Online MEC Offloading for V2V Networks

Fangming Liu*, *Senior Member, IEEE,* Jian Chen, Qixia Zhang, and Bo Li, *Fellow, IEEE*

Abstract—As an enabling technology for vehicle-to-vehicle (V2V) networks, multi-access edge computing (MEC) provides a feasible platform for sharing power and resources, and offloading some of the computation-intensive tasks between vehicles. This, however, is challenging with the unpredictable variations in road traffic conditions and vehicle mobility in MEC-enabled V2V networks. Consequently, such computation task offloading can be easily disrupted, which may require frequent switching of task offloading between vehicles and degrade the Quality of Service (QoS). In this paper, we focus on the computation offloading problem under unstable connections in MEC-enabled V2V networks. We first model this as a distributed online service optimization problem, which is proved to be NP-hard. In order to minimize the out-of-service time (i.e., the service mismatching, switching and compromise time), we propose a distributed Online Instability-aware Computation Offloading (OICO) heuristic algorithm to improve the service efficiency and quality. Specifically, in order to minimize the service mismatching rate, we design an efficient Service Path Matching (SPM) algorithm for matching pairs of customer vehicles (which require offload computing services) and server vehicles (which provide edge computing services) that share the longest matching path. We evaluate OICO through real-world traces, i.e., GAIA open dataset from DiDi. Extensive simulation results demonstrate that OICO can increase the service matching rate by 25% and reduce the power consumption by about 54% per customer vehicle compared with the existing schemes.

Index Terms—V2V Communication, Multi-access Edge Computing, Computation Offloading, Online Service Optimization.

1 Introduction

With the evolution of 5G vehicular networks and base station coordinated device-to-device (D2D) communications, short-range vehicle-to-vehicle (V2V) communication has become one of the key technologies in enhancing the transportation and information exchange [1], [2]. Due to the safety imperative, V2V communication usually has high-level Quality of Service (QoS) requirements in terms of ultra low latency and high reliability [3]. To meet these stringent QoS requirements, *multi-access edge computing* (MEC) emerges as a feasible solution for sharing power and resources [4], [5], and even offloading some computation-intensive services between vehicles, such as augmented reality (AR) and vehicular games.

In fact, modes of MEC vehicular service offloading can be categorized into two types: (1) offloading to edge computing facilities (e.g., edge server, edge cloud) via 4G/5G/V2I (Vehicle to Infrastructure), and (2) offloading to moving vehicles with idle computing resources via V2V communications [6], [7]. Compared with offloading to edge infrastructures via expensive V2I or 4G/5G cellular networks, offloading to vehicles via V2V communications can be much cheaper by making use of the idle computing resources of vehicles. This provides a feasible and cost-effective alternative for offloading services. With the growing number of smart vehicles in recent years, V2V communication can now support in-car entertainment (e.g., vehicular games in Tesla [8] or BYD electric cars) with reasonable performance. In this scenario, MEC-enabled vehicles with idle resources can act as "servers" to provide computing services so as to deal with the challenges like battery anxiety from customers [9]. This paradigm enables low-power vehicles to offload part of their value-added computing services to nearby vehicles via V2V communications [6]. In V2V networks, *customer vehicles* are the vehicles requiring to offload their computation services, while *server vehicles* are the ones providing computation resources and services. By offloading some of the computation-intensive tasks to nearby server vehicles, the power consumption of customer vehicles can be reduced and their endurance mileage can be extended. In contrast, server vehicles can potentially gain monetary profits by offering such services at the expense of their own computation resources and electric/chemical energy consumption.

Nevertheless, differing from offloading computation to MEC servers with fixed locations, offloading computation to vehicles is more challenging due to the unpredictable variations of traffic conditions and real-time mobility of vehicles [6], [9], [10]. In real-world cases, both customer vehicles and server vehicles may change their directions, speeds, and routes at any time. Under this circumstance, among a large number of candidate server vehicles, intuitively a customer vehicle prefers to select a server vehicle with the same start point, similar driving speed, and longer matching path which refers to the common driving path of two vehicles.

For example, as shown in Fig. 1, the white vehicle is a customer vehicle, while the No.1, No.2, and No.3 black vehicles are the candidate server vehicles. Ideally, No.1 server vehicle first matches the customer vehicle at point A, and drives along path A-B at the same speed to provide

- *F. Liu, Jian Chen, and Q. Zhang are with the National Engineering Research Center for Big Data Technology and System, the Services Computing Technology and System Lab, Cluster and Grid Computing Lab in the School of Computer Science and Technology, Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan 430074, China. E-mail: {fmliu, chenjian98, zhangqixia427}@hust.edu.cn.*
- *Bo Li is with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology.*

This work was supported in part by the NSFC under Grant 61761136014 and 61520106005, in part by National Key Research & Development (R&D) Plan under grant 2017YFB1001703. The research was supported in part by RGC RIF grant R6021-20, and RGC GRF grants under the contracts 16207818 and 16209120. (Corresponding author: Fangming Liu)

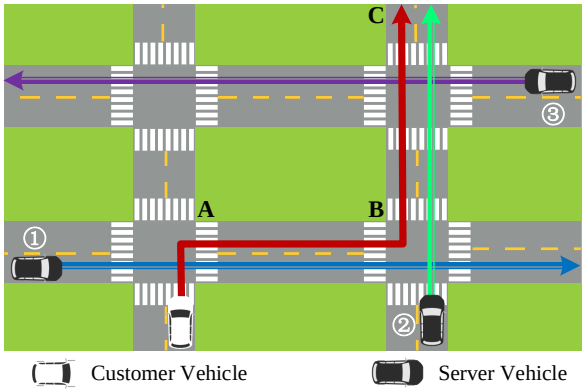

Fig. 1: An example of computation offloading and service matching in MEC-enabled V2V Networks.

offloading service. Then No.2 server vehicle serves the customer along path B-C. We define the *service matching time* by the period of time when a pair of customer and server vehicles matches and offloads tasks.

When two vehicles are out of the effective V2V communication distance [11], customer vehicle needs to find new server vehicles for offloading its tasks. Switching server vehicles will lead to further degraded QoS. Thus, how to jointly minimize the service mismatching time and service switching time is a significant challenge for customer vehicles.

As for the server vehicles, to ensure the connectivity and stability in computation offloading process, sometimes they have to make a compromise to decelerate or accelerate and maintain the same speed as customer vehicles, which is defined as *service compromise*. For example, if a server vehicle driving at 60 km/h matches a customer vehicle driving at 40 km/h, then it needs to decelerate to 40 km/h for providing continuous services. Generally, the profit of service is positively correlated with the cost of service. For example, in other aspects of life, such as online taxi service, service profit is positively correlated with some indicators, such as waiting time for passengers, distance, travel time and so on. So the profits of server vehicles are positively correlated with electric/chemical power consumption for providing computation offloading services under normal circumstances. They intend to maximize the service matching time and minimize the service compromise time. To jointly improve the profits of both customer and server vehicles, in this paper, we study the **online computation offloading problem under complex vehicle mobility for MEC-enabled V2V networks** with an objective of **minimizing the out-of-service time**, i.e., the service mismatching, service switching and service compromise time.

Currently, some efforts have been paid to tackle the computation offloading problem in MEC systems and/or V2V networks [5], [10], [12]–[16]. However, some existing solutions provide centralized algorithms without considering the mobility of vehicles and unstable services [16]. Some other works use lazy or passive task migration strategy [9], which will greatly increase the end-to-end latency and may not satisfy the strict QoS requirements of vehicular networks. Differing from them, in this paper, we take into account the service instability, service switching costs, power, resource, and latency constraints, and then we model the online computation offloading problem under complex vehicle mobility for V2V networks as a distributed online service optimization problem, which is proved NP-hard. To minimize the out-of-service time, we propose a distributed Online Instability-aware Computation Offloading (OICO) heuristic algorithm for MEC-enabled V2V networks, which can effectively improve the profits of both customer vehicles and server vehicles. Specifically, OICO can be distributed to server vehicles and does not need to know global information, thus ensuring less information exchange and lower execution delay than centralized algorithms. In OICO, we also design an efficient Service Path Matching (SPM) heuristic algorithm to match the most suitable server vehicle for each customer vehicle, which shares the longest matching path. We give theoretical analyses of algorithms in terms of both optimality and computation complexity. Then we conduct extensive evaluations based on real-world traces, i.e., 1.92 million GPS data from 1,000 vehicles in GAIA open dataset from DiDi [17]. Evaluation results demonstrate that OICO can efficiently minimize the out-of-service time and improve both customer and server vehicles' profits.

The main contributions are as follows:

We model the online computation offloading problem under complex vehicle mobility for MEC-enabled V2V networks as a distributed online service optimization problem, which is proved NP-hard. The service instability, power, resource, and latency constraints are all considered in our model.

