MCVCO: Multi-MEC Cooperative Vehicular Computation Offloading

Jianhang Liu, Kunlei Xue, Qinghai Miao, *Senior Member, IEEE*, Shibao Li, Xuerong Cui, Danxin Wang, and Kewen Li

***Abstract*—Mobile edge computing (MEC) has been envisioned as a promising paradigm that provides processing resources for vehicular computation-intensive tasks to accommodate the strict latency requirement. However, there is still a need to further enhance system performance to overcome challenges such as poor efficiency of data transmission and limited system resources. To improve the quality of service, this article proposes a multi-MEC cooperative vehicular computation offloading (MCVCO) scheme. Firstly, we propose a heat-aware task offloading strategy to capture the time-varying multi-link relations between vehicle and MEC nodes. Secondly, we design a multi-MEC resource compensation method based on fountain code which cooperatively collects the task data and improves the efficiency of data reception in the edge layer. Finally, we develop a parallel transmission and execution based dynamic scheduling algorithm to make the most of available resources. Extensive simulation results and analyses demonstrate that MCVCO outperforms other baseline schemes in various experimental settings. MCVCO achieves a 32% increase in success rate, up to a 47% reduction in end-to-end latency, and a 24% improvement in uploading quality.**

***Index Terms*—Computation offloading strategy, mobile edge computing(MEC), vehicular networks, Quality of Service.**

I. Introduction

With the development of intelligent transportation systems, the Internet of Vehicles (IoV) has drawn considerable attention from the industry and academia in recent years [1], [2], [3]. An intelligent transport network will be built in Europe and 130 million new energy vehicles are anticipated for transportation systems by 2035 [4]. This has led to a great development of vehicular applications aiming to improve traffic efficiency and enhance road safety, such as autonomous driving, augmented reality, as well as data distribution, etc [5]. These vehicular applications demand low latency, substantial processing resources, and reliable network connectivity, which are hard to accommodate by traditional local computing and cloud computing.

By offloading computation load at the edge rather than the remote cloud, Mobile Edge Computing (MEC) has been a promising solution for supporting vehicular applications with strict delay-constraint requirements. The MEC servers are deployed close to fixed wireless infrastructures, such as Access Points (APs) and Roadside Units (RSUs). This feature enables MEC to offload the high computation-intensive tasks from mobile vehicles via wireless communication, and significantly enhances the computing capacity of vehicles [6].

MEC-assisted networks are extensively employed for vehicular computation offloading. However, there are certain challenges that must be addressed to enhance the quality of service (QoS). These challenges encompass the following aspects.

1) The mobility of vehicles and the uneven spatial-temporal distribution of wireless infrastructures give rise to challenges such as frequent handoff and service interruption when vehicles traverse overlapping coverage areas of different MEC nodes. Moreover, the existing V2I communication architecture predominantly utilizes a single link model, which is susceptible to the fluctuating state of wireless channels. These aforementioned factors contribute to the occurrence of unexpected task offloading failures.
2) Given the intricate nature of the edge network environment, characterized by diverse edge server configurations and constrained computation resources, sequential task offloading to these servers may result in resource congestion and imbalances. Consequently, this can lead to inefficient resource utilization and inferior task execution performance.

Recently, many research efforts have been focused on computation offloading strategies and optimization schemes in MEC-assisted vehicle networks. It is widely acknowledged that the successful uploading of vehicle task data to the MEC nodes via a wireless network heavily relies on efficient data transmission. However, the majority of current research assumes stable channel conditions and constant communication bandwidth [7]. The noise and fading of wireless channels and unknown characteristics of MEC-assisted vehicle network conditions may decrease the successful transmission probability and prolong

Manuscript received 4 July 2023; accepted 21 July 2023. Date of publication 27 July 2023; date of current version 23 February 2024. This work was supported in part by the Natural Science Foundation of Shandong Province under Grant ZR2020MF005, in part by the National Natural Science Foundation of China (NSFC) under Grants 52171341, 61902431, and 61972417. *(Corresponding author: Qinghai Miao.)*

Jianhang Liu, Danxin Wang, and Kewen Li are with the College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China (e-mail: liujianhang@upc.edu.cn; wangdx@upc.edu.cn; likw@upc.edu.cn).

Kunlei Xue, Shibao Li, and Xuerong Cui are with the College of Oceanography and Space Informatics, China University of Petroleum (East China), Qingdao 266580, China (e-mail: kunleixue@163.com; Lishibao@upc.edu.cn; cuixuerong@163.com).

Qinghai Miao is with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: miaoqh@ucas.ac.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2023.3299381.

Digital Object Identifier 10.1109/TIV.2023.3299381

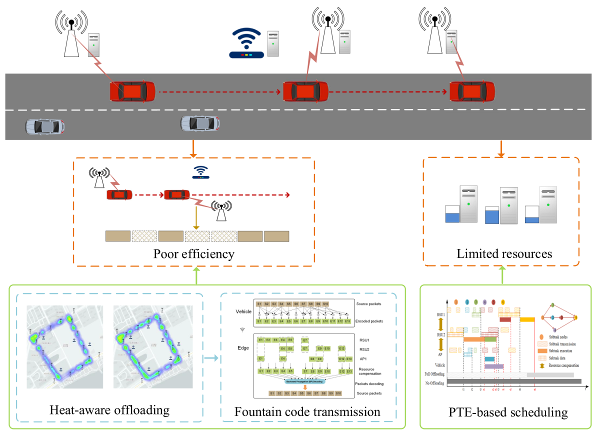

Fig. 1. Challenges and solutions of MEC-assisted vehicular computation offloading.

the task completion delay. Therefore, multi-link cooperative data uploading is of vital importance to improve system performance under the changing channel state. Furthermore, most existing works have designed offloading mechanisms according to the coarse-grained method, while the dependency and decomposability of the tasks are often neglected. However, a vehicular task can comprise multiple subtasks or threads [8]. For instance, an augmented reality task encompasses various components such as sensors, multimedia elements, and 3D models. The execution of these subtasks is dependent on their respective predecessors. Consequently, task dependency plays a crucial role in determining the scheduling strategy. Hence, compared with binary offloading method, a more comprehensive fine-grained task offloading scheme is required, e.g., considering the dependency relationship and the characteristics of MEC-assisted vehicular networks.

Based on the motivations above, we comprehensively investigate the scenario of cooperative task offloading in the MEC-assisted vehicular system, and propose a multi-MEC cooperative vehicular offloading (MCVCO) scheme to enhance the system performance in this article. Fig. 1 shows the challenges and our solutions in the MEC-assisted vehicular computation offloading scenario. To improve the efficiency of data uploading, we adopt a broadcast scheme to transmit fountain-encoded data. First, in order to effectively capture the dynamic relationships among multiple links between the vehicle and MEC nodes, we introduce the concept of “heat value” as a measure of the weighted duration of V2I alive links. Subsequently, we propose a heat-aware task offloading strategy, considering both the mobility characteristics of vehicles and the diverse distributions of MEC nodes. Then, based on the features of the fountain code, a multi-MEC resource compensation algorithm is designed. Specifically, the task data is encoded at the vehicle, and multiple MEC nodes collect the encoded packets in a cooperative manner to deal with the highly fluctuating wireless network and frequent handoff. Finally, to enhance the task transmission and execution performance, we propose a PTE-based dynamic computation scheduling algorithm that considers the dependency relationship of tasks, the available communication and computing resources.

To the best of our knowledge, there is a lack of research on integrating three key phases, i.e., offloading decision, data uploading, and computation scheduling to improve the system performance. The main contributions of our work are as follows.

1) In view of the mobility of the vehicle and frequent topology change problem, we propose a heat-aware task offloading strategy. This method could effectively take advantage of the multi-link data uploading mechanism and improve the efficiency of task offloading by 39%.
2) A multi-MEC cooperative resource compensation algorithm based on fountain code is proposed. Employing resource compensation and collision detection, the algorithm could avoid data confusion caused by wireless network interference and outperforms 24% in data-collecting.
3) In order to achieve high efficiency of scheduling, a parallel transmission and execution based dynamic scheduling method is designed. Compared with the single process based algorithm, the proposed method can reduce 32% computation delay by fully utilizing the limited computing and transmission resources.
4) The above three algorithms are integrated into MCVCO, and the performance of MCVCO is validated through extensive simulations. The results and analyses demonstrate that MCVCO could reduce end-to-end delay, enhance task offloading efficiency as well as the uploading quality.

The rest of this article is organized as follows. Section II reviews the related works in terms of different optimization schemes. Section III introduces the system models and problem formulation. Section IV presents the design of MCVCO. Section V compares MCVCO with the other four baselines and analyzes the results. Section VI summarizes this article.

II. Related Works

Mobile Edge Computing has been an efficacious paradigm for vehicular computation offloading and brings new vitality to VANET. In recent years, different studies have been explored, and a variety of optimization schemes and architectures were proposed to derive practical solutions. In this chapter, we will introduce the related studies from multiple aspects.

A. Optimization Schemes of Offloading Decision

Most vehicles are limited in resources [9], so it is of vital importance to deploy effective offloading decision-making schemes to enhance the quality of task service. Scholars have implemented plenty of research on offloading decision optimization. In response to the current problem of the mobility of the vehicle, Hoang et al. [10] put forward a mobility-aware computation offloading strategy by analysing the average costs according to the local and edge computation at the vehicle and the MEC nodes. Nowadays, MEC nodes are not deployed everywhere, causing unstable connectivity between MEC nodes and vehicles. Saeid et al. [11] proposed DIVINE which enables the vehicle to offload task data to edge nodes through V2V or V2I. DIVINE considers various factors to determine the offloading decision. These include the duration of connectivity between the target vehicle and the edge nodes, as well as other vehicles, the expected latency required to connect with MEC nodes, and the delay in establishing contact with nearby vehicles.

