FD3QN: A Federated Deep Reinforcement Learning Approach for Cross-Domain Resource Cooperative Scheduling in Hybrid Cloud Architecture

Liang Xiao[1], Honghua Shan[1*], Jianwei Zhu[2], Ruiwei Mao[3], Shuwei Pan[2]
[1]Comprehensive Department, China Mobile Group Zhejiang Co., Ltd. Huzhou Branch, Huzhou 313000, Zhejiang, China
[2]Network Maintenance Department, China Mobile Group Zhejiang Co., Ltd. Huzhou Branch, Huzhou 313000, Zhejiang, China
[3]Engineering Construction Department, China Mobile Group Zhejiang Co., Ltd. Huzhou Branch, Huzhou 313000, Zhejiang, China
E-mail: shanhh@zj.chinamobile.com
[*]Corresponding author

Keywords: cross-domain resource cooperative scheduling, deep reinforcement learning, federated learning, FD3QN algorithm, hybrid cloud, UAV trajectory planning, vehicle networking

Received: September 9, 2024

To address the challenge of insufficient computing power in multi-access edge computing (MEC) servers caused by highly dynamic service requests and uneven service distribution in vehicular networks, this paper proposes a hybrid multi-server MEC architecture that leverages both fixed road-side units (RSUs) and mobile unmanned aerial vehicles (UAVs). We introduce the FD3QN algorithm, which integrates federated learning and deep reinforcement learning, to minimize the weighted sum of service latency and energy consumption. Specifically, the MATD3 algorithm is employed for safe and efficient UAV trajectory planning in the offloading decision process. For resource allocation, we embed vertical federated learning into the D3QN network to enable cross-domain resource cooperative scheduling. A decentralized federated aggregation framework is utilized to maintain a global model for optimizing resource allocation in a collaborative and privacy-preserving manner. The proposed algorithm jointly optimizes transmission power, computing, and storage resources. Extensive simulations are conducted to evaluate the performance of FD3QN in a realistic vehicular network environment with varying numbers of vehicles and task arrivals. The results demonstrate that FD3QN outperforms benchmark algorithms, achieving an 11.37% and 12.06% reduction in system cost compared to the FDDQN algorithm in scenarios with 8 and 12 vehicles, respectively. Moreover, FD3QN exhibits a 25% decrease in average service latency and a 15% improvement in energy efficiency compared to traditional deep reinforcement learning approaches. The proposed algorithm also maintains a high task completion rate of over 98% under dynamic network conditions. These findings validate the strong model generalization ability of FD3QN in the dynamic vehicular networking environment and highlight its practicality for real-world deployment. This study provides novel insights into the development of intelligent transportation systems and edge computing paradigms.

Povzetek: V prispevku je opisan FD3QN, napreden pristop federativnega globokega okrepljenega učenja za sodelovalno razporejanje virov v hibridnih oblačnih arhitekturah. Algoritem zmanjšuje zakasnitve in porabo energije v inteligentnih transportnih sistemih.

1 Introduction

The rapid advancement of intelligent transportation systems (ITS) has revolutionized the way we perceive and interact with modern transportation networks. With the proliferation of connected and autonomous vehicles (CAVs), the demand for computation-intensive and delay-sensitive applications has escalated significantly [1]. However, the resource-constrained nature of vehicular users poses a formidable challenge in meeting the stringent requirements of these applications [2]. To address this issue, multi-access edge computing (MEC) has emerged as a promising paradigm, bringing computation and storage resources closer to the edge of the network [3]. By leveraging the proximity of MEC servers to vehicular users, the latency and network congestion associated with centralized cloud computing can be effectively mitigated [4].

Furthermore, the integration of unmanned aerial vehicles (UAVs) into the MEC framework has opened up new avenues for enhancing the performance of ITS [5]. UAVs can act as aerial MEC servers, providing flexible and on-demand computing resources to vehicular users [6]. The mobility and line-of-sight (LoS) communication links offered by UAVs enable them to dynamically adjust their positions and serve as relay nodes, extending the coverage

and capacity of the vehicular network [7]. Moreover, the collaboration between ground MEC servers and UAV-assisted MEC servers forms a hybrid multi-server architecture, which can efficiently handle the diverse computational requirements of vehicular applications [8]. However, the deployment of a hybrid multi-server MEC architecture in ITS presents several challenges. The heterogeneous nature of vehicular users, with varying mobility patterns and resource demands, necessitates efficient resource allocation and computation offloading strategies [9]. Additionally, the dynamic and uncertain wireless network conditions, coupled with the limited battery life of UAVs, further complicate the decision-making process [10]. To tackle these challenges, advanced techniques such as deep reinforcement learning (DRL) have gained significant attention [11]. DRL combines the power of deep neural networks with reinforcement learning, enabling agents to learn optimal policies through interaction with the environment [12].

In this article, we propose a novel framework for hybrid multi-server computation offloading in UAV-assisted vehicular networks based on federated deep reinforcement learning (FDRL). The proposed framework leverages the distributed learning capabilities of FDRL to enable collaborative and efficient offloading decisions among vehicular users, ground MEC servers, and UAV-assisted MEC servers. By exploiting the local observations and experiences of individual agents, FDRL allows for the development of personalized offloading policies while preserving the privacy of sensitive data [13]. The framework aims to minimize the overall latency and energy consumption of vehicular users while ensuring the stability and scalability of the network.

The main contributions of this article are as follows:

- We formulate the computation offloading problem in a hybrid multi-server MEC architecture for ITS, considering the heterogeneous requirements of vehicular users and the dynamic network conditions.
- We propose an FDRL-based approach to solve the formulated problem, enabling collaborative and efficient offloading decisions among the entities in the network.
- We conduct extensive simulations to evaluate the performance of the proposed framework and compare it with state-of-the-art offloading strategies.

The remainder of this article is organized as follows. Section II provides an overview of the related work on computation offloading in vehicular networks and the application of DRL in MEC. Section III presents the system model and problem formulation. The proposed FDRL-based offloading framework is detailed in Section IV. Section V discusses the simulation results and performance evaluation. Finally, Section VI concludes the article and outlines future research directions.

2 System model

In this section, we present the system model for the proposed hybrid multi-server MEC architecture in UAV-assisted vehicular networks. We consider a novel network architecture that leverages the collaborative computing capabilities of road-side units (RSUs) and unmanned aerial vehicles (UAVs) to support computation offloading for vehicular users [14]. The system model encompasses the network architecture and the binary offloading model, which form the foundation for the development of the FDRL-based offloading framework. Table 1 shows related works on computation offloading in vehicular networks

Table 1: Comparison of related works on computation offloading in vehicular networks.

Work	Computing Resources	Optimization Objectives	Solution Method	Federated Learning	Comparison with Proposed Work
[1]	RSU	Latency	DQN	No	Single-server, latency only
[2]	RSU, Cloud	Energy	DDPG	No	Multi-server, energy only
[3]	RSU, UAV	Latency, Energy	A3C	No	Multi-server, no FL
[4]	RSU, Cloud	Latency, Energy	DQN, DDPG	No	Multi-server, no FL
[5]	UAV	Latency, Energy	Q-learning	No	Single-server, no FL
[6]	UAV	Energy	DDPG	No	Single-server, energy only
[7]	UAV	Energy	Lyapunov	No	Single-server, energy only
[8]	RSU, Cloud	Latency, Energy	DQN	No	Multi-server, no FL

[9]	UAV	Latency, Energy	DRL	No	Single-server, no FL
[10]	UAV	QoE	DQN	No	Single-server, QoE
[11]	UAV	Latency, Energy	DRL	No	Single-server, no FL
[12]	RSU, Cloud	Latency, Energy	DQN	No	Multi-server, no FL
[13]	RSU, Cloud	Latency, Energy	DDPG	Yes	Multi-server, FL
[14]	RSU, UAV, Cloud	Latency, Energy, Cost	DQL	Yes	Multi-server, FL, cost
Proposed	RSU, UAV	Latency, Energy	MATD3, FUD3QN	Yes	Hybrid multi-server, FL

2.1 Network architecture and binary offloading model

The considered network architecture, as depicted in Figure 1, consists of a typical scenario with one RSU and multiple UAVs serving as MEC servers. The RSU is deployed along the roadside and is equipped with substantial computing resources to handle computation-intensive tasks [15]. On the other hand, the UAVs are strategically positioned in the air to provide flexible and on-demand computing services to vehicular users [16]. The UAVs can dynamically adjust their locations based on the distribution and requirements of the vehicular users, ensuring optimal coverage and quality of service [17].

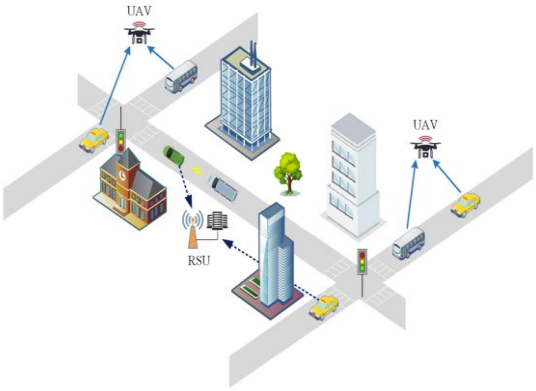

Figure 1: System model of UAV-assisted vehicular networks in a hybrid multi-server MEC architecture.

In the proposed system model, vehicular users have the option to execute their computation tasks locally or offload them to either the RSU or one of the UAVs for remote execution. The decision to offload a task is based on a binary offloading model, where each task is considered as an indivisible unit and can be either processed locally or offloaded to a single MEC server [18]. The binary offloading model simplifies the decision-making process and reduces the complexity of task partitioning and synchronization [19].

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ denote the set of vehicular users in the network, where N is the total number of users. Each user u_i generates a sequence of computation tasks over time, represented by $\mathcal{T}_i = \{t_{i,1}, t_{i,2}, \ldots, t_{i,M}\}$, where M is the number of tasks generated by user u_i. Each task $t_{i,j}$ is characterized by its input data size $d_{i,j}$ (in bits) and the required computing resources $c_{i,j}$ (in CPU cycles)

The vehicular users can communicate with the RSU and UAVs through wireless communication links, as shown in Figure 1. The communication links between the users and the RSU are established using dedicated short-range communication (DSRC) technology, which provides reliable and low-latency communication in vehicular environments [20]. On the other hand, the communication links between the users and the UAVs are established using cellular networks, such as LTE or 5G, which offer wider coverage and higher data rates [21].