In order to minimize out-of-service time (i.e., the service mismatching, service switching and service compromise time), we propose a distributed Online Instability-aware Computation Offloading (OICO) heuristic algorithm together with an efficient Service Path Matching (SPM) heuristic algorithm, which can effectively improve the profits of both customer and server vehicles.

Extensive evaluations based on real-world traces demonstrate that our proposed approach can increase the service matching rate by 25% and reduce the power consumption by about 54% per customer vehicle, as compared with other existing schemes.

Discuss the business model, and the privacy and security concerns under this pattern.

2 Related Work

There are many studies on improving the service quality and energy efficiency in edge computing systems, such as edge service allocation/migration [18]–[22] and energy-efficient mobile cloud computing [23]–[26]. In this section, we only discuss and summarize the closely-related literature: we first investigate the state-of-the-art on computation offloading and task scheduling for MEC systems. Then we focus on the latest related work on computation offloading for vehicular networks, including the V2V networks.

2.1 Computation Offloading for MEC Systems

In recent years there has been a number of studies on the computation offloading and task scheduling in MEC

systems. Specifically, Huang *et al.* [27] propose a stochastic control algorithm to decide which software component should be offloaded so as to minimize the long-term average energy consumption. Liu *et al.* [28] propose a stochastic computation task scheduling policy, incorporating different timescales in both task execution and channel fading process. Mao *et al.* [29] propose a Lyapunov optimization-based online computation offloading algorithm with energy harvesting devices under a strict processing latency requirement. And the authors in [30] also adopt the Lyapunov optimization to design their energy-efficient cloud offloading scheduling algorithm. Finally, Kwak *et al.* [31] solve the dynamic resource and task allocation problem in mobile cloud systems with heterogeneous types of mobile applications. These existing works mainly focus on the computation offloading and task scheduling in single-user [32], [33] or simple linear systems [34], while only a few works focus on designing methodologies for multi-user MEC systems [12], [35], [36].

To meet the demand of multi-user MEC systems, Apostolopoulos *et al.* [12] investigate the trade-off between the power consumption and latency reduction and propose a Lyapunov optimization-based computation offloading algorithm. Similarly, Zhang *et al.* [13] present an energy-aware offloading scheme to jointly optimize the communication and computation resource allocation under the limited energy and sensitive latency. Gao *et al.* [14], [37] also study the problem of joint optimization on the access networks selection and service placement for multi-user MEC systems. Compared to these multi-user MEC systems, our proposed approach supports large-scale real complex road conditions and high mobility of users. And we adopt the distributed method and do not need a central decision server for scheduling.

2.2 Computation Offloading for Vehicular Networks

With the evolution of MEC and 5G technologies, some efforts have been made to address the challenges (e.g., providing intelligent vehicle control and reliable communications) in vehicular networks. First, Zhang *et al.* [38] list five perspective works regarding MEC for vehicular networks. In particular, Huang *et al.* [39] explore the 5G-enabled software-defined vehicular networks (5G-SDVNs) and leverage MEC to strengthen network control of 5G-SDVN. Liu *et al.* [40] introduce an SDN-enabled MEC network architecture that integrates a heterogeneous vehicular network and offers reliable communication services. Sasaki *et al.* [41] propose an infrastructure-based vehicle control system that shares internal states between edge servers and cloud servers to reduce latency and balance the computational load. Besides, Yuan *et al.* [42] propose a two-level edge computing architecture for automated driving services in order to make full use of wireless connections for coordinated content delivery.

In addition, a few work has addressed the computation offloading, task scheduling, and migration in MEC-enabled vehicular networks. For instance, Zhang *et al.* [15] propose a task-file transmission strategy with a predictive V2V relay and an optimal predictive combination-mode offloading scheme for MEC-based vehicular networks. However, some of their assumptions are too strong for real-world cases, e.g., vehicles arrive as a Poisson distribution and move at a constant speed, and vehicular tasks are only offloaded once without dynamic migration, which ignores the complex vehicle mobility. Huang *et al.* [10] propose a joint SDN and MEC enabled architecture for V2V data offloading in cellular networks. However, they do not specify the many-to-many relationship between base stations and MEC servers. Ning *et al.* [16] investigate new methods to reduce the task offloading latency in MEC-enabled vehicular networks, while they do not take account of the real-time mobility of vehicles and fail to provide an efficient online task offloading strategy. Han *et al.* [9] use idle vehicle-mounted computing resources to assist the calculation, however, it is still essentially a single-user MEC system. Feng *et al.* [5] offload the calculations to surrounding vehicles with idle computing power. However, in the experiment, the vehicles are limited to a small area, which cannot fully reflect the vehicle mobility and service instability.

For V2V communication, Naqvi *et al.* [43] state that device-to-device (D2D) communication coordinated by base stations is suitable for short-distance wireless transmission of V2V communication. To reduce co-channel interference, Yang *et al.* [44] enable D2D connections between vehicles by reusing uplink spectrum resources while allocating base station orthogonal spectrum resources to D2D users and cellular users. Differing from these existing works, we propose a distributed online instability-aware offloading approach for MEC-enabled V2V networks with special consideration for complex vehicle mobility, service switching costs, power, resources and latency constraints, which can jointly improve the profits of both customer vehicles and server vehicles.

3 Model and Problem Formulation

In this section, we first introduce the MEC-enabled V2V networks model. Then we define the decision variables, including indicators that reflect the instability and efficiency in computation offloading process. Finally, we present the mathematical formulation of the online computation offloading problem under complex vehicle mobility for MEC-enabled V2V networks with the objectives and constraints. Key notations are listed in Table 1.

3.1 MEC-enabled V2V Networks Model

First of all, we use a set $U = \{u_1, u_2, \ldots, u_{|U|}\}$ to represent the set of customer vehicles, where each customer vehicle has some computation tasks to be offloaded. And we define the set of server vehicles by $V = \{v_1, v_2, \ldots, v_{|V|}\}$, where each server vehicle can offer computation offloading services. In each time slot of our model, the number of active U and V nodes and the relationship between supply and demand change dynamically from moment to moment, rather than simply assuming a fixed number. By computation offloading, customers in U can pay some money to address their battery/energy issue, while owners of server vehicles in set V can earn some money by providing computation services at the expense of their own energy consumption. To ensure the connectivity and communication efficiency of customer vehicles and server vehicles, we mainly consider

single-hop V2V communications in our model. Since most of the throughput related activities can be reduced to this basic single-hop problem [45], such as multi-hop network traffic [46] that is essentially dependent on routing and single-hop wireless link transmission planning and data aggregation scheduling [47].

When performing a computation offloading task between a server vehicle and a customer vehicle, taking game's offloading service as an example, each rendered frame of the game is transmitted back to customer vehicle in real time. To deal with real-time cases, we divide time into a set of time slots, defined as $T = \{t_1, t_2, \ldots, t_{|T|}\}$, where the length of each time slot is . Note that the offloaded tasks are executed during the period of two vehicles driving together. We assume that at the time slot $\forall t \in T$, the set of computation offloading tasks in the set U is S^t, where $S^t = \{s_1^t, s_2^t, \ldots, s_{|U|}^t\}$. For $\forall u_i \in U$, the computation offloading task related to vehicle u_i is s_i^t. There are different types of tasks in the set S^t, such as AR, vehicular games, map services, etc. The customer vehicle does not need service all the time, so in some time slots $t \in T$, the type of s_i^t may be idle. We assume that the set of types of tasks is D, where $D = \{d_1, d_2, \ldots, d_{|D|}\}$. Each type of task $d \in D$ has a computation offloading ratio $_d$, which represents the proportion of the task that can be offloaded to the server vehicles. Clearly, $1 - _d$ means the proportion of a type $d \in D$ task that can only be executed on customer vehicles and cannot be offloaded to other server vehicles. We use e_d^c to represent the power for computation offloading; while e_d^m represents the power for V2V communication when a task is offloaded to another vehicle.

