# Deep Reinforcement Learning for Vehicular Edge Computing: An Intelligent Offloading System

ZHAOLONG NING, PEIRAN DONG, and XIAOJIE WANG, Dalian University of Technology, China
JOEL J. P. C. RODRIGUES, National Institute of Telecommunications (Inatel), Santa Rita do Sapucaí–MG, Brazil; Instituto de Telecomunicações, Portugal; Federal University of Piauí, Teresina–PI, Brazil
FENG XIA, Dalian University of Technology, China

The development of smart vehicles brings drivers and passengers a comfortable and safe environment. Various emerging applications are promising to enrich users' traveling experiences and daily life. However, how to execute computing-intensive applications on resource-constrained vehicles still faces huge challenges. In this article, we construct an intelligent offloading system for vehicular edge computing by leveraging deep reinforcement learning. First, both the communication and computation states are modelled by finite Markov chains. Moreover, the task scheduling and resource allocation strategy is formulated as a joint optimization problem to maximize users' Quality of Experience (QoE). Due to its complexity, the original problem is further divided into two sub-optimization problems. A two-sided matching scheme and a deep reinforcement learning approach are developed to schedule offloading requests and allocate network resources, respectively. Performance evaluations illustrate the effectiveness and superiority of our constructed system.

CCS Concepts: • **Networks** → *Network management*;

This work has been partially supported by the National Natural Science Foundation of China under Grant 61572106 and 61971084, the China Postdoctoral Science Foundation under Grant 2018T110210, the State Key Laboratory of Integrated Services Networks, Xidian University (ISN20-01), the State Key Laboratory for Novel Software Technology, Nanjing University under Grant KFKT2018B04, Dalian Science and Technology Innovation Fund under Grant 2018J12GX048, National Natural Science Foundation of Chongqing under Grant cstc2019jcyj-msxmX0208. National Funding from the FCT—*Fundação para a Ciência e a Tecnologia*—through the UID/EEA/50008/2019 Project; by RNP, with resources from MCTIC, Grant No. 01250.075413/2018-04, under the *Centro de Referência em Radiocomunicações*—CRR project of the *Instituto Nacional de Telecomunicações* (Inatel), Brazil; and by Brazilian National Council for Research and Development (CNPq) via Grant No. 309335/2017-5.
Authors' addresses: Z. Ning, Dalian University of Technology, Key laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian, 116620, China, State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210008, China and Chongqing Key Laboratory of Mobile Communications Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; email: zhaolongning@dlut.edu.cn; P. Dong and X. Wang, Dalian University of Technology, School of Software, Dalian, 116620, China; emails: peiran_dong@outlook.com, wangxj1988@mail.dlut.edu.cn; J. J. P. C. Rodrigues (corresponding author), National Institute of Telecommunications (Inatel), Santa Rita do Sapucaí – MG, Brazil; Instituto de Telecomunicações, Portugal; Federal University of Piauí, Teresina – PI, Brazil; email: joeljr@ieee.org; F. Xia (corresponding author), Dalian University of Technology, Key laboratory for Ubiquitous Network and Service Software of Liaoning Province, School of Software, Dalian, 116620, China and School of Science, Engineering and Information Technology, Federation University Australia, Australia; email: f.xia@acm.org.

2157-6904/2019/10-ART60 $15.00
https://doi.org/10.1145/3317572

Additional Key Words and Phrases: Vehicular system, intelligent offloading, deep reinforcement learning, edge computing

**ACM Reference format:**
Zhaolong Ning, Peiran Dong, Xiaojie Wang, Joel J. P. C. Rodrigues, and Feng Xia. 2019. Deep Reinforcement Learning for Vehicular Edge Computing: An Intelligent Offloading System. *ACM Trans. Intell. Syst. Technol.* 10, 6, Article 60 (October 2019), 24 pages.
https://doi.org/10.1145/3317572

## 1 INTRODUCTION

With the rapid development of ubiquitous systems and smart vehicles, artificial intelligence–based vehicular networks [37], known as a subset of Cyber Physical Systems (CPS)[22], have drawn increased attention. Many researchers all over the world have been working on novel automotive applications to create a more comfortable and safer driving environment. However, how to execute these computing-intensive applications on vehicles still faces huge challenges, e.g., how to enable real-time feedbacks between vehicles and network servers based on Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication modes; how to provide efficient computing capabilities for resource-constraint applications and reasonable resource allocations for vehicles and infrastructures.

Initially, researchers propose the paradigm of Mobile Cloud Computing (MCC) [40], accumulating rich computing and storage resources into cloud servers. Despite its powerful computing capability, it is still difficult to satisfy the real-time response requirements of vehicular applications. Therefore, Mobile Edge Computing (MEC) is a promising alternative, the nodes of which are in proximity of users. Compared with MCC, MEC can greatly reduce communication latency, which scales exponentially with the increase of routing hops in vehicular networks [8]. Moreover, the diversity of MEC nodes significantly exploits potential computing resources over the network, which also alleviates the workload of the central Base Station (BS) [4]. It should be noted that more than RoadSide Units (RSUs)—any entities with the capability of computing, caching, or networking—can be the platform of MEC. Since the constraint resource limits the capability of MEC nodes, it is envisioned that the performance of traditional time- and energy-consuming networking methods [3, 5] can drop abruptly in vehicular networks. Therefore, it is urgent to develop an efficient solution for the MEC environment.

Deep Reinforcement Learning (DRL) is a prospective technology to replace traditional methods. Recently, machine learning has achieved remarkable achievements in many fields, such as image processing, pattern recognition, and natural language processing. It is also involved in computing-intensive applications, including autopilot and real-time navigation through V2V or V2I. However, machine learning in MEC-enabled vehicular networks (i.e., vehicular edge computing) is still in its infancy. A few researches attempt to leverage deep learning and convolutional neural networks to predict traffic flows. However, DRL is rarely considered. To construct an intelligent offloading system for vehicular edge computing and make it work well, there are three main challenges:

(1) Although DRL has achieved great success in Atari games and Go [17], its application in vehicular networks is almost nonexistent. This is because vehicular networks are highly dynamic, and the constraint in offloading systems is more implicit, flexible, and diverse than the explicit rules of chess.
(2) Both DRL and traditional vehicular networks are investigated based on a sequence of captured images. However, no sequential images exist in our intelligent offloading system. How to migrate DRL to vehicular networks without images is rather challenging.

(3) No matter playing chess or Atari games, there is usually one "agent" in a DRL model. Since multiple vehicles participate in the intelligent offloading systems, it is difficult to construct a suitable environment and build the corresponding DRL model.

In this article, a DRL method is integrated with vehicular edge computing to solve the computation offloading problem, where we jointly study the optimization of task scheduling and resource allocation in vehicular networks. First, we model the architecture of communication and edge computing, respectively. The channel state and the computation capability are finite continuous values varying with time, where the state at the next moment is only related to the previous moment. For ease of analysis, they are discretized and quantized into several levels and further modelled as Finite-State Markov Chains (FSMC) [30]. Moreover, we model the mobility of vehicles by discrete random jumps. The number of contacts between RSUs and vehicles follows a Poisson distribution, where the parameter represents the mobility intensity. Then, the joint optimization problem of traffic scheduling and resource allocation in vehicular networks is formulated. Since the formulated problem is constrained by different factors, and variables are coupled with each other, we divide the original problem into two sub-optimization problems. For the first one, we decide the priority of multiple vehicles by designing a utility function, reflecting the QoE level of users. Then, the second sub-problem is formulated as a Reinforcement Learning (RL) problem, where we illustrate four key elements: agents, system states, actions, and rewards, respectively. The main contributions are summarized as follows:

(1) Based on the finite-state Markov chains, we jointly study the task scheduling and resource allocation in vehicular networks and construct an intelligent offloading system. Specifically, DRL is integrated with vehicular edge computing. We formulate an optimization problem to maximize the QoE of users, while taking both energy consumption and execution delay into consideration.
(2) Due to the high complexity of the formulated problem, it is divided into two sub-optimization problems. In the first stage, we schedule tasks of multiple vehicles. A utility function is defined to quantize the level of QoE. A two-sided matching scheme is proposed to solve the formulated sub-problem, with the purpose of maximizing the total utilities.
(3) The decision making of resource allocation is resolved by leveraging a DRL algorithm in the second stage. A deep Q network is improved by applying dropout regularization and double deep Q networks to deal with the defect of overestimation. We identify the system state, the action, and the reward function in our DRL model, whose target is to maximize the cumulative reward through obtaining the optimal policy.
(4) Performance evaluations illustrate the effectiveness of our designed system, i.e., the proposed two-sided matching scheme can approximate the performance of an exhaustive searching scheme efficiently, and the improved DRL algorithm is superior to other methods with various system parameters.

The rest of this article is organized as follows: In Section 2, we review the related work. We illustrate the system model in Section 3. In Section 4, we formulate the optimization problem. Section 5 illustrates the constructed intelligent offloading system. Two algorithms are specified in Section 6 to solve the two sub-optimization problems, respectively. Performance evaluations are provided in Section 7, and Section 8 concludes our work.

## 2 RELATED WORK

Recently, deep learning and edge computing–based CPS have attracted the attention of many researchers. We review three categories, including deep learning–based CPS, MEC-based CPS, and deep learning for MEC. Since CPS contains various network systems, we choose vehicular networks as an example to introduce recent research progress.

