Multi-Agent Reinforcement Learning for Cooperative Task Offloading in Internet-of-Vehicles

Yuchen Lei[1], Kai Jiang[1], Zhenning Wang[1], Yue Cao[1*], Hai Lin[1], Liang Chen[2]
[1]School of Cyber Science and Engineering, Wuhan University, China. yue.cao@whu.edu.cn
[2] State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, China.

***Abstract*—The Internet of Vehicles (IoV) has witnessed a significant growth in the number of participants. This rapid expansion has led to increased demands for computing resources and quality of service (QoS), posing challenges for mobile edge computing (MEC) in the IoV domain. Efficiently allocating computing power to meet these service demands has become a crucial concern. Specifically, we need jointly optimize offloading decision and power allocation to achieve a trade-off between task latency and energy consumption. To address this challenge, in this paper, we propose a multi-agent reinforcement learning (MARL) method called multi-agent twin delayed deep deterministic policy gradient (MA-TD3). This algorithm improves performance and execution speed compared to its predecessor, multi-agent deep deterministic policy gradient (MADDPG). It solves the slow convergence problem caused by Q-value overestimation to some extent and reduces the computational cost. The experimental results illustrate that the proposed algorithm reaches an observable performance improvement.**

***Index Terms*—internet of vehicle, mobile edge computing, multi-agent deep reinforcement learning**

I. Introduction

The emergence of connected vehicles assisted by artificial intelligence (AI) has subverted the entire automotive industry, promoting the development of future mobility. However, the vast and diverse service demands of applications, including in-car entertainment applications, real-time image recognition, autonomous driving, etc, have brought significant challenges to vehicular computing.

Recently, mobile edge computing (MEC) has been recognized as a promising solution to handle intensive computation tasks of mobile applications. By introducing MEC into the Internet of Vehicles (IoV), vehicles have the ability to offload their computing tasks to nearby edge servers, in addition to using their own computing power. However, since computation offloading introduces additional transmission latency and energy consumption, the quality of the offloading decision is crucial for the efficiency of edge computing in the IoV.

In general, offloading decisions in IoV include offloading target selection, offloading power allocation, central processing unit (CPU) resource allocation, and wireless channel allocation [1], [2]. Generally, the combination of these actions is within a mixed discrete and continuous action space, thus is deemed as a huge challenge for optimization. Also, with the increasing number of vehicles in the network, the combination of states and actions will inevitably incredibly rise.

Along with extensive previous works applying typical optimization mechanisms such as game theory and linear programming exists, recent works have utilized emerging deep reinforcement learning (DRL). These works focus on developing novel task offloading and resource allocation strategies for IoV. For binary offloading policy or quantified continuous space, deep Q-network (DQN) is utilized to study the optimal decisions. Literature works [3], [4], [5] proposed different DQN-based learning frameworks for discrete resource allocation. To achieve the joint optimization of the assignment of continuous variables, the actor-critic (AC) and its derivative structures are applied. Literature works [6], [7], [8] formulated the optimization problem as a mixed integer non-linear programming (MINLP) problem for offloading decisions, service caching and resource allocation, etc. Specifically, distributed DRL such as asynchronous advantage actor-critic (A3C) is deployed [9] to improve learning efficiency.

Some studies have also attempted to use multi-agent reinforcement learning (MARL) to alleviate dependence on global information at runtime. For problems that can be solved by the MARL method, the MADDPG is a suitable solution. Yuan et al. [10] uses a two-branch convolution-based network to produce migration action and routing action separately. Some techniques like parameter aggregation are also introduced [11] for enhancing learning efficiency.

