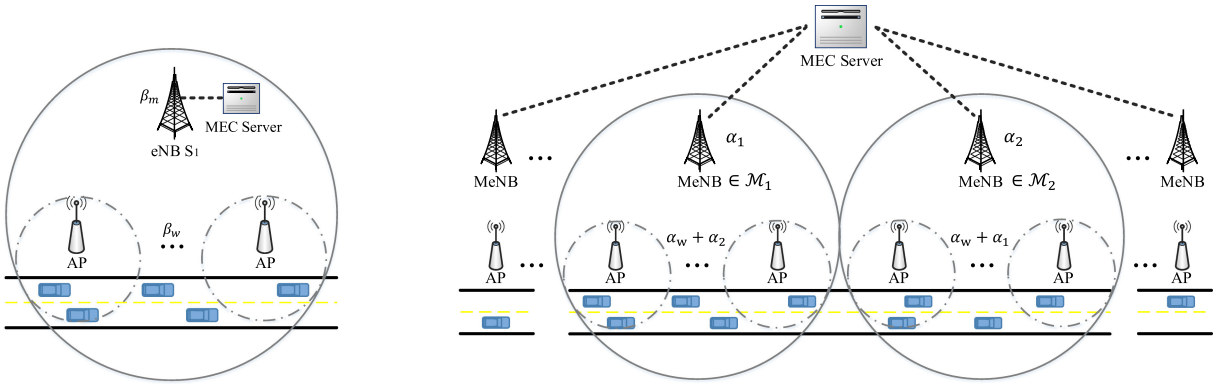
In this section, an MEC-based vehicular network architecture is presented, followed with models for spectrum management, vehicular communications, and computing/storing resource allocation under the considered MEC-based vehicular network.

A. MEC-Based Vehicular Network Architecture

Based on the network model in our previous work [12], we consider an MEC-based vehicular network architecture with one MEC server to support different vehicular applications. Through enabling computing and storing capabilities at the edge of the core network, the MEC server can help reduce the task overload on vehicles and provide a low response delay. To be adaptable to different application environments, we assume the MEC server is either placed at an MBS, such as a macro eNodeB (MeNB) or at an EN of the core network. Vehicles within the service area are allowed to access the computing/storing resources placed at the MEC server through different wireless access technologies. The service area of an MEC server is defined as the total coverage areas of the BSs connected to it.

Fig. 1 illustrates an MEC-based vehicular network model with the MEC server placed at an EN. Vehicles, including traditional manually driven vehicles and autonomous vehicles[1], can access the computing/storing resources placed at the MEC server through either cellular or Wi-Fi/DSRC technologies, i.e., a vehicle can offload computing tasks to the MEC server through an MeNB or a Wi-Fi AP. To achieve high utilization of spectrum, computing, and storing resources, a software defined networking (SDN) and network function virtualization (NFV) enabled controller is installed at each MEC server to

[1] Autonomous vehicles are either driving cooperatively in platoon or convoy forms or driving non-cooperatively same to the manually driven vehicles [29]



(a) For the MeNB-mounted MEC server

(b) For the EN-mounted MEC server

Fig. 2. Dynamic spectrum management frameworks.

centrally and dynamically manage the three dimensions of resources over its service area [12], [30]. By the controller, available spectrum resources from different wireless access technologies can be aggregated, and then dynamically sliced among the BSs and allocated among vehicles on demanding. Moreover, through collecting information about vehicle distribution and application requirements from vehicles in the service area, virtual computing, and storing resources allocated to each offloaded task can be adjusted by the controller.

B. Spectrum Management Framework

The topology of MEC-based vehicular network and distribution of task offloading requests in the service area change frequently due to high vehicle mobility and heterogeneous applications. To offload vehicles' computing tasks to the MEC server with acceptable communication delay to satisfy the QoS requirements of each offloaded task, dynamic spectrum management frameworks are developed for two considered scenarios, placing the MEC server at an MeNB and at an EN, as shown in Fig. 2. Considering vehicles on a two-lane straight country road with one lane for each direction, MeNBs and Wi-Fi APs are uniformly deployed in one side of the road with multiple Wi-Fi APs under the MeNB's coverage. In the following, we provide the detailed spectrum management procedures executed by the controllers installed at the MeNB- and EN-mounted MEC servers, including spectrum slicing among MeNBs and Wi-Fi APs and spectrum allocation among vehicles associated to the same BS.

MeNB-mounted MEC server:' Denote $\mathcal{A}_m/A_m$ as the set/number of Wi-Fi APs under the service area of an MeNB-mounted MEC server, i.e., $\mathcal{A}_m = \{W_i, i = 1, \ldots, A_m\}$. Then the spectrum management procedures executed by the controller can be summarized into three steps: (1) aggregating available spectrum resources from the MeNB and the A_m Wi-Fi APs into $R_{\max}$; (2) dynamically slicing the aggregated spectrum resources, $R_{\max}$, to the MeNB and the Wi-Fi APs with ratios β_m and β_w, respectively, where $\beta_m + \beta_w = 1$; and (3) allocating a proper fraction of spectrum to each vehicle's uplink transmission. To improve the spectrum efficiency with acceptable interference, spectrum reusing is enabled [31]. Vehicles associated to different Wi-Fi APs share spectrum $R_{\max}\beta_w$ to transmit computing tasks to the MeNB-mounted MEC server. Note that spectrum reusing among MeNBs is not considered in this scenario as it is beyond the scope of the control capability of the controller installed at the MeNB-mounted MEC server.

EN-mounted MEC server: To overcome the challenges caused by the limited serving time to each vehicle due to high vehicle mobility, the MEC server can be placed at an EN and with multiple MeNBs wired connected to it to enlarge its service area. According to the spatial adjacency relation among different MeNBs, the MeNBs under the service area of the EN-mounted MEC server are divided into two groups with sets/numbers $\mathcal{M}_1/M_1$ and $\mathcal{M}_2/M_2$, such that two MeNBs from the same group are not neighbored to each other. Spectrum slicing and spectrum reusing then are considered among the Wi-Fi APs and the two groups of MeNBs. By the controller installed at the EN-mounted MEC server, the aggregated spectrum resources, $R_{\max}$, are sliced with ratio set $\{\alpha_1, \alpha_2, \alpha_w\}$, and then reallocated to MeNBs in $\mathcal{M}_1$, MeNBs in $\mathcal{M}_2$, and Wi-Fi APs. In Fig. 2(b), we take two adjacent MeNBs from $\mathcal{M}_1$ and $\mathcal{M}_2$ as the target MeNBs to illustrate the spectrum slicing results. Let $\mathcal{A}_j/A_j$ be the set/number of Wi-Fi APs within the coverage of MeNB S_j. Then for uplink transmission under the service area, considered spectrum reusing includes, (1) vehicles associated to MeNBs in $\mathcal{M}_1$ (or in $\mathcal{M}_2$) reuse the spectrum resource $R_{\max}\alpha_1$ (or $R_{\max}\alpha_2$); (2) vehicles associated to Wi-Fi APs reuse the spectrum resource $R_{\max}\alpha_w$; and (3) vehicles associated to a Wi-Fi AP reuse the spectrum resources sliced to the MeNBs that with no overlapping with this Wi-Fi AP, such as vehicles associated to the Wi-Fi APs in $\mathcal{A}_j$ (where $S_j \in \mathcal{M}_1$) reuse the spectrum resource $R_{\max}\alpha_2$ that is sliced to MeNBs in $\mathcal{M}_2$.

C. Communication Model

Assume the transmit powers of the MeNB and the Wi-Fi AP are fixed, full signal coverage is provided on the considered road segment by the MeNB, and there is no overlapping between any two Wi-Fi APs. Based on the duality theorem for the transmission in the same coherence interval [18], we assume the channels between a vehicle and a BS for uplink and downlink transmissions are symmetry. Let P be the transmit power of a vehicle. In the following, we analyze the spectrum efficiency and transmission rate achieved at the BS from the associating vehicles under the two dynamic spectrum management frameworks presented in Section II-B.