### 2.1 Deep Learning–based CPS

Deep learning is generally utilized for pattern recognition and traffic prediction. The authors in Reference [18] utilize unsupervised learning to cluster the urban smart card data, involving the station-oriented and passenger-oriented views. The real-world data of the metropolitan area in Rennes, France, are collected to find the distribution of traffic flow and the similarity among passengers. A cooperative neural network is tailored to construct a structural lane detection system with the captured images in a real-world traffic scene [13]. First, a deep convolutional network is devised to detect traffic signs and their geometric attributes. Second, recurrent neural networks dealing with signal spatial distribution are difficult to be explicitly recognized. After millions of traffic images are captured, how to efficiently retrieve these large amounts of data is a challenging issue. A supervised hash coding scheme is designed to generate high-quality binary codes [35]. Convolutional neural networks are implemented to analyze the feature representation of images. The quantized loss function compels that similar images are encoded by similar codes. Authors in Reference [1] adopt Long Short Term Memory (LSTM)–based recurrent neural networks to predict the real-time taxi demand. Different from traditional prediction, they do not forecast a deterministic value, but leverage mixture density networks to predict the probability distribution of taxi demands. A new neural network training method is applied in Reference [16]. It takes the correlations between space and time into consideration to predict traffic flow. The key idea of the training method is using the greedy layerwise unsupervised learning scheme to preprocess the deep neural network layer-by-layer, which reduces training time significantly.

### 2.2 MEC-based CPS

A tensor-based cooperative mobile computing system is constructed in Reference [28], where cloud servers are in charge of processing large-scale and long-term data, such as global decision making. MEC servers process small-scale and short-term data, such as real-time responses. A CPS stream data processing pattern is proposed in Reference [36], which performs network services in a distributed manner through clustering edge devices. By investigating MEC-based CPS, authors in Reference [7] jointly study three cost-efficiency problems, including base station association, task scheduling, and virtual machine deployment. To minimize the overall cost, they formulate the optimization problem into a mixed-integer non-linear programming problem and linearize it into a mixed integer linear programming problem. Offloading in MEC-based vehicular networks has been well investigated in the past few years. In Reference [20], a non-orthogonal multiple access–based offloading scheme is designed for vehicular networks. Technologies of spectrum reuse and efficient computing are leveraged to increase the transmission rate and the offloading efficiency. Authors in Reference [27] put forward a three-layer real-time traffic management system in vehicular networks. Both parked and moving vehicles are taken into account, and moving vehicles are modelled as an $M/M/1$ queue. The optimization target is to minimize the average response time.

### 2.3 Deep Learning for MEC

The integration of deep learning and MEC has become a hot research area. A deep learning approach is developed in Reference [15] to find the optimal auction for computation resources of edge computing in blockchain networks. It devises a monotone transform function to anonymize bid prices. The softmax function and relu function are applied to compute the probability of winning and the price for network resources, respectively. The residual battery capability and renewable energy are considered in Reference [34]. It proposes a post-decision state–based RL algorithm,

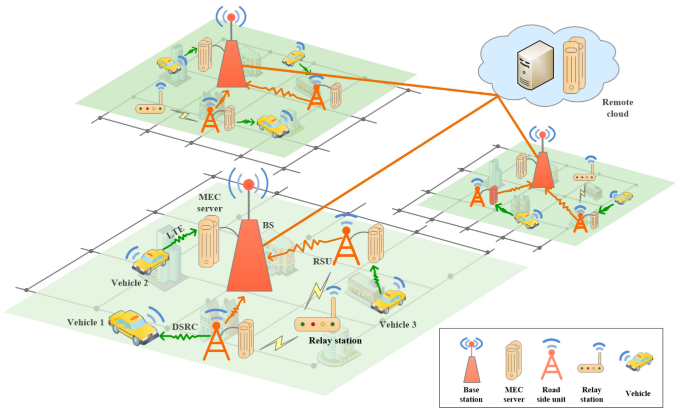

Fig. 1. The architecture of MEC-based vehicular networks.

which divides the power consumption of computation and green energy charging into two phases and makes an auto-scaling allocation in the middle state. An RL-based evolved NodeB selection algorithm is developed by jointly considering the blocking probability, communication rate, and load balancing [14]. To decrease the number of duplicated contents in networks, a DRL method for caching in smart cities is designed in Reference [10]. The agent in the system collects the status from MEC servers and base stations and learns to choose the optimal action to get the best policy for resource arrangement. Authors in Reference [9] study the device-to-device communications and social properties in network communication and design a trust-based social network framework for networking and computing.

To the best of our knowledge, there are few researches on DRL for vehicular edge computing. Although researches in References [23] and [11] study the DRL-based networking and caching, our work focuses on the task scheduling and resource allocation of computation offloading in vehicular networks, which achieves a good trade-off between the QoE of users and the profit of servers.

## 3 SYSTEM MODEL

As shown in Figure 1, a city-wide vehicular network can be divided into several zones according to streets or other criteria. In each zone, there is a central BS with abundant computation resources. Vehicles can communicate with the BS through Long Term Evolution (LTE). Similar with References [19, 41], we consider that cellular networks can fully cover urban areas. In addition, several RSUs are deployed along roads within each zone. Note that RSUs are equipped with MEC servers. Vehicles upload tasks to RSUs based on Dedicated Short Range Communications (DSRC), guaranteeing high quality of communications in a short range especially for one-hop communication. RSUs are connected to each other through relay nodes [31] and obtain global information of vehicular offloading tasks through the relay station. If RSUs merely communicate with vehicles and BS, then BS has to take the role of traffic management center, resulting in a great burden of communication and processing. For ease of description, we consider one zone in the urban area, and the model can be easily extended to other zones.

We consider a zone-based vehicular network, including one BS, $K$ RSUs, and $U$ vehicles. With the computation resources of MEC servers, RSUs can share the computing tasks of the BS, which helps alleviate the overload of the BS and reduce the communication latency. Let $\mathcal{K} = \{0, 1, \ldots, K\}$ and $\mathcal{U} = \{1, \ldots, U\}$ be the sets of RSUs and vehicles, respectively. We assume that vehicles only upload tasks to RSUs through one-hop DSRC when they are in the communication range of RSUs. In contrast, vehicles outside the communication range of RSUs can upload their tasks to the BS.

We assume that each vehicle spends a period of time to upload its offloading tasks. Let $\mathcal{T}$ be the duration time of communication, which can be divided into $T_i$ time slots. Vehicles can offload their tasks to the BS or RSUs. Let $a_{i,k}(t)$ denote the connection relationship among vehicles, RSUs and the BS in time slot $t$, where $a_{i,k}(t) = 1$ means that vehicle $i$ connects to RSU $k$ at time $t$; otherwise $a_{i,k}(t) = 0$. Note that $k = 0$ indicates the BS is selected for connection. Each vehicle can only connect to either one RSU or the BS during one time slot, thus the following constraint should be satisfied:

$$\sum_{k \in \mathcal{K}} a_{i,k}(t) = 1, \forall i \in \mathcal{U}. \tag{1}$$

The whole network is operated by a central controller, such as traditional mobile network operators. Network operators provide services of routing, caching and computing, and profit from them. Therefore, there is a trade-off between users' (i.e., vehicles) QoE and the profit of operators. The purpose of our research is to find the optimal scheduling strategy to maximize the QoE of vehicles while ensuring the revenue of network operators.

The mobility of vehicles is illustrated as follows. Although vehicles move randomly and their position changes frequently, their positions have relatively small changes during a short period of time. In discrete time periods, the movement of vehicles can be viewed as a discrete image that jumps from one position to the next. Thus, we model the mobility of vehicles by discrete random jumps, and the corresponding intensity is characterized by the average sojourn time among jumps. Let $M_{i,k}$ denote the number of contacts between RSU $k$ and vehicle $i$ within communication time $T_i$, which follows a Poisson distribution with parameter $\lambda_{i,k}$. Herein, $\lambda_{i,k}$ can be viewed as the connect frequency, accounting for the mobility intensity.

Whenever a vehicle enters or exits the wireless coverage of an RSU, a message is sent to the RSU to force that RSU to update its management list, and the RSU responds to the vehicle by transmitting a message containing the information of available computing resources. Note that the RSU may send a message for denial of service when the MEC server is overloaded. Total communication time $T_i$ can be divided into two parts: RSU connection time $T_i^R$ and BS connection time $T_i^B$.

## 3.1 Communication Model

Relying on BS, traditional network services will inevitably lead to high delays as the number of tasks increases. To overcome this communication bottleneck, MEC is leveraged to reduce the round-trip time of communication between vehicles and servers. In addition, information sharing between RSUs can also reduce the communication burden of the BS. Considering that the wireless connections between vehicles and BS/RSUs are time-varying and memoryless, we model the channel state as FSMC. There are several parameters that can determine the communication rate, where the channel gain reflects the channel quality. Let variable $\gamma_i^k$ denote the channel gain of the wireless link between vehicle $i$ and RSU $k$. Actually, the realistic wireless channel gain is a continuous variable. In our model, the value range of $\gamma_i^k$ is discretized and quantized into $L$ levels. Let $\mathcal{L} = \{\Upsilon_0, \ldots, \Upsilon_{L-1}\}$ denote the state space of Markov chain: $\Upsilon_0$, if $\gamma_0^* \leqslant \gamma_i^k < \gamma_1^*$; $\Upsilon_1$, if $\gamma_1^* \leqslant \gamma_i^k < \gamma_2^*$; $\ldots$ ; $\Upsilon_{L-1}$, if $\gamma_i^k \geqslant \gamma_{L-1}^*$. Furthermore, the realization of channel gain $\gamma_i^k$ at time slot $t$ is denoted by $\Gamma_i^k(t)$. We define the transition probability that $\Gamma_i^k(t)$ changes from one state $g_s$ to another $h_s$ as $\psi_{g_s,h_s}(t)$. Here $g_s$ and $h_s$ are two states of $\gamma_i^k$, which belong to $\mathcal{L}$. Therefore, we can obtain the following $L \times L$ channel state transition probability matrix for the communication between vehicle $i$ and RSU $k$:

$$\Psi_i^k(t) = \left[\psi_{g_s,h_s}(t)\right]_{L \times L}, \tag{2}$$

where $\psi_{g_s,h_s}(t) = \Pr\left(\Gamma_i^k(t+1) = h_s \mid \Gamma_i^k(t) = g_s\right)$, and $g_s, h_s \in \mathcal{L}$.

