Deep Reinforcement Learning for Shared Offloading Strategy in Vehicle Edge Computing

Xin Peng, Zhengke Han, Wenwu Xie, Chao Yu, Peng Zhu, Jian Xiao, and Jinxia Yang

***Abstract*—Vehicular edge computing (VEC) effectively reduces the computing load of vehicles by offloading computing tasks from vehicle terminals to edge servers. However, offloading of tasks increase in quantity the transmission time and energy of the network. In order to reduce the computing load of edge servers and improve the system response, a shared offloading strategy based on deep reinforcement learning is proposed for the complex computing environment of Internet of Vehicles (IoVs). The shared offloading strategy exploits the commonality of vehicles task requests, similar computing tasks coming from different vehicles can share the computing results of former task submitted. The shared offloading strategy can be adapted to the complex scenarios of the IoVs. Each vehicle can share the offloading conditions of the VEC servers, and then adaptively select three computing modes: local execution, task offloading, and shared offloading. In this article, the network state and offloading strategy space are the input of the deep reinforcement learning (DRL). Through the DRL, each task unit selects the offloading strategy with the optimal energy consumption at each time period in the dynamic IoVs transmission and computing environment. Compared with the existing proposals and DRL-based algorithms, it can effectively reduce the delay and energy consumption required for tasks offloading.**

***Index Terms*—Deep reinforcement learning (DRL), Internet of Vehicles (IoVs), task shared offloading, vehicular edge computing (VEC).**

I. Introduction

INTERNET of Vehicles (IoVs) have undertaken a wealth of application services, including autonomous driving, path planning, collision avoidance, and in-vehicle entertainment [1]. Applications of IoVs are usually computing and communication intensive tasks, requiring large computing and energy consumption, and it is difficult to rely on vehicles for localized processing. With the continuous popularization of 5G networks, a large number of servers are deployed on the edge side closed to network users, providing effective computing power support for IoVs. By offloading the computing tasks of vehicles to edge server for execution, vehicular edge computing (VEC) can effectively reduce the computing load of vehicles, and then reduces the response delay of the applications and extends the battery life of the vehicles. VEC servers are often deployed in network access points, such as base stations (BSs) and road side units (RSUs) in order to improve the flexibility of the VEC network [2]. RSU has been widely deployed along major streets and equipped with edge servers to provide extensive network access coverage [3].

Although the task offloading strategy of edge computing can effectively expand the computing power of vehicles, a large number of vehicles need to offload tasks to edge servers for execution that will cause network congestion, increase task response delay, and system energy consumption [4]. Therefore, choosing the efficient task offloading strategy of BS or RSU is a key issue for VEC applications [5]. Existing works mainly focus on current states of tasks and edge servers. Migration decisions are made based on task execution parameters, computing resources of edge servers [6], [7]. There is relatively little literature to integrate the global and time-varying information of the network for optimal decision making [8], [9]. The problems of global optimization are generally nonconvex functions, and global optimization cannot be effectively performed without satisfying convex optimization conditions [10], [11]. Globalized information is usually difficult to obtain for the distributed architecture of the IoVs [12]. The emergence of machine learning makes the realization of global optimization possible [13].

Reinforcement learning is an important branch of machine learning. Recently, there have been many literatures applying it to task offloading optimization of IoVs [14], [15]. Deep reinforcement learning (DRL) is a combination of deep learning and reinforcement learning. Its core idea is to express the decision-making process as a Markov decision process (MDP). DRL can also combine perception and decision-making capabilities to provide solutions for complex systems [16], [17].

Although the edge server has to process a large number of task computing requests from vehicles, many computing requests from different vehicles are actually similar computing tasks. If similar computing tasks can share computing results without repeated computing for each task, it will save a lot of computing cost and improve the response speed of the system. Based on this consideration, this article proposes a shared task offloading strategy for VEC. The successfully matched task unit does not need to be recomputed by the VEC server, and the VEC server directly retrieves the corresponding results from its memory and returns it to the user. This strategy does not need to execute the computation for each computing tasks, the offloading efficiency will be improved.

Manuscript received 1 January 2022; revised 17 May 2022; accepted 3 July 2022. Date of publication 26 July 2022; date of current version 8 June 2023. This work was supported in part by the Natural Science Foundation of China under Grant 61772195 and in part by the Natural Science Foundation of Hunan Province under Grant 2018JJ2156. *(Corresponding author: Xin Peng.)*

The authors are with the School of Information Science and Engineering, Hunan Institute of Science and Technology, Yueyang 414015, China (e-mail: peng-xin@foxmail.com; 805161770@qq.com; gavinxie@hnist.edu.cn; ycyuchao_001@hotmail.com; zhupeng@hnist.edu.cn; victor98@foxmail.com; yangjinxia@vip.hnist.edu.cn).

Digital Object Identifier 10.1109/JSYST.2022.3190926

The proposed strategy can construct the effective offloading scheme in the dynamic environment of IoVs without the prior information of the network model. Because each application service of IoVs is composed of subprograms with independent functions, we divide each computing task into relatively independent task units. Different from the overall task offloading, after the task is divided into task units, the decision space for task offloading will be larger. In the specific implementation process, a gradient strategy is used to solve the possible overfitting problem of DRL.

The contributions of this article are as follows.

1) We propose a shared offload strategy, in which, tasks are decomposed into relatively independent task units, and similar task units share the computing results in the task offloading process of VEC. Thus, the edge computing cost is reduced and the edge service response efficiency is improved.
2) A task shared offloading optimization model is established, with communication and computing cost as parameters. Based on the model, DRL method is used to maximize the system utility and minimize the energy consumption in the task offloading decision space.
3) We implemented the shared offloading strategy on Python. The simulation results under different system parameters are analyzed. Numerical results verify the advantages of shared offloading strategy in terms of energy consumption and delay optimization.

The structure of this article is as follows. In Section II, we discuss the related works. Section III proposes system model. Section IV introduces the shared offloading strategy. Section V gives the DRL solution of the shared offloading strategy. Section VI evaluates the performance of the algorithm through simulation. Finally, Section VII concludes this article.

II. Related Work

We summarize the existing works about edge computing task offloading for energetic optimization in this section.

Appropriate resource allocation of VEC can promote the data transmission rate, which can satisfy the requirements of latency-sensitive service deadline constraints [18]. Luo *et al.* [19] use graph theory and greedy algorithm to optimize the delay constraints in the process of vehicle data prefetching and data distribution. Hu *et al.* [10] used minority games to reduce the delay of task offloading. From the perspective of power saving and energy consumption, Zhang *et al.* [20] analyzed the occupied channel priority of MEC computing tasks, and then optimized the channel allocation resources for all computing tasks based on the priority.

During the task offloading process, the communication and computation processes consume energy and also produce time delay [21]–[23]. Joint optimization of time delay and energy consumption in computing, storage, and communication has become an important research direction. In the case of a given task offloading scheduling decision, Mao *et al.* [9] use convex optimization techniques to determine the optimal transmit power allocation. Tran *et al.* [24] utilize convex and quasi-convex optimization techniques to jointly optimize task offloading decision, mobile user uplink transmission power, and computing resource allocation at the MEC servers.

Recently, machine learning algorithm has been widely used in edge offloading and caching because of its powerful data processing and decision-making ability. Among machine learning algorithms, DRL has significant advantages in accuracy and efficiency compared with traditional methods. Some literatures apply DRL to the optimization of energy consumption and delay of task offloading [25], [26]. There are also literatures that take maximizing system utility as the optimization goal [15]–[17]. Zhan *et al.* [15] maximize the utility of multiple users through game theory and DRL. Chen *et al.* [16] proposed a resource allocation scheme that can dynamically coordinate computing and communication resources and improve the total return of network operators. Hu *et al.* [17] studied the problems of the joint communication, caching and computing problem, and established a DRL multitimescale model and its mobility-aware reward estimation method. Ning *et al.* [27] developed an intelligent flow control scheme by studying DRL, which can increase the system profit of mobile network operators and effectively allocate network resources.

