Video Streaming Caching and Transcoding for Heterogeneous Mobile Users

Jinbo Cai†, Mingjun Xiao†*, He Sun†, Junjie Shao†, Yu Zhao†, Tongxiao Zhang‡
†School of Computer Science and Technology & Suzhou Institute for Advanced Study,
University of Science and Technology of China, Hefei, P. R. China.
‡School of Computer and Communication Engineering, Northeastern University at Qinhuangdao, P.R.China
*Correspondence to: xiaomj@ustc.edu.cn

***Abstract*—With the increasing prevalence of video streaming media, users have higher requirements for the quality of video streams, which can be quantified as Quality of Experience (QoE). Caching and transcoding video chunks on edge servers are effective methods to improve users' QoE and reduce the cost of retrieving video chunks. In this paper, we investigate how to cache and transcode video chunks for mobile users to maximize their QoE, while taking into consideration the cooperation between edge servers, users' mobility, and heterogeneous preferences, simultaneously. We employ a Multi-Agent Reinforcement Learning (MARL) framework and propose an MARL-based Cache replacement and Transcoding (MACT) mechanism. More specifically, we formulate the problem as a multi-agent game and prove this game is a state-based potential game. Then, we decompose this multi-agent game into individual agent decision problems to be solved. Through extensive simulations, we demonstrate that the proposed MACT mechanism achieves about 19.3% performance improvement, compared to the state-of-the-art works.**

***Index Terms*—Video streaming, edge caching, quality of experience, multi-agent reinforcement learning**

I. Introduction

A. Background

With the advent of new applications like 360-degree videos, there has been a surge in video streaming traffic in recent years, dominating network traffic [1], [2]. Due to the increasing demand for video quality from users, QoE has emerged as a crucial concern in academia [3]–[5]. QoE quantifies user satisfaction with service performance and indicates user experience or perception, i.e., users' subjective feelings about the video streaming service performance provided by the mobile network. High-quality QoE typically comprises a high bitrate, minimal rebuffering time, low startup delay, and few bitrate switches, all of which are closely related to the users' viewing experience [6], [7].

Caching and transcoding video chunks at the edge servers can significantly improve the users' viewing experience [8], [9]. Compared to retrieving video chunks directly from the cloud, this strategy has the following advantages: (1) Reduced latency: Edge servers are geographically close to users, resulting in relatively low latency in retrieving video chunks. (2) Alleviated cloud load: Caching and transcoding video chunks at edge servers can alleviate the burden on cloud servers and enhance the performance of cloud servers.

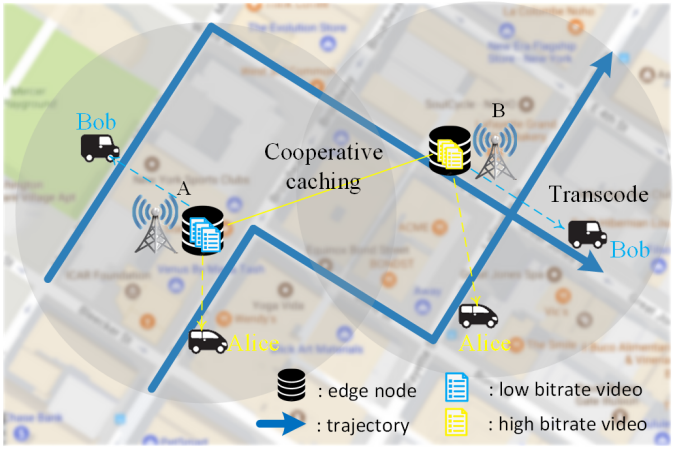

Fig. 1: A cooperative video chunk caching and transcoding scenario for users who might move across the areas covered by different edge servers and have heterogeneous preferences.

B. Motivations

Existing research has devoted significant efforts in enhancing user QoE. Some studies [10], [11] have investigated edge server caching and transcoding strategies to improve user experience while reducing the costs of retrieving video chunks. Other works [12]–[14] focus on pre-caching video chunks for users with high mobility, so as to provide smoother viewing experiences. Additionally, personalized caching and transcoding strategies are designed for heterogeneous users who have diverse preferences on video chunks [15]. However, none of them have taken into consideration the video chunk caching, transcoding, edge server cooperation, and users' mobility and heterogeneity simultaneously.

Fig. 1 illustrates a video chunk caching and transcoding scenario where users' mobility, heterogeneity and cooperation between edge servers are considered simultaneously. In this example, Alice prefers high bitrate while Bob prefers low latency. Edge servers A and B cache video chunks with low and high bitrates, respectively. When Alice is connected to server A and requests a high bitrate video chunk, edge server B can assist server A in fulfilling Alice's request. When Bob moves within the coverage area of edge server B and requests a

low bitrate video chunk, edge server B can transcode the high bitrate video chunk to the desired low-bitrate video chunk for Bob in advance.

C. Challenges

There are several challenges and obstacles that need to be overcome in enhancing users' QoE in above cooperative caching and transcoding scenario. Here are some crucial concerns:

- **Dealing with user mobility**: Due to users' mobility, a dynamic caching and transcoding strategy should support them in seamless transitions across the areas covered by different edge servers. This means that edge servers need to be highly flexible and adaptive, capable of cooperative determining the video chunk caching and transcoding strategy when users switch from one edge server to another.
- **Balancing transmission latency and video chunk bitrate**: The sizes of video chunks are typically proportional to the bitrate. Therefore, under the condition with the same bandwidth, a higher bitrate of video chunk will result in an increasing transmission delay. We need to strike a balance between reducing video transmission delay and increasing video chunk bitrate.