The limited spectrum resource has not been fully and efficiently utilized in Orthogonal Multiple Access (OMA) [21]. To handle this problem, NOMA has been proposed as a promising solution for 5G wireless networks, which is promising to combine the LTE-based Vehicle-to-Everything (V2X) service and cellular network architectures and reduce the end-to-end latency [6]. NOMA allows vehicles to access BS non-orthogonally. That is to say, multiple vehicles can upload data concurrently on the same channel, which improves the spectrum efficiency.

Since each RSU only accesses one vehicle at a time, and BS can serve multiple vehicles at the same time, we consider that the Orthogonal Frequency Division Multiple Access (OFDMA) technology is utilized for links between vehicles and RSUs, and the Non-Orthogonal Multiple Access (NOMA) technology is leveraged for the link between vehicles and the BS. Thus, there is no interference when vehicles communicate with RSUs. The achievable instant data transmission rate at time slot $t$ can be calculated by:

$$v_{i,k}(t) = b_{i,k}(t)\log_2\left(1 + \frac{p_{i,k}(t)\,(\Gamma_i^k(t))^2}{\sigma^2}\right), \tag{3}$$

where $b_{i,k}(t)$ represents the orthogonally allocated bandwidth from RSU $k$ to vehicle $i$, $k \in \mathcal{K}$ and $i \in \mathcal{U}$. Let $\mathcal{B}$ denote the whole available bandwidth in the zone. Thus, $\sum_{i\in\mathcal{U},k\in\mathcal{K}} b_{i,k}(t) \leqslant \mathcal{B}$. Variable $p_{i,k}(t)$ denotes the transmission power of vehicle $i$, and $\sigma^2$ is Gaussian white noise power.

To cope with the interference caused by channel sharing among multiple vehicles, the Successive Interference Cancellation (SIC) [24] can be adopted at the end-receivers (i.e., BS). Therefore, the received signal of BS from vehicle $i$ at time slot $t$ can be computed by:

$$y_{i,0}(t) = \sqrt{p_{i,0}(t)}\Gamma_i^0(t)\,x_{i,0}(t) + \sum_{n\neq i,\,n\in\mathcal{U}} \sqrt{p_{n,0}(t)}\Gamma_i^0(t)\,x_{n,0}(t) + \sigma, \tag{4}$$

where $x$ and $y$ represent the sent signal of vehicles and received signal of BS, respectively. The first part of $y_{i,0}$ in Equation (4) is the effective signal from the target vehicle; the second part is the interference signal from other vehicles sharing this channel; and the third part is the noise.

After receiving the signal, BS performs the SIC decoding scheme to reduce the interference from other vehicles based on the decreasing order of channel gains [38]. For example, there are two vehicles $u_i, u_j \in \mathcal{U}$. If $\gamma_i^0 > \gamma_j^0$, then BS treats $u_j$ as the interference to $u_i$, and cancels $u_i$ after decoding it. When BS decodes $u_j$, there is no interference. That is, for vehicle $i$, the interference signal is a signal set that those vehicles with a smaller equivalent channel gain over it. We consider that $N$ vehicles share the same channel in the descending order of their channel gains: $\gamma_1^0 \geqslant \gamma_2^0 \geqslant \cdots \geqslant \gamma_N^0$. Then the interference signals of vehicle $n$ can be calculated by:

$$I_n = \sum_{i=n+1}^{N} p_{i,0}\left(\gamma_i^0\right)^2. \tag{5}$$

We can obtain the data transmission rate between vehicle $i$ and BS as follows:

$$v_{i,0}(t) = b_{i,0}(t)\log_2\left(1 + \frac{p_{i,0}(t)\,(\Gamma_i^k(t))^2}{\sigma^2 + I_i}\right). \tag{6}$$

Finally the communication rate of vehicle $i$ can be obtained by:

$$R_{i,k}^{\text{comm}}(t) = a_{i,k}(t)\,v_{i,k}(t), \forall i \in \mathcal{U}, k \in \mathcal{K}. \tag{7}$$

The total communication rate of all vehicles accessed to RSU $k$ cannot exceed its capacity, i.e.,

$$\sum_{i \in \mathcal{U}} R_{i,k}^{\text{comm}}(t) \leqslant Z_k, \forall k \in \mathcal{K}. \tag{8}$$

Similarly, the total communication rate of vehicles in a zone cannot exceed the total capacity, thus the following constraint should be met:

$$\sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{U}} R_{i,k}^{\text{comm}}(t) \leqslant Z. \tag{9}$$

## 3.2 Computation Model

In this article, we mainly focus on those applications whose offloading tasks can be divided into several parts and processed on different platforms, such as online game, augmented reality, and natural language processing. We define the uploaded computation task of vehicle $i$ as $\xi_i = \{d_i, c_i\}$, where $d_i$ is the data size of the computation task, and $c_i$ is the required number of CPU cycles to accomplish the task. After that, RSU or BS sends the computation result back to vehicle $i$. Since MEC servers are located in the proximity of RSUs, the transmission time between them can be ignored [26]. Moreover, the output data size of task offloading is often much less than that of input data size. Thus, the transmission delay of backhaul (i.e., downlink) link is also ignored [2].

We define the computation capability (i.e., CPU cycles per second) of RSUs and BS $k$ allocated to vehicle $i$ as $f_{i,k}$. We assume that RSUs work in a preemptive manner, which means that they process communication requests of vehicles sequentially (i.e., $\sum_{i \in \mathcal{U}} a_{i,k} \leqslant 1, \forall k \in \mathcal{K}$). Despite this, multiple vehicles may share one MEC server when they move in the wireless coverage of the same RSU. Due to the resource constraint of MEC servers, it is impossible to guarantee that all vehicles are provided with full and sufficient computation capabilities. Thus, $f_{i,k}$ can be modelled as a random variable and divided into $N$ levels: $\varepsilon = \{\varepsilon_0, \varepsilon_1, \ldots, \varepsilon_{N-1}\}$, where $N$ denotes the number of available computation capability states. Let $F_{i,k}(t)$ be the instant computation capability at time slot $t$. Similar to channel gain $\gamma_i^k$, we model $f_{i,k}$ as FSMC. The computation capability transition probability matrix of instant computation capability $F_{i,k}(t)$ is presented as:

$$\Theta_{i,k}(t) = [\theta_{x_s,y_s}(t)]_{N \times N}, \tag{10}$$

where $\theta_{x_s,y_s}(t) = \Pr(F_{i,k}(t+1) = y_s \mid F_{i,k}(t) = x_s)$, and $x_s, y_s \in \varepsilon$.

The task execution time of computation task $\xi_i$ at RSU $k$ can be calculated by: $\Delta_{i,k} = c_i/f_{i,k}$. Thus, the computation rate (i.e., bits per second) can be obtained by:

$$r_{i,k}^{\text{comp}} = \frac{d_i}{\Delta_{i,k}} = \frac{f_{i,k} d_i}{c_i}. \tag{11}$$

The instant computation rate of RSU $k$ for vehicle $i$ at time slot $t$ is expressed as:

$$R_{i,k}^{\text{comp}}(t) = a_{i,k}(t) r_{i,k}^{\text{comp}}(t) = a_{i,k}(t) \frac{F_{i,k}(t) d_i}{c_i}, \tag{12}$$

where the data size of concurrent computation on the MEC server cannot exceed its computation capacity. Therefore, the following constraint should be satisfied:

$$\sum_{i \in \mathcal{U}} a_{i,k}(t) d_i \leqslant \mathcal{D}_k, \forall k \in \mathcal{K}, \tag{13}$$

where $\mathcal{D}_k$ denotes the maximum data size that can be simultaneously processed on the MEC server.

## 4 PROBLEM FORMULATION

We formulate the optimization problem in this section. Since the matching problem over a period of time is an NP-hard problem, the original problem is divided into two sub-optimization problems.

### 4.1 Optimization Objective

When a moving vehicle generates an offloading task, it detects available RSUs and BS around and records the task-related information as well as available computing resource information. Then, it sends this event to the nearby BS over LTE. After BS receives details of this event, it will immediately perform resource allocation and broadcast the obtained schedule through RSUs in a zone. Traditional BS is in charge of all tasks execution, which may cause a large traffic delay and excessive energy consumption. Not only the user's QoE cannot be guaranteed, but also the network operator's profit is very low. To maximize the QoE of vehicles while guaranteeing the revenue of network operators (i.e., RSUs and BS in this article), a cooperative offloading network system is constructed. The instant QoE of vehicles at time slot $t$ can be obtained by:

$$\mathcal{R}_{i,k}(t) = \mathcal{R}_{i,k}^{\text{comm}}(t) + \mathcal{R}_{i,k}^{\text{comp}}(t). \tag{14}$$

The joint optimization problem of Traffic Scheduling and Resource Allocation (TSRA) in our system is formulated as follows:

$$\max_{a_{i,k}(t)} \mathcal{R} = \sum_{i\in\mathcal{U}} \sum_{k\in\mathcal{K}} \sum_{t=1}^{M_{i,k}} \mathcal{R}_{i,k}(t), \tag{15}$$

$$\text{s.t.} = \begin{cases} a_{i,k}(t) \in \{0,1\}, \forall i \in \mathcal{U}, k \in \mathcal{K}, \\ Equations\ (1)\ ,(8)\ ,(9)\ ,(13)\ . \end{cases}$$

Since the formulated TSRA problem is constrained by different factors, coupled variables make the optimization problem difficult to resolve. To address this issue and make a trade-off between QoE of users and the revenue of network operators, we divide the original TSRA problem into two sub-optimization problems. In the first stage, we decide the priority of multiple vehicles by designing a utility function. Then, we leverage the improved Deep Q-Network (DQN) algorithm to obtain the scheduling results and map each user to the corresponding RSU or BS.

### 4.2 Offloading Task Scheduling

It is common for multiple vehicles to select the same RSU at one time slot, leading to constraint (1) not being satisfied. Therefore, the first sub-optimization problem takes QoE of users into account and attempts to find a reasonable scheduling list of all vehicles without conflicting with each other.

