Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH **Open Access**

Task offloading in hybrid-decision-based multi-cloud computing network: a cooperative multi-agent deep reinforcement learning

Juan Chen[1], Peng Chen[1*], Xianhua Niu[1], Zongling Wu[2], Ling Xiong[1] and Canghong Shi[1]

Abstract
Multi-cloud computing is becoming a promising paradigm to provide abundant computation resources for Internet-of-Things (IoT) devices. For a multi-device multi-cloud network, the real-time computing requirements, frequently varied wireless channel gains and changeable network scale, make the system more dynamic. It is critical to satisfy the dynamic nature of network with different constraints of IoT devices in multi-cloud environment. In this paper, we establish a continuous-discrete hybrid decision offloading model, each device should learn to make coordinated actions, including cloud server selection, offloading ratio and local computation capacity. Therefore, both continuous-discrete hybrid decision and coordination among IoT devices are challenging. To this end, we first develop a probabilistic method to relax the discrete action (e.g. cloud server selection) to a continuous set. Then, by leveraging a centralized training and distributed execution strategy, we design a cooperative multi-agent deep reinforcement learning (CMADRL) based framework to minimize the total system cost in terms of the energy consumption of IoT device and the renting charge of cloud servers. Each IoT device acts as an agent, which not only learns efficient decentralized policies, but also relieves IoT devices' computing pressure. Experimental results demonstrate that the proposed CMADRL could efficiently learn dynamic offloading polices at each IoT device, and significantly outperform the four state-of-the-art DRL based agents and two heuristic algorithms with lower system cost.

Keywords: Computation offloading, Continuous-discrete hybrid decision, Deep reinforcement learning, Internet of Things, Multi-cloud computing

Introduction

With the development of mobile communication networks, the number of Internet-of-Things (IoT) devices, such as smartphones, wearables, and sensors, has a rapid growth. Moreover, the new advanced applications with computation-intensive tasks are emerging [1–3]. However, IoT devices usually have limited computation, battery and communication capacity [4, 5]. To address the conflict between computation-intensive tasks and resource-limited IoT devices, cloud computing has been considered as an emerging paradigm [6, 7], which supports IoT devices to offload some computation tasks to the cloud servers with sufficient computation capability.

Nevertheless, it is still challenging for IoT devices to acquire satisfactory computation services [8]. On one hand, there may be a large number of IoT devices require computation intensive services simultaneously. With limited storage and computation resources, it will be hard for a single cloud server to provide its computation services, especially in hotspot scenario [9]. On the other hand, IoT application relies only on a single cloud server, which increases the risk of cloud server lock-in. Thus, it is more promising to study the scenario with multi-cloud collaboration. Although multi-cloud computing technology maintains satisfactory service requirements of IoT

*Correspondence: chenpeng@mail.xhu.edu.cn
[1] School of Computer and Software Engineering, Xihua University, Chengdu, China
Full list of author information is available at the end of the article

applications, it is still challenging to achieve efficient utilization of computation resources for service charging multi-cloud system. Hence, it is non-trivial to investigate the task offloading mechanism in multi-cloud networks.

So far, many researchers have dedicated to design computing offloading policies. For the case of static optimization, some strategies are proposed in [10–12]. In [10], the authors studied a problem of multi-cloud systems fault-tolerant workflow scheduling, and proposed a fault-tolerant cost-efficient workflow scheduling algorithm based on mathematic method to improve the scientific applications execution reliability and reduce their execution cost, respectively. Besides, ant-colony-based optimization technique was applied in [11] to derive optimal coalition of virtual machines (VMs), and then a first-price sealed-bid auction game was used to allocate and migrate the VMs for multi-cloud environments, where federation profit was improved at the expense of increased latency. Further, Alnoman et al. [12] applied dynamic programming and exhaustive search approach to jointly optimize the power consumption, cloud response time and user energy in heterogeneous cloud radio access cloud-edge networks. The traditional offloading strategies often require complete and accurate network information, which is difficult to obtain in real networks due to highly dynamic multi-cloud networks. Besides, for large-scale dynamic environment, some traditional approaches,normally need a considerable amount of iterations to achieve a satisfying local optimum. Meanwhile, the computation complexity of the above-mentioned traditional solutions increases significantly, which makes them very difficult to be suitable for dynamic environment.

Since deep reinforcement learning (DRL) based methods could make intelligent decision with no prior knowledge through exploring the dynamic network environments [13, 14]. Recently, some researchers apply DRL methods to deal with decision optimization problem in multi-cloud networks [15–21]. In [15], an asynchronous advantage actor critic (A3C) and residual recurrent neural network (R2N2) based scheduler were investigated for heterogeneous edge-cloud environment to obtain optimal energy consumption, response time, Service-Level-Agreement and running cost. Zhang et al. [16] considered a three-layer distributed multi-cloud multi-access edge, and proposed a multi-agent reinforcement learning to make task offloading and resource allocation strategy. Zhao et al. [17] studied a scheduling policy with DRL in a hybrid multi-cloud environment to maximize renewable energy utilization. In [18], a deep Q-network (DQN) based collaborative task placement algorithm was proposed to optimize system utility. In [19], by combining multiple parallel deep neural networks (DNNs) with Q-learning, a deep meta reinforcement learning-based offloading (DMRO) algorithm is applied to migrate complex tasks from IoT devices to edge-cloud servers. Chen et al. [20] proposed a multiple buffer deep deterministic policy gradient (MBDDPG) to learn preferable microservice-based service deployment strategy, and improve the average waiting time. Chen et al. [21] investigated the long-term dynamic task allocation and service migration (DTASM) problem in edge-cloud IoT systems, twin-delayed deep deterministic policy gradient (DDPG) was proposed to minimize the total computing load forwarded to the cloud server while satisfying the seamless service migration constraint.

However, these above methods modeled in either a discrete or a continuous action space, which restricted the optimization of offloading decisions in limited action space. In reality, the action space of offloading problem is generally continuous-discrete hybrid [22]. The agent should decide continuous actions (e.g., offloading ratio or local computation capacity) and discrete (e.g., whether to offload or which cloud server to select) actions to execute offloading computation. Thus, these methods may not perform well when the action space becomes large. On the other hand, if the number of IoT devices or cloud servers is large, the state and action may grow exponentially, which results in a serious performance of convergence and generalizability degradation.

To tackle these problems, this paper investigates hybrid-decision-based collaborative multi-cloud system, where multiple cloud servers are designed to offload computation tasks of IoT devices under time-varying wireless channels and task arrivals. The task offloading optimization problem is formulated to minimize the total system cost in terms of energy consumption of IoT devices and renting charge of cloud servers. Particularly, the decision of each IoT device is interdependent in the hybrid-decision-based multi-cloud environments. To solve the issues of hybrid decision and collaboration among different devices, we address the issues in two steps. To be specific, we first relax discrete action (e.g. cloud server selection) into a continuous set by designing a probabilistic method. Then, a cooperative multi-agent DRL (CMADRL) [23] based framework, which employs centralized training process and distributed execution strategy, is designed to obtain the optimal cloud server selection, offloading ratio and local computation capacity. The major contributions of our work are the following:

- We establish a computation offloading framework for multiple IoT devices with multi-cloud, where the task arrivals, channel gains and computation capacity of cloud servers are time-varying. The dynamic computation offloading problem is formulated to minimize the total system cost of energy consumption and renting charge, by jointly designing the cloud server selection, offloading ratio and local computation capacity.

- We relax discrete decision (i.e. cloud server selection) into a continuous set by designing a probabilistic method. Thus, the continuous-discrete hybrid decision is transformed as a continuous decision. Then, we design a novel CMADRL framework with each IoT device acting as an agent to stabilize the training and alleviate on-device computational burden. That is to say, we use global state information collecting at the proxy server to train a locally observable policy function for each IoT device.
- We conduct extensive simulations to evaluate the performance of the proposed CMADRL. The results demonstrate the superiority of the proposed algorithm by comparing with four state of art DRL-based frameworks and two heuristic algorithms, especially in terms of flexibility to the change of currently processed task, adaptability to the variation of communication resources, and generalizability to the extension of network scale.

The remainder of this paper is organized as follows. The system model and problem formulation are provided in "System model and problem formulation" section. The proposed CMADRL is introduced in "The proposed CMADRL" section. Simulation results section analyzes and discusses the experimental results, and "Conclusions and future work" section concludes this paper.

System model and problem formulation

In this section, we consider a multi-cloud computing system consisting of M cloud servers, base station (BS), a proxy server, and N IoT devices. Each IoT device can communicate with the BS with a wireless link, whereas the BS and cloud servers are connected by a wired link. The proxy server deploying near BS plays a role of training equipment, which assists the BS for centralized training and will be explained in detail in "Multi-agent DRL framework" section. As shown in Fig. 1, a set of cloud servers $\mathcal{M} = \{1, \ldots, M\}$ can provide offloading computing services for a set of IoT devices $\mathcal{N} = \{1, \ldots, N\}$. Without loss of generality, we assume each IoT device $n \in \mathcal{N}$ maintains a computation-intensive task to be processed during each time slot $t \in \mathcal{T}$, where $\mathcal{T} = \{1, \ldots, T\}$. We assume each task data are fine-grained and can be partitioned into subsets of any size [24]. Namely one part to be executed on IoT device n, and the other to be offloaded to one of the cloud servers $m \in \mathcal{M}$ for remotely processing.

