Multicast-Oriented Task Offloading for Vehicle Edge Computing

HAOTIAN LI1, XUJIE LI1,2,3, (Member, IEEE), MINGYUE ZHANG4, AND BUYANKHISHIG ULZIINYAM1,5

1College of Computer and Information, Hohai University, Nanjing 210098, China
2National Mobile Communications Research Laboratory, Southeast University, Nanjing 210096, China
3Key Laboratory of Wireless Sensor Network and Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China
4Jacobs School of Engineering, University of California at San Diego, San Diego, CA 92092, USA
5School of Information and Communication Technology, Mongolian University of Science and Technology, Ulaanbaatar 14191, Mongolia

Corresponding author: Xujie Li (lixujie@hhu.edu.cn)

This work was supported in part by the Postgraduate Research and Practice Innovation Program of Jiangsu Province under Grant KYCX20-0537, in part by the Open Research Fund of the National Mobile Communications Research Laboratory, Southeast University, under Grant 2020D16, in part by the Provincial Key Research and Development Program of Jiangsu under Grant BE2019017, in part by the Six Talent Peaks Project in Jiangsu under Grant DZX-008, and in part by the Open Research Fund of the Key Laboratory of Wireless Sensor Network and Communication, Chinese Academy of Sciences, under Grant 20190914.

ABSTRACT Vehicle edge computing (VEC) is expected to be widely used in 5G wireless network in the future owing to its low latency, high reliability, and easy deployment. Unfortunately, the computation resources of roadside units (RSUs) equipped with computing ability are limited, which makes us need to seek other computing nodes. In this article, we explore an offloading model to meet this challenge. Future vehicles equipped with computing abilities can provide VEC services in vehicular fog networks (VeFN), which greatly reduces task delay and improves the efficiency of transportation system. We consider that a package consisting of several tasks is offloaded to vehicles with different computing abilities via multicast, and these vehicles process tasks at the same time. First, we use the mathematical method to construct the system utility function, aiming to obtain low delay and low computing cost, and produce a 0-1 programming problem with inequality constraints. We then solve the slack form of the primary problem based on the interior point method. Finally, a low-complexity algorithm is proposed to optimize the task delay and cost. Numerical results show that the algorithm has fast convergence speed and superior performance.

INDEX TERMS Delay, multicast, task offloading, vehicle edge computing.

I. INTRODUCTION

In recent years, with the development of Internet of Things (IoT) and wireless communication technology, Internet of Vehicles (IoV) has been developed more and more, on-board sensors (e.g. cameras and radars) can play a key role in efficient and safe transportation systems [1], [2]. Smart vehicles accessing popular content and sharing transportation information are evolving as an emerging paradigm to support the advanced driver assistance systems and self-driving [3]. However, computationally intensive applications can put pressure on resource-constrained vehicles, causing bottlenecks and making it difficult to ensure the required level of quality of service (QoS). Fortunately, mobile edge computing (MEC) technology can provide computing services at the edge of wireless access networks and near mobile users, thereby alleviating the heavy computing needs of individual users. Inspired by MEC technology, vehicle edge computing (VEC) network has emerged at the historic moment, which can offload computation tasks to the edge of the network, e.g. roadside computing servers (RCS).

VEC as an efficient technology, alleviates the computing pressure of vehicle users and improves QoS [4], [5]. Wang et al. [6] proposed a permissioned vehicular blockchain in VEC, called Parking-chain, where the parked vehicles (PVs) could share their idle computation resources with service requesters (SRs). Zhang et al. [7] studied parked vehicles assisted edge computing with Stackelberg game, where the overall cost of a task publisher in terms of monetary cost and subjective dissatisfaction caused by un-offloaded
workloads was minimized. Toward the flexible and fine-grained resource usage, Huang et al. [8] integrated the container-based virtualization with parked vehicle edge computing for ensuring the task execution in parked vehicles with fast response, increased scalability, and high efficiency. Li et al. [9] investigated an energy-efficient parked vehicular computing paradigm, and they designed a contract-based incentive mechanism to motivate parked vehicles to contribute their idle on-board computation resources. The aforementioned work focused on the parked vehicles with idle computational resources providing edge computing services, which is applicable in the VEC network. Wang et al. [10]–[12] proposed a wireless powered MEC system, where a multi-antenna access point (AP) (integrated with an MEC server) broadcasts wireless power to charge users and each user node relies on the harvested energy to compute tasks. Building on this proposed model, they developed an innovative framework to improve the MEC performance, by optimizing the energy transmit beamforming at the AP, the central processing unit frequencies, the numbers of offloaded bits at the users, and the time allocation among users. Wang et al. [13] considered the utilization of multi-antenna non-orthogonal multiple access (NOMA) technology for multi-user computation offloading, and then different users could simultaneously offload their tasks to the multi-antenna base station (BS) over the same time/frequency resources. The aforementioned work focused on the multi-antenna base station (BS) providing edge computing services with great energy efficiency, which was applicable and innovative in the MEC network.

However, the computation resources are limited, which makes the traditional VEC technology not always applicable. In order to meet the emerging needs of autonomous driving, future vehicles will not only be equipped with a wealth of on-board sensors such as cameras and radars, but will also be equipped with the powerful computing capability to process sensor data and make driving decisions. Vehicles can contribute their computation resources and act as fog nodes in the fog computing environment, which can be considered as a vehicle fog network (VeFN) [14]–[21]. Unlike traditional mobile crowdsensing, fog-based vehicular nodes are introduced specifically to meet the requirements for location-specific applications and location-aware data management in vehicular ad hoc networks, including parking navigation, road surface monitoring, traffic collision reconstruction, and V2V energy swapping. Although edge cloud is beneficial in terms of low delay and less backhaul bandwidth consumption, its computing capability is relatively limited in the busy phase of computing business, and the vehicles with powerful computing capabilities can compose a novel cloud, which is referred to as vehicular cloud [14]–[17].

