Deep Reinforcement Learning-Guided Task Reverse Offloading in Vehicular Edge Computing

Anqi Gu*, Huaming Wu*, Huijun Tang* and Chaogang Tang†
*Center for Applied Mathematics, Tianjin University, Tianjin 300072, China
†School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China
Emails: 3095896998@qq.com, {whming, tanghuijune}@tju.edu.cn, cgtang@cumt.edu.cn

GLOBECOM 2022 - 2022 IEEE Global Communications Conference | 978-1-6654-3540-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/GLOBECOM48099.2022.10001474

***Abstract*—The rapid development of Vehicular Edge Computing (VEC) provides great support for Collaborative Vehicle Infrastructure System (CVIS) and promotes the safety of autonomous driving. In CVIS, crowd-sensing data will be uploaded to the VEC server to fuse the data and generate tasks. However, when there are too many vehicles, it brings huge challenges for VEC to make proper decisions according to the information from vehicles and roadside infrastructure. In this paper, a reverse offloading framework is constructed, which comprehensively considers the relationship balance between task completion delay and the energy consumption of User Vehicle (UV). Furthermore, in order to minimize the overall system consumption, we establish an adaptive optimal reverse offloading strategy based on Deep Q-Network (DQN). Simulation results demonstrate that the proposed algorithm can effectively reduce the energy consumption and task delay, when compared with the full local and fixed offloading schemes.**

***Index Terms*—Internet of Vehicle, Vehicular Edge Computing, Reverse Offloading, Deep Reinforcement Learning**

I. Introduction

In recent years, autonomous driving has emerged as a promising vehicle technology. With the vigorous development of Vehicular Edge Computing (VEC), its application in autonomous driving has attracted widespread attention. Internet of Vehicles (IoV) can improve traffic efficiency, ensure vehicle safety and reduce energy consumption [1]. Unfortunately, the safety of autonomous driving remains a concern, which is severely affected by many factors, e.g., complex road conditions, communication delays, and limitations of computational capacity [2]. The ES8 self-driving accident in 2021 showed that keeping the vehicle itself safe is still worthy of in-depth investigation, due to the fact that the area the vehicle can perceive is limited, not to mention the scarcity of on-board computing resources.

Collaborative Vehicle Infrastructure System (CVIS) has the ability to capture the real-time status of vehicles and road conditions and deliver corresponding message [3]. CVIS has been applied to receiving real-time messages and fusing them together for making decisions in a short period of time. Unlike existing works that offload tasks at the edge of Internet of Things (IoT), these tasks are created by the VEC rather than the vehicles. This poses a new challenge, due to the location of the server, and RSU can only receive the wireless signal within the valid area, a single RSU fails to meet the strict constraints of task latency, while using many servers would be prohibitively expensive. It is also impractical to draw support from the cloud, as it may bring about long delays and information security issues. Considering the close proximity between the vehicle and the VEC, and the remaining computational capacity of the vehicle terminal itself can be used to reduce the latency. It is necessary to reversely offload tasks from the VEC server to the vehicle terminal for execution. Making accurate offloading decisions in real-time and allocating computing resources reasonably have great significance to improve the service performance of VEC and improve the Quality of Service (QoS).

978-1-6654-3540-6/22$31.00 © 2022 IEEE

There are many studies focused on developing novel offloading-decision strategies in vehicular networks. Wang *et al.* [4] formulated the application offloading process as a Markov Decision Process (MDP) problem and proposed a Site-by-Site and Task-by-Task (SSTT) heuristic approach, where mobile devices can offload multiple tasks to adjacent vehicle-based Cloudlets for the reduction of energy consumption. Feng *et al.* [5], [6] proposed a greedy-based reverse offloading framework for IoV. The vehicle uploads sensor data to Road Side Units (RSUs) for information fusion, and then the tasks generated by the VEC can either be executed on the RSU or be offloaded in reverse to vehicles, where appropriate decisions are made based on the vehicle information fusion, thereby improving the safety of the vehicle. However, these heuristic offloading schemes still struggle to balance complexity and optimality, which are prone to fall into local extrema when the scale of the optimization problem is large [7]. In addition, only the system latency is considered, while the energy consumption at the VEC is ignored [6], which is equally important for User Vehicles (UVs).