However, these above methods neglect the decomposability and dependence of tasks. Compared with coarse-grained offloading, fine-grained task offloading could fully exploit the benefits of parallel transmission and execution [12]. In [13], to keep the balance between accuracy and efficiency, Ren et al. designed a fine-grained computation framework and balanced the multiple costs by managing the limited hierarchical computation resources. In addition, Ren et al. developed a complex AR application to prove the concept, and the simulation results demonstrated the advantages of this cooperative approach. Ding et al. [14] proposed a code-oriented computation offloading algorithm to determine the position, computation resources, and transmission power for user equipment (UE) to decrease the computation overhead. The authors assumed that the users keep stationary while transmitting the task data. But the users may in motion, which makes the scenario far away from the real world.

B. Optimization Schemes of Data Uploading

As far as the present situation is concerned, the research on optimizing data uploading mainly relies on the assistance of physical layer resources. He et al. [15] investigated a NOMA-assisted system with passive eavesdropping vehicles and designed a power management and task scheduling scheme to decrease the entire completion delay. In addition, the proposed scheme also utilizes the near vehicles as duplex relays to provide assistance for data uploading. And the better performance on data security and completion delay are demonstrated through digital simulation. To overcome the challenges of unsymmetric network resource allocation for transmission links in MEC-assisted wireless networks, Wang et al. [16] designed FAIR, which is an end-to-end automotive edge networking scheme. The FAIR could offer robust connected services for task offloading vehicles in complicated edge scenes. With the high altitude and maneuverable mobility, unmanned aerial vehicle (UAV) assisted-MEC is becoming a promising technology to cope with the vehicular task offloading [17], [18]. Based on the advances in centralized management of Software-Defined Networking (SDN), Zhao et al. [19] proposed an SDN-enabled and UAV-assisted vehicular task offloading architecture. In the architecture, the UAV is used as an assistant node to help transmit task data to the MEC nodes. Meanwhile, the UAV can work as an independent server to execute the offloading tasks. Nevertheless, the UAV wireless communication network encounters significant challenges in managing computation-intensive and time-sensitive vehicular tasks due to the constrained computing resources, storage capacity, and small size of UAVs.

C. Optimization Schemes of Execution Scheduling

Liao et al. [20] put forward Dependency-Aware Application Assigning and Scheduling (DAAS) to tackle the task assigning and scheduling problem based on the priority of each task in an online manner. However, DAAS does not consider the function configuration, while the MEC server has limited computation resources. Recently, vehicle as a resource (VaaR) is booming to enlarge the MEC-assisted vehicular network capacity by exploiting the resources of nearby vehicles. Based on VaaR, Pham et al. [21] utilized a game-theoretic approach to devise a distributed offloading task algorithm with low complexity. This algorithm enables the determination of optimal resource allocation and offloading proportion through the implementation of a subgradient method.

Artificial Intelligence (AI)'s rapid development has generated various schemes and AI involves a great number of roles in intelligent transport systems [22], [23], [24]. Taking into account real-world scenarios, where task characteristics and computational capabilities at multiple servers may exhibit time-varying behavior, Li et al. [25] proposed an offloading strategy using deep Q-network (DQN) to address dynamic scheduling challenges. The authors initially modeled the problem as a Markov decision process and subsequently defined the state and action spaces. Finally, they employed a DQN-based strategy to solve the formulated problem. This method embodied the superior performance of DQN for dynamic problems. Based on DRL, Zhan et al. [26] designed an offloading scheduling algorithm taking into account both where to schedule and when to schedule. Besides, a parameter-shared network framework with CNN is used to approximate both value function and policy, which could get the system features effectively.

D. Optimization Schemes of Joint Optimization

Dai et al. [27] first designed an asynchronous deep Q-learning method to decide the task offloading decisions. In addition, the scheme achieved rapid convergence by conducting training on the partial model at local server and asynchronously updating the global model. Secondly, the authors derived an optimal analytic formulas by decomposing the resource allocation problem into multiple subproblems. Due to the highly dynamic MEC-assisted vehicular network environment, these existing single-phase optimization methods usually cannot fulfill the processing requirements of the application in vehicular networks. Dai et al. [28] investigated a novel scene of vehicular task computation offloading in MEC-assisted architecture, including coordinated task uploading, data migration and computation capabilities between different layer servers. To tackle the cooperative computation offloading (CCO) problem, the authors proposed a online scheduling algorithm according to the derived offloading probability. To minimize task offloading the computation latency, Zhu et al. [29] formulated the problem as a nonconvex optimization problem. They put forward a joint optimization approach that considers both the data transmission duration and the allocation of task computation. This approach leverages the high efficiency of NOMA in various channel scenarios. Aiming to decrease the task offloading latency, Mahmood et al. [30] designed an optimal resources allocation framework including local and edge computational frequency, subtask partition, and transmission power.

In conclusion, the existing works mainly focus on the single phases optimization but neglect the consistency and interaction of MEC-assisted vehicular system. Distinguished from previous studies, this article focuses on a cooperative vehicular task

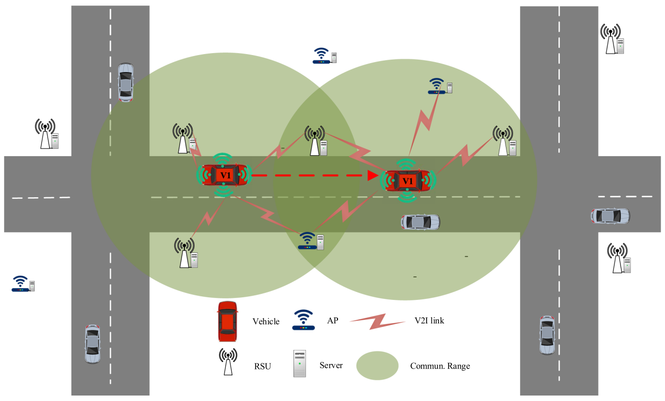

Fig. 2. System architecture.

offloading model that incorporates the processes of task offloading, data uploading, and resource scheduling. The model considers the challenges posed by vehicle mobility, uneven distribution MEC resources, high fluctuations in wireless network conditions, and heterogeneous capacities of MEC nodes.

III. System Model and Problem Formulation

A. Network Model

We investigate a heterogeneous architecture for cooperative task offloading in MEC-assisted vehicular networks. In this architecture, the MEC nodes, including roadside units (RSUs) and access points (APs), are deployed to provide coverage along the roads. However, due to the uneven spatial distribution and varying communication ranges of the wireless infrastructures, there exist overlapping sub-areas covered by multiple types of wireless interfaces. This introduces the possibility of the service vehicle V traversing different areas as a result of its high mobility. In addition, we assume that the vehicle is equipped with on-board units (OBUs), GPS, and other devices. System can get real-time positions (V_x, V_y) and velocity (v_x, v_y) of vehicle from GPS. The task data is uploaded to the edge network based on broadcast schemes. And the vehicle stops broadcasting encoded packets until the MEC nodes collect the complete data content. Moreover, the data uploading procedure is modeled as $N/N/1$ model, which indicates that other tasks have to wait until the data of the previous task data is uploaded. We assume wireless infrastructure and a MEC server constitute one MEC node M. And the MEC servers could offer computation service for vehicular offloading tasks. In addition, the MEC nodes have various computation capacities which are characterized by different processor numbers. The system architecture is shown in Fig. 2.

B. Task Model

In this article, we assume that the vehicular task is composed of several interdependent subtasks. Let Q denote the number of subtasks, and S_q denote the q-th subtask, where $1 \leqslant q \leqslant Q$. We describe the subtask by a tuple $S_q = (D_q, C_q, T_q)$, where D_q is the data size of S_q, $C_q = \alpha D_q$ is the required number of CPU

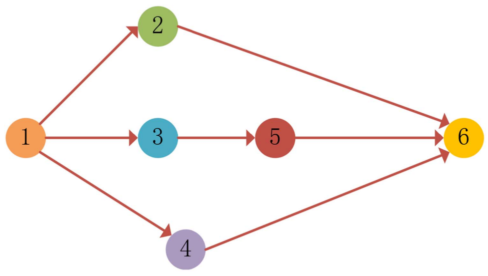

Fig. 3. Example of independent subtasks relationships.

cycles to execute the subtask, and T_q is the maximum delay tolerance. In addition, we model the dependency relationship of the subtasks by a DAG, G(S, R), where S $= \{S_j\}_Q$ is the subtask nodes set, R $= \{r_{j,k}\}_{Q*Q}$ is the relationship link matrix of the subtasks.

$$R_{Q\times Q} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1Q} \\ r_{21} & r_{22} & \cdots & r_{2Q} \\ \vdots & \vdots & \ddots & \vdots \\ r_{Q1} & r_{Q2} & \cdots & r_{QQ} \end{bmatrix} = [r_{jk}] \tag{1}$$

$r_{jk} = 1$ denotes that S_j is the predecessor subtask of S_k. Due to the dependency relationship, the successor subtasks cannot commence until the completion of the predecessor subtasks. As shown in Fig. 3, there is an example of subtasks relationships of a vehicle request. In this case, S_2, S_3, S_4 are the successor subtasks of S_1, so the r_{12}, r_{13}, $r_{14} = 1$. Note that the sequence defined here is an approximation for data uploading. Since the network environment may change, a dynamic scheduling is required, as shown in Section IV-C. The summary of key notations is presented in Table I.

