frontiers | Frontiers in Bioengineering and Biotechnology

ORIGINAL RESEARCH
published: 04 August 2022
doi: 10.3389/fbioe.2022.908056

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

Jianqiao Xu[1], *Zhuohan Xu*[2] *and Bing Shi*[2,3]*

[1]Department of Information Security, Naval University of Engineering, Wuhan, China, [2]School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China, [3]Shenzhen Research Institute of Wuhan University of Technology, Shenzhen, China

The rapid development of mobile device applications 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. According to the nature of cloud service, cloud service can be divided into private cloud and public cloud. In a private cloud environment, the edge node must allocate resources between the cloud node and the edge node. In a public cloud environment, since public cloud service providers offer various pricing modes for users' different computing demands, the edge node also must select the appropriate pricing mode of cloud service; which is a sequential decision problem. In this stydy, we model it as a Markov decision process and parameterized action Markov decision process, and we propose a resource allocation algorithm cost efficient resource allocation with private cloud (CERAI) and cost efficient resource allocation with public cloud (CERAU) in the collaborative cloud-edge environment based on the deep reinforcement learning algorithm deep deterministic policy gradient and P-DQN. Next, we evaluated CERAI and CERAU against three typical resource allocation algorithms based on synthetic and real data of Google datasets. The experimental results demonstrate that CERAI and CERAU can effectively reduce the long-term operating 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.

Keywords: collaborative cloud-edge computing, resource allocation, reinforcement learning, edge computing, Markov decision process

OPEN ACCESS

Edited by:
Gongfa Li,
Wuhan University of Science and Technology, China

Reviewed by:
Bo Li,
Shanghai Dianji University, China
Jianliang Gao,
Central South University, China
Mau-Luen Tham,
Tunku Abdul Rahman University, Malaysia

***Correspondence:**
Bing Shi
bingshi@whut.edu.cn

Specialty section:
This article was submitted to Bionics and Biomimetics, a section of the journal Frontiers in Bioengineering and Biotechnology

Received: *30 March 2022*
Accepted: *30 May 2022*
Published: *04 August 2022*

Citation:
Xu J, Xu Z and Shi B (2022) Deep Reinforcement Learning Based Resource Allocation Strategy in Cloud-Edge Computing System. Front. Bioeng. Biotechnol. 10:908056. doi: 10.3389/fbioe.2022.908056

1 INTRODUCTION

In recent years, the number of mobile devices, such as mobile phones, wearable devices, and sensors increased rapidly. Because of 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. To provide low latency and real-time services to mobile users, edge computing is proposed. Edge computing provides a virtual pool of configurable computing resources, and such resource instances are often referred to as virtual machines (VMs). Edge computing can provide users with low latency, location awareness, and high-quality service Mao et al. (2017); Weisong et al. (2019); Shi et al. (2016).

TABLE 1 | 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_c	Unit cost of VMs in private cloud
$p_{upfront}$	Customization price of reserved instance in public cloud
p_{od}	Unit cost of on-demand instance in public cloud
p_{re}	Unit cost of reserved instance in public cloud
p_t	Unit cost of spot instance in public cloud in time slot t

However, edge nodes usually do not exhibit enough storage and computing resources when processing massive mobile device data. Therefore, collaborative cloud-edge computing has been proposed. To provide users with computing services, cloud and edge nodes cooperate with each other Chang et al. (2014).

In the collaborative cloud-edge computing, the edge node with limited local resources can rent more resources from the cloud node and pay corresponding costs to meet users' demands. Because the computing cost of edge nodes changes dynamically according to their workload, if no strategic resource allocation exists when collaborative cloud-edge computing provides service, the computing resources in edge nodes with a relatively low computing cost cannot be used reasonably, and its computing cost will increase. Therefore, reducing the long-term operation cost on the premise of meeting the dynamic computing demands of users is a key problem in the collaborative cloud-edge computing.

In the real environment, cloud service can be divided into public cloud and private cloud according to their characteristics. In the private cloud, 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 the public cloud, cloud service providers offer various pricing modes for cloud service, so the edge node also 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 under different cloud services. Since it is a sequential decision-making problem, we propose two resource allocation algorithm Cost Efficient Resource Allocation with private cloud (CERAI) and Cost Efficient Resource Allocation with public cloud (CERAU) based on deep reinforcement learning algorithms, deep deterministic policy gradient (DDPG) and P-DQN. Furthermore, we ran experiments to evaluate our algorithm against three typical resource allocation algorithms 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. Next, we run experiments to evaluate the proposed algorithm in **Section 5** and conclude the paper in **Section 6**.

2 RELATED WORK

A lot of works about resource allocation in the cloud computing exists, such as Mao and Humphrey (2011); Abrishami et al. (2013); Malawski et al. (2015); Rodriguez and Buyya (2014). In Zhan et al. (2015), Zhan et al. did a deep survey about resource allocation in the cloud computing. Furthermore, some works about resource allocation in the edge computing also exist. Zhang et al. Zhang et al. (2019) 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. Guo et al. (2016) proposed an optimal policy based on a Markov decision process for scenarios with high mobile device density. Zhao et al. Zhao et al. (2015) proposed an offloading strategy that maximizes the number of tasks served while satisfying the users' latency requirements. Gross et al. Gross et al. (2020) 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, works about resource allocation in the cloud-edge system also exist. Wang et al. Wang and Wang (2021) proposed an optimization strategy for computing resource allocation of massive IoHT devices in cloud-edge computing environment. Yuan et al. Yuan and Zhou (2020) designed a collaborative computation offloading and resource allocation algorithm to maximize the profit of systems and meet the response time constraint. Next, Lin et al. Lin and Shen (2016) proposed a lightweight system called CloudFog to improve the quality of service for the corresponding delay problem in the cloud-based entertainment game. Then, Shen et al. Shen et al. (2020) 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. Zhao et al. (2019) proposed a collaborative approach in the cloud-edge computing to offload tasks to automobiles in vehicular networks. Wang et al. Wang et al. (2017) proposed an online algorithm to dynamically allocate resources in the collaborative cloud-edge system. Jiao et al. Jiao et al. (2017) 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. Dinh et al. (2020) considered a hybrid cloud-edge computing system where edge devices with limited local resources can rent more resources from cloud nodes.

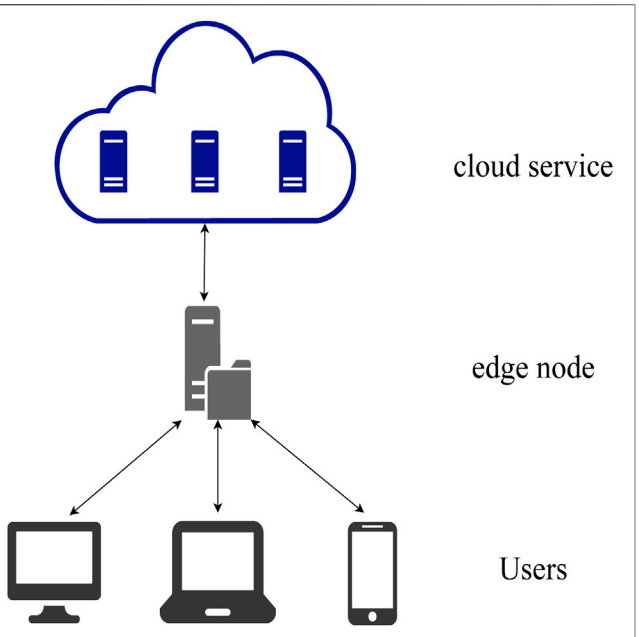

FIGURE 1 | Collaborative cloud-edge computing system.

