Edge Computing Task Offloading of Internet of Vehicles Based on Improved MADDPG Algorithm

Ziyang Jin[1], Yijun Wang [1*], and Jingying Lv [1]
[1] School of Electronic and Information Engineering, Changchun University of Science and Technology, Changchun,130022, China
[e-mail: wangyijun@cust.edu.cn]
[*]Corresponding author: Yijun Wang

Received May 30, 2023; revised October 20, 2023; accepted February 7, 2024; published February 29, 2024

Abstract

Edge computing is frequently employed in the Internet of Vehicles, although the computation and communication capabilities of roadside units with edge servers are limited. As a result, to perform distributed machine learning on resource-limited MEC systems, resources have to be allocated sensibly. This paper presents an Improved MADDPG algorithm to overcome the current IoV concerns of high delay and limited offloading utility. Firstly, we employ the MADDPG algorithm for task offloading. Secondly, the edge server aggregates the updated model and modifies the aggregation model parameters to achieve optimal policy learning. Finally, the new approach is contrasted with current reinforcement learning techniques. The simulation results show that compared with MADDPG and MAA2C algorithms, our algorithm improves offloading utility by 2% and 9%, and reduces delay by 29.6%.

Keywords: MADDPG, Edge Computing, IoV, Task offloading.

This research was supported by the Key R&D Projects of Jilin Provincial Science and Technology Development Plan, China (No. 20230203091SF). We express our gratitude for reviewing our manuscript.

http://doi.org/10.3837/tiis.2024.02.004 **ISSN : 1976-7277**

1. Introduction

With more vehicles on city streets and the advancement of wireless communication technologies, new Internet of Vehicles services are emerging, including computationally intensive and delay-sensitive services such as augmented reality, image processing, driverless path planning, and navigation. These applications and services require significant computational resources and strict delay constraints. However, vehicles need more computational resources to support many applications [1]. The traditional IoV(Internet of Vehicles) can no longer meet the above demands, and edge computing has emerged to provide a low-delay, highly reliable service experience for IoV services. Due to the limited resources of edge servers, the deployment of edge servers is complicated by increasing vehicle terminals, richer IoV services, and applications [2]. To improve local and remote execution of distributed tasks in vehicles, multiple edge nodes and distant clouds can collaborate thanks to roadside devices' storage and processing power with edge servers. Additionally, the cutting-edge distributed deep learning paradigm of federated learning creates fresh opportunities for edge computing [3].

Federation learning, a common machine learning method, transmits only a fraction of the training results to the cloud without storing the data on the device. When partial training results from millions of devices are collected in the cloud, they can be assembled into a new supermodel that can be sent back to the device in the next step [4]. Applying federation learning to the Internet of Vehicles can protect user privacy while speeding up vehicle processing tasks. However, a single global model may fail when the data distribution changes between clients. User heterogeneity is a major obstacle to federation learning, making it difficult for the global model in IOV to converge [5]. The key to solving the IoV task is using federated learning in edge computing for the Internet of Vehicles.

Considering the limited resources of edge servers, which leads to high delay and low offloading utility in edge computing for IoV, an Improved MADDPG (Multi-Agent Deep Deterministic Policy Gradient) [6] algorithm is proposed. This paper uses edge computing's distributed characteristics to adapt to the vehicle network's dynamic changes and resource limits to provide customers with low-delay services. The main contributions of this paper are as follows:

(1) The MADDPG algorithm is used in the Internet of Vehicles to obtain an effective task offloading policy through training to minimize the loss function with a limited resource budget.
(2) Propose an Improved MADDPG algorithm that adds federated learning to the MADDPG to enable adjustment of model aggregation frequencies better to learn parametric models on vehicles with different computing power to reduce delay and improve offloading utility.
(3) To confirm the efficacy of the suggested method, we ran extensive numerical simulations and compared them to other strategies.

2. Related Work

A critical issue in the Internet of Vehicles is how to employ limited communication and computer resources to deliver safe and dependable services to vehicle users. IoV and MEC (Mobile Edge Computing) integration has created new business models and information interaction for intelligent vehicles, which can solve this challenge.

To deliver better services to nearby users and achieve seamless real-time responses to mobile user service requests, Wang S proposed an offline microservice coordination algorithm based on dynamic programming [7]. Shuai suggested an adaptive network routing technique to find a balance between modeling accuracy and optimization efficiency in MEC systems with time-varying connection delays [8]. In heterogeneous vehicle networks with many random jobs, time-varying radio channels, and dynamic bandwidth, Ke proposed an adaptive depth-reinforcement-based computational offloading approach [9]. Qin Z suggested a hierarchical end-edge system that minimizes network overhead by deep collaboration between data communication, computation offloading, and content caching to efficiently propagate huge volumes of data or computation demands from nearby vehicles [10]. He X presented enhanced deep reinforcement learning PS-DDPG to find the best offloading pattern learning approach for vehicle tasks to improve stability and convergence [11]. IoV system with MEC layers was presented by Wang G in [12]. After rearranging jobs according to delay tolerance, a reinforcement learning technique allocates resources intelligently and automatically. To arrange work across MEC servers and reduce downlink energy usage, Ning Z presented a heuristic method [13].

The research as mentioned above, processes Internet of Vehicles tasks in edge computing using reinforcement learning and conventional algorithms, but the task processing speed is slightly slower. To increase service speed and flexibility, this paper considers using distributed federated learning in edge computing.

In recent years, the machine learning model for the Internet of Vehicles application has been primarily taught using data acquired by onboard devices, roadside units, and base stations. However, the vehicle's vast data resources make it tough for the central server to cope. Federated learning can alleviate resource constraints and lighten the load on the central server.

After evaluating the convergence of the gradient descent loss function, S. Wang developed a control strategy [14] to balance the tradeoff between local model updating and global model aggregation under resource restrictions. To safeguard the privacy of the updated local model and eliminate security risks brought on by centralized administration, Y. Lu proposed in [15] to add local differential privacy into cooperative learning. Samarakoon S proposed a federated learning-based distributed solution in [16] to solve the joint power and resource allocation problem of ultra-reliable low-delay communication in a vehicle network. Yang H developed an analysis model in [17] to evaluate wireless federated learning and examined federated learning model convergence rates with different scheduling approaches. Amiri M studied federated learning at the edge of wireless networks in [18], where low-power wireless devices collaborate with remote parameter servers to optimize end-to-end performance. Bao W presented an edge computing-based Internet of Vehicles client selection and networking approach in [19] to find the best behavior. D. Chen studied and described delay minimization in [20] for collaborative multitask learning in computer networks with numerous access points. Matching games determine the best task assignment technique. T Dinh described in [21] a personalized federated learning approach that uses the Moreau envelope as a regularization loss function to attain great performance on each client's task.

Existing approaches may need to be revised since the Internet of Vehicles scenario necessitates the usage of varied learning models to complete tasks; therefore, this study proposes an Improved MADDPG algorithm.

The following portions of this essay are as follows: Section III presents the system's overall architecture and problem modeling. The proposed algorithm's formulation is shown in Section IV. The experimental findings are analyzed in Section V. The entire paper is concluded in Section VI.

3. System Architecture and Problem Description

3.1 System Architecture

In the MEC system based on the Internet of Vehicles, the edge architecture is a widely used model. This model places the collaboration between vehicles and base stations at the core, achieving task collaboration through task offloading, computing resource sharing, and other methods. Specifically, we represent vehicles as set,$N = \{1,2,\ldots,N\}$, and each vehicle has an IoV data task Y_n to complete. For ease of description, we use tuples $Y_n = (C_n, D_n, TD_n)$ to represent the total computing resources, input data size, and maximum allowable delay time required for the vehicle n to complete the task Y_n. Among them, the variable C_n denotes the computing resources necessary to accomplish the given task Y_n, the quantity of central processing unit (CPU) cycles that must be performed, D_n indicates the size of input data required, TD_n indicates the maximum allowable delay time for completing a task Y_n.