Under the service area of the MeNB-mounted MEC server: As spectrum reusing is not considered among MeNBs in this scenario, the spectrum efficiency achieved at the MeNB from associating vehicle k can be given by

$$h_k^m = \log_2\left(1 + \frac{PG_k^m}{\sigma^2}\right), \tag{1}$$

where G_k^m (or $G_k'^i$) is the channel gain between vehicle k and the MeNB (or Wi-Fi AP W_i), which varies with the distance between vehicle k and the MeNB or Wi-Fi AP, and the calculating formulas are given in Section V in detail. σ^2 is the power spectrum density of the additive white Gaussian noise (AWGN). Considering interference generated by the transmission under Wi-Fi AP $W_e \in \mathcal{A}_m$ ($e \neq i$), spectrum efficiency achieved at Wi-Fi AP W_i from vehicle k can be given by

$$h_k'^i = \log_2\left(1 + \frac{PG_k'^i}{\sum_{W_e \in \mathcal{A}_m, e \neq i} PG_k'^e + \sigma^2}\right). \tag{2}$$

Let f_k^m (or $f_k'^i$) be the spectrum fraction allocated to vehicle k by the MeNB (or Wi-Fi AP W_i), namely, spectrum resource $R_{\max}\beta_m f_k^m$ (or $R_{\max}\beta_w f_k'^i$) are available for vehicle k's uplink transmission. Then the data rate of the uplink transmission from vehicle k to the MeNB (or Wi-Fi AP W_i) can be expressed as

$$R_k^m = R_{\max}\beta_m f_k^m h_k^m \ \ (\text{or } R_k'^i = R_{\max}\beta_w f_k'^i h_k'^i). \tag{3}$$

Under the service area of the EN-mounted MEC server: As shown in Fig. 2(b), vehicles under the service area of the EN-mounted MEC server can access the MEC server through either an MeNB in $\mathcal{M}_1$ or $\mathcal{M}_2$, or a Wi-Fi AP. According to the considered spectrum reusing, we can obtain the spectrum efficiency achieved at the MeNB and Wi-Fi AP in this scenario accordingly. Taking the uplink transmission from vehicle k to MeNB $S_e \in \mathcal{M}_1$ as an example, experienced interference includes that from uplink transmission to MeNB $S_j \in \mathcal{M}_1$ ($j \neq e$) and from uplink transmission to Wi-Fi AP $W_i \in \mathcal{A}_j$ ($S_j \in \mathcal{M}_2$). And the spectrum efficiency achieved at MeNB S_e from vehicle k then can be given by

$$h_k^e = \log_2\left(1 + \frac{PG_k^e}{\sum_{S_j \in \mathcal{M}_1, j \neq e} PG_k^j + \sum_{S_j \in \mathcal{M}_2} \sum_{W_i \in \mathcal{A}_j} PG_k'^i + \sigma^2}\right). \tag{4}$$

Let f_k^e be the spectrum fraction allocated to vehicle k by MeNB $S_e \in \mathcal{M}_1$. As the spectrum resource sliced to MeNBs in $\mathcal{M}_1$ is $R_{\max}\alpha_1$, the transmission rate from vehicle k to MeNB S_e is

$$R_k^e = R_{\max}\alpha_1 f_k^e h_k^e. \tag{5}$$

Similarly, for the uplink transmission from vehicle k to MeNB $S_u \in \mathcal{M}_2$, we can also get the achievable spectrum efficiency h_k^u and transmission rate R_k^u accordingly.

For uplink transmission from vehicle k to the associated Wi-Fi AP, the achievable spectrum efficiency includes two parts since spectrum sharing is considered between a Wi-Fi AP and an MeNB and between two Wi-Fi APs. Taking Wi-Fi AP $W_i \in \mathcal{A}_j$ ($S_j \in \mathcal{M}_1$) as an example. According to spectrum reusing shown in Fig. 2(b), both spectrum resource $R_{\max}\alpha_2$ and $R_{\max}\alpha_w$ are available for uplinks under Wi-Fi AP W_i. Using subscripts m and w to distinguish the reused spectrum resources that are sliced to MeNBs and Wi-Fi APs, respectively, i.e., $R_{\max}\alpha_2$ and $R_{\max}\alpha_w$. Then the corresponding spectrum efficiency achieved from these two parts can be described as

$$h_{m,k}'^i = \log_2\left(1 + \frac{PG_k'^i}{\sum_{W_g \in \mathcal{A}_{m_1}, g \neq i} PG_k'^g + \sum_{S_j \in \mathcal{M}_2} PG_k^j + \sigma^2}\right) \tag{6}$$

$$h_{w,k}'^i = \log_2\left(1 + \frac{PG_k'^i}{\sum_{W_g \in \{\mathcal{A}_{m_1} \cup \mathcal{A}_{m_2}\}, g \neq i} PG_k'^g + \sigma^2}\right), \tag{7}$$

respectively, where $\mathcal{A}_{m_1}$ and $\mathcal{A}_{m_2}$ are the sets of Wi-Fi APs within the coverages of MeNBs in $\mathcal{M}_1$ and in $\mathcal{M}_2$, respectively. Let $f_{m,k}'^i$ and $f_{w,k}'^i$ be the spectrum fractions allocated to vehicle k by Wi-Fi AP W_i from $R_{\max}\alpha_2$ and from $R_{\max}\alpha_w$, respectively. Then, the transmission rate achieved by the uplink from vehicle k to Wi-Fi AP W_i can be given by

$$R_{j,k}'^i = R_{\max}\alpha_2 f_{m,k}'^i h_{m,k}'^i + R_{\max}\alpha_w f_{w,k}'^i h_{w,k}'^i, \tag{8}$$

where the subscript j of $R_{j,k}'^i$ indicates that Wi-Fi AP W_i is within the coverage of MeNB S_j.

D. Computing and Storing Resource Allocation Models

For vehicles under the service area of the MEC server, we assume each vehicle randomly generates computing tasks, such as image processing tasks for assisting in automatic driving [32], and periodically sends them to the MEC server with their driving state information together. Denote $\{c_k^s, c_k^c, d_k\}$ as the computing task generated by vehicle k, where c_k^s is the data size of the computing task that needs to be transmitted to and cached by the MEC server, c_k^c denotes the required number of CPU cycles for task execution, and d_k is the maximum delay tolerated by the task. Without loss of generality, we assume a vehicle generates different computing tasks during different time slots and the vehicle distribution under the considered road segment changes with time.

Due to the high and stable data rate achieved by the wired connection between a BS and the MEC server, the transmission time in a wired link is relatively small and is neglected in this paper. Assume each MEC server is equipped with computing capabilities of $C_{\max}^c$ CPU cycles per second (i.e., Hz) and storing capabilities of $C_{\max}^s$ kbits. Let f_k^c be the fraction of computing resources allocated to vehicle k. Then, under the service area of an MeNB-mounted MEC server, the total time consumption on offloading and executing the computing task generated by vehicle k can be expressed as

$$T_k = \begin{cases} \frac{c_k^s}{R_k^m} + \frac{c_k^c}{C_{\max}^c f_k^c}, & \text{if } v_k^m = 1 \\ \frac{c_k^s}{R_k'^i} + \frac{c_k^c}{C_{\max}^c f_k^c}, & \text{if } v_k'^i = 1, \end{cases} \tag{9}$$

where binary variable v_k^m (or $v_k'^i$) is the association pattern between vehicle k and the MeNB (or Wi-Fi AP W_i), which

equals to 1 if vehicle k associates to the MeNB (or Wi-Fi AP W_i), and 0 otherwise. We assume a vehicle only associates to an MeNB when it is outside of the coverage of any Wi-Fi AP. For vehicles under the service area of an EN-mounted MEC server, let v_k^j (or $v_{j,k}^{\prime i}$) be the association pattern between vehicle k and MeNB S_j (or Wi-Fi AP W_i in $\mathcal{A}_j$). Then we can express the total time consumption on offloading vehicle k's task through MeNB S_j or Wi-Fi AP $W_i \in \mathcal{A}_j$ to and executing the task at the EN-mounted MEC server as

$$T_k = \begin{cases} \frac{c_k^s}{R_k^j} + \frac{c_k^c}{C_{\max}^c f_k^c}, & \text{if } v_k^j = 1 \\ \frac{c_k^s}{R_{j,k}^{\prime i}} + \frac{c_k^c}{C_{\max}^c f_k^c}, & \text{if } v_{j,k}^{\prime i} = 1. \end{cases} \tag{10}$$

Denote f_k^s as the fraction of storing resources allocated to vehicle k. Then, we call vehicle k's offloaded task is completed with satisfied QoS requirements when the following two conditions are satisfied: (1) at least c_k^s storing resources are allocated to vehicle k, i.e., $f_k^s C_{\max}^s \geq c_k^s$, and (2) the spectrum and computing resources allocated to vehicle k are enough for transmitting the c_k^s data to and executing the task at the MEC server with a total time cost less than d_k.

III. Problem Formulation and Transformation

In this section, we first formulate problems to jointly manage the spectrum, computing, and storing resources, and then transform the formulated problems based on RL.

A. Problem Formulation

In the MEC-based vehicular network, moving vehicles send their moving state, position, and task information, i.e., $\{c_k^s, c_k^c, d_k\}$, to the MEC server. Then based on the collected information, the controller centrally manages the spectrum, computing, and storing resources among vehicles with task offloading requests. How to efficiently allocate the limited three dimensions of resources to support as many task offloading requests with satisfied QoS requirements as possible is important to the MEC-based vehicular networks. To achieve this goal, problems are formulated to maximize the numbers of offloaded tasks that are completed with satisfied QoS requirements by the MeNB- and EN-mounted MEC servers. The one for the MeNB-mounted MEC server is as follows,

$$\max_{\beta_m, \beta_w, \mathbf{f}, \mathbf{f}^c, \mathbf{f}^s} \sum_{k \in \mathcal{N}} H(d_k - T_k) H(f_k^s C_{\max}^s - c_k^s), \tag{11}$$

$$\text{s.t.} \begin{cases} \beta_m, \beta_w \in [0,1] & (11a) \\ \beta_m + \beta_w = 1 & (11b) \\ f_k^m, f_k^{\prime i}, f_k^c, f_k^s \in [0,1], \quad k \in \mathcal{N} & (11c) \\ \sum_{k \in \mathcal{N}_m} f_k^m = 1 & (11d) \\ \sum_{k \in \mathcal{N}^i} f_k^{\prime i} = 1, \quad \forall i & (11e) \\ \sum_{k \in \mathcal{N}} f_k^c = 1 & (11f) \\ \sum_{k \in \mathcal{N}} f_k^s = 1, & (11g) \end{cases}$$

where T_k is given by equation (9), $\mathcal{N}$ is the set of vehicles under the service area of the MEC server, and $\mathcal{N}_m$ (or $\mathcal{N}^i$) is the set of vehicles associated to the MeNB (or Wi-Fi AP W_i).

