




Computation Offloading and Resource Allocation For Cloud Assisted Mobile Edge Computing in Vehicular Networks

Junhui Zhao , Senior Member, IEEE, Qiuping Li, Yi Gong , Senior Member, IEEE, and Ke Zhang 

Abstract—Computation offloading services provide required computing resources for vehicles with computation-intensive tasks. Past computation offloading research mainly focused on mobile edge computing (MEC) or cloud computing, separately. This paper presents a collaborative approach based on MEC and cloud computing that offloads services to automobiles in vehicular networks. A cloud-MEC collaborative computation offloading problem is formulated through jointly optimizing computation offloading decision and computation resource allocation. Since the problem is non-convex and NP-hard, we propose a collaborative computation offloading and resource allocation optimization (CCORAO) scheme, and design a distributed computation offloading and resource allocation algorithm for CCORAO scheme that achieves the optimal solution. The simulation results show that the proposed algorithm can effectively improve the system utility and computation time, especially for the scenario where the MEC servers fail to meet demands due to insufficient computation resources.

Index Terms—Collaborative computation offloading, computation resource allocation, mobile edge computing, vehicular networks.

I. INTRODUCTION

WITH the development of intelligent vehicles, prosperous computation-intensive applications, such as augmented

reality (AR) [1], autonomous driving (AD) [2], speech recognition [3], and natural language processing [4], are employed for assisting both drivers and passengers in a vehicular environment [5], [6]. Completing those applications with rigorous delay constraints usually needs to consume extensive computational resources [7], [8]. However, the constrained computing capability of the vehicles is usually difficult to meet the computation requirements of the applications [9]–[11]. The cloud computing with high-powered computing capability has served as an effective approach to solving the conflict between resource-intensive applications and resource-constrained vehicles, where the tasks can be offloaded to the remote cloud servers from a vehicle [12]–[14]. The work in [12] minimized the total power consumption of cloud computing in a vehicular cloud network. Yu *et al.* evaluated the cloud resource allocation for an effective resource management [13]. The authors developed a fully polynomial time approximation scheme and a greedy approximation for scheduling a sole task to minimize the total cost in [14]. Although cloud computing can extend the computing capacity of vehicles, the long distance data transmission and backhaul of the network with a huge amount of data will cause some potential problems, e.g. unacceptable latency and unstable connections, when the vehicles offload the computing tasks to the cloud computing servers. These deficiencies lead to the difficulties in achieving the low-latency requirements of the novel applications.

To supplement cloud computing to fully address these challenges, MEC has been conducted in the academia and industry [3]. The MEC shifts the cloud services to the radio access network (RAN) and provides the cloud-computing capability in close proximity to mobile users [15]. As a result, this can further reduce the delay and ensure high bandwidth connection in the computation offloading process [16]. Moreover, many studies involved in the MEC computation offloading in cellular networks have been proposed in recent years. The technology specifically focuses on enhancing the system performance gain, such as reducing the energy consumption or system cost by optimizing offloading decision and allocating the resources effectively [17]–[19]. By jointly optimizing the computation offloading strategy and resource allocation, the literature [17] minimized the overall cost of energy, computation, and delay for all users, and the energy consumption of users was minimized in [18] and [19].

Manuscript received November 13, 2018; revised March 14, 2019; accepted April 23, 2019. Date of publication June 25, 2019; date of current version August 13, 2019. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2018YJS008, in part by the National Natural Science Foundation of China under Grant 61661021, in part by Beijing Natural Science Foundation under Grant L182018, in part by the Open Research Fund of National Mobile Communications Research Laboratory, Southeast University under Grant 2017D14, in part by Science and Technology Program of Jiangxi Province under Grants 20172BCB22016 and 20171BBE50057, in part by the Shenzhen Science and Technology Program under Grant JCYJ20170817110410346, and in part by Peng Cheng Laboratory under Grant PCL2018KP002. The review of this paper was coordinated by Prof. P. Lorenz. (Corresponding author: Yi Gong.)

J. Zhao is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China, and also with the School of Information Engineering, East China Jiaotong University, Nanchang, China (e-mail: junhui Zhao@hotmail.com).

Q. Li is with the School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China, and also with Peng Cheng Laboratory, Shenzhen, China (e-mail: qiupingli@bjtu.edu.cn).

Y. Gong is with the Shenzhen Engineering Laboratory of Intelligent Information Processing for IoT, Southern University of Science and Technology, Shenzhen, China (e-mail: gongy@sustech.edu.cn).

K. Zhang is with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, China (e-mail: zhangke@uestc.edu.cn).

Digital Object Identifier 10.1109/TVT.2019.2917890

Intelligent vehicles are regarded as one of the important platforms to implement computing applications. The computation offloading under vehicular networks has gained widespread popularity recently [2], [20]–[25]. An optimal predictive combination-mode offloading scheme was proposed for the vehicular networks [20]. The authors in [21] recommended a distributed reputation management system to provide security protection and enhance network efficiency in the implementation of vehicular edge computing. In [22], the authors designed a contract-based computation offloading scheme in cloud-enabled vehicular networks, which maximized the benefit of the MEC service provider by optimizing the allocation of the computing resource. An efficient service delivery mechanism was designed in an online vehicles MEC environment [23]. The authors in [24] proposed an efficient distributed computation offloading algorithm to enable the optimal offloading decisions for vehicles in vehicular networks. To ensure the best use of underutilized vehicular computational resources, Feng *et al.* proposed an autonomous vehicular edge (AVE) framework in managing the idle computational resources of vehicles [2]. Hou *et al.* viewed slow-moving and parked vehicles as infrastructure in providing computational resources for vehicles with computation-intensive tasks in the vehicular edge computing [25].

Besides, considering the game-theory can effectively address the decision-making concerns among the multiple rational game players contrasting the goal, the research of computation offloading based on game theory has been discussed in [4], [26]–[30]. Yu *et al.* proposed a coalition game-based model for resource sharing among the different cloud service providers [26]. In [27] and [28], a game-theoretic analysis of a computation offloading for multiuser MEC systems was proposed, and a distributed algorithm was designed to achieve a Nash equilibrium (NE). The authors in [29] minimized the system cost through the evolutionary game. In [4], an optimal multilevel offloading scheme based on Stackelberg game theoretic was proposed, and the revenue of both vehicles and computing servers were maximized. The authors in [30] proposed a Bayesian coalition game to reduce energy consumption and improve the utilization of the computing resource in a vehicular mobile cloud.

However, aforementioned research mainly consider migrating computation tasks from the vehicle to the MEC server [4], [20]–[22] or cloud server [12]–[14], [26], [30], and few consider utilizing the MEC and cloud computing resources simultaneously. Compared to cloud computing, the computing resources of the MEC server are limited and it may be incapable of big data processing. Among the available affluent computation-intensive applications, the limited computing resources of MEC servers cannot fully attain all computation offloading requirements. In particular, this occurs when the number of computation tasks keeps increasing, while the resource bottleneck of MEC servers becomes increasingly prominent. On the other hand, although the cloud computing may experience long latency in the offloading processing, cloud computing can provide the adequate cloud computing resources. The cloud computing and MEC should be highly complementary. Therefore, designing a proper collaborative computation offloading scheme among the vehicles

to ensure full use of the MEC and cloud computing resources is of significant interest.

Meanwhile, some previous studies sought to optimize the computation offloading strategy or the computing resource allocation without optimizing both. Similar to the works in [10], [27]–[29], they only evaluated computation offloading strategy and did not include the optimal computation resource allocation, which was simplified, as each vehicle usually gets diverse computation tasks that require different computation resources in real life. Moreover, the tasks in [22] and [26] disregarded the optimization of computation offloading strategy, where all computation tasks were offloaded to the MEC or cloud server. The limited computing resources and communication resources are shared among the vehicles in the computation offloading, where each vehicle has to identify where to process its task and whether offloading is necessary at all. Hence, the computation offloading strategy and resource allocation should be optimized to obtain an improved system performance gain.

In filling in the gap, this paper evaluated the collaboration of cloud computing and MEC, including the impact of vehicle speed on the computation offloading scheme to design a collaborative computation offloading scheme for a cloud assisted-MEC in vehicular networks. The contributions of this paper are summarized as follows:

- This paper puts forward cloud assisted MEC for computation offloading optimization problem in vehicular networks. The stated problem seeks to maximize the system utility by optimizing both the offloading strategy and resource allocation under the delay constraints of the tasks. The system utility is defined based on the three metrics such as task processing delay, cost of computation resource, and the normalization factor.
- A collaborative computation offloading and resource allocation optimization (CCORAO) scheme is proposed to decouple the optimization problem into two of the sub-problems of computation offloading decision-making and resource allocation, where the offloading decisions are taken through game-theoretic approach. The resource allocation is achieved by using the Lagrange multiplier method. Moreover, the existence of Nash equilibrium (NE) is proven by introducing the definition of exact potential game.
- A distributed computation offloading and resource allocation (DCOR) algorithm is developed for the proposed scheme. It works iteratively between offloading decisions and resource allocation to obtain the NE, the optimal computation offloading strategy and computation resource allocation. The numerical results demonstrate that the proposed scheme outperforms the current approaches, which distinctly enhances the system utility and reduces the task processing delay.

The remainder of this paper is structured as follows. The system model and the total utility optimization problem are presented in Section II. The collaborative computation offloading and resource allocation optimization scheme and the algorithm designing for the proposed scheme are described in Section III.

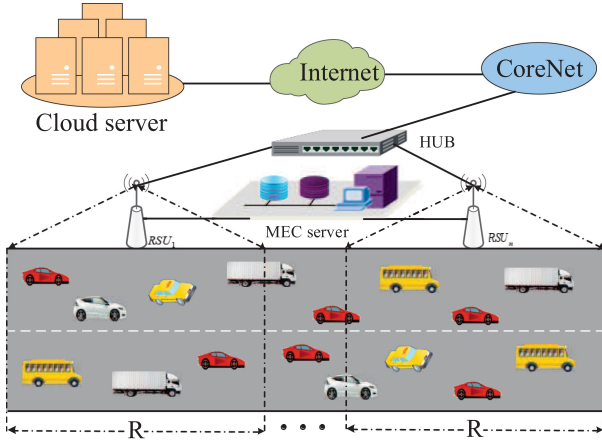


Fig. 1. The system model.