The present architectural design incorporates a Mobile Edge Computing server within the base station to furnish computational services to vehicles. Using task offloading, the vehicle can offload task Y_n to the base station for execution, thereby reducing its computational burden. Simultaneously, it is imperative to guarantee the quality and promptness of tasks; this architecture also supports the sharing and collaboration of computing resources; that is, when computing resources are idle or urgent tasks need to be completed, it can request sharing and collaboration of computing resources from other vehicles.

In summary, the MEC architecture based on the Internet of Vehicles effectively achieves cooperation between vehicles and base stations through task offloading, computing resource sharing, and other methods, improving the efficiency and quality of completing IoT data tasks.

The fundamental network elements of the Mobile Edge Computing framework, which is founded on the IoV framework, as shown in **Fig. 1**. Edge vehicles: Each vehicle manages a set of IoT sensors within its coverage area. QoE (Quality of Experience) requires vehicles to process data locally or wirelessly offload to near MEC servers. MEC server: The MEC server at the base station handles vehicle offloading, a computationally intensive data operation. Vehicles can offload, so MEC servers may calculate QoE-based QoS (Quality of Service) by analyzing job configuration files, including task size, channel circumstances, and available resources.

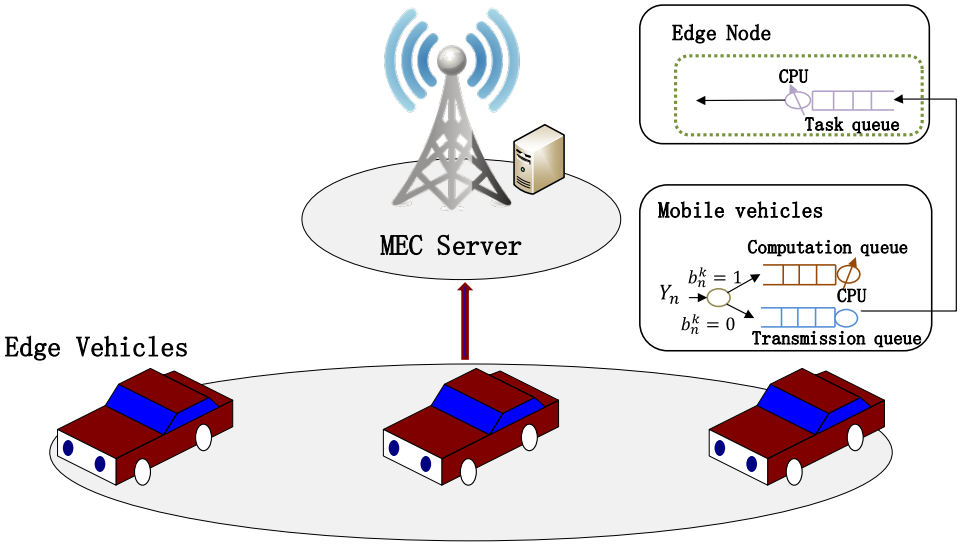

Fig. 1. System Architecture

Tasks in the model can be processed locally or offloaded to edge nodes for processing. It is assumed that at the beginning of each time slot, a new task arrives at the mobile device with a certain probability. When a new task arrives at the mobile device, it first needs to decide whether to process the task locally or offload it to the edge node. If the mobile device decides to process the task locally, its scheduler will put the task into the compute queue for processing. Otherwise, the mobile device needs to decide which edge node to offload the task to. For the computation (or transmission) queue, if the processing (or sending) of a task is completed within one-time slot, the next task in the queue will be processed (or transmitted) at the beginning of the next time slot.

Table 1. Partial Variable Interpretation

K	The set of available sub-bands
b_n^k	offloading strategy for binary variable definitions
b_n	global offloading decisions
Y_n	task
n	Index of Vehicle
D_n	the size of input data required for the task
TD_n	maximum allowable delay time for completing a task
B	includes all of the task offloading variables b_n^k .
N_n	a group of vehicles whose duties are delegated to the MEC server
M_j	Index of MEC servers
k	Index of sub-band
σ^2	background noise variance
T	time consumption
E	energy consumption
P	offloading decision policy
F	resource allocation policy
J_n^{off}	offloading utility
f_n^l	computational resource allocated to the vehicle n
o_n	vehicle's observation
$\phi_n(t)$	CPU resources
DS_j	the index of dataset
w	model parameter of MADDPG
$F_j(w)$	the loss function of the dataset DS_j at each node j
CB_m	the overall budget proposal for m-type resources
w^f	final model parameter
G	total number of global aggregations within ST iterations
l_m	resources consumed by each local update step at all nodes,
g_m	and each global aggregation step consumes

3.2 Problem Description

3.2.1 Communication Model

The set of available sub-bands at the base station is represented by $K = \{1,2,..,K\}$. Define a task offloading scheme using the binary variable $b_n^k (n \in N, k \in K)$, which also covers uplink subband scheduling; when b_n^k is equal to 1, it signifies that offloading
Y_n from vehicle n to the MEC server is carried out through sub-band k. The vehicle can execute computer tasks or offload to the MEC server:

$$\sum_{k \in K} b_n^k \leq 1, n \in N \tag{1}$$

The task offloading strategy B is defined in this article as $B = \{b_n^k | b_n^k = 1, n \in N, k \in K\}$ it includes all of the task offloading variablesb_n^k. Furthermore, $N_n = \{n \in N | \sum_{k \in K} b_n^k = 1\}$ it is represented as a fleet of vehicles assigned tasks managed by the MEC server.

It is posited that each vehicle and base station is equipped with a solitary antenna for uplink communication. The uplink channel gain between the vehicle n and the base station over the subband k is represented as h_n^k. The vehicle transmits power policy is $P = \{p_n^k | 0 < p_n^k \leq P_n^k, n \in N_n\}$, where p_n^k is vehicle's transmit power while offloading task Y_n to the base station over channel k with the maximum budget P_n^k. It is assumed that the MEC system operates within a total operating band B_{MEC}, partitioned into K sub-bands of equal size $W = B_{MEC}/K$. Hence, the data transmission rate of the vehicle n can be computed as follows:

$$R_n = W \log_2 \left(1 + \frac{p_n^k h_n^k}{\sigma^2 + \sum_{x \in N_{n,x \neq n}} (b_x^k p_x^k h_x^k)} \right) \tag{2}$$

The variable σ^2 denotes the background noise variance, while the second term in the denominator represents the interference that arises among vehicles sharing the same channel. The duration for the vehicle n to transmit its task input D_n through the uplink can be represented as:

$$T_n^{up} = \frac{D_n}{R_n}, \forall n \in N \tag{3}$$

Thus, the vehicle's offloading consumption of energy is:

$$E_n^{up} = p_n T_n^{up} = p_n \frac{D_n}{R_n}, \forall n \in N \tag{4}$$

where $p_n = \sum_{k \in K} p_n^k, \forall n \in N$.

3.2.2 Computing Model

Local: The variable f_n^l represents the computational resource assigned to the vehicle denoted as n responsible for executing the data task. F must be at most, the overall computational capacity F_n of the vehicle. Thus, it is possible to establish the computational resource allocation policy for the vehicle $F = \{f_n^l | 0 < f_n^l \leq F_n, n \in N\}$. The time it takes a task input D_n to execute on the vehicle n can be denoted as:

$$T_n^l = \frac{C_n}{f_n^l} \tag{5}$$

The local task performance of a vehicle is associated with its energy consumption.

$$E_n^l = \kappa \left(f_n^l\right)^2 C_n \tag{6}$$

where κ is the energy factor depending on the chip architecture [22] and is the CPU workload of the vehicle n.