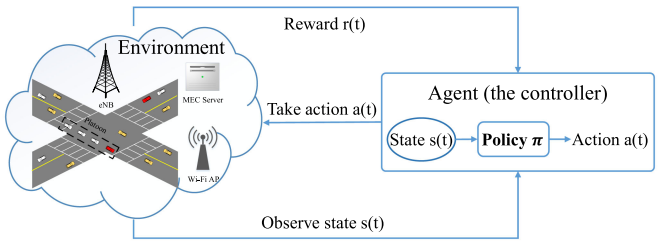


Fig. 3. The fundamental deep RL architecture in the MEC-based vehicular Network.

$\mathbf{f} = \{f_k^m, f_k^{\prime i}\}$ $(k \in \mathcal{N})$ is the spectrum allocation matrix to vehicles by the MeNB and Wi-Fi APs W_i. $\mathbf{f}^c$ and $\mathbf{f}^s$ are the matrices describing computing and storing resources allocated to vehicles by the MEC server, respectively. To describe an offloaded task with satisfied QoS requirements, the Heaviside step function denoted by $H(\cdot)$ is involved in the objective function, which is 1 if the variable is larger or equal to 0, and 0 otherwise. Then, for an offloaded task that is generated by vehicle k and completed with satisfied QoS requirements, we have $H(d_k - T_k)H(f_k^s C_{\max}^s - c_k^s) = 1$. Similarly, according to equation (10) and the communication model corresponding to Fig. 2(b), a joint resource management optimization problem can be also formulated for the scenario with EN-mounted MEC servers.

B. Problem Transformation With Deep RL

As the two formulated optimization problems are both non-convex due to the Heaviside step function, traditional optimization methods are infeasible without transforming the original objective functions. Due to the coupled relation among optimization variables β_m, β_w, $\mathbf{f}$, $\mathbf{f}^c$, and $\mathbf{f}^s$, the original problems have to be decomposed into subproblems and then leverage some alternate concave search algorithms to solve them [33]. Nevertheless, overlong solving time is resulted due to the alternating process among subproblems, which would be further increased with the number of vehicles under the service area of the MEC server.

It is critical to rapidly obtain an optimal resource allocation decision for a given dynamic environment state with delay-sensitive tasks. Thus, we model the above resource allocation decision making problems as Markov Decision Processes (MDPs) [34], and then adopt deep RL methods to solve them [35]. As shown in Fig. 3, the fundamental deep RL architecture consists of agent and environment interacting with each other [36]. The agent is implemented by the controller installed at each MEC server and everything beyond the controller is regarded as the environment. Through learning the best policy (called as resource allocation policy that maps the environment state to a resource allocation decision) to maximize the total accumulated reward, the above problems can be solved directly.

Environment state: As discussed before, each vehicle periodically sends the driving state information and task information to the MEC server. By collecting such information, the agent (i.e., the controller) can obtain the environment state. Denote $\mathcal{S}$ as the state space, $N(t) = |\mathcal{N}|$ as the number of vehicles under the service area of the MEC server at time slot t, and $x_k(t)$ and $y_k(t)$ as the x- and y- coordinates of the

position of vehicle k. Then, the environment state at time slot t, $s(t) \in \mathcal{S}$, can be described as

$$\begin{aligned} s(t) =&\{x_1(t), x_2(t), \ldots, x_{N(t)}(t), y_1(t), y_2(t), \ldots, \\ & y_{N(t)}(t), c_1^s(t), c_2^s(t), \ldots, c_{N(t)}^s(t), c_1^c(t), c_2^c(t), \\ & \ldots, c_{N(t)}^c(t), d_1(t), d_2(t), \ldots, d_{N(t)}(t)\}, \end{aligned} \tag{12}$$

According to each vehicle's position information, the uplink channel gain from a vehicle to a BS can be obtained by the agent. Note that the environment state should be adjusted according to the association patterns between vehicles and MeNBs (or Wi-Fi APs).

Action: Based on the observed environment states in $\mathcal{S}$, the agent will make resource allocation decisions according to the resource allocation policy π. Denote $\mathcal{A}$ as the action space. Then the action taken by the MeNB-mounted MEC server at time slot t, including the spectrum slicing ratio set $\{\beta_m(t), \beta_w(t)\}$, spectrum allocation fraction sets for the MeNB $f_k^m(t)$ and for each Wi-Fi AP $f_k'^i(t)$ ($W_i \in \mathcal{A}_m$), computing resource allocation fraction $f_k^c(t)$, and storing resource allocation fraction $f_k^s(t)$, can be given by

$$\begin{aligned} a(t) =&\{\beta_m(t), \beta_w(t), f_1^m(t), f_2^m(t), \ldots, f_{N(t)}^m(t), \\ & f_1'^i(t), f_2'^i(t), \ldots, f_{N^i(t)}'^i(t), f_1^s(t), f_2^s(t), \ldots, \\ & f_{N(t)}^s(t), f_1^c(t), f_2^c(t), \ldots, f_{N(t)}^c(t)\}, \forall W_i \in \mathcal{A}_m \end{aligned} \tag{13}$$

and that for the EN-mounted MEC server is

$$\begin{aligned} a(t) =& \{\alpha_1(t), \alpha_2(t), \alpha_w(t), f_1^m(t), f_2^m(t), \ldots, f_{N(t)}^m(t), \\ & f_1'^i(t), f_2'^i(t), \ldots, f_{N^i(t)}'^i(t), f_1^s(t), f_2^s(t), \ldots, f_{N(t)}^s(t), \\ & f_1^c(t), f_2^c(t), \ldots, f_{N(t)}^c(t)\}, \forall W_i \in \{\mathcal{A}_{m_1} \cup \mathcal{A}_{m_2}\}, \end{aligned} \tag{14}$$

where $N^i(t) = |\mathcal{N}^i|$ is the number of vehicles associated to Wi-Fi AP W_i at time slot t.

Reward: As shown in Fig. 3, once the agent takes action $a(t-1)$ based on the observed environment state $s(t-1)$, the environment will return an immediate reward $r(t)$ to the agent.[2] Then in the learning stage, the agent updates the resource allocation policy, π, based on the received reward until the algorithm converged. Indicated by equation (11), the delay requirement and requested storing resources should be simultaneously satisfied to guarantee the QoS requirements of an offloaded task. Thus, to maximize the number of offloaded tasks that are completed with satisfied QoS requirements by the MEC server at time slot t, we define the following two reward elements for offloaded task k that are corresponding to $H(d_k - T_k)$ and $H(f_k^s C_{\max}^s - c_k^s)$ of equation (11),

$$r_k^d(t) = \log_2\left(\frac{d_k(t-1)}{T_k(t-1)} + 0.00095\right) \tag{15}$$

$$r_k^s(t) = \log_2\left(\frac{C_{\max}^s f_k^s(t-1)}{c_k^s(t-1)} + 0.00095\right), \tag{16}$$

[2] In RL, the immediate reward at time slot t, $r(t)$, is the consequence of action token at the previous time slot, $a(t-1)$.

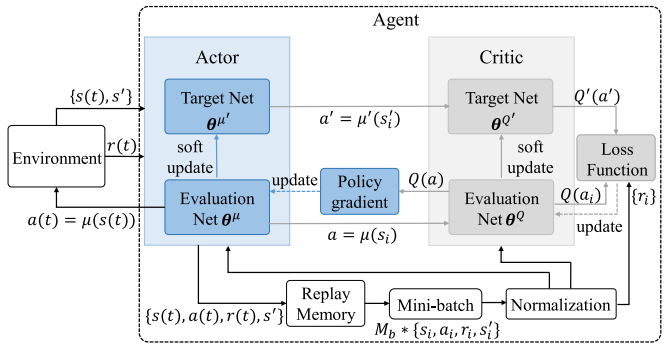


Fig. 4. The architecture of the DDPG learning.

where $r_k^d(t) \geq 0$ when the delay requirement $d_k(t-1)$ is satisfied by the allocated spectrum and computing resources, and $r_k^s(t) \geq 0$ when enough storing resources are allocated to task k, otherwise negative $r_k^d(t)$ and $r_k^s(t)$ are obtained. To improve the convergence performance of the RL algorithm and the fairness among vehicles, we use the logarithm function to define the reward elements. And a small value 0.00095 is added when calculating the logarithm reward to the base 2, such that the minimum value of each reward element is limited to -10 to avoid sharp fluctuation.

IV. DDPG Algorithm Based Solution

According to whether the agent can learn the environment for decision making in advance or not, deep RL algorithms can be classified into two categories: model-based and model-free. In the considered MEC-based vehicular network, the environment state dynamically changes over time due to the infinite channel states and dynamic task offloading requests. That is, the agent cannot make a resource allocation decision for the subsequent time slot according to the current observed environment state. Thus, we consider a model-free deep RL algorithm with an uncertain environment in this work. Moreover, the environment state and action vectors given in the previous section indicate that the sizes of state space $\mathcal{S}$ and action space $\mathcal{A}$ are infinite. Hence, policy-based[3] and model-free deep RL algorithms, such as policy gradient, actor-critic, deterministic policy gradient (DPG), and DDPG, should be adopted. In this work, we combine the DDPG with normalization and hierarchical learning architecture to solve the modeled MDPs.

