Hindawi
Scientific Programming
Volume 2022, Article ID 1163177, 15 pages
https://doi.org/10.1155/2022/1163177

Research Article

CSO-DRL: A Collaborative Service Offloading Approach with Deep Reinforcement Learning in Vehicular Edge Computing

Yuze Huang, Yuhui Cao, Miao Zhang, Beipeng Feng, and Zhenzhen Guo

School of Information Science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China

Correspondence should be addressed to Yuze Huang; huangyz@cqjtu.edu.cn

Received 3 April 2022; Revised 5 August 2022; Accepted 9 August 2022; Published 5 September 2022

Academic Editor: Ying Chen

In vehicular edge computing, vehicles move along the road and request the services from the nearest edge servers with low latency. Due to the limitation of computation capacity of vehicular devices, the services should be offloaded on RSUs equipped with edge servers to provide service with low latency. Noticed that the location of service offloading may affect the service requesting delay directly, and it may exist some interrelationship between interacting services; all of these are rarely considered in recent studies. To address such problems, we propose a collaborative service offloading approach with deep reinforcement learning in vehicular edge computing named CSO-DRL. Our approach first divides the road segments by k-means-based algorithm through analyzing the trajectory data of vehicles, and then the offloading location is determined by observing the vehicle running status. Secondly, the interacting services are discovered by a parallel frequent pattern-based algorithm efficiently. Furthermore, the collaborative service offloading algorithm is presented by the DDPG model for offloading the interacting services, which can minimize the service requesting delay and data communication delay between interacting services. Finally, the efficiency of the algorithm is evaluated by real-world data-based simulation experimental evaluations. The results show our algorithm can obtain a lower delay than other baseline algorithms in searching for the optimal service offloading strategy.

1. Introduction

With the development of intelligent transportation technologies, the internet of vehicles (IoV) has raised public attention and applications in the urban transportation system, which can be regarded as the evolution of vehicular ad hoc networks. In IoV, the vehicles equipped with intelligent devices are capable of vehicles to everything communication (V2X) through the wireless network [1]. Through the V2X communication, the vehicles can realize the information interaction among vehicles, roadside units (RSUs), and humans to assist the decision of auto-driving and avoid the traffic accidents [2]. In the IoV-based intelligent transportation system, the data including vehicles status information and traffic information are collected by numerous sensors for analysis and computation [3]. Since the vehicles have limited computation capability to execute complex computation tasks, the most direct approach is transmitting the traffic data to a cloud data center for computing. Generally speaking, a cloud data center is far away from the vehicles, which may bring some problems such as network congestion, time-consuming, and data privacy leaking [4]. With the complexity of urban traffic, the response time of cloud service will be increasingly difficult to satisfy the real-time decision for auto-driving. To solve such problems, edge computing regarded as the supplement of traditional cloud computing is used in IoV, which can efficiently provide service to users with low latency [5, 6].

In edge computing, the services are offloaded on edge servers (ESs) in close proximity to users, which mainly undertake the capacity of data computing and most services executions. While the intelligent devices are only responsible for the services of raw data preprocessing and few non-computation-intensive services executions, and then only a small amount of intelligent computation services are delivered to remote cloud servers for execution [7–9]. As an efficient approach, service offloading (also called computation offloading) can address the problems of insufficient

devices and the high latency for cloud services, which has raised public attention in edge computing [10–12]. During the service offloading, the users' experience may be greatly affected by service requesting delays and the energy consumption of intelligent devices [13, 14]. In such conditions, some offloading approaches are produced to minimize the latency and energy, but some defections are revealed in IoV service offloading.

In vehicular edge computing, when the vehicles move along the roads, the vehicles request the services from the nearest edge server to gain low-latency services. Due to the coverage limitation of the RSUs equipped with edge servers, when the vehicles reach the boundaries of edge servers, the edge servers must be switched to guarantee the continuity of service, so the selection of edge servers may influence the delay of service requesting to a large extent [15]. Thus, the speed and location of vehicles must be considered in service offloading. Moreover, the interrelationship between services is another factor that must be considered in service offloading. Many services collaborate with different services to complete a certain complex business goal, which may produce lots of transmission data among the interacting services. If the interacting services are offloaded separately, the data communication delay between them will be increased. As the number of data increases exponentially, the communication delay between interacting services becomes a nonignored problem for service offloading. Throughout the studies for services offloading [16, 17], although some works have a focus on the collaborative problem of cloud and edge servers [13, 18], a few studies considered the collaboration of services. In summary, the mobility of vehicles and the interrelationships between services bring new challenges for service offloading.

To address the mentioned challenges, a collaborative service offloading approach named CSO-DRL is proposed to search optimal offloading strategy for interacting services. In our approach, an algorithm for offloading location selection is presented to consider the impact of speed and location of vehicles. This algorithm divides the road segments by analyzing the trajectory of vehicles, and then the location of offloading is determined by the status of vehicles. Furthermore, the interacting services are discovered based on a parallel frequent pattern mining algorithm. Finally, we constructed the system models and put forward a collaborative offloading algorithm based on deep reinforcement learning to offload the interacting services while considering the optimization problem for minimizing the service requesting delay and data communication delay between interacting services. Specifically, the contribution of this paper is threefold as follows:

(i) An offloading location decision algorithm is presented by analyzing the trajectory of vehicles, which divides the road segments with the k-means clustering algorithm first, and then the offloading location is determined via the status of vehicles.

(ii) In order to reduce the service requesting delay and data communication delay, the interacting services are discovered by a parallel frequent pattern-based algorithm first, and then the offloading models are constructed by analyzing the communication environments and the computation models.

(iii) The collaborative service offloading algorithm with deep deterministic policy gradient is put forward to obtain the optimal service offloading strategy. This algorithm reveals the interrelationship of interacting services by minimizing the service requesting delay and data communication delay in service offloading.

The rest of this paper is organized as follows. Section 2 introduces the related work of this research. Section 3 presents an overall framework of the collaborative service offloading, and then an offloading location decision algorithm and the system model based on the interacting services discovered are produced in this section. Furthermore, a collaborative service offloading algorithm with deep deterministic policy gradient (DDPG) model is proposed to offload the interacting services in Section 4, which can realize the optimization of minimizing the service requesting delay and data communication delay between interacting services. Finally, the experimental evaluations are conducted in Section 5, and then we conclude this paper in Section 6.

2. Related Work

With the advent of the internet of everything, the data produced and collected by the various sensors in IoT are experiencing a steep rise [19–21]. Though the traditional cloud computing paradigm can solve the inefficient computation capacity of intelligent devices, it can also bring some problems, such as high latency, network congestion, and data privacy leaking. In order to address such problems, a new computing paradigm named edge computing is introduced [22]. In edge computing, the data are preprocessed by intelligent devices, and the edge servers nearby users undertake the majority of tasks for computing data and executing the services, while the remote cloud servers are only responsible for the training of deep neural network models [23, 24]. Due to the resource-constrained of end devices, the services should be offloaded on edge servers or cloud servers [25].

Service offloading (also called computation offloading) aims to address the inefficient storage capacity, computing resources, and energy of numerous devices at the user level [26]. In the depth of offloading researches, the service offloading mainly contains offloading decisions and offloading optimization. Offloading decision is mainly studies on whether the service should be offloaded or not, and which service should be offloaded. The main studies can be divided into two types, which are 0/1 offloading and partial offloading [27, 28]. 0/1 offloading means the service can be executed either locally or on the other devices fully, which cannot archive the optimal delay. In partial offloading, the service can be split into a quantity of subservices. The offloading strategy is responsible for determining which subservices should be executed locally and which part should be executed on edge or cloud. During the service offloading,

latency and energy are the main factors considered by researchers. For time-sensitive services, for example, VR gaming and multimedia services, the researchers mainly focus on how to reduce the latency [16]. Some studies have examined the optimization of latency and energy together, which is difficult to search for the balance optimal result in latency and energy consumption [29]. Some studies noticed that deep reinforcement learning (DRL) is a better way to search for the optimal offloading strategy than the traditional optimization algorithm. Since then, the task offloading approaches with deep reinforcement learning are increasingly, and some efficient approaches based on various DRL algorithms are introduced to get the optimal latency or energy [30–32]. Besides these studies, the load balance brings new challenges for resource allocation; some efficiency approaches are presented to address these challenges [33, 34].