Edge: In the case of offloading, the base station's MEC server can give computing services to many vehicles at the same time. The MEC server has more computing power f^e and a more stable energy source than the vehicle. On the MEC server, the task execution time can be determined as follows:

$$T_n^{eb} = \frac{C_n}{f^e}, \forall n \in N \tag{7}$$

Due to the typically small size of the data results, we did not model the portion of the downloaded data compared to the offloaded data. The equation that represents the cost incurred by a vehicle due to the delay in offloading its task is as follows:

$$T_n^{off} = T_n^{up} + T_n^{eb} = \left(\frac{D_n}{R_n} + \frac{C_n}{f^e}\right), \forall n \in N \tag{8}$$

For the energy cost, the energy cost consumed by vehicle n in offloading its task Y_n is only related to the data transfer and is given by the following equation:

$$E_n^{off} = E_n^{up} = p_n \frac{D_n}{R_n}, \forall n \in N \tag{9}$$

3.2.3 Problem Formulation

The present study formulates the MADDPG scheme's system utility by incorporating the offloading utility.

We use an offloading utility function that primarily takes task calculation time and energy usage into account. In the MEC system, these two factors are the main metrics used to characterize the offloading QoE, which can be expressed as:

$$T_n = T_n^{off} b_n + T_n^l (1 - b_n), \forall n \in N \tag{10}$$

$$E_n = E_n^{off} b_n + E_n^l (1 - b_n), \forall n \in N \tag{11}$$

where $b_n = \sum_{k \in K} b_n^k, \forall n \in N$, which represents the global offloading decision, all the vehicles involved in the decision work together to select the optimal offloading strategy by integrating the results of the offloading decisions of all sub-bands. This means that the intelligence do not only focus on the offloading of their own sub-bands but also considers the performance and resource utilization of the overall system to maximize the overall performance or balance the system efficiency.

We define a QoE-perceived utility function to calculate the usefulness of offloading by weighting the time and energy required to execute the activity locally against offloading it.

$$J_n^{off} = \lambda_n^T \left(\frac{T_n^l - T_n}{T_n^l}\right) + \lambda_n^E \left(\frac{E_n^l - E_n}{E_n^l}\right) \tag{12}$$

where $\lambda_n^T, \lambda_n^E \in [0,1]$ is set by the vehicle n to show the preference for time and energy costs when computing the task. The vehicle might increase the weighting factor for time consumption if the task is urgent. If a vehicle uses less energy, the energy consumption component should also be given priority.

We provide offloading method B, computational resource allocation method F, and transmission power method P. The offloading utility is the weighted sum J_n^{off} of all vehicle offloading utilities:

$$J^{off} = \sum_{n \in N} J_n^{off}(B, P, F) \tag{13}$$

Our goal is to maximize the offloading utility using the MADDPG algorithm:

$$\max_{X,P,F} J^{off} \tag{14}$$

$$b_n^k \in \{0,1\}, \tag{15}$$

$$\sum_{k\in K} b_n^k \le 1, \tag{16}$$

$$0 < p_n^k \le P_n^k, \tag{17}$$

$$0 < f_n^l \le F_n \tag{18}$$

Each task can be carried out locally or offloaded to the MEC server through a sub-channel, according to constraints (15) and (16). The transmission's power constraint for every vehicle is shown in (17). Each vehicle n is required to allocate computing resources to carry out computational activities, but they are not permitted to go above the entire computational budget, according to constraint (18).

The increase in high-dimensional system state space brought on by the rise of edge vehicles may make it more difficult to apply the suggested offloading model in real-world MEC scenarios. This article addresses the task offloading issue by using an Improved MADDPG algorithm.

4. Modeling based on Improved MADDPG Algorithm

4.1 Algorithm Overview

In the Improved MADDPG algorithm, we combine the MADDPG algorithm with the adaptive federated learning algorithm to reduce the time required to complete the task offload, as shown in **Fig. 2**.

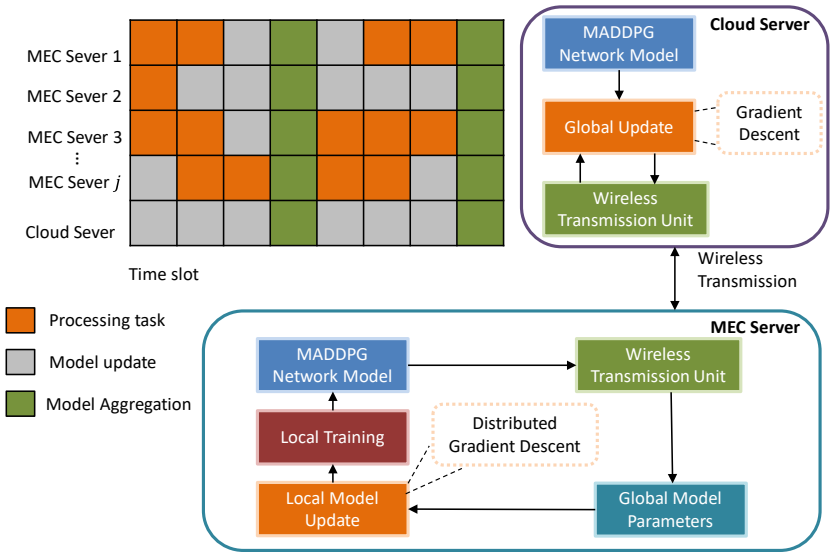

Fig. 2. Algorithm process

In this architecture, j service nodes are present. This study solely examines RSU as the roadside service node, which is represented as $M_1, M_2, \ldots, M_j$. Each RSU incorporates an MEC server. The roadside unit with an edge server receives consumer data and requests and updates local model parameters using a gradient descent algorithm. The edge server prioritizes the

tasks that are offloaded from the vehicle at the beginning of each time slot. During idle time slots, the edge server trains the MADDPG model parameters using data that was acquired locally. The edge server sends the parameters to the cloud server through wireless communication after completing numerous training sessions. The cloud server incorporates the model parameters from all of the edge servers and executes a global update after receiving the parameters from the edge servers and determining the right aggregation frequency based on the federated learning method. The gradient descent technique is used for this global update procedure. The cloud server transmits the new parameters back to the edge servers after the global update is finished. This enables the edge servers to process tasks with the new parameters.

4.2 Markov Modelling

When we study MEC-based systems, traditional single-intelligent or independent multi-intelligent plans do not yield vehicle cooperation strategies because of the non-smooth and locally observable environment. The O_n observed by the agent n can change when the strategies of other intelligence change; this differs from the cumulative reward $r_n(t)$ of its actual state action. In addition, the presence of non-collaboration in the independent multi-intelligence learning scheme leads to the fact that the agent n has only partial information and is not informed about the updates of the other intelligence. This impacts the agents' rewards and makes it difficult to ensure smooth convergence of the learning algorithm. For this reason, this paper adopts a MADDPG solution that combines centralized learning with decentralized execution. MADDPG is based on the idea of centralized learning and decentralized execution. As shown in **Fig. 3**.