Through vehicular cloud computing (VCC), not only the vehicle-related computation-intensive tasks can be fulfilled efficiently, but also the capacity of edge cloud can be enhanced by offloading computing tasks to the vehicular cloud. Sun et al. proposed an efficient task scheduling scheme in vehicular cloud by jointly considering the instability of resources, the heterogeneity of vehicular computing capabilities, and the inter-dependency of computing tasks [17]. Specifically, the vehicles with idle computation resources provided computing service in this scenario. In [18], the authors explored the possibility of moving vehicles as computing nodes. The authors considered that mobile vehicles, parked vehicles, RSUs, and even pedestrians’ smartphones with idle computing resources can provide computing services to other clients if necessary. For example, while the load of the data center or the traditional VEC servers (RSUs, base station, etc.) is relatively heavy, some tasks can be properly offloaded to the mobile vehicles to meet the intensive computing requirements of customer clients. The model proposed in [20] was different from the traditional VEC model, and the mobile nodes can provide computing services as traditional VEC servers. Hou et al. [19] studied the feasibility of vehicle fog computing and provided a quantitative analysis between capacity, vehicle mobility and connectivity. It is worth emphasizing that the authors believed that mobile cloudlets could perform remote cloud computing tasks and moving vehicles, especially slowly moving vehicles, would be an important part of vehicle fog computing infrastructures. The emergence of VeFN makes IoV realize computation resources sharing through task offloading, which will provide a wide range of fog applications.

The task offloading in VeFN is classified into three major modes: vehicle-vehicle offloading, vehicle-RSU-vehicle offloading, and pedestrian-RSU-vehicle offloading [18]. The authors proposed an energy-saving task offloading scheme to minimize the energy consumption of the MEC offloading system, which comprehensively optimized the offloading decision and the wireless resource allocation strategy [22]. A layered VEC offloading framework for vehicle network based on cloud computing was proposed in the work of Zhang et al. [23]. The authors studied on the task offloading mechanism and described it as a Stackelberg game model, and then adopt a distributed algorithm to obtain the optimal strategy of VEC servers. In the work of Dai et al. [24], a joint load balancing and offloading problem was proposed to maximize the system utility in the VEC network, and the authors introduced a joint optimization for selection, computing and offloading algorithm with low complexity, which jointly optimized selection decision, offloading ratio, and computation resources.

Sun et al. [25] proposed a new algorithm combining task replication and sequential learning to minimize average task offloading delay. In the work of Sun et al. [26], a task offloading scheme for a vehicle cloud computing system was designed, and a learning-based adaptive variable upper confidence bound (AVUCB) algorithm was proposed, which minimized the average offloading delay only based on historical delay observations. These two articles investigated vehicle-to-vehicle offloading issues and optimize the delay performance of offloading. In article [27], the author proposed a hierarchical offloading model for VEC: (i) RSUs
Multicast mode can greatly improve the efficiency of the transportation
system and save the tight computation resources of traffic
central servers or traditional edge/cloud servers.

Multicast mode can not only transmit data to multiple users
who need (join the group) at the same time, but also ensure
that it does not affect other communications of users who
do not need (join the group). Users who need the same data
stream join the same group to share a data stream, reducing
the load on the server. Since the multicast protocols copy and
forward the data stream according to the needs of the recipient,
the total service bandwidth of the server is not limited
by the bandwidth of the user access terminal. The multicast
mode greatly reduces the complexity of the process for the
offloading users, because the offloading users (e.g. RSUs)
do not need to establish communication links for many times,
but offload the package to the vehicles via multicast at a time.

The traditional VEC communication mode is always point-
to-point communication, which has a high requirement for
communication resources, and usually causes many unneces-
sary signal interference and equipment overload. Compared
with the traditional VEC communication mode, the multi-
cast mode has the characteristics of low load, small inter-
fERENCE, and favorable network resource management, which
effectively saves increasingly tight communication resources.
In addition, this offloading mode can avoid the risk of inter-
ruption/failure in the process of task computing, since mul-
tiple vehicles have the copies of offloading packages. Once
the original serving vehicle interrupts the task computing due
to unexpected conditions, other vehicles can take over the
leaving vehicle’s job and continue computing services, which
greatly improves the system reliability and QoS.

Our model makes full use of idle computation resources
of vehicles, which greatly improves the efficiency of the sys-
tem, relieves computing pressure of the RCSs, and processes
tasks with lower delay and fewer computing cost. The main
contributions of this article are as follows:

1) We establish a novel multicast-oriented VEC offload-
ing model. RSUs offload a task package to vehicle users
in the form of multicast and vehicles process these tasks
simultaneously.

2) We introduce a discrete-time system, which takes full
account of the real-time position of vehicles in each
time slot. Hence, we construct an optimization problem
which jointly consider the average delay and cost of
these tasks.

3) We propose an algorithm based on interior point
method to solve the optimization problem. Numerical
results prove that the algorithm has superior perfor-
mance and can quickly obtain satisfied results.

The remainder of this article is organized as follows.
Section II presents the system model and problem formu-
lation. Section III presents an algorithm jointly optimizing
the delay and cost for multicast-oriented tasks offloading
(JDCM). Section IV provides simulation results to validate
the performance of the proposed algorithm compared with
other benchmark schemes. Finally, Section V concludes our
contribution and discusses future work.
II. SYSTEM MODEL

As shown in Fig. 1, we consider a straight, one-way road with $M$ RSUs evenly distributed on the road, denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. There are $N$ vehicles on the road, denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$ and for a certain period of time, they are driving at constant speed along the centerline without changing lanes. The communication coverage of each RSU is regarded as a cellular, and there are $M$ cells in total, denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. There are $K$ tasks to be offloaded, written as $K = \{1, \ldots, K\}$. These $K$ tasks make up one offloading package which will be offloaded from RSUs to vehicles in a multicast manner. We assume that vehicles begin to process these tasks only after this package has been completely received. These tasks can be described by 3 parameters: (i) data size of tasks, denoted as $x_1, \ldots, x_K$ (bits), (ii) data size of results (ignored in this article), (iii) computation intensity of the task, denoted as $\lambda_k$ (CPU cycles per bit). Then, the computation resources (i.e., CPU cycles) required by task $k$, $k \in K = \{1, \ldots, K\}$, can be expressed as $\lambda_k x_k$ [30].

We assume that $N$ vehicles have mutually different computing abilities, written as: $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$, depending on the computing capabilities of on-board computers and working status of vehicles. Assume that this road has three lanes: lane A, lane B, lane C, and in different lanes the speeds of the vehicles are different. We consider that the speed of the vehicles in each lane is $v_{\text{lane}A}$, $v_{\text{lane}B}$, $v_{\text{lane}C}$, respectively and remain constant during a certain period of time. According to the design and planning of roads in our country, a road may usually have multiple lanes in a same direction to meet different requirements of vehicular speeds. This phenomenon is especially common on urban arterial roads or highways and in order to make our model more suitable for the practical traffic scenario, we respectively set the different speeds in three lines.