DDPG is an improved actor-critic algorithm, which combines the advantages of policy gradient and deep Q-network (DQN) algorithms. As the DDPG architecture shown in Fig. 4, the agent is more complex compared with the fundamental deep RL architecture shown in Fig. 3, and is mainly comprised of two entities, actor and critic. Similar to DQN, both actor and critic adopt target nets with soft updated parameters to achieve stable convergence. We use two deep neural networks (DNNs) with the same structure but different parameters, i.e., an

[3] As opposed to the value-based RL algorithm, the policy-based deep RL algorithm makes an action decision according to actions' probabilities, and it can be applied to scenarios with infinite state and action spaces.

evaluation net with real-time updated parameters and a target net with soft updated parameters, to realize the actor and critic. Let $\boldsymbol{\theta}^{\mu}$ and $\boldsymbol{\theta}^{\mu'}$ (or $\boldsymbol{\theta}^{Q}$ and $\boldsymbol{\theta}^{Q'}$) be the parameter matrices of the evaluation net and target net in the actor (or in the critic), respectively. Then the parameters of the target nets are soft updated with that of the evaluation nets as follows,

$$\boldsymbol{\theta}^{\mu'} = \kappa_a \boldsymbol{\theta}^{\mu} + (1-\kappa_a)\boldsymbol{\theta}^{\mu'} \tag{17}$$

$$\boldsymbol{\theta}^{Q'} = \kappa_c \boldsymbol{\theta}^{Q} + (1-\kappa_c)\boldsymbol{\theta}^{Q'} \tag{18}$$

with $\kappa_a \ll 1$ and $\kappa_c \ll 1$. Denote $\mu(\cdot)$ and $\mu'(\cdot)$ (or $Q(\cdot)$ and $Q'(\cdot)$) as the network functions of the evaluation and target nets in the actor (or in the critic). Same as the fundamental RL algorithm, the agent first makes a resource allocation action $a(t) = \mu(s(t))$ according to the observed environment state $s(t)$, and then waits for reward $r(t)$ and next subsequent state s' from the environment.

To improve the performance and stability of the evaluation and target nets and accelerate the convergence rate of DDPG, experience replay and input normalization are considered. As shown in Fig. 4, each experience of the agent, denoted by $\{s(t), a(t), r(t), s'\}$, is saved in the replay memory. Assume up to M_r experiences can be saved in the replay memory. Then once the number of experiences saved in the replay memory reaches to M_r, the first saved experience will be replaced by the new coming experience and the learning stage of the DDPG starts. In each step during the learning stage, the agent randomly chooses a mini batch of M_b experiences from the replay memory, denoted by $\{s_i, a_i, r_i, s_i'\}$ $(i = 1, \ldots, M_b)$, to update $\boldsymbol{\theta}^{\mu}$ and $\boldsymbol{\theta}^{Q}$. By randomly choosing experiences from the replay memory, the correlation among experiences are disrupted, and therefore accelerating the convergence rate. Moreover, considering a great difference among data elements of an experience would result in a large number of artificial neurons with inactive outputs,[4] each experience is locally normalized before inputted to the actor and critic. Based on the architecture shown in Fig. 4, two algorithms, i.e., DDPG- and HDDPG-based algorithms, are designed to solve the MDPs in the MEC-based vehicular network.

A. DDPG-Based Algorithm

The DDPG-based algorithm is with exactly the same architecture in Fig. 4. For the two considered scenarios with MeNB- and EN-mounted MEC servers, the environment state vector at time slot t is given by equation (12). According to the basic idea of DDPG [37], the goal of the agent is to find an optimal resource allocation policy π with the maximum achievable long-term average reward, $E\{Q^{\pi}(s,a)\}$, and can be approached by the learning stage step by step. In which, $Q^{\pi}(s,a)$ is the standard Q-value function used to evaluate the state-action value under policy π, and can be given by

$$Q^{\pi}(s,a) = E\left\{\sum_{\tau=0}^{\infty} \gamma^{\tau} r(t+\tau) \big| \pi, s = s(t), a = a(t)\right\}, \tag{19}$$

where $r(t)$ is the immediate reward received by the agent at time slot t, $\gamma \in (0,1)$ is a discount factor on $r(t)$, and τ is a time slot counting symbol. In order to achieve the objective functions of the formulated optimization problems, we define the reward achieved by vehicle k at time slot t, $r_k(t)$, as

$$r_k(t) = r_k^d(t) + r_k^s(t), \tag{20}$$

such that a positive reward can be guaranteed once vehicle k's offloaded task is completed with satisfied QoS requirements by the MEC server. Considering the vehicle density changes over time, the immediate reward, $r(t)$, then is defined as the average reward over vehicles within the service area of the MeNB- or EN-mounted MEC server, i.e., $r(t) = \frac{1}{N(t)} \sum_{k \in \mathcal{N}} r_k(t)$.

For the DDPG-based algorithm, the evaluation net of the actor is the one that makes the resource allocation action for each given environment state. Thus, how to update the parameter matrix, $\boldsymbol{\theta}^{\mu}$, during the learning stage is the key to find the optimal π in DDPG. As mentioned, in each step during the learning stage, a mini batch of experiences are randomly chosen from the replay memory and then inputted to the agent one by one. With each inputted experience, the actor and critic interact with each other to update the parameter matrices of their evaluation nets, $\boldsymbol{\theta}^{\mu}$ and $\boldsymbol{\theta}^{Q}$. Taking the i-th experience from the mini batch of experiences, $\{s_i, a_i, r_i, s_i'\}$, as an example, the interaction can be summarized as, (1) the actor makes resource allocation actions according to the two adjacent environment states, $a = \mu(s_i)$ and $a' = \mu'(s_i')$, by using the evaluation and target nets, respectively; (2) the critic evaluates a and a_i with its evaluation net[5], $Q(a)$ and $Q(a_i)$, and evaluates a' with its target net, i.e., $Q'(a')$; (3) the actor updates $\boldsymbol{\theta}^{\mu}$ according to the policy gradient to maximize $E\{Q(a)\}$, and the critic updates $\boldsymbol{\theta}^{Q}$ according to the loss function to minimize the temporal difference (TD)-error for each inputted experience. Here, the TD-error describes the difference between the estimated Q-value $Q(a_i)$ and the target Q-value $r_i + \gamma Q'(a')$, which can be given by

$$\varepsilon_i = Q(a_i) - (r_i + \gamma Q'(a')). \tag{21}$$

And then, the loss function can be described as $L(\boldsymbol{\theta}^{Q}) = E\{(\varepsilon_i)^2\}$.

From the critic perspective, the network functions of the evaluation and target nets are in charge of estimating the Q-value functions. As the achievable Q-value under the optimal policy π is $Q^{\pi}(s,a)$, i.e., $\pi(a|s) = \arg\max_a Q^{\pi}(s,a)$ [38], the parameter updating in the critic during the learning stage is similar to DQN to make $Q(a)$ approximate $Q^{\pi}(s,a)$. As mentioned, a mini batch of M_b experiences are adopted in each step during the learning stage. Assume that the loss function, $L(\boldsymbol{\theta}^{Q})$, is continuously differentiable with respect to $\boldsymbol{\theta}^{Q}$. Then

[4] An output that is on the boundary of the output range is called as an inactive output of the artificial neuron, which depends on the input data and the enabled activation function. For example, -1 and 1 are inactive outputs of an artificial neuron with tanh activation function. A large number of artificial neurons with inactive outputs would reduce the convergence rate of the learning algorithm.

[5] Even though s_i and a_i are the state-action pair and we have $a = \mu(s_i)$, a_i and a may be two different actions due to the updating on the parameter matrix $\boldsymbol{\theta}^{\mu}$. Thus, $Q(a)$ and $Q(a_i)$ may also be different.

the critic can update θ^Q with the gradient of $L(\theta^Q)$ as follows,

$$\Delta\theta^Q = \lambda_c \frac{1}{M_b} \sum_{\mathcal{M}_b} \varepsilon_i \nabla_{\theta^Q} Q(a_i), \tag{22}$$

where λ_c is the learning rate of the critic and $\nabla_{\theta^Q} Q(a_i)$ is the derivative of $L(\theta^Q)$ with respect to θ^Q.

The role of the actor is to make a resource allocation action for each given environment state. Considering the goal of the agent, the actor should update its parameter matrix to make $\mu(s_i)$ approximate the optimal resource allocation policy π to achieve the maximum $E\{Q^\pi(s,a)\}$. Thus, the policy objective function, used to evaluate the policy under a given parameter matrix θ^μ, can be defined as $J(\theta^\mu) = E\{Q(a)\}$ as in [39], where $a = \mu(s_i)$. The actor adopts the policy gradient method to update the parameter matrix θ^μ during the learning stage. Similar to the critic, when the policy objective function $J(\theta^\mu)$ is continuously differentiable with respect to θ^μ, θ^μ then can be updated with the gradients of $J(\theta^\mu)$ as follows,

$$\Delta\theta^\mu = \lambda_a \frac{1}{M_b} \sum_{\mathcal{M}_b} \nabla_a Q(a)|_{a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s_i), \tag{23}$$

where λ_a is the learning rate of the actor, $\nabla_a Q(a)|_{a=\mu(s_i)}$ is the derivative of $Q(a)$ with respect to a with $a = \mu(s_i)$, and $\nabla_{\theta^\mu}\mu(s_i)$ is the derivative of $\mu(s_i)$ with respect to θ^μ. Note that equations (22) and (23) are obtained by using the derivative method for compound function.

