Joint Content Popularity and Audience Retention-Aware Live Streaming over RSMA Edge Networks

Fayshal Ahmed[a], The-Vinh Nguyen[b], Nam-Phuong Tran[c], Nhu-Ngoc Dao[a,*], Sungrae Cho[d,*]

[a] *Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea*
[b] *R&D Department, LG Innotek Vietnam, Haiphong 180000, Viet Nam*
[c] *Centre for Wireless Communications, University of Oulu, Oulu 90014 , Finland*
[d] *School of Computer Science and Engineering, Chung-Ang University, Seoul 06974, South Korea*

Abstract

The exponential growth of high-quality live streaming services over cellular networks, particularly in heterogeneous environments facilitated by 6G, has underscored the need for novel wireless communication. To address this challenge, Rate Splitting Multiple Access (RSMA) has emerged as a promising interference management scheme in advanced cellular networks. This paper considers such a potential environment where the impacts of content popularity and audience retention are jointly investigated to maximize the average video resolution of live streaming services over RSMA edge networks. The complex problem is modeled as a Markov Decision Process and subsequently addressed using an appropriate reinforcement learning framework leveraging the Deep Deterministic Policy Gradient (DDPG) technique, named DDPG-BARMAS. Simulation results demonstrate that the proposed DDPG-BARMAS method significantly outperforms existing algorithms in terms of video resolution improvement, highlighting its potential as a robust solution for future wireless live-streaming services.

Keywords: Live video streaming, Multiuser uplink RSMA, Video bitrate maximization, Quality of Experience.

1. Introduction

In recent years, live video streaming platforms have garnered significant public attention and surged in popularity, driven by the rapid proliferation of mobile Internet of Things (IoT) devices equipped with 5G connectivity. The widespread appeal of live video streaming services can be surpassed by several factors, including the explosive growth of live broadcast platforms celebrated for their high entertainment value and robust real-time social interaction capabilities [1]. In the realm of live video streaming systems, the quality, popularity, and stability of streamer videos are pivotal in optimizing average video bitrate and enhancing Quality of Experience (QoE). High-quality streamer videos significantly impact follower engagement metrics such as total playback time and live stream views. In addition, the average bitrate plays a crucial role in determining the quality and viewer satisfaction of live video content [2]. High video bitrate for live streaming services in multiuser uplink networks is imperative for delivering superior live video streaming experiences. These advancements are crucial as the demand for seamless, high-quality streaming continues to grow in an increasingly connected digital landscape [3, 4].

On the side of mobile communication evolution, Rate Splitting Multiple Access (RSMA) has emerged as a transformative technology, capable of delivering robust, high data rates, and low latency essential for 6G networks and beyond [5, 6, 7]. Consequently, RSMA has garnered substantial attention from both academia and industry. Uplink RSMA operates on the principle of rate splitting, in which each message transmitted from the transmitter to a user is divided into two distinct segments [8]. By judiciously allocating disparate power levels, mobile gNodeBs employ a successive interference cancellation (SIC) technique to decode the incoming messages efficiently. In a mobile live streaming system, a 3GPP-standardized 5GMSu edge server, strategically positioned at the gNB, makes optimal decisions to maximize the QoE of live streaming services based on an analysis of streamer fame and watch time duration as well as mobile environment conditions.

In this paper, we propose an advanced bitrate maximization for average streamers approach that leverages the Deep Deterministic Policy Gradient (DDPG) technique (i.e., DDPG-BARMAS) in an uplink RSMA network. By jointly considering content popularity and retention rates, we effectively designate available streamers for video bitrate adjustments. Our primary objective is to identify an optimal solution that maximizes the overall video bitrate of the system. The key contributions of this paper are summarized as follows:

- We investigated the combined impact of content popularity and audience retention on the video resolution of live streaming services over an uplink RSMA network. Here, specific characteristics of uplink RSMA communication, i.e., the decoding order and transmit power, are

*Corresponding authors
Email addresses: 23110143@sju.ac.kr (Fayshal Ahmed), vinhnt@lginnotek.com (The-Vinh Nguyen), phuong.tran@oulu.fi (Nam-Phuong Tran), nndao@sejong.ac.kr (Nhu-Ngoc Dao), srcho@cau.ac.kr (Sungrae Cho)

optimized with content popularity and audience retention consideration to maximize average video bitrate to improve the system QoE.

- To resolve the above objective, we proposed an intelligent optimization framework, named DDPG-BARMAS, inspired by the DDPG technique. Addressing the complexity of non-convex optimization, we transformed it into a Markov decision process (MDP), defining state, action, and reward functions. By replacing discrete variables with normalized continuous ones, we navigate the high-dimensional state-action space and dynamic network channels. Our proposed DDPG-BARMAS framework focuses on maximizing long-term expected rewards and optimizing bitrate for average streamers.
- We conducted rigorous simulations to validate our approach. The simulation results demonstrate significant improvements in average video resolution compared to existing methodologies. This validation underscores the algorithm's effectiveness in enhancing video streaming quality through advanced bitrate optimization.