In our scenario, tasks are generated by many devices, such as pedestrians’ smartphones or wearable devices, traffic infrastructures generating computing data (RSUs, nearby traffic data centers, monitoring equipment, etc.), stationary or mobile vehicles. If these tasks cannot be completed by traditional VEC servers (usually VEC servers are integrated with RSUs), we consider that RSUs would offload these tasks to vehicles with idle computation resources. This process can be simplified as 3 parts: (i) tasks generated by smart devices; (ii) RSUs collection tasks; (iii) RSUs offload some tasks to mobile vehicles. We will focus on part (iii) in this article.

Since service continuity is important in the IoV, it is worthwhile to explore the problem of the handover between the adjacent RSUs. The authors considered that service continuity can be provided by using the radio access network (RAN) mechanisms such as multicast flows in handovers or more advanced techniques like RAN multicast area management [31]. They also proposed an end-to-end architecture for 5G multicast that goes beyond the current considerations for LTE broadcast, which exists solely as an isolated service [32]. Point-to-multipoint (PTM) transmissions could then be implemented in a flexible and dynamic manner as an essential RAN delivery tool such that PTM transmissions become a built-in RAN functionality without any special considerations in the core network, making it possible to dynamically and seamlessly switch between point-to-point (PTP) and PTM transmissions over the dynamically configurable RAN multicast area (RMA). Through such enhancements, 5G could provide a unified framework for PTP and multicast content delivery for relevant verticals and applications, including automotive, airborne, IoT, media and entertainment, and public warning and safety services. Nevertheless, the handover between the adjacent RSUs in our scenario is still an actual problem that required more consideration, and we will look for a more effective approach in the future work.

A. TASK OFFLOADING

As shown in Figure 1, RSUs are evenly distributed along the side of the road (near lane C). We consider that all the RSUs are identical, the length of the road covered by each RSU is $l_{\text{lane}}$, and the width of each lane is $w_{\text{lane}}$. For a certain period of time, vehicles in the same lane are driving at a constant speed along the centerline without changing lanes, i.e., the speed of vehicles in lane A is $v_{\text{lane}A}$, etc.

For convenience, we introduce a discrete-time system and divide the process into multiple identical time slots, with $d$ donating the length of each time slot, and each $d$ is small enough to approximate a continuous-time system. The speed of vehicle $n$ can be written as $v_n$, $n \in \mathcal{N} = \{1, 2, \ldots, N\}$, therefore the distance traveled by vehicle $n$ in each time slot is $v_n d$, and then the coordinates of each vehicle can be expressed as:

$$p_n(t) = p_n[0] + \sum_{i=1}^{t} v_n d[i]$$  

(1)

where, $p_n[0]$ is the initial coordinate of vehicle $n$, and $p_n[t]$ is the coordinate of vehicle $n$ in time slot $t$. We let $q_n[t]$ and $q_{m}^{RSU}$ respectively represent the vertical positions of vehicle $n$ and RSU $m$. Therefore, in slot $t$, the distance between vehicle $n$ and RSU $m$ can be expressed as:

$$D_{m,n}[t] = \sqrt{(p_n[t] - p_m^{RSU})^2 + (q_n[t] - q_{m}^{RSU})^2}$$  

(2)

In our scenario, these vehicles always communicate with the nearest RSU, that is, vehicle $n$ in cellular $m$ communicates
with RSU $m$ and if this vehicle arrives at the next cellular (i.e., cellular $m+1$), vehicle $n$ will continue to communicate with RSU $m+1$. Since the Doppler effect may have a certain influence on the communication quality, the channel quality of the moving vehicle and the RSU in the front may be better than the RSU in the rear. So, the channel between the nearest RSU and the user may not be better than the other RSUs. As shown in several existing works such as [18], the mobility of nodes in an intermittently connected network can be beneficial since mobility creates more chances of contacts between nodes, and thus the probability of communication and task offloading between the nodes also increases. The speeds of vehicles are limited in our scenario and we simply consider an ideal communication model. Therefore, the actual communication distance $D_{\text{off}}$ between vehicle $n$ and RSUs can be written as:

$$D_{\text{off}}[r] = \min_{m \in \mathcal{M}} D_{m,n}[r]$$  \hspace{1cm} (3)

Let $P_{\text{RSU}}$ denote the transmission power of RSU, so the transmission rate between RSUs and vehicles can be written as:

$$R_{\text{off}}[r] = W \log_2 \left( 1 + \frac{P_{\text{RSU}} \left( D_{\text{off}}[r] \right)^\alpha}{\sigma^2} \right)$$  \hspace{1cm} (4)

where, $\alpha$ is the path loss factor, and $\sigma^2$ is the white Gaussian noise power. $W$ is the transmit power of RSUs communicating with vehicles and they are identical. The channel small-scale effects are really common during the communication in urban environments. However, we only considered the ideal channel model for simplification. And the frequency interference is neglected since the OFDM scheme is used in our scenario. Then, we need to obtain the offloading delay $T_{\text{off}}$ of each vehicle. Firstly, according to formula (2), the longest transmission distance is defined as:

$$D_{\text{max}} = \sqrt{(3w_{\text{lane}})^2 + \left( \frac{1_{\text{lane}}}{2} \right)^2}$$  \hspace{1cm} (5)

Based on formula (5), the lower bound of the transmission rate between RSUs and vehicles can be written as:

$$R_{\text{min}} = W \log_2 \left( 1 + \frac{P_{\text{RSU}} \left( D_{\text{off}}[r] \right)^\alpha}{\sigma^2} \right)$$  \hspace{1cm} (6)

These $K$ tasks will be packaged into a multicast package and offloaded to the vehicles with idle computation resources, that is, these vehicles will receive these $K$ tasks at one-time. Then, the data size of the offloading package is $\sum_{k \in K} x_k$. So, the upper bound of the transmission delay can be written as:

$$T_{\text{off}} = \sum_{k \in K} x_k$$  \hspace{1cm} (7)

Therefore, we can express the upper bound of the transmission delay in a manner of time slot, which is given by:

$$t_{\text{max}} = \left\lceil \frac{T_{\text{off}}}{d} \right\rceil$$  \hspace{1cm} (8)

And then we define $t^*$:

$$t^* = \arg \min_{t} \left\lceil \sum_{k \in \mathcal{K}} R_{n,k} d_k - \sum_{k \in \mathcal{K}} x_k \right\rceil$$  \hspace{1cm} (9)