Let α_n^t denote offloading ratio, which can be consider as the percentage of the task's data size (in bit) to be offloaded to the cloud server, satisfying $\alpha_n^t \in [0,1]$. Let f_n^t and F_n^{max} denote local and maximum computation

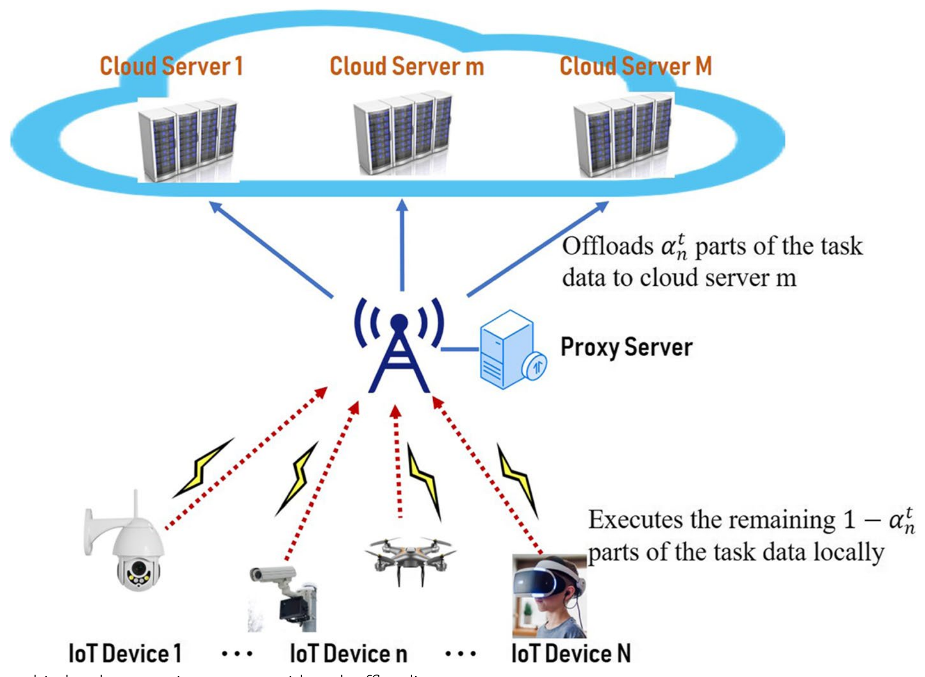

Cloud Server 1
Cloud Server m
Cloud Server M
Offloads α_n^t parts of the task data to cloud server m
Proxy Server
Executes the remaining $1-\alpha_n^t$ parts of the task data locally
IoT Device 1
IoT Device n
IoT Device N

Fig. 1 An example multi-cloud computing system with task offloading

capacity, which can be viewed as the CPU-cycle frequency to process the task data. We assume local computation capacity $f_n^t \in [0,F_n^{max}]$ is flexibly controlled via chip voltage adjustment using the dynamic voltage and frequency scaling (DVFS) technique [25].

At each time slot t, IoT device n needs to decide which cloud server m_n^t to offloading the task, then offloads α_n^t parts of the task data to the cloud server m_n^t for remote computing. Meanwhile, IoT device n executes the remaining 1-α_n^t parts of the task data locally. In other words, the task offloading need to be consider three decisions, including cloud server select m_n^t, offloading ratio α_n^t, the local computation capacity f_n^t.

Since both energy consumption and renting charge play a significant role in the performance evaluation of computation offloading for IoT devices, we consider these two objectives as total system cost. Following illustrates the detailed operation of task queue, local computing, offloading computing and problem formulation, respectively.

Task queue model

We adopt a task queue to represent this dynamic nature of the multi-cloud system. Due to the task may fail to be completed with the limited computation resource of IoT devices, the task execution result in current time slot is relevant to the task load in next time slot. Specifically, computation tasks of IoT device n in time slot t are denoted as Tas_n^t=$\{Ld_n^t,d_n^t,\bar{D}_n^t,c_n^t\}$, in which $Ld_n^t,d_n^t,\bar{D}_n^t$ and c_n^t indicate the size of computation data in the task queue (in bits), the currently processed task data size (in bits), the maximum tolerable delay, and required computational resources to complete the whole task (in CPU cycles/bit) [26]. In addition, we update Ld_n^{t+1} with the residual task in current time slot and the new arrived task in next time slot, which is given as

$$Ld_n^{t+1} = \left[Ld_n^t - d_n^t \mathbf{1}_{drop_n^t=False}\right]^+ + \breve{d}_n^{t+1} \tag{1}$$

where $[x]^+$=max(x,0), and $\breve{d}_n^{t+1}$ is a new arrived task generated in next time slot $t+1$. $drop_n^t$ is a bool variable, $drop_n^t$=*False* represents the task of IoT device n in time slot t is processed successfully.

Local computing

For partial task data $(1-\alpha_n^t)\cdot d_n^t$, the processing delay $D_n^{loc}(t)$ and energy consumption $E_n^{loc}(t)$ incurred on IoT device n, are given as [27]

$$D_n^{loc}(t) = \frac{(1-\alpha_n^t)\cdot d_n^t \cdot c_n^t}{f_n^t} \tag{2}$$

$$E_n^{loc}(t) = \kappa \cdot (f_n^t)^2 \cdot (1-\alpha_n^t)\cdot d_n^t \cdot c_n^t \tag{3}$$

where κ is an effectively switching capacitance constant.

Offloading computing

To take advantage of the rich computation resources of the multi-cloud servers, computation offloading involves three step. Firstly, the IoT device n offloads the partial task data to an appropriate cloud server m for remote execution. Then, cloud servers handle tasks offloaded from IoT device. Finally, cloud server returns the task execution results to the device. Specifically, computation offloading incurs both transmission delay and energy consumption between the IoT device and the selected cloud server. In this paper, we simplify the system model and assume that the size of task execution results obtained from the cloud server is small [28], therefore the transmission delay and energy consumption of feedback transmission are negligible compared with that for local computing of offloading. Moreover, we assume the multi-cloud server connected to the BS via optic fiber or copper wires, so we ignore the transmission delay between the BS and the selected cloud server.

In order to eliminate wireless channel interference among IoT devices, similar to [29], orthogonal frequency division multiple access (OFDMA) is adopted as a multiple access technology. Thus, the system bandwidth W can be divided into equivalent sub-bands distributed to each IoT device equally. According the Shannon formula, the uplink transmission rate for IoT device n to cloud server m, $v_{n,m}^{tra}(t)$ is

$$v_{n,m}^{tra}(t) = W \cdot \log_2(1 + (P_{n,m}^t \cdot g_{n,m}^t)/\varrho^2) \tag{4}$$

where $P_{n,m}^t$, $g_{n,m}^t$ and ϱ^2 are the transmission power, channel gain and noise power in time slot t, respectively.

The transmission delay and energy consumption incurred for offloading the partial input data $\alpha_n^t \cdot d_n^t$, from IoT device n to the cloud server m, $D_{n,m}^{tra}(t)$ and $E_{n,m}^{tra}(t)$, are

$$D_{n,m}^{tra}(t) = \frac{\alpha_n^t \cdot d_n^t}{v_{n,m}^{tra}(t)} \tag{5}$$

$$E_{n,m}^{tra}(t) = P_{n,m}^t \cdot D_{n,m}^{tra}(t) \tag{6}$$

According to Eq. (3), and Eq. (6), total energy consumption of IoT device n for processing the task data d_n^t , E_n^t, are formulated as

$$E_n^t = E_n^{loc}(t) + E_{n,m}^{tra}(t) \tag{7}$$

The processing delay for computing task in cloud server depends on the task data size and cloud server's computation capacity. For each cloud server, due to one cloud server may also support the computation requests from other IoT devices, its computation capability is time varying. Therefore, we assume the cloud server's remaining computation capacity varies randomly between different time slots but keep fixed in each time slot. Let $f_{occup}^{t} = \mathbf{Pr}(\varepsilon) \cdot f_{ser}^{unit}$ be the occupied computation resource, which is modeled as an i.i.d. Possion process with parameter ε, where f_{ser}^{unitis} is the occupied computation resource for each unit [22]. The computation capacity of server m, f_m^t, can be defined as $f_m^t = F_{ser}^{max} - f_{occup}^{t}$, where F_{ser}^{max} is the maximum computation capacity of cloud server $m \in \mathcal{M}$.

When processing the partial task data $\alpha_n^t \cdot d_n^t$ on cloud server m, the incurred execution delay can be expressed as

$$D_n^m(t) = \frac{\sum_{n=1}^{N} (\mathbf{1}_{m_n^t = m} \cdot \alpha_n^t \cdot d_n^t \cdot c_n^t)}{f_m^t} \tag{8}$$

The executing delay $D_n^m(t)$ brings server charge to an IoT device for a server provider in cloud computing. In other words, due to the IoT device rents the computing resources of the cloud server to execute the task, the cloud computing provider will charge the IoT device. Let $c(f_m^t) = e^{(-\eta)} \cdot (f_m^t - 1) \cdot \beta$ denote price per time unit at computing capability f_m^t [30], where η and β are two coefficients. Therefore, the service charge $C_n^m(t)$ required to execute IoT device n's partial task on cloud server m is obtain by

$$C_n^m(t) = c(f_m^t) \cdot D_n^m(t) \tag{9}$$

Problem formulation

The dynamic computing offloading problem concerned is to minimize the total system cost of the energy consumption E_n^t and the renting charge $C_n^m(t)$ in the long term, as formulated in Eq. (10).

$$\min_{m_n^t, \alpha_n^t, f_n^t} \left(\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{n=1}^{N} \left(\omega_1 \cdot E_n^t + \omega_2 \cdot C_n^m(t) \right) \right) \right) \tag{10}$$

s.t.

$$C1: \quad max(D_n^{loc}(t), D_{n,m}^{tra}(t) + D_n^m(t)) \leq \bar{D}_n^t, \ \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{10a}$$

$$C2: \quad m_n^t \in \mathcal{M}, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{10b}$$

$$C3: \quad 0 \leq \alpha_n^t \leq 1, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{10c}$$

$$C4: \quad 0 \leq f_n^t \leq F_n^{max}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{10d}$$

where ω_1 and ω_2 are the tradeoff weight. Constraint (10a) defines that for an arbitrary task, its actual task completion time cannot exceed its associated maximum tolerable delay. Constraint (10b) defines that for each IoT device, its task can be offloaded to only one of cloud servers. Constraint (10c) specifies offloading ratio is a variable between 0 and 1 for each task. Constraint (10d) states that for each IoT device, the local computation capacity cannot exceed its associated maximum computation capacity. Note that, m_n^t, α_n^t and f_n^t are the continue-discrete hybrid decision variables associated with IoT device n, where α_n^t and f_n^t is continue variable, and m_n^t is discrete variable.

Generally, the objective function and constraints in Eq. (10) are nonconvex, and the challenges of this dynamic computation offloading problem lies in three aspects: (1) *the decision process contains both continuous decisions and discrete decision;* (2) *the decision of IoT devices is highly dynamic with the large solution space;* (3) *the optimal offloading strategy should coordinate among IoT devices.* Therefore, it is intractable to find optimal policies through traditional optimization-based schemes.

The proposed CMADRL

In this section, we first relax the continue-discrete decision variable to continue decision variable. Then, we model a multi-agent Markov Decision Process (MDP) for task offloading optimal problem. Finally, the procedure of cooperative twin delayed DDPG (CMATD3) is introduced in detail.

