Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Collaborative on-demand dynamic deployment via deep reinforcement learning for IoV service in multi edge clouds

Yuze Huang[1*], Beipeng Feng[1], Yuhui Cao[1], Zhenzhen Guo[1], Miao Zhang[1] and Boren Zheng[1]

Abstract

In vehicular edge computing, the low-delay services are invoked by the vehicles from the edge clouds while the vehicles moving on the roads. Because of the insufficiency of computing capacity and storage resource for edge clouds, a single edge cloud cannot handle all the services, and thus the efficient service deployment strategy in multi edge clouds should be designed according to the service demands. Noticed that the service demands are dynamic in temporal, and the inter-relationship between services is a non-negligible factor for service deployment. In order to address the new challenges produced by these factors, a collaborative service on-demand dynamic deployment approach with deep reinforcement learning is proposed, which is named CODD-DQN. In our approach, the number of service request of each edge clouds are forecasted by a time-aware service demands prediction algorithm, and then the interacting services are discovered through the analysis of service invoking logs. On this basis, the service response time models are constructed to formulated the problem, aiming to minimize service response time with data transmission delay between services. Furthermore, a collaborative service dynamic deployment algorithm with DQN model is proposed to deploy the interacting services. Finally, the real-world dataset based experiments are conducted. The results show our approach can achieve lowest service response time than other algorithms for service deployment.

Keywords Service deployment, Internet of vehicles, Service demands, Deep reinforcement learning, Multi edge clouds

Introduction

Internet of Vehicles (IoV) creates the bridge between the vehicles and roadside units (RSUs) through the wireless communication technologies [1], which can be regarded as a typical IoT network and has been applied in urban transportation system. The IoV system can realize the data interaction between vehicles and RSUs, and make the decision for auto-driving [2].

In intelligent transportation system, the vehicles equipped with intelligent devices which are responsible for the collection of the vehicles moving status and traffic road condition data for analysis and computation [3]. The cloud computing based IoV system can address the problems produced by the computing capacity limitation of vehicles [4]. With the data collected by sensors increased, the cloud computing may bring the high service delay and the network congestion problems, which is difficult to satisfy the low-delay requirement for latency sensitive services [5]. Besides the low-latency requirement, we also noticed the mobility of vehicles is another important factor, which may bring the difficult to provide all services to vehicles relying on a single cloud, which may result in a serious performance degradation. To solve such

*Correspondence:
Yuze Huang
huangyz@cqjtu.edu.cn
[1] School of Information Science and Engneering, Chongqing Jiaotong University, Chongqing, China

problems, the edge computing has been produced and raised widely attentions of researches, which can not only provide low-latency service to users efficiently, but can also avoid the single cloud provider lock-in and guarantee the service performances [6, 7].

In edge computing, the intelligent devices are responsible for pre-processing raw data and offloading them to the edge clouds, which are closer to users, and mainly undertake to process the data [8]. Thus the edge computing can enhance the computing capacity of the edge of network [9, 10]. In reality, because of the insufficiency of computation capability and storage resource for edge clouds, the execution of the IoT services on edge clouds require the designing of service deployment strategy [11, 12]. Most of the studies concentrate on reducing the service response time and energy consumption of the intelligent devices [13–15]. With the deepening of researches, some studies noticed the heterogeneity of service requests among multi edge clouds. To address the problem of service requests imbalances among multi edge clouds, some studies proposed some efficient approaches of service deployment with computation workload scheduling strategies [16–18]. In sight of these studies, most of these schemes are produced based on the assumption of the known service demands. Generally speaking, the service demands are unknown in practice, which may result in the unreasonable deployment strategy and large service delay during the service deployment process. Thus, Hao et al. [19] presented a service deployment with the computation resource allocation strategy under the uncertainty of service demands in industrial cyber-physical system.

Along with the deepening of vehicular edge computing, we noticed the service deployment meet new challenges due to the particularity of the IoV environment. First, with the dramatic increase of mobile vehicles, the service demands are imbalance and highly dynamic in temporal, which may greatly influence the service delay to a large extent [20]. Thus the services should be deployed according to the service demands and the temporal dynamic of service demands should be considered for service deployment. Second, it is demonstrated that with the development of IoV, single atomic service cannot satisfy the complex business requirements. So the interacting services should complete the business goal with collaboration, and it exists large amount transmission data between the services [21]. Thus, the inter-relationship between services is another non-negligible factor for service deployment.

To deal with the above mentioned challenges, a collaborative service on-demand dynamic deployment approach is proposed to deploy the interacting services on multi edge clouds, which is named CODD-DQN. In our approach, a time aware service demands prediction algorithm is introduced to forecast the number of service request for each edge cloud, and then the interacting services are mined by a parallel algorithm. On this basis, the service response time models are formulated. Furthermore, we propose a collaborative service dynamic deployment algorithm via deep reinforcement learning to deploy the interacting services according to the forecasted the number of service request, which considers the minimization problem for service response time with data transmission delay between services. Specifically, the contributions of this paper can be threefold as the following descriptions.

- The number of service request for each edge cloud are forecasted by a time-aware service demands prediction algorithm based on the ARIMA model, which can investigate the temporal dynamic characteristics of service demands.
- Service response time models are formulated according to the inter-relationship between interacting services which have been discovered by a parallel mining algorithm.
- The collaborative service on-demand dynamic deployment algorithm via DQN model is presented to deploy the interacting services according to the forecasted value of service demands, which can reduce the service response time with data transmission delay between services.

The rest of this paper is organized as follows. we introduce the related work of this research in Related work section. Framework of collaborative service dynamic deployment section presents the framework of collaborative service dynamic deployment, and then a ARIMA based time-aware algorithm is presented to forecast the number of service request in Time-aware service demands prediction section. The system response time models are constructed to formulate the problem of service deployment in System model and problem formulation section. Furthermore, Algorithm for collaborative service dynamic deployment section proposes a collaborative service on-demand dynamic deployment algorithm with DQN to deploy the interacting services according to the service demands, aiming to solve the minimization problem of the service response time with data transmission delay between services. Finally, we evaluate the efficiency of our algorithms in Experimental evaluation section, and then Conclusion section concludes this paper.

Related work

In IoT environments, the data produced by the various intelligent devices are experiencing rise, which may lead to high latency and network congestion in IoT system. Thus the cloud computing cannot provide the low latency services for users [9]. To address such problems, edge computing is introduced and applied in wide areas. For edge computing, intelligent devices offload the preprocessed raw data to the edge clouds which is near to the users. While the edge clouds responsible to execute the services, and the cloud servers are only undertake to execute data-intensive services and train the deep neural network [22].

Currently, most studies have concentrated on task offloading, which mainly concentrate on how to design efficient offloading strategy to offload the tasks on edge clouds or remote cloud server [23]. In sight of these works, existing task offloading strategy can be divided into 0/1 offloading and partial offloading [24, 25]. Considering the insufficient computing capacity of intelligent devices and the limitation computation resource of edge clouds, the partial offloading is the reasonable task offloading manner, which can be formulated as a minimization problem of service request delay or energy consumption of devices [26, 27].

According the prior knowledge of the global information, the computing capacity or storage resource of a single edge cloud is insufficient, and all services cannot be executed on single edge cloud. Thus, an efficient services deployment strategy should be designed for deploying services on edge clouds or remote cloud server. For service deployment, some existing studies proposed efficient service deployment algorithms to reduce the service response time or allocate computation resource for edge computing [28–30]. For example, a fog configuration is presented to solve the minimization problem of energy consumption and request delay for industrial IoT [13, 31]. Wang et al. [14] proposed a edge server placement algorithm, which can minimize multi optimization objectives and balance the workloads between edge clouds. Noticed that the imbalance of service demands is another non-negligible factor on multi edge clouds, and then the optimization of service deployment joint with resource scheduling are investigated by some researchers. Ma et al. [17] introduced a cooperative schema combined service placement and workload scheduling for minimizing the service response time. Hao et al. [19] proposed an efficient service deployment strategy joint with resource allocation through considering the uncertain service demands. In summary, the service demands is another factor which must be take into consideration for service deployment.

In sight of the existing studies, internet of vehicles has been widely used in modern urban traffic system, and thus the edge computing based IoV has been widely concentrated by some investigations [32]. For vehicular edge computing, the vehicles invoke low-delay services from edge clouds which are closer to the vehicles. According to our prior knowledge, the service demands are uncertain and present temporal dynamic characteristics among the multi edge clouds. To design a reasonable service deployment strategy, the service demands uncertainty and temporal dynamic of service request must be considered for service deployment [20]. It is demonstrated the simple atomic service cannot satisfy the complex business requirements in reality, therefore the interacting services should collaborative work with each other to complete the business goal. It exists large amount transmission data between the interacting services, which is another non-negligible factor for service provisioning [21]. In our previous work [33], we studied the collaboration between interacting services for service offloading to minimize the service request delay and data transmission delay between services. Comparing with existing studies, we study the temporal dynamic characteristics of service demands and reveal the inter-relationship between services, aiming to solve the minimization problem of service response time with data transmission delay between services.