Recently, Deep Reinforcement Learning (DRL) has been widely applied in IoV for object detection and scene perception [8]. Owing to the combination of the powerful perception ability of deep learning and the exploratory ability of reinforcement learning, DRL can learn proper dynamic decisions by interacting with the environment and obtain feedback and rewards without knowing the prior knowledge of the environment. Huang *et al.* [9] proposed a DRL-based approach to address the joint task offloading and resource allocation problem. DREAM [10] is a queuing delay-aware task offloading algorithm based on DRL for maximization of the throughput of User Vehicle (UV) in a collaborative vehicle network under the long-term queueing delay constraint. Nevertheless, current DRL-based offloading schemes have seldom exploited the reverse offloading decision in IoV to overcome the uncertainties in VEC environments.

Inspired by the above research, in this paper, we adopt DRL in the task reverse offloading framework in the vehicle network scenario, considering both vehicular energy consumption and task delay. The main contributions of this paper can be summarized as follows:

- Aiming at the scenario of a single VEC server with multiple vehicles in the VEC environment, an optimization problem of reverse unloading decision is established. In each time slot, the VEC decides whether to reversely offload the task information of i^{th} vehicle back to the vehicle, and the system consumption minimization problem is formulated as a mixed integer nonlinear programming problem.
- We design an adaptive optimal reverse offloading strategy based on Deep Q-Network (DQN) to minimize the system consumption of UVs. To the best of our knowledge, this is the first work that introduces DRL into task reverse offloading while simultaneously considering both latency and energy consumption.
- We conduct extensive experiments to evaluate the performance of the proposed algorithm. Simulation results demonstrate that our scheme can always achieve the lowest system consumption compared to other schemes. The reverse offloading decision can be effectively optimized, so as to minimize the average consumption of the IoV system.

II. System Model and Problem Formulation

A. System Model

Since both data transmission and task execution consume the energy of the vehicle, while the battery capacity of the vehicle is limited, this paper attempts to choose appropriate tasks which are generated at the edge server, and offload them from the RSU to the vehicle for execution by making optimal offloading decisions with the goal of minimizing energy consumption and delay.

As depicted in Fig. 1, the VEC environment is composed of M vehicles and one RSU that can connect to the VEC and compute tasks through the edge server within it. M vehicles can be represented by a set $\mathcal{M} = \{1, 2, \cdots, M\}$. It is assumed that all vehicles have access to the VEC, and that both the vehicle and the VEC can compute, receive, and transmit signals.

We assume that all vehicles have perceptual data $i \in M$. The i^{th} vehicle data is represented by $D_i = \{l_i^p, l_i^o, c_i\}$, where l_i^p and l_i^o represent the size of input data and the size of output data, respectively, and c_i is the computational workload. Regarding the collection of information, the sensory data of all vehicles is uploaded to the VEC for data fusion. The reverse offloading is defined as the process of returning the sensory information data and tasks from the VEC to the original

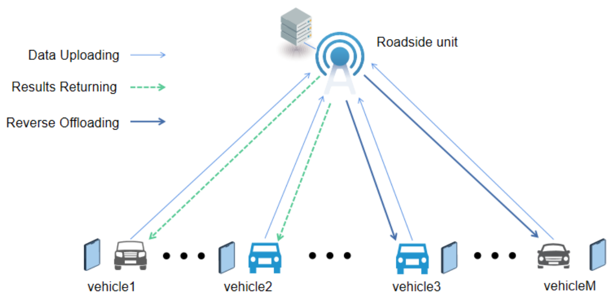


Fig. 1. System Model

vehicle for calculation. RSU performs data fusion and then generates a large number of computational tasks. With the powerful capabilities of edge servers, these tasks can either be performed on the VEC server or be reversely offloaded to the original vehicle.