However, the aforementioned works lack comprehensive understanding of the vehicular network. Many studies employ global information for offloading decisions, which is not applicable in reality. Some studies utilize multi-agent reinforcement learning. However, common approaches like MADDPG inevitably suffer from Q-value overestimation due to the use of only one critic network, which may lead to suboptimal policies and slower convergence. Meanwhile, these studies mostly neglect actual vehicular movement model. Thus, the main contributions are listed as follows:

- We formally define the task offloading and transmission power allocation problem for IoV and introduce quality of service (QoS) levels to enable priority differentiation among tasks to improve the overall user experience. Our optimization objective is a trade-off between task latency and energy consumption.
- We propose the multi-agent twin delayed deep deterministic policy gradient (MA-TD3) algorithm. It solves the slow convergence problem caused by Q-value overestimation. The critic network structure and training loops are improved to enable faster training.

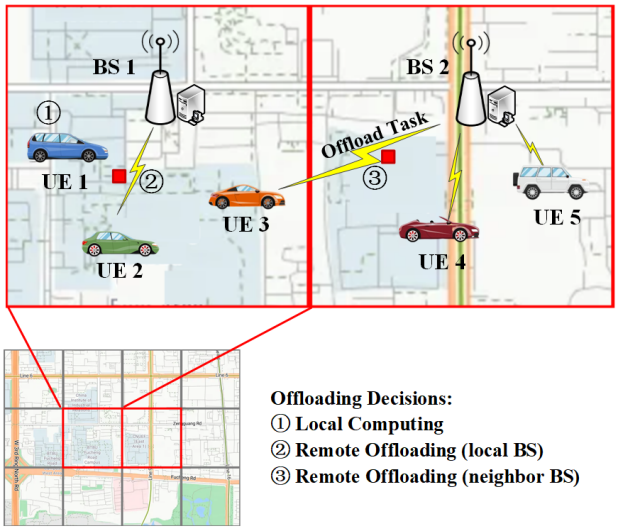

Fig. 1. System Model

II. System Model

In this paper, we consider a time-slotted dynamic MEC network where the system operates on fixed-length time slots, denoted as $\mathcal{T} = \{1, 2, \ldots, T\}$. The system state is varied between time slots. As Fig. 1 shows, the proposed MEC network consists of N vehicles denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$ and a series of base stations (BS) associated with edge servers. The base stations are placed in a grid divided by the area where the vehicles operate, respectively, denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$. The vehicles use MARL to make offloading decisions fully autonomously and individually.

A. Mobility Model

The vehicle's travel path will be based on real-world trajectories. The location of vehicle $i \in \mathcal{N}$ at time slot t is denoted as $\mathbf{u}^{i,t} = (u_x^{i,t}, u_y^{i,t})$, represents the coordinates of the vehicle on the x-axis and y-axis respectively. Unlike locations represented in a Cartesian coordinate vector, speed is demonstrated by direction $\theta^{i,t} \in (-\pi, \pi]$ and speed $v^{i,t}$ as $\mathbf{v}^{i,t} = (\theta^{i,t}, v^{i,t})$. These terms are calculated from its previous location by $\theta^{i,t} = \arctan2(v_y^{i,t}, v_x^{i,t})$ and $v^{i,t} = \sqrt{(v_x^{i,t})^2 + (v_y^{i,t})^2}$, where $v_x^{i,t} = u_x^{i,t} - u_x^{i,t-1}$, $v_y^{i,t} = u_y^{i,t} - u_y^{i,t-1}$.

B. Computing Task Model

We assume that each vehicle periodically generates tasks with different computation requirements and delay tolerance. Specifically, at the beginning of time slot t, a computing task will be generated by vehicle i with a probability of $prob_i$. The task can be characterized by $\psi_i = \{l_i, \rho_i, m_i, q_i\}$. The data size l_i is defined as the amount of data that the task needs to process and is measured in bits or bytes. The CPU cycles ρ_i required to process one bit represent the computational complexity of the task. The maximum delay tolerance m_i measured in seconds, represents the maximum amount of time allowed for the task to be completed without violating the user's requirements. The QoS is denoted by q_i. If any contract cannot be fulfilled, the contribution of task i will be $-q_i$ to reflect different priorities based on the QoS.