Combining with the parameter updating processes in the critic and actor, the DDPG-based algorithm can be summarized in Algorithm 1. To better observe the convergence of the algorithm, the total rewards per episode, r_{ep}, are considered in the DDPG-based algorithm. Let M_s be the number of steps in each episode, then for the episode from time slot t_0 to $t_0 + M_s$, we have $r_{ep} = \sum_{t=t_0}^{M_s+t_0} r(t)$.

When leveraging the DDPG-based algorithm to solve the MDPs in the scenarios with MeNB- and EN-mounted MEC servers, the state and action vectors are given by equations (12), (13), and (14). The three equations indicate that the sizes of state and action vectors increase linearly with the number of vehicles under the service area. That is more complex MDP would be modeled due to the increasing of vehicle density in or service area of an MEC server, and therefore resulting in longer convergence time of the DDPG-based algorithm. Moreover, the total time consumption on offloading and executing vehicle k's task, $T_k(t)$ (given by equations (9) or (10)), is co-determined by the spectrum and computing resource allocation results. In an action, the spectrum slicing ratios and the spectrum/computing allocation fractions are the elements related to spectrum and computing resource allocation. And the impacts of these elements on the reward achieved by vehicle k are only implied by r_k^d. With the MDP becoming more complex, it will be more difficult for the DDPG-based algorithm to learn the impacts of such elements, and therefore resulting in longer convergence time. To overcome the above issues, we combine the architecture shown in Fig. 4 with the hierarchical learning architecture [40], and propose a HDDPG-based algorithm in the next subsection.

Algorithm 1. The DDPG-based algorithm

```
/* Initialization phase */
Initialize the evaluation and target nets for the actor and critic with
parameter matrices θ^μ, θ^μ', θ^Q, and θ^Q';
Initialize the replay memory buffer.
/* Parameter updating phase */
foreach episode do
  Receive initial observation state s^0 and reset r_ep = 0.
  foreach step t do
    Make an action a(t) = μ(s(t)) with the parameter matrix θ^μ;
    Receive the subsequent state s' and reward r(t);
    if the number of experiences < M_r then
      Store the experience {s(t), a(t), r(t), s'} into the replay
       memory;
    else
      Replace the first saved experience with {s(t), a(t), r(t), s'}
      in the replay memory;
      Randomly select a mini batch of M_b experiences from the
      replay memory;
      Update the parameter matrices in the critic:
      θ^Q ← θ^Q + Δθ^Q
      θ^Q' = κ_c θ^Q + (1 − κ_c) θ^Q';
      Update the parameter matrices in the actor:
      θ^μ ← θ^μ + Δθ^μ
      θ^μ' = κ_a θ^μ + (1 − κ_a) θ^μ'.
    r_ep = r_ep + r(t).
```

B. HDDPG-Based Algorithm

According to the DDPG architecture shown in Fig. 4 and the hierarchical learning architecture, the main idea of the HDDPG-based algorithm is to decompose the original MDP into sub-MDPs. By involving a variable matrix $\boldsymbol{\nu} = \{\nu_1, \nu_2, \ldots, \nu_N\}$, defined as the proportion of delay allowed by each vehicle's data transmission, the original resource allocation decision making problem can be decomposed into two subproblems for spectrum allocation and computing/storing resource allocation. The spectrum allocation subproblem is in charge of spectrum slicing among BSs and spectrum allocation among vehicles while the computing/storing resource allocation subproblem is to manage the MEC server's computing and storing resources among the received tasks. Then we solve the two subproblems with two DDPG algorithms (spectrum DDPG algorithm and computing DDPG algorithm). For the spectrum DDPG algorithm, the state and action vectors can be given by

$$\begin{aligned} s_{sp}(t) = \{&x_1(t), x_2(t), \ldots, x_{N(t)}(t), y_1(t), y_2(t), \ldots, \\ &y_{N(t)}(t), c_1^s(t), c_2^s(t), \ldots, c_{N(t)}^s(t), \nu_1 d_1(t), \\ &\nu_2 d_2(t), \ldots, \nu_{N(t)} d_{N(t)}(t)\} \end{aligned} \tag{24}$$

$$\begin{aligned} a_{sp}(t) = \{&\alpha_1(t), \alpha_2(t), \alpha_w(t), f_1^m(t), f_2^m(t), \ldots, \\ &f_{N(t)}^m(t), f_1'^i(t), f_2'^i(t), \ldots, f_{N^i(t)}'^i(t)\}, \\ &\forall W_i \in \{\mathcal{A}_{m_1} \cup \mathcal{A}_{m_2}\} \end{aligned} \tag{25}$$

respectively. At time slot t, each vehicle's time consumption on data transmission, denoted by $\boldsymbol{T}' = \{T_1'(t), T_2'(t), \ldots, T_{N(t)}'(t)\}$, can be obtained by taking action $a_{sp}(t)$ on the current state. Then, for the spectrum DDPG algorithm, we define the reward achieved by vehicle k as

$$r'_k(t) = \log_2\left(\frac{v_k d_k(t-1)}{T'_k(t-1)} + 0.00095\right). \quad (26)$$

And the immediate reward received by the agent at time slot t, $r_{sp}(t)$, can be given by $r_{sp}(t) = \frac{1}{N(t)}\sum_{k\in\mathcal{N}} r'_k(t)$.

Similarly, we can define the environment state and action vectors for the computing DDPG algorithm as follows,

$$\begin{aligned} s_{cp}(t) = \{&T'_1(t), T'_2(t), \ldots, T'_{N(t)}(t), c^s_1(t), c^s_2(t), \\ &\ldots, c^s_{N(t)}(t), c^c_1(t), c^c_2(t), \ldots, c^c_{N(t)}(t), \\ &d_1(t), d_2(t), \ldots, d_{N(t)}(t)\} \end{aligned} \quad (27)$$

$$\begin{aligned} a_{cp}(t) = \{&f^s_1(t), f^s_2(t), \ldots, f^s_{N(t)}(t), \\ &f^c_1(t), f^c_2(t), \ldots, f^c_{N(t)}(t)\}. \end{aligned} \quad (28)$$

By updating equation (10) with $T_k(t) = T'_k(t) + c^c_k(t)/(C^c_{\max} f^c_k(t))$, the reward achieved by vehicle k in the computing DDPG algorithm can be given by equation (20). And the immediate reward received by the agent of the computing DDPG is $r_{cp}(t) = \frac{1}{N(t)}\sum_{k\in\mathcal{N}} r_k(t)$.

As indicated by the action vector in the computing DDPG algorithm, the computing DDPG algorithm correlates with the spectrum DDPG algorithm. Thus, we regard the action made by the agent of the spectrum DDPG as a part of the environment of the computing DDPG algorithm. To distinguish the two DDPG architectures in the HDDPG, a subscript is added to the notations defined in Section IV-A. For example, $\{s_{sp}(t), a_{sp}(t), r_{sp}(t), s'_{sp}\}$ and $\{s_{cp}(t), a_{cp}(t), r_{cp}(t), s'_{cp}\}$ are denoted as the agent's experiences at time slot t in the spectrum DDPG and computing DDPG, respectively. Similar to the DDPG-based algorithm, the HDDPG-based algorithm can be summarized in Algorithm 2, where $r_{eps} = \sum_{t=t_0}^{M_s+t_0} r_{sp}(t)$ and $r_{epc} = \sum_{t=t_0}^{M_s+t_0} r_{cp}(t)$ are the episode rewards for the spectrum DDPG and computing DDPG.