The decision variable x_i can be 0 or 1, when $x_i = 1$, the task of the i^{th} vehicle $task_i$ is processed on the VEC itself rather than reversely offloaded back to the vehicle, otherwise the task should be processed by the vehicle. The main notations used in this paper are summarized in Table I.

TABLE I
Notations and Their Meanings

Notation	Definition
M	The number of UVs
g_i^u	The uplink channel gain
g_i^d	The downlink channel gain
r_i^u	The upload rate
r_i^d	The download rate
δ	The path loss exponent
B_i	The bandwidth of the i^{th} vehicle
σ^2	The background noise power
p_i	The transmit power of the i^{th} vehicle
p^{RSU}	The transmit power of RSU
f_i	The CPU frequency of the vehicle
f_i^o	The CPU frequency of VEC server allocated to the i^{th} task
μ	The capacitance coeffificient
$p_i^{\max}$	The maximum transmit power of the i^{th} vehicle
E_i^{tran}	The transmission energy
E_i^{loc}	The local computation energy
T_i^u	The uploading latency
T_i^{ro}	The reverse offloading latency
T_i^l	The local computation latency
T_i^o	The VEC computation latency
T_i^d	The result download latency
x_i	The reverse offloading decision
d_i	The distance between the RSU and the i^{th} vehicle
ϕ	The loss function
γ	The discount factor
ω	The network parameter
ψ'	The learning rate for network updating

B. Edge Computing

Once a task is uploaded to the RSU, the wireless device passes its related information to the RSU through the wireless channel. In general, considering the small amount of feedback information and the VEC server can supply power continuously without being limited by energy consumption, we ignore the energy consumption when the edge server sends the calculation results back to the wireless device [11].

The total delay for the i^{th} task calculated by VEC can be written as:

$$T_i^R = T_i^u + T_i^o + T_i^d, \tag{1}$$

where $T_i^u = l_i^p / r_i^u$, $T_i^o = c_i l_i^p / f_i^o$ and $T_i^d = l_i^o / r_i^d$ are the uploading latency, the VEC computation latency in the RSU and the result download latency, respectively. f_i^o is the CPU frequency of the VEC server allocated to the task of the i^{th} vehicle $task_i$. r_i^d and r_i^u are the download rate and the upload rate, respectively, which can be calculated by [6]:

$$r_i^d = B_i \log_2\Big(1 + \frac{p^{\text{RSU}} g_i^d {d_i}^{-\delta}}{\sigma^2}\Big), \tag{2}$$

$$r_i^u = B_i \log_2\Big(1 + \frac{p_i g_i^u {d_i}^{-\delta}}{\sigma^2}\Big), \tag{3}$$

where p^{RSU} and p_i are the transmit power of the RSU and the i^{th} vehicle, respectively, g_i^u and g_i^d denote the uplink and downlink channel gains between the RSU and the i^{th} vehicle, respectively, d_i represents the distance between the i^{th} vehicle and the RSU, and δ represents the pass loss exponent.

In the meantime, the energy consumption of vehicles only includes the energy spent during the transmission, i.e.

$$E_i^{tran} = p_i \cdot T_i^u. \tag{4}$$

C. Local Computing with RSU Reverse offloading

When it is decided that the task should be reversely offloaded, the output data l_i^o of the RSU will be transmitted back to the i^{th} vehicle to deal with, where $x = 0$ indicates that the RSU reversely offloads the task of the i^{th} vehicle $task_i$ to the i^{th} vehicle, and then the vehicle executes its task.

