NLDDPG Based Joint Optimization Decision Scheme for Vehicular Network Offloading and Resource Allocation

Hui Hu[1], Bei Liu[2], Xin Su[3], Hui Gao[4], Xibin Xu[3]
[1]School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications Chongqing, China
[2]Beijing National Research Center for Information Science and Technology, Tsinghua University Beijing, China
[3]Department of electronic engineering, Tsinghua University Beijing, China
[4]Department of electronic engineering, Beijing University of Posts and Telecommunications, China
Corresponding Author: Hui Hu Email: s210101046@stu.cqupt.edu.cn

***Abstract*—In response to the explosive growth of data computation in vehicular terminals, computation offloading has emerged as a viable solution to mitigate the limitations of resources. Efficient offloading decisions not only meet the demanding requirements of complex vehicular tasks in terms of time, energy consumption, and computational performance but also minimize competition and resource consumption in the network. However, existing work on task offloading in vehicular networks often exhibits certain limitations, such as incomplete consideration of relevant factors or suboptimal utilization of available resources. This research presents the construction of a three-layer vehicular network environment, which is based on cloud and edge computing paradigms. The design entails the formulation of real-time vehicle location tracking and task priority metrics, while also considering the challenges posed by time-varying channels and signal blockage prevalent in vehicular network environments. In this paper, a novel variant of the Deep Deterministic Policy Gradient (DDPG) algorithm NLDDPG is proposed to iteratively train the model, aiming to optimize a weighted objective function. Simulation results show that this algorithm can improve the efficiency and optimize the task average utility.**

***Index Terms*—Offloading decision making, Deep reinforcement learning, Deep reinforcement learning, Cloud computing, Edge computing**

I. Introduction

The latest iteration of mobile communication network technology has greatly contributed to the rapid development and wide application of IoT. Among them, a large number of smart vehicle applications have been developed, such as autonomous driving, collision warning, virtual reality, etc. However, all of them require a large number of computing resources and strict quality of service (QoS) to achieve. Task offloading decision is one of the core problems of vehicular networks. Currently, offloading decisions usually consider time delay, energy consumption, joint time delay and energy consumption, system utility, or custom revenue as offloading objectives to meet real-time requirements. Luo [1] proposed a dynamic planning-based algorithm to achieve low time delay and joint optimization of computation and communication costs using available resources through collaborative computing of multiple edge servers and proximity communication at the edge. Ning [2] proposed a stochastic algorithm based on sample average approximation to approximate the expected future system utility. However, there are various uncertainties as well as dynamic changes in the vehicular network, such as vehicle mobility, network topology changes, and communication channel quality. Traditional static algorithms cannot consider the above factors, so dynamic offloading algorithms come into being.

Li [3] proposed a novel offloading strategy based on deep Q networks (DQN) to dynamically adjust the offloading ratio to ensure the joint optimization of task offloading time delay and energy consumption to improve system performance. Dai [4] designed a algorithm based on DQN to solve the joint optimization problem of bandwidth, computational resource allocation, and rental cost of heterogeneous servers. Wang [5] proposed a mobile edge computing resource allocation and offloading algorithm based on deep reinforcement learning in the AHP-DQN framework to solve the problem of low terminal storage capacity and to meet the diversification phenomenon of network services during task offloading. Liu [6] considered the dynamic requests of vehicles and time-varying communication conditions and proposes a deep reinforcement learning-based algorithm to achieve optimal decisions for joint optimization of task offloading and resource allocation. Li [7] proposed a joint optimization scheme based on Q-learning to implement a real-time energy-aware offloading scheme in vehicular edge computing. Although all the above works consider the dynamic nature of vehicular networks, most of the works only considered edge computing, while tasks that are computationally dense but have low real-time requirements in practical scenarios can be offloaded to cloud server computing, and tasks that are time delay sensitive or have small computation are suitable to be offloaded to edge server computing, so the architecture of cloud-edge collaboration is more applicable to practical scenarios. In addition, most of the above works only consideed minimizing time delay as the research goal, which is also not applicable to practical scenarios.