Finally, the time slots used by package offloading can be approximately expressed by:

$$t_{n,k}^* = \begin{cases} t^*, & \text{when } \sum_{t=1}^{t^*} R_{n,k} d_t - \sum_{k \in \mathcal{K}} x_k \geq 0 \\ t^* + 1, & \text{when } \sum_{t=1}^{t^*} R_{n,k} d_t - \sum_{k \in \mathcal{K}} x_k < 0 \end{cases}$$  \hspace{1cm} (10)

**B. TASK COMPUTING**

It is assumed that the idle computation resources of the vehicles remain fixed for a period of time, that is, the value of $\mathcal{F} = \{f_1, f_2, \ldots, f_N\}$ is consistent during the task processing. Therefore, the time for computing each task can be written as:

$$t_{n,k}^{\text{comp}} = \frac{\lambda_k x_k}{f_n}$$  \hspace{1cm} (11)

where $\lambda_k$ is the computation intensity of task $k$, $k \in \mathcal{K}$. Then, the time slots required by vehicle $n$ to compute task $k$ can be expressed by:

$$t_{n,k}^{\text{comp}} = \left\lceil \frac{T_{n,k}^{\text{comp}}}{d} \right\rceil$$  \hspace{1cm} (12)

**C. TASK DELAY**

The task delay consists of three parts: task offloading, task computing, and result feedback. If the RSUs receive the first feedback of task $k$ from a vehicle, the time spent by this vehicle is the delay of task $k$. In this article, the time of task return is ignored, owing to the result data of task is usually very small.

In order to reduce unnecessary resource consumption, not all the vehicles will provide computing services, since providing computing services consumes resources. We define $y \in \{0, 1\}$, that is, if $y_{n,k} = 0$, vehicle $n$ will not participate in computing task $k$; if $y_{n,k} = 1$, vehicle $n$ will participate in computing task $k$. Thus, we obtain a decision matrix $Y = \{y_{n,k}\}_{N \times K}$, where the rows of the matrix represent different vehicles and the columns represent different tasks.

In our model, we consider that $K$ tasks are packaged into a multicast package and offloaded to vehicles with idle computing resources, that is, these vehicles may receive the $K$ tasks for one time. Only if the vehicles completely received the offloading package, the CPUs would start to compute tasks. Therefore, the first part of the delay is the transmission time
of package with $K$ tasks. These $K$ tasks will be selectively computed by vehicle $n$, that is, vehicle $n$ processes 0, 1 or more tasks in this offloading package. We consider that a vehicle cannot compute multiple tasks simultaneously, and if it processes multiple tasks, it will process them sequentially (since the order of tasks does not affect the overall computing delay). For vehicle $n$, the delay of task $k$ includes the transmission time of the multicast package with $K$ tasks and the cumulative working time of CPU until task $k$ completed. Therefore, for the whole system, the delay of task $k$ is the time from the beginning of transmission until the earliest accomplishment of task $k$. However, not every vehicle take part in computing task $k$, and if vehicle $n$ did not participate in computing task $k$, its CPU occupied time would be 0 (only for task $k$). Therefore, in a cumulative time system, the minimum term obtained by the Minimum function is most likely not the delay of task $k$, since the time of many vehicular CPUs for computing task $k$ is 0. In order to deal with this problem, we assume that if vehicle $n$ did not compute task $k$, its CPU occupied time would be infinite. For convenience, we introduce a constant $Q$ with a large value to achieve this effect. Therefore, the delay of task $k$ can be obtained by:

$$t_k = \min_{n \in \mathbb{N}} \left\{ t_{\text{off}} + \sum_{i=1}^{k} y_{n,i}^{\text{comp}} + (1 - y_{n,k}) Q \right\}$$

(13)

D. PROBLEM FORMULATION

In this article, we jointly optimize the overall delay and computing cost of these $K$ tasks, that is, we aim to obtain great delay performance with few computing cost. First, the overall delay of these $K$ tasks can be written as:

$$t_{\text{sum}} = \max_{k \in \mathbb{K}} \left\{ \min_{n \in \mathbb{N}} \left\{ t_{\text{off}} + \sum_{i=1}^{k} y_{n,i}^{\text{comp}} + (1 - y_{n,k}) Q \right\} \right\}$$

(14a)

$$s.t. y_{n,k} \in \{0, 1\}, \quad k \in \mathbb{K}, \quad n \in \mathbb{N}$$

(14b)

$$\sum_{n \in \mathbb{N}} y_{n,k} \geq 1, \quad k \in \mathbb{K}, \quad n \in \mathbb{N}$$

(14c)

where (14c) guarantees that at least one vehicle provides computing service for each task. Moreover, since these tasks occupy the computing resources of these vehicles, and the vehicles should be paid some compensation, which can be regarded as the computing cost of tasks. The computing cost can be defined as a function logarithmically related to the vehicle’s computing time, and can be given by:

$$E = u \sum_{k \in \mathbb{K}} \sum_{n \in \mathbb{N}} y_{n,k} \log \left( 1 + t_{n,k}^{\text{comp}} \right)$$

(15a)

$$s.t. y_{n,k} \in \{0, 1\}, \quad k \in \mathbb{K}, \quad n \in \mathbb{N}$$

(15b)

$$\sum_{n \in \mathbb{N}} y_{n,k} \geq 1, \quad k \in \mathbb{K}, \quad n \in \mathbb{N}$$

(15c)

where $u$ is a constant, it can be adjusted according to the task’s tolerance for time delay. If these tasks are very sensitive to delay, that is, the system would like to pay more expenditure for computing service, and then $u$ can be smaller. If these tasks are not sensitive to delay, that is, the system would like to pay fewer expenditure for computing service, and then $u$ can be larger. Next, the system’s consumption can be defined as the weighted sum of $t_{\text{sum}}$ and $E$, and then we formulate the optimization problem as follows:

$$\min_{Y} \max_{k \in \mathbb{K}, n \in \mathbb{N}} \left\{ \min_{n \in \mathbb{N}} \left\{ t_{\text{off}} + \sum_{i=1}^{k} y_{n,i}^{\text{comp}} + (1 - y_{n,k}) Q \right\} \right\} + u \sum_{k \in \mathbb{K}} \sum_{n \in \mathbb{N}} y_{n,k} \log \left( 1 + t_{n,k}^{\text{comp}} \right)$$

(16a)

$$s.t. y_{n,k} \in \{0, 1\}, \quad k \in \mathbb{K}, \quad n \in \mathbb{N}$$

(16b)

$$\sum_{n \in \mathbb{N}} y_{n,k} \geq 1, \quad k \in \mathbb{K}, \quad n \in \mathbb{N}$$

(16c)

III. PROPOSED ALGORITHM FOR PROBLEM

A. PROBLEM TRANSFORMATION

The key challenge in solving this problem is the integer constraint $y_{n,k} \in \{0, 1\}$, which makes problem (16) become an integer non-linear programming problem. Additionally, the min $\{x_1, \ldots, x_n\}$ is a non-convex or non-linear function, so problem (16) is non-convex and NP-hard. In order to solve problem (16), we first transform it into an equivalent form as shown in Lemma 1.