C. Communication Model

We model the communication channel between the MEC nodes and vehicle as a flat block-fading Rayleigh channel. Let $d^{-\vartheta}$ denote the path loss, where d is the distance between vehicle and MEC nodes and $-\vartheta$ denotes the link path loss exponent. Furthermore, the transmit power of the vehicle, the channel bandwidth, the white Gaussian noise power, and the channel fading coefficient can be denoted as P_t, W_t, h, and N_0. Hence, we can obtain the transmission rate from

$$R_t = W_t \log_2 \left(1 + \frac{P_t d^{-\vartheta} |h|^2}{N_0}\right) \tag{2}$$

In this article, we do not consider the cost of retrieving the task execution results and exchanging intermediate data since the data size of these results is always much smaller than the task itself, which is widely adopted in [31].

D. Computation Model

In our study, we consider two processing models for vehicle tasks: the local computation model and the MEC computation model. The local computation model refers to the execution of tasks by the vehicle itself, while the MEC computation model

TABLE I
SUMMARY OF KEY NOTATIONS

Notation	Definition
V	The service vehicle
M	The MEC node
V_x	The horizontal position of V
v_x	The horizontal velocity of V
M_i^x	The horizontal position of M_i
f	The processing ability
R_t	The data transmission rate
S_q	The attribute tuple of subtasks
D_q	The data size of S_q
C_q	The amount of CPU cycles required for executing S_q
T_q	The maximum delay tolerance of S_q
r_{jk}	The dependency link among the subtasks
δ_q	The offloading decision of S_q
ε_q	The execution position decision of S_q
L_q^l	The local computation delay of S_q
L_q^o	The offloading computation delay of S_q
L_q^u	The transmission delay of S_q
L_l^a	The latency of awaiting local processor available.
$T_q^{e,s}$	The starting time slot of executing of S_q
$T_q^{e,c}$	The completing time slot of executing of S_q
$T_t^{l,a}$	The transmission resource available time of vehicle
$T_e^{l,a}$	The computation resource available time of vehicle
$T_e^{o,a}$	The computation resource available time of MEC nodes
$T_p^{e,c}$	The completion time of the preceding subtasks of S_q
LD_i	The link duration between M_i and V
P_i	The link reliability between M_i and V

involves offloading tasks to MEC nodes and executing them on MEC servers. To be more clear, we define a binary vector $\delta = \{\delta_q\}_Q$ to denote the offloading decisions, so that.

$$\delta_q = \begin{cases} 0, & \text{if } S_q \text{ is executed in the vehicle locally.} \\ 1, & \text{if } S_q \text{ is offloaded to MEC server.} \end{cases} \tag{3}$$

1) Local computation model: let f_l denote the local processing ability of vehicle, in terms of CPU cycles/s. According to task model defined above, we can get the local computation delay:

$$L_q^l = \frac{C_q}{f_l} \tag{4}$$

2) MEC computation model: After task data migration between MEC nodes, the entire data of S_q is transmitted to the optimal MEC node M_o considering the migration delay and execution delay. We define the variable $\varepsilon = \{\varepsilon_q\}_Q$ as the optimal MEC nodes decision. If S_q is executed at M_o, we set $\varepsilon_q = o$, otherwise $\varepsilon_q = 0$, i.e.,

$$\varepsilon_q = \begin{cases} o, & \text{if } S_q \text{ is executed at MEC node } M_o. \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Let f_o denote the processing ability of MEC nodes M_o, in terms of CPU cycles/s. Then we can get the offloading computation delay of S_q:

$$L_q^o = \frac{C_q}{f_o} \tag{6}$$

E. Problem Formulation

In the following, we will analyze the completion time of tasks with various dependency relationships and present the corresponding formulation for the optimization objective.

1) No Predecessor Subtasks: Please note that during the transmission of S_q to the MEC nodes, the wireless device remains occupied and cannot transmit additional subtasks concurrently. Let $T_t^{l,a}$ denote the transmission available time slot. Then, the executing starting time of the MEC computation model can be obtained by

$$T_q^{e,s} = \max\left\{T_t^{l,a} + L_q^u, T_e^{o,a}\right\} \tag{7}$$

where L_q^u denotes the time latency of uploading task data of S_q and $T_e^{o,a}$ denotes the computation resource available slot of optimal computing MEC nodes.

For the local computation model, task S_q can be executed only if the vehicle's computing resource is available. let $T_e^{l,a}$ denote the computation resource available slot of the local vehicle. Then we can get the executing starting time:

$$T_q^{e,s} = T_e^{l,a} \tag{8}$$

2) With Predecessor Subtasks: In general, subtasks have at least one predecessor subtask. And the successor subtasks can not be executed until all the predecessor subtasks have been finished. For the MEC computation model, we can infer the execution starting time by comparing the transmitting completing time, the waiting time until the executing resource becomes available, and the execution finishing time of predecessor subtasks. Then we can get

$$T_q^{e,s} = \max\left\{T_t^{l,a} + L_q^u, T_e^{o,a}, T_{p'}^{e,c}\right\} \tag{9}$$

$T_{p'}^{e,c}$ denotes the maximum completion time of the preceding subtasks of S_q, which can be calculated as

$$T_{p'}^{e,c} = \max_{r_{pq}=1} \left\{T_p^{e,c}\right\} \tag{10}$$

where $T_p^{e,c}$ is the completion time of S_p. In accordance with the definition in (1), S_p refers to the predecessor subtasks of S_q, specifically indicated by $r_{pq} = 1$.

For the local computation model, there is no need to upload task data. Then we can get the executing starting time by comparing the waiting time until the executing resource is available, and the executing finishing time of predecessor subtasks.

$$T_q^{e,s} = \max\left\{T_e^{l,a}, T_{p'}^{e,c}\right\} \tag{11}$$

According to the definition of δ_q, a task can only be executed at one model, i.e., MEC computation model and local computation model which correspond to (9) and (11). So executing starting time of each subtask can be expressed as

$$T_q^{e,s} = \delta_q \max\left\{T_t^{l,a} + L_q^u, T_e^{o,a}, T_{p'}^{e,c}\right\} + (1-\delta_q) \max\left\{T_e^{l,a}, T_{p'}^{e,c}\right\} \tag{12}$$

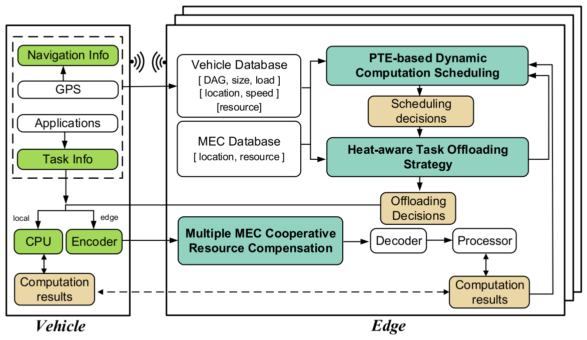
Navigation Info
GPS
Applications
Task Info
local
edge
CPU
Encoder
Computation results
Vehicle
Vehicle Database [DAG, size, load] [location, speed] [resource]
MEC Database [location, resource]
PTE-based Dynamic Computation Scheduling
Scheduling decisions
Heat-aware Task Offloading Strategy
Offloading Decisions
Multiple MEC Cooperative Resource Compensation
Decoder
Processor
Computation results
Edge

Fig. 4. System architecture of MCVCO.

The completion time of subtask S_q can be determined by adding the actual start time and the execution delay. Consequently, the total execution time of subtask S_q can be represented as

$$T_q^{e,c} = T_q^{e,s} + \delta_q L_q^o + (1-\delta_q) L_q^l \tag{13}$$

The overall completion delay of task can be expressed as

$$T_c = \max\{T_q^{e,c} \mid \forall 1 \leqslant q \leqslant Q\} \tag{14}$$

Therefore, we formulate the joint optimization problem, which aims to minimize maximal task completion delay, i.e.,

$$\begin{aligned} P1: \quad & \min_{\delta,\varepsilon} T_c \\ s.t. \quad & C_1: r_{q,q+1} \in R, 0 \leqslant q \leqslant Q-1 \\ & C_2: T_q^{e,c} \leqslant T_q \\ & C_3: \delta_q \in \{0,1\} \\ & C_4: \varepsilon_q \in \{0,1 \cdots Q\} \end{aligned} \tag{15}$$

where δ, ε are the offloading strategy set of S_q, the optimal MEC node strategy set of S_q respectively. C1 guarantees relationship dependency between the subtasks. C2 is the latency limitation of the subtasks. C3 represents that a task can only be executed by single computation model. C4 indicates that a task could be offloaded at an arbitrary optimal MEC node.

IV. Design of MCVCO

We design MCVCO, a Multi-MEC Cooperative Vehicular Computation Offloading system for enhancing computation offloading performance of the MEC-assisted vehicular architecture.

