Federated MADDPG-Based Collaborative Scheduling Strategy in Vehicular Edge Computing

Songxin Lei, Huijun Tang, *Member, IEEE*, Chuangyi Li, Xueying Zhang, Chenli Xu, and Huaming Wu, *Senior Member, IEEE*

***Abstract*—In Vehicular Edge Computing (VEC), vehicles offload computational tasks to Roadside Units (RSUs) equipped with edge servers to achieve efficient processing. Considering that vehicles switch connections between RSUs during high-speed movement, obtaining the state information of other RSUs is crucial for achieving global collaborative decision-making. However, frequent sharing of RSUs' state data during the training of scheduling models may result in privacy leakage risks. To address this issue, we federally train a joint scheduling model for task offloading and resource allocation without the need for state sharing among RSUs. We prove that the proposed task offloading problem influenced by resource allocation is a strict multi-node non-cooperative potential game problem, and use the potential function as the reward function for Multi-Agent Deep Deterministic Policy Gradient (MADDPG). Finally, we propose the Fed-MADDPG algorithm to find the equilibrium point of task offloading and apply the gradient descent method and the Lagrange multiplier method to maximize the average task completion rate among RSUs under constraints, ensuring the framework has optimal computational and transmission performance. We conduct simulation experiments using real-world datasets, and the results show that this method has superior performance compared to previous approaches.**

***Index Terms*—Vehicular edge computing, joint scheduling model, multi-agent deep reinforcement learning, federated learning.**

I. Introduction

THE Internet of Vehicles (IoV), a crucial component of intelligent transportation systems, connects vehicles, Roadside Units (RSUs) and the cloud to form a collaborative network that allows intelligent decision-making [1], [2], [3]. However, the increasing complexity and latency requirements of IoV applications demand more efficient computational capabilities. This need has led to the emergence of Vehicular Edge Computing (VEC), where vehicles offload computational tasks to RSUs equipped with edge servers, enabling efficient task processing closer to the source [3], [4], [5].

Task offloading allows devices to offload computing-intensive tasks to the edge efficiently. Wu et al. [6] offloaded computation-intensive and delay-sensitive tasks to minimize long-term energy consumption in Mobile Edge Computing (MEC) environments of limited battery capacity. Wang et al. [7] introduced a multiuser non-cooperative computation offloading game for MEC environments and defined the vehicle's utility functions to capture its actual benefits. In VEC, vehicles offload tasks to RSUs or volunteer vehicles [8], [9], [10]. Because of the limited storage and network bandwidth resources at edge nodes, several studies explored joint scheduling strategies, aiming to coordinate offloading decisions with resource allocation to boost overall system efficiency. Sun et al. [11] proposed a hybrid intelligent optimization algorithm to reduce task delay and resource consumption in VEC scenarios. Tran et al. [12] addressed joint task offloading and resource allocation in multi-cell networks, offering an efficient decomposition method to improve offloading utility. Joint scheduling of task offloading and resource allocation enables the efficient utilization of limited computational and communication resources in VEC systems, thereby improving overall system performance.

In VEC environments, different devices can collect heterogeneous and complementary information from their local environments. Effectively leveraging such diverse information sources enables efficient joint scheduling [13], [14]. Specifically, Jeremiah et al. [15] propose a digital twin-assisted collaborative training framework in which RSUs coordinate task offloading and workload sharing to achieve optimal scheduling. Xue et al. [16] designed a collaborative multi-agent reinforcement learning framework with inter-RSU task migration to address the instability of task returns in dynamic VEC environments. Shinde et al. [17] propose collaborative Q-learning methods that enable vehicles to share local information, thereby improving joint network selection and computation offloading decisions in heterogeneous multi-service IoV environments. However, the frequent exchange of sensitive information to collaboratively train a scheduling model poses significant privacy and security risks.

Some studies have explored the challenges and solutions for preserving privacy and security during information

Received 28 December 2024; revised 6 July 2025; accepted 16 July 2025. Date of publication 18 July 2025; date of current version 3 December 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62401190 and Grant 62071327, in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LQN25F020016, and in part by the Engineering and Physical Sciences Research Council under Grant EP/X040518/1 and Grant EP/Y037421/1. Recommended for acceptance by D. Xu. *(Songxin Lei and Huijun Tang contributed equally to this work.) (Corresponding author: Huaming Wu.)*

Songxin Lei, Chuangyi Li, and Chenli Xu are with the School of Mathematics, Tianjin University, Tianjin 300350, China (e-mail: lei_songxin@tju.edu.cn).

Huijun Tang is with the Department of Engineering, Durham University, DH1 3LE Durham, U.K. (e-mail: huijun.tang@durham.ac.uk).

Xueying Zhang and Huaming Wu are with the Center for Applied Mathematics, Tianjin University, Tianjin 300072, China (e-mail: zhang_xueying@tju.edu.cn; whming@tju.edu.cn).

Digital Object Identifier 10.1109/TMC.2025.3590747

sharing in collaborative training. Onieva et al. [18] surveyed security vulnerabilities in VEC systems and proposed a layered security framework combining authentication and intrusion detection to safeguard information exchange during collaborative operations. Kang et al. [19] leveraged consortium blockchain and smart contract technologies to enable secure and efficient data storage and sharing in VEC, ensuring privacy during collaborative information exchange among vehicles. However, existing studies have not placed sufficient emphasis on protecting the privacy of RSU state information [20], while direct sharing of state information among RSUs is prone to some cyberattacks under heterogeneous communication technologies.

To address the challenge, we propose an innovative joint optimization method for task offloading and resource allocation based on federated MADDPG to maintain the highest possible task completion rate without directly sharing state information among RSUs. The main contributions of our work are as follows:

- *Federated collaborative VEC scheduling model:* To train a collaborative scheduling model without directly sharing RSU state information, we propose a federated scheduling method to maintain the system's task completion rate, which jointly optimizes task offloading and resource allocation. By federated training, the state information of each RSU remains local to prevent information leakage caused by the frequent sharing of RSU states.
- *Theoretical proof of the task offloading and resource allocation modules:* Based on the proposed federated collaborative model, We construct the global potential function to capture the overall payoff dynamics on all RSUs, prove that the task offloading problem influenced by resource allocation is a multi-node non-cooperative potential game, and further prove it has at least one pure-strategy nash equilibrium, which ensures the convergence under the federated collaborative training. In addition, we prove the proposed resource allocation problem is a convex optimization problem.
- *BCD iterative scheduling optimization framework based on the Fed-MADDPG algorithm and KKT method:* Based on the theoretical proof, we utilize the Gause-Seidel-type Block Coordinate Descent (BCD) framework that iteratively solves the resource allocation module-based Karush-Kuhn-Tucker (KKT) method and federated task offloading module based on federated MADDPG, which utilizes the potential function as the reward function.
- *Effective performance on real-world data:* We conduct experiments using real-world vehicle trajectory data, and the results validate the effectiveness of our proposed method, showing significant improvements in joint scheduling efficiency.

The rest of the paper is organized as follows. Related works are investigated in Section II. Section III describes the system model and problem formulation. The proposed Fed-MADDPG algorithm is in Section IV. Experiments are presented in Section V, followed by concluding remarks in Section VI.

II. Related Work

A. Task Offloading and Resource Allocation in IoV

Optimization theory has been widely applied in the joint task offloading and resource allocation problem in IoV scenarios. These approaches aim to improve system performance by addressing challenges such as computational complexity, vehicle mobility, and resource constraints. Tan et al. [23] decomposed the joint optimization problem into task offloading and resource allocation subproblems, using dual decomposition for decentralized, near-optimal solutions. Zhang et al. [24] addressed the challenges of high vehicle mobility and dynamic networks by formulating the problem as a Mixed Integer Non-Linear Programming (MINLP). A matching-based algorithm was proposed to optimize system utility related to delay and computing costs, demonstrating competitive performance through simulations. Tang et al. [32] focused on optimizing Deep Neural Network (DNN) partitioning and resource allocation in MEC, proposing an iterative alternating optimization algorithm that achieves optimal solutions in polynomial time. Xiao et al. [25] focused on FL within VEC, highlighting the need to select appropriate vehicles for learning tasks due to energy constraints and varying data quality. The problem was modeled as a min-max optimization and solved using a greedy algorithm and Lagrangian dual method, achieving a balance between cost and fairness. However, optimization theory-based methods often struggle to address the interactive nature of IoV environments. To overcome these limitations, game theory has emerged as an alternative method for modeling multi-agent interactions and resource competition in IoV systems.

Game theory has also been extensively applied to the joint optimization of task offloading and resource allocation in IoV systems, offering effective solutions for multi-agent interactions and resource competition. Wang et al. [7] proposed a non-cooperative game for task offloading in Multi-access Edge Computing networks, where each vehicle independently adjusts its offloading strategy to maximize its utility. The proposed algorithm converges to a stable equilibrium, optimizing performance under dynamic network conditions. Fan et al. [31] addressed load imbalance issues in multi-RSU VEC scenarios using an Exact Potential Game (EPG). The study introduced a two-stage iterative algorithm to minimize task processing delays, enabling cooperation between overloaded and underloaded RSUs. Xu et al. [30] modeled the task offloading process as an exact potential game within a NOMA-based VEC architecture. The authors proposed a multi-agent deep reinforcement learning approach to achieve Nash equilibrium, optimizing service ratios by jointly managing intra- and inter-edge resources. However, game theory-based methods often rely on static models, which limit their adaptability to highly dynamic IoV environments.