Wang et al. Wang and Li (2021) 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 they 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.

3 BASIC SETTINGS

In this section, first, we introduce how the collaborative cloud-edge system works, and then, we introduce the basic settings. In conclusion, we describe our problems. We list the key symbols used in this paper in **Table 1**.

3.1 The Collaborative Cloud-Edge Environment

We consider the resource allocation problem in the multi-level collaborative computing environment Ren et al. (2019); Dinh et al. (2020) of "user-edge-cloud," as shown in **Figure 1**. Our research focuses on the resource allocation strategy under collaborative cloud-edge. Therefore, in order to simplify the problem model, our model includes a single cloud node and an edge node, which can also be extended the setting with multiple edge nodes. According to the character of a cloud service, the cloud service can be divided into private cloud and public cloud.

In a private cloud environment, the edge node exhibits 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. However, the cost of private cloud changes dynamically according to its physical computing cost, so the edge node needs to dynamically allocate resources at each time slot according to its resource allocation policy. After the edge node allocates resources through its policy, the computing cost of the edge node and private cloud in this time slot can be calculated and then transfer to the next time slot to receive new computing tasks.

In a public cloud environment, different from the private cloud, the public cloud provides a variety of cloud service pricing modes according to the demand characteristics of different demands. When using the cloud services of the public cloud, you need to pay according to the pricing mode. Cloud service providers such as Amazon, Microsoft, and Alicloud provided three different pricing modes[1], each of which exhibits 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.

3.2 User Setting

As described in **Section 3.1**, 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 the following:

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

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

3.3 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 exhibits no available resources, it will hand over all the arriving computing tasks to the cloud service for processing. Now, we demonstrate the following:

[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 approximately 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.

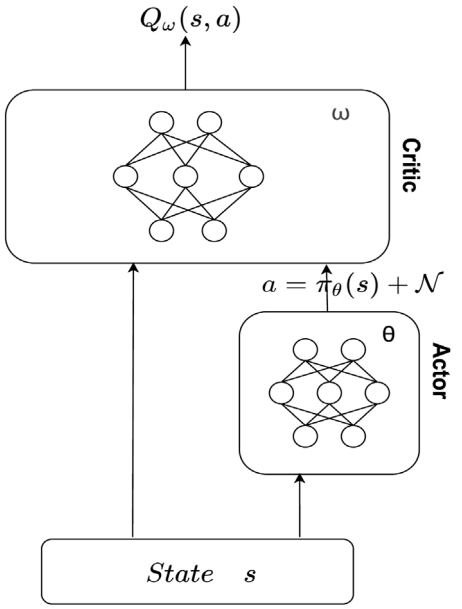

FIGURE 2 | Structure of deep deterministic policy gradient (DDPG).

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

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, where d_t^e is the number of VMs provided by the edge node in this allocation, and 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:

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

At the end of each time slot, the edge node traverses these m records in the allocation record list H and then subtract one 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 completed the computing task and are waiting to be released at the end of time slot t is defined as η_t. Then, we demonstrate the following:

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

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 the following:

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

Note that in order to quickly respond to users' computing demands, even if no computing demand is found, the machine still exhibits 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 the following:

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

$e_t p_e$ 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.

3.4 Cost of Collaborative Cloud-Side Computing in Private Cloud

In time slot t, the cost of collaborative cloud-edge in private cloud environment is the following:

$$C_t^{pri} = d_t^c p_c + C_t^e \tag{8}$$

d_t^c is the number of VMs provided by cloud node, p_c is the unit cost of VMs in private cloud, and C_t^e is the cost of the edge node.

3.5 Cost of Collaborative Cloud-Side Computing in Public Cloud

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

$$\begin{aligned} C_t^{pub} &= X_1 p_{od} d_t^c + X_2 p_{upfront} + X_3 p_{re} d_t^c + X_4 p_t d_t^c + C_t^e \\ X_i &= \begin{cases} 1, The\ service\ is\ used \\ 0, The\ service\ is\ not\ used \end{cases} \end{aligned} \tag{9}$$

$X_1 p_{od} d_t^c$ 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.

3.6 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 allocates demands to either cloud VMs or its own VMs according to its resource allocation strategy. In a public cloud environment, the edge node additionally determines the type of cloud service to be used. 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 $\sum_{t=1}^{T} C_t$.

4 RESOURCE ALLOCATION ALGORITHM

In this section, we first describe the Markov Decision Process and the parameterized action Markov decision process, and then, we introduce two resource allocation strategy in collaborative cloud-edge environment based on DDPG and P-DQN.

4.1 Markov Decision Process

Referring to the work in Chen et al. (2021b), first, we need to model the collaborative cloud-side computing scenario. 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. Markov decision process is a tuple (S, A, P, r, γ), where S is the finite set of states, A the finite set of actions, P is the probability of state transition, r and γ are the immediate reward and discount factor, respectively. Now, we introduce them in the following details.

- $s_t = (e_t, \eta_{t-1}, D_t, p_c) \in S$ 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, and p_t is the unit cost of VMs in private cloud in t.
- $a_t = (x_e, x_k) \in A$, where x_e is the ratio of the number of VMs provided by the edge node to the total number of VMs. Also, x_k is the ratio of the number of VMs provided by the cloud node to the total number of VMs.
- $r_t = -C_t^{pri}$ is the reward in each time slot. Note that we want to reduce the long-term operation cost $R = \sum_{i=1}^{T} r(s_i, a_i)$; therefore, the reward function is set as a negative value of the cost.

4.2 Parameterized Action Markov Decision Process

In the public cloud environment, first, the edge node 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 in each time slot t. 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) Masson et al. (2016) is used.

Similar to Markov decision process, PAMDP is a tuple (S, A, P, r, γ). The difference with the Markov decision process is that A is the finite set of parameterized actions. The specific modeling is as follows.

- $s_t = (e_t, \eta_{t-1}, D_t, p_t, \xi_t) \in S$, where p_t is the unit cost of spot instance in t, and 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 $K = \{k_1, k_2, k_3\}$ is the set of all discrete actions, k_1 is the on-demand instance, k_2 is the reserved instance, and k_3 is the spot instance.
- $r_t = -C_t^{pub}$ is the reward in each time slot.

4.3 Resource Allocation Based on Deep Deterministic Policy Gradient

Reinforcement learning Kaelbling et al. (1996) has been widely used to solve the sequential decision-making problems. When faced with large or continuous state space, conventional reinforcement learning suffers from "curse of dimensionality." In view of the widespread use of deep learning Huang et al. (2021); Jiang et al. (2021a,b); Huang et al. (2022), DeepMind combined deep learning with reinforcement learning and proposed deep reinforcement learning (DRL). In the collaborative cloud-side environment under the private cloud, the state space and the action space is a continuous space. Therefore, we use DDPG Lillicrap et al. (2015) to solve the resource allocation.

The DDPG algorithm is the classical algorithm of the Actor-Critic algorithm, where the Actor generates actions based on policies and interacts with the environment, while Critic evaluates Actor's performance through a value function that guides Actor's next action, thus improving its convergence and performance.