Framework of collaborative service dynamic deployment

The architecture of internet of vehicles is presented in Fig. 1. Typically, the architecture can be composed of three layers, which are remote cloud layer, edge network layer and vehicle user layer. Generally speaking, the vehicle user layer contains numerous vehicles, which mainly undertake the capacity of sensing the road environment and collect the data from vehicles. Due to the limited computation of vehicles, the vehicles only preprocess the raw data and transmit them to the RSUs, which often act as edge clouds in IoV. Comparing with vehicles devices, the edge clouds have rich communication, computation and storage resources. Thus the RSUs are responsible for the execution of computation-intensive services. By deploying the service on edge clouds, the edge clouds are beneficial for processing the strict latency requirements and deliver the low-latency service to vehicle users. In IoV, the cloud server with higher computing capacity and more storage capacity undertake to provide the global management and centralized decisions control in the system. We investigate the temporal dynamic of service demands and reveal the inter-relationship between services in this paper. Thus, the interacting services are deployed according the forecasted number of service request on multi edge clouds. The cloud servers only responsible for training the deep reinforcement

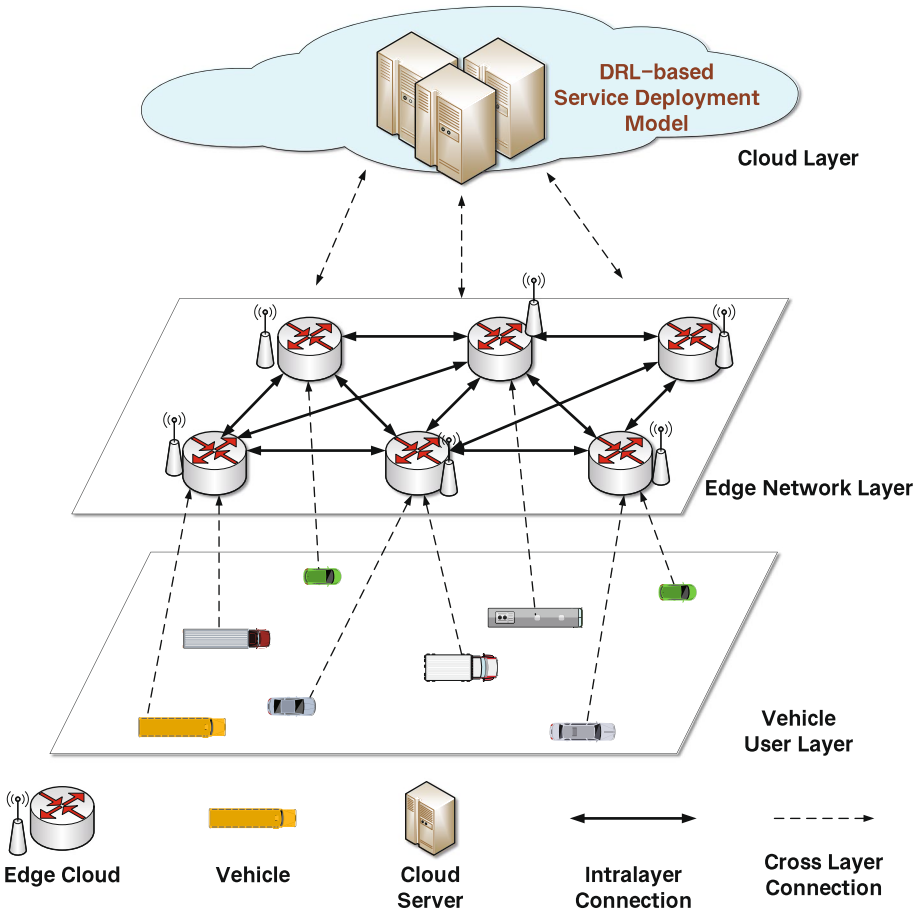

Fig. 1 Architecture of vehicular edge computing

learning based service deployment model, and then service deployment strategy will be send to the edge to perform for minimizing the service response time in the whole system.

As Fig. 2 shows, the service invoking logs are collected as the input of our approach, which contains the service request sequence and the number of service request on each edge cloud. First, to investigate the temporal dynamic characteristic of the service demands, a time-aware service demands prediction algorithm by ARIMA model is introduced to forecast the number of service request. Furthermore, we employ a parallel algorithm to discover the interacting services [34, 35]. Finally, the interacting services are deployed by the DQN-based collaborative service dynamic deployment algorithm according the forecasted number of service request, aiming to optimize the service response time with data transmission delay between services. The details can be found as follows.

Step1. Service invoking logs are exacted as the input of our approach, and then the ARIMA model based algorithm is put forward for forecasting the number of service request for each edge cloud, which can investigate the temporal dynamic characteristic of service demands.

Step2. Service response time models are constructed according to the inter-relationship between services, which have been discovered by our proposed algorithm [34, 35].

Step3. A collaborative service on-demand dynamic deployment algorithm based on DQN model is presented to deploy the interacting services, aiming to minimize the service response time with data transmission delay between services. This algorithm can obtain the optimal service deployment strategy through receiving environment status and performing the decision actions through iterative computing.

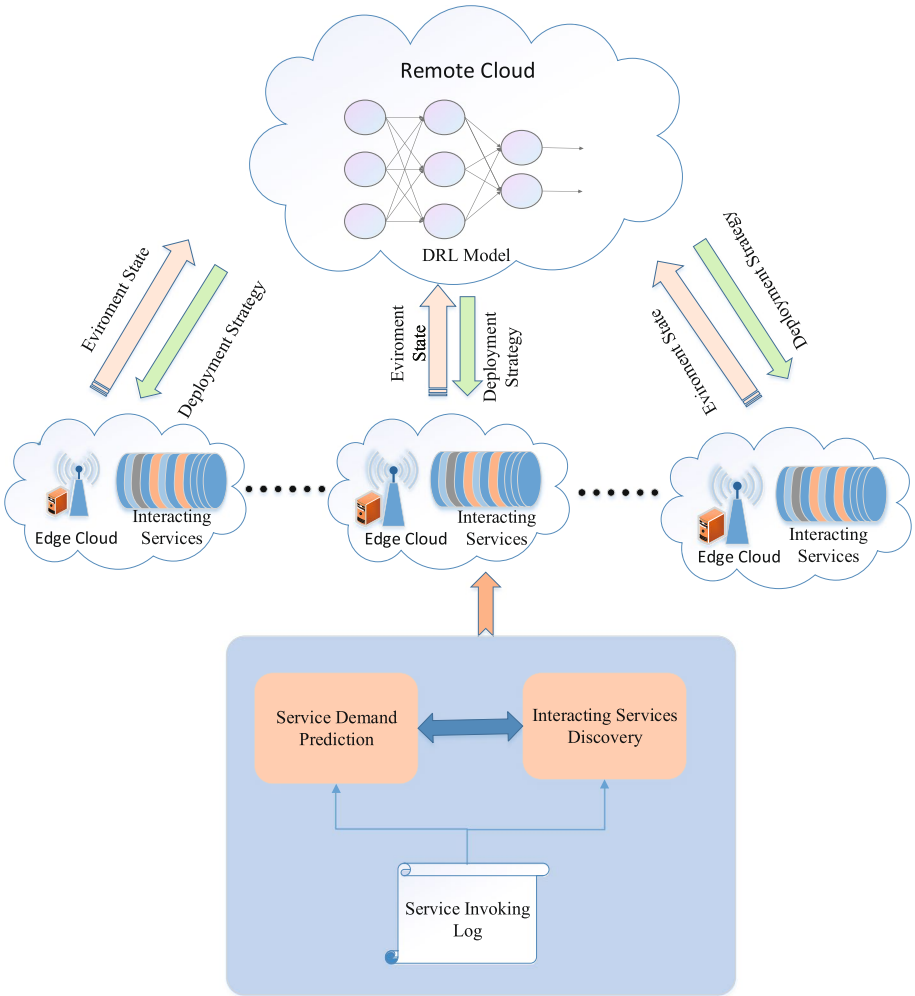

Fig. 2 Framework of CODD-DQN

Time-aware service demands prediction

In vehicular edge computing, due to the mobility of the vehicles, the service demands are imbalance and dynamic in temporal. According to the temporal characteristics of service demands, we put forward a time-aware algorithm to forecast the number of service request based on the ARIMA model. Next, we will present the time-aware service demands prediction algorithm to forecast the number of service request of each edge clouds.

In our system, the services $s = \{1, 2, ..., s\}$ are deployed on the edge clouds $E = \{1, 2, ..., i\}$. In order to investigate the temporal dynamic of the service demands, the number of service request for service k deployed on edge cloud i denoted as $\{c(i, k, t)|t = 0, 1, 2, ..., n\}$. The number of service request can be forecasted by our algorithm. The ARIMA integrates autoregressive (AR) and moving average (MA) model to formulated the time series data [36]. In this model, if the original data is non-stationary, the data should be transferred into a stationary data through d steps differences. Thus the time series denoted by $ARMA(p, q)$ can be modeled as follows.

$$c_t = \phi_0 + \sum_{i=1}^{p} \phi_i c_{t-i} + \sum_{j=1}^{q} \theta_j a_{t-j} + a_t, \quad (1)$$

where ϕ_0 is a constant item. θ_j and ϕ_i denote the parameter of MA and AR model, respectively. a_t denotes the white noisy. p, q are non-negative integer, which denote the order of AR model, MA model, respectively.

To our best knowledge, the most important step for ARIMA-based time series forecasting is constructing ARIMA model and determining the order of model to forecast the future data. In our algorithm, the pre-condition of constructing the ARIMA model is checking the series data is white noisy or not. For this step, the Ljung-Box test is used for white noisy checking. If the series data satisfy the pre-condition, the ARIMA model can be used for time series forecasting, else, we employ the simple moving

average to forecast the number of service request in this algorithm, which can be formulated as

$$\begin{aligned}\hat{c}(i,k,t+n) =&[c(i,k,1)+...+c(i,k,t)\\ &+\hat{c}(i,k,t+1)+\hat{c}(i,k,t+2)\\ &+...+\hat{c}(i,k,t+n-1)]/t+n-1,\end{aligned} \tag{2}$$

where $\hat{c}(i,k,t+n)$ represents the $n-th$ forecasted value of the number of service request, and $c(i,k,t)$ is the $t-th$ observed value.

