A Probabilistic Approach for Cooperative Computation Offloading in MEC-Assisted Vehicular Networks

Penglin Dai, *Member, IEEE*, Kaiwen Hu, Xiao Wu, *Member, IEEE*, Huanlai Xing, *Member, IEEE*, Fei Teng, *Member, IEEE*, and Zhaofei Yu, *Member, IEEE*

***Abstract*— Mobile edge computing (MEC) has been an effective paradigm for supporting computation-intensive applications by offloading resources at network edge. Especially in vehicular networks, the MEC server, is deployed as a small-scale computation server at the roadside and offloads computation-intensive task to its local server. However, due to the unique characteristics of vehicular networks, including high mobility of vehicles, dynamic distribution of vehicle densities and heterogeneous capacities of MEC servers, it is still challenging to implement efficient computation offloading mechanism in MEC-assisted vehicular networks. In this article, we investigate a novel scenario of computation offloading in MEC-assisted architecture, where task upload coordination between multiple vehicles, task migration between MEC/cloud servers and heterogeneous computation capabilities of MEC/cloud severs, are comprehensively investigated. On this basis, we formulate cooperative computation offloading (CCO) problem by modeling the procedure of task upload, migration and computation based on queuing theory, which aims at minimizing the delay of task completion. To tackle the CCO problem, we propose a probabilistic computation offloading (PCO) algorithm, which enables MEC server to independently make online scheduling based on the derived allocation probability. Specifically, the PCO transforms the objective function into augmented Lagrangian and achieves the optimal solution in an iterative way, based on a convex framework called Alternating Direction Method of Multipliers (ADMM). Last but not the least, we implement the simulation model. The comprehensive simulation results show the superiority of the proposed algorithm under a wide range of scenarios.**

***Index Terms*— Vehicular networks, mobile edge computing, computation offloading, queuing theory, optimization.**

I. Introduction

MOBILE edge computing (MEC) has been an effective paradigm for supporting computation-intensive applications with low latency requirements in vehicular networks by offloading resources at network edge [1]. Especially, the MEC, as one type of mainstream MEC-based architectures, is deployed as a mobility-enhanced small-scale data center at the fixed wireless infrastructures, such as Access Points (APs), Base Stations (BSs) and Roadside Units (RSUs), and capable of offloading the computation-intensive tasks from mobile vehicles in the local servers rather than uploading to the central cloud via wireless communication [2], [3], which greatly reduces task completion delay. Therefore, the MEC-based architecture is promised to be the fundamental of many intelligent transportation systems (ITSs), such as large-scale traffic sensing [4], pattern recognition with image processing [5] and virtual-reality-assisted driving [6]. However, due to the unique characteristics of vehicular networks [7], such as high mobility of vehicles, dynamic distribution of vehicle density, heterogeneous computation capacities of MEC/cloud servers, computation offloading still suffers from unbalance workload distribution among MEC servers, which can badly degrade the system performance.

Recently, great efforts have been focused on computation offloading in MEC-based vehicular networks. Some researchers proposed several MEC-based service architectures for computation offloading in vehicular networks. References [8], [9] applied software-defined network (SDN) to propose a programmable, flexible, and controllable network architecture, which can potentially improve resource utilization of MEC servers and achieve sustainable network development. Further, some researchers proposed several computation offloading strategies, which enable terminal users to adaptively offload excessive computation-intensive tasks to the MEC/fog nodes and improve overall system performance, such as energy consumption [10] and task completion delay [11]. However, these computation offloading strategies are implemented at terminal users based on the global knowledge of workload distribution, which takes high communication cost. Further, they have not considered the effect of vehicle mobility on wireless bandwidth competition of task upload. Particularly,

Manuscript received February 2, 2020; revised June 11, 2020 and July 17, 2020; accepted August 13, 2020. Date of publication August 28, 2020; date of current version February 2, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61802319, Grant 61772436, and Grant 61806011; in part by the China Postdoctoral Science Foundation Funded Project under Grant 2019M660245 and Grant 2020T130547; in part by the Foundation for Department of Transportation of Henan Province of China under Grant 2019J-2-2; in part by the Major Program of Scientific Frontier of Sichuan Province of China under Grant 2020YJ0207; in part by the Sichuan Science and Technology Program under Grant 2020YJ0272 and Grant 2019YJ0214; and in part by the Fundamental Research Funds for the Central Universities under Grant 2682018CX23. The Associate Editor for this article was J. Blum. *(Corresponding author: Penglin Dai.)*

Penglin Dai, Kaiwen Hu, Xiao Wu, Huanlai Xing, and Fei Teng are with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China, and also with the National Engineering Laboratory of Integrated Transportation Big Data Application Technology, Chengdu 611756, China (e-mail: penglindai@swjtu.edu.cn; kaiwenhu_swjtu@163.com; wuxiaohk@swjtu.edu.cn; hxx@home.swjtu.edu.cn; fteng@swjtu.edu.cn).

Zhaofei Yu is with the School of Electronics Engineering and Computer Science, Institute of Digital Media, Peking University, Beijing 100871, China (e-mail: yzf714@126.com).

Digital Object Identifier 10.1109/TITS.2020.3017172

cooperative task offloading between MEC/cloud servers by synthesizing multiple critical factors, including mobility features of vehicles, heterogeneous communication and computation capacities of cloud/MEC servers, are not investigated.

In this article, we comprehensively investigate the service scenario of cooperative computation offloading in MEC-assisted service architecture, where multiple MEC servers and remote cloud offload computation-intensive tasks in a cooperative way. Specifically, mobile vehicles are able to upload the task associated with data size and computation resources requirements to MEC servers at the coverage of wireless interface. The MEC is assumed to be wired connected to different types of wireless interfaces, which are characterized by different service rates of task upload. Further, the heterogeneous computation capacities of MEC servers are characterized by several key factors, such as processor number, computation rate, etc. The cloud server is supposed to own sufficient number of processors but far away from the end users. Hence, the task can always be assigned to an idle processor immediately, but has to tolerate additional transmission delay compared to MEC server. In this article, each MEC server is regarded as local scheduler and responsible for making online scheduling decision of new arrival tasks, which includes determining task upload, migration and computation server. To implement effective scheduling, the following issues are still critical to be addressed. First, wireless bandwidth competition among multiple vehicles may deteriorate the performance of task upload. Therefore, the coordination of task upload has to be designed to distribute task upload workload by considering mobility features of vehicles. Second, the dynamic workload distribution may make some MEC servers with weak computation capability overloaded, which results in serious service delay. Particularly, in dynamic vehicular environment, an online allocation approach has to be designed to achieve global optimal computation workload by synthesizing real-time workload and computation capacities of MEC/cloud servers. To sum up, as far as we know, this article is the first work dedicated to resolving the above issues of computation offloading in MEC-assisted vehicular networks. The contributions of this article are outlined as follows.

- We investigate the service scenario of computation offloading in MEC-assisted vehicular networks, where mobility features of vehicles, dynamic distributions of vehicle density and heterogeneous communication capabilities of MEC servers, are comprehensively studied. Particularly, the horizontal and vertical cooperations between MEC/cloud servers are utilized for balancing the workload distribution in dynamic vehicular environment.
- We formulate a problem of cooperative computation offloading (CCO) by theoretically modeling the procedure of task upload, migration and computation based on queuing theory, which aims at minimizing expected service delay. The task upload and migration procedure are characterized by $M/M/1$ model, where the time consumption is determined by dwelling time of vehicles, vehicle arrival rate and service rate of task upload and migration. Particularly, the task computation procedure of MEC and cloud server are simulated by $M/M/C$ and $M/M/\infty$ queuing models, where the heterogeneous capabilities of MEC servers are characterized by different processor numbers and processing rates.
- We propose a probabilistic computation offloading (PCO) algorithm, which enables the MEC server to online make scheduling decision independently based on a derived optimal allocation probability. Each MEC server has its own allocation probability, which is determined by parameter setting and adaptive to different service scenarios. Specifically, we first verify the convexity of the objective function by decomposing it into multiple components and then transforms the objective function of the CCO problem into the formulation of augmented Lagrangian. Based on the ADMM framework, the optimal solution is achieved in an iterative way.
- We build the comprehensive system model by integrating real-world map, realistic vehicular trace and schedule module. The extensive performance evaluation demonstrates the superiority of our proposed algorithms compared to four competitive algorithms in a wide range of service scenarios.

The remainder of this article is organized as follows. Section II reviews the related work. In Section III, the system model is presented. In Section IV, we formulate the CCO problem. Section V gives the algorithm design. Section VI gives a comprehensive simulation study. Finally, Section VII concludes this article.

II. Related Work

A. MEC-Based Service Architecture

In the last decades, several types of MEC-based service architectures have been proposed for improving system performance of vehicular networks from different perspectives. For edge caching, [12] proposed a novel cooperative content caching architecture, where RSUs, based stations and vehicles are regarded as edge cache servers and can predictively download data for reducing data access time. For edge communication, [13] proposed a data offloading architecture, where the MEC server is regarded as a small backhaul network for accelerating social context delivery and reducing the load of core network. Further, [14] proposed a multihomed nested vehicular networks, where mobile vehicles, such as buses and ambulances, are regarded as edge routers and responsible for choosing optimal routing path to support QoS-aware ITS services. For edge computing, [15] explored the novel collaborative vehicular edge computing network architecture, which includes design principles, corresponding functional modules, communication process, as well as installation and deployment ideas. They also discussed the promising technical challenges, including collaborative coalition formation, collaborative task offloading and mobility management. In particular, [16] proposed a comprehensive service architecture integrating edge caching, edge communication and edge computation, where resources sharing between vehicles is implemented by virtual machine migration. These studies focused on the architecture design of MEC-based vehicular networks, which reveals the potential benefits of MEC. However, more specific offloading models and mechanisms, particularly for computation offloading, which is the fundamental of ITS services, should be further investigated.

