RESEARCH ARTICLE OPEN ACCESS

Edge Server Deployment Based on Reinforcement Learning in Mobile Edge Computing

Zixiang Wang*, Jipeng Zhou**
*(Department of Computer Science, Jinan University, Guangzhou 510632, China
Email: 1263448722@qq.com)
**(Department of Computer Science, Jinan University, Guangzhou 510632, China
Email: jpzhoucn@sohu.com)

----------**********----------

Abstract:

Mobile Edge Computing (MEC) is to sink the resource of remote cloud computing center to the edge network to provide users with better services. Distinct from cloud computing, mobile edge computing is under the constraint of edge server computing resources, deployment location, wireless transmission bandwidth and etc. Although there has been significant research in the field of mobile edge computing, little attention has been given to understanding the placement of edge servers to optimize the mobile edge computing network performance. In this paper, we propose a server deployment scheme based on reinforcement learning. Firstly, we propose the MEC three-tier architecture, which takes the latency of the base station as the optimization goal. Then, we formulate the edge server deployment problem as a single-objective optimization problem, and propose the edge server deployment (Q-ESD) algorithm in this paper based on the Q-Learning algorithm. Finally, experimental results show that our approach outperforms several representative approaches in terms of access delay and workload balancing.

Keywords — **Mobile edge computing, Reinforcement learning, Server deployment, Load balancing**.

----------**********----------

I. INTRODUCTION

With the development of IoT and 5G networks in smart city environments, mobile communication traffic has experienced explosive growth over the past few years. Mobile intelligent devices have become increasingly important as tools for entertainment, learning, social networking, and businesses for smarter living [1]. The explosive growth of terminal equipment, such as various sensors, smartwatches, cameras, poses new challenges to the transmission capacity, transmission rate, data distribution and processing capability, and data security for the entire network. The development of the application requirements of the Internet of Everything has given birth to the edge big data processing model, namely edge computing model [2]. The European Telecommunications Standards Institute (ETSI) established a mobile edge computing specification working group in 2014 and formally promoted the standardization of mobile edge computing. The basic idea is tantamount to migrate the cloud computing platform from the core network to the edge of the mobile access network to achieve elastic utilization of computing and storage

resources [3][4]. Mobile edge computing is an effective way to alleviate the long latency problem of users and improve the current network architecture. In mobile edge computing, computing resources will sink to the user side with the deployment of edge servers, which can process user's requests closer to the user. Such a network architecture can bring two benefits: 1) For downstream data, edge servers play the role of cloud service providers, bringing computing resources close to end-users, so that the latency of service requests can be very low; 2) For upstream data, it helps to relieve the network transmission pressure on the core network [5]. In some high-bandwidth, low-latency innovative services, such as augmented reality [6], deep learning [7], smart city [8], and other application scenarios, many scholars have carried out realistic deployment analyses of them.

At present, a lot of research focuses on access delay and energy consumption of terminal devices. Reference [9] studied the problem of placing a limited number of edge servers and assigning users to edge servers in a large-scale wireless metropolitan area network to minimize the average waiting time for migration tasks. Reference [10] studied the placement of wireless APs in large-scale wireless metropolitan area networks. They considered placing micro-clouds in different strategic locations to lower user access latency. Then, by treating it as an integer linear programming problem, a heuristic solution was proposed. Reference [11] studied the problem of minimizing the number of edge servers while guaranteeing Quality of Service (QoS) constraints (such as access delay) and gave the integer linear programming formulation of the problem. Reference [12] took the energy consumption of the edge server as the sensing target, expressed the problem as a multi-objective optimization problem, and designed a particle swarm optimization algorithm to find the optimal deployment scheme to decrease the energy consumption of the edge servers. The results showed that this method can reduce energy consumption by more than 10% and increase the utilization rate of computational resources by more than 15%. Some scholars consider the user's access delay and the load balance of the edge server, and use traditional optimization algorithms to solve it. Reference [13] mainly studied the deployment of edge servers in smart city mobile edge computing environments. By placing servers at some base stations to reduce the access latency of mobile users and balance the workload of edge servers, they described the problem as of constrained optimization problem and employed mixed integer programming to find the optimal edge server layout.

