Received 8 September 2023, accepted 6 October 2023, date of publication 16 October 2023, date of current version 20 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3324718

RESEARCH ARTICLE

Federated Double Deep Q-Learning-Based Computation Offloading in Mobility-Aware Vehicle Clusters

WENHUI YE, KE ZHENG, YUANYU WANG, AND YULIANG TANG, (Member, IEEE)
School of Informatics, Xiamen University, Xiamen 361005, China

Corresponding author: Yuliang Tang (tyl@xmu.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61731012, Grant 61801409, and Grant 91638204.

ABSTRACT On the edge side of internet of vehicles (IoV), mobile edge computing (MEC) servers with certain computational resources are deployed to provide computational service for vehicles. However, with the generation of a series of vehicle computing intensive and delay sensitive applications such as virtual reality (VR) navigation and vehicle-mounted games, the quality of service (QoS) for vehicle applications cannot be guaranteed with the limited computational resources in MECs. In this article, adjacent vehicles in IoV are converged into clusters based on topological stability. The idle computational resources of vehicles within the cluster are integrated and scheduled uniformly to serve vehicles within the cluster with computation task requirements. This reduces the dynamic changes in idle computational resources caused by rapid vehicle movement, and effectively ensured reliable feedback of computing results after offloading computing tasks. In order to reduce the task completion time, the task is decomposed into multiple subtasks with time dependencies, which can be processed in parallel. A directed acyclic graph (DAG) is used to characterize the task model. To further minimize latency in completing computing tasks, a federated double deep Q network-based computation task offloading (FDTO) strategy is proposed for vehicle clusters. The optimal computation task offloading decision is obtained based on the collaborative training and updating of double deep Q network (DDQN). Unlike traditional machine learning algorithms that require centralized training, federated learning (FL) allows only the training model parameters to be passed between vehicles, which not only protects data privacy but also reduces communication overhead. Simulation results show that the proposed strategy can effectively reduce task completion latency compared to the benchmark strategies.

INDEX TERMS Internet of Vehicles, directed acyclic graph, DDQN, federated learning.

I. INTRODUCTION

With the development of intelligent vehicles, a large number of computing-intensive services will generate large amounts of communication data, such as artificial intelligence [1], augmented reality [2], and autonomous driving [3]. Lots of data are closely related to traffic safety and traffic efficiency, requiring lower computational processing latency and higher communication reliability, and therefore demanding sufficient computational resources for processing. How to use limited computational resources to meet the explosive growth of computational demand is one of the key issues to be studied in the internet of vehicles (IoV), which is also investigated in this paper. In response to the shortage of computational resources in IoV, many studies have addressed cloud computing as a solution to deal with computation-intensive services [4]. Utilizing cloud computing and cloud storage resources can greatly enhance the computing and storage capabilities of IoV, but it also brings larger latency, higher cost and energy consumption [5], [6]. In addition, the reliability of long-distance communication between vehicles and the central cloud is difficult to guarantee in the highly dynamic environment of IoV. To overcome the various limitations of cloud computing, mobile edge computing (MEC) has

The associate editor coordinating the review of this manuscript and approving it for publication was Usama Mir.

received a lot of attention. MEC is an extension of cloud computing, which can effectively solve the problems of communication latency and bandwidth limitation caused by remote cloud computing by deploying cloud services on the edge side of the network. Applying MEC to IoV effectively integrates the compute and storage resources of a large number of idle vehicle nodes as well as roadside units (RSUs) on the edge side of the network. Compared with cloud computing, MEC can effectively reduce communication latency, cost, and energy consumption, which greatly helps the IoV system handle computing-intensive and latency-sensitive computation tasks [7]. Computation task offloading where the vehicle offloads its computing task to RSUs for processing, is one of the key technologies for edge computing [8], [9]. Since RSUs are roadside fixed infrastructure with limited communication range, the communication link connection time between high-speed vehicles and RSUs is short. This poses a challenge for computing results to be sent back to the task vehicle. In addition, when offloading computing tasks to RSUs for processing, the limited computational resources of RSUs may not be sufficient to guarantee the quality of service (QoS) for certain latency-sensitive or computing-intensive services [10]. Therefore, to further improve the QoS, we think about other approaches besides taking MEC-assisted task offloading for IoV. With the increasing computational capacity of vehicles and the growing road traffic density [11], lots of idle in-vehicle resources are generated in IoV. For vehicles with relatively small speeds, the inter-vehicle communication link connection time is relatively longer, and by reasonably exploiting and scheduling these free in-vehicle resources, it is possible to effectively handle complex computing tasks, reduce task processing latency, and improve QoS [12]. The concept of vehicle cloud computing (VCC) was proposed in [13]. In VCC, resource vehicle nodes with free computational resources can be used as vehicle cloud servers to provide computation services to task vehicle nodes that have tasks to offload. In this paper, we assist the task offloading in IoV by integrating and scheduling the idle computational resources of vehicles uniformly through VCC. However, there are several challenges in using in-vehicle resources to handle computing tasks:

1) Limited in-vehicle resources. For some computing tasks with large computing requirements, if the computing task is completely offloaded, the computing power of a single resource node may not be enough to support the completion of the task, so a complex task needs to be divided into multiple subtasks.
2) Extremely unstable network topology. Vehicles are constantly moving and the highly dynamic network topology affects the stability of the communication environment and computational resources.
3) Rational allocation of computational resources for subtasks. A reasonable computation offloading policy can offload different subtasks to different resource nodes for parallel processing, which effectively reduces task latency.

To solve the above problems, we propose a federated double deep Q network-based computation task offloading (FDTO) strategy. The main contributions of this paper are summarized as follows:

1) A mobility-aware vehicle clustering mechanism is proposed to reduce the impact of dynamic changes in network topology and resources on task offloading strategies.
2) The computing task is modeled as a directed acyclic graph (DAG) structure, which characterizes the temporal dependence of vehicle cloud subtasks. The computation task offloading problem for vehicle clusters is modeled as a mixed-integer nonlinear programming problem to minimize task completion delay.
3) We propose an FDTO strategy to protect user data privacy, reduce communication overhead, and have lower task completion latency compared to the benchmark strategies.

The rest of this paper is organized as follows: Section II provides an overview of the related work. In Section III, we present the system model and problem formulation. The proposed FDTO strategy is presented in detail in Section IV. Section V discusses the simulation results and the conclusion is made in Section VI.

II. RELATED WORK

Many existing studies have analyzed the task offloading problem in IoV and used MEC as a solution [14], [15], [16], [17], [18], [19]. According to the different computing nodes, the computational task offloading scenarios can be categorized into five scenarios: single MEC server, multiple MEC servers, cloud-side server collaboration, multi-vehicle collaboration, and MEC server-vehicle collaboration. In a single MEC multi-user scenario, a two-layer optimization framework based on deep Q network (DQN) and gradient descent considering task latency and energy consumption was proposed in [14]. And a joint optimal MEC server selection and offloading algorithm were studied for a multi-user multi-server MEC system in [15]. A comprehensive task processing latency was used to formulate the system utility, taking into account both transmission and computation time. By applying the game theoretic analysis in the framework of MEC computation and offloading, a particle swarm optimization-based computation offloading algorithm was proposed to minimize the time latency and cost of task offloading [16]. Task completion latency in the task offloading problem is also considered as an important metric in this paper. In addition, the task offloading strategy also was studied based on deep learning [17], [18]. A shared offloading strategy was proposed in [17], where similar computing tasks from different vehicles can share the computational results of previously submitted tasks. Considering the available vehicle resources and service migration, an asynchronous deep reinforcement algorithm was used to find the optimal task and resource scheduling policy [18]. Considering vehicle mobility, the impact of time-varying channels during task transmission cannot be ignored. The impact of time-varying channels on task offloading strategy in the MEC scenario was considered in [19]. However, the above work mainly assumed that the roadside is equipped with MEC servers having sufficient computational resources, and does not apply to the scenario where no MEC servers are deployed at the roadside. Therefore, in order to further improve the QoS, we think about other approaches besides taking MEC-assisted task offloading for IoV.

VCC serves as a new solution to provide vehicular computational resources for task offloading. VCC enables the sharing of resources between vehicles, which extends the possibilities of vehicle functions and services. Utilizing the idle on-board resources of the parked vehicles, it was scheduled as a general-purpose computing node to process the computing tasks in [20]. A new load balancing strategy was proposed to reduce the task latency in [21], stationary vehicles were used as auxiliary fog computing nodes to process the task. However, the above work mainly utilized stationary vehicle resources, and for moving vehicles, although the VCC system is more flexible compared to the MEC system, it also faces new challenges. The mobility of vehicles leads to highly dynamic network topology, rapidly changing wireless channel states, and computational resources for vehicle nodes with time-varying and heterogeneous nature, which can greatly affect task computation latency. Several existing works have researched the highly dynamic changes in network topology during the computational task offloading process in IoV [22], [23], [24]. Based on the multi-armed bandit theory, an adaptive task offloading algorithm was proposed which enhanced its applicability in the highly dynamic environment of IoV through perceptual learning of vehicle nodes during computational task offloading [22]. In [23], an RSU-based heuristic clustering algorithm was proposed to improve topology stability, which recovers from the problem of the unavailability of cluster heads by using a weighting mechanism to select secondary cluster heads. A clustering problem was transformed into a cut-plot problem in [24], aiming to improve the average survival of all clusters. In this paper, we perform vehicle clustering by considering the motion attributes such as speed and direction of vehicle nodes to form a vehicle cloud, thus reducing the dynamics of the network topology.