In another works, the task offloading model is established with the joint constraints of communication resources and computing power [28]–[30]. For the multiple-access edge server, Maurice *et al.* [28] proposed a work that solves the problem of offloading and subcarrier allocation in the MEC system through DRL, which greatly improves the computation speed of the multiple-access edge computing system. Liu *et al.* [29] studied the problem of offloading and resource allocation in VEC. Taking into account the delay of computing tasks, a DRL-based vehicle-assisted offloading scheme was proposed and maximize the long-term utility of the network. In terms of system utility and offloading reliability, Zhang *et al.* [30] considered the choice of target server and the determination of data transmission methods, and proposed an optimal offloading strategy based on DRL.

Different from constructing optimal task offloading and transfer schemes based on the binary decision mechanism of task migration, some works have studied multiple offloading modes [31], [32]. For the computation offload scheduling problem, Zhan *et al.* [33] study the scheduling position and scheduling time of each task, trades off task delay and energy consumption through DRL, and minimizes long-term costs. DRL algorithms suffer from signaling overhead and computational complexity. Huang *et al.* [34] use multiple agents to share model parameters and minimize system energy consumption in edge computing scenarios while reducing the signaling overhead and computational complexity of the DRL algorithm.

We can find that the existing solutions use DRL on a limited decision space, and cannot give full play to the advantages in system perception and decision. The overfitting problem of DRL is not considered, resulting in the efficiency of task offloading and transmission decision-making reduced. We decompose the tasks into relatively independent task units, and let similar tasks

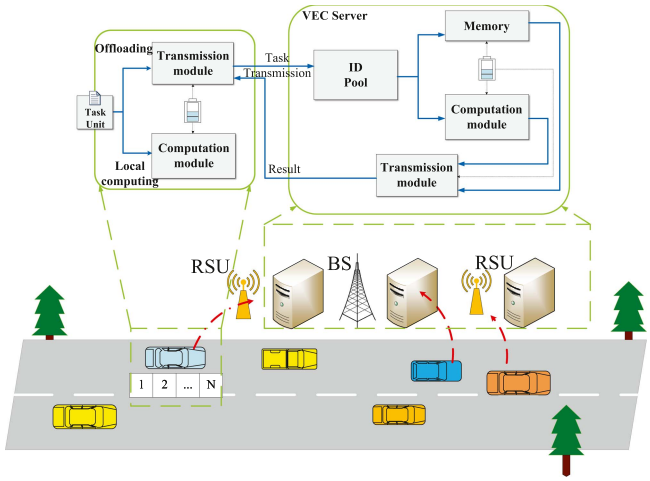


Fig. 1. Task offloading in IoVs.

share the computing results. Thus, the decision space of task offloading is effectively expanded, which is conducive to obtaining the optimal task offloading decision based on DRL.

III. System Model

Fig. 1 shows a fifth-generation (5G) network supported IoVs consisting of many vehicles equipped with dual wireless interface. RSUs provide services for the vehicles in its coverage. Each macro BS have wide area coverage. In the scenario, macro BS forms a wide area cell. Each RSU divides the highway into a microcell, which contain multiple vehicles. BS and RSU are equipped with VEC servers that provide computing service for vehicle users. All vehicles within the coverage of BS or RSU can be served by the VEC server of corresponding BS or RSU. The input of each VEC server is the offloading task coming from its served vehicles, and the output is the computed results of vehicles offloaded tasks.

When the vehicle enters the service range of the VEC, it is assumed that the length of the service segment of the BS currently serving is L_1, the length of the service segment of the RSU is L_2, and the average speed of the vehicle is v. Then, the time when the vehicle receives service at the VEC server of BS and RSU is $t_b^r = \frac{L_1}{v}$ and $t_r^r = \frac{L_2}{v}$, respectively. Vehicle applications are made up of subprogram modules. Therefore, we decompose vehicle application services into relatively independent task units according to the functions of the modules. Task unit is the basic component of vehicle computing task. Different vehicle computing tasks may have multiple identical task units. In the procedure of task offloading computation, the task units are cached in the VEC server first, and then the first-come, first-served strategy is adopted. The execution of the task unit will occupy the computing and communication resources of the VEC server. The roads and vehicles in the figure are covered by two different types of access networks that is cellular and DSRC networks. Cellular networks provided by BS and DSRC networks provided by RSU. The cellular network and DSRC operate on different and nonoverlapping spectrums. Compared with BS with seamless coverage and high data transmission cost, RSU provides uneven coverage and free access services. Vehicle users need to pay for using the cellular network and VEC servers. The vehicles can offload tasks to the VEC server through V2B and V2R. V2B is suitable for the scenario where the vehicles cannot obtain the network service provided by RSU.

This article assumes that a task is decomposed into N task units, and each task unit i is assigned an ID. The ID of ith task unit includes six fields that are the size d_i of the task unit, the calculation amount c_i required by the task unit, the task execution time limit $t_i^{\max}$, task function ζ_i, task unit generation time t_i^g, and period of validity Δt_i of task computation result. Task execution time limit $t_i^{\max}$ indicates that the task execution result must be returned within $t_i^{\max}$ time, otherwise the task fails. Task function ζ_i is determined by the functional attributes of the task unit itself. Task unit that has the same ζ_i field is with the same function. For the task unit generated by the vehicles, generation time t_i^g is the generation time of the task unit. While for the task unit cached by the VEC server, this field is the time when the task unit enters the VEC server ID pool. Effective duration of calculation result Δt_i represents the effective time of the computation result of the task unit.

When the task unit is offloaded to the VEC server for execution, its ID and computation results will be saved in the ID pool. VEC server ID pool has limited space. When the ID pool is full, the new arrived task unit ID will overwrite the previous data. The coverage mechanism adopts the principle of least recent use. Task IDs that have not been matched for a long time are also less likely to be used in the future. When new task units come in, these task units will be overwritten first. When a task unit need to offload to an VEC server, the task unit ID will be matched with the ID in ID pool of the VEC server. If the match is successful, the VEC server will directly return the cached results to the vehicle. This shared offloading strategy greatly reduces the energy consumption of the VEC servers and improves the efficiency of offloading.

For ease of reference, the main variates used in this article are summarized in Table I.

IV. Shared Offloading Strategy: Analysis and Problem Formulation

In this section, we first elaborate the matching scheme of task shared offloading. Then, we establish a task offloading optimization model.

A. Analysis of Matching Scheme of Task Shared Offloading

The proposed offloading strategy takes energy consumption, execution delay, and computation cost as the main optimization goal. Both local computing and offloading of tasks will bring energy consumption and delay. The task offloading decision first needs to make a balancing between local computing and offloading computing. When the vehicular computation capacity is insufficient, the task units can be offloaded to the VEC servers of the RSU or BS for computation through wireless communication links. Whether the task unit offloaded to the VEC server is computed directly or share the former cached result depends on the matching result of the task unit ID. Before

TABLE I
MAIN VARIABLES

VARIATE	DESCRIPTION
d_i	The data size of task unit i
c_i	The required CPU cycles of task unit i
r_i^B	The transmission rate of vehicle to BS
p_i^B	The power of data transmission between vehicle and BS
g_i^B	Channel gain between vehicle and BS
σ^2	Background noise power
r_i^R	The transmission rate of task unit i communicating with RSU
p_i^R	Data transmission power between vehicle and RSU
g_i^R	Channel gain between vehicle and RSU
e_i^B	The energy consumption of the VEC server on the BS for computing task unit i
f_B	The frequency of the CPU cycle of the VEC server in the BS
e_i^R	The energy consumption of the VEC server on the RSU for computing task unit i
f_R	The CPU frequency of the VEC server in the RSU
f_L	The local CPU frequency of the vehicle
t_i^B	Transmission delay of communication between task unit i and BS
t_i^R	Transmission delay of communication between task unit i and RSU
t_i^{max}	The maximum time limit of task unit i

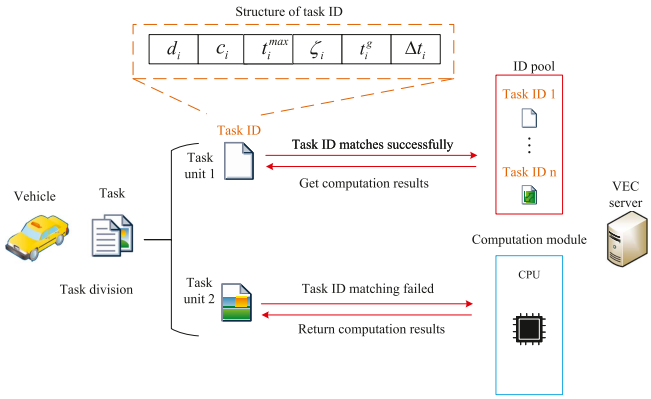


Fig. 2. Shared offloading strategy.

the task is transmitted to the VEC server, it has been divided into independent task units. Then, the shared offloading process consists of two steps: task matching and task execution.