This paper mainly focuses on the deployment of edge servers. In this section, we weaken the relationship between mobile users and the base stations they access and directly use the workload of each base station to let mobile users participate in the model. We design a logical three-tier network architecture, which mainly includes the base station layer, the edge server layer, the remote computing center layer. The waiting time of the base stations is considered as the optimization objective, and the load balance of the edge server is considered when allocating the base stations to the edge server. Then the system model in this paper is combined with the basic framework of reinforcement learning, and the system states, actions and rewards are defined in conjunction with the deployment of edge servers. Finally, the Q-Learning Edge Server Deployment (Q-ESD) algorithm based on reinforcement learning is designed to solve it. Simulation results show that the Q-ESD algorithm performs better than optimization algorithms such as the k-means for balancing server workload and reducing user access delay. The structure of this paper is reproduced below. The first section introduces the background and related works. Section II present a formulaic definition of the proposed model. Section III transform the corresponding problem into an optimization problem. In section IV we design the optimization algorithm based on reinforcement learning. For section V, numerical simulations of different methods are carried out. The conclusions of this article are located in Section VI.

II. SYSTEM MODEL

In mobile edge network, as shown in Fig1, edge network is considered as a three-tier architecture including cloud computing center level, edge server level, and the base station level. The

edge network consists of a set of base stations $\mathcal{B} = \{b_1, \dots, b_i, \dots, b_m\}$ connected through the Internet and a set of edge servers $\mathcal{S} = \{s_1, \dots, s_j, \dots, s_k\}$ providing computing resources. We use an undirected graph $G(V, E)$ to represent the relationship between base stations and edge servers. There are for two types of edges in the graph, the edge of base station b_i and base station b_o $((b_i, b_o) \in E)$ means that there is a one-hop distance between b_i and b_o, and the edge $((b_i, s_j) \in E)$ of base station b_i and edge server s_j means that base station b_i can communicate directly with edge server s_j through the edge (b_i, s_j). We assume that graph G is connected, which means that all base stations can be attached to each other through communication links. In addition, each edge server can directly access the remote cloud computing center through the Core Network.

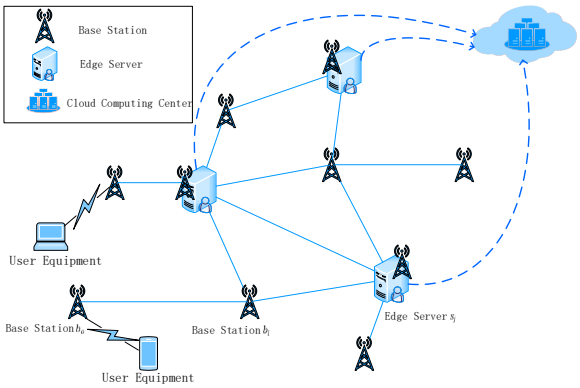

Fig1 Mobile Edge Network Architecture

In the edge network, mobile users will offload computing tasks to the neighboring base station through the wireless network. The base station forwards the user request to the edge server for processing. We define the user request received by the base station b_i $(b_i \in B, i = 1, \dots, m)$ as the workload of b_i, which is represented by the symbol w_{b_i}. Base stations are densely deployed intensively in edge networks. When selecting base stations to deploy edge servers, some base stations are selected to co-deploy. This solution not only brings a higher quality of service (QOS) to mobile users but also reduces the costs of operators. Taking Fig1 as an example, the workload of base station b_o needs to be forwarded to server s_j by base station b_i. We assume that base station b_i will allocate resources to process data forwarding, the data to be forwarded will not have a waiting delay at the base station b_i .To simulate the network delay of the edge network, $D \in \mathbb{R}^{m \times m}$ is used to represent the delay matrix of the edge network, where $D_{o,i}$ represents the delay from base station b_o to base station b_i, when $i = o$, there is $D_{o,i} = 0$.

