Hindawi
Mobile Information Systems
Volume 2022, Article ID 9218266, 13 pages
https://doi.org/10.1155/2022/9218266

Research Article

Deep Reinforcement Learning-Based Task Offloading for Parked Vehicle Cooperation in Vehicular Edge Computing

Hui Zhao,[1] **Jiwei Hua**,[2] **Zusheng Zhang**,[1] **and Jinqi Zhu**[2]

[1]*School of Cyberspace Security, Dongguan University of Technology, Dongguan, China*
[2]*School of Computer and Information Engineering, Tianjin Normal University, Tianjin, China*

Correspondence should be addressed to Jiwei Hua; huajiwei@tjnu.edu.cn

Received 8 September 2022; Revised 9 December 2022; Accepted 12 December 2022; Published 29 December 2022

Academic Editor: Claudio Agostino Ardagna

Vehicular edge computing (VEC) has greatly enhanced the quality of vehicle service with low latency and high reliability. However, in some areas not covered by roadside infrastructures or in cases when the infrastructures are damaged or fail, the offloaded tasks cannot have the chance to be performed. Even in the areas deployed with infrastructures, when a large number of offloaded tasks are generated, the edge servers may not be capable of processing them in time, owing to their computing resources constraint. Based on the above observations, we proposed the idea of parked vehicle cooperation in VEC, which uses roadside parked vehicles with underutilized computational resources to cooperate with each other to perform the compute-intensive tasks. Our approach aims to overcome the challenge brought by infrastructure lacking or failure and make up for the shortage of computing resources in VEC. In our approach, firstly, the roadside parked vehicles are managed as different parking clusters. Then, the optimal amount of resources required for each offloaded task is analyzed. Furthermore, a task offloading algorithm based on deep reinforcement learning (DRL) is proposed to minimize the total cost, which is composed of the task execution delay and the energy consumption overhead of the parked vehicles for executing the task. A large number of simulation results show that, compared with other algorithms, our approach not only has the highest task completion execution successful rate, but also has the lowest task execution cost.

1. Introduction

In recent years, with the rapid development of Internet of vehicle (IoV), more and more vehicles are equipped with wireless devices, trip computer, as well as a serial of sensors. These vehicles are called intelligent vehicles. Consequently, many feasible applications (e.g., real-time driving monitoring, dangerous vehicle recognition, and autonomous driving) over IoV are emerging to facilitate the drivers. IoV is expected to change the way we drive and improve our driving experience greatly in the near future.

Some applications over IoV demand a lot of computing resources and are delay sensitive, such as real-time road condition analysis and recognition, and the voice-based human vehicle dynamic interaction [1, 2]. These applications usually involve complex calculation, which are called compute intensive tasks. In addition, these applications often have strict delay limitation, which need to obtain the processing results in real time. Nevertheless, the characteristics of the on board equipment of vehicles such as insufficient computing capacity and limited storage capacity seriously affect the quality of the vehicle service, thus affecting the driving experience and vehicle safety.

Cloud computing can usually provide users with sufficient and secure computing and storage services. In cloud computing, vehicles can access the cloud platform and then use the huge resources on the network, which makes up for the shortage of resources in the vehicle itself. However, the transmission of data usually results in high delay in this method. Therefore, it still cannot solve the problem of task delay limitation. Vehicular edge computing (VEC) [3] is an efficient way to resolve the abovementioned problem. In VEC, multi-access edge computing (MEC) is introduced into the IoV, that is, by placing the server with powerful computing capability at the edge of the road, compute-intensive tasks can usually be offloaded to the nearby edge server for efficient execution.

Many studies [4, 5] have been focused on VEC and showed that VEC enhances the service quality of vehicle greatly. However, most of the existing studies generally assumed that the edge servers are deployed near roadside units (RSUs), and the task offloading is realized on such framework. This edge server deployment approach has the following disadvantages:

(i) The construction of RSUs and edge servers will incur high cost, and RSU has limited communication range. Thus, it is not possible for them to cover every area of the city.

(ii) RSUs may be damaged accidently. They may also fail in a disaster. For example, in July 2021, the 48 hours of heavy rain in Henan province of China incurred great damage to the communication infrastructures, which resulted in a long time of communication interruption in Zhengzhou, China.

(iii) The computing capacity of the edge servers is limited in VEC. When a large number of offloaded tasks are generated, the edge servers may not be capable of processing them in time owing to their computing resources constraint.

We find that there are a large number of parked vehicles in the city. These parked vehicles have stable and unused computing and storage resources. Meanwhile, many compute intensive tasks support distributed computing [6, 7]. Based on the above facts, in this paper, we propose the idea of parked vehicle cooperation in VEC, which uses parked vehicles on and off the street to cooperate with each other to perform compute-intensive tasks, that is, roadside parked vehicles are organized into static service nodes and multiple vehicles are utilized to process a task request in parallel. Specially, under the condition that infrastructures are not deployed or damaged, parked vehicles could still work. Meanwhile, as natural edge servers, organized parked vehicles can also provide assistance to the edge servers deployed in practice to perform offloaded tasks handling, with the aim of solving the problem of resource shortage of the edge servers.

The main contributions we have made in this paper are summarized as follows:

(i) We propose the idea of parked vehicle cooperation in VEC for task offloading. Our approach could overcome the challenge brought by the inadequate deployment of RSUs or the damage of RSUs when accidents or disasters happen, and make up for the shortage of resources in physical deployed edge servers.

(ii) We organize the parked vehicles within a certain range on the road into parking clusters, and then analyze the task execution delay and the energy consumption of parked vehicles when performing the task.

(iii) We establish an optimization model to analyze the optimal amount of resources allocated to each offloaded task by the parking cluster, and propose a task offloading strategy based on deep reinforcement learning (DRL) [8].

(iv) Experimental results show that the proposed strategy has better offloading performance compared with other offloading strategies.

2. Related Work

In recent years, scholars at home and abroad have conducted a lot of researches on task offloading in VEC, and these works basically fall into two categories.

The first one only uses physically deployed edge servers to handle the offloaded task. For example, given that a single MEC server cannot meet the large number of task offloading requirements, Zhang et al. [9] designed a joint task offloading scheme, which forwards tasks to surrounding edge servers when a server is overloaded. Tang et al. [10] used deep reinforcement learning to realize task offloading, so as to deliver the task efficiently to the target edge server.

Moreover, in [11], the authors combine MEC technology with social network to improve the service quality and user experience. In [12], the authors prove that the minimization of task execution delay is NP hard and propose a greedy algorithm to deal with task offloading. Throughout these works, the help of the edge servers relieves the resources shortage in vehicle to some extent. Nevertheless, owing to the computing resources constraint of the edge servers, when a large number of offloaded tasks are generated, some tasks may not be processed within the constrained time.

The second one handles the task requests of vehicle users through resources expansion. In [13, 14], the authors present the collaboration between MEC and remote clouds to provide users with powerful computing and storage capacity. As we have illustrated above, the delay of data transmission in cloud computing is intolerable for many vehicle applications. In [15], the authors show the fog computing system they designed, in which the mobile buses are used to perform the task shared by roadside units. Similarly, in [16–18], the authors put forward the assistance from the moving vehicles to execute the tasks offloaded to edge server. However, due to the high mobility of vehicles and the variable traffic conditions, the communication between the edge servers and the moving vehicle as well as among the moving vehicles are not stable, which makes the task difficult to be completely offloaded. Moreover, Wu [19] propose an edge computing task offloading framework based on parked vehicle to solve the problem of overloading in edge servers. Huang et al. [20] propose to dispatch the parked vehicles on demand through the deployed MEC server. However, these two works neglect the efficient organization of the parked vehicles and systematic management of their computing resources, which is of vital importance in task offloading.