The main idea of MCVCO involves three steps: 1) proposing an effective method to estimate the communication quality for task offloading strategy; 2) cooperatively collecting the offloading task data; and 3) further dynamic task scheduling given the different data collecting ratio and limited transmission and computation resources. Fig. 4 depicts the architecture and the workflow of MCVCO in the MEC-assisted vehicular network. First, MCVCO initiates the execution of the dynamic task scheduling module to obtain scheduling decisions based on the current task execution status and the task dependencies. Subsequently, the offloading decision module determines the appropriate task offloading strategy by considering the vehicular and MEC position information. Based on these offloading decisions, the vehicle either executes the subtask locally or offloads it to the edge. In the case of offloading, the vehicle broadcasts the encoded task data for the edge layer to receive, followed by the execution of the cooperative resource compensation module to collect the task data. In the following sections, we will provide a detailed introduction to these algorithms.

A. *Heat-Aware Task Offloading Strategy*

For traditional data uploading strategies, the vehicle communicates with a single wireless interface. However, the limited coverage region of individual interface and high mobility of vehicles tend to result in unexpected task failure and interruption of data uploading. In this article, we apply the broadcast scheme to upload encoded data packets to MEC servers. In this case, the task could be collected by multiple wireless interfaces. The higher density of wireless infrastructures, the greater probability of successfully collecting task data. In addition, the velocity of the vehicle greatly affects the link duration between the vehicle and wireless interfaces. So we need to consider the mobility characteristics to decide on the offloading strategy.

We define the heat value as the Weighted Alive Link Duration (WALD) between the vehicle and the infrastructures, which integrates the vehicle's mobility and the distribution of wireless infrastructures. To exhibit the V2I communication scene more clearly, Fig. 5 shows the heat maps of WALD under different MEC node distribution scenarios.

The positions of Vehicle V and MEC M_i are (V^x, V^y) and (M_i^x, M_i^y). The velocity of V is (v_x, v_y). We first define the relative distance D_i between V and M_i at a certain moment t as

$$\begin{aligned} D_i(t) &= [V^x - M_i^x]^2 + [V^y - M_i^y]^2 \\ &= |\triangle \mathbf{dis}|^2 t^2 + 2(\triangle \mathbf{dis} \cdot \triangle \mathbf{vel})t + |\triangle \mathbf{vel}|^2 \end{aligned} \tag{16}$$

where

$$\begin{cases} \triangle \mathbf{dis} = (V^x - M_i^x, V^y - M_i^y) \\ \triangle \mathbf{vel} = (v_x, v_y) \end{cases} \tag{17}$$

If $D_i(t)^2 \leqslant R$, M_i is within the communication range of vehicle, and V could communicate with M_i. Then we can get the link duration LD_i between V and M_i [32]

$$LD_i = \begin{cases} 0, \text{ if } |\triangle \mathbf{dis}| = R, \cos(\triangle \mathbf{vel}, \triangle \mathbf{vel}) \geqslant 0 \\ \infty, \text{ if } |\triangle \mathbf{dis}| \leqslant R, \triangle \mathbf{vel} = 0 \\ \frac{-b+\sqrt{b^2-4\times a\times c}}{2\times a}, \quad \text{otherwise} \end{cases} \tag{18}$$

where

$$\begin{cases} a = |\triangle \mathbf{vel}|^2 \\ b = 2 \times \triangle \mathbf{vel} \cdot \triangle \mathbf{dis} \\ c = |\triangle \mathbf{dis}|^2 - R^2 \\ \cos(\triangle \mathbf{dis}, \triangle \mathbf{vel}) = \frac{(\triangle \mathbf{dis}, \triangle \mathbf{vel})}{|\triangle \mathbf{dis}| \times |\triangle \mathbf{vel}|} \end{cases} \tag{19}$$

During the communication between V and M_i, the velocity of V obey Gaussian distribution represented as $v \sim N(\mu, \sigma^2)$. The link reliability P_i denotes the probability that the connection

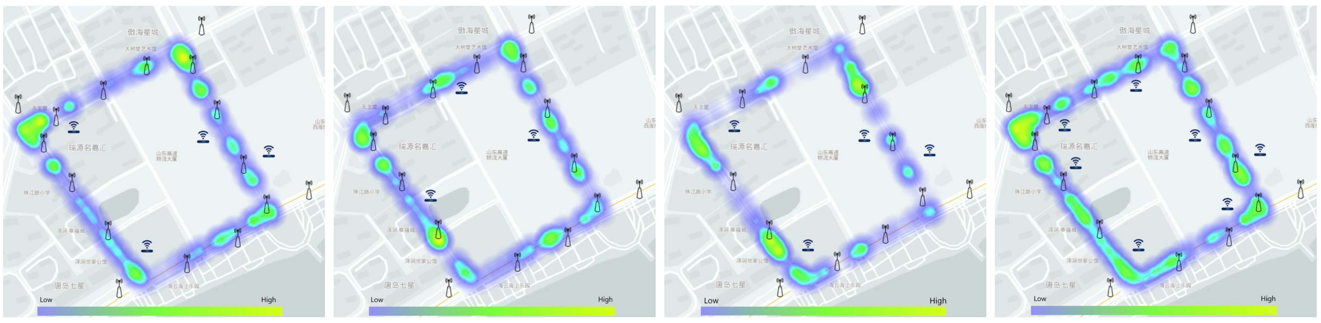
(a) Base scenario (b) Different distribution (c) Less interfaces (d) More interfaces

Fig. 5. Heat map of WALD under different MEC distribution scenes when the vehicular velocity obey Gaussian distribution (15, 10) m/s in clockwise direction.

between V and M_i remains active, as represented by (20).

$$P_i = \begin{cases} \int_t^{t+LD_i} f(\tau)d\tau, \text{if } LD_i > 0 \\ 0, \text{otherwise} \end{cases} \tag{20}$$

where

$$f(\tau) = \frac{2R}{\sqrt{2\pi}\sigma\tau^2} e^{-\frac{(\frac{2R}{\tau}-\mu)^2}{2\sigma^2}}, \text{if } \tau \geqslant 0 \tag{21}$$

Then the Alive Link Duration (ALD) of the link between Vehicle V and MEC M_i can denote by (18) and (20).

$$ALD_i = P_i \times LD_i \tag{22}$$

A higher value of P indicates a stronger link quality indicator between the vehicle and the wireless interfaces, resulting in an increased likelihood of successful offloading of data packets. In this article, we demonstrate the decision attitude based on the utility by the weighted mean of ALD which task P_i as the weighted parameter.

$$WALD = \sum_{i=1}^{I} \left\{ ALD_i \left(\frac{P_i}{P_{sum}} \right) \right\} \tag{23}$$

where P_{sum} is the sum of link reliability. Then we can get available upload data AUD based on WALD

$$AUD = WALD \times R_a \tag{24}$$

where R_a is the average data uploading rate. Moreover, we consider the latency of each task under offloading decisions should be no more than the local latency. Then we define the estimated offloading cost EC_o and estimated local cost EC_l as

$$EC_o = \frac{D_q}{R_a} + \frac{C_q}{f_a} \tag{25}$$

$$EC_l = \frac{C_q}{f_l} + L_l^a \tag{26}$$

where f_a is the the average processing ability, and L_l^a is the waiting delay of local processor available.

Based on the analysis above, we propose a heat-aware task offloading algorithm, i.e., Algorithm 1, to determine the offloading strategy. The time complexity is $O(I)$ as it involves calculating the alive link duration of MEC nodes in the communication range, assuming that there are I MEC nodes in the communication range.

Algorithm 1: Heat Aware Task Offloading Strategy.

```
input : The attributes of MEC positions, $M^x, M^y, I$;
        the attributes of vehicle, $V^x, V^y, v_x, v_y, R$,
        $\mu, \sigma$;
        the attributes of task data, $D_q, C_q$.
output: The offloading decision, $\delta_q$.
1  Set $P_{sum}$, $WALD = 0$;
2  Initiate $EC_o$, $EC_l$;
3  for $i$ to $I$ do
4      Calculate $LD_i$ based on Equation (18);
5      Calculate $P_i$ based on Equation (20);
6      Calculate $ALD_i$ based on Equation (22);
7      Update $P_{sum}$ = $P_{sum}$ + $P_i$;
8  end
9  Calculate WALD based on Equation(23);
10 Calculate AUD based on Equation(24);
11 Calculate $EC_o$ and $EC_l$ based on Equation(25 (26);
12 if $EC_o \leqslant EC_l$ then
13     if $AUD \geqslant \alpha D_q$ then
14         $\delta_q = 1$;
15     else
16         $\delta_q = 0$;
17     end
18 else
19     $\delta = 0$
20 end
21 return $\delta_q$
```

B. Multi-MEC Cooperative Resource Compensation Based on Fountain Code

Due to the rateless feature of the fountain code, the data receiver only concerns about whether it can collect enough encoded packets to recover source data. Based on this, we depend on cooperative resource compensation to improve the performance of task uploading in the MEC-assisted heterogeneous networks. On account of the randomness of degrees and selection of source packets, we can ignore same encoded packet. Moreover, the data receiver should get the adjacency relationship which is the degree information between the source data packet and encoded packet. This can be achieved by placing degrees d and adjacencies ad in the header of the encoded packet.