According to the discusses of time series forecasting, the process of ARIMA-based service demands prediction algorithm follows the following six steps.

Step 1: Stationarity Checking. With the white noisy checking completed, the stationarity of the number of service request series should be determined by the unit root test. If the time series data is not stationarity, the original data should be calculated through d steps differences, and transfer them into the stationary series.

Step 2: Model Identification. Model identification is the most important step in time series forecasting. During this process, the order of p and q should be determined for constructing the ARMA model. In this step, the ACF (autocorrelation function) and PACF (patrial autocorrelation function) are computed to assist the order selection, which can be obtained by the following expressions:

$$\rho_k = \frac{\gamma_k}{\sigma^2}, \tag{3}$$

$$\phi_{kk} = \frac{\rho_k - \sum_{j=1}^{k-1}\phi_{k-1,j}\rho_{k-j}}{1-\sum_{j=1}^{k-1}\phi_{k-1,j}\rho_j}, \tag{4}$$

where ρ_k represents the lag k ACF, and the γ_k represents the lag k auto-covariance function. The lag k PACF is denoted by ϕ_{kk}.

Since the ACF and PACF are computed by the former equations, the order of the ARIMA model is selected accordingly. If PACF is truncated at p-order and ACF decays, the $AR(p)$ model can be selected to constructed the model. If ACF is truncated at q-order and PACF decays, the $MA(q)$ model can be used to fit the series data. If ACF and PACF decay, the $ARMA(p,q)$ can be adopted as the model to fit the series data.

Step 3: Model Estimation. After the model order is selected, the parameters of the model should be estimated for the ARMA model. In this step, the maximum likelihood estimation is adopted to determined the parameters by the following expression.

$$l \propto (\sigma^2)^{-\frac{n}{2}} exp\{-\frac{1}{2\sigma^2}\sum_{t=1}^{n}(a_t)^2\}, \tag{5}$$

where l represents the likelihood function, and $a_t \sim N(o,\sigma^2)$ denotes the white noisy.

Step 4: Model Checking. In this step, the significance of models and parameters should be checked. If the significance test is satisfied, the model can be adopted to forecast the number of service request.

Step 5: Model Selection. Since the model checking is completed, the optimal model should be selected from all candidate models which have passed the significance test. The model selection according to the AIC (Akaike's Information Criterion) value in this step, the model which has the minimum AIC value should be selected to forecast the future data.

Step 6: Number of service request forecasting. Since the optimal model is selected, the number of service request are forecasted by the constructed model. In this algorithm, the $(n+1)-th$ value is calculated according to the $n-th$ forecasted value. Thus, As the steps increased, the forecasted error increases accordingly. The details of the algorithm can be found in Algorithm 1. In our system, the prediction algorithm is deployed on each edge cloud, and the number of service request for each edge cloud can be forecasted by this algorithm.

Input: The number of service request D
Output: Forecasted the number of service request values $\{c(i,k,t)|t=n,n+1,...,n+m\}$ for service k on edge cloud i

```
for Service s = 1 to k do
  if p-value of LB-test pv_lb > α then
    Forecast the future values by (2)
  else
    if the series data is non-stationary then
      {c(i,k,t)} ← diff({c(i,k,t)})
    end if
    The models for fitting the series data are identified
    The parameters of identified models are determined
    The significance of all candidate models are checked, and then significance models are added into the candidate model set
    The optimal model is selected as the fitted model through the AIC value
    The future values {c(i,k,t)|t = n, n+1, ..., n+m} are forecasted by the fitted model.
  end if
end for
```

Algorithm 1 Algorithm for Time-aware Service Demands Prediction

System model and problem formulation

In this section, the service response time models are presented to formulate the service deployment problem of our approach in the following contents.

System model

In reality, a complex service can be composed by a serial of sub-services, each of which processes certain data and accomplishes one piece of sub-task. In that cases, the precursor service should be executed and transmit the processed data to the subsequent service, and then the subsequent service should process the transmitted data to accomplish a certain task. Thus, it may exist the data communications between interacting services. In such cases, the inter-relationship between the services should be considered for service provisioning in edge computing.

In this paper, we construct the system model for service deployment during the time slots $T = \{1, 2, ..., t\}$. During the process, the services are deployed on edge clouds and the computation resource are allocation in each duration. In our system, the finite services are deployed on multi edge clouds upon the limited storage and computation resource, and the user requests the service from the proximity edge clouds. We assume there are a series services denoted as $K = \{1, 2, ..., k\}$, which are deployed on the multi edge clouds. The edge clouds can be denoted by $S = \{1, 2, ..., s\}$. We let $M(i)$ and $D(i)$ denote the computing and storage capacity of edge cloud, respectively. In contrast with previous works [19, 20], we study the temporal dynamic of service demands and consider the interrelationship between interacting services for service deployment, and thus the interacting services are deployed collaborative on the multi edge clouds. The remote cloud is only responsible to train the deep reinforcement learning model for searching the service deployment strategy. In the following contents, we present the system model with service response time and formulate the service deployment problem. The important notations of this paper are shown in Table 1.

As mentioned above, we construct the system model for services deployment with computation resource allocation in multi edge clouds. First, we define the service deployment function as $b(k, i, t) \in \{0, 1\}$, whose value is a binary variable. Thus, when the service is deployed on edge cloud, we let $b(k, i, t) = 1$, otherwise $b(k, i, t) = 0$. Due to the insufficiency of the storage capacity of edge cloud, the whole data size of the services cannot exceed the storage capacity of edge cloud.

$$\sum_{k=1}^{K} b(k, i, t) d(k) \leq D(k), \forall t, \tag{6}$$

where $d(k)$ represents the data size of the service k.

To improve the utilization of computation resource, a primer resource allocation scheme for service deployment is designed in multi edge clouds. We use $l(k, i, t) \in [0, 1]$ denote the proportion of computation resource allocation. Accordingly, if the service is not deployed on the edge cloud in this time, the $l(k, i, t) = 0$. Thus the computation resource allocation function is defined by $L(t) = \{l(k, i, t) | i \in S, k \in K\}$. Since the computing capacity of edge cloud is insufficient, the allocated proportion of computation resource to execute service cannot exceed 1, which can be expressed as

$$\sum_{k=1}^{K} l(k, i, t) \leq 1, \forall t. \tag{7}$$

Once the service is deployed on the edge cloud, the computation resource should be allocated according to the following scheme for executing this service. Thus, when the services are deployed on the edge cloud, the computation resource should be allocated as a certain proportion value, otherwise the allocated computation resource is 0. The relationship between $l(k, i, t)$ and $b(k, i, t)$ can be formulated as follows.

$$l(k, i, t) = \begin{cases} 0 & b(k, i, t) = 0 \\ g & b(k, i, t) = 1 \end{cases}, \quad g \in (0, 1], \ \forall t, \tag{8}$$

where g denotes the proportion value of computation resource allocated for executing the service.

To analyze the service response time in this system, we let $c(k, i, t)$ denote the number of service request, which can be forecasted by our proposed service demands prediction algorithm. In this paper, once the service cannot be deployed on such edge cloud, the service should be executed on another edge cloud through service scheduling. We notice that the data back haul delay for executing the service is much smaller than service request delay and data transmission delay, thus the delay of data back haul can be ignored in this paper.

In this paper, the edge clouds receive the service request and the data should be transmitted from vehicles to edge clouds, thus the data transmission delay between vehicles and edge clouds can be calculated by the following expression.

Table 1 List of important notations

Notations	Description
$D(i)$	Storage capacity of edge cloud i
$M(i)$	Computing capacity of edge cloud i
$d(k)$	Data size of service k
$m(k)$	Computing capacity for executing the service k
V_{v2e}	Network transmission rate between vehicles and edge clouds
V_{e2e}	Network transmission rate among edge clouds
$c(k, i, t)$	Number of service request for service
$d(kk^*)$	Data transmission size between interacting services

$$W_{v2e}^{tran} = \frac{C(k,t)d(k)}{V_{v2e}}, \tag{9}$$

where $C(k, t)$ denotes the total value of the number of service request for service k on all edge clouds, which can be computed through $C(k,t) = \sum_{i=1}^{S} c(k,i,t)$. The V_{v2e} denotes the network transmission rate between vehicles and edge clouds.

As mentioned above, the services should be executed through service scheduling in some cases. Therefore, the data transmission delay between edge clouds can be computed as

$$W_{e2e}^{tran} = b(k,i,t)\frac{(C(k,t) - c(k,i,t))d(k)}{V_{e2e}}, \tag{10}$$

where $C(k,t) - c(k,i,t)$ is the number of service request handled on other edge clouds, and V_{e2e} denotes the network transmission rate between edge clouds.

When the service is executed on the edge cloud, the computation delay can be calculated by the following expression.

$$W^{comp} = b(k,i,t)\frac{C(k,t)m(k)}{l(k,i,t)M(i)}, \tag{11}$$

where $m(k)$ is the computation resource requirement of service k.

