2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA)

Towards Efficient Task Offloading and Resource Allocation: Federated Multi-Agent Learning in Vehicular Edge Computing

1st Liang Zhao*
Shenyang Aerospace University
Shenyang, China
lzhao@sau.edu.cn

2nd Lu Sun
Shenyang Aerospace University
Shenyang, China

3rd Lexi Xu
China United Network Communications Corporation
Beijing, China
davidlexi@hotmail.com

4th Xiongyan Tang
China United Network Communications Corporation
Beijing, China
tangxy@chinaunicom.c

5th Ammar Hawbani
Shenyang Aerospace University
Shenyang, China
anmande@ustc.edu.cn

***Abstract*—Vehicle Edge Computing (VEC) improves traffic efficiency and driving experience. However, task offloading and resource allocation in VEC environments face bandwidth limitations, high latency, and dynamic network changes, which affect system performance and user experience. Therefore, this paper proposes a Hierarchical Attention Federated Multi-Agent Deep Deterministic Policy Gradient (HAFMADDPG). On the one hand, the algorithm utilizes Road Side Units (RSUs) as intermediaries to achieve hierarchical aggregation between vehicles and central servers, effectively reducing bandwidth and latency overhead and adapting to geographically dispersed dynamic network environments. On the other hand, a dynamic attention-based weighting mechanism in the critic network enables performance-aware model aggregation, enhancing system robustness and security. Simulation results show this scheme is better than the centralized Multi-Agent Deep Deterministic Policy Gradient (MADDPG) and Federated Averaging Algorithm (FedAvg).**

Keywords-Federated learning, deep reinforcement learning, task offloading, resource allocation, vehicle edge computing.

I. INTRODUCTION

With advancements in 5G and wireless communication technologies, the Internet of Vehicles (IoV) has become a pivotal enabler of autonomous driving systems [1]. Autonomous vehicles rely on real-time operations, such as environment perception, path planning, and decision-making, which generate computationally intensive tasks.

However, the constrained computational capabilities of vehicles are insufficient to meet these task demands [2]. Vehicle Edge Computing (VEC) has emerged as a promising paradigm, augmenting the computational capabilities of autonomous systems by offloading tasks to edge nodes near vehicles, providing robust support for the growing demands of computation-intensive applications [3], [4].

While VEC offers promising solutions, resource constraints and inefficient task scheduling strategies pose ongoing challenges in balancing computational efficiency and energy consumption. Existing research has proposed various optimization methods around task offloading and resource allocation. Liu et al. [5] propose a system design combining extreme value theory, Lyapunov optimization, and matching theory to optimize task offloading and resource allocation, achieving reliable computation with lower delay and reduced power consumption. Tran and Pompili [6] tackle joint task offloading and resource allocation in MEC-enabled multi-cell networks using convex optimization and heuristic algorithms. Raza et al. [7] introduce a game-theoretical scheme to improve computation efficiency while satisfying energy and time constraints. However, the above studies usually rely on fixed models and have limitations in dynamic and complex VEC environments. In contrast, Deep Reinforcement Learning (DRL) shows excellent potential in task offloading and resource allocation problems in VECs because it adaptively learns and optimizes policies in dynamic and uncertain environments [8]-[10]. However, traditional DRL methods mainly rely on centralized learning frameworks. As the number of connected vehicles and devices in VEC increases, resource management becomes increasingly complex, and the action space and parameter dimensions expand rapidly. This reduces training efficiency and brings security risks related to data privacy [11].

Federated Learning (FL) has gradually become a feasible solution to address the above challenges. By integrating with Multi-Agent Deep Reinforcement Learning (MADRL) algorithms, Federated Multi-Agent Deep Reinforcement Learning (FMADRL) algorithms not only effectively solve the problems of poor model scalability and privacy leakage in traditional centralized methods but also further deal with the complexity of multi-vehicle collaborative decision-making in dynamic environments [12]. However, traditional federated multi-agent deep reinforcement learning still faces challenges, including coordination problems between mobile vehicles and Road Side Units (RSUs) and high cross-device communication overhead.