Lemma 1: Problem (16) can be transformed into the following equivalent problem:

$$\min_{Y} \max_{k \in \mathbb{K}, n \in \mathbb{N}} \left\{ \left( t_{\text{off}} + \sum_{i=1}^{k} y_{n,i}^{\text{comp}} + (1 - y_{n,k}) Q \right)^{-1} \right\} + u \sum_{k \in \mathbb{K}} \sum_{n \in \mathbb{N}} y_{n,k}^2 \log \left( 1 + t_{n,k}^{\text{comp}} \right)$$

(17a)

$$s.t. (16b), (16c).$$

(17b)

Proof: First, min $\{x_1, \ldots, x_n\}$ is a non-convex or non-linear function and it can be transformed into a convex function max $\{x_1^{-1}, \ldots, x_n^{-1}\}$. So, we can obtain:

$$\min_{n \in \mathbb{N}} \left\{ t_{\text{off}} + \sum_{i=1}^{k} y_{n,i}^{\text{comp}} + (1 - y_{n,k}) Q \right\}$$

$$= \max_{n \in \mathbb{N}} \left\{ t_{\text{off}} + \sum_{i=1}^{k} y_{n,i}^{\text{comp}} + (1 - y_{n,k}) Q \right\}^{-1}$$

(18)

Note that $\sum_{k \in \mathbb{K}} \sum_{n \in \mathbb{N}} y_{n,k} \log \left( 1 + t_{n,k}^{\text{comp}} \right)$ is linear, and in order to make the convexity of objective function better, we can square the 0-1 variable $y_{n,k}$. And this transformation will not change the value of original objective function, since $y_{n,k}^2 = y_{n,k}$, which can be written as:

$$\sum_{k \in \mathbb{K}} \sum_{n \in \mathbb{N}} y_{n,k} \log \left( 1 + t_{n,k}^{\text{comp}} \right)$$

$$= \sum_{k \in \mathbb{K}} \sum_{n \in \mathbb{N}} y_{n,k}^2 \log \left( 1 + t_{n,k}^{\text{comp}} \right)$$

(19)

Therefore, problem (17) is equivalent to problem (16). □
Considering that problem (17) is a 0-1 integer programming problem, the decision variable \( y_{n,k} \in \{0, 1\} \) can be transformed into a slack variable \( z_{n,k} \in [0, 1] \), and the variable matrix is denoted as \( Z \). Thus, the original problem (16) is relaxed into a continuous variable problem:

\[
\min \sum_{k \in K, n \in N} \max \left\{ t_{n,k}^{\text{off}} + \sum_{i=1}^{k} z_{n,k}^{i} t_{n,i}^{\text{comp}} + (1 - z_{n,k}) Q \right\} \\
+ u \sum_{k \in K, n \in N} \sum_{i=1}^{k} z_{n,k}^{i} \log \left( 1 + t_{n,k}^{\text{comp}} \right)
\]

s.t. \( z_{n,k} \geq 1 \), \( k \in K \), \( n \in N \)

\[
\min \sum_{k \in K, n \in N} \max \left\{ t_{n,k}^{\text{off}} + \sum_{i=1}^{k} z_{n,k}^{i} t_{n,i}^{\text{comp}} + (1 - z_{n,k}) Q \right\} \\
+ u \sum_{k \in K, n \in N} \sum_{i=1}^{k} z_{n,k}^{i} \log \left( 1 + t_{n,k}^{\text{comp}} \right)
\]

s.t. \( z_{n,k} \leq 0 \), \( n \in N \), \( k \in K \)

\[
\min \sum_{k \in K, n \in N} \max \left\{ t_{n,k}^{\text{off}} + \sum_{i=1}^{k} z_{n,k}^{i} t_{n,i}^{\text{comp}} + (1 - z_{n,k}) Q \right\} \\
+ u \sum_{k \in K, n \in N} \sum_{i=1}^{k} z_{n,k}^{i} \log \left( 1 + t_{n,k}^{\text{comp}} \right)
\]

s.t. \( z_{n,k} - 1 \leq 0 \), \( n \in N \), \( k \in K \)

Proof: See Appendix A.

\[\square\]

Lemma 2: Problem (20) can be transformed into a convex optimization problem [33]:

\[
\sum_{k \in K, n \in N} \max \left\{ t_{n,k}^{\text{off}} + \sum_{i=1}^{k} z_{n,k}^{i} t_{n,i}^{\text{comp}} + (1 - z_{n,k}) Q \right\} \\
+ u \sum_{k \in K, n \in N} \sum_{i=1}^{k} z_{n,k}^{i} \log \left( 1 + t_{n,k}^{\text{comp}} \right)
\]

s.t. \( z_{n,k} \leq 0 \), \( n \in N \), \( k \in K \)

Proof: Note that the Maximum function is very difficult to handle, so we can focus on its analytic approximation [33]. In problem (21), the analytic approximation of the Maximum function can be written as:

\[
\max_{k \in K, n \in N} \left\{ t_{n,k}^{\text{off}} + \sum_{i=1}^{k} z_{n,k}^{i} t_{n,i}^{\text{comp}} + (1 - z_{n,k}) Q \right\} \\
\approx \log \left( \sum_{k \in K, n \in N} e^{t_{n,k}^{\text{off}} + \sum_{i=1}^{k} z_{n,k}^{i} t_{n,i}^{\text{comp}} + (1 - z_{n,k}) Q} \right) \]

Therefore, problem (22) is equivalent to problem (21).

\[\square\]

**B. INTERIOR POINT METHOD**

In this article, problem (22) can be solved by using interior point method. First, the penalty function \( \Phi (Z, r) \) [33] can be constructed as:

\[
\Phi (Z, r) = \log \left( \sum_{k \in K, n \in N} e^{t_{n,k}^{\text{off}} + \sum_{i=1}^{k} z_{n,k}^{i} t_{n,i}^{\text{comp}} + (1 - z_{n,k}) Q} \right) \\
+ u \sum_{k \in K, n \in N} \sum_{i=1}^{k} z_{n,k}^{i} \log \left( 1 + t_{n,k}^{\text{comp}} \right)
\]

Next, we set an initial penalty factor \( r \) and select the initial point \( Z \) in feasible region. And then, the extreme point of the penalty function can be obtained by an unconstrained optimization method: (a) use the gradient method, \( z_{n,k} = z_{n,k} + b \frac{\partial \Phi (Z, r)}{\partial z_{n,k}} \), to find the extreme point \( Z \); (b) let \( r = cr \), where \( c \) is the decreasing coefficient; (c) use new \( r \) to update equation (24); (d) iterate (a)-(c) until the margin of error is met.