At first, the source task data S_q is separated into several equal parts. Then each part is divided into K pieces i.e., source packets. In addition, an integer d is chosen from a given probability mass function. Here is the Robust Soliton distribution of our work:

$$\eta(d) = \frac{\rho(d) + \tau(d)}{\sum_{d=1}^{k} \rho(d) + \tau(d)} \tag{27}$$

and $\tau(d)$ and $\rho(d)$ are given by (28) and (29).

$$\tau(d) = \begin{cases} \frac{R}{dk}, & d = 1, 2, \ldots \frac{k}{R} - 1 \\ \frac{R}{k} \ln \frac{R}{\xi}, & d = \frac{k}{R} \\ 0, & d = \frac{k}{R} + 1, \ldots, k \end{cases} \tag{28}$$

$$\rho(d) = \begin{cases} \frac{1}{k}, & d = 1 \\ \frac{1}{d(d-1)}, & d = 2, 3 \ldots, k \end{cases} \tag{29}$$

where $R = c \ln(\frac{k}{\xi})\sqrt{k}$, and c is a constant,and ξ is the probability of decoding failure.

According to the above analysis, we propose a multiple MEC cooperative resource compensation algorithm for the task data uploading. The basic idea is summarized as follows:

1) Encode the source data: The vehicular encoder generates the data packets E according to degree distribution function $\eta(d)$ and XOR operation. And insert the degree d and adjacency ad into the packets header.
2) Determine the packets center MEC server: According to the (18) and (20), the MEC layer selects the highest available link duration MEC node M_t.
3) Packets Transmission: The OBU broadcasts encoded task packets to MEC interfaces.
4) Packets collision detection and resource compensation: For each packet received by MEC servers, the degree and adjacency are collected automatically. Then the information is transmitted to M_t to detect whether there is a collision or not. If yes, the packet is redundant and should be discarded. Otherwise, transmit the packet to M_t to the decoding phase. Repeat the above procedures until recovering the source packets.

To facilitate the understanding of the proposed strategy, we show an example in Fig. 6. Supposing the source data packets $\{S1, S2, \ldots, S10\}$ generate Fountain Code encoded packets $\{E1, E2, \ldots, E12, E13\}$. Due to the mobility of the vehicle and fluctuating wireless network, RSU1, RSU2 and AP receives $\{E1, E2, E3, E5, E7\}$, $\{E4, E5, E7, E9, E10, E12\}$ and $\{E1, E4, E8, E9, E12, E13\}$, respectively. $E6$ and $E11$ are not collected by any MEC and go astray. After packet collision detection and resource compensation, the cooperative system collects $\{E1, E2, E3, E4, E5, E7, E8, E9, E10, E12, E13\}$ which could recover the source packets. It can be seen from the figure that the fountain code-based resource compensation method can deal with packets loss effectively and improve the overall task-uploading performance.

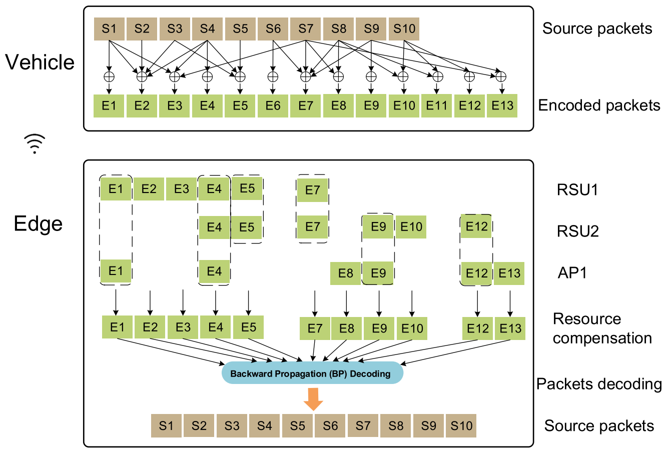
Vehicle
Source packets
Encoded packets
S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
E1 E2 E3 E4 E5 E6 E7 E8 E9 E10 E11 E12 E13
Edge
RSU1
RSU2
AP1
Resource compensation
Packets decoding
Backward Propagation (BP) Decoding
Source packets

Fig. 6. Cooperative resource compensation based on fountain code.

Algorithm 2: Multiple MEC Cooperative Resource Compensation.

```
input : The source task data $S_q$;
        the distribution function, $\eta(d)$;
output: The source task data $S_q$;
1  Encode $S_q$ based on Equation(27),(28),(29) and XOR
     operation;
2  Insert the degree $d$ and adjacencies $ad$ into encoded
     packets header;
3  for $i$ to $I$ do
4      Calculate $LD_i$ based on Equation (18);
5      Calculate $P_i$ based on Equation (20);
6      Calculate alive link duration $LD_i \times P_i$.
7  end
8  Select the MEC nodes $M_t$ with highest alive link
     duration value as the packets switching center.
9  The OBU broadcasts the data packets;
10 while The source data is not recovered do
11     for $j = 1$ to $M$ do
12         foreach Encoded packet $E_x$ received by $M_j$ do
13             Check $d$ and $ad$ with the $M_t$.
14             if No packet collision then
15                 Transmit the packet $E_x$ to $M_t$.
16             else
17                 Discard $E_x$.
18             end
19         end
20     end
21     $M_t$ decode source data with Backward propagation
         (BP) decoding algorithm.
22 end
23 return $S_q$.
```

The proposed multiple MEC cooperative resource compensation algorithm is described in Algorithm 2. The time complexity mainly depends on the fountain encoding and data recovering phrases. The encoding algorithm in Algorithm 2 exhibits a time complexity of $O(K * \ln K)$, while the data recovery process has

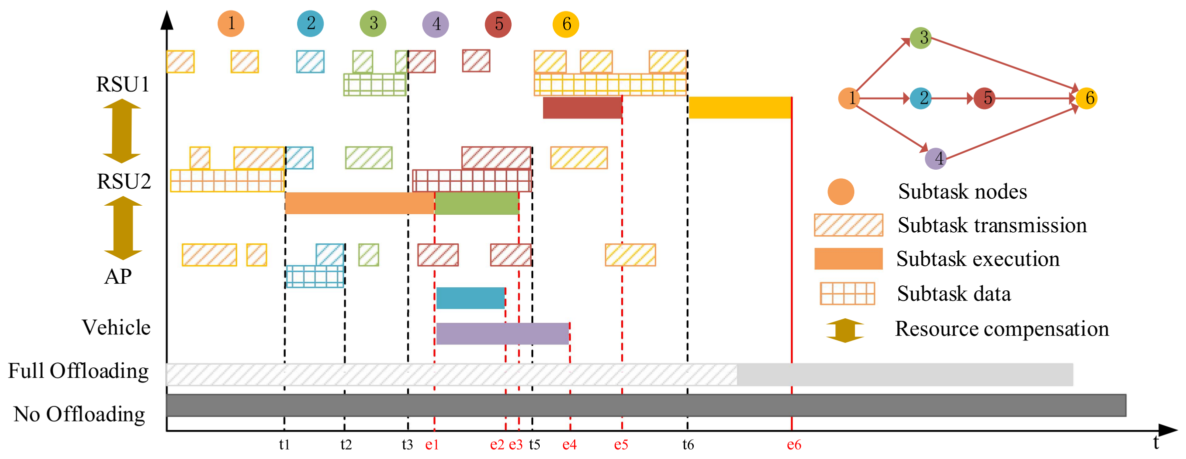

Fig. 7. PTE-based task transmission and execution procedures of MCVCO.

a time complexity of $O(K * M)$. Here, K denotes the number of source task data packets, and M represents the number of MEC nodes receiving the task data. Due to the much greater value of K compared to M, the overall time complexity of Algorithm 2 can be expressed as $O(K * \ln K)$.

C. PTE-Based Dynamic Computation Scheduling

In this work, based on Parallel Transmission and Execution (PTE) scheme, we propose a PTE-based dynamic computation scheduling algorithm, which maintains the simplicity and efficiency of DAG scheduling in distributed computing.

As an example, we plot the PTE procedures considering six subtasks and three MEC nodes scenario with cooperative data compensation in Fig. 7. Specifically, t_q denotes the transmission complete time and e_q denotes the execution complete time of subtask node S_q. The processing order of subtasks follows the numerical order shown in the figure. Next, we take subtask S_2 as an example to demonstrate the execution process. The transmission procedure of S_2 starts until the transmission of S_1 is finished according to the $N/N/1$ model. At t_2, the MEC nodes receive enough data packets and transmit the task data to AP nodes to execute. However, since S_1 is the predecessor subtask of S_2, the execution of S_2 has to wait until the execution of S_1 is completed. Then, S_2 is executed by the selected edge server. Based on the findings illustrated in Fig. 7, it is evident that the resource compensation and PTE scheme exhibit enhanced task offloading performance by effectively utilizing transmission and computation resources. This improvement surpasses that achieved through full offloading or local computing approaches.

According to data uploading status and processing status of subtask, we first classify the subtasks into two categories, i.e., prepared stage subtasks and unprepared stage subtasks. Prepared stage refers to the tasks which have been uploaded successfully and all the predecessor subtasks have finished or no predecessor subtasks; unprepared stage refers to some predecessor subtasks under processing or under uploading. Let ϕ_p and ϕ_u denote the set of prepared stage subtasks and unprepared stage subtasks.

The dynamic nature of available resources can evidently affect the status of subtasks. While striving to minimize the completion time of a specific subtask, it is crucial to acknowledge that this may lead to timeouts in other concurrent subtasks, ultimately resulting in deteriorated overall task performance. Hence, we need to give overall consideration to the subtasks and quantify the execution priority of all subtasks. This study investigates the primary factors that affect task completion time, including the dependency relationships among subtasks, execution time on the server, and transmission time among servers.

Let P_q^t denote the transmitting cost of S_q among MEC servers, and the data size is D_q, then P_q^t can be defined as

$$P_q^t = \frac{D_q}{\bar{t}} \tag{30}$$

where $\bar{t}$ is the average transmission rate between MEC servers.