We define a flexible utility function as the user satisfaction level. The value of the utility function depends on the task, communication channel state, and distance between the vehicle and the RSU. There are four parameters in the utility function: priority, urgency, channel gain, and distance. Channel gain $\gamma_i^k$ has been introduced in Section III-C, which reflects the communication channel state. Distance $\triangle d_{i,k}$ is the Euclidean distance between the vehicle and the RSU or BS. Priority $\pi(p)$ sets the upper bound of the utility function, where priority level $p \in \{critical, high, medium, low\}$. If an offloading request of a vehicle is responded to immediately, then its utility function takes the upper bound of the corresponding task priority level, denoted by $\pi(p)$. Otherwise, the value of its utility function will decrease over time. Urgency $\rho(r)$ is defined to model the exponential decay rate of vehicles' utility functions with the increase of response delay, where urgency level $r \in \{extreme, high, medium, low\}$. The higher the urgency level of the task is, the faster the utility function decreases as the delay increases. The value of utility functions with different urgency levels is illustrated in Reference [12] when the task priority level is fixed.

As illustrated in Reference [29], it is critical to incorporate the sigmoidal behavior into the vehicles' utility function for resource allocation. We adopt the sigmoid-like function to model the utility of vehicle $i \in \mathcal{U}$ with parameters priority $\pi^{(i)}(p)$, urgency $\rho^{(i)}(r)$, channel gain $\gamma_i^k$, and distance $\triangle d_{i,k}$ as follows:

$$\mathcal{Y}_{i,k} = \frac{\pi^{(i)}(p)}{\triangle d_{i,k} + \exp(-\rho^{(i)}(r)(\gamma_i^k - b_{i,k}))}, \tag{16}$$

where constant parameter $b_{i,k}$ is used to fine-tune the utility function. Parameter $\pi^{(i)}(p)$ is similar to the traditional weight factor, and $\rho^{(i)}(r)$ controls the steepness of $\mathcal{Y}_{i,k}$. The larger $\rho^{(i)}(r)$ is, the faster $\mathcal{Y}_{i,k}$ increases with $\gamma_i^k$.

We aim to maximize the average value of vehicles' utilities. The optimization function is as follows:

$$\max \quad \frac{1}{|\mathcal{U}|} \sum_{i \in \mathcal{U}} a_{i,k} \mathcal{Y}_{i,k}, \tag{17}$$

$$\text{s.t.} = \begin{cases} p \in \{critical, high, medium, low\}, \\ r \in \{extreme, high, medium, low\}, \\ 0 < \triangle d_{i,k} \leqslant \Delta, \\ a_{i,k} \in \{0,1\}, \forall i \in \mathcal{U}, \\ \sum_{i \in \mathcal{U}} a_{i,k}(t) = 1, \forall k \in \mathcal{K}, \\ Equation\ (1), \end{cases}$$

where $\Delta$ indicates the radius of RSUs' wireless coverages. When vehicles independently select different RSUs, there are different utility values, and vehicles strive to maximize their utilities to ensure their own QoE. Due to the constraint that one RSU can only be accessed by one vehicle at one time, it may conflict among offloading decisions.

## 4.3 Deep Reinforcement Learning–based Offloading

In this subsection, we formulate the resource allocation optimization problem as a DRL process. After obtaining the service queue of vehicles by solving the task scheduling problem, we aim to maximize the overall QoE of vehicles by determining the offloading decision of vehicles. Since variables, the immense space of environment states, and system actions change dynamically over time, it is almost impossible to solve this complicated problem with traditional optimization methods. Thus, we take advantage of the recent progressive DQN to yield system actions for vehicles effectively and efficiently.

To reduce the round trip time, vehicles select the BS as the agent of DQN, responsible for interacting with the environment and making decisions. We assume that the computing state of the MEC server is updated in real time and shared among RSUs. The agent collects the status from MEC servers and vehicles. The mobility of vehicles is obtained in real time through broadcasting, as mentioned before. After that, the agent can make the offloading decision by constructing the system state and choosing the optimal action. Finally, all optimal actions can be broadcast to vehicles.

In the following, we identify the system state, actions, and the reward function in our DQN model:

(1) System State
The state of communication channel gain and available computing capabilities are determined by the realization of channel gain $\gamma_i^k$ and computation capability $f_{i,k}$, respectively.

Consequently, the composite state $\chi_i(t) \in R^{2\times K}$ can be expressed as follows:

$$\chi_i(t) = \quad [\Gamma_i^1(t), \Gamma_i^2(t), \ldots, \Gamma_i^K(t), F_{i,1}(t), F_{i,2}(t), \ldots, F_{i,K}(t)]. \tag{18}$$

(2) System Action
In a DQN, the agent is responsible to choose RSUs or BS to process the offloading task of vehicle $i$. The offloading assignment is defined as a vector of binary variable $a_{i,k}(t)$, which is presented by:

$$a_i(t) = [a_{i,1}(t), a_{i,2}(t), \ldots, a_{i,K}(t)]. \tag{19}$$

(3) Reward Function
We aim to maximize the comprehensive QoE of vehicles, including the lease cost of the spectrum bandwidth and the computing resource. Therefore, QoE $\mathcal{R}_i(t)$ is set as the reward of our system. In addition, network operators charge vehicles for task execution and virtual network accessing. Their unit prices are defined as $\phi_i$ per Mbps and $\tau_i$ per Mbps, respectively. However, operators need to pay for bandwidth leasing, defined as $\delta_k$ per Hz for RSU $k$. In addition, the energy consumption of task execution should be taken into consideration. The unit cost of computation for RSU $k$ is denoted by $\eta_k$ per Joule. For unification, $\varsigma_k$ is defined as the energy consumption of running one CPU cycle for RSU $k$, whose unit is Watts per Hz. What is more, the proportion of offloading tasks handled by RSUs is represented by $\varrho_i$. We assume that offloading tasks can be divided into several parts and processed by MEC servers and BS separately. When vehicles drive out of the communication range of RSUs before accomplishing the offloading task, BS can continue to process the remaining tasks. Finally, we define the reward function of vehicle $i$ as:

$$\begin{aligned}
\mathcal{R}_i(t) &= \sum_{k\in\mathcal{K}} \mathcal{R}_{i,k}^{\text{comm}}(t) + \sum_{k\in\mathcal{K}} \mathcal{R}_{i,k}^{\text{comp}}(t) \\
&= \sum_{k=1}^{K} \left(\tau_i R_{i,k}^{\text{comm}} - \delta_k b_{i,k}(t)\right) + \left(\tau_i R_{i,0}^{\text{comm}} - \delta_k b_{i,0}(t)\right) \\
&\quad + \sum_{k=1}^{K} \left(\varrho_i \phi_i R_{i,k}^{\text{comp}} - \eta_k c_{i,k} \varsigma_k\right) + \left((1-\varrho_i)\phi_i R_{i,0}^{\text{comp}} - \eta_0 c_{i,0} \varsigma_0\right) \\
&= \sum_{k=1}^{K} \left(\tau_i b_{i,k}(t) \log_2\left(1 + \frac{p_{i,k}(t)(\Gamma_i^k(t))^2}{\sigma^2}\right) - \delta_k b_{i,k}(t)\right) \\
&\quad + \left(\tau_i b_{i,0}(t) \log_2\left(1 + \frac{p_{i,0}(t)(\Gamma_i^k(t))^2}{\sigma^2 + I_i}\right) - \delta_k b_{i,k}(t)\right) \\
&\quad + \sum_{k=1}^{K} a_{i,k}(t)\varrho_i \left(\phi_i \frac{F_{i,k}(t) d_i}{c_i} - \eta_k c_{i,k} \varsigma_k\right) + a_{i,0}(t)(1-\varrho_i)\left(\phi_i \frac{F_{i,0}(t) d_i}{c_i} - \eta_0 c_{i,0} \varsigma_0\right).
\end{aligned} \tag{20}$$

The agent can obtain immediate reward $\mathcal{R}_i(t)$ by performing the chosen action $a_{i,k}(t)$ at time slot $t$. The goal of DQN is to maximize the cumulative reward through obtaining the optimal policy. Thus, the optimization problem can be formulated as follows:

$$\mathcal{R}_i = \max_{a_{i,k}(t)} \sum_{t=0}^{T-1} \epsilon^t \mathcal{R}_{i,k}(t), \tag{21}$$

$$\text{s.t.} = \begin{cases} a_{i,k} \in \{0,1\}, \forall k \in \mathcal{K}, \\ 0 \leqslant \varrho_i \leqslant 1, \forall i \in \mathcal{U}, \\ Equation\,(8)\,,(9)\,,(13)\,, \end{cases}$$

where parameter $\epsilon$ is the weight of QoE in each time slot , $\epsilon \in (0,1]$.

## 5 INTEGRATED DEEP REINFORCEMENT LEARNING IN VEHICULAR NETWORKS

Before solving the formulated problem, this section briefly introduces the overview of RL and integrated DQN in vehicular networks.

### 5.1 Reinforcement Learning in Vehicular Networks

There are four key elements in RL: agents, environment states, rewards, and actions. For each episode, the agent chooses an action according to the current environment state and can get a reward after executing the chosen action. The environment states are typically modeled as a Markov Decision Process (MDP). As a result, an RL problem can be formulated as an optimal control problem in MDP. The space of environment states and actions are finite and explicit. The purpose of leveraging RL for the agent is to maximize the total reward by taking a series of actions when it interacts with the environment [39]. Since the computing and caching capabilities of vehicles are limited, it is unreasonable to deploy the computation-intensive deep neural networks applications at each vehicle. Therefore, BS plays the role of the agent in our model, which intends to gain profit (i.e., reward) through providing network services. Time-varying parameters channel state $\gamma_i^k$ and computation capability $f_{i,k}$ are environment states. The action space is available offloading servers. The BS chooses actions to schedule vehicles for profit maximization. It can be viewed as a typically RL problem.