3.2 Decision Variables

As vehicles move in real time, the computation offloading process will be affected by the complex vehicle mobility, i.e., the real-time V2V communication distance. This means that if two vehicles are within effective V2V communication distance, customer vehicle can offload its tasks to server vehicle; otherwise, the tasks cannot be offloaded. Based on IEEE 802.11p [48] and 5G D2D technology and communication standards, we use M to represent the effective V2V communication distance [11], [49]. Note that V2V communication distance can be modified under other communication standards. At each time slot $t \in T$, we first use a binary indicator $x_{u,v}^t$ to indicate whether server vehicle $u \in U$ is in the effective V2V communication distance of customer vehicle $v \in V$. Naturally, we use $x_{u,v}^t = 1$ to represent the case that when the distance between $v \in V$ and $u \in U$ is less than M at time slot $t \in T$, the tasks can be offloaded; otherwise, $x_{u,v}^t = 0$ means the tasks cannot be offloaded by any other server vehicle. Thus, we have a constraint for $x_{u,v}^t$ as follows:

$$x_{u,v}^t \in \{0,1\}, \forall u \in U, \forall v \in V, \forall t \in T. \quad (1)$$

Due to the real-time mobility of customer and server vehicles, the indicator $x_{u,v}^t$ can be changed as the time slots move on, reflecting the complex vehicle mobility in computation offloading process, which is a significant characteristic of V2V communications. As defined, $|U|$ customer vehicles have $|D|$ types of tasks. To indicate the specific type of task and the power consumption of each vehicle, we introduce the second binary indicator $y_{u,d}^t$ to indicate whether the vehicle $u \in U$ needs the type $d \in D$ task at time slot $t \in T$. In fact, each customer vehicle may require different types of services at different time slots. Thus, $y_{u,d}^t = 1$ means that vehicle $u \in U$ needs the type $d \in D$ task at this time slot $t \in T$, and $y_{u,d}^t = 0$ means $u \in U$ does not need type d task. For simplification, we assume that a customer vehicle only has one type of service at a time slot, which can be represented as follows:

$$\sum_{d \in D} y_{u,d}^t = 1, \forall u \in U, \forall t \in T. \quad (2)$$

$$y_{u,d}^t \in \{0,1\}, \forall u \in U, \forall d \in D, \forall t \in T. \quad (3)$$

At each time slot $t \in T$, we use a binary indicator $z_{u,v}^t$ to indicate whether customer vehicle $u \in U$ is served by server vehicle $v \in V$ at time slot $t \in T$. Thus, $z_{u,v}^t = 1$ means that

TABLE 1: Key Notations

Symbol	Description
U	The set of customer vehicles
V	The set of server vehicles
D	The set of tasks' types
T	The set of time slot
	The length of each time slot
S^t	The set of computation offloading tasks at time slot $t \in T$
$_d$	The proportion of a type $d \in D$ task that can be offloaded to server vehicles
e_d^c	The electric power for computation offloading of a type $d \in D$ task
e_d^m	The electric power for V2V communication of a type $d \in D$ task
f_v^p	The computation capacity of $v \in V$
f_d^c	The computation demand of $d \in D$
p_u	The power capacity of customer vehicle $u \in U$ for processing the local tasks
q_v	The power capacity of server vehicle $v \in V$ for processing the offloaded tasks
M	The effective V2V communication distance
$x_{u,v}^t$	1 if customer vehicle $u \in U$ is within the service communication distance of server vehicle $v \in V$ at time slot $t \in T$, 0 otherwise
$y_{u,d}^t$	1 if customer vehicle $u \in U$ needs a type $d \in D$ task at time slot $t \in T$, 0 otherwise
$z_{u,v}^t$	1 if customer vehicle $u \in U$ is served by server vehicle $v \in V$ at time slot $t \in T$, 0 otherwise
t_u	The expected total service time of $u \in U$
t_u^{mis}	The service mismatching time of $u \in U$
t_u^{mt}	The service matching time of $u \in U$
t_d^s	The service switching time of a type $d \in D$ task
t_u^{swc}	The total service switching time of $u \in U$
t_v^c	The service compromise time of $v \in V$

customer vehicle u is served by vehicle v at this time slot $t \in T$; otherwise, $z^t_{u;v} = 0$. In addition, as we stated before, not all parts of the task can be offloaded to server vehicles (i.e., only the $\_d$ parts of task can be offloaded). We use $z^t_{u;0} = 1$ to indicate that vehicle $u \in U$'s task is running locally instead of offloading to any other vehicle at time slot $t \in T$. Thus we have constraints for $z^t_{u;v}$ as follows:

$$\sum_{v \in \{0\} \cup V} z^t_{u;v} = 1; \forall u \in U; \forall t \in T: \tag{4}$$

$$z^t_{u;v} \in \{0; 1\}; \forall u \in U; \forall v \in \{0\} \cup V; \forall t \in T: \tag{5}$$

$$\sum_{v \in \{0\} \cup V} z^t_{u;v} \quad x^t_{u;v} \quad 1; \forall u \in U; \forall t \in T: \tag{6}$$

Constraint (6) means one task of customer vehicle can only be served by one server vehicle or locally at any time slot t.

3.3 Objective and Constraints

Now we formally propose the mathematical formulation of the **online computation offloading problem under complex vehicle mobility for MEC-enabled V2V networks**. We begin with the constraints.

First of all, each server vehicle $v \in V$ has a computation capacity f^p_v (i.e., CPU frequency [32]) for dealing with offloaded tasks, while each type $d \in D$ task has a computation demand f^c_d. Thus, at time slot $\forall t \in T$, the total computation demand of all tasks offloaded to server vehicle v should not exceed its computation capacity f^p_v, which can be represented by:

$$\sum_{u \in U} z^t_{u;v} \sum_{d \in D} f^c_d \quad y^t_{u;d} \quad f^p_v; \forall v \in V; \forall t \in T: \tag{7}$$

In terms of energy costs, we also consider the transmission and computing energy costs of task offloading. In related work, it makes sense to consider energy costs, Mao *et al.* [50] also consider time costs and energy costs jointly. Since the power used for providing computation offloading services should not affect the driving of the vehicle. So we put some constraints on the power consumption of the vehicle for MEC service. We use q_v to represent the available power capacity of server vehicle $v \in V$ for dealing with offloaded tasks, and we have:

$$\sum_{t \in T} \sum_{u \in U} z^t_{u;v} \quad \Big(\sum_{d \in D} y^t_{u;d} \quad (e^c_d \quad \_d + e^m_d) \Big) \quad q_v: \tag{8}$$

Besides, the power of each customer vehicle $u \in U$ is limited, we use p_u to represent the available power capacity for processing the local tasks (i.e., $(1 \quad \_d)$ parts of task cannot be offloaded). Thus, all the power consumption of local tasks of customer vehicle u should not exceed its power capacity, which can be expressed as follows:

$$\sum_{t \in T} \sum_{v \in V} z^t_{u;v} \quad \Big(\sum_{d \in D} y^t_{u;d} \quad (e^c_d \quad (1 \quad \_d) \quad e^m_d) \Big) \quad p_u: \tag{9}$$

Now we introduce the time slot-based customer-related parameters. First, each customer vehicle $u \in U$ has an *expected total service time* of t_u, which represents that customer vehicle u expects to get this certain period of time for offloading its computation tasks. However, due to the complex vehicle mobility, the customer vehicle cannot always "match" a server vehicle for offloading its computation tasks. Thus, we use t^{mt}_u to represent the *service matching time* of customer vehicle $u \in U$. Within t^{mt}_u, customer vehicle u can offload its tasks to other server vehicles, which can be expressed as follows:

$$t^{mt}_u = \sum_{t \in T} \sum_{v \in V} z^t_{u;v} \quad : \tag{10}$$

We also define a *service mismatching time* of customer vehicle $u \in U$, represented by t^{mis}_u. Thus, we have:

$$t^{mis}_u = t_u \quad t^{mt}_u = t_u \sum_{t \in T} \sum_{v \in V} z^t_{u;v} \quad : \tag{11}$$

In fact, due to the complex vehicle mobility in V2V networks, matching a server vehicle does not mean that a customer vehicle can totally offload its tasks in the service matching time. When a server vehicle drives out of the V2V communication distance, the computation offloading process will be interrupted and the customer vehicle needs to find a new available server vehicle for its unprocessed tasks. Due to the real-time mobility of vehicles, sometimes a customer vehicle needs to switch its server vehicles frequently. The sum of these periods of time is called by the *service switching time*. When switching the service to another vehicle, the sandbox or file needs to be loaded into memory and initialized. We assume that each type $d \in D$ task has its service switching time t^s_d. For each customer vehicle $u \in U$, we define the cumulative time of total switching time by t^{swc}_u. Thus, the total switching time t^{swc}_u of customer vehicle u can be expressed as follows:

$$t^{swc}_u = \sum_{t=2}^{|T|} \Big(\sum_{v=0}^{|V|} \frac{|z^t_{u;v} \quad z^{t \quad 1}_{u;v}|}{2} \quad \Big(\sum_{d \in D} t^s_d \quad y^t_{u;d} \Big) \Big): \tag{12}$$

Note that since a service switching involves one customer vehicle and two server vehicles, the sum of $|z^t_{u;v} \quad z^{t \quad 1}_{u;v}|$ in Eq. (12) needs to be divided by 2.

We also define the *total in-service time* as $(t^{mt}_u \quad t^{swc}_u)$, which represents the total time of customer vehicle $u \in U$ getting served by a server vehicle for computation offloading.

