Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH **Open Access**

COCAM: a cooperative video edge caching and multicasting approach based on multi-agent deep reinforcement learning in multi-clouds environment

Ruohan Shi[1], Qilin Fan[1*], Shu Fu[2], Xu Zhang[3], Xiuhua Li[1,4] and Meng Chen[1]

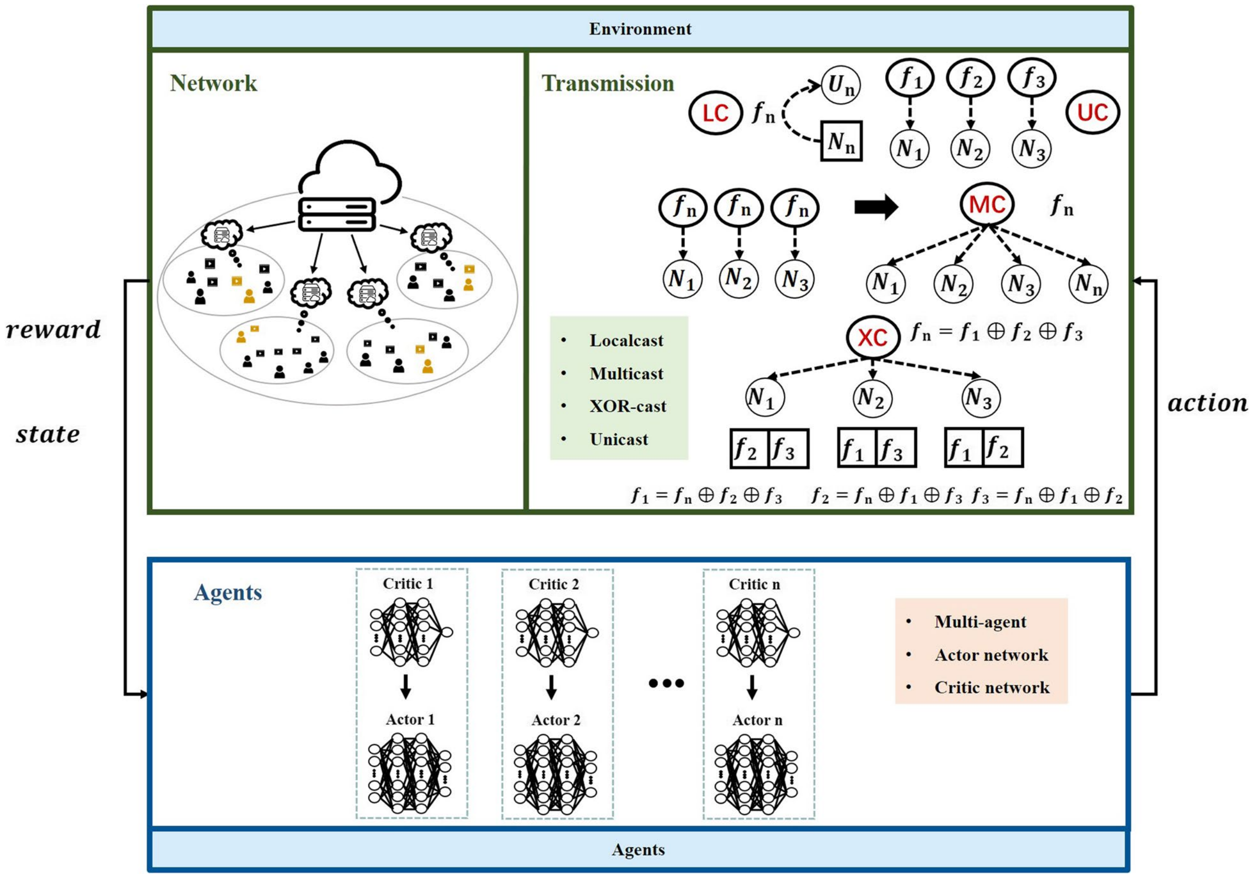
Abstract

The evolution of the Internet of Things technology (IoT) has boosted the drastic increase in network traffic demand. Caching and multicasting in the multi-clouds scenario are effective approaches to alleviate the backhaul burden of networks and reduce service latency. However, existing works do not jointly exploit the advantages of these two approaches. In this paper, we propose COCAM, a cooperative video edge caching and multicasting approach based on multi-agent deep reinforcement learning to minimize the transmission number in the multi-clouds scenario with limited storage capacity in each edge cloud. Specifically, by integrating a cooperative transmission model with the caching model, we provide a concrete formulation of the joint problem. Then, we cast this decision-making problem as a multi-agent extension of the Markov decision process and propose a multi-agent actor-critic algorithm in which each agent learns a local caching strategy and further encompasses the observations of neighboring agents as constituents of the overall state. Finally, to validate the COCAM algorithm, we conduct extensive experiments on a real-world dataset. The results show that our proposed algorithm outperforms other baseline algorithms in terms of the number of video transmissions.

Keywords Multi-clouds, Edge caching, Multicasting, Deep reinforcement learning

*Correspondence:
Qilin Fan
fanqilin@cqu.edu.cn
Full list of author information is available at the end of the article

Graphical Abstract



Introduction

As the Internet of Things (IoT) technology evolves, users are becoming increasingly interconnected with their electronic devices [1]. The advent of new wireless networks such as the fifth-generation (5G) network, the proliferation of smart devices, and users' high usage of diverse applications such as video streaming, online gaming, and virtual reality have resulted in a profound surge in video traffic. According to the Cisco report [2], studies predicted that the traffic of video types would account for 79% of all Internet traffic worldwide by 2022. The extensive prevalence of video traffic and the stringent quality of experience (QoE) requirements have put tremendous backhaul pressure on networks [3]. Therefore, the issue of minimizing the network resource consumption during transmission while simultaneously satisfying user demand has become one of the most critical concerns of network operators [4].

In traditional cloud environments, the service process requires moving data to remote data centers for centralized computing and storage. This leads to high network transmission latency, which can negatively impact the performance of mobile applications. To address this problem and provide reliable services for latency-sensitive applications, researchers have explored deploying small-scale cloud servers at the edge so that these edge cloud servers can provide resources closer to edge IoT devices [5–7]. Edge cloud servers are equipped with finite resources and can be utilized to deliver bandwidth-optimized services at the edge, thus enabling the provision of fast and immediate services [8, 9]. The multi-clouds architecture, including the remote cloud and edge clouds, is a promising paradigm to improve the QoE of users and reduce energy consumption [10, 11]. This potential stems from its ability to facilitate ubiquitous caching and efficient content delivery for end users, as highlighted by several studies.

During the content request phase, the network engages in content searching upon receiving a user's request. To alleviate traffic congestion, edge caching is an efficient manner of caching popular files on edge cloud servers closer to their requesters. It tackles the problem of which content to be cached in the edge cloud [12]. Recent scholarly investigations have substantiated the effectiveness of collaborative caching, which has attracted considerable scholarly attention. Collaborative caching works by allowing edge clouds to collectively distribute content through internal connections. Song et al. [13] presented an adaptive cooperative caching scheme that incorporates an enhanced quantum genetic algorithm to address the energy-delay tradeoff problem. Zhang et al. [14] proposed a spatially cooperative caching strategy for a two-tier heterogeneous network. The objective of this strategy is to minimize storage usage for duplicated content with caching while maximizing the likelihood of successful content retrieval (hit probability).

