Deep Reinforcement Learning Based Resource Allocation Strategy in Cloud-Edge Computing System

Zhuohan Xu[1], Zeheng Zhong[1], and Bing Shi[1,2,*]

[1]School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan 430070, China
[2]Shenzhen Research Institute of Wuhan University of Technology, Shenzhen 518000, China
[*]Corresponding author. Email: bingshi@whut.edu.cn

2022 International Joint Conference on Neural Networks (IJCNN) | 978-1-7281-8671-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/IJCNN55064.2022.9892029

***Abstract*—The rapid development of mobile devices applications has put tremendous pressure on edge nodes with limited computing capabilities, which may cause poor user experience. To solve this problem, collaborative cloud-edge computing is proposed. In the cloud-edge computing, an edge node with limited local resources can rent more resources from a cloud node. Since cloud service providers offer a variety of pricing modes for users' different computing demands, edge node needs to select appropriate pricing mode of cloud service and allocates resources between the cloud node and the edge node. It is a sequential decision problem. In this paper, we model it as a Parameterized Action Markov Decision Process, and propose a resource allocation algorithm Cost Efficient Resource Allocation under collaborative Cloud-Edge (CERACE) based on the deep reinforcement learning algorithm Parametrized Deep Q-learning (P-DQN). We evaluate CERACE against three typical resource allocation algorithms Edge First + On Demand (E+O), Edge + Random (E+R) and Random + Random (R+R) based on synthetic data and real data of Google dataset. The experimental results show that CERACE can effectively reduce the long-term operation cost of collaborative cloud-side computing in various demanding settings. Our analysis can provide some useful insights for enterprises to design the resource allocation strategy in the collaborative cloud-side computing system.**

***Index Terms*—Collaborative Cloud-Edge Computing, Resource allocation, Reinforcement learning**

I. Introduction

In recent years, the number of mobile devices such as mobile phones, wearable devices and sensors has increased rapidly. Due to the limitations of computing power, memory and battery capacity of mobile devices, it is usually unable to meet the requirements of the complex computing demands. In order to provide low latency and real-time services to mobile users, edge computing is proposed. Edge computing can provide users with low latency, location awareness and high-quality service [1]–[3]. However, edge nodes usually do not have enough storage and computing resources when processing massive mobile device data. Therefore, collaborative cloud-edge computing has been proposed. Cloud nodes and edge nodes cooperate with each other to provide users with computing services [4].

In the collaborative cloud-edge computing, the edge node with limited local resources can rent more resources from the cloud node. When the users' computing demands randomly reach the edge node, the edge node needs to decide how to reasonably allocate resources between the cloud and edge nodes to satisfy users' demands. In addition, cloud service providers may offer various pricing modes for cloud service. Edge node needs to select appropriate pricing mode of cloud service for collaborative computing according to users' demands. In this paper, we will analyze how to allocate resources efficiently to reduce the long-term operation cost in the cloud-edge computing system to satisfy the dynamic demands of users.

Specifically, the existing research on resource allocation in collaborative cloud-side usually assumes that the users' demands are known, and generally does not consider that cloud service providers offer different types of pricing modes. Therefore, in this paper, we consider an edge node with limited computing capacity and a cloud node that provides multiple different pricing modes for cloud service. The edge side needs to select appropriate pricing mode of cloud service and allocates resources between the cloud node and the edge node. Since it is a sequential decision-making problem, we propose a resource allocation algorithm based on hybrid action deep reinforcement learning algorithm Parametrized Deep Q-learning (P-DQN), named Cost Efficient Resource Allocation under collaborative Cloud-Edge (CERACE). We further run experiments to evaluate our algorithm against three typical resource allocation algorithms Edge First + On Demand (E+O), Edge + Random (E+R) and Random + Random (R+R) based on synthetic data and real data of Google dataset. The experimental results show that the algorithm proposed in this paper can achieve the lowest operation cost under different strength of demanding amount and computing time duration.

The structure of this paper is as follows. In Section 2, we introduce the related work. In Section 3 we describe the basic settings. In Section 4, we model the problem as a Parameterized Action Markov Decision Process and introduce the resource allocation algorithm CERACE based on P-DQN.

978-1-7281-8671-9/22/$31.00 ©2022 IEEE

We run experiments to evaluate the proposed algorithm in Section 5 and conclude the paper in Section 6.

II. Related Work

There exist a lot of works about resource allocation in the cloud computing, such as [5]–[8]. In [9], Zhan et al. did a deep survey about resource allocation in the cloud computing. Furthermore, there also exist some works about resource allocation in the edge computing. Zhang et al. [10] used a Stackelberg game based approach to solve the multi-user offloading problem when the edge computing resource is limited in order to reduce the energy consumption. Guo et al. [11] proposed an optimal policy based on a Markov decision process for scenarios with high mobile device density. Zhao et al. [12] proposed an offloading strategy that maximizes the number of tasks served while satisfying the users' latency requirements. Gross et al. [13] proposed a dynamic cost model to minimize the total time consumed by IoT devices in the mobile edge computing environment.