Different from traditional machine learning methods (such as supervised learning), RL cannot learn from tagged historical data even when they are provided by an experienced supervisor. The trial-and-error search and the delayed reward are two remarkable features of RL [11]. The former is to make a trade-off between exploration and exploitation; the latter allows the agent to consider the accumulated rewards of vehicles. Generally, RL algorithms include Q learning, SARSA, and DQN.

### 5.2 Integrated DQN

Traditional DRL has many drawbacks in practical applications, such as slow convergence and overestimation. We adopt two methods to improve the DQN algorithm.

*5.2.1 Dropout Regularization.* Regularization decreases the number of network parameters and transforms deep and complex neural networks into linear and simple networks to reduce the variance of DQN. After that, the parameter matrix becomes a sparse matrix. In our model, dropout regularization reduces the complexity of parameter matrix $\theta$ by inactivating random parts of neurons and setting their weights to zero. Since neurons are randomly discarded in each layer, the trained neural network is much smaller than the normal network, and the over-fitting problem can be avoided. In addition, the whole network cannot be biased towards certain features (e.g., the weight values of features are very large), because every feature can be discarded in dropout regularization. Thus, the weight of each feature can be given a small value and is similar to L2 regularization ($||\theta||_2^2 = \sum_{j=1}^{\#neurons} \theta_j^2 = \theta^{\mathrm{T}}\theta$). The most significant parameter in dropout regularization is inactivation probability. For example, if we set the inactivation probability to 0.2, 20 percent neurons can be inactivated. Random inactivation will reduce the expected result value. When we perform dropout regularization on the hidden layer, the expected value of its output decreases by the inactivation probability, which affects future predictions. Thus, we divide the result by inactivation probability to keep its expected value unchanged. The specific implementation is as follows:

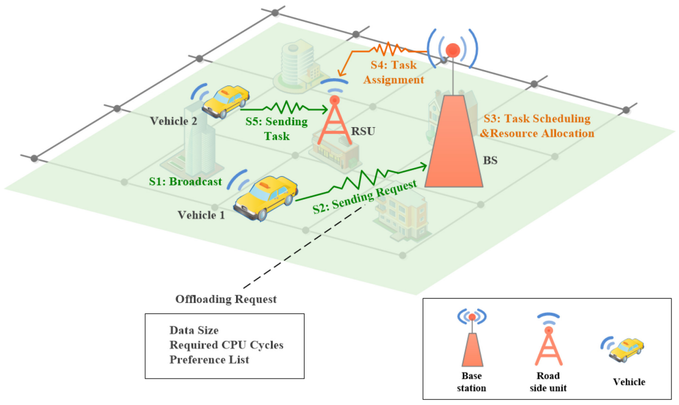

Fig. 2. The procedure of the intelligent offloading system.

$$\begin{aligned} keep.prob &= 0.8; \\ d3 &= \text{np.random.rand}(a3.shape[0], a3.shape[1]) < keep.prob; \\ a3 &= \text{np.multiply}(a3, d3); \\ a3 &= a3/keep.prob; \\ z4 &= \text{np.dot}(w4, a3) + b4. \end{aligned} \tag{22}$$

*5.2.2 Double DQN (DDQN).* Based on the framework of Q learning, DQN uses convolutional neural networks to represent the action-value function. However, DQN cannot overcome the inherent shortcomings of Q learning, i.e., overestimation.

To solve this problem, Hasselt [25] proposes DDQN to evaluate the selection of actions and the evaluation of actions by using different value functions. We compare the differences among Q learning, DQN, and DDQN from the formulation value function as follows:

$$\begin{aligned} Y_t^Q &= R_t + \gamma \max_a Q(S_{t+1}, a; \theta_t), \\ Y_t^{DQN} &= R_t + \gamma \max_{a'} Q(S_{t+1}, a'; \theta_t^-), \\ Y_t^{DDQN} &= R_t + \gamma Q\Big(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \theta_t); \theta_t^-\Big). \end{aligned} \tag{23}$$

In both Q learning and DQN, their action selection strategies are greedy. However, DDQN uses a neural network to evaluate the selection strategy and approximates the true value function.

## 6 A DEEP REINFORCEMENT LEARNING–BASED INTELLIGENT OFFLOADING SYSTEM

We have formulated the TSRA problem in Section 3 and divided it into two sub-optimization problems. In this section, we illustrate the intelligent offloading system, which contains two modules. The first one is task scheduling among multiple vehicles, and a two-sided matching algorithm is proposed to solve it. The second is resource allocation, and we implement the integrated DRL method to resolve it.

### 6.1 System Overview

The whole offloading procedure is presented in Figure 2. On the first step, all vehicles broadcast their position information and update their available RSUs list. After that, vehicles calculate util-

Table 1. System Variables

| Variable | Description |
|---|---|
| $V_i$ | Vehicle $i$ |
| $k$ | RSU $k$ |
| $q_v$ | The number of RSUs that one vehicle can access simultaneously |
| $q_R$ | The number of vehicles that one RSU can serve simultaneously |
| $\mathcal{P}_i$ | The preference list of vehicle $i$ |
| $\mathcal{Y}_{i,k}$ | The utility value of vehicle $i$ with the serving of RSU $k$ |
| $\mathcal{A}_k$ | The accepted set of RSU $k$ |
| $\mathcal{F}_k$ | The forbidden list of RSU $k$ |
| $\theta$ | The weights of the evaluated Deep Q-Network |
| $\theta^-$ | The weights of the target Deep Q-Network |

ity values and construct the corresponding preference list. Then, they send the offloading requests to BS. On the third step, BS performs task scheduling and resource allocation and sends the task assignment to RSUs. Finally, all vehicles send their offloading tasks to the corresponding RSUs. Table 1 summarizes the mainly used system variables in this paper. The pseudo-code of our intelligent offloading system is shown in Algorithm 1.

**ALGORITHM 1:** The Pseudo-code of the Intelligent Offloading System.

```
Vehicles broadcast their position information;
Vehicles send offloading requests to BS;
Task scheduling = Algorithm 3;
for each vehicle $i$ in Task scheduling do
    Algorithm 4;
end
BS sends task assignments to RSUs;
RSUs perform computation offloading;
```

### 6.2 Two-sided Matching for Task Scheduling

To resolve occurred conflicts during the scheduling of offloading requests, the scheduling process is modelled as a two-sided matching model, and a Dynamic V2I Matching algorithm (DVIM) is developed to find the optimal match. We consider that a vehicle can access up to $q_v$ RSUs simultaneously. An RSU can serve at most $q_R$ vehicles at one time. In addition, each RSU maintains two lists: forbidden list $\mathcal{F}_k$ and accepted set $\mathcal{A}_k$. Traditional static matching algorithms traverse the complete set every time, which is time-consuming and wastes computing resources. To reduce the computational complexity of our designed algorithm, users rejected by RSU $k$ are added to the forbidden list. By default, they cannot be selected again in this round. Similarly, the accepted list is leveraged to record current accepted offloading requests by RSU $k$.

We now elaborate the whole process of the DVIM algorithm. It starts with the initialization of the forbidden list and accepted set. Next, each vehicle calculates its utility $\mathcal{Y}_{i,k}$ if its task is offloaded to RSU $k$, then all vehicles construct their preference list $\mathcal{P}_i$ in descending order of $\mathcal{Y}_{i,k}$. In the matching iteration phase, each vehicle that has been matched with less than $q_v$ RSUs sends offloading requests to the most preferred RSU in $\mathcal{P}_i$ and then removes this RSU from the preference list of this vehicle. After all vehicles propose their requests, RSUs that have received proposals will

**ALGORITHM 2:** The Pseudo-code of the Utility-based Selection Algorithm.

```
for each vehicle $i = 1, 2, \ldots, U$ do
    if vehicle i-matched-number $< q_v$ and $\mathcal{P}_i \neq \emptyset$ then
        Vehicle $i$ proposes itself to $\mathcal{P}_i[1]$;
        vehicle-propose = 1;
        Remove $\mathcal{P}_i[1]$ from $\mathcal{P}_i$;
        for each RSU $k = 1, 2, \ldots, K$ do
            if RSU k has received any proposal, denoted as $V_i$ then
                if $\mathcal{Y}_k(\{V_i\} \cup \mathcal{A}_k)) > \mathcal{Y}_k((\mathcal{A}_k))$ then
                    RSU $k$ accepts the proposal from $V_i$;
                    if $|\mathcal{A}_k| \geqslant q_R$ then
                        for each vehicle $i \in \mathcal{A}_k$ do
                            Calculate $\mathcal{Y}_k((\mathcal{A}_k \setminus V_i))$;
                        end
                        Find the largest utility function in the above loop, denoted as $\mathcal{Y}_k((\mathcal{A}_k \setminus V_n))$;
                        Opposite number RSU $k$ unmatch with $V_n$;
                        Add vehicle $n$ into $\mathcal{F}_k$;
                    end
                    else
                        Refuse the proposal from $V_m$;
                        Add vehicle $m$ into $\mathcal{F}_k$;
                    end
                end
            end
        end
    end
end
```

decide whether to accept them or not. Generally, RSUs accept those proposals that can increase overall utility values. If RSU $k$ has already matched up with $q_R$ vehicles, then it will unmatch with the least important vehicle. Then all responses will be sent back to vehicles. Vehicles continue to send requests when they match with less than $q_v$ RSUs and their preference lists are not empty. The algorithm terminates when no more vehicles would like to send offloading requests.