The purpose of the base station allocation strategy in the edge network is to allocate base stations to the adjacent edge server. After determining the physical location of the edge server s_j, the base station with the minimum network delay with the edge server s_j is assigned to s_j. Since the edge server is deployed in coordination with the base station in the physical location, the network delay between the base station and s_j can be obtained by the delay matrix D. The specific forwarding strategy is shown in Algorithm 1

Algorithm 1: Base Station Allocation Strategy

Input: Undirected graph G, base station number q co-deployed by edge server s_j, base station set $\mathcal{B}$, Delay matrix D, The number of edge servers k, the number of base stations m

Output: The set of base stations B_j processed by the edge server s_j

$B_j \leftarrow \emptyset, R \leftarrow m/k, index \leftarrow -1, x_q \leftarrow 1$
For $i = 1$ *to* R **do**
  $minDelay \leftarrow Float.MAX\_VALUE$
  For $v = 1$ *to* m **do**
    If $D_{v,q} < minDelay$
      $minDelay \leftarrow D_{v,q}$
      $index \leftarrow v$
    End If
  End For
  $y(index, j) \leftarrow 1,\ B_j \leftarrow B_j \cup b_{index}$
  For $v = 1$ *to* m **do**
    $D_{index,v} \leftarrow Float.MAX\_VALUE$
  End For
End For
return B_j

In the algorithm 1, x_q is a binary variable, and x_q=1 indicates that an edge server is deployed at the base station b_q. Let $B_j = \{b_i | y(i, j) = 1\}$ denote the

set of base stations that forward user requests to edge server s_j, where $y(i,j)$ is a binary variable, and $y(i,j) = 1$ means base station b_i forwards user requests to edge server s_j for processing, otherwise $y(i,j) = 0$. Algorithm 1 balances the workload of edge servers by limiting the number of base stations allocated to edge server. When base stations allocated to the edge server s_j exceed the average number of base stations that each edge server should be responsible for in the edge network, no more base stations are allocated to the edge server s_j. Lines 11 to 13 of the algorithm to avoid secondary allocation of base stations to edge server.

III. PROBLEM FORMULATION

A. Computation Model

In the edge network, the edge server we deploy have the same computing resources, assuming that the maximum processing power of the edge server CPU is f. In Fig1, if the base station b_i is assigned to the edge server s_j, the calculation delay of the request received by the base station is expressed as

$$T_j^{cal} = \frac{w_{b_i}}{f} \tag{1}$$

In the edge network, base stations are densely deployed around mobile users. Typically assigning multiple base stations to the nearby edge server. It is assumed that the task processing of the edge server is regarded as a queuing system. According to the queuing theory, we assume that the average arrival rate of task requests is λ_j, the average service rate is μ_j, and the service density is ρ_j. The queuing delay of the task at the edge server s_j is expressed as

$$T_j^{queue} = \frac{\rho_j}{1-\rho_j}\frac{1}{\mu_j} \tag{2}$$

Therefore, for the set of base stations B_j assigned to the edge server s_j, the delay in processing the workload of the base stations in B_j is expressed as

$$T_j = \sum_{b_i \in B_j} (D_{i,j} + \frac{w_{b_i}}{f} + \frac{\rho_j}{1-\rho_j}\frac{1}{\mu_j}) \tag{3}$$

The total delay of all base stations in the edge network is defined as

$$T = \sum_{j=1}^{k} T_j \tag{4}$$

B. Optimization Objective and Problem Description

We introduce two sets of variables X and U to represent the location where edge servers are co-deployed and the assignment relationship between base stations and edge servers. The values of the two sets of variables can be obtained according to the allocation strategy. $X = \{x_i | 1 \le i \le m\}$且$U = \{B_j | 1 \le j \le k\}$. We denote the workload set of base stations by $W = \{w_{b_i} | b_i \in \mathcal{B}\}$.