Because of the development of collaborative cloud-edge system, there also exist works about resource allocation in the cloud-edge system. Wang et al. [14] proposed an optimization strategy for computing resource allocation of massive IoHT devices in cloud-edge computing environment. Yuan et al. [15] designed a collaborative computation offloading and resource allocation algorithm to maximize the profit of systems and meet the response time constraint. Lin et al. [16] proposed a lightweight system called CloudFog to improve the quality of service for the corresponding delay problem in the cloud-based entertainment game. Shen et al. [17] proposed a dynamic task unloading method DOM with minority game in cloud-edge computing to address the vehicle computing resource shortage in the Internet of vehicles. Zhao et al. [18] proposed a collaborative approach in the cloud-edge computing to offload tasks to automobiles in vehicular networks. Wang et al. [19] proposed an online algorithm to dynamically allocate resources in the collaborative cloud-edge system. Jiao et al. [20] designed an online algorithm which decouples the original off-line problem by constructing a series of regularized subproblems to reduce the cost. Dinh et al. [21] considered a hybrid cloud-edge computing system where edge devices with limited local resources can rent more resources from cloud nodes. Wang et al. [22] proposed a dynamic multi-winner incomplete information game to offload tasks and allocate resource for multiple end users.

To the best of our knowledge, existing works about resource allocation in the collaborative cloud-edge system usually did not consider the cost of cloud-side resource, and only consider a simple pricing mode. Furthermore, users' demands are usually dynamically arriving in the real world, and existing works usually consider how to allocate resources when users' demands are known. In this paper, we analyze the resource allocation problem in the collaborative cloud-edge computing given users' dynamic demands and various pricing modes of cloud service.

TABLE I
Key symbols

Notation	Description
T	Total time slots
D_t	Demand information submitted by the user in time slot t
d_t	Number of VMs requested of D_t
l_t	Computing time duration of D_t
E	The total number of VMs of the edge node
e_t	The number of remaining VMs of edge node in time slot t
d_t^e	Number of VMs provided by edge node
d_t^c	Number of VMs provided by cloud node
h_t	Resource allocation record for D_t
η_t	Number of VMs released by edge nodes in time slot t
p_e	Standby cost of one VM in the edge node
p_f	Computing cost of one VM in the edge node
$p_{upfront}$	Customization price of reserved instance in cloud service
p_{od}	Unit cost of on-demand instance in cloud service
p_{re}	Unit cost of reserved instance in cloud service
ξ_t	Remaining usage time of reserved instance
p_t	Unit cost of spot instance in cloud service in time slot t

III. Basic Settings

In this section, we first introduce how the collaborative cloud-edge system works, and then introduce the basic settings. Finally, we describe our problems. We list the key symbols used in this paper in Table I.

A. The Collaborative Cloud-Edge Environment

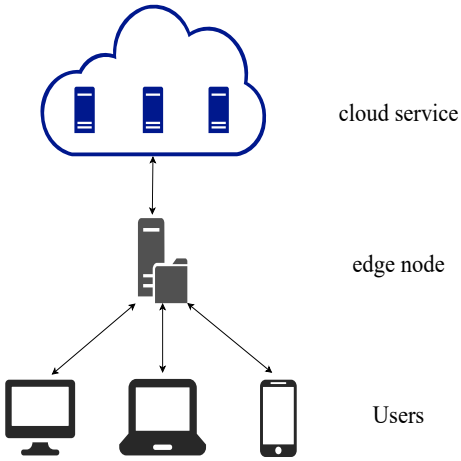


Fig. 1. Collaborative cloud-edge computing system

We consider the resource allocation problem in the multi-level collaborative computing environment [21], [23] of "user-edge-cloud", as shown in Fig. 1. The environment contains a single cloud node and an edge node, which can also be extended the setting with multiple edge nodes.

The edge node has its own VMs to process users' demands. However, because the number of VMs requested by the user may exceed the edge node's capacity, the edge node needs to rents VMs from the cloud node to scale up its capacity. Note that cloud service providers such as Amazon, Microsoft and

Alicloud have provided three different pricing modes[1], each of which has a different cost structure. Edge node needs to select appropriate pricing mode of cloud service and allocates users' demands to either rented VMs or its own VMs.

B. User Setting

As described in Section III-A, the time is discretized into T time slots. We assume that in each time slot t, the demand submitted by the user can be defined as:

$$D_t = (d_t, l_t) \tag{1}$$

where d_t is the number of VMs requested of D_t, l_t is the computing time duration of D_t.