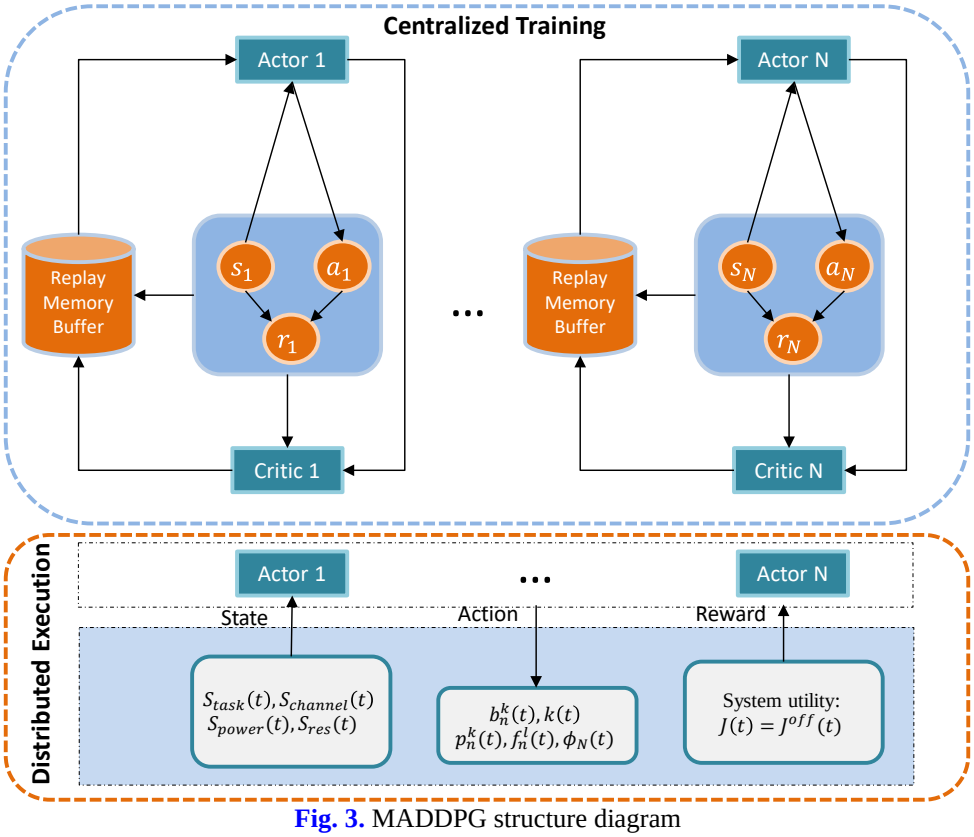

Fig. 3. MADDPG structure diagram

The objective function is converted from the maximization of system utility to the maximization of revenue. The task offloading problem is formulated using MDP (Markov Decision Process), represented by the tuple $\langle N, S, A, O \rangle$. Every vehicle is an intelligent agent that determines how to get the most out of the system by observing the environment while cooperating with other intelligent agents as the number of vehicles.

$N = \{1,2,\ldots,N\}$ is used. $S = \{s_1, s_2, \ldots, s_N\}$ represents the set of states, $A = \{a_1, a_2, \ldots, a_N\}$ represents the set of intelligence actions, and $O = \{o_1, o_2, \ldots, o_N\}$ stands for the set of vehicles' observations. Define each tuple element in each time interval, as shown below.

4.2.1 State

The environmental state is comprised of four distinct components, namely the task state $S_{task}(t)$, channel state $S_{channel}(t)$, power state $S_{power}(t)$, and resource state $S_{res}(t)$. The definition of the system state is as follows:

$$S(t) = \{S_{task}(t), S_{channel}(t), S_{power}(t), S_{res}(t)\} \tag{19}$$

where $S_{task}(t) = [D_n(t), C_n(t)]$ represents the relationship between the computational task size denoted by $D_n(t)$ a given vehicle and the number of input CPU cycles required to complete the task data $C_n(t)$.

When a vehicle utilizes subchannel k in time slot t:

$$S_{channel}(t) = c_n^k(t) = \begin{bmatrix} c_{1,1} & \cdots & c_{1,k} \\ \cdots & \cdots & \cdots \\ c_{N,1} & \cdots & c_{N,k} \end{bmatrix} \tag{20}$$

where $c_n^k(t)$ is the answer, if so, then $c_n^k(t) = 1$, otherwise $c_n^k(t) = 0$.

The power states are:

$$S_{channel}(t) = c_n^k(t) = \begin{bmatrix} p_{1,1} & \cdots & p_{1,k} \\ \cdots & \cdots & \cdots \\ p_{N,1} & \cdots & p_{N,k} \end{bmatrix} \tag{21}$$

where $p_n^k(t)$ is the vehicle's transmit power level in the *kth* sub-channel, satisfying $0 < p_n^k(t) \leq P_n^k$.

Moreover, the resource state is represented by the symbol $S_{res}(t) = \{v_1(t), v_2(t), \ldots, v_N(t)\}$, in which $v_N(t)$ comprises the state of the CPU resources $\phi_n(t)$ for the vehicle and the computing resources $f_n^l(t)$ that are now accessible.

4.2.2 Action

Through the observation of the system status, each vehicle must process data at every stage, encompassing crucial tasks such as offloading decisions and channel selection. The representation of the action space can be expressed as follows:

$$A(t) = \{b_n^k(t), k(t), p_n^k(t), f_n^l(t), \phi_N(t)\} \tag{22}$$

Offloading decision $b_n^k(t)$: $b_n^k(t) \in \{0,1\}$. Based on the current task state, the vehicle decides to locally execute the task $b_n^k(t) = 0$ or offload it to the MEC server through channel k, $b_n^k(t) = 1$.

Channel selection $k(t)$: $k(t) = [1,2,\ldots,K]$. Every vehicle selects one of the available channels based on the current channel state to offload the task to the MEC server.

Transmit power selection $p_n^k(t)$: $p_n^k(t) \in (0, P_n^k)$. In consideration of the present task state and channel state, each vehicle determines the appropriate transmission power level for

transmitting the data task to the MEC server.

Calculate resource allocation using $f_n^l(t)$: $f_n^l(t) = [f_1^l(t), f_2^l(t), \ldots, f_N^l(t)]$. Every vehicle allocates a portion of its computing resources to tasks based on the resource and task states.

CPU resource allocation $\phi_n(t)$: $\phi_n(t) = [\phi_1(t), \phi_2(t), \ldots, \phi_N(t)]$. Every vehicle allocates CPU resources based on the state of the available resources.

4.2.3 Reward

The system reward at a time slot t is the sum of the rewards of all vehicles. After performing each possible action $a_n(t)$, each vehicle n will receive a reward $r(s_n(t), a_n(t))$ in a particular state $s_n(t)$. In this paper, the system reward function should be positively related to the objective function in the optimization problem (13)-(18), aiming to maximize the system utility of all vehicles. Then, we can specify the system reward function of the offloading network for each time slot t as:

$$r(s(t), a(t)) = \sum_{n \in N} r(s_n(t), a_n(t)) = J(t) \tag{23}$$

the system utility of the MEC is denoted by $J(t) = J^{off}(t)$.

4.3 Improved MADDPG Algorithm

In the MADDPG algorithm, $\pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ is defined as the collection of all agent policies and $\theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$ the set of parameters of the corresponding policy. To establish the best possible policy $\pi_{\theta_n}^* = \text{argmax}\pi_{\theta_n}\mathcal{J}(\theta_n)$, each agent modifies its $\theta_n$parameters, where $\mathcal{J}(\theta_n)$ is the objective function of the intelligence n as defined in (22). So we can get the gradient of each agent:

$$\nabla_{\theta_n}\mathcal{J}(\theta_n) = \mathbb{E}_{a\sim\pi_n}[\nabla_{\theta_n}\log\pi_n(a_n|o_n)Q_n^{\pi}(O, a_1, \ldots, a_N)] \tag{24}$$

where o_n denotes the observation of the agent $O = \{o_1, o_2, \ldots, o_N\}$, Q_n^{π} is the critic network.