Deep reinforcement learning (DRL) theory has also been widely used in the VCC task offloading problem [12], [25], [26], [27]. A vehicular fog computing framework was designed in [25] to encourage vehicles to share their idle computational resources with the task vehicles and use deep deterministic policy gradient (DDPG) and twin delayed DDPG algorithms for task offloading. Considering the mobility of the vehicles, task priority, and service availability of the vehicles, the task offloading problem was formulated as a markov decision process (MDP) to minimize the average latency of the tasks [12]. To reduce the overall system energy consumption along with satisfying user latency constraints, a three-tier energy-efficient offloading scheme based on DRL was constructed for IoV [26]. To solve the many-to-many task offloading problem in a dynamic state vehicle environment, [27] proposed a distributed dynamic many-to-many task offloading framework based on the vehicle-to-vehicle (V2V) transaction paradigm. Unlike [12], [27] formulated the transaction process as a partially observable MDP and employed multi-intelligent to solve the problem. However, obtaining a DRL model with excellent performance generally requires a long training time on cloud servers based on large amounts of data, which leads to increased task latency and decreased communication efficiency. DRL model training based on mobile edge networks will also generate a large number of data interactions and will bring the problems of data privacy leakage and communication cost waste.

Federated learning (FL) provides an effective solution to the above problem. The core idea of FL is that each entity uses its own data to train a model locally, and only sends the model updates to a central server, where each computing node communicates with each other based on its own model, and ultimately obtains a global model through model aggregation [28]. Federated stochastic gradient descent (FedSGD) [29] is the fastest algorithm for model updating in FL, but it increases the number of communications as the distributed nodes need to upload the model parameters once for every gradient descent they do. Federated averaging algorithm (FedAVG) can effectively reduce the number of communications in FL [30]. FL is widely used in many domains for its ability to protect local data privacy well, reduce the communication cost incurred by data interaction in centralized learning, and obtain smarter models. For example, an intelligent traffic flow forecasting model was proposed in [31] by integrating a spatial-temporal graph-based deep learning model into the devised multilevel federated learning framework. To deploy neural network models in internet of medical things devices, a federated learning algorithm with minimum square quantization error was proposed [32]. The application of FL in MEC was summarized in [33]. However, there are fewer studies related to the application of federated learning to vehicular edge computing, and in this paper, we will take advantage of FL to study computation task offloading in IoV.

Against this background, we will fully exploit and schedule the idle vehicle resources, and investigate the task offloading problem in VCC by combining vehicle mobility and the time-dependent relationship of computing tasks. To solve the problems of traditional DRL, we propose an FDTO strategy to protect user data privacy and reduce communication overhead.

III. SYSTEM MODEL AND PROBLEM FORMULATION

As shown in Fig. 1, we consider a computation task offloading scenario for vehicle clusters consisting of N

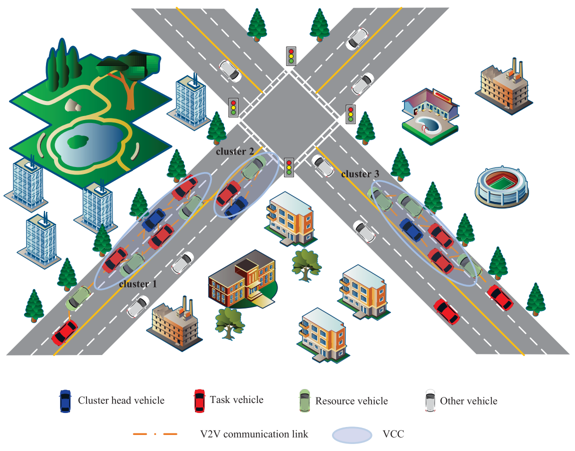


FIGURE 1. **Computation task offloading scenario base on vehicle cluster in IoV.**

vehicles. To ensure the stability of vehicle clusters, the same direction of travel is used as a prerequisite for vehicles to be classified into the same vehicle cluster. We reduce the high dynamic of vehicle network topology by grouping vehicle nodes with small relative motion speeds into the same cluster, electing a cluster head that manages the vehicle members in the cluster according to a certain criterion. The specific vehicle clustering algorithm is given in Section IV. Assume that all vehicles are equipped with global positioning system (GPS) devices that know their position, speed, and direction at any moment. Each vehicle terminal has certain computing, storage, and communication capabilities. Sufficient spectrum resources are assumed for intra-vehicle cluster communication, and vehicles are connected by V2V communication links.

Each circle in Fig. 1 is a vehicle cluster, and different vehicle types are used to represent cluster head vehicles, task vehicles, resource vehicles, and others. The set of vehicles on the current road can be denoted as $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$, the set of vehicle clusters N_{vc} is denoted as $VC = \left\{VC_1, VC_2, \ldots, VC_{N_{vc}}\right\}$ and the corresponding set of cluster heads is denoted as $CH = \left\{CH_1, CH_2, \ldots, CH_{N_{vc}}\right\}$. The vehicle cluster VC_i containing cluster head CH_i and ω_i cluster member nodes can be given as $VC_i = \left\{u_{i,0}, u_{i,1}, u_{i,2}, \ldots, u_{i,\omega_i}\right\}$, where $u_{i,0}$ is the cluster head node CH_i of VC_i. There are several resource vehicle nodes and task vehicle nodes in the vehicle cluster VC_i, which can be represented as $Rs_i = \left\{rs_{i,1}, rs_{i,2}, \ldots, rs_{i,n1}\right\}$ and $Ts_i = \left\{ts_{i,1}, ts_{i,2}, \ldots, ts_{i,n2}\right\}$ respectively.

In each vehicle cluster, the task vehicle needs to handle several latency-sensitive or computation-intensive tasks and will make a computation task offloading request when its computational capacity is insufficient. Resource vehicles in the vehicle cluster have several free computational resources that can provide computational services for task vehicles with computation task offloading requirements.

For some computation-intensive tasks, if the computing task is completely offloaded (i.e., the computing task is considered as a whole to be processed locally or offloaded to other computation nodes), the computing power of a single resource node may not be sufficient to support the completion of the task. Although the vehicle clustering mechanism allows highly dynamic vehicles to be divided into vehicle clusters with relatively stable network topology, the vehicle clusters are still in constant update, and if the computing task is complex and requires a long task processing delay, it may occur that the task vehicle or resource vehicle has left the current vehicle cluster when the task processing is completed. When the task vehicle and the resource vehicle are not within communication range of each other, the task results cannot be returned in time. Therefore, the computing tasks need to be reasonably cut and offloaded to the resource vehicles for processing to obtain the minimum processing delay of tasks.

A. MOBILITY MODEL

The instantaneous position of vehicle u_k at moment t is characterized by two-dimensional coordinates $L_k^t = \left(x_k^t, y_k^t\right)$. The instantaneous velocity set of vehicle nodes in the vehicle set $\mathcal{U}$ is defined as $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_N}\}$. The instantaneous distance between vehicle u_k and u_q is $D_{k,q} = \left\| L_k^t - L_q^t \right\|_2$, $k, q \in N$, and the angle of travel direction can be noted as $\theta_{k,q} = \arccos \frac{\boldsymbol{v_k v_q}}{|\boldsymbol{v_k}||\boldsymbol{v_q}|}$.

The vehicle velocity distribution is expressed using the truncated normal distribution [34], [35]:

$$\widehat{f_V}(v) = \frac{f_V(v)}{\int_{v_{\min}}^{v_{\max}} f_V(s)ds}, \tag{1}$$

where $f_V(v) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(v-\mu)^2}{2\sigma^2}}$ denotes the vehicle speed probability density function, μ is the average speed of all vehicle nodes, and σ is the variance of all vehicle speeds, the vehicle speed values $v \in [v_{\min}, v_{\max}]$. In addition, related studies have shown that vehicle spacing follows an exponential distribution [36], which can be written as

$$P\left(D_{k,q} > x\right) = e^{-\lambda x}. \tag{2}$$

B. TASK MODEL

To simplify the model, the problem modeling subsection will only discuss the computation task offloading problem within a vehicle cluster. A vehicle cluster VC_m which contains ω vehicle cluster members (CM) can be denoted as $VC_m = \{u_0, u_1, u_2, \ldots, u_\omega\}$, where u_0 is the cluster head node CH_m. The vehicle is used as a resource vehicle when it has free resources and will make a task offloading request and become a task vehicle when its local computing power is insufficient to handle the task. The resource vehicle set and the task vehicle set of the vehicle cluster can be denoted as $Rs \subseteq VC_m$, and $Ts \subseteq VC_m$, respectively. According to [37], the arrival of the vehicle u_k task obeys the Poisson distribution with arrival rate λ_k, i.e.,

$$P(X_k < x) = \sum_{m<x} \frac{\lambda_k^m e^{-\lambda_k}}{m!}. \tag{3}$$

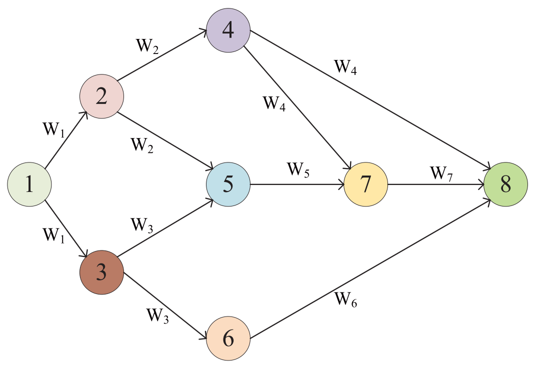


FIGURE 2. Time dependent subtask model represented by DAG.