C. Task Offloading Model

In our scenario, the BS can provide computing services to vehicles, which are also called user equipment (UE). We consider binary offloading scenarios, where tasks can be computed locally by UEs, or fully offloaded to the BS. To avoid channel decay due to long distances, we restrict the offloading decisions of UEs to local or neighboring cells. Specifically, there can be three offloading strategies for each task generated by the vehicles:

1) Local Computing: The vehicle can perform the task locally.
2) Remote Offloading via local cell: The vehicle can offload the task to the BS in the local cell.
3) Remote Offloading via neighboring cell: The vehicle can offload the task to a BS in an adjacent cell.

Next, we describe the different computation models.

1) Local Computing: When tasks are executed locally, the time and energy consumption depend on factors such as task length or CPU intensity. Here, different tasks require different computation time and energy consumption. The computing delay and the energy consumption in local computing are described respectively as follows:

$$\begin{aligned} T_l &= \frac{l_i \rho_i}{f_i}, \\ E_l &= T_l \kappa f_i^2 = l_i \rho_i \kappa f_i, \end{aligned} \tag{1}$$

where f_i is the CPU frequency of UE$_i$ and κ is a constant of CPU effectiveness.

2) Remote Offloading: Remote execution is defined as the transfer of tasks to the BS for processing. The requirements of tasks, including transmission time and energy consumption, depend on the time-varying channel conditions between the vehicle and the BS. In the MEC environment, the main concern is the total task time and the energy consumed by the user device. In our scenario, we consider the link transmission delay and the edge computation delay. For energy consumption, only transmission power is considered.

At time slot t, given the channel gain $g_{i,j}$ between UE$_i$ and BS$_j$, the transmission power p_i of UE$_i$, the transmit rate $r_{i,j}$ can be calculated by:

$$r_{i,j} = B \cdot \log_2(1 + \frac{p_i \cdot g_{i,j}}{\text{NP}}), \tag{2}$$

where B indicates channel width and NP is the noise power.

From the obtained communication rate, the transmission delay denoted as T_t between UE$_i$ and BS$_j$ can be formed by $T_t = l_i / r_{i,j}$.

After receiving the task, the BS will be responsible for the computation task processing. Denoting f_j as the CPU frequency of BS$_j$, the task computation of UE$_i$ at time slot t denoted as T_c can be formed as $T_c = l_i / f_j$.

Accordingly, the total time consumption of remote execution can be formed as:

$$\begin{aligned} T_r &= T_t + T_c \\ &= \frac{l_i}{r_{i,j}} + \frac{l_i}{f_j}. \end{aligned} \tag{3}$$

In addition, we also need to consider the transmission energy consumption of the UE. When the transmission power $p_{i,j}$ and transmission delay T_t are known, the energy consumption can be calculated by:

$$\begin{aligned} E_r &= p_{i,j} \cdot T_t \\ &= \frac{p_{i,j} l_i}{r_{i,j}}. \end{aligned} \tag{4}$$

D. Problem Formulation

We use binary variables $a_{ij} \in \{0, 1\}$ to denote the offloading decision for time slot $t \in T$. Specifically, $a_{ij} = 1$ means vehicle i offloads tasks to BS j. When $a_{i,0} = 1$, it indicates local computation.

This paper aims to minimize the total delay of all tasks and the energy consumption of vehicles. With a $\omega \in [0, 1]$ as the weight to balance energy consumption and latency. We describe the cost of UE$_i$ at time slot t by the following cost function:

$$\begin{aligned} C_i(t) &= \omega T_i + (1-\omega) E_i \\ &= \omega(a_{i,0} T_l + (1 - a_{i,0}) T_r) + \\ &\quad (1-\omega)(a_{i,0} E_l + (1 - a_{i,0}) E_r). \end{aligned} \tag{5}$$

The cost is based on weighted average delay and energy consumption.