Algorithm 2. HDDPG-based algorithm

/* `Initialization phase` */
Initialize the evaluation and target nets for the actor and critic in the spectrum DDPG with parameter matrices $\boldsymbol{\theta}^{\mu}_{sp}$, $\boldsymbol{\theta}^{\mu'}_{sp}$, $\boldsymbol{\theta}^{Q}_{sp}$, and $\boldsymbol{\theta}^{Q'}_{sp}$;
Initialize the evaluation and target nets for the actor and critic in the computing DDPG with parameter matrices $\boldsymbol{\theta}^{\mu}_{cp}$, $\boldsymbol{\theta}^{\mu'}_{cp}$, $\boldsymbol{\theta}^{Q}_{cp}$, and $\boldsymbol{\theta}^{Q'}_{cp}$;
Initialize the replay memory buffers for both spectrum DDPG and computing DDPG.
/* `Parameter updating phase` */
foreach *episode* **do**
  Receive initial observation state s^0_{sp}, and reset $r_{eps} = 0$ and $r_{epc} = 0$.
  foreach *step* t **do**
    Make an action $a_{sp}(t) = \mu_{sp}(s_{sp}(t))$ with the parameter matrix $\boldsymbol{\theta}^{\mu}_{sp}$, receive the subsequent state s'_{sp} and reward $r_{sp}(t)$, and obtain $\boldsymbol{T}' = \{T'_1(t), T'_2(t), \ldots, T'_{N(t)}(t)\}$;
    Make an action $a_{cp}(t) = \mu_{cp}(s_{cp}(t))$ with the parameter matrix $\boldsymbol{\theta}^{\mu}_{cp}$, and receive the subsequent state s'_{cp} and reward $r_{cp}(t)$;
    if *the number of experiences* $< M_r$ **then**
      Store $\{s_{sp}(t), a_{sp}(t), r_{sp}(t), s'_{sp}\}$ and $\{s_{cp}(t), a_{cp}(t), r_{cp}(t), s'_{cp}\}$ into the replay memory buffers of the spectrum DDPG and computing DDPG, respectively;
    else
      Replace the first saved experiences with $\{s_{sp}(t), a_{sp}(t), r_{sp}(t), s'_{sp}\}$ and $\{s_{cp}(t), a_{cp}(t), r_{cp}(t), s'_{cp}\}$ in the two replay memory buffers;
      Randomly select two mini batches of M_b experiences from the replay memory buffers of the spectrum DDPG and computing DDPG;
      Update the parameter matrices in the critic of the spectrum DDPG:
      $\boldsymbol{\theta}^{Q}_{sp} \leftarrow \boldsymbol{\theta}^{Q}_{sp} + \Delta\boldsymbol{\theta}^{Q}_{sp}$
      $\boldsymbol{\theta}^{Q'}_{sp} = \kappa_c \boldsymbol{\theta}^{Q}_{sp} + (1-\kappa_c)\boldsymbol{\theta}^{Q'}_{sp}$;
      Update the parameter matrices in the actor of the spectrum DDPG:
      $\boldsymbol{\theta}^{\mu}_{sp} \leftarrow \boldsymbol{\theta}^{\mu}_{sp} + \Delta\boldsymbol{\theta}^{\mu}_{sp}$
      $\boldsymbol{\theta}^{\mu'}_{sp} = \kappa_a \boldsymbol{\theta}^{\mu}_{sp} + (1-\kappa_a)\boldsymbol{\theta}^{\mu'}_{sp}$;
      Update the parameter matrices in the critic of the computing DDPG:
      $\boldsymbol{\theta}^{Q}_{cp} \leftarrow \boldsymbol{\theta}^{Q}_{cp} + \Delta\boldsymbol{\theta}^{Q}_{cp}$
      $\boldsymbol{\theta}^{Q'}_{cp} = \kappa_c \boldsymbol{\theta}^{Q}_{cp} + (1-\kappa_c)\boldsymbol{\theta}^{Q'}_{cp}$;
      Update the parameter matrices in the actor of the computing DDPG:
      $\boldsymbol{\theta}^{\mu}_{cp} \leftarrow \boldsymbol{\theta}^{\mu}_{cp} + \Delta\boldsymbol{\theta}^{\mu}_{cp}$
      $\boldsymbol{\theta}^{\mu'}_{cp} = \kappa_a \boldsymbol{\theta}^{\mu}_{cp} + (1-\kappa_a)\boldsymbol{\theta}^{\mu'}_{cp}$.
    $r_{eps} = r_{eps} + r_{sp}(t)$;
    $r_{epc} = r_{epc} + r_{cp}(t)$.

V. Simulation Results and Analysis

To demonstrate the performance of the proposed deep RL-based resource management schemes for the vehicular scenarios with MeNB- and EN-mounted MEC servers, simulation results are presented in this section. The simulation procedures of a RL-based algorithm can be summarized into two stages, i.e., learning stage to learn the model and inferring stage to test the learned model [41]. In this section, we first learn the DDPG- and HDDPG-based models for the two considered scenarios. Then, the learned models are tested under the scenarios with different amounts of available resources to measure the performance of the proposed resource management schemes.

We consider a two-lane straight country road with one for each direction, where the traffic flow on the road is generated by PTV Vissim [42]. For the scenario with the MeNB-mounted MEC servers, one MeNB and two Wi-Fi APs (AP 1 and AP 2) are deployed on one side of the road to support vehicular applications. And for the scenario with the EN-mounted MEC servers, we assume the MEC server is placed at an EN, where two adjacent MeNBs ($S_j \in \mathcal{M}_1$ and $S_{j+1} \in \mathcal{M}_2$) and four Wi-Fi APs (AP 1 and AP 2 in $\mathcal{A}_j$, and AP 3 and AP 4 in $\mathcal{A}_{j+1}$) are wired connected to the EN. In the simulation, we assume a vehicle chooses to associate to a Wi-Fi AP when it is under the coverage of that Wi-Fi AP, otherwise associate to the MeNB. The transmit power of each vehicle is set as 1 watt (i.e., 30 dBm), and the uplink channel gain between a vehicle and an MeNB (or a Wi-Fi AP) is described as $L_m(d') = -30 - 35\log_{10}(d')$ (or $L_w(d') = -40 - 35\log_{10}(d')$) [33], where d' is the distance between the vehicle user and the MeNB (or the Wi-Fi AP). Taking a type of delay-sensitive computing task (e.g., the analysis of the surveillance content of special road segments for approaching vehicles [43], [44]) as an example, the delay bound for the offloaded computing task is set as 50 ms, i.e.,

TABLE I
PARAMETERS FOR THE LEARNING STAGE

Parameter	Value
Data size of a computing task	$[0.8, 1.2]$ kbits
Number of CPU cycles required to execute a computing task	$[80, 120]$ Mcycles/s
Amount of aggregate spectrum resources at an MeNB-/EN-mounted MEC server	$10/20$ MHz
Computational capability at an MeNB-/EN-mounted MEC server	$100/200$ GHz
Capacity of storing at an MeNB-/EN-mounted MEC server	$60/120$ kbits
Communication range of an MeNB/Wi-Fi AP	$600/150$ m
Straight-line distance between the MeNB (or Wi-Fi AP) and the road	225 m (or 3 m)
Service range of an EN-mounted MEC server	1100 m
Background noise power	-104 dBm
Discount factor on immediate reward	0.92
κ_a/κ_c	0.005
Replay memory size	10000
Size of a mini batch of experiences	32
Learning rate of the actor/critic	$0.00005/0.0005$

$d_k = 50$ ms for $k \in \mathcal{N}$. Other parameters for the learning stage are listed in Table I. Unless specified otherwise, parameters for the inferring stage are same to that of the learning stage.

Figs. 5 and 6 demonstrate the convergence performance of the DDPG- and HDDPG-based algorithms in the scenarios with MeNB- and EN-mounted MEC servers, respectively. As we can see from the six subfigures, among the 1,200 episodes during the learning stage, the total rewards per episode fluctuate sharply and are relatively small in the first few hundreds episodes and then tend to be a relatively stable and high value. As mentioned in Algorithms 1 and 2, all parameters of the actors and critics of the DDPG- and HDDPG-based algorithms are initialized by the TensorFlow. Once 10,000 of experiences are saved in the replay memory buffers, the learning stages of the DDPG- or HDDPG-based algorithms start to update the parameters of the actor and critic. Thus, the total rewards per episode fluctuate sharply in the beginning of the learning stage and then increase with the parameters gradually being optimized. Moreover, to direct the agent to satisfy more tasks' QoS requirements, we clip $r_k^d(t)$ to be $[-8, 0.2]$ and $r_s^d(t)$ to be $[-7, 0.2]$. And the average number of steps in each episode is 1100. Hence, the maximum total rewards per episode achieved by the DDPG-based algorithm and the computing DDPG of the HDDPG-based algorithm is 440, as shown in Figs. 5(a), 5(c), 6(a), and 6(c). Similarly, the maximum total rewards per episode achieved by the spectrum DDPG of the HDDPG-based algorithm is 220, as shown in Figs. 5(b) and 6(b) is 220.

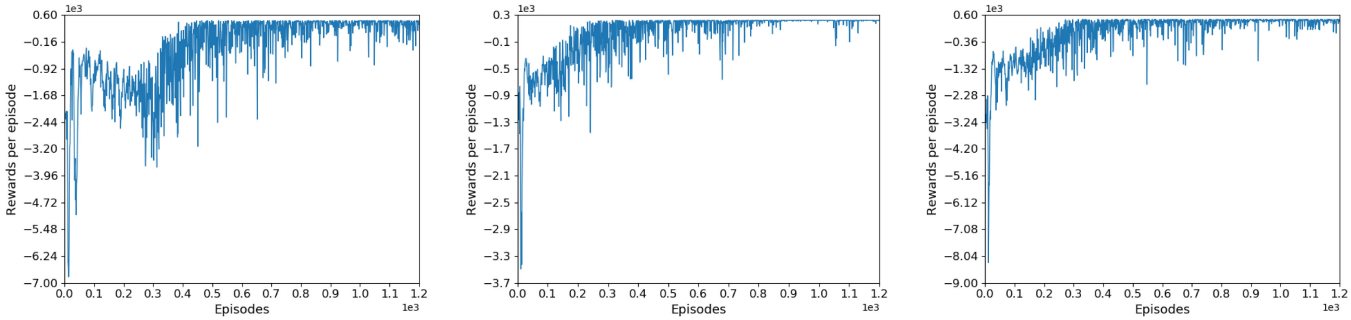
As the outputs of the spectrum DDPG are parts of the inputs of the computing DDPG in the HDDPG-based algorithm, we regard the convergence performance of the computing DDPG as that of the HDDPG-based algorithm. From Figs. 5(a) and 5(c) (or Figs. 6(a) and 6(c)), the HDDPG-based algorithm converges within a smaller number of episodes than the DDPG-based algorithm in the scenario with an MeNB-mounted MEC server (or with an EN-mounted MEC server). For example, in the scenario with an MeNB-mounted MEC server, the DDPG-based algorithm starts to converge after around 400 episodes and that of the HDDPG-based algorithm is around 300 episodes. That is because the impacts of the spectrum slicing and spectrum allocation on the final rewards are learned by the spectrum DDPG in the HDDPG-based algorithm. However, since two DDPG models need to be trained, the training time of 1200 episodes for the HDDPG-based algorithm is physically longer than that for the DDPG-based algorithm. Moreover, Figs. 5 and 6 indicate that more episodes are required for the proposed algorithms to converge in the scenario with an EN-mounted MEC server comparing to the scenario with an MeNB-mounted MEC server. This is due to the more complex MDP caused by the increased numbers of vehicles and BSs under the EN-mounted MEC server.