Assuming that each computing task can be divided into several subtasks with time-dependent relationships, allowing multiple subtasks to be processed in parallel, the subtask relationships of computing tasks are characterized by DAG, as shown in Fig. 2.

The computation task offloading decision for a task vehicle will be decided within each scheduling cycle, and each scheduling cycle will process limited computing task offloading requests in the order of the request arrival time. The next scheduling cycle will handle the remaining requests that cannot be handled in the current cycle.

A computing task $task_k$ proposed by task vehicle $u_k \in Ts$ in vehicle cluster VC_m that needs to be offloaded at the tth scheduling cycle can be expressed as $G_k = \{\mathcal{T}_k, E_k, W_k\}$ by DAG, where $\mathcal{T}_k = \left\{task_{k,1}, task_{k,2}, task_{k,3}, \dots, task_{k,N_k}\right\}$, $task_{k,i}$ is the i-th subtask of $task_k$ and N_k is the number of subtasks for $\mathcal{T}_k$. E_k is the set of all edges in the directed graph G_k, $E_k = \left\{e_{k,i\,i'} \mid i, i' \in (1, 2, 3, \dots, N_k)\right\}$, $e_{k,i\,i'} \in E_k$ shows that there is a time dependence between $task_{k,i}$ and $task_{k,i'}$, i.e., the processing of $task_{k,i'}$ must be done after the completion of $task_{k,i}$. W_k is the weights set of all edges in the directed graph G_k, $W_k = \left\{w_{k,i} \mid i \in (1, 2, 3, \dots, N_k)\right\}$, and $w_{k,i}$ is the amount of data generated after subtask $task_{k,i}$ is processed. when $i = N_k$, w_{k,N_k} represents the amount of data calculated by task $\mathcal{T}_k$.

The task vehicle node u_k will make an offloading decision for $task_k$ based on the task size, sub-task dependencies, available computing resources, V2V communication link status, etc. The offloading decision is denoted as $\mathcal{A}_k = \left\{a_{k,i} \mid i \in \{1, 2, \dots, N_k\}, a_{k,i} \in \{1, \dots, \omega\}\right\}$, where $a_{k,i} = j$ indicates that the subtask $task_{k,i}$ of $task_k$ is offloaded to the resource vehicle for processing, and $a_{k,i} = k$ indicates that the subtask $task_{k,i}$ is processed locally in the task vehicle u_k.

C. COMMUNICATION MODEL

When $e_{k,i\,i'} \in E_k$, $task_{k,i}$ is a preorder node of $task_{k,i'}$. When $task_{k,i}$ and $task_{k,i'}$ are offloaded to different vehicle nodes $u_{a_{k,i}}$ and $u_{a_{k,i'}}$ for processing, denote $a_{k,i} = j$, $a_{k,i'} = j'$ for convenience. Before $u_{j'}$ can perform $task_{k,i'}$, u_j must provide date $w_{k,i}$ to $u_{j'}$. When $e_{k,i\,i'} \notin E_k$ or $e_{k,i\,i'} \in E_k$ but $task_{k,i}$ and $task_{k,i'}$ are assigned to the same vehicle node for processing, there is no need for data transfer. Define $w_{k,i\,i'}$ as the data to be transferred after $task_{k,i}$ has been completed and before $task_{k,i'}$ will be processed, then

$$w_{k,i\,i'} = \begin{cases} 0, & \text{if } e_{k,i\,i'} \notin E_k \text{ or } a_{k,i} = a_{k,i'} \\ w_{k,i}, & \text{if } e_{k,i\,i'} \in E_k \text{ and } a_{k,i} \neq a_{k,i'}, \end{cases} \tag{4}$$

where $i \in (1, 2, 3, \dots, N_k)$.

Only the resource vehicle within the one-hop V2V communication distance R_{max} of the task vehicle is considered for offloading the task. The V2V communication between vehicles is accessed using orthogonal frequency division multiple access. Using the instantaneous transmission rate of the V2V communication link between resource vehicle nodes u_j and $u_{j'}$ to characterize the data transmission rate, i.e.,

$$r_{jj'} = B\log_2\left(1 + \frac{p_{jj'}h_{jj'}}{N_0}\right), \tag{5}$$

where B is the channel transmission bandwidth of the current V2V communication link, $p_{jj'}$ is the transmit power from u_j to $u_{j'}$, $h_{jj'}$ is the channel gain model between u_j and $u_{j'}$, and N_0 is the noise power. So the transfer time required for subtasks $task_{k,i}$ and $task_{k,i'}$ to be assigned to resource vehicles u_j and $u_{j'}$, respectively, is calculated as

$$t_{k,i\,i'}^{\text{trans}} = \frac{w_{k,i\,i'}}{r_{jj'}}. \tag{6}$$

Denoting $N_k = m$, $N_k + 1 = m'$. When $a_{k,m'} = k$, the transmission time required to return computation task results of task vehicle u_k is calculated as

$$t_{k,m'}^{\text{trans}} = \frac{w_{k,m\,m'}}{r_{a_{k,m}a_{k,m'}}}, \tag{7}$$

where $w_{k,m\,m'}$ is the amount of data returned from the calculation. When the last subtask $task_{k,m}$ is calculated locally, there is no requirement for result return, when $task_{k,m}$ is offloaded to other resource vehicles for calculation, the data size returned from the calculation is $w_{k,m}$, i.e.,

$$w_{k,m\,m'} = \begin{cases} 0, & \text{if } a_{k,m} = k \\ w_{k,m}, & \text{if } a_{k,m} \neq k. \end{cases} \tag{8}$$

D. COMPUTATION MODEL

We define that subtasks can be processed locally or offloaded to other resource vehicles within the vehicle cluster. Assuming that the CPU cycles required to compute subtask $task_{k,i}$ are $c_{k,i}$ cycles, the set of CPU cycles required for $task_k$ is $C_k = \left\{c_{k,1}, c_{k,2}, \dots c_{k,m}\right\}$. When the CPU frequency of locally available computational resources of task vehicle u_k is f_k, the computational latency of $task_{k,i}$ in local computation is expressed as

$$t_k^{\text{comp}_{k,i}} = \frac{c_{k,i}}{f_k}. \tag{9}$$

Denote the number of unprocessed computing tasks in the task vehicle as $N_{k,wait}$ and the list of queued subtasks as $TW_k = \left\{task_{k,q}, task_{k,o}, \dots, task_{k,z}\right\}$. Then the waiting delay required to calculate locally in the task vehicle can be expressed as

$$t_k^{\text{wait}_{k,i}} = \sum_{task_{k,o \in TW_k}} t_k^{\text{comp}_{k,o}}. \tag{10}$$

The currently available resources of resource vehicle u_j are denoted as f_j, and u_j does not necessarily allocate all computational resources to the same offloading task. When subtask $task_{k,i}$ is offloaded to resource vehicle u_j, the computational resources allocated to $task_{k,i}$ is written as $f_j^{k,i}$, $0 < f_j^{k,i} \leq f_j$. Then the computational latency of $task_{k,i}$ being offloaded to the resource vehicle u_j is calculated as

$$t_j^{\text{comp}_{k,i}} = \frac{c_{k,i}}{f_j^{k,i}}. \tag{11}$$

Since multiple subtasks are selecting the same resource vehicle for processing, assume that subtask $task_{k,i}$ arrives at resource vehicle u_j with $N_{j,wait}$ subtasks unprocessed and the list of queued subtasks is $TW_j = \left\{task_{k_1,i_1}, task_{k_1,i_2}, \ldots, task_{k_\xi,i_\eta}\right\}$. The waiting delay required for $task_{k,i}$ to process at resource vehicle u_j is calculated as

$$t_j^{\text{wait}_{\text{k,i}}} = \sum_{task_{k_q,i_o \in TW_j}} t_j^{\text{comp}_{k_q,i_o}}. \tag{12}$$

In summary, the total task processing time $t_{k,i}^{process}$ for subtask $task_{k,i}$ is expressed as

$$t_{k,i}^{proc} = t_j^{\text{comp}_{k,i}} + t_j^{\text{wait}_{k,i}}, \tag{13}$$

when $j = k$, the subtask $task_{k,i}$ is processed locally in the task vehicle. Denote the set of computational resources allocated to $task_k$ as $\mathcal{F}_k = \left\{f_{a_{k,1}}^{k,1}, f_{a_{k,2}}^{k,2}, \ldots, f_{a_{k,m}}^{k,m}\right\}$, and $f_{a_{k,1}}^{k,1}$ represents the size of computational resources available when $task_{k,i}$ is assigned to vehicle node $u_{a_{k,i}}$ for processing.