C. Computing Resources and Cost of Edge Nodes

The total computing resources owned by the edge node are represented by E. As the resource is allocated to users, we use e_t to represent the number of remaining VMs of edge node in time slot t. The number of VMs provided by the edge node is expressed as d_t^e.The number of VMs provided by the cloud node is expressed as d_t^c. It should be noted that if the edge node has no available resources, it will hand over all the arriving computing tasks to the cloud service for processing. Now we have:

$$d_t^e = \begin{cases} d_t - d_t^c, & e_t \geq 0 \\ 0, & e_t = 0 \end{cases} \tag{2}$$

When the resource allocation can be successfully performed on the edge node, each demand processed by the edge node will generate an allocation record:

$$h_t = (d_t^e, l_t) \tag{3}$$

which consists of two parts, d_t^e is the number of VMs provided by the edge node in this allocation, l_t is the remaining computing time of this demand. When a new demand arrives and resource allocation is completed, an allocation record will be generated and added to an allocation record list:

$$H = < h_1, h_2, \cdots, h_m > \tag{4}$$

At the end of each time slot, the edge node traverses these m records in the allocation record list H and then subtract 1 from the l_i in its record h_i. If the remaining computing time of an allocation record is 0, it means that the demand has been completed. The edge node needs to release the corresponding VMs according to its record and delete the allocation record from the list. The number of VMs that have completed the computing task and are waiting to be released at the end of time slot t is defined as η_t. Then we have:

$$\begin{aligned} \eta_t &= \sum_{i=1}^{m} d_i^e \\ s.t.\ l_i &= 0, h_i \in H \end{aligned} \tag{5}$$

Furthermore, the number of remaining VMs of the edge node at the next time slot $t+1$ is the number of remaining VMs at the beginning of the time slot t minus the quantity allocated in the end of time slot t, plus the quantity released because of the completion of the computing task in the time slot t. Then the number of remaining VMs of the edge node at the time slot $t+1$ is:

$$e_{t+1} = e_t - d_t^e + \eta_t \tag{6}$$

Note that in order to quickly respond to users' computing demands, even if there is no computing demand, the machine still has standby cost. Therefore, we consider that the cost of edge nodes consists of standby cost and computing cost. The standby cost of one VM in the edge node is p_e. The computing cost of one VM in the edge node is p_f. Now the cost of the edge node in the time slot t is:

$$C_t^e = e_t p_e + (E - e_t) p_f \tag{7}$$

$e_t p_e$ is the standby cost of the edge node in the time slot t, and $(E - e_t)p_f$ is the computing cost of the edge node.

D. Cost of Collaborative Cloud-Side Computing

In time slot t, the cost of collaborative cloud-edge includes the computing cost of cloud nodes and the cost of edge node, which is:

$$\begin{aligned} C_t = X_1 p_{od} d_t^c + X_2 p_{upfront} + X_3 p_{re} d_t^c + X_4 p_t d_t^c + C_t^e \\ X_i = \begin{cases} 1, The\ service\ is\ used \\ 0, The\ service\ is\ not\ used \end{cases} \end{aligned} \tag{8}$$

$X_1 p_{od} d_t^c$ is the cost of on-demand instance and p_{od} is the unit cost of on-demand instance. $X_2 p_{upfront} + X_3 p_{re} d_t^c$ is the cost of reserved instance, where $p_{upfront}$ is the customization price of reserved instance and p_{re} is the unit cost of reserved instance. $X_4 p_t d_t^c$ is the cost of spot instance, where p_t is the unit cost of spot instance, which is dynamically set by the cloud service provider. C_t^e is the cost of the edge node.

E. Problem Formulation

We divide the whole time into T time slots. At the beginning of each time slot t, the user submits its demand to the edge node. Once receiving it, the edge node determines the pricing mode of cloud service to be used and allocates demands to either cloud VMs or its own VMs according to its resource allocation strategy. According to the allocation and the price of the corresponding cloud service set by the cloud service provider, the cost C_t of the current time slot t can be calculated, and then the system enters the next time slot. Since the system will run for multiple time slots, we intend to minimize the long-term cost of the system $\sum_{t=1}^{T} C_t$.

[1]On-demand instance: This pricing mode allows users to pay with the fixed time granularity set by the platform, and the price is fixed in a long period of time.

Reserved instance: This pricing mode requires the user to submit the reservation time in advance and pay the corresponding reservation fee to have the instance within the contract time. It is applicable to users with a large number of computing demands, and the unit price is usually about 50% lower than that of on-demand instances

Spot instance: The instance of this mode is obtained by bidding, and its price changes in different time slots, which is usually used for short-term computing demands.

IV. Resource Allocation Algorithm

In this section, we first describe the Parameterized Action Markov Decision Process, and then introduce the resource allocation strategy in collaborative cloud-edge environment based on P-DQN.

A. Parameterized Action Markov Decision Process

Given users' dynamical demands over the time, the resource allocation problem is a sequential decision-making problem, which can be modeled as a Markov decision process. In each time slot t, the edge node first needs to select the pricing mode of cloud service to be used, and then determine the resource segmentation between the edge node and the cloud node. The resource allocation action can be described by parametric action. In order to describe this parameterized action sequential decision, Parameterized Action Markov Decision Process (PAMDP) [24] is used.