From the MEC server perspective, an efficient resource management scheme should be able to serve as many users as possible with the given available resources. And for a vehicle user, satisfied delay or QoS requirements is critical. Thus, to measure the performance of the two proposed deep RL-based resource management schemes, we define delay/QoS satisfaction ratio as the number of vehicles with satisfied delay/QoS requirements over the total number of vehicles within the service area and use them as the evaluation criterion in the inferring stage. Two comparisons are considered, the DPG-based scheme and the random resource management scheme. Considering the relatively large number of vehicles under the service area and the large size of a resource allocation action, it is unbefitting to discretize each action. Thus, we choose the DPG-based scheme, which is also applicable to MDPs with continuous state and action spaces as one of our comparisons. The other comparison is the random resource management scheme,[6] which can obtain resource allocation decisions rapidly, same as our proposed schemes.

Fig. 7 demonstrates the average delay/QoS satisfaction ratios over 5000 adjacent environment states in the scenario with the MeNB-mounted MEC server, with respect to different amounts of aggregated spectrum resources, computation capabilities, and capacity of storing, respectively. The three subfigures show that, except the cases with short supply available resources, higher average delay satisfaction ratios and doubled average QoS satisfaction ratios are achieved by the proposed DDPG- and HDDPG-based schemes compared with the DPG-based scheme[7] and the random resource

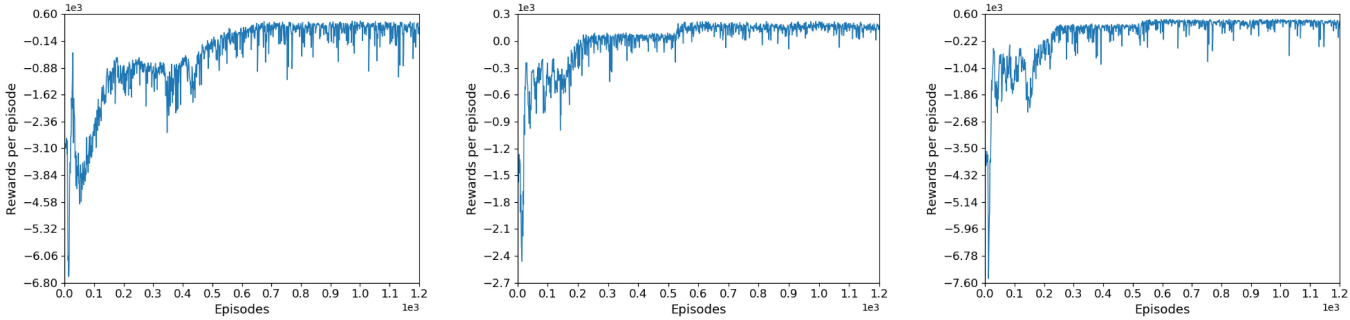
[6] with the random resource management scheme, the MEC server randomly allocates the spectrum, computing, and storing resources among vehicles under its service area.

[7] The DPG-based algorithm is trained with the same parameter setting as the two proposed schemes. Without deep learning, the convergence performance of the DPG-based algorithm cannot be guaranteed, especially in scenario with large sizes of environment state and action vectors. Thus, low delay/QoS satisfaction ratios are obtained by the DPG-based scheme.



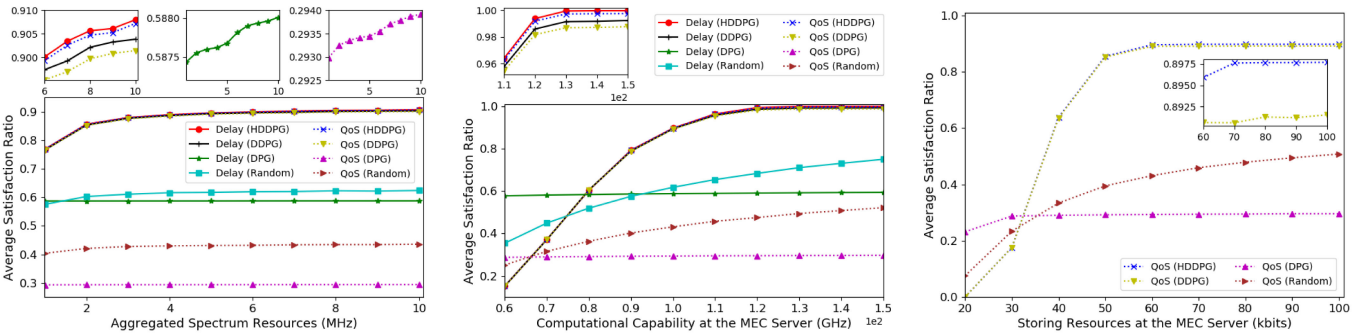
(a) DDPG-based algorithm (b) Spectrum DDPG in HDDPG-based algorithm (c) Computing DDPG in HDDPG-based algorithm

Fig. 5. The total rewards of each episode in scenario with an MeNB-mounted MEC server.



(a) DDPG-based algorithm (b) Spectrum DDPG in HDDPG-based algorithm (c) Computing DDPG in HDDPG-based algorithm

Fig. 6. The total rewards of each episode in scenario with an EN-mounted MEC server.

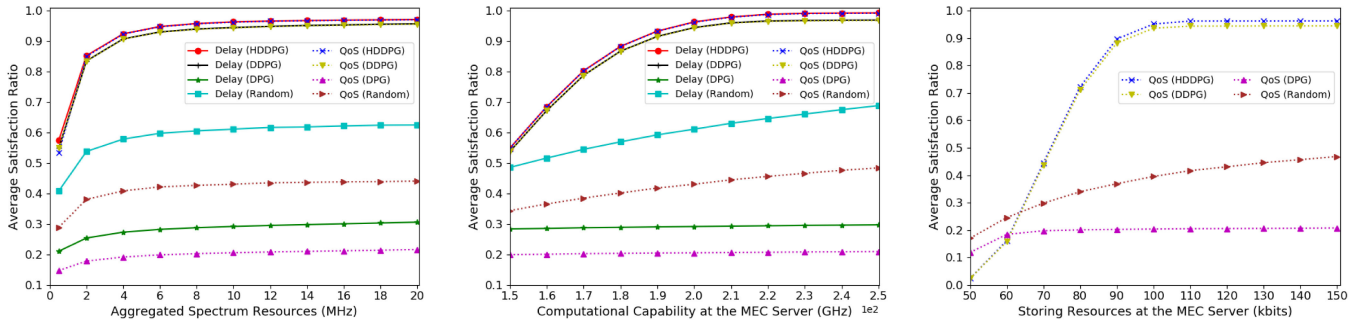


(a) Versus the aggregate spectrum resources (b) Versus the computation capabilities (c) Versus the capacity of storing

Fig. 7. Average delay/QoS satisfaction ratio over the vehicles under the service area of the MeNB-mounted MEC server.

management scheme. With the increasing of the amount of the available spectrum/computing/storing resources, the average satisfaction ratios achieved by the proposed schemes and the two comparisons increase due to the increasing of the amount of resources allocated to vehicle users. As mentioned before, the delay requirement is a part of its QoS requirement for an offloaded task, i.e., for a task with satisfied delay requirement, its QoS requirement would not be satisfied if no enough storing resources allocated to it. And the delay satisfaction of an offloaded task is co-determined by the amounts of spectrum and computing resources allocated to it. Thus, with the increasing of the amount of spectrum resources (or computing/storing), the delay/QoS satisfaction ratios achieved by the proposed schemes tend to be saturate due to the fixed amounts of the computing and storing resources (or the spectrum and storing/computing resources). Moreover, the gaps between the delay satisfaction ratios and QoS satisfaction ratios of the proposed schemes are much smaller than that of the two comparisons. That is because the proposed schemes have overall managed the spectrum/computing/storing resources to satisfy the QoS requirements for as many vehicles' tasks as possible.