For a deterministic strategy, the gradient can be expressed as:

$$\nabla_{\theta_n}\mathcal{J}(\pi_n) = \mathbb{E}_{a\sim D}[\nabla_{\theta_n}\pi_n(a_n|o_n)\nabla_{a_n}Q_n^{\pi}(O, a_1, \ldots, a_N)|_{a_n=\pi_n(o_n)}] \tag{25}$$

where D is the replay buffer, the actor network is updated using this gradient. In contrast, the critic network's parameters are updated using their mean square deviation from the target network as a loss.

$$L(\theta_n) = \mathbb{E}_{o,a,r,o'}[Q_n^{\pi}(O, a_1, \ldots, a_N) - y)^2], y = r_i + \gamma Q_n^{\pi'}(O', a_1', \ldots, a_N')] \tag{26}$$

where $Q_n^{\pi'}$ is the target critic network.

In our system, we suppose M edge nodes, each containing datasets DS_1, DS_2, ..., DS_j, DS_M from different vehicles, and the loss function of the dataset DS_j at each node j is:

$$F_j(w) = \frac{1}{DS_j}\sum_{i \in DS_j} f_i(w) \tag{27}$$

where w represents the model parameter of MADDPG and symbol i represents the training sample. The objective of the Improved MADDPG approach is to optimize the integrated global network model:

$$\mathcal{B} = arg\min_{\beta} F_j(w) \tag{28}$$

where β is the model of edge node j, and $\mathcal{B}$ denotes the ensemble of all local network models β . In this paper, $F_j(w) = L(\theta_n) = \mathbb{E}_{o,a,r,o'}[Q_n^{\pi}(O, a_1, \ldots, a_N) - y^2]$, which is the loss

function of the MADDPG model.

Different nodes have different data sets; the definition of the global loss function for all distributed datasets is:

$$F(w) = \frac{\sum_{j=1}^{M} DS_j F_j(w)}{DS} \tag{29}$$

where DS denotes the concatenation of all data sets, and $F(w)$ cannot be calculated directly if the information cannot be shared among the nodes. In this study, the gradient descent algorithm determines w to get $F(w)$ to reach the minimum value, denoted as w^*. The local model is updated locally, and the time of uploading to the cloud server is selected according to the aggregation frequency. The global model will be updated after aggregation, and then the updated network will be transmitted to each edge server.

Stochastic gradient descent is used to obtain the minimum value in (18), where each node has a local parameter model $w_j(t)$, $t = 0,1,2,$ ...indicates the iteration index. When t is 0, all node's j's local parameters are set to the same value. When t>0, gradient descent is used to update $w_j(t)$ based on the previous $t-1$ parameter values, and each node's local loss function is called a local update. After one or more local updates, the cloud server performs global aggregation; after aggregation, the parameters at each node are changed to reflect an average of the values at all other nodes. Local updates and global aggregation are defined for each iteration. After global aggregation, the parameter $w_j(t)$ changes. Using $\widetilde{w_j}$ to denote the parameter at the cloud server, the update rule for each node after global aggregation is as follows:

$$w_j(t) = \widetilde{w_j}(t-1) - \eta \nabla F_j\left(\widetilde{w_j}(t-1)\right) \tag{30}$$

where η denotes the step size, and for all iterations t:

$$w(t) = \frac{\sum_{j=1}^{M} K_j w_j(t)}{K} \tag{31}$$

The τ step local update is executed at each node before the system performs global aggregation; ST is the sum of the number of times each node's local iteration was performed, assuming that ST is a multiple of τ.

A distributed gradient descent approach determines the ideal values for ST and τ to minimize the resource-limited global loss function. Denote the total number of global aggregates in ST iterations by $G = \frac{ST}{\tau}$. Then it can be defined as:

$$w^f = \underset{w \in \{w(G\tau)\}}{\operatorname{argmin}} F(w) \tag{32}$$

where w^f is the obtained final model parameter, each node j first computes $F_j(w)$ and transmits the result to the aggregator, which then computes $F(w)$.

We think about *DK's* different kinds of resources. Each resource is represented by d, where $d \in \{1,2,\ldots,DK\}$, and l_m are used to define the resources consumed by each local update step at all nodes, and each global aggregation step consumes g_m resources, so the total amount of resources consumed is $(ST+1)l_m + (G+1)\,g_m$. To represent the overall budget proposal for m-type resources, we use the notation C_m:

$$\min_{\tau, N \in \{1,2,3\ldots\}} F\left(w^f\right) \tag{33}$$
$$s.t.\,(ST+1)l_m + (G+1)g_m \le CB_m$$

The estimation of l_m and g_m values relies on the kind of resource taken into account; for example, if m is a bandwidth resource, g_m is the sum of all nodes' bandwidth consumption. These values are based on measurements of resource consumption made on the edge nodes and the aggregator. Each type's resource consumption is determined by the aggregator, which

compares it to the CB_m budget, and if the consumed resources reach the limit, the learning process will stop and return the result.

To solve the problem (33), the adaptive aggregation frequency algorithm is used to adjust the frequency of local updates and aggregation of the edge server to solve for the value of τ. The edge server performs the process of local update, and the resultant aggregation is performed on the cloud server; we suppose the aggregator kicks off the training phase by broadcasting $w(0)$, the initial value for the model parameter, to all edge nodes. The method of aggregation frequency mentioned in [14] can be applied to this study to improve the performance of edge computing for vehicular networking.

5. Analysis of experimental results

The hardware environment for this experiment uses the 11th Gen Intel Core i5-11400H @ 270GHz for the CPU, NVIDIA GeForce RTX 3050 for the GPU, and Python 3.7 and Pytorch 1.10.2 for the experimental environment.

To fit the study of Internet of Vehicles, this paper uses a vehicle dataset, which needs to be converted to an MNIST dataset format according to the experimental setup in [14]. This dataset originates from the database of the California Institute of Technology, we divided the dataset into eight categories which were divided into two cases with and without vehicles, and the two cases included far, middle, left, and right. The dataset is converted to a grayscale map and *idx* format. As shown in **Fig. 4**.

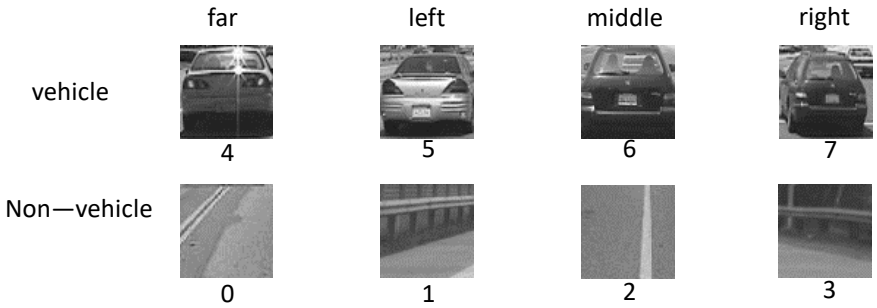

Fig. 4. Vehicle classification

Due to the consideration of experimental cost, we conduct the experiments in a simulated environment with three nodes, and the experimental computer is used as a node along with the cloud server. Heterogeneous clients were generated based on the class and size of the local training data, and two classes were extracted for each client j. These nodes have different two-class datasets on which the local model training is performed.

To better interpret the results, setting M to 1 and the high delay requirement in vehicular networking, time is used as a single resource type in the experiment. l_m and g_m denote the time required to update at the edge server and the time required to perform global aggregation at the cloud server, respectively.

This paper examines four different modes for distributing data to different nodes. Mode 1: randomly assigning a set of data to edge servers; Mode 2: every edge server's set of data has the same label; Mode 3: each edge server has the complete data set; Mode 4: the other data are assigned to the later part of the servers, while the first half of the servers are given the first four labels. The experimental results are shown in **Fig. 5a** and **Fig. 5b**.