Similar to Markov decision process, parameterized action Markov decision process is a tuple (S, A, P, r, γ) where S is the finite set of states, A the finite set of parameterized actions, P is the probability of state transition, r and γ is the immediate reward and discount factor. We now introduce them in the follow in details.

- $s_t = (e_t, \eta_{t-1}, D_t, p_t, \xi_t) \in S$ is used to describe the state of the edge node at the beginning of each time slot, where e_t is the number of remaining VMs of the edge node in t, η_{t-1} is the number of VMs returned in the previous time slot, D_t is the user's demand information in t, p_t is the unit cost of spot instance in t, ξ_t is the remaining usage time of reserved instance. When the edge node does not use this type of cloud service or it expires, this value is 0.
- $a_t = (x_e, (k, x_k)) \in A$, where x_e is the ratio of the number of VMs provided by the edge node to the total number of VMs. $k \in K$, where $K = \{k_1, k_2, k_3\}$ is the set of all discrete actions, k_1 is the on-demand instance, k_2 is the reserved instance, k_3 is the spot instance. x_k is the ratio of the number of VMs provided by the cloud node to the total number of VMs.
- $r_t = -C_t$ is the immediate reward. Note that we want to reduce the long-term operation cost, and therefore the immediate reward of each time slot is set as a negative value of the cost.

B. Resource Allocation based on P-DQN

Reinforcement learning [25] has been widely used to solve the sequential decision-making problems. In the resource allocation of the collaborative cloud-edge system, the edge node needs to select the pricing mode of cloud service to be used, and then determine the number of VMs to be rented from the cloud node. This is a mixture of discrete action space and continuous action space. Therefore, we use P-DQN [26] to solve the resource allocation.

The basic idea of P-DQN is as follows. For each action $a \in A$ in the parametric action space, due to $x_e + x_k = 1$, we can only consider k and x_k in the action value function, that is $Q(s, a) = Q(s, k, x_k)$, where $s \in S$,$k \in K$ is the discrete action selected in the time slot t, $x_k \in X_k$ is the parameter value corresponding to k. Similar to DQN, deep neural network $Q(s, k, x_k; \omega)$ is used in P-DQN to estimate $Q(s, k, x_k)$, where ω is the neural network parameter. In addition, for $Q(s, k, x_k; \omega)$, P-DQN uses the determined policy network $x_k(\cdot; \theta) : S \to \mathcal{X}_k$ to estimate the parameter value $x_k^Q(s)$, where θ is used to represent the policy network. That means the goal of P-DQN is to find the corresponding parameters θ when ω is fixed. It can be written as:

$$Q(s, k, x_k(s; \theta); \omega) \approx Q(s, k, x_k; \omega) \tag{9}$$

Similar to DQN, the value of ω can be obtained by minimizing the mean square error by gradient descent. Specifically, in step t, ω_t and θ_t is the parameters of value network and deterministic policy network respectively. Then y_t can be written as:

$$y_t = r + \max_{k \in [k]} Q(s', k, x_k(s', \theta_t); \omega_t) \tag{10}$$

where s' is the next state after taking the mixed action $a = (k, x_k)$. The loss function of value network can be written as:

$$l^Q(\omega) = \frac{1}{2}[Q(s, k, x_k; \omega) - y]^2 \tag{11}$$

Similarly, the loss function of a policy network can be written as:

$$l^{\Theta}(\theta) = -\sum_{k=1}^{K} Q(s, k, x_k(s; \theta); \omega) \tag{12}$$

We now propose the resource allocation algorithm based on P-DQN, which is called Cost Efficient Resource Allocation under collaborative Cloud-Edge (CERACE), as shown in Algorithm 1.

V. Experiment

In this section, we run experiments to analyze our resource allocation algorithm. We first introduce the parameter settings and dataset, and then discuss the experiment results.

A. Parameter Settings

The parameter settings used in this experiment are shown in Table II. In this experiment, the initial number of VMs of edge nodes is set $E = \{60, 70, 80, 90, 100\}$. The service time of collaborative cloud-edge computing is 100 time slots, i.e. $T = 100$. The number of VMs and the computing time duration requested by users can be described by normal distribution [27], [28], i.e.$d_t \sim N(\mu_a, \sigma_a^2)$, $l_t \sim N(\mu_l, \sigma_l^2)$. The standby cost of one VM in the edge node is $p_e = 0.03$. The computing cost of one VM in the edge node is $p_f = 0.2$. The unit cost of on-demand instance is $p_{od} = 3.0$. The customization price of reserved instance is $p_{upfront} = 800$. After the reserved instance is started, it can be used at a unit cost $p_{re} = 1.5$ within $T_r = 20$. The unit cost of spot instance is set by the cloud service provider. In this experiment, it is assumed that its price fluctuation follows a normal distribution $p_t \sim N(1.5, 1)$. Because the spot instance is mainly aimed at the needs of