The edge server deployment problem in the mobile edge network $G = (V, E)$ is defined as follows. Given the edge network G and system model parameters D, W, λ, μ, k, m, find X (the location where the edge servers are co-deployed at the base station) and U (the assignment relationship between base stations and edge servers) to minimize base stations waiting time T in the edge network:

$$\min_{X,U} T \tag{5}$$

IV. DEPLOYMENT ALGORITHM BASED ON Q-LEARNING

A. Reinforcement learning related concepts

Based on the Q-Learning algorithm, the Q-ESD algorithm is proposed according to the characteristics of the server deployment problem. This section mainly describes the relevant definitions of reinforcement learning applied to edge server deployment algorithm, and defines the state, action, and reward required by the deployment algorithm.

State: Before defining the system state, the state-space is presented. The state-space can be represented by a matrix $State$, which has k rows and m columns. Each state is commensurate with a row of the matrix. In the edge server deployment problem, the system state refers to the position of an edge server determined at a certain moment, that is, a row of the $State$ matrix is determined. Each state can be represented by a vector $state_j = [c_{j,1}, \dots, c_{j,i}, \dots c_{j,m}]$, $c_{j,i}$ in the vector is a binary variable, which represents whether the edge server

s_j is deployed at the base station b_i, $c_{j,i} = 1$ means that the edge server s_j is deployed in cooperation with the base station b_i, otherwise $c_{j,i} = 0$. In the edge server deployment problem, the edge server s_j can only be deployed with one base station, so there are the following constraints for $state_j$

$$\sum_{i=1}^{m} c_{j,i} = 1 \tag{6}$$

Update a row of the matrix, where $c_{j,i} = 1$. It is understood that the edge server s_j represented by row j and the base station b_i represented by column i are co-deployed. Updating the matrix represents a complete deployment scheme. The state-space is expressed as

$$State = \begin{bmatrix} state_1 \\ state_2 \\ \cdots \\ state_j \\ \cdots \\ state_k \end{bmatrix} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,m} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,m} \\ \cdots & \cdots & \cdots & \cdots \\ c_{j,1} & c_{j,2} & \cdots & c_{j,m} \\ \cdots & \cdots & \cdots & \cdots \\ c_{k,1} & c_{k,2} & \cdots & c_{k,m} \end{bmatrix} \tag{7}$$

Action: In the edge server deployment problem, the action consists of two parts, determining the deployment location of the server s_j; assigning base stations to s_j according to the base station allocation strategy. The action vector is expressed as $a_j = (state_j, B_j)$. The action-space is expressed as

$$A = \begin{bmatrix} a_1 \\ a_2 \\ \cdots \\ a_j \\ \cdots \\ a_k \end{bmatrix} \tag{8}$$

Reward: In the edge server deployment problem, performing an action adjustment is to determine the location of an edge server s_j and the base stations set B_j allocated to the edge server s_j. From formula (3), it can be known that the time delay for the edge server s_j to process the request in the set B_j is T_j.

In reinforcement learning model, the agent's goal is tantamount to maximize long-term cumulative rewards. The optimization goal of the edge server deployment problem proposed in this paper is to minimize the total delay of the edge network. We set the reward of the action to be a function representation that is negatively related to the delay. The reward function can be expressed as

$$reward_j = e^{-T_j} \tag{9}$$

Evaluation Function: The evaluation function in the Q-Learning algorithm includes immediate reward value, Q-value function, and discount rate. The Q-value table saves the estimated value of each state-action pair $(state, a)$ [14]. According to the given policy $h(x)$, the evaluation function of the Q-Learning algorithm is expressed as

$$Q^{t+1}(state_t, a_t) = Q^t(state_t, a_t) + \alpha[reward + \gamma max\{Q^t(state', a')\} - Q^t(state_t, a_t)] \tag{10}$$

In the formula: $\alpha \in (0,1)$ is the learning rate; $\gamma \in (0,1)$ is the discount rate, which determines how much weight the agent considers future rewards; t is the time step; the reward is the immediate reward when taking the current $(state_t, a_t)$; the max function indicates that the algorithm will evaluate $(state', a')$ according to the maximum value of the predicted value in the next $(state', a')$; in the formula, $reward + \gamma max Q^t(state', a') - Q^t(state_t, a_t)$ is defined as the time difference error (TD error) [15].