The paper is structured as follows. Section 2 provides related work analysis. Section 3 elucidates the problem statement, channel model, service model, and optimization problem. Section 4 delineates the DDPG-based BARMAS optimization framework. In Section 5, we sequentially present performance evaluation and convergence analysis comparisons. Finally, Section 6 concludes and describes prospective contributions for further enhancements.

2. Related Works

The literature review is structured into two primary domains. First, it investigates the consideration of RSMA techniques in streaming systems. Second, it explores the utilization of the DDPG technique to enhance video bitrates as well as QoE.

2.1. Live Video Streaming in RSMA Communication Environments

RSMA method has received significant attention due to its ability to achieve higher spectrum and energy efficiencies compared to conventional multiple access techniques. For instance, Dao et al. explored content quality in edge caching systems for multi-user Adaptive Bitrate Streaming (ABS) and additive bitrate streaming services under challenging transmission conditions [9]-[10]. Their dynamic policy for cache decision-making and quality-level selection during cache cycles, along with a drift-plus-penalty balancing optimization, demonstrated outstanding performance across various video streaming popularity models. On the other hand, Ma et al. proposed QAVA, a QoE-aware additive video bitrate aggregation scheme for HTTP live streaming using edge computing [11]. QAVA adapts bitrate dynamically based on network conditions, client states, and video characteristics using a DRL-based algorithm deployed on intelligent proxy servers at the edge. In [12], joint computational resource assignment and bitrate adaptation in edge caching systems were optimized using the asynchronous advantage actor-critical algorithm, assuming a discrete-time Markov chain (DTMC) for the channel with variable time. Fu et al. introduced a soft actor-critic network in [13] to maximize video bitrate while minimizing time delays and bitrate variations, focusing on improving streaming quality. In [14], a perceptual content-aware bitrate adaptation algorithm for HTTP streaming services was proposed. This approach considers visual perception's impact on user QoE, modeling the bitrate adaptation problem as a MDP and developing a segmented value iteration method to solve it.

2.2. DDPG Utilization for QoE Improvement

The DDPG algorithm has emerged as a compelling solution capable of developing optimal policies through interaction with the environment, without requiring prior knowledge of the network system model. For example, in [15], a robust proposal was introduced for a high-altitude platform mounted MEC (HAMEC) system within an RSMA environment. This proposal focused on jointly configuring parameters such as offloading decisions, splitting ratios, transmit power, and decoding to minimize processing costs in terms of response latency and energy consumption, using the DDPG algorithm. Kim et al. [16] formulated a problem to maximize sum rates by jointly optimizing Intelligent Reflecting Surface (IRS) beamformers, Base Station (BS) combiners, and User Equipment (UE) transmit powers. They proposed a multi-agent deep reinforcement learning algorithm to address this challenge. In [17], a multi-agent deep reinforcement learning (MADRL) based algorithm was introduced to jointly optimize resource block (RB) allocation and power control, aiming to maximize average spectrum efficiency (SE) while meeting Quality of Service (QoS) constraints. In [18], Liu et al. aimed to minimize the energy consumption of the base station during complete video streaming sessions, with an additional constraint to prevent interruptions in video playback. They used the DDPG algorithm to manage the continuous state and action spaces inherent in their formulated problem. Meanwhile, in [19], the focus was on device-to-device communication to enable uplink cell-free communication between external users and gNB base stations. An effective deep reinforcement learning (DRL) scheme was introduced to optimize rates for worst-case users and dynamically allocate power for both external and cellular users.

While these studies primarily focused on enhancing QoE as well as video bitrates, especially in the downlink channel, there remains a significant gap in research concerning the optimization of video resolution for live streaming services in multi-user uplink RSMA networks.

3. Problem Statement

We consider a system model consisting of an RSMA-enabled gNB, which provides wireless access for K streamers to simultaneously distribute their content to followers via the mobile network, as shown in Fig. 1. A list of key notations is described in Table 1.