Reinforcement learning has emerged as a powerful approach for dynamic scheduling in VEC. These methods adapt well to dynamic environments and complex decision-making processes. Liu et al. [27] introduced a vehicle-assisted offloading scheme, leveraging Q-learning and Deep Reinforcement Learning (DRL) to maximize long-term utility in dynamic VEC networks. The proposed method optimizes task offloading and resource allocation considering stochastic traffic and varying communication conditions. Ju et al. [26] focused on secure offloading in multi-user VEC networks, where a DRL-based SORA scheme was developed to jointly optimize transmit power, spectrum access, and resource allocation. This approach improved system delay performance and secured wireless offloading against eavesdropping, ensuring reliable Vehicle-to-Vehicle (V2V) communication. Ning et al. [28] combined DRL with task scheduling in VEC. It used a two-sided matching scheme and an enhanced DRL algorithm to optimize task scheduling and resource allocation, maximizing user Quality of Experience (QoE). However, dynamic interactions lead to frequent connection changes among devices, especially in the training process of the scheduling model, which requires state information sharing for coordination and poses risks to privacy.

B. Federated Learning in Vehicular Networks

Federated Learning (FL) has emerged as a transformative approach in Vehicular Networks (VNs), addressing key challenges such as data privacy, scalability, and heterogeneity in the IoV [33]. FL has been applied to domains including traffic prediction, autonomous driving, and vehicular communication optimization. For instance, Yuan et al. [34] proposed an FL framework for traffic state estimation to enhance road planning decisions while preserving driver privacy. By integrating LSTM models with DRL, the approach enhances prediction accuracy while effectively balancing computation and communication costs. Yu et al. [35] introduced a mobility-aware proactive caching scheme in vehicular networks. Using FL and an adversarial autoencoder, it predicts content popularity and dynamically manages edge caching, enhancing cache performance and reducing communication costs.

However, integrating FL in vehicular networks introduces unique challenges [36], such as heterogeneous resources, intermittent connectivity and high communication overhead. Xu et al. [37] addressed the communication overhead in cloud-based federated learning by proposing a hierarchical FL system. It optimizes edge aggregation intervals and resource allocation, reducing training latency and improving learning performance through iterative optimization. Guo et al. [38] developed a federated edge learning scheme, featuring a residual feedback mechanism and V2V communication to maximize gradient utilization. Lyapunov optimization enhances uploading efficiency, improving learning performance under energy constraints.

C. Comparison of Selected Related Studies

Our work introduces a unified framework that jointly optimizes task offloading and resource allocation, effectively addressing their interdependence. We model task offloading as a non-cooperative potential game and solve it using the MADDPG algorithm, ensuring decentralized decision-making. The entire process is dynamic and iterative, with task offloading decisions feeding into the resource allocation module, which is optimized via nonlinear convex optimization. Unlike existing static methods, our approach can dynamically generate decisions

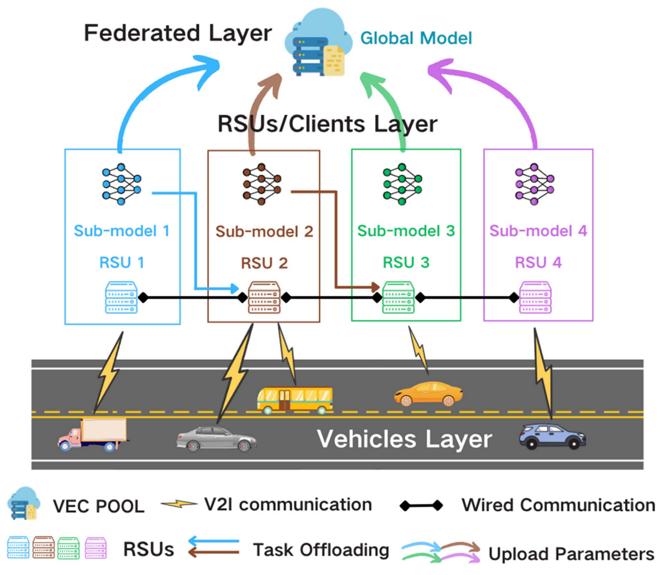


Fig. 1. System model. RSUs operate as clients, while the VEC POOL functions as a central server coordinating interactions between all clients.

to adapt to the highly dynamic nature of vehicular networks. At the same time, we introduce federated learning to enable collaborative training across vehicles, addressing the data silo problem. Table I provides a detailed comparison of our study with existing research.

III. System Model

A. System Architectures

As depicted in Fig. 1, RSUs embedded with computational resources serve as critical infrastructure in the VEC framework. Vehicles in motion delegate their computational tasks to these RSUs through Vehicle-to-Infrastructure (V2I) communication links. During each iteration, following the completion of local model training, RSUs transmit their policy network parameters to the VEC POOL. This central pool aggregates the received data and redistributes the refined parameters back to the RSUs, initiating the next training phase. This cycle repeats iteratively until model convergence is achieved. The RSUs are interconnected by a wired network, allowing them to manage tasks locally or redistribute them to other nodes as necessary [30].

Within a specified interval, each vehicle offloads tasks to exactly one RSU, whereas an RSU concurrently processes tasks from multiple vehicles if computational capacity permits. Consider a set of discrete time steps denoted by $\mathcal{T} = \{1, \ldots, t, \ldots, T\}$ and a set of vehicles $\mathcal{V} = \{1, \ldots, v, \ldots, V\}$. At time t, a computational task requested by vehicle v is represented by $k_v^t = (d_k, c_k, t_k)$, where d_k, c_k, and t_k indicate the data size, CPU cycle requirement, and deadline of the task, respectively. The set of RSUs is denoted as $\mathcal{E} = \{1, \ldots, e, \ldots, E\}$, with each RSU e characterized by its computational frequency c_e, V2I communication range u_e, and physical location s_e. The distance between vehicle v and RSU e at time t is denoted as $d_{v,e}^t$, and the set of vehicles within RSU e's communication range at time t is given by V_e^t. Consequently,

TABLE I
COMPARISON OF SELECTED RELATED STUDIES IN VEC SCENARIOS

Reference	Joint Scheduling	Dynamic Decision	Multi-RSU Collaboration	Federated Training	Method	Objective
Zhan *et al.* [21]	✗	✓	✗	✗	PPO	Energy + Delay
Li *et al.* [22]	✗	✓	✗	✓	Dueling Double Deep Q-learning	Transmit rates
Tan *et al.* [23]	✓	✗	✗	✗	Decentralized Convex Optimization	Energy + Delay
Zhang *et al.* [24]	✓	✗	✗	✗	MINLP	Cost + Delay
Xiao *et al.* [25]	✓	✗	✗	✓	Nonlinear Programming	Cost
Ju *et al.* [26]	✓	✓	✗	✗	Double Deep Q-learning	Secrecy + Delay
Liu *et al.* [27]	✓	✓	✗	✗	Q-Learning	VEC Network utility
Ning *et al.* [28]	✓	✓	✗	✗	Double Deep Q-Network	Experience quality
Zhao *et al.* [29]	✓	✓	✗	✗	Stackelberg Game	Vehicle utility
Xu *et al.* [30]	✓	✓	✓	✗	EPG + Convex Optimization	Task completion rates
Fan *et al.* [31]	✓	✓	✓	✗	EPG	Delay
Ours	✓	✓	✓	✓	**Fed MADDPG + KKT**	**Task completion rates**

TABLE II
A LIST OF THE MAIN NOTATIONS

Notations	**Description**
k_v^t	Task requested by vehicle v
$q_{v,e}^t$	Whether task k_v^t needs to be offloaded to RSU e (binary)
$c_{v,e}^t$	Computing resources allocated by RSU e to vehicle v
$d_{v,e}^t$	Distance between vehicle v and RSU e
$u_{v,e}^t$	Transmission time of training data from vehicle v to RSU e
$x_{v,e}^t$	Execution time of task k_v^t in RSU e
$w_{v,e}^t$	Wired task transmission time sent from node e' to RSU e
$n_{v,e}^t$	Task processing time of task k_v^t at RSU e
$\psi_{v,e}^t$	Service time for task k_v^t at RSU e
$\mathcal{T}$	Set of time points
$\mathcal{V}$	Set of vehicles
K_v	Set of tasks requested by vehicle v
$\mathcal{E}$	Set of RSUs
Q	Set of task offloading strategies
Q^t	Set of task offloading strategies at time t
C	Set of resource allocation strategies
d_k	Data size of task k
c_k	CPU cycle frequency of task k
t_k	Deadline of task k
c_e	Computation frequency of RSU e
u_e	V2I communication range of RSU e
s_e	Location of RSU e
ψ_e^t	The task completion rate of RSU e
V_e^t	Set of vehicles within the coverage area of RSU e
$Dis_{V_e^t}$	Set of distance between each vehicle v in V_e^t and RSU e;
K_e^t	Set of uploaded tasks within the coverage area of RSU e
$D_{K_e^t}$	Set of data size of each task k_v^t in K_e^t;
$T_{K_e^t}$	Set of deadline of each task k_v^t in K_e^t.
$K_{q_e}^t$	Set of tasks offloaded to RSU e at time t
$C_{K_e^t}$	Set of CPU cycle frequency;
B_e^t	Upload bandwidth of RSU e at time t
$D_{v,e}^t$	Data size of d_k to be transferred from vehicle v to RSU e

K_e^t represents the set of tasks uploaded by vehicles within V_e^t. Table II summarizes the key notations.