At present, there are many studies [21–25] using deep neural network to solve the task offloading problem in VEC.

Xu et al. [26] conduct an exhaustive survey on utilizing AI in edge service optimization in IoV. However, the shortcoming of most of these works is that the assumption of the scenario is generally too simple, and the situation space is quite small. For example, in [21], the authors only considered the scenario that one vehicle offloads tasks to multiple nearby edge servers. In fact, there are many vehicles on the road generate tasks, and one vehicle may generate more than one task simultaneously.

In this paper, we add the underutilized roadside parked vehicles to VEC, and consider the actual situation in which multiple mobile vehicles generate multiple tasks at the same time. As far as we know, this is the first attempt to solve the simultaneous task offloading from multiple vehicles to parked vehicles using DRL.

3. Framework Overview

3.1. Assumptions.

(1) We assume each offloaded task can only be processed on only one edge server, as researchers in [4, 27]. The edge server we focus on in this paper is the virtual server formed by the parked vehicles within an area.

(2) Since the computing resources of the vehicle are limited, we assume that only one task can be processed in a parked vehicle at a time.

(3) We assume that some parked vehicle owners have the willingness to share the resources of their vehicle. In [28, 29], the authors proposed that the parked vehicles with rich and underutilized computation resources in city areas could be explored for the purpose of providing third-party services. Furthermore, literature [30] attempted to use parked vehicles to assist content distribution in vehicular ad hoc networks. We believe that there will appear effective incentive mechanisms, such as task offloading priority, free parking or monetary reward, which would encourage the vehicle owners to share their vehicle resources(computation and storage) while parking.

3.2. System Framework. The system framework in this paper is given in Figure 1, which includes moving vehicles on a road and parked vehicles on the roadside. Data transmission between vehicles is carried out in vehicle-to-vehicle (V2V) mode [31]. In order to organize the parked vehicles effectively and make them cooperate well with each other, the parked vehicles on a street are managed as different parking clusters. Parking clusters serve as virtual edge servers to perform the computing tasks offloaded from surrounding moving vehicles. In order to reduce the task transmission delay, the maximum length of a parking cluster is limited to L meters. For each parking cluster, one vehicle located in the middle of the cluster is elected as the cluster head. After the cluster head is determined, other remaining vehicles are acted as cluster members.

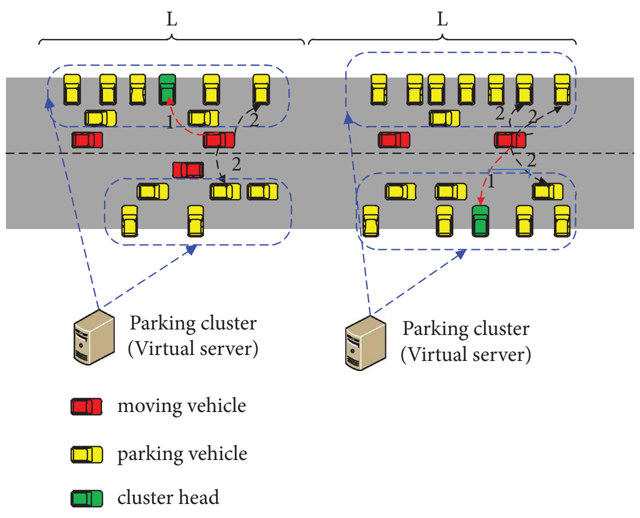

L
L
1
2
2
2
1
2
2
Parking cluster (Virtual server)
Parking cluster (Virtual server)
moving vehicle
parking vehicle
cluster head

Figure 1: System framework.

Let the set of parked vehicles in a parking cluster be $\mathrm{SeV} = \{1, 2, \ldots, M\}$, each element in the set marks a parked vehicle in this cluster, and the total number of parked vehicles is M. Members of the parking cluster regularly send beacon messages to tell their status and information, such as ID number, remaining battery capacity, and whether a task is being performed to the cluster head. The cluster head manages vehicles member and maintains the cluster structure according to the received information. The cluster head is also responsible for collecting task requests from the moving vehicles on the road where the cluster is located and responsible for the allocation of resources of members. Since the cluster head may move away at any time, a neighbor of the cluster head is used as a backup cluster head to maintain a copy of the information stored by the cluster head to enhance the robustness of the cluster.

Assuming that the task set generated by moving vehicles is $Q = \{1, 2, \ldots, N\}$, and each task is associated with a quadruple. For example, task Q_i is represented as $(c_i, s_i, y_i, t_i^{\max})$, where the four parameters indicate the size of the task, the amount of resources required to execute the task (number of CPU cycles), the size of the result returned by the task, and the maximum delay tolerated by the task, respectively. According to literature [11], the relationship between these four parameters can be expressed as follows:

$$\begin{aligned} s_i &= \alpha c_i, \\ y_i &= \beta c_i, \end{aligned} \tag{1}$$

where α represents the computational complexity of the task and β is the ratio between the size of the task completion result and the size of the task. In order to simplify the model, α as well as β is assumed to have the same value for all the computational tasks in this paper.

In Figure 1, intelligent vehicles moving on the road generate computing tasks randomly. The source vehicles have limited computing and storage capabilities. Thus, for some tasks that the local executing time doesn't exceed the maximum tolerated delay, they could be processed locally.

For other tasks that cannot be completed locally within the maximum tolerated delay, they should be offloaded to the parking cluster for execution.

The source vehicle will send the driving speed, current location, and the task information carried by itself to the cluster head of the parking cluster in its range (phase 1 in Figure 1). Based on the collected requests, the cluster head analyzes the optimal amount of resources required by the task and determines the optimal location for each task to be offloaded according to the DRL-based task offloading policy described in Section 5. The task offloading is performed accordingly (phase 2 in Figure 1). When the available resources in the parking cluster are not sufficient to perform a task, this task is executed locally. If the time required to obtain the task result exceeds the maximum allowable delay of the task, the task execution is considered to be failed.

3.3. Stability Analysis. The status of the parked vehicles on the road is dynamic. The reason is that the parked vehicles may join or drive away from the cluster anytime. In order to provide reliable computing support for offloaded tasks, the parking cluster must be relatively stable. In this section, the distribution function model of the change of vehicle numbers in the parking cluster is established.