Therefore, the optimization problem proposed in this paper can be defined as:

$$\mathcal{P}_1 : \min_{a_{ij}, p_i} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i \in N} C_i(t). \tag{6}$$

with the following constraints:

$$a_{ij} \in \{0, 1\}, \tag{7a}$$

$$\sum_{j=0}^{M} a_{ij} = 1, i \in \mathcal{N}, \tag{7b}$$

$$T_i \leq m_i, i \in \mathcal{N}. \tag{7c}$$

Constraint (7a) denotes a_{ij} as the binary offloading decision indicator. Constraint (7b) ensures that only one offloading target is allowed. Constraint (7c) guarantees that the task execution delay should not violate the tolerance.

Since the objective function contains both integer and continuous variables, the problem is a MINLP. Even if the information is complete, it is nearly impossible to solve using traditional optimization methods. In order to obtain the optimal solution of the optimization problem $\mathcal{P}_1$, it is necessary to know the state information related to vehicles and BSs. Collecting such information in a decentralized and autonomous MEC network is not practical.

Nonetheless, each vehicle can still obtain its own status, task requirements, and channel information with surrounding BSs. Therefore, the optimization problem can be modeled using a decentralized partially observable Markov decision process (Dec-POMDP).

III. MA-TD3 FOR OFFLOADING DECISION MAKING

This section models the optimization process as Dec-POMDP and tackles the corresponding problem based on the framework of MA-TD3.

A. State, Observation, Action, and Reward

Since obtaining global information for optimal decisions is impractical, we expect each vehicle to make decisions distributedly with collaborating. We employ each vehicle as an agent in MARL. Tuple $\{\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}\}$ can characterize a formal definition of Dec-POMDP in each time slot. Four critical elements are identified as follows:

a) Global State $\mathcal{S}$: The global state involves four parts $\mathcal{S} = \{\mathbf{u}, \mathbf{v}, \mathbf{\Psi}, \mathbf{G_{N \times M}}\}$, where $\mathbf{u}, \mathbf{v}$ denote the position and velocity of all vehicles, respectively; $\mathbf{\Psi}$ indicates the task information generated by all vehicles (i.e., task size, CPU demand density, maximum allowable delay); and matrix $\mathbf{G_{N \times M}}$ encodes the channel state between vehicles and the base station. The formal expression is as follows:

$$\begin{aligned} \mathcal{S} =& \{\mathbf{u}, \mathbf{v}, \mathbf{\Psi}, \mathbf{G_{N \times M}}\} \\ =& \{\{\psi_1, \psi_2, \ldots, \psi_N\}, \{g_{i,j} | i \in \mathcal{N}, j \in \mathcal{M}\}, \\ & \{\mathbf{u}^i, | i \in \mathcal{N}\}, \{\mathbf{v}^i, | i \in \mathcal{N}\}\}. \end{aligned} \tag{8}$$

b) Local Observation $\mathcal{O}$: For each vehicle i, observed information comprises its task information and the channel status of BSs it can offload to. As a subset of the global state, the observation is expressed as follows:

$$\mathcal{O}_i = \{\mathbf{u}^i, \mathbf{v}^i, \phi_i, g_{i,0'}, g_{i,1'}, \ldots, g_{i,4'}\}, \tag{9}$$

where $g_{i,0'}, g_{i,1'}, \ldots, g_{i,4'}$ denotes channel states between vehicles and possible offloading target BSs.

c) Action Space $\mathcal{A}$: Each vehicle independently makes offloading decisions based on the policy after obtaining its observations, which includes offloading target decisions a_{ij} and offloading power allocation p_i. The action space of vehicle i is as follows:

$$\mathcal{A}_i = \{\mathbf{a}_{i,j}, p_i\}. \tag{10}$$

d) Reward $\mathcal{R}$: In this MEC network, we assume that all agents make decisions cooperatively, requiring them to share the payoff. To minimize the objective function while not violating task constraints, the reward function is set to the negative of the objective function. Additionally, a penalty is applied to tasks if fail to satisfy the QoS q_i. Consequently, the reward function can be expressed as follows:

$$\mathcal{R} = -\sum_{i \in \mathcal{N}} [C_i - I(i) q_i], \tag{11}$$

where $I(i) \in \{0, 1\}$ indicates whether the requirements of the task produced by vehicle i can not be satisfied.