**Algorithm 1 Joint Optimization of Delay and Cost for Multicast-Oriented Tasks Offloading Algorithm (JDCM)**

1: **Initialization**: Set an initial matrix \( Z \) of problem (22).
2: **repeat**
3: Obtain the penalty function \( \Phi (Z, r) \) according to the current matrix \( Z \) and \( r \), such as (24).
4: Use the gradient method, \( z_{n,k} = z_{n,k} + b \frac{\partial \Phi (Z, r)}{\partial z_{n,k}} \), to find the extreme point \( Z \).
5: Set a decision threshold \( \beta \).
6: if \( z_{n,k} < \beta \) then
7: \( y_{n,k} = 0 \).
8: else if \( z_{n,k} \geq \beta \) then
9: \( y_{n,k} = 1 \).
10: **end if**
11: Set \( r = cr \).
12: **until** convergence.

**C. ALGORITHM DESIGN**

The feasible matrix \( Z \) of problem (22) needs to be converted to a matrix of integers, since each \( z_{n,k} \) is integer (0 or 1, i.e., \( y_{n,k} \in \{0, 1\} \)). According to the slack variable \( z_{n,k} \) obtained by each iteration of the interior point method, we can firstly set a decision threshold \( \beta \), that is, if \( z_{n,k} < \beta \), let \( y_{n,k} = 0 \); if \( z_{n,k} \geq \beta \), let \( y_{n,k} = 1 \), and then, we can get the matrix \( Y \) of original problem. Finally, the optimal matrix \( Y \) can be obtained by using the iteration method above until convergence, which is summarized in Algorithm 1. The time complexity of our algorithm can be calculated as: \( T(n) = 1 + n (NK + NK + 1 + NK) = (3NK + 2)n + 1 \), which could...
be denoted as $O(n)$. It can be seen that the proposed algorithm has the characteristics of low complexity, good convergence and high efficiency. In the scenario of VEC, in order to ensure low latency and high reliability of the system, it is very important to design efficient and low complexity algorithms.

IV. SIMULATION AND DISCUSSION

In this section, we give some simulation results to evaluate the effectiveness of our JDCM algorithm. We consider a one-way road where vehicles move at a constant speed and their positions are approximately uniformly distributed. As shown in Figure 1, RSUs are uniformly distributed along the road and near the side of the slow lane (i.e. lane C). We choose a certain cellular as the starting cellular and the vehicles in the cellular as the target vehicles. RSUs offload the package containing several tasks to these vehicles based on multicast. With the iteration of the algorithm, in order to obtain a low delay with as little computing cost as possible, the vehicles participating in computing are constantly changing, and the system finally reach optimization.

A. SIMULATION PARAMETER

In the simulation scenario, we consider a one-way road without curves and intersections. The road has three lanes, that is, the super-fast lane, the fast lane, and the slow lane, which are respectively written as lane A, lane B, lane C, and the width of each lane is 3.6 m. We set the number of vehicles 10, and the number of vehicles in each lane is 2 (lane A), 3 (lane B), 5 (lane C) and these vehicles are uniformly distributed. The vehicle keeps driving at a constant speed, namely $v_{laneA}=20m/s$, $v_{laneB}=15m/s$, $v_{laneC}=10m/s$. Without loss of generality, we assume that the vehicles will not change lanes during a short period of time, that is, it will not change its driving state during the progress of task offloading. The computation resources of these vehicles are denoted as $F = \{f_1, f_2, \ldots, f_N\}$, and subject to uniform distribution in the range of (50,100).

The main simulation parameters are shown in Table 1.

B. EVALUATION OF SIMULATION RESULTS

In this section, we will verify the performance of the JDCM algorithm proposed in Algorithm 1, and compare it with other methods to evaluate its superiority.

Figure 2 compare the performance under different offloading schemes. As shown in this figure, the task consumption is rapidly reduced by using the JDCM algorithm proposed in this article. It converges within 6 iterations and eventually obtains the same optimal solution as the Exhaust algorithm. Compared with the JDCM algorithm in this article, the classic Genetic algorithm shows poor performance and it converge slowly and easily fall into local optimization.

As shown in Figure 3, our algorithm has very good performance with different vehicles’ speed. And, speed will affect the transmission delay of tasks, that is, proper increase in speed may increase transmission efficiency, but too fast speed is counterproductive [18].

Next, we consider the performance of the JDCM algorithm when the number of vehicles increases, that is, the vehicle density increases. From Figure 4 we can see that the performance of the JDCM algorithm is still superior. Due to the increasing dimension of solution space, the convergence speed of our algorithm may finitely decrease. Compared with JDCM algorithm, the other algorithms are very poor, and they are easy to fall into local optimization.

TABLE 1. Simulation parameters.

| Parameters | Description | Value |
|------------|-------------|-------|
| $W$        | Bandwidth of RSUs for each vehicle | 1MHz |
| $\alpha$   | Pass-loss exponent of communication | 4 |
| $N_0$      | Noise power | $10^{-15}$W |
| $P_{RSU}$  | Transmitted power of RSUs | 2W |
| $\psi_{lane}$ | Width of each lane | 3.6m |
| $l$        | Length of a cellular covered by RSUs | 100m |
| $K$        | Number of tasks | 3 |
| $x_k$      | Data size of task $k$ | 100M |
| $\lambda$  | Computing intensity of tasks | 1 |
| $f$        | Computation resources of vehicles | U(50,100) |
| $N$        | Number of vehicles | 10 |
| $u$        | Coefficient | 1 |
| $M$        | Number of RSUs | 20 |
| $v_{laneA}$ | Speed of vehicles in lane A | 20m/s |
| $v_{laneB}$ | Speed of vehicles in lane B | 15m/s |
| $v_{laneC}$ | Speed of vehicles in lane C | 10m/s |
| $d$        | Length of a time slot | 0.1s |

As shown in Figure 3, our algorithm has very good performance with different vehicles' speed. And, speed will affect the transmission delay of tasks, that is, proper increase in speed may increase transmission efficiency, but too fast speed is counterproductive [18].