Literature [32] proved that the number of members joining and driving away from the parking cluster followed the Poisson process of B cars per second and K cars per second, respectively. Let the number of cars joining and driving away be $B(t)$ and $K(t)$ with the probability distribution as follows:

$$\begin{aligned} P(B(t) = b) &= \frac{(Bt)^b}{b!} e^{-Bt}, b = 0, 1, 2 \ldots, \\ P(K(t) = k) &= \frac{(Kt)^k}{k!} e^{-Kt}, k = 0, 1, 2 \ldots. \end{aligned} \tag{2}$$

Then, the probability of n more members joining in the cluster, that is, the probability of $B(t) - K(t) = n$ is as follows:

$$P(B(t) - K(t) = n) = \sum_{i=0}^{\infty} P(B(t) = i + n)P(K(t) = i) = \sum_{i=0}^{\infty} \frac{(Bt)^{i+n}(Kt)^i}{(i+n)!i!} e^{-(B+K)t}. \tag{3}$$

Similar to equation (3), the probability of n members missing in the cluster within this time is as follows:

$$P(K(t) - B(t) = n) = \sum_{i=0}^{\infty} \frac{(Bt)^i (Kt)^{i+n}}{(i+n)!i!} e^{-(B+K)t}. \tag{4}$$

Figure 2(a) describes the distribution function of increasing number of parked vehicles after different time t. According to literature [29], $B = 0.000$ and $K = 0.0003$, that is, on average, 3 cars enter the parking cluster every hour and 1 car drives away from it. Figure 2(b) describes the distribution function of the reduction of parked vehicles number after different time t, where $b = 0.0008$ and $K = 0.0013$. As shown in Figure 2, after 20 seconds, the probability of increasing or decreasing 1 car in the cluster is less than 2%, and after 60 seconds, the probability of increasing or decreasing 1 car in the cluster is less than 5%. Therefore, the members of the parking cluster are relatively stable, and the change of the vehicle numbers in the parking cluster has slight influence on the calculation of the offloading tasks.

4. Task Execution Cost Calculation

4.1. Task Execution Delay. For the task Q_i, if it is executed by the vehicle terminal locally, the execution delay is as follows:

$$T_i = T_i^{\text{local}} = \frac{s_i}{f_0}, \tag{5}$$

where f_0 is the computing capability of the moving vehicle.

If T_i^{local} does not exceed the maximum tolerated delay of the task $t_i^{\max}$, the task can be successfully executed locally. Otherwise, the task should be offloaded to the parking cluster. And multiple parked vehicles in the parking cluster will execute the task in parallel to speed up the execution of the task.

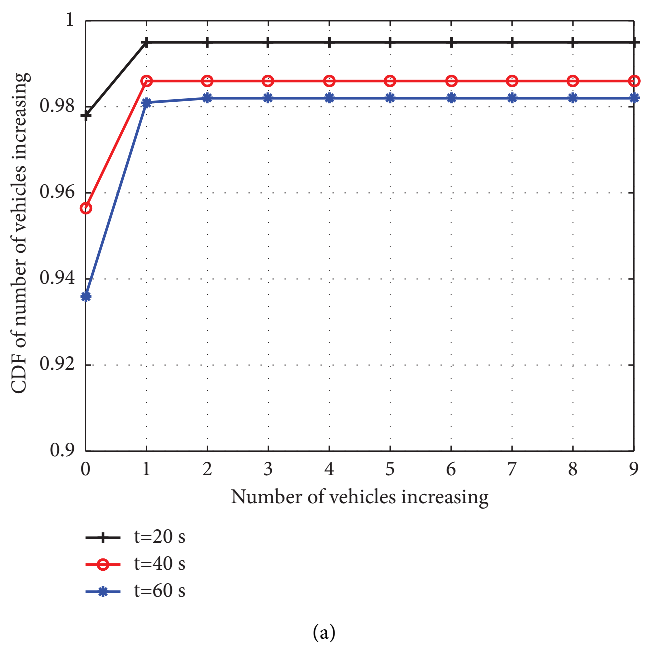
If task Q_i needs to be offloaded, the task should be divided into several subtasks according to the resources obtained from the parking cluster. The execution delay of each subtask is composed of three parts, namely, data offloading delay T_{ij}^{up}, calculation delay T_{ij}^{compute} on parked vehicle j, and result return delay T_{ij}^{down}. The task is completed only when all subtasks are fully calculated and the execution result is returned to the moving vehicle. Therefore, the maximum execution delay of each subtask is taken as the task execution delay. The completion delay of this task is expressed as follows:

$$T_i = \max_j \left(T_{ij}^{\text{up}} + T_{ij}^{\text{compute}} + T_{ij}^{\text{down}}\right). \tag{6}$$

The delay of each subtask is calculated as follows:

$$\begin{aligned} T_{ij}^{\text{up}} &= \sum_{k=1}^{h_{ij}} \frac{c_{ij}}{C_{\text{vel}}} + \frac{l_{ij} - hR}{v_j}, \\ T_{ij}^{\text{compute}} &= \frac{s_{ij}}{f_j}, \\ T_{ij}^{\text{down}} &= \sum_{k=1}^{h_{ij}'} \frac{y_{ij}}{C_{\text{vel}}} + \frac{l_{ij}' - h_{ij}'R}{v_j}. \end{aligned} \tag{7}$$

where f_j represents the computing capacity of parked vehicle j, and C_{ij} is the size of subtask offloaded on the vehicle j, and s_{ij} is the total computing resources required by the subtask. These two parameters can be obtained according to Section 5.1, l_{ij} is the distance between the moving vehicle



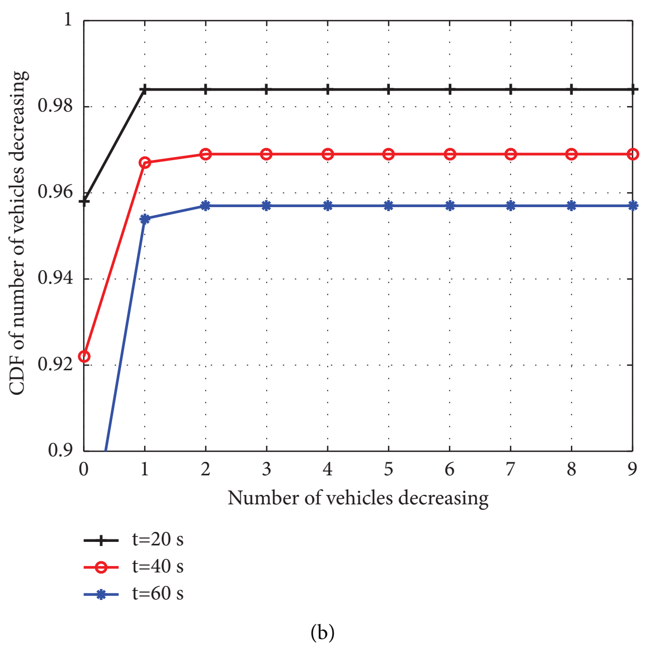


Figure 2: Distribution function of change number of vehicles in cluster (a) vehicle increasing (b) vehicle decreasing.

generating the task and the parked vehicle j performing the task, and l_{ij}' is the distance between the parked vehicle j and the moving vehicle when the task results are returned, which can be predicted by the speed and initial position of the moving vehicle. Data is transmitted in a store-and-forward way among vehicles. The subtask reaches the parked vehicle j after h_{ij} hops between vehicles, and the task result is returned after h_{ij}' hops after completion of calculation. C_{vel} represents the throughput of data forwarding between vehicles, R is the transmission radius of vehicles, v_j is the average speed of moving vehicles on the road where the parking cluster is located. For the hop number h_{ij}, we have the following formula:

$$h_{ij} = \frac{\rho l_{ij} \int_0^R \rho e^{-\rho z} \mathrm{d}z \int_0^R z\rho e^{-\rho z} \mathrm{d}z}{R} = \frac{l_{ij}}{R}\left(1 - e^{-\rho R}\right), \tag{8}$$

In the above formula, ρ is the vehicle density on the street where the parking cluster is located, which can be obtained from the electronic map of vehicle equipment. The calculation method of h_{ij}' is similar to equation (8).