The delay of the reverse offloading task can be represented as follows:

$$T_i^{R-off} = T_i^u + T_i^{ro} + T_i^l, \tag{5}$$

where T_i^{ro} and T_i^l are the reverse offloading latency and the local computation latency, respectively, which can be formulated as:

$$T_i^{ro} = \frac{\beta_i l_i^p}{r_i^d}, \tag{6}$$

$$T_i^l = \frac{c_i \beta_i l_i^p}{f_i}. \tag{7}$$

The local execution energy is calculated by [6]:

$$E_i^{loc} = c_i \beta_i l_i^p \mu f_i^2. \tag{8}$$

where μ is the capacitance coefficient, f_i is the CPU frequency of the vehicle. β_i means the overhead ratio of the reverse offloading of the i^{th} vehicle, and its value should be more than one [5].

The total delay of the task for each vehicle i can be represented as:

$$total\_lat_i = (1 - x_i) \cdot T_i^{o-off} + x_i \cdot T_i^o. \tag{9}$$

Similarly, the total energy consumption per vehicle i can be expressed as:

$$total\_power_i = E_i^{tran} + (1 - x_i) \cdot E_i^{loc}. \tag{10}$$

D. Problem Formulation

Our purpose is to minimize the total delay and corresponding energy consumption of all tasks in the whole procedure, we first introduce a system consumption function $L(\boldsymbol{x})$, which is defined as the product of energy consumption and task latency g^{th} power. The specific form is as follows:

$$L(\boldsymbol{x}) = \sum_{i=1}^{M} (total\_lat_i{}^g \times total\_power_i), \tag{11}$$

where $\boldsymbol{x} = \{x_i | i \in M\}$ is the offloading-decision vectors and g indicates the importance of the task latency to the energy consumption.

Then, we establish an optimization problem ($\boldsymbol{\mathcal{P}_1}$) to minimize $L(x)$ in the whole process by optimizing the reverse offloading decision, which is formulated as follows [6]:

$$\begin{aligned} (\boldsymbol{\mathcal{P}_1}): \quad & \min_{\boldsymbol{x}} L(\boldsymbol{x}) \\ \text{s.t.}: \quad & 0 \le f_i \le f_i^{\max}, & (12) \\ & 0 \le p_i \le p_i^{\max}, & (13) \\ & 0 \le f_i^R \le \frac{F_R^{\max}}{M}, & (14) \\ & x_i \in \{0, 1\}. & (15) \end{aligned}$$

where $F_R^{\max}$ is the maximum CPU cycle frequency of the VEC server, the first and third constraints represent the limitation of the computational resources, indicating that the CPU frequencies of all vehicles and the CPU frequencies of the VEC server assigned to the i^{th} vehicle cannot exceed the maximum frequency. The second constraint represents the restriction on the transmit power of the vehicles. The fourth constraint is the vehicle decision, which indicates that the binary variable $\boldsymbol{x}$ in the objective function and the constraint condition can only be 0 or 1.

It is found that $\boldsymbol{\mathcal{P}_1}$ is a non-convex problem, which is difficult to solve in polynomial time. With the increase in the number of vehicle terminals and the scale of computing tasks, the computational complexity of this problem also increases rapidly. Due to the curse of high-dimensional data, traditional heuristic optimization methods are inefficient and generally difficult to solve such complex problems [12].

III. DRL-Aided Reverse Offloading Approach

To conquer the above disadvantages, in this paper, we propose a DRL-based approximation algorithm to efficiently solve $\boldsymbol{\mathcal{P}_1}$.

We take the optimization of the vehicles on the road in a time slot T into consideration, and then divide T into many small slots for the reverse offloading decision. Therefore, the problem can be translated into the optimization problem of the long-term reward of the vehicle on the road, which can be further modeled as a Markov Decision Process (MDP). This paper proposes a DQN-based task offloading algorithm for VEC networks to obtain the reverse offloading policy $\boldsymbol{x}$. The state, action, the new state and the reward are stored in the memory and then sample memories in each training epoch of deep Q-learning algorithm to train the Q network. DQN is repeatedly trained through multiple iterative processes until the optimal offloading policy is achieved.