Next, we consider the performance of the JDCM algorithm when the number of vehicles increases, that is, the vehicle density increases. From Figure 4 we can see that the performance of the JDCM algorithm is still superior. Due to the increasing dimension of solution space, the convergence speed of our algorithm may finitely decrease. Compared with JDCM algorithm, the other algorithms are very poor, and they are easy to fall into local optimization.

In Figure 5, we verify the effectiveness of our algorithm with different numbers of tasks. As shown in Figure 5, the results obtained by JDCM algorithm in this article are very close to the optimal solution. In addition, as the number of tasks increases, the performance of JDCM algorithm remains superior. On the contrary, the performance of Genetic algorithm becomes worse as the dimension increasing.
Finally, we compare the performance with different computing intensity of tasks. As shown in Figure 6, the results obtained by JDCM algorithm in this article are very close to the optimal solution. From this figure, we observe that as the computing density of tasks increases, the consumption of tasks increases linearly. Compared with JDCM algorithm in this article, the classic Genetic algorithm shows poor performance as it is easy to fall into local optimization.

Without loss of generality, we consider the scenario that the instantaneous speed of each vehicle changes with respect to time and subjects to the Gaussian distribution with the average value is 20(m/s). As shown in Figure 7, the algorithm we proposed still has superior performance as vehicles driving at variable speeds.

Based on the results above, JDCM has the characteristics of low complexity, excellent convergence, and high efficiency, which can effectively reduce the task consumption. Therefore, we believe that the algorithm proposed in this article can solve problem (16) with superior performance.

V. CONCLUSION

In this article, a novel vehicle edge computing model is proposed, and the joint optimization problem of delay and...
cost for task offloading is proposed in this model. In VeFN, we used a multicast-oriented method to offload a package with multiple tasks to vehicles on the road. Vehicles can choose to compute one or several tasks in this multicast package, and even not compute tasks. With the JDCM algorithm proposed in this article, task consumption is optimized, and some vehicles do not provide computing services due to high task delay or computing cost. Numerical results show that this scheme has superior performance, which can significantly reduce task consumption compared with other schemes. In future work, we will consider more general scenarios, such as more complex roads, direction changes of vehicles, and timeliness of tasks, and in this case, the model can be transformed into a stochastic model, which remains to be further explored.

APPENDIX A
PROOF OF LEMMA 2

Note that, there are $N \times K$ subfunctions in the maximum function in (20a) and we can firstly let:

$$h_{n,k}(z_n) = \left( t_{\text{off}} + \sum_{i=1}^{k} z_{n,i} t_{\text{comp}}^{\text{off}} + (1 - z_{n,k}) Q \right)^{-1}$$

where $h_{n,k}(z_n)$ is a function of the vector $z_n$, and $z_n$ is the $n$th row of the matrix $Z$. Thus, $\frac{\partial^2 h_{n,k}(z_n)}{\partial z_{n,k}}, \frac{\partial^2 h_{n,k}(z_n)}{\partial z_{n,j}} |_{j<k}$ and $\frac{\partial^2 h_{n,k}(z_n)}{\partial z_{n,k} \partial z_{n,j}} |_{j<k}$ can be respectively expressed as:

$$\frac{\partial^2 h_{n,k}(z_n)}{\partial z_{n,k}} = 2 \left( t_{\text{off}} + \sum_{i=1}^{k} z_{n,i} t_{\text{comp}}^{\text{off}} + (1 - z_{n,k}) Q \right)^{-3}$$

$$\frac{\partial^2 h_{n,k}(z_n)}{\partial z_{n,j}} |_{j<k} = 2 \left( t_{\text{off}} + \sum_{i=1}^{k} z_{n,i} t_{\text{comp}}^{\text{off}} + (1 - z_{n,k}) Q \right)^{-3}$$

$$\frac{\partial^2 h_{n,k}(z_n)}{\partial z_{n,k} \partial z_{n,j}} |_{j<k} = 2 t_{\text{off}} t_{\text{comp}}^{\text{off}} \left( t_{\text{off}} + \sum_{i=1}^{k} z_{n,i} t_{\text{comp}}^{\text{off}} + (1 - z_{n,k}) Q \right)^{-3}$$

Due to space constraints, for convenience, we let

$$a_k = \frac{t_{\text{off}}}{t_{\text{off}} + \sum_{i=1}^{k} z_{n,i} t_{\text{comp}}^{\text{off}} + (1 - z_{n,k}) Q}$$

$$b = 2 \left( t_{\text{off}} + \sum_{i=1}^{k} z_{n,i} t_{\text{comp}}^{\text{off}} + (1 - z_{n,k}) Q \right)^{-3}$$

Then, the Hessian matrix of $h_{n,k}(z_n)$ is:

$$\nabla^2 h_{n,k}(z_n) = \begin{bmatrix} a_1 a_1 & \cdots & a_k a_1 & (a_1 - Q) \\ \vdots & \ddots & \vdots & \vdots \\ a_1 a_{k-1} & \cdots & a_k a_{k-1} & a_{k-1} (a_1 - Q) \\ a_1 (a_1 - Q) & \cdots & a_k (a_1 - Q) & (a_1 - Q)^2 \end{bmatrix}$$

Since $b > 0$, we can only pay attention to the matrix in the above formula, and we let

$$\mathbf{S} = \begin{bmatrix} a_1 a_1 & \cdots & a_k a_1 & (a_1 - Q) \\ \vdots & \ddots & \vdots & \vdots \\ a_1 a_{k-1} & \cdots & a_k a_{k-1} & a_{k-1} (a_1 - Q) \\ a_1 (a_1 - Q) & \cdots & a_k (a_1 - Q) & (a_1 - Q)^2 \end{bmatrix}$$