The simulation results are shown in Section IV. Finally the conclusion is presented in Section V.

II. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Model

In Fig. 1, a network is used, consisting of N vehicles, M roadside units (RSUs), a MEC server with the total computing resource F , and a cloud computing server. Here, the RSUs located along the highway have the same coverage range R , whose id set is denoted as $\mathcal{M} = \{1, 2, \dots, M\}$. Therefore, the research road segment can be divided into M segments, and all vehicles random distribution in the research highway segment. Then the relation between the number of vehicles and the length of the research road segment can be obtained, which is identified in the latter part. Moreover, these RSUs are linked to MEC and cloud servers through the fiber-wired link. The $\mathcal{N} = \{1, 2, \dots, N\}$ is denoted as a set of vehicles. Each vehicle has a computation task $\varphi_i = \{C_i, D_i, t_i^{\max}\}$, where $C_i = \bar{h}_i D_i$, D_i is the size of input data for the computation, C_i is the required computation resource to complete task φ_i , \bar{h}_i is the service coefficient describing the relationship between C_i and D_i , and t_i^{\max} is the tolerable maximum latency for the task completion.

Each computation task can either be processed locally, or offloaded to the MEC server, or accomplished on the cloud server. The vehicles offload their computation tasks to the MEC or cloud servers through RSU. It can be noted that the vehicles can only access to RSU m for further offloading their tasks when the vehicles run within the m th segment. Moreover, the wireless access technique between a vehicle and an RSU is generally based on IEEE 802.11p. The IEEE 802.11p provides a high-speed vehicle-to-vehicle (V2V) and vehicle-to-RSU (V2R) communication, which works at the 5.9 GHz band (5.85-5.925GHz). At the same time, other related studies noted V2R communications of the IEEE 802.11p have been held in the academic. In [31], the connectivity characteristics of V2R for platoon-based IEEE 802.11p VANET was investigated. The authors evaluated the V2R connectivity to provide intelligent transportation system

(ITS) applications, such as the curve speed and traffic signal violation warnings, in IEEE 802.11p VANET [32]. The authors in [33] analyzed the computation offloading problem based on IEEE 802.11p. Therefore, the computation task is offloaded from the vehicles to MEC or cloud servers based on IEEE 802.11p in this paper.

We then define $S = \{s_i | s_i \in \{s_i^{loc}, s_i^{mec}, s_i^{ccc}\}, s_i^j \in \{0, 1\}, i \in \mathcal{N}, j \in \{loc, mec, ccc\}\}$ as the offloading strategy of vehicles. Here $\mathcal{A} = \{loc, mec, ccc\}$ denotes the offloading decision set, which computes locally, offloads to MEC server and offloads to cloud server, respectively. Moreover, $s_i = s_i^j = 1$ shows that the task of vehicle i is completed by selecting decision j , otherwise $s_i = s_i^j = 0$.

When $s_i^{mec} = 1$ or $s_i^{ccc} = 1$, it can be assumed that $t_i^{sen} = \min\{t_i^{\max}, t_i^m\}$ is actual delay tolerance of a task. This ensures that the task can be completed when the vehicles shift from connecting the RSU to another adjacent unit and the computation task delay is satisfied. Here, t_i^m is the time that vehicle i left the linking RSU m and can be expressed as

$$t_i^m = \frac{Rm - d_i}{u_v}, \quad (1)$$

where d_i is the position of vehicle i , m denotes vehicle i running within the m th segment, $m = \lceil \frac{d_i}{R} \rceil$, $\lceil \cdot \rceil$ is the ceiling function, and μ_v is the vehicular velocity, which is detailed in following.

Initially, it has widely been accepted in the vehicle traffic theory where the speeds follow a Gaussian distribution. Then, a truncated version of this distribution can be adapted to ensure the research is more appropriate with the practical environment (i.e. avoid dealing with the negative speeds) [34]. That is, the Truncated Gaussian distribution [35] is used to describe the vehicular velocity. Moreover, the Truncated Gaussian probability density function (PDF) is denoted as

$$\tilde{f}_v(v) = \frac{f_v(v)}{\int_{V_{\min}}^{V_{\max}} f_v(s) ds} = \frac{2f_v(v)}{\text{erf}(\frac{V_{\max}-\mu}{\sigma\sqrt{2}}) - \text{erf}(\frac{V_{\min}-\mu}{\sigma\sqrt{2}})}, \quad (2)$$

where $f_v(v) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(v-\mu)^2}{2\sigma^2})$ is the Gaussian PDF, μ and σ are the average speed and standard deviation of the vehicular speed, respectively, $\text{erf}(\cdot)$ is error function, and $V_{\max} = \mu + 3\sigma$ and $V_{\min} = \mu - 3\sigma$ are the maximal and minimal vehicular velocities, respectively.

Therefore, based on (2), an equivalent speed ($V_{\min} \leq \mu_v \leq V_{\max}$) is given by

$$\mu_v = \frac{1}{\int_{V_{\min}}^{V_{\max}} \frac{\tilde{f}_v(v)}{v} dv} = \frac{\text{erf}(\frac{V_{\max}-\mu}{\sigma\sqrt{2}}) - \text{erf}(\frac{V_{\min}-\mu}{\sigma\sqrt{2}})}{\frac{2}{\sigma\sqrt{2\pi}} \int_{V_{\min}}^{V_{\max}} \frac{\exp(-\frac{(v-\mu)^2}{2\sigma^2})}{v} dv}. \quad (3)$$

It is assumed that L is the length of the research road segment while β is the arrival rate of the vehicles. Based on the vehicle traffic and probability theory, the number of vehicles at the

TABLE I
 NOTATION DEFINITIONS

Symbol	Description
m	the RSU index $m \in \mathcal{M}$
i	the vehicle index $i \in \mathcal{N}$
j	the processing model index $j \in \mathcal{A}$
R	the coverage range of RSU
μ_v	the equivalent speed
μ, σ	the average speed / the variance of the vehicular speed
\mathcal{M}, M	set / number of RSUs
\mathcal{N}, N	set / number of vehicles
C_i, D_i, t_i^{\max}	required computing resource / the size of input data/ tolerable maximum latency of φ_i
$s_i^{loc}, s_i^{mec}, s_i^{ccc}$	offloading decision of vehicle i
β	the arrival rate of vehicles
F	total computing resource of the MEC server
$t_i^{loc}, t_i^{mec}, t_i^{ccc}$	total duration time for processing task i in local / MEC / cloud computing
$u_i^{loc}, u_i^{mec}, u_i^{ccc}$	the utility of vehicle i in local / MEC / cloud computing processing
f_i^{loc}	the computing resource of vehicle i
f_i^{mec}, f_i^{ccc}	the computing resource of MEC / cloud allocated to vehicle i
p_m, p_c	computation resources price coefficient of MEC / cloud
γ_i	available transmission rate of vehicle i
τ	transmission delay from RSU to cloud server
\mathcal{S}	the offloading strategy set of vehicles
d_i	the positions of vehicle i
γ	the maximum rate of RSU
p_i^t	the successful transmission probability between vehicle and RSU
p_i^c	the collision probability
T_i^t	the successful transmission period
t_i^{sen}	the actual delay tolerance of computation task
t_i^n	the connection time between vehicle i and RSU m
$AIFS$	the interval of arbitration inter-frame spacing
RTS	request to send interval
θ_i	the weight coefficient

research road segment can be expressed as

$$N = \frac{L\beta}{\mu_v} = \frac{2L\beta}{\sigma\sqrt{2\pi}} \frac{\int_{V_{\min}}^{V_{\max}} \frac{\exp(-\frac{(v-\mu)^2}{2\sigma^2})}{v} dv}{\text{erf}(\frac{V_{\max}-\mu}{\sigma\sqrt{2}}) - \text{erf}(\frac{V_{\min}-\mu}{\sigma\sqrt{2}})}. \quad (4)$$

For the clarity of the following analysis, we show notations mainly used in this paper and their description in Table I. After that, based on the proposed system, the task total duration time can be analyzed for the task in local processing, MEC and cloud computing.

B. The Total Duration Under Different Offloading Decisions

Local processing model: When vehicle i process its computation task locally, the computational time is dependent on its own computing resource. It is assumed that f_i^{loc} is the computing resource of vehicle i , and this is determined by on-board unit (OBU) placed at the vehicle, and taken via offline measurement [2], [4]. Then, the local computational time can be represented as

$$t_i^{loc} = \frac{C_i}{f_i^{loc}}. \quad (5)$$

MEC processing model: The task is conducted by offloading to the MEC server in close proximity ($s_i^{mec} = 1$). Like any other studies [4], [22], many applications such as speech recognition, the time of receiving the computation results is negligible, where the size of the computation results is much smaller than the size of the input data. In this case, the total duration time specifically involves the computation execution time and uplink transmission time. The total duration time of vehicle i is shown as

$$t_i^{mec} = \frac{C_i}{f_i^{mec}} + \frac{D_i}{\gamma_i}, \quad (6)$$

where f_i^{mec} is the amount of computation resource that the MEC assigns to vehicle i , γ_i is the available transmission rate of vehicle i accessing to the RSU, and $\gamma_i = \min\{\gamma, \gamma'_i\}$, γ is the maximum rate of RSU, and γ'_i is the uplink data rate of vehicle i . Moreover, similar to [33], the wireless technology in the vehicular environment is widely used in the contention based on the IEEE 802.11p standard, while γ'_i is represented as

$$\gamma'_i = \frac{D_i p_i^t}{(1 - \varsigma_i)^{N_m} \delta + T_i^t p_i^t + p_i^c (RTS + AIFS + \delta)}. \quad (7)$$

In (7), δ is propagation delay, RTS is Request To Send interval, $AIFS$ is the interval of arbitration inter-frame spacing, ς_i is the probability that vehicle i offload its task to the cloud or MEC server, N_m is the number of vehicles, which is designed offload the tasks under the coverage of m th RSU, p_i^t is successful transmission probability between vehicle and RSU, p_i^c is the collision probability and T_i^t is the successful transmission period. p_i^c can be expressed as

$$p_i^c = 1 - (1 - \varsigma_i)^{N_m} - N_m \varsigma_i (1 - \varsigma_i)^{N_m - 1}. \quad (8)$$

Moreover, p_i^t and T_i^t can be formulated respectively as

$$p_i^t = N_m \varsigma_i (1 - \varsigma_i)^{N_m - 1} \quad (9)$$

$$T_i^t = \Phi + \frac{D_i}{\omega_m \log(1 + P_i h_{im})} \quad (10)$$

where Φ is specific to the MAC protocol, ω_m is the bandwidth of RSU m , h_{im} denotes the channel gain between vehicles i and RSU m , and P_i is the transmission power of vehicle i .