Discrete decision variable relaxation

To address these challenges in Eq. (10), for the discrete decision variable m_n^t , we adopt a probabilistic method to convert it as a continuous variable. In particular, let $Pro(m_n^t) \in [\frac{m-1}{M}, \frac{m}{M}]$ be the probability of the task offloads to cloud server m. In other words, if the IoT device n chooses a continue decision variable $Pro(m_n^t)$, which can be considered as the IoT device n selects cloud server m to offload its task at time slot t. Taking the number of cloud servers $M = 5$ as an example, if the $Pro(m_n^t) \in \left[\frac{1}{5}, \frac{2}{5}\right]$, we can choose server $m = 2$ to offload the task. Therefore, the total system cost minimization problem can be reformulated as follows,

$$\min_{Pro(m_n^t), \alpha_n^t, f_n^t} \left(\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \left(\sum_{n=1}^{N} \left(\omega_1 \cdot E_n^t + \omega_2 \cdot C_n^m(t) \right) \right) \right) \tag{11}$$

s.t.

$$C1: \quad max(D_n^{loc}(t), D_{n,m}^{tra}(t) + D_n^m(t)) \leq \bar{D}_n^t, \ \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11a}$$

$$C2: \quad 0 \leq Pro(m_n^t) \leq 1, \forall m \in \mathcal{M}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11b}$$

$$C3: \quad 0 \le \alpha_n^t \le 1, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11c}$$

$$C4: \quad 0 \le f_n^t \le F_n^{max}, \forall n \in \mathcal{N}, \forall t \in \mathcal{T} \tag{11d}$$

Based on this setting, DRL agent will be introduced to solve this dynamic offloading problem. For centralized decision making DRL, which require the BS to collect the environment state from all IoT devices. However, this would increase the communication overhead. Therefore, this paper aims at obtaining promising computing offloading solutions with multiagent settings.

Since the network environment is non-stationarity, other agents change their policy in the training process, this leads to the performance of traditional multiagent DRL becomes unstable. In order to guarantee convergence, we design a cooperative multi-agent deep reinforcement learning based framework, which leverages the strategy of centralized training and distributed execution by using locally executable actor networks and fully observable critic networks [31].

MDP Formulation

We model the task offloading optimal problem as a multiagent Markov decision process (MDP). The multi-agent MDP can be denoted by a 4 tuple $(\mathcal{N}, S_n, A_n, r_n)$, where $\mathcal{N}$ is the agent space, S_n is the state space of agent n, A_n is the action space of agent n, and r_n is the reward function of agent n, respectively.

- *Agent space* $\mathcal{N} : \mathcal{N} = \{1, \dots, N\}$, where N is the number of IoT devices, each IoT device acts as an agent. For agent n, $n = 1, 2, \dots, N$, by determining the server selection $Pro(m_n^t)$, offloading ratio α_n^t, and local computation capacity f_n^t, the agent can obtain the minimum total system cost.
- *State space* S_n: For IoT device n, the state $\mathbf{s}_n^t$ is composed of computation task, the channel gain between IoT device n and BS in time slot t, and computation capacity of all cloud servers in the multi-cloud system.

$$\mathbf{s}_n^t = \left\{ Tas_n^t; g_n^t; f_1^t, \cdots, f_m^t \right\} \tag{12}$$

- *Action space* A_n: Since each IoT device is required to determine probability of its selected cloud server $Pro(m_n^t)$, offloading ratio α_n^t and local computing capability f_n^t, the action space can be given by

$$\mathbf{a}_n^t = \left\{ Pro(m_n^t), \alpha_n^t, f_n^t \right\} \tag{13}$$

- *Reward function* r_n: To obtain the near-optimal policy for the task offloading optimization problem in Eq. (11), the numbers of agent should cooperate to minimize the total system cost. In other words, reward function r_n^t is set to instruct agent working at IoT device n to learn to make decisions that satisfy the constraints. The action is successful if the decision variables corresponding to the action do not violate any of the constraints defined in Eqs. (11a) - (11d), then the reward is defined as the product of the reciprocal of the weight sum $(\omega_1 \cdot E_n^t + \omega_2 \cdot C_n^m(t))$ and a constant C_1. Otherwise, the reward is defined as a negative value, which represents a punishment, denoted by $-C_2$. The immediate reward obtained at each time slot t is expressed as

$$r_n(t) = \begin{cases} \frac{C_1}{\omega_1 \cdot E_n^t + \omega_2 \cdot C_n^m(t)} & \mathbf{a}_n^t \textit{ is successful} \\ -C_2 & \textit{otherwise} \end{cases} \tag{14}$$

where C_1 and C_2 are positive constants. It is noted that, to maximize the estimation of discounted accumulative rewards, for a successful action, lower total system cost corresponding multi-cloud offloading decision leads to higher immediate reward.

Multi-agent DRL framework

For centralized decision making DRL, which requires the BS to collect the environment state from all IoT devices and cloud servers. However, with the number of IoT devices or cloud servers increasing, the communication overhead would increase, as well as the state-action space may grow exponentially, resulting in the poor convergence efficiency. To deal with these challenges, we aim at obtaining promising computation offloading solutions with multiagent DRL settings. However, traditional multi-agent DRLs still hit bottlenecks of overestimate and high variance, considering the high-dimensional discreate-continuous action space, The TD3 algorithm is designed to find efficient probability of selected cloud server $Pro(m_n^t)$, offloading ratio α_n^t and local computing capability f_n^t, based on dynamic multi-cloud environments.

In the multi-cloud offloading system, take advantage of local observations at each IoT device, IoT device n determines their server selection $Pro(m_n^t)$, task offloading ratio α_n^t and local compution capacity f_n^t. Thus, twin delayed DDPG (TD3) agent is employed to learn distributed computation offloading policies by jointly optimizing above three variables for each IoT device. This is referred to as the cooperative multi-agent TD3 (CMATD3) framework [32]. Figure 2 is the framework of the CMATD3 in multi-cloud system. Following the centralized training and distributed execution strategy, each agent's actor network makes offloading decision according to the local observation of the network

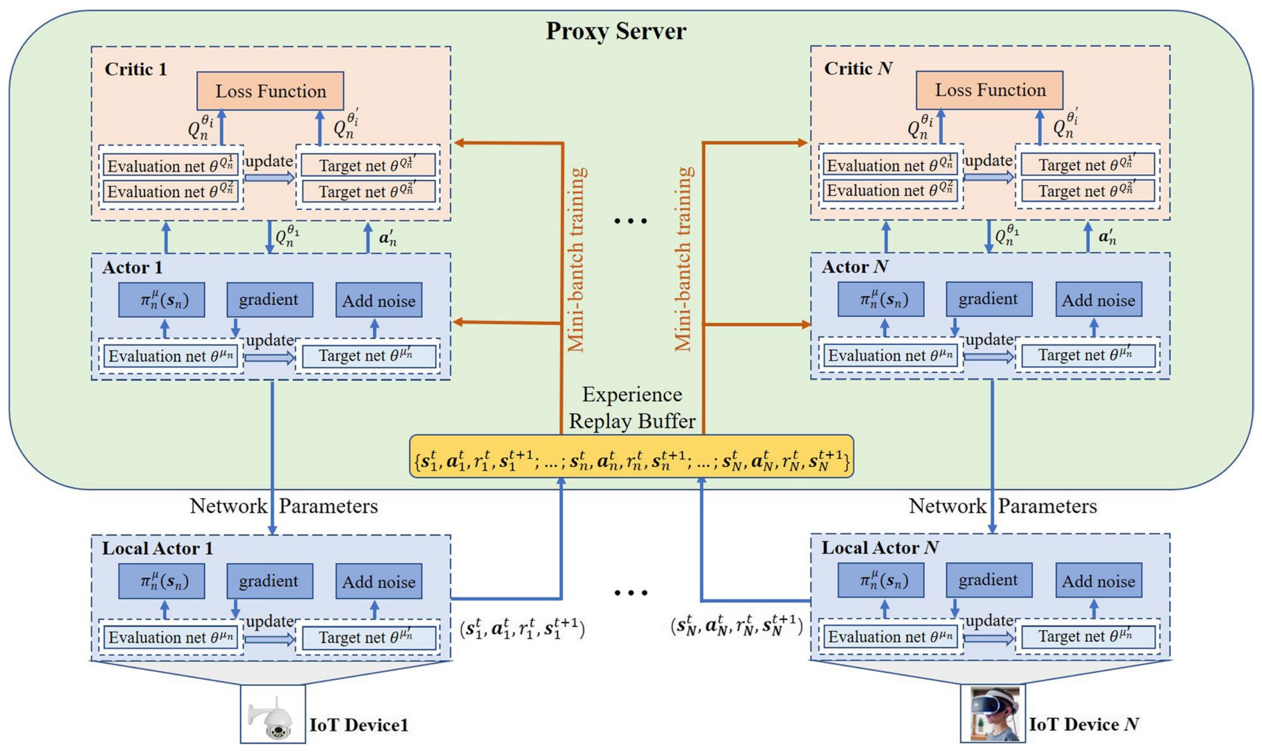

Proxy Server
Critic 1
Loss Function
$Q_n^{\theta_i}$
$Q_n^{\theta_i'}$
Evaluation net $\theta^{Q_n^1}$
Evaluation net $\theta^{Q_n^2}$
update
Target net $\theta^{Q_n^{1'}}$
Target net $\theta^{Q_n^{2'}}$
$Q_n^{\theta_1}$
a_n'
Actor 1
$\pi_n^{\mu}(s_n)$
gradient
Add noise
Evaluation net θ^{μ_n}
Target net $\theta^{\mu_n'}$
Mini-bantch training
Mini-bantch training
Critic N
Actor N
Experience Replay Buffer
$\{s_1^t, a_1^t, r_1^t, s_1^{t+1}; ...; s_n^t, a_n^t, r_n^t, s_n^{t+1}; ...; s_N^t, a_N^t, r_N^t, s_N^{t+1}\}$
Network Parameters
Local Actor 1
Local Actor N
$(s_1^t, a_1^t, r_1^t, s_1^{t+1})$
$(s_N^t, a_N^t, r_N^t, s_N^{t+1})$
IoT Device1
IoT Device N

Fig. 2 The framework of proposed CMATD3 in multi-cloud system

state, which will also be trained in proxy server located near BS. Then, the training parameters are periodically synchronized to each agent's actor network. On the other hand, each agent's two-critic network with global observation is deployed in the proxy server, i.e. states and actions of all agents. Therefore, from the perspective of each agent, the learning environment is stationary, regardless of any agent's policy changes.