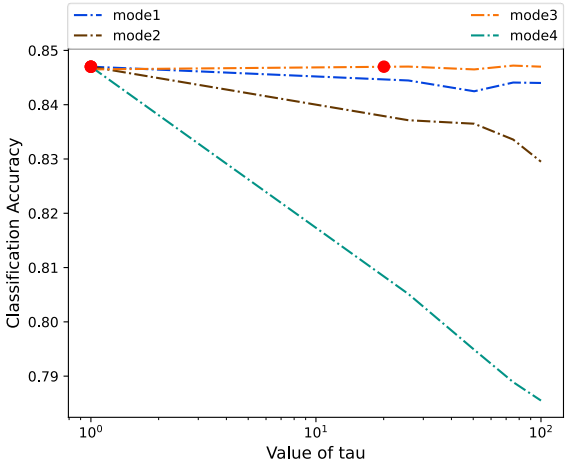

Fig. 5a. Classification accuracy

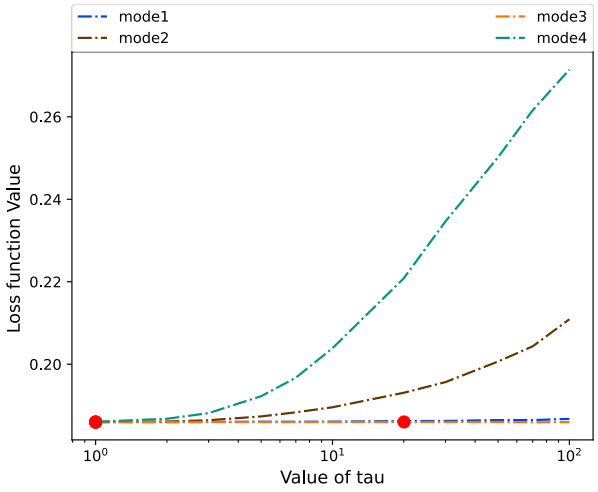

Fig. 5b. Loss function value

The values determined by the frequency of model aggregation by the adaptive federated learning method on the edge server are indicated by the dots in the above image. The experiments in **Fig. 5a** and **Fig. 5b** make use of three edge computing nodes. The aggregation frequency is used to maximize classification accuracy, as shown in **Fig. 5a**. **Fig. 5b** shows the variation of the loss function values in the four modes. To minimize the loss function, the aggregation frequency at the locations is employed. As a result, this technique calculates the ideal model value faster and with less delay. Thus, the adaptive federated learning method applies to MADDPG-based task offloading and is acceptable for the low delay scenarios required in the IoV environment.

Table 2. Partial simulation parameters

Number of vehicles N	[0,6]
Number of Edge nodes	3
CPU workload C_n	[0-1.5]Gcyles
Energy coefficient κ	$5*10^{-27}$
batch size	64
Learning rate	0.01
noise variance σ^2	100dBm

To conduct this experiment, some parameters are set, as shown in **Table 2**, and the results are averaged over 2000 simulations. First, we evaluate the offloading utility of the proposed algorithm under different algorithms. To verify the superiority of the collaborative Improved MADDPG algorithm designed in this paper, the algorithm is verified by comparing the cutting-edge intelligent body collaboration schemes such as MADDPG, MAA2C [23] (Multi-Agent Advantage Actor-Critic), and so on. MADDPG and MAA2C algorithms are used for multi-intelligent reinforcement learning, aiming to maximize the reward of multiple intelligences on a set of tasks. MAA2C uses a centralized critic to estimate the value function for all agents, while MADDPG uses a decentralized critic that estimates the value function for each agent separately ; but neither algorithm is combined with federated learning. The network structure of MAA2C as shown in **Fig. 6**.

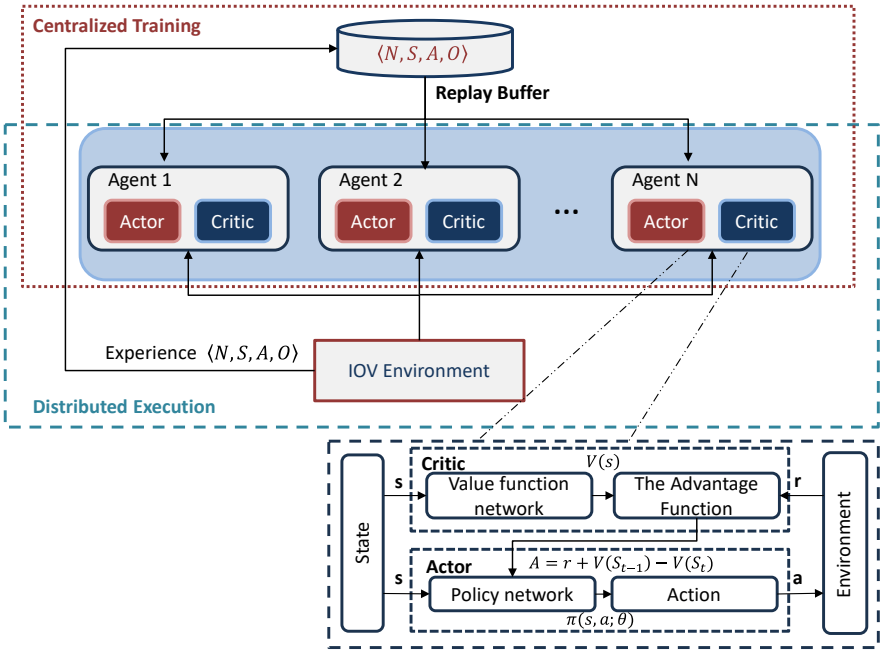

Fig. 6. MAA2C network structure

The training and execution phases of the algorithm can be separated. The agent interacts with the IoV environment during execution to produce experiences, which are then saved in the replay buffer. The agent uses a replay buffer as a source of training experiences for their actor-network and critic network. Each agent accounts for additional knowledge about the states and activities of the other agents through the experience extrapolated from the shared replay buffer, and their interactions are straightforward, and estimate the value function of the current state using a value function network and calculates the advantage value of each action, which represents the benefit of the current action relative to the average action value.

When employing Improved MADDPG, **Fig. 7** shows the relationship between the number of vehicles and the number of channels. As more users use the same number of channels, there can be channel congestion and an increase in delay in the system. Because Improved MADDPG supports distributed computing and task unloading, as the number of users grows, so do the available computing resources in the system, and edge computing resources can be leveraged more completely to process tasks. This distributed computing and task offloading can speed up task processing while decreasing delay. However, as the number of channels rises, the delay falls when the number of cars stays the same. This is because adding more channels can enhance the system's capacity for parallel transmission, lessen the likelihood of channel congestion, and allow for the simultaneous transmission of data by more users. This can shorten each user's wait time, which can shorten their transmission delay. However, an excessive growth in the number of channels could result in higher costs. To support each channel, associated hardware, and spectrum resources are needed, and obtaining these resources requires cost. We limited the number of channels to 3 to save cost.