D. Main Results and Key Contributions

In order to address the aforementioned challenges, we employ a Multi-Agent Reinforcement Learning (MARL) framework and formulate the problem as a multi-agent game, where each edge server is treated as an agent. We propose an MARL-based Cache replacement and Transcoding (MACT) mechanism. The main contributions of this paper are summarized as follows:

- By considering the mobility and heterogeneity of users as well as the cooperation among edge servers, we introduce a cache replacement and transcoding problem for edge servers, which aims at enhancing user QoE and reducing the cost of retrieving video chunks from the cloud. To the best of our knowledge, this is the first QoE work which integrates user mobility, heterogeneous preferences, cooperative video chunk caching and transcoding simultaneously.
- We propose the MACT mechanism by formulating the problem as a multi-agent game to be solved, which can efficiently handle user mobility and heterogeneity by pre-caching and transcoding video chunks, ensuring the video experience for users.
- We demonstrate that the multi-agent game is a state-based potential game. Consequently, by maximizing the reward function of each agent individually, the MACT mechanism can achieve the maximum global reward function value.
- We have validated the effectiveness of the MACT mechanism through extensive simulations. The results show that MACT mechanism achieves about 19.3% performance improvement, compared to the state-of-the-art works.

The remainder of the paper is organized as follows. Section II reviews related works. Section III formalizes the problem of managing video chunk reception for highly mobile users in adaptive video stream system. Section IV details our proposed multi-agent reinforcement learning framework. Section V presents simulation results and relevant analyses. Finally, Section VI summarizes the key contributions and discusses possible future research directions.

II. Related Works

Caching video chunks on edge servers can effectively improve user experience [16]–[18]. The impact of collaborative mobile edge caching joint QoE and backhaul data traffic was examined in [19]. In order to address the optimization problem, a low-complexity self-tuned bitrate selection algorithm was devised, accompanied by an efficient cache replacement strategy. In [16], a comprehensive investigation was conducted on a mobile edge caching placement optimization problem for dynamic adaptive video streaming.

The caching and transcoding strategies of edge servers have been studied in [**?**], [11], [20]. Guo *et al.* [20] model the wireless channel as a finite-state Markov channel and formulate the optimization problem as a stochastic optimization problem involving the joint allocation of computational resources and adaptive video quality. Tran *et al.* [11] propose an integrated caching and processing cooperative framework that determines the placement of video variants in the cache and the scheduling of video requests to cache servers, aiming to minimize the expected latency cost of video retrieval. This framework significantly improves the cache hit rate while reducing backhaul traffic and content access latency.

Some studies have incorporated considerations of user mobility into the research on caching strategies for edge servers [12]–[14], [21]. Yun *et al.* [13] proposed a quality-aware deep reinforcement learning algorithm for real-time video streaming in infrastructure-assisted connected vehicles. By dynamically adjusting the transmission strategy between vehicles and infrastructure, they achieved high-quality video transmission while maintaining low latency and low data loss. Qiao *et al.* [14] modeled the proposed system as a cache management problem and achieved optimal video streaming quality by using the Markov decision process to dynamically allocate the proper cache memory space of each base station to mobile users.

III. The Proposed Framework Description

A. System Overview, Model, and Problem Formulation

The system mainly consists of three components: users who request video chunks, edge servers with the ability to cache and transcode video chunks, and a content server located in the central cloud. Fig. 2 illustrates the workflow of the system, i.e., the process for users requesting and receiving video chunks. (1) Each edge server makes cache replacement and transcoding decisions based on the system state from the previous time slot. (2) Users might move within a edge server or switch from one edge server to another. (3) Users request video chunks from the

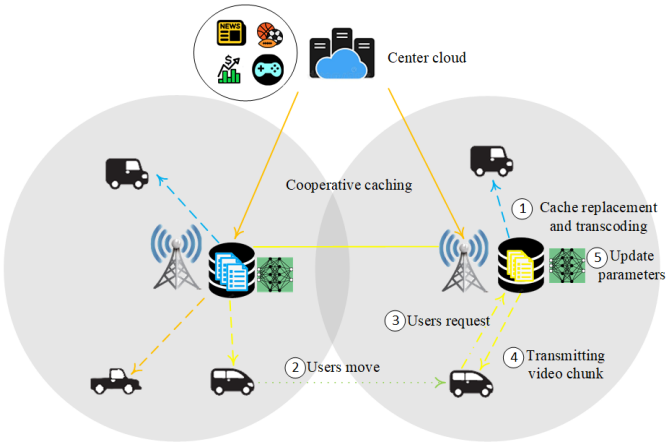

Fig. 2: The workflow of users requesting and receiving video chunks from edge servers.

HBS. (4) The system selects the video chunk delivery method that maximizes users' QoE to fulfill the users' request. If the nearby edge servers cannot fulfill the users' request, the video chunk will be sent from the cloud. (5) The system records the state and reward of this action and updates the parameters of the reinforcement learning neural network.