DDPG introduces the idea of DQN and contains four networks, where the main Actor network selects the appropriate action a, according to the current state, s and interacts with the environment:

$$a = \pi_\theta(s) + \mathcal{N} \tag{10}$$

where $\mathcal{N}$ is the added noise. For the Critic master network, the loss function is the following:

$$\nabla J(\omega) = \frac{1}{m} \sum_{j=1}^{m} \left(y_j - Q\left(\phi\left(s_j\right), a_j, \omega\right)\right)^2 \tag{11}$$

where y_j is the target Q value and is calculated as the following:

$$y_j = r_j + \gamma Q'\left(\phi\left(s_j'\right), \pi_\theta\left(\phi\left(s_j'\right)\right), \omega'\right) \tag{12}$$

For the Actor master network, the loss function is the following:

$$\nabla J(\theta) = \frac{1}{m} \sum_{j=1}^{m} \left[\nabla_a Q(s_i, a_i, \omega) \mid_{s=s_i, a=\pi_\theta(s)} \nabla_\theta \pi_\theta(s) \mid_{s=s_i} \right] \tag{13}$$

The parameters ω of the Actor target network and the parameters θ of the Critic target network are updated using a soft update:

$$\begin{aligned} \omega' &\leftarrow \tau\omega + (1-\tau)\omega' \\ \theta' &\leftarrow \tau\theta + (1-\tau)\theta' \end{aligned} \tag{14}$$

DDPG structure is shown in **Figure 2** and the CERAI algorithm is shown in Algorithm 1. The input of the algorithm contains information about the user requests demands D_t and the unit cost of VMs in private cloud p_c. At

the beginning of each iteration of the algorithm, the edge node first needs to obtain the state s_t of the collaborative cloud-edge environment and then pass the state as the input of the neural network into the main Actor network to obtain the action a_t. After the edge node gets the action, the number of demands to be processed by the edge node and the number of demands to be processed by the private cloud will be calculated by the action value, i.e., d_t^e and d_t^c, respectively. Then, interaction with the environment based on d_t^e and d_t^c, to get the next state, reward, and termination flag. Storing this round of experience to the experience replay pool, CERAI will sample from the experience replay pool and calculate the loss functions of Actor and Critic to update the parameters of the master and target networks. After one round of iterative, the training will be continued to the maximum number of training rounds set to ensure the convergence of the resource allocation policy.

Algorithm 1. Cost efficient resource allocation with private cloud (CERAI).

Initialize Actor main network and target network parameters θ, θ', Critic main network and target network parameters ω, ω', soft update coefficient τ, number of samples for batch gradient descent m, maximum number of iterations M, random noise $\mathcal{N}$ and experience replay pool K;
for $i = 1$ *to* M **do**
  Receive user task information and obtain the status s of collaborative cloud-edge computing environment;
  Actor main network selects actions according to s: $a = \pi_\theta(\phi(s)) + \mathcal{N}$;
  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 K;
  Update status: $s = s'$;
  Sample m samples from experience replay pool P, calculate the target Q value y according to the (12);
  Calculate the loss function according to (11) and update the parameters of the Critic main network;
  Calculate the loss function according to (13) and update the parameters of the Actor main network;
  update the parameters of the Critic and Actor target network according to (14) ;
  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

4.4 Resource Allocation Based on P-DQN

In the public cloud environment, 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 Xiong et al. (2018) 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, because of $x_e + x_k = 1$, 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, and $x_k \in X_k$ 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 $x_k(\cdot; \theta)$: $S \rightarrow \mathcal{X}_k$ 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 the following:

$$Q(s, k, x_k(s; \theta); \omega) \approx Q(s, k, x_k; \omega) \tag{15}$$

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

$$y = r + \max_{k \in [k]} Q(s', k, x_k(s', \theta_t); \omega_t) \tag{16}$$

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 the following:

$$l^Q(\omega) = \frac{1}{2}\left[Q(s, k, x_k; \omega) - y\right]^2 \tag{17}$$

In a similar manner, the loss function of a policy network can be written as the following:

$$l^\Theta(\theta) = -\sum_{k=1}^{K} Q(s, k, x_k(s; \theta); \omega) \tag{18}$$

P-DQN structure is shown in **Figure 3**. Now, we propose the resource allocation algorithm based on P-DQN, which is called Cost Efficient Resource Allocation with public cloud (CERAU), as shown in Algorithm 2. The input of the algorithm contains information about the user requests demands D_t and the unit cost of spot instance in public cloud in time slot t p_t. At the beginning of each iteration of the algorithm, the edge node first needs to obtain the state s_t of the collaborative cloud-edge environment and then pass the state as the input of the neural network into the strategy network to obtain the parameter values of each discrete action. After the edge node gets the action, it will select the appropriate public cloud instance type based on the discrete values in the action and determine the number of public cloud instances to be used based on the parameter values. Then, interaction with the environment occurs, to get the next state, reward, and

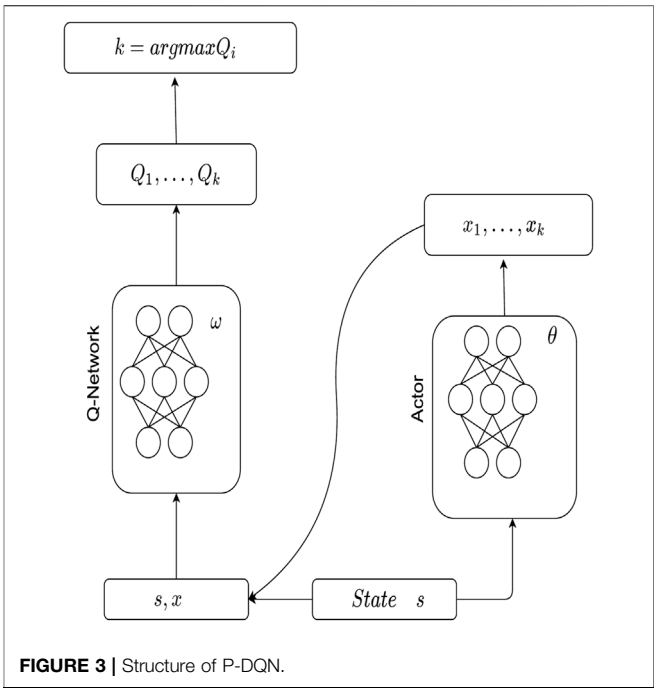

FIGURE 3 | Structure of P-DQN.

TABLE 2 | 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
$p_c \sim U(1, 5)$	The uniform distribution of the computing cost of private cloud
$p_{od} = 3.0$	Unit price of on-demand instance in public cloud
$p_{upfront} = 800$	Customization price of reserved instances in public cloud
$p_{re} = 1.5$	Unit price of reserved instance in public cloud
$T_r = 20$	Reserved duration of reserved instance in public cloud
$p_t \sim N(1.5, 1)$	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
$p = 100{,}000$	Size of experience pool
$m = 128$	Size of the batch sample in the experience pool

termination flag. Storing this round of experience to the experience replay pool, CERAU will sample from the experience replay pool and calculate the gradient of the value network and the policy network. Then, it will update the parameters of the corresponding networks. After one round of iterative, to ensure the convergence of the resource allocation policy, the training will be continued to the maximum number of training rounds set.

Algorithm 2. Cost Efficient Resource Allocation with public cloud (CERAU).