The training stages of CMATD3 agent is illustrated as follows. In each time slot t, for each agent n, the global observable two-critic network in the proxy server is exploited to train actor network, so as to obtain the computation offloading strategy. In addition, to stabilize training process and improve the training effectiveness, for each agent n, the local experience transition $\left(\mathbf{s}_n^t, \mathbf{a}_n^t, r_n^t, s_n^{t+1}\right)$ will be store in the experience replay buffer deployed in the proxy server, which concatenates the local experience transition of all agents together as a global experience replay buffer $\mathcal{B}$, expressed as $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}) = \left(\mathbf{s}_1^t, \mathbf{a}_1^t, r_1^t, \mathbf{s}_1^{t+1}; \cdots; \mathbf{s}_n^t, \mathbf{a}_n^t, r_n^t, \mathbf{s}_n^{t+1}; \cdots; \mathbf{s}_N^t, \mathbf{a}_N^t, r_N^t, \mathbf{s}_N^{t+1}\right)$.

Then, for each agent $n, n = 1, 2, \cdots, N$, the actor function is approximated by DNN with parameter θ^{μ_n} as $\pi_n^{\mu}(\mathbf{s}_n)$, which takes the state $\mathbf{s}_n$ as input. Besides, the two-critic network Q-function is also approximated by two DNN with parameter $\theta^{Q_n^i}$ as $Q_n^{\theta_i}(\mathbf{s}, \mathbf{a}|\theta^{Q_n^i})$, $i = 1, 2,,$ which takes the global state $\mathbf{s} = (\mathbf{s}_1, \cdots, \mathbf{s}_N)$ and action set $\mathbf{a} = (\mathbf{a}_1, \cdots, \mathbf{a}_N)$ as input. During the training process, each agent randomly samples a mini-batch $\{\mathbf{s}_j, \mathbf{a}_j, r_j, \mathbf{s}_j'\}_{\mathbf{j=1}}^{\mathbf{I}}$ from the global experience replay buffer $\mathcal{B}$. The policy gradient of the evaluation actor network can be derived as

$$\nabla_{\theta^{\mu_n}} J(\theta^{\mu_n}) \approx \mathbb{E}\left[\nabla_{\theta^{\mu_n}} \pi_n^{\mu}(\mathbf{s}_j) \cdot \nabla_{a_n} Q_n^{\theta_1}(\mathbf{s}, \mathbf{a}|\theta^{Q_n^1})|_{a_n = \pi_n^{\mu}(s_n)}\right] \quad (15)$$

In addition, to avoid over-fitting on the narrow peaks of Q-values, the target action $\mathbf{a}_j'$ is defined as $\mathbf{a}_j' = \pi_n^{\mu'}(\mathbf{s}_j') + \mathbb{N}$, where $\mathbb{N} \backsim clip(\mathfrak{N}(0, \breve{\sigma}^2), -1, 1)$ is clipped noise adding to target actor network with mean 0 and standard deviation $\breve{\sigma}$. This noise helps TD3 to achieve smoother state-action estimation. Based on the target policy smoothing scheme above, the target values y_j is defined as

$$y_j = r_j + \gamma \min_{i=1,2} Q_n^{\theta_i'}\left(\mathbf{s}_j', \mathbf{a}_j'|\theta^{Q_n^i}\right), i = 1, 2. \quad (16)$$

Then, as mentioned above, the two Q-functions, including $Q_n^{\theta_1}(s_j, \mathbf{a}_j)$ and $Q_n^{\theta_2}(s_j, \mathbf{a}_j)$, are concurrently obtained from two-critic network. The weight parameters $\theta^{Q_n^i}$ of $Q_n^{\theta_i}(s_j, \mathbf{a}_j), i = 1, 2$, are updated by minimizing the loss function $L(\theta_i)$, given as

$$L(\theta^{Q_n^i}) \approx \mathbb{E}\left[y_j - Q_n^{\theta_i}(s_j, \mathbf{a}_j)\right]^2, i = 1, 2. \quad (17)$$

Next, based on the Eq. (15) and Eq. (17), let λ be the learning rate, the weight of evaluation actor network and two evaluation critic networks are updated by

$$\begin{aligned}\theta^{\mu_n} &\leftarrow \theta^{\mu_n} - \lambda \nabla_{\theta^{\mu_n}} J(\theta^{\mu_n}) \\ \theta^{Q_n^i} &\leftarrow \theta^{Q_n^i} - \lambda \nabla_{\theta^{Q_n^i}} L(\theta^{Q_n^i}), i = 1, 2.\end{aligned} \tag{18}$$

In the end, to reduce temporal difference (TD) error, each agent updates the evaluation actor network's weights with a lower frequency. Here, the each IoT device updates the evaluation actor network every Γ time slots.

```
1: Initialize each IoT device's actor networks with random weights $\theta^{\mu_n}$ and $\theta^{\mu'_n}$, respectively;
2: Initialize proxy server's critic networks with $\{\theta^{Q^i_n}\}_{i=1,2}$ and $\{\theta^{Q^{i'}_n}\}_{i=1,2}$, respectively;
3: Initialize the proxy server's experience replay buffer $\mathcal{B}$ ;
4: for episode = 1 to $K_{max}$ do
5:   Reset simulation parameters for multi-cloud offloading environment;
6:   Obtain state $s_n$ from for each IoT device agent $n \in \mathcal{N}$ ;
7:   for $t = 1$ to $T$ do
8:     for each IoT device agent $n \in \mathcal{N}$ do
9:       Select an action $a^t_n = \pi^{\mu}_n(s^t_n) + \mathbb{N}$ by applying the local
10:      policy network $\theta^{\mu_n}$ and exploration noise $\mathbb{N}$ to decide cloud server selection, offloading ratio, local computation capacity.
11:      Execute the action $a^t_n$ independently at the IoT device $n$, obtain immediate reward $r^t_n$ and state $s^{t+1}_n$ from IoT device $n$;
12:      Form transition sample $(s_t, a_t, r_t, s_{t+1})$, store it into $\mathcal{B}$;
13:    end for
14:    Proxy server concatenates of experience transition all agents together as $(s_t, a_t, r_t, s_{t+1})$;
15:    for each IoT device agent $n \in \mathcal{N}$ do
16:      Randomly sample a mini-batch $\{s_j, a_j, r_j, s'_j\}^I_{j=1}$ from $\mathcal{B}$;
17:      Update weights $\{\theta^{Q^i_n}\}_{i=1,2}$ of evaluation critic networks by minimizing loss function $L(\theta^{Q^i_n})$ according to Eq. (17);
18:      if $t$ mod $\Gamma$ then
19:        Update weights $\theta^{\mu_n}$ of evaluation actor network by policy gradient according to Eq. (15);
20:        Updates the weights of target actor network and target two critic networks according to Eq. (19);
21:      end if
22:    end for
23:  end for
24: end for
```

Algorithm 1 Training stage for computation offloading via CMATD3Finally, aiming at stabilize the training process, each agent copys the weights of corresponding evaluation networks, and updates the weights of target actor network and target two-critic network. Thus, the the weights of target actor network and target two critic network are obtained as

$$\begin{aligned}\theta^{\mu'_n} &= \eta\theta^{\mu_n} + (1-\eta)\theta^{\mu'_n} \\ \theta^{Q^{i'}_n} &= \eta\theta^{Q^i_n} + (1-\eta)\theta^{Q^{i'}_n}, i = 1, 2.\end{aligned} \tag{19}$$

where η is the updating rate.

The time complexity of Algorithm 1 mainly depends on the number of IoT devices, as well as the structure of the neural networks for executing the actor and two-critic network of each TD3 agent. For each TD3 agent, we assume that number of fully connected layers of actor network and two-critic network is J and $2L$, respectively. Thus, the time complexity can be calculated as

$$\begin{aligned} & N \cdot \left(2 \sum_{l=0}^{L} \left(u_{A,l} \cdot u_{A,l+1} + 4 \sum_{j=0}^{J} \left(u_{C,j} \cdot u_{C,j+1}\right)\right)\right) \\ &= \mathcal{O}\left(N \cdot \left(\sum_{l=0}^{L} \left(u_{A,l} \cdot u_{A,l+1} + \sum_{j=0}^{J} \left(u_{C,j} \cdot u_{C,j+1}\right)\right)\right)\right) \end{aligned} \tag{20}$$

where N is the number of agents in the multi-device, multi-cloud environment, $u_{A,l}$ stands for the unit number of layer l in the actor network, $u_{C,j}$ represents the unit number of layer j in the two-critic network. Note that $u_{A,0}$ and $u_{C,0}$ are the same as input size of actor network and two-critic network, respectively.

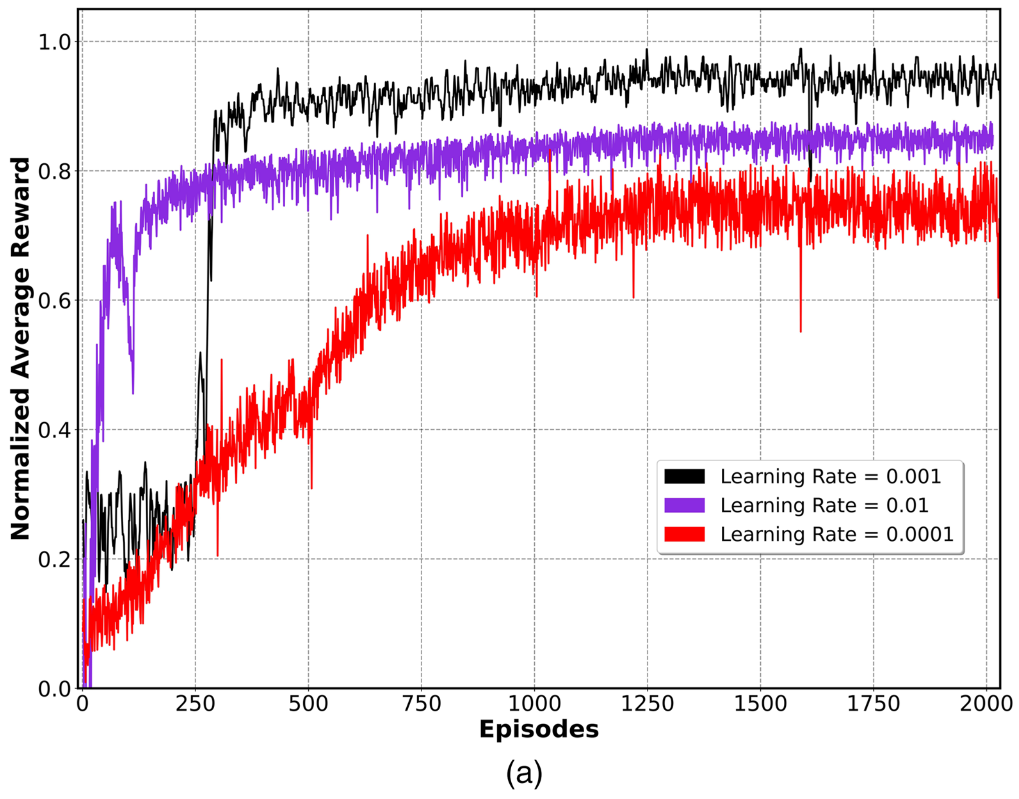
Simulation results

Experiment setup

In this experiment, IoT devices are devised to interact with multi-cloud servers, which is to present in detail how the offloading policy changes with the environment. The $\breve{d}^t_n$ satisfies a Poisson process with the mean data arriving rate 300 kbps. The d^t_n is uniformly distributed in [1, 7] Mbits, the $\bar{D}^t_n$ is uniformly generated in [2, 5] s, the c^t_n is uniformly generated in [200, 500] cycles/bit. Each time slot last 1 s.