Algorithm 1: Cost Efficient Resource Allocation under collaborative Cloud-Edge (CERACE)

Initialize exploration parameters ϵ, soft update coefficient τ_1 and τ_2, number of samples for batch gradient descent m, maximum number of iterations M, random noise function $\mathcal{N}$ and experience replay pool P;
for $i = 1$ *to* M **do**
  Receive user task information and obtain the status s of collaborative cloud-edge computing environment;
  Calculate the parameter value of each instance type in the cloud service:$x_k \leftarrow x_k\left(s_t, \theta_t\right) + \mathcal{N}$;
  Selects discrete actions according to ϵ-greedy strategy:

$$a = \begin{cases} \text{random discrete action}, rnd < \epsilon \\ \left(k, x_k\right), k = \arg\max_{k \in [K]} Q\left(s, k, x_k; \omega\right), rnd \geq \epsilon \end{cases};$$

  The edge node performs action a and obtains the next status s', reward r and termination flag $isend$;
  The edge node generates an allocation record h_i according to the allocation operation. Add it to the allocation record list H;
  Add the state transition tuple $(s, a, r, s', isend)$ in the experience replay pool D;
  Sample m samples from experience replay pool P, calculate the target Q value y according to the (10);
  Update status: $s = s'$;
  Calculate gradient $\nabla_\omega l^Q(\omega)$ and $\nabla_\theta l^\Theta(\theta)$ according to (11) and (12);
  Update network parameters:$\omega' \leftarrow \omega - \tau_1 \nabla_\omega l^Q(\omega)$, $\theta' \leftarrow \theta - \tau_2 \nabla_\theta l^\Theta(\theta)$;
  Update allocation record H and release computing resources for completed tasks;
  If s' is terminated, complete the current round of iteration, otherwise go to step 3;
end

small-scale and short-time computing tasks, we assume that only tasks with duration $T_m = 6$ or less can choose this type of instance.

B. Experimental Dataset

In order to evaluate the performance of the algorithm under different initial states and different user demanding intensities. We run the experiments on the synthetic data and the realistic data on Google dataset respectively. Firstly, we introduce the synthetic data. Similar to [21], [29], we investigate the impact of different demanding intensities on the algorithm. The size of the demanding intensity is described by the mean and variance of the demanding instances in the normal distribution. The greater the mean and variance, the greater the demanding

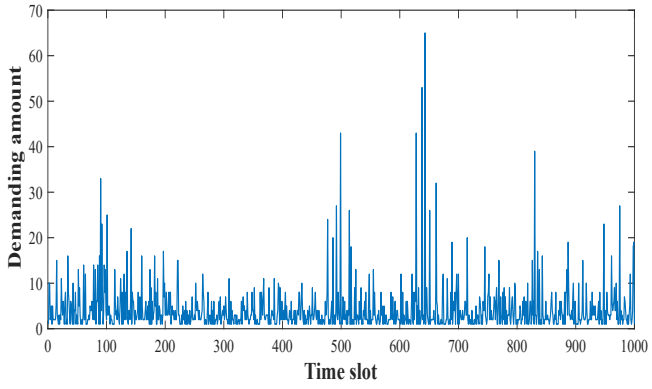


Fig. 2. Fluctuation of demanding amount

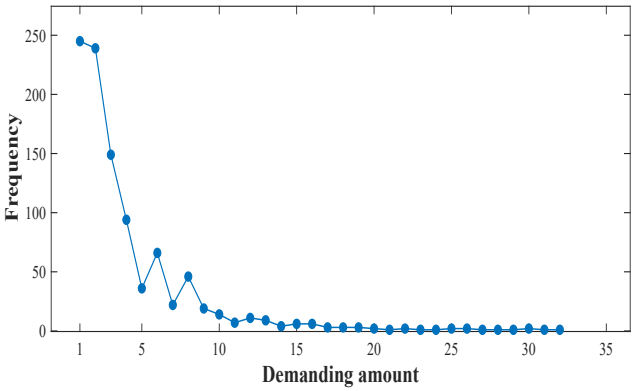


Fig. 3. Frequency of demanding amount

intensity. Specifically, since the demand D_t consists of the number of VMs requested d_t and the computing time duration l_t (see Equation (1)), we consider the demanding intensity from the demanding amount and computing time duration respectively. In more detail, in terms of the demanding amount d_t, we consider three different groups of intensities:

- Group 1 is a low intensity group with normal distribution $d_t \sim N(5, 5^2)$
- Group 2 is a medium intensity group with normal distribution $d_t \sim N(10, 10^2)$
- Group 3 is a medium intensity group with normal distribution $d_t \sim N(15, 15^2)$

where for each group, we assume that the computing time duration is at a medium level, $l_t \sim N(10, 10^2)$. In terms of the computing time duration l_t, we also consider three different groups of intensities:

- Group 1 is a low intensity group with normal distribution $l_t \sim N(5, 5^2)$
- Group 2 is a medium intensity group with normal distribution $l_t \sim N(10, 10^2)$
- Group 3 is a medium intensity group with normal distribution $l_t \sim N(15, 15^2)$

TABLE II
PARAMETER SETTINGS

Notation	Description
$T = 100$	Total time slots
$E = \{60, 70, 80, 90, 100\}$	The total number of VMs of the edge node
$d_t \sim N(\mu_a, \sigma_a^2)$	The normal distribution of the number of VMs requested by the user
$l_t \sim N(\mu_l, \sigma_l^2)$	The normal distribution of the computing duration requested by the user
$p_e = 0.03$	Standby cost of one VM in the edge node
$p_f = 0.2$	Computing cost of one VM in the edge node
$p_{od} = 3.0$	Unit price of on-demand instance in cloud service
$p_{upfront} = 800$	Customization price of reserved instances in cloud service
$p_{re} = 1.5$	Unit price of reserved instance in cloud service
$T_r = 20$	Reserved duration of reserved instance in cloud service
$p_t \sim N(1.5, 1)$	Normal distribution of unit price of spot instance
$T_m = 6$	Maximum service duration of spot instance
$\gamma = 0.99$	Discount factor
$\tau = 0.001$	Soft update parameters of target network
$\alpha_1 = 0.0001$	Learning rate of actor
$\alpha_2 = 0.00001$	Learning rate of critic
$P = 100000$	Size of experience pool
$m = 128$	Size of the batch sample in the experience pool

where for each group, we assume that the demanding amount is at a medium level, $d_t \sim N(10, 10^2)$.

We also use Google cluster-usage traces data to further evaluate the proposed algorithm. Because there is no demanding information about computing time duration in this data, we assume that the computing time duration satisfies the normal distribution $l_t \sim N(10, 10^2)$. Fig. 2 and Fig. 3 show the fluctuation and frequency of user demands in Google dataset. As can be seen from Fig. 2, in the first 500 time slots, the demanding amount fluctuates less, while in the last 500 time slots, it fluctuates more. From Fig. 3, the demanding amount is mainly between 1 and 10, but some requests can reach 60 or more. According to this analysis of Google dataset, we set the initial capacity of the edge node in this experiment as $E = \{40, 50, 60, 70, 80\}$.

C. Benchmark Algorithms

We now introduce the benchmark algorithms used in the experiment. Note that the existing similar works usually do not consider the various pricing modes of cloud service [29], [30]. Therefore, we cannot evaluate our algorithm against these works. Instead, we evaluate our algorithm against the following three algorithms in terms of the long-term operation cost:

- E+O (Edge First + On Demand): this algorithm gives priority to the edge node to process user's requests. When the edge node has insufficient capacity to provide services, only the on-demand instance is used to process user's requests.
- E+R (Edge + Random): this algorithm gives priority to the edge node to process user requests. When the capacity of the edge node is insufficient to provide computing services, the pricing mode of cloud service is randomly selected for collaborative computing.
- R+R (Random + Random): this algorithm randomly selects the pricing mode of cloud service and randomly determine the quantities for allocation.

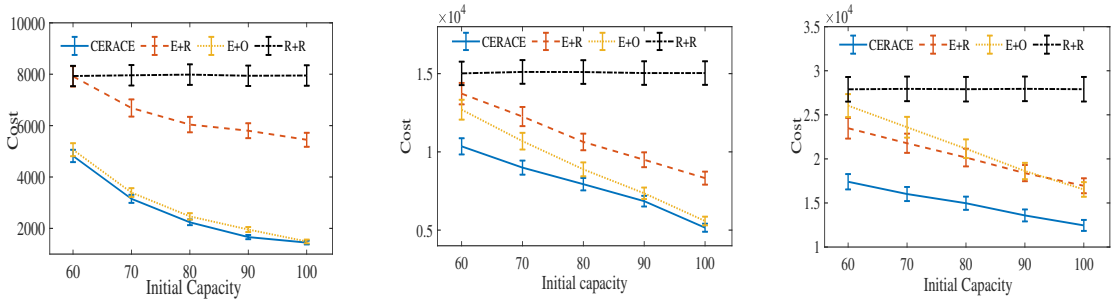
D. Experimental Analysis

1) Impact of demanding amount on the cost: the experimental results of the impact of demanding amount on the cost are shown in Figure.4. It can be seen that CERACE algorithm performs better than the other three algorithms under different request intensities.