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 $\mathcal{N}$ 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: $x_k \leftarrow x_k(s_t, \theta_t) + \mathcal{N}$;
  Selects discrete actions according to ϵ-greedy strategy:
  $a = \begin{cases} \text{random discrete action}, rnd < \epsilon \\ (k, x_k), k = \arg\max_{k \in [K]} Q(s, k, x_k; \omega), rnd \geq \epsilon \end{cases}$;
  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 (16);
  Update status: $s = s'$;
  Calculate gradient $\nabla_\omega l^Q(\omega)$ and $\nabla_\theta l^\Theta(\theta)$ according to (17) and (18);
  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

5 EXPERIMENT

5.1 Parameter Settings

The parameter settings used in this experiment are shown in **Table 2**. 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 Shao et al. (2006); Zhou and Chen (2008), 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 uniform distribution of the computing cost of private cloud is $p_c \sim U(1, 5)$. 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, the assumption exists 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 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.

5.2 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.

First, we introduce the synthetic data. Similar to Dinh et al. (2020); Wang et al. (2014), 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 intensity. In particular, since the demand D_t consists of the number of VMs requested d_t and the computing time duration l_t (see **Eq. 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 $d_t \sim N(5, 5^2)$

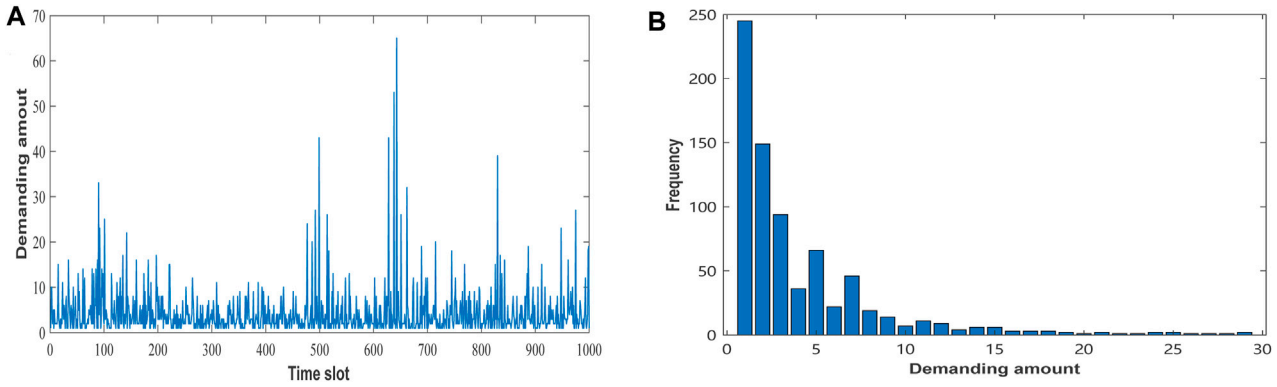

FIGURE 4 | Demanding amount of Google dataset.

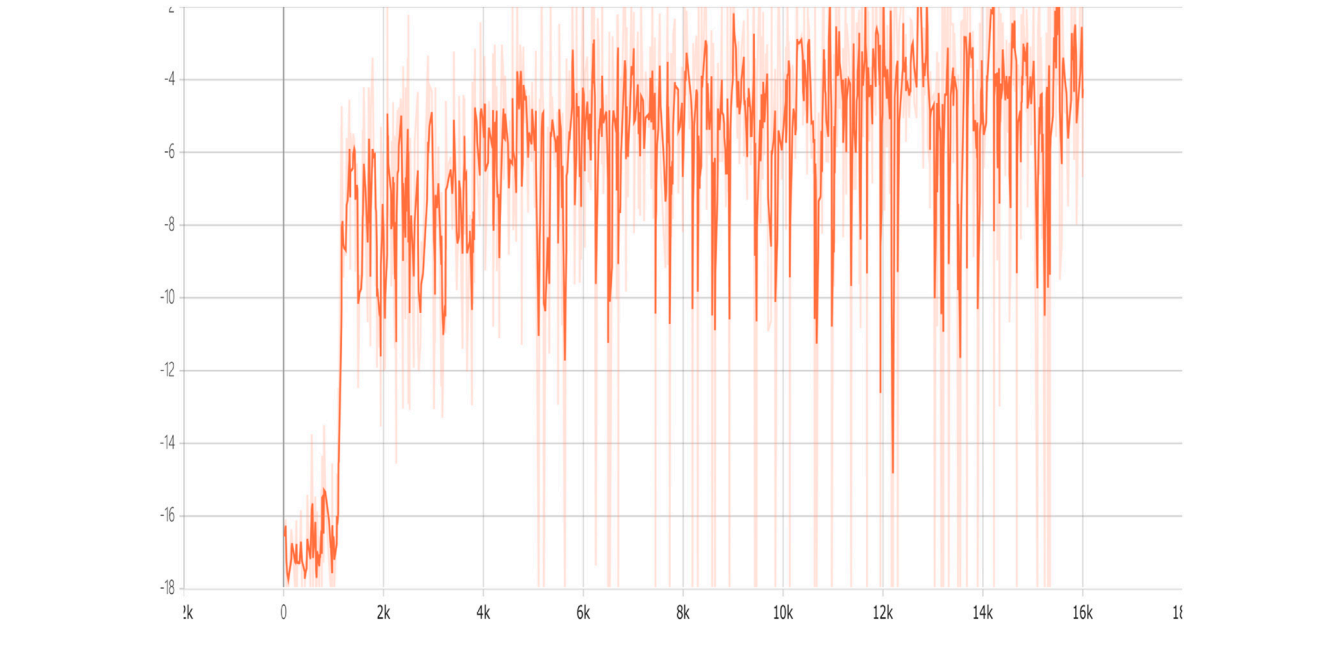

FIGURE 5 | Accumulated reward versus training episodes of CERACI.

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

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

Also, we use Google cluster-usage traces data to further evaluate the proposed algorithm. Because no demanding information exists about computing time duration in this data, we assume that the computing time duration satisfies the normal distribution $l_t \sim N(10, 10^2)$. **Figure 4** show the fluctuation and frequency of user demands in Google dataset. As can be seen from **Figure 4A**, in the first 500 time slots, the demanding amount fluctuates less, while in the last 500 time slots, it fluctuates more. From **Figure 4B**, the demanding amount is mainly between 1 and

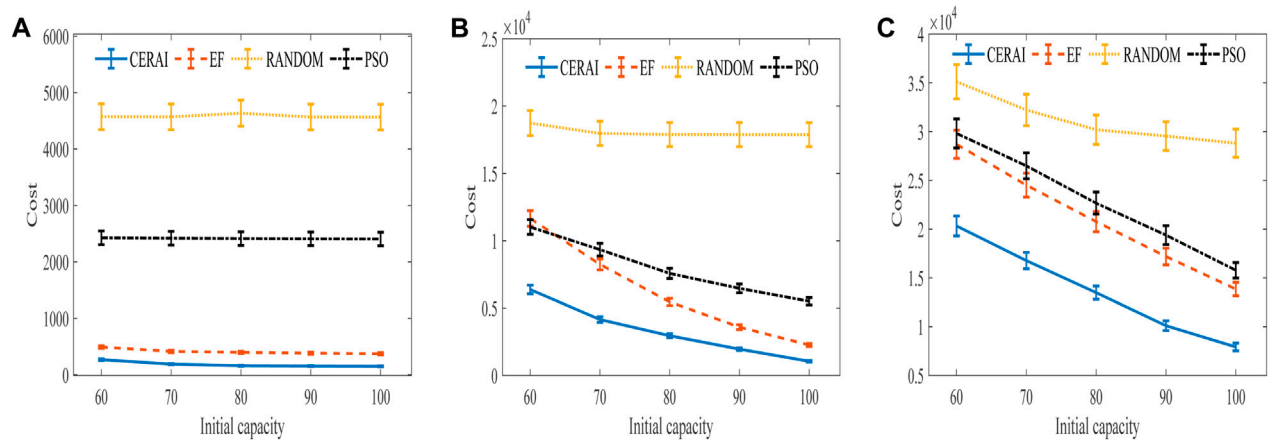

FIGURE 6 | Experimental results of the impact of demanding amount on cost with private cloud. **(A)** Experimental results of low intensity group. **(B)** Experimental results of medium intensity group. **(C)** Experimental results of high intensity group.