B. Edge Server Deployment Algorithm Design

Based on the Q-Learning algorithm, combined with the MEC edge server deployment scenario, the time slot is added to the model. The design points of the q-learning edge server deployment(Q-ESD) algorithm proposed in this paper are as following:

(1) In time slot t, in the state-space $State^t = \{state_1^t, \ldots, state_j^t \ldots, state_k^t\}$, each state means the deployment of an edge server in slot t.

(2) In time slot t, the action-space $A^t = \{a_1^t, \ldots, a_j^t, \ldots, a_k^t\}$, each action corresponds to an edge server deployment and the allocation relationship between base stations and server in slot t.

(3) A $k \times m$ Q-table matrix is established, all Q-values are initialized to 0. The jth row and the ith column of the matrix represent the Q-value of the edge server s_j deployed at the base station b_i.

(4) Under the current state $state^t$, find the action with the largest Q-value in the action-space and update the state to $state^{t+1}$.

(5) Calculate the immediate reward $reward^t$ according to the formula (9).
(6) Calculate the Q-value according to formula (10) and update it.
(7) Iterate multiple times until the number of iterations is N

The pseudocode of Algorithm 2 is as follows:

Algorithm2:
$Q-Learning\ Edge\ Server\ Deployment(Q-ESP)$

Input: $G,\ D,\ W,\ \mathcal{B},\ k,\ m,\ \varepsilon,\ \alpha,\ \gamma,\ N$
Output: $Q\text{-}table,\ X,\ U$

For episode $i = 1\ to\ N$ **do**
  randomly select a state $state^1$
  For slot $t = 1\ to\ k$ **do**
    generate p at random
    If $p \leq \varepsilon$
      randomly select an action a^t
    Else
      select action $argmax_{a^t}Q(state^t, a^t)$
    End if
    execute action a_t, obtain reward $reward^t(state^t, a^t)$ according to equation (9), observe the next state $state^{t+1}$
    calculate Q-value and update in Q-table according to equation (10)
  End for
End for

V. SIMULATION RESULTS

In reality, the edge network topology is difficult to obtain. With the increase of access terminals and APs, the edge network is constantly expanding and growing, and new nodes tend to be connected to more connected nodes when they join. For example, the topology of the Internet follows a power-law distribution [16]. Referring to this feature, we use a scale-free network model to simulate the topology of the edge network. In simulation experiments, we utilize the Barabasi-Albert model [17] to generate random scale-free networks as edge networks.

The experimental scene in this paper simulates a square area of 2000m×2000m.We set up 50 base stations deployed in the area to receive user requests. The communication range of each base station is 200m, and the topology structure between base stations is generated by the Barabasi-Albert model. The task data that each base station needs to calculate is 200~500MB, and the number of CPU cycles required to process 1 bit is 1000 cycles. We will select k $(1 \leq k \leq 10)$ base station locations to co-deploy edge servers to process requests. The service range of each edge server is 500m. In each experiment, the average arrival rate λ_j of the edge server is determined by the normal distribution $\lambda_j \sim N(0.2, 0.5)$, the average is 0.2, the variance is 0.5 and $0 \leqslant \lambda_j \leqslant 0.4$, the average service rate of the edge server is 0.5, and the processing capacity of the edge server is f =10GHz/s. In addition, the learning rate of Q-learning is $\alpha = 0.8$, the empirical reward discount factor is $\gamma = 0.1$, the number of iterations is set to $N = 1000$, and the ε in the $\varepsilon - greedy$ algorithm is set to $\varepsilon = 0.01$.