Table 1: List of Notations

Notation	Description
$\mathcal{K}$	The set of all streamers
K	The total number of streamers
s_k	The transmitted message of streamer k
p_{ki}	The transmit power of streamer k
$P_k^{\max}$	The maximum transmit power of streamer k
s_{ki}	The sub-messages received from streamer k
s_0	The total received signal at the gNB
h_k	Channel gain between user k and the gNB
n	The additive white Gaussian noise (AWGN)
π	The decoding order at the gNB
Π	Set of all possible decoding orders of k streams
π_{ki}	The decoding order of sub-messages s_{ki}
r_{ki}	The achievable throughput of s_{ki}
B	The bandwidth of the gNB
σ^2	Power spectral density of the Gaussian noise
r_k	The total uplink achievable rate of streamer k
$\mathcal{V}$	The available set of video bitrates
$\tilde{p}_k$	The content popularity of streamer k
p_k	The popularity of streamer k
f_k	The audience retention rate of stream k
$\mathcal{U}_k, U_k$	Set of currently active followers of stream k
t	Time active followers are viewing the stream k
j	List of followers watching the stream k
d_{kj}	Watching time of follower j
τ	Timeslot duration
t_{kj}	Time when follower j starts to view the stream k
$\bar{V}$	Average video bitrate of all streams k
v_k	Desired maximized video bitrate of stream k
$\mathcal{S}_t, s_t$	State space at time t
$\mathcal{A}_t, a_t$	Action at time t
$\mathcal{P}, \mathcal{R}, \gamma$	Probability, reward, and discount factor at time t
Q	Deep queue learning function

3.1. Channel Model

Let $\mathcal{K} = \{1, 2, \ldots, K\}$ defines the set of K streamers. Without loss of generality, we assume that each message from streamer $k \in \mathcal{K}$ is split into two sub-messages for transmission on RSMA uplinks to the gNB [15]. Let denote $\mathcal{I} = \{s_{k1}, s_{k2}\}$ is the set of sub-messages of s_k. Consequently, the transmitted message s_k from the streamer k is given by

$$s_k = \sum_{i=1}^{2} \sqrt{p_{ki}} s_{ki}, \quad \forall k \in \mathcal{K}, \tag{1}$$

where s_{ki} is the i sub-message with $\mathbb{E}\left[\mid s_{ki} \mid^2\right] = 1$ and p_{ki} is the transmit power of sub-message s_{ki} from streamer k. The total transmission signal power p_{ki} of each streamer k is limited to $P_k^{\max}$ and is restricted by $\sum_{i=1}^{2} p_{ki} \leq P_k^{\max}$. Thus, the power matrix $p_{ki} = \left\{1, 2, \ldots, p_k^{max}\right\}, \forall k \in \mathcal{K}$ is the maximum transmit power of streamer k.

Consequently, the total received signal s_0 obtained at the gNB is given by

$$s_y = \sum_{k=1}^{K} \sqrt{h_k} s_k + \eta = \sum_{k=1}^{K} \sum_{i=1}^{2} \sqrt{h_k p_{ki}} s_{ki} + \eta, \tag{2}$$

where $h_k \in \mathbb{C}^{N \times 1}$ is the gain of the uplink channel between streamer k. Furthermore, gNB and η are additive white Gaussian noise modeled as complex Gaussian random variables with zero mean and uniform variance of σ^2 such as $\eta \sim \mathcal{N}\left(0, \sigma^2\right)$.

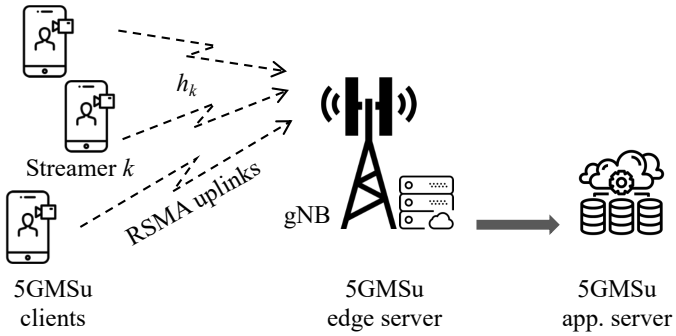


Figure 1: Live streaming services over 3GPP standard 5G media stream uplink (5GMSu) systems.

Adopting RSMA technology, the gNB utilizes the Successive Interference Cancellation (SIC) technique to decode all sub-messages s_{ki} of the streamer k from the received signal s_y. There exists a total $M = K \times I$ sub-messages form K streamers. Assuming that the decoding order of sub-messages at the gNB is denoted as the set $\pi = \{s_{ki} : k \in \mathcal{K}, i \in \mathcal{I}\}$ in which first element is decoded first and second is decoded second, respectively. The decoding order vector at the gNB is denoted by a permutation π belonging to the set Π, which is the set of all possible decoding orders of all sub-messages from K streamers. Let $\pi_{ki} \in \mathcal{M}$ where $\mathcal{M} = \{1, \ldots, M\}$ denotes the decoding order of the submessages s_{ki}. In particular, for the sub-message s_{ki}, the gNB successfully decodes and eliminates all sub-messages that have a lower decoding order than s_{ki} and treats the remaining sub-messages as interference except the sub-message s_{ki}. Therefore, the signal-to-noise ratio (SINR) in the uplink RSMA system for sub-message s_{ki} can be demonstrated as

$$\gamma_{ki} = \frac{h_k p_{ki}}{\sum_{\{(l,m) \in Q_{ki} | \pi_{lm} > \pi_{ki}\}} h_l p_{lm} + \sigma^2} \tag{3}$$