In contrast to the existing work, this paper considers a network of edge-cloud collaborative architecture with multiple vehicles, multiple edge servers, and a cloud server. Where

This work is supported by National Key R&D Program of China, No.2020YFB1806702

each vehicle has one task in a time slot, and these tasks can be computed locally or offloaded to the edge server and cloud server for executing. In order to guarantee the QoS of the network and minimize the utility of the system, the main contributions of this study are summarized as follows:

- For the complex environment of vehicular networks, this paper considers task priority, channel access, vehicle mobility, time-varying channels, and signal blocking based on the other works to obtain task offloading decisions that are more applicable to real scenarios to minimize the average task utility, where the average task utility is defined as the weighted average of the offloading delay and execution energy consumption of all tasks.
- We propose an improved algorithm NLDDPG to train the neural network model and obtain the optimal decision for the optimization problem.NLDDPG adds the LSTM (Long Short Term Memory) mechanism to DDPG (Deep Deterministic Policy Gradient) to improve the stability and convergence of the algorithm.We use NLDDPG to implement joint optimization of task offloading and resource allocation decisions in vehicular networks. Simulation results show that the model and algorithm used in this paper can significantly optimize the average task utility compared to other algorithms.

The framework for the content of this paper is described next. In Section I, we introduce the background of this paper, related works and the main contributions of this paper. In Section II, we introduce the system model used in this paper and the optimization problem of this paper. In Section III, we describe the specific Markov process and the algorithm used in this paper. In Section IV, we give the simulation parameter settings, simulation results, and related analysis of this paper. Finally, we concluded in Section V.

II. System Model and Problem Formulation

A. Network model

Fig. 1 shows we consider an environment of a macro Base station(BS) with $\boldsymbol{M}$ small-cells, where each small-cell has a small-cell BS, and each small-cell BS with one corresponding MEC server that serves the vehicles in its coverage. The coverage area of the small-cell BS is a circle and considering the overlapping coverage area of adjacent BS, we use a square inside the circle to approximate the coverage area of the BS. The task offloading decision in this paper is represented as follows:

$$\begin{cases} x_{i,0}=1 & \text{Local computing} \\ x_{i,j}=1, \forall j \in \{1,2,\ldots,M\} & \text{MEC server computing} \\ x_{i,M+1}=1 & \text{Cloud server computing} \end{cases} \tag{1}$$

where local computing denotes that the computation task of the i-th vehicle is computed on the vehicle, MEC server computing denotes that the computation task of the i-th vehicle is computed on the MEC server, and Cloud server computing denotes that the computation task of the i-th vehicle is computed on the Cloud server.

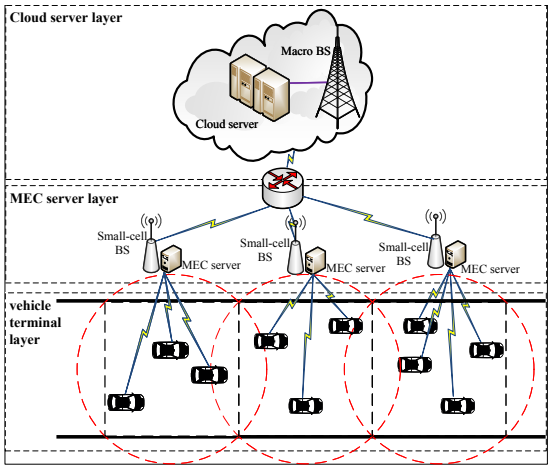

Fig. 1. Network model.

B. Communication model

Because the time delay and energy consumption of the downlink transmission are small compared to the uplink, they can be ignored [8]. The location of the server is fixed, so the position of the RSU can be denoted as coordinate $\boldsymbol{z_j} = \left[z_j^x, z_j^y, H_j\right]^T$, where H_j is the height of the j-th RSU, and the position of the vehicle is moving, so the coordinate of the i-th vehicle of the t-th time slot is denoted as $\boldsymbol{f_i(t)} = [f_i^x(t), f_i^y(t), 0]^T$. Assuming that the direction of the vehicle is constant during the time interval Δt, the coordinate of the vehicle can be denoted as:

$$\begin{aligned} \boldsymbol{f_i(t+\Delta t)} = [&f_i^x(t) + v(t)\cdot\Delta t\cdot\cos\beta(t), \\ &f_i^y(t) + v(t)\cdot\Delta t\cdot\sin\beta(t), 0]^T \end{aligned} \tag{2}$$