Besides, the system parameters are set as: maximum computation capacity $F^{max}_n = 0.5GHz$, the noise power $\varrho^2 = -174dBm/Hz$ [33], transmission power $P_n = 2Watt$, and effectively switching capacitance constant $\kappa = 10^{-27}$. Channel gain g^t_n is exponentially distributed with mean $g_0 \cdot (rad_0/rad_n)^e$, where the path-loss constant $g_0 = -30dB$, the reference distance, $rad_0 = 1m$, the distance between BS and IoT device n, rad_n, and the path-loss exponent $e = 3$, respectively.The computing capability of each cloud server f^t_m is uniformly generated in [2, 6] GHz.

For the proposed CMATD3 framework, both the actor and two-critic networks are four-layer fully connected neural network with two hidden layers, where the number of neurons in the two layers are 400 and 300, respectively. The learning rates of the actor network is initialized as 0.0001. We set the maximum experience



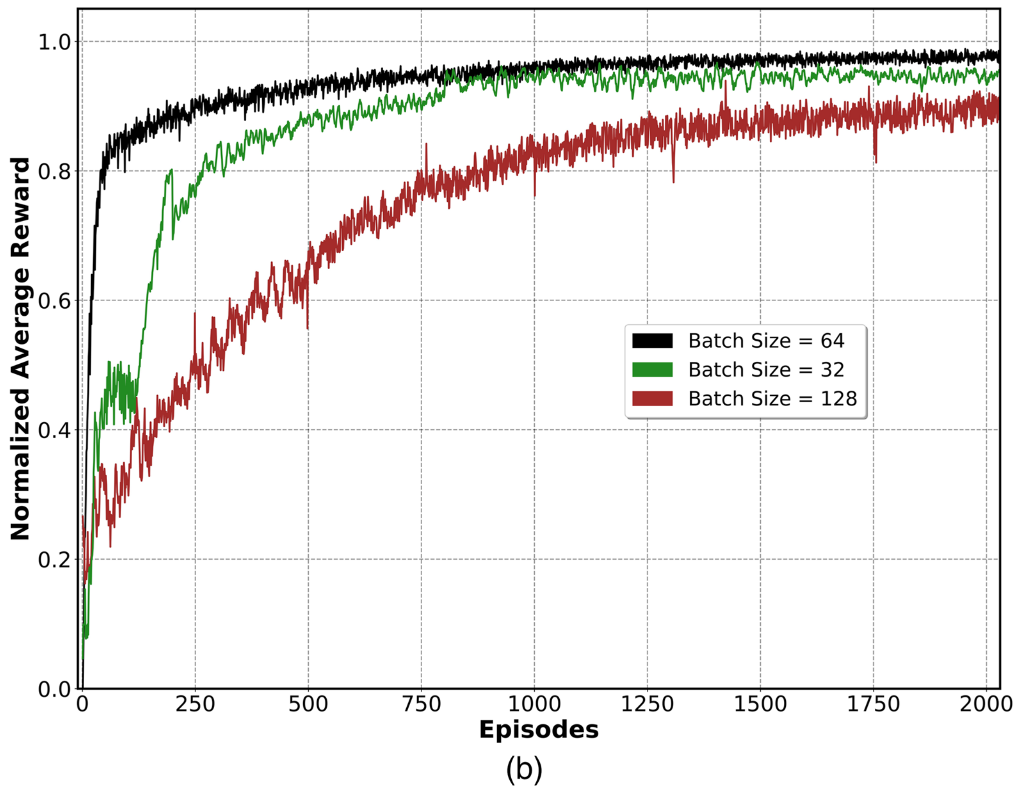


Fig. 3 Normalized average rewards of CMATD3 agent with (**a**) different learning rates and (**b**) different batch sizes

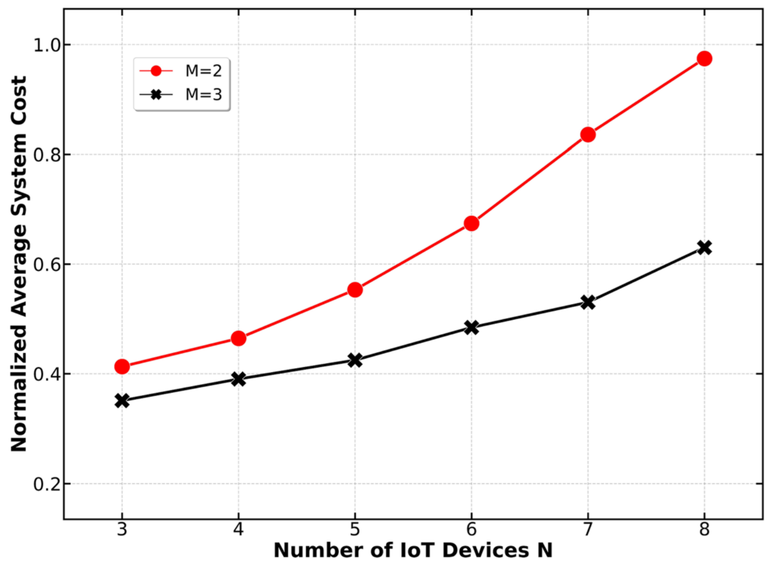


Fig. 4 Normalized average system costs with different number of IoT devices in the case of $M = 2$ and $M = 3$

replay buffer size $\mathcal{B} = 2.5 \times 10^5$, the target net update rate $\eta = 0.005$, and the discount factor $\gamma = 0.99$, respectively. In the training stage, the total number of episodes $K_{max} = 2000$, and maximal time slots in each episode is $T = 200$. The Adam optimizer is used to optimize the loss function during training. In the testing stage, the results obtained in 100 runs are averaged.

We run all experiments on a workstation with Intel Xeon E5-2667V4 8Core CPU×2 @3.2GHz, 128 GB RAM, and 4×NVIDIA GTX Titan V 12G GPU. It takes around 130 sec to run an episode on average.

Parameter study of CMATD3 agent

To verify the training efficiency, we study the impact of parameters on the performance of the proposed CMATD3 agent, including the learning rate and batch size, as shown in Fig. 3(a) and (b). The training process of CMATD3 agent is usually conducted offline. The number of cloud servers M is set to 3, the number of IoT devices N is set to 3. Figure 3(a) shows the normalized average reward of CMATD3 with different learning rates in two-critic networks. With a small learning rate, i.e., 0.0001, the CMATD3 agent cannot reach to high

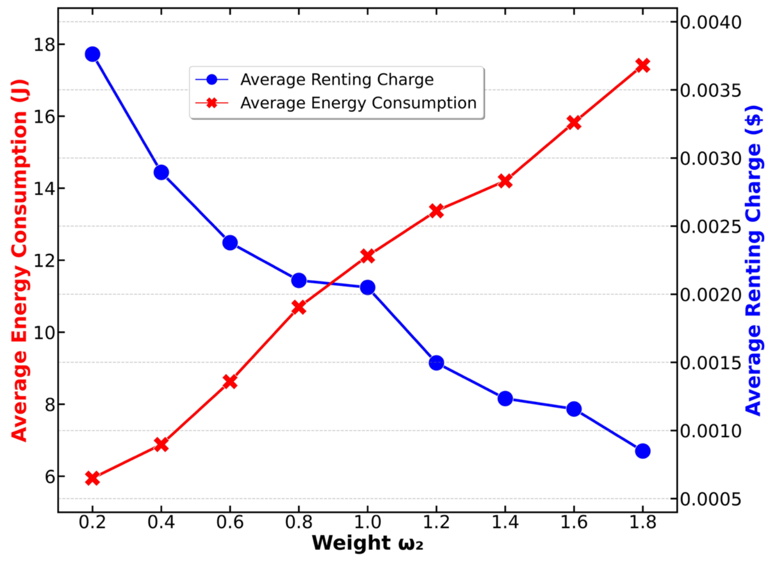


Fig. 5 Energy consumption and renting charge vs weight ω_2

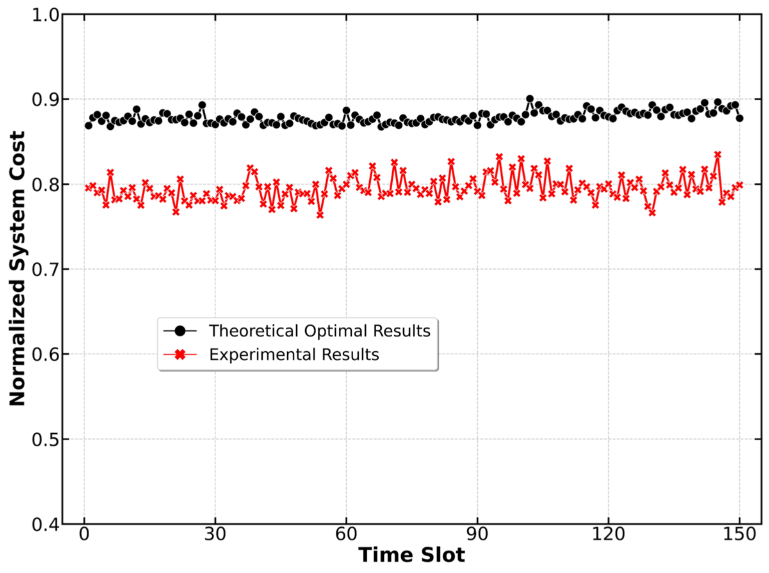


Fig. 6 Performance gap between the proposed CMATD3 and the theoretical optimal result

reward values, since the update of DNN' parameters is trivial. On the contrary, a large learning rate, i.e., 0.01, may leads to rapid changes to the weight parameters of DNN. Obviously, 0.001 is more appropriate than 0.01 and 0.0001. Thus, we hereafter fix the learning rate to 0.001. Figure 3(b) depicts the normalized average rewards of CMATD3 with different batch sizes. As shown in Fig. 3(b), both 32 and 128 lead to a deteriorated training performance, and the cumulative reward curve oscillates at low values. This is because a small batch size cannot efficiently cover the majority of transitions stored in the experience replay buffer. While a large batch size may lead to previously non-effective transitions are frequently sampled and trained from the experience replay buffer. Hence, we hereafter set the batch size to 64.