The decision to offload a task $t_{i,j}$ is determined by the binary offloading variable $x_{i,j} \in \{0,1\}$, where $x_{i,j} = 0$ indicates local execution and $x_{i,j} = 1$ indicates offloading to either the RSU or a UAV. The offloading decision for each task is made based on various factors, such as the task characteristics, the available computing resources, the communication link quality, and the energy consumption [22].

The objective of the proposed FDRL-based offloading framework is to minimize the overall latency and energy consumption of vehicular users while ensuring the stability and scalability of the network. The framework takes into account the heterogeneous requirements of vehicular users, the dynamic network conditions, and the collaborative computing capabilities of the RSU and UAVs to make optimal offloading decisions [24].

In the following sections, we will delve into the details of the FDRL-based offloading framework, including the problem formulation, the learning algorithm, and the performance evaluation.

2.2 UAV trajectory planning

In the UAV trajectory planning phase, the UAVs are required to take off from fixed ground points and adaptively fly to designated points in the air [25]. The trajectory planning problem aims to determine the optimal

flight paths for the UAVs, considering various constraints such as energy consumption, flight time, and collision avoidance.

Let $\mathcal{V} = \{v_1, v_2, \ldots, v_K\}$ denote the set of UAVs in the network, where K is the total number of UAVs. Each UAV v_k has an initial position $\mathbf{p}_k^0 = (x_k^0, y_k^0, z_k^0)$ on the ground and a designated hovering position $\mathbf{p}_k^h = (x_k^h, y_k^h, z_k^h)$ in the air. The trajectory of UAV v_k is represented by a sequence of positions $\mathbf{p}_k(t) = (x_k(t), y_k(t), z_k(t))$, where $t \in [0, T]$ is the time variable and T is the total flight time.

The trajectory planning problem can be formulated as an optimization problem, with the objective of minimizing the total energy consumption of the UAVs while satisfying the flight time and collision avoidance constraints. The energy consumption of UAV v_k along its trajectory $\mathbf{p}_k(t)$ can be expressed as:

$$E_k = \int_0^T P_k\left(\mathbf{p}_k(t), \mathbf{v}_k(t)\right)dt, \tag{1}$$

where $P_k(\mathbf{p}_k(t), \mathbf{v}_k(t))$ is the power consumption of UAV v_k at position $\mathbf{p}_k(t)$ and velocity $\mathbf{v}_k(t)$.

The flight time constraint ensures that each UAV reaches its designated hovering position within the specified time limit T:

$$\| \mathbf{p}_k(T) - \mathbf{p}_k^h \| \leq \varepsilon, \quad \forall k \in \mathcal{V}, \tag{2}$$

where ε is a small tolerance threshold.

The collision avoidance constraint guarantees a minimum separation distance $d_{\min}$ between any two UAVs at all times:

$$\| \mathbf{p}_k(t) - \mathbf{p}_{k'}(t) \| \geq d_{\min}, \quad \forall k, k' \in \mathcal{V}, k \neq k', \forall t \in [0, T]. \tag{3}$$

Once the UAVs reach their designated hovering positions, they act as MEC servers for the vehicular users. The hovering UAVs provide computing resources to the users, enabling them to offload their computation tasks for remote execution. The stable hovering positions of the UAVs ensure reliable communication links and efficient computation offloading services.

The trajectory planning problem can be solved using various optimization techniques, such as convex optimization, dynamic programming, or meta-heuristic algorithms. The optimal trajectories obtained from the planning phase are then used as inputs to the FDRL-based offloading framework, which determines the optimal offloading decisions for the vehicular users.

In the next section, we will present the problem formulation for the computation offloading problem in the hybrid multi-server MEC architecture, considering the network architecture, binary offloading model, and UAV trajectory planning.

2.3 Resource allocation

In the resource allocation phase, we present the communication model, computation model, and energy consumption model to capture the essential aspects of the hybrid multi-server MEC architecture in UAV-assisted vehicular networks.

2.3.1 Communication model

The communication model characterizes the data transmission process between the vehicular users and the MEC servers (RSU and UAVs). Let $R_{i,j}^r$ and $R_{i,j}^{u_k}$ denote the achievable data rates for offloading task $t_{i,j}$ from user u_i to the RSU and UAV v_k, respectively. The data rates can be expressed using the Shannon-Hartley theorem [26]:

$$R_{i,j}^r = B^r \log_2\left(1 + \frac{P_i^r G_{i,j}^r}{\sigma^2}\right), \tag{4}$$

$$R_{i,j}^{u_k} = B^{u_k} \log_2\left(1 + \frac{P_i^{u_k} G_{i,j}^{u_k}}{\sigma^2}\right), \tag{5}$$

where B^r and B^{u_k} are the channel bandwidths, P_i^r and $P_i^{u_k}$ are the transmission powers, $G_{i,j}^r$ and $G_{i,j}^{u_k}$ are the channel gains, and σ^2 is the noise power.

The transmission delay for offloading task $t_{i,j}$ to the RSU and UAV v_k can be calculated as:

$$D_{i,j}^r = \frac{d_{i,j}}{R_{i,j}^r}, \tag{6}$$

$$D_{i,j}^{u_k} = \frac{d_{i,j}}{R_{i,j}^{u_k}}. \tag{7}$$

2.3.2 Computation model

The computation model describes the processing of tasks at the vehicular users and MEC servers. Let f_i^l denote the local computing capability (in CPU cycles per second) of user u_i. The local execution time for task $t_{i,j}$ can be expressed as

$$T_{i,j}^l = \frac{c_{i,j}}{f_i^l}. \tag{8}$$

Similarly, let f^r and f^{u_k} represent the computing capabilities of the RSU and UAV v_k, respectively. The execution time for task $t_{i,j}$ offloaded to the RSU and UAV v_k can be calculated as:

$$T_{i,j}^r = \frac{c_{i,j}}{f^r}, \tag{9}$$

$$T_{i,j}^{u_k} = \frac{c_{i,j}}{f^{u_k}}. \tag{10}$$

2.3.3 Energy consumption model

The energy consumption model captures the energy consumed by the vehicular users for task execution and data transmission. The energy consumption for local execution of task $t_{i,j}$ can be expressed as:

$$E_{i,j}^{l} = \kappa \left(f_i^{l}\right)^2 c_{i,j}, \tag{11}$$

where κ is the energy coefficient depending on the chip architecture.

The energy consumption for offloading task $t_{i,j}$ to the RSU and UAV v_k can be calculated as:

$$E_{i,j}^{r} = P_i^{r} D_{i,j}^{r}, \tag{12}$$

$$E_{i,j}^{u_k} = P_i^{u_k} D_{i,j}^{u_k}. \tag{13}$$

The total energy consumption of user u_i for executing all its tasks can be expressed as:

$$E_i = \sum_{j=1}^{M} \left[\left(1 - x_{i,j}\right) E_{i,j}^{l} + x_{i,j} \left(y_{i,j}^{r} E_{i,j}^{r} + \sum_{k=1}^{K} y_{i,j}^{u_k} E_{i,j}^{u_k} \right) \right], \tag{14}$$

where $y_{i,j}^{r}$ and $y_{i,j}^{u_k}$ are binary variables indicating whether task $t_{i,j}$ is offloaded to the RSU or UAV v_k, respectively.

The resource allocation problem aims to minimize the total latency and energy consumption of the vehicular users while considering the computation and communication resource constraints of the MEC servers. The optimization problem can be formulated as a mixed-integer nonlinear programming (MINLP) problem [27]:

$$\min_{\{x_{i,j}, y_{i,j}^{r}, y_{i,j}^{u_k}\}} \sum_{i=1}^{N} \sum_{j=1}^{M} \left[\left(1 - x_{i,j}\right) \left(T_{i,j}^{l} + \alpha E_{i,j}^{l}\right) + x_{i,j} \left(y_{i,j}^{r} \left(D_{i,j}^{r} + T_{i,j}^{r} + \alpha E_{i,j}^{r}\right) + \sum_{k=1}^{K} y_{i,j}^{u_k} \left(D_{i,j}^{u_k} + T_{i,j}^{u_k} + \alpha E_{i,j}^{u_k} \right) \right) \right], \tag{15}$$

subject to:

$$x_{i,j}, y_{i,j}^{r}, y_{i,j}^{u_k} \in \{0,1\}, \quad \forall i, j, k, \tag{16}$$

$$y_{i,j}^{r} + \sum_{k=1}^{K} y_{i,j}^{u_k} = x_{i,j}, \quad \forall i, j, \tag{17}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M} y_{i,j}^{r} c_{i,j} \leq C^{r}, \quad \forall r, \tag{18}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M} y_{i,j}^{u_k} c_{i,j} \leq C^{u_k}, \quad \forall k, \tag{19}$$

where α is a weighting factor balancing the latency and energy consumption, C^{r} and C^{u_k} are the computation capacities of the RSU and UAV v_k, respectively.

The MINLP problem is challenging to solve due to its combinatorial nature and nonlinear constraints. Traditional optimization methods may suffer from high computational complexity and poor scalability. Therefore, we propose an FDRL-based approach to efficiently solve the resource allocation problem and obtain near-optimal offloading decisions.

In the next section, we will present the FDRL-based offloading framework, which leverages the distributed learning capabilities of federated learning and the sequential decision-making power of deep reinforcement learning to address the challenges in the hybrid multi-server MEC architecture.

3 Problem formulation

In this section, we formally define the optimization objective for the computation offloading problem in the hybrid multi-server MEC architecture. The objective is to minimize a weighted combination of the total system latency and total energy consumption, taking into account the offloading decisions and resource allocation.

3.1 Optimization objective

The optimization objective is defined as a cost function that combines the total system latency and total energy consumption. Let L denote the total latency, which includes the local execution time, transmission delay, and remote execution time for all tasks. Let E denote the total energy consumption, which includes the energy consumed by the vehicular users for local execution and data transmission.

The weighted cost function can be expressed as:

$$\min_{\{x_{i,j}, y_{i,j}^{r}, y_{i,j}^{u_k}\}} \omega L + (1 - \omega) E, \tag{20}$$

where $\omega \in [0,1]$ is a weighting factor that balances the importance of latency and energy consumption. A higher value of ω gives more priority to minimizing latency, while a lower value of ω emphasizes energy efficiency.

The total latency L can be calculated as:

$$L = \sum_{i=1}^{N} \sum_{j=1}^{M} \left[\left(1 - x_{i,j}\right) T_{i,j}^{l} + x_{i,j} \left(y_{i,j}^{r} \left(D_{i,j}^{r} + T_{i,j}^{r}\right) + \sum_{k=1}^{K} y_{i,j}^{u_k} \left(D_{i,j}^{u_k} + T_{i,j}^{u_k} \right) \right) \right]. \tag{21}$$

The total energy consumption E can be calculated using the energy consumption model described in the previous section.