Algorithm 1: MA-TD3 Algorithm for Task Offloading

Input: batch size B, policy update interval ui

Initialize actor networks μ_i for each agent with parameter θ_i
Initialize actor networks μ_i for each agent with parameter θ_i
Initialize critic network Q_1, Q_2 with parameter ϕ_1, ϕ_2
Initialize target actor networks μ_i' for each agent with parameter θ_i'
Initialize target critic networks Q_1' and Q_2' with parameter ϕ_1', ϕ_2'
replay buffer R
for *each episode* **do**
  Initialize environment state s;
  for *for each slot of episode* i **do**
    for *each agent* i **do**
      Select action $\mathcal{A}_i$ using strategy based on μ_i and current observation $\mathcal{O}_i$;
    end
    Execute joint action $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N)$;
    Observe new state $\mathcal{S}'$ and obtain reward $\mathcal{R}$;
    Store transition $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{S}')$ in replay buffer R;
    Sample random minibatch of transitions (s, a, r, s') from R;
    Set target actions a' using target actor networks and next states s' ;
    Set target Q-values $y = \mathbf{r} + \gamma \min(Q_1'(s', a'), Q_2'(s', a')$;
    Update critic networks Q_1 and Q_2 by minimizing the loss (14);
    if t % $pi = 0$ **then**
      for *each agent* i **do**
        Update actor network μ_i using the gradient from (16);
      end
      Update target actor networks by (19) ;
      Update target critic networks by (18) ;
    end
  end
end

B. MA-TD3 for task offloading

With knowledge of how TD3 makes improvements to DDPG, we propose MA-TD3 to resolve the caveats of MADDPG. The framework of this algorithm is shown in Fig. 2, and the detail is present in Algorithm 1. Specifically, for every time slot t, given the current state $\mathcal{S}$, every vehicle, acts as an agent and chooses the action $\mathcal{A}_i$ from its observation $\mathcal{O}_i$ by policy μ_i with parameter θ. The action selection process can be formulated as $\mathcal{A}_i = \mu(\mathcal{O}_i; \theta)$. This method utilizes twin critic networks and delayed updates, which are described as follows.

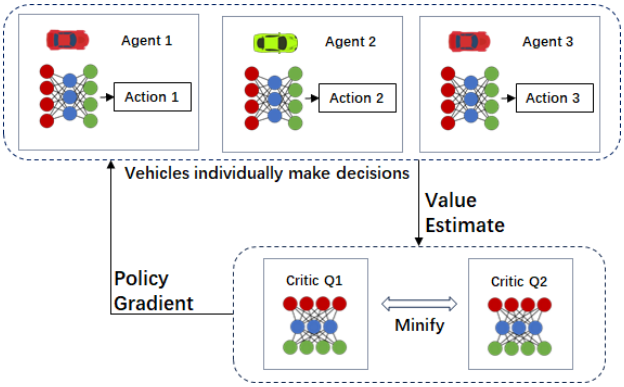

Fig. 2. MA-TD3 Model

1) Twin global critics: The critic's objective is to approximate the optimal Q-function, which provides an estimate of the expected future rewards for an agent following a specific policy. Agents interact with the environment through their actions. Then evaluate them using the global critic networks. To address the Q-value overestimation issue, we utilize two critic networks to estimate the Q-values for all agents more accurately. After we sample the transitions from the replay memory, including current states s, action a, reward r, and next states s'. Each agent selects the next action by their target policy networks, which is then aggregated to a'. With a discount factor γ, the target Q-value estimation from target value networks Q_1', Q_2' can be expressed as:

$$y = \mathbf{r} + \gamma \min(Q_1'(s', a'), Q_2'(s', a')). \tag{12}$$

The introduction of twin critic networks brings additional training costs. Because each agent has its own critic network in the original structure, simply multiplying each critic network directly would double the computation required for training. To mitigate the impact of this improvement, we can observe each Q_i evaluates the value function by observations and actions of all agents in the original MADDPG. Building on this, we discover that for any pair of Q_i, Q_j $(i, j \in \mathcal{N})$, their network inputs and optimization objectives are invariant. Given that Q_i and Q_j are interchangeable, it naturally follows to propose transforming each agent's homogeneous twin critic networks into global.