In our VEC environment, there are M vehicle terminals, and each of the terminal information can be transferred to a remote server or reversely offloaded to local computing. When the task information is updated to the server nearby or reverse offloaded to the local, the speed may change correspondingly, and the vehicle transmission power p_i and the vehicle CPU calculation frequency are also changing. Therefore, all the vehicles can be regarded as an agent. The elements of MDP, namely, state space, action space and reward function can be defined as follows.

A. State Space

Considering the influence of the distance between the vehicle and the server, task size, upload rate and download rate, we treat the vehicles transmit power, computational capacity, upload rate and download rate as states, which are severely affected by the status of the vehicle, and the distance between the vehicle and the RSU.

For the minimization of the overall energy consumption and latency, we use a DQN to find the optimal offloading decision x_i of the i^{th} server and program p_i, f_i, r_i^u and r_i^d of the user task into the system state s_t as an input to the DQN. Specifically, the state space s_t consists of the allocated CPU frequency of the VEC to the i^{th} vehicle, vehicle CPU frequency, upload and download rate, which can be represented as follows:

$$s_t = \Big\{p_1(t), p_2(t), \cdots, p_M(t), f_1(t), f_2(t), \cdots, f_M(t), r_1^u(t), r_2^u(t), \cdots, r_M^u(t), r_1^d(t), \cdots, r_M^d(t)\Big\}. \quad (16)$$

B. Action Space

The output of the DQN is the Q-value (in Eq. 11) of the corresponding action. When the agent selects the proper operation with the Q-value, the execution result of the action is to adjust the offloading decision accordingly.

We choose the decision-making variable $\boldsymbol{x}$ as the action, which can be represented as follows:

$$a_t = \{x_1(t), x_2(t), x_3(t), \cdots, x_M(t)\}. \quad (17)$$

C. Reward Function

We redefine the reward function in DQN, by taking the system consumption function as the goal of optimization. The reward reduces the computational complexity in the iterative process of the algorithm, accelerates the convergence of the algorithm, and ensures the lowest energy consumption of the system.

Since we want to minimize $L(\boldsymbol{x})$, the reward function of agents is defined as:

$$R_t = -L(\boldsymbol{x})/M, \quad (18)$$

which represents the average system consumption of the M vehicles.

Based on the analysis above, the details of the algorithmic process are illustrated in Algorithm 1.

Algorithm 1: DQN-based Task Reverse Offloading in Vehicular Edge Computing

```
Input: Network State $s_t$
Output: Reverse offloading decision $x_m$
1  Initialize the empirical pool storage space capacity and
     the parameters of the initial evaluation target $Q$ network
2  foreach episode = $1, \cdots, G$ do
3      Reset the environment, Initialize the environment
         state, rewards and losses
4      foreach time slot = $1, \cdots, T$ do
5          $a(t)$ is selected randomly with the probability of
             $\varepsilon$, otherwise select the action that maximizes the
             Q value, i.e., $a(t) = \arg\max_{a(t)} Q(s_t, a(t)\,|\theta)$
6          Execute $a(t)$, and offload the task to the selected
             server.
7          Calculated $r_t$ according to its definition and then
             transfer to the next state $s_{t+1}$
8          Store $(s_t, a_t, r_t, s_{t+1})$ in the experience pool
9          Memories were sampled uniformly randomly
             from the empirical pool
10         Compute the loss function by
             $\phi = r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
11         Update the parameters of the evaluation network
             using a back propagation algorithm
             $\omega = \omega + \psi' \nabla_\omega \phi^2$
12         The parameters of the evaluated network were
             replicated to the target network at every $n = 10$
             steps.
13         If $s_{t+1}$ is a termination state, the current iteration
             process is terminated
14     end
15 end
16 return Reverse offloading decision $x_m$;
```