If the i-th vehicle is connected to the j-th RSU through the n-th sub-channel, then the corresponding signal-to-noise ratio can be expressed as:

$$\gamma_{i,j}[n] = \frac{P_{i,j}[n]\, h_{i,j}[n]}{\left(\sigma^2 + d_{i,j}[n]\, P_{\text{loss}} + I_{i,j[n]}\right)\cdot\left\|f_i(t+\Delta t) - z_j\right\|^2} \tag{3}$$

where σ^2 is the noise power, $P_{i,j}[n]$ and$h_{i,j}[n]$ denote the transmission power and the channel gain per unit distance of the i-th vehicle to the j-th RSU respectively. The signal blocking flag $d_{i,j}[n]$ is added to distinguish the signal transmission capability, and $d_{i,j}[n]\, in 0, 1$, if $d_{i,j}[n] = 1$, it means that there is signal blocking for the transmission from the i-th vehicle to the j-th RSU through the n-th sub-channel, and vice versa, $d_{i,j}[n] = 0$.

The interference channel from the i-th vehicle to the j-th RSU through the n-th subchannel is denoted as $I_{i,j}[n]$, which can be expressed as:

$$I_{i,j}[n] = \sum_{g=1}^{U_I}\sum_{l=1,l\neq j}^{M} s_{g,j}[n]\, p_{g,j}[n]\, h_{g,j}[n] \tag{4}$$

where l denotes the other RSUs that does not include the j-th RSU.

Assuming that W is the bandwidth of each sub-channel, therefore, if the i-th vehicle transmits information to the j-th

RSU through the n-th sub-channel, the corresponding uplink transmission rate can be expressed as:

$$R_{i,j}[n] = W \log_2 (1 + \gamma_{i,j}[n]) \tag{5}$$

Then, the total uplink transmission rate from the i-th vehicle to the j-th RSU can be expressed as [9]:

$$R_{i,j} = \sum_{n=1}^{L} s_{i,j}[n] R_{i,j}[n] \tag{6}$$

where $s_{i,j}[n] \in 0, 1$, and $s_{i,j}[n] = 1$ means that the i-th vehicle transmits the information to the j-th RSU through the n-th sub-channel to execute the task, and vice versa, $s_{i,j}[n] = 0$. In addition, the uplink communication rate used to offload the computational tasks on the cloud server can be expressed as follows [9]:

$$R_{i,M+1} = \max_{j \in M} R_{i,j} \tag{7}$$

Then, the time for the i-th vehicle to offload the task is:

$$T_i^o = \sum_{j=0}^{M+1} \frac{a_i}{R_{i,j}} x_{i,j} \tag{8}$$

Therefore, the total communication time and energy used to execute the task of the i-th vehicle are expressed as:

$$\begin{cases} T_i^{c_1} = T_i^o + v x_{i,j} \\ E_i^{c_1} = P_i^c T_i^o \end{cases} \tag{9}$$

where v denotes the transmission delay of the i-th vehicle computing task between the MEC server and the Cloud server.

C. Computation model

For local computing, since different vehicles may have different computational capabilities, in addition, all vehicles may execute computational tasks locally. Therefore, the time delay and energy consumption for the local execution of the computational task of the i-th vehicle can be expressed as:

$$\begin{cases} T_i^l = \frac{c_i}{f_i^l} \\ E_i^l = \theta_i c_i \end{cases} \tag{10}$$

where f_i^l denotes the computational capacity of the i-th vehicle and θ_i is a factor indicating the energy consumed per CPU cycle. For remote execution, the computational tasks of the i-th vehicle will be offloaded and executed via a wireless channel connected to an MEC server or via a wireless channel connected to an MEC server to a Cloud server. The time of the MEC server j or the Cloud server $M + 1$ to execute the computational task of the i-th vehicle can be expressed as:

$$\begin{cases} T_i^e = \frac{c_i}{f_{i,j}^e} \\ T_i^c = \frac{c_i}{f_{i,M+1}^c} \end{cases} \tag{11}$$

where $f_{i,j}^e$ and $f_{i,M+1}^c$ denote the computing power allocated to the i-th vehicle by the MEC server j and the Cloud server $M+1$, respectively. Resource allocation, which plays a crucial role in optimizing system performance, is an essential aspect that will be comprehensively examined and discussed in this paper later.