We assume that time is evenly divided into T time slots. This paper considers a system of J cache servers, denoted as $\mathcal{J}$={1, 2, ..., J}. Especially, j = 0 denotes the content server on the center cloud. The cache capacity, computational capacity, number of users, video chunks, and bandwidth of the j^{th} edge server are denoted as W_j, P_j, U_j, I_j, and R_j, respectively. The edge servers are connected via the backhaul network. The set of all video chunks and the set of encoded versions for each video chunk are represented by $\mathcal{I}$={1, 2, ..., I} and $\mathcal{Q}$={1, 2, ..., Q}, respectively. We denote the i^{th} video chunk with the q^{th} bitrate as V_i^q, where i $\in \mathcal{I}$ and q $\in \mathcal{Q}$ .The set of all users is indexed as $\mathcal{U}$={1, 2, ..., U}. We assume that the video stream consists of continuous chunks, and users watch these video chunks in sequence. The edge server to which the user belongs is called Home Base Station(HBS), and the edge server adjacent to the local edge server is called Neighbor Base Station(NBS).

B. System Model

1) Cache Limitations For Edge Servers: We can use a binary variable $b_i^{j,q}(t) \in \{0,1\}$ to represent the caching state of $V_{i,q}$ on the j^{th} edge server at time slot t. Specifically, the cache limitations of edge servers can be formalized as follows.

$$\sum_{i}^{I}\sum_{q}^{Q}(b_i^{j,q}(t)\cdot S_i^q) \le W_j \tag{1}$$

where S_i^q represents the size of the $V_{i,q}$. Equation (1) indicates that the sum of the sizes of video chunks cached on each edge server should not exceed the capacity limit W_j.

$$\sum_{j}^{J}\sum_{q}^{Q} b_i^{j,q}(t) \le 1 \tag{2}$$

$$b_i^{j,q}(t+1) = \begin{cases} 1 - \sum_{i' \in I_j, i' \neq i}^{I} c_{i',i}^{j,q}(t), if\ b_i^{j,q}(t) = 1 \\ \sum_{i' \in I_j, i' \neq i}^{I} c_{i,i'}^{j,q}(t), if\ b_i^{j,q}(t) = 0 \end{cases} \tag{3}$$

where $c_{i,i'}^{j,q}(t)$ represents whether $V_{i,q}$ has replaced $V_{i',q}$ on the j^{th} edge server at time slot t. In order to simplify the problem, it is assumed that only video chunks with the same bitrate can replace each other.

2) Transcoding Limitations For Edge Servers: The transcoding decision at time t is denoted as $t_{u,i}^{j,q,q'}(t) \in \{0,1\}$, where $q' > q$. If $t_{u,i}^{j,q,q'}(t) = 1$, it means that the video chunk V_i^q with a higher bitrate will be transcoded to $V_i^{q'}$ with a lower bitrate. Otherwise, if $t_{u,i}^{j,q,q'}(t) = 0$, no transcoding operation will be performed.

$$t_{u,i}^{j,q,q'}(t) \le b_i^{j,q}(t), q' > q \tag{4}$$

$$t_{u,i}^{j,q,q'}(t) \cdot b_i^{j,q'}(t) = 0, q' > q \tag{5}$$

3) Delay For Users: We introduce the following constraints to account for the limitations of bandwidth and computational resources on the edge servers.

$$d_{u,i}^{j,q}(t) = \frac{S_i^q}{R_u^j(t)} \tag{6}$$

$d_{u,i}^{j,q}(t)$ denotes the delay incurred when the video chunk is sent from the HBS to the user. $R_u^j(t)$ signifies the bandwidth allocated to the user during time slot t.

$$\sum_{u}^{U_j} R_u^j(t) \le R_j \tag{7}$$

$$d_{u,i}^{j,q,q'}(t) = \frac{P_i^{q,q'} \cdot t_{u,i}^{j,q,q'}(t)}{P_u^j}, q' > q \tag{8}$$

$d_{u,i}^{j,q,q'}(t)$ and $P_i^{q,q'}$ respectively denote the time and resources required to transcode the video chunk from bitrate q' to q. $P_u^j(t)$ represents the computational resources allocated to the user.

$$\sum_{u}^{U_j} P_u^j(t) \le P_j \tag{9}$$

In this paper, to simplify the problem, we assume that the edge server evenly distributes bandwidth and computational resources to the users.

As in Fig. 2, users have multiple ways to retrieve video chunks besides hitting directly at the HBS. Here are the four potential paths for a user to retrieve a video chunk:

- If the HBS caches the video chunk, the user can directly retrieve the video chunk from the HBS, resulting in a delay equal to the time required to send the video chunk.
- If the HBS caches a high-bitrate version of the video chunk and has transcoded it, the user can also get the video chunk directly from the HBS. The delay includes both the time to send and transcode the video chunk.
- If the NBS caches the video chunk, the video chunk needs to be transferred from the NBS to the HBS, and then sent

to the user. The delay consists of the time to send the video chunk and the time to transfer it from the NBS.
- If none of the above cases apply, we can consider that the user cannot get the video chunk in edge servers and needs to retrieve it from the cloud. In this case, the delay is the time required to send the video chunk from the cloud.

Therefore, the delay for the user to retrieve the video chunk can be represented as:

$$D_{u,i}^{j,q}(t) = min(\frac{1}{b_i^{j',q}(t-1)+\delta}(d_{u,i}^{j,q}(t) + d^{j,j'}), \frac{1}{b_i^{j,q'}(t-1)\cdot t_{u,i}^{j,q,q'}(t-1)+\delta}(d_{u,i}^{j,q}(t) + d_{u,i}^{j,q,q'}(t)), \frac{1}{b_i^{j,q}(t)+\delta}d_{u,i}^{j,q}(t), d_{u,i}^{0,q}(t)), q' > q \tag{10}$$

j' represents the neighboring servers of edge the j^{th} server. δ is a small constant used to avoid division by zero.