1) Task Matching: Fig. 2 shows the task unit ID structure and task unit offloading computation procedure. The task generated by a vehicle is decomposed into relatively independent task units. Then, each decomposed task unit is given a task ID according to its attributes. The structure of the task ID contains

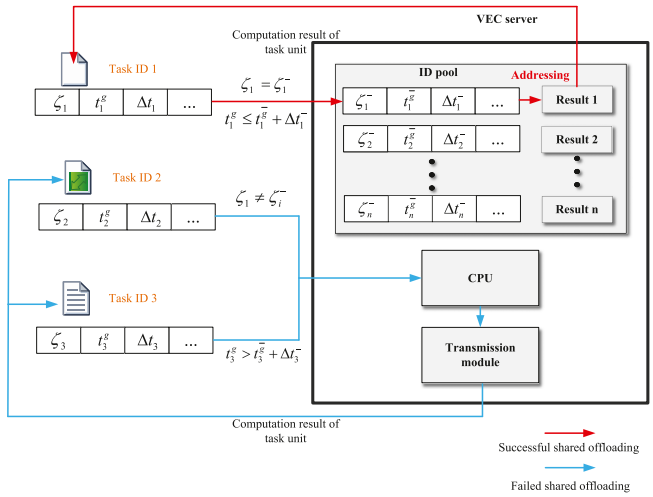


Fig. 3. Process of task unit in shared offload strategy.

the six fields mentioned earlier. In task ID, only the task function ζ_i, task unit generation time t_i^g, and period of validity Δt_i are used for task ID matching. Task units with the same task ID function field are similar task units. Only similar task units can match successfully.

During task unit matching, it is necessary to compare task ID function field ζ_i first. Then, it is compared that the difference between the generation time t_i^g of the new arrived task unit i and the generation time t_j^g of the task unit j cached in ID pool of VEC server. If the generation time t_i^g of new arrived task unit i is within the validity period of cached task unit j of VEC server, that is $t_i^g < t_j^g + \Delta t_j$, then the matching between task unit i and j is successful. The VEC server can directly fetch the cached result of task unit j as the result of task unit i and return it to the vehicle. Similarly, if the generation time t_i^g of new arrived task unit i is not within the validity period of cached task unit j of VEC server, that is $t_i^g \ge t_j^g + \Delta t_j$, then the matching between task unit i and j is failed. At this point, the VEC server will have to execute the task unit i, and return the result to the vehicle.

2) Task Execution: The task unit ID is first transmitted to the VEC server through vehicle to BS (V2B) or vehicle to RSU (V2R) links. Then, the task unit ID matches the task ID cached in the VEC server ID pool. If the task unit ID is successfully matched, the VEC server fetches the results of matched task units from the ID pool and returns it to the vehicle. For the task unit whose ID matching fails, the computation result is returned to the vehicle after the task unit is computed by the CPU of the VEC server.

The execution procedure is shown in Fig. 3. Before task offloading, the function field of task ID 1 matches the task function field of VEC server ID pool cached successfully ($\zeta_1 = \zeta_1^-$), and the generation time of task ID 1 is within the effective time of VEC server ID pool cache task ($t_1^g < \bar{t}_1^g + \Delta t_1^-$), then task ID 1 matches successfully. However, although function field of task ID 3 matched successfully, its generation time are out of effective time of cached task in VEC server ID pool. Therefore, task ID 3 match failed.

The length of validity period Δt_j depends on the task function. Give an example, if the cached task unit j is the weather at a certain time, its validity period may be 12 h. However, if the cached task unit j is the traffic condition of a road section at a certain time, its validity period may not exceed 1 h. Because the BS and RSU have limited coverage, the vehicles cannot offload the task sometimes. If without BS and RSU, vehicle task units can only be computed locally. In this case, if the vehicle does not have sufficient computing power, the computation of the task cannot be completed. Different computing locations have different computing power and computing cost. The required computing power and maximum time limit of each task unit impacts the computing location of the task unit.

B. Problem Formulation

Task offloading energy consumption consists of task transmission and task computing energy consumption. Each task unit is indivisible, and it can only have one computation mode in period *t*. We use $I_*^{i,t}|_{*=b,bs,r,rs,l}$ as the offload mode flag of task unit i at time t, and $I_*^{i,t} = \{0, 1\}$. Specifically, $I_b^{i,t} = 1$ represents that the task unit i is offloaded to the VEC server of BS. $I_{bs}^{i,t} = 1$ indicates that the task unit i is executed the shared offloading strategy on the VEC server of the BS. $I_r^{i,t}$ represents that the task units i is offloaded to the VEC server of RSU. $I_{rs}^{i,t} = 1$ indicates that the task unit i is executed the shared offloading strategy on the VEC server of the RSU. $I_l^{i,t} = 1$ represents that the task unit i is computed locally.

Because each task unit can only have one computation mode in period *t*, we have

$$I_b^{i,t} + I_{bs}^{i,t} + I_r^{i,t} + I_{rs}^{i,t} + I_l^{i,t} = 1. \tag{1}$$

The local computing time t_i^L of task unit i can be expressed as

$$t_i^L = \frac{c_i}{f_L}. \tag{2}$$

The local execution energy consumption e_i^L of the task unit i is given by [35]

$$e_i^L = \omega_L d_i c_i f_L^2 \tag{3}$$

where ω_L is a coefficient of chip structure of vehicle.

If the vehicle migrates their tasks to VEC server on the BS or RSU for execution, we also need to consider the transmission overhead of the task. It is assumed that the vehicle has a fixed transmission power. In the case of the V2B communication mode, the spectrum of the vehicles under the cellular networks are orthogonal, and the channel of the V2B communication does not collide with each other. For this reason, according to Shannon's formula, the transmission rate between the vehicles and the BS VEC server is given by

$$r_i^B = W \log_2 \left(1 + \frac{p_i^B g_i^B}{\sigma^2}\right) \tag{4}$$

where W is the channel bandwidth. The vehicle uses DSRC communication in V2R mode. DSRC has K service channels, which adopt competitive access mode. When the number of users is less than K, each channel does not interfere with each other. When the number of vehicles is greater than K, channel collision will occur, and the extra user transmission behavior will cause interference. The number of other vehicles communicating with the RSU in the current service area is M. The simultaneous interference power ξ_i^R is defined as

$$\xi_i^R = \begin{cases} \sum_{k=K+1}^{M} \rho \cdot \mathrm{g}_k^R \cdot p_k^R, M \geq K+1 \\ 0 \quad , M < K+1 \end{cases} \tag{5}$$

where ρ is the average interference coefficient generated by excessive users, usually $\rho = 1 - \frac{K}{M}$. When the number of vehicles exceeds the number of available channels, ξ_i^R represents the noise generated by extra vehicles' communicating with the RSU. In this case, the transmission rates of each vehicle to the VEC server of RSU is expressed as

$$r_i^R = W \log_2 \left(1 + \frac{p_i^R g_i^R}{\sigma^2 + \xi_i^R}\right). \tag{6}$$

If the task ID matches successfully in the VEC server pool, it could be in shared offloading mode. At this time, only the task unit ID needs to be transmitted to the VEC server. However, when the task ID matching is unsuccessful, the complete task unit needs to be migrated to the VEC server for offloading computation. In V2B and V2R transmission mode, the time taken for the complete task unit to be transmitted to the BS or RSU is expressed as $t_i^{B,tr}$ and $t_i^{R,tr}$, respectively. As shown in the following:

$$t_i^{B,tr} = \frac{\delta_i + I_b^{i,t} \cdot d_i}{r_{i,k}^B} \quad \text{and} \quad t_i^{R,tr} = \frac{\delta_i + I_r^{i,t} \cdot d_i}{r_{i,k}^R} \tag{7}$$

where δ_i is the ID size of task unit i.