To address these challenges, this paper proposes a Hierarchical Attention Federated Multi-Agent Deep Deterministic Policy Gradient (HAFMADDPG) to solve the joint optimization problem of task offloading and resource allocation in VEC. The main contributions of this article are as follows:

- To make up for the shortcomings of the traditional centralized DRL algorithm in dealing with large-scale dynamic VEC scenarios, the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) is used to solve task offloading and resource allocation problems. FL is introduced in the training process of MADDPG, which significantly enhances data privacy protection and system scalability.
- This paper proposes a novel hierarchical FL architecture, which uses Road Side Units (RSUs) as intermediate aggregation nodes between vehicles and the central server. This hierarchical aggregation structure improves the global model performance through more efficient data integration, while effectively reducing the communication overhead generated during model aggregation.
- This paper introduces an attention-based dynamic weighting mechanism within the critic network to mitigate the negative effects of low-performance or malicious updates on the global model. This approach can adaptively enhance the weights of high-performance agents and prevent any single agent from prematurely dominating the aggregation process, thereby establishing an efficient and robust transition from individual optimization to global strategy optimization.

The remainder of this paper is organized as follows. Section II presents the system model and problem formulation. Section III describes the main design of the algorithm. Section IV presents the simulation results. Finally, Section V concludes the paper.

II. System Model

This section first introduces the network model, then models the task offloading and resource allocation strategies, then details the system cost incurred by each task computation method, and finally formulates a joint optimization problem of task offloading and resource allocation.

A. *Network Model*

The Vehicular Edge Computing Network (VECN) comprises a Base Station (BS), multiple RSUs, and multiple mobile vehicles. As a central control node, the BS coordinates and manages the entire network to ensure the effective integration and reasonable allocation of model parameters. RSUs are distributed along the road and act as edge computing nodes, providing computing resources to support vehicle task offloading. Each vehicle generates computing tasks according to a certain probability while driving. These tasks can be represented by $(\sigma_n, \theta_n, \varphi_n)$ represents the data size of the *n*-th task, $\theta_n$represents the deadline of the *n*-th task, and φ_n represents the computing demand of the *n*-th task. In this system, vehicles and RSUs communicate using C-V2X wireless communication.

B. *Task Offloading Model*

The task offloading model aims to determine the optimal processing location for the tasks generated by the vehicle to balance resource utilization, reduce latency, and reduce energy consumption. In this paper, the vehicle can compute the task locally, offload the task to an Assisted Vehicle (AV), or offload the task to an RSU. The decision process is based on the vehicle state, task characteristics, surrounding environment information, and status information of other vehicles and RSUs. The communication distance between the vehicle and the target node must be guaranteed during the task offloading process. Within the communication time $t_i \in t$, the distance between the source vehicle and the target node can be calculated by (1) and (2), respectively.

$$d_{V2V}(t_i) = d_{V2V} + (v_i - v_j)t_i \tag{1}$$

$$d_{V2R}(t_i) = d_{V2R} + v_i t_i \tag{2}$$

where v_i is the moving speed of the source vehicle, v_j is the moving speed of the target vehicle, d_{V2V} is the initial distance between the two vehicles, and d_{V2R} is the initial distance between the vehicle and the RSU.

C. *Communication and Computing Model*

Shannon's theorem primarily determined the transmission rates for V2V and V2R communication. Let B_{v2v} denote the maximum communication bandwidth for V2V, $\frac{S_{v2v}}{N_{v2v}}$ represent the signal-to-noise ratio of the V2V channel, and β indicate the proportion of communication resources allocated to the *n*-th task. Based on Shannon's theorem, the transmission rates for V2V and V2R can be calculated using (3) and (4), respectively.

$$r_{v2v} = \beta_n B_{v2v} \, log_2\left(1 + \frac{S_{v2v}}{N_{v2v}}\right) \tag{3}$$

$$r_{v2r} = \beta_n B_{v2r} \, log_2\left(1 + \frac{S_{v2r}}{N_{v2r}}\right) \tag{4}$$

1) Local computing: Local computing can be preferred when the vehicle's computing resources are sufficient, and the task's computational complexity is not high. Local computing only uses the vehicle's computing resources, and its delay and energy consumption are calculated by (5) and (6), respectively.

$$T_i^l = \frac{\varphi_n}{f_n^v} \tag{5}$$

$$E_i^l = T_i^l \cdot p_v^{com} \tag{6}$$

where f_n^v is the computing resource allocated to the *n*-th task by the vehicle, φ_n is the CPU frequency required to execute the *n*-th task, and p_v^{com} is the computing power of the *i*-th vehicle.