E. PROBLEM FORMULATION

To better model the task offloading problem, we first give the following four definitions:

Subtask start time: the earliest moment when the subtask can start execution. This means that the subtask has already received the calculation result data of all previous subtasks. The preorder subtask set of subtask $task_{k,i}$ is denoted as $pre\left(task_{k,i}\right)$. The start time of $task_{k,i}$ can be expressed as

$$\text{ST}_{k,i} = \max_{task_{k,j} \in pre\left(task_{k,i}\right)} \left(\text{FT}_{k,j} + t_{k,j\,i}^{\text{trans}}\right). \tag{14}$$

Subtask starts execution time: the moment when the subtask actually starts to be calculated. After receiving all the required data, subtasks may need to queue for computational resources before starting their computation. Therefore, the start execution time of subtask $task_{k,i}$ can be expressed as

$$\text{ET}_{k,i} = \text{ST}_{k,i} + t_{a_{k,i}}^{\text{wait}_{k,i}}. \tag{15}$$

Subtask completion time: the moment when the subtask has completed the calculation and generated the result data. The completion time of subtask $task_{k,i}$ can be expressed as

$$\text{FT}_{k,i} = \text{ET}_{k,i} + t_{a_{k,i}}^{\text{comp}_{k,i}}. \tag{16}$$

Task completion time: the moment when the task vehicle node receives the result of the whole calculation task. The completion time of the calculation task $task_k$ can be expressed as

$$t_k^{\text{complete}} = \left(\max_{task_{k,j} \in \mathcal{T}_k} \text{FT}_{k,j}\right) + t_{k,mm'}^{\text{trans}}. \tag{17}$$

To minimize the task completion latency of $task_k$, it is necessary to allocate the optimal resource vehicle and computational resources for each subtask of $task_k$. Thus, for the task vehicle u_k, the optimization problem can be formulated as

$$\begin{aligned} \text{P0:}\ & \min_{\mathcal{A}_k, \mathcal{F}_k} \left(t_k^{\text{complete}}\right), \\ \text{s.t.}\ & C1: D_{a_{k,i} pre\left(a_{k,i}\right)} < R_{\max}, \\ & C2: a_{k,i} \in \{1, \ldots, \omega\}, \\ & C3: 0 < f_{a_{k,i}}^{k,i} \leq f_{a_{k,i}}, \\ & C4: i \in (1, 2, 3, \ldots, N_k), \end{aligned} \tag{18}$$

where constraint C1 ensures that any resource vehicle offloaded by a subtask should be within one hop of the V2V link with the resource vehicles offloaded by its predecessor subtasks. Constraint C2 indicates that the resource vehicles offloaded by any subtask should be in the same vehicle cluster as the task vehicle. Constraint C3 is the computational resource allocated for the offloaded subtask, which is non-negative and must not exceed the maximum computational resource available from the resource vehicle.

Multiple task vehicle nodes may exist simultaneously within a vehicle cluster, and the set of computation task offloading requests for vehicle cluster VC_m at period t is denoted as $\zeta = \left\{\mathcal{T}_k, \mathcal{T}_q, \ldots, \mathcal{T}_o\right\}$. The computation offloading decision of each task vehicle may have an impact on the offload of other computing tasks within the same vehicle cluster. To comprehensively consider the interaction of task vehicles within a vehicle cluster, we further minimize the computational task completion latency of all vehicle nodes within a vehicle cluster as the optimization objective, and the computation task offloading decision optimization problem for the vehicle cluster can be formulated as

$$\begin{aligned} \text{P1:}\ & \min_{\mathcal{A}, \mathcal{F}} \left(\max_{task_k \in \zeta} t_k^{\text{complete}}\right), \\ \text{s.t.}\ & C1: D_{k,a_{k,i}} < R_{\max}, \\ & C2: a_{k,i} \in \{1, \ldots, \omega\}, \\ & C3: 0 < f_{a_{k,i}}^{k,i} \leq f_{a_{k,i}}, \\ & C4: i \in (1, 2, 3, \ldots, N_k), \\ & C5: k \in (1, 2, 3, \ldots, \omega), \end{aligned} \tag{19}$$

where constraints C1, C2, and C3 have the same meaning as (18). In the optimization problem, the variables $\mathcal{A}$ and $\mathcal{F}$ represent the set of integers and the set of continuous real numbers, respectively. P1 is a mixed integer nonlinear programming problem (MINLP), which is a non-deterministic polynomial-time (NP) problem with polynomial complexity and is non-convex.

IV. OPTIMIZATION SOLUTION

Since MINLP is difficult to solve by traditional methods, an FDTO strategy for vehicle clusters is put forward to solve the optimization problem. We first describe the mobility-based vehicle clustering mechanism, followed by the double deep Q network (DDQN)-based single-node computation task offloading strategy. As the traditional centralized DDQN training process will generate a large amount of training data interactions, an FDTO strategy for vehicle cluster computing tasks is further presented which is more applicable to the communication resource-constrained vehicular network scenario. The FDTO strategy which is based on FL and DDQN allows each distributed node to train a local DDQN model locally using its local data, and only interact with the central node for model parameters, which reduces the bandwidth consumption and communication overhead associated with training data transfer.

A. VEHICLE CLUSTERING MECHANISM

To ensure the stability of vehicle clusters, it is specified that only vehicle nodes traveling in the same direction can be divided into the same vehicle cluster. When the speed angle $\theta_{k,q} \leq \pi/4$ between vehicle nodes u_k and u_q, it can be regarded as traveling in the same direction, otherwise, the traveling direction is opposite. Express the relative mobility of u_k and u_q using their relative velocities, i.e.,

$$\mathrm{RV}(k,q) = \left|\boldsymbol{v_k} - \boldsymbol{v_q}\right|. \tag{20}$$

In the one-hop V2V communication range with R_{max} as the communication radius, the n_k direct neighbor vehicles of u_k is recorded as the direct neighbor list $DNL_k = \left\{u_1, u_2, \ldots, u_{n_k}\right\}, |DNL_k| = n_k$. Then the relative mobility of u_k and its direct neighbor nodes can be expressed as

$$\mathrm{RV}(k) = \frac{1}{n_k}\sum_{u_q \in DNL_k} \mathrm{RV}(k,q). \tag{21}$$

According to (21), the smaller $\mathrm{RV}(k)$ is, the lower the relative speed of u_k and its direct neighbor nodes. In other words, u_k may have a longer communication hold time with its neighbor nodes. It is similarly defined that the relative distance between u_k and its direct neighbor nodes is

$$\mathrm{RD}(k) = \frac{1}{n_k}\sum_{u_q \in DNL_k} D_{k,q}. \tag{22}$$

Based on $\mathrm{RV}(k)$ and $\mathrm{RD}(k)$, define the cluster head factor of u_k as

$$M(k) = \alpha_1 \mathrm{RV}(k) + \alpha_2 \mathrm{RD}(k) + \alpha_3 \frac{1}{n_k}, \tag{23}$$

where α_1, α_2, α_3 are weighting factors, $\alpha_1, \alpha_2, \alpha_3 \in (0,1)$, $\alpha_1 + \alpha_2 + \alpha_3 = 1$. The smaller M_k is, the more likely for u_k to be a cluster head. Denote the cluster head of a vehicle cluster VC_m as CH_m, and there are currently ω_m vehicle nodes in VC_m, the average speed of this vehicle cluster is

$$\overline{v_m} = \frac{1}{\omega_m}\sum_{k=1}^{\omega_m} |\boldsymbol{v_k}|. \tag{24}$$

To further discuss the computation task offloading problem of the vehicle cluster, fr_m is introduced to represent the idleness of computational resources in VC_m, i.e.,

$$fr_m = \frac{N_{task}(m)}{N_{res}(m)}, \tag{25}$$

where $N_{task}(m)$ and $N_{res}(m)$ are the number of task vehicles and resource vehicles in VC_m, respectively. The computational resources in the vehicle cluster are relatively tight When $fr_m > \epsilon$, and fr_m should be kept below the threshold value ϵ as much as possible.

For the neighbor node u_k of CH_m, its cluster entry factor can be defined as

$$J_{k,m} = \lambda_1 ||\boldsymbol{v_k}| - \overline{v_m}| + \lambda_2 fr_m, \tag{26}$$

where λ_1, λ_2 are the influence factors of speed and the degree of computational resource idleness, respectively, $\lambda_1, \lambda_2 \in [0,1]$ and $\lambda_1 + \lambda_2 = 1$. When $J_{k,m} < \mu$, u_k can join the vehicle cluster VC_m, μ is a constant. Specify that the vehicle cluster VC_m can contain up to N_{max} cluster members.