Where $\forall k \in \mathcal{K}$, and $\forall i \in \mathcal{I}$ and Q_{ki} it is worth noting that it is the set of all sub-messages that have a greater decoding order than s_{ki} such as $Q_{ki} = \{(l, m) : \pi_{lm} > \pi_{ki}\}$.

Therefore, the achievable throughput r_{ki} of decoding sub-message s_{ki} can be calculated using the equation (3) as

$$\begin{aligned} r_{ki} &= B \log_2 (1 + \gamma_{ki}) \\ r_{ki} &= B \log_2 \left(1 + \frac{h_k p_{ki}}{\sum_{\{((l,m) \in Q_{ki}) | \pi_{lm} > \pi_{ki}\}} h_l p_{lm} + \sigma^2}\right), \end{aligned} \tag{4}$$

where B is the uplink bandwidth, σ^2 is the power spectral density of the Gaussian noise.

Finally, the uplink throughput of streamer k is given by

$$r_k = \sum_{i=1}^{2} r_{ki}, \qquad \forall k \in \mathcal{K}. \tag{5}$$

3.2. Service Model

As specified in the 3GPP TS 26.501 standard [20], 5G networks natively enable edge computing capability to support

mobile media streaming services, as depicted in Fig. 1. In this architecture, the 5G media stream uplink (5GMSu) edge server located at the gNB manages media sessions established on streamers in terms of content quality (i.e., video bitrate) configuration depending on environmental changes and follower engagement. Let $\mathcal{V} = \{v_1, v_2, \ldots, v^{\max}\}$ denote the set of available video bitrates in the streaming system. Typically, follower engagement can be measured as a combination of two popular metrics, such as content popularity and audience retention rate.

Let $\tilde{p}_k$ denote the content popularity of streamer k in the whole system at the considered timeslot t. In addition, $\tilde{p}_k$ is measured by the central 5GMSu application server as a ratio of the number of followers watching streamer k over the total active followers in the system. In a large-scale system measurement, $\tilde{p}_k$ has been proved following the Zipf distribution [21]. In a small-scale observation at the 5GMSu edge server, the content popularity p_k of streamer k can be normalized as

$$p_k = \frac{\tilde{p}_k}{\sum_{i=1}^{K} \tilde{p}_i}, \quad \forall k \in \mathcal{K}. \tag{6}$$

The popularity of the content is greatly affected by the fame of the streamer and the streaming event. On the other hand, audiovisual quality and content attractiveness are dominant factors in retaining followers to watch the stream continuously. To quantify this observation, the audience retention rate is considered. The audience retention rate f_k of the stream k can be determined by the percentage representing the amount of time followers consume to watch the stream during the current time slot. Let $\mathcal{U}_k$ and U_k denote the set of current active followers of stream k during timeslot t and its cardinality, respectively. In addition, d_{kj} denotes the watching time of follower j, $j \in \mathcal{U}_k$. Furthermore, the view window w_{kj} of follower j is determined as

$$\begin{cases} w_{kj} = \tau & \text{if } t_{kj} \leq \tau t, \\ w_{kj} = \tau(t+1) - t_{kj} & \text{otherwise}, \end{cases} \tag{7}$$

where τ is the timeslot duration and $t_{kj}, 0 \leq t_{kj} < t+1$, is the time point when follower j starts to watch the stream k. Then, f_k is given by

$$f_k = \frac{1}{U_k} \sum_{j=1}^{u_k} \frac{d_{kj}}{w_{kj}}, \quad \forall k \in \mathcal{K}. \tag{8}$$

It is worth noting that f_k is monitored at the central 5GMSu application server during the previous adjacent timeslot. Without loss of generality, the follower engagement of the stream k can be represented by the production of p_k and f_k [22].