Internet of vehicles (IoV) is a typical IoT, which connects vehicles and RSUs through wireless communication technologies. As the number of vehicles increases exponentially, as a complex system, the cloud computing-enabled IoV cannot meet the real-time demands for vehicular service, while vehicular edge computing (VEC) can provide lower data process delay and a more stable network transmission rate. Some studies have introduced for VEC framework and task scheduling in VEC [35, 36]. In IoV, since the vehicles move along the roads, the task offloading in VEC is a dynamic process. The new challenges for task offloading in VEC have been produced, and some efficient approaches have been introduced. Wang et al. [37] put forward a solution for task offloading in the fog-based IoV system, which decomposes the offloading optimization problem into two subproblems and schedules the traffic queue between different edge servers, the experimental evaluation results show that the algorithm can obtain the lower latency than other algorithms. In [38], the privacy conflicts of offloading are considered, and the NSGA-II algorithm is designed to solve the multiobjective optimization for reducing the service computation delay and the energy consumption of devices while guarding against privacy conflicts. He et al. [39] considered the QoE of services, and then a DRL-based algorithm is introduced to save energy consumption and improve the QoE value.

In the sight of these studies, few works have examined the location selection and the data communication delay between interacting services, which are the nonignored factors for service offloading. In view of the importance of location decisions for offloading and the interrelationships between services, this paper proposes a collaborative service offloading approach with deep reinforcement learning to address the above challenges.

3. System Model and Problem Formulation

In this section, we present the framework of the collaborative service offloading approach first, and then the algorithm for offloading location selection is produced. Finally, the system model, contained communication model, and computation model for our approach are put forward to formulate the offloading problem, which should be solved in the following sections.

3.1. Framework of Collaborative Service Offloading. In vehicular edge computing, the vehicular devices collect the data from numerous sensors and then transmit the preprocessed raw data to RSUs equipped with edge servers. The RSUs receive the data and execute the service to send back the decisions to vehicle users. Due to the limitation of computation capacity of vehicular devices, the computing-intensive services should be offloaded to the edge servers, and the vehicular devices only undertake the task of preprocessing the raw data and execute the noncomputing-intensive services. Thus, how to offload the services with low latency is important for vehicular edge computing.

As Figure 1 shows, the vehicles move along the road and communicate with RSUs through a wireless network. Due to the coverage limitation of RSU, the vehicle cannot request the service from a fixed RSU. When the vehicle moves at the boundary of RSU, the network should be handoff and the vehicle accesses another RSU nearby the vehicle. Therefore, the mobility of the vehicles may affect the service requesting a delay. In the sight of the studies for service offloading, most of them focus on the optimization of service requesting a delay, and few of them study the location selection for service offloading. In order to obtain a lower delay, during the process of service offloading, the location selection for offloading is an important problem, which must be considered. Besides offloading location selection, the interrelationship of services is another nonignored problem for service offloading. We noticed some services may interact with each other to complete a complex business goal, which may produce lots of transmission data among them. If the interacting services are offloaded separately, the data communication delay between them will be increased. Thus, the data communication between these interacting services is a nonignored factor.

In order to address the problems mentioned above, we propose a collaborative service offloading approach with deep reinforcement learning named CSO-DRL to offload the interacting services. As Figure 2 shows, the k-means-based road segments dived algorithm is presented first, and then the suitable RSU for offloading is selected based on the mobility of vehicles. Furthermore, the interacting services are discovered by analyzing the service requesting log to reveal the relationship between interacting services. Finally, a DDPG-based collaborative service offloading algorithm is presented for offloading the interacting services, which can optimize the service requesting delay and data communication delay between interacting services, the detailed process can be expressed as follows.

Step 1. The trajectory data of vehicles are collected, and then a k-means-based road segments dividing algorithm is put forward to obtain the road segments information. For each segment, an RSU is deployed as the location of service offloading in this segment.

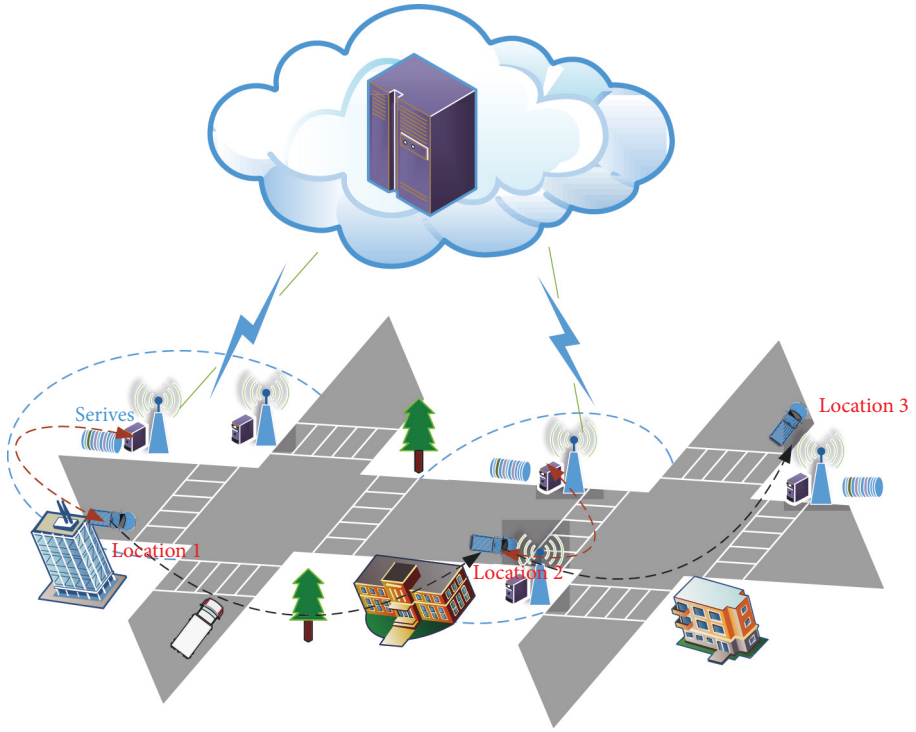

Figure 1: Vehicular edge computing scenario.

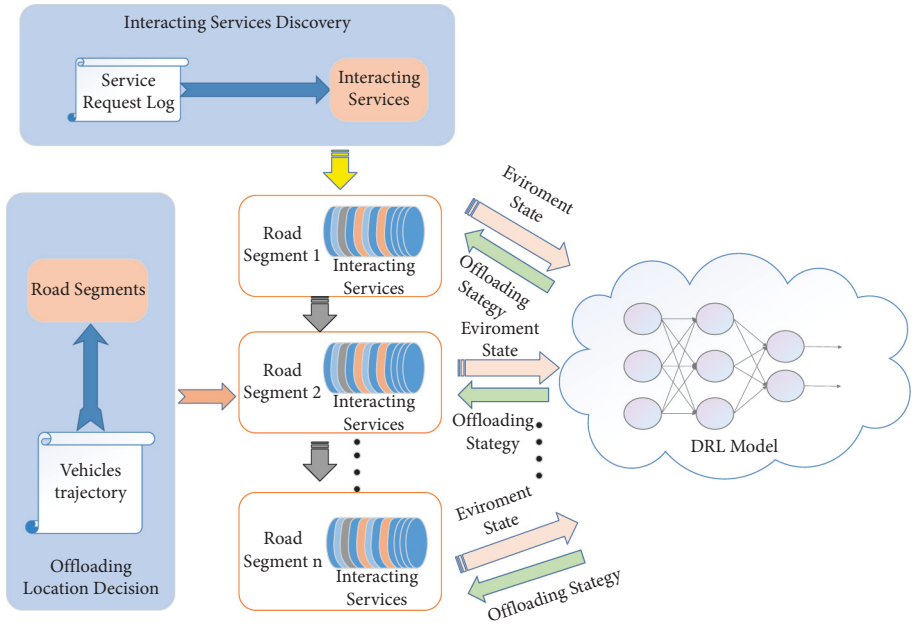

Figure 2: Framework of CSO-DRL in VEC.

Step 2. A vehicle is randomly selected to observe the running status. When the vehicle reaches the boundary of this segment, the RSU should not be switched to reduce the time consumption produced by network handoff; otherwise, another RSU is selected for offloading.

Step 3. The service requesting log is exacted to discover the interacting relationship for services. A parallel frequent pattern-based algorithm is presented to mine the frequent service 2-itemsets for interacting services, which can be found in our previous works [40, 41].