D. Task priority

In this paper, we abstract a criterion for rating task priorities to guide the development of task offloading policies. Because task size and maximum tolerated delay of tasks are important indicators affecting task offloading, linking the two can better reflect the importance and urgency of tasks for task scheduling and offloading and the priority of tasks can be defined as [10]:

$$p_i = p_{sf} \frac{c_i}{T_i^{max}} \tag{12}$$

where p_{sf} is a scaling factor to adjust the degree to which task size or time delay affects priority.

E. Resource allocation scheme based on task priority

When the task of the i-th vehicle is offloaded to the MEC server j or the Cloud server $M + 1$ for processing, the resource allocation factor for the corresponding task can be expressed as follows, respectively:

$$\begin{cases} \rho_{i,j} = \frac{p_i}{\sum_{i \in mec_j} p_i} \\ \rho_{i,M+1} = \frac{p_i}{\sum_{i \in mcc_{M+1}} p_i} \end{cases} \tag{13}$$

Where mec_j denotes the set of all vehicles for task processing at the MEC server j, and mcc_{M+1} denotes the set of all vehicles for task processing at the cloud server $M+1$. Then, the MEC server resources or Cloud server resources allocate to the task of the i-th vehicle for execution are denoted as follows, respectively:

$$\begin{cases} f_{i,j}^e = \rho_{i,j} F_{max}^e \\ f_{i,M+1}^e = \rho_{i,M+1} F_{max}^c \end{cases} \tag{14}$$

where F_{max}^e denotes the maximum computing resource of the MEC server j and F_{max}^c denotes the maximum computing resource of the cloud server $M + 1$.

F. Problem formulation

Based on the obtained formulas for time delay and energy consumption, the total time delay and total energy consumption for all tasks used to perform the i-th vehicle can be obtained as:

$$\begin{cases} T_i^{c_2} = T_i^l x_{i,0} + T_i^e x_{i,M+1} + \sum_{j=1}^{M} T_i^e x_{i,j} \\ E_i^{c_2} = E_i^l x_{i,0} + \sum_{j=1}^{M+1} \theta x_{i,j} \end{cases} \tag{15}$$

where θ is a constant indicates the energy consumed by the vehicle in the idle case. Therefore, the total time delay and energy consumption for the task of the i-th vehicle to be offloaded and executed can be expressed as:

$$\begin{cases} T_i = T_i^{c_1} + T_i^{c_2} \\ E_i = E_i^{c_1} + E_i^{c_2} \end{cases} \tag{16}$$

On the basis of the time taken to offload and calculate the task, combined with the consideration of energy consumption the utility of vehicle i for task offloading can be obtained as:

$$u_i = w_i^t T_i + w_i^e E_i \tag{17}$$

where both w_i^t and w_i^e denote the preference coefficients of delay and energy consumption for task offloading decision by the i-th vehicle, respectively, indicating the importance of delay and energy consumption by the user and satisfying the

constraints $w_i^t + w_i^e = 1, w_i^t \in [0,1]$ and $w_i^e \in [0,1]$. Combining the above analysis, the joint optimization objective of task offloading and resource allocation is expressed in this paper as follows:

$$\min_{x_{i,j},\rho_{i,j},\rho_{i,j}} U = \frac{1}{N}\sum_{i=1}^{n} u_i \tag{18}$$