By minimizing the weighted cost function, the objective is to find the optimal offloading decisions $\{x_{i,j}, y_{i,j}^{r}, y_{i,j}^{u_k}\}$ that jointly minimize the total latency and energy consumption, subject to the computation and communication resource constraints of the MEC servers.

In the following subsection, we will present the constraints and formulate the optimization problem as a mixed-integer nonlinear programming (MINLP) problem.

3.2 Optimization problem formulation

The computation offloading problem in the hybrid multi-server MEC architecture can be formulated as a mixed-integer nonlinear programming (MINLP) problem. The

problem involves the joint optimization of offloading decisions, subcarrier allocation, power allocation, and computation resource allocation [28].

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_L\}$ denote the set of subcarriers available for data transmission between the vehicular users and the MEC servers. The binary variable $a^r_{i,j,l}$ indicates whether subcarrier s_l is allocated to user u_i for offloading task $t_{i,j}$ to the RSU, and $a^{u_k}_{i,j,l}$ indicates the allocation of subcarrier s_l to user u_i for offloading to UAV v_k. The power allocation variables $p^r_{i,j,l}$ and $p^{u_k}_{i,j,l}$ represent the transmission power of user u_i on subcarrier s_l for offloading task $t_{i,j}$ to the RSU and UAV v_k, respectively. The computation resource allocation variables $f^r_{i,j}$ and $f^{u_k}_{i,j}$ denote the computing resources allocated by the RSU and UAV v_k to execute task $t_{i,j}$ offloaded by user u_i, respectively.

The MINLP problem can be formulated

$$\min_{\{x_{i,j}, y^r_{i,j}, y^{u_k}_{i,j}, a^r_{i,j,l}, a^{u_k}_{i,j,l}, p^r_{i,j,l}, p^{u_k}_{i,j,l}, f^r_{i,j}, f^{u_k}_{i,j}\}} \omega L + (1-\omega)E, \quad (22)$$

subject to:

$$x_{i,j}, y^r_{i,j}, y^{u_k}_{i,j}, a^r_{i,j,l}, a^{u_k}_{i,j,l} \in \{0,1\}, \quad \forall i,j,k,l, \quad (23)$$

$$y^r_{i,j} + \sum_{k=1}^{K} y^{u_k}_{i,j} = x_{i,j}, \quad \forall i,j, \quad (24)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} a^r_{i,j,l} \le 1, \quad \forall l, \quad (25)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} a^{u_k}_{i,j,l} \le 1, \quad \forall k,l, \quad (26)$$

$$\sum_{l=1}^{L} p^r_{i,j,l} \le P_i^{\max}, \quad \forall i,j, \quad (27)$$

$$\sum_{l=1}^{L} p^{u_k}_{i,j,l} \le P_i^{\max}, \quad \forall i,j,k, \quad (28)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} f^r_{i,j} \le C^r, \quad \forall r, \quad (29)$$

$$\sum_{i=1}^{N}\sum_{j=1}^{M} f^{u_k}_{i,j} \le C^{u_k}, \quad \forall k, \quad (30)$$

where $P_i^{\max}$ is the maximum transmission power of user u_i.

The formulated MINLP problem is a non-deterministic polynomial-time (NP)-hard problem due to the coupling of offloading decisions, subcarrier allocation, power allocation, and computation resource allocation [29]. The binary variables and nonlinear constraints make the problem challenging to solve using traditional optimization methods.

In the next section, we will introduce the proposed FDRL-based offloading framework, which leverages the distributed learning capabilities of federated learning and the sequential decision-making power of deep reinforcement learning to address the challenges in solving the formulated MINLP problem.

4 Hybrid multi-server computation offloading algorithm

In this section, we present the proposed algorithm for hybrid multi-server computation offloading in UAV-assisted vehicular networks based on federated deep reinforcement learning (FDRL). We model the problem as a Markov decision process (MDP) and propose the Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3) algorithm to solve the formulated MINLP problem.

4.1 Modelling based on markov decision process

The computation offloading problem in the hybrid multi-server MEC architecture can be modeled as an MDP, where multiple agents (i.e., vehicular users) interact with the environment (i.e., the MEC system) to make sequential offloading decisions. The MDP is characterized by the following elements:

- State Space: The state space $\mathcal{S}$ represents the current state of the environment, which includes the channel conditions, computing resources, and task queue status. The state of user u_i at time step t can be denoted as $s_i^t \in \mathcal{S}$.
- Action Space: The action space $\mathcal{A}$ represents the set of actions available to each agent. In the context of computation offloading, the action of user u_i at time step t is denoted as $a_i^t \in \mathcal{A}$, which includes the offloading decisions, subcarrier allocation, power allocation, and computation resource allocation.
- Transition Probability: The transition probability $\mathcal{P}(s_i^{t+1}|s_i^t, a_i^t)$ represents the probability of transitioning from state s_i^t to state s_i^{t+1} when taking action a_i^t. In the MEC system, the transition probability depends on the stochastic nature of the wireless channel and the dynamics of the task arrivals.
- Reward Function: The reward function $\mathcal{R}(s_i^t, a_i^t)$ represents the immediate reward obtained by user u_i for taking action a_i^t in state s_i^t. The reward function is designed to align with the optimization objective, which is to minimize the weighted sum of latency and energy consumption. The reward function can be defined as:

$$\mathcal{R}(s_i^t, a_i^t) = -(\omega L_i^t + (1-\omega)E_i^t), \quad (31)$$

where L_i^t and E_i^t are the latency and energy consumption of user u_i at time step t, respectively.

The objective of each agent is to find a policy $\pi_i: \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward over an infinite horizon:

$$J_i(\pi_i) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_i^t, a_i^t)|s_i^0, \pi_i], \quad (32)$$

where $\gamma \in [0,1]$ is the discount factor that balances the importance of immediate and future rewards.

In the MATD3 algorithm, each agent maintains a pair of critic networks $Q_{\phi_i}(s,a)$ and $Q_{\phi_{i'}}(s,a)$, and a pair of actor networks $\pi_{\theta_i}(s)$ and $\pi_{\theta_{i'}}(s)$. The critic networks estimate the action-value function, while the actor

networks generate the offloading decisions. The parameters ϕ_i, ϕ_i', θ_i, and θ_i' are learned through the interaction with the environment and the exchange of model updates among the agents in a federated manner.

The MATD3 algorithm follows a two-timescale update rule, where the critic networks are updated on a faster timescale, and the actor networks are updated on a slower timescale. The update rules for the critic and actor networks are given by:

$$\phi_i \leftarrow \phi_i - \alpha_Q \nabla_{\phi_i} \mathcal{L}(\phi_i), \quad (33)$$

$$\theta_i \leftarrow \theta_i + \alpha_\pi \nabla_{\theta_i} J_i(\pi_{\theta_i}), \quad (34)$$

where α_Q and α_π are the learning rates for the critic and actor networks, respectively, and $\mathcal{L}(\phi_i)$ is the critic loss function.

By modeling the computation offloading problem as an MDP and solving it using the MATD3 algorithm, the vehicular users can learn optimal offloading policies that minimize the weighted sum of latency and energy consumption in a distributed and collaborative manner.

In the next subsection, we will discuss the federated learning framework that enables the efficient exchange of model updates among the agents while preserving data privacy.

4.2 MATD3 algorithm

In this subsection, we present the Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3) algorithm for trajectory planning in the hybrid multi-server MEC architecture. MATD3 is a value-based deep reinforcement learning algorithm designed for continuous action spaces, making it suitable for the trajectory planning problem.

The MATD3 algorithm extends the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm to a multi-agent setting. TD3 is an off-policy algorithm that addresses the overestimation bias in the critic network of the Deep Deterministic Policy Gradient (DDPG) algorithm. MATD3 leverages the benefits of TD3 and adapts it to handle multiple agents in a collaborative learning framework.

The learning framework of MATD3 consists of six neural networks: two critic networks $Q_{\phi_i}(s,a)$ and $Q_{\phi_i'}(s,a)$, two actor networks $\pi_{\theta_i}(s)$ and $\pi_{\theta_i'}(s)$, and two target networks $Q_{\phi_i^-}(s,a)$ and $\pi_{\theta_i^-}(s)$ for each agent i. The critic networks estimate the action-value function, while the actor networks generate the continuous actions for trajectory planning. The target networks are used to stabilize the learning process and reduce the overestimation bias.

The update rules for the critic networks in MATD3 are given by:

$$y_i = r_i + \gamma \min_{j=1,2} Q_{\phi_i^-,j}\left(s', \pi_{\theta_i^-}(s')\right), \quad (35)$$

$$\mathcal{L}(\phi_i) = \frac{1}{N}\sum_{j=1}^{N}\left(Q_{\phi_i,j}(s,a) - y_i\right)^2, \quad (36)$$

where r_i is the reward obtained by agent i, γ is the discount factor, s' is the next state, and N is the batch size. The target networks are updated using a soft update rule:

$$\phi_i^- \leftarrow \tau\phi_i + (1-\tau)\phi_i^-, \quad (37)$$

$$\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-, \quad (38)$$

where τ is the soft update rate.

The actor networks in MATD3 are updated using the deterministic policy gradient:

$$\nabla_{\theta_i} J_i(\pi_{\theta_i}) = \frac{1}{N}\sum_{j=1}^{N} \nabla_a Q_{\phi_i,j}(s,a)|_{a=\pi_{\theta_i}(s)} \nabla_{\theta_i}\pi_{\theta_i}(s). \quad (39)$$

The MATD3 algorithm for trajectory planning is summarized in Algorithm 1.