B. Task Offloading Model

Given the disparity in computational power between local devices and RSUs, it is assumed that tasks are by default processed at RSUs. Let B_e^t represent the upload bandwidth of RSU e at time t, and $D_{v,e}^t$ denote the data size of d_k transferred from vehicle v to RSU e, where $v \in V_e^t$. The transmission duration is expressed as:

$$u_{v,e}^t = \frac{D_{v,e}^t}{B_e^t}. \tag{1}$$

The task offloading decision is indicated by $q_{v,e}^t$, which reflects whether the task from vehicle v at time t is offloaded to RSU e. This indicator can take values from the set $\{0, 1\}$. Each task is designated to a single RSU, ensuring $\sum_{e\in\mathcal{E}} q_{v,e}^t = 1$ for all vehicles v. Hence, the collection of tasks offloaded to RSU e is defined as:

$$K_{q_e}^t = \{k_v^t \mid q_{v,e}^t = 1, v \in V_e^t\}. \tag{2}$$

The computational resources (CPU clock cycles) allocated by RSU e for task $k_v^t \in K_{q_e}^t$ are denoted by $c_{v,e}^t$. The total computational resources allocated must not exceed the processing capacity of RSU e, i.e., $\sum_{k_v^t\in K_{q_e}^t} c_{v,e}^t \leqslant c_e$, where c_e denotes the CPU clock frequency of RSU e. Based on the model convergence accuracy analysis under the FL framework in Section III-B2, the execution duration of task k_v^t at RSU e is calculated by:

$$x_{v,e}^t = \frac{d_k c_k}{c_{v,e}^t}, \tag{3}$$

where c_k is the CPU cycles for one data unit of task k_v^t.

Considering task migration over wired connections between RSUs, $w_{v,e}^t$ represents the wired transmission duration from RSU e' to RSU e. Following the definition in [30] and supported by the cornerstone references [39] and [40], the delay is modeled as:

$$w_{v,e}^t = \begin{cases} 0, & k_v^t \in K_e^t \cap K_{q_e}^t, \\ \frac{\zeta d_k dis_{e,e'}^t}{z}, & k_v^t \in K_e^t \cap K_{q_{e'}}^t, \end{cases} \tag{4}$$

where z denotes the transmission rate, $dis_{e,e'}^t$ is the distance between RSU e and e', and ζ represents a discount factor. The overall processing time of k_v^t at RSU e combines execution and wired transmission times, denoted as $n_{v,e}^t$, contingent on the task offloading decision and defined as:

$$n_{v,e}^t = w_{v,e}^t + \sum_{e'\in\mathcal{E}} q_{v,e'}^t x_{v,e'}^t. \tag{5}$$

The total service time $\psi_{v,e}^t$ for task $k_v^t \in K_e^t$ is the sum of the upload and processing durations, expressed as:

$$\psi_{v,e}^t = u_{v,e}^t + n_{v,e}^t. \tag{6}$$

C. Problem Formulation

Due to the constrained computational resources available in vehicles, task offloading to RSUs is maximized within limited time frames to leverage their computing capabilities. Offloading tasks to RSUs with different CPU frequencies results in varying probabilities of meeting deadlines. Consequently, optimizing scheduling strategies is crucial to enhance the likelihood of tasks being completed on time.

A task is successfully executed if its service time does not exceed the deadline t_k, which is purposefully shorter than the interval between successive time slots. This design ensures that external factors, such as network conditions and resource distribution, remain stable throughout task execution.

The task completion rate at RSU e is defined as the ratio of successfully completed tasks to the total tasks requested, expressed as ψ_e^t:

$$\psi_e^t = \frac{\sum_{k_v^t \in K_e^t}}{I\left\{\psi_{v,e}^t \leqslant t_k\right\}} |K_e^t|. \tag{7}$$

where $|K_e^t|$ denotes the total number of tasks within RSU e's coverage area, and $I\{\psi_{v,e}^t \leqslant t_k\}$ is an indicator function.

A solution (Q, C) consists of the task offloading strategy Q and the computational resource allocation strategy C, formulated as:

$$\begin{cases} Q = \left\{q_{v,e}^t \mid \forall v \in V_e^t, \forall e \in \mathcal{E}, \forall t \in \mathcal{T}\right\}, \\ C = \left\{c_{v,e}^t \mid \forall v \in V_e^t, \forall e \in \mathcal{E}, \forall t \in \mathcal{T}\right\}. \end{cases} \tag{8}$$

The goal is to maximize the cumulative task completion rate across RSUs by jointly scheduling offloading and resource allocation strategies. The joint scheduling problem can be expressed as:

$$\mathcal{P}: \max_{Q,C} f_1 = \sum_{t \in T} \sum_{e \in \mathcal{E}} \psi_e^t \tag{9}$$

$$\text{s.t. } C_1: \sum_{k_v^t \in K_{qe}^t} c_{v,e}^t \leqslant c_e, \forall e \in \mathcal{E}, \forall t \in \mathcal{T}, \tag{9a}$$

$$C_2: q_{v,e}^t \in \{0,1\}, \forall v \in V, \forall e \in \mathcal{E}, \forall t \in \mathcal{T}, \tag{9b}$$

$$C_3: \sum_{e \in \mathcal{E}} q_{v,e}^t = 1, \forall v \in V, \forall t \in \mathcal{T}, \tag{9c}$$

where constraint C_1 ensures that the allocated computational resources remain within the RSU's capacity. Constraints C_2 and C_3 enforce binary task offloading decisions and guarantee that each task is assigned to exactly one RSU.

Theorem 1: The joint scheduling problem $\mathcal{P}$ is NP-hard.

Proof: To prove that problem $\mathcal{P}$ is NP-hard, we reduce a known NP-hard problem, the *Knapsack Problem* [41], to a special case of $\mathcal{P}$.

Consider a special case of problem $\mathcal{P}$ where: *a)* There is a single RSU e, i.e., $|\mathcal{E}| = 1$, and a single time slot, i.e., $|\mathcal{T}| = 1$. *b)* The tasks k_v^t correspond to items in the Knapsack Problem, each with an associated weight and value. *c)* The allocated computational resource $c_{v,e}^t$ for each task maps to the weight w_i of item i. *d)* The task completion rate ψ_e^t corresponds to the total value v_i of the selected items. Under a single RSU and single time slot, ψ_e^t becomes a direct sum of the completion values for tasks chosen to be offloaded. *e)* The computational capacity of the RSU c_e corresponds to the knapsack capacity W. Under these conditions, constraints C_2 and C_3 imply a 0-1 choice of tasks, and $\mathcal{P}$ reduces to selecting a subset of tasks whose total resource usage does not exceed c_e, while maximizing ψ_e^t. This is exactly the Knapsack Problem, which is known to be NP-hard.

Therefore, if there were a polynomial-time algorithm for solving $\mathcal{P}$, we could apply it to solve this Knapsack Problem, which is a special case of $\mathcal{P}$, in polynomial time as well. Since the Knapsack Problem is NP-hard, this implies that $\mathcal{P}$ must also be NP-hard. ■

This proof demonstrates that the joint scheduling problem $\mathcal{P}$ is computationally intractable, highlighting the necessity of advanced algorithmic solutions.

IV. Proposed Solutions

We adopt a Gauss-Seidel-type BCD approach to solve the joint scheduling model, which iteratively optimizes task offloading and computational resource allocation in an alternating manner. The solution framework is as follows: First, we model the task offloading problem as a non-cooperative potential game among RSUs, demonstrating that the dynamic offloading process over time steps forms a Markov Decision Process (MDP). Leveraging the MADDPG algorithm, we iteratively derive optimal task offloading decisions for each RSU, which dynamically adapt to the current system state. Second, within a federated learning framework, we aggregate, validate, and distribute the policy network parameters of each RSU to enable the global sharing of heterogeneous data, ensuring collaborative optimization across RSUs until the global model converges. Importantly, the task offloading decisions incorporate variables related to computational resources, which are treated as dynamic inputs during the iterative optimization.

Following the optimization of task offloading decisions, we solve the resource allocation problem as a nonlinear convex optimization, taking the globally optimized task offloading results as inputs. By applying the Karush-Kuhn-Tucker (KKT) conditions from the Lagrange multiplier method, we derive the optimal computational resource allocation strategy based on the current observed state and propagate these results back into the system. The entire dynamic iterative process embodies the principles of the Gauss-Seidel-type BCD approach. The solution process is illustrated in Fig. 2.

Based on the Gauss-Seidel-type Block Coordinate Descent iterative optimization approach, we optimize two interdependent components in an alternating manner. Specifically, at each time step, the problem is addressed by iteratively solving the task offloading problem and the resource allocation problem, ensuring that both are effectively refined within the framework.

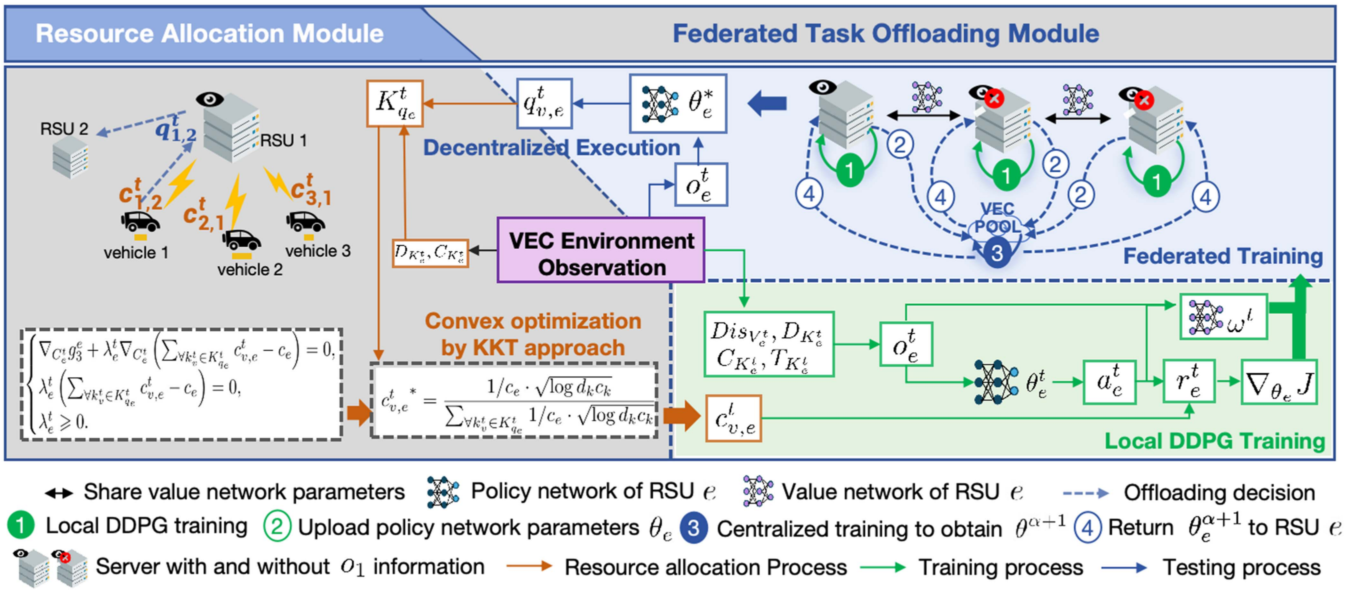

Resource Allocation Module
Federated Task Offloading Module
RSU 2
RSU 1
vehicle 1
vehicle 2
vehicle 3
Decentralized Execution
VEC Environment Observation
Federated Training
VEC POOL
Convex optimization by KKT approach
Local DDPG Training
Share value network parameters
Policy network of RSU e
Value network of RSU e
Offloading decision
Local DDPG training
Upload policy network parameters θ_e
Centralized training to obtain $\theta^{\alpha+1}$
Return $\theta_e^{\alpha+1}$ to RSU e
Server with and without o_1 information
Resource allocation Process
Training process
Testing process