And then we set $\mathbf{x} = (a_1, a_2, \cdots, a_k - Q)^T$, so matrix $\mathbf{S}$ can be expressed as $\mathbf{S} = \mathbf{x} \mathbf{x}^T$. Let $\lambda$ be any one of the characteristic values of matrix $\mathbf{S}$, and $\mathbf{g}$ be the characteristic vector. So, we have $\mathbf{Sg} = \lambda \mathbf{g}$, and then we can get:

$$\mathbf{Sg} = \lambda \mathbf{g}$$

$$\mathbf{x} \mathbf{x}^T \mathbf{g} = \lambda \mathbf{g}$$

$$\mathbf{g}^T \mathbf{x} \mathbf{x}^T \mathbf{g} = \lambda \mathbf{g}^T \mathbf{g}$$

$$\mathbf{g}^T \mathbf{g} = \lambda \mathbf{g}^T \mathbf{g} \geq 0$$

So $\lambda \geq 0$, and the matrix $\mathbf{S}$ is a positive semidefinite matrix, that is $\nabla^2 h_{n,k}(z_n) \succeq 0$ (in other words, $h_{n,k}(z_n)$ are all convex functions). According to the operations that preserve convexity, the pointwise maximum function of multiple convex functions is convex [33], so $\max_{k \in K, n \in N} \max_{k} h_{n,k}(z_n)$ is a convex function. Obviously, $z_{n,k} \log \left( 1 + t_{\text{comp}}^{\text{off}} \right)$ is a typical convex function, since it is a quadratic function, and nonnegative weighted sums are operations that preserve convexity, so eq. (20a) is a convex function. Constraint (20b) and (20c) can be transformed into two linear inequalities and one affine inequality constraint, such as (21b) (21c) (21d). In summary, problem (20) is convex.

REFERENCES

[1] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, “A survey on mobile edge networks: Convergence of computing, caching and communications,” IEEE Access, vol. 5, pp. 6757–6779, Mar. 2017.

[2] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” IEEE Commun. Surveys Tuts., vol. 19, no. 4, pp. 2322–2358, Mar. 2017.

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

[4] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, “Mobile-edge computing for vehicular networks: A promising network paradigm with predictive offloading,” IEEE Veh. Technol. Mag., vol. 12, no. 2, pp. 36–44, Jun. 2017.
[5] X. Huang, R. Yu, J. Kang, and Y. Zhang, “Distributed reputation management for secure and efficient vehicular edge computing and networks,” IEEE Access, vol. 5, pp. 25408–25420, 2017.

[6] S. Wang, X. Huang, R. Yu, Y. Zhang, and E. Hossain, “Permissioned blockchain for efficient and secure resource sharing in vehicular edge computing,” 2019, arXiv:1906.06319. [Online]. Available: http://arxiv.org/abs/1906.06319.

[7] J. Zhang, X. Huang, and R. Yu, “Optimal task assignment with delay constraint for parked vehicle assisted edge computing: A stackelberg game approach,” IEEE Commun. Lett., vol. 24, no. 3, pp. 598–602, Mar. 2020.

[8] X. Huang, P. Li, and R. Yu, “Social welfare maximization in container-based task scheduling for parked vehicle edge computing,” IEEE Commun. Lett., vol. 23, no. 8, pp. 1347–1351, Aug. 2019.

[9] C. Li, S. Wang, X. Huang, X. Li, R. Yu, and F. Zhao, “Parked vehicular computing for energy-efficient Internet of Vehicles: A contract theoretic approach,” IEEE Internet Things J., vol. 6, no. 4, pp. 6079–6088, Aug. 2019.

[10] F. Wang, J. Xu, X. Wang, and S. Cui, “Joint offloading and computing optimization in wireless powered mobile-edge computing systems,” IEEE Trans. Wireless Commun., vol. 17, no. 3, pp. 1784–1797, Mar. 2018.

[11] F. Wang, F. Xu, and S. Cui, “Optimal energy allocation and task offloading policy for wireless powered mobile edge computing systems,” IEEE Trans. Wireless Commun., vol. 19, no. 4, pp. 2443–2459, Apr. 2020.

[12] F. Wang, H. Xing, and J. Xu, “Optimal resource allocation for wireless powered mobile edge computing with dynamic task arrivals,” in Proc. IEEE Int. Conf. Commun. (ICC), Shanghai, China, May 2019, pp. 1–7.

[13] F. Wang, J. Xu, and Z. Ding, “Multi-antenna NOMA for computing offloading in multiuser mobile edge computing systems,” IEEE Trans. Commun., vol. 67, no. 3, pp. 2450–2463, Mar. 2019.

[14] J. Ni, A. Zhang, X. Lin, and X. S. Shen, “Security, privacy, and fairness in fog-based vehicular crowdsensing,” IEEE Commun. Mag., vol. 55, no. 6, pp. 146–152, Jun. 2017.

[15] J. Ni, X. Lin, and X. S. Shen, “Toward edge-assisted Internet of Things: From security and efficiency perspectives,” IEEE Netw., vol. 33, no. 2, pp. 50–57, Mar. 2019.

[16] M. Wang, M. Ismail, R. Zhang, X. Shen, E. Serpedin, and K. Qaraqe, “Spatio-temporal coordinated V2V energy swapping strategy for mobile PEVs,” IEEE Trans. Smart Grid, vol. 9, no. 3, pp. 1566–1579, May 2018.

[17] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen, “Cooperative task scheduling for computation offloading in vehicular cloud,” IEEE Trans. Veh. Technol., vol. 67, no. 11, pp. 11049–11061, Nov. 2018.

[18] S. Zhou, Y. Sun, Z. Jiang, and Z. Niu, “Exploiting moving intelligence: Deep reinforcement learning for vehicular fog computing,” IEEE Trans. Veh. Technol., vol. 67, no. 5, pp. 49–55, May 2019.

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

[20] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, “Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach,” IEEE Trans. Veh. Technol., vol. 68, no. 4, pp. 3113–3125, Apr. 2019.

[21] N. Chen, Y. Yang, T. Zhang, M.-T. Zhou, X. Luo, and J. K. Zao, “Fog as a service technology,” IEEE Commun. Mag., vol. 56, no. 11, pp. 95–101, Nov. 2018.

[22] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, “Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks,” IEEE Access, vol. 4, pp. 5896–5907, 2016.

[23] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, “Optimal delay constrained offloading for vehicular edge computing networks,” in Proc. IEEE Int. Conf. Commun. (ICC), Paris, France, May 2017, pp. 1–6.

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

[25] Y. Sun, J. Song, S. Zhou, X. Guo, and Z. Niu, “Task replication for vehicular edge computing: A combinatorial multi-armed bandit based approach,” in Proc. IEEE Global Commun. Conf. (GLOBECOM), Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–7.

[26] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, “Learning-based task offloading for vehicular cloud computing systems,” in Proc. IEEE Int. Conf. Commun. (ICC), Kansas City, MO, USA, May 2018, pp. 1–7.

[27] H. Li, X. Li, and W. Wang, “Joint optimization of computation cost and delay for task offloading in vehicular fog networks,” Trans.Emerg. Telecommun. Technol., vol. 31, no. 2, pp. 1–16, Feb. 2020.