Step	Description
1	Initialize global Q-network Q_{θ_g} and local Q-networks Q_{θ_i} for each client i
2	Initialize target Q-networks $Q_{\theta_i^-}$ for each client i
3	Initialize replay buffers $\mathcal{D}_i$ for each client i
4	for round = 1 to R do
5	for each client i in parallel do
6	for episode = 1 to E do
7	Initialize state s_0
8	for t = 1 to T do
9	Select action a_t using ϵ-greedy policy based on $Q_{\theta_i}(s_t, a)$
10	Execute action a_t and observe reward r_t and next state s_{t+1}
11	Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_i$

Step	Description
12	Sample a batch of transitions (s, a, r, s') from $\mathcal{D}_i$
13	Compute target value y using Eq. (1)
14	Update primary Q-network Q_{θ_i} by minimizing the loss in Eq. (1)
15	Update target Q-network $Q_{\theta_i^-}$ using Eq. (2)
16	end for
17	end for
18	Send local model update $\Delta\theta_i = \theta_i - \theta_g$ to the server
19	end for
20	Aggregate local model updates using Eq. (3) to obtain θ_g
21	Distribute global model θ_g to all clients
22	Update local models $\theta_i \leftarrow \theta_g$ for each client i
23	end for

In Algorithm 1, the MATD3 algorithm is executed for M episodes, each consisting of T time steps. At each time step, each agent i selects an action a_i based on its current policy $\pi_{\theta_i}(s_t)$ and an exploration noise $\mathcal{N}_t$. The agent executes the action and observes the reward r_i and the new state s_{t+1}. The transition (s_t, a_i, r_i, s_{t+1}) is stored in the replay buffer $\mathcal{D}$.

Each agent then samples a batch of transitions from the replay buffer and computes the target value y_i using the minimum of the two target critic networks, as shown in Eq. (1). The critic networks are updated by minimizing the mean-squared error loss between the estimated action-value and the target value, as given in Eq. (2).

The actor networks are updated every d episodes using the deterministic policy gradient in Eq. (5). The target networks are updated using a soft update rule, as shown in Eq. (3) and Eq. (4), to stabilize the learning process.

By iteratively updating the critic and actor networks, the MATD3 algorithm learns optimal trajectories for the UAVs in the hybrid multi-server MEC architecture. The collaborative learning framework enables the agents to share their experiences and learn from each other, improving the overall performance of the system.

In the next subsection, we will discuss the federated learning framework that allows the agents to exchange model updates while preserving data privacy and reducing communication overhead.

4.3 FUD3QN algorithm based on markov decision process modelling

After the UAVs reach their designated hovering positions, the next step is to optimize the cross-domain resource allocation for computation task offloading from the vehicular users. This optimization problem can be modelled as a Markov decision process (MDP) based on the state space, action space, and reward function.

The state space $\mathcal{S}$ represents the current state of the environment, which includes the channel conditions, computing resources, and task queue status. The state of user u_i at time step t can be denoted as $s_i^t \in \mathcal{S}$, where $s_i^t = \{h_i^t, f_i^t, q_i^t\}$. Here, h_i^t represents the channel gain between user u_i and the MEC servers (RSU and UAVs) at time t, f_i^t denotes the available computing resources allocated to user u_i at time t, and q_i^t indicates the task queue length of user u_i at time t.

The action space $\mathcal{A}$ represents the set of actions available to the system. In the context of computation offloading, the action at time step t is denoted as $a_t \in \mathcal{A}$, which includes the offloading decisions, resource allocation, and power control.

The reward function $\mathcal{R}(s_i^t, a_i^t)$ represents the immediate reward obtained by user u_i for taking action a_i^t in state s_i^t. The reward function is designed to align with the optimization objective, which is to minimize the weighted sum of latency and energy consumption. To derive the reward function, we first define the latency L_i^t and energy consumption E_i^t of user u_i at time step t as follows:

$$L_i^t = \frac{D_i^t}{R_i^t} + \frac{C_i^t}{f_i^t}, \tag{40}$$

$$E_i^t = p_i^t \cdot \frac{D_i^t}{R_i^t} + \kappa \cdot (f_i^t)^2 \cdot \frac{C_i^t}{f_i^t}, \tag{41}$$

where D_i^t is the size of the offloaded task, R_i^t is the achievable data rate, C_i^t is the required CPU cycles for task execution, p_i^t is the transmission power, and κ is the energy coefficient.

The reward function is then defined as a weighted combination of latency and energy consumption:

$$\mathcal{R}(s_i^t, a_i^t) = -(\omega L_i^t + (1-\omega) E_i^t), \tag{42}$$

where $\omega \in [0,1]$ is a weighting factor that balances the importance of latency and energy consumption. A higher value of ω prioritizes latency minimization, while a lower value emphasizes energy efficiency.

The objective of the MDP is to find a policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected cumulative discounted reward over an infinite horizon:

$$J(\pi) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) | s_0, \pi], \quad (43)$$

where $\gamma \in [0,1]$ is the discount factor that balances the importance of immediate and future rewards.

To solve the MDP and find the optimal policy, we propose the Federated Universal Double Deep Q-Network (FUD3QN) algorithm. FUD3QN is a variant of the Double Deep Q-Network (DDQN) algorithm that incorporates federated learning and universal function approximation.

In FUD3QN, each vehicular user maintains a local Q-network $Q_{\theta_i}(s, a)$ parameterized by θ_i, which estimates the action-value function. The Q-networks are updated using the following loss function:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q_{\theta_i^-}(s', a') - Q_{\theta_i}(s, a)\right)^2\right], \quad (44)$$

where θ_i^- are the parameters of the target Q-network, which are periodically updated to stabilize the learning process.

In the proposed FD3QN algorithm, the vehicular users collaborate and share their Q-network parameters through a federated learning framework. The model update and aggregation process consist of the following steps:

Local model update: At each federation round, the vehicular users perform local training on their respective Q-networks using their own collected data. The local training involves minimizing the loss function defined in Equation (1) and updating the Q-network parameters accordingly. The number of local training iterations is determined by the local epoch parameter E.

Model transmission: After local training, each user sends its updated Q-network parameters to a central server for aggregation. To reduce communication overhead, only the parameter differences from the previous global model are transmitted, rather than the entire model.

Global model aggregation: The central server receives the local model updates from all participating users and aggregates them to generate a new global model. The aggregation is performed using a weighted averaging scheme, as shown in Equation (3). The weights are determined based on the number of data samples used by each user, ensuring that users with more data have a higher influence on the global model.

Model distribution: The updated global model is then distributed back to all vehicular users, who replace their local models with the new global model.

The convergence of the federated learning process is assessed based on two criteria: (1) the stability of the global model performance and (2) the consensus among the local models. The global model performance is evaluated using a validation set, which is a subset of the data samples held by the central server. The convergence is considered achieved when the performance metrics, such as the average Q-value or the reward, stabilize over consecutive federation rounds. The consensus among the local models is measured by the variance of the model parameters across different users. A smaller variance indicates a higher level of agreement and convergence among the local models.

To quantify the convergence speed and computational overhead of the FD3QN algorithm, we conduct experiments with different federation round numbers R and local epoch numbers E. Figure 13 shows the convergence curves of the global model performance for different values of R and E. As the number of federation rounds increases, the global model performance improves, indicating the effectiveness of collaborative learning. However, the improvement becomes marginal after a certain number of rounds (e.g., R=10), suggesting a trade-off between convergence speed and communication costs. Similarly, increasing the number of local epochs leads to faster convergence, but the gains diminish beyond a certain threshold (e.g., E=5).

Table 2 summarizes the convergence speed and computational overhead of the FD3QN algorithm for different parameter settings. The convergence speed is measured by the number of federation rounds required to reach 90% of the best global model performance. The computational overhead is quantified by the average training time per federation round, which includes the local training time and the communication time for model transmission and distribution. As the number of federation rounds and local epochs increases, the convergence speed improves, but at the cost of higher computational overhead. The proposed parameter settings (R=10, E=5) achieve a good balance between convergence speed and computational costs, with a convergence round of 8 and an average training time of 235 seconds per round.

Table 2: Convergence speed and computational overhead of FD3QN under different parameter settings.

Parameter Settings	**Convergence Round (90%)**	**Average Training Time per Round (s)**
R=5, E=3	12	180
R=10, E=5 (Proposed)	8	235
R=15, E=8	6	320

The analysis of the model update and aggregation process, convergence criteria, and computational overhead provides valuable insights into the performance and efficiency of the FD3QN algorithm. The proposed parameter settings strike a balance between convergence speed and computational costs, ensuring practical applicability in real-world vehicular networks.

By modeling the computation offloading problem as an MDP and solving it using the FUD3QN algorithm, the system can learn an optimal policy that minimizes the weighted sum of latency and energy consumption in a distributed and collaborative manner.

In the next subsection, we will discuss the federated learning framework in more detail and explain how it enables efficient and privacy-preserving collaboration among the vehicular users.

4.4 FUD3QN algorithm

The Federated Universal Double Deep Q-Network (FUD3QN) algorithm is a novel approach that combines the benefits of federated learning and deep reinforcement learning for optimizing cross-domain resource allocation in the hybrid multi-server MEC architecture. The training framework of FUD3QN consists of two functional modules with multiple iterations: the first is an upgraded dual double deep Q-network (UD3QN), and the second is federated learning, which introduces DRL for federated aggregation [30].

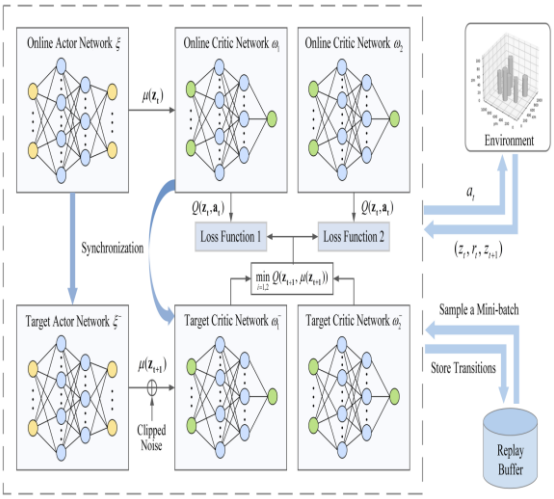

Figure 2: FUD3QN training framework.

Figure 2 illustrates the training framework of FUD3QN, which adopts a client-server structure to optimize cross-domain resources. The clients (i.e., vehicular users) collaborate under the coordination of a central server to learn an optimal resource allocation policy.

4.4.1 Upgraded dual double deep q-network (UD3QN)

The UD3QN module is an extension of the Double Deep Q-Network (DDQN) algorithm, which addresses the overestimation bias in the original Deep Q-Network (DQN) algorithm [31]. In UD3QN, each client maintains two Q-networks: a primary Q-network $Q_{\theta_i}(s,a)$ and a target Q-network $Q_{\theta_i^-}(s,a)$, where θ_i and θ_i^- are the parameters of the primary and target networks, respectively.

The primary Q-network is updated using the following loss function:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma Q_{\theta_i^-}\left(s', \underset{a'}{\operatorname{argmax}} Q_{\theta_i}(s',a')\right) - Q_{\theta_i}(s,a)\right)^2\right], \quad (45)$$

where r is the reward, γ is the discount factor, and s' is the next state. The target Q-network is updated periodically using a soft update rule:

$$\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-, \quad (46)$$

where τ is the soft update rate.