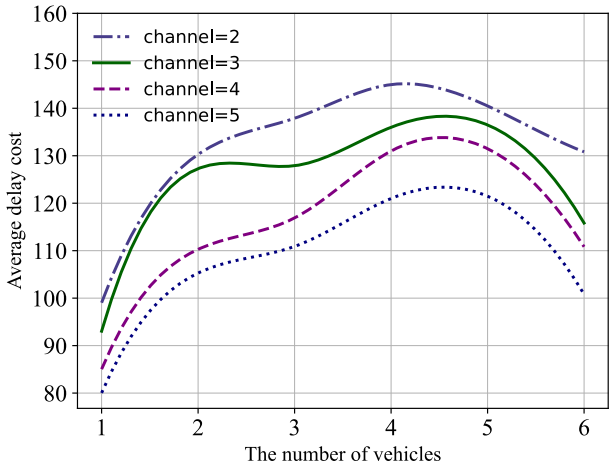

Fig. 7. The impact of channel numbers on different vehicle numbers

Fig. 8 illustrates the performance of the average offloading utility across varying numbers of vehicles. It can be seen that our proposed algorithm demonstrates the best offloading utility, the average offloading utility increases with the number of vehicles for less than four because the MEC system has sufficient spectrum and computational resources to manage all of the tasks offloaded from the vehicles in this instance. When thresholds are exceeded, the system's utility drops due to the higher number of vehicles, offloading tasks, and resource utilization competition. Our Improved MADDPG scheme still achieves optimal utility performance compared to other learning schemes due to the adaptive algorithm. With five vehicles, the average offloading utility of the improved MADDPG is 2% and 9% higher than that of the MADDPG and MAA2C schemes, respectively. In summary, the Improved MADDPG algorithm works well.

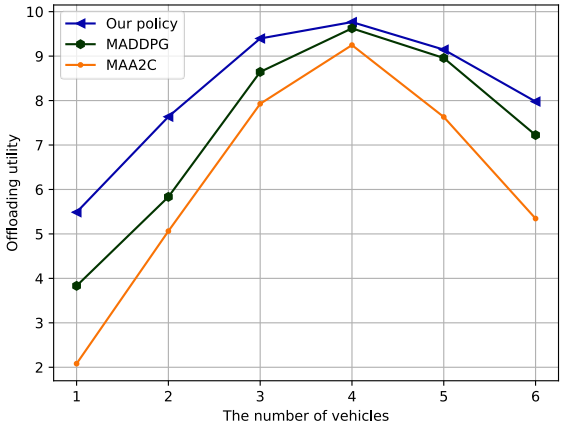

Fig. 8. The utility of different algorithms

To test whether the improved MADDPG's performance is affected by the number of vehicles, the global model iteration tests in this research vary from 1 to 6. As illustrated in **Fig. 9**, when the number of iterations is small, the utility is minimal or even negative; as the number of iterations increases, the results tend to stabilize gradually. And the number of users affects model quality; when there are four vehicles, the offloading utility is the greatest, because the vehicles have to interact with one another to exchange model parameters and learning results. As the number of vehicles grows, rises the communication overhead. Initially, as the number of vehicles increases, more model updates become accessible, improving the offloading utility. However, when the number of vehicles is too large, the communication overhead increases and can outweigh the benefits of the offloading operation itself, resulting in a drop in utility.

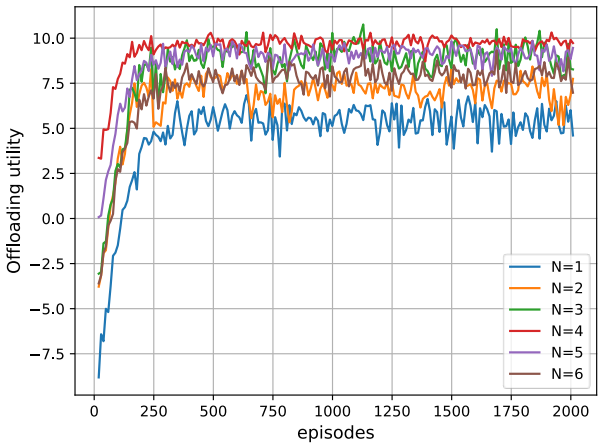

Fig. 9. Performance of Improved MADDPG under different numbers of vehicles

Fig. 10 shows the simulation of the utility when performing task offloading with different sizes of bandwidth from 20 bps to 120 bps on the IoV environment. As the bandwidth increases, the offload utility of all three methods increases, which is because the larger the bandwidth, the faster the data transfer will be, and thus the task offload can be done faster. Also, more bandwidth means more reliable data transfer because there is less probability of error or loss

during data transfer. Compared with the other two schemes, Improved MADDPG can obtain higher utility; for example, when the bandwidth is 100bps, the average offload utility of the Improved MADDPG scheme is 3.6% and 6.8% higher than that of the MADDPG, MAA2C scheme, respectively.

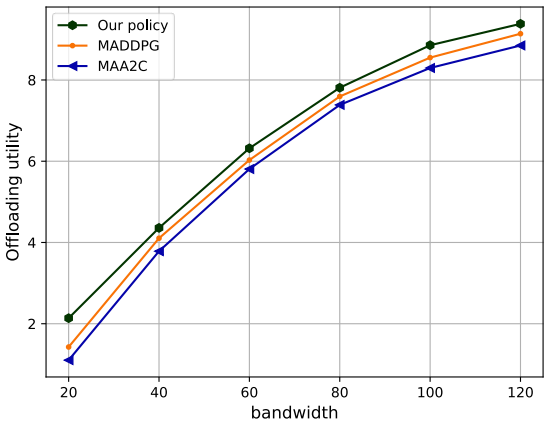

Fig. 10. Offloading utility under different bandwidths

Fig. 11 shows the significant reduction in overall delay with the Improved MADDPG algorithm. However, as the quantity of vehicles grows, the delay grows as well due to the correlation between the number of users and the delay. The complexity of the tasks to be processed increases accordingly. When the number of participating vehicles is all 3, the Improved MADDPG algorithm reduces the delay by 29.6% compared to the MADDPG algorithm.

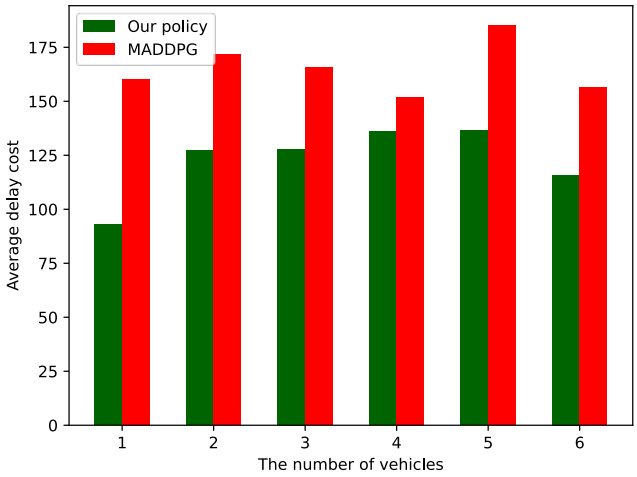

Fig. 11. Comparison of delay between two algorithms

We analyzed the vehicle's computing capacity as well as the processing effectiveness of tasks that were locally executed, as shown in **Fig. 12**. Due to the vehicle's limited computational capacity and the tasks' poor processing efficiency, the results demonstrate that local execution causes the system consumption to be at its greatest. For local users, using MEC

to do tasks has a cheaper computational cost than doing it locally. However, sending tasks to edge servers for offloading presents connection quality uncertainty, and various link states can occasionally cause transmission congestion. Although there is no local computing overhead in this scenario, the network's intricacy makes it difficult to reduce the total overhead. The random offloading decision algorithm, which mixes edge and local execution strategies, was also studied. Because the transmission calculation is only executed when the link quality is good, we discovered throughout our studies that the random offloading strategy results in significant overhead.

We provide an Improved MADDPG algorithm that has the lowest cost out of the three methods mentioned above. By producing work scheduling decisions at the lowest possible cost, it enhances the quality of the user experience. Experiments demonstrate that compared to the other three ways, our suggested algorithm uses fewer system resources under the same circumstances.