The computation delay of the task unit in the VEC server of the BS and RSU is expressed as

$$t_i^{B,co} = \frac{C_i}{f_B} \quad \text{and} \quad t_i^{R,co} = \frac{C_i}{f_R}. \tag{8}$$

Then, the energy consumption $e_i^B$and e_i^R can be expressed as

$$e_i^B = p_i^B \cdot t_i^{B,tr} + I_b^{i,t} \omega_B c_i f_B^2 \tag{9}$$

$$e_i^R = p_i^R \cdot t_i^{R,tr} + I_r^{i,t} \omega_R c_i f_R^2 \tag{10}$$

where $p_i^B \cdot t_i^{B,tr}$ and $p_i^R \cdot t_i^{R,tr}$ are the transmission energy consumption to the VEC server. $\omega_B c_i f_B^2$ and $\omega_R c_i f_R^2$ are the computing energy consumption on the VEC server. ω_B and ω_R are the coefficient of chip structure of BS and RSU VEC server, respectively.

According to the aforementioned analysis, different task offloading modes and task computing locations have different energy consumption. For task unit i, its overall energy consumption can be expressed as

$$E_{i,t} = I_b^t e_i^B + I_r^t e_i^R + I_l^t e_i^L + I_{bs}^{i,t} e_i^B + I_{rs}^{i,t} e_i^R. \tag{11}$$

Due to the limited network transmission capacity, the task unit needs to wait for a period of time before it is migrated to the VEC server. Let t_i^{ready} denotes the waiting time for the task

unit i to be transmitted to the VEC server. It can be given by

$$t_i^{\text{ready}} = \begin{cases} \sum_{\text{k}=1}^{i-1} \begin{pmatrix} (I_b^{k,t}(t_k^{B,tr}+t_k^{B,co}) + I_b^{k,t}(t_k^{R,tr}+t_k^{R,co}) \\ +I_{bs}^{k,t}t_k^{B,tr} + I_{rs}^{k,t}t_k^{R,tr}) \end{pmatrix}, i>1 \\ 0, i=1 \end{cases} \tag{12}$$

t_i^{ready} includes the transmission and computation delay of previous task units. The total duration of each task unit in offloading execution is expressed as

$$\Phi(i) = I_b^{i,t}(\max\{t_i^{\text{ready}}, t_{i-1}^{B,co}\} + t_i^{B,co}) + I_r^{i,t}(\max\{t_i^{\text{ready}}, t_{i-1}^{R,co}\} + t_i^{R,co}) + I_l^{i,t}t_i^L + I_{bs}^{i,t}t_i^{B,tr} + I_{rs}^{i,t}t_i^{R,tr} \tag{13}$$

where $t_{i-1}^{B,co}$ represents the time for task unit $i-1$ offloaded to the BS, and $t_{i-1}^{R,co}$ represents the time for task unit $i-1$ offloaded to the RSU.

The optimal task shared offloading problem take energy consumption, time delay, and cost as the optimization objectives. Therefore, we define the utility function $R_i(t)$ of task unit i, which include three parts, which are the remaining duration time of the task unit, the computing energy consumption of the task unit, and the cost of offloading services. The longer the remaining time of the task unit, the greater the utility function. When the computation delay exceeds the maximum delay $t_i^{\max}$ of the task unit, there is a timeout penalty. The cost of offloading services includes the transmission fee from the task unit to the BS and the computing fee on the VEC server. The RSU communication is free. The service fee normalized value L_i is defined as (14) shown at the bottom of this page, where υ_B is the transmission billing rate of the BS, and υ_C is the computing billing rate of the VEC server. $R_i(t)$ is given in Definition 1.

Definition 1: Utility function $R_i(t)$. The utility function is the normalized weighted sum of the remaining duration time, the offloading energy consumption, and the offloading service cost during the offloading process of the task unit i. The utility function is expressed as

$$R_i(t) = \alpha_1 \left(\frac{t_i^{\max} - \phi^{i,t}}{t_i^{\max}} \mu(t_i^{\max} - \Phi(i)) - \lambda_i \mu(\Phi(i) - t_i^{\max}) \right) - \alpha_2 \cdot (k_1)^{E_i} - \alpha_3 \cdot (k_2)^{L_i} \tag{15}$$

where α_1, α_2, and α_3 are the weight coefficients, and $\alpha_1 + \alpha_2 + \alpha_3 = 1$. k_1 and k_2 are constants close to 1. λ_i is an arbitrary constant. $\mu(\cdot)$is a step function.

Based on the utility function $R_i(t)$, we define the overall utility value of all task units offloading as a reward function.

Definition 2: Reward function *U*(*t*). The reward function is the overall utility value generated by offloading all task units, which is used to evaluate the current task unit offloading scheme. The reward function is expressed as

$$U(t) = \sum_{i \in N} R_i(t). \tag{16}$$

Each task unit has many different offloading strategies. The proposed algorithm needs to find the optimal offloading strategy for the task unit. The definition of task unit offloading strategy space is given here.

Definition 3: Strategy space. The strategy space of the task offloading algorithm is the collection of all possible offloading modes of the task unit i in the period t, expressed as

$$\Pi_t = \{(I_b^{i,t}, I_{bs}^{i,t}, I_r^{i,t}, I_{rs}^{i,t}, I_l^{i,t}) | i \in N, I_x^{i,t} \in \{0,1\}\}, \quad x \in \{b, bs, r, rs, l\}.$$

We transform the optimization objective into maximizing the reward function of the task unit in period t. The optimal task offloading model is formulated as follows:

$$\begin{aligned} &\max_{\Pi_t} U(t) = \sum_{i \in N} R_i(t) \\ &\text{s.t. } C1: \frac{\left(e^{\frac{d_i \cdot \ln 2}{W \cdot t_i^{\max}}} - 1\right)(\xi_i^R + \sigma^2)}{g_i^R} \le p_i^R \\ &\quad C2: \frac{\left(e^{\frac{d_i \cdot \ln 2}{W \cdot t_i^{\max}}} - 1\right)\sigma^2}{g_i^B} \le p_i^B \\ &\quad C3: I_b^{i,t} + I_{bs}^{i,t} + I_r^{i,t} + I_{rs}^{i,t} + I_l^{i,t} = 1 \\ &\quad C4: I_{bs}^{i,t} = 0, if(\varsigma_i \in IDStack_{BS}) \cup (\varsigma_i \in IDStack_{BS}, |t_i^g - t_j^g| \le \Delta t_j) \\ &\quad C5: I_{rs}^{i,t} = 0, \text{if}(\varsigma_i \in IDStack_{RSU}) \cup (\varsigma_i \in IDStack_{RSU}, |t_i^g - t_j^g| \le \Delta t_j) \\ &\quad C6: \Phi(i) \le t_b^r \text{and} \Phi(i) \le t_r^r \end{aligned} \tag{17}$$

where *C*1 and *C*2 represent the lowest transmission power of vehicles for RSU and BS, respectively. *C*3 represents each task unit makes a unique offloading decision at the period because of its indivisibility. *C*4 represents that if the task unit *i* matches failed on VEC server of BS, it cannot use the shared offloading mode on the BS server. *C*5 represents that if the task unit i matches failed on VEC server of RSU, it cannot use the shared offloading mode on the RSU server. *C*6 represents that the total duration of the offloading execution should be within the receiving time of the VEC server service of the BS and RSU.

For the tenability of *C*1 and *C*2, we give the following lemmas.

Lemma 1: If the task unit is offloaded to the BS for execution, the transmission power P_i^B of the vehicle satisfies $\frac{(e^{\frac{d_i \cdot \ln 2}{W \cdot t_i^{\max}}} - 1)\sigma^2}{g_i^B} \le p_i^B$.