Then, we define the priority factor of executing delay P_q^e as

$$P_q^e = \frac{C_q}{\bar{f}} \tag{31}$$

where $\bar{f}$ is the average service rate of MEC servers.

Moreover, we have to make sure that the subtasks are in correct order and avoid reversing the dependency relationships. Hence, we differ the priority level by adopting a recursive scheduling scheme and define the priority of S_q as

$$P_q = \max_{r_{qk}=1} \left\{ \left(P_q^t + P_q^e \right) + P_k \right\} \tag{32}$$

In accordance with the definition in (1), S_k refers to the successor subtasks of S_q, specifically indicated by $r_{qk} = 1$.

After determining the status set and priority of each subtask, we design a greedy method-based computation assigning strategy which takes account of executing delay and data migration delay. Let T_q^e denote the total time delay of S_q executing at M_e, which can be calculated by

$$T_q^e = \frac{D_q}{r_t} + \frac{C_q}{f_e} \tag{33}$$

where r_t is the transmitting rate of M_t which recover the encoded task data. Subsequently, we are able to select the MEC node with the minimum transmission and execution time to execute tasks.

Based on the above analysis, the PTE-based dynamic computation scheduling is shown as Algorithm 3. The complexity

Algorithm 3: PTE-based dynamic computation scheduling algorithm.

```
input  : The workload $C_q$ and data size $D_q$ of each
         subtask;the DAG of task data, $G(S, R)$;the
         attributes of each MEC, $f, r$
output: The execution decisions , $\{\varepsilon_q\}$;
1  Initiate $\phi_p$, $\phi_u$;
2  for $i = 1$ to $Q$ do
3      Calculate $P_q^t$ based on Equation (30);
4      Calculate $P_q^e$ based on Equation (31);
5      for $j = 1$ to $Q$ do
6          Calculate $P_q$ based on Equation (32);
7      end
8  end
9  Sort the $P_q$ sequence in non-decreasing order P;
10 while Task is not completed do
11     foreach Prepared subtask in $\phi_p$ do
12         Select the highest priority subtask $S_h$;
13         for $k = 1$ to $I$ do
14             Calculate total time delay based on
                 Equation (33);
15         end
16         Determine execution decision $\varepsilon_h$;
17         if $\varepsilon_h$ != $t$ then
18             Transmit the source data from $M_t$ to $M_h$.
19         else
20             Execute subtask in $M_t$.
21         end
22     end
23     Update $\phi_p$ and $\phi_u$;
24 end
25 return $\{\varepsilon_q\}$.
```

depends on the number of subtasks Q, and the priority factor of each subtask and predecessor subtask needs to be calculated. Therefore, the time complexity of Algorithm 3 is $O(Q^2)$.

V. Performance Evaluation and Analysis

A. Simulation Setup

The road map for the simulations was obtained from OpenStreetMap. A simulated area of 1500 * 1500 meters in Qingdao was chosen, as illustrated in Fig. 8(a). We represent the road sections as a grid environment, as shown in Fig. 8(b). Our simulations are implemented by Matlab, and generated the vehicle motility features using the simulation tool (Matlab Driving Scenario Designer). Furthermore, we conduct a comparative analysis of different types of typical vehicular tasks with distinct dependency relationships, encompassing chain query, video processing, and data analysis tasks. Fig. 9 describes the typical serial relationship (chain query task) or highly parallel relationship (the VR/AR processing and the complex data analysis tasks). The computational capabilities of the vehicle server, and edge server are 1 GHz and 10–50 GHz. The default size and complexity of the subtasks are 200 KB and 1000 CPU cycles/bit,

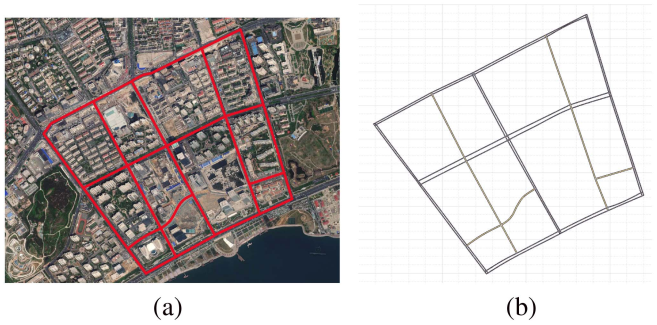

Fig. 8. Simulation map. (a) The red lines are the roads segments for simulations. (b) Simulation roads abstracted from real world maps.

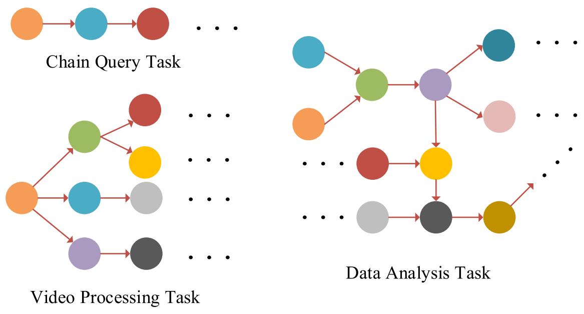

Fig. 9. DAG of representative tasks.

respectively. We assume a commonly applied Poisson process for the arrival of tasks, allowing us to explore different task densities by varying the mean number of vehicular tasks. Note that these parameters are adopted by [27] and [28].

Besides, we compare MCVCO schemes with the following four base lines, namely, NO, CO, FO, PCO. The details of the base lines are described as follows.

- NO: No Offloading. The tasks are executed by vehicle only. Compared to MCVCO, the NO does not require consideration of wireless network fluctuations and system resources scheduling. But NO encounters the challenges related to inadequate computation resources.
- CO: Coarse-grained Offloading. In this approach, the tasks are executed in a coarse-grained manner by MEC servers only. Compared to MCVCO, the CO scheme upload the task data utilizing a single link model and execute tasks on a specific MEC node. Therefore, CO encounters challenges related to poor transmission efficiency and inefficient resource scheduling.
- FO: Fine-grained Offloading. The tasks are executed in a fine-grained manner by both MEC servers and local servers. Compared to MCVCO, the FO scheme offloads the task utilizing a single link model. Consequently, The FO scheme encounters challenges related to poor transmission efficiency.
- PCO: Probabilistic Computation Offloading [28]. In this baseline, PCO minimizes the task completion delay of tasks via task uploading coordination, horizontal cooperation between MEC layer servers, and vertical cooperation between cloud and other layers.

To compare the effectiveness of the offloading scheme, we collect the following statistics: the total number of tasks and the number of tasks executed within the deadline, denoted by Q^{total} and $Q^{success}$; the submission time and finish time of each task, denoted by T_i^{start} and T_i^{finish}; upload time and upload data of each task, denoted by T_i^{upload} and D_i^{upload}. On this basis, we evaluate MCVCO and other schemes using the following metrics:

1) Average success ratio (ASR): It indicates the average percentage of the number of computation tasks executed within the deadline in each episode, which is calculated as

$$ASR = \frac{Q^{success}}{Q^{total}} \tag{34}$$

The higher value of ASR represents that vehicle receives more useful computation results, which brings a higher QoS.

2) Average completion time (ACT): This metric represents the average completion delay of successfully offloaded tasks, measuring the time it takes from their offloading at the MEC server to the computation result back to the vehicle.

$$ACT = \frac{\sum_{i=1}^{Q^{total}} \left(T_i^{finish} - T_i^{start}\right)}{Q^{total}} \tag{35}$$

The lower value of ACT indicates that the task spends less time getting the result.

3) Average uploading quality (AUQ): This metric is defined as the average value of the ratio of task uploading data and task uploading time of all the successful tasks, which is calculated as

$$AUQ = \frac{\sum_{i=1}^{Q^{success}} \frac{D_i^{upload}}{T_i^{upload}}}{Q^{success}} \tag{36}$$

The higher value of AUQ indicates that the lower the impact of wireless network on the vehicle, the higher the data transmission quality.

B. Simulation Results and Analysis

1) Impact of Vehicle's Velocity: In the simulations, we set the MEC nodes number, subtask number, and bandwidth to 50, 20, and 10 MHz, respectively. We first compare the impact of the velocity of the vehicle on ASR, as shown in Fig. 10(a). The simulation results reveal that the ASR of three typical offloading algorithms decreased as the velocity increases. This phenomenon occurs due to the increased velocity, which in turn results in more frequent handoffs between different edge nodes and a higher rate of offloading packet loss. As a consequence, there is a significant degradation in performance. Nevertheless, the MCVCO exhibits limited sensitivity to variations in velocity, as evidenced by the nearly constant curves depicted in the figure. Compared with the typical offloading and local method, the MCVCO considers the vehicle's mobility characteristics and distributions of MEC nodes in offloading strategy decision-making. In this case, our proposed scheme could effectively eliminate the negative impact of speed on data offloading.

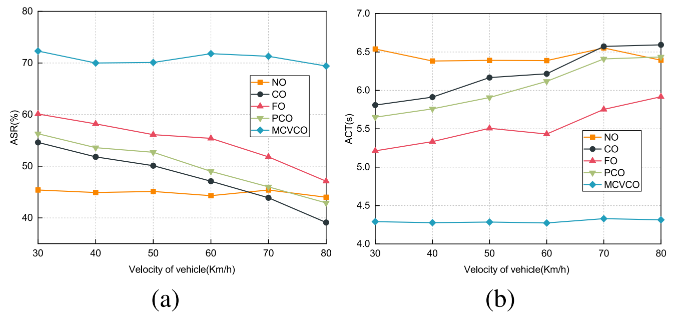

Fig. 10. Impact of different velocity on system performance. (a) ASR versus velocity. (b) ACT versus velocity.