3.3. Optimization Problem

Given the current uplink throughput r_k of streamer k, the 5GMSu edge server decides the video bitrate v_k of stream k closest to its upper bound r_k as

$$v_k = v|_{0 \leq \min(r_k - v), \forall v \in \mathcal{V}}. \tag{9}$$

Accordingly, the average video bitrate $\overline{V}$ of all streams managed by the 5GMSu edge server at the gNB is given by

$$\overline{V} = \frac{1}{K} \sum_{k=1}^{K} (v_k p_k f_k). \tag{10}$$

Therefore, the optimization problem ($\mathcal{P}1$) to maximize $\overline{V}$ can be represented as

$$(\mathcal{P}1): \max_{\pi, \{p_{ki}\}} \overline{V} \tag{11}$$

$$\text{subject to (4), (5), (9),}$$

$$\sum_{i=1}^{2} p_{ki} \leq P_k^{\max}, \quad \forall k \in \mathcal{K}, \tag{12}$$

$$v_k \leq r_k, \quad \forall k \in \mathcal{K}, \tag{13}$$

$$v_k \in \mathcal{V}, \pi \in \Pi. \tag{14}$$

Derived from (11), the optimization problem can be reformulated as

$$(\mathcal{P}1) \equiv \max_{\pi, \{p_{ki}\}} \sum_{k=1}^{K} \sum_{i=1}^{2} (p_k f_k \log_2(h_{ki})), \tag{15}$$

subject to (4), (5), (9), (12), (13), and (14). Consequently, the objective function ($\mathcal{P}1$) exhibits a linear proportionality with respect to variations in p_k and f_k, while demonstrating a logarithmic proportionality with respect to variations in h_{ki}. In other words, the parameters p_k and f_k predominantly influence the model's decision-making process for optimizing the system performance. The developed optimization problem is completely consistent with our core idea of exploiting the joint impact of content popularity and audience retention to maximize the average video resolution in live streaming services over RSMA edge networks.

4. Proposed Solution

4.1. Markov Decision Processes Transformation

As the decoding order π is a discrete variable, it is challenging to reach an optimal solution due to the discrete value space. To address this problem, we first reform the decoding order π to continuous variable processes so that we obtain continuous action values. We denote $\Pi_n = \{\phi_{ki} \in (1, \ldots, M), \quad \forall k \in \mathcal{K}\}$ is the set of decoding priority of all sub-signals, where $\phi \in \Pi_n$ is the decoding priority of sub-signal s_{ki} of streamer k.

We consider a standard reinforcement learning setup where an agent interacts with an environment in continuous timesteps. Initially, we translate the problem into a Markov Decision Process (MDP), where the 5GMSu application server serves as the agent. At each timestep t, the agent observes the environment state s_t, decides on an action a_t, and receives a cumulative reward r_t. The agent's objective is to maximize the bitrate for the average streamer, with the remainder of the system comprising the network environment. A conventional Deep Reinforcement Learning (DRL) architecture typically involves five main components: $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$. Here, $\mathcal{S}$

represents the state space, $\mathcal{A}$ denotes the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ defines the transition probability function specifying the likelihood of transitioning from one state to another given an action $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{R}$ represents the continuous reward function where rewards range within $\mathcal{R}_{\max} \in \mathbb{R}^+$ (*e.g.*, $[0, \mathcal{R}_{\max}]$), and $\gamma \in [0, 1)$, used to discount future rewards.

This setup enables the agent, represented by the 5GMSu application server, to autonomously learn an optimal policy to maximize the average streamer's bitrate within the network environment.

State Space. In our environment, the state space s_t comprises all streamers k authorized for wireless access through the RSMA uplink enabled gNB, initiating their live streams at time t. However, network conditions are subject to fluctuation and can vary significantly between environments over time. On the other hand, the 5GMSu edge server evaluates the popularity p_k of each streamer and their retention rate f_k, respectively. Hence, the state environment of our RL model is characterized as:

$$s_t = (h_k, p_k, f_k) \tag{16}$$

Action Space. In our system environment, the action involves decisions aimed at optimizing the average video bitrate $\overline{V}$ for all streamers and ensuring adequate power allocation for video QoE and seamless uplink live stream transmission. This can be formulated as follows:

$$a_t = \{\phi, p_{ki}\}. \tag{17}$$

Transition Probability. A thorough analysis reveals that content popularity p_k and retention rate f_k are critical determinants for the 5GMSu edge server when selecting the action a_t within the state environment s_t. We assume that the channel gain probability of streamer k at each time t is independent. Consequently, the probability of maximizing the average video bitrate and transitioning to the subsequent state s_{t+1} can be expressed by the following state transition function:

$$\mathcal{P}(s_{t+1}|s_t, a_t) = \mathcal{P}(h_{k(t+1)}, p_{k(t+1)}, f_{k(t+1)} \mid h_k, p_k, f_k, \phi, p_{ki}) \tag{18}$$

Here, the state information s_{t+1} for the next iteration $t + 1$ depends only on the current state s_t and the action chosen a_t at time t.