IV. Performance Evaluation

A. Parameter Settings

We assume that there are 10 vehicle terminals and an RSU with a VEC. The relevant parameter values are set according to [6], [10]. We set the path loss index δ=3 and the background noise power of 70 dBm, $p_i^R = 0.5$ W and $p_i = 0.1$ W, capacitance coefficient is $\mu = 10^{-28}$, the available bandwidth between the i^{th} vehicle and the RSU is $B = 180$ kHz, weight coefficient $g = 0.5$ means more emphasis on energy saving, the distance between the i^{th} vehicle and the RSU $d_i(t)$ is decreases evenly over time. The channel gain of uplink and downlink between the vehicle and the VEC server was set as $127 + 30 \times \log d_i(t)$. The input data size, computing requirements, and reverse offloading overhead ratios follow $l_i^s \in [0.5, 1] \times 10^5$ bits, $c_i \in [500, 1000]$ cycle/bits, $\beta \in [1, 2]$. The output task data size was set to $l_i^o = 0.1 l_i^s$ bits, and the maximum CPU cycle frequency per vehicle and VEC server were $F^{\max} = 1.5 \times 10^8$ cycles/bits and $F_R^{\max} = 8 \times 10^8$ cycle/bits, respectively. The channel model is based on the Rayleigh fading model, assuming all vehicles have uniform straight motion and the path loss model is ${d_i}^{-\delta}$ [1].

B. Convergence Performance

The performance of system loss at different learning rates is shown in Fig. 2, which are 0.00001 and 0.0001, respectively.

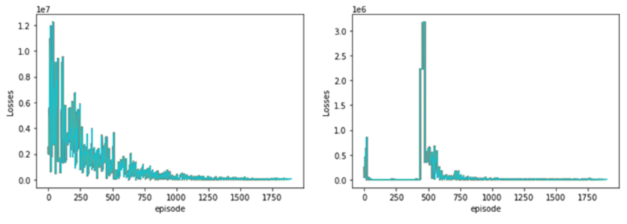

(a) Learning rate=0.00001 (b) Learning rate=0.0001

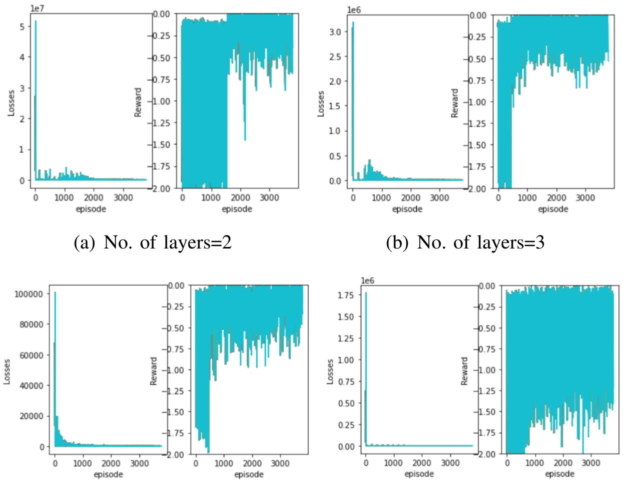
Fig. 2. Convergence performance at different learning rates.

It can be seen that when the learning rate is 0.00001, the convergence process is fast, and it gradually converges after the $1,000^{\text{th}}$ training step. When the learning rate is 0.0001, it still converges quickly, but the intermediate brief converges with drastic fluctuations and eventually converges before 1,000 times training steps. Thus, as the learning rate increases, it is more likely to find a locally optimal solution rather than a globally optimal solution. Therefore, we need to select an appropriate learning rate according to the specific situation.

The convergence performance for different numbers of DNN layers is as shown in Fig. 3. It can be seen that as the number of DNN layers increases, the model first converges faster and then tends to be stable. When the number of layers is 2, it converges after the $1,000^{\text{th}}$ training step; when the number of layers is 4, its fluctuation is slightly larger than that of three layers; when the number of layers is 5, the convergence rate increases. Thus, we can conclude that the more DNN layers are, the faster the convergence is, however, the convergence rate increases slowly after exceeding three layers. Therefore, in this paper, we choose the three-layer DNN structure as an approximation of the Q value.