During the content delivery phase, traditional unicast mechanisms for distributing content from remote cloud to edge clouds and user ends (UEs) result in inefficient delivery. Multicasting, on the other hand, can leverage the available network bandwidth to deliver the same content to multiple receivers, benefiting from the similarity of users' preferences for content in close geographic locations. This mechanism reduces traffic generated during delivery by delivering the requested file through a single multicast rather than multiple unicasts [15]. Significant efforts have been devoted to video coding and multicast transmission [16–19]. For instance, Guo et al. [18] proposed a layer-based multi-quality multicast beamforming scheme based on scalable video coding. Wu et al. [19] designed an adaptive video streaming scheme using named data network multicast. However, these algorithms, while addressing video coding and multicast transmission, did not consider the integration of coded multicasting with caching in a cooperative environment.

Intuitively, Caching reduces latency and network bandwidth consumption by serving frequently requested content locally at the edge clouds [10, 20]. Multicasting further reduces bandwidth usage by efficiently delivering popular content to multiple users simultaneously, especially in scenarios with concurrent requests for the same content. Joint consideration of caching and multicasting can enhance the overall network performance and resource utilization by dynamically allocating caching and multicasting resources based on real-time user demand and network conditions. This adaptive strategy optimizes the content availability and delivery efficiency, leading to an improved user experience. Notably, it facilitates the deployment of various latency-sensitive applications and services [21, 22]. In the context of large-scale cache-enabled wireless networks, Jiang et al. [23] applied an iterative numerical algorithm to analyze and optimize caching and multicasting. Various coding multicasting mechanisms have been proposed in different scenarios [24–27]. Nevertheless, in large-scale cooperative caching scenarios, finding a balance between edge caching and multicasting to improve resource efficiency remains a challenging task.

In this paper, we exploit the benefits of mobile edge caching with multicasting in the multi-clouds environment to reduce network transmission consumption. We investigate the collaborative caching among different edge clouds to effectively adapt to dynamic edge environments. We propose a multi-agent DRL-based approach for COoperative video CAching and Multicasting named COCAM to minimize the average transmission number, thereby enhancing video delivery efficiency. Our main contributions are summarized as follows:

- We investigate the cooperative video edge caching and multicasting issue to reduce the transmission number in the multi-clouds scenario. Moreover, we present the problem formulation as a multi-agent Markov decision process (MDP).
- A novel multi-agent actor-critic algorithm is designed to address the formulated MDP. Specifically, each agent learns a local caching strategy and further encompasses the observations of neighboring agents as constituents of the overall state. Multiple agents work in collaboration to efficiently adapt to the dynamic network environment.
- Extensive trace-driven simulations demonstrate that our proposed algorithm outperforms other baselines in terms of video transmission number.

The rest of this paper is organized as follows. In Related work section, we introduce the related works. System model and problem formulation section presents the system model and problem formulation. The details of the COCAM approach are presented in The COCAM approach section. We compare the experimental performance and analyze the results in Performance evaluation section. Conclusion section concludes the paper.

Related work

Caching algorithms

Edge caching stores popular content locally on edge clouds, allowing them to deliver the requested content directly to users. It significantly reduces network latency and network consumption. Li et al. [28] investigated a cost-effective greedy algorithm with consideration for different video characteristics. It optimized the mobile edge cache placement problem for QoE-driven dynamic adaptive video streams. Tran et al. [29] proposed a federated collaborative caching and processing framework based on integer linear programming to accommodate adaptive bitrate video streams in mobile edge computing networks. The caching decision process in wireless communication networks can be represented as an MDP, and reinforcement learning has been commonly employed in this domain. Based on a multi-agent framework, Wang et al. [30] proposed a deep actor-critic reinforcement learning algorithm to address the dynamic control of caching decisions by enabling each edge to learn an optimal policy through self-adaptation. However, the existing research primarily focused on content caching policies and did not incorporate consideration for the content delivery process.

Multicasting algorithms

Multicast transmission is extensively utilized in edge networks, demonstrating its efficacy in enhancing network performance by reducing bandwidth, routing, and cost [31]. Damera et al. [32] constructed a new feasible architectural model to transmit the required content to the user using the multicell transmission. The Signal Noise Ratio was improved using the multicell transmission. The optimized MEC scheduling algorithm showed better performance compared to the existing model. Zahoor et al. [33] proposed a suggested enhanced eMBMS network architecture to address the significant limitations of the standard eMBMS architecture, i.e., a network architecture using network function virtualization (NFV) and MEC. The proposed architecture allows the multicasting of crowdsourcing live streams. Ren et al. [15] considered the fundamental issues of NFV-enabled multicast in mobile edge clouds and designed a heuristic algorithm. Qin et al. [34] studied the multicast traffic for IoT applications in edge networks under the delay-oriented network slicing problem. Nevertheless, these works focused on the network architecture and multicast protocols without integration with the practical applications of the edge cloud servers.

Joint caching and multicasting algorithms

The utilization of multicast transmission at the base station, enabling concurrent servicing of distinct user requests for the identical file, is recognized as a highly efficacious approach for supporting the delivery of extensive content over wireless networks. This approach is regarded as an effective strategy in wireless communications to meet the constantly increasing demand for content transmission. Maddah-Ali et al. [35] used the joint encoding of multiple files and the multicasting feature of downlink channels to optimize content placement and delivery under encoded multicast. They also evaluated the caching gain and demonstrated that the joint optimization problem could improve the caching gain. Liao et al. [36] used the benefits of multicast content delivery and collaborative content sharing jointly to develop a compound caching technique (multicast-aware cooperative caching). He et al. [37] designed partial caching bulk transmission and partial caching pipelined transmission to reduce the delivery latency of cache-enabled multi-group multicast networks. Somuyiwa et al. [38] combined active caching and multicast transmission to model the single-user multi-request problem as an MDP and used a DRL approach to solve the problem. Since traditional approaches are difficult to adapt to this highly diverse and dynamic environment under multi-clouds cooperative caching, we propose a COCAM-based framework to maximize the traffic consumption during the video delivery phase.

System model and problem formulation

In this section, we introduce the cooperative video edge caching and multicasting model and give concrete definitions. Then, we state the corresponding cache decision-making problem. For convenience, we summarize some key modeling parameters and notations in Table 1.

Network model

We consider the multi-clouds system, which consists of three types of layers: the remote cloud layer, the edge cloud layer, and the UE layer. Assuming that the remote cloud provides all the requested video files $\mathcal{F} = \{1, 2, \cdots, F\}$. Since video service generally fragments a video into equally sized chunks, we assume all files are unit-sized. The set of edge cloud servers can be denoted as $\mathcal{N} = \{1, 2, \cdots, N\}$. We denote the time slot of requests as $\mathcal{T} = \{1, 2, \cdots, T\}$. The edge clouds receive the requests and make the caching decision at each time slot t. At each time slot t, the edge clouds receive requests and determine caching decisions. The request received by edge cloud n for file f is denoted as $q_{t,n}^{f} \in \{0, 1\}$, where $q_{t,n}^{f} = 1$ represents a request for file f, and $q_{t,n}^{f} = 0$ signifies no request for file f. A variable $x_{t,n}^{f}$ is used to denote the transmission decision, i.e., whether the requested video f is transmitted from the remote cloud to the edge cloud n at time t. If no, we have $x_{t,n}^{f} = 0$, and $x_{t,n}^{f} \in (0, 1]$ otherwise. $x_{t,n}^{f} = 1$ means a transmission channel is fully used by edge cloud n and it occurs only under unicast