$$\text{s.t.}: \quad \sum_{j=1}^{M+1} X_{i,j} = 1 \tag{18a}$$

$$\sum_{i \in mec_j} \rho_{i,j} \leq 1, \forall i \in N \tag{18b}$$

$$\sum_{i \in mcc_j} \rho_{i,M+1} \leq 1, \forall i \in N \tag{18c}$$

$$0 \leq \gamma_{M+1} \leq 1 \tag{18d}$$

$$\sum_{i=1}^{N} x_{i,M+1} c_i \leq F_{\max}^c \tag{18e}$$

$$\sum_{i=1}^{N} x_{i,j} c_i \leq F_{\max}^e, \forall j \in M \tag{18f}$$

$$0 \leq f_i^c, \forall i \in N \tag{18g}$$

$$0 \leq f_i^e, \forall i \in N \tag{18h}$$

$$\sum_{i=1}^{N} s_{i,j}[n] \leq 1, \forall j \in M, i \in N \tag{18i}$$

$$f(t) \in [0 \sim 1000, 0 \sim 3000] \tag{18j}$$

where U denotes the average utility of the task. In summary, the optimization objective of this paper involves multiple factors and several constraints, and is a multi-objective joint optimization problem, which is difficult to solve by conventional methods, so the NLDDPG algorithm is used to solve the problem.

III. NLDDPG-BASED TASK OFFLOADING DECISION

Since traditional offloading decision-making schemes are limited by manual design features and complex a priori knowledge, etc., in this paper, we train the optimization problem using a neural network that possesses a high degree of adaptivity and nonlinear modeling capability; in addition, the neural network can take into account the task features and the network state so as to take advantage of the rich information and achieve effective dynamic decision-making on the task's computational requests.

In this paper, we use an algorithm NLDDPG, which can better extract the task specificity to eliminate the size difference of different state values. And this algorithm achieves delayed updates by introducing a fixed-length empirical playback buffer, which reduces the correlation and oscillation of updates and improves the stability and efficiency of learning. Markov decision process (MDP) is a mathematical model commonly used in decision-making [11]. In this section, the three core components of MDP, i.e., state, action, and reward, are first introduced, and afterwards the framework of the algorithm is introduced.

A. State

The state space consists of various state information of servers and vehicles that affect the offloading decision, including Cloud server computing resources, individual MEC server computing resources, task data size, coordinates of vehicles, and block flags. Therefore, the state space of the i-th vehicle at the t-th time slot in the vehicular network is:

$$\begin{aligned} s_i(t) = \big(&F_r^c(t), F_r^1(t), F_r^2(t), \ldots, F_r^m(t), \\ &f_1(t), f_2(t), \ldots, f_n(t), a_1(t), a_2(t), \ldots, a_n(t), \\ &c_1(t), c_2(t), \ldots, c_n(t), d_1(t), d_2(t), \ldots, d_n(t)\big) \end{aligned} \tag{19}$$

where $F_r^c(t)$ denotes the remaining computational resources of the cloud server at the t-th time slot ,$F_r^1(t), F_r^2(t), \ldots, F_r^m(t)$ denote the remaining computational resources of each MEC server at the t-th time slot, $f_1(t), f_2(t), \ldots, f_n(t)$ denote the coordinates of each vehicle at the t-th time slot, $a_1(t), a_2(t), \ldots, a_n(t)$ denote the total data size of the computational tasks that can be offloaded by each vehicle at the t-th time slot, $c_1(t), c_2(t), \ldots, c_n(t)$ denote the priority of the computational tasks of each vehicle at the t-th time slot, $d_1(t), d_2(t), \ldots, d_n(t)$ denotes the block flags of each vehicle at the t-th time slot.

B. Action

The i-th vehicle based on the state $s_i(t)$ observed at the t-th time slot, the vehicle outputs an action as the offloading policy taken at the current time slot. The action $a_i(t)$ of the i-th vehicle at the t-th time slot can be expressed as:

$$a_i(t) = (x_1(t), x_2(t), \ldots, x_n(t)) \tag{20}$$

where $x_i(t)\,(i \in N)$denotes the task offloading decision of the i-th vehicle at the t-th time slot. If $x_i(t) = 0$, the task of the i-th vehicle at the t-th time slot will be executed locally; if $x_i(t) = 1$, the i-th vehicle at the t-th time slot will be executed at the nearby MEC server; if $x_i(t) = 2$, the i-th vehicle at the t-th time slot will be executed at the nearby Cloud server.