The vehicle clustering mechanism can be split into two parts, i.e., the vehicle cluster generation process and the dynamic maintenance process of vehicle clusters, as shown in Algorithm 1. In the initial stage, all vehicles are isolated nodes. Taking isolated vehicle node u_k as an example, the vehicle nodes that are in the communication range of u_k with the same direction of motion are recorded in the list of direct neighbor nodes DNL_k. According to (26), each node will compute its own $M(k)$ and broadcast it to its direct neighbor nodes. If the $M(k)$ of vehicle node u_k is less than the cluster head factor of all nodes in DNL_k, u_k will be elected as the cluster head, otherwise, the node with the smallest cluster head factor $M(k)$ in DNL_k will become the cluster head. After the cluster head election is completed, a vehicle cluster VC_m with vehicle node u_h as the cluster head CH_m will be generated. Then CH_m will broadcast the cluster head message to the nodes in its direct neighbor list DNL_m. A vehicle node u_0 which receives the cluster head message sends a cluster request to CH_m. If u_0 receives cluster head messages from more than one cluster head node, u_0 will choose the cluster head node with the smallest cluster head factor to send the cluster request. And CH_m will calculate the cluster head factor of each node after receiving the cluster requests from other nodes and add them to the cluster member list (CML) in order of the smallest to the largest cluster head factor, then reply to the requests. If the number of members in the CML of vehicle cluster VC_m exceeds N_{max} or the cluster entry factor of u_0 exceeds the threshold μ, the cluster request of u_0 may be

Algorithm 1 Mobility-Aware Vehicle Clustering Mechanism

```
Input: Maximum vehicle cluster capacity $N_{max}$, constant $\mu$, vehicle set $\mathcal{U}$ and its information
Output: Number of vehicle clusters $N_{vc}$, vehicle cluster $VC = \left\{VC_1, VC_2, \ldots, VC_{N_{vc}}\right\}$, set of cluster head $CH = \left\{CH_1, CH_2, \ldots, CH_{N_{vc}}\right\}$, list of cluster members $CML = \left\{CML_1, CML_2, \ldots, CML_{N_{vc}}\right\}$
 1: Each vehicle node in $\mathcal{U}$ sends Hello packets and generates the set of DNL lists $DNL = \{DNL_1, DNL_2, \ldots, DNL_N\}$.
 2: Each vehicle node $u_k \in \mathcal{U}$ calculates the angle of motion with the nodes in $DNL_k$ and updates DNL.
 3: Calculate the value of $M(k)$ for each vehicle node according to (26) and send it to the nodes in $DNL_i$.
 4: for $u_k \in \mathcal{U}$ do
 5:   if exists cluster head $CH_m \in DNL_m$ and its $M(m)$ is the smallest in $DNL_k$ then
 6:     if $N_{\max} > \omega_m$ and $J_{k,m} < \mu$ then
 7:       Add $u_k$ to the list of vehicle cluster members $CML_m$ where $CH_m$ is located.
 8:       for $u_j \in CML_m$ do
 9:         if No periodic message from $u_j$ is received by $CH_m$ in one cycle or $||\mathbf{v}_j| - \bar{v}_i| > \rho$ then
10:           $CH_m$ removes $u_j$ from $CML_m$
11:         end if
12:       end for
13:     else
14:       Reject the application $u_k$ to join the cluster
15:     end if
16:   else if $M(k)$ is the smallest in $DNL_k$ then
17:     $u_k$ is elected as cluster head $CH_o$ and send cluster head messages to all nodes in $DNL_k$
18:   else
        The $u_k$ becomes an isolated node and tries to join other vehicle clusters
19:   end if
20: end for
```

rejected. Isolated nodes not in a cluster on the current road are treated as self-contained clusters, but these isolated nodes will keep trying to join the cluster. After the vehicle cluster VC_m is generated, to maintain the stability of the vehicle cluster network topology, CH_m needs to maintain the vehicle cluster dynamically, until the vehicle cluster VC_m is dissolved. When CH_m does not receive the periodic message from a cluster member node u_0 within a cycle or the difference between the speed of cluster member u_0 and the average speed of VC_m exceeds the set threshold value ρ, u_0 will be removed from the cluster member list.

Through the above mobility-based vehicle clustering mechanism, the highly dynamic vehicles can be divided into vehicle clusters with relatively stable network topology. The clustering mechanism takes into account the location of vehicles, travel speed, the number of members that can be accommodated in the vehicle cluster, and the idleness of computational resources to ensure that the communication load of cluster head vehicle is within a tolerable range, the vehicle cluster structure tends to be stable and the distribution of task vehicles and resource vehicles in the cluster is balanced.

B. DDQN-BASED SINGLE-NODE TASK OFFLOADING STRATEGY

We solve the computation task offloading problem P1 for vehicle clusters based on FL. FL is a special kind of distributed learning where the model training process involves several distribution nodes and a central node. Each task vehicle in the vehicle cluster can be considered an agent, namely a distribution node, and the cluster head is considered the central node. In this paper, a single-node computational task offloading strategy based on DDQN is first given to illustrate the model training process for distributed nodes in FL, which is equivalent to solving the optimization problem P0. DDQN is a common method of DRL, which can better solve the overestimation problem in Q learning. Compared to DQN, it is able to converge faster and better. According to DRL theory, the computation task offloading problem of a single node is modeled as an MDP, and the MDP can be defined as $(\mathcal{S}, \mathcal{A}, \mathcal{P}(S_{t+1} \mid S_t, A_t), \mathcal{R}(S_t, A_t))$, corresponding to the state space $\mathcal{S}$, action space $\mathcal{A}$, state transfer probability $\mathcal{P}$ and reward $\mathcal{R}(S_t, A_t)$ after acting, respectively. Based on MDP, the state space, action space, and reward function for the computation task offloading problem for each task vehicle node u_k are defined as follows:

State Space: the state set $\mathcal{S}$, of task vehicle u_k. At time slot t, the state $S_t \in \mathcal{S}$ of task vehicle u_k can be expressed as

$$S_t = \{loc_t, v_t, \mathcal{G}_t, C_t, VNL_t\}, \tag{27}$$

where loc_t, v_t, $\mathcal{G}_t$, C_t, VNL_t represent the position, speed, computing task, computational resource requirements, and the list of direct neighboring vehicles within the same vehicle cluster as u_k at time slot t, respectively.

Action Space: the action set A of task vehicle u_k. At time slot t, the action $A_t \in \mathcal{A}$ of task vehicle u_k can be expressed as

$$\mathcal{A}_t = \{a_{k,i}, f_{a_{k,i}}^{k,i} \mid i \in \{1, 2, \ldots, N_k\}, \\ a_{k,i} \in \{1, \ldots, \omega\}, 0 < f_{a_{k,i}}^{k,i} < f_{a_{k,i}}\}, \tag{28}$$

where $a_{k,i} = j$ represents the subtask $task_{k,i}$ of computational task $\mathcal{T}_k$ of task vehicle u_k offloaded to resource vehicle u_j. for processing, and $f_{a_{k,i}}^{k,i}$ is the computational resources allocated to $task_{k,i}$ by u_j.

Reward Function: the reward value that can be obtained by u_k based on state S_t performing action A_t. The reward function plays a key role in the performance of DDQN and allows the algorithm to continuously converge to the optimal value. The reward function is inversely proportional to the computational task completion latency, given as

$$\mathcal{R}(S_t, A_t) = \frac{1}{t_k^{complete}}. \tag{29}$$

Define the discount factor $\gamma \in [0,1]$, long-term cumulative discount rewards can be expressed as

$$G_t = \sum_{t=1}^{T} \gamma^{t-1}\mathcal{R}(t). \tag{30}$$

In DRL, the behavior of the agent depends on the strategy ϑ, which is a mapping from the state space to the action space, expressed as $\vartheta : \mathcal{S} \to \mathcal{A}$. For each agent, the optimization problem P1 can be equated to finding an optimal strategy ϑ^* to maximize the long-term cumulative return G_t. The action value function $Q_\vartheta(s,a)$ is expressed as

$$Q_\vartheta(s,a) = \mathbb{E}_\vartheta\left[G_t \mid S_t = s, A_t = a\right]. \tag{31}$$