Fig. 2. The overall architecture of the proposed Fed-MADDPG algorithm. Taking RSU 1 as an example, the offloading decisions of RSU 1 are input to the resource allocation module, while computational resource allocation dynamically influences the task offloading environment in the federated task offloading module.

A. Potential Game-Based Task Offloading

The first component, related to Q^t, which refers to the set of task offloading strategies at time t, involves determining the task offloading decisions for RSUs. Its formulation can be expressed as:

$$\mathcal{P}_1: \max_{Q^t} g_1 = \sum_{\forall e \in \mathcal{E}} \psi_e^t \tag{10}$$

$$\text{s.t. } C_4: q_{v,e}^t \in \{0,1\}, \forall v \in \mathcal{V}, \forall e \in \mathcal{E}, \tag{10a}$$

$$C_5: \sum_{e \in \mathcal{E}} q_{v,e}^t = 1, \forall v \in \mathcal{V}. \tag{10b}$$

Then, $\mathcal{P}_1$ is modeled as a non-cooperative game among RSUs, where each RSU independently determines its task offloading strategy. The game is represented as $G = \{\mathcal{E}, \mathcal{S}, \{U_e\}_{e\in\mathcal{E}}\}$, where: *a)* $\mathcal{E}$ is the set of players (RSUs). *b)* $\mathcal{S} = S_1 \times S_2 \times \cdots \times S_E$ is the strategy space of the game, with S_e being the set of all possible strategies for RSU e. *c)* U_e is the utility function of RSU e.

Each strategy profile $\zeta \in \mathcal{S}$ is a tuple $\zeta = (\zeta_1, \zeta_2, \ldots, \zeta_E)$, which can also be written as $\zeta = (\zeta_e, \zeta_{-e})$, where $\zeta_{-e} \in S_{-e}$ represents the combined strategies of all RSUs except e and $S_{-e} = S_1 \times \cdots S_{e-1} \times S_{e+1} \cdots \times S_E$ is the strategy space formed by all RSUs except the RSU e. $U_e(\zeta)$ represents the utility function of RSU e, defined as follows:

Definition 1: **Utility function of RSU** e It is defined as the task completion rate of RSU e under strategy ζ, i.e.,

$$U_e(\zeta) = \psi_e^t.$$

Definition 2: A game $G = \{\mathcal{E}, \mathcal{S}, \{U_e\}_{e\in\mathcal{E}}\}$ is a **potential game** if there exists a function $F: \mathcal{S} \to \mathbb{R}$ such that, for all $e \in \mathcal{E}$, all $\zeta_{-e} \in S_{-e}$, and all $\zeta_e, \zeta_e' \in S_e$,

$$U_e(\zeta_e, \zeta_{-e}) - U_e(\zeta_e', \zeta_{-e}) = F(\zeta_e, \zeta_{-e}) - F(\zeta_e', \zeta_{-e}).$$

Theorem 2: The game $G = \{\mathcal{E}, \mathcal{S}, \{U_e\}_{e\in\mathcal{E}}\}$ is a potential game with the potential function:

$$F(\zeta) = \sum_{e\in\mathcal{E}} \psi_e^t.$$

Proof: The potential function $F(\zeta)$ for the system is defined as the sum of RSUs' utility functions (task completion rates):

$$F(\zeta) = \sum_{e\in\mathcal{E}} U_e(\zeta) = \sum_{e\in\mathcal{E}} \psi_e^t.$$

Thus:

$$F(\zeta) = \sum_{e\in\mathcal{E}} \frac{\sum_{k_v^t \in K_e^t} I\left\{\frac{D_k}{B_e^t} + w_{v,e}^t + \sum\limits_{e'\in\mathcal{E}} q_{v,e'}^t x_{v,e'}^t \le t_k\right\}}{|K_e^t|}.$$

Let $\zeta = (\zeta_e, \zeta_{-e})$ represent the initial strategy and $\zeta' = (\zeta_e', \zeta_{-e})$ represent the strategy where only RSU e has changed strategy from ζ_e to ζ_e'. Then, we analyze how changing the strategy from ζ_e to ζ_e' impacts both $U_e(\zeta)$ and $F(\zeta)$ in various cases. Specifically, whether task k_v meets the deadline at RSU e or another RSU e' will determine these changes.

According to the constraint C_5, we analyze each possible scenario for how the indicator function $I\{\psi_{v,e}^t \le t_k\}$ changes under this strategy change:

- *Case 1:* Assume that under strategy ζ_e, task k_v is completed within the deadline t_k (i.e., $\psi_{v,e}^t \le t_k$). If changing to ζ_e' causes $\psi_{v,e}^t$ to exceed t_k, the indicator function changes from 1 to 0, reducing $U_e(\zeta)$ and $F(\zeta)$.
- *Case 2:* Assume that under strategy ζ_e, task k_v fails to meet the deadline t_k (i.e., $\psi_{v,e}^t > t_k$), so $I\{\psi_{v,e}^t \le t_k\} = 0$. If strategy ζ_e' leads to $\psi_{v,e}^t \le t_k$, the indicator changes from 0 to 1, increasing both $U_e(\zeta)$ and $F(\zeta)$.

$$U_e(\zeta_e,\zeta_{-e})-U_e(\zeta_e',\zeta_{-e})=\psi_e^t(\zeta_e,\zeta_{-e})-\psi_e^t(\zeta_e',\zeta_{-e})$$
$$=\frac{\sum_{k_v^t\in K_e^t} I\left\{\frac{D_k}{B_e^t}+w_{v,e}^t+\sum_{e'\in\mathcal{E}} q_{v,e'}^t x_{v,e'}^t\le t_k\right\}}{|K_e^t|}$$
$$-\frac{\sum_{k_v^t{}'\in K_e^t{}'} I\left\{\frac{D_k}{B_e^t}+w_{v,e}^t{}'+\sum_{e'\in\mathcal{E}} q_{v,e'}^t x_{v,e'}^t\le t_k\right\}}{|K_e^t|}.$$

Based on the above cases, we can mathematically express the change in the utility function $U_e(\zeta)$ as follows:

- *Case 1:* $U_e(\zeta_e,\zeta_{-e})-U_e(\zeta_e',\zeta_{-e})=-\frac{1}{|K_e^t|}$.
- *Case 2:* $U_e(\zeta_e,\zeta_{-e})-U_e(\zeta_e',\zeta_{-e})=\frac{1}{|K_e^t|}$.

Similarly, the change takes the same form for the potential function $F(\zeta)$. Since only RSU e's strategy changes, the change in $F(\zeta)$ is given by:

$$F(\zeta_e,\zeta_{-e})-F(\zeta_e',\zeta_{-e})=U_e(\zeta_e,\zeta_{-e})-U_e(\zeta_e',\zeta_{-e}).$$

■

Theorem 3: Every finite potential game has at least one pure-strategy Nash equilibrium.

Proof: Since G is a finite potential game and the potential function $F(\zeta)$ is bounded over the finite strategy space $\mathcal{S}$, there exists a strategy profile ζ^* that maximizes F:

$$\zeta^*=\arg\max_{\zeta\in\mathcal{S}} F(\zeta).$$

At ζ^*, no single RSU e can unilaterally change its strategy to increase its utility, because such a change would increase F, contradicting the maximality of $F(\zeta^*)$. Therefore, ζ^* is a pure-strategy Nash equilibrium. ■

Our potential game formulation ensures alignment between the system optimum and the potential function. This implies that decentralized RSU utility maximization inherently achieves optimality and crucially, and information sharing introduces no decision bias. Motivated by these properties, we hypothesize that strategic coordination within a decentralized framework can enhance offloading. This insight motivates our integration of FL into the MARL framework, in which RSUs periodically share their policy network parameters during the task offloading process while preserving privacy.

B. KKT-Based Resource Allocation

C^t involves the allocation of computational resources, and the resource allocation problem is described as follows:

$$\boldsymbol{\mathcal{P}_2}:\ \min_{C^t} g_2=\sum_{e\in\mathcal{E}}\sum_{k_v^t\in K_e^t} n_{v,e}^t \tag{11}$$
$$\text{s.t.}\ \ C_6:\sum_{k_v^t\in K_{q_e}^t} c_{v,e}^t\leqslant c_e,\forall e\in\mathcal{E}. \tag{11a}$$

The optimization problem can be proved to be an approximately convex programming problem. $\boldsymbol{\mathcal{P}_2}$ can be further decomposed into multiple simpler subproblems, each associated with an RSU e, formulated as follows:

$$\boldsymbol{\mathcal{P}_3}:\ \min_{C_e^t} g_3^e=\sum_{k_v^t\in K_{q_e}^t} x_{v,e}^t \tag{12}$$
$$\text{s.t.}\ \ C_7:\sum_{k_v^t\in K_{q_e}^t} c_{v,e}^t\leqslant c_e, \tag{12a}$$

where C_e^t represents the variables in C^t related to RSU e. Then, problem $\boldsymbol{\mathcal{P}_3}$ can be proved to be a convex optimization problem, with the objective function being convex and the constraints being linear.