C. Baseline Comparison

To evaluate the performance of the proposed DQN-based task reverse offloading algorithm, we adopt the following two schemes as baselines:


(a) No. of layers=2 (b) No. of layers=3

(c) No. of layers=4 (d) No. of layers=5

Fig. 3. Convergence performance for different numbers of DNN layers.

- **Fixed Decision (FD)**: In this scheme, we set a random half of the tasks for reverse offloading, and the other half to be used for VEC computation.
- **Full Local (FL) or Full Reverse Offloading**: In this scheme, each task is reversely offloaded to the original vehicle for calculation.

In our experiment, each simulation result was obtained over 100 replicates, but each initial environment was randomly generated. For convenience, we put all the constraints of the state variable into the constraint of random generation.

Fig. 4 studies the performance of various types of schemes with different g-parameters. It can be seen that the system consumption of each scheme increases as the g-value increases, and the proposed DQN-based task reverse offloading algorithm achieves the best performance. In addition, with the increase in the delay proportion, the gap between the proposed algorithm and other algorithms gradually increases, indicating that our algorithm has a significant effect on reducing the system consumption compared with other algorithms.

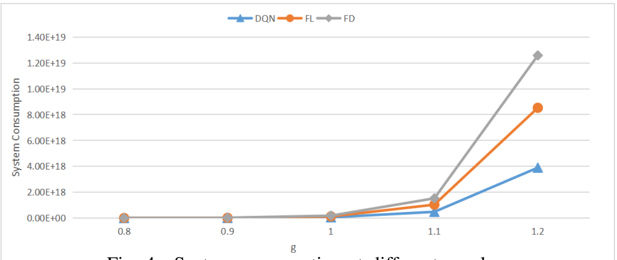


Fig. 4. System consumption at different g-values.

Furthermore, we compare the performance of different offloading schemes when the number of vehicle terminals varies. As can be seen from Fig. 5, the system consumption increases as the number of vehicle terminals increases. However, no matter the increase or decrease in the number of vehicles, the system consumption of our algorithm is always the lowest compared with the other two algorithms. In addition, the proposed algorithm can learn according to the actual information transmission situation, and make appropriate task offloading decisions, effectively taking the latency and the reduction of energy consumption into account.

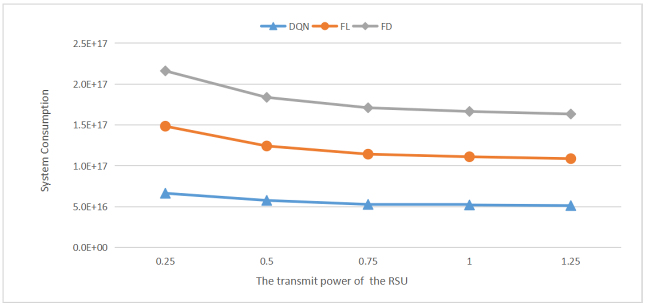


Fig. 5. System consumption at different transmit power of the RSU.

Fig. 6 shows the relationship between the system consumption and the VEC computational capacity under different scenarios. We can see that the system consumption of the proposed algorithm is significantly better than the other two algorithms, and the FL scheme is independent of the VEC CPU frequency, so its relationship with F_R^{max} remains unchanged. Besides, regardless of the computational capacity of the VEC server, the proposed algorithm can also produce less system consumption than the FL, because the task is assigned to the VEC or offloaded to the vehicle terminal for processing, thus reducing the load of a single server. Moreover, when the computational resources of the VEC server are sufficient, the system consumption of these three algorithms gradually decreases to approximately 0, so in this case, the bottleneck of the system performance is the radio resources.