Fig. 8 shows the average delay/QoS satisfaction ratios in the scenario with the EN-mounted MEC server. Similar to the scenario with the MeNB-mounted MEC server, the subfigures show that higher delay/QoS satisfaction ratios can be obtained by the



(a) Versus the aggregate spectrum resources (b) Versus the computation capabilities (c) Versus the capacity of storing

Fig. 8. Average delay/QoS satisfaction ratio over vehicles under the service area of the EN-mounted MEC server.

proposed resource management schemes compared with the DPG-based scheme and random resource management scheme. Except the scenarios with short supply available resources, such as the scenarios with 2 MHz or less spectrum resources available for the EN-mounted MEC server with 100 or more vehicles under its service area, over 90% of tasks are with satisfied QoS requirements under different environment states with the proposed schemes. Also, from Fig. 8 and the zoom in figures of Fig. 7, the HDDPG-based scheme outperforms the DDPG-based scheme in terms of the delay/QoS satisfaction ratios in addition to improved convergence. That is because two DDPG models are adopted to optimally manage the spectrum and computing/storing resources in the HDDPG-based scheme. Moreover, as the HDDPG-based algorithm is more applicable to the scenario with complex MDPs, compared to the scenario with the MeNB-mounted MEC server, more performance enhancements are achieved by the HDDPG-based algorithm than the DDPG-based algorithm in the scenario with the EN-mounted MEC server.

VI. Conclusions

In this paper, we have investigated the joint spectrum, computing, and storing resource management problem to accommodate delay-sensitive applications in the MEC-based vehicular network. Particularly, we have considered two typical MEC architectures, i.e., with MeNB- and EN-mounted MEC servers, under which two resource optimization problems have been formulated to maximize the number of offloaded tasks that are completed with satisfied QoS requirements. As the formulated problems are computationally intractable in real time, we have exploited the deep RL to transform and solve them and devised the DDPG- and HDDPG-based algorithms. Extensive simulation results have shown that our proposed DDPG- and HDDPG-based resource management schemes can converge within acceptable training episodes and outperform the DPG-based scheme and random resource management scheme in terms of delay/QoS satisfaction ratios.

Acknowledgment

The first author would like to thank Conghao Zhou, Hongli He, Yanglong Sun, and Hao Ye for their available comments on this paper and their help in the simulation code.

References

[1] J. Wang, J. Liu, and N. Kato, "Networking and communications in autonomous driving: A survey," *IEEE Commun. Surv. Tut.*, vol. 21, no. 2, pp. 1243–1274, Apr.–Jun., 2019.
[2] F. Lyu *et al.*, "Characterizing urban vehicle-to-vehicle communications for reliable safety applications," *IEEE Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2586–2602, Jun. 2020.
[3] L. Liang, H. Ye, and G. Y. Li, "Toward intelligent vehicular networks: A machine learning framework," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 124–135, Feb. 2019.
[4] H. Peng *et al.*, "Resource allocation for cellular-based inter-vehicle communications in autonomous multiplatoons," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 11 249–11 263, Dec. 2017.
[5] C. Zhang, K. Ota, J. Jia, and M. Dong, "Breaking the blockage for big data transmission: Gigabit road communication in autonomous vehicles," *IEEE Commun. Mag.*, vol. 56, no. 6, pp. 152–157, Jun. 2018.
[6] S. Darbha, S. Konduri, and P. R. Pagilla, "Benefits of V2V communication for autonomous and connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1954–1963, May 2019.
[7] H. Peng, L. Liang, X. Shen, and G. Y. Li, "Vehicular communications: A network layer perspective," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1064–1078, Feb. 2019.
[8] 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.
[9] 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.
[10] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. Quek, "Computation offloading for mobile edge computing enabled vehicular networks," *IEEE Access*, vol. 7, pp. 62 624–62 632, May 2019.
[11] Y. He, F. R. Yu, N. Zhao, V. C. Leung, and H. Yin, "Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 31–37, Dec. 2017.
[12] H. Peng, Q. Ye, and X. Shen, "SDN-based resource management for autonomous vehicular networks: A multi-access edge computing approach," *IEEE Wireless Commun.*, vol. 26, no. 4, pp. 156–162, Aug. 2019.
[13] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling," *IEEE Veh. Technol. Mag.*, vol. 14, no. 1, pp. 28–36, Mar. 2019.
[14] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 48–54, Aug. 2018.
[15] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
[16] 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. 11 158–11 168, Nov. 2019.

[17] H. Peng, Q. Ye, and X. Shen, "Spectrum management for multi-access edge computing in autonomous vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 3001–3012, Jul. 2020.
[18] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor-critic deep reinforcement learning," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2061–2073, Apr. 2019.
[19] J. Zhang *et al.*, "Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4283–4294, Jun. 2019.
[20] H. Li, K. Ota, and M. Dong, "Deep reinforcement scheduling for mobile crowdsensing in fog computing," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–18, Apr. 2019.
[21] L. Liang, H. Ye, G. Yu, and G. Y. Li, "Deep-learning-based wireless resource allocation with application to vehicular networks," *Proc. IEEE*, vol. 108, no. 2, pp. 341–356, Feb. 2020.
[22] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Wireless Commun.*, vol. 24, no. 2, pp. 98–105, Apr. 2017.
[23] Y. He *et al.*, "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Trans. Veh. Tech.*, vol. 66, no. 11, pp. 10 433–10 445, Nov. 2017.
[24] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Dec. 2018.
[25] 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. 10 190–10 203, Nov. 2018.
[26] Y. Sun, M. Peng, and S. Mao, "Deep reinforcement learning-based mode selection and resource management for green fog radio access networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1960–1971, Apr. 2019.
[27] Y. He, N. Zhao, and H. Yin, "Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach," *IEEE Trans. Veh. Tech.*, vol. 67, no. 1, pp. 44–55, Jan. 2018.
[28] N. Cheng *et al.*, "Space/aerial-assisted computing offloading for IoT applications: A learning-based approach," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 5, pp. 1117–1129, May 2019.
[29] Y. Li, C. Tang, K. Li, X. He, S. Peeta, and Y. Wang, "Consensus-based cooperative control for multi-platoon under the connected vehicles environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2220–2229, Jun. 2019.
[30] Q. Ye, W. Zhuang, X. Li, and J. Rao, "End-to-end delay modeling for embedded vnf chains in 5 g core networks," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 692–704, Feb. 2019.
[31] L. Kuang, X. Chen, C. Jiang, H. Zhang, and S. Wu, "Radio resource management in future terrestrial-satellite communication networks," *IEEE Wireless Commun.*, vol. 24, no. 5, pp. 81–87, Oct. 2017.
[32] L. Li, K. Ota, and M. Dong, "Humanlike driving: Empirical decision-making system for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 67, no. 8, pp. 6814–6823, Aug. 2018.
[33] Q. Ye, W. Zhuang, S. Zhang, A. Jin, X. Shen, and X. Li, "Dynamic radio resource slicing for a two-tier heterogeneous wireless network," *IEEE Trans. Veh. Tech.*, vol. 67, no. 10, pp. 9896–9910, Oct. 2018.
[34] M. L. Puterman, *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.
[35] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surv. Tut.*, vol. 21, no. 4, pp. 3133–3174, Oct.-Dec. 2019.
[36] 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.
[37] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[38] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.
[39] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, and M. Lee, "Natural actor-critic algorithms," *Automatica*, vol. 45, no. 11, pp. 2471–2482, 2009.
[40] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Proc. Advances Neural Inf. Process. Syst.*, 2016, pp. 3675–3683.
[41] D. Zeng, L. Gu, S. Pan, J. Cai, and S. Guo, "Resource management at the network edge: A deep reinforcement learning approach," *IEEE Netw.*, vol. 33, no. 3, pp. 26–33, May/Jun. 2019.
[42] "PTV vissim," [Online]. Available: https://en.wikipedia.org/wiki/PTV_VISSIM/, Accessed: Sep. 19, 2019.
[43] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of Vehicles," in *Proc. IEEE*, vol. 108, no. 10, pp. 246–261, Feb. 2019.
[44] Z. Su, Y. Hui, and T. H. Luan, "Distributed task allocation to enable collaborative autonomous driving with network softwarization," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2175–2189, Oct. 2018.

Haixia Peng (Student Member, IEEE) received the M.S. and Ph.D degrees in electronics and communication engineering and computer Science from Northeastern University, Shenyang, China, in 2013 and 2017, respectively. She is currently working toward the Ph.D degree with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. Her current research focuses on autonomous vehicular network, resource management, and reinforcement learning. She served as a TPC member in IEEE VTC-fall 2016 and 2017, IEEE ICCEREC 2018, IEEE Globecom 2016–2020, and IEEE ICC 2017–2020 conferences.

Xuemin (Sherman) Shen (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Rutgers University, New Brunswick, NJ, USA, in 1990. He is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research focuses on network resource management, wireless network security, Internet of Things, 5G and beyond, and vehicular ad hoc and sensor networks. He is a Registered Professional Engineer of Ontario, Canada, Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, a Chinese Academy of Engineering Foreign Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

Dr. Shen was the recipient of R.A. Fessenden Award in 2019 from IEEE, Canada, James Evans Avant Garde Award in 2018 from the IEEE Vehicular Technology Society, Joseph LoCicero Award in 2015 and Education Award in 2017 from the IEEE Communications Society. He has also received the Excellent Graduate Supervision Award in 2006 and Outstanding Performance Award five times from the University of Waterloo and the Premier's Research Excellence Award (PREA) in 2003 from the Province of Ontario, Canada. He served as the Technical Program Committee Chair/Co-Chair for the IEEE Globecom 2016, the IEEE Infocom 2014, the IEEE VTC 2010 Fall, the IEEE Globecom 2007, the Symposia Chair for the IEEE ICC 2010, the Tutorial Chair for the IEEE VTC 2011 Spring, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He was an Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL and the Vice President on Publications of the IEEE Communications Society.