4.2. Energy Consumption Analysis of Parked Vehicles. If the task is offloaded to the parked vehicle for execution, the parked vehicle will receive the task data, execute the task, and output the results, and the energy consumption of the parked vehicle is the sum of all the three parts of energy consumption. For the owners of parked vehicles, they want to reduce their energy consumption as much as possible while performing tasks. Moreover, if the battery energy is exhausted due to the execution of tasks, the vehicle cannot be started, resulting in serious consequences. Thus, the energy saving in parking vehicles should be considered.

Given that the vehicle transmits power P^u and the vehicle receives power P^d, the total energy consumption of the parked vehicle for performing task Q_i is as follows:

$$E_i = \sum_j \left(P^u T_{ij}^{\text{up}} + \delta f_j^2 T_{ij}^{\text{compute}} + P^d T_{ij}^{\text{down}} \right), \tag{9}$$

where δ represents the calculated energy consumption coefficient of the task.

4.3. Task Execution Cost. The proposed task offloading scheme should pursue the smallest task completion delay. Meanwhile, considering the battery capacity of the parked vehicle, it is necessary to try to reduce the energy consumption of the parked vehicle when performing the task. Therefore, the execution cost of each task is defined as the weighted sum of the two, and the corresponding execution cost of task Q_i is calculated as follows:

$$\text{cost}_i = \lambda T_i + (1 - \lambda) E_i, \tag{10}$$

where λ represents the weight factor. Then, the total task execution cost is as follows:

$$\text{cost} = \sum_{i=1}^{N} \text{cost}_i, \tag{11}$$

where N represents the total number of all the tasks, while T_i and E_i could be obtained by equations (6) and (9)

5. Task Offloading Scheme

5.1. Optimal Resources Determination. In order to realize reasonable task offloading, it is necessary to determine the optimal resources allocated to each offloading task. Assume that each task has a resource set $R = \{1, 2 \ldots, r_i^m, \ldots r_{\max}\}$, where $r_{\max}$ is the maximum computing resources that can be assigned to each task. When the resources assigned to task Q_i is r_i^m, the execution time is c_i/r_i^m. In this paper, the resource allocation problem is expressed as the following optimization problem, with the purpose of minimizing the execution time of all of the tasks offloaded in total.

$$
\begin{aligned}
&P\colon \min \quad f = \sum_{i=1}^{N}\sum_{m=1}^{r_{\max}} \frac{c_i}{r_i^m} x_i^m, \\
&\text{s.t.} \quad x_i^m \in \{0,1\}, \forall i \in \{1,2,\ldots N\}, \forall m \in \left(1,2,\ldots,r_{\max}\right), \\
&\sum_{i=1}^{N}\sum_{m=1}^{r_{\max}} r_i^m x_i^m \le R_j, \\
&\sum_{m=1}^{r_{\max}} x_i^m \le 1, \forall i \in \{1,2,3,\ldots N\}.
\end{aligned}
\tag{12}
$$

In the above optimization model, constraint (12) indicates that x_i^m can only be 0 or 1, where $x_i^m = 0$ indicates that the resource item is not selected and $x_i^m = 1$ otherwise. Constraint (12) guarantees that the sum of resources obtained by the tasks from the parking cluster does not exceed the available resources of the parking cluster. R_j in this formula represents the maximum available resources of parking cluster j. Constraint (12) ensures that each task can only take at most one source item from set R.

If each parking cluster is regarded as a knapsack, and the maximum weight constraint of knapsack j is R_j, then P is a knapsack problem as follows: within the limit of the maximum weight of the knapsack, for each task, one element is selected from its resource set and put into the backpack, so that the total value in the knapsack is minimized. In this paper, as shown in Algorithm 1, a greedy algorithm is used to solve this problem.

Assuming that the set of tasks is T, the maximum resources can be assigned to each task is $r_{\max}$, and the total available resources of the parking cluster is R_j. We used R as the current available computing resources of the parking cluster, which is initialized as R_j. As depicted from line 3 to line 14, as long as there are still some tasks in T, the task i_0 and resource item m_0 with minimized execution time are selected for resources allocation.

Since we assume that only one task can be processed in a parked vehicle at a time, after the optimization scheme is solved, the number of total parked vehicles assigned to each task can be obtained, and therefore the number of subtasks of the offloaded task is gotten.

5.2. Task Offloading Algorithm Based on DRL. Compared with traditional machine learning methods, deep reinforcement learning (DRL) is capable of dynamic learning, which could learn new handling strategies based on new complex situations encountered by the vehicles, and even generalize to other similar situations. It will also constantly adjust the offloading strategy to achieve the best return. Therefore, here, DRL is used to assign offloaded tasks to parked vehicle members in parking cluster. The main elements of DRL model used in this paper are as follows:

(1) Agent: as an intelligent agent in the deep reinforcement learning model, it is responsible for making action decisions while interacting with the environment. In the hypothetical scenario of this paper, the cluster head acts as the agent of the model. It collects characteristic information of moving vehicles and tasks, assigns specific parked vehicles to the offloaded task, and keeps learning in an iterative way to achieve the optimal task offloading with minimized total costs.

(2) State: an appropriate representation of feature states is crucial for DRL models. In this model, the size of the task, the maximum tolerant delay, the current position and speed of the moving vehicle all affect the offloading decision, so $s_i = (C_i, L_i, V_i, D_i)$ is used to represent the state characteristics of the environment, where C_i, L_i, V_i, D_i is the set of task size, moving vehicle position, driving speed and maximum tolerable delay in the current state, respectively.

(3) Action: the action set in the state of s_i is represented as $a_i = (a_{i1}, a_{i2}, \ldots, a_{iM})$, where 1,2, . . ., M indicate the serial number of parked vehicles within the parking cluster. We have $a_{ij} \in \{0,1\}$, and $a_{ij} = 1$ indicates that task Q_i would be offloaded to the parked vehicle j. Subject to the tolerated delay, each task might be performed locally or offloaded to the parked vehicles. If the task is executed locally, we have $\sum_{j=1}^{M} a_{ij} = 0$. If multiple parked vehicles are selected as the destination of an offloaded task, we have $\sum_{j=1}^{M} a_{ij} > 1$. Generally in DRL, the sum of a_{ij} is either 1 or 0. However, in our DRL model, the sum of a_{ij} is greater than 1, which is different from any other works based on DRL.

(4) Reward: Every time the agent makes a decision, it can get an immediate reward. Considering that we wish the task completion rate is as high as possible while the task execution cost is minimized, the reward calculation method of task Q_i is as follows:

```
(1) Input the set of tasks $\mathbf{T}$, the maximum resources can be assigned to each task $r_{\max}$, and the total available resources of the parking cluster $R_j$.
(2) use $R$ as the current available computing resources of the parking cluster, initialize $R = R_j$.
(3) while $T! = \varnothing$ do
(4)   if $R > 0$ then
(5)     for each task $i$ in set $T$ do
(6)       for each $m$ ($m \le r_{\max}$ and $m < R$) do
(7)         Select the $i$ and $m$ with minimized $c_i/r_i^m$, denoted as $i_0$ and $m_0$
(8)       end for
(9)     end for
(10)  else
(11)    break;
(12)  allocate $m_0$ resources to task $i_0$
(13)  $T := \mathrm{T} - i_0$
(14)  $R := R - m_0$
(15)  end if
(16) end while
```

Algorithm 1: The optimal resources allocation.

$$r_i = \begin{cases} \dfrac{\tau s\_\text{rate}}{\lambda T_i + (1-\lambda)E_i} & \text{if } T_i \le t_i^{\max}, \\ 0 & \text{else.} \end{cases} \tag{13}$$

where s_rate represents the completion rate of the task after the end of the current iteration cycle, and the larger the completion rate, the better the decision of the current round. τ is the expansion factor of the reward, with the aim of preventing the reward value from being too small to lead to the slow gradient descent speed. In this paper, the value of τ is 100.