To train these two value networks, Q_1 and Q_2, we utilize mean squared Temporal-Difference (TD) error as the loss function of the value network. It can be defined as:

$$\mathcal{L}_Q = \frac{1}{K} \sum_{k=1}^{K} (y - Q(s_k, a_k))^2. \tag{13}$$

where K is the number of sampled transitions, y represents the target Q-value computed above, and $Q(s_k, a_k)$ is the estimated Q-value by critic network for each transition k.

To update the critic networks, the gradients of the critic's loss function with respect to its parameters (ϕ) can be computed using the chain rule:

$$\nabla_\phi \mathcal{L}_Q = \frac{2}{K} \sum_{k=1}^{K} \left(y - Q(s_k, a_k)\right) \nabla_\phi Q(s_k, a_k). \tag{14}$$

where $\nabla_\phi Q(s_k, a_k)$ represents the gradient of the estimated Q-value with respect to the critic's parameters ϕ for the critic network.

2) Delayed update: The actor networks of each agent exhibit different characteristics. TD3 updates the actor and critic networks at different intervals. By updating the actor networks less frequently than the critic networks, the critic network is able to provide more accurate and reliable value estimates before updating the actor network. In addition, this structure helps to reduce the variance in the policy updates and improves the convergence properties of the algorithm.

Specifically, as indicated by policy update interval ui in Algorithm 1, we update the critic network in every epoch, but only update the actor network every ui epoch, while performing a soft update on all target networks.

The objective of actors is to maximize the expected return. The loss function for the actor network of agent i can be formulated as the negative of the expected Q-value with respect to the actions taken by all agents while keeping the other agents' actions fixed.

$$\mathcal{L}_{\mu_i} = -\mathbb{E}_{\mathcal{S}, \mathcal{A}_{-i} \sim \mathcal{D}}[Q(\mathcal{S}, \mathcal{A})]. \tag{15}$$

The gradient of the actor loss function with respect to its parameters (θ) can be computed using the chain rule:

$$\nabla_{\theta_i} \mathcal{L}_{\mu_i} = \mathbb{E}_{s, a_{-i} \sim \mathrm{R}} \left[\nabla_{a_i} Q_i(s, a)|_{a_i = \mu_i(\mathcal{O}_i; \theta_i)} \nabla_{\theta_i} \mu_i(\mathcal{O}_i; \theta_i) \right]. \tag{16}$$

where a_{-i} represents the actions of all agents except agent i, R denotes the replay buffer that stores past experiences, and $\mu(\mathcal{O}_i; \theta_i)$ represents the output of the actor network for agent i given its partial observation $\mathcal{O}_i$ and parameters θ_i.

Offloading decision a_{ij} satisfies (7a) (7b). This implies that a_i needs to be a one-hot encoding, rather than a series of continuous real numbers. However, argmax operation producing one-hot encoding will vanish the gradient. To address this issue, instead of quantizing the network's output for action selection, we opted to apply the gradient to the argmax operation. Specifically, we apply gumbel noise g_j [12] to each output that represents a potential destination, followed by the softmax operation. The final result can be approximated as the desired one-hot vector. This approach allows us to leverage the benefits of gradient descent optimization while also benefiting from the intuitive choice brought by the argmax operation. The final action is calculated by

$$a_{i,j} = \frac{\exp(\log a_{i,j} + g_j)}{\sum_{j=1}^{M} \exp(\log a_{i,j} + g_j)} \tag{17}$$