DDQN consists of a target network and an evaluation network. In each update of the Q-value, DDQN uses the evaluation network to determine the action and combines the identified action with the target network to estimate its reward. During the initial phase of the algorithm, network parameters will be generated for the target and evaluation networks, and the evaluation network collects samples to put in the experience playback pool by interacting with the environment. When the experience replay pool is full, a batch of samples will be randomly selected from the experience replay pool for training in each iteration, and the evaluation network will select action A_t based on the $\mathcal{E}$-greedy strategy and the current state S_t. Then the target Q-value of the target network will be calculated as

$$\begin{aligned} y^t &= \mathcal{R}(S_t, A_t) \\ &\quad + \gamma Q\left(S_{t+1}, \arg\max_{A^{t+1}} Q\left(S_{t+1}, A_{t+1}; \delta^t\right); \delta^-\right), \end{aligned} \tag{32}$$

where δ^- is the current parameter of the target network, δ^t represents the parameter of the evaluation network in the tth iteration, and $\arg\max_{A^{t+1}} Q\left(S_{t+1}, A_{t+1}; \delta^t\right)$ is the optimal action of the next state determined by the evaluation network. The loss function can be written as

$$Loss\left(\delta^t\right) = E\left[\left(y^t - Q\left(S_t, A_t; \delta^t\right)\right)^2\right]. \tag{33}$$

The parameters of the model are updated by gradient descent and the gradient which is obtained by differentiating the loss function by δ^t can be calculated as

$$\begin{aligned} \nabla_{\delta^t} Loss\left(\delta^t\right) = E\big[2\left(Q\left(S_t, A_t; \delta^t\right) - y^t\right) \\ \cdot \nabla_{\delta^t} Q\left(S_t, A_t; \delta^t\right)\big]. \end{aligned} \tag{34}$$

The parameters for evaluating network updates can be expressed as

$$\delta^t \leftarrow \delta^{t-1} - \xi \nabla_{\delta^t} Loss\left(\delta^t\right), \tag{35}$$

Algorithm 2 Single-Node Computation Task Offloading Strategy Based on DDQN

```
Input: Main network, target network, discount factor $\gamma$
Output: Computation task offloading strategy $\vartheta^*$
1:  Initialization: Experience Pool D, mini-batch b, master
    network model Q, parameter $\delta$, target network model $\widehat{Q}$,
    parameter $\hat{\delta}$
2:  Set the probability value $\varepsilon$ in the $\varepsilon$-greedy policy
3:  for $t = 1, T$ do
4:      Get the current state $S_t$, action space $\mathcal{A}$ based on the
        environment
5:      Randomly select action $A_t$ with probability $\varepsilon$, or select
        the current optimal action $a_t = \max_A Q^*(S_t, A; \delta)$
        according to the model
6:      Perform action $A_t$ to get state $S_{t+1}$ and reward $A_{t+1}$
7:      Deposit experience $(S_t, A_t, R_{t+1}, S_{t+1})$ into D
8:      while t exceeds the maximum number of iterations or
        D is full do
9:          Sampling d experiences $\left(S_j, A_j, R_{j+1}, S_{j+1}\right)$ from D
10:         Calculate the target Q-value
            $y_j = \begin{cases} R_{j+1}, S_{j+1} \text{ is the final state} \\ R_{j+1} + \gamma Q\left(S_{j+1}, \arg\max_{A^{j+1}} Q\left(S_{j+1}, A_{j+1}; \delta\right); \hat{\delta}\right)) \\ , S_{j+1} \text{ is not the final state} \end{cases}$
11:         Perform gradient descent on the loss function and
            update the main network parameters $\delta$
12:         Update the target network parameter $\hat{\delta} \leftarrow \delta$ every
            C rounds
13:         Jump to step 12
14:     end while
15:     Training is over, jump to step 6
16: end for
17: Input the current state based on the trained model, and
    output the computational task offloading policy
```

where ξ is the study rate. The target network parameters are updated per C iterations according to the evaluation network parameters, i.e.,

$$\delta^- \leftarrow \delta^t. \tag{36}$$

The model training of DDQN is completed when the value of the loss function tends to converge or reaches the maximum number of iterations. The algorithm flow is presented in Algorithm 2.

C. FEDERATED DDQN-BASED COMPUTATION TASK OFFLOADING STRATEGY FOR VEHICLE CLUSTERS

To further solve the optimization problem P1, we proposed an FDTO strategy which is based on FL and DDQN for vehicle cluster computing tasks. FL allows multiple task vehicles within a vehicle cluster to be trained collaboratively using

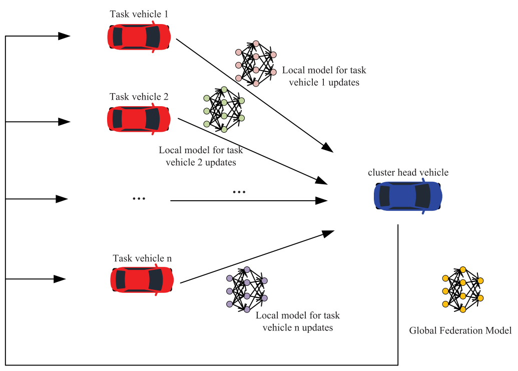

Task vehicle 1
Local model for task vehicle 1 updates
Task vehicle 2
Local model for task vehicle 2 updates
cluster head vehicle
...
...
Task vehicle n
Local model for task vehicle n updates
Global Federation Model

FIGURE 3. **FL-based task offloading architecture for vehicle clusters.**

their local data, which enables each intelligence to obtain a better model than just its own training. Figure 3 depicts an FL-based vehicle cluster computation task offloading architecture. This architecture is essentially a distributed learning architecture, where each cluster head vehicle is considered a temporary central node responsible for aggregating training models and distributing aggregation results, and task vehicle nodes in the vehicle cluster are treated as distribution nodes responsible for training their local models according to Algorithm 2 and uploading local model parameters.

We use the federated averaging algorithm (FedAVG) to update the model parameters in FL. FedAVG reduces the number of communications in the federal stochastic gradient descent algorithm by updating the local model parameters through multiple local model training at each distribution node. The model training process for FedAVG can be described specifically as follows:

1) Model Download: In each round of communication, the K vehicle nodes involved in FL training download the current global model parameter ω_m from the cluster head node, where m denotes the mth round training.
2) Local Model Training: In each FL training step, each distribution node performs E iterative updates of its local model parameter ω_m^i based on its local dataset and the downloaded model parameters, where $i \in [1, K]$. The FL parameters of the distributed node u_k at the tth iteration of the mth round training are updated as

$$\omega_{m,t}^k \leftarrow \omega_{m,t-1}^k - \eta \nabla f\left(\omega_{m,t-1}^k, \mathcal{D}_{m,t}^k\right), \tag{37}$$

where $\omega_{m,t}^k$ and $\omega_{m,t-1}^k$ are the model parameters that are updated by the local FL model of node u_k after the tth and $(t-1)$th iterations of the local mth round training, respectively, η represents the learning rate, $\mathcal{D}_{m,t}^k$ is the training sample set for this iteration of node u_k, $f\left(\omega_{m,t-1}^k, \mathcal{D}_{m,t}^k\right)$ is the loss function, $\nabla f\left(\omega_{m,t-1}^k, \mathcal{D}_{m,t}^k\right)$ represents the gradient of $f\left(\omega_{m,t-1}^k, \mathcal{D}_{m,t}^k\right)$ to $\omega_{m,t-1}^k$, $\omega_{m,0}^k = \omega_m$. When the distribution node u_k has completed the mth round training, its updated local model parameters can be written as

$$\omega_{m+1}^k \leftarrow \omega_m^k - \eta \nabla f\left(\omega_m^k, \mathcal{D}_m^k\right), \tag{38}$$

where $\mathcal{D}_{m,t}^k$ is the set of training samples for the mth round of the distribution node. When each node has finished updating its local model, the model parameters are uploaded to the cluster head node.
3) Global Model Update: The cluster head node will aggregate the received FL parameters and then update the global FL model, which is formulated as

$$\omega_{m+1} \leftarrow \omega_m - \eta \sum_{k=1}^{K} \frac{n_k}{n} \omega_{m+1}^k, \tag{39}$$

where n_k is the number of training samples of the distribution node u_k, $n = \sum_{k=1}^{K} n_k$. Keep updating the local and global FL parameters iteratively using the above steps until the model converges, and then the optimal global parameters $\omega_* = \omega_*^1 = \cdots = \omega_*^K$ will be obtained.

Combining Algorithm 1 and Algorithm 2, the FDTO strategy is presented in Algorithm 3.

Algorithm 3 The FDTO Strategy for Vehicle Clusters

```
Input: Distribution of vehicle nodes, Maximum number of cluster members $N_{max}$, constant $\epsilon$, constant $\mu$, mini-batchsize b, Number of iterations E
Output: Computation task offloading strategy
1:  Initialization: Execution of Algorithm 1 clusters the vehicles based on mobility and obtains the vehicle cluster number $N_{vc}$, the vehicle cluster set, and the cluster head vehicle set.
2:  for Each iteration do
3:    for Task vehicles $u_k$ in each vehicle cluster do
4:      if Task vehicles have sufficient computing resources to complete computing tasks then
5:        $u_k$ performs its computational task locally
6:      else
7:        Download the current global model $\omega_m$ from the cluster head node
8:        for $t$=1,$E$ do
9:          Update local model parameters based on local dataset and Algorithm 2
10:         $\omega_{m+1}^k \leftarrow \omega_m^k - \eta \nabla f\left(\omega_m^k, \mathcal{D}_m^k\right)$
11:       end for
12:       $u_k$ uploads $\omega_{m+1}^k$ to the cluster head node
13:       Cluster head node aggregation local model parameters
14:       $\omega_{m+1} \leftarrow \omega_m - \eta \sum_{k=1}^{K} \frac{n_k}{n} \omega_{m+1}^k$
15:     end if
16:   end for
17: end for
18: Training ends
```

TABLE 1. Simulation parameters.