Proof: The vehicle needs to maintain sufficient transmitting power to successfully transmit the task to the BS for execution. The transmission time of task unit i shall meet the maximum

$$L_i = \frac{(I_b^{i,t} + I_{bs}^{i,t})\upsilon_B \cdot (\delta_i + I_b^{i,t} \cdot d_i) + (I_b^{i,t} + I_{bs}^{i,t} + I_r^{i,t} + I_{rs}^{i,t})\upsilon_C \cdot c_i}{\upsilon_B \cdot (\delta_i + I_b^{i,t} \cdot d_i) + \upsilon_C \cdot c_i} \tag{14}$$

deadline of the task, which is shown as

$$\frac{d_i}{r_i^B} \le t_i^{\max}. \tag{18}$$

Substitute formula (4) into the aforementioned formula, there is

$$\frac{d_i}{W\log_2\left(1+\frac{p_i^B g_i^B}{\sigma^2}\right)} \le t_i^{\max}. \tag{19}$$

Then, using the base changing formula, we have the following:

$$\frac{d_i}{W\cdot t_i^{\max}} \le \frac{\ln\left(1+\frac{p_i^B g_i^B}{\sigma^2}\right)}{\ln 2}. \tag{20}$$

By using logarithmic relation and shifting terms, we can derive

$$e^{\frac{d_i\cdot\ln 2}{W\cdot t_i^{\max}}} \le 1+\frac{p_i^B g_i^B}{\sigma^2}. \tag{21}$$

Keep the power p_i^B on the right of the inequality, we have

$$\frac{\left(e^{\frac{d_i\cdot\ln 2}{W\cdot t_i^{\max}}}-1\right)\sigma^2}{g_i^B} \le p_i^B. \tag{22}$$

Lemma 1 is proved.

Similarly, we can prove Lemma 2, and the proof process will not be repeated.

Lemma 2: If the task unit is offloaded to the RSU for execution, the transmission power p_i^R of the vehicle satisfies

$$\frac{\left(e^{\frac{d_i\cdot\ln 2}{W\cdot t_i^{\max}}}-1\right)\left(\xi_i^R+\sigma^2\right)}{g_i^R} \le p_i^R.$$

V. DRL-Based Solution

In this section, we first give the initialization method of the optimization model, and then give the DRL offloading algorithm based on optimization model.

A. Optimization Model Initialization

Before model optimization, the task unit offloading mode needs to be initialized. The initialization process follows the following principles. When the local computing power cannot meet the maximum deadline requirement of the task unit, i.e., $t_i^L > t_i^{\max}$, the offloading mode is selected according to the size of the task unit and the maximum deadline in the task unit. If the task unit is only offloaded to the VEC server of the BS through V2B communication, the computation delay $t_i^{B,co}$ and the transmission delay $t_i^{B,tr}$ of the task unit satisfy the maximum deadline $t_i^{\max}$of task unit, i.e., $t_i^{B,tr}+t_i^{B,co} < t_i^{\max}$ and $t_i^{R,tr}+t_i^{R,co} > t_i^{\max}$, and the task unit does not match successfully on the BS server, the offload mode indicator initializes $I_b^{i,t}=1$. If the task unit matches successfully on the BS server, the offloading strategy indicator initializes $I_{bs}^{i,t}=1$. Similarly, if the task unit is offloaded to the VEC server of the RSU through V2R communication, the computation delay $t_i^{R,co}$ and the transmission delay $t_i^{R,tr}$ of the task unit satisfy the maximum deadline $t_i^{\max}$of the task unit, i.e., $t_i^{R,tr}+t_i^{R,co} < t_i^{\max}$, and the task unit does not match successfully on the RSU server, the offload mode indicator initializes $I_r^{i,t}=1$. In this case, if the task unit matches successfully, the offloading mode indicator initializes $I_{rs}^{i,t}=1$. If the local computation of the vehicle can meet the deadline $t_i^{\max}$ of the task unit, i.e., $t_i^{L,co} < t_i^{\max}$, or the network of BS and RSU is unavailable, the local computing mode will be the only choice, and the offloading mode indicator initializes $I_l^{i,t}=1$. On the basis of the aforementioned analysis, the offloading optimization model initialization process is shown in Algorithm 1.

Algorithm 1: Strategy Initialization.

```
Initialization
For each task unit i ∈ N Do {
If local computing power is enough and t_i^L ≤ t_i^max;
  The task unit will be computed in local and I_l^{i,t} = 1;
Else If t_i^{B,tr} + t_i^{B,co} < t_i^max and
t_i^{R,tr} + t_i^{R,co} > t_i^max;
    If Task ID is successfully matched in BS;
      Task unit is in shared offloading mode, and
      I_bs^{i,t} = 1;
    Else Task unit is computed in VEC server of BS
    and I_b^{i,t} = 1;
  Else If t_i^{B,tr} + t_i^{B,co} > t_i^max and
  t_i^{R,tr} + t_i^{R,co} < t_i^max
    If Task ID is successfully matched in RSU;
      Task unit is in shared offloading mode, and
      I_rs^{i,t} = 1;
      Else Task unit is computed in VEC server of
      RSU and I_r^{i,t} = 1;
    Else Select an offloading strategy randomly;}
End
```

B. DRL-Based Task Offloading Solution

By maximizing the optimization objective of the model (17), the optimal offloading strategy $\pi_t^* \in \Pi_t$ of the task unit in period t can be obtained. Then, the overall optimal offloading strategy P^* in the task offloading process can be expressed as

$$P^* = \arg\max_{p\in\{(\pi_1,\pi_2,\ldots,\pi_T)|\pi_t\in\Pi_t,t\in T\}} E\left(\sum\nolimits_{t\in T}\gamma^{t-1}U(t)\right) \tag{23}$$

where γ is discount factor and satisfies $0<\gamma<1$. T is the time period set.$E(\cdot)$ is mathematical expectation. γ is used to indicate the impact of future reward on current offloading mode.

The selection of the overall optimal offloading strategy P^* depends on the current network channel, the historical caching of the task unit in the VEC server, the computing power of the VEC server, and the reward function $U(t)$. Therefore, we need to define the state space and value function for DRL algorithm.

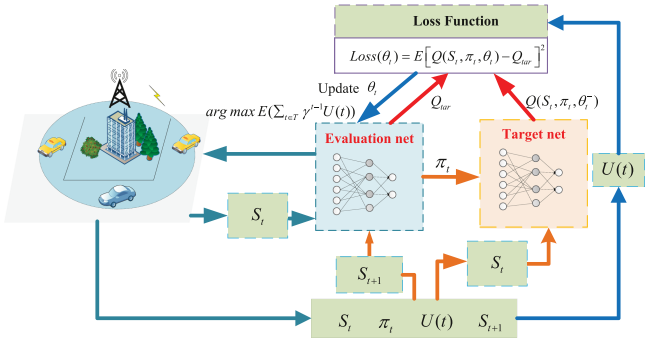


Fig. 4. Process of DRL.

Definition 4: The state space of the DRL-based task offloading method is $S_t = [r_i^B, r_i^R, f_B, f_R, f_L, t_i^{\text{ready}}, ID_i, IDStack_{BS}, IDStack_{RSU}]$. $IDStack_{BS}$ and $IDStack_{RSU}$ are ID pools of BS server and RSU server, respectively. ID_i indicates the complete ID of task unit i.

Definition 5: Value function. The task offloading process is equivalent to MDP, so the value function is defined as long-term expectation of task offloading reward function based on state space S_t and task unit offloading strategy π_t

$$Q(S_t,\pi_t) = E\left(\sum_{t\in T}\gamma^{t-1}U(t)\right). \tag{24}$$

Fig. 4 shows the process of our DRL algorithm in VEC task offloading. The agent-based DRL determines the optimal offloading strategy according to the current state and reward. The algorithm is deployed on the VEC server side. We adopt the double DQN method in the algorithm. Double DQN includes two convolution neural networks, namely target net and evaluation net. The evaluation net is used to decide strategy according to $Q(S_t,\pi_t,\theta_t)$. The target net is used to evaluate the strategy decided by evaluation net according to $Q(S_t,\pi_t,\theta_{t-1})$. Loss function is used to update the parameters θ_t of evaluation net. This makes the difference between the target network and the evaluation network gradually decreases. We use the random gradient descent of loss function to update the parameters, so that the network gradually converges. Meanwhile, the value of loss function is converged to the lowest value through iteration, and the generalization of the algorithm is improved. The buffer is used to temporarily store the state parameters, delay the update of the neural network, and cut off the correlation of the network parameters, which is expressed as buffer $= [S_t, \pi_t, U(t), S_{t+1}]$.