TABLE I
SIMULATION PARAMETERS

Parameters	Value
Number of vehicles	100
Number of cells	64
Operating area	8km × 8km
The size of tasks l_i	[10, 30] Mbits
Computation requirement	[500,800] cycles/bit
Background noise NP	-100 dBm
Channel bandwidth B	20 Mhz
Max transmission power of UE	1 W
Computation capability of BS	30 GHz
Computation capability of UE	2 GHz
QoS priority	[10, 1000]

In addition to updating the policy networks of the actor in every certain cycle, a soft update of the target networks is performed using the following approach:

$$\phi'_{\{1,2\}} = (1 - \tau)\phi'_{\{1,2\}} + \tau \phi_{\{1,2\}}, \tag{18}$$

$$\theta'_i = (1 - \tau)\theta'_i + \tau \theta_i, i \in \mathcal{N}. \tag{19}$$

where τ is a small soft update rate.

When trained centrally until the network converges, each agent should be able to learn how to independently make decisions that maximize the benefit of the full network.

IV. PERFORMANCE EVALUATION

In this section, we will evaluate the advantages of the proposed algorithm from two aspects: algorithm performance and execution. After introducing the parameters used for simulation, we then analyze the simulation results.

A. Simulation Parameters

We model the vehicle movement using real cab trajectory data in Beijing [13] and choose 100 paths of equal duration and collapse them into an area with an 8 km side length. The area is covered by a series of square cells with a side length of 1 km, totaling 64 cells, i.e., we consider 64 base stations in the network serving a total of 100 vehicles. Other parameter preferences not explicitly noted are indicated in Table I.

We implement our algorithm on PyTorch and used AdamW as the optimizer. For comparing the performance of the proposed algorithm, we use the following algorithm as the baseline. In addition to demonstrating the advantage of the algorithm in terms of training time, we also include a comparison with the original MADDPG algorithm.

1) *Random Offloading*: Each vehicle randomly selects a base station with equal probability for task offloading.
2) *Greedy*: Each vehicle chooses an offload target that minimizes the current cost.
3) *Target Selection Only*: Each vehicle uses the proposed algorithm only for offload target selection.
4) *Original MADDPG*: The basic MADDPG employs separate critic networks for each individual agent.

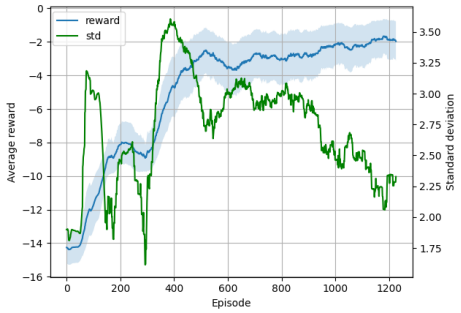

Fig. 3. Convergence of the Proposed Algorithm

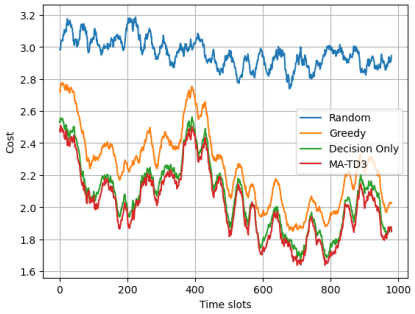

Fig. 4. Performance Comparison

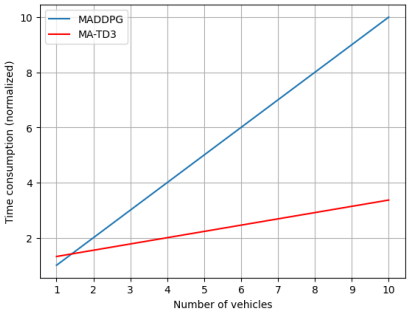

Fig. 5. Time Consumption (normalized)

B. Result and Analysis

We first examine the convergence of the algorithm. We compute the moving average on training cost history while simultaneously calculating the standard deviation in moving window. In Fig. 3, we can see that the average reward for each episode increases rapidly over the first 500 training episodes and then converges to a relatively stable value at last. After the reward stabilizes, we can observe a significant decrease in the standard deviation, indicating that the model becomes more stable in making decisions as the training progresses.