(5) Policy: due to the large action space, it is difficult to get a good result by predicting action value through neural network. Therefore, this paper directly takes softmax output of neural network as a strategy, and the probability of choosing parking vehicle j to execute the task in state s_i is as follows:

$$\pi_\theta\left(s_i, a_{ij}\right) = P\left(a_{ij} \mid s_i, \theta\right). \tag{14}$$

The actions are chosen according to the above formula in the actual decision. For example, if the task is divided into three subblocks, then the three actions with the highest probability are chosen.

(6) Objective function: the objective of the proposed scheme in this paper is to find the optimal offloading strategy π^* and maximize the expected reward value of future multiple scheduling cycles (i.e., minimize the task execution cost). Therefore, the objective function based on strategy entropy [33] is adopted to update the parameter θ by gradient descent. The objective function is as follows:

$$L(\theta) = -E_i\left[Q_\pi\left(s_i, a_i\right)E_j\left(\log\pi_\theta\left(s_i, a_{ij}\right)\right)\right], \tag{15}$$

where $Q_\pi(s_i, a_i)$ represents the action value obtained by executing action a_i in state s_i, and the calculation method is as follows:

$$Q_\pi\left(s_i, a_i\right) = \begin{cases} r_i & \text{if } s_i \text{ is the terminated state,} \\ E\left(r_i + \gamma Q_\pi\left(s_{i+1}, a_{i+1}\right)\right) & \text{else.} \end{cases} \tag{16}$$

In the above equation, γ is the attenuation factor. Performing the action a_i in state s_i will get an immediate reward r_i. The next state is denoted as s_{i+1}. The pseudo-code of DRL based task allocation in this paper is shown in Algorithm 2.

This DRL based task offloading algorithm use the feature vector $\varphi(s_i)$ of state s_i as the input of the neural network and use the softmax output of the neural network as the probability of selecting each action. As depicted in line 13, the parameters θ of the neural network are updated using batch gradient descent algorithm.

6. Simulation Results

In order to evaluate the performance of the proposed DRL-based task offloading strategy, simulation experiments in Python are conducted in this section. In the simulation, the road is a two-way lane, and the length of

```
(1) Initialize step length Step, attenuation factor $\gamma$, sample number of gradient descent batch_size.
(2) Initialize the parameters $\theta$ of the neural network randomly and initialize the experience replay buffer $D$.
(3) for each episode do
(4)    Initialize the environment state, get its feature vector $\varphi(s_0)$.
(5)       for each iteration do
(6)       Use $\varphi(s_i)$ as the input, obtain the softmax output of the neural network. Select action $a_i$ according to equation (14)
(7)       Execute the action $a_i$, observe the new environment state $s_{i+1}$, and gets the corresponding immediate reward $r_i$
(8)       Put the quadruple $\varphi(s_i), a_i, r_i, \varphi(s_{i+1})$ into the experience replay buffer
(9)    if $s_{i+1}$ is the terminated state then
(10)          break; //end this iteration
(11)      end if
(12)   end for
(13)   Obtain batch_size samples from the experience replay buffer $D$, and update the parameters $\theta$ of neural network through
     minimizing the objective function in equation (16) using batch gradient descent algorithm.
(14) end for
```

Algorithm 2: DRL based task offloading.

the parking cluster is 1000 meters. The speed of moving vehicles obeys the uniform distribution among [40 km/s, 80 km/s]. The vehicle-to-vehicle communication complies with IEEE 802.11p protocol, and the communication radius is 250 m. Each moving vehicle generates calculation tasks randomly. The size of each task obeys the uniform distribution of [2 Mb, 4 Mb]. The computing resources of each vehicle is 0.4 GHz, and the vehicle transmission and reception power is $P^u = P^d = 0.1\text{W}$. For the resources required by the calculation task and the return size of the result, $\alpha = 1.25, \beta = 0.1$. The weight parameters of time cost and energy cost are 0.6 and 0.4, respectively. The neural network model used in this paper includes 2 hidden layers, and the number of neurons is 512 and 256, respectively. Relu is selected as the activation function, and different state features are placed in the same order of magnitude through normalization before training, so as to reduce the influence of features with large variance on the model.

In the process of training, the learning rate controls the step size of gradient descent. Too large learning rate may make the model exceed the optimal value and fail to converge; too small learning rate may lead to low model optimization rate and even make the model fall into local optimal. Therefore, it is very important to choose an appropriate learning rate. In Figure 3, the training effects of the model proposed in this paper under different learning rates are analyzed. As can be seen from Figure 3, when the learning rate is 0.001, the completion rate of computational tasks does not change significantly with the progress of training. Obviously, the model converges slowly owing to the fact that the learning rate is too small. When the learning rate is 0.1 and 0.05, although the model converges quickly in the early stage, the completion rate fluctuates greatly in the later stage, this is because the learning step is too large, which leads to the model oscillating around the optimal value. When the learning rate is 0.01, although the model converges slowly in the early stage, certain optimization effect can still be achieved in the later stage. Therefore, as shown in Figure 3, the learning rate used in this paper gradually decreases with the training. The initial value of learning rate is 0.1, and it decreases by 0.005 after 1000 training cycles, which not only ensures the model optimization rate in the early stage, but also ensures stable convergence in the later stage.

Attenuation factor is also an important hyper parameter in the deep reinforcement model, this can be explained by the fact that in a decision problem with a long period, it is necessary to consider not only the immediate reward but also the future reward. Figure 4 reflects the influence of different attenuation factors on model training. It can be seen from the figure that when the attenuation factor is small, the model in this paper can achieve better training effect.

After parameter adjustment, the important hyper parameter settings of DRL network model proposed in this paper are shown in Table 1.

The performance of the proposed scheme is verified by comparing with other two task offloading schemes, which are local computing and random offloading.

Local computing (LC): all tasks are executed locally. If the completion time exceeds the maximum allowed delay, the task fails to be executed.

Random offloading (RF): the task is executed locally if the local execution time meets the delay requirement. Otherwise, several parked vehicles are randomly selected to execute the task. If the completion time exceeds the maximum delay, the task fails to be executed.

6.1. Impact of Computing Resource Quantity on Offloading Performance.

This group of experiments mainly evaluate the impact of task computational complexity on the performance of each task offloading scheme. The computational complexity of the task, that is, the amount of resources required per bit of data, varies from 900 round/bit to 1200 round/bit. The experimental results are shown in Figures 5 and 6.

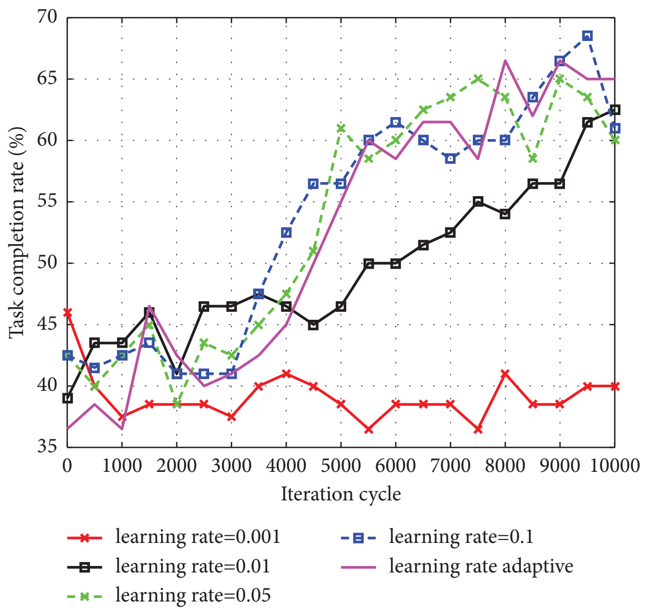


Figure 3: Effects of different learning rates on model training.