Step 4. A DDPG-based collaborative service offloading algorithm is proposed to offload the interacting services, which can optimize the service requesting delay and data communication delay between interacting services. In this

Input. Data set of vehicles trajectory V_t; number of road segments k
Output. Road segments $R = \{r_1, r_2, \ldots, r_k\}$

(1) Select the k samples as the initial vector $\{u_1, u_2, \ldots, u_k\}$.
(2) **while** True **do**
(3) num = 0
(4) **for** $i = 0, 1, \ldots, k$ **do**
(5) $C_i = \Phi$
(6) **end for**
(7) **for** $j = 1, 2, \ldots, n$ **do**
(8) Compute the distance of v_j from each vector $u_i\,(1 \le i \le k)$ according to equation (1).
(9) Determine the clusters' label according to the nearest vector, $\tau_j = \text{argmin}_{i \in \{1,2,\ldots,k\}} \mathrm{d}_{ji}$
(10) $r_{\tau_i} = r_{\tau_i} \cup v_j$
(11) **end for**
(12) **for** $i = 1, 2, \ldots, k$ **do**
(13) $u_i' = 1/C_i \sum_{x \in C_i} x$
(14) **if** $u_i' \neq u_i$ **then**
(15) $u_i = u_i'$
(16) **else**
(17) num + +
(18) **end if**
(19) **end for**
(20) **if** num = k **then**
(21) break
(22) **end if**
(23) **end while**
(24) **return** $R = \{r_i, r_2, \ldots, r_k\}$

Algorithm 1: Algorithm for dividing road segments.

algorithm, the training model is deployed on the cloud servers to receive the vehicle running environment state, and then the optimal offloading strategy is obtained and performed after numerous interaction computing.

3.2. Offloading Location Decision. In VEC, the mobility of vehicles may affect the service requesting a delay. Due to the coverage limitation of each RSU, the vehicles access the different RSUs when the vehicles reached different locations. The selection of offloading location is an important problem, which should not be ignored during the service offloading. In this section, we put forward a service offloading location selection approach to assist service offloading action decisions.

In reality, it is difficult to deploy the edge servers on all the RSUs, as it is a time-consuming and resource-consuming process. Hence, a k-means-based algorithm is introduced to divide the road segment and select the RSU to deploy the edge server as the offloading location. In this algorithm, the trajectory data of vehicles are collected as the input of our proposed algorithm, and then the segments of the road are divided by a k-means clustering-based algorithm. In our algorithm, the distance of time t_i and t_j can be expressed as d_{ij}, which is computed by the Euclidean distance as follows:

$$\mathrm{d}_{ij} = \sqrt{\left(\mathrm{lon}_{t_i} - \mathrm{lon}_{t_j}\right)^2 + \left(la_{t_i} - la_{t_j}\right)^2}, \quad (1)$$

where lon_{t_i} denotes the longitude of the vehicles in time episode t_i and the latitude of the vehicle in time episode t_i can be denoted as la_{t_i}.

The mobility of vehicles may produce the dynamic of service requesting delay. During the services offloading, the main purpose is deciding the strategy of offloading and optimizing the service requesting delays, but the location decision is also a nonignore problem, which may affect the time delay. Thus, we put forward an algorithm for dividing road segments based on the k-means clustering algorithm [42]. Details of this algorithm can be found as Algorithm 1 shows.

With the road segments divided, we select an RSU in each segment to equip with the edge server, which mainly undertakes the task of receiving the data transmitted from vehicles and executing the computing-intensive services requested by the vehicles running in the area of this road segment. For each segment, a vehicle is randomly selected as the observed object. When the vehicle moves in the segment and the corresponding edge server is selected as the offloading location. With the running of the vehicle, the vehicle may reach the boundary of the segment; the edge server should not be switched to reduce the time consumption produced by the network handoff. In each road segment, the vehicle is observed, and the environment states are transferred to the collaborative offloading model, which is deployed on the cloud server. After the iteration computing, the offloading actions are obtained and sent to the vehicle for performing. If the vehicle moves out of the coverage of the current segment, the edge server must be switched and another RSU will be selected as the offloading location, and then the training steps are adopted as previous actions.

Table 1: List of important notations.

Notations	Description
H	Channel power gain
P	Transmission power
σ^2	Noise power
W	Transmission bandwidth
C_i	Computation capacity for executing service s_i
f_{dev}	Computation capacity of devices
f_{edge}	Computation capacity of edge servers
D_i^{off}	Data transmission size from devices to edge servers
C_i^{off}	Computation capacity for executing the offloaded service
C_i^{local}	Computation capacity for executing the service deployed on the devices
D_j^{re}	Data communication size between interacting services

3.3. System Model. In this section, the system model, contained communication model, and computation model are presented to formulate the problem of our approach.

Compared with previous studies, our approach offloads the interacting services to optimize the service requesting delay and data communication delay. In our previous works [40, 41], a parallel frequent pattern-based algorithm is presented to discover the interacting services. In this algorithm, the service requesting log denoted as $EL \triangleq \{\text{cid}, Ts, \prod\}$ is adapted as the input of the interacting services discovery algorithm, where cid denotes the nonempty set of service requesting case ID and the Ts denotes the finite serial of time stamps. The $\prod$ is the finite serial of services. We noticed that a composition service is composed of a series subservices to complete a certain complex business goal, which can be denoted as $S_i = \{s_1, s_2, \ldots, s_n\}$. In these subservices, it may exist the interrelationships between subservices. In our algorithm, the fine-grained frequent 2-itemsets of services (also called interacting services pairs) are discovered by a parallel frequent pattern-based algorithm directly, which can reveal the interrelationships between them. The details can be found in our previous works [40, 41]. Finally, the interacting services are offloaded by our CSO-DRL algorithm, which can be found in Section 4.

3.3.1. Communication Model. In order to make this paper clear, we give the important notations in our paper first, which can be found in Table 1.

Since the signal will be affected Gaussian white noise during the transmission process, we construct the communication model with Gaussian white noise to reflect the real communication channel. Here, we assume the transmission power is fixed, which can be denoted as P. The standard path loss propagation index can be denoted as θ. In IoV, the vehicles move along the road and access the service from the nearest edge server at RSU; thus, the distance between the vehicle and edge server in a time slot t can be denoted as d_t. The signal-to-noise ration (SNR) of the communication channel can be expressed as follows:

$$\text{SNR} = \frac{\text{PHd}_t^{-\theta}}{\sigma^2}, \tag{2}$$

where σ^2 is the power of Gaussian white noise and H denotes the channel gain. So the transmission rate can be given as follows:

$$R = W\log(1 + \text{SNR}), \tag{3}$$

where W denotes the channel bandwidth.

3.3.2. Computation Model. In this paper, we considered the interrelationships between interacting services. If some services are offloaded to remote cloud servers, which may produce lots of data communications between them. In the VEC system, due to the computation limitation of vehicular devices, the vehicular devices offload the portion services to RSUs equipped with edge servers, while the remaining services are locally executed at devices. The RSUs will receive the preprocessed data and execute the computing-intensive services, and then the decision will be sent back to vehicular users. Hence, the partial service offloading manner is adapted in each time slot T. The offloading manner contains three parts, which are locally computing, partial offloading on edge servers, and fully edge computing. Next, we will give the service delay model for these three parts:

(A) Locally computing. In our approach, the services without high computation capacity can be computed on the devices directly. So the computation delay can be given as follows:

$$T_i^{\text{local}} = \frac{C_i}{f_{\text{dev}}}, \tag{4}$$

where f_{dev} denotes the computation capacity of the vehicular device and the computation capacity for executing the service s_i can be denoted as C_i.

(B) Partial offloading. Due to the computation capacity limitation of the vehicular devices, the services are offloaded by a partial offloading strategy. In these composition services, some subservices are executed on vehicular devices, but some computation-intensive subservices should be offloaded on RSUs equipped with the edge servers; therefore, the data produced by these services should be transmitted to the edge servers. The time delay can be expressed as follows:

$$T_i^{\text{par}} = \max\left(T_i^{\text{local}}, 2T_i^{\text{tran}} + T_i^{\text{edge}}\right), \tag{5}$$

where T_i^{tran} denotes the transmission delay for data transmitting and T_i^{edge} denotes the computation delay for the edge computing part. These can be found as follows:

$$\begin{aligned} T_i^{\text{tran}} &= \frac{D_i^{\text{off}}}{R}, \\ T_i^{\text{edge}} &= \frac{C_i^{\text{off}}}{f_{\text{edge}}}, \end{aligned} \tag{6}$$

where D_i^{off} denotes the transmission data to edge, C_i^{off} denotes the computation capacity for executing the offloaded services, and f_{edge} denotes the computation capacity of the edge server. Thus, the delay for partial offloading can be expressed as follows:

$$T_i^{\text{par}} = \max\left(\frac{C_i^{\text{local}}}{f_{\text{dev}}}, 2\frac{D_i^{\text{off}}}{R} + \frac{C_i^{\text{off}}}{f_{\text{edge}}}\right), \tag{8}$$

where C_i^{local} denotes the computation capacity for executing the services deployed on devices.

(C) Fully offloading. For the computation-intensive services, all of these should be offloaded on edge servers; thus, the delay can be expressed as follows:

$$T_i^{\text{full}} = 2\frac{D_i}{R} + \frac{C_i}{f_{\text{edge}}}, \tag{9}$$

where D_i is the data for executing these services and the C_i denotes the computation capacity for executing the service.