4) User Mobility Model: According to [22], [23], user movement direction and position information can be tracked and recorded when users move on the highway. This allows for predicting the probability of a user entering the next edge server at the next moment. The transition probability ρ can be derived from the vehicle speed and the coverage of each edge server. The vehicle speed distribution can be obtained from historical data. Suppose the range of the current edge server is x meters, and the vehicle speed distribution follows a uniform distribution from a m/s to b m/s. After time t, the probability of the user entering the next edge server is given by:

$$\rho(t) = \begin{cases} 0, t \le \frac{x}{a} \\ \frac{b-\frac{x}{t}}{b-a}, \frac{x}{a} < t \le \frac{x}{b}, \\ 1, \frac{x}{b} < t \end{cases} \tag{11}$$

In this paper, for convenience, we assume that the users' speed distribution follows a uniform distribution within a certain range. In actual scenarios, the users' speed distribution may vary and follow different types of distributions. However, we can still calculate the probability of a user reaching the next edge server. This assumption does not affect the effectiveness of the algorithm we proposed.

C. Problem Formulation

In the following section, we will introduce the optimization objective of the system. Firstly, we must consider the QoE for users. According to [7], the main factors influencing user QoE are the bitrate and latency of retrieving video chunks. We use $x_{u,1}$ and $x_{u,2}$ to represent users' QoE preferences for video bitrate and delay, respectively. Q(·) represents the users' satisfaction function with respect to bitrate changes. Therefore, users' QoE can be represented as $x_{u,1} \cdot Q(q) - x_{u,2} \cdot D_{u,i}^{j,q}(t)$. Secondly, we also need to consider the cost of retrieving video chunks from the cloud, denoted as $C_{u,i}^{j,q}$. Therefore, our joint optimization objective comprises three components: video chunk bitrate, latency, and the cost of obtaining video chunks from the cloud. These are detailed below:

$$F_{u,i}^{j}(t) = \max_{q}(r_{u,i}^{j}(t)(k_1(x_{u,1}\cdot Q(q) - x_{u,2}\cdot D_{u,i}^{j,q}(t)) - k_2\cdot \alpha_{u,i}^{j,q}(t)\cdot C_{u,i}^{j,q})) \tag{12}$$

$r_{u,i}^{j}(t)$ denotes the u^{th} user's request. $\alpha_{u,i}^{j,q}(t)$ indicates whether the user obtains the video chunk from the cloud. k_1, k_2 respectively represent the weights assigned to user QoE and the cost of obtaining video chunks from the cloud, reflecting the system's emphasis on these two aspects

The decision variables of the system are denoted as C(t) and T(t). C(t) represents the cache replacement policy of the system, while T(t) represents the transcoding policy. Therefore, the optimization objective of the system can be expressed as follows:

$$\max_{C(t),T(t)} \sum_{t}^{T}\sum_{u}^{U}\sum_{i}^{I}\sum_{j}^{J} F_{u,i}^{j}(t) \quad s.t.(1)-(12) \tag{13}$$

$$\{b_i^{j,q}(t), c_{i,i'}^{j,q}(t), r_{u,i}^{j}(t), t_{u,i}^{j,q,q'}(t)\} \in \{0,1\}, q' > q \tag{14}$$

The above optimization problem involves a large search space of cache states and computing resource states, as well as two distinct decision variables. Finding the optimal solution within polynomial time becomes a highly challenging task. Furthermore, considering that user requests are dynamically changing, traditional methods struggle to effectively solve this problem. Therefore, we propose employing the method of multi-agent reinforcement learning to tackle this issue.

IV. The MACT mechanism

In this section, we turn the collaborative transcoding and caching process of video streams in the edge environment as a Markov Decision Process (MDP). We assign an agent to each edge server to determine its cache replacement and transcoding strategy, and formulate the control problem of multiple edge servers as a game $G = \{\mathbb{I}, \mathbb{S}, \mathbb{A}, \mathbb{R}\}$, where $\mathbb{I}$ represents the set of agents, and $\mathbb{S}, \mathbb{A}, \mathbb{R}$ represent the joint state space, action space, and reward space of all agents, respectively.

A. Problem Decoupling

In the following, we establish that the proposed game is a state-based potential game in multi-agent systems, which must satisfy the following condition [24]:

$$r_i(s, a_i, a_{-i}) - r_i(s, a_i', a_{-i}) = \Phi(s, a_i, a_{-i}) - \Phi(s, a_i', a_{-i}) , \forall (a_i, a_{-i}), (a_i, a_{-i}) \in \mathbb{A}, \forall s \in \mathbb{S}, \forall i \in \mathbb{I} \tag{15}$$

where a_i and a_{-i} represent the action of agent i and the actions of all other agents in the system except for agent i, respectively. r_i denotes the reward of agent i. Φ is a measurable function that represents the mapping from states and actions to rewards. We define the potential function as

$\Phi(s,a) = \sum_j r_j(s,a)$. By substituting the definition into (15), we obtain:

$$r_i(s,a_i,a_{-i}) - r_i(s,a_i',a_{-i}) = \sum_j r_j(s,a_i,a_{-i}) - \sum_j r_j(s,a_i',a_{-i}) \tag{16}$$

According to (10), each agent's current reward function is solely influenced by its own actions, so we can obtain: $r_i(s,a_i,a_{-i}) = r_i(s,a_i)$. Substituting it into the right-hand side of (16):

$$\begin{aligned} &\sum_j r_j(s,a_i,a_{-i}) - \sum_j r_j(s,a_i',a_{-i}) \\ &= \sum_j r_j(s,a_i) - \sum_j r_j(s,a_i') \\ &= r_i(s,a_i,a_{-i}) - r_i(s,a_i',a_{-i}) \end{aligned} \tag{17}$$

Therefore, the multi-agent game we propose is a state-based potential game. By individually maximizing each agent's reward function, we can achieve the objective of maximizing the global potential function. Therefore, the multi-agent game problem can be decoupled into individual agent decision problems. Consequently, each agent employs an independent learning algorithm to maximize the global reward function.