B. Computation Offloading Model in MEC-Based Vehicular Networks

Many efforts have been paid on investigating the computation offloading models of MEC-based vehicular networks based on different service scenarios. Reference [17] proposed a vehicular fog computing (VFC) model, where mobile vehicles are considered as the edge devices for task offloading without deployment cost. However, due to stochastic behavior of vehicles, the capability of VFC is unstable and the task may suffer from unexpected delay. Reference [18] conceived the base station (BS) as MEC servers to support real-time computation-intensive task, where the computation capability of MEC server is formulated as queuing model. They investigated the cooperation mechanism between multiple MEC servers via horizontal wired connection for improving the users' QoS in terms of both time saving and energy consumptions. However, as the computation capacities of MEC server are still limited, the task may tolerate serious delay when the MEC server is overloaded. To resolve this issue, [19] proposed a hybrid computation offloading model, where the task workload can be optimal distributed among MEC and cloud servers based on a theoretically derived threshold. To further improve resource utilization, [20] proposed a three-layer hierarchical architecture of computation offloading, which synthesizes the resources of mobile vehicles, RSUs and remote cloud. An optimization model integrating network, transmission and computation, is formulated for task offloading decision by minimizing energy consumption. However, the unique characteristics of vehicular networks, such as high mobility of vehicles, dynamic distribution of vehicle density, the heterogeneity of MEC servers, are not fully considered. Particularly, the computation offloading model combing both horizontal and vertical cooperation are not comprehensively investigated.

C. Optimization Mechanism for MEC-Based Computation Offloading

In addition, many studies have investigated various optimization mechanisms to derive efficient solutions of computation offloading in MEC-based vehicular networks. Reference [21] proposed a hybrid intelligent optimization algorithm based on genetic algorithm and heuristic rules to solve a joint task scheduling problem. Specifically, the problem is formulated as a mixed integer non-linear programming problem (MINLP), which not only determines where the tasks are performed, but also indicates the execution order of the tasks on the server. Reference [22] proposed a convex-based algorithm, called Interior point method (IPM) to achieve the optimal solution of a multi-objective optimization model based on queuing theory by considering energy consumption, execution delay, and payment cost. However, these two methods are iterative-based, which cannot be applied to online scheduling due to high time complexity in large scale network. To be more scalable, [23] designed a deep reinforcement learning (DRL) framework, where data cache, communication and computation are operated at multi-timescale levels. The mobility-aware reward estimation is proposed for fast parameter configuration to mitigate the complexity due to the large action space. However, the proposed DRL-based model is based on centralized scheduling, which cannot be applied to the distributed scenario, where each MEC server is regarded as an independent agent. To resolve this issue, [24] proposed a multiuser non-cooperative computation offloading game, where multiple vehicles compete for MEC resources. A distributed offloading algorithm is proposed to achieve optimal solution by converging to a unique Nash equilibrium. Further, [25] formulated the cooperative computation offloading game for task offloading between MEC server and terminal users based on group utility maximization. They proposed a social-aware Nash equilibrium and a reinforcement learning algorithm for strong and weak information cases, respectively. However, the game-based model requires information exchange between vehicles, such as decision and result feedback, where the communication overhead cannot be neglected with high vehicle density. Further, they only focused on the offloading between vehicles and MEC server, which cannot be directly applied to the MEC/cloud hybrid architecture.

Distinguished from previous works, our paper investigates a comprehensive computation offloading model by synthesizing both horizontal and vertical cooperation, where the mobility features of vehicles, dynamic vehicle density, heterogeneous capacities of MEC servers are comprehensively modeled through the procedure of task uploading, migration and computation offloading. Further, the proposed PCO algorithm is optimized offlined in advance based on the prediction of arrival task workload and enables each MEC server to schedule real-time task online in a distributed way.

III. System Model

In this section, we present a three-layer service architecture for cooperative computation offloading in MEC-assisted vehicular networks, as shown in Fig. 1, which includes mobile user layer, MEC layer and cloud layer. Specifically, in the mobile user layer, road network is divided into multiple sub-areas and each one is assumed to be covered by one type of wireless interfaces, such as APs, BSs and RSUs. Due to high mobility of vehicles, the vehicle density in one sub-area may be time-varying and evenly distributed among these sub-areas. Further, the vehicles in the same sub-area may compete for the wireless bandwidth of one wireless interface for task upload. The heterogeneous communication capacities of wireless interfaces are characterized by different service rates of task upload. For simplicity, the $M/M/1$ model is used for modeling task upload procedure, which indicates that vehicles have to wait in the task upload queue until the data of previous tasks is totally uploaded.

In the MEC layer, servers are assumed to be close to wireless interfaces and communicate with each other via short

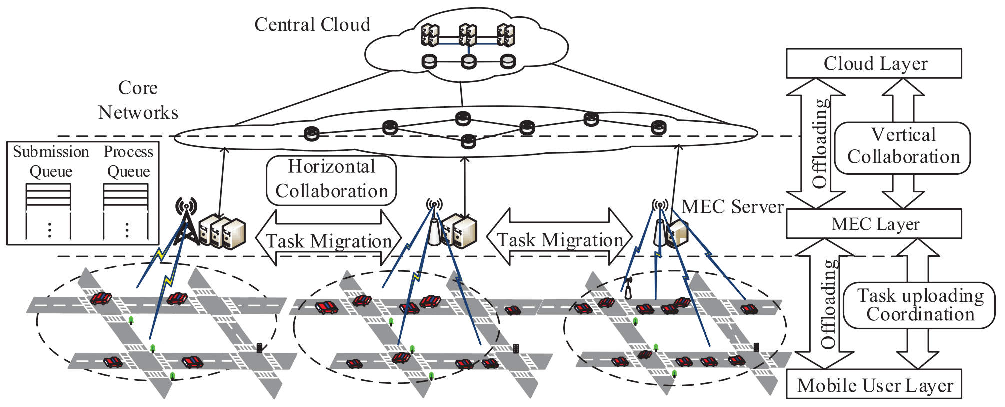


Fig. 1. The service architecture for cooperative computation offloading in MEC-assisted vehicular networks.

wired connection, as shown in Fig. 1. The task data uploaded at wireless interface is then transmitted to MEC server via wired connection. Further, the computation capacities of the MEC servers are characterized by different processor numbers, varying queuing capacities and diverse service rates of task computation. Each processor is assumed to compute one task at one time. When all the processors are busy, the task has to wait in the queue until at least one of the processors becomes available. The queuing capacity limits maximum number of pending tasks in the queue simultaneously. Therefore, the $M/M/C$ queuing model is adopted for simulating the task computation procedure of each MEC server.

In the cloud layer, the central cloud server is assumed to own unlimited number of processors, which indicates the task arrived at cloud server can be immediately assigned to an idle processor without pending delay. The $M/M/\infty$ queuing model is adopted to simulate the task computation procedure of the cloud. However, compared with MEC layer, task computation in the cloud layer has to take extra communication time of migrating task from the MEC layer to the cloud layer. When vehicle density is unevenly distributed, horizontal collaboration can effectively balance the workload in the MEC layer by efficiently migrating tasks from overloaded servers to underloaded servers. On the other hand, when the workload in MEC layer is overloaded, vertical migration can effectively avoid intolerant pending delay at MEC server by offloading extra tasks to the cloud server. Particularly, the $M/M/1$ queuing model is used for simulating task migration among MEC/cloud servers.

The MEC server is regarded as the scheduler for making scheduling decision for each submitted task in its service range. The detail scheduling procedure of the MEC server is described as follows. First, the beacon message periodically broadcast by the vehicles is monitored by the wireless interface, which includes task submission information, such as task ID and submission time, and mobility features, such as the velocity and driving direction. By receiving the collected information from the wired connected wireless interface, the MEC server maintains a submission queue for storing these new tasks. Second, the MEC server makes the scheduling decision of each task in the submission queue, including task upload server, task migration server and task computation server. Third, if the task upload server is equal to the dwelling server, the task ID will be pushed into the upload queue and the vehicle waits for uploading task data. Otherwise, the vehicle keeps silence until driving into the coverage of determined task upload server. Fourth, after task upload, the MEC server will check whether the task computation server is equal to the task upload server. If so, the task is then pushed into the computation queue in the local server. Otherwise, the task is pushed into the migration queue and migrated to the determined task computation server. Fifth, after task migration, the task waits in the computation queue until one of the processor is available. In this article, the cost of retrieving the computation result is not considered since the data size of computation result is always much smaller the task itself, which is commonly adopted in related literatures [26].