Table 1 Summary of important notations

Notations	Definition
β	The hyperparameter of the entropy term
γ	The discount factor
$N, \mathcal{N}$	The number and set of edge clouds
G	The video set through the XC scheme
$F, \mathcal{F}$	The number and set of videos
$x_{t,n}^{f}$	The variable whether the requested video *f* is transmitted from the remote cloud to edge cloud *n* at time *t*
$y_{t,n}^{f}$	The variable whether the requested video *f* has been stored in edge cloud *n* at time *t*
$q_{t,n}^{f}$	The request for file *f* received by the edge cloud *n* at time *t*
C	The maximum capacity of edge cloud
$s_{t,n}$	The state of agent *n* at time *t*
$\hat{s}_{t,n}$	The joint observation state of an agent *n*
$\pi_{t,n}$	The policy of agent *n*
$a_{t,n}$	The action of agent *n*
ω_n	The parameter of the critic network for agent *n*
θ_n	The parameter of the actor network for agent *n*
$R_{t,n}$	The expected value equation for edge cloud *n*
r_n	The global reward
B	The replay buffer memory
ζ	The target network update parameter *n*
V	The value function of the critic network
$\tilde{A}_{t,n}$	The advantage function
$\mathcal{N}_n$	The neighborhoods set of agent *n*

conditions. Otherwise, if multiple edge cloud servers share a channel under one of the multicast conditions, we assume these edge clouds share the channel equally.

Caching model

At each time t, we assume that only one UE under the edge cloud server n will request the video. For each UE, if the requested video has been cached in the upper edge cloud, the edge cloud server can deliver it to the UE directly. Else the edge cloud server requests the file from the remote cloud.

Each edge cloud has the same maximum capacity C. We use a binary variable $y_{t,n}^{f}$ to indicate whether the requested video f has been stored in edge cloud n at time t. If yes, we have $y_{t,n}^{f} = 1$, and 0 otherwise. Each server stores content limited to its maximum storage capacity:

$$\sum_{f \in \mathcal{F}} y_{t,n}^{f} \leq C. \tag{1}$$

After the edge cloud gets the requested video, the edge cloud will decide whether to cache the content or not. If the edge cloud storage capacity is not fully filled, we store the video directly. Otherwise, we update our caching space based on the policy.

Transmission model

The remote cloud delivers the videos to the requested edge clouds. Figure 1 gives four schemes in our cooperative transmission model which are described as follows:

- **Localcast (LC)**: If the requested video has been cached in the local edge cloud server at time t, the UE can fetch it from the edge cloud directly without requesting from the remote cloud. We use $\mathcal{N}_{LC} = \{n | y_{t,n}^{f} = 1, \forall n \in \mathcal{N}, \forall f \in \mathcal{F}\}$ to denote the set of edge clouds from which UEs can get videos at time t through LC schema without fetching from the remote cloud. We have:

$$x_{t,n}^{f} = 0, \forall n \in \mathcal{N}_{LC}, \tag{2}$$

as shown in the LC part of Fig. 1, N_1 requests f_1, f_1 has been stored in N_1.
- **Multicast (MC)**: If the requested video has not been cached in the edge cloud, then the edge cloud requests the file from the remote cloud. If there are other different edge clouds requesting the same video f at time t, then these edge clouds can obtain the requested video f through MC schema. We use

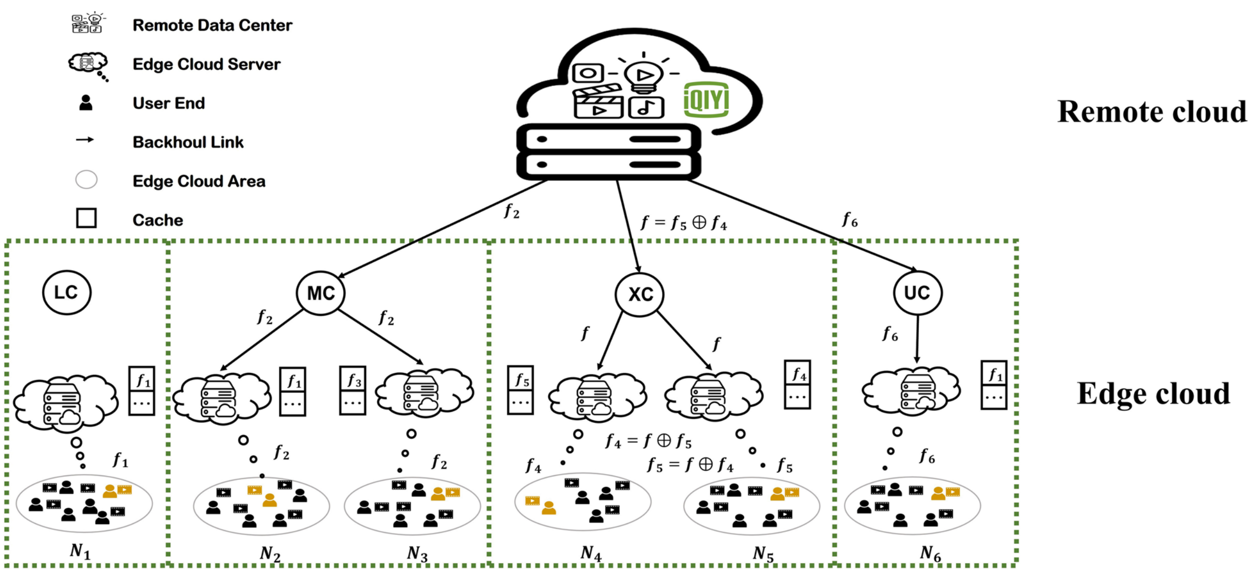


Fig. 1 System model

the $\mathcal{N}_{MC}^{f} = \{n | q_{t,n}^{f} = 1, y_{t,n}^{f} = 0, \forall n \in \mathcal{N} \setminus \mathcal{N}_{LC}\}$ to denote the set of edge clouds that can use multicast transmission to obtain the requested video *f*. We have:

$$\sum_{n \in \mathcal{N}_{MC}^{f}} x_{t,n}^{f} = 1, \tag{3}$$

as shown in the MC part of Fig. 1, N_2 and N_3 simultaneously request f_2 that have not been cached.

- **XOR-cast (XC)**: We form a special edge cloud set named exclusive OR (XOR) set where each edge cloud in the set stores the video files requested by the other edge clouds. We denote this set as:

$$\mathcal{N}_{XC}^{G} = \Big\{ n \mid q_{t,n}^{f} = 1, y_{t,n}^{f} = 0, y_{t,n'}^{f} = 1, \forall n \in \mathcal{N} \setminus \left(\mathcal{N}_{LC} \cup \mathcal{N}_{MC}^{f} \right), \forall n' \in \mathcal{N}_{XC}^{G} \setminus n, \forall f \in G \Big\}, \tag{4}$$

where the video set through the XC scheme can be denoted as:

$$G = \Big\{ f \mid q_{t,n}^{f} = 1, y_{t,n}^{f} = 0, y_{t,n}^{f'} = 1, \forall n \in \mathcal{N} \setminus \left(\mathcal{N}_{LC} \cup \mathcal{N}_{MC}^{f} \right), \forall f' \in G \setminus f \Big\}. \tag{5}$$

The XOR set receives the XOR-encoded bit stream by one transmission. Then, each edge cloud restores its video by decoding the received bit stream with the contents stored in its cache. We have:

$$\sum_{n \in \mathcal{N}_{XC}^{F}} \sum_{f \in F} x_{t,n}^{f} = 1, \tag{6}$$

as shown in the XC part of Fig. 1, N_4 and N_5 simultaneously request f_5 and f_4 that have been cached not by themselves but by each other. We denote the coded XOR information as *f*. If there are multiple XC combinations, we choose the combination that will generate the smallest number of XC sets with the participation of the same number of edge clouds. This preference is based on the effectiveness of our proposed XC approach in significantly reducing internal energy consumption during unicast transmission. While this paper does not explicitly consider the energy consumption associated with XOR operations, it is important to acknowledge that such operations still entail a non-negligible energy overhead. Considering a fixed number of edge clouds, our objective is to minimize the number of XC combinations to mitigate the impact of XOR energy consumption.