3.4. Problem Formulation. Based on the description of the system problem in the above section, the delay can be expressed as follows:

$$T_i = \begin{cases} \dfrac{C_i}{f_{\text{dev}}}, \\ \max\left(\dfrac{C_i^{\text{local}}}{f_{\text{dev}}}, 2\dfrac{D_i^{\text{off}}}{R} + \dfrac{C_i^{\text{off}}}{f_{\text{edge}}}\right), \\ 2\dfrac{D_i}{R} + \dfrac{C_i}{f_{\text{edge}}}. \end{cases} \tag{10}$$

Besides the system computation model produced in the above section, we noticed that the data communication delay between interacting services is the nonignore problem. In this system, we assume the number of interacting service pairs is n, and the number of interacting service pairs executed on vehicular devices is a; thus, the latency of the system can be described as the following equation:

$$T_i = \begin{cases} \dfrac{C_i}{f_{\text{dev}}}, \\ \max\left(\dfrac{C_i^{\text{local}}}{f_{\text{dev}}}, 2\dfrac{D_i^{\text{off}}}{R} + \sum_{j=1}^{n-a}\dfrac{D_j^{re}}{R} + \dfrac{C_i^{\text{off}}}{f_{\text{edge}}}\right), \dfrac{D_j^{re}}{R} + \dfrac{C_i}{f_{\text{edge}}}. \\ 2\dfrac{D_i}{R} + \sum_{j=1}^{n} \end{cases} \tag{11}$$

where D_j^{re} denotes the data communication size between the interacting services.

In our approach, we let $\lambda \in [0, 1]$ denotes the ratio of service offloading. So $D_i^{\text{off}} = \lambda * D_i$. Finally, the latency of our system can be given as follows:

$$\begin{aligned} T_i^{\text{sum}} = \max\Bigg(& (1-\lambda)\frac{C_i^{\text{local}}}{f_{\text{dev}}}, \\ & \lambda\left(2\frac{D_i}{R} + \frac{C_i^{\text{off}}}{f_{\text{edge}}}\right) + \sum_{j=1}^{n-a}\frac{D_j^{re}}{R}\Bigg). \end{aligned} \tag{12}$$

In this paper, the purpose is to optimize the latency, which combines the service requesting delay and data communication delay between interacting services. Thus, the optimization function can be given as follows:

$$\text{Min}\left(T_i^{\text{sum}}\right). \tag{13}$$

Based on the analysis of the environments, the constraint conditions can be expressed as follows:

$$\begin{aligned} C_i^{\text{local}} &\le C_i \le f_{\text{dev}}, \\ C_i^{\text{off}} &\le C_i \le f_{\text{edge}}. \end{aligned} \tag{14}$$

4. Algorithm for DRL-Based Collaborative Service Offloading

In this section, an algorithm of collaborative service offloading based on DDPG is presented to offload the interacting services. The details of this algorithm can be described in the next contents.

Deep reinforcement learning is the algorithm mixed with deep learning and reinforcement learning. The reinforcement learning model is an iteration process, which can be divided into environment and agent. For each step, the agent observes the state s_t and takes actions according to the current policy π. When the state of the environment changes to s_{t+1}, the reward value r_t is received in the next step. The process for state transition of the environment and the reward computing can be constructed as a Markov process; thus, the probability and reward of the state transition only depend on the environment state s_t and action a_t. The agent receives these quantities according to the decision policy and interacts with the environment, and then the state is back propagated for maximizing the expected reward r_t.

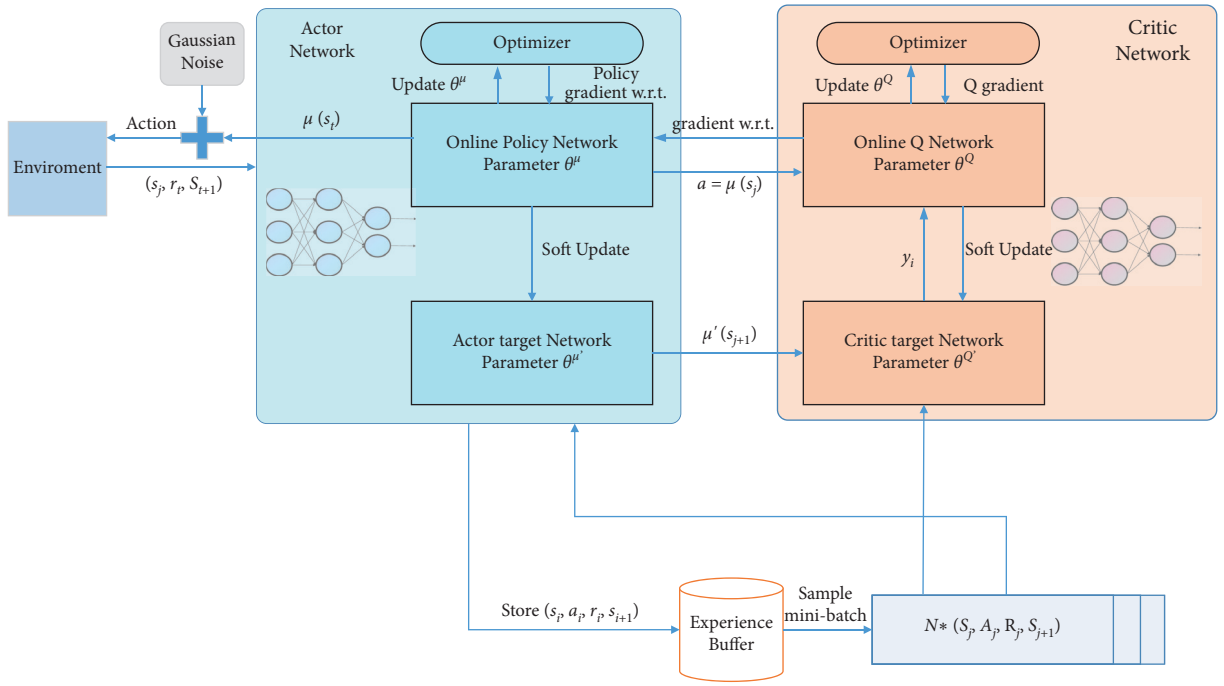

Figure 3: Deep deterministic policy gradient (DDPG) model.

The DDPG algorithm is a model-free nonpolicy action-critic algorithm based on DNN ideas, which can learn policies in the continuous action spaces [43]. The action-critic algorithm consists of the policy function and the q-value function, where the policy function undertakes the task to generate actions like an actor and the q-value function undertakes the task to evaluate the actor's performance and direct the actor's following actions.

As Figure 3 shows, the structure of DDPG is similar to actor-critic algorithm [44]. DDPG can be divided into actor network and critic network, which continues the idea of a fixed target network such as DQN [45]. First, the actor network generates $\mu(s_i)$ after the training step, and then the action space with Gaussian noise is constructed. After performing the a_t in the environment, the agent observes the next state and the immediate reward. Second, the transition is stored in the experience replay butter, and N transitions are selected by the model to make up a minibatch, which can be transferred into the actor network and critic network. With the minibatch, the actor target network outputs the action to the critic target network, and then the critic network calculates the target value. Furthermore, the critic network will be updated, and the actor network gives the minibatch action to the critic network to achieve the action's gradient. Finally, the DDPG agent updates the critic target network and the actor target network. The details can be found in [43].

Compared with the traditional DDPG algorithm, we adapt a state normalization algorithm to preprocess the observed states and add the behavior noise to the environments, which takes the difference between the maximum and minimum of each variable as the scaling factor [46]. The state normalization algorithm can solve the problem of the magnitude difference of input variables. According to the previous work [46], we can find that after normalizing the state and adding the behavior noise, our algorithm can obtain a better performance than the traditional algorithm.

4.1. State Space. In our approach, the users' requirements, network conditions, and channel and resource conditions are used as the state of the system, which will produce the state space explosion problems. In order to simplify the system, we only consider the offloading requirement in a single user scenario, which can reduce the dimension of the state space increasingly and make the algorithm easy to converge. So the state space can be expressed as follows:

$$S_t = \left(q(t), p(t), D_{\text{remain}}(t), D(t), D_{re}(t)\right), \tag{15}$$

where $q(t)$ represents the location information of the edge server and $p(t)$ denotes the location information of the vehicle. The data size of the remaining subservices is D_{remain}, and $D(t)$ denotes the task size requested by the vehicle; finally, $D_{re}(t)$ denotes the data communication size between the interacting services.

4.2. Action Space. We define the action space as the size of the offloading task, which can be expressed as follows:

$$a_t = (k(t), \lambda(k)), \tag{16}$$

where $k(t)$ indicates the edge server that undertakes the task for executing the services and $\lambda(k)$ denotes the service offloading ratio.