The offloading strategy adopted by each task in the period t depends on the current and future states of the VEC system. Therefore, we formulate (23) as a Markov decision method. Meanwhile, the algorithm finds the optimal task offloading strategy of each task unit in the time series by updating the value function. The value function update process uses the time difference method. According to Definition 5, we can get

$$Q(S_t,\pi_t) = E\left(U(t) + \gamma Q(S_{t+1},\pi_{t+1})\right). \tag{25}$$

Therefore, the optimal value of the value function can be expressed as

$$Q^*(S_t,\pi_t) = E\left(U(t) + \gamma \max_{t+1} Q^*(S_{t+1},\pi_{t+1})\right). \tag{26}$$

According to Q-learning algorithm, the update process of the value function is expressed as follows:

$$Q(S_t,\pi_t) \leftarrow Q(S_t,\pi_t) + \beta\left(U(t) + \gamma \max_{\pi_{t+1}} Q^*(S_{t+1},\pi_{t+1}) - Q(S_t,\pi_t)\right) \tag{27}$$

where β is learning rate. Optimal value of the value function can be obtained by maximizing the function with parameter γ.

Double DQN method is used for the aforementioned formula. Then, the optimal strategy P^* can be expressed as

$$P^* = \arg \max_{p\in\{(\pi_1,\pi_2,\ldots,\pi_T)|\pi_t\in\Pi_t, t\in T\}} Q(S_t,\pi_t,\theta_t). \tag{28}$$

The target net needs to compute Q_{tar} value, which can be expressed as

$$Q_{\text{tar}} = U(t) + \gamma Q(S_{t+1}, \arg\max_{\pi_{t+1}} Q^*(S_{t+1},\pi_{t+1},\theta_t), \theta_{t-1}). \tag{29}$$

Let the loss function $\text{Loss}(\theta_t)$ be the difference between the target network and the evaluation network, which can be expressed as

$$\text{Loss}(\theta_t) = E\left[Q(S_t,\pi_t,\theta_t) - Q_{\text{tar}}^t\right]^2. \tag{30}$$

We use the gradient descent method to correct θ_t. The gradient $\nabla_{\theta_t}\text{Loss}(\theta_t)$ is obtained by

$$\nabla_{\theta_t}\text{Loss}(\theta_t) = 2E[\nabla_{\theta_t} Q(S_t,\pi_t,\theta_t)(Q(S_t,\pi_t,\theta_t) - Q_{\text{tar}}^t)]. \tag{31}$$

Then, update θ_t according to the following formula:

$$\theta_t \leftarrow \theta_t - \varpi \nabla_{\theta_t}\text{Loss}(\theta_t) \tag{32}$$

where ϖ is the scalar· size.

Follow the aforementioned procedure, the strategies in the strategy space are selected according to the state space, so that the algorithm is optimized toward the optimization objective. In this way, each task unit has an optimal offloading strategy, which consumes less energy, less delay, and less cost. That is when the task unit satisfies the multiple offloading strategies, the proposed offloading algorithm will guide it to find the strategy with the least energy and delay consumption.

VI. Numerical Results

To illustrate the performance of the proposed shared offloading algorithm, we consider an area of 120 km^2. Each cell has 1 BS and 0–3 RSUs. Each cell has 1–3 vehicles performing computing tasks in each period. Other evaluation parameters are listed in Table II.

We compared the proposed DRL-based shared offloading algorithm with other algorithms, including Q-learning-based offloading, greedy offloading, unshared offloading, offloading only, and local only. Greedy offloading algorithm executes the task unit locally first, and offloading is only carried out when the

TABLE II
EXPERIMENT PARAMETERS

VARIABLE	DESCRIPTION
The frequency corresponding to the CPU cycle of the VEC server in BS f_B	100GHZ
The frequency corresponding to the CPU cycle of the VEC server in RSU f_R	20GHZ
The frequency corresponding to the CPU cycle of the vehicle user f_L	0.5GHZ
Background noise σ^2	-100dBm
Speed of the vehicle v	20 meter/sec
The maximum delay of task unit t_i^{max}	10ms
The size of each task d_i	5-10 bites
Step size ϖ	0.01
Activation function	Relu
Buffer size	500
Batch	32
Discount factor γ	0.9
Number of rounds episode	200
The coverage length of the RSU	500m
The coverage length of the BS	1000m

Algorithm 2: DRL for Task Offloading.

1. Initialization by Algorithm 1;
2. **While** ($Loss(\theta_t)>\eta$)
3. { **For** $t = 1, 2, \ldots, T$ **Do**
4. {Execute strategy π_t and compute reward $U(t)$;
5. Update $S_{t+1} \leftarrow S_t$;
6. Store the experience $(S_t, \pi_t, U(t), S_{t+1})$ into the experience replay buffer;
7. Get a batch of M samples $(S_t, \pi_t, U(t), S_{t+1})$ from the replay memory;
8. Compute the Q-value by $Q^*(S_t, \pi_t) = E(U(t) + \gamma \max_{t+1} Q^*(S_{t+1}, \pi_{t+1}))$;
9. Update the main deep Q-network by minimizing $Loss(\theta_t) = E[Q(S_t, \pi_t, \theta_t) - Q_{tar}]^2$;
10. Update the target deep Q-network parameters with $\theta_t \leftarrow \theta_t - \varpi \nabla_{\theta_t} Loss(\theta_t)$;
11. Update the: choose strategy π_t with maximum Q function;}
12. Return P^* by $P^* = arg \max_{p \in \{(\pi_1, \pi_2, \ldots, \pi_T) | \pi_t \in \Pi_t, t \in T\}} E(\sum_{t \in T} \gamma^{t-1} U(t))$;}

local execution cannot meet the deadline. Unshared offloading is DRL-based offloading method, but without computing result sharing mechanism. Offloading only method offload all task units to VEC server. Local only method executes all task units on vehicular computing platform.

In the shared offloading algorithm, the learning rate of the neural network model is 0.1. The exploration degree is 0.9, which means that 90% of probability choose the best strategy, and 10% choose random strategy, allowing the algorithm to explore all the possibilities in the environment. There are two

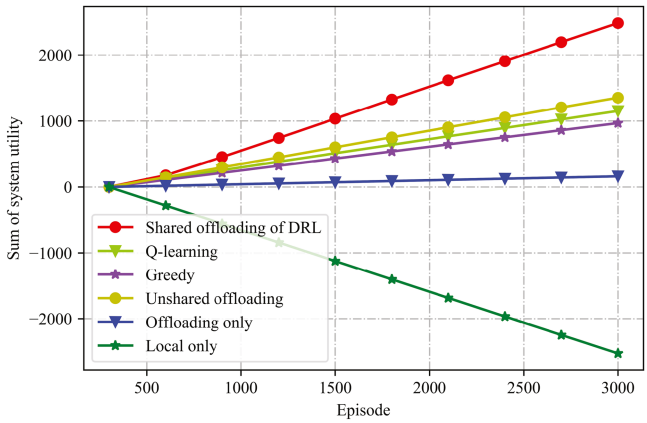


Fig. 5. Accumulation of reward value.

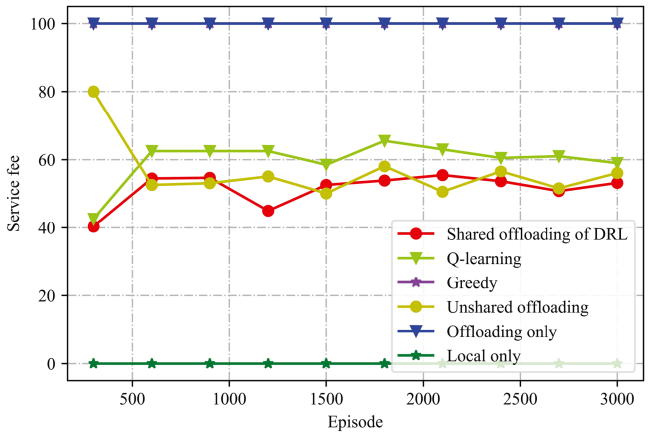


Fig. 6. Service fee of different algorithms.

unrelated networks in the shared offloading algorithm, each with two layers of neural networks. Each layer is a fully connected layer of 20 neurons and uses the Relu activation function. When the network selects a strategy, the environment will feed back the energy consumption and the remaining delay of the strategy to the agent, and then update the network in turn. The buffer size is 500 and the batch size is 32. In Q-learning, the learning rate is 0.01, and the discount coefficient is 0.9. The proposed algorithm and Q-learning both have 3000 episodes.