2) AV computing: In scenarios where vehicles are densely populated, and some vehicles have idle computing resources, V2V offloading can be an option. The delay and energy consumption caused by V2V offloading are calculated by (7) and (8), respectively.

$$T_i^v = \frac{\varphi_n}{f_n^v} + \frac{\sigma_n}{r_{v2v}} \tag{7}$$

$$E_i^l = \frac{\varphi_n}{f_n^v} \cdot p_v^{com} + \frac{\sigma_n}{r_{v2v}} \cdot p_v^{trans} \tag{8}$$

where σ_n represents the data size of the *n*-th task, r_{v2v} represents the transmission rate between vehicles, and p_v^{trans} represents the transmission power of the vehicle.

3) RSU Computing: When the vehicle computing resources cannot meet the task computing requirements, it can contact RSU for assistance. The delay and energy consumption generated by RSU processing tasks are calculated by (9) and (10), respectively.

$$T_i^r = \frac{\varphi_n}{f_n^r} + \frac{\sigma_n}{r_{v2r}} \tag{9}$$

$$E_i^r = \frac{\varphi_n}{f_n^r} \cdot p_r^{com} + \frac{\sigma_n}{r_{v2r}} \cdot p_r^{trans} \tag{10}$$

where f_n^r is the computing resource allocated by RSU to the *n*-th task, r_{v2r} represents the transmission rate between the vehicle and RSU, and p_r^{com} and p_r^{trans} represent the computing and transmission power of RSU, respectively.

Therefore, the total delay and energy consumption generated by executing the *n*-th task is expressed as (11) and (12), respectively.

$$T_n(t) = a_i^{d,l}(t) \cdot T_i^l + a_i^{d,v}(t) \cdot T_i^v + a_i^{d,e}(t) \cdot T_i^r \tag{11}$$

$$E_n(t) = a_i^{d,l}(t) \cdot E_i^l + a_i^{d,v}(t) \cdot E_i^l + a_i^{d,e}(t) \cdot E_i^r \tag{12}$$

D. Problem Formulation

We formulate a joint optimization problem of task offloading and resource allocation. This multi-objective optimization problem ensures that tasks are completed on time while minimizing processing tasks' delay and energy consumption. The formulation of this optimization problem is as follows:

$$\begin{aligned} P:\ & \min_{\forall t \in T} \sum_{j=1}^{n} [\,(1-\alpha)\cdot T_j(t) + \alpha \cdot E_j(t)] \\ s.t.\ & C_1: a_i^{d,l}(t), a_i^{d,v}(t), a_i^{d,e}(t) \in \{0,1\} \\ & C_2: a_i^{d,l}(t) + a_i^{d,v}(t) + a_i^{d,e}(t) = 1 \\ & C_3: d_{V2V}(t_i) \le R_{V2V}, \forall t_i \in t \\ & C_4: d_{V2R}(t_i) \le R_{V2R}, \forall t_i \in t \end{aligned} \tag{13}$$

where C_1 and C_2 indicate that the tasks are inseparable and only one calculation method can be selected for each task; C_3 and C_4 indicate that the distance between the vehicle and the target node cannot exceed its maximum communication range.

III. Algorithm Design

This section first models the above optimization problem as a Markov Decision Process (MDP) and then proposes a HAFMADDPG algorithm to solve this problem. The algorithm integrates a novel hierarchical FL architecture based on the MADDPG algorithm. It adopts a two-layer aggregation process: initial aggregation using performance-based dynamic weighting at the RSU level and then at the central server global aggregation. This layered structure effectively reduces communication overhead while significantly improving the stability and scalability of the learning process, thereby ensuring efficient policy fusion in dynamic and diverse vehicle environments. The specific steps are shown in Algorithm 1.