Cloud computing processing model: For the offloading decision ccc ($s_i^{ccc} = 1$), vehicle i will offload its computation task to the cloud server for thousands of miles away via the fiber and core networks. Hence, the uploading time for transmitting the input data from the RSU to the cloud server has to be considered. Moreover, although the amount of output data is much smaller than the input data, the downloading time for sending back the outcome from the cloud server to RSU is also non-negligible. In this case, the total duration time of vehicle i is expressed as

$$t_i^{ccc} = \frac{D_i}{\gamma_i} + \frac{C_i}{f_i^{ccc}} + (D_i + D_i^o) \tau, \quad (11)$$

where τ is the transmission delay from the RSU to the cloud server, D_i^o is the output data size, and f_i^{ccc} is the computation resource of cloud server allocated to vehicle i .

C. Utility Function Under Different Offloading Decisions

In this sub-section, a logic utility function is defined. It is used to quantify the level of satisfaction, which is taken by a vehicle as a result of its computation offloading decision. At the same time, the utility of vehicle i is identified by considering the following metrics.

- Task processing delay. Compared to energy consumption, the task processing delay is a more critical metric for vehicles. In general, the utility function should monotonically decrease with the increased task processing delay. Since the task processing delay is shorter, the vehicle may obtain a higher satisfaction. Moreover, the satisfaction of the vehicle in terms of task processing delay should be non-negative.
- Cost of computation resource. Similar to the existing works [4], [17], [21], [25], each vehicle with offloading its computation task to the MEC or cloud computing server needs to pay for the service. Moreover, as a fact of matter, the vehicle that uses more computing resource will lead to higher cost.
- Normalization factor. Aside from MEC and cloud computing processing, the vehicle can process its own computation task by using its computation resource in the local processing, where the vehicle does not need to pay for the service. However, when the task processing delay of local processing is longer than the maximum tolerable delay of the task, and the task processing delay of MEC/cloud processing is lower than the maximum tolerable delay, it is necessary to ensure that the utility of local execution of the vehicle is lower than that of MEC and cloud computing processing.

Therefore, the utility function of MEC and cloud computing processing for vehicle i can be expressed, respectively, as

$$u_i^{mec} = \theta_i \ln(1 + (t_i^{sen} - t_i^{mec})^+) - (1 - \theta_i) \rho_m f_i^{mec}, \quad (12)$$

$$u_i^{ccc} = \theta_i \ln(1 + (t_i^{sen} - t_i^{ccc})^+) - (1 - \theta_i) \rho_c f_i^{ccc}, \quad (13)$$

where $(x)^+ = \max(x, 0)$, which normalizes the satisfaction to be non-negative in terms of task processing delay, θ_i is the weight coefficient, ρ_m is the unit cost of the computing resource of MEC server, and ρ_c is the unit price of the computation resource of cloud server.

Then, the utility function of local processing for vehicle i can be expressed as

$$u_i^{loc} = \ln(1 + (t_i^{max} - t_i^{loc})^+) - \varpi I(t_i^{max} < t_i^{loc}), \quad (14)$$

where $I(x)$ is an indicator function, which is equal to 1 if x is true and 0 otherwise, and ϖ is used to normalize the u_i^{loc} . ϖ guarantees $u_i^{loc} < s_i^{mec} u_i^{mec} + s_i^{ccc} u_i^{ccc}$ when $t_i^{max} < t_i^{loc}$ and $s_i^{ccc} t_i^{ccc} + s_i^{mec} t_i^{mec} < t_i^{sen}$.

D. Problem Formulation

In this sub-section, the optimization problem of computation offloading and resource allocation is designed for the cloud assisted MEC system. This problem seeks to maximize the system utility by optimizing both computation offloading strategy \mathcal{S} and computation resource allocation \mathcal{F} , under the

maximum delay constraints of tasks, and is formulated as

$$\begin{aligned} \max_{\mathcal{S}, \mathcal{F}} \quad & \sum_{j \in \mathcal{A}} \sum_{i \in \mathcal{N}} s_i^j u_i^j, \\ \text{s.t.} \quad & C1: f_i^{loc} \geq 0 \quad \forall i \in \mathcal{N}, \\ & C2: 0 \leq f_i^{mec} \leq s_i^{mec} F, \quad \forall i \in \mathcal{N}, \\ & C3: \sum_{i=1}^N f_i^{mec} \leq F, \quad i \in \mathcal{N}, \\ & C4: s_i^{mec} + s_i^{ccc} + s_i^{loc} \leq 1, \quad \forall i \in \mathcal{N}, \\ & C5: s_i^j = \{0, 1\}, i \in \mathcal{N}, j \in \{loc, mec, ccc\}, \\ & C6: s_i^j t_i^j \leq (s_i^{mec} + s_i^{ccc}) t_i^{sen} + s_i^{loc} t_i^{max}, \end{aligned} \quad (15)$$

where $\mathcal{S} = \{\mathcal{S}^{loc}, \mathcal{S}^{mec}, \mathcal{S}^{ccc}\}$ and $\mathcal{F} = \{f_1^{mec}, f_2^{mec}, \dots, f_N^{mec}\}$ are the execution indicator vector and computation resource allocation, respectively. $C1$ is the available local computing resource, which is non-negative, $C2$ is the constraint of the available computational resource allocation for vehicle i in MEC server, and $C3$ is the constraint of total computation resource of MEC server. The computation resource of cloud computing server is unconstrained. For each task, only one implementation approach can be selected, which is shown in $C4$ and $C5$. $C6$ is the latency constraint of each computation task.

The above mixed-integer programming problem is difficult to solve. Moreover, (15) is a classical maximum cardinality bin packing problem, which is NP-hard [36]. Generally, the exhaustion method named the enumeration computation offloading and resource allocation (ECORA) scheme can be applied to solve this problem, which leads to an optimal solution. However, its computation complexity is very high because it has to enumerate all optional offloading decision combinations. A CCORAO scheme with low computation complexity is presented to analyze the cloud assisted MEC collaborative computation offloading problem (15) in the next section, which decomposes (15) into the subproblems of offloading decisions and resource allocation. Moreover, it can be used to obtain a near-optimal solution compared to the ECORA scheme in Section IV.

III. COLLABORATIVE COMPUTATION OFFLOADING AND RESOURCE ALLOCATION OPTIMIZATION SCHEME

In this section, a CCORAO scheme is proposed, which decouples the original optimization problem into two subproblems to obtain the maximal system utility. One is the offloading strategy, which is taken via the potential game. Another is the resource allocation, which achieves the optimal computation resource allocation for the MEC and cloud computing processing vehicles. That is, the offloading strategy is obtained under achieved computing resources allocation, and then computing resources allocation is identified under given offloading strategy. This process is iterated until convergence. Moreover, the offloading strategy and resource allocation are detailed in Sections III-A and III-B, respectively. Finally, the details of CCORAO scheme are described and its complexity is analyzed in Section III-C.

A. Computation Offloading Strategy

In the computing decision-making problem among multiple vehicles, the offloading decision of vehicle n not only depends on its own offloading requirement but also on the offloading strategies of other vehicles. The inherent competitive nature of the computation offloading decision making problem shows that game theory is feasible to address the problem. The game theory uses a rigorous mathematical model to solve the conflict of interest among the decision-making bodies in the real world to achieve the best strategy for optimal combination. Thus, a game theoretic approach is used to address the computation offloading decision-making problem.

1) *Game Formulation*: In this sub-section, the computing decision-making problem among multiple vehicles is modeled as a computation offloading strategy game. In this game, the players are all vehicles which have computation tasks while there are players competing for resources to maximize respective utility. The computation offloading strategy game can be defined as $\Gamma = \{\mathcal{N}, (\mathcal{S}_i)_{i \in \mathcal{N}}, (u_i)_{i \in \mathcal{N}}\}$, where \mathcal{S}_i is a set of offloading strategy of vehicle i . The utility function of vehicle i is $u(s_i, s_{-i})$, where $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N)$ stands for the offloading decisions of all vehicles apart from vehicle i . Each vehicle seeks to select a profitable offloading strategy $(s_i^{mec} + s_i^{ccc} + s_i^{loc} = 1)$ to maximize its utility, i.e.,

$$\max_{s_i} u(s_i, s_{-i}) = s_i^{mec} u_i^{mec} + s_i^{ccc} u_i^{ccc} + s_i^{loc} u_i^{loc}. \quad (16)$$

Then, a vital concept of NE is introduced to solve the proposed computation offloading strategy game.

Definition 1: A strategy $\mathcal{S}^* = (\mathcal{S}^{mec*}, \mathcal{S}^{ccc*}, \mathcal{S}^{loc*})$ is the NE for the game, if for any $i \in \mathcal{N}$, we have the following relationship.

$$u(s_i^*, s_{-i}^*) \geq u(s_i, s_{-i}^*). \quad (17)$$

The NE is a steady state for the system, where no vehicle has any intention to unilaterally break away from this steady state for getting additional revenue. Then, the existence of NE of the computation offloading strategy game can be certified by rolling out the exact potential game.