In addition, to further measure the service switching cost, we use a service switching indicator n_u to represent the number of service switching times of a customer vehicle $u \in U$, which can be expressed as follows:

$$n_u = \sum_{t=2}^{|T|} \sum_{v=0}^{|V|} \frac{|y^t_{u;v} \quad y^{t \quad 1}_{u;v}|}{2}: \tag{13}$$

For the drivers of the server vehicles, they intend to provide the computation capacity and power of their own vehicles in exchange for profits. To ensure the connectivity and stability in computation offloading process, sometimes the server vehicles need to decelerate and maintain the same speed as the customer vehicle. For example, as plotted in Fig. 2, a customer vehicle $u_1 \in U$ driving at 30 km/h matches a server vehicle $v_1 \in V$ driving at 60 km/h at time slot $t_1 \in T$. They are both driving from point A to point B at a distance of 10 km on the same route. However, since

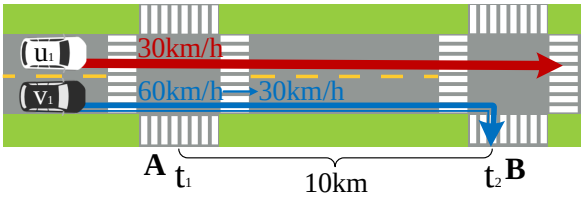

Fig. 2: An example for the service compromise of server vehicle.

vehicle v_1 drives faster than u_1, v_1 is supposed to arrive at B in 10 minutes, while u_1 is supposed to arrive in 20 minutes. In this case, since v_1 is the server vehicle providing services, it has to compromise with the customer vehicle u_1. v_1 will decelerate to 30 km/h and will also arrive at B in 20 minutes at time slot t_2. Thus, the *service compromise time* $t_v^c = 20 \quad 10 = 10$ minutes, which represents the speed compromise made by the server vehicle to maintain the stability of the service. To measure t_v^c, we use $t_{u;v;t}^c$ to represent the service compromise time per time slot, when customer vehicle $u \in U$ matches a server vehicle $v \in V$. In particular, $t_{u;v;t}^c$ is calculated by the service matching distance at time slot t divided by the speed difference between vehicles u and v. More details can be found in Sec. IV.

Thus, for $8v \in V$, the service compromise time t_v^w can be expressed as follows:

$$t_v^c = \sum_{t \in T} \sum_{u \in U} z_{u;v}^t \quad t_{u;v;t}^c; \tag{14}$$

From a customer vehicle $u \in U$ point of view, the objective is to maximize the service matching time while minimizing the number of service switching times. From a server vehicle $v \in V$ point of view, its profits are positively correlated with the power consumption for providing computation offloading services. Thus, a server vehicle also aims to maximize the service matching time while minimizing the service compromise time. In short, in order to jointly improve the profits of both customer vehicles and server vehicles, our objective is to **minimize the out-of-service time**, including the service mismatching, service switching and service compromise time. Note that as stated in Eq. (11), minimizing the service mismatching time t_u^{mis} is equivalent to maximizing the service matching time t_u^{mt}. We use C as the comprehensive performance metric, i.e., the total *out-of-service time* of all vehicles in V2V networks. Thus, the **computation offloading problem under complex vehicle mobility for MEC-enabled V2V networks** can be formulated as follows:

$$\begin{aligned} &\min C \\ &= \min \sum_{t \in T} \sum_{u \in U} \sum_{v \in V} \left(\frac{z_{u;v}^t \quad z_{u;v}^{t\ 1;j}}{2} \sum_{d \in D} t_d^s \, y_{u;d}^t \quad z_{u;v}^t \quad + z_{u;v}^t \, t_{u;v;t}^c \right) \\ &s.t.: \ (1);\ (2);\ (3);\ (4);\ (5);\ (6);\ (7);\ (8);\ (9); \end{aligned} \tag{15}$$

4 Algorithm Design and Analysis

In this section, we first analyze the problem complexity of formulation Eq. (15). Then, we propose a distributed Online Instability-aware Computation Offloading (OICO) heuristic algorithm for MEC-enabled V2V networks, together with the Service Path Matching (SPM) algorithm. Afterward, we provide a theoretical analysis of our proposed algorithms in terms of optimality and computation complexity.

4.1 Problem Complexity

To analyze the problem complexity, we first focus on solving the sub-problem—minimizing the total service compromise time. At time slot $t \in T$, this sub-problem can be represented by:

$$\min \sum_{v \in V} \sum_{u \in U} z_{u;v}^t \quad t_{u;v;t}^c \tag{16}$$
$$s.t.: \ (1);\ (2);\ (3);\ (4);\ (5);\ (6);\ (7);\ (8);\ (9);$$

Theorem 1. *The sub-problem defined in Eq.* (16) *is NP-hard.*

Proof. We construct a polynomial-time reduction to Eq. (16) from the resource constrained generalized assignment problem (RGAP), a classic NP-hard combinatorial optimization problem [51]:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{i;j} \quad x_{i;j} \tag{17}$$
$$s.t.: \sum_{i=1}^{n} x_{i;j} = 1; 8j = 1; :::; n; \tag{18}$$
$$\sum_{j=1}^{m} a_{i;j} \quad x_{i;j} \quad b_i; 8i = 1; :::; m; \tag{19}$$
$$x_{i;j} \in \{0; 1g; 8i = 1; :::; m; j = 1; :::; n: \tag{20}$$

We relax the constraints of equation (16) to (2), (3), (4), (5), (7). Then we can reduce an RGAP problem to (16). In Eq. (17), $c_{i;j}$ can be mapped to $t_{u;v;t}^c$. Constraint (18) means that only one machine can be assigned to perform each task. Constraint (19) indicates that the number of resources consumed by each machine during the execution of tasks cannot exceed the fixed number of resources provided, where $a_{i;j}$ represents the amount of resources consumed by the machine during execution. In constraint (19), $a_{i;j}$ can be mapped to $\sum_{d \in D} f_d^m \, y_{u;d}^t$ and b_i can be mapped to f_v^p. Clearly, this reducing can be done in polynomial time. Therefore, if there exists an instance solution for the RGAP problem, it also solves the sub-problem defined in Eq. (16). Thus, we can conclude that the sub-problem defined in Eq. (16) is NP-hard. □

Theorem 2. *The online computation offloading problem under complex vehicle mobility defined in Eq.* (15) *is NP-hard.*

Proof. The proof process is similar to Theorem 1. As proved, the sub-problem defined in Eq. (16) is proved NP-hard, which is part of the Eq. (15) problem. Since Eq. (15) consists of three time costs, we first conduct a relaxed formulation of the problem (15) by setting the service mismatching time $\sum_{u \in U} t_u^{mis}$ and the total switching time $\sum_{u \in U} t_u^{swc}$ to 0.

We can reduce Eq. (15) to Eq. (16). Clearly, this reduction can be finished in polynomial time. Based on Theorem 1, the reducing from RGAP (17) formulation to Eq. (16) can also be finished in polynomial time. Thus, we can conclude that the problem defined in Eq. (15) is NP-hard. □

4.2 Instability-aware Computation Offloading Algorithm

To deal with the NP-hard problem of Eq. (15), we propose an Online Instability-aware Computation Offloading (OICO) heuristic algorithm, considering the service instability, service switching costs, power, resource, and latency constraints. Specifically, OICO can efficiently offload real-time services from each customer vehicle to the most suitable server vehicles, which are selected by a Service Path Matching (SPM) algorithm for the longest matching path. We first explain how OICO works. Since OICO is a distributed algorithm that does not require a unified central controller or server, the algorithm distributed to the server vehicle is listed in Algorithm 1 and the customer vehicle version is listed in Algorithm 2. Some common variable formats have emerged in several algorithms, such as the Check Point which contains latitude and longitude information as well as time information.

For the server vehicle version of OICO (Algorithm 1), firstly we explain individual variables of the input and output parts. Three-dimensional array $CP1_j$ contains check points, which include encrypted longitude, latitude and time information. The navigation route is equivalent to $CP1_j$. The method of encryption is determined by using the conventions of both parties. Each server vehicle broadcasts to the surrounding area to provide service information (the encrypted navigation route, the vehicle's various resource capacity limits, line 4), while vehicles that are already providing service do not need to broadcast (line 2). We choose to let the service vehicle broadcast its encrypted track and other information instead of the user vehicle to better protect the user vehicle's privacy. If multiple customer vehicles choose the same optimal candidate vehicle, then at this time, the candidate vehicle will match and serve a customer vehicle with the longest service time (lines 10-12). At a certain time, there may be no vehicle around to request service from the server vehicle, then the server vehicle will try for $n1$ times at most. $n1$ is a constant to indicate the maximum number of loop attempts (line 5). The return state 0, 0 means no service is available in the current time period for v_j. Matching duration td is used to indicate the number of time slots that two matched vehicles are driving together.

For the customer vehicle version of OICO (Algorithm 2), each unmatched customer vehicle $u \ 2 \ U$ uses V2V communication to search for the serviceable vehicles nearby (line 4). Then, Algorithm 2 compares the power consumption of each vehicle, and deletes the candidate vehicles that do not meet the power demand and computation capacity. The optimal server vehicle candidate will be selected from the candidate set. Each customer vehicle accepts a set of server vehicles that provide service within V2V distance. Then, the optimal set of top $n1$ candidate server vehicles is obtained by calling SPM (line 8). Up to now, if the idle server vehicles set $S2$ becomes an empty set, it is unfortunate that the vehicle can only run the service locally during the time slot, consuming the power of its own (line 10). The user vehicle defines the end point of the co-drive path as a Separation Point. The user vehicle sends the requested service information including Separation Point to the currently optimal service vehicle (line 17). A failed competitor needs to compete for the remaining sub-optimal server vehicles (line 12) until the competition is successful (line 22) or the candidate set becomes an empty set (line 14).