Parameter	Value
The length of road (km)	5
Minimum vehicle speed v_{min} (km/h)	20
Maximum vehicle speed v_{max} (km/h)	50
V2V communication distance (m)	200
Task arrival rate	5
Vehicle cluster maximum tolerable number of members N_{max}	50
Communication bandwidth per vehicle (MHz)	20
Vehicle transmitting power (dBm)	23
Noise power (dBm)	-114
V2V path loss	148.1+40log(d[km])
V2V Shadow decay standard deviation (dB)	3
Vehicle local computing resources (GHz)	[1,3]
Computation resource requirements for computing tasks (MHz)	[100-1200]
Mini-batch size	500
Exploration rate	0.001
Discount factor	0.9
Experience pool capacity	3000
Learning rate	0.001

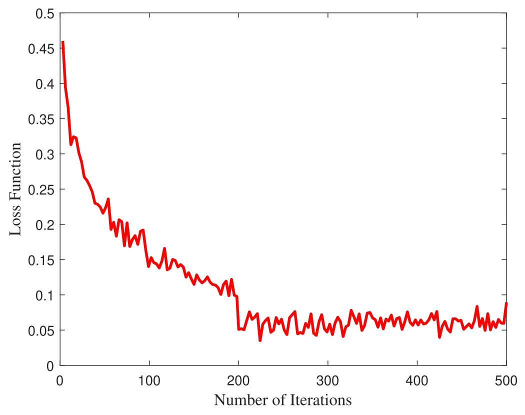


FIGURE 4. Algorithm convergence verification.

V. SIMULATION EVALUATION

In this section, we present simulation results to demonstrate the performance of our proposed FDTO strategy. The convergence of the proposed strategy is validated in Section IV-A and the performance is evaluated in Section IV-B.

Consider an urban two-way four-lane scenario with a road length of 5000 meters, where the velocity and distribution of vehicles are generated according to (1) and (2), and the computing tasks of vehicles obey the Poisson distribution with an arrival rate of five. The strategy allows the task vehicle to perform the training process distributively using its local data. The relevant important parameters in the simulation are set as shown in Table 1.

A. CONVERGENCE OF ALGORITHM

Convergence can be accelerated by introducing a server (cluster head vehicle) learning rate η during the training process. By adjusting the learning rate on the server side, local model updates from different task vehicles can be fused faster during the global model aggregation process. In this paper we set the learning rate to 0.001 and the experience pool capacity to 3000. Figure 4 shows the relationship between the loss function and the number of iterations of the FDTO algorithm. It can be seen that when the number of training is greater than 200, the value of the loss function tends to stabilize and the algorithm tends to converge.

B. PERFORMANCE OF ALGORITHM

Figure 5 shows the relationship between the number of vehicles and the end-to-end latency for different vehicle clustering mechanisms. The minimum ID clustering mechanism and the MOBIC clustering mechanism [38] were selected for comparison with the mobility-based vehicle clustering mechanism. The minimum ID clustering mechanism only assigns a unique ID to the vehicle based on the physical address and selects the node with the smallest ID value as the cluster head, while the MOBIC clustering mechanism only clusters the vehicles based on the relative movement between two nodes. The simulation result shows that the mobility-based vehicle clustering mechanism proposed in this paper has the lowest end-to-end latency with the different numbers of vehicles.

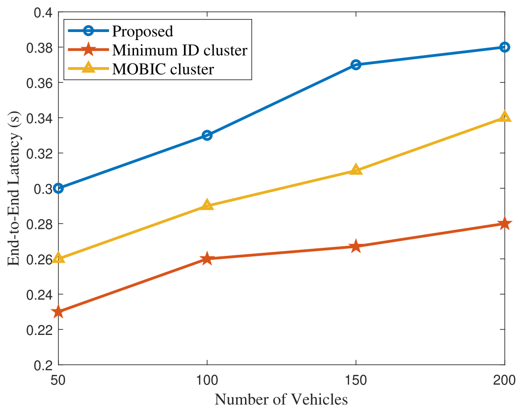


FIGURE 5. End-to-end latency with different clustering mechanisms.

Communication and computation cost: the communication cost of FL is mainly due to the interaction of model parameters (or gradients) between the client and the server during the training process. During the training process, the cluster head vehicle needs to communicate with K participants (task vehicles), thus its communication cost is $O(K)$. Each participant only needs to communicate with the cluster head vehicle, hence its communication cost is $O(1)$. The cluster head vehicle computes the global model based on the K models uploaded by the participants, each model contains n parameters, and for each parameter, its

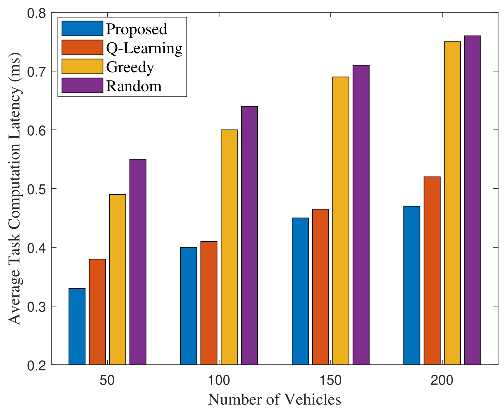


FIGURE 6. Average task computation latency.

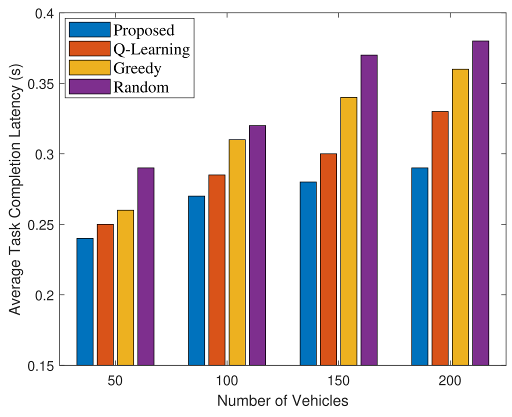


FIGURE 7. Average task completion latency.

average is computed, hence its time complexity is $O(Kn)$. Participants optimize the model based on local data, the number of local training rounds is E, t_1 denotes the time for local model training on each device, and the communication time complexity includes the time required for uploading the parameters from the task vehicle to the cluster-head vehicle and the subsequent download of the updated global model from the cluster-head vehicle to the task vehicle, denoted as t_2, and the time complexity of updating the model parameters by the neural network optimization algorithm, which consists of one time of forward propagation and back-propagation in each round, is thus $O(E(t_1 + t_2))$. Figures 6 and 7 compare the performance of different vehicle cluster computation task offloading strategies. To analyze the performance of the FDTO strategy for vehicle clusters, we selected the following three strategies for performance comparison.

1) Centralized Q-learning-based offloading strategy for vehicle cluster computing tasks: this strategy differs from the proposed one in that the cluster head vehicle collects data from all vehicles within the vehicle cluster and performs the entire training process using Q-learning.
2) Random offloading strategy (Random): each task vehicle randomly selects a resource vehicle within its one-hop V2V range for computation task offloading.
3) Greedy offloading strategy (Greedy): each task vehicle chooses the resource vehicle that gives it the shortest task completion latency for computation task offloading.

Figure 6 shows the average computation latency of the task versus the number of vehicles. It is observed from Fig. 6 that as the number of vehicles increases, the overall average computation latency of the tasks increases as well. The reason for this is that as the vehicle density increases, more computation task offloading requests follow and computational resources become more strained, thus generating more computation latency. For the same number of vehicles, the lowest average task computation latency is obtained with the FDTO strategy. When the number of vehicles is 150, the FDTO strategy reduces the average task computation latency by 36.1% and 32.3% compared to the random offloading strategy and the greedy offloading strategy, respectively.

Figure 7 presents the average task completion latency as a function of the number of vehicles in the vehicle cluster. As the number of vehicles continues to increase, the average task completion latency tends to increase. This is because the increase in the number of vehicles leads to an increase in computation task offloading requests, thus generating more computational latency, queuing latency, and communication latency. When the number of vehicles is 150, the proposed FDTO strategy reduces the average task completion latency by 24.3% and 17.6% compared to the random offloading strategy and the greedy offloading strategy, respectively. In addition, the data results show that the magnitude of task computation latency is much smaller than that of task completion latency, which indicates that the communication latency during computation offloading has a significant impact on task completion latency. When the number of vehicles increases from 50 to 200, the average completion latency of the Q-learning policy grows from 1.04 times the average completion latency of the proposed FDTO policy to 1.14 times, due to the fact that reducing data interactions through FL can effectively reduce the communication latency.