C. Reward

At the t-th time slot, the environment gives back a reward $r_i(t)$ immediately after the i-th vehicle gives the corresponding action $a_i(t)$ based on the observed state $s_i(t)$, since the objective of this paper is to minimize the average utility of the task, the reward can be set according to the (18) as:

$$r_i(t) = -U \tag{21}$$

D. NLDDPG algorithm

The purpose of the NLDDPG algorithm is to learn the optimal policy for maximizing long-term gains. The inputs to the NLDDPG algorithm are the vehicular edge computing environment parameters and and the output is the optimization decision. The NLDDPG algorithm process is shown in Fig. 2, which begins with randomly initializing the online policy network and online Q-network weights and copying them to the corresponding target network parameters. Then, the experience playback buffer is cleared and the iteration minibatch is set. This is followed by entering the vehicle edge computing environment and starting the iteration process. The last is to obtain the optimal weights of the actor network after the iteration is completed.

The specific implementation process of the NLDDPG algorithm is as Algorithm 1 shows.

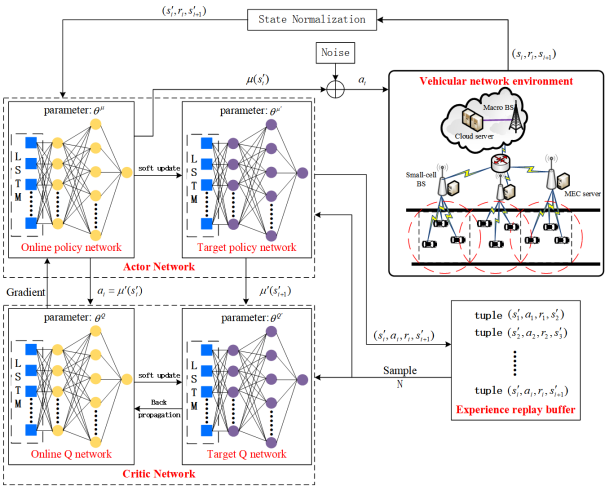

Fig. 2. NLDDPG algorithm architecture.

Algorithm 1 NLDDPG Algorithm Process for Joint Optimization Decision Scheme of Vehicular Network Offloading and Resource Allocation

Input: Initialize the parameters θ^{μ} and θ^{Q} of the Actor Network and Critic Network, target Actor Network $\theta^{\mu'}$, target Critic Network $\theta^{Q'}$ and environment parameters.

Output: Optimal decision-making

Initialize vehicular environment, vehicle generation task request

for episode = 1 to Max_Episode **do**

For each time slot = 1 to T do

Normalize the state s_i to s'_i

Get the action a_i and execute it

Calculate the reward r_i and get the new state s'_{i+1}

Store quaternions $\left(s'_i, a_i, r_i, s'_{i+1}\right)$ to the experience replay R

Randomly select N sample from the R as a small batch of training data

Processing through LSTM networks

Calculation Q value by online Q network: $y_i = r_i + \gamma Q\left(s'_{i+1}, \mu\left[s'_{i+1}\right] \mid \theta^{Q}\right)$

Update the weights of online Q-networks using policy gradients: $L\left(\theta^{Q}\right) = \frac{1}{N}\sum_{j=1}^{N}\left(y_i - Q\left[s'_i, a_i \mid \theta^{Q}\right]\right)^2$

Update the weights of the online strategy network:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_{j=1}^{N}\left[\nabla_a Q\left(s, a \mid \theta^{\mu}\right)\Big|_{s=s'_i, a=\mu\left(s'_i\right)\mid\theta^{\mu}} \nabla_{\theta^{\mu}}\mu\left(s \mid \theta^{\mu}\right)\Big|_{s=s'_i}\right]$$

Soft update for target networks

Use the ϵ-greedy to get the optimal task offloading decision

end for

IV. Simulation and Analysis of Results

To verify the effectiveness of the algorithm, this paper simulates a vehicular network environment using Python. This environment has one Cloud server, three MEC servers, and several vehicles, where each small-cell covers part of the same number of vehicles. The parameters need to be set based on the actual environment, and for the reference project [12] and we set similar values in the simulation as TABLE I.