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 $E = \{40, 50, 60, 70, 80\}$.

5.3 Benchmark Algorithms

First, we introduce the benchmark algorithms used in the private cloud environment. We evaluate our algorithm with three benchmark algorithms commonly used in existing collaborative cloud-side computing resource allocation works:

- EF (Edge First): this algorithm gives priority to the edge node to process user's requests. Since the unit cost of edge nodes is lower than the unit cost of private clouds, this algorithm can be considered as a greedy algorithm.
- RANDOM: this algorithm randomly selects the edge node or the private cloud service. We add the random allocation algorithm to compare with the state-based resource allocation algorithm to show the strategy of the algorithm and its performance.
- Particle Swarm Optimization (PSO)Clerc (2010): PSO performs well on a wide range of optimization problems Liu et al. (2022). Unlike the above three resource allocation algorithms, PSO directly optimizes the resource allocation actions for each time slot to generate a resource allocation algorithm. That is to say, a sequential decision problem is transformed into a classical optimization problem.

Then, we introduce the benchmark algorithms used in the public cloud environment. Note that the existing similar works usually do not consider the various pricing modes of cloud service Wang et al. (2014); Champati and Liang (2015) in the public cloud. 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 demonstrates 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 determines the quantities for allocation.

5.4 Experimental Analysis for Private Cloud

5.4.1 Impact of Demanding Amount on the Cost

We depict the training curve of CERAI in **Figure 5**, and the experimental results of the impact of demanding amount on the cost are shown in **Figure 6**. It can be seen that CERAI performs better than the other three algorithms under different request intensities.

In the low intensity group (**Figure 6A**), all algorithms vary little for different initial capacities of edge node, but the difference in cost required is large. The RANDOM algorithm demonstrates the highest cost, and the PSO demonstrates the second highest cost. This is because of the high dimensionality of the solution space of this problem and the lack of optimization capability when using PSO, which leads to its inability to search for the optimal solution. In contrast, the strategy-based algorithms EF and CERAI exhibit lower costs and show better performance in this group of experiments. CERAI updates its policy through continuous learning and iteration, which enables it to maintain a high performance, even when resources are abundant. CERAI reduces the cost by more than 45% compared to the suboptimal algorithm EF for different initial capacity of edge node.

In the medium intensity group (**Figure 6B**), it can be seen that RANDOM performance is still the worst, and it remains basically the same for different initial capacities of edge node. Next, the cost of EF, PSO, and CERAI decrease gradually as the initial

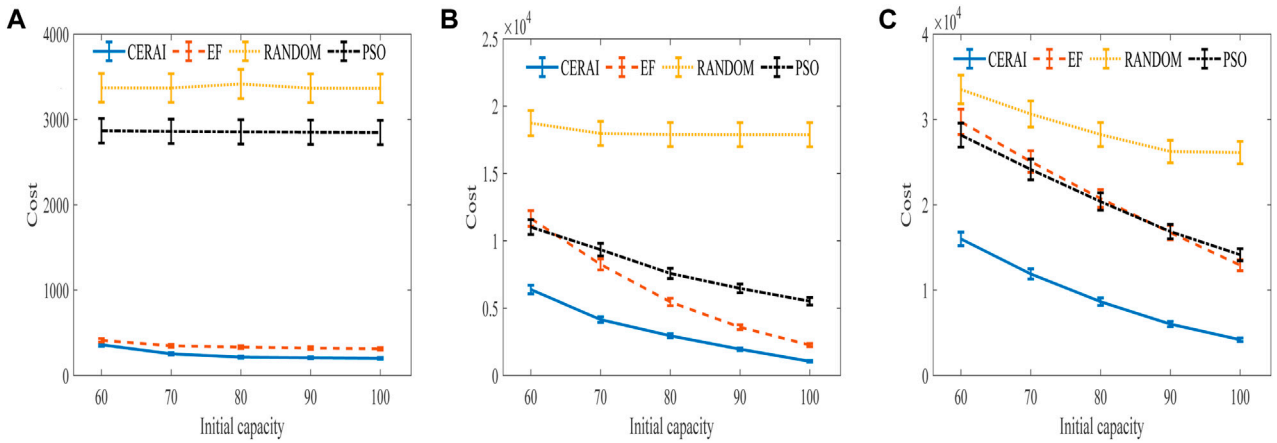

FIGURE 7 | Experimental results of the impact of computing time duration on cost with private cloud. **(A)** Experimental results of low intensity group. **(B)** Experimental results of medium intensity group. **(C)** Experimental results of high intensity group.

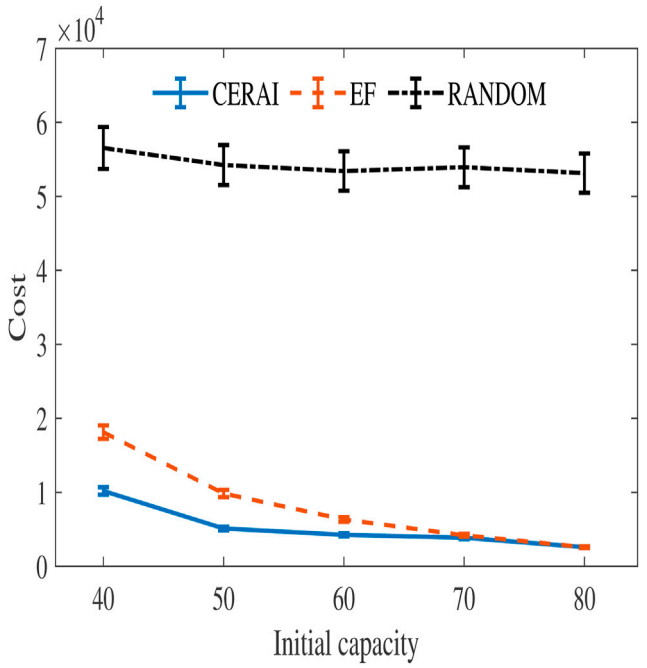

FIGURE 8 | Experimental results based on Google dataset with private cloud.

capacity of the edge nodes rises. CERAI still shows the best performance in this group because CERAI does not use up all the resources of the edge node when user demand arrives, but it strategically reserves some of the resources for upcoming tasks. This results in cost savings by allowing the edge nodes to be used to handle the tasks when demanding amount is high.