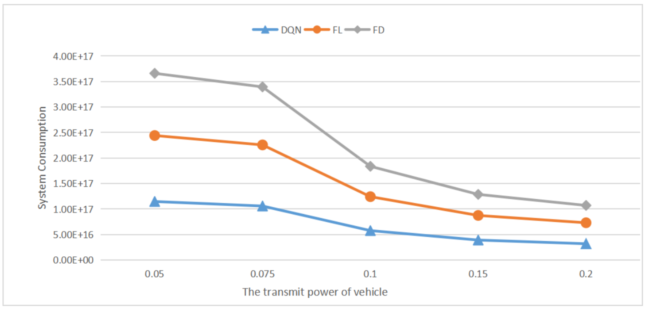


Fig. 6. System consumption under different transmit power of vehicle when the no. of vehicle terminals is 10.

V. Conclusion

In CVIS, crowdsensing data from vehicles and roadside facilities are is to the VEC, and then proper decisions are made. When there are too many vehicles, VEC can offload some computing tasks back to the vehicles for the efficient accomplishment of the tasks. In order to minimize the total computation and communication energy consumption, as well as the transmission delay between vehicle terminals and VEC and the computational delay of VEC, taking full advantage of local computational resources, we use DRL to help find the optimal reverse offloading decision.

This paper investigates the problem of task reverse offloading in a collaborative vehicle network. Firstly, the system consumption function is introduced to evaluate the energy consumption and delay of the system. Then, according to the actual constraints, the constraints related to the unloading decisions are improved. By optimizing the reverse offloading decision, a reverse task offloading algorithm based on DQN is proposed to optimize the task offloading decision and minimize the system consumption of the multi-vehicle edge network, including total energy consumption and delay in completing the task. The simulation results verify the accuracy of the algorithm and effectively reduce system consumption compared with the existing full local, fixed strategy algorithms. It can make accurate decisions in real-time with different states of the vehicle by choosing states appropriately.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant Number 62071327. The corresponding author is Huaming Wu.

References

[1] Z. Xia, J. Wu, L. Wu, Y. Chen, J. Yang, and P. S. Yu, “A comprehensive survey of the key technologies and challenges surrounding vehicular ad hoc networks,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 12, no. 4, pp. 1–30, 2021.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, “A survey of autonomous driving: Common practices and emerging technologies,” *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[3] S. Li, N. Zhang, S. Lin, L. Kong, A. Katangur, M. K. Khan, M. Ni, and G. Zhu, “Joint admission control and resource allocation in edge computing for internet of things,” *IEEE Network*, vol. 32, no. 1, pp. 72–79, 2018.

[4] Z. Wang, D. Zhao, M. Ni, L. Li, and C. Li, “Collaborative mobile computation offloading to vehicle-based cloudlets,” *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 768–781, jan 2021.

[5] W. Feng, N. Zhang, S. Li, S. Lin, R. Ning, S. Yang, and Y. Gao, “Latency minimization of reverse offloading in vehicular edge computing,” *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2022.

[6] W. Feng, S. Yang, Y. Gao, N. Zhang, R. Ning, and S. Lin, “Reverse offloading for latency minimization in vehicular edge computing,” in *ICC 2021 - IEEE International Conference on Communications*. IEEE, jun 2021.

[7] H. Wu, Z. Zhang, C. Guan, K. Wolter, and M. Xu, “Collaborate edge and cloud computing with distributed deep learning for smart city internet of things,” *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8099–8110, 2020.

[8] A. Gupta, A. Anpalagan, L. Guan, and A. S. Khwaja, “Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues,” *Array*, vol. 10, p. 100057, 2021.

[9] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, “Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing,” *Digital Communications and Networks*, vol. 5, no. 1, pp. 10–17, feb 2019.

[10] Z. Jia, Z. Zhou, X. Wang, and S. Mumtaz, “Learning-based queuing delay-aware task offloading in collaborative vehicular networks,” in *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1–6.

[11] S. Pan, Z. Zhang, Z. Zhang, and D. Zeng, “Dependency-aware computation offloading in mobile edge computing: A reinforcement learning approach,” *IEEE Access*, vol. 7, pp. 134 742–134 753, 2019.

[12] G. Qu, H. Wu, R. Li, and P. Jiao, “Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing,” *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, 2021.