## 6.3 Deep Reinforcement Learning–based Mobility-Aware Offloading

The second sub-optimization problem is a joint optimization problem of resource allocation and offloading decision based on Markov chains. It is complicated when environment states in a series of time slots are taken into account. Therefore, an improved DRL method is designed. First, experience replay memory $D$ is initialized, which can hold $N$ transitions. Action-value function $Q$ is initialized with random weight $\theta$, and the target Q-network used to calculate the Temporal Difference (TD) target is also initialized with the same weight, i.e., $\theta^- = \theta$. Next, offloading requests are scheduled in the task. For each step in one event, a random RSU is selected from the available accessing list with probability $\varepsilon$. Otherwise, a greedy strategy is leveraged to select the RSU with the largest $Q$-value of the current action-value function. After RSU $k$ is chosen, the immediate reward $r_t$ as well as the next states is observed. Therefore, a set of transition $(observation, action, reward, next\ observation)$ can be obtained and stored in the replay buffer. In the neural network learning phase, DQN randomly samples a mini-batch transition

**ALGORITHM 3:** The Pseudo-code of the DVIM Algorithm.

```
Initialize the forbidden list $\mathcal{F}_k$ to be empty;
Initialize the accepted set $\mathcal{A}_k$ to be empty;
for each vehicle $i = 1, 2, \ldots, U$ do
    for each RSU $k = 1, 2, \ldots, K$ do
        if vehicle $i \notin \mathcal{F}_k$ then
            Calculate utility function value $\mathcal{Y}_k(V_i)$;
        end
    end
end
Vehicle $i$ constructs preference list $\mathcal{P}_i$;
matching-iteration = 1;
vehicle-propose = 0;
while matching-iteration = 1 or vehicle-propose = 1 do
    matching-iteration += 1;
    Algorithm 1;
end
```

from replay buffer $D$. For each sample, whether the next state is the termination state of the event should be determined. If so, the TD target is $r_j$, otherwise the target DQN is used to calculate the TD target: $y_j = r_j + \gamma Q(x_{j+1}, arg\max_{a'} Q(x_{j+1}, a'; \theta); \theta^-)$. Then gradient descent with the goal of minimizing the variance-error is performed to update the evaluated Q-network parameters: $\Delta\theta = \alpha[r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)]\nabla Q(s, a; \theta)$. Finally, the TD target network parameters and random probability $\varepsilon$ are updated every $C$ steps, which guarantees that the target Q-network fits the action-value function well and accelerates the convergence speed. The above process is shown in Algorithm 4.

## 7 PEFORMANCE EVALUATION

In this section, the performance of our proposed algorithms, i.e., DVIM and Mobility-Aware Double DQN (MADD), are evaluated. For the first module, we compare DVIM algorithm with exhaustive searching, a greedy method, and a random sorting method. Simulation results demonstrate that DVIM achieves a good trade-off between network performance and execution time. For the second module, MADD is evaluated to be superior to traditional DQN, Q learning, and two baseline algorithms.

### 7.1 Simulation Setup

Before implementing DVIM algorithm, parameters in the function (Equation (17)) are clarified. Authors in Reference [12] provide a joint probability distribution, where most offloading tasks have *medium* or *low* priorities and urgencies. Generally, significant tasks, such as traffic jams or even traffic accidents, have *critical* or *high* priorities and *extreme* or *high* urgencies. Therefore, we set the ratio of tasks at different levels as $[0.1, 0.2, 0.4, 0.3]$ [12].

Priority levels define the maximum utility value that the offloading task can achieve, which are set by $\pi^{(i)}(critical) = 8, \pi^{(i)}(high) = 4, \pi^{(i)}(medium) = 8, \pi^{(i)}(low) = 1$. Urgency levels determine the exponential decay rate of the utility function, which are set by $\rho^{(i)}(extreme) = 0.6, \rho^{(i)}(high) = 0.2, \rho^{(i)}(medium) = 0.1, \rho^{(i)}(low) = 0.01$ [12]. Furthermore, we consider that there are five RSUs and some vehicles with computation offloading requests within the communication range of BS. Performance indicators are as follows:

**ALGORITHM 4:** The Pseudo-code of the Mobility-Aware DDQN.

```
Input: system states $\chi_i(t)$, available action space $\mathcal{K}$;
Output: maximum QoE $\mathcal{R}_i$;
Initialize the experience replay buffer $D$ into capacity $N$ ;
Initialize the evaluated Deep Q-Network with weights $\theta$ ;
Initialize the target Deep Q-Network with weights $\theta^- = \theta$ ;
for each episode $i = 1, 2, \ldots, U$ do
    Initialize observation $s_1$, and pre-process sequence $x_1 = \varphi(s_1)$;
    for $t = 1, 2, \ldots, T-1$ do
        With probability $\varepsilon$ select a random action $a_{i,k}(t)$ ;
        Otherwise, select $a_t = arg\max_{a_{i,k}(t)} Q(x, a; \theta)$;
        Execute action $a_{i,k}(t)$ ;
        Observe the immediate reward $r_t = \mathcal{R}_i(t)$ and the next observation $s_{t+1}$.;
        Process $s_{t+1}$ to be the next state $x_{t+1} = \varphi(s_{t+1})$).;
        Store transition $(x_t, a_t, r_t, x_{t+1})$ into $D$;
        Sample random mini-batch of transitions $(x_j, a_i, r_j, x_{j+1})$ from $D$;
        if episode terminates at step $j+1$ then
            the target $Q$-value $y_j = r_j$;
        end
        else
            $y_j = r_j + \gamma Q(x_{j+1}, arg\max_{a'} Q((x_{j+1}, a'; \theta)); \theta^-)$;
        end
        Perform gradient decent on $(y_j - Q((x_j, a_j; \theta)))^2$;
        Every $C$ steps, update the target Deep Q-Network parameters and probability $\varepsilon$ with rate $\sigma$ and $\mu$;
            $\theta^- = \sigma\theta + (1-\sigma)\,\theta^-,$
            $\varepsilon = \varepsilon - \mu\varepsilon.$
    end
end
```

(1) Total utilities: The optimization target of the DVIM algorithm is to maximize total utilities of all vehicles, which measures QoE of users.
(2) Execution time: The time consumed for algorithms to obtain the task scheduling result.
(3) Average QoE: Average profit earned by network operators from vehicles.

For the second module, the key parameters are stated in Table 2. The transition probability matrix is set as follows [11]:

$$\Theta = \begin{bmatrix} 0.5 & 0.25 & 0.125 & 0.0625 & 0.0625 \\ 0.0625 & 0.5 & 0.25 & 0.125 & 0.0625 \\ 0.0625 & 0.0625 & 0.5 & 0.25 & 0.125 \\ 0.125 & 0.0625 & 0.0625 & 0.5 & 0.25 \\ 0.25 & 0.125 & 0.0625 & 0.0625 & 0.5 \end{bmatrix}.$$

For communication states, we set transition probability $\gamma_i^k = 0.7$, and $\psi_{g_s,h_s}(t) = 0.3$. To implement the DDQN-based MADD algorithm, TensorFlow 0.12.1 is employed with Python Anaconda 4.3 on Ubuntu 16.04 LTS. Four schemes are compared with the MADD algorithm:

Table 2. Simulation Parameters

| Parameter | Value | Description |
|---|---|---|
| $\mathcal{U}$ | 10 | The number of vehicles. |
| $K$ | 5 | The number of RSUs. |
| $d_i$ | 10MB | The data size of the offloading task. |
| $c_i$ | 100Mcycles | Required CPU cycles to complete a task. |
| $b_{i,k}/b_{i,0}$ | 1/4MHz | The bandwidth of RSU $k$/BS allocated to vehicle $i$. |
| $\tau_{i,k}/\tau_{i,0}$ | 10/20 units/Mbps | The unit charging-price for accessing the virtual network. |
| $\delta_{i,k}/\delta_{i,0}$ | 2/20 units/MHz | The unit paid-price for leasing bandwidth. |
| $\phi_k/\phi_0$ | 20/10 units/Mbps | The unit charging-price for task execution. |
| $\eta_k/\eta_0$ | 0.3/1 units/J | The unit paid-price for energy consumption of computation. |
| $\varsigma_k/\varsigma_0$ | 0.1/0.2 W/Hz | The unit energy consumption of running one CPU cycle. |
| $f_{i,k}$ | [10,12,14,16,18]GHz | The realization of the computation capability. |

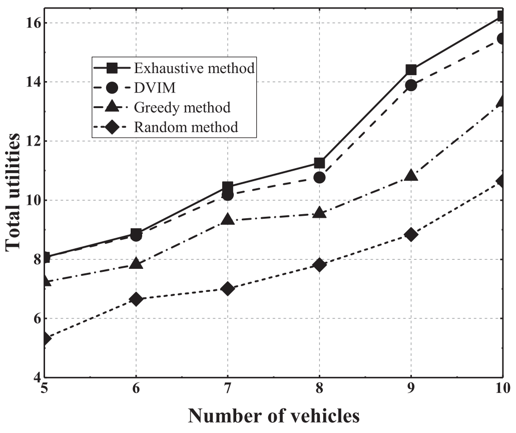

Fig. 3. Comparison of total utilities under different numbers of vehicles.

(1) DQN: Traditional DQN methods use one value function to evaluate the selection of actions. Generally, DQN methods tend to choose the action that maximizes the reward value of the next step, leading to overestimation inevitably.
(2) Q learning: As a classic temporal difference algorithm, it always chooses the largest action value at the next moment as the target. In addition, it also needs to record rewards of all state-action pairs.
(3) A greedy method: Contrary to reinforcement learning, the greedy algorithm chooses the action with the largest reward value at the current moment.
(4) Local computing: It is a baseline algorithm, where all vehicles offload their tasks to the local BS.

## 7.2 Simulation Results

This subsection illustrates the performance evaluation of the intelligent offloading system, including two modules: task scheduling and resource allocation.