IV. Problem Formulation

A. Preliminary

In this section, we briefly introduce the notations used in this article. The set of MEC servers is denoted by M. Each $m_i \in M$ is assumed to be wired connected to one type of wireless interfaces, where the service range of m_i is determined by the wireless coverage. For each $m_i \in M$, the arrival pattern of vehicles follows the Poisson process with parameter λ_i. Each vehicle is assumed to submit one task, which requires unit communication resource (denoted by δ_s) and unit computation resource (denoted by δ_c). The dwelling time of vehicles in the service range of m_i is denoted by l_i. Additionally, ρ_{ij} denote the turning probability of vehicles driving from the service range of m_i to m_j. The communication capability of m_i is determined by the connected wireless interface, which is characterized by the service rate of uploading one task, denoted by u_i^u. The computation capability of $m_i \in M$ is characterized by three-tuple (u_i^p, p_i, c_i), where u_i^p, c_i and n_i represent the service rate of computing task, the number of processors owned by m_i and the maximum number of tasks tolerated by m_i at the same time, respectively. The service rate of migrating one task between m_i and m_j is denoted by u_{ij}^m. The cloud server is denoted by m_0. The service rate of any processors owned by m_0 is denoted by u_0^p. It is noted that the service time of task upload, migration and computation are assumed to follow the exponential distribution with the corresponding service rate based on the queuing

TABLE I
SUMMARY OF NOTATIONS

Symbol	Description
M	The set of MEC servers, where $M = \{m_i\}$
δ_s	The unit communication resource
δ_c	The unit computation resource
λ_i	The arrival rate of vehicles in the coverage of m_i
ρ_{ij}	The probability of vehicles that drive from m_i to m_j
l_i	The dwelling time of vehicles in the coverage of m_i
u_i^u	The service rate of uploading one task by m_i
c_i	The number of processors owned by m_i
n_i	The queuing capacity of m_i
u_i^p	The service rate of computing a task by m_i
u_{ij}^m	The service rate of migrating a task from m_i to m_j
$P_{ij\to k}^{(l)}$	The probability that the task of m_i is uploaded at m_i ($l=1$) or m_j ($l=2$) and computed by m_k.

theory. Further, the allocation probability $P_{ij\to k}^{(l)}$ is used for indicating the probability that the task submitted to m_i is uploaded at m_i ($l = 1$) or m_j ($l = 2$) and computed by m_k. The primary notations are listed in the Table I.

B. Cooperative Computation Offloading Problem

In this section, we formally introduce the cooperative computation offloading (CCO) problem in detail, which consists of task upload model, task migration model and task computation model, respectively.

First, we establish the task upload model for each MEC server $m_i \in M$ based on $M/M/1$ queuing model. For each $m_i \in M$, given arrival rate λ_i, the expected number of new submitted tasks in m_i choosing m_i for task upload is computed by $\sum_{j=1}^{M}\sum_{k=0}^{M} \lambda_i \rho_{ij} P_{ij\to k}^{(1)}$. Similarly, for $m_j \in M$, $j \neq i$, the expected number of new arrival tasks that choose m_i for task upload is computed by $\sum_{k=0}^{M} \lambda_j \rho_{ji} P_{ji\to k}^{(2)}$. Then, the expected upload workload of m_i is formulated as follows.

$$\lambda_i^u = \sum_{j=1}^{M}\sum_{k=1}^{M}\left(\lambda_i \rho_{ij} P_{ij\to k}^{(1)} + \lambda_j \rho_{ji} P_{ji\to k}^{(2)}\right)\cdot \delta_s \tag{1}$$

Based on $M/M/1$ queuing model, given expected upload workload λ_i^u and service rate of task upload u_i^u, the expected task upload time of m_i is formulated as follows.

$$T_i^u = \frac{1}{u_i^u - \lambda_i^u} \tag{2}$$

As the vehicle has to complete task upload before it leaves the service range of m_i, the expected upload time at each m_i cannot exceed the dwelling time l_i, expressed as follows.

$$T_i^u \le l_i, \quad \forall m_i \in M \tag{3}$$

Therefore, the expected task upload time of the system is computed as follows.

$$E(T^u) = \frac{\sum_{i=1}^{M} \lambda_i^u T_i^u + \sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_i \rho_{ij} P_{ij\to k}^{(2)} l_i}{\sum_{i=1}^{M} \lambda_i} \tag{4}$$

where $\sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_i \rho_{ij} P_{ij\to k}^{(2)} l_i$ is the summation of dwelling time tolerated by vehicles before entering the service range of task upload server.

Second, we model the procedure of task migration between two MEC servers based on $M/M/1$ queuing model. Specifically, the workload of task migration from m_i to m_j, denoted by λ_{ij}^m, is calculated as the product of the number of tasks that choose m_i for task upload and m_j for task computation and unit communication resource, which is expressed as follows.

$$\lambda_{ik}^m = \sum_{j=1}^{M} (\lambda_i \rho_{ij} P_{ij\to k}^{(1)} + \lambda_j \rho_{ji} P_{ji\to k}^{(2)}) \cdot \delta_s \tag{5}$$

Based on $M/M/1$ queuing model, the expected task migration time between m_i and m_k, is computed as follows.

$$T_{ik}^m = \frac{1}{u_{ik}^m - \lambda_{ik}^m} \tag{6}$$

Further the expected migration workload λ_{ik}^m cannot exceed the service rate of task migration u_{ik}^m, otherwise the service time will achieve ∞. Therefore the following constraint has to be satisfied.

$$\lambda_{ik}^m \le u_{ik}^m, \quad \forall m_i, m_j \in M \tag{7}$$

Particularly, task migration workload from the MEC layer to the cloud layer is the summation of the workload of task migration from $m_i \in M$ to m_0, which is computed as follows.

$$\lambda_0^m = \sum_{i=1}^{M}\sum_{j=1}^{M} \lambda_i \left(\rho_{ij} P_{ij\to 0}^{(1)} + \rho_{ij} P_{ij\to 0}^{(2)}\right)\cdot \delta_s \tag{8}$$

Then, the expected task migration time from the MEC layer to the cloud layer is computed as follows.

$$T_0^m = \frac{1}{u_0^m - \lambda_0^m} \tag{9}$$

Similarly, the following constraint should be satisfied.

$$\lambda_0^m \le u_0^m \tag{10}$$

Therefore, the expected task migration time of the system is computed by the summation of task migration time between MEC and cloud servers divided by the total expected number of new arrival tasks in the system.

$$E(T^m) = \frac{\sum_{i=1}^{M}\sum_{k=1}^{M} \lambda_{ik}^m T_{ik}^m + \lambda_0^m T_0^m}{\sum_{i=1}^{M} \lambda_i} \tag{11}$$

It is noted the task migration time is affected by dynamic distribution of vehicle density. Higher uneven distribution of vehicles, more numbers of tasks need to be migrated between MEC servers for load balance. Accordingly, the value of λ_{ik}^m increases and results in the increasing of task migration time.

Third, we utilize $M/M/C$ and $M/M/\infty$ queuing model to establish the task computation model of MEC and cloud servers, respectively. For each $m_i \in M$, the expected computation workload assigned to m_k is computed as follows.

$$\lambda_i^p = \sum_{k=1}^{M}\sum_{j=1}^{M} \lambda_k \rho_{kj} \left(P_{kj\to i}^{(1)} + P_{kj\to i}^{(2)}\right)\delta_c, \quad \forall m_i \in M \tag{12}$$

Based on $M/M/C$ queuing model, given expected task computation workload λ_i^p, the processor number c_i, service rate of task computation u_i^p and queuing capacity n_i, the expected task computation time of m_i is formulated as follows.

$$T_i^p = \frac{\sum_{k=0}^{c_i} k \frac{(\lambda_i^p/u_i^p)^k}{k!} + \sum_{k=c_i+1}^{n_i} k \frac{\left(\frac{\lambda_i^p}{c_i u_i^p}\right)^k \cdot c_i{}^{c_i}}{c_i!}}{\lambda_i^p \left(\sum_{k=0}^{c_i} \frac{(\lambda_i^p/u_i^p)^k}{k!} + \sum_{k=c_i+1}^{n_i} \frac{\left(\frac{\lambda_i^p}{c_i u_i^p}\right)^k \cdot c_i{}^{c_i}}{c_i!}\right)} \tag{13}$$

For cloud server m_0, based on $M/M/\infty$ model, the expected computation workload and computation time are formulated as follows.

$$\lambda_0^p = \sum_{k=1}^{M}\sum_{j=1}^{M} \lambda_k \rho_{kj} \left(P_{kj\to 0}^{(1)} + P_{kj\to 0}^{(2)}\right)\delta_c \tag{14}$$

$$T_0^p = \frac{1}{u_0^p} \tag{15}$$

Based on Eqs. (12) $\sim$ (15), the expected task computation time of the system is formulated as follows.

$$E(T^p) = \frac{\lambda_0^p \cdot T_0^p + \sum_{i=1}^{M} \lambda_i^p \cdot T_i^p}{\sum_{i=1}^{M} \lambda_i} \tag{16}$$

It is noted that even with the same workload, the computation time of tasks can still differ with each other due to heterogeneous computation capabilities of MEC/cloud servers, which indicates unbalanced workload distribution.