- **Unicast (UC)**: When the relationship between the requests from edge clouds and the cache list does not satisfy any of the above cases, edge clouds fetch videos directly from the remote cloud by establishing a transmission channel. We denote the UC set as $\mathcal{N}_{UC} = \Big\{ N \setminus \left(\mathcal{N}_{LC} \cup \mathcal{N}_{MC}^{f} \cup \mathcal{N}_{XC}^{F} \right) \Big\}$. We have:

$$x_{t,n}^{f} = 1, \forall n \in \mathcal{N}_{UC}. \tag{7}$$

as shown in the UC part of Fig. 1, the edge cloud gets content from the remote cloud.

To use fewer transmissions to deliver all the data during the delivery process, we use the network coding technique. The transmitted content is encoded at the network nodes and then decoded at the destination. We use XOR coding techniques. These edge clouds have not cached the requested video but have cached the video requested by other edge clouds. The caching policy determines what will be cached in the edge cloud, and then the remote cloud classifies the transmission based on the cache state in the edge clouds. According to the above four cases, we can formulate the joint multicast transmission and cache replacement problem that aims to minimize the total number of transmissions from the remote cloud to the edge cloud as:

$$\min \quad \sum_{n\in\mathcal{N}}\sum_{f\in\mathcal{F}}\sum_{t\in\mathcal{T}}\frac{q_{t,n}^{f}x_{t,n}^{f}}{N} \tag{8}$$

$$s.t. \quad (1), (2), (3), (6), (7) \tag{9}$$

$$0 \le x_{t,n}^{f} \le 1 - y_{t,n}^{f}, \tag{10}$$

$$y_{t,n}^{f} \in \{0,1\}. \tag{11}$$

The COCAM approach

Our modeling problem is a mixed integer programming (MIP) problem [22], which is strictly NP-hard. Solving MIP problems with traditional computational methods has been proven challenging in natural caching systems with low computational efficiency. Thus we consider a learning approach. We explore the collaboration between different edge cloud servers with a multi-agent reinforcement learning-based algorithm to better adapt to dynamic edge environments.

In this section, each edge cloud operates as an independent agent, while maintaining a cooperative relationship with other edge clouds. We model the cache decision-making problem as a multi-agent extension of the Markov Decision Process (MDP) and introduce a novel multi-agent actor-critic-based caching approach. Our proposed approach aims to minimize the average number of transmissions during the request transmission process. Multi-agent reinforcement learning consists of agents and the environment. Based on the state and the reward from the environment, each agent executes an action according to its certain strategy. Then the environment changes to a new state. An MDP is a mathematical framework for modeling sequential decision-making consisting of state, action, transition probability, and reward. Each agent learns the optimal decision-making sequence through continuous interaction with the environment. We define the basic elements of a multi-agent MDP as follows:

State

The state of agent n at time t be denoted as $s_{t,n} = \{y_{t,n}, q_{t,n}^{f}\}$, where $q_{t,n}^{f}$ indicates the current request demands and $y_{t,n} = \{y_{t,n}^{f}\}_{\forall f\in\mathcal{F}}$ denotes the caching state of edge cloud n. We define the neighborhoods that can be observed by the agent n as $\mathcal{N}_n$. We use $\pi_{t,n}$ to denote the policy of agent n. Thus, the adjacent agent policy of agent n can be denoted as $\pi_{t,\mathcal{N}_n}$. Each agent can observe the states and policies of the neighborhoods. Therefore, the joint state of an agent n to be fed into the input network is $\hat{s}_{t,n} = \{s_{t,m}\}_{\forall m\in\{n,\mathcal{N}_n\}}$.

Action

An agent decides which video should be replaced from the cache list based on its policy. We denote the action of agent n as $a_{t,n} = v$, where $v \in \{0, 1, 2, \cdots, C\}$. If $v = 0$, the requested video will not be cached. Else, the v-th content in the cache space of edge cloud n will be replaced by the current requested video.

Reward

The goal is to minimize the average transmission number. We define the negative value of the transmission number as the reward:

$$r_{t,n} = -\sum_{f\in\mathcal{F}}\frac{q_{t,n}^{f}x_{t,n}^{f}}{N}. \tag{12}$$

So the global reward is calculated as:

$$r_t = \sum_{n\in\mathcal{N}} r_{t,n}. \tag{13}$$

Network architecture

As shown in Fig. 2, each agent consists of two parts: actor network (as θ) and critic network (as ω). The actor network and critic network are essential components of a policy network. The actor network receives environmental states as input and generates corresponding action outputs, aiming to learn an optimal policy π_{θ_n} that maximizes the expected return or value function associated with accumulated rewards. On the other hand, the critic network serves as a value function estimation network, evaluating the quality of actions chosen by the actor network within a given state. Its primary objective is to learn