Comparing with other studies, we investigate the data transmission delay between services. Assuming it exists some interacting services, which can be divided to the pre-service k and successor service k^*. In that case, the number of service request for service k handled on edge cloud i can be denoted as $c_{comp}(k,i,t)$, and the total value of the number of service request for service k handled at time slot t can be computed by $C_{comp} = \sum_{i=1}^{S} c_{comp}(k,i,t)$. Thus, the data transmission delay between services can be calculated by

$$\begin{aligned} W_{s2s}^{tran}(kk^*,i,t) =& [b(k,i,t)(C_{comp}(k,t) \\ & - c_{comp}(k,i,t))d(kk^*)]/V_{e2e}, \end{aligned} \tag{12}$$

where $d(kk^*)$ denotes the data transmission size between interacting services. In that case, the computation delay for executing the successor service k^* can be calculated by the following expression.

$$W_{suc}^{comp}(k^*,i,t) = b(k^*,i,t)\frac{C_{comp}(k,t)m(k^*)}{l(k,i,t)M(i)}, \tag{13}$$

where $m(k^*)$ is the computation resource requirement of successor service k^*.

Problem formulation

With the system models are constructed, the response time for handling the interacting services can be obtained as follows.

$$\begin{aligned} W^{sum}(kk^*,t) = \sum_{i=1}^{S} [& W_{v2e}^{tran}(k,i,t) \\ & + W_{e2e}^{tran}(k,i,t) \\ & + W^{comp}(k,i,t) \\ & + W_{s2s}^{tran}(kk^*,i,t) \\ & + W_{suc}^{comp}(k^*,i,t)]. \end{aligned} \tag{14}$$

In addition, the service response time for handling the single atomic services can be obtained by the following expression.

$$\begin{aligned} W_{single}^{sum}(k,t) = \sum_{i=1}^{S} \big[& W_{v2e}^{tran}(k,i,t) + W_{e2e}^{tran}(k,i,t) \\ & + W^{comp}(k,i,t) \big]. \end{aligned} \tag{15}$$

In summary, the total delay for handling all services can be obtained as

$$W^{sum}(k,t) = W^{sum}(kk^*,t) + W_{single}^{sum}(k,t). \tag{16}$$

In this paper, our purpose is minimizing the service response time for service deployment based on the service demands prediction. So we formulate the service deployment problem as

$$\min_{B,L} = \frac{1}{T}\sum_{t=1}^{T}\sum_{k=1}^{K} W^{sum}(k,t), \tag{17}$$

$$s.t.\ C1: \quad \sum_{k=1}^{K} b(k,i,t)d(k) \le D(i), \forall t, \tag{18a}$$

$$C2: \quad \sum_{k=1}^{K} l(k,i,t) \le 1, \forall t, \tag{18b}$$

$$C3: \quad b(k,i,t) \in \{0,1\}, k \in K, i \in S, \tag{18c}$$

$$C4: \quad l(k,i,t) \in [0,1], k \in K, i \in S. \tag{18d}$$

As mentioned above, the service deployment problem is formulated as a mixed integer nonlinear programming, which is an NP-hard problem. We noticed that deep reinforcement learning algorithms have its natural

advantages on solving this kind of problem [37], so a DQN-based algorithm is designed to address this problem, which will be described in the next content.

Algorithm for collaborative service dynamic deployment

In this section, the interacting services is deployed by a collaborative service dynamic deployment algorithm with DQN model. The detailed information of this algorithm can be found as follows.

DQN algorithm is a typical deep reinforcement learning algorithm which is produced from the Q-learning algorithm [38]. As Fig. 3 shows, the DQN model contains two Q-networks with the same structure and the same initial parameters, which are current value network and target value network. In DQN algorithm, two neural networks are updated with the different frequency through a iterative computation process. During the training process, the model obtain the initial state and the initial action which are selected based on the greedy policy at first, and then the next state is obtained by calculating the rewards. Secondly, the $(s_t^*, a_t, R_t, s_{t+1}^*)$ is stored in the replay memory. With the training steps increased, the parameters of the Q-network are updated and the action value can be calculated to be performed. The details of DQN model can be found in [38].

As the description of DQN model in the above content, we construct state space and action space, and then the reward function is formulated for MDP process. Next we will describe these three elements as below.

State space: In our vehicular edge computing system, the DQN model on cloud servers receives the state of edge clouds at each time slot. Thus, the state space can be expressed as

$$s^*(i,t) = \{c(k,i,t), M(i), D(i), l(k,i,t)\}. \tag{19}$$

Action space: Assuming there are K services deployed on S edge clouds. As mentioned in System model section, we defined the service deployment function $b(k,i,t) \in \{0,1\}$. Therefore, the action space of services deployment is 2^{S*K}. Besides the services deployment, we also considered the computation resource allocation during the services deployment progress. In this vehicular edge computing system, we defined the minimum allocation unit is $\Delta l(k,i,t)$, thus the schema of computation resource allocation follows the below expression.

$$l(k,i,t) = \{\Delta l(k,i,t), ...m\Delta l(k,i,t), ...1\}. \tag{20}$$

Therefore, the action space of edge cloud i at time slot t can be formulated as

$$A_i(t) = \{b(k,i,t), \Delta l(k,i,t), k \in K\}. \tag{21}$$

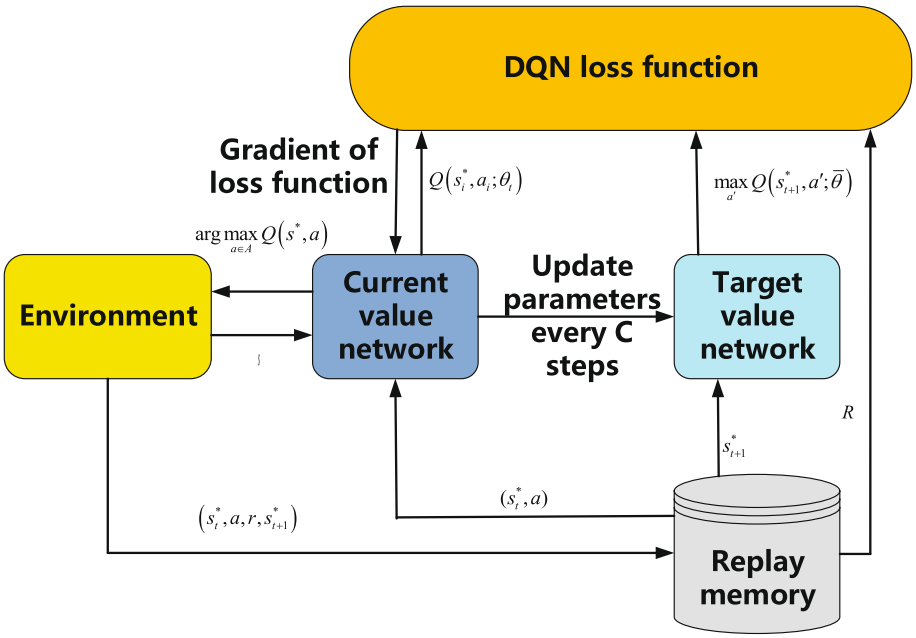

Fig. 3 DQN model

Reward: The purpose of this paper is searching the optimal deployment strategy and solving the minimization problem of service response time with data transmission delay between interacting services. We let $P = \sum_{t=1}^{T} \sum_{k=1}^{K} W^{sum}(k,t)$. Thus, the reward function can be obtained as

$$R(t) = \frac{\Delta w}{T}. \tag{22}$$

In this model, the action A_t is performed, and then the state of next time slot s^*_{t+1} is obtained. We use Δw denote the difference of response time between two states, which can be calculated as

$$\Delta w = \frac{1}{a}[P(s^*_{t+1} \mid s^*_t, A_t) - P(s^*_t)], \tag{23}$$

where a is a constant item.

As the Equation. 22 shows, our purpose can be transferred into the optimization for maximizing the reward function. So the action value function $Q(s^*, a)$ can be calculated as

$$Q(s^*, a) = \mathbb{E}(\sum_{t=1}^{T} \gamma^t R_t \mid s^*_t = s^*, A_t = a), \tag{24}$$

where γ denotes the discount factor, and $\gamma \in [0, 1]$. Thus, the action of searching the optimal service deployment a^* can be expressed as the optimization for maximizing the action value.

$$a^* = arg \max_{a \in A} Q(s^*, a). \tag{25}$$

During this progress, the loss function $L(\theta_t)$ can be obtained by

$$L(\theta_t) = \frac{1}{|J|} \sum_{j=1}^{|J|} [R_j + \gamma \max_{a'} Q(s^*_{i+1}, a'; \bar{\theta}) - Q(s^*_i, a_i; \theta_t)]^2. \tag{26}$$

The gradient descent method is employed to update the parameter θ, which can be expressed as

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t), \tag{27}$$

where η denotes the learning rate, and the parameter θ can be updated through $\mathcal{C}$ steps.

With the MDP process described, the interacting services are deployed by CODD-DQN algorithm, which is a iterative process. The details can be found in Algorithm 2.