In order to investigate the scalability of our proposed CMATD3 agent, we evaluate the performance with different numbers of cloud servers and IoT devices, as shown in Fig. 4. We can find that with the number of IoT devices increases, there are more computation waiting to be offloading, which results in the higher total system cost. On the other hand, there are more cloud servers participating in computation offloading, as the number of cloud server M increasing. Besides, when the number of IoT devices and tasks is constant, the more cloud server participating in, the lower total system cost will be obtained. Nevertheless, it is unnecessary for more cloud servers to participate in computing offloading with a few of IoT devices and tasks. Take the number of IoT devices $N = 3$ as an example, the performance of $M = 2$ is almost similar to that of $M = 3$. Moreover, when the number of cloud server M fixes to 3 and the number of IoT device N increases to 8, the CMATD3 agent is still competent for the multi-cloud computation offloading problem. These above verify the high scalability of the proposed CMATD3 agent with regard to cloud servers, state and action spaces.

Figure 5 displays the relationship between the energy consumption and charge renting with the weight parameter ω_2. Note that the weigh parameter ω_1 is set to 1. Specifically, the ω_1 and ω_2 indicate the relative importance of energy consumption and renting charge, respectively. For example, a small ω_2 means more weight putting on the energy consumption. In the Fig. 5, as the weight ω_2 increases from 0.2 to 1.8, the renting charge gets more emphasized, and less tasks are offloaded to cloud server, which results in less renting charge and more energy consumption. Nevertheless, when the ω_2 increases to 1.6 and 1.8, the curve of renting charge decreases slow down. This is because the computation capacity of the cloud servers offered are limited, and less offloaded task data lead to higher energy consumption of IoT devices.

Figure 6 is the performance gap between the proposed CMATD3 and the theoretical optimal result. We obtain the theoretical optimal result at each time slot, and mark it as black line. Besides, the experimental results by implementing CMATD3 according to experiment setup are get. It can be observed that the theoretical optimal results are almost close to 0.9, while the normalized system costs

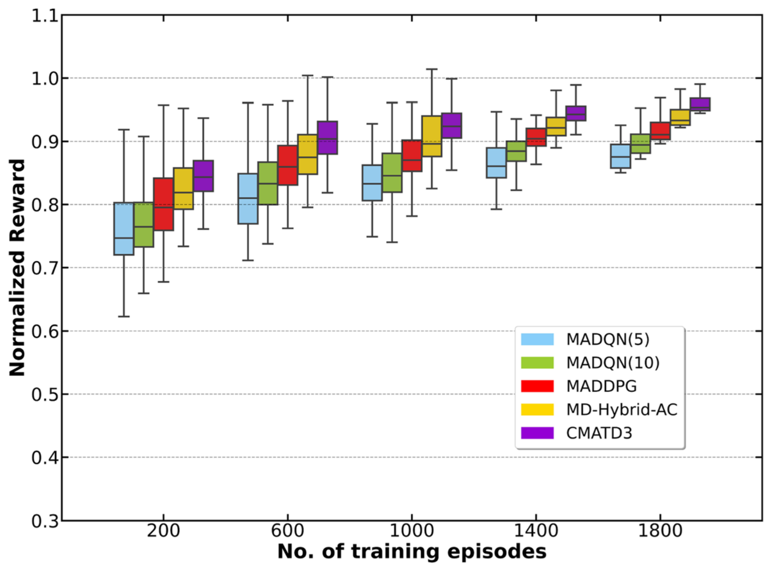


Fig. 7 Normalized reward values obtained during training

oscillate around 0.8. The average gap between optimal result and experimental result is less than 0.1. This is why our proposed CMATD3 can achieve near optimal results.

Performance evaluation and analysis

To validate the effectiveness and advantage of the proposed CMATD3 algorithm for multi-cloud task offloading, we conduct extensive comparative experiments with changing system parameters. On one hand, the performance of four DRL based algorithms (i.e., MADQN(5), MADQN(10), MD-Hybrid-AC [22], and MADDPG [34]) are assessed. On the other hand, two heuristic algorithms (i.e. ACO [11] and SPSO-GA [12]) are also evaluated as follows.

- **MADQN**: Action values will be quantized firstly when coping with dynamic multi-cloud offloading problems with a continuous-discrete hybrid action space. We develop two multi-agent based on the different number of discretized levels. MADQN(5): For an agent allocated at an IoT device, range of both decision variables, e.g. offloading ratio α_n^t and computation capacity f_n^t, are equally divided the into 5 levels. In addition, the range of cloud server selection m_n^t is 3. Thus, the action dimension of each agent is 13. MADQN(10): For each agent, the range of both decision variables, e.g. offloading ratio α_n^t and computation capacity f_n^t, are equally divided the into 10 levels.
- MD-Hybrid-AC [22]: The improvement of actor-critic architecture to tack the continuous-discrete hybrid decision based computation offloading problem, with centralized training and decentralized execution framework adopted.
- MADDPG [34]: A cooperative multi-agent DDPG framework, which is employed to learn decentralized dynamic computation offloading policies.
- CMATD3: The proposed agent in this paper.

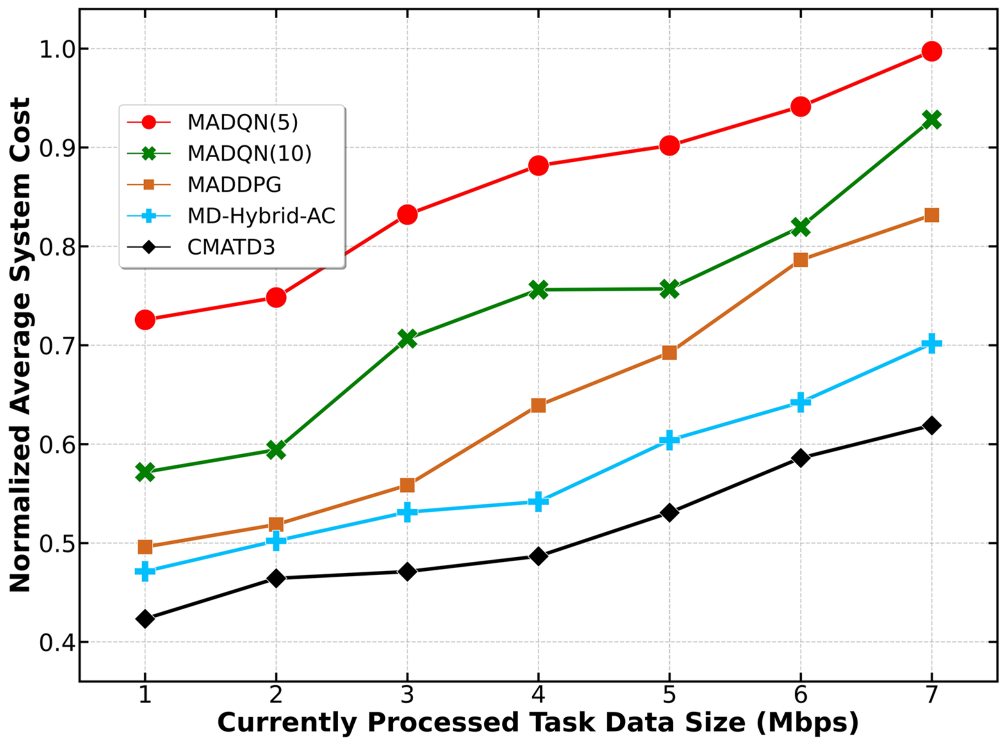
In the MADQN(5), MADQN(10), MD-Hybrid-AC, and MADDPG, the hyperparameters for the DNNs networks are exactly the same with CMATD3.

Convergence of the five algorithms

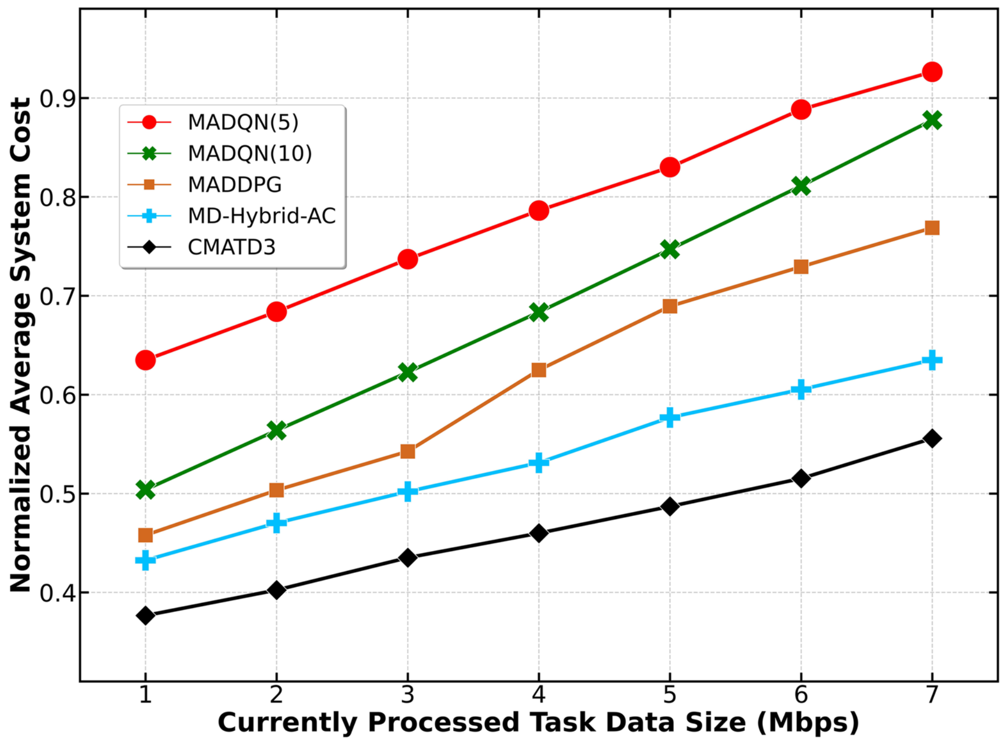
Figure 7 shows the convergence of the five agents during training. We can easily observe that the normalized reward steadily grows up, with training episodes increasing. A larger episode leading to a higher normalized reward. One can see clearly that CMATD3 result in best convergence among all algorithms in terms of the normalized reward. This is because the two independent critic networks in TD3 can efficiently alleviate the overestimation issue, improving the training stability and effectiveness.