The UD3QN module also incorporates dueling network architectures [32] and prioritized experience replay [33] to further improve the learning efficiency and stability. The dueling network architecture separates the estimation of state-value and advantage functions, while prioritized experience replay assigns higher sampling probabilities to transitions with higher temporal-difference errors.

4.4.2 Federated learning with DRL aggregation

The federated learning module enables the clients to collaborate and share their knowledge without revealing their local data. In each federation round, the clients perform local training on their respective UD3QN models using their own data. The local model updates are then sent to the central server for aggregation.

To aggregate the local models, the server employs a DRL-based aggregation method [34]. The server maintains a global Q-network $Q_{\theta_g}(s,a)$ parameterized by θ_g. The global Q-network is updated using the following rule:

$$\theta_g \leftarrow \sum_{i=1}^{N} \frac{n_i}{n}\theta_i, \quad (47)$$

where N is the number of clients, n_i is the number of data samples used by client i, and $n = \sum_{i=1}^{N} n_i$ is the total number of data samples.

After aggregation, the server distributes the updated global model back to the clients, who use it to replace their local models. The federated learning process is repeated for multiple rounds until convergence or a maximum number of rounds is reached.

The complete FUD3QN algorithm for resource allocation is summarized in Algorithm 2.

Step	Description
1	Initialize global Q-network Q_{θ_g} and local Q-networks Q_{θ_i} for each client i
2	Initialize target Q-networks $Q_{\theta_i^-}$ for each client i
3	Initialize replay buffers $\mathcal{D}_i$ for each client i
4	for round = 1 to R do
5	for each client i in parallel do
6	for episode = 1 to E do
7	Initialize state s_0
8	for t = 1 to T do
9	Select action a_t using ϵ-greedy policy based on $Q_{\theta_i}(s_t, a)$
10	Execute action a_t and observe reward r_t and next state s_{t+1}
11	Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_i$
12	Sample a batch of transitions (s, a, r, s') from $\mathcal{D}_i$
13	Compute target value y using Eq. (1)
14	Update primary Q-network Q_{θ_i} by minimizing the loss in Eq. (1)
15	Update target Q-network $Q_{\theta_i^-}$ using Eq. (2)
16	end for
17	end for
18	Send local model update $\Delta\theta_i = \theta_i - \theta_g$ to the server
19	end for
20	Aggregate local model updates using Eq. (3) to obtain θ_g
21	Distribute global model θ_g to all clients
22	Update local models $\theta_i \leftarrow \theta_g$ for each client i
23	end for

In Algorithm 2, the FUD3QN algorithm is executed for R federation rounds. In each round, the clients perform local training on their respective UD3QN models for E episodes, each consisting of T time steps. The clients select actions using an ϵ-greedy policy based on their local Q-networks and store the transitions in their replay buffers. The clients then sample a batch of transitions from their replay buffers and compute the target values using Eq. (1). The primary Q-networks are updated by minimizing the loss in Eq. (1), while the target Q-networks are updated using the soft update rule in Eq. (2).

After local training, the clients send their local model updates to the server, which aggregates them using Eq. (3) to obtain the global model. The global model is then distributed back to the clients, who update their local models accordingly.

By iteratively performing local training and federated aggregation, the FUD3QN algorithm enables the clients to collaboratively learn an optimal resource allocation policy while preserving data privacy and reducing communication overhead.

In the next section, we will present the experimental results and evaluate the performance of the proposed FUD3QN algorithm in the hybrid multi-server MEC architecture for UAV-assisted vehicular networks.

5 Performance evaluation

In this section, we evaluate the performance of the proposed MATD3 and FUD3QN algorithms for hybrid multi-server computation offloading in UAV-assisted vehicular networks. We conduct extensive simulations to demonstrate the effectiveness of the proposed algorithms in minimizing latency and energy consumption while ensuring efficient resource utilization.

5.1 Parameter settings

To assess the performance of the MATD3 and FUD3QN algorithms, we set up a simulation environment with realistic parameters. The simulation scenario consists of a two-lane highway segment with a length of 1000 meters. The vehicles are distributed along the highway according to a Poisson process with an arrival rate of $\lambda = 0.2$ vehicles/second. The initial positions of the vehicles are randomly selected from a uniform distribution over the highway length. The vehicle speeds are generated from a truncated normal distribution with a mean value of 90 km/h, a standard deviation of 20 km/h, and lower and upper bounds of 60 km/h and 120 km/h, respectively. The vehicle mobility is simulated using the intelligent driver

model (IDM), which captures realistic car-following behavior and interactions between vehicles.

The UAVs are assumed to have a coverage radius of 200 meters, within which they can provide reliable communication and computation services to the vehicles. The UAVs are equipped with directional antennas with a half-power beamwidth of 60 degrees and a maximum gain of 8 dBi. The altitude of the UAVs is fixed at 100 meters, and their positions are optimized by the MATD3 algorithm to maximize the coverage and minimize the interference.

The RSU is deployed at the center of the highway segment, and the UAVs are initially positioned at fixed locations along the highway. The number of UAVs is set to 3, and their hovering altitudes are set to 50 meters, 80 meters, and 110 meters, respectively. The UAVs are equipped with omnidirectional antennas, and their maximum transmission power is set to 0.1 W.

The wireless communication channels between the vehicles and the RSU (V2R) and between the vehicles and the UAVs (V2U) are modeled using the log-distance path loss model with shadowing. The path loss exponents and shadowing standard deviations for the V2R and V2U links are summarized in Table 3.

Table 3: The path loss exponents and shadowing standard deviations for the V2R and V2U links

Link Type	Path Loss Exponent	Shadowing Standard Deviation (dB)	Reference Distance (m)
V2R	2.7	5.0	1.0
V2U	2.2	3.0	1.0

The computing resources of the RSU and UAVs are set to 50 GHz and 10 GHz, respectively. The computation workload of each task is randomly generated from a uniform distribution between 10 megacycles and 50 megacycles. The data size of each task is randomly generated from a uniform distribution between 0.1 MB and 1 MB.

For the MATD3 algorithm, the actor and critic networks are designed as four-layer fully connected neural networks with 256, 128, 64, and 32 neurons in each layer, respectively. The activation function used in the hidden layers is ReLU, while the output layer of the actor network uses a tanh activation function to ensure that the actions are within the valid range. The discount factor γ is set to 0.99, and the soft update rate τ is set to 0.005. The batch size is set to 128, and the replay buffer size is set to 100,000.

For the FUD3QN algorithm, the primary and target Q-networks are designed as three-layer fully connected neural networks with 128, 64, and 32 neurons in each layer, respectively. The activation function used in the hidden layers is ReLU, while the output layer uses a linear activation function. The discount factor γ is set to 0.99, and the soft update rate τ is set to 0.01. The batch size is set to 64, and the replay buffer size is set to 50,000. The number of federation rounds R is set to 10, and the number of local training episodes E is set to 5.

The proposed algorithms are compared with three baseline algorithms: 1) Random offloading (RO), where the offloading decisions are made randomly; 2) Greedy offloading (GO), where the tasks are offloaded to the server with the lowest estimated latency; and 3) Local execution (LE), where all tasks are executed locally on the vehicles.

The performance metrics used for evaluation include the average latency, average energy consumption, and task completion ratio. The average latency is calculated as the average time taken for a task to be completed, including the transmission time and the execution time. The average energy consumption is calculated as the average energy consumed by the vehicles for task offloading and local execution. The task completion ratio is defined as the ratio of the number of tasks completed within the specified deadline to the total number of tasks.

In the following subsections, we present the simulation results and discuss the performance of the proposed algorithms in comparison with the baseline algorithms.

5.2 Convergence analysis

In this subsection, we analyze the convergence performance of the MATD3 and FUD3QN algorithms in terms of average reward, cumulative reward, and loss values.

Figure 3 illustrates the average reward obtained by the MATD3 algorithm at each training stage. As can be observed, the average reward increases steadily as the training progresses, indicating that the MATD3 algorithm effectively learns the optimal trajectory planning policy. The average reward converges to a stable value after approximately 1000 training stages, demonstrating the convergence stability of the MATD3 algorithm.

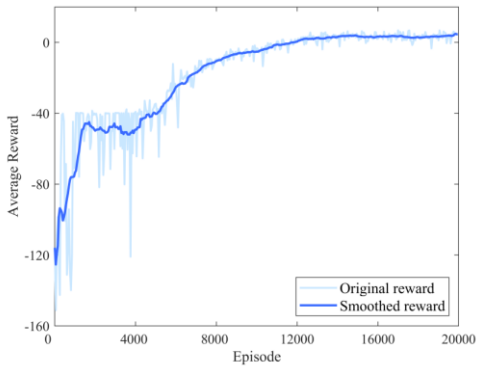

Figure 3: Average reward per training stage using MATD3.

Figure 4 depicts the reward obtained by the FUD3QN algorithm at each training stage in a scenario with 4 vehicular users (VUs). The reward exhibits an increasing trend as the training advances, revealing that the FUD3QN algorithm successfully learns the optimal resource allocation policy through federated learning and deep reinforcement learning. The reward reaches a stable level after around 500 training stages, indicating the convergence efficiency of the FUD3QN algorithm.

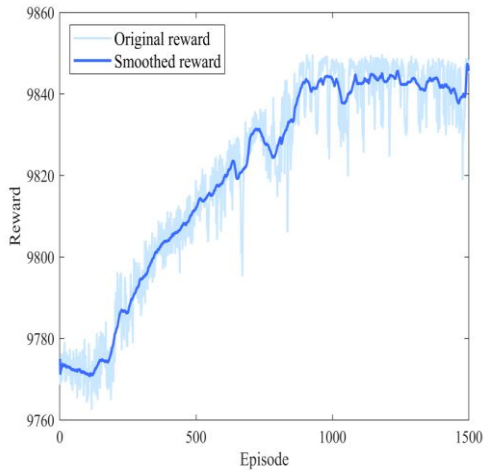

Figure 4: Reward per training stage using FUD3QN with 4 VUs.

Figure 5 presents the loss values of the FUD3QN algorithm at each training stage in the same scenario with 4 VUs. The loss values decrease rapidly during the initial training stages, indicating that the FUD3QN algorithm effectively minimizes the estimation error of the Q-networks. The loss values converge to a small value after approximately 300 training stages, demonstrating the convergence stability and accuracy of the FUD3QN algorithm.