A. MDP problem

1) State space: The state space is defined as $s(t) = \{v(t), g(t), r(t), i(t), e(t)\}$. $v(t)$ denotes the state of each vehicle, encompassing attributes such as location, speed, acceleration, and remaining energy. $g(t)$ describes task characteristics, including data size, remaining time constraints, and computational requirements. $r(t)$ represents the state of each RSU, incorporating attributes such as available bandwidth, computational resources, and communication range. $i(t)$ reflects the network conditions, such as the average channel quality and bandwidth availabilit. $e(t)$ represents environmental factors, including vehicle density and road traffic flow. All features are normalized to ensure consistency across dimensions and maintain a uniform scale.

2) Action space: The action space set of time slot t is $a(t) = \{a_1(t), a_2(t), \dots, a_i(t)\}$, where $a_i(t)$ represents the action space of each agent i, which consists of discrete action space $a_i^d(t)$ and continuous action space $a_i^c(t)$. It can be expressed as (14).

$$\begin{gathered} a_i(t) = [a_i^d(t), a_i^c(t)] \\ a_i^d(t) = \{a_i^{d,l}(t), a_i^{d,v}(t), a_i^{d,e}(t)\} \\ a_i^c(t) = [p_{i,j}^c(t), p_{i,j}^k(t)] \end{gathered} \tag{14}$$

where $a_i^{d,l}(t) \in \{0,1\}$ indicates whether the task is calculated locally, $a_i^{d,v}(t) \in \{0,1\}$ indicates whether the task is offloaded to AV, $a_i^{d,e}(t) \in \{0,1\}$ indicates whether to offload the task to RSU. $p_{i,j}^c(t)$ and $p_{i,j}^k(t)$represent the resource allocation ratio between the *i*-th vehicle and the *j*-th target node. $p_{i,j}^c(t) \in [0,1]$ represents the computing resource allocation ratio, and $p_{i,j}^k(t) \in [0,1]$ represents the communication resource allocation ratio.

3) Reward Mechanism: We aim to minimize system costs and improve overall performance while satisfying constraints. The reward function is the weighted sum of latency, energy consumption, task failure penalty, and resource utilization efficiency reward.

$$r(t) = \sum_{j=1}^{n} [\lambda_4 \cdot U(t) - \lambda_1 \cdot T_j(t) - \lambda_2 \cdot E_j(t) - \lambda_3 \cdot F_j(t)] \tag{15}$$

where $T_j(t)$ and $E_j(t)$ are the total delay and total energy consumption caused by executing tasks, $F_j(t)$ is the number of times a task is not completed before its deadline θ_n, and $U(t)$ represents the overall resource utilization efficiency, λ_1, λ_2, λ_3, λ_4 are weight coefficients that balance the importance of each component.

B. Proposed HAFMADDPG Algorithm

In the HAFMADDPG algorithm, we adopt the attention-weighted aggregation mechanism to combine model parameters from multiple agents effectively. This attention-like mechanism is inspired by the self-attention mechanism in the Transformer architecture. It uses a softmax-based weighting scheme to assign dynamic weights to each agent model according to its performance indicators, ensuring that agents with higher performance significantly impact the aggregated global model, thereby improving the overall learning efficiency and robustness of the FL process. Specifically, for each agent i, the weight w_i is calculated as (16).

$$w_i = \frac{exp\left(\frac{perf_i}{\tau}\right)}{\sum_{j=1}^{N} exp\left(\frac{perf_j}{\tau}\right)} \tag{16}$$

where $perf_i$ represents the performance metric of the i-th agent,τ is the temperature parameter used to control the smoothness of the weight distribution, and N is the total number of agents.

1) Local vehicle training: Each vehicle within the network maintains dedicated actor and critic networks characterized by parameters $(\theta_i^{\pi}, \theta_i^{Q})$. Each vehicle also maintains target networks with parameters $(\theta_i^{\pi'}, \theta_i^{Q'})$, which are periodically updated using a soft update mechanism. During each training iteration, a vehicle observes its current state s_i, generates an action $a_i = [a_i^d, a_i^c]$ through its actor-network, and interacts with the environment to receive the subsequent state $s_{i'}$, a reward r_i, and a termination signal d_i. These interactions are encapsulated as tuples $(s_i, a_i, s_i', r_i, d_i)$ and stored in a local replay buffer B_i.