4.3. Reward Function. In this paper, the main purpose of our algorithm is to minimize the latency, which contains the service requesting delay and communication delay. The r_t is

Input. Training data set I; critic learning rate λ_c; actor learning rate λ_a; discount factor γ; soft update factor τ; experience replay buffer B_m; minibatch size N; Gaussian noise n_i, current state $q_i, p_i, D_{\text{remain}}(i), D(i), f_i$; state parameters with normalized $\gamma_{D_{\text{remain}}}, \gamma_{D_{ue}}, \gamma_x, \gamma_y$
Output. Reward r_i

(1) Initialize the critic network and the actor network with weights θ^Q and θ^μ
(2) Initialize the target network with weights $\theta^{Q'} \longleftarrow \theta^Q$, $\theta^{\mu'} \longleftarrow \theta^\mu$
(3) Empty the experience replay buffer B_m
(4) **for** episode = 1 to M **do**
(5) Reset the simulation parameters and receive the initial observation state s_1
(6) **for** $t = 1, 2, \ldots, T$ **do**
(7) Initialize s_t as the current state
(8) Normalize state $s_t' \longleftarrow s_t$, and obtain the feature vector $\phi(s')$
(9) Get the action $a_t = \pi\theta^\mu(\phi(s')) + n_t$
(10) Execute the action to get the reward re_t, and observe the next state s_{t+1}
(11) Normalize state $s_{t+1}' \longleftarrow s_{t+1}$
(12) **if** B_m is not full **then**
(13) Store $(s_t', a_t, r_t, s_{t+1}')$ to B_m
(14) **else**
(15) Randomly replace a transition in B_m with $(s_t', a_t, r_t, s_{t+1}')$
(16) Randomly sample a mini-batch from B_m
(17) $y_j = r_j + \gamma Q'(s_{j+1}', \mu'(s_{j+1}'|\theta^{\mu'}), \theta^{Q'})$
(18) $L(\theta^Q) = 1/N \sum_{j=1}^{N} (y_j - Q(s_j', (a_j|\theta^Q))^2)$
(19) Update θ^μ by the sampled policy gradient
(20) Soft update the critic target network and actor target network
(21) **end if**
(22) **end for**
(23) **end for**
(24) Get $\mu(s'|\theta^\mu)$
(25) **for** episode = 1, 2, ..., E **do**
(26) Reset the VEC parameters and receive the initial observation s_1
(27) **for** $t = 1, 2, \ldots, T$ **do**
(28) Normalize state $s_t' \longleftarrow s_t$
(29) $a_t = \mu(s'|\theta^\mu)$
(30) Execute the action a_t and get the reward r_t
(31) **end for**
(32) **end for**

Algorithm 2: Algorithm for collaborative service offloading with DDPG.

defined as the reward when the action a_t is performed at state s_t. In order to reduce the system latency while guaranteeing the sufficient computation resources of devices, the reward function can be given as follows:

$$r_t = -T_i^{\text{sum}}. \tag{17}$$

The action value function is used to describe the expected return of policy π at time episode t in the reinforcement learning algorithm, which can be expressed as follows:

$$\begin{aligned} Q^\pi(s_t, a_t) &= E_{r_t, s_{t+1}} \\ &\sim E\left[r_t(s_t, a_t) + \gamma E_{a_{t+1} \sim \pi}\left[Q^\pi(s_{t+1}, a_{t+1})\right]\right]. \end{aligned} \tag{18}$$

Therefore, if the update of the target policy is continuous, the target policy can be described as a function μ: $A \leftarrow S$, which can be expressed as follows:

$$\begin{aligned} Q^\pi(s_t, a_t) &= E_{r_t, s_{t+1}} \\ &\sim E\left[r_t(s_t, a_t) + \gamma E_{a_{t+1} \sim \pi}\left[Q^\pi(s_{t+1}, a_{t+1})\right]\right]. \end{aligned} \tag{19}$$

Algorithm 2 demonstrates the details of our CSO-DRL algorithm, which is an iteration process. During the training process, the parameters of the critic network and the actor network are trained with the iteratively update process, and then the trained actor network parameters are adopted to offload the interacting services.

5. Experimental Evaluation

In this section, we conduct experiments to evaluate the performance of our algorithm. First, all the simulation parameters are indicated, and then the efficiency of our algorithm is conducted to compare with other baseline algorithms in the same simulation environment.

5.1. Experiment Setup. To ensure the practicability of the algorithm, we use a real-world data set of taxi trajectories in Shanghai (https://cse.hkust.edu.hk/scrg/) to evaluate the experiment in this paper. This data set records the GPS trajectory data on February 20, 2007, in Shanghai, which contains the longitude and latitude of 4316 taxies. Here, we

Table 2: Simulation parameters.

Parameters	Numerical value	Unit
Channel power gain, H	−50	dB
Transmission bandwidth, W	100	MHz
Noise power, σ^2	−100	dBm
Devices computation capacity, f_{dev}	0.8	GHz
Edge computation capacity, f_{edge}	3	GHz
Task size with random values, D_i	< 60	Mbit
Data communication size between interacting services, D_j^{re}	$< M_i$	Mbit

conduct the experiment in the real world to obtain the road segments and determine the location of service offloading. Furthermore, the simulation experiments are conducted to compare the latency with other baseline algorithms, which are described as follows:

(i) Local-only: executing all services locally

(ii) Offload-only: offloading all services to RSUs deployed with edge servers

(iii) AC: actor-critic-based service offloading algorithm [44]

(iv) DQN: DQN-based service offloading algorithm [45]

We set the channel power gain $H = -50$ dB when the distance is 1 m, and the transmission bandwidth is $W = 100$ MHz. The noise power accepted without signal blocking is $\sigma^2 = -100$ dBm. The computation capacity of the vehicular devices and edge servers are set to $f_{\text{dev}} = -0.8$ GHz and $f_{\text{edge}} = 3$ GHz, respectively. We set the distance between the vehicular devices and the edge servers d_t randomly. The size of the task is randomly assigned, and $f_i = \epsilon D_i$ is the required computation resource, where ϵ is set at 0.5 Gcyles/MB. The detailed simulation parameters are listed in Table 2.

Since the data set records the location information of the vehicles every 60 s, so the time slot is set $t = 60$ s, and the total time is 24 hours. For fairness, we conduct all the algorithms in the same simulation environment. During the time slots, all the services are offloaded continually by our CSO-DRL algorithm, and then the delay is conducted to evaluate the efficiency of our algorithm.

5.2. Experimental Results. The hyperparameters in our algorithm affect the overall performance of the algorithm seriously. The experiments are conducted to determine the optimal values of these hyperparameters in our algorithm. First, we conduct experiments to select the optimal hyperparameters. Figure 4 shows the convergence of the CSO-DRL algorithm under different learning rates. In the DDPG model, there are two training networks, which are the actor network and the critic network. We set the learning rates of the critic network and actor network separately. According to the training, we find that the algorithm cannot converge when the learning rate is $\lambda_a = 0.1$ and $\lambda_c = 0.2$. When the learning rate is $\lambda_a = 0.001$ and $\lambda_c = 0.002$, the algorithm can get a better convergence value. Therefore, the optimal learning rate of actor network $\lambda_a = 0.01$, and the learning rate of critic network $\lambda_c = 0.02$.

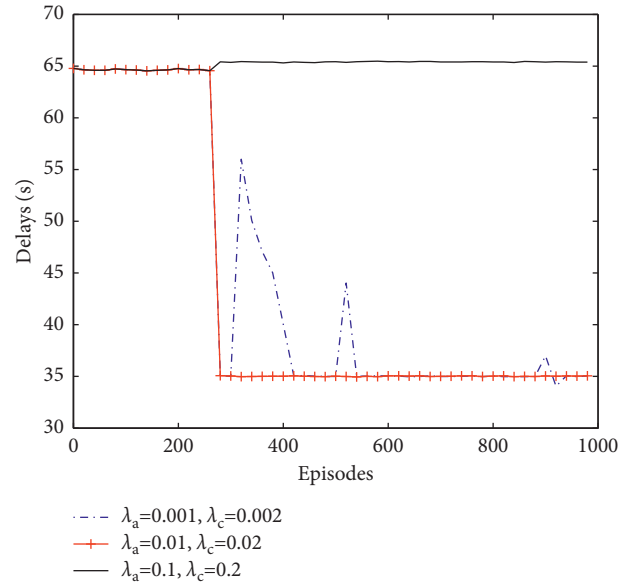

Figure 4: Convergence of CSO-DRL under different learning rates.

Next, the convergence of the algorithm under different discount factors γ are conducted, which can indicate the weight of the value for the next action. The value of γ can affect the efficiency of the algorithm. As Figure 5 shows, the trained service offloading strategy performs best when the discount factor $\gamma = 0.5$. Therefore, we set the value $\gamma = 0.5$ in the following experiments.