Based on Eqs. (4), (11) and (16), the expected service delay is defined as the summation of expected task upload time, expected task migration time and expected task computation time, which is formulated as follows.

$$E(T^{delay}) = E(T^u) + E(T^m) + E(T^p) \tag{17}$$

The unbalanced workload distribution among MEC servers will increase the value of $E(T^u)$, $E(T^m)$ and $E(T^p)$, which leads to overlong service delay of tasks.

The formulation of the CCO problem is presented as follows. Given the arrival rate λ_i and the dwelling time l_i, service rate of task upload u_i^u, task migration u_{ij}^u and task computation u_i^p, the objective of the CCO problem is to minimize both expected service delay by searching the optimal allocation probability $\boldsymbol{P} = [P_{ij\to k}^{(l)}]$, $\forall i, j \in [1, M], k \in [0, M]$, $l \in \{1, 2\}$, which is formulated as follows.

$$\begin{aligned}
&\min_{\forall P_{ij\to k}^l} \ E(T^{delay})\\
&s.t.\ C_1: T_i^u \le l_i, \quad \forall i \in [1, M]\\
&\qquad C_2: \lambda_0^u \le u_0^u,\\
&\qquad C_3: \lambda_i^p \le c_i u_i^p, \quad \forall i \in [1, M]\\
&\qquad C_4: \lambda_{ij}^m \le u_{ij}^m, \quad \forall i, j \in [1, M]\\
&\qquad C_5: \sum_{k=1}^{M}\sum_{l=1}^{2} P_{ij\to k}^l = 1, \quad \forall i, j \in [1, M]\\
&\qquad C_6: 0 \le P_{ij\to k}^l \le 1, \quad \forall i, j \in [1, M],\\
&\qquad\quad k \in [0, M], l \in 1, 2
\end{aligned} \tag{18}$$

Based on the formulated model in Eq. (18), we can have the following observations. First, the high mobility of vehicles is characterized by the dwelling time of vehicles in the coverage of MEC server and the turning probability of vehicles that drive from the coverage of one MEC server to another. Second, the dynamic distribution of vehicle density is characterized by different vehicle arrival rates at the coverage of MEC servers, which represents varying workload distribution. Third, the heterogeneous computation capacities of MEC/cloud servers are characterized by different computation models, where MEC and cloud server are characterized by $M/M/C$ queuing model and $M/M/\infty$ queueing model. Particularly, MEC servers have different processor numbers and varying service rates. Fourth, the above factors can jointly cause unbalanced workload distribution among MEC servers, which results in serious service delay. Therefore, the objective of the optimization model is to minimize the service delay by deriving the optimal offloading probability.

V. Algorithm Design

In this section, we propose probabilistic computation offloading (PCO) algorithm, where the basic principle of the PCO is described as follows. First, The PCO achieves the optimal allocation probability for each MEC server by minimizing the service delay with satisfying the constraints of the CCO model. Accordingly, each MEC server can independently schedule the offloading decision for each pending task in its coverage in a probabilistic way. The allocation probability is determined by the given parameters including mobility features of vehicles (i.e., dwelling time and turning probability), vehicle density distributions (i.e., vehicle arrival rate) and computation capabilities of MEC/ loud servers (i.e., the number of processors and the computation rate). Therefore, the allocation probability differs in MEC servers with different service scenarios, which can adaptively achieve balanced workload distribution among MEC/cloud server. Specifically, we first prove the convexity of the objective function by decomposing it into multiple components and analyzing the convexity of each component. Further, we achieve the optimal allocation probability in an iterative way and implement the computation offloading strategy at each MEC server based on derived allocation probability.

A. The Convexity of Objective Function

In this section, we analyze the convexity of the objective function $E(T^s)$. Based on Eq. (17), $E(T^s)$ can be decomposed into three components: $E(T^u)$, $E(T^m)$ and $E(T^c)$, which will be analyzed respectively as follows.

First, based on Eq. (4), $E(T^u)$ is the affine mapping of $\lambda_i^u T_i^u$ and $\lambda_i \rho_{ij} P_{ij\to k}^{(2)}$. Based on convex theory [27], affine mapping preserves the convexity property. Therefore, we only need to prove the convexity of $\lambda_i^u T_i^u$. We derive the derivative and the second derivative of a general function $f(\lambda) = \frac{\lambda}{u-\lambda}$ with respect to $\lambda \in [0, u)$, which is shown in Eq. (19).

$$f'(u) = \frac{u}{(u-\lambda)^2}, \quad f''(u) = \frac{2u}{(u-\lambda)^3} \tag{19}$$

As $f''(u) > 0$ with respect to $\lambda \in [0, u)$, $f(\lambda)$ is proved to be a convex function. Further, λ_i^u is the affine mapping of $P_{ij\to k}^l$, which preserves the convexity. Therefore, the first component T^u is proved to be convex with respective to $\lambda_i^u \in [0, u_i^u], i = 1, 2, \ldots, M$.

Second, based on Eq. (11), $E(T^m)$ is the affine mapping of $\lambda_{ik}^m T_{ik}^m$ and $\lambda_0^m T_0^m$. Since the structure of $\lambda_{ik}^m T_{ik}^m$ is the same to $\lambda_i^u T_i^u$, then $\lambda_{ik}^m T_{ik}^m$ is convex with respect to $\lambda_{ik}^m \in [0, u_{ik}^m]$. Therefore, the second component T^m is proved to be convex with respective to $\lambda_{ik}^m \in [0, u_{ik}^m], i, k = 1, 2, \ldots, M$.

Third, based on Eq. (16), $E(T^p)$ is the affine mapping of $\lambda_i^p T_i^p$, which indicates that we only need to prove the convexity of $\lambda_i^p T_i^p$. For generality, we analyze the general function, which is formulated as follows.

$$f(x) = \frac{\sum_{k=0}^{c} k\frac{c^k}{k!}x^k + \sum_{k=c+1}^{n} k\frac{c^c}{c!}x^k}{\sum_{k=0}^{c} \frac{c^k}{k!}x^k + \sum_{k=c+1}^{n} \frac{c^c}{c!}x^k} \tag{20}$$

where c, n and $x = \frac{\lambda}{cu}$ are processor number, queuing capacity and the ratio of task computation workload and computation rate of MEC server, respectively. Since the second derivative of Eq. (16) is too complicated to be calculated and analyzed, we plot the curves of the relationship between $f(x)$, processor number c and queuing capacity n, respectively. In practical application, the processor number of MEC server and queuing capacity is usually not large due to high deployment cost and overlong delay pending in the computation queue. Therefore, we test the value of $\lambda_i T_i^p$ under processor number and queuing capacity varying with [1,10] and [20,100], respectively, shown in Figs. (2) and (3). It is observed that $f(x)$, i.e., $\lambda_i T_i^p$ under different processor numbers and queuing capacities are presented to be convex with respect to $x \in [0, 1)$. Though the convexity of $f(x)$ is not proved in the theoretical way, we can boldly make the assumption that the $\lambda_i T_i^p$ is convex with respect to $\lambda_i^p \in [0, u_i^p c_i)$ with the constraint of $c_i \in [2, 10]$ and $n_i \in [20, 100]$. This assumption will be further validated by extensive simulation results that the solution will finally converge based on the proposed convex-based approach in Section VI.

B. Probabilistic Computation Offloading Algorithm

In this article, we will introduce the probabilistic computation offloading (PCO) algorithm based on ADMM. The ADMM [28] is a powerful optimization method, which can be applied to a variety of applications, especially popularly applied in wireless network [29], [30]. Particularly, the ADMM approach is still useful in solving non-convex problem by achieving a local or global optimal solution, which is suitable for solving the CCO problem. We use the PCO algorithm to derive the allocation probability offline, which can be solved by the powerful cloud server. Further, the solution can be performed online at each MEC server independently with neglected communication overhead of control message between MEC and cloud servers.

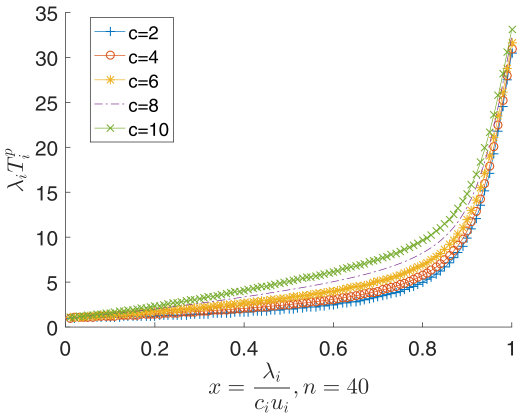


Fig. 2. The curve of $\lambda_i T_i^p$ under different processor numbers.

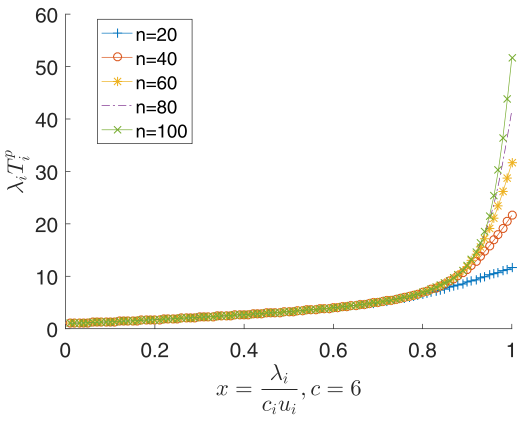


Fig. 3. The curve of $\lambda_i T_i^p$ under different queuing capacities.