Theorem 4: $\boldsymbol{\mathcal{P}_3}$ is a convex optimization problem.

Proof: The feasible region for problem $\boldsymbol{\mathcal{P}_3}$ is $[0,c_e]$, which is obviously a convex set. Let f be the objective function, $f=\sum_{k_v^t\in K_{q_e}^t}\frac{D_k c_k}{c_{v,e}^t}$. Then $f'=-\sum_{k_v^t\in K_{q_e}^t}\frac{D_k c_k}{(c_{v,e}^t)^2}$, $f''=\sum_{k_v^t\in K_{q_e}^t}\frac{2D_k c_k}{(c_{v,e}^t)^3}$. On the defined convex set $[0,c_e]$, f'' is always greater than 0, so the objective function f is a convex function.

Thus, problem $\boldsymbol{\mathcal{P}_3}$ is a convex optimization problem. Due to the nonlinearity of the objective function, problem $\boldsymbol{\mathcal{P}_3}$ is further a nonlinear convex optimization problem. ■

The Lagrange multiplier method can be used to solve the problem. By introducing the Lagrange multiplier, the Lagrange function is constructed. The optimization problem with constraints is transformed into an unconstrained optimization problem, which is expressed as:

$$\boldsymbol{\mathcal{P}_4}:\ \min_{\lambda_e^t,C_e^t} g_4=g_3^e+\lambda_e^t\left(\sum_{k_v^t\epsilon K_{q_e}^t} c_{v,e}^t-c_e\right) \tag{13}$$
$$\text{s.t.}\ \ C_8:\lambda_e^t\geqslant 0. \tag{13a}$$

The optimal solution is given by the KKT conditions in the Lagrange multiplier method, which are necessary and sufficient due to the convexity. By the KKT conditions, we have:

$$\begin{cases}\nabla_{C_e^t} g_3^e+\lambda_e^t\nabla_{C_e^t}\left(\sum_{k_v^t\in K_{q_e}^t} c_{v,e}^t-c_e\right)=0,\\ \lambda_e^t\left(\sum_{k_v^t\in K_{q_e}^t} c_{v,e}^t-c_e\right)=0,\\ \lambda_e^t\geqslant 0.\end{cases} \tag{14}$$

Solving the system of equations, the optimal resource allocation strategy for task k_v^t is:

$$c_{v,e}^t{}^*=\frac{1/c_e\cdot\sqrt{\log d_k c_k}}{\sum_{k_v^t\in K_{q_e}^t} 1/c_e\cdot\sqrt{\log d_k c_k}},\forall k_v^t\in K_{q_e}^t. \tag{15}$$

C. MADDPG Algorithm

Theorem 5: If the computing task of each vehicle overflows the computing power of an RSU with a probability of $\frac{1}{p}$ and task overflow only affects the next round of decision-making, then the dynamic unloading process over time steps is a Markov Decision Process.

The relevant proofs of the theorem are included in Appendix A, available online. The proof for the special case is provided in Appendix A-A, available online, while the proof for the general case is provided in Appendix A-B, available online. According to Theorem 5, the dynamic unloading process is a Markov decision process. To conduct algorithm design based on it, it is necessary to mathematically define the state of the RSU at time step t.

The state of RSU e at time t is represented as:

$$o_e^t = \left\{e, t, Dis_{V_e^t}, D_{K_e^t}, C_{K_e^t}, T_{K_e^t}\right\}, \tag{16}$$

where $Dis_{V_e^t}$ represents the set of spatial distance between vehicle $v \in V_e^t$ and RSU e, $D_{K_e^t}$, $C_{K_e^t}$ and $T_{K_e^t}$ represent the set of data size d_k, CPU cycle frequency c_k, and deadline t_k of task $k_v^t \in K_e^t$. Therefore, the state space of the system at time t can be represented as $o^t = \{o_1^t, \ldots, o_e^t, \ldots, o_E^t\}$.

The action space of each RSU e consists of the task offloading decisions requested by $v \in V_e^t$, represented as:

$$a_e^t = \left\{q_{v,e'}^t \left| \forall e' \in \mathcal{E}, \forall v \in V_e^t \right.\right\}, \tag{17}$$

where $q_{v,e'}^t \in \{0,1\}$ indicates whether the task k_v^t is offloaded at the RSU e'. The set of actions for RSUs is represented as $a^t = \{a_e^t | \forall e \in \mathcal{E}\}$. $P(o^{t+1}|o^t, a^t) \in (0,1)$ denotes the probability of state transition from the current state o^t to the next state o^{t+1} obtained by randomly sampling the action of the policy function.

In the potential game model, our purpose is to offload tasks to proper RSUs so as to maximize the system's cumulative reward. Therefore, we use the potential function of the system to represent its reward function at time t, expressed as:

$$r\left(a^t \left| o^t\right.\right) = \sum_{e\in\mathcal{E}} U_e(\zeta) = \sum_{e\in\mathcal{E}} \psi_e^t. \tag{18}$$

Under the system state o^t, the reward function of the system is used to construct the reward for RSU e with action a_e^t:

$$r_e^t = r\left(a^t \left| o^t\right.\right) - r\left(a_{-e}^t \left| o^t\right.\right), \tag{19}$$

where $r(a_{-e}^t|o^t)$ is the system reward obtained when RSU e makes no contribution. The reward set of RSUs is represented as $r^t = \{r_1^t, \ldots, r_e^t, \ldots, r_E^t\}$. In Fed-MADDPG, the expected return optimized by each e is represented as $R_e^t = \sum_{i\geqslant 0} \gamma^i r_e^{t+i}$, where γ is the discount factor.

D. Federated Training

In the federated learning framework, parameter sharing occurs across all agents' policy networks while maintaining the confidentiality of diverse tasks. This method addresses the restricted observability issue in centralized training and execution within traditional MADDPG. Consequently, we introduce the Fed-MADDPG algorithm, which facilitates the potential game model's convergence to the Nash equilibrium with enhanced accuracy, ensuring the privacy of vehicle owners.

At the outset, the policy and value network parameters for RSUs are randomly initialized locally, represented by θ_e and ω. Simultaneously, the parameters of the target policy and target value networks in the learner are initialized identically to the local networks, expressed as θ_e^- and ω^-. A replay buffer $\mathcal{H}$ is initialized with a maximum capacity of $|\mathcal{H}|$ to archive past experiences. The procedural flow of the Fed-MADDPG training is depicted in Algorithm 1.

During the local training process, the cooperative relationship among multiple agents adopts a centralized training and decentralized execution approach. As a result, different agents have distinct policy networks but share the same value network. During the initialization and experience collection stages, each RSU e is regarded as an agent that gains experience through interactions with the environment and stores experience in the replay buffer. It is equipped with the capability for local training based on the DDPG algorithm and can share its policy network within the FL framework. Thus, the task offloading action of RSU e at time t is:

$$a_e^t \sim \mu\left(\cdot \left| o_e^t; \theta_e\right.\right) + \varepsilon \mathbb{N}_t, \tag{20}$$

where $\mathbb{N}_t$ is the exploration noise and ε is the exploration constant. Execute action a^t and get the reward according to the reward function. Finally, the state o_e^t, action a_e^t, reward r_e^t,

Algorithm 1: Fed-MADDPG Algorithm.

```
Input  : $Dis_{V_e^t}$: Set of distance between each vehicle
         $v$ in $V_e^t$ and RSU $e$;
         $D_{K_e^t}$: Set of data size of each task $k_v^t$ in $K_e^t$;
         $C_{K_e^t}$: Set of CPU cycle frequency;
         $T_{K_e^t}$: Set of deadline of each task $k_v^t$ in $K_e^t$.
Output: Optimized policy network parameters.
1  Randomly initialize local network weights $\theta_e$ and $\omega$;
2  Copy the local network weights to the target network;
3  Initialize replay buffer $\mathcal{H}$;
4  for $\alpha$=1 to Max-iterations do
5      for $t$=1 to $T$ do
6          for $e$=1 to $E$ do
7              Generate the action $a_e^t$ according to
                 Eq. (20) ;
8              Execute the action $a_e^t$ and obtain $o_e^{t+1}$ and
                 $r_e^t$;
9              Store $(o_e^t, a_e^t, o_e^{t+1}, r_e^t)$ into replay buffer $\mathcal{H}$;
10             Randomly sample $M$ samples from the
                 replay buffer of length $N$;
11             Compute the TD target $Y_e^i$ based on the
                 samples and Eq. (21);
12             Compute TD error $\delta^i$ based on Eq. (22);
13             Update the parameters of the local value
                 network based on Eq. (23);
14             Update the parameters of the local policy
               network based on Eq. (24);
15         if $t = k \cdot t_{tg}, \forall k \in Z$ then
16             Update target network weights based on
                 Eq. (25);
17     if $\alpha = k \cdot \alpha_{tg}, \forall k \in Z$ then
18         for $e$=1 to $E$ do
19             Send local policy network parameters $\theta^{\alpha+1}$
                 based on Eq. (26);
20         Aggregate policy network parameters $\theta^{\alpha+1}$
             based on Eq. (27);
21         Distribute the aggregated parameters to the
             policy network of each agent based on
             Eq. (28);
22 end
```

and the next state o_e^{t+1} are added into the replay buffer $\mathcal{H}$. The training process continues until the model converges.