In the high intensity group (**Figure 6C**), it can be seen that the cost of all algorithms decreases with the increase of the initial capacity of the edge node. The resources of the edge node are relatively scarce in this group of experiments, so private cloud nodes will be used more. Therefore, the gap between RANDOM and the other three algorithms is relatively reduced in this group compared to the other groups of experiments. The cost difference between the EF and the PSO is smaller because the EF strategy of prioritizing the use of edge nodes limits its performance in the presence of scarce resources of edge nodes. CERAI continues to show the best performance in this group of experiment.

5.4.2 Impact of Computing Time Duration on the Cost

The experimental results of the impact of computing time duration on the cost are shown in **Figure 7**. It can be seen that **Figure 7A,B** are almost the same as **Figure 6A,B**. The difference from the previous group is that in this group of high intensity experiments, PSO outperforms EF when the initial capacity of edge node is less. CERAI continues to show the best performance in this group of experiments. Also, it is worth noting that since the experimental parameters of the medium intensity group in both sets of experiments are identical, the experimental results are also identical.

5.4.3 Experiment Based on Google Dataset

In order to further evaluate the performance of the CERAI in the private cloud environment, we run the experiments on the Google dataset. The amount of data in the Google dataset experiment is large, and its solution space is too large for the PSO, so it cannot be solved using PSO. Therefore, the experiments in this group compare CERAI with EF and RANDOM. The experimental results are shown in **Figure 8**, and it can be seen that CERAI demonstrates a greater advantage over the EF when the initial resources of the edge node are small. The advantage of CERAI is weakened when the resources of edge node are relatively sufficient.

5.5 Experimental Analysis for Public Cloud

5.5.1 Impact of Demanding Amount on the Cost

The training curve of CERAU are shown in **Figure 9**, and the experimental results of the impact of demanding amount on

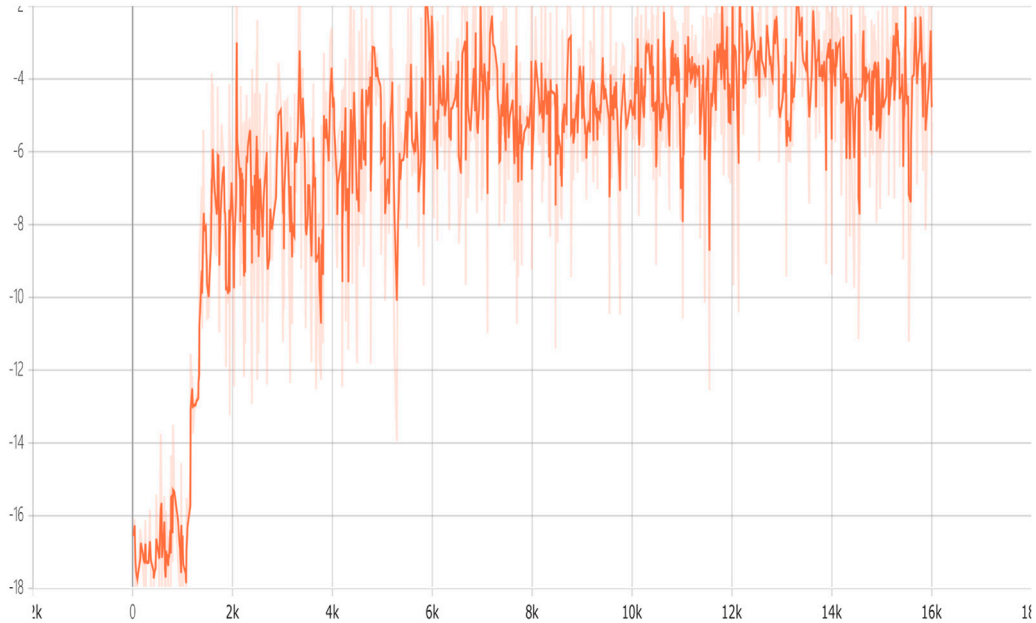

FIGURE 9 | Accumulated reward versus training episodes of CERACU.

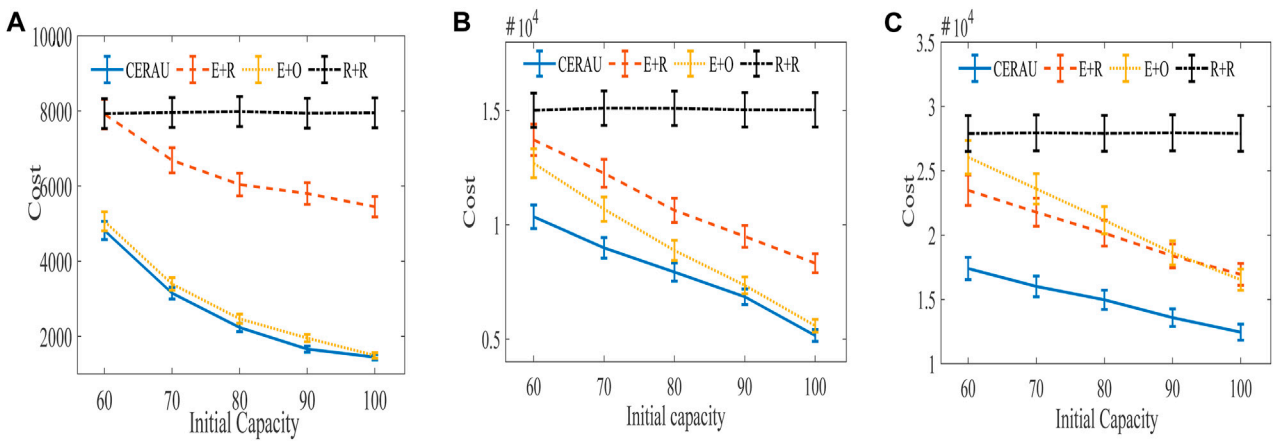

FIGURE 10 | Experimental results of the impact of demanding amount on cost with public cloud. **(A)** Experimental results of low intensity group. **(B)** Experimental results of medium intensity group. **(C)** Experimental results of high intensity group.

the cost with public cloud are shown in **Figure 10**. It can be seen that CERAU performs better than the other three algorithms under different request intensities.

In the low intensity group (**Figure 10A**), R + R demonstrates 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. CERAU and E + O demonstrate 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 demonstrate a good performance on the cost. At this moment, no need to use reserved instances in cloud services is found. Since CERAU 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 (**Figure 10B**), 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. Next, the cost of the other three algorithms decreases gradually with the increase of the capacity of edge nodes. When the initial capacity is low, CERAU exhibits obvious advantages over the algorithm E+O.