First, in order to utilize the ADMM approach, the COO problem has to be transformed into the form of augmented Lagrangian. We add an N_z-dimension virtual variable vector $\boldsymbol{Z} = \{z_i\}$, where $\boldsymbol{Z} \in R_+^{N_z}$ and $N_z = M^3 + 3M^2 + 2M + 1$, to transform the inequality constraints into the set of equations. Then, Eq. (18) is transformed as follows.

$$\begin{aligned}
&\min_{\forall P_{ij\to k}^l} \; E(T^{delay}) \\
&s.t.\; C_1 : \lambda_i^u + z_i - l_i = 0, \quad \forall i \in [1, M] \\
&\quad\;\; C_2 : \lambda_0^u + z_{1+M} - u_0^u = 0, \\
&\quad\;\; C_3 : \lambda_i^p + z_{i+M+1} - c_i \cdot u_i^p = 0, \quad \forall i \in [1, M] \\
&\quad\;\; C_4 : \lambda_{ij}^m + z_{(i+1)M+j+1} - u_{ij}^m = 0, \quad \forall i, j \in [1, M]
\end{aligned}$$

$$\begin{aligned} C_5 &: \sum_{k=1}^{M}\sum_{l=1}^{2} P_{ij\to k}^{l} + z_{(i+1+M)M+j+1} - 1 = 0, \\ &\forall i, j \in [1, M] \\ C_6 &: P_{ij\to k}^{l} + z_{i(2M^2+2M)+j(2M+2)+2k+l-2M-3} - 1 = 0, \\ &\quad \forall i, j \in [1, M], k \in [0, M], l \in \{1, 2\} \end{aligned} \tag{21}$$

It is observed that th constraints $C_1 \sim C_6$ are linear, then Eq. (21) can be formulated as follows.

$$\begin{aligned} &\min_{\forall \boldsymbol{P}, \boldsymbol{Z}} f(\boldsymbol{P}) + g(\boldsymbol{Z}) \\ &s.t.\ A\boldsymbol{P} + B\boldsymbol{Z} = C \end{aligned} \tag{22}$$

Particularly, $g(\boldsymbol{Z})$ is an indicator function where $g(\boldsymbol{Z}) = 0$ if each $z_i \in \boldsymbol{Z} \in [0, 1]$. Otherwise, $g(\boldsymbol{Z}) = +\infty$. $\boldsymbol{P}$ is an N_p dimensional vector ($N_p = 2M^2(M+1)$), where the index of the element $P_{ij\to k}^{l}$ is $(i-1)*2M(M+1)+(j-1)*2(M+1)+(k-1)*2+l$. Accordingly, A is a $N_z \times N_p$ matrix, where each element a_{kl} is the coefficient of l_{th} element of $\boldsymbol{P}$ in the k_{th} constraint. Then, B is an $N_z \times N_z$ diagonal matrix and C is an N_z dimensional vector.

On this basis, we can transform Eq. (22) into the augmented Lagrangian, which is formulated as follows.

$$\begin{aligned} L_\rho(\boldsymbol{P}, \boldsymbol{Z}, \boldsymbol{U}) &= f(\boldsymbol{P}) + g(\boldsymbol{Z}) + \boldsymbol{U}(A\boldsymbol{P} + B\boldsymbol{Z} - C) \\ &\quad + \frac{\rho}{2}||A\boldsymbol{P} + B\boldsymbol{Z} - C||_2^2 \end{aligned} \tag{23}$$

where $\boldsymbol{U}$ is the dual variable vector and $\rho > 0$ is the step size for controlling the convergence speed.

$$\begin{aligned} \boldsymbol{P}^{k+1} &= arg \min_{\forall \boldsymbol{P}} L_\rho(\boldsymbol{P}, \boldsymbol{Z}^k, \boldsymbol{U}^k) \\ &\Leftrightarrow arg \min_{\forall \boldsymbol{P}} \{f(\boldsymbol{P}) + g(\boldsymbol{Z^k}) + \boldsymbol{U^k}(A\boldsymbol{P} + B\boldsymbol{Z}^k) \\ &\quad + \frac{\rho}{2}||A\boldsymbol{P} + B\boldsymbol{Z}^k - C||_2^2\} \end{aligned} \tag{24}$$

$$\begin{aligned} \boldsymbol{Z}^{k+1} &= arg \min_{\forall \boldsymbol{Z}} L_\rho(\boldsymbol{P}^{k+1}, \boldsymbol{Z}, \boldsymbol{U}^k) \\ &\Leftrightarrow arg \min_{\forall Z} \{g(\boldsymbol{Z^k}) + \boldsymbol{U^k}B\boldsymbol{Z} \\ &\quad + \frac{\rho}{2}||A\boldsymbol{P}^{k+1} + B\boldsymbol{Z} - C||_2^2\} \end{aligned} \tag{25}$$

$$\boldsymbol{U}^{k+1} = \boldsymbol{U}^k + \rho(A\boldsymbol{P}^{k+1} + B\boldsymbol{Z}^{k+1} - C) \tag{26}$$

Based on the ADMM approach, the solution consists of three iterations, which are formulated in Eq.(24)~(26). It is noticed that the Eqs. (24) and (25) are unconstrained convex functions, which can be solved by standard convex optimization tools. Further, to evaluate the stopping criterion, the primal and dual residuals are formulated in Eq. (27), respectively, which is used for checking whether to terminate the iteration or not by comparing with two thresholds ϵ^{pri} and ϵ^{dual} at each iteration.

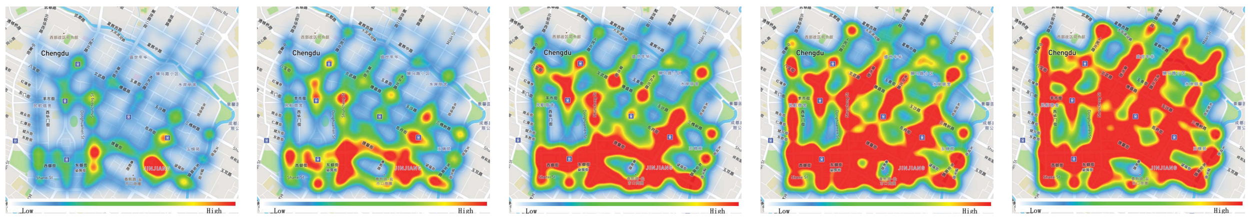
$$r = A\boldsymbol{P}^k + B\boldsymbol{Z}^k - C \tag{27}$$

$$s = A^T B(\boldsymbol{Z}^k - \boldsymbol{Z}^{k-1})\boldsymbol{P} + B\boldsymbol{Z} - C \tag{28}$$

On this basis, we propose the PCO algorithm, where the Pseudocode is shown in Alg. 1 and the detail procedure is described as follows. First, we initialize the related parameters, including ϵ^{pri}, ϵ^{dual}, $\boldsymbol{P}^0$, $\boldsymbol{Z}^0$ and $\boldsymbol{U}^0$. Specifically, each element $P_{ij\to k}^{l} \in \boldsymbol{P}$ is set to $\frac{1}{2M^2(M+1)}$ and $\boldsymbol{Z}, \boldsymbol{U}$ are set to zero vectors. To guarantee the efficiency, ϵ^{pri} and ϵ^{dual} are set to 0.5 for accelerating convergence. Second, we update $\boldsymbol{P}^k$, $\boldsymbol{Z}^k$ and $\boldsymbol{U}^k$ in sequential order, which are based on Eqs. (24), (25) and (26), respectively. Third, two residuals r and s are computed based on Eq. 27. If both $||r||_2 \le \epsilon^{pri}$ and $||s||_2 \le \epsilon^{dual}$ are satisfied, the iteration terminates. Otherwise, the iteration continues until the maximum iteration number is achieved. After that, the allocation probability P is obtained and the cumulative probability function $P_{ij}^c(k, l)$ is computed

Algorithm 1 Probabilistic Computation Offloading (PCO)

```
Step 1 (Offline): derive the optimal allocation probability
1: Initialize $\epsilon^{pri}$, $\epsilon^{dual}$, $\boldsymbol{P}^0$, $\boldsymbol{Z}^0$ and $\boldsymbol{U}^0$
2: while convergence_indicator == 0 do
3:   Update $\boldsymbol{P}(t)$ based on Eq. (24)
4:   Update $\boldsymbol{Z}(t)$ based on Eq. (25)
5:   Update $\boldsymbol{U}(t)$ based on Eq. (26)
6:   Calculate $||r(t)||_2 = ||A\boldsymbol{P}(t) + B\boldsymbol{Z}(t) - C||_2$
7:   Calculate $||s(t)||_2 = ||A^T B(\boldsymbol{Z}(t) - \boldsymbol{Z}(t-1))||_2$
8:   if ($||r(t)||_2 \le \epsilon^{pri}$ and $||s(t)||_2 \le \epsilon^{dual}$) or $t >= t_{max}$ then
9:     convergence_indicator = 1
10:  end if
11:  $t = t + 1$
12: end while
13: $\boldsymbol{P} \leftarrow \boldsymbol{P}(t)$
14: Compute $\boldsymbol{P}^c$ based on Eq. (29)
Step 2 (Online): make scheduling decision of new tasks
15: while there exists a new task submitted by vehicles do
16:   $m_i \leftarrow$ the dwelling server, $m_j \leftarrow$ the next dwelling server
17:   Randomly generate $tmp$ from interval [0, 1]
18:   for $k$ from 0 to $||M||$ do
19:     for $l$ from 1 to 2 do
20:       if $k == 1$&& $l == 1$ && $tmp < P_{ij}^c(k, l)$ then
21:         select_indicator =1
22:       else if $k == 1$ && $l == 2$ && $tmp \ge P_{ij}^c(k+1, l-1)$ && $tmp < P_{ij}^c(k, l)$ then
23:         select_indicator ==1
24:       else if $tmp \ge P_{ij}^c(k-1, l)$ && $tmp < P_{ij}^c(k, l)$ then
25:         select_indicator ==1
26:       end if
27:       if select_indicator ==1 then
28:         if $l == 1$ then
29:           $m^u \leftarrow i$
30:         else
31:           $m^u \leftarrow j$
32:         end if
33:         $m^c \leftarrow k$
34:         if $m^u == m^c$ then
35:           $m^c \leftarrow \emptyset$
36:         else
37:           $m^c \leftarrow k$
38:         end if
39:       end if
40:     end for
41:   end for
42: end while
```