TABLE I
Related Parameters of NLDDPG

Simulation parameters	Parameters Value
Small-cell BS coverage/(m^2)	1000
Layers	4
Neurons	100,10,300,10
Batch size	64
Learning rate	6e-7
Soft update factor	0.01
Episodes	600
Maximum MEC calculation capability/(Gbit)	10
Maximum Cloud calculation capability/(Gbit)	50
Task size/(Mb)	[1,2],[20,30],[100,150]
The CPU cycles needed to compute/(cycles/bit)	1000

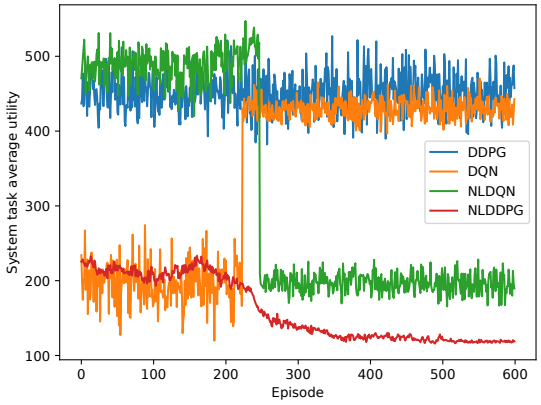

Fig. 3. System average task utility change using NLDDPG.

In this paper, we conduct experiments to evaluate the performance of the proposed NLDDPG algorithm by comparing it with three other popular algorithms, namely DQN, DDPG, and NLDQN. The objective was to assess the average task utility of the entire system, considering a scenario where a small-cell base station covers a group of five vehicles. Fig. 3 shows that the use of the NLDDPG algorithm results in the lowest average task utility and the best stability of the system at the 400th episode as the number of iterations increases. resources by exclusively assigning all computational tasks to the server. The results obtained using the DQN algorithm are the opposite of what we want to achieve in this paper while using NLDQN gives better performance after the 247th episode, but its stability and convergence are poor to get the optimal results. After using the DDPG algorithm, the system will get different results with the increase of episodes but its optimality finding algorithm is poor because there are a large

number of tasks and vehicles in the complex vehicle network, which exceeds the capacity of the neural network, therefore, this paper adds LSTM to the neural network to extract the environmental features.

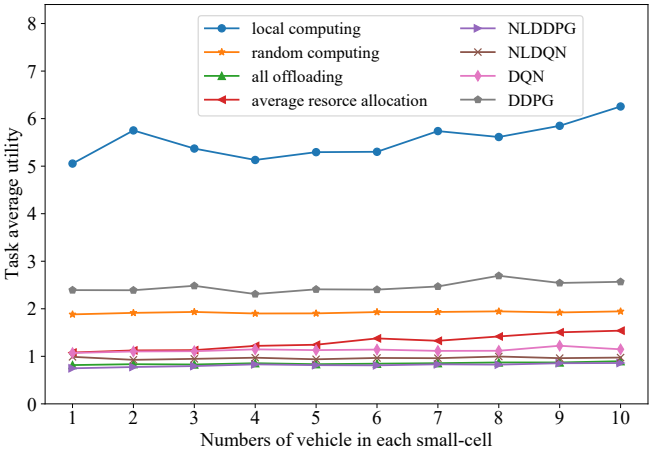

Fig. 4. Task average utility using different methods.

To further validate the performance of the NLDDPG algorithm, we further compare the full offloading, random offloading, local computation and average resource allocation strategies. Among them, the full offloading strategy implies that all tasks of the vehicle are offloaded to the server for execution, random offloading implies that the tasks are offloaded to the local or the server for computation, and local computing indicates that the tasks are processed entirely using local resources.

Fig. 4 shows the variation in average task utility among the different methods as the number of vehicles in a small cell, and consequently the number of tasks, increases. Notably, the performance of the local computation strategy is observed to be the least favorable, exhibiting the lowest average task utility. Conversely, both the NLDDPG algorithm and the all-offloading strategy demonstrate superior performance, displaying the highest average task utilities. However, it is important to note that the all-offloading strategy disregards the utilization of local resource.