In the high intensity group (**Figure 10C**), compared with other algorithms, the cost of CERAU can be reduced by more than 25%

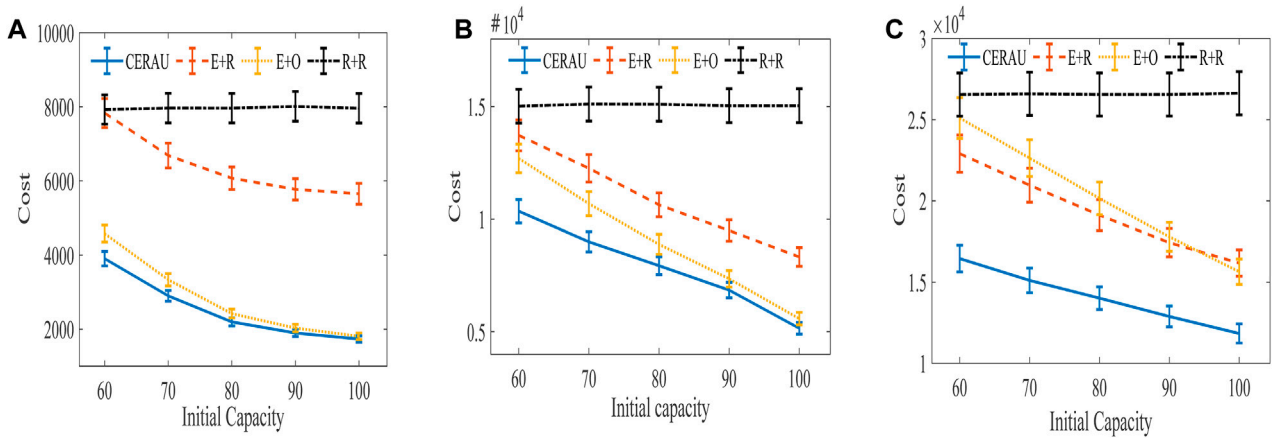

FIGURE 11 | Experimental results of the impact of demanding amount on cost with public cloud. **(A)** Experimental results of low intensity group. **(B)** Experimental results of medium intensity group. **(C)** Experimental results of high intensity group.

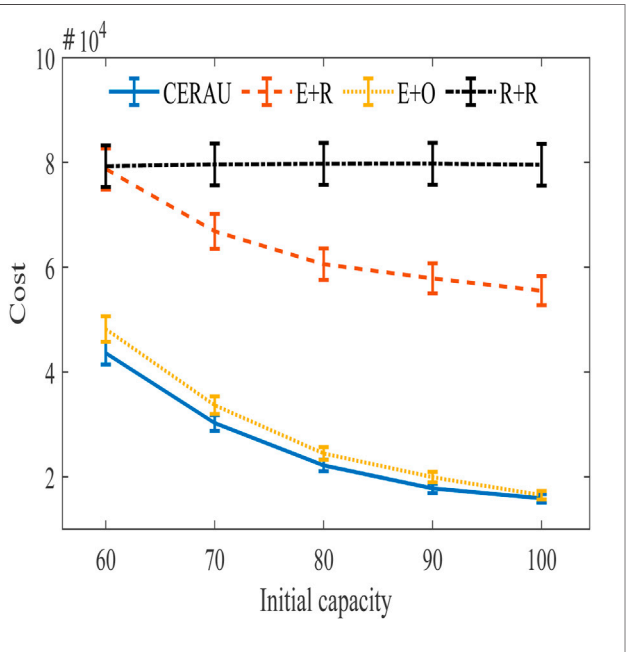

FIGURE 12 | Experimental results based on Google dataset with public cloud.

under different initial capacity of edge nodes. Compared to **Figure 10B**, we find that E + R perform a little better than E + O. This is because given high demand, the capacity of edge nodes is relatively scarce. E + O uses more on-demand instances; therefore, its cost is higher than E + R, which chooses instance type randomly.

5.5.2 Impact of Computing Time Duration on the Cost

The experimental results of the impact of computing time duration on the cost are shown in **Figure 11**. It can be seen that the results are similar to **Figure 6**. In more detail, **Figure 11A,B** are almost the same as **Figure 10A,B**. A slight difference is found between Figure 10(C) and **Figure 11C**. This shows that when the initial capacity of the edge node is insufficient and the user's request intensity is low, CERAU is more sensitive to the change of the user's demanding amount. At this time, CERAU can more effectively reduce the cost. When the user's request intensity is high, CERAU demonstrates almost the same sensitivity to the user's demanding amount and computing time duration, and we can effectively reduce the cost of the system. Similar to CERAI's experiment, since the experimental parameters of the medium intensity group in both sets of experiments are identical, their experimental results are also identical.

5.5.3 Experiment Based on Google Dataset

The experimental results of the Google dataset are shown in **Figure 12**. 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 CERAU. The cost difference between E + O and CERAU 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.

6 CONCLUSION

In this paper, we analyze resource allocation problem in the collaborative cloud-edge computing under private and public clouds, respectively. We model the problem as Markov decision process and PAMDP, and then, we propose the resource allocation algorithm CERAI and CERAU based on the DRL algorithm DDPG and P-DQN. In conclusion, 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 CERAI and CERAU 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. This scenario can be solved by combining it with game theory by referring to the work in Chen et al. (2021a). For example, the edge node can use the idle resources of adjacent nodes for the collaborative computing.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/supplementary material, and further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

JX, ZX, and BS contributed to conception and design of the study. ZX wrote the first draft of the manuscript. JX, and BS wrote sections of the manuscript. All authors contributed to manuscript revision, read, and approved the submitted version.

REFERENCES

Abrishami, S., Naghibzadeh, M., and Epema, D. H. J. (2013). Deadline-constrained Workflow Scheduling Algorithms for Infrastructure as a Service Clouds. *Future Gener. Comput. Syst.* 29, 158–169. doi:10.1016/j.future.2012.05.004

Champati, J. P., and Liang, B. (2015). "One-restart Algorithm for Scheduling and Offloading in a Hybrid Cloud," in 2015 IEEE 23rd International Symposium on Quality of Service (IWQoS) (Portland, OR, USA: IEEE), 31–40. doi:10.1109/iwqos.2015.7404699

Chang, H., Hari, A., Mukherjee, S., and Lakshman, T. (2014). "Bringing the Cloud to the Edge," in 2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) (Toronto, ON, Canada: IEEE), 346–351. doi:10.1109/infcomw.2014.6849256

Chen, T., Peng, L., Yang, J., Cong, G., and Li, G. (2021a). Evolutionary Game of Multi-Subjects in Live Streaming and Governance Strategies Based on Social Preference Theory during the Covid-19 Pandemic. *Mathematics* 9, 2743. doi:10.3390/math9212743

Chen, T., Yin, X., Yang, J., Cong, G., and Li, G. (2021b). Modeling Multi-Dimensional Public Opinion Process Based on Complex Network Dynamics Model in the Context of Derived Topics. *Axioms* 10, 270. doi:10.3390/axioms10040270

Clerc, M. (2010). *Particle Swarm Optimization, Vol. 93*. Hoboken, NJ, USA: John Wiley & Sons.