Input: State of edge cloud $s^*(i,t)$; the hyper-parameters of DQN; the initial value of θ
Output: Strategy of service dynamic deployment
for episode =1 to E **do**
  The initial state s^*_t is received
  for time slot $t = 1$ to T **do**
    The initial action a_t is selected to be performed through $\varepsilon-$ greedy policy
    The action a_t is performed, and then the next state s^*_{t+1} is obtained by calculating the rewards r_t.
    $(s^*_t, a_t, R_t, s^*_{t+1})$ is stored in the replay memory J
    A mini-batch $J = (s^*_i, a_i, R_i, s^*_{i+1})_{i=1}^{|J|}$ is sampled from the replay memory
    The loss function $L(\theta_t)$ is calculated by Eq. 26
    The parameter θ is updated by Eq. 27
    After $\mathcal{C}$ steps, update θ
  end for
end for
Get the parameter θ
Compute the action value function $Q(s^*_t, a; \theta)$
Calculate $a_t = arg \max Q(s^*_t, a; \theta)$

Algorithm 2 Algorithm for Collaborative Dynamic Service Deployment with DQN

Experimental evaluation

Next, we evaluate the efficacy of proposed algorithms, including service demands algorithm and CODD-DQN algorithm. First, the accuracy of service demands prediction algorithm is evaluated by real-world dataset, and then the simulation experiments are conducted to evaluate the efficiency of CODD-DQN by comparing with other baseline algorithms.

Experiment setting

In this paper, a real-life ISP dataset in China is employed to evaluated the accuracy of service demands prediction, which records more than 480,000 records of mobile users invoking about 16,000 base stations in three cities [39]. We random select continuous 80 hours records from the dataset to record the service demands from these base stations. We conduct the experiments with four metrics to evaluate the accuracy of service demands prediction algorithm, which are root mean square error (RMSE), mean square error (MSE), mean absolute percentage error (MAPE) and mean absolute error (MAE). We vary the proportion of observation data from 50% to 90% to forecast the remain data values and compared with other common prediction algorithms, which are simple exponential smoothing (SES), move average (MA) and autoregressive (AR).

Besides the accuracy of our service demands prediction approach, we also conduct the CODD-DQN algorithm with simulation experiments and compare the average response time with following algorithms.

- Random: Deploying the services randomly under the constraint of the data size of services and the storage capacity of edge clouds.
- Greedy: Deploying the services and allocating the computation resource according to the computation requirement for executing the service. Thus the service with high computation requirement are deployed on the edge cloud with priority.
- Frequency: Deploying the services and allocating the computing resource according to the frequency of the service request.
- Q-Learning: Q-Leaning based service deploying algorithm [40].
- DQN w.o. collaboration: DQN-based service deploying algorithm without considering the interrelationship between interacting services.

In this paper, we set the network transmission rate between the edge clouds V_{e2e} and the network transmission rate between the vehicles and edge clouds V_{v2e} are $100Mbps$. The data size of the services follow the random value from $2GB$ to $8GB$, and the value of computation requirements for executing services are randomly from $1gigacycles$ to $5gigacycles$. To indicate the heterogeneity of the edge clouds, the storage capacity of edge clouds are set as the random value from $10GB$ to $30GB$, and the computing capacity of edge clouds follows the random value from $5GHz$ to $10GHz$. In DQN algorithm, we set the size of experience pool as 3000 and construct neural network with a single hidden layer, whose number of nodes is 128. In our algorithm, the ε-greedy strategy is used, where the initial value of ε is 0.9, and decreases with 0.0005 decrement. After several test, we set batch-size is 64. All of the simulation parameters can be found in Table 2.

Results analysis

First, we evaluate the accuracy of the service demands prediction using the real-life dataset, and vary the proportion of observation data from 50% to 90% to forecast the future number of service request. In this paper, our algorithm are compared with other baseline algorithms. From Fig. 4, we find the accuracy of our algorithm is higher than other baseline algorithms. As Fig. 4a shows, with the training set increases from 50% to 90%, the MSE decreases from 4489 to 100. When the training set is 90%, the MSE value remain 100, which indicate the higher accuracy can be obtained by our service demand prediction algorithm, therefore we have rich time to caching the service beforehand. Besides the MSE, we also conduct the experiments by other metrics. In Fig. 4b, we know the RMSE value decreases from 67 to 12 rapidly, when the proportion increases from 50% to 70%. The RMSE remains 10 when the proportion is 90%. As Fig. 4c and d show that with the proportion increases, the accuracy of prediction increases following. From Fig. 4c we can find, with the proportion increases from 50% to 70%, the MAE of our algorithm decreases rapidly, and achieves at 11.1 when the proportion is 70%. As the proportion increases from 70% to 90%, the MAE decreases slowly, and achieves at 8.83 when the proportion is 90%. As Fig. 4d shows, as the proportion increases from 50% to 70%, the MAPE of our algorithm decreases from 19.8% to 3.64%, and achieves at 3.27% when the proportion is 90%.

With the accuracy of the service demands prediction evaluated, we also evaluate the efficiency of service dynamic deployment algorithm with simulation experiments. In DQN model, we set the initial value of the greedy strategy parameter ε is 0.9 and decrement value is 0.0005. First, the hyper-parameters in our algorithm are determined through the training progress. As Fig. 5

Table 2 Simulation parameters

Parameters	Value
Data size of service, $d(k)$	$[2, 8]GB$
Computing capacity for executing service, $m(k)$	$[1, 5]gigacyles$
Storage capacity of edge cloud, $D(i)$	$[10, 30]GB$
Computing capacity of edge cloud, $M(i)$	$[5, 10]GHz$
Network transmission rate between multi edge clouds, v_{e2e}	$100Mbps$
Network transmission rate between vehicles and edge clouds, v_{v2e}	$100Mbps$
Initial value of greedy strategy parameter, ε	0.9
Decrement value of ε	0.0005
Number of nodes in hidden layer	128
Size of experience pool	3000
Batch size	64

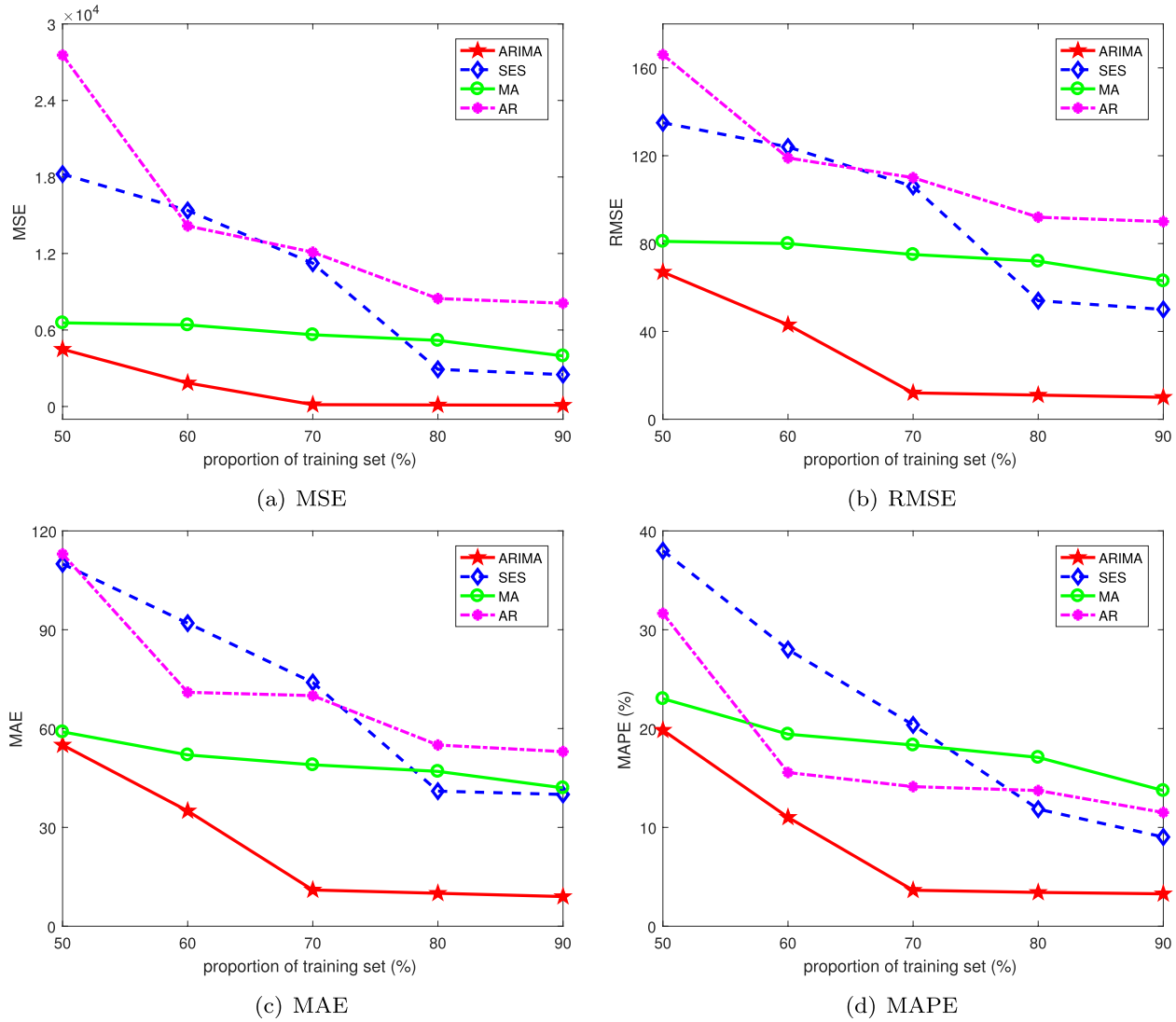

Fig. 4 Comparison between different service demands prediction algorithms

shows, the algorithm can obtain the best performance when the discount factor γ is 0.9, The average response time can reach about 0.65s when the episode decreases at 400. So the optimal discount factor is set as 0.9.