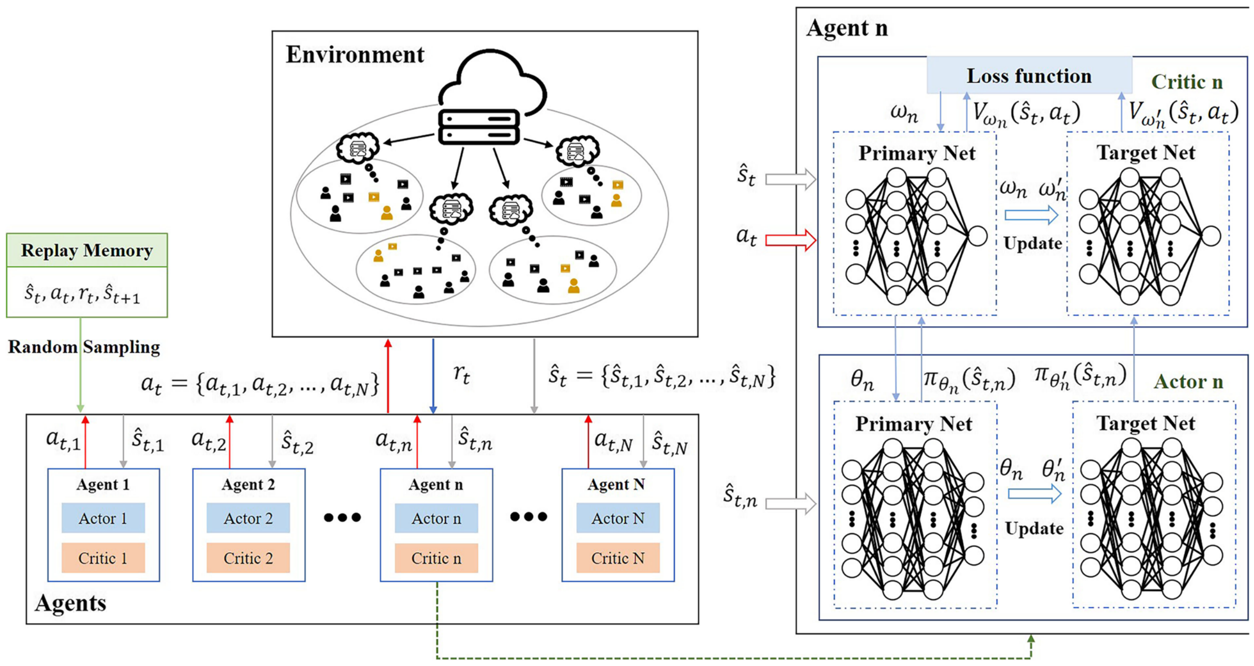


Fig. 2 The COCAM approach

a value function V_{ω_n} capable of estimating the expected return or value based on the current state and the actions selected by the actor network. The actor network consists of two fully connected hidden layers with ReLU activation functions, where the dimensions are determined by the variable state size of the cache. Its output layer is a fully connected layer utilizing a hyperbolic tangent (tanh) activation function. Similarly, the critic network shares the same architecture as the actor network, comprising two fully connected hidden layers with ReLU activation functions. The critic network's output layer consists of a single unit activated by a linear function. Each network contains a target network and a primary network of the same network structure. We use the target network to improve the stability and convergence of training. After the primary network learns a certain number of times, the parameters of the primary network are used to update the parameters of the target network.

Agents get the policies π based on their actor networks. The actor network is denoted as a function to seek optimal policy $\pi_{t,n} = \pi_{\theta_n}(a_{t,n}|\hat{s}_{t,n}, \pi_{t-1,\mathcal{N}_n})$, where θ_n denotes the actor network parameter of agent n. An agent gets the action by random sampling with the policy distribution. We denote the parameter of the critic network for agent n as ω_n. Thus, V_{ω_n} denotes the value function of the critic network trained as an estimate of the expected reward.

We formulate the expected value equation for edge cloud n as:

$$R_{t,n} = r_t + \gamma V'_{w_n}\left(\hat{s}_{t+1,n}, \pi_{t,\mathcal{N}_n}\right), \tag{14}$$

where γ denotes the discount reward factor. At each time t, the agent stores the experience $(\hat{s}_{t,n}, a_{t,n}, r_{t,n}, \hat{s}_{t+1,n})$ in replay memory B.

We use the temporal difference (TD) algorithm to update the critic network. The loss function of the critic network can be calculated as:

$$L(w_n) = \frac{1}{2|B|}\sum_t \left(R_{t,n} - V_{w_n}\left(\hat{s}_{t,n}, \pi_{t-1,\mathcal{N}_n}\right)\right)^2. \tag{15}$$

The actor network is updated by the policy gradient (PG) algorithm. The loss function of the actor network can be defined as:

$$L(\theta_n) = -\frac{1}{|B|}\sum_t (\log \pi_{\theta_n}\left(a_{t,n} \mid \hat{s}_{t,n}, \pi_{t-1,\mathcal{N}_n}\right)\tilde{A}_{t,n} - \beta \sum \pi_{\theta_n} \log \pi_{\theta_n}\left(a_{t,n} \mid \hat{s}_{t,n}, \pi_{t-1,\mathcal{N}_n}\right)), \tag{16}$$

where the β denotes a hyperparameter to control the entropy term, and the advantage function $\tilde{A}_{t,n} = R_{t,n} - V_{w_n}\left(\hat{s}_{t,n}, \pi_{t-1,\mathcal{N}_n}\right)$ is the discounted reward minus a baseline.

Then we update the target network parameters for each agent n as:

$$\theta'_n = \zeta\theta_n + (1-\zeta)\theta'_n, \tag{17}$$

$$\omega'_n = \zeta \omega_n + (1-\zeta)\omega'_n, \tag{18}$$

where ζ denotes the target network update parameter. The target network is updated every τ step.

After the training is completed, each agent can get the most effective action strategy in the current state according to its own state in each execution step.

```
1  /**Training Process**/
2  Initialize the actor parameters ${\theta_n}_{n \in N}$ and critic
     parameters ${\omega_n}_{n \in N}$.
3  Initialize the global environment for agents.
4  for $episode = 1 \rightarrow E_{max}$ do
5      Initialize the global environment.
6      for $timestep\ t = 1 \rightarrow T_{max}$ do
7          /**Environmental Interaction**/
8          for $agent\ n = 1 \rightarrow N$ do
9              Extract state $s_{t,n}$ of the agent from the
                 environment.
10             Construct the joint state $\hat{s}_{t,n}$ by incorporating
                 states from the agent's neighbors.
11             Random sample action $a_{t,n} = v$ through local
                 policy $\pi_{\theta_n}$.
12             Execute the action and update the caching
                 space.
13             Observe the new state $\hat{s}_{t+1,n}$ and calculate the
                 global reward $r_t$.
14             Store the $(\hat{s}_{t,n}, a_{t,n}, r_t, \hat{s}_{t+1,n})$ into replay
                 memory $B$.
15         end
16         /**Parameter Update**/
17         for $agent\ n = 1 \rightarrow N$ do
18             Sample a random batch of
                 $(\hat{s}_{t,n}, a_{t,n}, r_t, \hat{s}_{t+1,n})$ from replay memory $B$.
19             Update parameter $\omega_n$ of critic primary network
                 and the parameter $\theta_n$ of actor primary
                 network via (15)(16).
20         end
21         if $t \mod \tau == 0$ then
22             Update the target networks via (17)(18).
23         end
24     end
25 end
```

Algorithm 1 The COCAM AlgorithmThe COCAM algorithm is given in Algorithm 1. Each local agent collects the experience tuple by following the current policy until enough samples are collected for batch updating (lines 8 to 15). Then a batch will be sampled randomly to update the actor and the critic network (lines 17 to 20). For every τ step, the target network is updated (lines 21 to 23).

Performance evaluation

Experiment setup

We conduct experiments on a real-world dataset from iQIYI which contains 300,000 individual videos watched by 2 million users over two weeks. We randomly select 10,000 records from it. Figure 3 illustrates a descending order trend in video request preferences observed in the iQIYI dataset. The popularity distribution of videos exhibits notable skewness, adhering to a Zipf distribution. This implies that a small subset of highly popular videos significantly contributes to the majority of access volume, while a large number of other videos receive minimal attention. Popular videos are frequently accessed, necessitating regular updates to their cached content. Conversely, a substantial proportion of less popular videos are rarely accessed, rendering them ineffective for caching purposes. However, despite their limited popularity, these less popular videos still contribute to users' demand. Therefore, it becomes imperative to design an adaptive cooperative caching and multicasting strategy that captures the distribution and dynamics in video popularity. We divide the dataset into 30 edge areas based on geographic information with the K-means algorithm [39]. We select 20 to deploy edge cloud servers (i.e., agents) to provide the video service for users. By default, we set the cache size to 50. We assume that each agent can observe the states of all the other agents from the environment. The key experimental parameters are listed in Table 2.