Fig. 5 shows that with the accumulation of period, the reward of shared offloading algorithm gradually reaches the maximum, and finally the reward value is obviously higher than that based on Q-learning and greedy algorithm. The nonintelligence of the greedy algorithm makes the processing of time delay and energy consumption poor in each period, which is a penalty for overtime. The shared offloading algorithm calls a deep neural network, which can more intelligently consider the task offloading strategy and improve the overall energy benefits of the system.

Fig. 6 shows the service charges under different strategies. Because the local computing strategy does not consume the offloading cost of the BS, the service fee is zero. However, a large number of task units fail to execute due to timeout. The service fee for offloading all tasks to the VEC server is

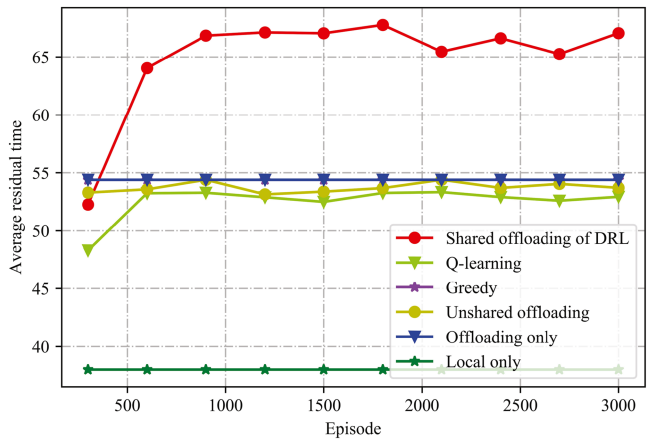


Fig. 7. Accumulation of residual time.

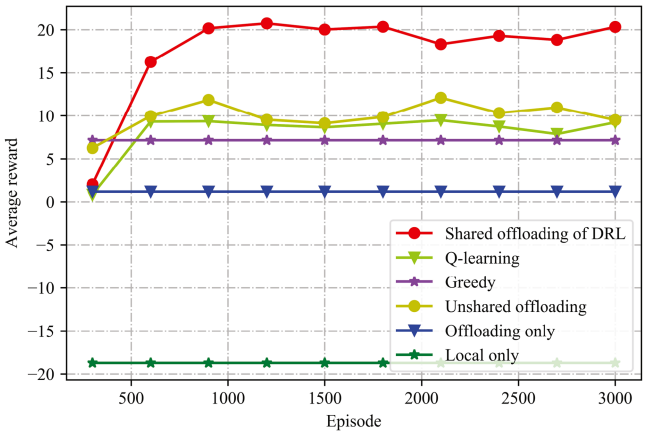


Fig. 9. Average reward performance of different DRL algorithms.

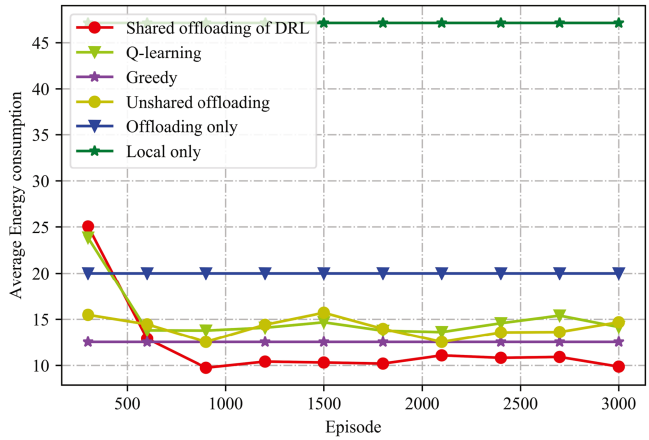


Fig. 8. Energy consumption of different algorithm.

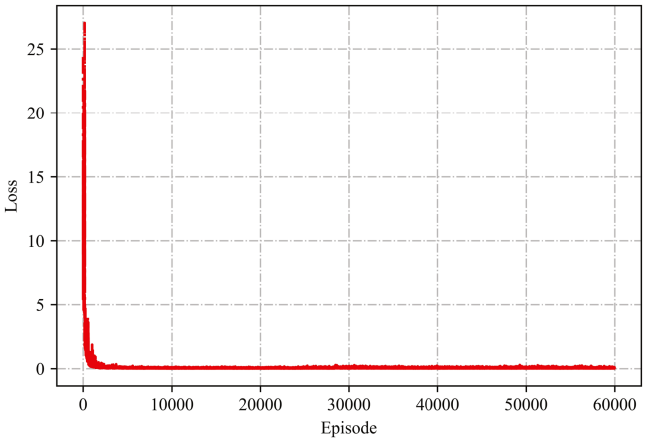


Fig. 10. Relationship between the value function and the number of training steps.

the highest. Greedy offloading algorithm aims to complete the task calculation within the time limit and does not consider the service fee generated during the offloading process, so the service fee is relatively high. The service fee of our algorithm is lower than that of other algorithms except local only algorithm, because the shared offloading process only generates the service fee of transmission and does not generate the service fee of computation.

Fig. 7 shows the results of the average residual time of task units in each algorithm. After 3000 episodes, the residual time of task units in our algorithm is obviously more than other algorithm. This is because shared offloading mechanism effectively reduce the computing time and transmission time of task units. Due to limited computing power, local only algorithm has least residual time. Offloading only algorithm needs to migrate all task to VEC server, this will cause network congestion, and further reduce the remaining time of task unit. Q-learning can make a comprehensive decision on task offloading, and its performance is close to our algorithm.

Fig. 8 shows the comparison of the energy consumption of each algorithm. After 3000 episodes, the energy consumption generated by the proposed algorithm in this article is better than the comparison algorithms. This is because the shared offloading algorithm reduces the amount of data transmission and task computation. Since there is no data transmission in the local only computing mode, the structural characteristics of its local CPU lead to high energy consumption when computing task units. The offloading only algorithm requires a large number of task transmission, so the energy consumption is relatively high. Q-learning and unshared offloading algorithms take energy consumption as one of the optimization objectives, and also have relatively low energy consumption.

Fig. 9 shows that the reward value obtained by our algorithm is significantly better than other algorithms. This shows that our algorithm has made significant improvement in the comprehensive optimization of energy consumption, delay, and service cost. The reward value of Q-learning and unshared offloading algorithms are second only to that of our algorithm. Due to the learning mechanism, the reward values of the three algorithms have a gradual increase process. Local only and offloading only algorithms have the lowest reward value due to the lack of flexible decision-making mechanism.

Fig. 10 shows the convergence of our algorithm. It is not difficult to see from the figure that after 400 episodes of iterative

training, the value function of the algorithm in this article tends to be stable. This is because the algorithm adopts the gradient descent method for the loss function, which effectively improves the convergence performance. In addition, the rapid convergence also improves the adaptability of the algorithm.

VII. Conclusion

In this article, we have proposed a task offloading algorithm for 5G supported VEC by integrating DRL. To minimize the offloading overhead of similar tasks, we propose a shared offloading strategy for VEC. In the strategy, similar computing tasks coming from different vehicles can share the computing results of former task submitted to VEC server. We establish the overall utility function of integrating computing energy, transmission energy, task remaining time, and service cost, and establish the objective function and a task offloading optimization model on this basis. We also propose an optimization scheme with joint task offloading computing mode decision and VEC server selection in a DRL approach. The numerical results illustrate that the DRL-inspired task shared offloading scheme significantly outperforms the comparison schemes.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2322–2358, Oct.–Dec. 2017.