Furthermore, we determine the learning rate by several experiments. Figure 6 shows the convergence performance comparison of the algorithm with different learning rates η. From this Figure, we notice the CODD-DQN performs best performance when $\eta = 0.0001$, while the algorithm is not convergence when $\eta = 0.001$ and $\eta = 0.0005$. Therefore, we set the value of learning rate as 0.0001.

Since the hyper-parameters are determined, we evaluate the performance of our algorithm to compare with other algorithms. Figure 7 shows the average response time of different algorithms. We can see that our CODD-DQN algorithm can achieve the lowest average response time than the four algorithms. As Fig. 7 shows, with the number of episode increases, the Q-learning algorithm is not convergence, while our CODD-DQN algorithm can obtain the average response time about 0.65s when the episode is 400. Compared with DQN w.o. collaboration algorithm, our algorithm achieves the lower average response time than DON w.o. collaboration algorithm, and converges at 400 episodes, while the DQN w.o. collaboration algorithm converges at about 600 episodes. Because the DQN w.o. collaboration algorithm deploys the services without considering the relationships between interacting services, which may increase the data communication delay between interacting services.

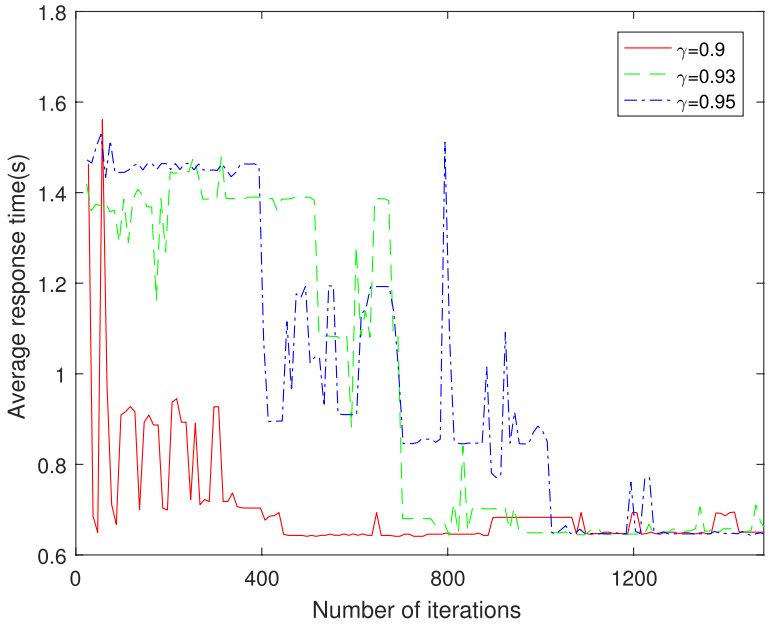

Fig. 5 Convergence performance of CODD-DQN algorithm with different discount factors

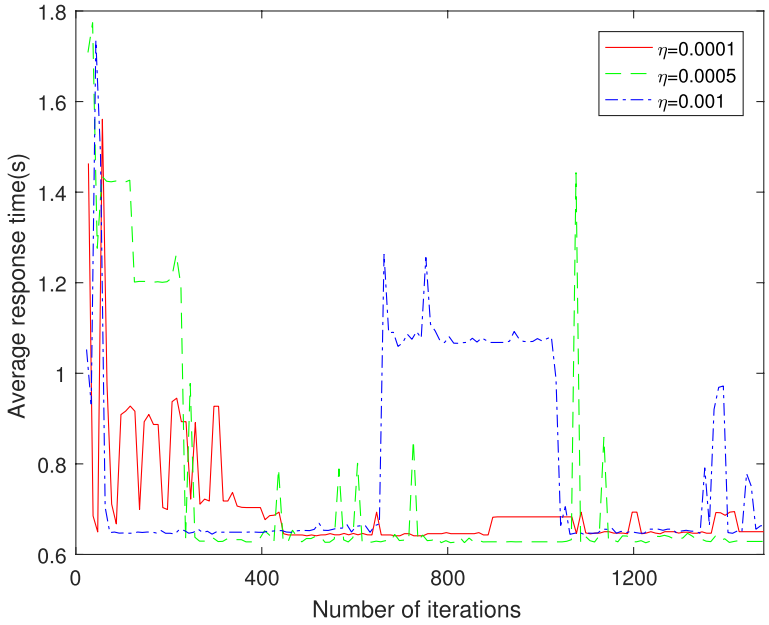

Fig. 6 Convergence performance of CODD-DQN algorithm with different learning rates

We also conduct the experiments under different system simulation parameters. Since the Q-learning algorithm cannot converge, we only compare the average response time of our algorithm with other baseline algorithms. First, we evaluate the service response time with different values of storage capacity. Figure 8 show the convergence performance and service response time comparison under different storage capacity. The performance of CODD-DQN algorithm and DQN w.o. collaboration algorithm can be found in Fig. 8a. We notice that the smaller the storage capacity of edge clouds, the higher response time of the algorithm. The CODD-DQN algorithm can achieve the lower response time than DQN w.o. collaboration algorithm,

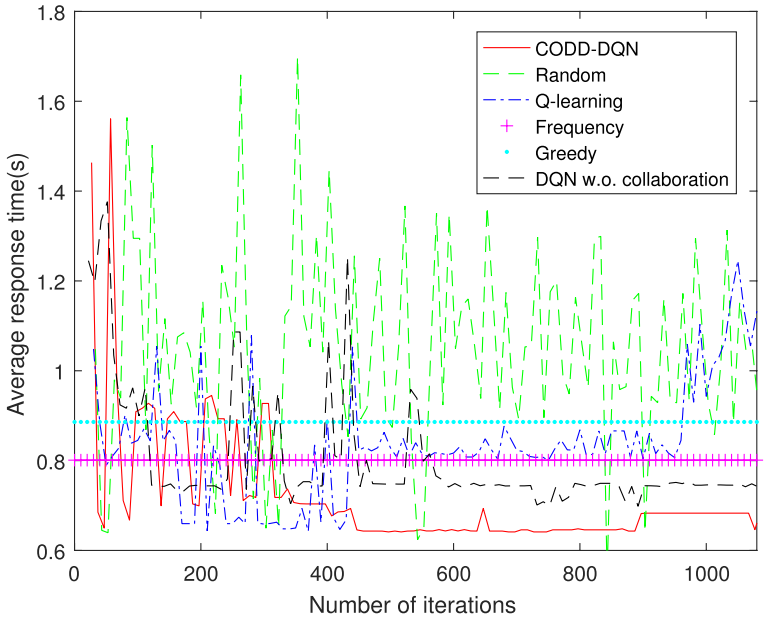

Fig. 7 Convergence performance comparison of different algorithms

and converges about at 0.7*s* when the storage capacity is 20*GB*. Figure 8b shows the service response time comparison between CODD-DQN algorithm and other baseline algorithms under different storage capacity of edge clouds. From the Figure, we can see the CODD-DQN algorithm can obtain the lowest response time than other algorithms. With the storage capacity increased from 10*GB* to 30*GB*, the response time decreases following, and the response time of our CODD-DQN algorithm remains at about 0.67*s* when the storage capacity increases at 30*GB*.

Figure 9 show the convergence performance and service response time comparison of the algorithms under different values of the number of services. From the Fig. 9a we know the service response time of two DRL-based algorithms with the number of services is 10 are higher than that when the number of services is 8. Thus, the more services, the higher response time in our system. We also found that the CODD-DQN algorithm can obtain the lower response time than DQN w.o. collaboration algorithm, and Converges at about 0.59*s* when the number of services is 8. Figure 9b shows the service response time comparison between CODD-DQN algorithm and other baseline algorithms under different values of the number of services. With the number of services increased from 4 to 12, the response time of CODD-DQN algorithm increases from 0.31*s* to 1.28*s*, and achieves the lowest response time than other algorithms.

Besides these experiments, we also conduct the experiments under other different system parameters. we vary the computing capacities of the edge clouds and conduct the performance of different algorithms. Figure 10 show the group results of convergence performance and service response time comparison of the algorithms under different computing capacities of edge clouds. From the Fig. 10a we know the service response time of two DRL-based algorithms with the computing capacity of edge clouds is 6 *GHZ* are higher than that when the computing capacity of edge clouds is 8 *GHZ*. Thus, the higher computing capacity of edge clouds , the lower response time in our system. We also found that the CODD-DQN algorithm can obtain the lower response time than DQN w.o. collaboration algorithm. Figure 10b shows the service response time comparison between CODD-DQN algorithm and other baseline algorithms under different computing capacities. With the computing capacities increased from 6 *GHZ* to 10 *GHZ*, the response time of CODD-DQN algorithm decreases from 0.85*s* to 0.74*s*, and achieves the lowest response time than other algorithms.