Fig. 4 shows the variation of the average task utility between the different approaches as the number of vehicles in a small cell increases. In this, the local computation strategy gives the poorest average task utility and shows the worst performance. Whereas all offloading, random offloading, and average resource allocation strategies all give relatively better performance, it can be seen that the use of deep reinforcement learning algorithms also gives relatively better performance. Among them, the NLDDPG algorithm and the full offloading strategy show the best performance, however, it is important that the full offloading only utilizes the local resources, which can lead to the waste of server resources as well as an increase in latency, etc.

Furthermore, the algorithm proposed in this paper showcases an improvement in utility compared to the average assignment algorithm. This improvement highlights the effectiveness of the priority-based resource allocation approach presented in this study, which facilitates enhanced resource utilization. By leveraging the prioritization mechanism, the proposed algorithm optimizes the allocation of resources, leading to increased task utility and more efficient utilization of available resources.

V. Conclusion

To alleviate the contradiction between increasingly rich in-vehicle applications and limited vehicle resources, this paper constructs a cloud-side collaboration-based offloading decision architecture for vehicular networks, and models the task offloading problem as a Markov decision problem with the average task utility composed of time delay and energy consumption as the optimization objective, and proposes a joint optimization scheme for task offloading and resource allocation based on the NLDDPG algorithm. By considering the task priority, the average task utility in the system is optimized and the task processing latency is reduced. The experimental results show that the proposed scheme in this paper improves the effectiveness and accuracy of decision-making to a certain extent and achieves the minimization of the average task utility of the current policy.

References

[1] J. Luo, X. Deng, H. Zhang and H. Qi, "Ultra-low latency service provision in edge computing", *Proc. IEEE Int. Conf. Commun. (ICC)*, pp. 1-6, May 2018.

[2] Z. Ning, P. Dong, X. Wang, S. Wang, X. Hu, S. Guo, et al., "Distributed and dynamic service placement in pervasive edge computing networks", *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1277-1292, Jun. 2021.

[3] C. Li et al., "Dynamic offloading for multiuser muti-CAP MEC networks: A deep reinforcement learning approach", *IEEE Trans. Veh. Technol.*, vol. 70, no. 3, pp. 2922-2927, Mar. 2021.

[4] P. Dai, K. Hu, X. Wu, H. Xing and Z. Yu, "Asynchronous deep reinforcement learning for data-driven task offloading in MEC-empowered vehicular networks", *Proc. IEEE INFOCOM Conf. Comput. Commun.*, pp. 1-10, 2021.

[5] J. Wang and L. Wang, "Mobile edge computing task distribution and offloading algorithm based on deep reinforcement learning in Internet of Vehicles", *J. Ambient Intell. Humanized Comput.*.

[6] Y. Liu, H. Yu, S. Xie and Y. Zhang, "Deep reinforcement lear'ning for offloading and resource allocation in vehicle edge computing and networks", *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11158-11168, Nov. 2019.

[7] X. Li, "A computing offloading resource allocation scheme using deep reinforcement learning in mobile edge computing systems," *J. Grid Comput.*, vol. 19, no. 3, pp. 1–12, 2021.

[8] C. You and K. Huang, "Multiuser resource allocation for mobile-edge computation offloading", *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, pp. 1-6, 2016.

[9] L. Huang, X. Feng, L. Zhang, L. Qian and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks", *Sensors*, vol. 19, no. 6, 2019.

[10] J. Yu, L. Lingyun, and L. Xiang, "Edge-cloud collaborative task offloading mechanism based on ddqn in vehicular networks," *Comp. Eng.*, vol. 48, no. 12, pp. 156–164, 2022.

[11] K. Wang, X. Wang, X. Liu and A. Jolfaei, "Task offloading strategy based on reinforcement learning computing in edge computing architecture of Internet of Vehicles", *IEEE Access*, vol. 8, pp. 173779-173789, 2020.

[12] J. Li, H. Gao, T. Lv and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC", *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, pp. 1-6, Apr. 2018.