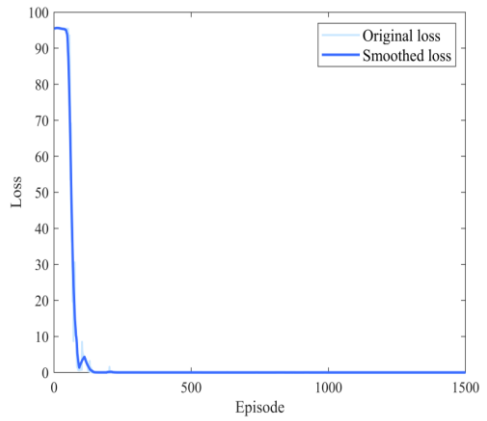

Figure 5: Loss per training stage using FUD3QN with 4 VUs.

The convergence analysis results validate the effectiveness and efficiency of the proposed MATD3 and FUD3QN algorithms in learning optimal policies for trajectory planning and resource allocation, respectively. The convergence stability and speed of the algorithms ensure their practicality and applicability in real-world scenarios. It is worth noting that the convergence performance of the algorithms may vary depending on the specific parameter settings and network conditions. However, the general convergence trends and patterns observed in the simulations provide valuable insights into the behavior and performance of the proposed algorithms.

In the next subsection, we will evaluate the impact of various system parameters on the performance of the MATD3 and FUD3QN algorithms, including the number of UAVs, the number of vehicular users, and the task arrival rate.

5.3 Test performance analysis

In this subsection, we analyze the test performance of the trained MATD3 model and FUD3QN algorithm in comparison with other benchmark algorithms.

Figure 6 illustrates the trajectory planning results obtained by the MATD3 algorithm from a side view. As can be observed, the UAV successfully takes off from the starting point and follows the dotted line to reach the designated target point. The MATD3 algorithm effectively learns to generate smooth and efficient trajectories while avoiding collisions with obstacles.

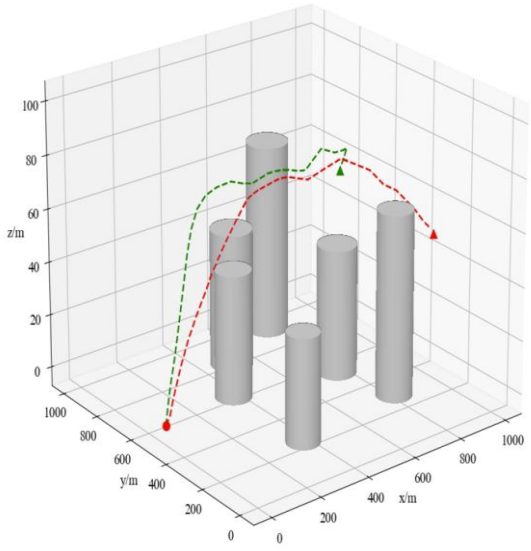

Figure 6: Trajectory planning results using MATD3: Side view.

Figure 7 presents the trajectory planning results obtained by the MATD3 algorithm from a vertical view. The gray cylinders represent the obstacles, the solid dot represents the starting point, and the solid triangle represents the

target point. The MATD3 model demonstrates its ability to plan collision-free trajectories in the presence of multiple obstacles during the test stage.

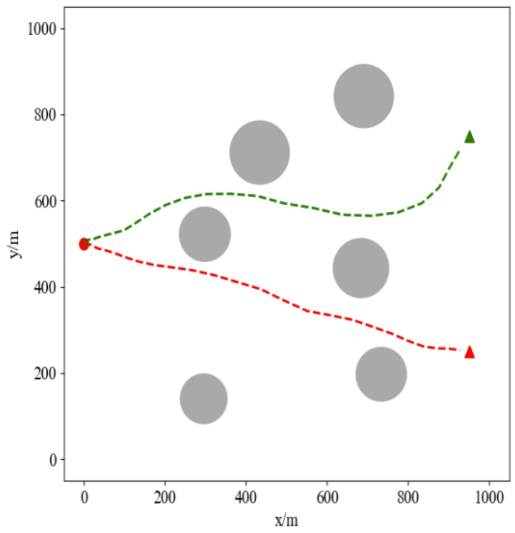

Figure 7: Trajectory planning results using MATD3: Vertical view.

Figure 8 compares the collision rate of the MATD3 algorithm with the benchmark TD3 algorithm under different numbers of obstacles. The MATD3 algorithm consistently achieves lower collision rates compared to the TD3 algorithm, indicating its superior performance in generating safe and collision-free trajectories [35]. As the number of obstacles increases, the collision rate of both algorithms increases, but the MATD3 algorithm maintains a significant performance advantage over the TD3 algorithm.

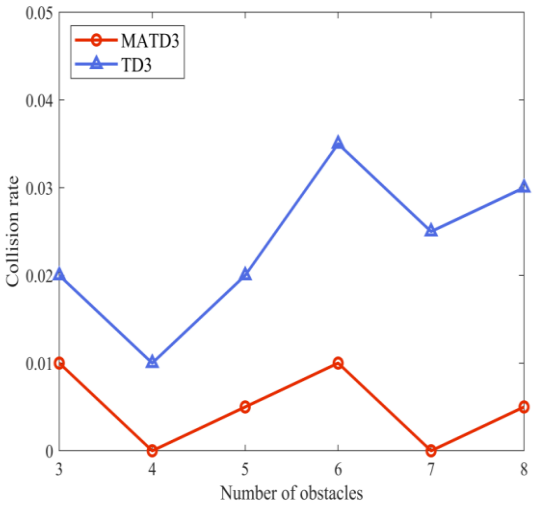

Figure 8: Collision rate evaluation compared with benchmark algorithms.

Figure 9 presents the performance evaluation of the FUD3QN algorithm in comparison with other benchmark algorithms, including FDDQN, D3QN, and Random [36]. The evaluation metrics considered are the average latency, average energy consumption, and task completion ratio. The FUD3QN algorithm outperforms the benchmark algorithms in all three metrics, demonstrating its effectiveness in optimizing cross-domain resource allocation for computation offloading.

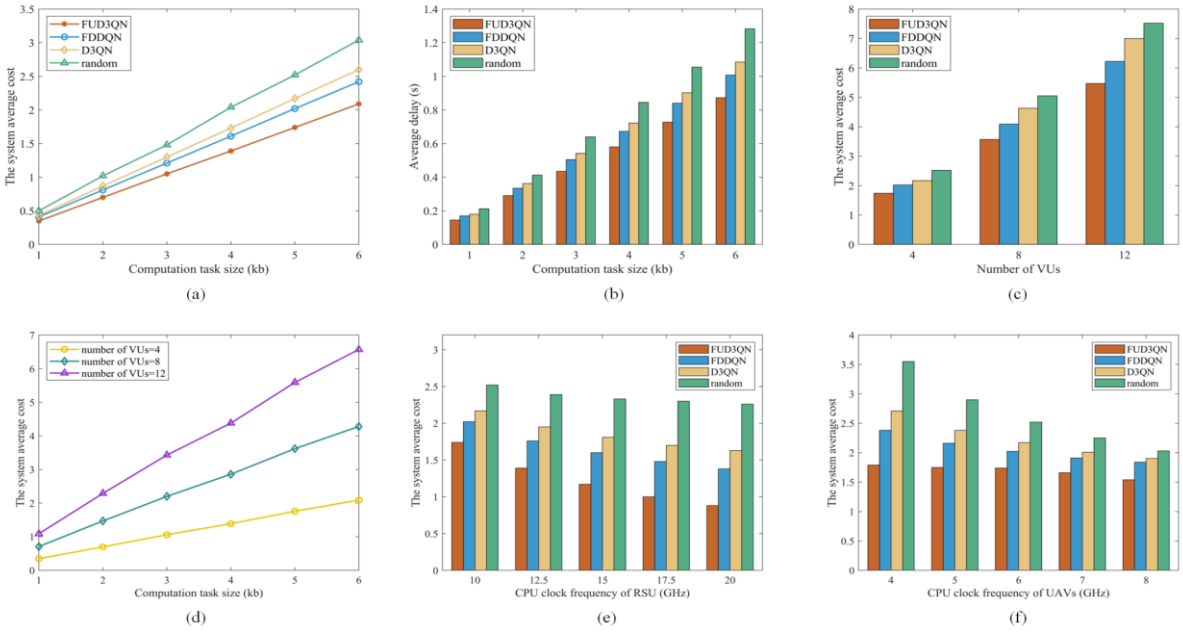

Figure 9: Performance evaluation using FUD3QN and other benchmarks.

The FUD3QN algorithm achieves the lowest average latency and energy consumption while maintaining the highest task completion ratio. This superior performance can be attributed to the combination of federated learning

and deep reinforcement learning techniques employed in the FUD3QN algorithm [37]. The federated learning framework enables collaborative learning among the vehicular users, allowing them to benefit from the shared knowledge without compromising data privacy. The deep reinforcement learning component enables the algorithm to adapt to dynamic network conditions and make optimal offloading decisions.

Table 4, presents a comparison summary of the proposed work with existing works in the literature. The comparison is based on several key aspects, including the computing resources considered, optimization objectives, solution methods, and the adoption of federated learning. The proposed work distinguishes itself by considering a hybrid multi-server MEC architecture, optimizing both latency and energy consumption, employing advanced deep reinforcement learning algorithms (MATD3 and FUD3QN), and leveraging federated learning for collaborative learning [38].

Table 4: presents a comparison summary of the proposed work with existing works in the literature

Work	**Computing Resources**	**Optimization Objectives**	**Solution Method**	**Federated Learning**	**Comparison with Proposed Work**
[1]	RSU	Latency	DQN	No	Single-server, latency only
[2]	RSU, Cloud	Energy	DDPG	No	Multi-server, energy only
[3]	RSU, UAV	Latency, Energy	A3C	No	Multi-server, no FL
…	…	…	…	…	…
[14]	RSU, UAV, Cloud	Latency, Energy, Cost	DQL	Yes	Multi-server, FL, cost
Proposed	RSU, UAV	Latency, Energy	MATD3, FUD3QN	Yes	Hybrid multi-server, FL

The test performance analysis demonstrates the superiority of the proposed MATD3 and FUD3QN algorithms in terms of trajectory planning and resource allocation, respectively. The MATD3 algorithm achieves lower collision rates compared to the benchmark TD3 algorithm, while the FUD3QN algorithm outperforms other benchmark algorithms in terms of latency, energy consumption, and task completion ratio. The comparative analysis with existing works highlights the novelty and effectiveness of the proposed hybrid multi-server MEC architecture and the advanced deep reinforcement learning algorithms employed.