D3QN (Dueling Double Deep Q-Network) is a reinforcement learning algorithm that combines the strengths of Double DQN and Dueling DQN. By incorporating the Dueling network architecture, D3QN can accurately estimate the value of each action, leading to improved learning efficiency and convergence speed. Consequently, we adopt the D3QN algorithm to train each agent.

B. State Space

In time slot t, the system encompasses five distinct states, which are described as follows: (1) The user's location $L(t)$, indicating the current edge server to which the user is connected. (2) The requested video chunk by the user $R(t)$. (3) User preference $U(t)$. (4) User mobility probability $P(t)$. (5) The cache status of the edge server $C'(t)$. It is important to note that all these states possess the Markov property. Hence, the state of the defined MDP can be represented as $S_t = (L(t), R(t), U(t), P(t), C'(t))$, where $L(t) \in \mathbb{L}$, $R(t) \in \mathbb{R}$, $U(t) \in \mathbb{U}$, $P(t) \in \mathbb{P}$, and $C'(t) \in \mathbb{C}'$. Here, $\mathbb{L}$, $\mathbb{R}$, $\mathbb{U}$, $\mathbb{P}$, and $\mathbb{C}'$ respectively denote the sets of all possible user locations, user requests, user preferences, user mobility probabilities, and system cache states.

C. Action Space

As mentioned previously, in our proposed solution, each edge server has two actions: (1) Cache replacement decision $C(t)$ and (2) Transcoding decision $T(t)$. Therefore, the action taken at time t can be defined as $A_t = (C(t), T(t))$, where $C(t) \in \mathbb{C}$ and $T(t) \in \mathbb{T}$. Here, $\mathbb{C}$ and $\mathbb{T}$ represent all possible cache replacement decisions and transcoding decisions, respectively. Specifically, $C(t)$ involves determining which video chunks to eliminate and which ones to cache, while $T(t)$ includes selecting which video chunks to transcode and deciding the transcoding bitrate.

D. Reward Function

Setting the reward function is crucial for reinforcement learning algorithms. In the previous section, we demonstrated that by defining the reward function for each agent as (10), the proposed game can be proven to be a state-based potential game. By individually maximizing the reward function for each agent, we can achieve the global optimal solution. Therefore, the reward function for each agent can be represented as follows:

$$r_t = \sum_u^U \sum_i^I \sum_j^J F_{u,i}^j(t) \tag{18}$$

E. The Detailed Algorithm

Algorithm 1: MACT

Initialize:Initialize each agent's evaluation network and target network with random weights. Initialize the replay buffer $\mathcal{M}$.
for $episode = 1, 2, \ldots, E$ **do**
  Reset the environment and get an initial observation state s_0. **for** $t = 1, 2, \ldots, T$ **do**
    for $i = 1, 2, \ldots, I$ **do**
      Each agent select actions $a_{i,t}$ based on current state and random exploration probability.
      Each agent executes $a_{i,t}$, that is, executes corresponding cache replacement and transcoding actions.
    end
    Users move and request video chunks from local edge servers.
    Based on the current cache situation and transcoding chunks, select the video chunk that maximizes the users' QoE for each request and obtain the reward.
    The environment return the state of the next time slot s_{t+1} and reward r_t.
    Store sample (s_t, a_t, r_t, s_{t+1}) in $\mathcal{M}$.
    Sample a random minbatch of K samples from $\mathcal{M}$.
    for $i = 1, 2, \ldots, I$ **do**
      Each agent update the evaluation network weight θ_i^E with loss function according to (19)
      Each agent update the evaluation network weight θ_i^T with loss function according to (20)
    end
  end
end

The training method for MACT is described in Algorithm 1. At each time slot t, each agent selects actions $a_{i,t}$ based on the current state and a random exploration probability

and executes the actions. Then, the users move and request video chunks. Based on the accuracy of the prediction, each agent receives corresponding rewards. The next state s_{t+1} and reward r_t are returned by the environment. The sample (s_t, a_t, r_t, s_{t+1}) is stored in the experience replay buffer $\mathcal{M}$. Finally, each agent updates its parameters based on K randomly selected samples. The parameters of the evaluation network and the target network are updated using the following formula:

$$L(\theta^E) = \frac{1}{K}\sum_{i=1}^{K}[r_{j+1} + \gamma Q^T(s_{j+1}, a_E^*; \theta^T) - Q^E(s_j, a_j; \theta^E)]^2 \quad (19)$$

$$\theta^T = \tau\theta^E + (1-\tau)\theta^T \quad (20)$$

where $a_E^* = \underset{a'}{argmax} Q^E(s_{j+1}, a'; \theta^E)$, Q^T and Q^E represent the Q functions of the target network and evaluation network, respectively. θ^T and θ^E are the weights of the target network and evaluation network, respectively. Finally, the parameters of the target network are updated slightly by τ.