Fig. 4 shows the result of comparing the system cost of the proposed algorithm with three baseline offloading methods. The differences in performance may result from two aspects. Firstly, the local observation information of each agent contains its own position and velocity, enabling the agent to gain foresight. Secondly, agents learned how to optimize the team reward using the twin global critic networks during centralized training, resulting in improved performance.

Lastly, as shown in Fig. 5, we vary the number of vehicles and examine the total training time required. In particular, we specify the training time when there is only one vehicle in origin MADDPG as the baseline. The time consumed under other parameters is expressed as a proportional value relative to this time. Since we only consider the time taken for network training here, the results show an almost perfect linear relationship. Due to employing delayed actor updates and global critic networks, MA-TD3 only incurs longer training time compared to the original MADDPG when it degrades to a single-agent scenario. However, as the number of vehicles increases, the advantages of MA-TD3 become more prominent.

V. Conclusion

In this paper, we propose the MA-TD3 algorithm to solve the task offloading decision problem in autonomous vehicle networking systems. The proposed MA-TD3 minimizes the long-term average cost including task latency and vehicle energy consumption. By introducing the idea of twin critic networks and delayed update, our method solves Q-value overestimation and reduces the computational cost. Simulation results show that these improvements make the training faster while having a faster convergence rate. The proposed algorithm also has a significant advantage over the various baseline algorithms used for comparison.

References

[1] J. Lu, L. Chen, J. Xia, F. Zhu, M. Tang, C. Fan, and J. Ou, "Analytical offloading design for mobile edge computing-based smart internet of vehicle," *EURASIP journal on advances in signal processing*, vol. 2022, no. 1, p. 44, 2022.

[2] Z. Ning, J. Huang, X. Wang, J. J. Rodrigues, and L. Guo, "Mobile edge computing-enabled internet of vehicles: Toward energy-efficient scheduling," *IEEE Network*, vol. 33, no. 5, pp. 198–205, 2019.

[3] L. Huang, S. Bi, and Y.-J. A. Zhang, "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, 2019.

[4] J. Gao, Z. Kuang, J. Gao, and L. Zhao, "Joint offloading scheduling and resource allocation in vehicular edge computing: A two layer solution," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3999–4009, 2023.

[5] Q. Luo, T. H. Luan, W. Shi, and P. Fan, "Deep reinforcement learning based computation offloading and trajectory planning for multi-uav cooperative target search," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 2, pp. 504–520, 2023.

[6] Q. Qi, J. Wang, Z. Ma, H. Sun, Y. Cao, L. Zhang, and J. Liao, "Knowledge-driven service offloading decision for vehicular edge computing: A deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4192–4203, 2019.

[7] A. M. Rahimi, A. Ziaeddini, and S. Gonglee, "A novel approach to efficient resource allocation in load-balanced cellular networks using hierarchical drl," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 5, pp. 2887–2901, 2022.

[8] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast adaptive task offloading in edge computing based on meta reinforcement learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 1, pp. 242–253, 2020.

[9] H. Zhou, Z. Wang, H. Zheng, S. He, and M. Dong, "Cost minimization-oriented computation offloading and service caching in mobile cloud-edge computing: An a3c-based approach," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1326–1338, 2023.

[10] Q. Yuan, J. Li, H. Zhou, T. Lin, G. Luo, and X. Shen, "A joint service migration and mobility optimization approach for vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 8, pp. 9041–9052, 2020.

[11] S. Hu, T. Ren, J. Niu, Z. Hu, and G. Xing, "Distributed task offloading based on multi-agent deep reinforcement learning," in *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2021, pp. 575–583.

[12] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *International Conference on Learning Representations*, 2016.

[13] W. Tong, Y. Tong, C. Xia, J. Hua, Q. A. Li, and S. Zhong, "Understanding location privacy of the point-of-interest aggregate data via practical attacks and defenses," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, pp. 2433–2449, 2023.