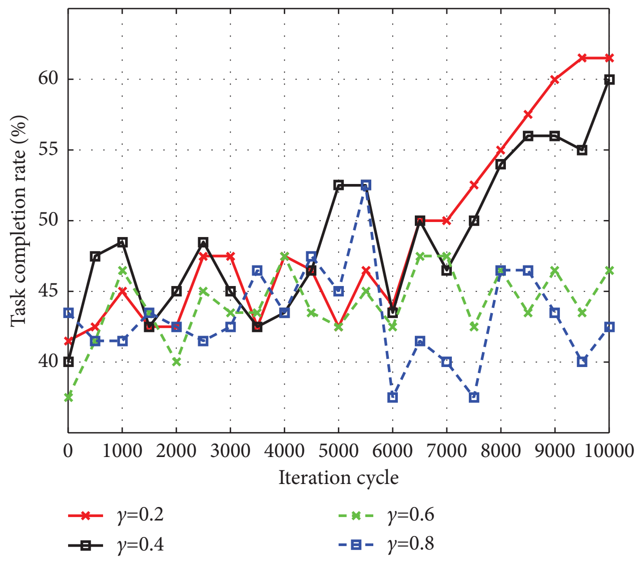


Figure 4: Effects of different decay factors on model training.

Table 1: Hyper parameters of task offloading model.

Parameter	Value	Annotation
Batch_size	80	Size of the training batch
γ	0.2	Attenuation factor
Learning_rate	0.1	Initial value of learning rate
Step	80	Step
Layer 1	512	Number of neurons in the first layer
Layer 2	256	Number of neurons in the second layer

As can be seen from Figure 5, with the increase of task computational complexity, the task completion rates of the three schemes declined. This is because when the local computing resources and the computing resources of parked vehicles remain unchanged, the more computing resources required by a task, the longer the computing time is required by each task and subtask. It results in an increase in the task execution delay. Therefore, the task completion rate keeps decreasing when the task tolerance delay is constant. When the computational complexity is 1200 round/bit, the task

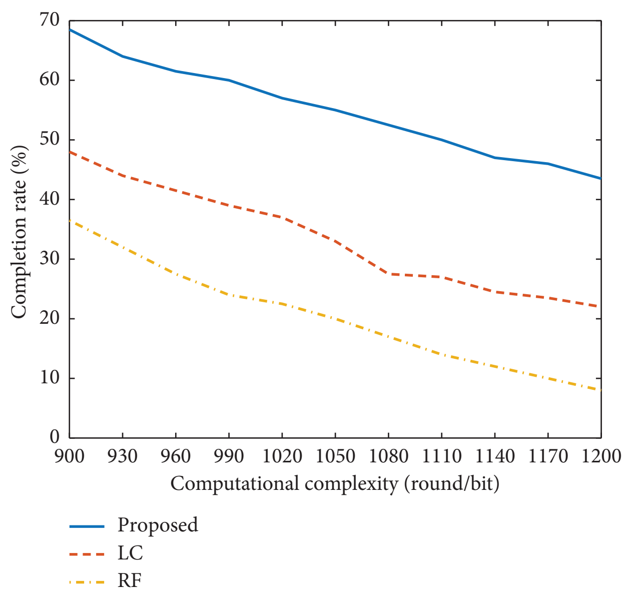


Figure 5: Relationship between task completion rate and computational complexity.

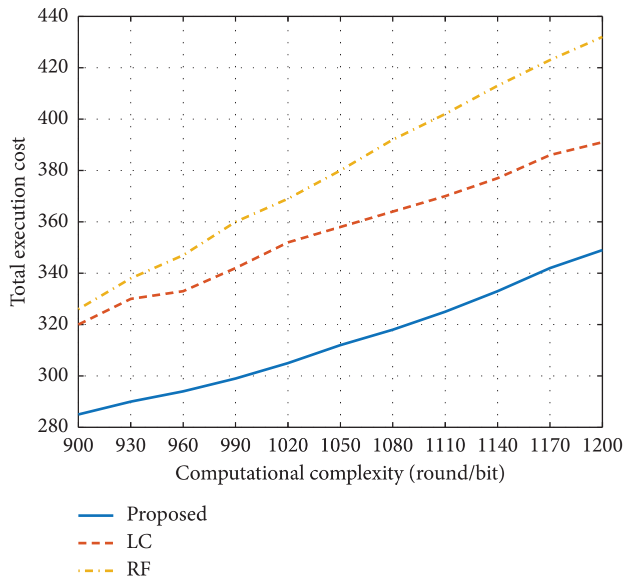


Figure 6: Relationship between total execution cost and computational complexity.

completion rate of the local computing scheme is only 9% and that of the random unloading scheme is 22%, while that of the scheme proposed in this paper is 45%. It can be seen that the task offloading success rate of the strategy proposed in this paper is always higher than that of the comparison strategy. Compared with random offloading scheme and local execution scheme, the task completion rate of the proposed scheme is increased by 23% and 36%, respectively. This shows that even with the increasing computational complexity, the strategy proposed in this paper can allocate parking resources better through reinforcement learning, thus improving the task completion rate.

Figure 6 shows that with the increase of task computational complexity, the total task execution costs of the three offloading schemes all show an upward trend. This is because when the computational complexity increases, the task calculation time increases and the resulting calculation energy consumption of parked vehicles increases. It can also be seen from Figure 6 that the total task cost of the algorithm proposed in this paper is the lowest. When the computational complexity is the highest, compared with the local execution scheme, the total consumption of random offloading scheme is reduced by 40, while the total cost of the scheme proposed in this paper is reduced by 90. As can be seen from Figures 5 and 6, the scheme proposed in this paper can guarantee the success rate of the task and at the same time have a low energy consumption of parked vehicles.

6.2. Impact of Task Number on Offloading Performance. This group of experiments mainly discuss the changes of two performance indexes, task completion rate and total execution cost of each strategy while the number of tasks changes. In the experiment, the computational complexity of tasks remained at 1000 round/bit, and the number of tasks gradually increased from 8 to 80.

While the number of tasks is increasing, the limited parking vehicle resources are not enough to provide task offloading and computing services for the excessive tasks. Therefore, in Figure 7, as the number of tasks increases, the task completion rate of LC, random offloading and the strategy proposed in this paper presents a downward trend, while the completion rate of the strategy proposed in this paper is far higher than other strategies. As Figure 8 shows, in the case of constant computational task complexity, with the increase in computing tasks, the total task execution cost of the three schemes increases. While the number of tasks is less than 10, the difference among the total cost of the three schemes is not big. However, with the increase of number of tasks, the gap of the total task execution cost of the proposed scheme and that of the two comparison schemes is more and more big. When the number of tasks is greater than 48, the reduction rate of the completion rate of the proposed offloading scheme in Figure 7 is significantly accelerated, and the total task execution cost of the three schemes in Figure 8 increases at approximately the same rate. The reason is that when there are too many tasks, the parked vehicles on the roadside cannot meet the task execution request. As a result, computing tasks can only be executed locally. By comparing the changes of total cost and completion rate of the three schemes with the number of tasks, it can be seen that the scheme proposed in this paper can better cope with a large number of tasks and complete as many tasks as possible with a smaller cost.