During the parameter update phase, from the buffer M, each mini-batch of decision sequence of length N is selected to train the local policy and value network, and transformed into $(o^{i:i+N}, a^{i:i+N-1}, r^{i:i+N-1})$. First, we compute the TD target for agent e based on the parameters of the target critic network and the target policy network:

$$Y_e^i = \sum_{n=0}^{N-1} \left(\gamma^n r_e^{i+n}\right) + \gamma^N Q^- \left(o_e^{i+N}, \mu^- \left(\cdot \left| o_e^{i+N}; \theta_e^- \right.\right) \left| \omega^- \right.\right). \tag{21}$$

where $a^{i+N} = \{a_1^{i+N}, \ldots, a_e^{i+N}, \ldots, a_E^{i+N}\}$.

Next, we compute the TD error based on the locally observed Q-value and the TD target:

$$\delta^i = \frac{1}{M} \sum_i \left(Q\left(o_e^i, a_e^i \left| \omega\right.\right) - Y_e^i\right)^2. \tag{22}$$

Then, we compute the gradient by $p_\omega^i = \frac{\partial Q(o_e^i, a_e^i|\omega)}{\partial \omega}|_{\omega=\omega^i}$, and update the local value network by minimizing the TD error during the backpropagation process:

$$\omega^{i+1} = \omega^i - \sigma \delta^i p_\omega^i. \tag{23}$$

Next, we compute the policy gradient of the local policy network of RSU e by $p_{\theta_e}^i = \frac{1}{M}\sum_i \left(\frac{\partial \log \mu(\cdot|o_e^i;\theta_e)}{\partial \theta_e}|_{\theta_e=\theta_e^i}\right)$, and update its parameters using gradient ascent:

$$\theta_e^{i+1} = \theta_e^i + \rho Q\left(o_e^i, a_e^i \left|\omega^i\right.\right) p_{\theta_e}^i. \tag{24}$$

Finally, if $t_{tg} \mid t$, the agents update the target network parameters, with t_{tg} being the target network parameter update cycle. The update process is as follows:

$$\begin{cases} \theta_e^- \leftarrow n\theta_e + (1-n)\,\theta_e^-, \\ \omega^- \leftarrow n\omega + (1-n)\,\omega^-. \end{cases} \tag{25}$$

The application of a federated learning framework can be seen as introducing a global perspective to the model. In each iteration, after the local model completes its training, each local model uploads its policy network parameters for aggregation. The aggregated parameters are then distributed back to the local models, enhancing the model's ability to evaluate decision-making completeness from a global perspective. α_{tg} is used as the aggregation period to control how often the server aggregates the policy network parameters from multiple clients in the federated learning framework. Every α_{tg} iterations, after the local model of each agent is well-trained, the parameters from the local value model are temporarily stored as they are no longer needed, and the parameters from the local policy network are sent to the VEC POOL by:

$$\theta_1^\alpha, \theta_2^\alpha, \ldots, \theta_E^\alpha \rightarrow \Theta, \tag{26}$$

where $\Theta = \{\theta_e^\alpha | e \in 1, 2, \ldots, E\}$ is stored in the VEC POOL temporarily. Then, the VEC POOL aggregates the parameters from the clients into new parameters:

$$\theta^{\alpha+1} \leftarrow \frac{1}{E}\sum_{e=1}^{E} \theta_e^\alpha, \theta_e^\alpha \in \Theta. \tag{27}$$

TABLE III
SIMULATION PARAMETERS

Parameter	Value
Max-iterations	6100
Loss rate γ	0.990
Number of training batches M	256
Maximum capacity of experience pool $\lvert\mathcal{H}\rvert$	10^6
Exploration rate ε	0.5
Learning rate for value network	10^{-4}
Learning rate for policy network	10^{-5}
Updating frequency of target network weights t_{tg}	150
Aggregation Period α_{tg}	500

Finally, the aggregated policy network parameters are distributed to each agent for training in a new iteration:

$$\theta_1^{\alpha+1}, \theta_2^{\alpha+1}, \ldots, \theta_E^{\alpha+1} \leftarrow \theta^{\alpha+1}. \tag{28}$$

V. PERFORMANCE EVALUATION

A. Experimental Setup

This section describes the experimental configuration, including parameter settings, datasets, baselines, and evaluation metrics, to validate the performance of our algorithm.

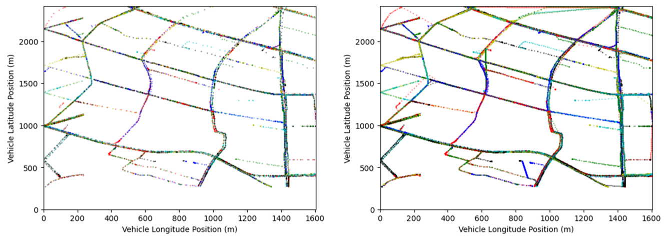
The experiments were conducted in a virtual environment configured using Python 3.9, TensorFlow 2.8.0, CUDA 11.2, and CuDNN 8.1.1, managed through Conda. To accommodate the distributed nature of FL, certain dependencies utilized built-in versions of CUDA and CuDNN, such as 'nvidia-cuda-nvrtc-cull==11.7.99'. The simulation model was executed on an Ubuntu 20.04.1 server equipped with an Intel(R) Xeon(R) Silver 4214R 12-core processor at 2.40 GHz, five NVIDIA GeForce RTX 3090 GPUs, and 503 GB of memory.

1) Parameters Setting: The simulation environment was designed to represent a realistic VEC scenario, with 8 RSUs evenly distributed along a 3 km road. The computing power of the RSUs ranged between 3 GHz and 10 GHz, while the V2I communication range was set to 500 m. Network bandwidth settings included 54 Mbps (802.11 g), 450 Mbps (802.11n), and 866.7 Mbps (802.11ac wave1), with values selected from the theoretical maximum data rates in [42].

The parameters used in the Fed-MADDPG algorithm are based on the configurations in [30], as detailed in Table III.

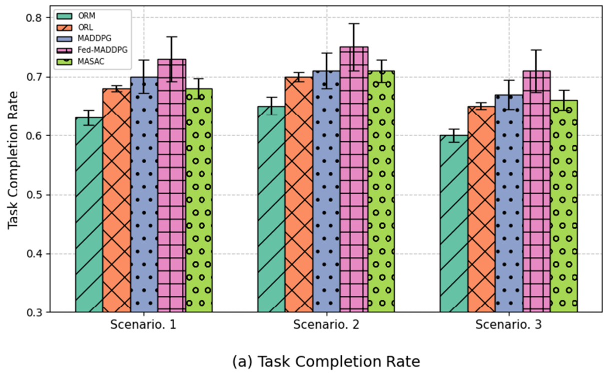
2) Datasets: The training data was sourced from the Didi GAIA Open Data, collected on November 16, 2016. This dataset includes vehicle trajectory data across a 9 km^2 section in the Qingyang District of Chengdu. For this study, a 3 km-long and 35 m-wide expressway was selected as the experimental focus area.

To simulate different service scenarios, data from three time periods (8 am - 8:05 am, 1 pm - 1:05 pm, and 6 pm - 6:05 pm) was utilized. During the three periods, the vehicle trajectories are most dense at noon, followed by the morning, with the evening period being the least dense. Based on the statistical data, we can calculate the total number of vehicle trajectories and the number of vehicles passing through each grid per second in three different scenarios [30]. These metrics will be used to

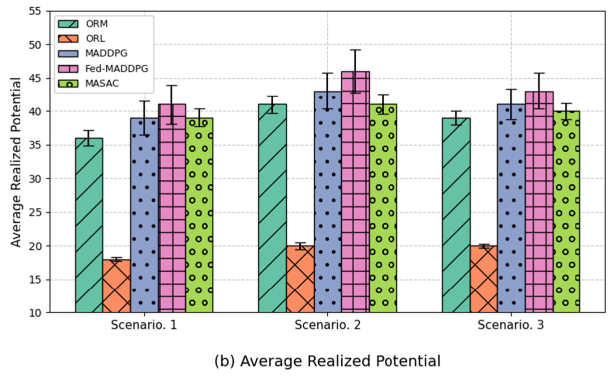


(a) Before Filling (b) After Filling

Fig. 3. Comparison of Trajectories Before and After Filling



(a) Task Completion Rate under Different Traffic Volumes



(b) Average Realized Potential under Different Traffic Volumes

Fig. 4. Comparison of Algorithm Performance under Different Traffic Volumes

describe the distribution of vehicles in both the time and spatial dimensions as environmental information.

As shown in Fig. 3, the original trajectory data had long collection intervals, which are insufficient for algorithm training. Missing data points were filled using the mean value to ensure higher data density. Fig. 3(a) and (b) illustrate the trajectory data before and after filling.

3) Baselines: To evaluate the performance of our algorithm, we conducted comparative experiments with four baselines:

- *Offloading-Request Migration (ORM) [30]:* This method involves task offloading and resource allocation, with RSUs migrating tasks to other RSUs to balance the workload.
- *Offloading-Request Local (ORL) [43]:* RSUs tend to compute tasks locally without offloading to other nodes, reducing communication overhead but potentially limiting computational efficiency.
- *Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [30]:* Each RSU acts as an independent agent and implements MADDPG for task offloading decisions influenced by resource allocation.
- *Multi-Agent Soft Actor-Critic (MASAC) [44]:* Each RSU acts as an independent agent and implements MASAC for task offloading decisions influenced by resource allocation.

4) Metrics: To comprehensively evaluate the proposed algorithm, the following metrics were employed:

- *Task Completion Rate:* The ratio of successfully completed tasks to the total number of tasks within a given timeframe, which can be obtained by (7).
- *Cumulative Reward:* The total reward accumulated by the system during the entire scheduling period that reflects the overall performance of the system, which can be obtained by (18).
- *Average Realized Potential:* The rewards divided by RSUs, which is defined as $\sum_{e \epsilon \mathcal{E}} \sum_{t \epsilon T} r_e^t / E$ [30], measures the advantages of constructing the reward function through the potential function.