Table 2 Simulation parameters

Parameter	Description	Value
β	Hyperparameter of the entropy term	0.01
γ	Discount factor	0.99
τ	Parameter of update time	64
ζ	Parameter of soft update	0.01
$\lvert B\rvert$	Batch size	64

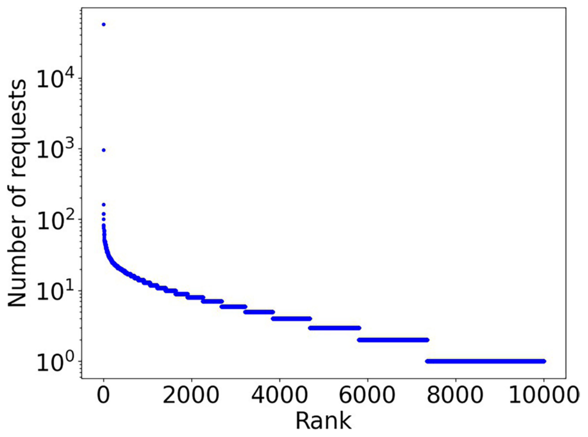


Fig. 3 Number of requests of a content versus its rank on iQIYI dataset

Comparisons and results

The contents in different edge cloud servers are related to each other in multicast delivery, leading to the tendency of multicast caches to store similar contents. In contrast, for cooperative caching, the cached contents in different units should be mutually exclusive for better utilization of the limited storage space. The combination is balanced by using multi-agent reinforcement learning in the combination.

Figure 4 shows the variation of transmission number of COCAM with the increasing training episode. We can see that the transmission number decreases linearly at the beginning and steadily converges at around 150 episodes.

According to Eq. (8), we measure the performance of our proposed algorithm using the average number of transmissions during the entire process of requesting. The average number of transmissions can show the efficiency of multicast transmission, which is affected by the caching decision. A lower average number of transmissions means fewer channel resources and higher multicast transmission efficiency for the same request, which can effectively relieve network transmission pressure.

To evaluate the performance of the COCAM algorithm, we compare it with other algorithms in different cases.

Comparison with non-cooperative caching algorithms

In Fig. 5, we compare the COCAM algorithm with non-cooperative caching algorithms under cooperative transmission in terms of the number of transmissions.

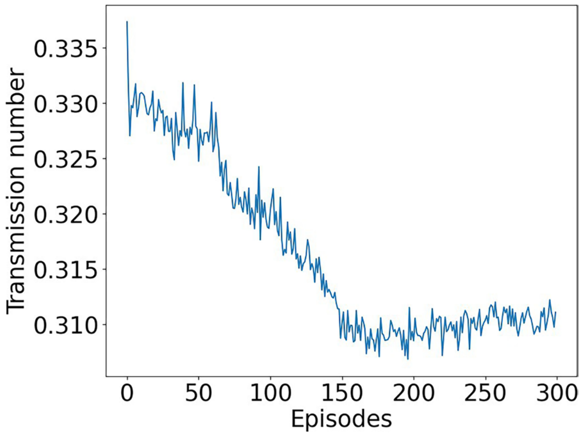


Fig. 4 The values of transmission number in the training process of COCAM

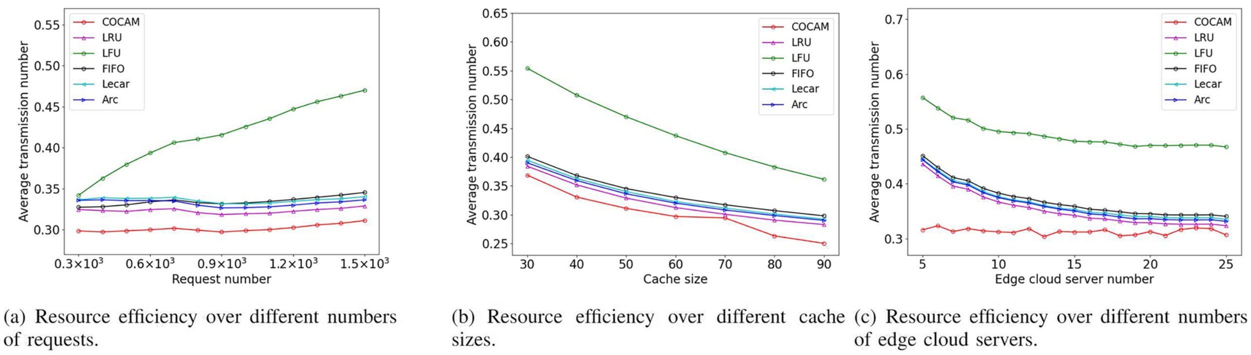
LRU [40]: The new content will replace the cached content which has been least recently requested. **LFU** [40]: The new content will replace the cached content which has been least frequently requested. **FIFO** [41]: The new content will replace the cached content which has been stored earliest. **Lecar** [42]: It adopts LRU or LFU algorithm to update the cache according to the weight adaption by regret minimization technique. **Arc** [43]: It dynamically adjusts the size of the two queues and performs cache updates based on the LRU algorithm.

In these caching algorithms, each edge cloud server individually caches the content based on its caching decision without combining the cooperative caching among the edge clouds.

Figure 5a shows the comparison result under different numbers of requests. The request numbers are set ranging from 300 to 1500. Our COCAM algorithm performs better than the other baselines, with an average improvement of 2% to 15% in global benefits. Besides, the variations in the number of requests hardly affect the performance except for the LRU algorithm. It is because LRU works better for popular content and tends to lead to cache pollution in smooth datasets. Figure 5b shows the performance comparison under different edge cloud cache sizes. The cache size ranges from 30 to 90.

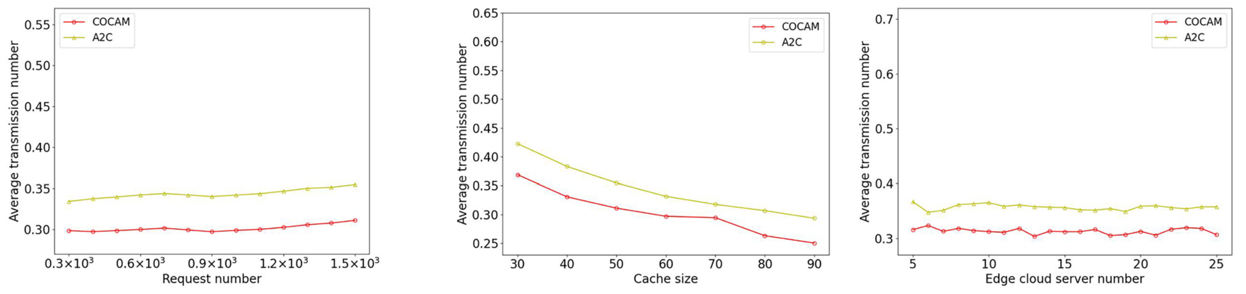
From Fig. 5b, it is observed that the transmission number decrease as the edge cloud cache sizes increase for all methods. Since the requested videos are more likely to be hit locally or built a multicast transmission as the cache capacity increases, our COCAM algorithm performs better than the other baselines.

Figure 5c shows the comparison under different amounts of edge clouds. We set the edge cloud server numbers ranging from 5 to 25 with a cache size of 50. We compare the results after 1500 requests. We can see that COCAM achieves the minimum transmission number. The performance of our algorithm is significantly better



(a) Resource efficiency over different numbers of requests. (b) Resource efficiency over different cache sizes. (c) Resource efficiency over different numbers of edge cloud servers.

Fig. 5 Performance comparison with non-cooperative caching algorithms



(a) Resource efficiency over different numbers of requests. (b) Resource efficiency over different cache sizes. (c) Resource efficiency over different numbers of edge cloud servers.