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3 (d) Scenario 4 (e) Scenario 5

Fig. 4. The heat map of the distribution of vehicle densities under different traffic scenarios.

based on Eq. (29).

$$P_{ij}^{c}(k,l) = \begin{cases} P_{ij\to k}^{(l)} & if\ k == 1, l == 1 \\ P_{ij}^{c}(M+1, l-1) + P_{ij\to k}^{(l)} & if\ k == 1, l == 2 \\ P_{ij}^{c}(k-1, l) + P_{ij\to k}^{(l)} & otherwise \end{cases} \tag{29}$$

The above procedure shown in lines 1 $\sim$ 14 can be implemented offline efficiently by powerful cloud server, since the vehicle arrival rate can be predicted or estimated based on traffic flow prediction techniques [31]. Based on $P_{ij}^{c}(k,l)$, the MEC server can make the scheduling decision for the new tasks online. For each new task, a random variable tmp is generated from [0,1]. Then, the cloud server determines the probability interval that tmp belongs to in an iterative way. Once the probability interval is determined, the MEC server determines task upload, migration and computation servers. The detail procedure is shown in lines 17 $\sim$ 42 of Alg. 1. The oneline phase only takes $O(2M)$ time for scheduling each new task, which is linear to the number of MEC servers.

VI. Performance Evaluation

A. Default Setting

In this section, we implement the simulation model based on the system model presented in Section III. The real-world map is based on the core area within the first ring road, Chengdu City, China. The realistic vehicular trajectories, are extracted on 7th, November, 2018, as traffic inputs, which is provided by Didi Chuxing GAIA Initiative [32]. We have examined five service scenarios with different time periods, where the traffic characteristics are shown in Table II. Detailed statistics includes average arrival rate (AAR) of vehicles at MEC server, the variance of arrival rate (VAR), average dwelling time (ADT) of vehicles at MEC server and variance of dwelling times (VDT). Specifically, the traffic workload increases from Scenario 1 to 5, as well as the dwelling time. The high value of VAR and VDT indicates the dynamic mobility feature of vehicles. To better exhibit the system workload under different scenarios, Fig. 4 shows the heat maps of vehicle distribution under different traffic scenarios. It is noted that the vehicle density varies greatly with different scenarios and is highly uneven distributed in each scenario, which demonstrates serious unbalanced workload.

There are 1 cloud and 9 MEC servers simulated in the system, where the MEC servers are uniformly distributed in the road map. The probability of generating a computation-intensive task by one vehicle is set to 0.2. Each task is assumed to require 50 Giga CPU cycles with 5 MB data. For each MEC server, the service rate of task upload u_i^u and task migration u_{ij}^m are randomly generated from the interval [4.8, 7.2] and [9.6, 12] tasks per minute, which represents for the transmission rate of [3.2, 4.8] Mb/s and [6.4, 8] Mb/s, respectively. Further, the task computation rate u_i^p is randomly selected from [1.2, 2.4] tasks per minute, which represents for the computation rate of [1.0, 2.0] GHz. Particularly, the processor number and queuing capacity of each MEC server is randomly generated from [1, 4] and [25, 35], respectively. For central cloud, the number of processors is unlimited and the computation rate is set to 2.4 task per minute. The similar parameter settings are also adopted in these existing literatures [33]–[35]. For algorithm implementation, ϵ^{pri} and ϵ^{dual} are set to 0.5. The step size ρ starts from 0.2 and decreases with 0.01 until ρ achieves 0.01. Unless stated otherwise, the simulation is conducted under default settings.

For performance comparison, since there exists no feasible solution which can be directly applied in the CCO problem, two heuristic algorithms, i.e., Local Server Only (LSO) and Uniform Selection (US) are adopted for performance evaluation. Besides, two competitive algorithms in the literatures, called game-based computation offloading (GCO) [24] and cooperative computation offloading at MEC (CCO-MEC) [18], are tailored to be suited to the proposed model. The details are described as follows.

- GCO, is a multi-user non-cooperative computation offloading game, where each vehicle adjusts the offloading probability by striking the best load balance between MEC and cloud servers via vertical cooperation.
- CCO-MEC, is a convex-optimization-based algorithm, which minimizes the service delay of tasks via horizontal cooperation between MEC servers.
- LSO, always prefers to upload and compute the tasks in the local server.
- US, randomly chooses task upload and computation servers among available candidate with equal probability.

For performance evaluation, we collect the following statistics: the submission time and completion time of each task, denoted by $Q_i^{submission}$ and Q_i^{finish}; upload time and computation time of each task, denoted by Q_i^{upload} and $Q_i^{computation}$; the total number of task and the number of completed task, denoted by Q^{total} and $Q^{completed}$. On this basis, we define the following metrics.

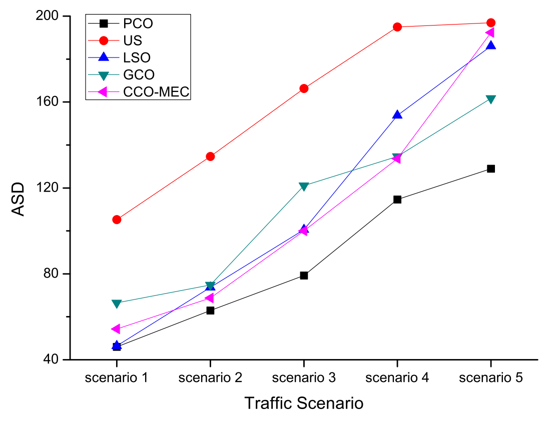


Fig. 5. The ASD of five algorithms under different traffic scenarios.

TABLE II
TRAFFIC CHARACTERISTICS

Scenario	Time period	AAR(veh/min)	VAR	ADT(s)	VDT
1	0:00 - 2:00	6.8	10.2	98.7	186.7
2	22:00 - 24:00	12.7	42.8	109.2	212.6
3	20:00 - 22:00	19.4	109.0	122.6	143.3
4	16:00 - 18:00	25.1	186.5	150.7	421.0
5	8:00 - 10:00	31.7	100.9	150.6	389.9

- Average service delay (ASD): it is defined as the summation of service delay of all completed tasks divided by the number of completed tasks, which is computed as follows.

$$ASD = \frac{\sum_{i=1}^{Q^{completed}} (Q_i^{finish} - Q_i^{submission})}{Q^{total} - Q^{fail}} \tag{30}$$

Low value of ASD indicates that the vehicle takes less delay for receiving the computation result, which brings better user experience.

- Average upload time (AUT): it is defined as the mean value of task upload time of all the completed tasks, which is computed as follows.

$$AUT = \frac{\sum_{i=1}^{Q^{completed}} Q_i^{upload}}{Q^{completed}} \tag{31}$$

Higher value of AUT indicates that more tasks compete for wireless bandwidth, which degrades the ASD.

- Average computation time (ACT): it is defined as the mean value of task computation time of all the completed tasks, which is computed as follows.

$$ACT = \frac{\sum_{i=1}^{Q^{completed}} Q_i^{computation}}{Q^{completed}} \tag{32}$$

Higher value of AUT indicates higher pending delay for task uploading, which further degrades the ASD.

B. Simulation Result

1) Effect of Traffic Workload: Fig. 5 compares the ASD of five algorithms under different traffic workloads. As noted in Table II, the vehicle arrival rate increases from Scenario 1 to 5, which results in heavier system workload. Accordingly, the ASD of five algorithms increases. Specifically, the ASD of CCO-MEC and LSO achieves at a low lever at light workload

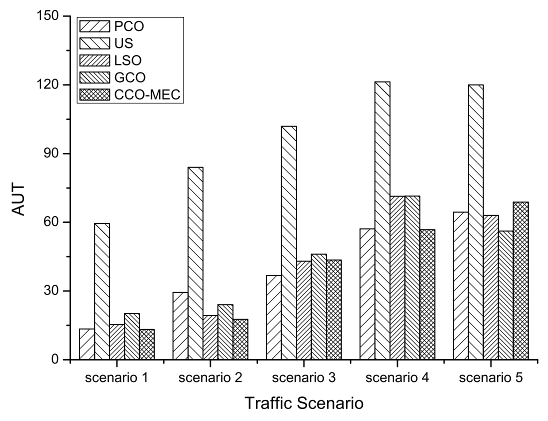


Fig. 6. The AUT of five algorithms under different traffic scenarios.