Periodically, the vehicle samples mini-batches from B_i to update its actor and critic networks. The critic network is updated by minimizing the Temporal Difference (TD) error, defined by (17), the target value y_i is computed as (18).

$$\mathcal{L}_i^Q = \mathbb{E}\left[\left(Q\left(\boldsymbol{s}_i, \boldsymbol{a}_i \mid \theta_i^Q\right) - y_i\right)^2\right] \tag{17}$$

$$y_i = r_i + \gamma(1 - d_i)Q'\left(s_i', \pi'(s_i') \mid \theta_i^{Q'}\right) \tag{18}$$

where γ denotes the discount factor. Concurrently, the actor-network updates its policy parameters by maximizing the expected Q-value, as captured by (19).

$$\mathcal{L}_i^{\pi} = -\mathbb{E}\left[Q\left(\boldsymbol{s}_i, \pi(\boldsymbol{s}_i \mid \theta_i^{\pi}) \mid \theta_i^Q\right)\right] \tag{19}$$

The target networks $(\theta_i^{\pi'}, \theta_i^{Q'})$are updated using a soft update strategy to ensure training stability:

$$\theta_i^{\pi'} \leftarrow \tau\theta_i^{\pi} + (1 - \tau)\theta_i^{\pi'} \tag{20}$$

$$\theta_i^{Q'} \leftarrow \tau\theta_i^{Q} + (1 - \tau)\theta_i^{Q'} \tag{21}$$

where τ is a small constant that controls the update rate.

2) Hierarchical aggregation: After local training, the RSU collects the local model parameters After local training, the RSU collects the local model parameters $(\theta_i^{\pi}, \theta_i^{Q})$ and performance metrics $pref_v$ of the vehicles within its communication range and then calculates the normalized attention weight w_v of each vehicle using (16). Each RSU performs a preliminary aggregation of the locally trained actor and critic parameters according to (22) and (23) to obtain$(\theta_{agg,r}^{\pi}, \theta_{agg,r}^{Q})$.

$$\theta_{agg,r}^{\pi} = \sum_{i=1}^{N} w_i\, \theta_i^{\pi} \tag{22}$$

$$\theta_{agg,r}^{Q} = \sum_{i=1}^{N} w_i\, \theta_i^{Q} \tag{23}$$

Subsequently, the central server collects these RSU-level aggregations and calculates the weight w_r of each RSU using (16). The server synthesizes the RSU-level aggregates into a comprehensive global model through a soft update strategy:

$$\theta_{global}^{\pi} \leftarrow \alpha \sum_{r=1}^{R} w_r\, \theta_{agg,r}^{\pi} + (1 - \alpha)\theta_{global}^{\pi} \tag{24}$$

$$\theta_{global}^{Q} \leftarrow \alpha \sum_{r=1}^{R} w_r\, \theta_{agg,r}^{Q} + (1 - \alpha)\theta_{global}^{Q} \tag{25}$$

where α is the global aggregation rate parameter controlling the influence of newly aggregated parameters on the existing global model, this soft update mechanism ensures a balanced integration of new knowledge while maintaining the stability and continuity of the global policy.

Algorithm 1 HAFMADDPG

```
1: Initialize: $\theta_i^{\pi}$, $\theta_i^{Q}$, $\theta_i^{\pi'} \leftarrow \theta_i^{\pi}$, $\theta_i^{Q'} \leftarrow \theta_i^{Q}$, $\mathcal{B}_i$
2: for each episode = 1 to $M$ do
3:   for each $t = 1$ to $T$ do
4:     for each agent $i$ do
5:       Select action $a_i^t = \pi(s_i^t \mid \theta_i^{\pi})$
6:       Execute action, observe $s_i^{t+1}$, $r_i^t$, $d_i^t$
7:       Store $(s_i^t, a_i^t, s_i^{t+1}, r_i^t, d_i^t)$ in $B_i$
8:     end for
9:     for each agent $i$ do
10:      Compute target $y_i$ as in (18)
11:      Update critic and actor by (17) and (19)
12:      Update target networks using (20) and (21)
13:    end for
14:  end for
15:  for each RSU $r = 1$ to $R$ do
16:    Collect $(\theta_i^{\pi}, \theta_i^{Q}, perf_i)$ from connected agents
17:    Compute attention weights $w_i$ using (16)
18:    Aggregate parameters $\theta_{agg,r}^{\pi}$ and $\theta_{agg,r}^{Q}$
       using (22) and (23)
19:  end for
20:  Collect $(\theta_{agg,r}^{\pi}, \theta_{agg,r}^{Q}, perf_r')$ from all RSUs
21:  Compute global attention weights $w_r'$ using (16)
22:  Update global model parameters and distribute updated
     global parameters $\theta_{global}^{\pi}$ and $\theta_{global}^{Q}$
23: end for
```