V. Evaluation

A. Simulation Settings

In our simulation, we consider an adaptive video stream system composed of three edge servers. Given that for some users, the marginal improvement of perceived quality diminishes at higher bitrates, we assume that user satisfaction grows at a logarithmic rate with bitrate. Like [7], we assume that video chunks are encoded at a constant bitrate, and thus the size of the video chunk is proportional to the bitrate. The latency ratio for acquiring video chunks of the same bitrate from local edge servers, neighboring edge servers, and the cloud is set at 1:2:5. The transcoding delay is proportional to the bitrate, being 0.2 times the video chunk transmitted from HBS. Each video chunk is available in four bitrates, and like [25], the relative bitrates of video chunks are set to 1:2.5:5:10. User video requests arrive following a Zipf distribution, with a parameter α set at 0.8. The size of the cache in the cluster is set at 8% of the total library size. In (12), the trade-off weights k_1 and k_2 are empirically set at 0.8 and 0.2, respectively, to balance user QoE and transmission cost. As for user preference, the ratio range between $x_{u,1}$ and $x_{u,2}$ is $(0, \frac{3}{8})$.

In implementing the DRL algorithm, our neural network consists of two fully-connected hidden layers, each containing 1024 neurons. We set the capacity of the experience replay buffer to 100000 and the batch size to 4096. The discount factor γ in the reward function is set to 0.95, and the network update parameter τ is set to 0.005. To ensure the algorithm operates within the valid action space, we constrain the output of the neural network to fall within the allowable action range. Any selected action outside of this range will not be executed.

The total reward is the primary metric used to evaluate the performance of our proactive caching solution. To demonstrate the effectiveness of our proposed algorithm from various perspectives, we also employ the hit rate of video chunks in the system, the QoE of users, and the cost-saving rate in the cloud as performance evaluation metrics. Additionally, in this section, we will analyze the influence of the total cache size of the edge servers and different Zipf parameters on the total reward.

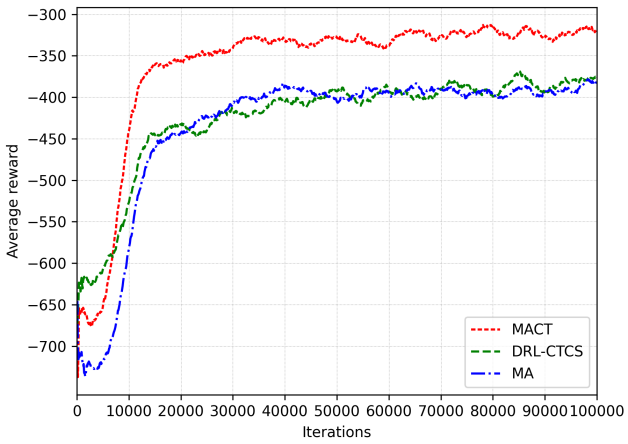

Fig. 3: The convergence of MACT and the two compared algorithms.

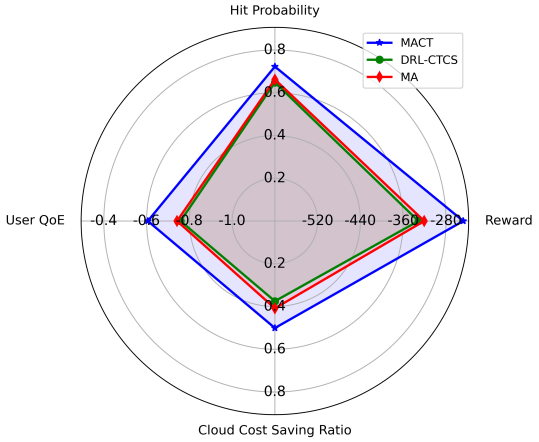

Fig. 4: The hit rate , the QoE of users and the cloud cost-saving of MACT and two compared algorithms

B. Convergence

In this section, we evaluate the performance of MACT, comparing it with two alternative methods.

DRL-CTCS: The work in [10] proposes a cooperative edge transcoding and caching strategy for video streams in edge clusters. However, this strategy assumes that the number of requesting users is fixed, unchanged corresponding edge servers for users, and does not consider varying preferences among users. We have adapted this algorithm to fit our MARL model and incorporated consideration for user preferences.

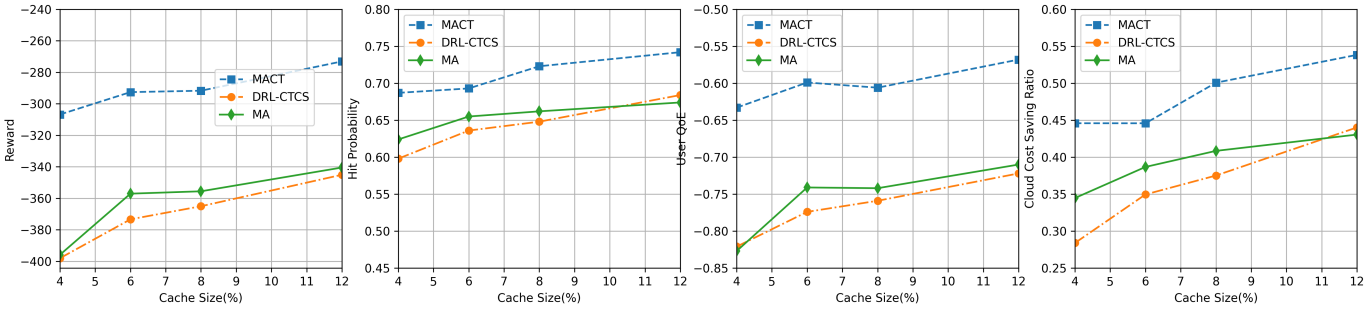

Fig. 5: Performance comparison of MACT and two compared algorithms with varying cache sizes

Mobile-based Algorithm(MA): The studies in [12]–[14], although considering user mobility, overlook user preferences and collaboration between edge servers. Additionally, they do not consider the transcoding feature of video chunks. As a result, we have adapted this algorithm to fit our MARL model, which does not use user preference information for decision-making, and users cannot obtain video chunks from adjacent edge servers.