To construct the network delay matrix $D \in \mathbb{R}^{m \times m}$, we set a weighted value for each link in the edge network topology graph G, which represents the delay between the two base stations connected by this link. The weight of each edge is randomly generated according to the normal distribution, with an average value of 0.2 and a variance of 0.1, the restricted value range is between 0.1 and 0.4. We use Dijkstra algorithm to calculate the delay $D_{i,j}$ between base stations b_i and b_j in graph G.

To evaluate the specific performance of the Q-ESD algorithm proposed in the paper, this section builds a simulation experiment environment on MATLAB R2017b. Taking the waiting time of the system in the edge network as an indicator, the comparison algorithms include the deployment algorithm based on the maximum load, the deployment algorithm based on K-means clustering, the random deployment algorithm. The detailed description of the comparison algorithm is as following:

(1) Top-K. The algorithm is to place K edge servers at the top-K base stations in front of the workload.
(2) K-means. This approach is a commonly used classical clustering algorithm, and its purpose is to automatically divide a dataset into K clusters. We use the K-means algorithm to separate all base stations into K clusters, and then deploy k edge servers at the centroids of these K clusters to reduce the system delay of the edge network.

(3) Random. The algorithm randomly selects K base stations to deploy edge servers.

A. Impact of the Number of Servers

In this experiment, we mainly study the impact of the numbers of edge servers *k* on the performance of the edge server in terms of the access delay. We performed experiments using 50 base stations by varying *k* from 1 to 10. The experimental results of the four deployment algorithms on the total system delay with the increase of edge servers are shown in Fig2.

Fig2 shows that with the increase of the number of edge servers, the total delay of the edge network will gradually decrease and become flat. Because, as the number of edge server increases, each base station has the opportunity to forward user requests to closer edge servers, the total latency of the base station tends to decrease. Regardless of the number of edge servers, the performance of the random algorithm is the worst, and the total latency of the Q-ESD deployment algorithm is less than that of the other three algorithms. In the Q-ESD algorithm proposed in this paper, each iteration is a complete deployment scheme, and the Q value of each server deployment is recorded. During 1000 iterations, the deployment plan is continuously adjusted according to the total delay of the edge network after deployment. Compared with K-means and Random, Q-ESP reduces the system delay by 15.9% and 19.1% on average.

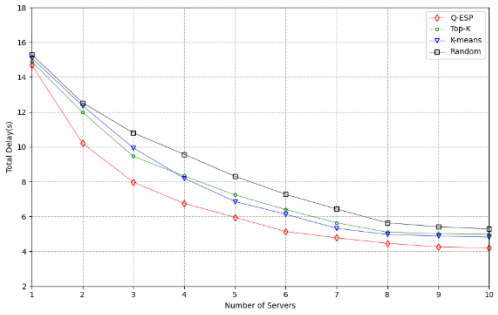

Fig2 Total Delay with Different Number of Servers

When the number of edge server does not exceed 3, the Top-K algorithm performs better than the K-means. Because the Top-K algorithm deploys the edge server at the *k* base stations with the highest load, it can serve as many user requests as possible, which can quickly alleviate the delay caused by insufficient computing resources. When the number of edge servers exceeds 3, the performance of the K-means algorithm is better than the Top-K. Since the K-means algorithm is deployed more evenly, the number of network hops between the base station and the target edge server is reduced. The Top-K algorithm needs to connect the remote base station to the server after multiple forwarding, so the transmission time is higher.

When the number of edge servers exceeds 8, the performance of all four algorithms tends to be flat. When there are more than 8 edge servers, in a square area of 2000m × 2000m, the computing of edge servers can meet the need of the current edge network.