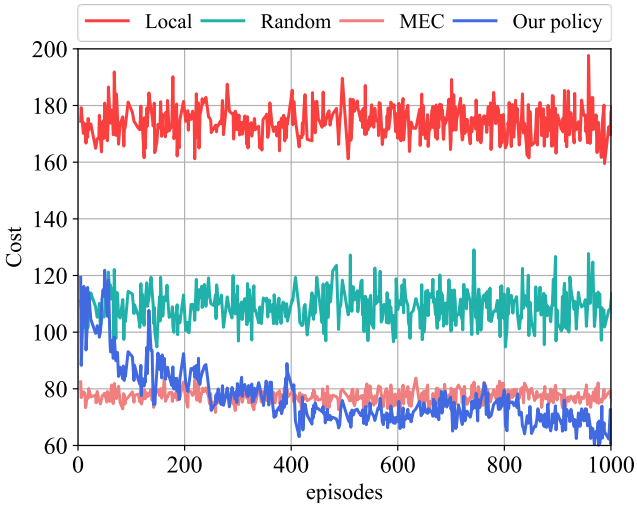

Fig. 12. Cost under different task processing methods

6. Conclusion

The purpose of this study is to use an Improved MADDPG algorithm to make appropriate task offloading decisions in the Internet of Vehicles. Experiments have demonstrated that this algorithm substantially reduces delay while also improving offloading utility. The MADDPG method incorporates federated learning to provide the optimal trade-off between local update and overall aggregation, fixing the high real-time requirements of distributed edge computing in the Internet of Vehicles. Further study is needed in the future for situations with widely spread and numerous edge nodes to achieve the effectiveness of task offloading in heavy traffic areas.

References

[1] Shichao Li et al., “Joint Admission Control and Resource Allocation in Edge Computing for Internet of Things,” *IEEE Network*, vol. 32, no. 1, pp. 72-79, Jan.-Feb. 2018.
Article (CrossRef Link).

[2] Ilija Hadžić, Yoshihisa Abe, and Hans Christian Woithe, “Server Placement and Selection for Edge Computing in the ePC,” *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 671-684, 1 Sept.-Oct. 2019. Article (CrossRef Link).
[3] Abhishek Hazra et al., “Federated-Learning-Aided Next-Generation Edge Networks for Intelligent Services,” *IEEE Network*, vol. 36, no. 3, pp. 56-64, May/June 2022. Article (CrossRef Link).
[4] Niranjan K. Ray, Deepak Puthal, and Dhruva Ghai, “Federated Learning,” *IEEE Consumer Electronics Magazine*, vol. 10, no. 6, pp. 106-107, Nov. 2021. Article (CrossRef Link).
[5] Zhu Z, Hong J, Zhou J, “Data-free knowledge distillation for heterogeneous federated learning,” *International conference on machine learning (PMLR)*, pp. 12878-12889, July 2021. Article (CrossRef Link)
[6] Lowe R, Wu Y I, Tamar A, et al., “Multi-agent actor-critic for mixed cooperative-competitive environments,” *Part of Advances in Neural Information Processing Systems (NIPS)*, 2017 Article (CrossRef Link)
[7] S. Wang, Y. Guo, N. Zhang, P. Yang, A. Zhou and X. Shen, “Delay-Aware Microservice Coordination in Mobile Edge Computing: A Reinforcement Learning Approach,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 939-951, March 2021. Article (CrossRef Link).
[8] R. Shuai, L. Wang, S. Guo and H. Zhang, "Adaptive Task Offloading in Vehicular Edge Computing Networks Based on Deep Reinforcement Learning," in *Proc. of 2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 260-265, 2021. Article (CrossRef Link).
[9] H. Ke, J. Wang, L. Deng, Y. Ge and H. Wang, "Deep Reinforcement Learning-Based Adaptive Computation Offloading for MEC in Heterogeneous Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7916-7929, July 2020. Article (CrossRef Link).
[10] Z. Qin, S. Leng, J. Zhou and S. Mao, "Collaborative Edge Computing and Caching in Vehicular Networks," in *Proc. of 2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1-6, 2020. Article (CrossRef Link)
[11] 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," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2252-2261, April 2021. Article (CrossRef Link).
[12] Wang G, Xu F, Zhao C, “QoS‐enabled resource allocation algorithm in internet of vehicles with mobile edge computing,” *IET Communications*, vol 14, no 14, pp, 2326-2333, 2020. Article (CrossRef Link)
[13] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues and L. Guo, "Mobile Edge Computing-Enabled Internet of Vehicles: Toward Energy-Efficient Scheduling," *IEEE Network*, vol. 33, no. 5, pp. 198-205, Sept.-Oct. 2019. Article (CrossRef Link).
[14] S. Wang et al., "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205-1221, June 2019. Article (CrossRef Link).
[15] Y. Lu, X. Huang, Y. Dai, S. Maharjan and Y. Zhang, "Differentially Private Asynchronous Federated Learning for Mobile Edge Computing in Urban Informatics," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 2134-2143, March 2020. Article (CrossRef Link).
[16] S. Samarakoon, M. Bennis, W. Saad and M. Debbah, "Distributed Federated Learning for Ultra-Reliable Low-Latency Vehicular Communications," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1146-1159, Feb. 2020. Article (CrossRef Link).
[17] H. H. Yang, Z. Liu, T. Q. S. Quek and H. V. Poor, "Scheduling Policies for Federated Learning in Wireless Networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 317-333, Jan. 2020. Article (CrossRef Link).
[18] M. M. Amiri and D. Gündüz, "Federated Learning Over Wireless Fading Channels," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3546-3557, May 2020. Article (CrossRef Link).

[19] W. Bao, C. Wu, S. Guleng, J. Zhang, K. -L. A. Yau and Y. Ji, "Edge computing-based joint client selection and networking scheme for federated learning in vehicular IoT," *China Communications*, vol. 18, no. 6, pp. 39-52, June 2021. Article (CrossRef Link).
[20] D. Chen et al., "Matching-Theory-Based Low-Latency Scheme for Multitask Federated Learning in MEC Networks," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11415-11426, July, 2021. Article (CrossRef Link).
[21] Canh T Dinh, Nguyen Tran, and Josh Nguyen, "Personalized federated learning with moreau envelopes," in *Proc. of 34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, pp. 21394–21405, 2020. Article (CrossRef Link)
[22] Y. Zhou, H. Yu, Z. Li, J. Su and C. Liu, "Robust Optimization of a Distribution Network Location-Routing Problem Under Carbon Trading Policies," *IEEE Access*, vol. 8, pp. 46288-46306, 2020. Article (CrossRef Link).
[23] Xiao Y, Tan W, Amato C, "Asynchronous Actor-Critic for Multi-Agent Reinforcement Learning," *arXiv preprint arXiv:2209.10113*, 2022. Article (CrossRef Link).

Ziyang Jin received the B.S. degree in Communication Engineering from Northeast Electric Power University, Jilin, China in 2020, She is pursuing a master's degree at Changchun University of Science and Technology, Changchun, China. Her research interests include edge computing and Internet of Vehicles.

YiJun Wang received the B.E. degree in communication engineering from College of Electronic and Information Engineering, Lanzhou Jiaotong University, Lanzhou, China in 2006, and the M.E. degree in signal and information processing from College of Electronic and Information Engineering, Changchun University of Science and Technology, Changchun, China in 2009 and the Ph.D. degrees in College of Communications Engineering from Jilin University, Changchun, China in 2012. He is an Associate Professor with the Changchun University of Science and Technology. His research interests include 5G/6G, Ad Hoc Network.

Jingying Lv received the B.S. degree in Communication Engineering from Changchun University of Science and Technology, Changchun, China in 2020, She is pursuing a master's degree at Changchun University of Science and Technology, Changchun, China. Her research interests include edge computing and Internet of Things.