2) *The Existence of Nash Equilibrium*: Every exact potential game with finite strategy sets always has a NE and provides the finite improvement property (FIP) [37]. Moreover, a game is known as an exact potential game if it admits potential function $\psi(s)$ for every vehicle when the offloading strategy unilaterally changes from s_i to s'_i and $s_{-i} \in \prod_{j \neq i} \mathcal{A}_j$, $s_i, s'_i \in \mathcal{S}_i$, we can obtain the following relationship.

$$u(s_i, s_{-i}) - u(s'_i, s_{-i}) = \psi(s_i, s_{-i}) - \psi(s'_i, s_{-i}). \quad (18)$$

The potential function $\psi(s)$ accurately reflects the unilateral change produced by the utility function of any player [17], [28], [38].

Lemma 1: The computation offloading strategy game is an exact potential game with the potential function $\psi(s)$ as shown in (19), and always converges to the NE and has the FIP.

Proof: See Appendix A. ■

$$\begin{aligned} \psi(s) = & s_i^{loc} \sum_{n=1}^N \left(\ln \left(1 + (t_n^{max} - t_n^{loc})^+ \right) \right. \\ & \left. - \varpi I(t_n^{max} < t_n^{loc}) \right) + (1 - s_i^{loc}) \\ & \times \left\{ \theta_i \ln \left(1 + (t_i^{sen} - s_i^{ccc} t_i^{ccc} - s_i^{mec} t_i^{mec})^+ \right) \right. \\ & \left. - (1 - \theta_i) (s_i^{mec} \rho_m f_i^{mec} + s_i^{ccc} \rho_c f_i^{ccc}) \right. \\ & \left. + \sum_{n=1, n \neq i}^N \left(\ln \left(1 + (t_n^{max} - t_n^{loc})^+ \right) \right) \right. \\ & \left. - \varpi I(t_n^{max} < t_n^{loc}) \right\}. \end{aligned} \quad (19)$$

After the resource allocation step, the computation offloading strategies are updated by the potential game until obtaining NE. After offloading strategies are given by all vehicles, the computation resource allocation of MEC and cloud computing needs optimization to maximize the utility of all offloading vehicles. By mutual iteration, the system goes into a steady state, and the optimal solution is obtained. Moreover, the optimal resource allocation will be discussed in the next sub-section.

B. The Optimal Computation Resource Allocation

In this sub-section, the optimal computation resource allocation is evaluated for vehicles that are required to offload the tasks to MEC or cloud. Initially, the optimal computation resource allocation of MEC is analyzed, which seeks to maximize the utility of vehicles with offloading the computation tasks to the MEC server. Then, the utility of vehicles is maximized that chooses cloud computing implementation by optimizing the computation resource allocation of cloud server.

1) *Optimal Computation Resource Allocation of MEC*: When the vehicles offload their computation tasks to the MEC server, the optimal computation resource allocation needs to be obtained, using

$$\begin{aligned} \max_{\mathcal{F}} \sum_{i \in \mathcal{N}_0} & \theta_i \ln(1 + (t_i^{sen} - t_i^{mec})^+) - (1 - \theta_i) \rho_m f_i^{mec} \\ s.t. \quad C7: & \frac{C_i}{f_i^{mec}} + \frac{D_i}{\gamma_i} \leq t_i^{sen}, \\ & C2, \quad C3, \quad \forall i \in \mathcal{N}_0, \end{aligned} \quad (20)$$

where \mathcal{N}_0 is a set of vehicles required to offload the tasks to the MEC server.

Lemma 2: The problem in (20) is convex.

Proof: See Appendix B. ■

Since (20) is convex, the Slater condition is satisfied. Hence, the problem can be solved by using partial Lagrange function, and it is formulated as

$$\begin{aligned} L(\mathcal{F}, \lambda) = & \sum_{i \in \mathcal{N}_m} \theta_i \ln(1 + (t_i^{sen} - t_i^{mec})^+) \\ & - (1 - \theta_i) \rho_m f_i^{mec} + \lambda \left(\sum_{i=1}^N f_i^{mec} - F \right), \end{aligned} \quad (21)$$

Algorithm 1: Bisection Method for Computation Resource Allocation (BMACR) Algorithm.

Initialization:

1: Maximum tolerance $\varepsilon > 0$, $\lambda^{\min} = 0$, $\lambda^{\max} = \lambda^{\text{bound}}$.

Bisection Search:

2: **while** $\lambda^{\max} - \lambda^{\min} > \varepsilon$ **do**

3: Define $\lambda = (\lambda^{\min} + \lambda^{\max})/2$, and compute f_i^{mec} according to substitute λ into (22).

4: If $\sum_{i=1}^N f_i^{\text{mec}} < F$, update $\lambda^{\max} = \lambda$, otherwise update $\lambda^{\min} = \lambda$.

5: **end while**

6: Optimal computation resource allocation scheme can be derived by substituting λ into (22).

Output: $\mathcal{F}^* = \{f_1^{\text{mec}*}, f_2^{\text{mec}*}, \dots, f_n^{\text{mec}*}\}$

where λ is the Lagrange multiplier related to the computation resource constraint of MEC server and $\lambda \geq 0$.

Subsequently, the KKT conditions are used to take the optimal computation resource allocation \mathcal{F}^* . By differentiating $L(\mathcal{F}, \lambda)$ with respect to f_i^{mec} ($i \in \mathcal{N}_0$), and makes it equal 0, it can be achieved that

$$f_i^{\text{mec}*} = \frac{C_i + \sqrt{C_i^2 - 4 \left(1 + t_i^{\text{sen}} - \frac{D_i}{\gamma_i}\right) \left(-\frac{C_i \theta_i}{(1-\theta_i)\rho_m + \lambda^*}\right)}}{2 \left(1 + t_i^{\text{sen}} - \frac{D_i}{\gamma_i}\right)}. \quad (22)$$

Then, the optimal computation resource allocation solution $f_i^{\text{mec}*}$ is achieved by applying the bisection algorithm. The bisection method for computation resource allocation (BMCR) algorithm is shown in Algorithm 1.

2) *Optimal Computation Resource Allocation of Cloud Computing:* Despite that the cloud server always has adequate computing resources when the vehicle offloads its computation task to the cloud server, the optimal computation resource allocation should be computed to maximize the utility of vehicle. As the vehicle uses more computation resource, more cost incurs. Similar to MEC processing, the resource allocation of cloud server can be obtained.

Differentiating u_i^{ccc} with respect to f_i^{ccc} , it can be obtained

$$\frac{\partial u_i^{\text{ccc}}}{\partial f_i^{\text{ccc}}} = \frac{\theta_i}{\Lambda_i - \frac{C_i}{f_i^{\text{ccc}}}} \frac{C_i}{(f_i^{\text{ccc}})^2} - (1 - \theta_i)\rho_c. \quad (23)$$

The second-order derivative of u_i^{ccc} with respect to f_i^{ccc} is

$$\frac{\partial^2 u_i^{\text{ccc}}}{\partial (f_i^{\text{ccc}})^2} = -\frac{\frac{\theta_i C_i}{(f_i^{\text{ccc}})^3} \left(2 \left(\Lambda_i - \frac{C_i}{f_i^{\text{ccc}}}\right) + \frac{C_i}{f_i^{\text{ccc}}}\right)}{\left(\Lambda_i - \frac{C_i}{f_i^{\text{ccc}}}\right)^2}, \quad (24)$$

where $\Lambda_i = 1 + t_i^{\text{sen}} - \frac{D_i}{\gamma_i} - (D_i + D_i^o)\tau$.

It is quite obvious that the utility function is concave for $\frac{\partial^2 u_i^{\text{ccc}}}{\partial (f_i^{\text{ccc}})^2} < 0$. By using the first-order optimality condition,

Algorithm 2: The Algorithm for Sub-offloading Strategy.

Initialization:

Vehicle set $\mathcal{N} = \{1, 2, \dots, N\}$, computation task

$\varphi_i = \{C_i, D_i, t_i^{\text{max}}\}$, vehicular speed v

$\sim \mathcal{CN}(\mu, \sigma^2)$, vehicle position d_i

1: **for all** Vehicle $i \in \mathcal{N}$ **do**

2: Calculate $f_i^{\text{ccc}*}$ by (25), calculate t_i^{loc} and u_i^{loc} according to (5) and (14) respectively, calculate t_i^{ccc} and u_i^{ccc} by substituting $f_i^{\text{ccc}*}$ into (11) and (13) respectively.

3: **if** $t_n^{\text{max}} < t_i^{\text{loc}}$ and $t_n^{\text{sen}} < t_i^{\text{ccc}}$ **then**

4: $u_i^{\text{lc}} = -\infty$, $s_i^{\text{loc}} = 0$, $s_i^{\text{ccc}} = 0$

5: **else**

6: If $u_i^{\text{ccc}} > u_i^{\text{loc}}$, update $s_i^{\text{ccc}} = 1$, $s_i^{\text{loc}} = 0$, otherwise update $s_i^{\text{loc}} = 1$, $s_i^{\text{ccc}} = 0$

7: **end if**

8: **end for**

9: $u^{\text{lc}*} = \{u_1^{\text{lc}*}, u_2^{\text{lc}*}, \dots, u_N^{\text{lc}*}\}$ can be derived.

Output: $u_i^{\text{lc}*}$

$\frac{\partial u_i^{\text{ccc}}}{\partial f_i^{\text{ccc}}} = 0$, it can be derived that

$$f_i^{\text{ccc}*} = \frac{C_i + \sqrt{C_i^2 - 4\Lambda_i C_i}}{2\Lambda_i}, \quad (25)$$

where $C_i = -\frac{\theta_i C_i}{(1-\theta_i)\rho_c}$. Consequently, $u_i^{\text{ccc}*}$ is taken via substituting (25) into (13).