Furthermore, the convergence of the algorithm under different exploration parameters σ is also conducted. Noticed that, the larger value for σ allows the agent to choose a prudent strategy, and make the evaluation of the q-value become more accurate, and then the training process will become more stable. As Figure 6, when our algorithm converges at $\sigma = 0.1$, the delay fluctuates at 35s. The initial delay with the $\sigma = 0.1$ is lower than the situation with $\sigma = 0.01$. We noticed that when $\sigma = 1$, the algorithm decreases at 280 episodes, but the convergence is unstable. Hence, we set $\sigma = 0.1$ to obtain better performance in the following experiments.

With the parameters determined, we compare the delay of our algorithm with baseline algorithms. Figure 7 shows the performance of different algorithms. From this figure, it can be observed that the delay of the CSO-DRL algorithm is lower than the other baseline algorithms significantly under the same service size. As the number of iterations increases,

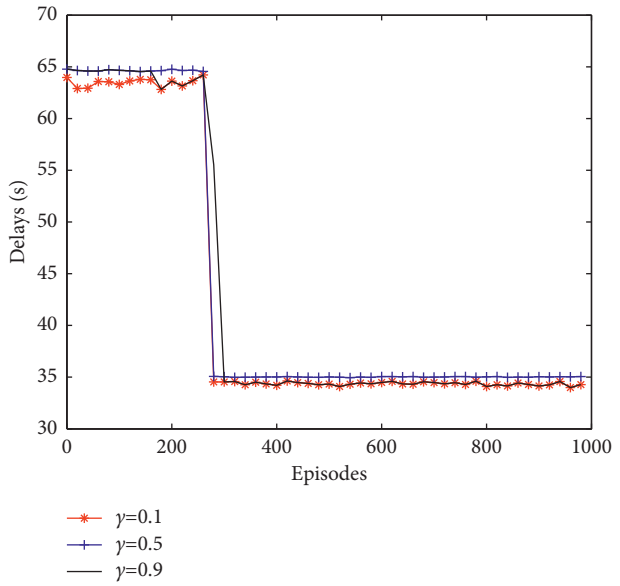

Figure 5: Convergence of CSO-DRL under different discount factors.

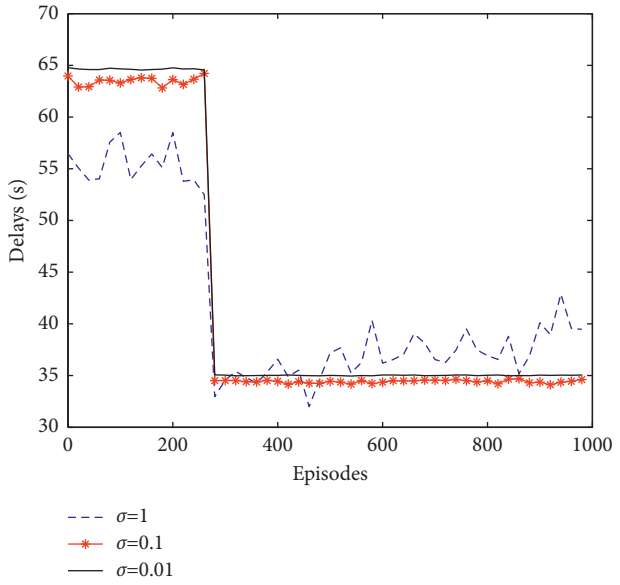

Figure 6: Convergence of CSO-DRL under different exploration parameters.

the AC algorithm cannot converge, while the DQN algorithm and CSO-DRL algorithm can achieve convergence. Because the actor network and critic network of the AC algorithm are updated at the same time, the action selection of the actor network depends on the value function of the critic network, but the critic network is difficult to converge. In contrast, since both DQN and DDPG have a dual network structure. While DQN is only suitable for discrete action spaces and the state space for service offloading is continuous, DQN cannot search the optimal offloading strategy accurately. On the other hand, the CSO-DRL algorithm explores a continuous action space and takes precise actions, which may result in an optimal policy and make the delay

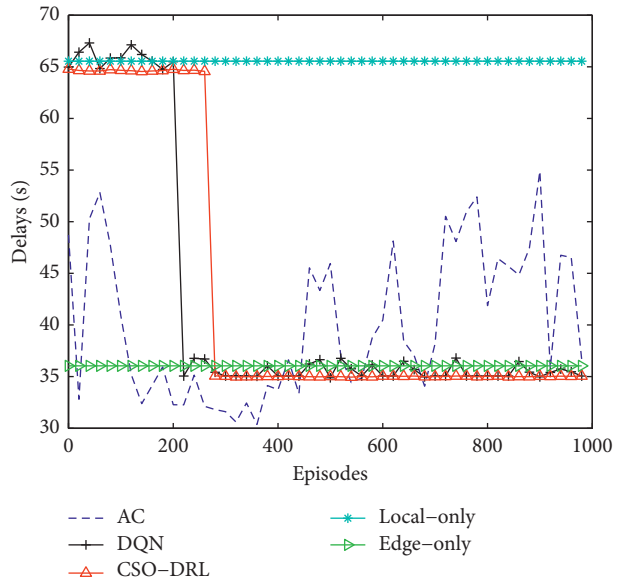

Figure 7: Convergence performance of different algorithms.

reduce significantly. We can also find that the CSO-DRL algorithm and the DQN algorithm can converge at 300 episodes, and the CSO-DRL algorithm obtains a better delay than DQN and converges at 35, which is more stable than the DQN algorithm.

Besides the comparison with the AC algorithm and the DQN algorithm, we also conduct the experiments by comparing the CSO-DRL algorithm with the local-only algorithm and the offload-only algorithm. In Figure 7, it is obvious that the delay of local computing is too large due to the limitation of the computation capacity of the vehicular devices. Since the offload-only algorithm depends on the running state and the computation resources of the edge server, and the vehicles keep the state with resource competition for a long time, the CSO-DRL algorithm can obtain a better delay than offload-only algorithm.

We also conduct the experiment under different service sizes (also called task sizes). Since the AC algorithm cannot converge, we compare the delay with other algorithms. As Figure 8 shows, with the task sizes increased, the delay of the CSO-DRL algorithm is always lower than the other baseline algorithm for the same task size. We notice that the delay of CSO-DRL increases much slower than other baseline algorithms, which is significant when the task sizes increase doubled.

In order to explore the effect of the number of edge servers for the delay, we compare the delay of our proposed algorithm with other algorithms under different numbers of edge servers. As Figure 9 shows, the delay of all algorithms except DQN is almost constant as the number of edge servers increases. With the increase in the number of edge servers, the DQN algorithm fluctuates at about 40 s. Besides, the proposed CSO-DRL algorithm achieves the lowest delay. The reason is DDPG model can find the optimal value in the continuous action and obtain the optimal control policy.

Besides the delay comparison with other algorithms under different numbers of edge servers, we compare these

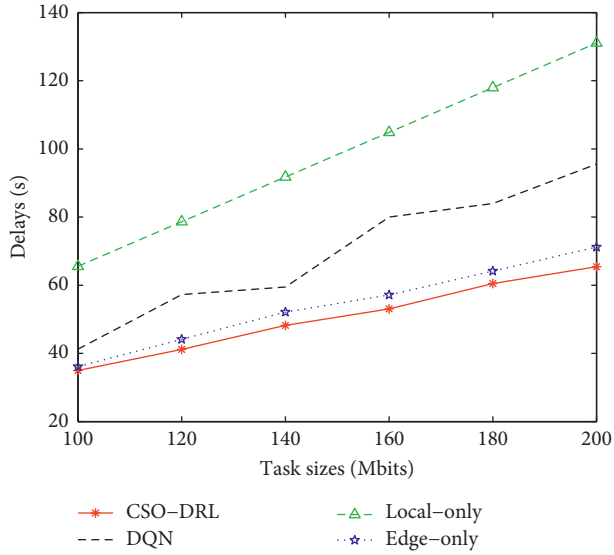

Figure 8: Delay comparison under different task sizes.

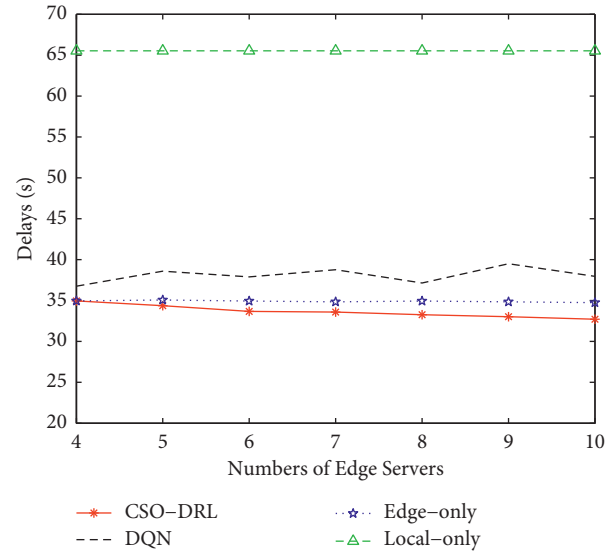

Figure 9: Delay comparison under different numbers of edge servers.

algorithms under different computing capabilities of edge servers and computing capabilities of devices, which can be found in Figures 10 and 11, respectively. Due to the non-convergence of the AC algorithm, we only compared our proposed CSO-DRL algorithm with the DQN algorithm.