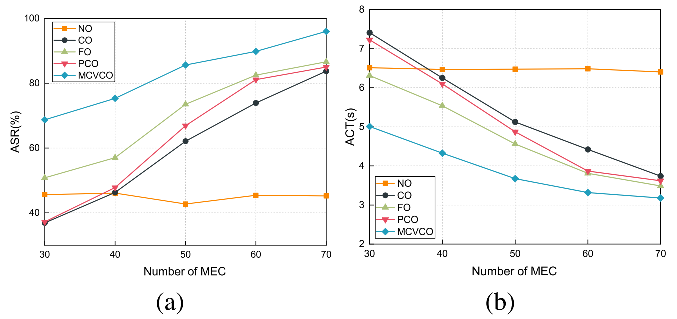

Fig. 11. Impact of number of MEC on system performance. (a) ASR versus number of MEC. (b) ACT versus number of MEC.

To illustrate the better performance of MCVCO at high-speed scenes, we further analyze the ACT in Fig. 10(b). From the figure, we can see that MCVCO performs the best among all schemes and performs better in high speed. By comparison, the ACT of the others three offloading schemes is low when the velocity is low. Considering the limited occurrence of handoffs and data migration between MEC nodes in this case, as well as potential interference from the wireless environment, the impact of vehicle movement on the overall performance is negligible. As the velocity of vehicles increases, the ACT of typical offloading schemes skyrockets while the MCVCO's ACT stays around 4.3 s, saving 27–51%. MCVCO also outperforms NO by 49%, which highlights the advantage of PTE-based scheduling. Next, we show the AUQ in Fig. 12(a). The overwhelming majority of MCVCO's AUQ exceeds 2 Mbit/s, and even in the worst-case scenario, efficient uploading of task data can still be achieved through resource compensation.

2) Impact of MEC Nodes Amount: In this group of simulations, we set the average velocity, subtask number, and bandwidth to 60 Km/h, 20, and 10 MHz, respectively. Fig. 11(a) and (b) depict the variation of ASR and ACT of five schemes under the different numbers of MEC nodes. From the figures, we can find that MCVCO works well under different MEC distributions. As the number of MEC nodes increases from 30 to 70, the available computation and transmission resources total resources become sufficient. Thus, the ASR of four computation

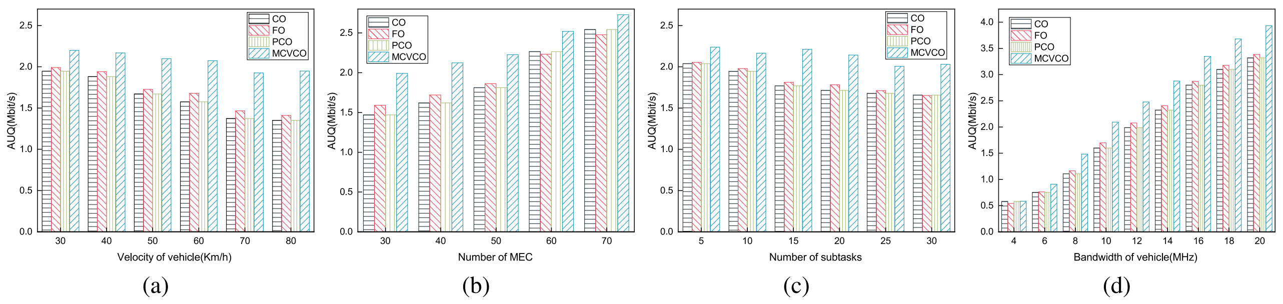

Fig. 12. Impact of different velocity on system performance. (a) AUQ versus velocity. (b) AUQ versus number of MEC. (c) AUQ versus number of subtasks. (d) AUQ versus bandwidth.

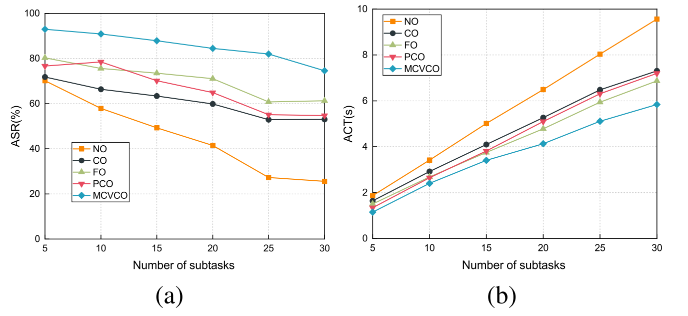

Fig. 13. Impact of number of subtasks on system performance. (a) ASR versus number of subtasks. (b) ACT versus number of subtasks.

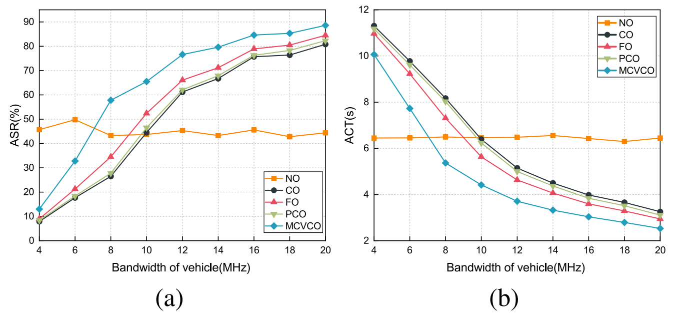

Fig. 14. Impact of number of bandwidth of vehicle on system performance. (a) ASR versus bandwidth. (b) ACT versus bandwidth.

offloading algorithms increases and reaches a high level eventually. Nonetheless, conventional computation offloading schemes exhibit subpar performance in scenarios characterized by low deployment density, performing even worse than NO when the number of MEC nodes is limited to 30. The application of the resource compensation algorithm in MCVCO results in superior performance compared to other schemes, particularly in sparse MEC deployment circumstances. MCVCO outperforms other schemes by a significant margin of 39%. However, as the number of edge servers increases, the advantage of MCVCO diminishes. In dense deployment conditions, the ACT of all offloading schemes is shorter, whereas performance optimization quantity is reduced when the amount of MEC nodes increases. The reason is that the resources assigned to the offloaded task are adequate for transmitting and executing. In addition, as shown in Fig. 12(b), the AUQ also reaches saturation as the MEC nodes increase. Hence, we can deduce that the resources are no longer the factor affecting the delay, but the scheduling algorithms become the biggest influencing factors. Besides, due to economic and geographic, it is impossible to deploy MEC nodes at such a high density in the real world.

3) Impact of the Number of Subtasks: In the group of simulations, we set the average velocity, the number of MEC nodes, and bandwidth to 60 Km/h, 50, and 10 MHz, respectively. We adjust the mean number of subtasks to compare the ASR and ACT in different offloading schemes. Fig. 13 depicts the result with an increasing mean number of subtasks. Fig. 13(a) and (b) show the NO scheme gets the lowest ASR and longest ACT among the four schemes. This is due to the insufficient computational capacity of vehicles to handle these computation-intensive tasks. Compared with the NO, the computation-intensive tasks can be offloaded to the edge layer and executed by edge servers. Therefore, the ASR of the four offloading go up. Moreover, as shown in 12(c), the AUQ of four offloading schemes decrease when the number of tasks increases. This argument is sound as the number of subtasks increases, leading to a greater allocation of wireless and computational resources. Consequently, this combined increase in resource demands prolongs the ACT.

In conclusion, we can find that:1) MCVCO consistently outperforms typical offloading schemes across different numbers of subtasks, improving the ASR by as large as 28%, and 2) MCVCO performs far better (up to 32% margin) as the subtasks number increases. With the assistance of PTE-based scheduling and data compensation, MCVCO provides the better performance. Further considering data compensation so that MEC nodes can help each other by Fountain code, MCVCO provides the best performance.

4) Impact of Bandwidth: In this group of simulations, we set the average velocity, the number of MEC nodes, and the number of subtasks to 60 Km/h, 50, and 20, respectively. In theory, the ASR should exhibit an increase in performance when the vehicle's bandwidth improves. This is attributed to the positive utility gained from the enhanced transmitting capacity in offloading schemes. This inference is supported by the observations presented in Fig. 14. Then, it is more meaningful to care about the AUQ with the increase of bandwidth, which can measure the system capacity. It can be seen intuitively from Fig. 12(d) that the AUQ increases as the bandwidth increases. In addition, it is noteworthy that the performance gap between the MCVCO and typical offloading schemes decreases with the bandwidth increases. In more detail, the AUQ is only 0.2 higher than that of other offloading schemes when the bandwidth is 6 MHz, while the MCVCO increases the AUQ by 0.7 when the bandwidth is 18 MHz. It demonstrates that the system overall performance can be improved by cooperative resource compensation of the MEC nodes.

VI. Conclusion

In this article, we focus on the vehicular computation offloading problem in a MEC-assisted system and designed a Multi-MEC Cooperative Vehicular Computation Offloading scheme. Based on extensive simulations and result analyses, we demonstrate the effectiveness of MCVCO under different vehicular velocities, various numbers of MEC nodes, the number of subtasks, and bandwidth compared with other baseline methods. However, MCVCO is a combination of ad-hoc based searching protocols, and is facing the challenges of integrated management. In our future work, we will try to solve this problem by using digital-twin approach to optimize the algorithms in the digital world. In addition, we will expand our research by adopting V2V collaborative task offloading and considering lightweight artificial intelligence algorithms to deal with the dynamic scheduling problem.