Algorithm 1 Online Instability-aware Computation Offloading Algorithm (Server vehicle version)

Input: Check points set array $CP1_j$ for each server vehicle of v_j within T; Current time slot t; Time slot length ;
Output: Matching customer vehicle u_i; Matching duration slots td;
1: **if** v_j has provided service **then**
2: **return** The current state of service;
3: **end if**
4: Through V2V broadcast the encrypted navigation route $CP1_j$, the current power that can provide for service, and the maximum idle computing capacity of v_j to the surrounding;
5: **for** m = 1 to n1 **do**
6: Accept a set $S1$ of user vehicles that request for matching in a certain amount of time =$n1$;
7: **if** $S1$ = = ? **then**
8: **continue**;
9: **end if**
10: Accept a set $S2$ that represents common path separation points of $S1$;
11: Calculate the maximum common path customer vehicle u_i, the service matching duration slots td according to $S2$ by calling SPM;
12: Match v_j with u_i, and send a confirmation message to u_i;
13: Update binary indicators x, y, z;
14: **return** u_i, td;
15: **end for**
16: **return** 0, 0;

Some variables with the same name appear in Algorithm 1 and Algorithm 2, such as $S1$ and $S2$, which are independent of each other in different algorithms. And the fourth line of Algorithm 1 and the fourth line of Algorithm 2 describe the process of moving path, resource and energy information transfer between vehicles.

OICO not only determines the allocation of vehicles in the current time slot, but also makes use of the advantages of ride-hailing to allocate the partial decisions of vehicles in future multiple time slots in advance (such as Algorithm 2, line 8). Then, OICO calls SPM to find the current best candidate vehicles for customer vehicle u. As mentioned, if u is selected to match the server vehicle out of all candidate customer vehicles, the unselected candidate customer vehicles will move on to compete for other server vehicles. Once a pair of server vehicle and customer vehicle matches successfully, they will share the same period of time slots along the common route. The speeds of two vehicles are then adjusted to the same to ensure the continuity of computation offloading service. OICO can achieve a stable MEC service by maintaining the same speed of two vehicles as much as possible without affecting the main business of the vehicle. In OICO, the customer's vehicle does not need to send its navigation information to any vehicle, only an encrypted common path separation coordinate point is sent to the target server vehicle, which protects the path privacy of the customer vehicle and avoids disclosing the passengers' track to a certain extent.

Algorithm 2 Online Instability-aware Computation Offloading Algorithm (Customer vehicle version)

Input: Check points set array $CP2_i$ for each customer vehicle of u_i within T; Current time slot t; Time slot length ;
Output: Matching customer vehicle v_j;
1: **if** u_i has been serviced **then**
2: **return** The current state of service;
3: **end if**
4: Accept a collection $S1$ of server vehicles that provide service via V2V;
5: **if** $S1$ = ? **then**
6: **return** 0;
7: **end if**
8: Select top $n1$ idle server vehicles $S2$ from set $S1$ that satisfy (4), (5), (6), (7), (8), (9), sorting by common driving path length according to SPM;
9: **if** $S2$ == ? **then**
10: **return** 0;
11: **end if**
12: **for** m = 1 to n1 **do**
13: **if** $S2$ == ? **then**
14: **return** 0;
15: **end if**
16: Select the first candidate service vehicle v_j from $S2$;
17: Send common path Separation Point to v_j;
18: **if** v_j refuses to provide the service **then**
19: Remove v_j from $S2$, wait a certain amount of time =$n1$;
20: **continue**;
21: **else**
22: Service matching with v_j, update binary indicators x, y, z;
23: **return** v_j;
24: **end if**
25: **end for**
26: **return** 0;

4.3 Service Path Matching Algorithm

The Service Path Matching (SPM) algorithm, is used to assist OICO to select the most suitable server vehicles from the set of candidate vehicles $V2$. The procedure of SPM is listed in Algorithm 3. The current order route for ride-hailing is known. We match the current route of the vehicle u in order to be served and the vehicles in $V2$. In line 1, $p_i^u[2]$ means the *check_time* in p_i^u. In each cycle, we remove all candidate vehicles whose current time slot does not meet the relevant constraints, leaving only one (line 7). The time range of our filtering method is small enough to flag the common path. One of the evaluation indexes, waiting time, is reflected in SPM. At each time slot, we detect whether the vehicles in the set $V2$ are out of the V2V communication distance of the customer vehicle u (lines 5-7). Note that we have extended the detection range of vehicles in $V2$ to x , where x is an artificially controlled constant. In this way, we can control that the waiting time of potential server vehicles does not exceed x .

At the same time, a more stable V2V edge computing service can be provided by controlling the server vehicle to adjust its running speed within a certain acceptable time. The $t^c_{u;v;t}$ mentioned above in Sec. II can be given by SPM. And if only one server vehicle $v \in V$ is passed in the incoming server parameter, SPM will give $t^c_{u;v;t}$ for that vehicle $v \in V$ if it serves $u \in U$ at time slot $t \in T$. Specifically, if the input includes $V2 = v$, and the check point information set is related to u, the *check_time* is in time slot t, then $t^c_{u;v;t} = \frac{jt^{end1} - t^{end2}j}{n}$. Meanwhile, in the time interval $[t; t + n - 1]$, the value of $t^c_{u;v;t}$ equals to $\frac{jt^{end1} - t^{end2}j}{n}$.

Algorithm 3 Service Path Matching Algorithm

Input: The vehicle u_i that request service; Check point information set p_i^u = [*longitude*; *latitude*; *check_time*] of vehicle u_i; Potential server vehicle set array $V2$; Check points set array $P2$ for each customer vehicle of $V2$;
Output: The selected best trajectory matching server vehicle v^s; Number of slots for the duration of the service n ; Service ending time point t^{end1}; Selected server vehicle's service adjustment ending time point t^{end2};
1: $t^{end1} = p_i^u[2]$, $t^{end2} = p_i^u[2]$, $t^{temp} = p_i^u[2]$, $n = 1$;
2: **for** true **do**
3: Take the check point p_i^{temp} = [*longitude*; *latitude*; *check_time*] of u_i, where the check time is the maximum time data check time of u_i in range $[t^{temp}; t^{temp} +]$;
4: **for** each $v2_j \in V2$ **do**
5: Extract all check points of $v2_j$ of the time range $[Maxfp_i^{temp}[2] \quad x \quad ; p_i^u[2]g; p_i^{temp}[2] + x \quad]$ as $P3$;
6: Select the check point closest to p_i^{temp} in $P3$ as p^{close};
7: **if** the distance between check point p^{close} and p_i^{temp} is greater than M or not satisfy (4), (5), (6), (7), (8), (9) **then**
8: **if** $jV2j$==1 **then**
9: **return** $v2_j; n \ ; t^{end1}; t^{end2}$;
10: **end if**
11: Remove $v2_j$ from $V2$;
12: **else**
13: $t^{end2} = p^{close}[2]$;
14: $v^s = v2_j$;
15: **end if**
16: **end for**
17: $t^{temp} = t^{temp} + \ ;$
18: $n = n + 1$;
19: $t^{end1} = p_i^{temp}[2]$;
20: **end for**
21: **return** $v^s; n \ ; t^{end1}; t^{end2}$;

4.4 Algorithm Optimality Analysis

Now we give a brief optimality analysis of OICO. We analyze the worst-case performance bound of OICO algorithm. We use (OPT) to represent the optimal out of service time.

Theorem 3. *The worst-case performance bound of OICO is* $\frac{H+jTjl_{max}}{H+jTjl_{min}}$, *where* $H = \sum_{u=1}^{u=\mathbb{P}^{Uj}} t_u$, $l_{max} = \max_{t2T} l_t$, $l_{min} = \min_{t2T} l_t$, $l_t = \sum_{v=1}^{v=\mathbb{P}^{Vj}} z^t_{u;v} + \sum_{u=1}^{u=\mathbb{P}^{Uj}} t^c_{u;v;t} \ z^t_{u;v} + \sum_{v=0}^{v=\mathbb{P}^{Vj}} jz^t_{u;v} \quad z^t_{u;v}{}^1 j \ P_u$, $P_u = \sum_{d=1}^{d=\mathbb{P}^{Dj}} t^{swc}_d \ y^t_{u;d}$=2, *and* $z^0_{u;v} = z^1_{u;v}$.