C. Algorithm Process of DCORA

In this section, we take the advantage of the FIP of computation offloading strategy game and design an effective algorithm identified as DCORA algorithm for CCOROA scheme. The main objective of the DCORA algorithm is to maximize the system utility. Specifically, considering the cloud server always has enough computational resources, the algorithm of sub-offloading strategy in the DCORA Algorithm is applied, which is detailed in Algorithm 2. In Algorithm 2, the vehicles can make a pre-selection in the cloud computing processing model or local processing model. In simple terms, the vehicles decide to complete the computation task by choosing cloud computing or local processing model when $s_i^{\text{mec}} = 0$. Moreover, it can be noted $s_i^{\text{loc}} u_i^{\text{loc}} + s_i^{\text{ccc}} u_i^{\text{ccc}} = u_i^{\text{lc}}$ in Algorithm 2.

Finally, the details of the DCORA Algorithm are shown in Algorithm 3. At first, a initial offloading strategy is assumed. Then, u_i^{mec} and u_i^{lc} are obtained by using Algorithms 1 and 2, respectively. The offloading strategy is updated by comparing the size of u_i^{mec} and u_i^{lc} . Then, the algorithm undertakes the iterative process. In each iterative process, for all vehicles, their utility is computed separately based on the current offloading strategy. Then, each vehicle updates the offloading strategy to increase the respective utility. The iteration process is terminated when no vehicle has motivation to change its offloading strategy. Based on the DCORA algorithm, the CCORAO

Algorithm 3: The Distributed Computation Offloading and Resource Allocation (DCORA) Algorithm.
Initialization:

- Vehicle set $\mathcal{N} = \{1, 2, \dots, N\}$, Computation task $\varphi_i = \{C_i, D_i, t_i^{max}\}$, $i \in \mathcal{N}$, vehicular speed $v \sim \mathcal{CN}(\mu, \sigma^2)$, vehicle position d_i , initial offloading strategy \mathcal{S}_Θ
- 1: Allocate computation resources f_i^{mec*} to the vehicles by use the Algorithm 1, calculate u_i^{mec} according to (12), and obtain the sub-offloading strategy s_i^{loc} , s_i^{ccc} and u_i^{lc} according to Algorithm 2
 - 2: **for all** vehicle $i \in \mathcal{N}$ **do**
 - 3: If $u_i^{lc} > u_i^{mec}$ update $s_i^{mec} = 0$, $s_i^{ccc} = s_i^{ccc}$, $s_i^{loc} = s_i^{loc}$, otherwise $s_i^{mec} = 1$, $s_i^{ccc} = 0$, $s_i^{loc} = 0$
 - 4: Update $\mathcal{S}(t) = \{\mathcal{S}^{loc}(t), \mathcal{S}^{mec}(t), \mathcal{S}^{ccc}(t)\}$
 - 5: **end for**
 - 6: Let $t = t + 1$
 - 7: **while** $\mathcal{S}(t-1) \neq \mathcal{S}_\Theta$ **do**
 - 8: $\mathcal{S}_\Theta = \mathcal{S}(t-1)$, $i = 1$
 - 9: **while** $i \leq N$ **do**
 - 10: Update $s'_i = s_i^{mec} = 1$, and calculate $u_{update}(i) = u(s'_i, s_{-i}(t-1))$ based on Algorithms 1 and 2
 - 11: $i = i + 1$
 - 12: **end while**
 - 13: **for each** vehicle $i \in \mathcal{N}$ **do**
 - 14: If vehicle i has maximal u_{update} , or $s_\Theta(i) = s_i^{mec} = 1$, update $s_i(t) = s'_i$, otherwise $s_i(t) = s_i(t-1)$
 - 15: **end for**
 - 16: $t = t + 1$
 - 17: **end while**

Output: The optimal computation resource allocation \mathcal{F}^* and offloading strategy \mathcal{S}^* .

 TABLE II
SIMULATION PARAMETERS

Parameter	Value
The maximum rate of RSU (Mbps)	2.5 [39]
The arrival rate of vehicles (veh/s)	0.1
The interval of arbitration inter-frame spacing (μs)	0.1
The request to send interval (μs)	144
The propagation delay (μs)	2
The coverage length of RSU (km)	1
The number of RSUs	4
The computation resource price coefficient of cloud (\$/GHz)	0.015
The computation resource price coefficient of MEC (\$/GHz)	0.03
The average vehicular velocity (km/h)	120
The variance of vehicular velocity	21

 TABLE III
ACRONYMS DEFINITIONS

Acronyms	Description
MEC	Mobile Edge Computing
ELP scheme	Entire Local Processing scheme
ECP scheme	Entire Cloud computing Processing scheme
EMP scheme	Entire MEC Processing scheme
COSO scheme	Computation Offloading Strategy Optimization scheme
CORAO scheme	Computation Offloading and Resource Allocation Optimization scheme
ECORA scheme	Enumeration Computation Offloading and Resource Allocation scheme
CCORAO scheme	Collaborative Computation Offloading and Resource Allocation Optimization scheme
BMACR Algorithm	Bisection Method for Computation Resource Allocation Algorithm
DCORA Algorithm	Distributed Computation Offloading and Resource Allocation Algorithm

IV. THE SIMULATION ANALYSIS

In this section, the representative numerical results are presented. Initially, the convergence of the proposed algorithm is validated in Section IV-A. Then the performance of CCORAO scheme is evaluated in Section IV-B. For the numerical analysis, the computation resource of vehicles, input data size, service coefficient and maximum latency constraint of each computation task take a uniform distribution in the range of [10, 15] GHz, [10, 20] KB, [0.2, 0.4] GHz/KB and [0.2, 1] s, respectively. The values of other simulation parameters used in our simulations are listed in Table II. The simulation parameters of Table II are set as default unless otherwise specified.

Moreover, for the clarity of performance evaluation, a list of acronyms is provided in Table III.

A. Convergence of Algorithms

Fig. 2 highlights the convergence performance of the BMACR Algorithm under different computation capacities of the MEC server. In the BMACR Algorithm, initially, it can be assumed that $\lambda^* = 0$, where the optimal resource allocation is the same for the different computing resources of MEC server based on (22). However, the allocated computation resources are higher than the total computation resource of MEC server at the start. Therefore, to meet the constraint of the computing resource of MEC server, BMACR Algorithm reduces the computation resource

scheme obtains the optimal solutions (offloading decision making and computation resource allocation), where NE of computation offloading strategy game is taken and no vehicle deviates from this offloading strategy. The NE means each vehicle gets the last offloading decision and no vehicle deviates from this offloading strategy. Based on DCORA algorithm, the vehicles select the most reasonable offloading strategy to complete the task.

The computation complexity of Algorithm 3 is analyzed in the following. In the CCORAO scheme, the computational complexity of Algorithm 1 is $O(\log_2((\lambda^{\max} - \lambda^{\min})/\varepsilon))$, and the computational complexity of Algorithm 2 is $O(N)$. The while-loop in Algorithm 3 needs K iterations to converge, thus the computational complexity of the CCORAO scheme is $O(KN(\log_2((\lambda^{\max} - \lambda^{\min})/\varepsilon) + N))$. For the ECORA scheme, however, all the offloading decision combinations need to be traversed and the running time is $O(3^N)$. Therefore, the computational complexity of the ECORA scheme is $O(3^N(\log_2((\lambda^{\max} - \lambda^{\min})/\varepsilon) + N))$.

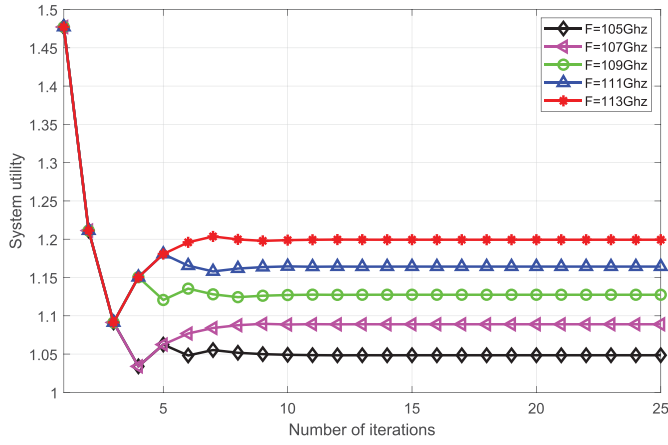


Fig. 2. Convergence of the BMACR algorithm.

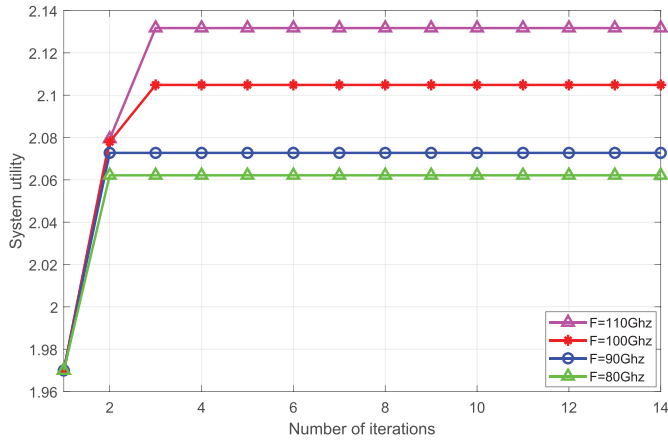


Fig. 3. Convergence of the DCORA algorithm.

allocation by adjusting λ . At a certain point, the optimal computing resource allocation satisfies the constraint of the computing resource of the MEC server and the utility becomes constant. In Fig. 2, it is observed that the BMACR algorithm converges with a few iterations.