Fig. 6 Performance comparison with cooperative caching algorithms

than other algorithms and stabilizes when there are fewer edge clouds.

Comparison with cooperative caching algorithms

In Fig. 6, we compare the COCAM algorithm with A2C algorithms that apply cooperative caching under cooperative transmission. **A2C** [44]: This algorithm uses a single-agent advantage actor-critic algorithm to select the action with the best reward.

As seen in the figure, the two cooperative caching algorithm curves converge, with the COCAM significantly outperforming the A2C algorithm. Compared to the A2C algorithm, our proposed algorithm results in an average improvement of 4%. It is mainly because COCAM yields more intelligent decision-making that learns the dynamic request pattern based on the global state. The performance of the learning-based algorithm can adapt well to the multicast environment and is not significantly affected by the variations in the number of edge clouds. With the cooperation of different agents, COCAM shows better and more stable performance.

Comparison of algorithms with different multicasting schemes

Figure 7 shows the performance of multicast transmission and coding transmission during the delivery phase in the cooperative caching scenario. COCAM-w/o-MC &XC: We design the COCAM-w/o-MC &XC by using COCAM without using the part of MC and XC. COCAM-w/o-XC: We design the COCAM-w/o-XC by using COCAM without using the part of XC.

The experimental results illustrate that our proposed COCAM algorithm works better than the design-altered COCAM algorithms. It shows that our proposed MC and XC schemes effectively reduce the transmission number. As shown in the figure, the

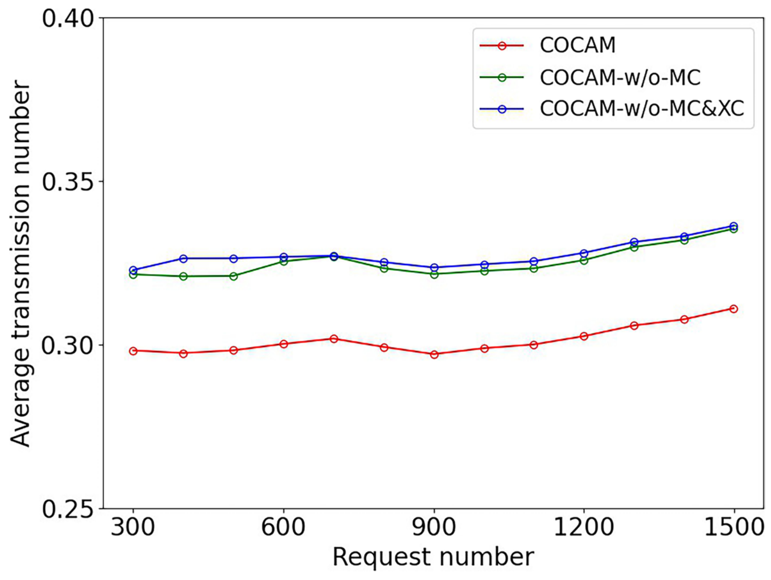


Fig. 7 Performance comparison in different parts

two altered algorithm curves are closer in results, indicating MC scheme is less effective on this dataset. This phenomenon can be attributed to the observation that users within the same region tend to have similar request preferences, while their activities of accessing the same content may vary across different time slots. It indicates that our XC scheme can effectively leverage this insight to achieve superior performance in the integrated caching and multicasting scenario.

Conclusion

In this paper, we have proposed a joint cache replacement and multicast transmission strategy in the multi-clouds scenario. This strategy could reduce the transmission number efficiently for video delivery. We have designed a multi-agent actor-critic algorithm named COCAM, enabling multiple edge clouds to cooperate to achieve intelligent caching decisions. In addition, we have conducted experiments on a real-world dataset. The evaluation results have shown that our COCAM algorithm could reduce the average transmission number by cooperation between different agents compared to other baselines. In our future work, we will further enhance reinforcement learning algorithms to achieve improved adaptation in resource-constrained and bandwidth-limited multi-clouds environment at a large scale.

Abbreviations

IoT	Internet of Things
QoE	Quality of experience
UE	User end
MEC	Mobile edge computing
NFV	Network function virtualization
MDP	Markov decision process
DRL	Deep reinforcement learning
MIP	Mixed integer programming
TD	Temporal difference
PG	Policy gradient

Acknowledgements
The authors would like to thank to anonymous reviewers for their valuable comments on the manuscript.

Authors' contributions

Ruohan Shi carried out the experiment and wrote the manuscript with support from Qilin Fan, Shu Fu, Xu Zhang, Xiuhua Li and Meng Chen. Qilin Fan supervised the whole project. All authors have read and agreed to the published version of the manuscript.

Funding
This work is supported in part by the National NSFC (Grant No. 62102053), the Natural Science Foundation of Chongqing, China (Grant No. CSTB2022NSC-QMSX1104), the General Program of Chongqing Science & Technology Commission (Grants No. CSTB2022TIAD-GPX0017 and CSTB2022TIAD-STX0006), the Science and Technology Plan Project of Chongqing Economic and Information Commission (Grant No. 2211R49R03), the Key Research Program of Chongqing Science & Technology Commission (Grant No. cstc2021jscx-dxwtBX0019), the Regional Innovation Cooperation Project of Sichuan Province (Grant No. 2023YFQ0028), the Regional Science and Technology Innovation Cooperation Project of Chengdu City (Grant No. 2023-YF11-00023-HZ), and the EU Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 898588, and the Chongqing Key Laboratory of Digital Cinema Art Theory and Technology (Grant No. 2021KF01). This article reflects only the authors' view. The European Union Commission is not responsible for any use that may be made of the information it contains.

Availability of data and materials

We are actively disclosing data with data providers due to signed confidentiality agreements. The motion model parameters we used are listed in the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Author details
[1]School of Big Data and Software Engineering, Chongqing University, Chongqing 400044, China. [2]College of Microelectronics and Communication Engineering, Chongqing University, Chongqing 400044, China. [3]Faculty of Engineering and Physical Sciences, University of Leeds, Leeds LS2 9JT, UK. [4]Haihe Laboratory of Information Technology Application Innovation, Tianjin University, Tianjin 300072, China.

Received: 30 December 2022 Accepted: 14 August 2023
Published online: 24 August 2023

References
1. Chen Y, Hu J, Zhao J, Min G (2023) QoS-aware computation offloading in leo satellite edge computing for IoT: a game-theoretical approach. Chin J Electron. https://doi.org/10.23919/cje.2022.00.412
2. Cisco (2022) Cisco Annual Internet Report (2018-2023). https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. Accessed 10 Mar 2020
3. Llorca J, Tulino AM, Guan K, Esteban J, Varvello M, Choi N et al (2013) Dynamic in network caching for energy efficient content delivery. In: 2013 Proceedings IEEE INFOCOM. IEEE, Turin, pp 245–249
4. 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
5. Chen Y, Zhao J, Hu J, et al (2023a) Distributed task offloading and resource purchasing in NOMA-enabled mobile edge computing: Hierarchical game theoretical approaches. ACM Trans Embed Comput Syst. https://doi.org/10.1145/3597023
6. Chen Y, Zhao J, Zhou X, et al (2023b) A distributed game theoretical approach for credibility-guaranteed multimedia data offloading in MEC. Inf Sci. https://doi.org/10.1016/j.ins.2023.119306
7. Pan J, McElhannon J (2017) Future edge cloud and edge computing for internet of things applications. IEEE Internet Things J 5(1):439–449
8. Chen Y, Zhao J, Wu Y et al (2022) QoE-aware decentralized task offloading and resource allocation for end-edge-cloud systems: a game-theoretical approach. IEEE Trans Mob Comput. https://doi.org/10.1109/TMC.2022.3223119
9. Huang J, Wan J, Lv B, Ye Q et al (2023) Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning. IEEE Syst J 17(2):2500–2511
10. Zhong C, Gursoy MC, Velipasalar S (2019) Deep multi-agent reinforcement learning based cooperative edge caching in wireless networks. In: ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, Shanghai, pp 1–6
11. Chen Y, Gu W, Xu J, et al (2022) Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning. China Commun. https://doi.org/10.23919/JCC.ea.2022-0372.202302