Performance comparison against currently processed task data size

Figure 8(a) and (b) display the influence of different currently processed task data size, d_n^t, on the performance of total system cost with the case of cloud server $M = 2$ and $M = 3$. As the task data size increasing, the total energy consumption steadily grows up, leading to the performance of the five DRL agents deteriorates. Furthermore, with more cloud server joining in task offloading, the

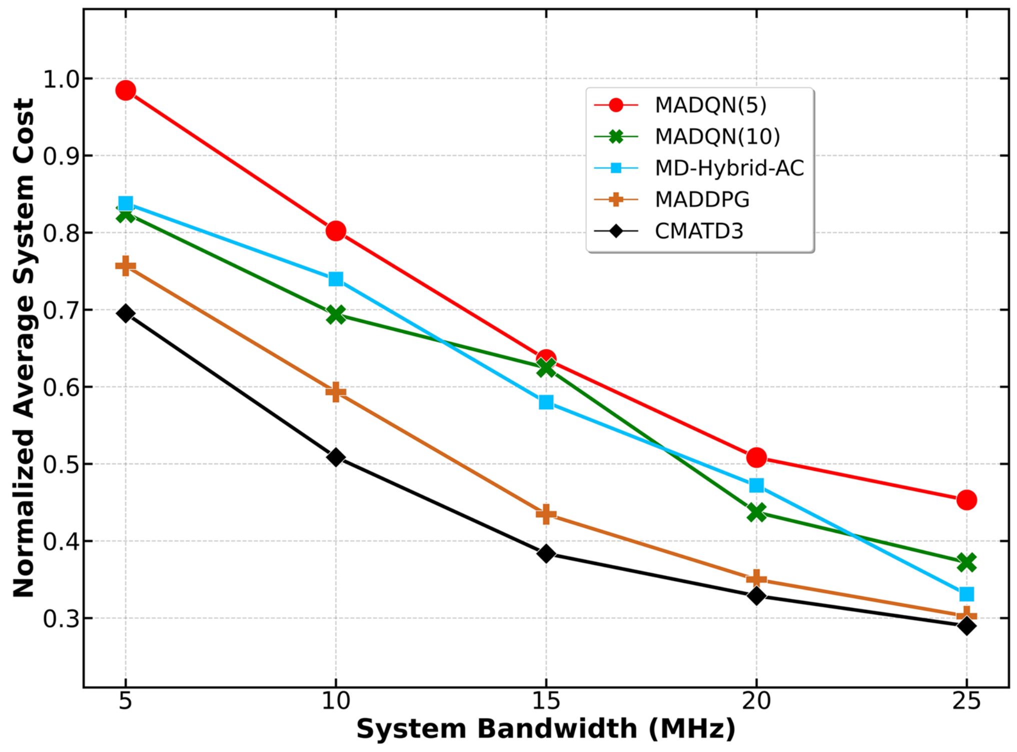


(a)

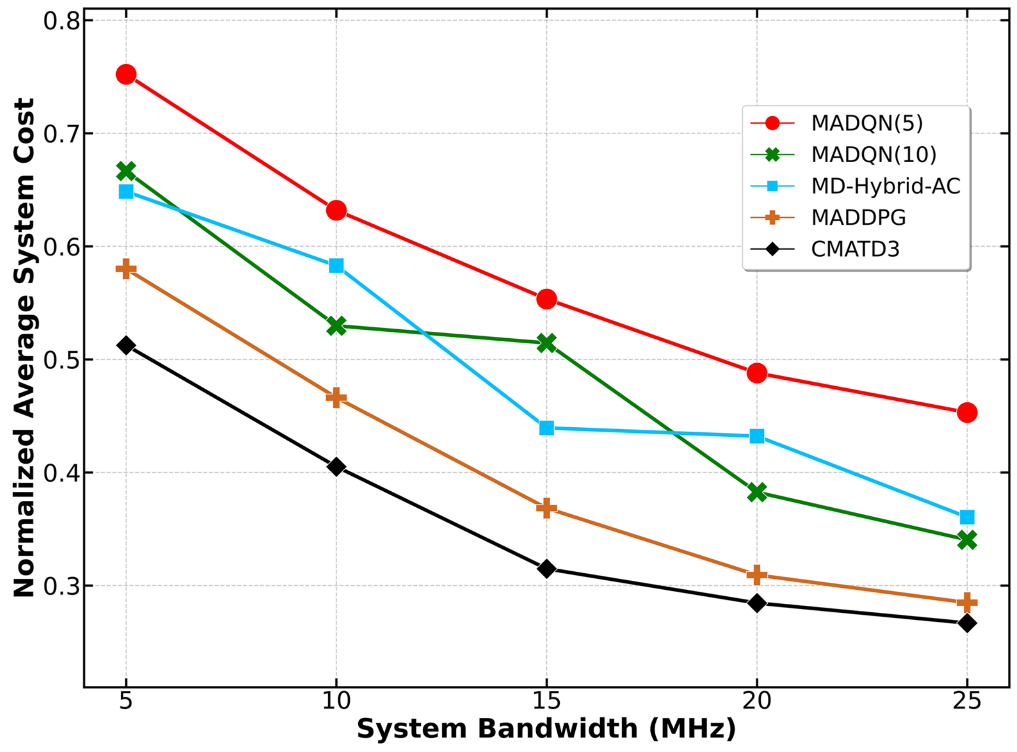


(b)

Fig. 8 Results of normalized average system cost vs. d_m^t. with **a** the number of cloud server $M = 2$ and **b** the number of cloud server $M = 3$



(a)



(b)

Fig. 9 Results of normalized average system cost vs. W with **a** the number of cloud server $M = 2$ and **b** the number of cloud server $M = 3$

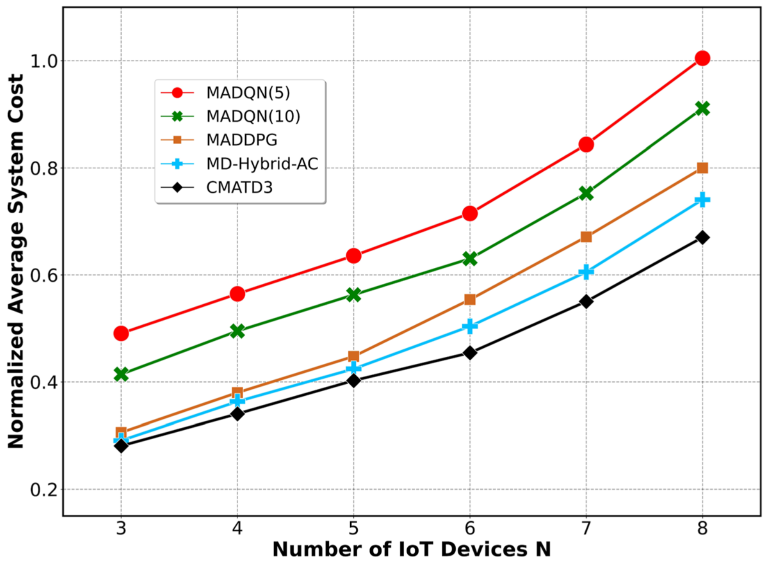


Fig. 10 Results of normalized average system cost vs. *N*

Table 1 Performance comparation with different number of IoT devices

Algorithms	3	4	5	6	7	8
ACO [10]	0.511	0.565	0.646	0.715	0.844	0.998
SPSO-GA [11]	0.309	0.391	0.483	0.571	0.718	0.821
CMATD3	0.281	0.340	0.402	0.454	0.550	0.670

lower total system cost can be achieved, which is shown in Fig. 8(a) and (b). Figure 8 shows the MADQN(5), MADQN(10) consistently have high system costs because their inflexible and naive behaviors. The MD-hybrid-AC agent has a comparable performance with MADDPG when the task data size is not heavy, and the performance deteriorates even more with the increase of the task data size. Besides, the CMATD3 agent outperforms MD-Hybrid-AC with lower system cost on average distribution range of d_n^t, which means that CMATD3 adapts to new learning task better due to the coordination among IoT device agents.

Performance comparison against system bandwidth

We evaluate the total system cost of five DRL-based agents with different system bandwidths in the scenarios of cloud server $M = 2$ and $M = 3$. In the Fig. 9, it can easily observe that as the system bandwidth W increases, the total system cost of DRL-based optimization methods goes down. The is because the transmission rate for IoT device n to cloud server m gradually goes up, which results in low transmission energy consumption. Then, the total system cost decreases in each DRL-based agent with different number of cloud servers. Clearly, compared with the case of cloud server $M = 2$ in Fig. 9(a), more cloud server will participate in computation offloading with the number of cloud server $M = 3$ in Fig. 9(b), which contributes to lower total system cost in five DRL-based optimization methods. Obviously, the results show that the CMATD3 decreases gradually and still maintains a lowest system cost among other schemes when the system bandwidth increases. This is because CMATD3 makes better decisions on server selection, offloading ratio and local computation capacity , compared with MADQN(5), MADQN(10), MADDPG, and Hybrid-AC agents.

Performance comparison against number of IoT devices

In multi-cloud environments, coordination among IoT devices is more challenging since the number of IoT devices may change with some one leaves or arrives. Therefore, to further analyze the scalability of the five DRL-based agents, we discuss the impact of N on the total system cost. Besides, for the sake of simplicity, all IoT devices are assumed to randomly scattered between 500m and 1000m to the BS, the number of cloud server is set to 3.

Figure 10 shows the performance of the five DRL-based computation offloading schemes with different *N*. In the Fig. 10, as the number of IoT devices increases, the average cost of each agent gradually grows up. The reason is explained below. A larger *N* leads to a higher probability that more IoT devices communicate with the cloud servers at the same time, resulting into more severe interference among IoT devices. In this case, it takes more energy consumption to transmit a given amount of data, which leads to more average normalized system cost during the uplink data transmission process.