Figure 3 shows the comparison of total utilities under different numbers of vehicles. When the number of vehicles is relatively small, DVIM algorithm can almost perform as well as the

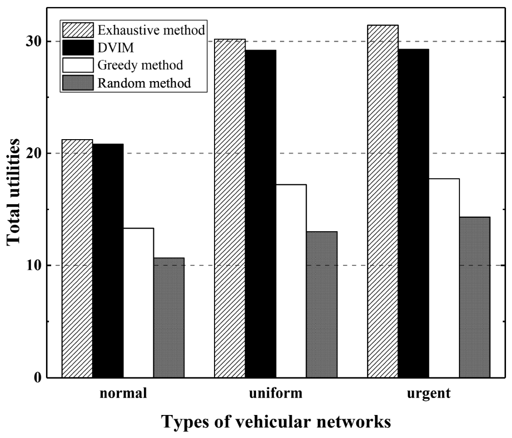

Fig. 4. Comparison of total utilities in different types of vehicular networks.

exhaustive algorithm. Due to the randomness of the offloading task, total utilities increase non-linearly with the increasing number of vehicles. When the number of vehicles is large (e.g., $\mathcal{U}$ = 9), DVIM algorithm still maintains a good performance, which is only 5% lower than that of the exhaustive algorithm. The greedy algorithm has a declined performance due to the increased resource competition. The total utility of DVIM is 22% higher than that of the greedy method. In summary, the performance of our proposed DVIM algorithm is close to the exhaustive algorithm. It is superior to the greedy algorithm and the random algorithm, when the number of vehicles is relatively large, i.e., the resource competition is fierce.

Figure 4 illustrates the comparison of total utilities in different types of vehicular networks. In normal vehicular networks, the proportion of the four-level offloading tasks (*critical* or *extreme*, *high*, *medium*, and *low*) is [0.1,0.2,0.4,0.3]. According to Reference [12], it is reasonable to assume that 70% of tasks have *medium* or *low* priorities as well as urgencies. The result of the exhaustive algorithm can be viewed as the upper bound. The proposed DVIM algorithm can achieve 98% of the upper bound performance. It is 20% better than that of the greedy algorithm, and nearly 50% better than that of the random algorithm. In a local area or within a specific time period (e.g., morning and evening traffic peaks), the proportion of tasks in each level changes with the state of vehicular networks. High priority and urgent tasks related to traffic jams or accidents are uploaded frequently. Thus, we also evaluate the performance of the DVIM algorithm in uniform and urgent vehicular networks, where proportions are [0.25,0.25,0.25,0.25] and [0.4,0.4,0.1,0.1], respectively. In uniform vehicular networks, the performance of DVIM is 3.4% lower than that of the exhaustive searching and 41% higher than that of the greedy method. In urgent vehicular networks, DVIM can reach 93% of the upper bound, and increase 65% comparing with the greedy method. The priority and urgency levels of offloading tasks in a normal environment are lower than their counterparts in uniform and urgent environments. Therefore, the achievable upper bound of utilities is lower than those of the other two circumstances. In summary, considering both personal priority and overall utilities, the performance of the DVIM algorithm can approximate to the upper bound obtained by the exhaustive algorithm and is 40% higher on average than the greedy algorithm.

Execution time of different algorithms is compared in Figure 5. When there are five vehicles, the execution time differences among the proposed DVIM, the greedy method, and the random method are very close. The execution time of exhaustive searching is 10 times higher than that of the other three algorithms. Moreover, the execution time of DVIM, the greedy method, and the

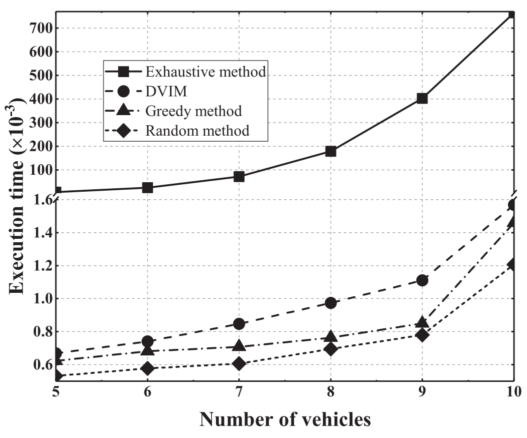

Fig. 5. Execution time of different algorithms.

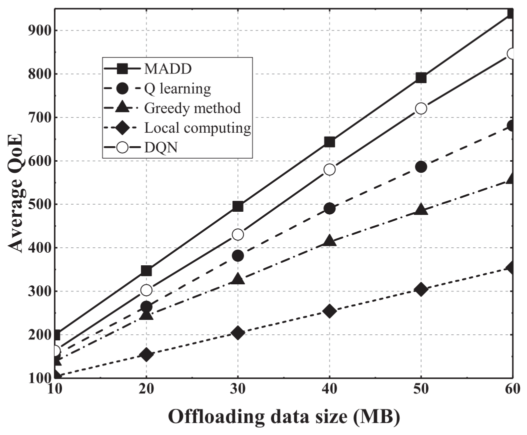

Fig. 6. The average QoE varying from offloading data size.

random method grows slowly with the increasing number of vehicles, while that of the exhaustive method grows exponentially, since the searching space increases dramatically. When the number of vehicles equals to 10, the execution time of exhaustive searching is 500 times higher than that of the other three algorithms. Although the exhaustive method can reach the upper bound, it is not practical due to its high time complexity. In addition, it is demonstrated that our proposed DVIM algorithm can approximate to the performance of the exhaustive algorithm with much lower time consumption.

Figure 6 shows the effect of offloading data size $d_i$, varying from 10MB to 60MB. The overall average QoE grows steadily with the increase of data size. The increasing speed of local computing (i.e., all computation tasks are fulfilled by the BS) is the slowest among schemes. This is because BS consumes more resources and energy than RSUs to transmit and compute large amounts of data. With the burgeoning requirement of proximity and preemptive services, RSUs are suitable to deal with large amounts of data. Since the DQN method cannot overcome the drawback of overestimation, the performance of our proposed MADD algorithm is 15% higher than that of the DQN method on average. Q learning and the greedy method do not fully take the dynamic changes of network states into account, whose performances are 25% and 35% lower than that of the MADD

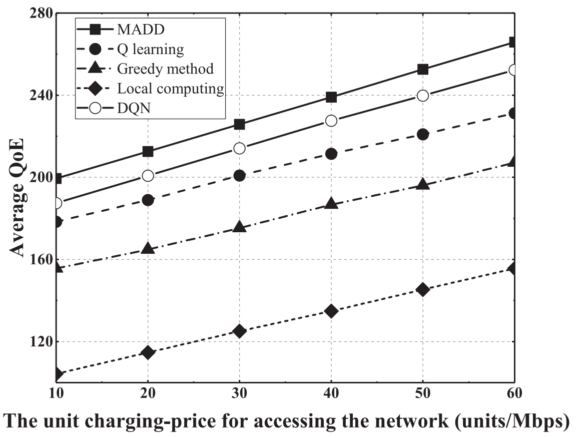

Fig. 7. The average QoE varying from the unit charging-price for virtual network accessing.

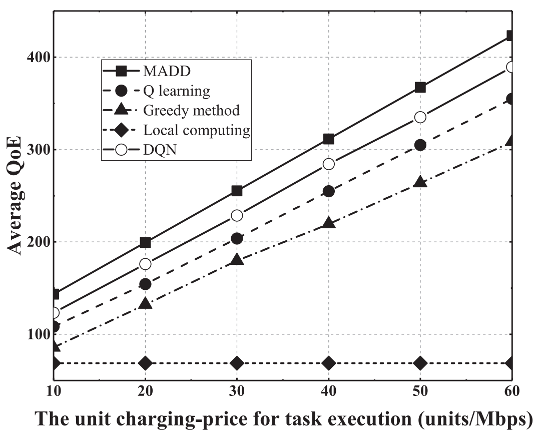

Fig. 8. The average QoE varying from the unit charging-price for task execution.

algorithm, respectively. In summary, the proposed MADD algorithm performs better than other existing schemes. When the amount of the offloading data increases, the performance advantage increases.

As shown in Figure 7, we evaluate the average QoE of vehicles varying with the change of unit charging-price $\tau_{i,k}$ for virtual network accessing. We notice that the accessing fee is doubled from 10 to 20, the overall QoE obtained by MADD only increases by 6.5% and 5.2% when the accessing fee increases from 50 to 60. Thus, it is unreasonable that network operators gain more profit by increasing the accessing fee without restriction. When the unit charging price is high, price increasing may prompt users to choose BS.

The effect of $\phi_k$, which is the unit charging price for task execution of RSU $k$, is shown in Figure 8. When $\phi_k$ rises, the incoming of MEC-based offloading generally increases. Therefore, the average QoE increases linearly with the rise of the unit charging price for task execution. Since local computing does not occupy MEC servers, the average QoE stays the same. The performance of the proposed MADD algorithm is about 12% and 20% higher than those of the DQN method and the greedy method, respectively. This is because the target network is built separately, and the overestimation problem is solved by applying DDQN frameworks. In general, the MADD algorithm performs well in the local vehicular networks.

## 8 CONCLUSION

In this article, we focus on deep reinforcement learning for vehicular edge computing and construct an intelligent offloading system. Network states are modeled as finite-state Markov chains. The mobility of vehicles and non-orthogonal multiple access are also taken into consideration. The offloading system contains two modules, i.e., task scheduling module and resource allocation module. The joint optimization problem for these two modules is formulated with the objective of maximizing total QoE of vehicles. Due to the NP-hardness of the formulated problem, it is divided into two sub-optimization problems. For the first one, a two-sided matching approach is designed to schedule offloading requests, with the purpose of maximizing utilities of vehicles. A DDQN-based algorithm is developed to solve the second sub-problem. Numerical results demonstrate that the matching algorithm in the first module can reach 95% of the exhaustive algorithm in different network scenarios and decrease the execution time by more than 90%. For the second module, DDQN-based algorithm performs 10% to 15% better than that of the traditional DQN method. Therefore, our offloading system is efficient and effective. In our future work, we will consider how to realize energy-efficient vehicular edge computing with favorable security [32, 33].

## REFERENCES

[1] Zhiguang Cao, Hongliang Guo, and Jie Zhang. 2018. A multiagent-based approach for vehicle routing by considering both arriving on time and total travel time. *ACM Trans. Intell. Syst. Technol.* 9, 3 (2018), 25.