12. Fan Q, Li X, Li J, He Q, Wang K, Wen J (2021) PA-Cache: evolving learning-based popularity-aware content caching in edge networks. IEEE Trans Netw Serv Manag 18(2):1746–1757
13. Song J, Song Q, Wang Y, Lin P (2021) Energy-delay tradeoff in adaptive cooperative caching for energy-harvesting ultradense networks. IEEE Trans Comput Soc Syst 9(1):218–229
14. Zhang S, Sun W, Liu J (2019) Spatially cooperative caching and optimization for heterogeneous network. IEEE Trans Veh Technol 68(11):11260–11270
15. Ren H, Xu Z, Liang W, Xia Q, Zhou P, Rana OF, Galis A, Wu G (2020) Efficient algorithms for delay-aware NFV-enabled multicasting in mobile edge clouds with resource sharing. IEEE Trans Parallel Distrib Syst 31(9):2050–2066
16. Araniti G, Rinaldi F, Scopelliti P, Molinaro A, Iera A (2019) A dynamic MBSFN area formation algorithm for multicast service delivery in 5G NR networks. IEEE Trans Wirel Commun 19(2):808–821
17. Condoluci M, Araniti G, Molinaro A, Iera A (2015) Multicast resource allocation enhanced by channel state feedbacks for multiple scalable video coding streams in lte networks. IEEE Trans Veh Technol 65(5):2907–2921
18. Guo C, Cui Y, Ng DWK, Liu Z (2018) Multi-quality multicast beamforming with scalable video coding. IEEE Trans Commun 66(11):5662–5677
19. Wu F, Yang W, Ren J, Lyu F, Ding X, Zhang Y (2020) Adaptive video streaming using dynamic ndn multicast in wlan. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). IEEE, Toronto, pp 97–102
20. Chen S, Yao Z, Jiang X, Yang J, Hanzo L (2020) Multi-agent deep reinforcement learning-based cooperative edge caching for ultra-dense next-generation networks. IEEE Trans Commun 69(4):2441–2456
21. Poularakis K, Iosifidis G, Sourlas V, Tassiulas L (2016) Exploiting caching and multicast for 5G wireless networks. IEEE Trans Wirel Commun 15(4):2995–3007
22. Sun Y, Chen Z, Tao M, Liu H (2020) Bandwidth gain from mobile edge computing and caching in wireless multicast systems. IEEE Trans Wirel Commun 19(6):3992–4007
23. Jiang D, Cui Y (2019) Analysis and optimization of caching and multicasting for multi-quality videos in large-scale wireless networks. IEEE Trans Commun 67(7):4913–4927
24. Huang W, Huang Y, He S, Yang L (2020) Cloud and edge multicast beamforming for cache-enabled ultra-dense networks. IEEE Trans Veh Technol 69(3):3481–3485
25. Dani MN, So DK, Tang J, Ding Z (2021) NOMA and coded multicasting in cache-aided wireless networks. IEEE Trans Wirel Commun 21(4):2506–2520
26. Bilal K, Shuja J, Erbad A, Alasmary W, Alanazi E, Alourani A (2022) Addressing challenges of distance learning in the pandemic with edge intelligence enabled multicast and caching solution. Sensors 22(3):1092
27. Hassanzadeh P, Tulino A, Llorca J, Erkip E (2016) Cache-aided coded multicast for correlated sources. In: 2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC). IEEE, Brest, pp 360–364
28. Li C, Toni L, Zou J, Xiong H, Frossard P (2017) Qoe-driven mobile edge caching placement for adaptive video streaming. IEEE Trans Multimedia 20(4):965–984
29. Tran TX, Pompili D (2018) Adaptive bitrate video caching and processing in mobile-edge computing networks. IEEE Trans Mob Comput 18(9):1965–1978
30. Wang F, Wang F, Liu J, Shea R, Sun L (2020) Intelligent video caching at network edge: a multi-agent deep reinforcement learning approach. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, Toronto, pp 2499–2508
31. Singal G, Laxmi V, Gaur MS, Rao DV, Kushwaha R, Garg D, Kumar N (2021) Qos-aware mesh-based multicast routing protocols in edge ad hoc networks: Concepts and challenges. ACM Trans Internet Technol (TOIT) 22(1):1–27
32. Damera B, Babu PC (2014) Broadcast the user chosen content over LTE in a simplified approach using evolved multimedia broadcast multicast service (eMBMS) with optimized MCE scheduling algorithm. In: 2014 International Conference on Intelligent Computing Applications. IEEE, pp 90–94
33. Zahoor K, Bilal K, Erbad A, Mohamed A (2020) Service-less video multicast in 5G: Enablers and challenges. IEEE Netw 34(3):270–276
34. Qin Y, Xia Q, Xu Z, Zhou P, Galis A, Rana OF, Ren J, Wu G (2020) Enabling multicast slices in edge networks. IEEE Internet Things J 7(9):8485–8501
35. Maddah-Ali MA, Niesen U (2014) Fundamental limits of caching. IEEE Trans Inf Theory 60(5):2856–2867
36. Liao J, Wong KK, Zhang Y, Zheng Z, Yang K (2017) Coding, multicast, and cooperation for cache-enabled heterogeneous small cell networks. IEEE Trans Wirel Commun 16(10):6838–6853
37. He S, Ren J, Wang J, Huang Y, Zhang Y, Zhuang W, Shen S (2019) Cloud-edge coordinated processing: Low-latency multicasting transmission. IEEE J Sel Areas Commun 37(5):1144–1158
38. Somuyiwa SO, György A, Gündüz D (2019) Multicast-aware proactive caching in wireless networks with deep reinforcement learning. In: 2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC). IEEE, Cannes, pp 1–5
39. Hartigan JA, Wong MA (1979) Algorithm as 136: A k-means clustering algorithm. J R Stat Soc Ser C (Appl Stat) 28(1):100–108
40. Lee D, Choi J, Kim JH, Noh SH, Min SL, Cho Y et al (1999) On the existence of a spectrum of policies that subsumes the least recently used (LRU) and least frequently used (LFU) policies. In: Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. ACM, Atlanta, pp 134–143
41. Bramson M (1994) Instability of FIFO queueing networks. Ann Appl Probab 4(2):414–431
42. Vietri G, Rodriguez LV, Martinez WA, Lyons S, Liu J, Rangaswami R et al (2018) Driving cache replacement with ML-based LeCaR. In: 10th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 18). USENIX, Boston, pp 928–936
43. Megiddo N, Modha DS (2003) ARC: A Self-Tuning, low overhead replacement cache. In: 2nd USENIX Conference on File and Storage Technologies (FAST 03), vol 3. USENIX, San Francisco, pp 115–130
44. Ban TW, Lee W, Ryu J (2020) An efficient coded streaming using clients' cache. Sensors 20(21):6220

Publisher's Note