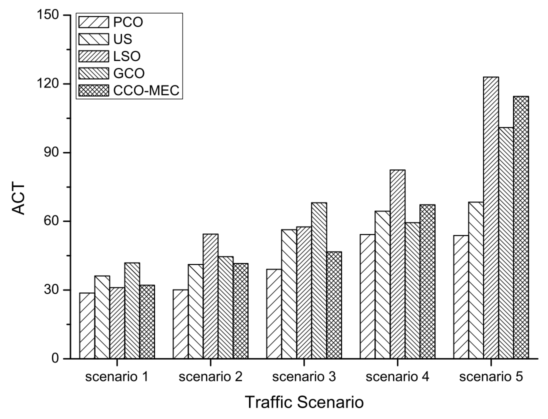
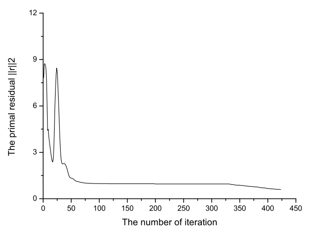


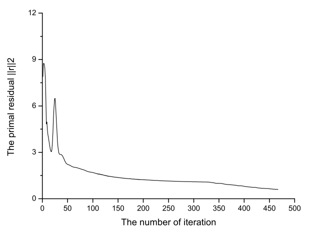
Fig. 7. The ACT of five algorithms under different traffic scenarios.

and increases dramatically when the workload becomes heavy. The reason behind is that neighboring MEC can cooperate with each other to handle the unbalanced workload. However, when the workload exceeds the computation capability of MEC servers, the computation time will explosively increase. It is demonstrated in Fig. 7, where the ACT of LSO and CCO-MEC becomes much higher than other algorithms at heavy workload. Even though, the ACT of CCO-MEC can be lower than that of LSO since CCO-MEC uses horizontal cooperation to alleviate workload unbalance. Reversely, the ASD of GCO is higher than that of CCO-MEC at first but then becomes lower than CCO-MEC. It is because GCO can adaptively offload the task to the MEC or the cloud based on real-time workload. The US always achieves the worst performance since the US uniformly assigns task upload to MEC servers and tasks tolerate long delay before the vehicles drives into the assigned server. It is verified by Fig. 6, which shows that the AUT of GCO achieves much higher than other algorithms. Particularly, the PCO achieves the lowest ASD across all the service scenarios. Fig. 8 shows the primal dual variable $||r||^2$ under different traffic scenarios, which demonstrates that the PSO can always achieve convergence in an iterative way. Based on the observation, this set of simulation result shows the scalability of the PSO against varying system workloads.

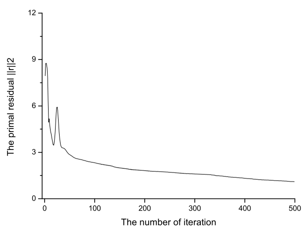
2) Effect of Task Computation Rate: Fig. 9 shows the ASD of five algorithms under different task computation rates. Higher value of task computation rates indicates that



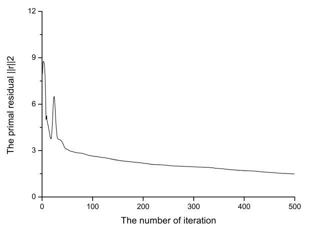
(a) Scenario 1



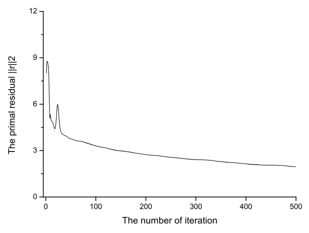
(b) Scenario 2



(c) Scenario 3



(d) Scenario 4



(e) Scenario 5

Fig. 8. The curves of the primal dual variable $||r||^2$ under different traffic scenarios.

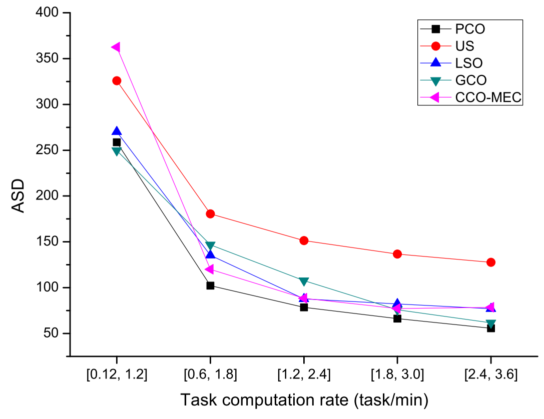


Fig. 9. The ASD of five algorithms under different task computation rates.

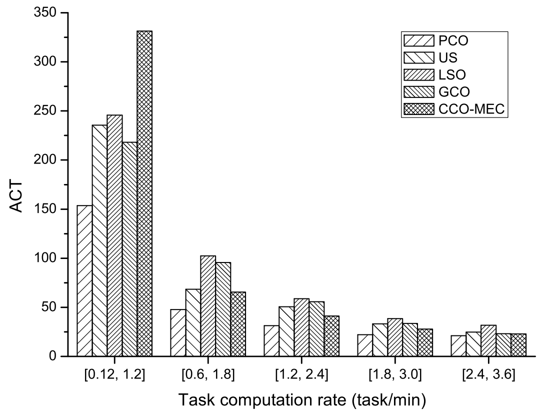


Fig. 11. The ACT of five algorithms under different task computation rates.

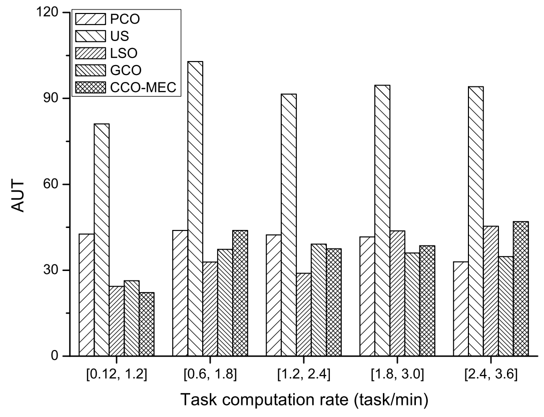


Fig. 10. The AUT of five algorithms under different task computation rates.

individual task can be served with less computation time, which indicates lighter service workload. Accordingly, the ASD of five algorithms decreases dramatically at first with increasing task computation rates and then the trend slows down when the task computation rate keeps increasing. This phenomenon is explained as follows. Figs. 10 and 11 show the AUT and the ACT of five algorithms under different task computation rates, respectively. The ACT and AUT of five algorithms are 200s and 50s on average at [0.12, 1.2] task/min, respectively, which indicates that the ACT dominates the ASD at first. When the computation rate increases, the ACT of five algorithms decreases dramatically and becomes less than 50s on average after the point of [1.2, 2.4] task/min. Since the service rate of task upload remains the same, the AUT maintains at a stable level and dominates the ASD when the task computation rate keeps increasing. Further, the US achieves the worst ASD since it consumes overhigh time for uniform task uploading among different MEC severs. In addition, the CCO-MEC achieves the highest ASD at beginning since the CCO-MEC only considers to offload task to MEC servers, which results in high ACT when the computation capability of MEC server is weak. Particularly, it is observed that the PCO still achieves the lowest ASD among five algorithms across all the scenarios. It is because that the PCO can balance the AUT and ACT by optimally determining workload distribution among MEC/cloud severs. This set of simulation results show the effectiveness of the PCO against varying computation capabilities.

VII. Conclusion

In this article, we present an MEC-assisted architecture for cooperative computation offloading in vehicular networks, where MEC servers are deployed at the roadside and responsible for making task offloading decision from the mobile terminal users to the MEC/cloud servers. Specifically, the mobility features of vehicles are characterized by dwelling time and turning probability between MEC servers. The heterogeneous communication capacities of MEC servers are characterized by different service rates of task upload and migration, denoted by u_i^u and u_j^u, respectively. Further, the heterogeneity of computation capacities is characterized by different processor numbers, varying service rates of task computation and diverse queuing capacities. With such an architecture, we model the task upload and migration models based on the $M/M/1$ queuing model and the task computation model of MEC and cloud severs based on $M/M/C$ and $M/M/\infty$ queuing model. By synthesizing the above models, we formulate the problem of cooperative computation offloading (CCO), which aims at minimizing the expected system service delay by searching the optimal allocation probability. Then, we analyze the convexity of the objective function and propose a probabilistic approach, which consist of oneline and offline phase. In offline phase, the PCO utilizes the ADMM approach to transform the objective function into augmented Lagrangian by adding dual variables and derives the optimal solution in an iterative way by establishing three iterations. In online phase, the PCO uses a probabilistic approach to determine the scheduling decision of each new task based on the optimal allocation probability. Finally, we build the simulation model and implement the proposed algorithm as well as four competitive algorithms. The comprehensive simulation results validate the superiority of the PCO under a wide range of service scenarios.