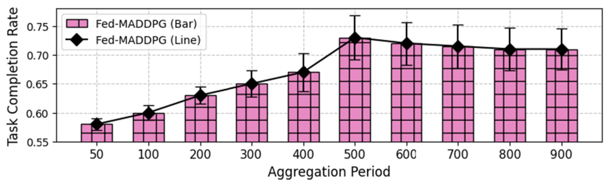
B. Experimental Results and Analysis

1) Impact of Traffic Volume: Regarding the impact of traffic volume on algorithm performance, the results are shown in Fig. 4. Fig. 4(a) shows the task completion rate and the error bars of the five algorithms under different traffic volumes. The Fed-MADDPG algorithm significantly outperforms other algorithms. As traffic volume increases, the performance gap between Fed-MADDPG and the others becomes more pronounced, indicating that Fed-MADDPG is better equipped to handle heavy traffic conditions. This improvement suggests that Fed-MADDPG efficiently balances task offloading and resource allocation, resulting in higher task completion rates.

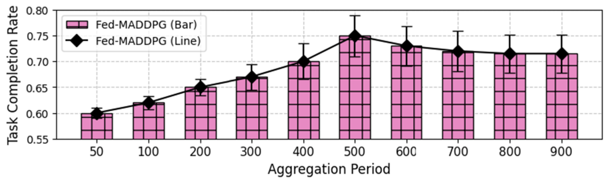
Fig. 4(b) shows the average realized potential and the error bars of the five algorithms under different traffic volumes. The results indicate that Fed-MADDPG provides significantly higher realized potential compared to the other algorithms. This reflects its superior ability to allocate resources effectively and distribute rewards more efficiently among RSUs, especially as the traffic volume increases.

With increasing traffic volume, all algorithms exhibit improvements in task completion rate and average realized potential, yet Fed-MADDPG consistently outperforms the others. This indicates that although higher traffic levels necessitate greater resource allocation, Fed-MADDPG demonstrates superior scalability, effectively enhancing task completion and reward distribution concurrently.

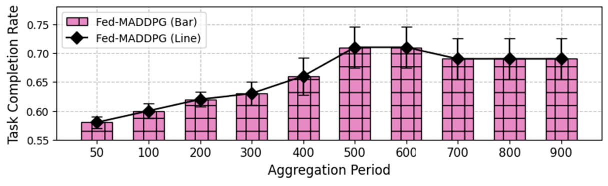
2) Impact of Aggregation Period α_{tg}: Fig. 5(a), (b), and (c) illustrate the relationship between aggregation period α_{tg} and the task completion rate under different traffic volumes. The x-axis represents the aggregation period, ranging from 50 to 900, while the y-axis denotes the task completion rate. The bars show a clear trend: as the aggregation period increases



(a) Scenario 1

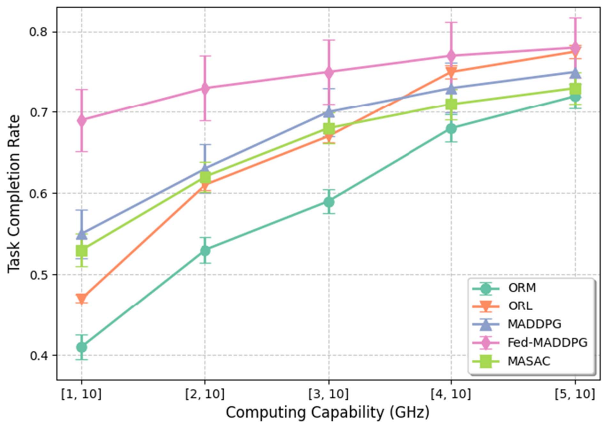


(b) Scenario 2

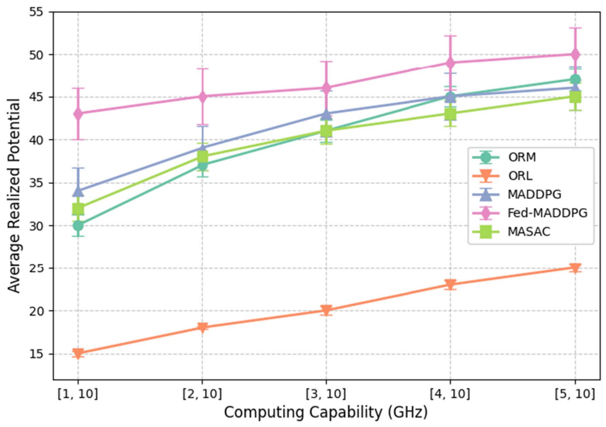


(c) Scenario 3

Fig. 5. Effect of aggregation period on task completion rate under different traffic volumes



(a) Task Completion Rate



(b) Average Realized Potential

Fig. 6. Comparison of Algorithm Performance under Different RSU Computational Capabilities

from 50 to 500, the average task completion rate improves significantly. However, after reaching an aggregation period of 500, the improvement slows and plateaus, with only marginal differences from 500 to 900.

This trend can be attributed to the trade-off between the global system's performance and efficiency. A lower aggregation period allows for faster model convergence due to the frequent aggregation of the policy network parameters, but fails to capture the global dynamics, resulting in suboptimal performance. Conversely, a longer aggregation period provides more time for local updates, which helps in capturing global relationships but slows down convergence. The trend after $\alpha_{tg} = 500$ suggests that further increasing the aggregation period has a negligible impact on the model performance, as the global model has already captured sufficient dynamics. Therefore, we selected $\alpha_{tg} = 500$ as the aggregation period for subsequent experiments.

3) Impact of RSU Computational Capabilities c_e: Regarding the impact of different RSU computational capabilities on algorithm performance, the results are presented in Fig. 6. As expected, the stronger the computational capability of the RSUs, the more tasks can be executed within the specified time, leading to better algorithm performance.

Fig. 6(a) shows the task completion rate and the error bars of the five algorithms under different RSU computational capabilities. As the computational power of the RSUs increases, all algorithms experience a noticeable improvement. However, Fed-MADDPG consistently outperforms the other algorithms across all computational capabilities, demonstrating its superior ability to handle task offloading and resource management. The improvement in task completion rate with stronger RSU computational capabilities reflects the importance of computational resources in ensuring timely task execution and the overall effectiveness of the system.

Fig. 6(b) illustrates the average realized potential and the error bars of the five algorithms with varying RSU computational capabilities. This metric reveals that as RSU computational capabilities increase, the Fed-MADDPG algorithm provides significantly higher realized potential compared to the other algorithms. This indicates that Fed-MADDPG is not only better at handling task offloading but also excels at efficiently distributing rewards to RSUs. The improvement in realized potential emphasizes the value of the potential function-based reward structure, which allows RSUs to contribute to the system performance more effectively.

In summary, these results highlight the critical role of RSU computational capabilities in improving both task completion rates and the efficient distribution of rewards. Based on these insights, we choose to distribute the RSU computational capabilities c_e evenly across multiple RSUs in the range of [3, 10]. This distribution ensures that the system has sufficient capacity

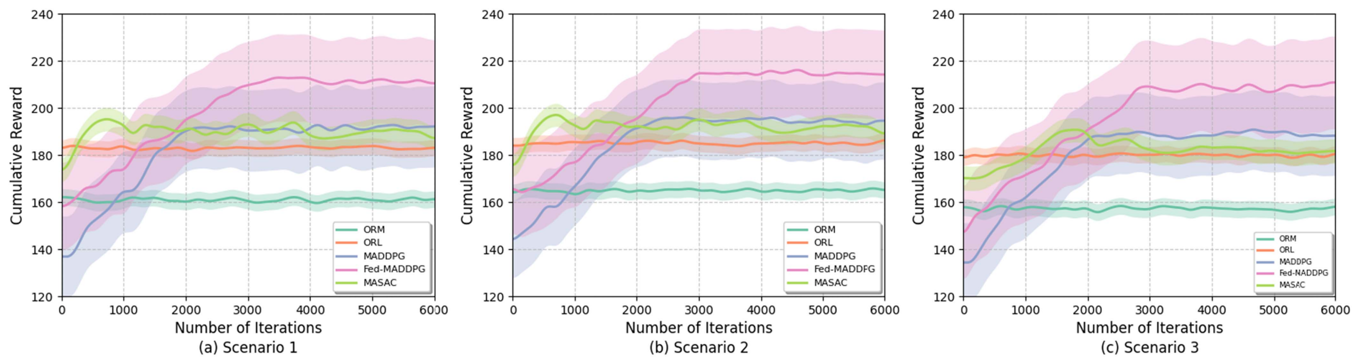


Fig. 7. Comparison of Algorithm Convergence under Different Traffic Conditions.

for varying traffic scenarios while maintaining the flexibility to adapt to different computational requirements.

4) Convergence Analysis: Fig. 7 provides a comparative analysis of the cumulative rewards of the five algorithms under different traffic scenarios. The figure was generated by recording the cumulative reward every 10 iterations, visualizing the entire training process. To enhance clarity, we applied Gaussian smoothing with a standard deviation of 10, where larger shaded regions indicate greater variability in the results. The observed larger variability in Fed-MADDPG and MADDPG can be attributed to the task heterogeneity introduced by multi-agent computing.

In terms of performance, the figure highlights several key observations. For each subfigure, under the current parameter settings, Fed-MADDPG demonstrates superior performance compared to the other algorithms. Although it converges at around 3,000 iterations, which means that its convergence speed is slightly slower than that of MADDPG, the trade-off is acceptable given its higher cumulative rewards. On a broader scale, our algorithm performs consistently well across all traffic scenarios. Notably, in Scenario 2, which corresponds to peak traffic conditions during midday, the denser the traffic, the better the algorithm's performance. This indicates the robustness of our approach to managing diverse traffic conditions effectively.