6.3. Impact of Vehicle Speed on Offloading Performance. This group of experiments studies the influence of moving vehicle speed on offloading performance. The experimental results are shown in Figures 9 and 10. In the experiment, the

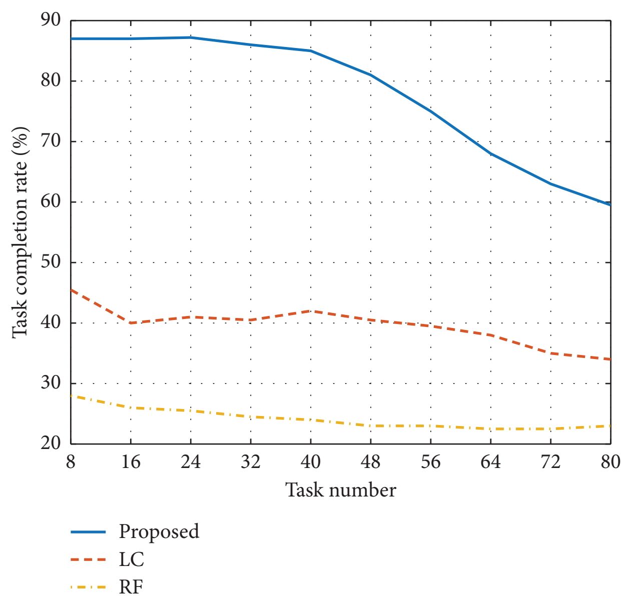


Figure 7: Relationship between task completion rate and task number.

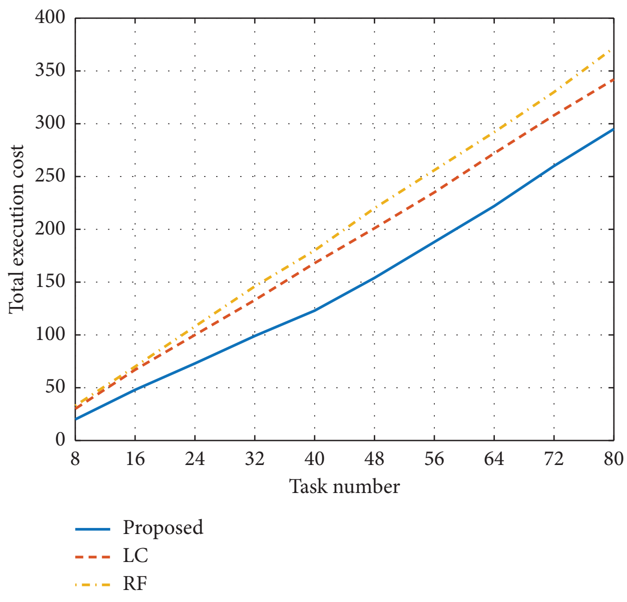


Figure 8: Relationship between total execution cost and task number.

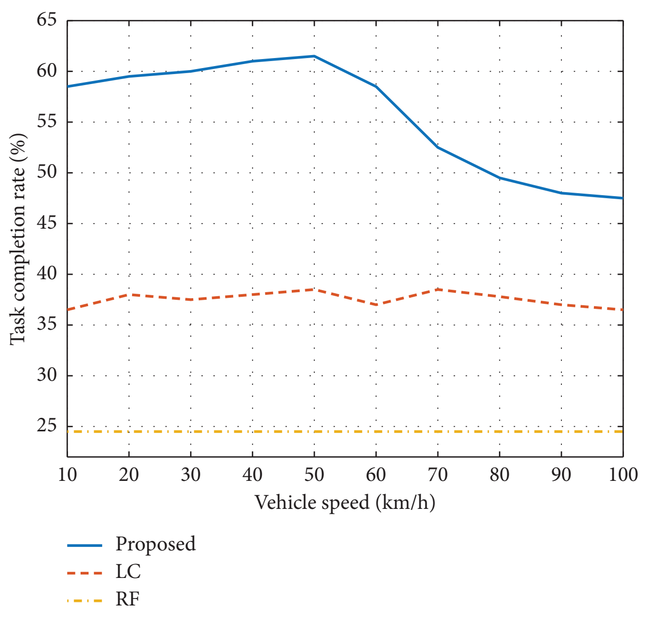


Figure 9: Relationship between task completion rate and vehicle speed.

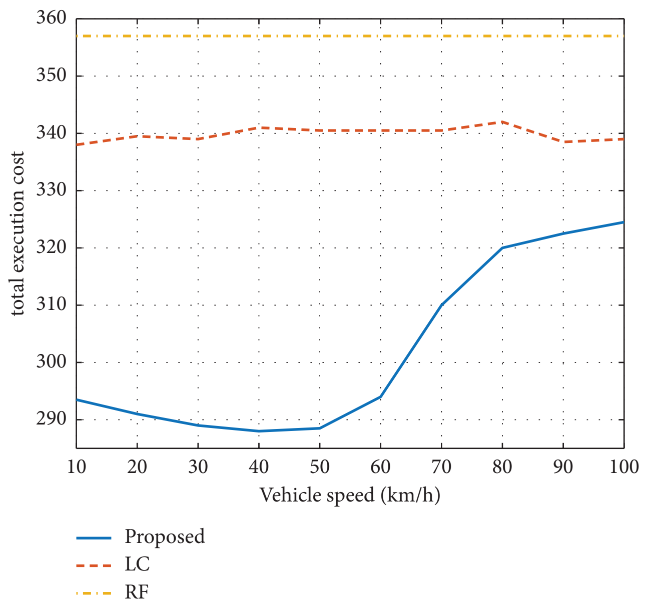


Figure 10: Relationship between total execution cost and vehicle speed.

computational complexity of tasks is kept at 1000 round/bit, the number of tasks is 80, and the value range of moving vehicle speed is [40 km/h, 80 km/h]. As shown in Figures 9 and 10, when the moving vehicle speed increases, the task completion rate of the scheme proposed in this paper increases first and then decreases, while the total execution cost decreases first and then increases. When the moving vehicle speed is about 50 km/h, the task completion rate is the highest and the execution cost is the lowest. This is because when the vehicle speed is low, with the increase of the speed, the connectivity between vehicles is enhanced and the transmission delay becomes smaller, so the execution time of the task is reduced and the completion rate is increased. However, when the speed exceeds 50 km/h, due to the excessive speed, the moving vehicle will be far away from the signal range of the parked vehicle performing the task quickly, and the task result needs multiple hops to return, which leads to excessively long transmission delay and transmission energy consumption. This group of experiments shows that the offloading scheme in this paper can make maximum use of the resources of the parking cluster to complete the offloading task and effectively control the execution cost of the task.

7. Conclusion

Inspired by the widespread distribution of parked vehicles with a large number of idle computing resources in urban areas, this paper proposes to make parked vehicles cooperate with each other to perform compute-intensive tasks in VEC. Specially, our approach could still work in the event of a lack of infrastructure or infrastructure failure due to disasters. Meanwhile, it could also solve the problem of resources constraints of the edge servers. For task scheduling and allocation, an optimization model is proposed to analyze the optimal amount of resources allocated to each offloading task by parking clusters, and a task offloading algorithm based on DRL is elaborately designed to determine the offloaded destination of each subtask. The simulation results show that the proposed algorithm can efficiently utilize the parked vehicle resources, improve the offloading success rate of the task, and reduce the execution cost of the task. In the future, the DRL algorithm and model structure will be optimized, and complex environmental factors such as radio interference will be considered to cope with the complex and changeable IoV environment.