Dinh, T. Q., Liang, B., Quek, T. Q. S., and Shin, H. (2020). Online Resource Procurement and Allocation in a Hybrid Edge-Cloud Computing System. *IEEE Trans. Wirel. Commun.* 19, 2137–2149. doi:10.1109/twc.2019.2962795

Gross, J. L. G., Matteussi, K. J., dos Anjos, J. C. S., and Geyer, C. F. R. (2020). "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 (Berlin, Germany: Springer), 101–109. doi:10.1007/978-3-030-65310-1_8

Guo, X., Singh, R., Zhao, T., and Niu, Z. (2016). "An Index Based Task Assignment Policy for Achieving Optimal Power-Delay Tradeoff in Edge Cloud Systems," in 2016 IEEE International Conference on Communications (ICC) (Kuala Lumpur, Malaysia: IEEE), 1–7. doi:10.1109/icc.2016.7511147

Huang, L., Chen, C., Yun, J., Sun, Y., Tian, J., Hao, Z., et al. (2022). Multi-scale Feature Fusion Convolutional Neural Network for Indoor Small Target Detection. *Front. Neurorobot.* 16, 881021. doi:10.3389/fnbot.2022.881021

Huang, L., Fu, Q., He, M., Jiang, D., and Hao, Z. (2021). Detection Algorithm of Safety Helmet Wearing Based on Deep Learning. *Concurrency Comput. Pract. Exp.* 33, e6234. doi:10.1002/cpe.6234

Jiang, D., Li, G., Sun, Y., Hu, J., Yun, J., and Liu, Y. (2021a). Manipulator Grabbing Position Detection with Information Fusion of Color Image and Depth Image Using Deep Learning. *J. Ambient. Intell. Hum. Comput.* 12, 10809–10822. doi:10.1007/s12652-020-02843-w

Jiang, D., Li, G., Tan, C., Huang, L., Sun, Y., and Kong, J. (2021b). Semantic Segmentation for Multiscale Target Based on Object Recognition Using the Improved Faster-Rcnn Model. *Future Gener. Comput. Syst.* 123, 94–104. doi:10.1016/j.future.2021.04.019

Jiao, L., Tulino, A. M., Llorca, J., Jin, Y., and Sala, A. (2017). Smoothed Online Resource Allocation in Multi-Tier Distributed Cloud Networks. *IEEE/ACM Trans. Netw.* 25, 2556–2570. doi:10.1109/tnet.2017.2707142

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement Learning: A Survey. *J. Artif. Intell. Res.* 4, 237–285. doi:10.1613/jair.301

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous Control with Deep Reinforcement Learning. *arXiv Prepr. arXiv: 1509.02971*.

Lin, Y., and Shen, H. (2016). Cloudfog: Leveraging Fog to Extend Cloud Gaming for Thin-Client Mmog with High Quality of Service. *IEEE Trans. Parallel Distributed Syst.* 28, 431–445.

Liu, Y., Jiang, D., Yun, J., Sun, Y., Li, C., Jiang, G., et al. (2022). Self-tuning Control of Manipulator Positioning Based on Fuzzy Pid and Pso Algorithm. *Front. Bioeng. Biotechnol.* 9, 817723. doi:10.3389/fbioe.2021.817723

Malawski, M., Juve, G., Deelman, E., and Nabrzyski, J. (2015). Algorithms for Cost- and Deadline-Constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds. *Future Gener. Comput. Syst.* 48, 1–18. doi:10.1016/j.future.2015.01.004

Mao, M., and Humphrey, M. (2011). "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 (Seattle, WA, USA: IEEE), 1–12. doi:10.1145/2063384.2063449

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutorials* 19, 2322–2358. doi:10.1109/comst.2017.2745201

Masson, W., Ranchod, P., and Konidaris, G. (2016). "Reinforcement Learning with Parameterized Actions," in Thirtieth AAAI Conference on Artificial Intelligence.

Ren, J., Yu, G., He, Y., and Li, G. Y. (2019). Collaborative Cloud and Edge Computing for Latency Minimization. *IEEE Trans. Veh. Technol.* 68, 5031–5044. doi:10.1109/tvt.2019.2904244

Rodriguez, M. A., and Buyya, R. (2014). Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds. *IEEE Trans. Cloud Comput.* 2, 222–235. doi:10.1109/tcc.2014.2314655

Shao, H., Lam, W. H. K., and Tam, M. L. (2006). A Reliability-Based Stochastic Traffic Assignment Model for Network with Multiple User Classes under Uncertainty in Demand. *Netw. Spat. Econ.* 6, 173–204. doi:10.1007/s11067-006-9279-6

Shen, B., Xu, X., Dar, F., Qi, L., Zhang, X., and Dou, W. (2020). "Dynamic Task Offloading with Minority Game for Internet of Vehicles in Cloud-Edge Computing," in 2020 IEEE International Conference on Web Services (ICWS) (Beijing, China: IEEE), 372–379. doi:10.1109/icws49710.2020.00055

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet Things J.* 3, 637–646. doi:10.1109/jiot.2016.2579198

Wang, B., and Li, M. (2021). Cooperative Edge Computing Task Offloading Strategy for Urban Internet of Things. *Wirel. Commun. Mob. Comput.* 2021, 1–21. doi:10.1155/2021/9959304

Wang, B., and Li, M. (2021). Resource Allocation Scheduling Algorithm Based on Incomplete Information Dynamic Game for Edge Computing. *Int. J. Web Serv. Res. (IJWSR)* 18, 1–24. doi:10.4018/ijwsr.2021040101

Wang, L., Jiao, L., Li, J., and Mühlhäuser, M. (2017). "Online Resource Allocation for Arbitrary User Mobility in Distributed Edge Clouds," in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS) (Atlanta, GA, USA: IEEE), 1281–1290. doi:10.1109/icdcs.2017.30

Wang, W., Niu, D., Liang, B., and Li, B. (2014). Dynamic Cloud Instance Acquisition via Iaas Cloud Brokerage. *IEEE Trans. Parallel Distributed Syst.* 26, 1580–1593.

Weisong, S., Xingzhou, Z., Yifan, W., and Qingyang, Z. (2019). Edge Computing: State-Of-The-Art and Future Directions. *J. Comput. Res. Dev.* 56, 69.

Xiong, J., Wang, Q., Yang, Z., Sun, P., Han, L., Zheng, Y., et al. (2018). Parametrized Deep Q-Networks Learning: Reinforcement Learning with Discrete-Continuous Hybrid Action Space. *arXiv Prepr. arXiv:1810.06394.*

Yuan, H., and Zhou, M. (2020). Profit-maximized Collaborative Computation Offloading and Resource Allocation in Distributed Cloud and Edge Computing Systems. *IEEE Trans. Automation Sci. Eng.* 18, 1277. doi:10.1109/TASE.2020.3000946

Zhan, Z.-H., Liu, X.-F., Gong, Y.-J., Zhang, J., Chung, H. S.-H., and Li, Y. (2015). Cloud Computing Resource Scheduling and a Survey of its Evolutionary Approaches. *ACM Comput. Surv.* 47, 1–33. doi:10.1145/2788397

Zhang, K., Mao, Y., Leng, S., Maharjan, S., Vinel, A., and Zhang, Y. (2019). Contract-theoretic Approach for Delay Constrained Offloading in Vehicular Edge Computing Networks. *Mob. Netw. Appl.* 24, 1003–1014. doi:10.1007/s11036-018-1032-0

Zhao, J., Li, Q., Gong, Y., and Zhang, K. (2019). Computation Offloading and Resource Allocation for Cloud Assisted Mobile Edge Computing in Vehicular Networks. *IEEE Trans. Veh. Technol.* 68, 7944–7956. doi:10.1109/tvt.2019.2917890

Zhao, T., Zhou, S., Guo, X., Zhao, Y., and Niu, Z. (2015). "A Cooperative Scheduling Scheme of Local Cloud and Internet Cloud for Delay-Aware Mobile Cloud Computing," in 2015 IEEE Globecom Workshops (GC Wkshps) (Piscataway, NJ, USA: IEEE), 1–6. doi:10.1109/glocomw.2015.7414063

Zhou, Z., and Chen, A. (2008). Comparative Analysis of Three User Equilibrium Models under Stochastic Demand. *J. Adv. Transp.* 42, 239–263. doi:10.1002/atr.5670420304

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.