Reword Function. Our optimization objective is to maximize the bitrate of video streaming for average streamers. In other words, our goal is to maximize the long-term cumulative discounted reward r_t after selecting the action a_t in the state environment s_t that can be expressed as

$$\begin{aligned} r_t =& r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ =& \sum_{i=0}^{\infty} \gamma^i r_{t+i} \\ =& \sum_{i=0}^{\infty} \gamma^i \sum_{k=1}^{K} \overline{V}(t+i), \end{aligned} \tag{19}$$

The discount factor, $\gamma \in [0, 1]$, signifies the balance between present and future rewards. As lower $\gamma = 0$, the agent prioritizes immediate rewards. As γ increases, the agent emphasizes future rewards, facilitating more strategic decision-making.

To achieve our goal, we require a policy π which maps states to actions ($\pi : \mathcal{S} \rightarrow \mathcal{A}$). The policy π guides the agent by specifying the action a_t to be taken in state s_t to maximize the expected reward r_t. Maximizing the objective function is therefore equivalent to identifying the optimal policy, denoted π^*.

In conventional RL problems, the Q-value function, often denoted as $Q(st, a_t)$, plays a crucial role [23]. This function represents the expected accumulated reward starting from state s_t, taking action a_t, and then following the policy π, is instrumental in solving RL problems and can be expressed as follows:

$$\begin{aligned} Q_\pi(s_t, a_t \mid \theta^Q) &= \mathbb{E}[R_t|s_t, a_t] = \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i}|s_t, a_t\right] \\ &= \mathbb{E}[r_t + \gamma Q_\pi(s_{t+1}, a_{t+1})|s_t, a_t]. \end{aligned} \tag{20}$$

The corresponding optimal Q value function $Q_{\pi^*}\left(s_t, a_t \mid \theta^Q\right)$ is the function that has the highest Q value and it follows the optimal policy π^*. Therefore, this recursive relationship, known as the Bellman equation, can be expressed mathematically as follows:

$$\begin{aligned} Q_{\pi^*}\left(s_t, a_t \mid \theta^Q\right) &= \max_\pi Q_\pi(s_t, a_t) \\ &= \max_\pi \mathbb{E}\left[R_t|s_t, a_t\right] \\ &= \max_\pi \mathbb{E}\left[r_t + \gamma R_{t+1}|s_t, a_t\right] \\ &= \max_\pi \mathbb{E}\left[r_t + \gamma Q_\pi(s_{t+1}, a_{t+1})|s_t, a_t\right] \\ &= \mathbb{E}\left[r_t + \gamma \max_{a_{t+1}} Q_{\pi^*}(s_{t+1}, a_{t+1})|s_t, a_t\right] \end{aligned} \tag{21}$$

By formulating such a reward function, the RL agent is incentivized to maximize the bitrate for average streamers, leading to higher rewards. In contrast, it faces penalties, possibly receiving zero rewards, for policies that fail to meet system requirements. This mechanism encourages the RL agent to select policies that achieve the highest bitrate v_k while avoiding those that do not satisfy minimum bitrate requirements. Therefore, the reward function r_t, as described in 19, can be summarized as follows:

$$r_t(s_t, a_t) = \sum_{k=1}^{K} \left(\overline{V} - \omega\right). \tag{22}$$

We introduce ω as the penalty incurred for actions that violate the optimization constraints. This constraint ensures that the maximum bitrate $v_k \in \mathcal{V}$ never exceeds the available throughput r_k.

A deep neural network (DNN) is well-suited to address our optimization problem because it efficiently approximates complex mapping functions using a fixed number of parameters. This stability in parameter count is advantageous regardless of the state and action space size, making it particularly effective for solving continuous problems like the DDPG.

4.2. Our Proposed DDPG-BARMAS Solution

To address the defined MDP model, we propose the structure of our proposed DDPG-BARMAS algorithmic framework. The BARMAS algorithm tailored for an RSMA-enabled uplink live streaming service, adeptly manages large continuous state and action spaces that are time-varying. Moreover, our proposed DDPG-based BARMAS algorithm is a model-free, off-policy DRL method, originally detailed in [24]. It utilizes two principal sets of Deep Neural Networks (DNNs): an actor network and a critic network.

The actor network selects actions based on the existing state s_t, while the critic network evaluates the actions a_t, generated by the actor network. Specifically, we initialize a critic network $Q(s, a; \theta_Q)$ with parameter θ_Q and an actor network $\mu\left(s_t; \theta_\mu\right)$ with parameter θ_μ. The actor network is tasked with selecting deterministic actions for each input state s_t. In contrast, the critic network assesses these actions and provides feedback to the actor network in the form of a gradient signal, which guides the actor's learning process. Furthermore, the DDPG framework employs an experienced replay buffer and target networks. These include a target actor network $\mu'\left(s_t; \theta_{\mu'}\right)$ with parameter $\theta_{Q'}$ and a target critic network $Q'(s, a; \theta_{Q'})$ with parameter $\theta_{\mu'}$. The experience replay buffer stores transition for use during training, while the target networks help reduce the correlation of the training data and enhance training stability. The algorithm framework is illustrated in Fig. 2.