Data Availability

The data used in this study are available from Hui Zhao (zhaoh@dgut.edu.cn) upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Natural Science Fund, China (nos. 61872083 and 61872081) and in part by the Science and Technology Project of Guangdong Province (no. 2020ZDZX3054).

References

[1] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary v2x technologies toward the internet of vehicles: challenges and opportunities," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 308–323, 2020.

[2] J. Nyambal and R. Klein, "Automated parking space detection using convolutional neural networks," in *Proceedings of the 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, IEEE, Bloemfontein, South Africa, November 2017.

[3] Q. Qi, J. Wang, Z. Ma et al., "Knowledge-Driven service offloading decision for vehicular edge computing: a deep reinforcement learning approachfloading decision for vehicular edge computing: a deep reinforcement learning approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4192–4203, 2019.

[4] X. Q. Pham, T. D. Nguyen, V. D. Nguyen, and E. N. Huh, "Joint node selection and resource allocation for task offloading in scalable vehicleassisted multi-access edge computing," *Symmetry*, vol. 11, no. 1, pp. 1–17, 2019.

[5] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular edge computing: architecture, resource management, security, and challenges," *ACM Computing Surveys*, vol. 55, no. 1, pp. 1–46, 2023.

[6] J. Dean, G. S. Corrado, R. Monga, C. Kai, and A. Y. Ng, "Large scale distributed deep networks," *Advances in Neural Information Processing Systems*, vol. 1, pp. 1223–1231, 2013.

[7] B. C. Ooi, W. Yuan, Z. Xie, M. Zhang, and A. Tung, "Singa: a distributed deep learning platform," in *Proceedings of the 23rd ACM international conference*, ACM, Brisbane, Australia, October 2015.

[8] R. Sutton and A. Barto, *Reinforcement Learning:An Introduction*, MIT Press, Cambridge, MA, USA, 1998.

[9] J. Zhang, S. Dai, and B. Zhang, "Joint offloading method based on task urgency in the vanets," *Journal of Electronic Measurement and Instrument*, vol. 34, no. 11, pp. 66–71, 2020.

[10] D. Tang, X. Zhang, M. Li, and X. Tao, "Adaptive inference reinforcement learning for task offloading in vehicular edge computing systems," in *Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, IEEE, Dublin, Ireland, June 2020.

[11] F. Lin, X. Lu, I. You, and X. Zhou, "A novel utility based resource management scheme in vehicular social edge computing," *IEEE Access*, vol. 6, pp. 66673–66684, 2018.

[12] Y. Wu, J. Wu, L. Chen, J. Yan, and Y. Luo, "Efficient task scheduling for servers with dynamic states in vehicular edge computing," *Computer Communications*, vol. 150, pp. 245–253, 2020.

[13] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *Proceedings of the IEEE Globecom Workshops*, IEEE, San Diego, CA, USA, December 2015.

[14] F. Dressler, G. S. Pannu, F. Hagenauer, M. Gerla, and O. Altintas, "Virtual edge computing using vehicular micro clouds," in *Proceedings of the 2019 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, Honolulu, HI, USA, February 2019.

[15] D. Ye, M. Wu, S. Tang, and Y. Rong, "Scalable fog computing with service offloading in bus networks," in *Proceedings of the 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*, IEEE, Beijing, China, June 2016.

[16] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11158–11168, 2019.

[17] W. Chen, Z. Su, Q. Xu, T. H. Luan, and R. Li, "Vfc-based cooperative uav computation task offloading for post-disaster rescue," in *Proceedings of the IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, IEEE, Toronto, ON, Canada, July 2020.

[18] X. Huang, R. Yu, D. Ye, L. Shu, and S. Xie, "Efficient workload allocation and user-centric utility maximization for task scheduling in collaborative vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3773–3787, 2021.

[19] Z. Wu, *Research on resource optimization and data sharing in vehicular edge computing*, Ph.D. dissertation, Guangdong University of Technology, Guangzhou, China, 2019.

[20] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked vehicle edge computing: exploiting opportunistic resources for distributed mobile applications," *IEEE Access*, vol. 6, pp. 66649–66663, 2018.

[21] X. He, H. Lu, Y. Mao, and K. Wang, "Qoe-driven task offloading with deep reinforcement learning in edge intelligent iov," in *Proceedings of the GLOBECOM 2020 - 2020 IEEE*

Global Communications Conference, IEEE, Taipei, Taiwan, December 2020.

[22] S. Xu, C. Guo, R. Q. Hu, and Y. Qian, "Learning iov in edge: deep reinforcement learning for edge computing enabled vehicular networks," in *Proceedings of the ICC 2021 - IEEE International Conference on Communications*, IEEE, Montreal, QC, Canada, June 2021.

[23] X. He, H. Lu, M. Du, Y. Mao, and K. Wang, "Qoe-based task offloading with deep reinforcement learning in edge-enabled internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2252–2261, 2020.

[24] S. M. A. Kazmi, S. Otoum, R. Hussain, and H. T. Mouftah, "A novel deep reinforcement learning-based approach for task-offloading in vehicular networks," in *Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, Madrid, Spain, December 2021.

[25] J. Dong, W. Wu, Y. Gao, X. Wang, and P. Si, "Deep reinforcement learning based worker selection for distributed machine learning enhanced edge intelligence in internet of vehicles," *Intelligent and Converged Networks*, vol. 1, no. 3, pp. 234–242, 2020.

[26] X. Xu, H. Li, W. Xu, Z. Liu, L. Yao, and F. Dai, "Artificial intelligence for edge service optimization in Internet of Vehicles: a surveyficial intelligence for edge service optimization in internet of vehicles:a survey," *Tsinghua Science and Technology*, vol. 27, no. 2, pp. 270–287, 2022.

[27] 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 Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[28] N. Liu, L. Ming, L. Wei, G. Chen, and J. Cao, "Pva in vanets: stopped cars are not silent," in *Proceedings of the 2011 Proceedings IEEE INFOCOM*, IEEE, Shanghai, China, April 2011.

[29] M. Abuelela and S. Olariu, "Taking vanet to the clouds," in *Proceedings of the Eighth International Conference on Advances in Mobile Computing and Multimedia*, pp. 7–21, ACM, Paris, Francevol, November 2010.

[30] N. Liu, M. Liu, G. Chen, and J. Cao, "The sharing at roadside: vehicular content distribution using parked vehicles," in *Proceedings of the 2012 Proceedings IEEE INFOCOM*, pp. 2641–2645, IEEE, Orlando, FL, USA, March 2012.

[31] P. Tg, *Wireless Access in Vehicular Environments (Wave)*, IEEEP, Bristol, England, 2006.

[32] A. Adiv and W. Wang, "On-street parking meter behavior," *Transportation Quarterly*, vol. 41, no. 3, pp. 1–37, 2013.

[33] J. W. Liu, F. Gao, X. L. Luo, and D. O. Automation, "Survey of deep reinforcement learning based on value function and policy gradient," *Chinese Journal of Computers*, vol. 42, no. 6, pp. 1406–1438, 2019.