At this time, requests from the base station can be quickly forwarded to the nearest edge server, fewer base stations are allocated to the edge server, and the server can process base station requests promptly, so the total system delay tends to be flat. If you continue to increase the number of edge servers, it will not greatly improve the QoS of users but will increase the cost of operators.

B. Impact of the Base Stations Load

In this experiment, we mainly consider the performance of different deployment algorithms when the base station load changes. We choose 8 edge servers, and use set W to record the base station load of this scene. We consider expanding the load of all base stations to 1.1, 1.2, 1.3, 1.4, and 1.5 times the load in the set W. The total system delay comparison of the four algorithms is presented in Fig3.

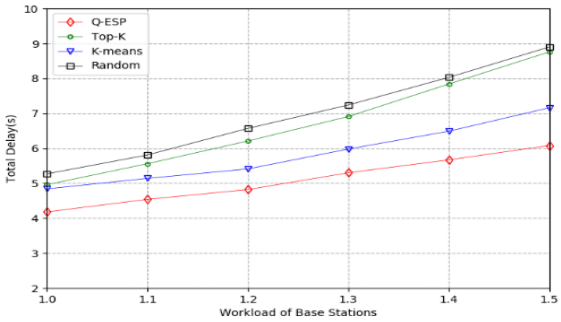

Fig3 Total Delay with Different Workload of Base Stations

As the load of the base station increases, the workload of the edge server also increases accordingly, and the total system delay of the four

algorithms shows an upward trend. The performance of the Q-ESD algorithm is the best. It can be seen that the performance of the K-means algorithm and the Q-ESD algorithm is significantly improved compared to the Random algorithm and the Top-K algorithm. The K-means algorithm ensures that the base station is closer to the edge server to some extent. The Q-ESD algorithm improves its deployment performance through multiple iterations and interaction with the environment. It considers both the calculation delay and the transmission delay between base stations, so the Q-ESD algorithm has the best performance.

At the same time, we found that with the increase of base station load, the total delay of the system increases faster and faster. When the load just begins to increase, the edge server can still roughly satisfy the computing request of the base station. As the load of the base station continues to grow, a awaiting queue will gradually be generated at the edge server, and the length of the waiting queue will increase rapidly as the load increases, so the total system delay will increase faster and faster.

C. Performance of Q-ESP Algorithm

This experiment mainly uses the reward value in the framework of reinforcement learning to evaluate the performance of our proposed Q-ESD algorithm. We choose 8 edge servers in the experiment. According to Fig2, 8 edge servers can reduce the total delay of the entire edge network and will not waste the computing power of edge servers. The performance of the Q-SEP algorithm is shown in Fig4.

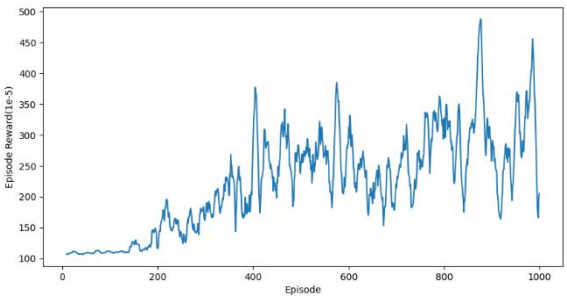

Fig4 Episode Reward with Number of Iterations

With a learning rate $\alpha = 0.8$ and an empirical reward discount factor $\gamma = 0.1$, as the number of iterations increases, the reward obtained by the Q-ESD algorithm in each iteration cycle oscillates around 250×10^{-5}. In the first 160 iterations, the system rewards did not increase significantly, indicating that the algorithm is still in an exploratory state. From the 160th to the 420th iteration period, the system returns showed a clear upward trend. After 420 iterations, the system return value fluctuated around 250×10^{-5}, and the Q-ESP algorithm gradually stabilized.