Proof. Assume that at any time slot $t \in T$, (OICO) represents the out of service time of OICO as defined in Eq. (15), while (OPT) represents the optimal out of service time.

$$\frac{\text{(OICO)}}{\text{(OPT)}} = \frac{\sum_{u=1}^{|U|}(t_u^{mis} + t_u^{swc}) + \sum_{v=1}^{|V|} t_v^c}{\sum_{u=1}^{|U|}(t_u^{mis} + t_u^{swc}) + \sum_{v=1}^{|V|} t_v^c}$$

$$\frac{\sum_{u=1}^{|U|} t_u + \sum_{t=1}^{|T|}\left(\sum_{v=1}^{|V|} z_{u;v}^t\right) + \sum_{u=1}^{|U|} t_u^{swc} + \sum_{v=1}^{|V|} t_v^c}{\sum_{u=1}^{|U|} t_u + \sum_{t=1}^{|T|}\left(\sum_{v=1}^{|V|} z_{u;v}^t\right) + \sum_{u=1}^{|U|} t_u^{swc} + \sum_{v=1}^{|V|} t_v^c}$$

$$\frac{H + \sum_{t=1}^{|T|}\left(\sum_{u=1}^{|U|} t_{u;v;t}^c \; z_{u;v}^t \sum_{v=1}^{|V|} z_{u;v}^t\right) + \sum_{u=1}^{|U|} t_u^{swc}}{H + \sum_{t=1}^{|T|}\left(\sum_{u=1}^{|U|} t_{u;v;t}^c \; z_{u;v}^t \sum_{v=1}^{|V|} z_{u;v}^t\right) + \sum_{u=1}^{|U|} t_u^{swc}}$$

$$\frac{H + \sum_{t=1}^{|T|}\left(\sum_{u=1}^{|U|} t_{u;v;t}^c \; z_{u;v}^t \sum_{v=1}^{|V|} z_{u;v}^t + EQ_{u;t}^{swc}\right)}{H + \sum_{t=1}^{|T|}\left(\sum_{u=1}^{|U|} t_{u;v;t}^c \; z_{u;v}^t \sum_{v=1}^{|V|} z_{u;v}^t + EQ_{u;t}^{swc}\right)}$$

$$\frac{H + |T|\,|U| \; t_{max}}{H + \sum_{t=1}^{|T|}\left(\sum_{u=1}^{|U|} t_{u;v;t}^c \; z_{u;v}^t \sum_{t=1}^{|V|} z_{u;v}^t + EQ_{u;t}^{swc}\right)}$$

$$\frac{H + |T|\,|U| \; t_{max}}{H + |T|\,|U| \; t_{min}}$$

where $z_{u;v}^t$ is the optimal solution of OPT, $EQ_{u;t}^{swc} = \sum_{v=0}^{|V|} |z_{u;v}^t - z_{u;v}^{t-1}| \; P_u$ and $EQ_{u;t}^{swc} = \sum_{v=0}^{|V|} |z_{u;v}^t - z_{u;v}^{t-1}| \; P_u$. □

We started with the relationship between expected total service time and service mismatching time, and made a series of scaling. Then we obtain the ratio of OICO and the optimal result (OPT) through multi-step mathematical deduction. The worst-case performance bound of OICO is $\frac{H+|T|\,|U|\,t_{max}}{H+|T|\,|U|\,t_{min}}$.

4.5 Algorithm Complexity Analysis

SPM, which is used to assist OICO to select the most suitable server vehicles from the set of candidate vehicles $V2$, cyclically matches the A candidate vehicles passing by (Algorithm 3, line 4). Suppose that each vehicle, not in service currently, has a maximum number of A candidate vehicles within the effective V2V communication distance. A's number can be determined by space constraints. Assume that the maximum check times for all server vehicles in the total time slot is B times (Algorithm 3, line 2). Among A candidate vehicles, the maximum order duration is B time slots. Thus, SPM has a time complexity $O(A \quad B)$.

For OICO, the complexity mainly comes from the user vehicle algorithm. Decisions are made for server vehicles at each time slot. Then, each customer vehicle that has not met its demand of service calculates the serviceability of surrounding vehicles, which will cost a constant time (Algorithm 1, line 4; Algorithm 2, line 4, respectively). For the distributed algorithm of each vehicle, the accumulated time complexity so far is A. OICO algorithm then calls SPM algorithm to select the best candidate vehicle. Combined with the complexity of SPM, OICO obtained the ranking of the candidate vehicle set. At the same time, distributed OICO competed for the candidate service set in turn, which lasts at most A times (Algorithm 2, line 8). So far, OICO has a time complexity of $O(A^2 \quad B)$. To sum up, the time complexity of distributed OICO is $O(A^2 \quad B)$.

5 Evaluation

In this section, we first introduce the evaluation setup based on real-world traces and the schemes for comparison. Then we discuss the performance evaluation results of OICO and provide some analyses and insights.

5.1 Simulation Setup

We adopt the KDD CUP 2020 driving track data set from DiDi's GAIA open dataset [17]. The data set is obtained from the track data and order data of DiDi vehicle platform in a local area of Chengdu. The acquisition interval of locus points is 2-4 s (including latitude, longitude and time information). The locus point has been treated with road binding, which ensures that the data can correspond to the actual road information. The driver and order information is encrypted, desensitized and anonymized. We randomly select the data of 1,000 vehicles, where we select 200 vehicles as customer vehicles and the other 800 vehicles as server vehicles. The experiment data set contains more than 1.92 million check-in records. We set the effective V2V communication distance M as 200 m [49]. In the experiment, we set the time slot to 60 seconds. According to the common calculation and communication power consumption, we give a reasonable power consumption range. Depending on the type of service, we set the power consumption for computation offloading ranging from 200 to 400 watts. Similarly, we set power consumption for V2V communication between 0.5 and 2 watts. The number of computation task types is 5, and the corresponding offloading ratio $_d$ ranges from 0.1 to 0.9. Tasks requested for each customer vehicle are not needed all the time. For a customer vehicle that needs service, we assume that it needs service only during the period when there are passengers in the vehicle. Whether there are passengers in the vehicle is not simulated, but the real behavior in DiDi's driving data set.

We compare OICO algorithm with the following two algorithms:

FSM. First-hit Service Matching algorithm, where customer vehicles match the first-hit nearest server vehicle [52].

NMM. Non-waiting Maximum Matching algorithm, where customer vehicles greedily select the server vehicle with the maximum matching path but do not compromise on deceleration. Heuristic algorithms based on basic greedy strategy are common in MEC related work [53].

FSM and NMM are both the existing frameworks of centralized computing offloading system. Compared with those existing frameworks of centralized computing offloading system, the proposed method OICO is distributed, requiring only local information to make decisions.

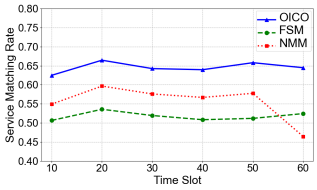

Fig. 3: The average service matching rate per ten minutes.

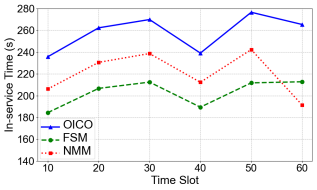

Fig. 4: The average total in-service time per ten minutes.

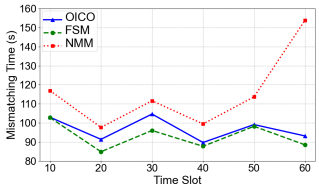

Fig. 5: The average service mismatching time per ten minutes.

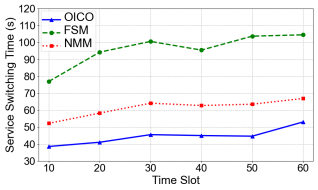

Fig. 6: The average service switching time per ten minutes.

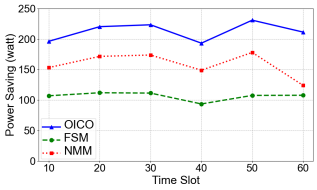

Fig. 7: The average power saving of a customer vehicle per ten minutes.

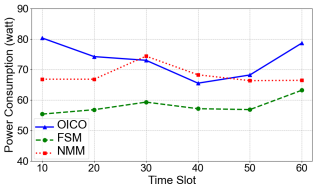

Fig. 8: The average power consumption of a server vehicle per ten minutes.

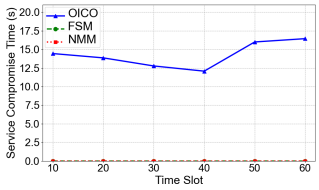

Fig. 9: The average service compromise time per ten minutes.

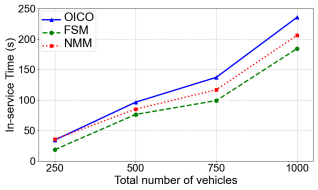

Fig. 10: The trend of total in-service time with total vehicle size per ten minutes.