In order to indicate the performance of algorithms under different number of edge clouds, we also vary the number of edge clouds and compare the performances of different algorithms. Figure 11 show the group results of convergence performance and service response time comparison of the algorithms under different number of edge clouds. From the Fig. 11a we know the service

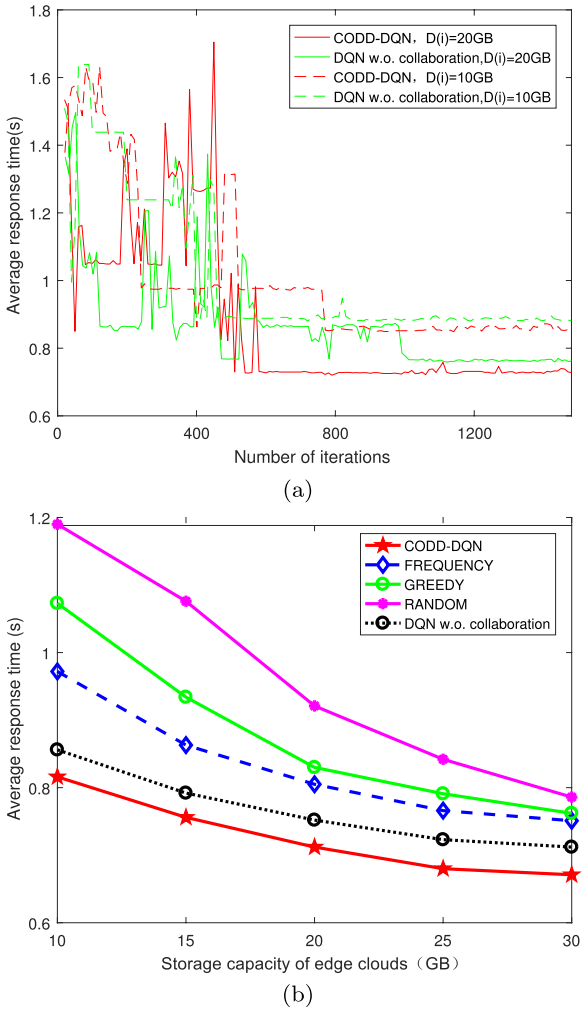

Fig. 8 **a** Convergence performance of different algorithms with different storage capacity of edge clouds. **b** Comparison between different algorithms

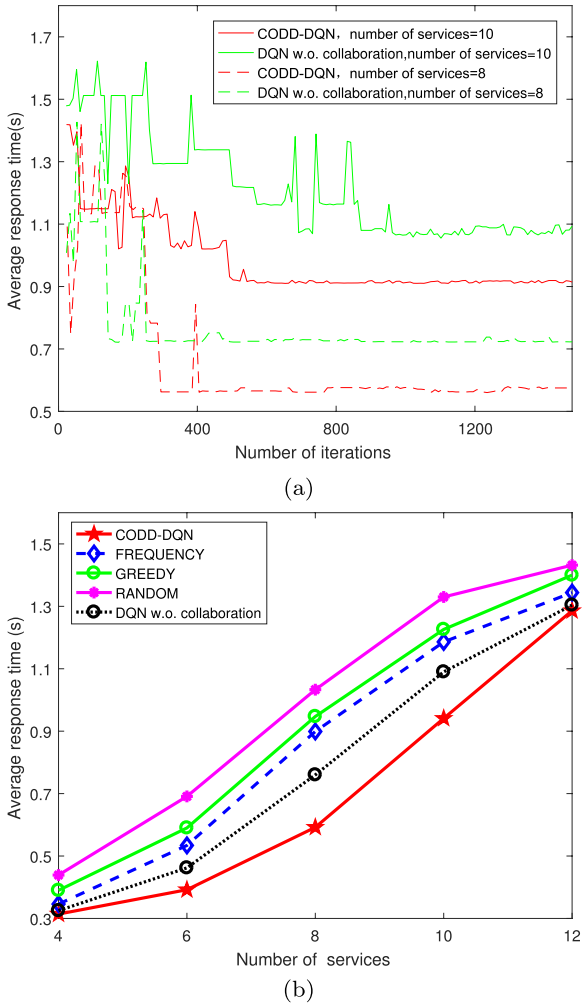

Fig. 9 **a** Convergence performance of different algorithms with different number of services. **b** Comparison between different algorithms

response time of two DRL-based algorithms with the number of edge clouds is 3 are higher than that when the number of edge clouds is 5. Thus, the more edge clouds, the lower response time in our system. We also found that the CODD-DQN algorithm can obtain the lower response time than DQN w.o. collaboration algorithm. Figure 11b shows the service response time comparison between CODD-DQN algorithm and other baseline algorithms under different values of the number of edge clouds. With the number of edge clouds increased from 3 to 7, the response time of CODD-DQN algorithm decreases from 0.81*s* to 0.68*s*, and achieves the lowest response time than other algorithms.

Conclusion

In this paper, A collaborative service on-demand dynamic deployment approach via DQN model is proposed in vehicular edge computing, which is named CODD-DQN. To investigate the temporal dynamic characteristics of service request, a time-aware service demands prediction algorithm by ARIMA model is produced to forecast the number of service request for each edge cloud, and then the interacting services are discovered through the analysis of the service invoking logs. Furthermore, the service response time models are constructed to formulate the service deployment as an optimization problem, and the collaborative service deployment algorithm is presented

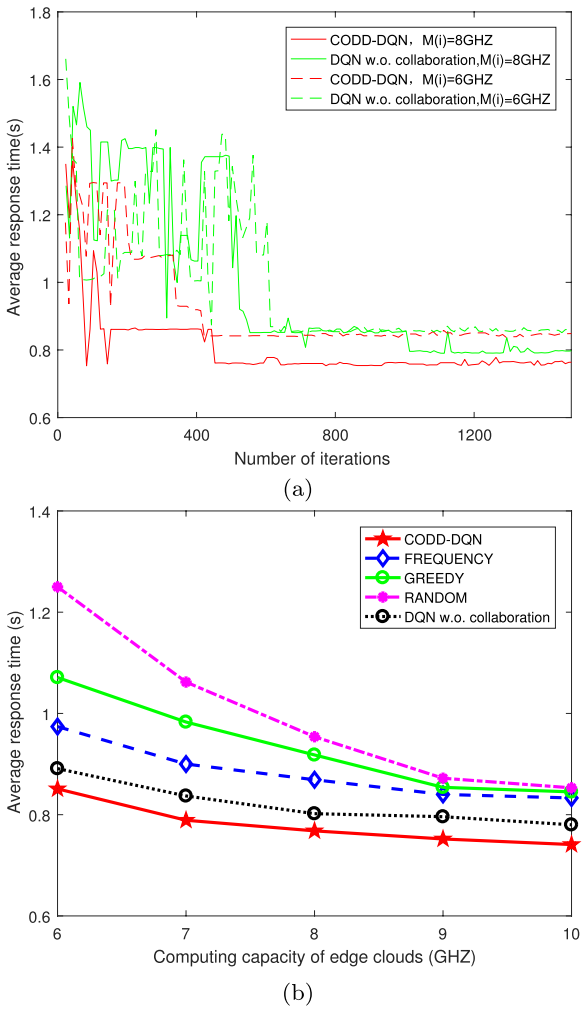

Fig. 10 **a** Convergence performance of different algorithms with different computing capacities of edge clouds. **b** Comparison between different algorithms

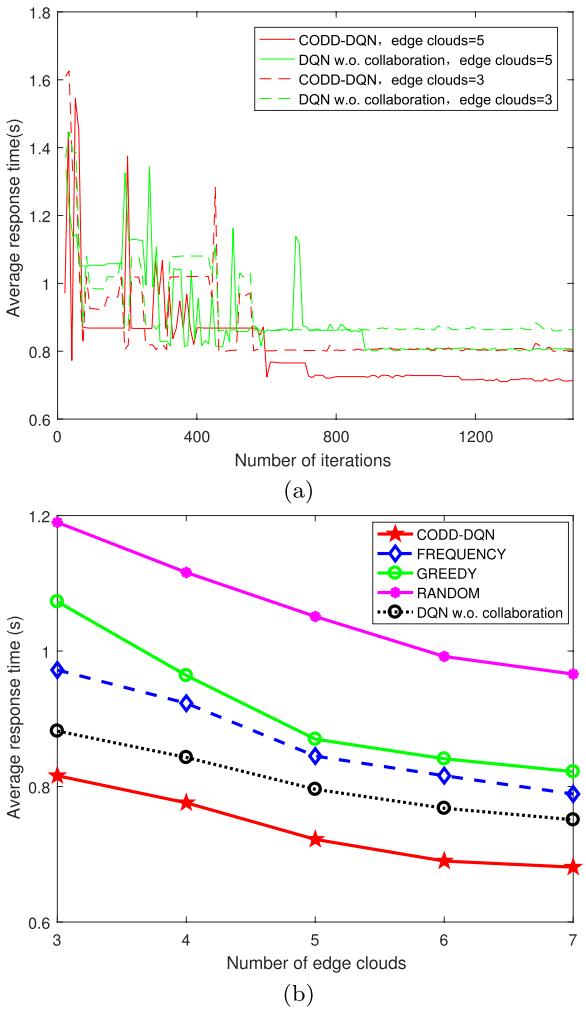

Fig. 11 **a** Convergence performance of different algorithms with different number of edge clouds. **b** Comparison between different algorithms

by DQN model to deploy the interacting services, which can solve the minimization problem of service response time with data transmission delay. Finally, the real-life dataset based experiments are conducted to evaluate the efficiency of the algorithms. The results show proposed CODD-DQN algorithm can achieve lowest service response time than other algorithms on deploying the interacting services.

Noticed that our purpose is to design approach for service dynamic deployment by forecasting the number of service request with efficiency. To improve the utilization of the computation resource, we also design a primer resource allocation function during service deployment. Note that the resource allocation is a complex problem which is need to be studied, and thus the detail schema of resource should be designed. In the future, we plane to design a detail resource allocation strategy to improve the utilization of the resource. Besides this, we also notice the efficacy of our algorithms are only evaluated by simulation experiment in laboratory environments due to the limitation of hardware. We will construct the real vehicular edge computing environment to evaluate efficiency and improve the performance of the algorithms.