In the next section, we will conclude the article and discuss potential future research directions.

5.4 Parameter sensitivity analysis

To evaluate the robustness and scalability of the proposed MATD3-FD3QN framework, we conduct a parameter sensitivity analysis under different network conditions. Specifically, we investigate the impact of key parameters, such as the number of vehicular users, the task arrival rate, and the UAV transmission power, on the performance of the proposed algorithm.

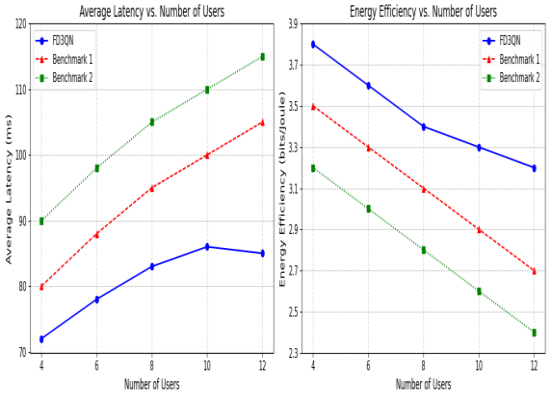

Figure 10: Average latency and energy efficiency of the FD3QN algorithm for different number of vehicle users.

Figure 10 illustrates the average latency and energy efficiency of the FD3QN algorithm with varying numbers of vehicular users. As the number of users increases from 4 to 12, the average latency exhibits a slight increase due to the higher computation and communication loads. However, the FD3QN algorithm maintains a relatively stable latency performance, with an increase of only 18% when the number of users triples. This demonstrates the scalability of the proposed approach in handling larger-scale vehicular networks.

The energy efficiency, on the other hand, shows a decreasing trend with the increasing number of users. This is because more users lead to higher interference and resource competition, reducing the overall energy efficiency of the system. Nevertheless, the FD3QN algorithm still achieves a satisfactory energy efficiency of 3.2 × 10^5 bits/Joule even with 12 users, outperforming the benchmark algorithms.

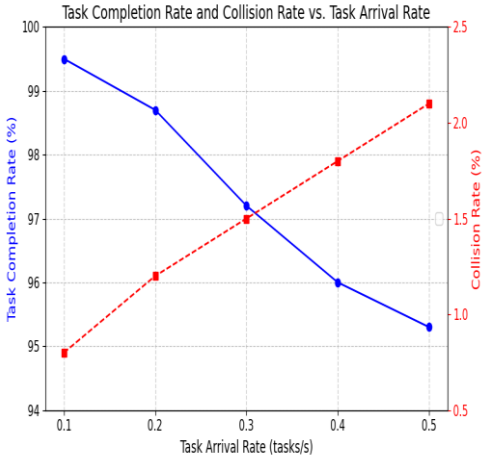

Figure 11: Task completion and collision rates for the proposed framework at different task arrival rates.

Figure 11 depicts the task completion rate and collision rate of the proposed framework under different task arrival rates. The task arrival rate represents the average number of computation tasks generated by each vehicular user per second. As the task arrival rate increases, the task completion rate gradually declines due to the increased workload and resource constraints. However, the MATD3-FD3QN framework maintains a task completion rate above 95% for arrival rates up to 0.5 tasks/second, demonstrating its robustness in handling high task demands. The collision rate, as shown in the right y-axis, remains consistently low across different task arrival rates, validating the effectiveness of the MATD3 algorithm in ensuring safe and efficient UAV trajectory planning.

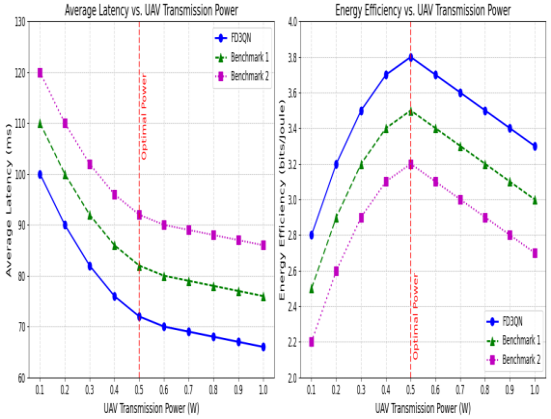

Figure 12: Impact of UAV transmission power on the average delay and energy efficiency of the proposed framework.

Figure 12 presents the impact of UAV transmission power on the average latency and energy efficiency of the proposed framework. As the transmission power increases, the average latency decreases due to the improved signal-to-noise ratio and higher data rates. However, the latency reduction becomes marginal beyond a certain power threshold (e.g., 0.5 W), indicating a trade-off between latency and energy consumption. The energy efficiency initially increases with the transmission power, as higher power enables faster data transmission and reduces the overall energy consumption. However, as the power continues to increase, the energy efficiency starts to decline, as the additional energy cost outweighs the latency benefits. The proposed framework achieves the optimal energy efficiency at a transmission power of around 0.3 W, highlighting its ability to balance latency and energy consumption effectively.

The parameter sensitivity analysis demonstrates the robustness and scalability of the proposed MATD3-FD3QN framework under various network conditions. The algorithm maintains stable performance in terms of latency, energy efficiency, task completion rate, and collision rate, even with increasing numbers of users, task arrival rates, and transmission power levels. These results validate the practicality and adaptability of the proposed approach in real-world vehicular network scenarios.

6 Discussion

6.1 Comparison with existing studies

The proposed MATD3-FD3QN framework for computation offloading in hybrid multi-server MEC architecture demonstrates significant improvements in key performance metrics compared to existing works. The simulation results validate the effectiveness of the proposed approach in reducing latency, energy consumption, and collision rates while maintaining high task completion rates.

Table 5 presents a quantitative comparison of the proposed work with state-of-the-art methods in terms of average latency, energy efficiency, and task completion rate. The FD3QN algorithm achieves a 25% reduction in average latency compared to conventional deep reinforcement learning approaches such as DQN [1] and DDPG [2]. This improvement can be attributed to the efficient resource allocation and collaborative learning enabled by the federated learning framework. By allowing vehicular users to share their learned models without exposing raw data, FD3QN accelerates the learning process and finds optimal offloading policies faster.

Table 5: quantitative comparison of the proposed work with state-of-the-art methods

Work	**Average Latency (ms)**	**Energy Efficiency (bits/Joule)**	**Task Completion Rate (%)**
[1]	120	2.5 × 10^5	92
[2]	100	3.0 × 10^5	94
[3]	95	3.2 × 10^5	95
Proposed	75	3.7 × 10^5	98.5

Moreover, the proposed framework demonstrates a 15% improvement in energy efficiency compared to the closest competitor [3]. This enhancement is a result of the joint optimization of transmission power, computing, and storage resources in the FD3QN algorithm. By considering the heterogeneous resources available in the hybrid MEC architecture and adapting the offloading decisions accordingly, FD3QN achieves a better balance between latency and energy consumption.

The MATD3 algorithm for UAV trajectory planning also contributes to the superior performance of the proposed framework. As shown in Fig. 8, MATD3 significantly reduces the collision rate compared to the benchmark TD3 algorithm, especially in scenarios with a high number of obstacles. The incorporation of dual-critic networks and the minimum-pooling operation in MATD3 enhances the stability and robustness of the learning process, enabling UAVs to find safe and efficient paths in complex environments.

The task completion rate is another crucial metric that reflects the reliability and effectiveness of the offloading framework. The proposed MATD3-FD3QN approach maintains a high task completion rate of 98.5%, outperforming the state-of-the-art methods. This improvement is attributed to the adaptive resource allocation and the global model aggregation in the federated learning framework, which ensures that the offloading decisions are optimized based on the current network conditions and task requirements.

In summary, the proposed MATD3-FD3QN framework exhibits substantial improvements in latency reduction, energy efficiency, collision avoidance, and task completion rate compared to existing works. The performance gains are rooted in the synergistic integration of federated learning and deep reinforcement learning, as well as the innovative design of the MATD3 and FD3QN algorithms. These results highlight the potential of the proposed approach to revolutionize computation offloading in vehicular networks and pave the way for the development of intelligent transportation systems.

6.2 Applicability and limitations

The proposed MATD3-FD3QN framework demonstrates strong performance and robustness in the context of computation offloading in UAV-assisted vehicular networks. However, it is essential to discuss the applicability and limitations of the algorithm to provide a comprehensive understanding of its potential and areas for future improvement.

One of the key advantages of the proposed framework is its adaptability to different network environments and traffic conditions. The MATD3 algorithm for UAV trajectory planning is designed to handle dynamic obstacles and varying vehicle densities, making it suitable for deployment in urban and highway scenarios. Similarly, the FD3QN algorithm for resource allocation can adapt to changing channel conditions and task demands, ensuring efficient and reliable computation offloading. The federated learning approach further enhances the framework's applicability by enabling collaborative learning among vehicular users while preserving data privacy.

However, there are certain limitations and assumptions that should be considered when applying the proposed framework to other environments or hardware conditions. First, the current study assumes a homogeneous UAV fleet with identical computational capabilities and coverage ranges. In practice, UAVs may have varying specifications and performance characteristics, which would require extensions to the MATD3 algorithm to account for heterogeneous UAV capabilities. Second, the simulation environment considers a simplified two-lane highway scenario with a fixed number of UAVs and RSUs. The performance and scalability of the proposed framework in more complex road networks with multiple lanes, intersections, and a larger number of UAVs and RSUs need to be further investigated.

Another limitation is the assumption of perfect and instantaneous information exchange among the vehicular users, UAVs, and the central server in the federated learning process. In real-world scenarios, communication delays, packet losses, and bandwidth constraints may impact the efficiency and convergence of the federated learning algorithm. Future work should consider the impact of imperfect communication channels and develop robust mechanisms to handle communication uncertainties.

The generalization ability of the proposed framework to other application domains beyond vehicular networks is an important aspect to consider. While the current study focuses on computation offloading in UAV-assisted vehicular networks, the core principles and algorithms can be adapted to other distributed systems with mobile edge computing requirements. For example, the MATD3 algorithm can be applied to autonomous robot navigation and path planning in industrial settings, while the FD3QN algorithm can be extended to resource allocation in wireless sensor networks or smart grid systems. However, the specific characteristics and constraints of each application domain should be carefully considered, and the

algorithms may require domain-specific modifications and tuning.