[2] C. Zhang, X. Lin, R. Lu, and P. Ho, "RAISE: An efficient RSU-aided message authentication scheme in vehicular communication networks," in *Proc. IEEE Int. Conf. Commun.*, 2008, pp. 1451–1457.

[3] 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 Netw.,* vol. 33, no. 5, pp. 198–205, Sep./Oct. 2019.

[4] Z. Xiao et al., "Vehicular task offloading via heat-aware MEC cooperation using game-theoretic method," *IEEE Internet Things J.,* vol. 7, no. 3, pp. 2038–2052, Mar. 2020.

[5] X. Jiang, F. R. Yu, T. Song, and V. C. M. Leung, "Intelligent resource allocation for video analytics in blockchain-enabled internet of autonomous vehicles with edge computing," *IEEE Internet Things J.,* to be published, doi: 10.1109/JIOT.2020.3026354.

[6] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.,* vol. 66, no. 5, pp. 810–819, May 2017.

[7] T. G. Rodrigues, K. Suto, H. Nishiyama, N. Kato, and K. Temma, "Cloudlets activation scheme for scalable mobile edge computing with transmission power control and virtual machine migration," *IEEE Trans. Comput.,* vol. 67, no. 9, pp. 1287–1300, Sep. 2018.

[8] Y. Sun et al., "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.,* vol. 68, no. 4, pp. 3061–3074, Apr. 2019.

[9] Y. Mao, J. Zhang, and K. B. Letaief, "Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2017, pp. 1–6.

[10] M. Hu, Z. Xie, D. Wu, Y. Zhou, X. Chen, and L. Xiao, "Heterogeneous edge offloading with incomplete information: A minority game approach," *IEEE Trans. Parallel Distrib. Syst.,* vol. 31, no. 9, pp. 2139–2154, Sep. 2020.

[11] H. Peng, Q. Ye, and X. Shen, "Spectrum management for multi-access edge computing in autonomous vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 7, pp. 3001–3012, Jul. 2020.

[12] Y. Pan, M. Chen, Z. Yang, N. Huang, and M. Shikh-Bahaei, "Energy-efficient NOMA-based mobile edge computing offloading," *IEEE Commun. Lett.,* vol. 23, no. 2, pp. 310–313, Feb. 2019.

[13] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[14] Z. Li, C. Wang, and C.-J. Jiang, "User association for load balancing in vehicular networks: An online reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.,* vol. 18, no. 8, pp. 2217–2228, Aug. 2017.

[15] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Trans. Comput.,* vol. 69, no. 6, pp. 883–893, Jun. 2020.

[16] M. Chen, T. Wang, K. Ota, M. Dong, and A. Liu, "Intelligent resource allocation management for vehicles network: An A3C learning approach," *Comput. Commun.*, vol. 151, pp. 485–494, Feb. 2020.

[17] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.,* vol. 67, no. 11, pp. 10190–10203, Nov. 2018.

[18] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li, "Collaborative learning of communication routes in edge-enabled multi-access vehicular environment," *IEEE Trans. Cogn. Commun. Netw.,* vol. 6, no. 4, pp. 1155–1165, Dec. 2020.

[19] G. Luo, Q. Yuan, H. Zhou, N. Cheng, Z. Liu, and F. Yang, "Cooperative vehicular content distribution in edge computing assisted 5G-VANET," *China Commun.,* vol. 15, no. 7, pp. 1–17, Jul. 2018.

[20] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[21] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mobile Comput.,* vol. 21, no. 2, pp. 598–611, Feb. 2022.

[22] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.,* vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[23] X. Chen, C. Wu, Z. Liu, N. Zhang, and Y. Ji, "Computation offloading in beyond 5G networks: A distributed learning framework and applications," *IEEE Wireless Commun.,* vol. 28, no. 2, pp. 56–62, Apr. 2021.

[24] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. Veh. Technol.,* vol. 68, no. 1, pp. 856–868, Jan. 2019.

[25] Z. Chen and X. Wang, "Decentralized computation offloading for multi-user mobile edge computing: A deep reinforcement learning approach," in *Proc. EURASIP J. Wireless Commun. Netw.*, 2020, Art. no. 188.

[26] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, pp. 54074–54084, 2020.

[27] Z. Ning, R. Kowk, K. Zhang, X. Wang, and B. Sadoun, "Joint computing and caching in 5G-envisioned internet of vehicles: A deep reinforcement learning-based traffic control system," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5201–5212, Aug. 2021.

[28] M. Nduwayezu, Q. Pham, and W. Hwang, "Online computation offloading in NOMA-based multi-access edge computing: A deep reinforcement learning approach," *IEEE Access*, vol. 8, pp. 99098–99109, 2020.

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

[30] K. Zhang, Y. Zhu, S. Leng, Y. He, S. Maharjan, and Y. Zhang, "Deep learning empowered task offloading for mobile edge computing in urban informatics," *IEEE Internet Things J.,* vol. 6, no. 5, pp. 7635–7647, Oct. 2019.

[31] F. Jiang, X. Zhu, and C. Sun, "Double DQN based computing offloading scheme for fog radio access networks," in *Proc. IEEE/CIC Int. Conf. Commun. China*, 2021, pp. 1131–1136.

[32] Z. Yu, Y. Tang, L. Zhang, and H. Zeng, "Deep reinforcement learning based computing offloading decision and task scheduling in internet of vehicles," in *Proc. IEEE/CIC Int. Conf. Commun. China*, 2021, pp. 1166–1171.

[33] W. Zhan et al., "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[34] X. Huang, S. Leng, S. Maharjan, and Y. Zhang, "Multi-agent deep reinforcement learning for computation offloading and interference coordination in small cell networks," *Trans. Veh. Technol.,* vol. 70, no. 9, pp. 9282–9293, Sep. 2021.

[35] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "DRL-based V2V computation offloading for blockchain-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, to be published, doi: 10.1109/TMC.2022.3153346.

Xin Peng received the B.S. degree in communication engineering and the M.S. and Ph.D. degrees in computer science from Hunan University, Changsha, China, in 2003, 2008, and 2011, respectively.

He was a Visiting Researcher with Auburn University, Auburn, AL, USA, in 2014. He is currently a Professor with the School of Information Science and Engineering, Hunan Institute of Science and Technology, Yueyang, China. His main research interests include Internet of Things, CPS, and cloud computing.

Zhengke Han received the M.S. degree in information and communication engineering from the Hunan Institute of Technology, Hengyang, China, in 2022.

His major research interests include vehicle edge computing and deep reinforcement learning.

Wenwu Xie received the B.S., M.S., and Ph.D. degrees in communication engineering from Huazhong Normal University, Wuhan, China, in 2004, 2007, and 2017, respectively.

He is currently an Associate Professor with the School of Information Science and Engineering, Hunan Institute of Science and Technology, Yueyang, China. His research interests include communication algorithm, such as channel estimation, equalizer, encoding/decoding, etc.

Chao Yu received the Ph.D. degree in communication engineering with Hoseo University, Cheonan, South Korea.

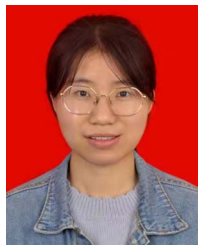
He is currently an Assistant Professor with the School of Information Science and Engineering, Hunan Institute of Science and Technology, Yueyang, China. His research interests include UAV-aided communications, intelligent reflecting surface, physical layer security, wireless power transfer, and mobile edge computing.

Peng Zhu received the B.S. and Ph.D. degrees in space physics from Wuhan University, Wuhan, China.

He is currently an Associate Professor with the School of Information Science and Engineering, Hunan Institute of Science and Technology, Yueyang, China. His major research interests include signal processing and communication techniques.

Jian Xiao received the M.S. degree in information and communication engineering from the Hunan Institute of Technology, Hengyang, China, in 2022.

His major research interests include reconfigurable intelligent surface and machine learning.



Jinxia Yang received the M.S. degree in information and communication engineering from the Hunan Institute of Technology, Hengyang, China, in 2022.

Her major research interests include reconfigurable intelligent surface and convex optimization problem.