To regulate the exploration procedure, the actor network $\mu\left(s \mid \theta_\mu\right)$ is augmented with noise to generate action a_t as follows:

$$a_t = \mu\left(s_t; \theta_\mu\right) + \mathcal{N}_t(0, \sigma), \tag{23}$$

where $\mathcal{N}_t$ represents the Gaussian noise (GN) added to the action to encourage exploration.

Moreover, at each time step t, the agent observes the state of the environment s_t and executes the action a_t according to the policy π^*. The environment then provides an immediate reward r_t and the next state s_{t+1} following the action a_t. As the actor network processes the state s_t and the action a_t to obtain the rewords r_t, a new state s_{t+1} is immediately generated post action execution. According to the exploration policy, the transition tuple (s_t, a_t, r_t, s_{t+1}) is stored in the central experience relay buffer $\mathcal{B}$, which is used to update the network parameters. A randomly sampled mini-batch of $\mathcal{D}$ transition tuples is extracted from $\mathcal{B}$ to update the actor and critic networks. Once the replay buffer reaches its capacity, older samples are discarded, and the actor and critic networks begin their gradual learning process.

During the training process, the parameters of the critic network are adjusted by minimizing the loss function between the action value function $Q(s_i, a_i; \theta_Q)$ and the target value y_i. This loss function is expressed as:

$$L(\theta_Q) = \frac{1}{N} \sum_i^N \left(y_i - Q(s_i, a_i; \theta_Q)\right)^2, \tag{24}$$

where $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}; \theta_{\mu'}\right); \theta_{Q'}\right)$. Here, N represents the number of samples in the mini-batch $\mathcal{D}$ which is randomly selected from the experience replay buffer $\mathcal{B}$. Each sample (s_i, a_i, r_i, s_{i+1}) represents a sample i, that includes the state s_t, action a_t, reward r_t, and the next state s_{t+1}.

The actor network is updated using the deterministic policy gradient method, using the gradient information from the critic network. This is expressed as follows:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i^N \nabla_a Q\left(s, a \left|\theta_Q\right.\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu\left(s \left|\theta_\mu\right.\right)\Big|_{s_i}. \tag{25}$$

To improve training stability, a soft update method is employed to update the parameters of the target networks using a small constant $\epsilon \ll 1$. This process is formulated as follows:

$$\begin{aligned} \theta_{\mu'} &\leftarrow \epsilon\theta_\mu + (1-\epsilon)\,\theta_{\mu'} \\ \theta_{Q'} &\leftarrow \epsilon\theta_Q + (1-\epsilon)\,\theta_{Q'} \end{aligned} \tag{26}$$

During the execution phase, the critic networks are inactive and only the actor network is utilized. The trained actor network generates actions based on the current state s_t observed by the agent in the 5GMSu RSMA-assisted uplink live streaming system, as illustrated in Fig. 2 and Algorithm 1.

In Algorithm 1, the steps begin at a specific time t. Each time, an agent conducts an observational investigation into the state environment s_t. Based on the agent's observation, the action a_t is determined using the policy π. The actor network $\mu\left(s; \theta_\mu\right)$ with parameter θ_μ determines the action a_t, subsequently creating a list of tuples as $(h_k, p_k, f_k, \phi, p_{ki})$. The power allocation p_{ki} and total achievable uplink throughput r_k of streamer k are calculated. In addition, the popularity of streamer content p_k and the audience retention rate f_k are evaluated by the 5GMSu edge server.

Furthermore, the critic network $Q\left(s, a; \theta^Q\right)$ is initialized with random weights θ_Q. We establish a target actor network $\mu'\left(s; \theta_{\mu'}\right)$ with parameter $\theta_{\mu'}$ and a target critic network $Q'(s, a; \theta_{Q'})$ with parameter $\theta_{Q'}$. An experience replay buffer $\mathcal{B}$ is then created to store the transitions during the training phase when necessary. In the aforementioned second step, the agent (5GMSu edge server) interacts with the state environment s_t to collect training data, which are stored in the replay buffer of the experience $\mathcal{B}$.