Acknowledgements
The authors would like to thank the anonymous reviewers for their insightful comments and suggestions on improving this paper.

Authors' contributions
Yuze Huang conceived the initial ideal and designed the algorithms, and wrote the paper. Beipeng Feng designed system model and carried out the experiments. Yuhui Cao analyzed the experimental data. Zhenzhen Guo contributed to data collection and analysis. Miao Zhang and Boren Zheng proofread the manuscript. The authors read and approved the final manuscript.

Funding
This work is sponsored by Natural Science Foundation of Chongqing, China (No. CSTB2022NSCQ-MSX0368), and Young Project of Science and Technology Research Program of Chongqing Education Commission of China (No. KJQN202200702, No. KJQN201900708).

Availability of data and materials
The datasets used during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 28 December 2022 Accepted: 16 July 2023
Published online: 08 August 2023

References

1. Contreras-Castillo J, Zeadally S, Ibáñez JAG (2018) Internet of Vehicles: Architecture, Protocols, and Security. IEEE Internet Things J. 5(5):3701–3709
2. Wang X, Ning Z, Hu X, Wang L, Hu B, Cheng J et al (2019) Optimizing Content Dissemination for Real-Time Traffic Management in Large-Scale Internet of Vehicle Systems. IEEE Trans Veh Technol. 68(2):1093–1105
3. Singh D, Singh M (2015) Internet of vehicles for smart and safe driving. International Conference on Connected Vehicles and Expo, ICCVE 2015, October 19-23, 2015. IEEE, Shenzhen, pp 328–329
4. Hussain R, Kim D, Son J, Lee J, Kerrache CA, Benslimane A et al (2018) Secure and Privacy-Aware Incentives-Based Witness Service in Social Internet of Vehicles Clouds. IEEE Internet Things J. 5(4):2441–2448
5. Zhang M, Wang S, Gao Q (2020) A joint optimization scheme of content caching and resource allocation for internet of vehicles in mobile edge computing. J Cloud Comput. 9:33
6. Wu L, Zhang R, Li Q, Ma C, Shi X (2022) A mobile edge computing-based applications execution framework for Internet of Vehicles. Frontiers Comput Sci. 16(5):165506
7. Zhang J, Letaief KB (2020) Mobile Edge Intelligence and Computing for the Internet of Vehicles. Proc IEEE. 108(2):246–261
8. Chen Y, Zhao J, Zhou X et al (2023) A Distributed Game Theoretical Approach for Credibility-guaranteed Multimedia Data Offloading in MEC. Inf Sci. 644:119306. https://doi.org/10.1016/j.ins.2023.119306
9. Zhang Y (2022) Mobile Edge Computing, vol 9. Springer, Cham
10. Ning Z, Huang J, Wang X, Rodrigues JJPC, Guo L (2019) Mobile Edge Computing-Enabled Internet of Vehicles: Toward Energy-Efficient Scheduling. IEEE Netw. 33(5):198–205
11. Wang S, Urgaonkar R, He T, Chan K, Zafer M, Leung KK (2017) Dynamic Service Placement for Mobile Micro-Clouds with Predicted Future Costs. IEEE Trans Parallel Distrib Syst. 28(4):1002–1016
12. Hao Y, Chen M, Cao D, Zhao W, Petrov I, Antonenko VA et al (2020) Cognitive-Caching: Cognitive Wireless Mobile Caching by Learning Fine-Grained Caching-Aware Indicators. IEEE Wirel Commun. 27(1):100–106
13. Chen L, Zhou P, Gao L, Xu J (2018) Adaptive Fog Configuration for the Industrial Internet of Things. IEEE Trans Ind Inform. 14(10):4656–4664
14. Wang L, Jiao L, He T, Li J, Mühlhäuser M (2018) Service Entity Placement for Social Virtual Reality Applications in Edge Computing. 2018 IEEE Conference on Computer Communications, INFOCOM 2018, April 16-19, 2018. IEEE, Honolulu, pp 468–476
15. Aït-Salaht F, Desprez F, Lebre A (2021) An Overview of Service Placement Problem in Fog and Edge Computing. ACM Comput Surv 53(3):65:1-65:35
16. Poularakis K, Llorca J, Tulino AM, Taylor IJ, Tassiulas L (2019) Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks. 2019 IEEE Conference on Computer Communications, INFOCOM 2019, April 29 - May 2, 2019. IEEE, Paris, pp 10–18
17. Ma X, Zhou A, Zhang S, Wang S (2020) Cooperative Service Caching and Workload Scheduling in Mobile Edge Computing. 39th IEEE Conference on Computer Communications, INFOCOM 2020, July 6-9, 2020. IEEE, Toronto, pp 2076–2085
18. Chen Y, Zhao J, Hu J et al (2023) Distributed Task Offloading and Resource Purchasing in NOMA-enabled Mobile Edge Computing: Hierarchical Game Theoretical Approaches. ACM Trans Embed Comput Syst. early access. https://doi.org/10.1145/3597023
19. Hao Y, Chen M, Gharavi H, Zhang Y, Hwang K (2021) Deep Reinforcement Learning for Edge Service Placement in Softwarized Industrial Cyber-Physical System. IEEE Trans Ind Informatics. 17(8):5552–5561
20. Wang R, Kan Z, Cui Y, Wu D, Zhen Y (2021) Cooperative Caching Strategy With Content Request Prediction in Internet of Vehicles. IEEE Internet Things J. 8(11):8964–8975
21. Hui Y, Ma X, Su Z, Cheng N, Yin Z, Luan TH et al (2022) Collaboration as a Service: Digital-Twin-Enabled Collaborative and Distributed Autonomous Driving. IEEE Internet Things J. 9(19):18607–18619
22. Chen H, Qin W, Wang L (2022) Task partitioning and offloading in IoT cloud-edge collaborative computing framework: a survey. J Cloud Comput. 11:86
23. Huang J, Gao H, Wan S et al (2023) AoI-aware energy control and computation offloading for industrial IoT. Futur Gener Comput Syst. 139:29–37
24. Chen Y, Zhao J, Wu Y et al (2022) QoE-aware Decentralized Task Offloading and Resource Allocation for End-Edge-Cloud Systems: A Game-Theoretical Approach. IEEE Trans Mob Comput. early access.1–17. https://doi.org/10.1109/TMC.2022.3223119
25. Chen Y, Hu J, Zhao J, Min G (2023) QoS-Aware Computation Offloading in LEO Satellite Edge Computing for IoT: A Game-Theoretical Approach. Chin J Electron. early access. https://doi.org/10.23919/cje.2022.00.412
26. LiWang M, Gao Z, Hosseinalipour S, Dai H (2020) Multi-Task Offloading over Vehicular Clouds under Graph-based Representation. 2020 IEEE International Conference on Communications, ICC 2020, June 7-11, 2020. IEEE, Dublin, pp 1–7
27. Chen Y, Gu W, Xu J et al (2022) Dynamic Task Offloading for Digital Twin-empowered Mobile Edge Computing via Deep Reinforcement Learning. Chin Commun. early access. 1–12. https://doi.org/10.23919/JCC.ea.2022-0372.202302
28. Hegyi P (2022) Service deployment design in latency-critical multi-cloud environment. Comput Netw. 213:108975
29. Lima D, Miranda H (2022) A geographical-aware state deployment service for Fog Computing. Comput Netw. 216:109208
30. Huang J, Lv B, Wu Y et al (2022) Dynamic Admission Control and Resource Allocation for Mobile Edge Computing Enabled Small Cell Network. IEEE Trans Veh Technol. 71(2):1964–1973
31. Chen Y, Xing H, Ma Z, et al (2022) Cost-Efficient Edge Caching for NOMA-enabled IoT Services. Chin Commun
32. Huang J, Wan J, Lv B, Ye Q et al (2023) Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning. IEEE Syst J. 17(2):2500–2511. https://doi.org/10.1109/JSYST.2023.3249217
33. Huang Y, Cao Y, Zhang M, Feng B, Guo Z (2022) CSO-DRL: A Collaborative Service Offloading Approach with Deep Reinforcement Learning in Vehicular Edge Computing. Sci Prog. 2022:1163177. https://doi.org/10.1155/2022/1163177
34. Huang Y, Huang J, Cheng B, Yao T, Chen J (2017) Poster: Interacting Data-Intensive Services Mining and Placement in Mobile Edge Clouds. Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking, MobiCom 2017, October 16 - 20, 2017. ACM, Snowbird, pp 558–560
35. Huang Y, Huang J, Liu C, Zhang C (2020) PFPMine: A parallel approach for discovering interacting data entities in data-intensive cloud workflows. Future Gener Comput Syst. 113:474–487

36. Box GEP, Jenkins GM (2015) Time Series Analysis: Forecasting and Control, 5th edn. Wiley, Hoboken
37. Chen W, Qiu X, Cai T, Dai H, Zheng Z, Zhang Y (2021) Deep Reinforcement Learning for Internet of Things: A Comprehensive Survey. IEEE Commun Surv Tutorials. 23(3):1659–1692
38. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG et al (2015) Human-level control through deep reinforcement learning. Nat. 518(7540):529–533
39. Liu H, Li Y, Wang S (2022) Request Scheduling Combined with Load Balancing in Mobile Edge Computing. IEEE Internet of Things. 9(21):20841–20852. https://doi.org/10.1109/JIOT.2022.3176631
40. Suton RS, Barto AG (2018) Reinforcement Learning, 2nd edn. MIT Press, Cambridge

Publisher's Note