IV. Simulation Results

The simulation environment used in this paper is built based on the OpenAI Gym framework. It simulates a dynamic road network environment containing multiple mobile vehicles, RSUs, and base stations. The simulation parameters are listed in Table I. We compare the HAFMADDPG algorithm with FedAvgMADDPG and MADDPG, where MADDPG uses centralized agent training. Evaluation of the performance of the proposed HAFMADDPG algorithm involves simulating the total cost under various parameter conditions.

TABLE I: SIMULATION PARAMETER SETTINGS

Parameter	***Value***	***Parameter***	***Value***
Lr_actor	$1e-4$	Time slot	[1,100]
Lr_critic	$3e-4$	Vehicle communication range	150m
Batch size	128	RSU communication range	300m
Replay buffer	50,000	V2V total bandwidth	20MHz
Road range	1000m	V2R total bandwidth	15MHz

The convergence of several algorithms for 10 and 50 vehicles can be seen in Figure 1 and Figure 2. The two algorithms considering FL outperform the MADDPG algorithm. When the maximum number of vehicles in the scene is 50, the HAFMADDPG algorithm proposed shows a faster convergence speed. This indicates that when more agents are on the scene, the federated average algorithm still has limitations in improving the model's generalization ability. The HAFMADDPG algorithm proposed integrates an attention-based dynamic performance weighting mechanism in the critic network, which enables it to integrate heterogeneous information from other agents better, effectively alleviating the impact of low-quality updates and malicious nodes, thereby improving the robustness of the model and its performance in multi-agent scenarios.

As shown in Figure 3, when the number of vehicles in the scene gradually increases, the performance of HAFMADDPG and FedAvgMADDPG algorithms is better than MADDPG. When the number of vehicles increases from 30 to 40, the total cost of the MADDPG algorithm increases from 583.411 to 1640.831, an increase of 1.81 times.

Figure 4 shows the performance differences of several algorithms under different RSU computing power. As the computing power of RSU increases from 6GHz to 14GHz, the total cost of several algorithms shows a downward trend, among which the total cost of the MADDPG algorithm decreases by about 35.6%, with the most significant change, indicating that the algorithm is highly dependent on the increase of computing resources to make up for the lack of multi-agent collaboration efficiency. When the computing power of FedAvgMADDPG increases from 6GHz to 12GHz, the total cost decreases by 24.9% but rebounds to 146.626 at 14GHz, indicating deficiencies in task allocation and resource utilization optimization. The total cost of the HAFMADDPG algorithm decreased by about 22.4%, which is a minor decrease and maintains a stable optimization effect under all computing powers, proving that the performance improvement of the algorithm depends on its efficient resource allocation ability rather than on the simple increase of computing resources.

Figure 5 shows the performance differences of several algorithms under different V2R allocatable total bandwidths. As the V2R allocatable total bandwidth increases from 15 MHz to 55 MHz, the total cost of the HAFMADDPG algorithm decreases by about 47.5%, the most significant decrease and no rebound trend, which shows that the algorithm can effectively reduce communication overhead and dynamically optimize task allocation. The total cost of the FedAvgMADDPG algorithm first decreases from 180.819 to 101.554 and then increases to 140.328, reflecting that it has a bottleneck in resource utilization efficiency under high communication conditions. The total cost of the MADDPG algorithm decreases by about 39.2%. Its cost reduction is mainly reflected in the initial increase in bandwidth. Still, the cost tends to be stable under high bandwidth conditions, indicating that its optimization ability in resource utilization efficiency is limited.