Fig. 3 illustrates the training process of MACT and two compared algorithms. After 100,000 training iterations, all three algorithms have converged. The average reward of MACT converges to -291, while the DRL-CTCS and MA converge to -365 and -355, respectively. We calculated the improvement of MACT relative to the compared algorithms using the reward -687 achieved by the random algorithm as the baseline value. MACT achieves about 23.0% and 19.3% performance improvement in rewards compared to DRL-CTCS and MA, respectively. Clearly, MACT outperforms both compared algorithms. The performance of DRL-CTCS is the worst because it does not consider user mobility and cannot predict which video chunks need to be cached in advance. Additionally, it may cache video chunks for users who are about to leave the coverage area of the edge server, leading to resource waste.

Fig. 4 demonstrates that MACT outperforms the other two in terms of hit rate, user QoE, and cloud cost savings. MACT achieves a hit rate of 70.0% by predicting user movement and enabling collaboration between edge servers, while DRL-CTCS and MA achieve hit rates of 61.9% and 65.3%, respectively. In terms of user QoE, MACT significantly enhances the perceived quality by considering individual preferences while maintaining a high hit rate. In contrast, MA shows moderate performance, and DRL-CTCS performs the worst. This demonstrates the significant cost-saving effectiveness of MACT and its ability to alleviate pressure on the cloud, while the two compared algorithms require more frequent assistance from the cloud.

C. Impact of Cache Size

The cache size has a significant impact on the algorithm's performance. Fig. 5 compares the performance of the three algorithms across different cache capacities, ranging from 4% to 12%. All other parameters are kept identical to those mentioned in the first subsection.

As shown in Fig. 5(a), the rewards of all three algorithms improve as the cache size increases. It's noteworthy that the rate of improvement is more pronounced at smaller cache sizes and starts to diminish as the cache size becomes larger. This phenomenon can be attributed to the marginal improvement that increasing the cache space can bring to system performance. MACT consistently outperforms the two compared algorithms in terms of hit rate, achieving an average performance gain of 7.0% and 5.4%, respectively. Fig. 5(c) illustrates the QoE of users under different strategies. As the cache space increases, the hit rate of DRL-CTCS shows a greater improvement compared to MA because it can request assistance from NBS. Furthermore, the user QoE of all three algorithms improves as the cache space increases.

D. Impact of Zipf parameter

To investigate the impact of varying video popularity on the three strategies, we set three different Zipf parameters: 0.5, 0.8, and 1, with the cache size set to 8% of the video library. The Zipf parameter plays a crucial role in algorithm performance, as an increase in the Zipf parameter leads to a more concentrated distribution of users' requests on a few video chunks. As shown in Fig. 6, when users request the same video chunk more frequently, the performance of the three schemes improves to varying extents across all metrics. Compared to the two compared algorithms, MACT exhibits a significant increase in the average reward by 77.7 and 67.9, respectively. DRL-CTCS has the lowest reward, followed by the Mobile-based Algorithm. MACT demonstrates a more pronounced advantage when the Zipf parameter is smaller.

VI. Conclusion

In this paper, we tackle the challenge of enhancing QoE in edge networks by introducing a Multi-Agent Reinforcement Learning framework. Our framework is specifically designed to tackle dynamic caching and transcoding strategies on edge servers, considering various complexities including high user mobility, user heterogeneity, and the latency-bitrate trade-off. We have conducted comprehensive evaluations of our proposed model, and the results demonstrate the effectiveness of our approach, with significant improvements observed in key

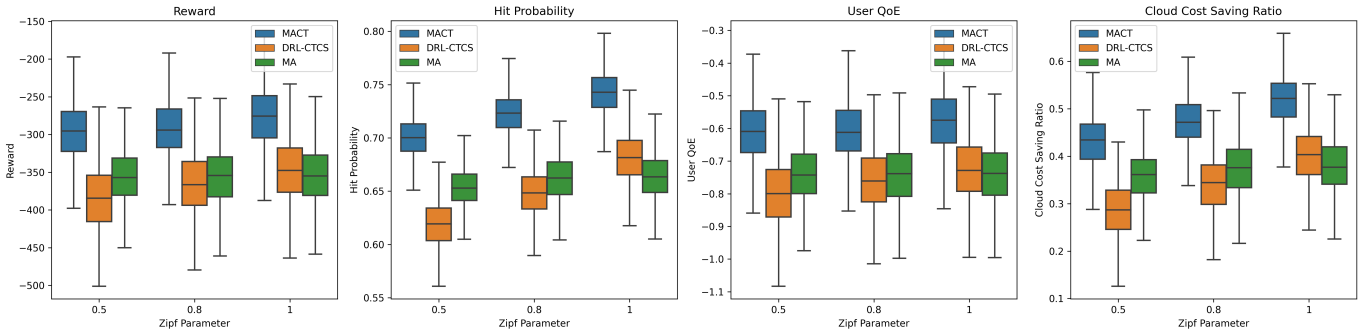

Fig. 6: Performance comparison of MACT and two compared algorithms with varying Zipf parameters