References

[1] C. Zhao, Y. Lv, J. Jin, Y. Tian, J. Wang, and F.-Y. Wang, “DeCAST in transverse for parallel intelligent transportation systems and smart cities: Three decades and beyond,” *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 6, pp. 6–17, Nov./Dec. 2022.

[2] L. Li et al., “Parallel testing of vehicle intelligence via virtual-real interaction,” *Sci. Robot.*, vol. 4, no. 28, 2019, Art. no. eaaw4106.

[3] L. Chen et al., “Milestones in autonomous driving and intelligent vehicles: Survey of surveys,” *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1046–1056, Feb. 2023.

[4] S. Colle, T. Mortier, P. Micallefc, M. Coltelli, A. Horstead, and M. Aveta, “Can utilities turn EVS into a grid asset?,” 2023. [Online]. Available: https://evision.eurelectric.org/event/2022/report/

[5] S. Ansari, F. Naghdy, and H. Du, “Human-machine shared driving: Challenges and future directions,” *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 499–519, Sep. 2022.

[6] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, “Mobile edge computing: A survey,” *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[7] H. Ke, J. Wang, L. Deng, Y. Ge, and H. Wang, “Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks,” *IEEE Trans. Veh. Technol.*, vol. 69, no. 7, pp. 7916–7929, Jul. 2020.

[8] J. Li et al., “Multiobjective oriented task scheduling in heterogeneous mobile edge computing networks,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8955–8966, Aug. 2022.

[9] Z. Li, G. Xiong, Z. Wei, Y. Lv, N. Anwar, and F.-Y. Wang, “A semisupervised end-to-end framework for transportation mode detection by using GPS-enabled sensing devices,” *IEEE Internet Things J.*, vol. 9, no. 10, pp. 7842–7852, May 2022.

[10] V. Huy Hoang, T. M. Ho, and L. B. Le, “Mobility-aware computation offloading in MEC-based vehicular wireless networks,” *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 466–469, Feb. 2020.

[11] S. Jahandar, L. Kouhalvandi, I. Shayea, M. Ergen, M. H. Azmi, and H. Mohamad, “Mobility-aware offloading decision for multi-access edge computing in 5G networks,” *Sensors*, vol. 22, no. 7, 2022, Art. no. 2692.

[12] K. Guo, M. Sheng, T. Q. S. Quek, and Z. Qiu, “Task offloading and scheduling in fog RAN: A parallel communication and computation perspective,” *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 215–218, Feb. 2020.

[13] P. Ren, X. Qiao, Y. Huang, L. Liu, C. Pu, and S. Dustdar, “Fine-grained elastic partitioning for distributed DNN towards mobile web AR services in the 5G era,” *IEEE Trans. Serv. Comput.*, vol. 15, no. 6, pp. 3260–3274, Nov./Dec. 2022.

[14] Y. Ding, C. Liu, X. Zhou, Z. Liu, and Z. Tang, “A code-oriented partitioning computation offloading strategy for multiple users and multiple mobile edge computing servers,” *IEEE Trans. Ind. Inform.*, vol. 16, no. 7, pp. 4800–4810, Jul. 2020.

[15] L. He, M. Wen, Y. Chen, M. Yan, and B. Jiao, “Delay aware secure offloading for NOMA-assisted mobile edge computing in Internet of Vehicles,” *IEEE Trans. Commun.*, vol. 70, no. 8, pp. 5271–5284, Aug. 2022.

[16] H. Wang, J. Xie, and M. M. A. Muslam, “Fair: Towards impartial resource allocation for intelligent vehicles with automotive edge computing,” *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1971–1982, Feb. 2023.

[17] W. Yao et al., “Evolutionary utility prediction matrix-based mission planning for unmanned aerial vehicles in complex urban environments,” *IEEE Trans. Intell. Veh.*, vol. 8, no. 2, pp. 1068–1080, Feb. 2023.

[18] P. Zhang, C. Wang, C. Jiang, and A. Benslimane, “UAV-assisted multi-access edge computing: Technologies and challenges,” *IEEE Internet Things Mag.*, vol. 4, no. 4, pp. 12–17, Dec. 2021.

[19] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, and Z. Liu, “A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3664–3674, Jun. 2021.

[20] H. Liao, X. Li, D. Guo, W. Kang, and J. Li, “Dependency-aware application assigning and scheduling in edge computing,” *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4451–4463, Mar. 2022.

[21] X.-Q. Pham, T. Huynh-The, E.-N. Huh, and D.-S. Kim, “Partial computation offloading in parked vehicle-assisted multi-access edge computing: A game-theoretic approach,” *IEEE Trans. Veh. Technol.*, vol. 71, no. 9, pp. 10220–10225, Sep. 2022.

[22] F.-Y. Wang, X. Wang, L. Li, and L. Li, “Steps toward parallel intelligence,” *IEEE/CAA J. Autom. Sinica*, vol. 3, no. 4, pp. 345–348, Oct. 2016.

[23] X. Li, K. Wang, Y. Tian, L. Yan, F. Deng, and F.-Y. Wang, “The paralleleye dataset: A large collection of virtual images for traffic vision research,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2072–2084, Jun. 2019.

[24] X. Li, P. Ye, J. Li, Z. Liu, L. Cao, and F.-Y. Wang, “From features engineering to scenarios engineering for trustworthy AI: I&I, C&C, and V&V,” *IEEE Intell. Syst.*, vol. 37, no. 4, pp. 18–26, Jul/Aug. 2022.

[25] C. Li et al., “Dynamic offloading for multiuser muti-CAP MEC networks: A deep reinforcement learning approach,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2922–2927, Mar. 2021.

[26] W. Zhan et al., “Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing,” *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[27] P. Dai, K. Hu, X. Wu, H. Xing, and Z. Yu, “Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks,” in *Proc IEEE INFOCOM*, 2021, pp. 1–10.

[28] P. Dai, K. Hu, X. Wu, H. Xing, F. Teng, and Z. Yu, “A probabilistic approach for cooperative computation offloading in MEC-assisted vehicular networks,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 899–911, Feb. 2022.

[29] B. Zhu, K. Chi, J. Liu, K. Yu, and S. Mumtaz, “Efficient offloading for minimizing task computation delay of NOMA-based multiaccess edge computing,” *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3186–3203, May 2022.

[30] A. Mahmood, Y. Hong, M. K. Ehsan, and S. Mumtaz, “Optimal resource allocation and task segmentation in IoT enabled mobile edge cloud,” *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 13294–13303, Dec. 2021.

[31] 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.

[32] X. Zhang, X. Cao, L. Yan, and D. K. Sung, “A street-centric opportunistic routing protocol based on link correlation for Urban VANETs,” *IEEE Trans. Mobile Comput.*, vol. 15, no. 7, pp. 1586–1599, Jul. 2016.

Jianhang Liu received the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2013. He is currently an Associate Professor with the Computer Science Department, China University of Petroleum, Qingdao, China. His research interests include intelligent transport system and digital twin. He is the Member of CCF Network and Data Communications.

Kunlei Xue received B.Eng. degree in communication engineering from the China University of Petroleum, Qingdao, China, in 2021, where he is currently working toward the M.E. degree with the Department of Information and Communication Engineering. His research interest includes mobile edge computing and intelligent vehicular networking.

Qinghai Miao (Senior Member, IEEE) received the Ph.D. degree in control theory and control engineering from the Graduate University of Chinese Academy of Sciences, Beijing, China, in 2007. From 2017 to 2018, he was a Visiting Scholar with the School of Informatics, the University of Edinburgh, Edinburgh, U.K. He is currently an Associate Professor with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China. His research interests include parallel intelligence, machine learning, computer graphics, and intelligent transportation systems.

Shibao Li was born in 1978. He received the B.S. and M.S. degrees in computer science from the University of Petroleum, Qingdao, China, in 2002 and 2009, respectively. He is currently a Professor with the China University of Petroleum, Qingdao, China. His research interests include indoor localization technology, the Internet of Things, wireless networks, and mobile computing

Xuerong Cui received the master's degree in computer application technology from the China University of Petroleum, Qingdao, China, in 2003, and the Ph.D. degree in information science and engineering from the Ocean University of China, Qingdao, China, in 2012. From 2015 to 2016, he was a Visiting Scholar with the University of Victoria, Victoria, BC, Canada. His research interests include Big Data and artificial intelligence; and wireless positioning and navigation.

Danxin Wang received the B.S. degree in information security from Northeastern University, Shenyang, China, in 2015, and the Ph.D. degree in information security from Wuhan University, Wuhan, China, in 2022. She is currently a Lecturer with the Qingdao Institute of Software, College of Computer Science and Technology, China University of Petroleum (East China), Qingdao, China. Her research interests include data security and privacy, Cybersecurity for various IoT/IoVs applications, Blockchain, and federated learning.

Kewen Li received the Ph.D. degree in information management and information system from Tianjin University, Tianjin, China. He was a Visiting Scholar with Florida Atlantic University, Boca Raton, FL, USA. He is currently a Professor with the College of Computer and Communication Engineering, China University of Petroleum (East China), Qingdao, China. His research interests are in artificial intelligence, software engineering, data mining, and machine learning.