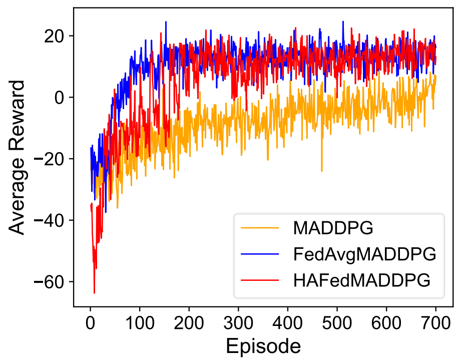

Figure 1: Average reward at 10 vehicles.

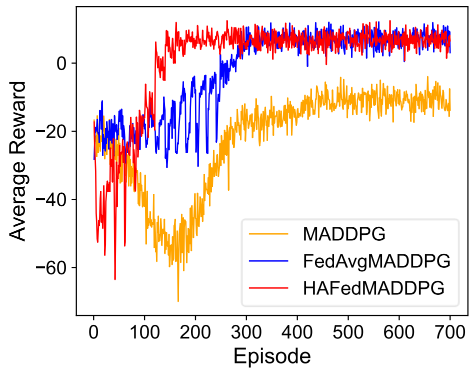

Figure 2: Average reward at 50 vehicles.

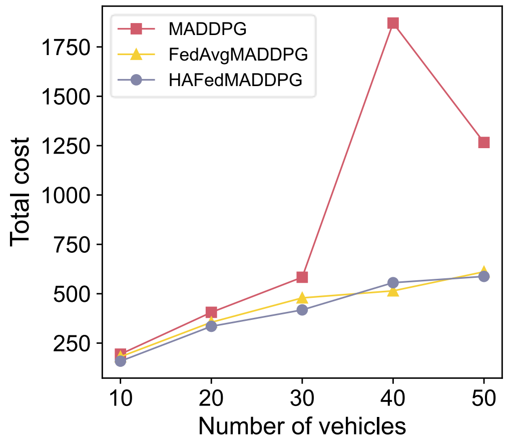

Figure 3: Total cost vs. vehicle number thresholds.

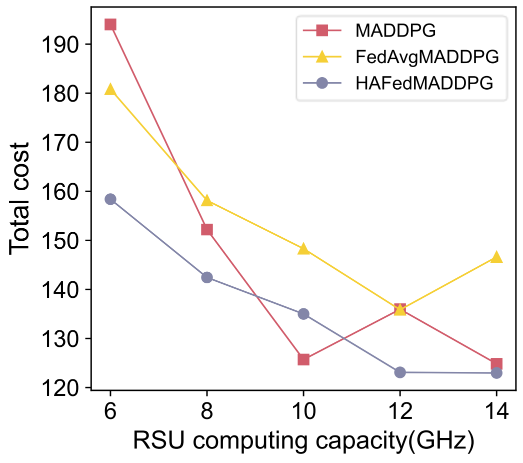

Figure 4: Total cost vs. RSU computing capacity.

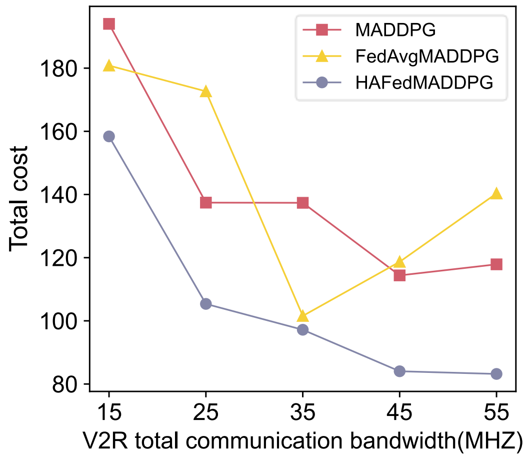

Figure 5: Total cost vs. V2R communication total bandwidth.