5.2 Performance Evaluation Results

Service Matching Rate. The service matching rate is defined by the service matching time divided by the expected total service time, which reflects the performance of OICO in providing stable and continuous computation offloading. As shown in Fig. 3, the average service matching rates of OICO, FSM, and NMM are 64.57%, 51.77%, and 55.51%, respectively. Compared with NMM and FSM, OICO improves the service matching rate by 16.32% and 24.73%, respectively.

Total In-service Time. During the service process, the matching and switching of the service affect the actual service time. The actual total in-service time reflects the user's real experience. Fig. 4 shows that the actual service time of users is affected by the results of service matching and service switching. In particular, the average actual service time of OICO, FSM and NMM per ten minutes for each customer vehicle is 4.30 minutes, 3.38 minutes and 3.67 minutes, respectively. In other words, OICO improves the in-service time by up to 27.22% as compared with other schemes.

Service Mismatching Time. As plotted in Fig. 5, FSM's service mismatching time is minimal because it does not consider service interruptions caused by service switching. Since FSM is based on greedy concepts and chooses the first-matched vehicle. However, OICO takes the service interruptions caused by frequent switching into account, and sometimes it would rather not match when there is no suitable candidate. In particular, the average service mismatching time per ten minutes of OICO, FSM and NMM is 96.82 seconds, 93.01 seconds and 115.54 seconds, respectively.

Service Switching Time. A service switching occurs when computing hardware resources are replaced. In fact, OICO takes into account the service interruptions caused by frequent switching, and neither FSM nor NMM optimizes the costly service switching overhead. As illustrated in Fig. 6, the average service switching time per ten minutes for OICO is 44.65 seconds, and that for FSM and NMM is 95.93 seconds and 61.32 seconds, respectively. In short, OICO can efficiently reduce the average switching time by 53.45% and by 27.17% as compared with FSM and NMM.

Power Saving of Customer Vehicle. Customer vehicle gets the offloading service of mobile edge computing, and it achieves energy saving by transferring the computing to server vehicle with sufficient power. Fig. 7 plots the average power savings of three algorithms per customer vehicle in ten minutes, where OICO saves averagely 212.66 watts, FSM and NMM save averagely 106.56 watts and 158.29 watts of power per vehicle, respectively. Since the average power consumption of processing all the computations in customer vehicle (without offloading to another vehicle) is 302.82 watts, OICO can reduce the power consumption by about 54.06% and 37.62%, as compared with FSM and NMM.

Power Consumption of Server Vehicle. The server vehicles can gain some profits by offering services at the expense of their own computation resources and power consumption. Fig. 8 shows the average power consumption of three algorithms every ten minutes. OICO provides the highest average power consumption, about 73.35 watts. The NMM algorithm achieves the second-largest average power consumption, about 68.20 watts, while FSM consumes about 58.13 watts. In the experiment of Power Consumption of Server Vehicle, OICO performed slightly worse than comparison method NMM in some time slots. One of the main factors is: Compared with NMM, OICO comprehensively considers the optimization of subsequent time slots, so the optimization does not show much in some time slots, but more benefits are obtained in subsequent time slots. In short, the server vehicles using OICO can achieve the highest total profits as compared with other schemes.

Total Out-of-service Time. As we defined in Sec. III, the total out-of-service time is our comprehensive performance metric. The total out-of-service time of FSM is the highest in most cases. In contrast, OICO and NMM show the same trend in the total out-of-service time. In particular, the average total out-of-service time per ten minutes of OICO, FSM and NMM is 405.64 minutes, 448.58 minutes and 416.96 minutes.

Service Compromise Time. To ensure the connectivity and stability in computation offloading process, the server

vehicles using OICO sometimes need to decelerate to the matched customer vehicle's speed. As plotted in Fig. 9, the average service compromise time for a server vehicle is 14.26 seconds per ten minutes. On the contrary, FSM and NMM do not adjust the vehicle speed, so the service compromise time is always zero. However, this will lead to disconnection of a pair of matched customer and server vehicles, since they are driving at different speeds. Sooner or later, they will drive out of the effective V2V communication distance and the service will be interrupted. Thus, even though OICO pays a little cost of the service compromise time, it efficiently increases the in-service time and reduces the service switching times.

Scalability Analysis. In-service time is an important metric in this scenario. We perform scalability analysis on the In-service time. Total in-service time reflects the user's real experience. Fig. 10 shows the trend of total in-service time with total vehicle size per ten minutes. The ratio of server vehicles to customer vehicles remains 8: 2. The results show that the proposed framework is scalable and applicable in large-scale settings.

In-service time, service matching time and power saving can intuitively reflect the efficiency and quality of edge computing services based on V2V networks. Extensive real-world evaluations have shown that OICO outperforms other existing solutions, specifically improving service matching by 25% and reducing power consumption by about 54% per customer vehicle. At the same time, the expansibility of the proposed method is analyzed, and the results show that the proposed method has good expansibility.

6 Discussion

V2V-based edge computing services are suitable for ride-hailing companies such as Uber. They can use navigation information to improve the QoS of MEC and increase the additional profits of server vehicle owner. In this section, we discuss privacy and security concerns under this pattern.

Privacy and Personal Safety. Since there is no binding association between passengers and online ride-hailing vehicles, passenger information is anonymous when a vehicle driving track is transmitted between online ride-hailing vehicles. In this way, the private track data of passengers can be protected. At the same time, as online ride-hailing becomes a public service, the established route of online ride-hailing is under supervision of the responsible organization or company. Besides, ride-hailing companies are paying more and more attention to the safety of passengers. For example, in many places around the world, when online ride-hailing vehicles deliberately deviate from the predetermined navigation route, alarm information will be seen immediately to ensure the safety of passengers.

Under the MEC-enabled V2V pattern, the matched server vehicle and customer vehicle are agreed to share the planned route. By monitoring the driving route of vehicles, the ride-hailing companies can send alarms to customers when their cars deviate from the planned routes. In case that the MEC-enabled V2V is carried out, two vehicles will drive in coordination, which improves the safety of passengers to a certain extent and prevents the infringement of passengers by drivers. For example, DiDi will continuously track the driving trajectory during the journey of customers. And DiDi will monitor and intervene for abnormal trajectory and yaw behavior [54]. Through the European Strategy on Cooperation-Intelligent Transportation Systems (C-ITS), the EU has identified a number of V2V services as candidates for early deployment, where V2V can be applied for connected and cooperative navigation, vulnerable road user protection [55].

Traffic Safety. In terms of traffic safety, server vehicle and customer vehicle can cooperate with each other at the same speed through vehicle queue control technology during the service process. Collaborative driving is beneficial to collision avoidance, emergency braking, improving road capacity and increasing the safety of driving process [56], and reducing energy consumption by bringing vehicles together to reduce air drag [57], [58]. Under these circumstances, our proposed method can not only improve the service stability, but also indirectly improve the traffic safety and driving safety.

7 Conclusion

In this paper, we study the online computation offloading problem under complex vehicle mobility in MEC-enabled V2V networks, considering unpredictable variations of traffic conditions and real-time mobility of vehicles. The problem is modeled as a distributed online service optimization problem and proved to be NP-hard, which takes account of service instability, service switching costs, power, resources, and delay constraints. In order to minimize the out-of-service time (i.e., service mismatching, service switching and service compromise time), we propose a distributed OICO algorithm together with an efficient SPM algorithm, which can effectively improve the profits of both customer vehicles and server vehicles. Extensive evaluations based on real-world traces demonstrate that OICO outperforms other existing schemes, specifically, it increases service matching rate by 25% and reduces power consumption by about 54% per customer vehicle.

References

[1] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for V2V communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[2] L. Wang, J. Yan, K. Yu, and D. Deng, "Research of D2D communications mode for 5G vehicular networks," in *2019 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2019, pp. 1–6.

[3] N. Alliance, "5G white paper," *Next generation mobile networks, white paper*, vol. 1, 2015.

[4] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5G," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[5] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling," *IEEE Vehicular Technology Magazine*, vol. 14, no. 1, pp. 28–36, 2018.

[6] C. Chen, L. Chen, L. Liu, S. He, X. Yuan, D. Lan, and Z. Chen, "Delay-optimized V2V-based computation offloading in urban vehicular edge computing and networks," *IEEE Access*, vol. 8, pp. 18 863–18 873, 2020.

[7] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless communications*, vol. 20, no. 3, pp. 14–22, 2013.

[8] "Vehicular game in tesla electric cars." [Online]. Available: https://www.theverge.com/2021/1/27/22253258/tesla-model-s-ps5-xbox-series-x-next-gen-10-teraflop

[9] D. Han, W. Chen, and Y. Fang, "A dynamic pricing strategy for vehicle assisted mobile edge computing systems," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 420–423, 2018.

[10] C.-M. Huang, M.-S. Chiang, D.-T. Dao, W.-L. Su, S. Xu, and H. Zhou, "V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture," *IEEE Access*, vol. 6, pp. 17 741–17 755, 2018.