One can see clearly that the performance of proposed CMATD3 agent significantly better than the other four DRL based agents. Then, in the case of $N < 6$, the total system cost of MADDPG agent is closer to both of CMATD3 and MD-Hybrid-AC agents, this is because the task data incurred on each IoT device is uniformly distributed. The performances of MADQN(5) and MADQN(10) do not exhibit well since the searching space of them is extremely large as the number of IoT devices increases and thus resulting in a serious performance degradation. Compared with MADQN(5), MADQN(10) improves the performance slightly with the increase of quantized levels, but far lower than the proposed CMATD3. The reason is that the quantization process induces quantization noise, which loses many features of action and impedes MADQN to find the optimal policy. Besides, MD-Hybrid-AC has less performance degradation than CMATD3 under different *N* since the MD-Hybrid-AC cannot efficiently adapt to the states of network scale.

Table 1 is the performance comparation of CMATD3 with heuristic algorithms, including ACO [11] and SPSO-GA [12], under different number of IoT devices. Obviously, STDPG outperforms ACO algorithm and SPSO-GA algorithm as it always obtains the smallest normalized system costs. For instance, when the number of IoT devices $\mathrm{N} = 8$, The normalized system costs of our proposed CMATD3 is 0.67 as against 0.998 and 0.821for ACO and SPSO-GA. The following explains why. The CMATD3 algorithm takes advantage of centralized training and distributed executing. For ACO and SPSO-GA algorithms, both of them normally need a considerable amount of iterations to achieve a near optimum. As the number of IoT devices increasing, the search action grows exponentially, they may easily fall into local optimum during optimal processing.

Conclusions and future work

This paper investigated the dynamic computation offloading problem in a hybrid-decision-based collaborative multi-cloud computing network, in which the time-varying computing requirements, wireless channel gains and network scale are comprehensively considered. The optimization problem was formulated to obtain the minimum long-term average total system cost of energy consumption of IoT devices and renting charge of cloud servers. To solve the issues of hybrid decision and collaboration among different IoT devices, we addressed the issues by two steps. Specifically, we first relaxed discrete action (e.g. cloud server selection) into a continuous set by designing a probabilistic method. Then, a cooperative multi-agent DRL (CMADRL) based framework with each IoT device acting as an agent, was developed to obtain the optimal cloud server selection, offloading ratio and local computation capacity. Experimental results have been performed to verify the effectiveness and superiority of the proposed CMADRL based framework over the other four state of the art DRL-based frameworks.

For our future work, we will consider to establish edge-cloud computing network system to execute computing tasks collaboratively. Moreover, we will study how the computation complexity and communication overhead of the training process are reasonably decreased, we will try to task advantage of federated learning based DRL, which only requires BS agents to share their model parameters instead of local training data.

Abbreviations
DRL: Deep reinforcement learning; IoT: Internet-of-Things; BS: Base station; CMADRL: Cooperative multi-agent deep reinforcement learning; DDPG: Deep deterministic policy gradient; TD3: Twin delayed DDPG; MATD3: Multi-agent TD3.

Acknowledgements
The authors would like to thank all the staf and students of school of computer and software engineering in Xihua university for contribution during this research process.

Authors' contributions
Problem formulation: Juan Chen, Peng Chen. The proposed algorithm: Peng Chen, Xianhua Niu. Computer simulations: Ling Xiong, Canghong Shi. Article preparation: Juan Chen, Zongling Wu. The authors read and approved the final manuscript.

Funding
The work of this paper is supported by the National Science Foundation of China (No. 62171387).

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

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Computer and Software Engineering, Xihua University, Chengdu, China. [2]School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China.

Received: 28 July 2022 Accepted: 19 November 2022
Published online: 08 December 2022

References

1. Gai K, Guo J, Zhu L, Yu S (2020) Blockchain meets cloud computing: a survey. IEEE Commun Surv Tutorials 22(3):2009–2030
2. Li K, Zhao J, Hu J et al (2022) Dynamic energy efficient task offloading and resource allocation for noma-enabled iot in smart buildings and environment. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109513
3. Chen C, Zeng Y, Li H, Liu Y, Wan S (2022) A multi-hop task offloading decision model in mec-enabled internet of vehicles. IEEE Internet Things J: 1. https://doi.org/10.1109/JIOT.2022.3143529
4. Chen Y, Zhao F, Lu Y, Chen X () Dynamic task offloading for mobile edge computing with hybrid energy supply. Tsinghua Sci Technol https://doi.org/10.26599/TST.2021.9010050
5. Chen Y, Xing H, Ma Z, et al (2022) Cost-efficient edge caching for noma-enabled iot services. China Communications
6. Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I et al (2010) A view of cloud computing. Commun ACM 53(4):50–58
7. Dinh HT, Lee C, Niyato D, Wang P (2013) A survey of mobile cloud computing: architecture, applications, and approaches. Wirel Commun Mob Comput 13(18):1587–1611
8. Huang J, Tong Z, Feng Z (2022) Geographical poi recommendation for internet of things: A federated learning approach using matrix factorization. Int J Commun Syst e5161 https://doi.org/10.1002/dac.5161
9. Apostolopoulos PA, Fragkos G, Tsiropoulou EE, Papavassiliou S (2021) Data offloading in uav-assisted multi-access edge computing systems under resource uncertainty. IEEE Trans Mob Comput: 1. https://doi.org/10.1109/TMC.2021.3069911
10. Tang X (2021) Reliability-aware cost-efficient scientific workflows scheduling strategy on multi-cloud systems. IEEE Trans Cloud Comput: 1. https://doi.org/10.1109/TCC.2021.3057422
11. Addya SK, Satpathy A, Ghosh BC, Chakraborty S, Ghosh SK, Das SK (2021) CoMCLOUD: Virtual machine coalition for multi-tier applications over multi-cloud environments. IEEE Trans Cloud Comput: 1. https://doi.org/10.1109/TCC.2021.3122445
12. Chen X, Zhang J, Lin B, Chen Z, Wolter K, Min G (2022) Energy-efficient offloading for dnn-based smart iot systems in cloud-edge environments. IEEE Trans Parallel Distrib Syst 33(3):683–697. https://doi.org/10.1109/TPDS.2021.3100298
13. Chen Y, Zhao J, Wu Y (2022) QoE-aware decentralized task offloading and resource allocation for end-edge-cloud systems: A game-theoretical approach. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2022.3223119
14. Xu J, Li D, Gu W et al (2022) Uav-assisted task offloading for iot in smart buildings and environment via deep reinforcement learning. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109218
15. Tuli S, Ilager S, Ramamohanarao K, Buyya R (2020) Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks. IEEE Trans Mob Comput 21(3):940–954. https://doi.org/10.1109/TMC.2020.3017079
16. Zhang Y, Di B, Zheng Z, Lin J, Song L (2020) Distributed multi-cloud multi-access edge computing by multi-agent reinforcement learning. IEEE Trans Wirel Commun 20(4):2565–2578
17. Chen Z, Hu J, Min G, Luo C, El-Ghazawi T (2022) Adaptive and efficient resource allocation in cloud datacenters using actor-critic deep reinforcement learning. IEEE Trans Parallel Distrib Syst 33(8):1911–1923. https://doi.org/10.1109/TPDS.2021.3132422
18. Zhou P, Wu G, Alzahrani B, Barnawi A, Alhindi A, Chen M (2021) Reinforcement learning for task placement in collaborative cloud-edge computing. In: 2021 IEEE Global Communications Conference (GLOBECOM). IEEE, Madrid, pp 1–6
19. Qu G, Wu H, Li R, Jiao P (2021) Dmro: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing. IEEE Trans Netw Serv Manag 18(3):3448–3459
20. Chen L, Xu Y, Lu Z, Wu J, Gai K, Hung PC, Qiu M (2020) Iot microservice deployment in edge-cloud hybrid environment using reinforcement learning. IEEE Internet Things J 8(16):12610–12622
21. Chen Y, Sun Y, Wang C, Taleb T (2022) Dynamic task allocation and service migration in edge-cloud iot system based on deep reinforcement learning. IEEE Internet Things J 9(18):16742–16757. https://doi.org/10.1109/JIOT.2022.3164441
22. Zhang J, Du J, Shen Y, Wang J (2020) Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach. IEEE Internet Things J 7(10):9303–9317
23. Oroojlooyjadid A, Hajinezhad D (2019) A review of cooperative multi-agent deep reinforcement learning. https://doi.org/10.48550/arXiv.1908.03963
24. Muñoz O, Pascual-Iserte A, Vidal J (2015) Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. IEEE Trans Veh Technol 64(10):4738–4755. https://doi.org/10.1109/TVT.2014.2372852
25. Chen Y, Zhao F, Chen X, Wu Y (2022) Efficient multi-vehicle task offloading for mobile edge computing in 6g networks. IEEE Trans Veh Technol 71(5):4584–4595. https://doi.org/10.1109/TVT.2021.3133586
26. Chen J, Wu Z (2021) Dynamic computation offloading with energy harvesting devices: A graph-based deep reinforcement learning approach. IEEE Commun Lett 25(9):2968–2972. https://doi.org/10.1109/LCOMM.2021.3094842
27. Chen J, Xing H, Xiao Z, Xu L, Tao T (2021) A drl agent for jointly optimizing computation offloading and resource allocation in mec. IEEE Internet Things J 8(24):17508–17524. https://doi.org/10.1109/JIOT.2021.3081694
28. Chen C, Jiang J, Zhou Y, Lv N, Liang X, Wan S (2022) An edge intelligence empowered flooding process prediction using internet of things in smart city. J Parallel Distrib Comput 165:66–78
29. Huang J, Gao H, Wan S et al (2023) Aoi-aware energy control and computation offloading for industrial iot. Futur Gener Comput Syst 139:29–37
30. Chen C, Li H, Li H, Fu R, Liu Y, Wan S (2022) Efficiency and fairness oriented dynamic task offloading in internet of vehicles. IEEE Trans Green Commun Netw
31. Lowe R, Wu Y, Tamar A, Harb J (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. https://doi.org/10.48550/arXiv.1706.02275
32. Fujimoto S, Hoof HV, Meger D (2018) Addressing function approximation error in actor-critic methods. https://doi.org/10.48550/arXiv.1802.09477
33. Chen Y, Gu W, Xu J, et al (2022a) Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning. China Commun
34. Chen Z, Zhang L, Pei Y, Jiang C, Yin L (2022) Noma-based multi-user mobile edge computation offloading via cooperative multi-agent deep reinforcement learning. IEEE Trans Cogn Commun Netw 8(1):350–364. https://doi.org/10.1109/TCCN.2021.3093436

Publisher's Note