In the low intensity group (Fig. 4(a)), R+R has little changes on the cost with respect to the different initial capacity of edge nodes. Its performance is the worst. The performance of E+R is the same as R+R when the initial capacity of edge nodes is 60. With the increase of the initial capacity of edge nodes, its cost decreases. CERACE and E+O have almost the same performance of the cost. This is because when users' request intensity is low and the computing resources of edge nodes are relatively sufficient, E+O can also have a good performance on the cost. At this moment, there is no need to use reserved instances in cloud services. Since CERACE can use the spot instance with cheaper price than the on-demand instance, its performance is a little better than E+O.

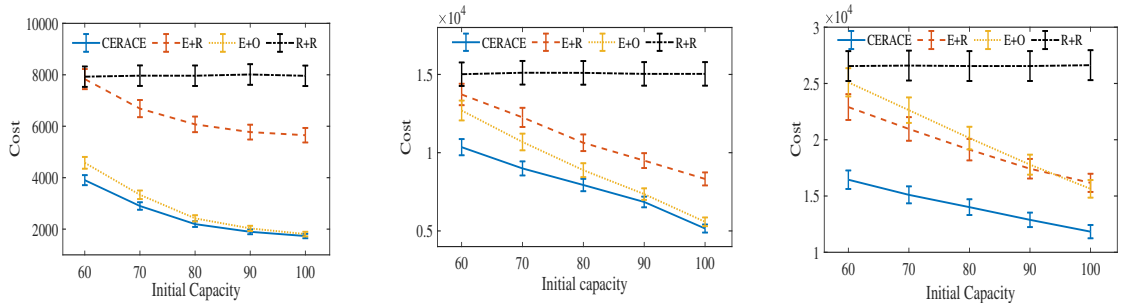
In the medium intensity group (Fig. 4(b)), our algorithm can still outperform other algorithms. We find that the cost of R+R under different initial capacity of edge nodes is still the highest. The cost of the other three algorithms decreases gradually with the increase of the capacity of edge nodes. When the initial capacity is low, CERACE has obvious advantages over the algorithm E+O.

In the high intensity group (Fig. 4(c)), compared with other algorithms, the cost of CERACE can be reduced by more than 25% under different initial capacity of edge nodes. Compared to Fig. 4(b), we find that E+R perform a little better than E+O. This is because given high demanding, the capacity of edge nodes is relatively scarce. E+O uses more on-demand instances, and therefore, its cost is higher than E + R which chooses instance type randomly.



(a) Low intensity group (b) Medium intensity group (c) High intensity group

Fig. 4. Experimental results of the impact of demanding amount on cost



(a) Low intensity group (b) Medium intensity group (c) High intensity group

Fig. 5. Experimental results of the impact of computing time duration on cost

2) Impact of computing time duration on the cost: the experimental results of the impact of computing time duration on the cost are shown in Fig. 5. It can be seen that the results are similar to Fig. 4. In more detail, Fig. 5(b) and Fig. 5(c) are almost the same as Fig. 4(b) and Fig. 4(c). There is a slight difference between Figure.4(a) and Fig. 5(a). This shows that when the initial capacity of the edge node is insufficient and the user's request intensity is low, CERACE is more sensitive to the change of the user's demanding amount. At this time, CERACE can more effectively reduce the cost. When the user's request intensity is high, CERACE has almost the same sensitivity to the user's demanding amount and computing time duration, and can effectively reduce the cost of the system.

3) Experiment based on Google dataset: in order to further evaluate the performance of the proposed resource allocation algorithm in the collaborative cloud-edge environment, we run the experiments on the Google dataset. The experimental results are shown in Fig. 6. The cost of the four algorithms under different initial capacity of edge nodes from high to low is R+R, E+R, E+O and CERACE. The cost difference between E+O and CERACE is small. This is because the demanding amount in Google dataset is mainly between 1 and 10 which is similar to the experiment of low intensity group.

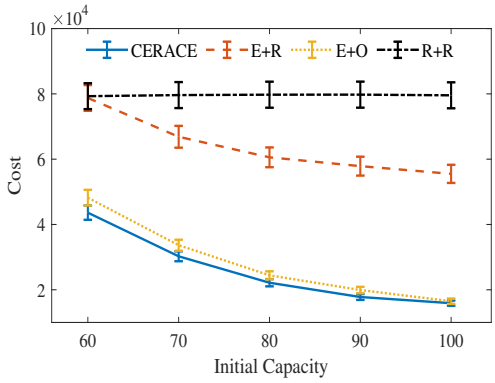


Fig. 6. Experimental results based on Google dataset

VI. Conclusion

In this paper, we analyze resource allocation problem in the collaborative cloud-edge computing given dynamic users' demands and various pricing modes of cloud service. We model the problem as a Parameterized Action Markov Decision Process, and then propose a resource allocation algorithm CERACE based on deep reinforcement learning algorithm P-DQN. Finally, we run experiments to evaluate our strategy against three typical algorithms on the synthetic data and the real data of Google dataset. The experimental results show that CERACE can effectively reduce the long-term operation cost of collaborative cloud-side computing system in various settings. In this paper, we do not consider the cooperation of edge nodes. In the future, we want to extend the analysis to the case with more edge nodes, where edge nodes can cooperate with each other to accomplish the users' tasks. For example, the edge node can use the idle resources of adjacent nodes for

the collaborative computing.