Fig. 3 shows the convergence of the outer loop of the DCORA Algorithm, under the different computational resources of MEC server. As shown, the system utility keeps on increasing until it converges. The reason is that the DCORA Algorithm seeks to maximize the system utility in the iteration by optimizing offloading decisions and computation resource allocation. Moreover, the DCORA Algorithm converges usually in several iterations.

Fig. 4 shows the impact of the vehicular velocity on the actual delay tolerance of task (t_i^{sen}), where the maximum time delay of vehicle i is 1s. The actual delay tolerance of task reduces with the increase of vehicular velocity. When the vehicular velocity is higher, the time that the vehicle i leaves the connecting RSU is smaller compared with the maximum time delay of task. The delay constraint of the task is converted to t_i^m . In contrast, the delay constraint of the task is equal to the maximum time delay of each task t_i^{\max} when the vehicular velocity is lower. This purpose is to guarantee smooth transmission of the task in the MEC processing model or cloud computing execution model.

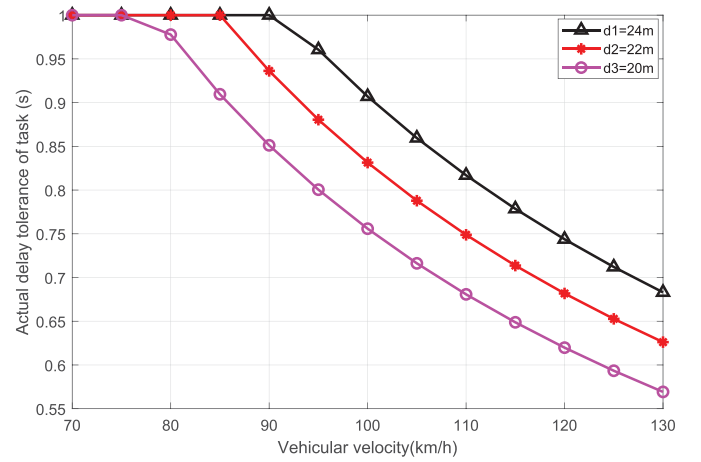


Fig. 4. The delay tolerance of the task versus various vehicular velocity.

B. Performance of CCORAO Scheme

This paper evaluates the performance of the proposed CCORAO scheme in comparison with the following benchmark schemes.

- *The entire local processing (ELP)* scheme, where all vehicles execute computation tasks on themselves.
- *The entire cloud computing processing (ECP)* scheme, where all vehicles offload their computation tasks to the cloud server.
- *The entire MEC processing (EMP)* scheme, where all vehicles offload their computation tasks to the MEC.
- *Computation offloading strategy optimization (COSO)* scheme, where only offloading decisions are optimized to maximize the system utility, under a given computation resource.
- *Computation offloading and resource allocation optimization (CORAO)* scheme, where the computation tasks are completed by computing locally or offloading to the MEC server. In simple terms, the major difference between CORAO and CCORAO schemes is that CORAO scheme does not involve the cloud computing execution model.

The impact of the required computing resource of computation task C_i on the system utility of all schemes is shown in Fig. 5. From the figure, it is obvious that the system utility keeps dropping with the increase of C_i for all schemes. Moreover, compared to COSO, EMP, ECP and ELP schemes, the proposed CCORAO scheme always achieves striking performance gain. This reason is that COSO schemes only optimizes computation offloading strategy, and EMP and ECP schemes only optimize the computation resource allocation while the proposed CCORAO scheme jointly optimizes both. Further, although the CORAO scheme jointly optimizes computation offloading strategy and computation resource allocation, the CORAO scheme just makes the vehicle offload the computation task to the MEC server, it does not consider cloud computing processing. When the required computing resource of computation task becomes larger, the MEC server will be unable to provide the required computation resource of computation task, resulting in a low system utility. Therefore, when the required computing resource of

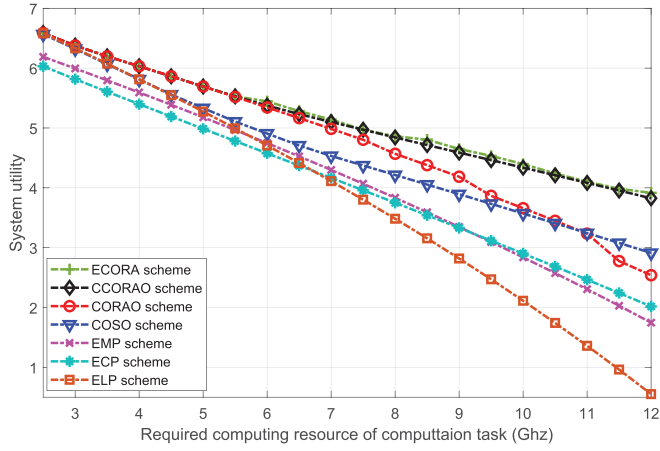


Fig. 5. The system utility variance with the required computing resource of computation task.

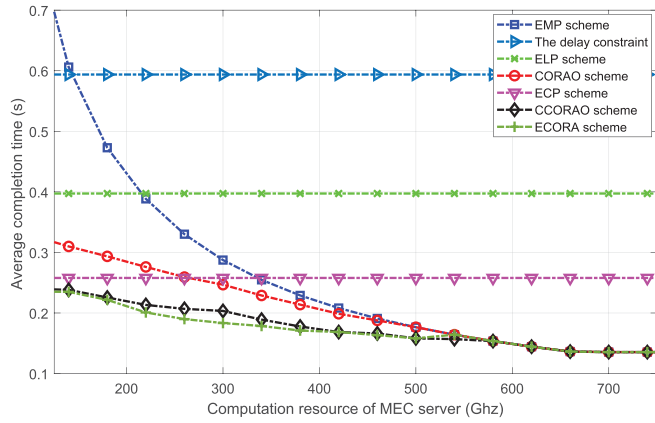


Fig. 6. The average completion time of the task with regard to the computation resources of MEC server.

computation task is relatively small, the CCORAO and CORAO scheme have the same utility. However, the system utility gap between CCORAO and CORAO becomes larger with the required computing resource of computation tasks increasing. Our proposed CCORAO scheme better uses the computing resource of MEC and cloud computing servers, jointly optimizes computation offloading strategy and computation resource allocation. ECORA scheme obtains the optimal system utility by enumerating all possible solutions. It can be observed that the proposed scheme can obtain a near-optimal solution to (15), which closely estimates to the ECORA scheme. However, the complexity of the proposed CCORAO scheme is less than that of the ECORA scheme, where the complexity of the proposed CCORAO and the ECORA schemes are $O(KN(\log_2(\frac{\lambda^{\max}-\lambda^{\min}}{\epsilon}) + N))$ and $O(3^N(\log_2(\frac{\lambda^{\max}-\lambda^{\min}}{\epsilon}) + N))$, respectively. They are discussed in Section III-D. With that, the proposed scheme not only achieves a near-optimal solution but also has a higher computational efficiency compared to the ECORA scheme.

Fig. 6 shows the comparative results with the different completion time of task among CCORAO, CORAO, ECORA, ELP, ECP and EMP schemes. The completion time includes both data transmission and task processing times. In Fig. 6, it can be seen

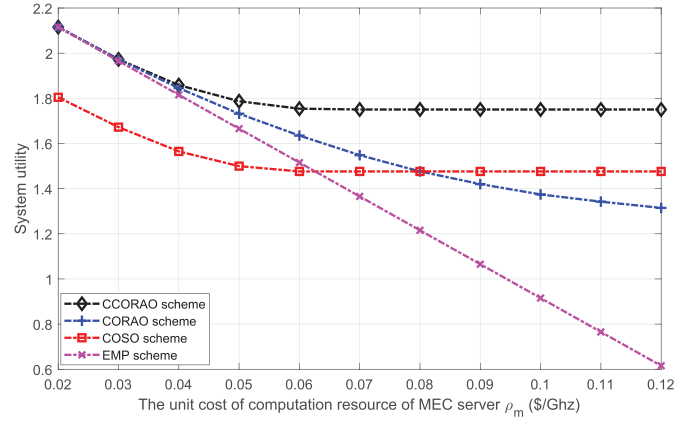


Fig. 7. The system utility variance with the unit cost of computation resource of MEC server.

that the completion time is invariant for ELP and ECP schemes as they do not consider offload computation task to the MEC server. The completion time of computation task keeps declining with the computing resource of MEC server increasing for CCORAO, CORAO and EMP schemes. Moreover, it can be noted that the completion time of CCORAO scheme is always lower compared with other schemes until the computing resources of MEC server get larger. This is because the tasks of MEC processing have fewer computation resource than cloud computing execution when the computing resource of MEC server is relatively small. Hence, the completion time can be more significantly reduced by offloading to the centralized cloud computing server. When the computing resource of MEC server becomes larger, the vehicles are more likely to offload their tasks to the MEC server while the completion time of a task is the same for CCORAO, CORAO and EMP schemes in this case. Furthermore, as shown in Fig. 6, the completion time of the EMP scheme is higher than the delay constraint when the computing resource of the MEC server is quite low. Also, the reason is that the computing resources of the MEC server cannot fully meet the demands of computation. The proposed CCORAO scheme effectively use MEC and cloud computing resources, and reduces the task processing task. In addition, it can be observed that the average completion time of computation tasks that is obtained in the proposed CCORAO scheme is very close to the optimal solution in the ECORA scheme.