Figure 10 shows the same group of experiments in terms of convergence performance and delay under different computing capabilities of edge servers. We vary the computation capabilities of edge servers from 2 GHz to 4 GHz. Figure 10(a) shows the convergence performance of the DQN algorithm and our CSO-DRL algorithm. We find that when the computation capability of the edge server is 3 GHz, the delay of two algorithms is higher than that when the computation capability of the edge server is 4 GHz. Thus, the smaller the computation capability of the edge server, the slower the service execution speed of the system at the same time, which results in a larger delay of the algorithm. We also notice that the CSO-DRL algorithm obtains a lower delay than DQN and converges at 35 s, which is stable than DQN algorithm. Figure 10(b) shows the delay comparison between the DQN algorithm and the CSO-DRL algorithm under different CPU frequency of edge servers. It is obvious that the CSO-DRL algorithm achieves a lower delay than DQN. As the CPU frequencies of edge servers increase from 2 GHz to 4 GHz, the delay decreases following and remains at 27 s when the CPU frequency of edge server is 4 GHz.

Figure 11 shows the same group of experiments in terms of convergence performance and delay under different computation capabilities of edge servers. We vary the computation capabilities of edge servers from 0.6 GHz to 1 GHz. Figure 11(a) shows the convergence performance of the DQN algorithm and our CSO-DRL algorithm. We find

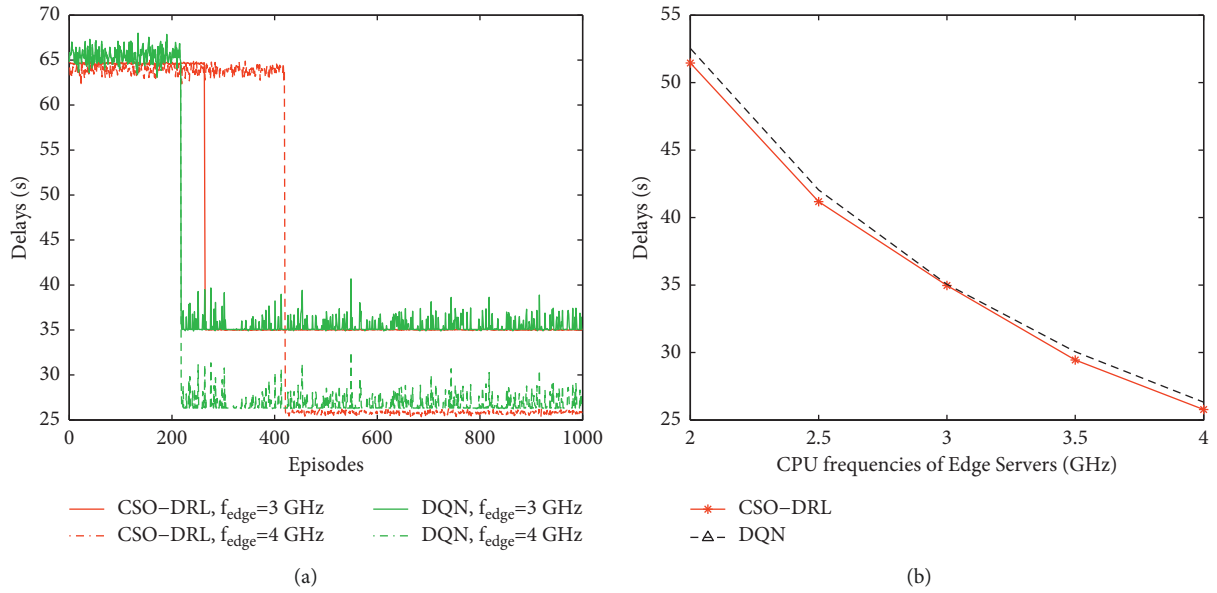

Figure 10: (a) Convergence of different algorithms under different computation capabilities of the edge server. (b) Comparison between different algorithms.

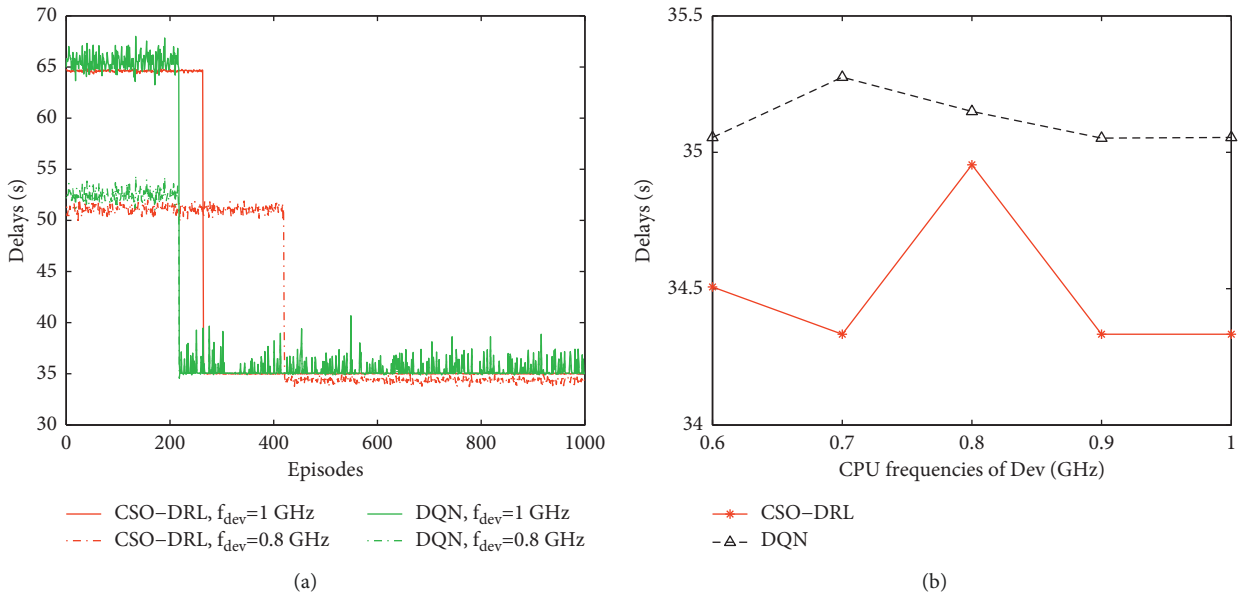

Figure 11: (a) Convergence of different algorithms under different computation capabilities of devices. (b) Comparison between different algorithms.

that when the computation capability of devices is 0.8 GHz, the delay of two algorithms is higher than that when the computation capability of devices is 1 GHz. Thus, the smaller the computation capability of devices, the slower the service execution speed of the system at the same time, which results in a larger delay of the algorithm. We also notice that the CSO-DRL algorithm obtains a better delay than DQN and converges at 35 s, which is stable than the DQN algorithm. Figure 11(b) shows the delay comparison between the DQN algorithm and the CSO-DRL algorithm under different CPU frequencies of devices. The delay of CSO-DRL algorithm increases slowly from 34.5 s to 35 s as the CPU frequency of devices increases from 0.7 GHz to 0.8 GHz and remains at about 34.3*s* when the CPU frequency is 1 GHz. It is obvious that the CSO-DRL algorithm achieves a lower delay than DQN under the same CPU frequency of devices.

6. Conclusion

In this paper, we propose a collaborative service offloading approach with deep reinforcement learning in vehicular edge computing named CSO-DRL. Our approach first

divides the road segments by k-means-based algorithm through analyzing the trajectory data of vehicles, and then the offloading location is determined by observing the vehicle running status. Secondly, the interacting services are discovered by a parallel frequent pattern-based algorithm efficiency. Furthermore, the collaborative service offloading algorithm is presented by the DDPG model for offloading the interacting services, which can reduce the service requesting delay and communication delay together. Finally, the efficiency of the algorithms is evaluated by real-world data-based simulation experimental evaluations. The results show our algorithm can get a better delay in obtaining the optimal service offloading strategy than other baseline algorithms.