ACKNOWLEDGMENT

This paper was funded by the Shenzhen Fundamental Research Program (Grant No.JCYJ20190809175613332), the Humanity and Social Science Youth Research Foundation of Ministry of Education (Grant No.19YJC790111) and the Philosophy and Social Science Post-Foundation of Ministry of Education(Grant No.18JHQ060).

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] S. Weisong, Z. Xingzhou, W. Yifan, and Z. Qingyang, "Edge computing: State-of-the-art and future directions," *Journal of Computer Research and Development*, vol. 56, no. 1, p. 69, 2019.

[3] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[4] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman, "Bringing the cloud to the edge," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2014, pp. 346–351.

[5] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2011, pp. 1–12.

[6] S. Abrishami, M. Naghibzadeh, and D. H. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 158–169, 2013.

[7] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," *Future Generation Computer Systems*, vol. 48, pp. 1–18, 2015.

[8] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222–235, 2014.

[9] Z. Zhan, X. Liu, Y. Gong, J. Zhang, H. S. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, pp. 1–33, 2015.

[10] K. Zhang, Y. Mao, S. Leng, S. Maharjan, A. Vinel, and Y. Zhang, "Contract-theoretic approach for delay constrained offloading in vehicular edge computing networks," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 1003–1014, 2019.

[11] X. Guo, R. Singh, T. Zhao, and Z. Niu, "An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems," in *2016 IEEE International Conference on Communications (ICC)*. IEEE, 2016, pp. 1–7.

[12] T. Zhao, S. Zhou, X. Guo, Y. Zhao, and Z. Niu, "A cooperative scheduling scheme of local cloud and internet cloud for delay-aware mobile cloud computing," in *2015 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2015, pp. 1–6.

[13] J. L. G. Gross, K. J. Matteussi, J. C. dos Anjos, and C. F. R. Geyer, "A dynamic cost model to minimize energy consumption and processing time for iot tasks in a mobile edge computing environment," in *International Conference on Service-Oriented Computing*. Springer, 2020, pp. 101–109.

[14] J. Wang and L. Wang, "A computing resource allocation optimization strategy for massive internet of health things devices considering privacy protection in cloud edge computing environment," *Journal of Grid Computing*, vol. 19, no. 2, pp. 1–14, 2021.

[15] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Transactions on Automation Science and Engineering*, 2020.

[16] Y. Lin and H. Shen, "Cloudfog: Leveraging fog to extend cloud gaming for thin-client mmog with high quality of service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 431–445, 2016.

[17] B. Shen, X. Xu, F. Dar, L. Qi, X. Zhang, and W. Dou, "Dynamic task offloading with minority game for internet of vehicles in cloud-edge computing," in *2020 IEEE International Conference on Web Services (ICWS)*. IEEE, 2020, pp. 372–379.

[18] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[19] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1281–1290.

[20] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin, and A. Sala, "Smoothed online resource allocation in multi-tier distributed cloud networks," *IEEE/ACM Transactions On Networking*, vol. 25, no. 4, pp. 2556–2570, 2017.

[21] T. Q. Dinh, B. Liang, T. Q. Quek, and H. Shin, "Online resource procurement and allocation in a hybrid edge-cloud computing system," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 2137–2149, 2020.

[22] B. Wang and M. Li, "Resource allocation scheduling algorithm based on incomplete information dynamic game for edge computing," *International Journal of Web Services Research (IJWSR)*, vol. 18, no. 2, pp. 1–24, 2021.

[23] J. Ren, G. Yu, Y. He, and G. Y. Li, "Collaborative cloud and edge computing for latency minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5031–5044, 2019.

[24] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement learning with parameterized actions," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[25] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[26] J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, and H. Liu, "Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space," *arXiv preprint arXiv:1810.06394*, 2018.

[27] H. Shao, W. H. Lam, and M. L. Tam, "A reliability-based stochastic traffic assignment model for network with multiple user classes under uncertainty in demand," *Networks and Spatial Economics*, vol. 6, no. 3, pp. 173–204, 2006.

[28] Z. Zhou and A. Chen, "Comparative analysis of three user equilibrium models under stochastic demand," *Journal of Advanced Transportation*, vol. 42, no. 3, pp. 239–263, 2008.

[29] W. Wang, D. Niu, B. Liang, and B. Li, "Dynamic cloud instance acquisition via iaas cloud brokerage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1580–1593, 2014.

[30] J. P. Champati and B. Liang, "One-restart algorithm for scheduling and offloading in a hybrid cloud," in *2015 IEEE 23rd International Symposium on Quality of Service (IWQoS)*. IEEE, 2015, pp. 31–40.