Fig. 7 indicates the system utility versus the unit cost of computation resource of MEC server. The proposed CCORAO scheme acquires higher system utility than other schemes with the unit cost of computation resource increasing. The main reason is that the MEC processing needs to cost more than the cloud computing processing as the unit cost of computation resource of MEC server increases. Hence, vehicles offload their computation tasks to the cloud server to obtain higher system cost. Additionally, although COSO scheme makes the vehicles offload their computation task to the cloud server, it only optimizes the computation offloading strategy as described in Section III-A, without consideration of computation resource allocation as described in Section III-B. Moreover, with the further increase of the unit cost of computation resource of MEC

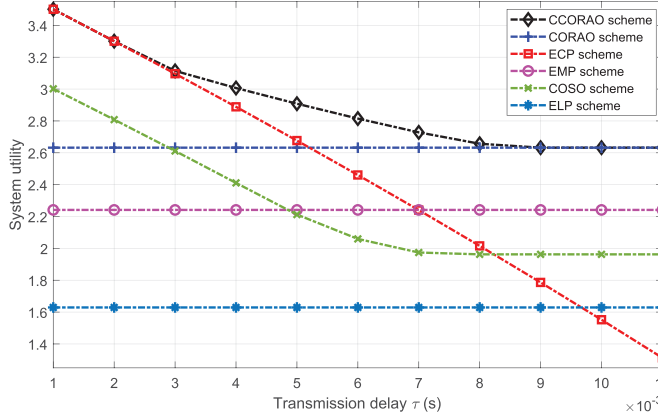


Fig. 8. The system utility variance with the transmission delay from RSU to cloud server.

server, the performance of the COSO scheme outperforms the CORAO scheme. The proposed CCORAO scheme achieves significant performance gain all the time.

Fig. 8 illustrates the relationship between system utility and transmission delay from RSU to cloud server. From this figure, it can be observed that the system utility of ECP scheme always drops off rapidly as the transmission delay increases. For COSO and CCORAO schemes, the system utility keeps dropping with the increase of transmission delay when the transmission delay is relatively small, and they become constant after the transmission delay becomes relatively large. Moreover, the proposed CCORAO schemes always can obtain higher system utility than the COSO scheme. This is due to the proposed CCORAO scheme utilizes the MEC and cloud computing resources simultaneously, and jointly optimizes computation offloading strategy and computation resource allocation while COSO scheme only optimizes computation offloading strategy, and other schemes ignores the powerful computing resources of the cloud server. In addition, cloud computing processing results in a higher task processing delay when the transmission delay becomes large, and the utility of cloud computing processing achieves a relatively small value. Hence, vehicles intend to offload their computation task to the MEC server when the transmission delay becomes relatively large, and the system utility is same in this moment for CCORAO and CORAO schemes.

Fig. 9 shows the system utility with different vehicular speed. The system utility keeps decreasing with the average vehicular speed increasing. This is brought by the growing mobility of vehicles that causes the reduction of vehicular density in road segment, letting the number of vehicles N to reduce. Moreover, our proposed CCORAO scheme achieves highest system utility than other schemes when the vehicular velocity is relatively small. The reason is that the MEC server cannot provide all vehicles required computing resource when the number of vehicles is large. However, the proposed CCORAO scheme uses the computing resource of cloud server, which makes vehicles offload computation tasks to the cloud server to meet their computation requires. Therefore, the proposed can keep relatively high value even when the number of vehicles is large. Until the vehicular velocity becomes relatively large, which results in small the

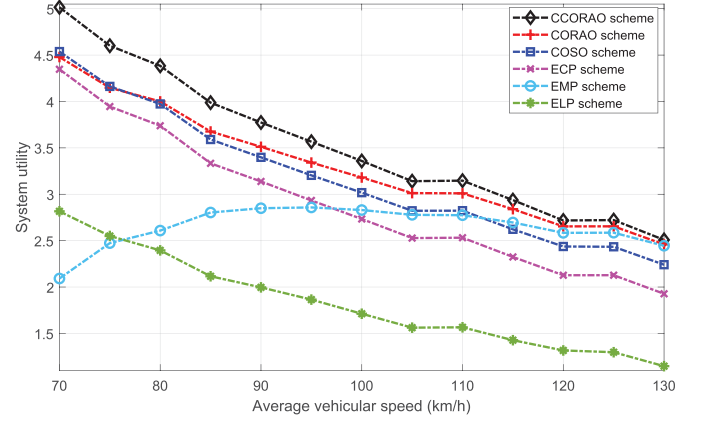


Fig. 9. The system utility with different average vehicular speed.

number of vehicles, the system utility in this moment is close for CCORAO, CORAO and EMP schemes. This is because MEC server can meet computation requires of vehicles well, therefore the vehicles tend to choose MEC processing.

In conclusion, based on the simulation results, it is clear to observe the necessity of introducing cloud computing in MEC, especially when there are many resource hungry computation-intensive tasks, the computation capacities of MEC servers are not enough or the unit cost of computation resource of MEC server is higher. Moreover, the proposed CCORAO scheme makes the best use of computation resource of MEC and centralized cloud computing servers, and provides a practical and feasible collaborative computation offloading scheme to reduce the completion time. The offloading decision ($s_i^j = 1, j \in \{loc, mec, ccc\}$) can be determined dynamically by vehicles to satisfy the time delay and maximize the system utility.

V. CONCLUSION

In this paper, a collaborative computing offloading problem is formulated for vehicular networks where MEC and cloud computing coexist. This problem is identified as a constrained optimization by jointly optimizing computation offloading decisions and computation resource allocation to maximize the system utility. In addition, a CCORAO scheme is proposed to solve this problem, which includes computation offloading strategy game and resource allocation. A DCORA algorithm is developed for CCORAO scheme, which can largely decrease the system complexity without loss of the performance. The numerical results showed the improved performance of our proposed CCORAO scheme than the existing schemes in terms of the computation time and system utility. The advantage is more obvious when the computing resources of MEC servers decrease, the number of computation-intensive tasks increases, or the unit cost for the MEC computation resource increases.

APPENDIX A PROOF OF THE LEMMA 1

To prove the existence of NE of the computation offloading strategy game, a potential function is created as

shown in (19) [17], [40]. Then, for every vehicle i ($i \in \mathcal{N}$), $s_{-i} \in \prod_{j \neq i} \mathcal{S}_j$, the potential function is demonstrated to satisfy formula (18) when i updates its current offloading decision s_i to s'_i . Moreover, the following three cases are being considered: *Case 1.* $s_i = s_i^{loc} = 1$, $s'_i = s_i^{ccc} = 1$; *Case 2.* $s_i = s_i^{mec} = 1$, $s'_i = s_i^{loc} = 1$; *Case 3.* $s_i = s_i^{mec} = 1$, $s'_i = s_i^{ccc} = 1$.

Case 1 means that vehicle i decides to offload the computation task to centralized cloud computing server or compute locally. Based on (19), it is shown that

$$\begin{aligned} & \psi(s_i^{loc}, s_{-i}) - \psi(s_i^{ccc}, s_{-i}) \\ &= \sum_{n=1}^N \left(\ln \left(1 + \left(t_n^{max} - \frac{C_n}{f_n^{loc}} \right)^+ \right) - \varpi I \left(t_n^{max} < \frac{C_n}{f_n^{loc}} \right) \right) \\ & \quad - \sum_{n=1, n \neq i}^N u_n^{loc} - \theta_i \ln \left(1 + (t_i^{sen} - t_i^{ccc})^+ \right) \\ & \quad + (1 - \theta_i) \rho_c f_i^{ccc} \\ &= \ln \left(1 + \left(t_i^{max} - \frac{C_i}{f_i^{loc}} \right)^+ \right) - \varpi I(t_i^{max} < t_i^{loc}) \\ & \quad + (1 - \theta_i) \rho_c f_i^{ccc} - \theta_i \ln \left(1 + (t_i^{sen} - t_i^{ccc})^+ \right) \\ &= u(s_i^{loc}, s_{-i}) - u(s_i^{ccc}, s_{-i}). \end{aligned} \quad (26)$$

Case 2 implies that vehicle i decides to offload the computation task to the MEC server or compute locally, and it can be noted that

$$\begin{aligned} & \psi(s_i^{mec}, s_{-i}) - \psi(s_i^{loc}, s_{-i}) \\ &= \sum_{n=1, n \neq i}^N u_n^{loc} + \theta_i \ln \left(1 + (t_i^{sen} - t_i^{mec})^+ \right) \\ & \quad - (1 - \theta_i) \rho_m f_i^{mec} - \sum_{n=1}^N \left(\ln \left(1 + \left(t_n^{max} - \frac{C_n}{f_n^{loc}} \right)^+ \right) \right. \\ & \quad \left. - \varpi I \left(t_n^{max} < \frac{C_n}{f_n^{loc}} \right) \right) \\ &= u(s_i^{mec}, s_{-i}) - u(s_i^{loc}, s_{-i}). \end{aligned} \quad (27)$$

Case 3 means that vehicle i decides to offload the computation task to the MEC or centralized cloud computing servers. Based on (19), it can be derived that

$$\begin{aligned} & \psi(s_i^{mec}, s_{-i}) - \psi(s_i^{ccc}, s_{-i}) \\ &= \sum_{n=1, n \neq i}^N u_n^{loc} + \theta_i \ln \left(1 + (t_i^{sen} - t_i^{mec})^+ \right) \\ & \quad - (1 - \theta_i) \rho_m f_i^{mec} - \sum_{n=1, n \neq i}^N u_n^{loc} \\ & \quad - \theta_i \ln \left(1 + (t_i^{sen} - t_i^{ccc})^+ \right) + (1 - \theta_i) \rho_c f_i^{ccc} \\ &= u(s_i^{mec}, s_{-i}) - u(s_i^{ccc}, s_{-i}). \end{aligned} \quad (28)$$

Based on the results in the three cases cited above, the potential function $\psi(s)$ is seen that it always meets (18) for any

two offloading decisions of vehicle i . Therefore, computation offloading strategy game is an exact potential game.

APPENDIX B

PROOF OF THE LEMMA 2

To prove the problem in (20) is convex, it is demonstrated that utility function u_i^{mec} is a concave function with regard to f_i^{mec} .