VI. Conclusion and Future Work

In this paper, we propose a federated learning-based scheduling model that jointly optimizes task offloading and resource allocation to maximize task completion rates. We demonstrate that the task offloading process, influenced by resource allocation, can be framed as a non-cooperative potential game. Building on this, we introduce the Fed-MADDPG algorithm, which leverages the potential function as the reward signal. Simulation experiments demonstrate that Fed-MADDPG exhibits superior performance in terms of task offloading completion and future offloading completion trends across various traffic scenarios. In future work, we aim to extend the system model to include collaborative cloud-side processing, integrating onboard edge computing with federated learning in a more complex environment to complete the training of the collaborative model.

References

[1] X. Long, Y. Zhao, H. Wu, and C.-Z. Xu, "Deep reinforcement learning for integrated sensing and communication in RIS-assisted 6G V2X system," *IEEE Internet Things J.*, vol. 11, no. 24, pp. 39834–39849, Dec. 2024.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.

[3] Y. Liang, H. Tang, H. Wu, Y. Wang, and P. Jiao, "Lyapunov-guided offloading optimization based on soft actor-critic for ISAC-aided Internet of Vehicles," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 14708–14721, Dec. 2024.

[4] H. Tang, H. Wu, G. Qu, and R. Li, "Double deep Q-network based dynamic framing offloading in vehicular edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 3, pp. 1297–1310, May/Jun. 2023.

[5] X. Shen et al., "AI-assisted network-slicing based next-generation wireless networks," *IEEE Open J. Veh. Technol.*, vol. 1, pp. 45–66, 2020.

[6] H. Wu, J. Chen, T. N. Nguyen, and H. Tang, "Lyapunov-guided delay-aware energy efficient offloading in IIoT-MEC systems," *IEEE Trans. Ind. Informat.*, vol. 19, no. 2, pp. 2117–2128, Feb. 2023.

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

[8] J. Yang et al., "Service-aware computation offloading for parallel tasks in VEC networks," *IEEE Internet Things J.*, vol. 12, no. 3, pp. 2979–2993, Feb. 2025.

[9] F. Zeng, Q. Chen, L. Meng, and J. Wu, "Volunteer assisted collaborative offloading and resource allocation in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3247–3257, Jun. 2021.

[10] J. Geng, Z. Qin, and S. Jin, "Dynamic resource allocation for cloud-edge collaboration offloading in VEC networks with diverse tasks," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 12, pp. 21235–21251, Dec. 2024.

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

[12] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[13] L. Qian and J. Zhao, "User association and resource allocation in large language model based mobile edge computing system over 6G wireless communications," in *Proc. IEEE 99th Veh. Technol. Conf.*, 2024, pp. 1–7.

[14] M. Z. Alam and A. Jamalipour, "Multi-agent DRL-based hungarian algorithm (MADRLHA) for task offloading in multi-access edge computing internet of vehicles (IoVS)," *IEEE Trans. Wireless Commun.*, vol. 21, no. 9, pp. 7641–7652, Sep. 2022.

[15] S. R. Jeremiah, L. T. Yang, and J. H. Park, "Digital twin-assisted resource allocation framework based on edge collaboration for vehicular edge computing," *Future Gener. Comput. Syst.*, vol. 150, pp. 243–254, 2024.

[16] J. Xue, L. Wang, Q. Yu, and P. Mao, "Multi-agent deep reinforcement learning-based partial offloading and resource allocation in vehicular edge computing networks," *Comput. Commun.*, vol. 234, 2025, Art. no. 108081.

[17] S. S. Shinde and D. Tarchi, "Collaborative reinforcement learning for multi-service Internet of Vehicles," *IEEE Internet Things J.*, vol. 10, no. 3, pp. 2589–2602, Feb. 2023.

[18] J. A. Onieva, R. Rios, R. Roman, and J. Lopez, "Edge-assisted vehicular networks security," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8038–8045, Oct. 2019.

[19] J. Kang et al., "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4660–4670, Jun. 2019.
[20] A. Jolfaei, K. Kant, and H. Shafei, "Secure data streaming to untrusted road side units in intelligent transportation system," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./13th IEEE Int. Conf. Big Data Sci. Eng.*, 2019, pp. 793–798.
[21] 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.
[22] X. Li, L. Lu, W. Ni, A. Jamalipour, D. Zhang, and H. Du, "Federated multi-agent deep reinforcement learning for resource allocation of vehicle-to-vehicle communications," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8810–8824, Aug. 2022.
[23] K. Tan, L. Feng, G. Dán, and M. Törngren, "Decentralized convex optimization for joint task offloading and resource allocation of vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 71, no. 12, pp. 13226–13241, Dec. 2022.
[24] Y. Zhang, X. Qin, and X. Song, "Mobility-aware cooperative task offloading and resource allocation in vehicular edge computing," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops*, 2020, pp. 1–6.
[25] H. Xiao, J. Zhao, Q. Pei, J. Feng, L. Liu, and W. Shi, "Vehicle selection and resource optimization for federated learning in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11073–11087, Aug. 2022.
[26] Y. Ju et al., "Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 5, pp. 5555–5569, May 2023.
[27] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158–11168, Nov. 2019.
[28] Z. Ning, P. Dong, X. Wang, J. J. Rodrigues, and F. Xia, "Deep reinforcement learning for vehicular edge computing: An intelligent offloading system," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–24, 2019.
[29] L. Zhao, S. Huang, D. Meng, B. Liu, Q. Zuo, and V. C. M. Leung, "Stackelberg-game-based dependency-aware task offloading and resource pricing in vehicular edge networks," *IEEE Internet Things J.*, vol. 11, no. 19, pp. 32337–32349, Oct. 2024.
[30] X. Xu et al., "Joint task offloading and resource optimization in noma-based vehicular edge computing: A game-theoretic DRL approach," *J. Syst. Archit.*, vol. 134, pp. 1383–7621, 2023.
[31] W. Fan et al., "Game-based task offloading and resource allocation for vehicular edge computing with edge-edge cooperation," *IEEE Trans. Veh. Technol.*, vol. 72, no. 6, pp. 7857–7870, Jun. 2023.
[32] X. Tang, X. Chen, L. Zeng, S. Yu, and L. Chen, "Joint multiuser DNN partitioning and computational resource allocation for collaborative edge intelligence," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9511–9522, Jun. 2021.
[33] W. Y. B. Lim et al., "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 22, no. 3, pp. 2031–2063, Third Quarter 2020.
[34] X. Yuan, J. Chen, N. Zhang, C. Zhu, Q. Ye, and X. S. Shen, "FedTSE: Low-cost federated learning for privacy-preserved traffic state estimation in IoV," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2022, pp. 1–6.
[35] Z. Yu, J. Hu, G. Min, Z. Zhao, W. Miao, and M. S. Hossain, "Mobility-aware proactive edge caching for connected vehicles using federated learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5341–5351, Aug. 2021.
[36] Q. Xia, W. Ye, Z. Tao, J. Wu, and Q. Li, "A survey of federated learning for edge computing: Research problems and solutions," *High-Confidence Comput.*, vol. 1, no. 1, 2021, Art. no. 100008.
[37] B. Xu, W. Xia, W. Wen, P. Liu, H. Zhao, and H. Zhu, "Adaptive hierarchical federated learning over wireless networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 2, pp. 2070–2083, Feb. 2022.
[38] S. Guo and B.-J. Hu, "Energy and gradient aware dynamic scheduling for V2V aided federated edge learning," *IEEE Commun. Lett.*, vol. 28, no. 2, pp. 323–327, Feb. 2024.
[39] L. L. Peterson and B. S. Davie, *Computer Networks: A Systems Approach*. New York, NY, USA: Elsevier, 2007.
[40] W. Stallings, *Data and Computer Communications*. Chennai, India: Pearson Education India, 2007.
[41] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*, 6th Ed. Berlin, Germany: Springer, 2018.
[42] Intel, "Different Wi-Fi protocols and data rates," Oct. 28, 2021, Accessed: Nov. 10, 2024, [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000005725
[43] 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.
[44] Y. Pu, S. Wang, R. Yang, X. Yao, and B. Li, "Decomposed soft actor-critic method for cooperative multi-agent reinforcement learning," 2021. [Online]. Available: https://arxiv.org/abs/2104.06655

Songxin Lei received the BSc degree from Tianjin University, China, in 2020. He is currently working toward the MS degree with Guangzhou Campus, The Hong Kong University of Science and Technology. His research interests include spatio-temporal data mining, mobile edge computing, and deep reinforcement learning.

Huijun Tang (Member, IEEE) received the BSc degree from Jinan University, China, in 2016 and the MS and PhD degrees from Tianjin University, China, in 2018 and 2022, respectively. She is currently a post-doctoral research associate with Durham University, U.K. Her research interests include Internet of Things, mobile edge computing, deep learning and complex networks.

Chuangyi Li is currently working toward the BSc degree with Tianjin University, China. His research interests include machine learning, deep learning, and efficient intelligent computing.

Xueying Zhang received the BSc degree from the China University of Petroleum (East China), China, in 2024. She is currently working toward the MS degree with the Center for Applied Mathematics, Tianjin University, China. Her research interests include edge computing, Internet of Things, and federated learning.

Chenli Xu received the BSc degree from Hunan University, China, in 2021. She is currently working toward the PhD degree with Tianjin University, China. Her research interests include game theory and its applications, and sparse optimization.

Huaming Wu (Senior Member, IEEE) received the BE and MS degrees in electrical engineering from the Harbin Institute of Technology, China, in 2009 and 2011, respectively, and the PhD degree with the highest honor in computer science from Freie Universität Berlin, Germany, in 2015. He is currently a professor with the Center for Applied Mathematics, Tianjin University, China. His research interests include mobile cloud computing, edge computing, Internet of Things, deep learning, complex networks, and DNA storage.