[2] Xu Chen, Lei Jiao, Wenzhong Li, and Xiaoming Fu. 2016. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Trans. Netw.* 5 (2016), 2795–2808.

[3] Ruilong Deng, Rongxing Lu, Chengzhe Lai, and Tom H. Luan. 2015. Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing. In *Proceedings of the IEEE International Conference on Communications (ICC'15).* IEEE, 3909–3914.

[4] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H. Luan, and Hao Liang. 2016. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Int. Things J.* 3, 6 (2016), 1171–1181.

[5] Ruilong Deng, Zaiyue Yang, Jiming Chen, Navid Rahbari Asr, and Mo-Yuen Chow. 2014. Residential energy consumption scheduling: A coupled-constraint game approach. *IEEE Trans. Smart Grid* 5, 3 (2014), 1340–1350.

[6] Boya Di, Lingyang Song, Yonghui Li, and Geoffrey Ye Li. 2017. Non-orthogonal multiple access for high-reliable and low-latency V2X communications in 5G systems. *IEEE J. Select. Areas Commun.* 35, 10 (2017), 2383–2397.

[7] Lin Gu, Deze Zeng, Song Guo, Ahmed Barnawi, and Yong Xiang. 2017. Cost efficient resource management in fog computing supported medical cyber-physical system. *IEEE Trans. Emerg. Topics Comput.* 5, 1 (2017), 108–119.

[8] Gabriel Guerrero-Contreras, Jose Luis Garrido, Sara Balderas-Diaz, and Carlos Rodriguez-Dominguez. 2017. A context-aware architecture supporting service availability in mobile cloud computing. *IEEE Trans. Serv. Comput.* 10, 6 (2017), 956–968.

[9] Ying He, Chengchao Liang, Richard Yu, and Zhu Han. 2018. Trust-based social networks with computing, caching, and communications: A deep reinforcement learning approach. *IEEE Trans. Netw. Sci. Eng.* DOI: 10.1109/TNSE.2018.2865183

[10] Ying He, F. Richard Yu, Nan Zhao, Victor C. M. Leung, and Hongxi Yin. 2017. Software-defined networks with mobile edge computing and caching for smart cities: A big data deep reinforcement learning approach. *IEEE Commun. Mag.* 55, 12 (2017), 31–37.

[11] Ying He, Nan Zhao, and Hongxi Yin. 2018. Integrated networking, caching, and computing for connected vehicles: A deep reinforcement learning approach. *IEEE Trans. Vehic. Technol.* 67, 1 (2018), 44–55.

[12] Bhavesh Khemka, Ryan Friese, Luis D. Briceno, Howard Jay Siegel, Anthony A. Maciejewski, Gregory A. Koenig, Chris Groer, Gene Okonski, Marcia M. Hilton, Rajendra Rambharos et al. 2015. Utility functions and resource management in an oversubscribed heterogeneous computing environment. *IEEE Trans. Comput.* 64, 8 (2015), 2394–2407.

[13] Jun Li, Xue Mei, Danil Prokhorov, and Dacheng Tao. 2017. Deep neural network for structural prediction and lane detection in traffic scene. *IEEE Trans. Neural Netw. Learn. Syst.* 28, 3 (2017), 690–703.

[14] Yu-Jui Liu, Shin-Ming Cheng, and Yu-Lin Hsueh. 2017. eNB selection for machine type communications using reinforcement learning based Markov decision process. *IEEE Trans. Vehic. Technol.* 66, 12 (2017), 11330–11338.

[15] Nguyen Cong Luong, Zehui Xiong, Ping Wang, and Dusit Niyato. 2018. Optimal auction for edge computing resource management in mobile blockchain networks: A deep learning approach. In *Proceedings of the IEEE International Conference on Communications (ICC'18)*. IEEE, 1–6.

[16] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, Fei-Yue Wang et al. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intell. Transport. Syst.* 16, 2 (2015), 865–873.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[18] K. Mohamed, Etienne Côme, Latifa Oukhellou, and Michel Verleysen. 2017. Clustering smart card data for urban mobility analysis. *IEEE Trans. Intell. Transport. Syst.* 18, 3 (2017), 712–728.

[19] Zhaolong Ning, Xiping Hu, Zhikui Chen, MengChu Zhou, Bin Hu, Jun Cheng, and Mohammad S. Obaidat. 2018. A cooperative quality-aware service access system for social Internet of vehicles. *IEEE Int. Things J.* 5, 4 (2018), 2506–2517.

[20] Zhaolong Ning, Xiaojie Wang, and Jun Huang. 2019. Mobile edge computing-enabled 5G vehicular networks: toward the integration of communication and computing. *IEEE Vehic. Technol. Mag.* 14, 1 (2019), 54–61.

[21] Zhaolong Ning, Xiaojie Wang, Feng Xia, and Joel Jose Rodrigues. 2019. Joint computation offloading, power allocation, and channel assignment for 5G-enabled traffic management systems. *IEEE Trans. Industr. Inform.* 15, 5 (2019), 3058–3067.

[22] Zhaolong Ning, Feng Xia, Noor Ullah, Xiangjie Kong, and Xiping Hu. 2017. Vehicular social networks: Enabling smart mobility. *IEEE Commun. Mag.* 55, 5 (2017), 49–55.

[23] Le Thanh Tan and Rose Qingyang Hu. 2018. Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning. *IEEE Trans. Vehic. Technol.* 67, 11 (2018), 10190–10203.

[24] David Tse and Pramod Viswanath. 2005. *Fundamentals of Wireless Communication.* Cambridge University Press.

[25] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'05)*, Vol. 2. 5.

[26] Chenmeng Wang, Chengchao Liang, F. Richard Yu, Qianbin Chen, and Lun Tang. 2017. Computation offloading and resource allocation in wireless cellular networks with mobile edge computing. *IEEE Trans. Wirel. Commun.* 16, 8 (2017), 4924–4938.

[27] Xiaojie Wang, Zhaolong Ning, and Lei Wang. 2018. Offloading in Internet of vehicles: A fog-enabled real-time traffic management system. *IEEE Trans. Industr. Inform.* 14, 10 (2018), 4568–4578.

[28] Xiaokang Wang, Laurence T. Yang, Xia Xie, Jirong Jin, and M. Jamal Deen. 2017. A cloud-edge computing framework for cyber-physical-social services. *IEEE Commun. Mag.* 55, 11 (2017), 80–85.

[29] Zehua Wang, Derrick Wing Kwan Ng, Vincent W. S. Wong, and Robert Schober. 2017. Robust beamforming design in C-RAN with sigmoidal utility and capacity-limited backhaul. *IEEE Trans. Wirel. Commun.* 16, 9 (2017), 5583–5598.

[30] Yifei Wei, F. Richard Yu, and Mei Song. 2010. Distributed optimal relay selection in wireless cooperative networks with finite-state Markov channels. *IEEE Trans. Vehic. Technol.* 59, 5 (2010), 2149–2158.

[31] Jinming Wen, Chao Ren, and Arun Kumar Sangaiah. 2018. Energy-efficient device-to-device edge computing network: An approach offloading both traffic and computation. *IEEE Commun. Mag.* 56, 9 (2018), 96–102.

[32] Guangquan Xu, Jia Liu, Yanrong Lu, Xianjiao Zeng, Yao Zhang, and Xiaoming Li. 2018. A novel efficient MAKA protocol with desynchronization for anonymous roaming service in global mobility networks. *J. Netw. Comput. Appl.* 107 (2018), 83–92.

[33] Guangquan Xu, Yao Zhang, Arun Kumar Sangaiah, Xiaohong Li, Aniello Castiglione, and Xi Zheng. 2019. CSP-E2: An abuse-free contract signing protocol with low-storage TTP for energy-efficient electronic transaction ecosystems. *Inform. Sci.* 476 (2019), 505–515.

[34] Jie Xu, Lixing Chen, and Shaolei Ren. 2017. Online learning for offloading and autoscaling in energy harvesting mobile edge computing. *IEEE Trans. Cog. Commun. Netw.* 3, 3 (2017), 361–373.

[35] Chenggang Yan, Hongtao Xie, Dongbao Yang, Jian Yin, Yongdong Zhang, and Qionghai Dai. 2018. Supervised hash coding with deep neural network for environment perception of intelligent vehicles. *IEEE Trans. Intell. Transport. Syst.* 19, 1 (2018), 284–295.

[36] Xianjiao Zeng, Guangquan Xu, Xi Zheng, Yang Xiang, and Wanlei Zhou. 2018. E-AUA: An efficient anonymous user authentication protocol for mobile IoT. *IEEE Int. Things J.* 6, 2 (2018), 1506–1519. DOI: https://doi.org/10.1109/JIOT.2018.2847447.

[37] Desheng Zhang, Tian He, and Fan Zhang. 2018. Real-time human mobility modeling with multi-view learning. *ACM Trans. Intell. Syst. Technol* 9, 3 (2018), 22.

[38] Shuhang Zhang, Boya Di, Lingyang Song, and Yonghui Li. 2017. Sub-channel and power allocation for non-orthogonal multiple access relay networks with amplify-and-forward protocol. *IEEE Trans. Wirel. Commun.* 16, 4 (2017), 2249–2261.

[39] Zixing Zhang, Jürgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Björn Schuller. 2018. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Trans. Intell. Syst. Technol* 9, 5 (2018), 49.

[40] Bowen Zhou and Rajkumar Buyya. 2018. Augmentation techniques for mobile cloud computing: A taxonomy, survey, and future directions. *ACM Comput. Surv.* 51, 1 (2018), 13.

[41] Chao Zhu, Giancarlo Pastor, Yu Xiao, Yong Li, and Antti Ylae-Jaeaeski. 2018. Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In *Proceedings of the 15th IEEE International Conference on Sensing, Communication, and Networking (SECON'18).* IEEE, 1–9.

Received December 2018; revised February 2019; accepted March 2019