Differentiating u_i^{mec} with respect to f_i^{mec} , it can be obtained

$$\frac{\partial u_i^{mec}}{\partial f_i^{mec}} = \frac{\theta_i}{1 + t_i^{sen} - \frac{C_i}{f_i^{mec}} - \frac{D_i}{\gamma_i}} \frac{C_i}{(f_i^{mec})^2} - (1 - \theta_i) \rho_m, \quad (29)$$

The second-order derivative of u_i^{mec} with respect to f_i^{mec} is expressed as

$$\begin{aligned} & \frac{\partial^2 u_i^{mec}}{\partial (f_i^{mec})^2} \\ &= - \frac{\frac{\theta_i C_i}{(f_i^{mec})^3} \left(2 \left(1 + t_i^{sen} - \frac{C_i}{f_i^{mec}} - \frac{D_i}{\gamma_i} \right) + \frac{C_i}{f_i^{mec}} \right)}{\left(1 + t_i^{sen} - \frac{C_i}{f_i^{mec}} - \frac{D_i}{\gamma_i} \right)^2}, \end{aligned} \quad (30)$$

where $t_i^{sen} - \frac{C_i}{f_i^{mec}} - \frac{D_i}{\gamma_i} > 0$ and $\frac{\partial^2 u_i^{mec}}{\partial (f_i^{mec})^2} < 0$.

REFERENCES

- [1] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *Proc. 55th Annu. Conf. Soc. Instrum. Control. Eng. Jpn. (SICE)*, Tsukuba, Japan, Sep. 2016, pp. 1122–1127.
- [2] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Ave: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Jun. 2017.
- [3] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper* no. 11, pp. 1–16, Sep. 2015.
- [4] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Con. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.
- [5] J. Zhao, Y. Chen, and Y. Gong, "Study of connectivity probability of vehicle-to-vehicle and vehicle-to-infrastructure communication systems," in *Proc. IEEE 83rd Vehicular Technol. Conf. (VTC Spring)*, Nanjing, China, May 2016, pp. 1–4.
- [6] X. Cheng, C. Chen, W. Zhang, and Y. Yang, "5G-enabled cooperative intelligent vehicular (5GenCiv) framework: When Benz meets Marconi," *IEEE Intell. Syst.*, vol. 32, no. 3, pp. 53–59, May 2017.
- [7] W. Lu, Y. Gong, X. Liu, J. Wu, and H. Peng, "Collaborative energy and information transfer in green wireless sensor networks for smart cities," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1585–1593, Apr. 2018.
- [8] S. Maharjan, Q. Zhu, Y. Zhang, S. Gjessing, and T. Basar, "Dependable demand response management in the smart grid: A Stackelberg game approach," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 120–132, Feb. 2013.
- [9] X. Li, C. You, S. Andreev, Y. Gong, and K. Huang, "Wirelessly powered crowd sensing: Joint power transfer, sensing, compression, and transmission," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 2, pp. 391–406, Oct. 2019.
- [10] N. Kumar, S. Zeadally, and J. J. Rodrigues, "Vehicular delay-tolerant networks for smart grid data management using mobile edge computing," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 60–66, Oct. 2016.
- [11] J. Zhao, Y. Liu, Y. Gong, C. Wang, and L. Fan, "A dual-link soft handover scheme for C/U plane split network in high-speed railway," *IEEE Access*, vol. 6, pp. 12473–12482, Jan. 2018.
- [12] A. Alahmadi, A. Q. Lawey, T. E. El-Gorashi, and J. M. Elmirghani, "Distributed processing in vehicular cloud networks," in *Proc. IEEE 8th Int. Con. Netw. Future*, London, U.K., Nov. 2017, pp. 22–26.
- [13] R. Yu, Y. Zhang, W. Xia, and K. Yang, "Toward cloud-based vehicular networks with efficient resource management," *IEEE Netw.*, vol. 27, no. 5, pp. 48–52, Oct. 2013.

- [14] M. Nabi, R. Benkoczi, S. Abdelhamid, and H. S. Hassanein, "Resource assignment in vehicular clouds," in *Proc. IEEE Int. Con. Commun. (ICC)*, Paris, France, May 2017, pp. 1–6.
- [15] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for Internet of Everything," *China Commun.*, vol. 16, no. 3, pp. 32–41, Mar. 2019.
- [16] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.
- [17] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19324–19337, Mar. 2018.
- [18] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Dec. 2017.
- [19] P. Zhao, Q. C. Tian, Hui, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access.*, vol. 5, pp. 11255–11268, Jun. 2017.
- [20] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Predictive offloading in cloud-driven vehicles: Using mobile-edge computing for a promising network paradigm," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.
- [21] 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, Nov. 2017.
- [22] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang, "Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks," in *Proc. 8th Int. Workshop Resilient Net. Design Model. (RNDM)*, Halmstad, Sweden, Sep. 2016, pp. 288–294.
- [23] T. Nguyen, T.-D. Nguyen, V. Nguyen, X.-Q. Pham, and E.-N. Huh, "Cost-effective resource sharing in an internet of vehicles-employed mobile edge computing environment," *Symmetry*, vol. 10, no. 11, pp. 594, Nov. 2018.
- [24] Q. Liu, Z. Su, and Y. Hui, "Computation offloading scheme to improve QoE in vehicular networks with mobile edge computing," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WSCP)*, Hangzhou, China, Oct. 2018, pp. 1–5.
- [25] 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, Feb. 2016.
- [26] R. Yu *et al.*, "Cooperative resource management in cloud-enabled vehicular networks," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7938–7951, Sep. 2015.
- [27] X. Ma, C. Lin, X. Xiang, and C. Chen, "Game-theoretic analysis of computation offloading for cloudlet-based mobile cloud computing," in *Proc. ACM Int. Conf. Model. Anal. Simulat. Wireless Mobile Syst.*, Cancun, Mexico, Nov. 2015, pp. 271–278.
- [28] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.
- [29] J. Zhang *et al.*, "An evolutionary game for joint wireless and cloud resource allocation in mobile edge computing," in *Proc. 9th Int. Conf. Wireless Commun. Signal Process. (WSCP)*, Nanjing, China, Oct. 2017, pp. 1–6.
- [30] N. Kumar, S. Zeadally, N. Chilamkurti, and A. Vinel, "Performance analysis of Bayesian coalition game-based energy-aware virtual machine migration in vehicular mobile cloud," *IEEE Netw.*, vol. 29, no. 2, pp. 62–69, Apr. 2015.
- [31] C. Shao, S. Leng, Y. Zhang, A. Vinel, and M. Jonsson, "Performance analysis of connectivity probability and connectivity-aware MAC protocol design for platoon-based VANETs," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5596–5609, Dec. 2015.
- [32] C. Campolo and A. Molinaro, "Improving V2R connectivity to provide ITS applications in IEEE 802.11 p/WAVE VANETs," in *Proc. 18th Intern. Conf. Telecommun.*, Ayia Napa, Cyprus, May 2011, pp. 476–481.
- [33] 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.
- [34] S. Yousefi, E. Altman, R. El-Azouzi, and M. Fathy, "Analytical model for connectivity in vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3341–3356, Jul. 2008.
- [35] S. Durrani, X. Zhou, and A. Chandra, "Effect of vehicle mobility on connectivity of vehicular ad hoc networks," in *Proc. IEEE 72nd Vehicular Technol. Conf.*, Ottawa, Canada, Sep. 2010, pp. 1–5.
- [36] K. Zhang *et al.*, "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, Aug. 2016.
- [37] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.
- [38] E. Kalai, *Games and Economic Behavior*. Academic Press, 1989.
- [39] Z. Hu, Z. Zheng, T. Wang, L. Song, and X. Li, "Roadside unit caching: Auction-based storage allocation for multiple content providers," *IEEE Trans. Wireless Commun.*, vol. 16, no. 10, pp. 6321–6334, Oct. 2017.
- [40] H. Guo, J. Liu, H. Qin, and H. Zhang, "Collaborative computation offloading for mobile-edge computing over fiber-wireless networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Singapore, Dec. 2017, pp. 1–6.



Junhui Zhao (S'00–M'04–SM'09) received the M.S. and Ph.D. degrees in information and communication engineering from Southeast University, Nanjing, China, in 1998 and 2004, respectively. From 1998 to 1999, he worked with the Nanjing Institute of Engineers at ZTE Corporation. From 2004 to 2007, he was as an Assistant Professor with the Faculty of Information Technology, Macau University of Science and Technology, Macau, China. In 2008, he joined as an Associate Professor with Beijing Jiaotong University, Beijing, China, where he is currently a Professor at the School of Electronics and Information Engineering. He was also a short term Visiting Scholar with Yonsei University, South Korea, in 2004 and a Visiting Scholar with Nanyang Technological University, Singapore, from 2013 to 2014. Since 2016, he is with the School of Information Engineering in East China Jiaotong University, Nanchang, China. His current research interests include wireless and mobile communications and related applications, containing 5G mobile communication technology, high-speed railway communications, vehicle communication network, wireless localization, and cognitive radio. Dr. Zhao was the recipient of the IEEE WCSP 2017 Best Paper Award.



Qiuping Li received the B.Sc. degree in electronic information science and technology from China West Normal University, Sichuan, China, in 2015. She is currently working towards the Ph.D. degree in communication and information systems from Beijing Jiaotong University, Beijing, China. She is currently working as a Visiting Student with Peng Cheng Laboratory, Shenzhen, China. Her research interests include heterogeneous cellular network, vehicular networks, mobile edge computing, resource allocation, and optimization techniques.



Yi Gong (S'99–M'03–SM'07) received the Ph.D. degree in electrical engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2002. He was with the Hong Kong Applied Science and Technology Research Institute, Hong Kong and Nanyang Technological University, Singapore. He is currently a Professor with the Southern University of Science and Technology, Shenzhen, China. From 2006 to 2018, he was on the Editorial Board of the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS and IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY. His research interests include cellular networks, mobile computing, and signal processing for wireless communications and related applications.



Ke Zhang received the Ph.D. degree in information and communication engineering from the University of Electronic Science and Technology of China, in 2017. He is currently a Lecturer with the School of Information and Communication Engineering, University of Electronic Science and Technology of China. His research interests include scheduling of mobile edge computing, design and optimization of next-generation wireless networks, smart grid, and the Internet of Things.