[11] L. Liang, H. Peng, G. Y. Li, and X. Shen, "Vehicular communications: A physical layer perspective," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 647–10 659, 2017.

[12] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[13] J. Zhang, X. Hu, Z. Ning, E. C.-H. Ngai, L. Zhou, J. Wei, J. Cheng, and B. Hu, "Energy-latency tradeoff for energy-aware offloading in mobile edge computing networks," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2633–2645, 2017.

[14] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *IEEE International Conference on Computer Communications (INFOCOM)*. IEEE, 2019, pp. 1459–1467.

[15] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[16] Z. Ning, X. Wang, and J. Huang, "Mobile edge computing-enabled 5G vehicular networks: Toward the integration of communication and computing," *IEEE vehicular technology magazine*, vol. 14, no. 1, pp. 54–61, 2018.

[17] "Data source: DiDi chuxing." [Online]. Available: https://gaia.didichuxing.com

[18] P. Jin, X. Fei, Q. Zhang, F. Liu, and B. Li, "Latency-aware vnf chain deployment with efficient resource reuse at network edge," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 267–276.

[19] M. Li, Q. Zhang, and F. Liu, "Finedge: A dynamic cost-efficient edge resource management platform for nfv network," in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*. IEEE, 2020, pp. 1–10.

[20] Q. Zhang, F. Liu, and C. Zeng, "Online adaptive interference-aware vnf deployment and migration for 5g network slice," *IEEE/ACM Transactions on Networking*, 2021.

[21] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "On-edge multi-task transfer learning: Model and practice with data-driven task allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1357–1371, 2019.

[22] S. Chen, L. Jiao, F. Liu, and L. Wang, "Edgedr: An online mechanism design for demand response in edge clouds," *IEEE Transactions on Parallel and Distributed Systems*, 2021.

[23] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, and B. Li, "etime: Energy-efficient transmission between cloud and mobile devices," in *2013 Proceedings IEEE INFOCOM*. IEEE, 2013, pp. 195–199.

[24] T. Zhang, X. Zhang, F. Liu, H. Leng, Q. Yu, and G. Liang, "etrain: Making wasted energy useful by utilizing heartbeats for mobile data transmissions," in *2015 IEEE 35th International Conference on Distributed Computing Systems*. IEEE, 2015, pp. 113–122.

[25] X. Yi, L. Pan, Y. Jin, F. Liu, and M. Chen, "Edirect: Energy-efficient d2d-assisted relaying framework for cellular signaling reduction," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 860–873, 2020.

[26] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE transactions on computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[27] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[28] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *2016 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2016, pp. 1451–1455.

[29] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[30] Z. Jiang and S. Mao, "Energy delay tradeoff in cloud offloading for multi-core mobile devices," *IEEE Access*, vol. 3, pp. 2306–2316, 2015.

[31] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[32] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[33] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–7.

[34] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26 652–26 664, 2019.

[35] I. Ketykó, L. Kecskés, C. Nemes, and L. Farkas, "Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing," in *2016 European Conference on Networks and Communications (EuCNC)*. IEEE, 2016, pp. 225–229.

[36] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432–7445, 2017.

[37] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Transactions on Mobile Computing*, 2021.

[38] Y. Zhang, J. Lopez, and Z. Wang, "Mobile edge computing for vehicular networks [from the guest editors]," *IEEE Vehicular Technology Magazine*, vol. 14, no. 1, pp. 27–108, 2019.

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

[40] 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 Communications Magazine*, vol. 55, no. 7, pp. 94–100, 2017.

[41] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 2016, pp. 1122–1127.

[42] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Network*, vol. 32, no. 1, pp. 80–86, 2018.

[43] S. A. R. Naqvi, H. Pervaiz, S. A. Hassan, L. Musavian, Q. Ni, M. A. Imran, X. Ge, and R. Tafazolli, "Energy-aware radio resource management in D2D-enabled multi-tier hetnets," *IEEE Access*, vol. 6, pp. 16 610–16 622, 2018.

[44] Y. Yang, G. Song, W. Zhang, X. Ge, and C. Wang, "Neighbor-aware multiple access protocol for 5G mmtc applications," *China Communications*, vol. 13, no. 2, pp. 80–88, 2016.

[45] X. Xu, Y. Chen, Y. Zhao, S. He, and H. Song, "Delay efficient D2D communications over 5G edge-computing mobile networks," in *Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019)*. Springer, 2020, pp. 1249–1260.

[46] P.-J. Wan, "Multiflows in multihop wireless networks," in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*, 2009, pp. 85–94.

[47] X. Jiao, W. Lou, X. Wang, J. Cao, M. Xu, and X. Zhou, "Data aggregation scheduling in uncoordinated duty-cycled wireless sensor networks under protocol interference model." *Ad Hoc Sens. Wirel. Networks*, vol. 15, no. 2-4, pp. 315–338, 2012.

[48] C. F. Mecklenbrauker, A. F. Molisch, J. Karedal, F. Tufvesson, A. Paier, L. Bernadó, T. Zemen, O. Klemp, and N. Czink, "Vehicular channel characterization and its implications for wireless system design and performance," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1189–1212, 2011.

[49] M. Sepulcre and J. Gozalvez, "Experimental evaluation of cooperative active safety applications based on V2V communications," in *Proceedings of the ninth ACM international workshop on Vehicular inter-networking, systems, and applications*, 2012, pp. 13–20.

[50] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1405–1418, 2020.
[51] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Management science*, vol. 32, no. 9, pp. 1095–1103, 1986.
[52] P. Zhang, M. Durresi, and A. Durresi, "Mobile privacy protection enhanced with multi-access edge computing," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2018, pp. 724–731.
[53] M. Guo, X. Huang, W. Wang, B. Liang, Y. Yang, L. Zhang, and L. Chen, "Hagp: A heuristic algorithm based on greedy policy for task offloading with reliability of mds in mec of the industrial internet," *Sensors*, vol. 21, no. 10, p. 3513, 2021.
[54] "DiDi improved ability to detect route abnormalities." [Online]. Available: https://technode.com/2020/03/30/didi-has-resumed-late-night-hours-for-carpooling-service-hitch
[55] B. M. Masini, A. Bazzi, and A. Zanella, "A survey on the roadmap to mandate on board connectivity and enable V2V-based vehicular sensor networks," *Sensors*, vol. 18, no. 7, p. 2207, 2018.
[56] E. Mitsakis and I. S. Anapali, "Recent developments on security and privacy of V2V & V2I communications: A literature review," *Periodica Polytechnica Transportation Engineering*, vol. 48, no. 4, pp. 377–383, 2020.
[57] F. Lin, K. Wang, Y. Zhao, and S. Wang, "Integrated avoid collision control of autonomous vehicle based on trajectory re-planning and V2V information interaction," *Sensors*, vol. 20, no. 4, p. 1079, 2020.
[58] T. R. Gonçalves, V. S. Varma, and S. E. Elayoubi, "Vehicle platooning schemes considering V2V communications: A joint communication/control approach," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2020, pp. 1–6.

Fangming Liu (S'08, M'11, SM'16) received the B.Eng. degree from the Tsinghua University, Beijing, and the Ph.D. degree from the Hong Kong University of Science and Technology, Hong Kong. He is currently a Full Professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and edge computing, datacenter and green computing, SDN/NFV/5G and applied ML/AI. He received the National Natural Science Fund (NSFC) for Excellent Young Scholars, and the National Program Special Support for Top-Notch Young Professionals. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019, ACM e-Energy 2018 and IEEE GLOBECOM 2011, the First Class Prize of Natural Science of Ministry of Education in China, as well as the Second Class Prize of National Natural Science Award in China.

Jian Chen received his B.Eng. degree from School of Computer Science and Technology, Huazhong University of Science and Technology, China in 2019. He is currently a master student in School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include Internet of Vehicles, cloud computing and edge computing, 5G network and network slicing.

Qixia Zhang received his B.Eng. degree from School of Computer Science and Technology, Huazhong University of Science and Technology, China in 2016. He is currently a Ph.D. student in School of Computer Science and Technology, Huazhong University of Science and Technology. His research interests include network function virtualization, cloud computing and edge computing, datacenter and green computing, 5G network and network slicing. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019.

Bo Li (S'89-M'92-SM'99-F'11) is a Chair Professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, which he has been affiliated with since 1996. He held the Cheung Kong Chair Professor in Shanghai Jiao Tong University between 2010 and 2015. His research interests cover broad areas in networking and distributed systems, with recent focuses on big data and machine learning systems, cloud and edge computing. He was a co-recipient of seven Best Paper Awards from IEEE including the Test-of-Time Paper Award from IEEE INFOCOM (2015) and the Best Paper Award from IEEE INFOCOM (2021). He received his PhD in the Electrical and Computer Engineering from University of Massachusetts at Amherst, and his B. Eng. (summa cum laude) in the Computer Science from Tsinghua University, Beijing, China.