References

[1] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, “A scalable and quick-response software defined vehicular network assisted by mobile edge computing,” *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 94–100, Jul. 2017.

[2] H. Alshaer and E. Horlait, “An optimized adaptive broadcast scheme for inter-vehicle communication,” in *Proc. IEEE 61st Veh. Technol. Conf.*, vol. 5, May 2005, pp. 2840–2844.

[3] H. Alshaer and J. M. H. Elmirghani, “Road safety based on efficient vehicular broadcast communications,” in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2009, pp. 1155–1160.

[4] N. Mitrovic, M. T. Asif, J. Dauwels, and P. Jaillet, “Low-dimensional models for compressed sensing and prediction of large-scale traffic data,” *IEEE Trans. Intell. Transport. Syst.*, vol. 16, no. 5, pp. 2949–2954, Oct. 2015.

[5] B. Kaur and J. Bhattacharya, “A convolutional feature map-based deep network targeted towards traffic detection and classification,” *Expert Syst. Appl.*, vol. 124, pp. 119–129, Jun. 2019.

[6] T. Sawabe, M. Kanbara, and N. Hagita, “Diminished reality for acceleration—Motion sickness reduction with vection for autonomous driving,” in *Proc. IEEE Int. Symp. Mixed Augmented Reality (ISMAR-Adjunct)*, Sep. 2016, pp. 297–299.

[7] P. Dai, K. Liu, X. Wu, Z. Yu, H. Xing, and V. C. S. Lee, “Cooperative temporal data dissemination in SDN-based heterogeneous vehicular networks,” *IEEE Internet Things J.*, vol. 6, no. 1, pp. 72–83, Feb. 2019.

[8] X. Huang, R. Yu, J. Kang, Y. He, and Y. Zhang, “Exploring mobile edge computing for 5G-enabled software defined vehicular networks,” *IEEE Wireless Commun.*, vol. 24, no. 6, pp. 55–63, Dec. 2017.

[9] P. Dai *et al.*, “Temporal information services in large-scale vehicular networks through evolutionary multi-objective optimization,” *IEEE Trans. Intell. Transport. Syst.*, vol. 20, no. 1, pp. 218–231, Jan. 2019.

[10] Z. Chang, Z. Zhou, T. Ristaniemi, and Z. Niu, “Energy efficient optimization for computation offloading in fog computing system,” in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.

[11] H. Cao and J. Cai, “Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach,” *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.

[12] 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.

[13] H.-R. Cheon and J.-H. Kim, “Social-aware mobile data offloading algorithm through small cell backhaul network: Direct and indirect user influence perspectives,” *Comput. Netw.*, vol. 165, Dec. 2019, Art. no. 106951.

[14] H. Alshaer, T. Ernst, and A. de La Fortelle, “A QoS architecture for provisioning high quality in intelligent transportation services,” in *Proc. IEEE Netw. Oper. Manage. Symp.*, Apr. 2012, pp. 595–598.

[15] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, “Collaborative vehicular edge computing networks: Architecture design and research challenges,” *IEEE Access*, vol. 7, pp. 178942–178952, 2019.

[16] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang, “Toward cloud-based vehicular networks with efficient resource management,” *IEEE Netw.*, vol. 27, no. 5, pp. 48–55, Sep. 2013.

[17] 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.

[18] W. Fan, Y. Liu, B. Tang, F. Wu, and Z. Wang, “Computation offloading based on cooperations of mobile edge computing-enabled base stations,” *IEEE Access*, vol. 6, pp. 22622–22633, 2018.

[19] P. Dai *et al.*, “Multi-armed bandit learning for computation-intensive services in MEC-empowered vehicular networks,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7821–7834, Jul. 2020.

[20] Z. Zhou, J. Feng, Z. Chang, and X. Shen, “Energy-efficient edge computing service provisioning for vehicular networks: A consensus ADMM approach,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5087–5099, May 2019.

[21] J. Sun, Q. Gu, T. Zheng, P. Dong, A. Valera, and Y. Qin, “Joint optimization of computation offloading and task scheduling in vehicular edge computing networks,” *IEEE Access*, vol. 8, pp. 10466–10477, 2020.

[22] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, “Multiobjective optimization for computation offloading in fog computing,” *IEEE Internet Things J.*, vol. 5, no. 1, pp. 283–294, Feb. 2018.

[23] L. T. Tan and R. Q. Hu, “Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning,” *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.

[24] Y. Wang *et al.*, “A game-based computation offloading method in vehicular multi-access edge computing networks,” *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

[25] L. Tang, X. Chen, and S. He, “When social network meets mobile cloud: A social group utility approach for optimizing computation offloading in cloudlet,” *IEEE Access*, vol. 4, pp. 5868–5879, 2016.

[26] F. Mehmeti and T. Spyropoulos, “Performance analysis of mobile data offloading in heterogeneous networks,” *IEEE Trans. Mobile Comput.*, vol. 16, no. 2, pp. 482–497, Feb. 2017.

[27] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[28] S. Boyd, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2010.

[29] J. F. C. Mota, J. M. F. Xavier, P. M. Q. Aguiar, and M. Püschel, “D-ADMM: A communication-efficient distributed algorithm for separable optimization,” *IEEE Trans. Signal Process.*, vol. 61, no. 10, pp. 2718–2723, May 2013.

[30] C. Liang, F. R. Yu, H. Yao, and Z. Han, “Virtual resource allocation in information-centric wireless networks with virtualization,” *IEEE Trans. Veh. Technol.*, vol. 65, no. 12, pp. 9902–9914, Dec. 2016.

[31] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transport. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[32] *Data Source: Didi Chuxing Gaia Initiative*. Accessed: Jan. 6, 2019. [Online]. Available: https://gaia.didichuxing.com

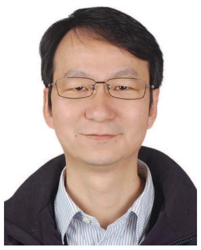
[33] B. Coll-Perales, J. Gozalvez, and M. Gruteser, "Sub-6GHz assisted MAC for millimeter wave vehicular communications," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 125–131, Mar. 2019.

[34] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.

[35] H. Cheny, D. Zhao, Q. Cheny, and R. Chai, "Joint computation offloading and radio resource allocations in wireless cellular networks," in *Proc. 10th Int. Conf. Wireless Commun. Signal Process. (WCSP)*, Oct. 2018, pp. 1–6.

Penglin Dai (Member, IEEE) received the B.S. degree in mathematics and applied mathematics and the Ph.D. degree in computer science from Chongqing University, Chongqing, China, in 2012 and 2017, respectively. He is currently an Assistant Professor with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China. His research interests include intelligent transportation systems and vehicular cyber-physical systems.

Kaiwen Hu received the B.E. degree in vehicle engineering from Southwest Jiaotong University, Chengdu, China, in 2019, where he is currently pursuing the master's degree with the School of Information Science and Technology. His research interests include vehicular networks and applications of convex optimization in vehicular networks.



Xiao Wu (Member, IEEE) received the B.Eng. and M.S. degrees in computer science from Yunnan University, Yunnan, China, in 1999 and 2002, respectively, and the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2008.

He was with the Institute of Software, Chinese Academy of Sciences, Beijing, China, from 2001 to 2002. He was a Research Assistant and a Senior Research Associate with the City University of Hong Kong from 2003 to 2004 and from 2007 to 2009. He was a Visiting Scholar with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, and the School of Information and Computer Science, University of California at Irvine, Irvine, CA, USA, from 2006 to 2007 and from 2015 to 2016. He is currently a Professor and the Assistant Dean of the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China. He has authored/coauthored more than 70 research articles in well-respected journals, such as the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE TRANSACTIONS ON MEDICAL IMAGING and prestigious proceedings like Conference on Computer Vision and Pattern Recognition and ACM Multimedia. His research interests include artificial intelligence, multimedia information retrieval, image/video computing, and computer vision. He received the Second Prize of Natural Science Award of the Ministry of Education, China, in 2016, and the Second Prize of Science and Technology Progress Award of Henan Province, China, in 2017.

Huanlai Xing (Member, IEEE) received the B.Eng. degree in communications engineering from Southwest Jiaotong University, Chengdu, China, in 2006, the M.Sc. degree in electromagnetic fields and wavelength technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K., in 2013. He is currently an Associate Professor with the School of Information Science and Technology, Southwest Jiaotong University. His research interests include evolutionary computation, cloud computing, multiobjective optimization, network function virtualization, and software defined networks. He has authored or coauthored over 30 peer-reviewed journal articles and conference papers. He is a member of ACM.

Fei Teng (Member, IEEE) was born in 1984. She received the Ph.D. degree from the École Centrale Paris in 2011. She is currently an Associate Professor and the master's Supervisor with Southwest Jiaotong University. Her research interests include parallel computing, cloud computing, and edge computing.

Zhaofei Yu (Member, IEEE) received the B.S. degree from the Hongshen Honors School, College of Optoelectronic Engineering, Chongqing University, Chongqing, China, in 2012, and the Ph.D. degree from the Automation Department, Tsinghua University, Beijing, China, in 2017. He is currently a Post-Doctoral Fellow with the National Engineering Laboratory for Video Technology, School of Electronics Engineering and Computer Science, Peking University, Beijing. His current research interests include artificial intelligence, brain-inspired computing, and computational neuroscience.