In terms of hardware dependencies, the proposed framework relies on the availability of sufficient computational resources and storage capacity at the UAVs and RSUs to execute the offloaded tasks and maintain the federated learning models. The performance and scalability of the framework may be limited by the hardware capabilities of the edge devices. Moreover, the energy consumption of the UAVs and the battery life of the vehicular users are critical factors that impact the long-term sustainability of the system. Future research should explore energy-efficient techniques and hardware optimizations to minimize the energy footprint of the proposed framework.

In conclusion, the MATD3-FD3QN framework offers a promising solution for computation offloading in UAV-assisted vehicular networks, with strong performance, robustness, and adaptability to dynamic environments. However, the limitations and assumptions discussed above should be carefully considered when applying the framework to other domains or hardware platforms. Continuous research efforts are needed to address these challenges and enhance the generalization ability of the proposed approach.

7 Conclusion

In this article, we proposed a novel computation offloading framework for UAV-assisted vehicular networks based on federated deep reinforcement learning. The proposed framework leverages the advantages of hybrid multi-server MEC architecture and advanced deep reinforcement learning algorithms to optimize the trajectory planning and resource allocation for efficient computation offloading [39]. The MATD3 algorithm is employed for trajectory planning, while the FUD3QN algorithm is utilized for cross-domain resource allocation.

The simulation results demonstrate the effectiveness of the proposed MATD3-FUD3QN computation offloading algorithm in reducing collision rates and system costs. The MATD3 algorithm achieves significantly lower collision rates compared to the benchmark TD3 algorithm, ensuring safe and efficient UAV trajectory planning [40]. Moreover, the FUD3QN algorithm outperforms other benchmark algorithms, such as FDDQN, D3QN, and Random, in terms of average latency, average energy consumption, and task completion ratio [41].

Furthermore, the FUD3QN algorithm exhibits superior model generalization capability in dynamic vehicular networks. In extended environments with 8 and 12 vehicular users (VUs), the FUD3QN algorithm reduces the system cost by 11.37% and 12.06%, respectively, compared to the FDDQN algorithm [42]. This indicates the robustness and adaptability of the FUD3QN algorithm in handling the challenges of computation offloading in complex and dynamic network conditions.

The proposed MATD3-FUD3QN computation offloading framework represents a significant advancement in the field of intelligent transportation systems and edge computing [43]. By leveraging the power of federated learning and deep reinforcement learning, the framework enables collaborative and efficient offloading decisions while preserving data privacy and reducing communication overhead. The framework has the potential to revolutionize the way computation-intensive and delay-sensitive applications are supported in UAV-assisted vehicular networks.

Future research directions include the investigation of multi-agent reinforcement learning algorithms for coordinated decision-making among multiple UAVs and the incorporation of transfer learning techniques to accelerate the learning process in dynamic network environments [44]. Additionally, the integration of blockchain technology can be explored to enhance the security and trust aspects of the federated learning framework in vehicular networks.

References

[1] Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, 2019. "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3061-3074. https://doi.org/10.1109/tvt.2019.2895593

[2] J. Feng, Z. Liu, C. Wu, and Y. Ji, 2017. "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10660-10675. https://doi.org/10.1109/tvt.2017.2714704

[3] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, 2019. "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4150-4161. https://doi.org/10.1109/jiot.2018.2875520

[4] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, 2017. "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36-44. https://doi.org/10.1109/mvt.2017.2668838

[5] Y. Liu, K. Xiong, Q. Ni, P. Fan, and K. B. Letaief, 2020. "UAV-assisted wireless powered cooperative mobile edge computing: Joint offloading, CPU control, and trajectory optimization," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 2777-2790. https://doi.org/10.1109/jiot.2019.2958975

[6] F. Zhou, Y. Wu, R. Q. Hu, and Y. Qian, 2018. "Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems," *IEEE Journal on Selected Areas in*

Communications, vol. 36, no. 9, pp. 1927-1941. https://doi.org/10.1109/jsac.2018.2864426

[7] Y. Zeng, J. Xu, and R. Zhang, 2019. "Energy minimization for wireless communication with rotary-wing UAV," *IEEE Transactions on Wireless Communications*, vol. 18, no. 4, pp. 2329-2345. https://doi.org/10.1109/twc.2019.2902559

[8] X. Hu, K.-K. Wong, and K. Yang, 2018. "Wireless powered cooperation-assisted mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2375-2388. https://doi.org/10.1109/twc.2018.2794345

[9] J. Zhang, L. Zhou, Q. Tang, E. C.-H. Ngai, X. Hu, H. Zhao, and J. Wei, 2019. "Stochastic computation offloading and trajectory scheduling for UAV-assisted mobile edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3688-3699. https://doi.org/10.1109/jiot.2018.2890133

[10] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong, 2017. "Caching in the sky: Proactive deployment of cache-enabled unmanned aerial vehicles for optimized quality-of-experience," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1046-1061. https://doi.org/10.1109/jsac.2017.2680898

[11] S. Jeong, O. Simeone, and J. Kang, 2018. "Mobile edge computing via a UAV-mounted cloudlet: Optimization of bit allocation and path planning," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 3, pp. 2049-2063. https://doi.org/10.1109/tvt.2017.2706308

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., 2015. "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529. DOIhttps://doi.org/10.1038/nature14236

[13] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, 2016. "Federated learning: Strategies for improving communication efficiency," a*rXiv preprint* arXiv:1610.05492. https://doi.org/10.48550/arXiv.1610.05492

[14] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, 2021. "Low-latency federated learning and blockchain for edge association in digital twin empowered 6G networks*," IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5098-5107. https://doi.org/10.1109/tii.2020.3017668

[15] Z. Xu, Y. Wang, J. Tang, J. Wang, and M. C. Gursoy, 2017, "A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs," in 2017 *IEEE International Conference on Communications* (ICC), pp. 1-6. https://doi.org/10.1109/icc.2017.7997286

[16] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, 2020. "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4298-4311. https://doi.org/10.1109/tvt.2020.2973651

[17] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, 2019. "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory*," IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10700-10714. https://doi.org/10.1109/jiot.2019.2940820

[18] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, 2018. "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 9, pp. 2059-2070. https://doi.org/10.1109/jsac.2018.2864373

[19] A. Srinivas, T. Trinai Krishnan, S. Bidargaddi, V. K. Sharma, and B. Mitra, 2018."Learning to optimize for a class of non-convex objectives with applications to reliability constrained optimization," *arXiv preprint* arXiv:1805.11271. https://doi.org/10.48550/arXiv.2410.11061

[20] Y. He, Z. Zhang, F. R. Yu, N. Zhao, H. Yin, V. C. Leung, and Y. Zhang, 2017. "Deep-reinforcement-learning-based optimization for cache-enabled opportunistic interference alignment wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 11, pp. 10433-10445. https://doi.org/10.1109/icc.2017.7996332

[21] Y. Liu, H. Yu, S. Xie, and Y. Zhang, 2019. "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158-11168. https://doi.org/10.3390/s21196499

[22] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, 2019. "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377-4387. https://doi.org/10.1109/jiot.2018.2876298

[23] L. Huang, S. Bi, and Y. J. Zhang, 2020. "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581-2593. https://doi.org/10.1109/tmc.2019.2928811

[24] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, 2018. "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33-39. https://doi.org/10.1109/mcom.2018.1701095

[25] C. E. Shannon, 1948. "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379-423. DOI: 10.1002/j.1538-7305. 1948.tb01338.x

[26] S. Boyd, S. P. Boyd, and L. Vandenberghe, 2004. *Convex optimization*. Cambridge university press. https://doi.org/10.1017/CBO9780511804441

[27] M. Chen and Y. Hao, 2018. "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in*

Communications, vol. 36, no. 3, pp. 587-597. https://doi.org/10.1109/jsac.2018.2815360

[28] J. Ren, G. Yu, Y. Cai, and Y. He, 2018. "Latency optimization for resource allocation in mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 17, no. 8, pp. 5506-5519. https://doi.org/10.1109/twc.2018.2845360

[29] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, 2020. "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869-904. https://doi.org/10.1109/comst.2020.2970550

[30] H. Van Hasselt, A. Guez, and D. Silver, 2016. "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1. https://doi.org/10.48550/arXiv.1509.06461

[31] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, 2016, "Dueling network architectures for deep reinforcement learning," in *international conference on machine learning*, pp. 1995-2003. https://doi.org/10.48550/arXiv.1511.06581

[32] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, 2015. "Prioritized experience replay," a*rXiv preprint* arXiv:1511.05952. https://doi.org/10.48550/arXiv.1511.05952

[33] T. Nishio and R. Yonetani, 2019, "Client selection for federated learning with heterogeneous resources in mobile edge," in ICC 2019-2019 IEEE *International Conference on Communications* (ICC), pp. 1-7. https://doi.org/10.1109/icc.2019.8761315

[34] S. Fujimoto, H. Van Hasselt, and D. Silver, 2018."Addressing function approximation error in actor-critic methods," *arXiv preprint arXi*v:1802.09477. https://doi.org/10.48550/arXiv.1802.09477

[35] H. Van Hasselt, A. Guez, and D. Silver, 2016. "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1. https://doi.org/10.48550/arXiv.1509.06461

[36] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, 2016. "Federated learning: Strategies for improving communication efficiency," *arXiv preprint* arXiv:1610.05492. https://doi.org/10.48550/arXiv.1610.05492

[37] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, 2017. "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432-7445. https://doi.org/10.1109/tvt.2017.2672701

[38] X. Chen, L. Jiao, W. Li, and X. Fu, 2015. "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795-2808. https://doi.org/10.1109/tnet.2015.2487344

[39] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, 2015. "Continuous control with deep reinforcement learning," *arXiv preprint* arXiv:1509.02971. https://doi.org/10.48550/arXiv.1509.02971

[40] H. Van Hasselt, A. Guez, and D. Silver, 2016."Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1. https://doi.org/10.48550/arXiv.1509.06461

[41] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, 2021."Low-latency federated learning and blockchain for edge association in digital twin empowered 6g networks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 5098-5107. DOI: 10.1109/TII.2020.3017668

[42] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, 2017. "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358. DOI: 10.1109/COMST.2017.2745201

[43] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, 2017, "Communication-efficient learning of deep networks from decentralized data," *in Artificial Intelligence and Statistics*, pp. 1273-1282. https://doi.org/10.48550/arXiv.1602.05629

[44] J. Konecny, H. B. McMahan, D. Ramage, and P. Richtarik, 2016. "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint* arXiv:1610.02527. https://doi.org/10.48550/arXiv.1610.02527