metrics such as hit rate, user QoE, and cloud cost. In the future, we aim to integrate more lifelike constraints, like evolving user behavior, the content correlation in video streams, and user collaboration, while also delving into the scalability of our approach in more extensive network environments.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 62172386, 61872330, 61572457, the Natural Science Foundation of Jiangsu Province in China under Grants BK20231212, BK20191194, and BK20131174, and the Teaching Research Project of the Education Department of Anhui Province in China (No. 2021jyxm1738).

REFERENCES

[1] U. Cisco, "Cisco annual internet report (2018–2023) white paper," *Cisco: San Jose, CA, USA*, vol. 10, no. 1, pp. 1–35, 2020.

[2] L. Sun, F. Duanmu, Y. Liu, Y. Wang, Y. Ye, H. Shi, and D. Dai, "Multi-path multi-tier 360-degree video streaming in 5g networks," in *Proceedings of the 9th ACM multimedia systems conference*, 2018, pp. 162–173.

[3] S. Cheng, H. Hu, X. Zhang, and Z. Guo, "Rebuffering but not suffering: Exploring continuous-time quantitative qoe by user's exiting behaviors."

[4] G. Lv, Q. Wu, W. Wang, Z. Li, and G. Xie, "Lumos: Towards better video streaming qoe through accurate throughput prediction," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 650–659.

[5] J. Li, Z. Li, Q. Wu, and G. Tyson, "On uploading behavior and optimizations of a mobile live streaming service," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1299–1308.

[6] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," *IEEE/ACM Transactions On Networking*, vol. 28, no. 4, pp. 1698–1711, 2020.

[7] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 325–338.

[8] E. Baccour, A. Erbad, K. Bilal, A. Mohamed, and M. Guizani, "Pccp: Proactive video chunks caching and processing in edge networks," *Future Generation Computer Systems*, vol. 105, pp. 44–60, 2020.

[9] H. Zhao, Q. Wang, J. Wang, B. Wan, and Z. Wu, "Popularity-based and version-aware caching scheme at edge servers for multi-version vod systems," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 3, pp. 1234–1248, 2020.

[10] T. Yang, Z. Tan, Y. Xu, and S. Cai, "Collaborative edge caching and transcoding for 360° video streaming based on deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 25 551–25 564, 2022.

[11] T. X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 18, no. 9, pp. 1965–1978, 2018.

[12] Z. Zhang, Y. Yang, M. Hua, C. Li, Y. Huang, and L. Yang, "Proactive caching for vehicular multi-view 3d video streaming via deep reinforcement learning," *IEEE Transactions on Wireless Communications*, vol. 18, no. 5, pp. 2693–2706, 2019.

[13] W. J. Yun, D. Kwon, M. Choi, J. Kim, G. Caire, and A. F. Molisch, "Quality-aware deep reinforcement learning for streaming in infrastructure-assisted connected vehicles," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 2002–2017, 2021.

[14] J. Qiao, Y. He, and X. S. Shen, "Proactive caching for mobile video streaming in millimeter wave 5g networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 7187–7198, 2016.

[15] X. Zuo, J. Yang, M. Wang, and Y. Cui, "Adaptive bitrate with user-level qoe preference for video streaming," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1279–1288.

[16] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "Qoe-driven mobile edge caching placement for adaptive video streaming," *IEEE Transactions on Multimedia*, vol. 20, no. 4, pp. 965–984, 2017.

[17] X. Chen, L. He, S. Xu, S. Hu, Q. Li, and G. Liu, "Hit ratio driven mobile edge caching scheme for video on demand services," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2019, pp. 1702–1707.

[18] M. F. Tuysuz and M. E. Aydin, "Qoe-based mobility-aware collaborative video streaming on the edge of 5g," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 11, pp. 7115–7125, 2020.

[19] A. Mehrabi, M. Siekkinen, and A. Ylä-Jaaski, "Qoe-traffic optimization through collaborative edge caching in adaptive mobile video streaming," *IEEE Access*, vol. 6, pp. 52 261–52 276, 2018.

[20] Y. Guo, F. R. Yu, J. An, K. Yang, C. Yu, and V. C. Leung, "Adaptive bitrate streaming in wireless networks with transcoding at network edge using deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 3879–3892, 2020.

[21] Y. Cai, Y. Chen, M. Ding, P. Cheng, and J. Li, "Mobility prediction-based wireless edge caching using deep reinforcement learning," in *2021 IEEE/CIC International Conference on Communications in China (ICCC)*. IEEE, 2021, pp. 1036–1041.

[22] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[23] Y. Yuan, W. Wang, Y. Wang, S. S. Adhatarao, B. Ren, K. Zheng, and X. Fu, "Vsim: Improving qoe fairness for video streaming in mobile environments," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1309–1318.

[24] J. R. Marden, "State based potential games," *Automatica*, vol. 48, no. 12, pp. 3075–3088, 2012.

[25] J. Kim and E.-S. Ryu, "Feasibility study of stochastic streaming with 4k uhd video traces," in *2015 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2015, pp. 1350–1355.