V. Conclusions

This paper proposes a HAFMADDPG algorithm to solve the joint optimization problem of task offloading and resource allocation in VEC. By combining the improved FL with MADDPG, the algorithm achieves efficient task management in a dynamic environment with limited resources. In addition, by combining hierarchical attention-weighted parameter fusion with the distributed actor-critic update, the communication overhead of model aggregation is effectively reduced, the learning stability and scalability are improved, and robust strategy fusion is guaranteed in a dynamic and complex vehicle environment. Simulation results show that HAFMADDPG can maintain a relatively reasonable total cost under different network conditions. As the number of agents increases, the algorithm can continuously optimize the task offloading and resource allocation strategies.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Grant 62372310 and W2421018, and in part by the Liaoning Province Applied Basic Research Program under Grant 2023JH2/101300194, and in part by the LiaoNing Revitalization Talents Program under Grant XLYC2203151, and in part by THE ROYAL SOCIETY-International Exchanges 2021 Cost Share (NSFC) under Grant IEC\ NSFC\211034.

References

[1] N. Zhang, S. Liang, K. Wang, Q. Wu and A. Nallanathan, "Computation Efficient Task Offloading and Bandwidth Allocation in VEC Networks," in *IEEE Transactions on Vehicular Technology*, vol. 73, no. 10, pp. 15889-15893, Oct. 2024, doi: 10.1109/TVT.2024.3407356.

[2] R. Shen, M. Gao, W. Li and Y. Li, "Dynamic Task Offloading in Distributed VEC Networks: An Exploration and Exploitation Assisted Contract-Theoretic Approach," in *IEEE Transactions on Vehicular Technology*, vol. 73, no. 4, pp. 5717-5729, April 2024, doi: 10.1109/TVT.2023.3332956.

[3] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza and W. U. Khan, "Task Offloading and Resource Allocation for IoV Using 5G NR-V2X Communication," in *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 10397-10410, 1 July1, 2022, doi: 10.1109/JIOT.2021.3121796.

[4] X. He, H. Lu, M. Du, Y. Mao and K. Wang, "QoE-Based Task Offloading With Deep Reinforcement Learning in Edge-Enabled Internet of Vehicles," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2252-2261, April 2021, doi: 10.1109/TITS.2020.3016002.

[5] C. -F. Liu, M. Bennis, M. Debbah and H. V. Poor, "Dynamic Task Offloading and Resource Allocation for Ultra-Reliable Low-Latency Edge Computing," in *IEEE Transactions on Communications*, vol. 67, no. 6, pp. 4132-4150, June 2019, doi: 10.1109/TCOMM.2019.2898573.

[6] T. X. Tran and D. Pompili, "Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856-868, Jan. 2019, doi: 10.1109/TVT.2018.2881191.

[7] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza and W. U. Khan, "Task Offloading and Resource Allocation for IoV Using 5G NR-V2X Communication," in *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 10397-10410, 1 July1, 2022, doi: 10.1109/JIOT.2021.3121796.

[8] Y. Liu, H. Yu, S. Xie and Y. Zhang, "Deep Reinforcement Learning for Offloading and Resource Allocation in Vehicle Edge Computing and Networks," in *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158-11168, Nov. 2019, doi: 10.1109/TVT.2019.2935450.

[9] F. Zhang, G. Han, L. Liu, M. Martínez-García and Y. Peng, "Deep Reinforcement Learning Based Cooperative Partial Task Offloading and Resource Allocation for IIoT Applications," in *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 5, pp. 2991-3006, 1 Sept.-Oct. 2023, doi: 10.1109/TNSE.2022.3167949.

[10] L. Zhao, X. Dong, A. Hawbani, Y. Bi, Q. He and Z. Liu, "Optimizing Task Offloading in VEC: A PDQKM Scheme Combining Deep Reinforcement Learning and Kuhn-Munkres Matching," in *IEEE Transactions on Intelligent Transportation Systems*, doi: 10.1109/TITS.2024.3491168.

[11] B. Hazarika, K. Singh, S. K. Singh, C. Pan and T. Q. Duong, "Asynchronous Federated Learning-Based Resource Management in URLLC-IoV Networks," *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, Kuala Lumpur, Malaysia, 2023, pp. 6874-6879, doi: 10.1109/GLOBECOM54140.2023.10437135.

[12] Z. Hu, S. Liu, D. Zhou, F. Xu, J. Ma and X. Ning, "Deep Reinforcement Learning for Task Offloading and Resource Allocation in UAV Cluster-Assisted Mobile Edge Computing," in *IEEE Journal on Miniaturization for Air and Space Systems*, doi: 10.1109/JMASS.2024.3518576.