Although our approach considers the mobility of vehicles in service offloading and reveals the impact of interrelationship between services on service offloading. There are some avenues for our future studies. In reality, the vehicles are in a complex scenario with resource competition. Thus, how to design offloading strategies for multiusers VEC by considering the computation resource competition is a nonignored problem. We also noticed that due to the limitation of device power, we will present the service offloading strategy to investigate the multiobjective optimization of reducing the energy consumption of vehicular devices and latency in VEC.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Young Project of Science and Technology Research Program of Chongqing Education Commission of China (Nos. KJQN201900708 and KJQN202100738) and the National Natural Science Foundation of China (No. 62101080).

References

[1] J. Contreras-Castillo, S. Zeadally, and J. A. Guerrero-Ibanez, "Internet of vehicles. Architecture, protocols, and security," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3701–3709, 2018.

[2] X. Wang, Z. Ning, X. Hu et al., "Optimizing content dissemination for real-time traffic management in large-scale internet of vehicle systems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1093–1105, 2019.

[3] D. Singh and M. Singh, "Internet of vehicles for smart and safe driving," in *Proceedings of the International Conference on Connected Vehicles and Expo, ICCVE*, pp. 328-329, IEEE, Shenzhen, China, October, 2015.

[4] R. Hussain, D. Kim, J. Son et al., "Secure and privacy-aware incentives-based witness service in social internet of vehicles clouds," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2441–2448, 2018.

[5] L. Wu, R. Zhang, Q. Li, C. Ma, and X. Shi, "A mobile edge computing-based applications execution framework for internet of vehicles," *Frontiers of Computer Science*, vol. 16, no. 5, Article ID 165506, 2022.

[6] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 246–261, 2020.

[7] Y. Chen, H. Xing, and M Zhuo, "Cost-efficient edge caching for noma-enabled iot services," *China Communications*, 2022.

[8] Y. Zhang, *Mobile Edge Computing*, Vol. 9, Springer, , Berlin/Heidelberg, Germany, 2022.

[9] 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 Network*, vol. 33, no. 5, pp. 198–205, 2019.

[10] J. Wang, H. Ke, X. Liu, and H. Wang, "Optimization for computational offloading in multi-access edge computing. A deep reinforcement learning scheme," *Computer Networks*, vol. 204, Article ID 108690, 2022.

[11] F. Song, H. Xing, X. Wang, S. Luo, P. Dai, and K. Li, "Offloading dependent tasks in multi-access edge computing. A multi-objective reinforcement learning approach," *Future Generation Computer Systems*, vol. 128, pp. 333–348, 2022.

[12] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, vol. 10, 2021.

[13] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "A collaborative computational offloading strategy for latency-sensitive applications in fog networks," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4565–4572, 2022.

[14] X. Gu, G. Zhang, M. Wang, W. Duan, M. Wen, and P. H. Ho, "Uav-aided energy-efficient edge computing networks. Security offloading optimization," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4245–4258, 2022.

[15] J. Huang, Z. Tong, and Z. Feng, "Geographical poi recommendation for internet of things. A federated learning approach using matrix factorization," *International Journal of Communication Systems*, vol. n/a, Article ID e5161, 2022.

[16] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay-sensitive task offloading in the 802.11p-based vehicular fog computing systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 773–785, 2020.

[17] Y. Chen, F. Zhao, X. Chen, and Y. Wu, "Efficient multi-vehicle task offloading for mobile edge computing in 6g networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4584–4595, 2022.

[18] Z. Tong, X. Deng, J. Mei, B. Liu, and K. Li, "Response time and energy consumption co-offloading with SLRTA algorithm in cloud-edge collaborative computing," *Future Generation Computer Systems*, vol. 129, no. 64–76, pp. 64–76, 2022.

[19] A. A. Shah, N. A. Bhatti, K. Dev, and B. S. Chowdhry, "MUHAFIZ. iot-based track recording vehicle for the damage analysis of the railway track," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9397–9406, 2021.

[20] H. Attaullah, T. Kanwal, A. Anjum et al., "Fuzzy-logic-based privacy-aware dynamic release of iot-enabled healthcare data," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4411–4420, 2022.

[21] J. C.-W. Lin, G. Srivastava, Y. Zhang, Y. Djenouri, and M. Aloqaily, "Privacy-preserving multiobjective sanitization model in 6g iot environments," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5340–5349, 2021.

[22] R. Roman, J. López, and M. Mambo, "Mobile edge computing, fog et al. A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[23] W. Chen and B. Yang, "Energy efficiency analysis of e-commerce customer management system based on mobile edge computing," *Scientific Programming*, vol. 2022, Article ID 5333346, 9 pages, 2022.

[24] S. Wang, Y. Li, and S. Pang, "A task scheduling strategy in edge-cloud collaborative scenario based on deadline," *Scientific Programming*, vol. 2020, Article ID 3967847, 9 pages, 2020.

[25] Y. Zhou, L. He, B. Wang, Y Su, and H. Chen, "MCAF. developing an annotation-based offloading framework for mobile cloud computing," *Scientific Programming*, vol. 2020, no. 9, Article ID 5304612, 9 pages, 2020.

[26] K. Gasmi, S. Dilek, S. Tosun, and S. Ozdemir, "A survey on computation offloading and service placement in fog computing-based IoT," *The Journal of Supercomputing*, vol. 78, no. 2, pp. 1983–2014, 2022.

[27] Z. Hong, W. Chen, H. Huang, S. Guo, and Z. Zheng, "Multi-hop cooperative computation offloading for industrial iot-edge-cloud computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2759–2774, 2019.

[28] Z. Yu, Y. Gong, S. Gong, and Y. Guo, "Joint task offloading and resource allocation in uav-enabled mobile edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3147–3159, 2020.

[29] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless iot environment," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4411–4422, 2019.

[30] Y. Chen, W. Gu, and K. Li, "Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning," *International Journal of Communication Systems*, vol. n/a, Article ID e5154, 2022.

[31] J. Chen and Z. Wu, "Dynamic computation offloading with energy harvesting devices. A graph-based deep reinforcement learning approach," *IEEE Communications Letters*, vol. 25, no. 9, pp. 2968–2972, 2021.

[32] J. Xu, D. Li, W. Gu, and Y. Chen, "Uav-assisted task offloading for iot in smart buildings and environment via deep reinforcement learning," *Building and Environment*, vol. 222, Article ID 109218, 2022.

[33] J. Huang, C. Zhang, and J. Zhang, "A multi-queue approach of energy efficient task scheduling for sensor hubs," *Chinese Journal of Electronics*, vol. 29, no. 2, pp. 242–247, 2020.

[34] J. Huang, B. Lv, Y. Wu, Y. Chen, and X. Shen, "Dynamic admission control and resource allocation for mobile edge computing enabled small cell network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1964–1973, 2022.

[35] X. Xu, R. Gu, F. Dai, L. Qi, and S. Wan, "Multi-objective computation offloading for internet of vehicles in cloud-edge computing," *Wireless Networks*, vol. 26, no. 3, pp. 1611–1629, 2020.

[36] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles. Model and algorithm," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5913–5925, 2021.

[37] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles. A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[38] X. Xu, Y. Xue, L. Qi et al., "An edge computing-enabled computation offloading method with privacy preservation for internet of connected vehicles," *Future Generation Computer Systems*, vol. 96, no. 89–100, pp. 89–100, 2019.

[39] 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, 2021.

[40] Y. Huang, J. Huang, B Cheng, T. Yao, and J. Chen, "Poster. Interacting data-intensive services mining and placement in mobile edge clouds," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom*, pp. 558–560, ACM, Snowbird, UT, USA, October, 2017.

[41] Y. Huang, J. Huang, C. Liu, and C. Zhang, "PFPMine. A parallel approach for discovering interacting data entities in data-intensive cloud workflows," *Future Generation Computer Systems*, vol. 113, pp. 474–487, 2020.

[42] L. Rokach, "A survey of clustering algorithms," in *Proceedings of the Oded Maimon and Lior Rokach, editors, Data Mining and Knowledge Discovery Handbook, 2nd ed*, pp. 269–298, Springer, Berlin/Heidelberg, Germany, 2010.

[43] T. P. Lillicrap, J. J. Hunt, and P. Alexander, "Continuous control with deep reinforcement learning," in *Proceedings of the Yoshua Bengio and Yann LeCun, editors, 4th International Conference on Learning Representations, ICLR 2016*, Conference Track Proceedings, San Juan, Puerto Rico, May, 2016.

[44] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation,"vol. 12, pp. 1057–1063, in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 12, pp. 1057–1063, The MIT Press, Denver, Colorado, USA, November, 1999.

[45] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[46] Y. Wang, W. Fang, Y Ding, and N. Xiong, "Computation offloading optimization for uav-assisted mobile edge computing. a deep deterministic policy gradient approach," *Wireless Networks*, vol. 27, no. 4, pp. 2991–3006, 2021.