VI. CONLUSION

Edge computing is an important emerging technology that can be used to extend the computation and storage capabilities by offloading processing workload from the cloud in order to reduce mobile edge computing network latency on mobile devices. In this study, we propose a three-layer network model consisting of data center, edge servers, and base stations, and the edge servers have computing capabilities and can provide computing services for base stations. Therefore, an edge server deployment optimization problem is considered. To get a solution that minimizes the total delay of the system, we use a reinforcement learning algorithm to solve it. Firstly, we define the key elements of the algorithm according to the framework of reinforcement learning: state, action, and reward. Then, the designed solution algorithm is composed of two sub-algorithms. When the location of each edge server is selected, Q-ESP algorithm is utilized to solve it. After determining the location of an edge server, we determine the base station assigned to the edge server according to the delay matrix. Finally, we performed experiments to evaluate the performance of the proposed approach, and we compare the results with classical algorithm. The experimental results demonstrate that our proposed algorithm outperforms several representative approaches in terms of the access delay in edge network.

REFERENCES

[1] Taleb, T., Dutta, S., Ksentini, A., Iqbal, M., & Flinck, H. (2017). Mobile edge computing potential in making cities smarter. *IEEE Communications Magazine*, *55*(3), 38-43.

[2] Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020, July). A theoretical analysis of deep Q-learning. In *Learning for Dynamics and Control* (pp. 486-489). PMLR.

[3] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., & Young, V. (2015). Mobile edge computing—A key technology towards 5G. *ETSI white paper*, *11*(11), 1-16.

[4] Abbas, N., Zhang, Y., Taherkordi, A., & Skeie, T. (2017). Mobile edge computing: A survey. *IEEE Internet of Things Journal*, *5*(1), 450-465.

[5] Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE internet of things journal*, *3*(5), 637-646.

[6] Ha, K., Chen, Z., Hu, W., Richter, W., Pillai, P., & Satyanarayanan, M. (2014, June). Towards wearable cognitive assistance. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services* (pp. 68-81).

[7] Li, D., Salonidis, T., Desai, N. V., & Chuah, M. C. (2016, October). Deepcham: Collaborative edge-mediated adaptive deep learning for mobile object recognition. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)* (pp. 64-76). IEEE.

[8] Tang, B., Chen, Z., Hefferman, G., Wei, T., He, H., & Yang, Q. (2015). A hierarchical distributed fog computing architecture for big data analysis in smart cities. In *Proceedings of the ASE BigData & SocialInformatics 2015* (pp. 1-6).

[9] Jia, M., Cao, J., & Liang, W. (2015). Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks. *IEEE Transactions on Cloud Computing*, *5*(4), 725-737.

[10] Xu, Z., Liang, W., Xu, W., Jia, M., & Guo, S. (2015). Efficient algorithms for capacitated cloudlet placements. IEEE Transactions on Parallel and Distributed Systems, 27(10), 2866-2880.

[11] Ren, Y., Zeng, F., Li, W., & Meng, L. (2018, July). A low-cost edge server placement strategy in wireless metropolitan area networks. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)* (pp. 1-6). IEEE.

[12] Li, Y., & Wang, S. (2018, July). An energy-aware edge server placement algorithm in mobile edge computing. In *2018 IEEE International Conference on Edge Computing (EDGE)* (pp. 66-73). IEEE.

[13] Wang, S., Zhao, Y., Xu, J., Yuan, J., & Hsu, C. H. (2019). Edge server placement in mobile edge computing. *Journal of Parallel and Distributed Computing*, *127*, 160-168.

[14] Shah, S. M., & Borkar, V. S. (2018). Q-learning for Markov decision processes with a satisfiability criterion. *Systems & Control Letters*, *113*, 45-51.

[15] Clifton, J., & Laber, E. (2020). Q-learning: theory and applications. Annual Review of Statistics and Its Application, 7, 279-301.

[16] Albert, R., Jeong, H., & Barabási, A. L. (1999). Diameter of the world-wide web. *nature*, *401*(6749), 130-131.

[17] Barabási, A. L., & Albert, R. (1999). Emergence of scaling in random networks. *science*, *286*(5439), 509-512.