VI. CONCLUSION

In this paper, we propose an FDTO strategy for computation task offloading. We cluster vehicles based on their mobility to reduce the effect of the highly dynamic changes of the network topology in IoV. The computing task is modeled as a time-dependent directed graph model, and an FDTO strategy is used to make the computation task offloading decision to minimize the task completion latency. The proposed strategy enables the nodes in the vehicle cluster to directly train the local model locally without uploading local data to the cluster head node. The cluster head node only needs to aggregate the local model parameters to update the global model parameters. The simulation shows that the proposed computation task offloading strategy for vehicle clusters has better performance in terms of average task computation latency and average task completion latency. In the future, we will consider using heterogeneous resources such as RSU and other edge servers to provide better computation offloading service for IoV, and jointly consider multiple optimization objectives such as latency and energy consumption.

REFERENCES

[1] X. Xu, H. Li, W. Xu, Z. Liu, L. Yao, and F. Dai, "Artificial intelligence for edge service optimization in Internet of Vehicles: A survey," *Tsinghua Sci. Technol.*, vol. 27, no. 2, pp. 270–287, Apr. 2022.

[2] M. R. Dey, S. Sharma, R. C. Shit, C. P. Meher, and H. K. Pati, "IoV based real-time smart traffic monitoring system for smart cities using augmented reality," in *Proc. Int. Conf. Vis. Towards Emerg. Trends Commun. Netw. (ViTECoN)*, Mar. 2019, pp. 1–6.

[3] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of Vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[4] Y. Li and S. Xu, "Collaborative optimization of edge-cloud computation offloading in Internet of Vehicles," in *Proc. Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2021, pp. 1–6.

[5] 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. 26652–26664, 2019.

[6] A. Boukerchea and R. E. De Grande, "Vehicular cloud computing: Architectures, applications, and mobility," *Comput. Netw.*, vol. 135, pp. 171–189, Apr. 2017.

[7] J. Lee and W. Na, "A survey on vehicular edge computing architectures," *Proc. 13th Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Jeju Island, South Korea, 2022, pp. 2198–2200.

[8] M. Abuelela and S. Olariu, "Taking VANET to the clouds," in *Proc. 8th Int. Conf. Adv. Mobile Comput. Multimedia*, Nov. 2010, pp. 6–13.

[9] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, "Mobile edge computing-enabled Internet of Vehicles: Toward energy-efficient scheduling," *IEEE Netw.*, vol. 33, no. 5, pp. 198–205, Sep. 2019.

[10] C. Wu, X. Chen, T. Yoshinaga, Y. Ji, and Y. Zhang, "Integrating licensed and unlicensed spectrum in the Internet of Vehicles with mobile edge computing," *IEEE Netw.*, vol. 33, no. 4, pp. 48–53, Jul. 2019.

[11] S. Banerjee, C. Chakraborty, and S. Chatterjee, "A survey on IoT based traffic control and prediction mechanism," in *Internet of Things and Big Data Analytics for Smart Generation* (Intelligent Systems Reference Library), vol. 154. Cham, Switzerland: Springer, 2019, pp. 53–75.

[12] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16067–16081, Dec. 2020.

[13] M. Whaiduzzaman, M. Sookhak, A. Gani, and R. Buyya, "A survey on vehicular cloud computing," *J. Netw. Comput. Appl.*, vol. 40, pp. 325–344, Apr. 2014.

[14] J. Gao, Z. Kuang, J. Gao, and L. Zhao, "Joint offloading scheduling and resource allocation in vehicular edge computing: A two layer solution," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3999–4009, Mar. 2023.

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

[16] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Minimizing the delay and cost of computation offloading for vehicular edge computing," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2897–2909, Sep. 2022.

[17] X. Peng, Z. Han, W. Xie, C. Yu, P. Zhu, and J. Xiao, "Deep reinforcement learning for shared offloading strategy in vehicle edge computing," *IEEE Syst. J.*, vol. 17, no. 2, pp. 2089–2100, Jun. 2023, doi: 10.1109/JSYST.2022.3190926.

[18] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, and M. Xiao, "Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transport. Syst.*, early access, Mar. 6, 2023, doi: 10.1109/TITS.2023.3249745.

[19] S. Li, S. Lin, L. Cai, W. Li, and G. Zhu, "Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3384–3398, Mar. 2020.

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

[21] A. J. Kadhim and J. I. Naser, "Proactive load balancing mechanism for fog computing supported by parked vehicles in IoV-SDN," *China Commun.*, vol. 18, no. 2, pp. 271–289, Feb. 2021.

[22] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[23] O. Senouci, S. Harous, and Z. Aliouat, "A new heuristic clustering algorithm based on RSU for Internet of Vehicles," *Arabian J. Sci. Eng.*, vol. 44, no. 11, pp. 9735–9753, Nov. 2019.

[24] G. Liu, N. Qi, J. Chen, C. Dong, and Z. Huang, "Enhancing clustering stability in VANET: A spectral clustering based approach," *China Commun.*, vol. 17, no. 4, pp. 140–151, Apr. 2020.

[25] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, "DRL-based resource allocation for computation offloading in IoV networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 11, pp. 8027–8038, Nov. 2022.

[26] Z. Ning, P. Dong, X. Wang, L. Guo, J. J. P. C. Rodrigues, X. Kong, J. Huang, and R. Y. K. Kwok, "Deep reinforcement learning for intelligent Internet of Vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cognit. Commun. Netw.*, vol. 5, no. 4, pp. 1060–1072, Dec. 2019.

[27] Z. Wei, B. Li, R. Zhang, X. Cheng, and L. Yang, "Dynamic many-to-many task offloading in vehicular fog computing: A multi-agent DRL approach," in *Proc. IEEE Global Commun. Conf.*, Riode Janeiro, Brazil, Dec. 2022, pp. 6301–6306.

[28] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication efficient learning of deep networks from decentralized data," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, vol. 54, 2017, pp. 1273–1282.

[29] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, Apr. 2017, vol. 54, pp. 1–10.

[30] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, and V. Ivanov, "Towards federated learning at scale: System design," in *Proc. Mach. Learn. Syst.*, vol. 1, 2019, pp. 374–388.

[31] L. Liu, Y. Tian, C. Chakraborty, J. Feng, Q. Pei, L. Zhen, and K. Yu, "Multilevel federated learning-based intelligent traffic flow forecasting for transportation network management," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 2, pp. 1446–1458, Jun. 2023.

[32] Z. Xu, Y. Guo, C. Chakraborty, Q. Hua, S. Chen, and K. Yu, "A simple federated learning-based scheme for security enhancement over Internet of Medical Things," *IEEE J. Biomed. Health Informat.*, vol. 27, no. 2, pp. 652–663, Feb. 2023.

[33] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 2031–2063, 3rd Quart., 2020.

[34] R. P. Roess, E. S. Prassas, and W. R. McShane, *Traffic Engineering*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2004.

[35] S. Yousefi, E. Altman, R. El-Azouzi, and M. Fathy, "Analytical model for connectivity in vehicular ad hoc networks," *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3341–3356, Nov. 2008.

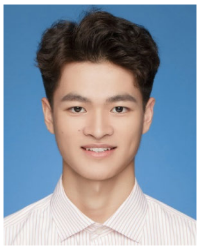
[36] N. Wisitpongphan, F. Bai, P. Mudalige, V. Sadekar, and O. Tonguz, "Routing in sparse vehicular ad hoc wireless networks," *IEEE J. Sel. Areas Commun*, vol. 25, no. 8, pp. 1538–1556, Oct. 2007.

[37] C.-C. Lin, D.-J. Deng, and C.-C. Yao, "Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3692–3700, Oct. 2018.

[38] R. Sarumathi and V. Jayalakshmi, "Study of clustering schemes in mobile ad hoc networks," in *Proc. 6th Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Madurai, India, May 2022, pp. 694–700.

WENHUI YE received the B.S. degree from Fuzhou University, Fuzhou, China, in 2018. She is currently pursuing the Graduate degree with the School of Informatics, Xiamen University, China. Her research interests include vehicle networking communications, 5G networks, and flying ad-hoc networks.

KE ZHENG received the B.E. degree from the Changsha University of Science and Technology, Changsha, China, in 2019, and the M.S. degree from the Department of Informatics and Communication Engineering, Xiamen University, in 2022. Her research interests include vehicle networking communications, NOMA, and 5G networks.



YUANYU WANG received the B.S. degree in communication engineering from the Wuhan University of Technology, Wuhan, China, in 2017. He is currently pursuing the Graduate degree with the School of Informatics, Xiamen University, China. His research interests include vehicular ad-hoc networks, space-air-ground integrated networks, and flying ad-hoc networks.

YULIANG TANG (Member, IEEE) received the M.S. degree from the Beijing University of Posts and Telecommunications, China, in 1996, and the Ph.D. degree in informatics and communication engineering from Xiamen University, in 2009. He is currently a Professor with the Department of Information and Communication Engineering. He has published more than 90 papers in journals and international conferences and has been granted over 20 patents in his research areas. His research interests include wireless communication, 5G and beyond, and vehicular ad-hoc networks.