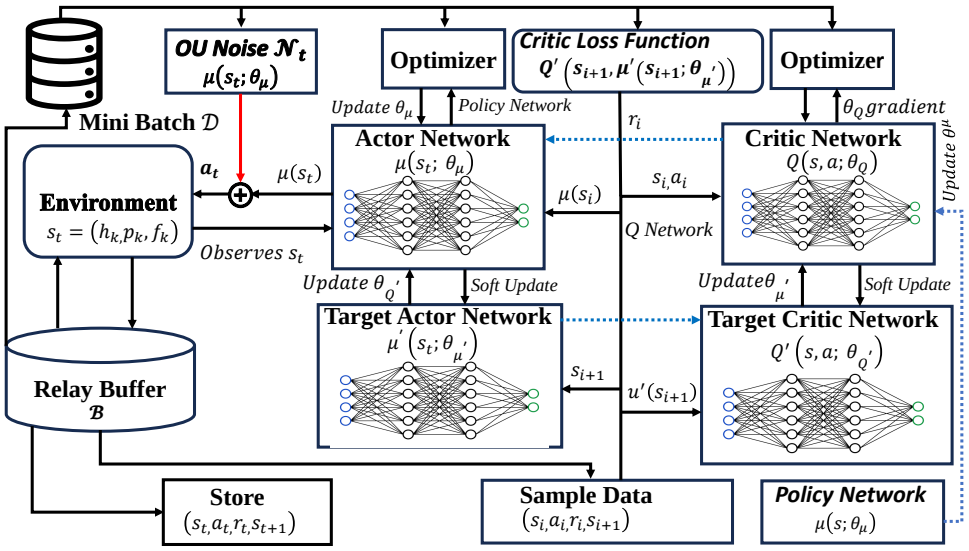


Figure 2: DDPG-BARMAS Algorithm Framework.

Algorithm 1 The proposed DDPG-BRMAS

Initialize actor network $\mu\left(s_t;\theta_\mu\right)$ and critic network $Q\left(s,a;\theta_Q\right)$ with weight parameters θ_μ and θ_Q.
Initialize target networks μ' and Q' with weight parameters $\theta_{\mu'} \leftarrow \theta_\mu, \theta_{Q'} \leftarrow \theta_Q$.
Establish experience replay buffer $\mathcal{B}$.
for *episode* $\in \{1,\ldots,M\}$ **do**
  Obtain initial observation state s_1.
  for $t \in \{1,\ldots,T\}$ **do**
    Select action with exploration noise:

$$a_t = \mu\left(s_t;\theta_\mu\right) + \mathcal{N}_t\left(0,\sigma\right).$$

    Execute action a_t, receive a reward r_t, and move to the new state s_{t+1}.
    Store transition (s_t, a_t, r_t, s_{t+1}) in the replay buffer $\mathcal{B}$.
    Update state: $s_t \leftarrow s_{t+1}$.
    Arbitrarily sample a mini-batch of $\mathcal{D}$ transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{B}$.
    Compute the critic loss function:

$$L\left(\theta^Q\right) = \frac{1}{N}\sum_i \left(y_i - Q\left(s_i,a_i;\theta^Q\right)\right)^2,$$

    where $y_i = r_i + \gamma Q'\left(s_{i+1},\mu'\left(s_{i+1}\mid\theta^{\mu'}\right)\mid\theta^{Q'}\right)$.
    Update the parameters of the critic network θ^Q using backpropagation.
    Compute the actor policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s,a;\theta^Q\right)\Big|_{s=s_i,a=\mu(s_i;\theta^\mu)} \nabla_{\theta^\mu}\mu\left(s;\theta^\mu\right)\Big|_{s=s_i}.$$

    Update the actor network parameters θ^μ using the computed gradient.
    Perform soft updates to the target networks:

$$\theta^{\mu'} \leftarrow \epsilon\theta^\mu + (1-\epsilon)\theta^{\mu'}, \quad \theta^{Q'} \leftarrow \epsilon\theta^Q + (1-\epsilon)\theta^{Q'}.$$

  end for
end for
return the trained actor network $\mu\left(s;\theta_\mu\right)$.

Finally, in the third step, a mini-batch $\mathcal{D}$ of samples is randomly selected from the replay buffer of the experience $\mathcal{B}$ to train the network. The algorithm then uses the loss function $L(\theta_Q)$ and the policy gradient ascent method to update the parameters of the critic and actor networks, respectively. To ensure training stability and prevent divergence, the target network's parameters are softly updated using a constant value ϵ. However, the overall training process concludes once the desired number of episodes has been reached. This results in a well-trained actor network that can be effectively utilized during the online execution phase.

4.3. Computational Complexity Analysis

In this section, we analyze the complexity of optimizing the RSMA-aided uplink live streaming bitrate maximization problem using the proposed DDPG-BARMAS algorithm. Specifically, the complexity of the algorithm is determined based on the training of neural networks within a framework that employs fully connected deep neural networks (DNNs) comprising an input layer, two hidden layers, and an output layer. The computational complexity of each training step is influenced by several factors, including the size of the input state and action, the number of layers, and the number of neurons in each layer of the DNNs [25]. During training, the BARMAS algorithm utilizes a finite number of DNNs and requires $M \times I \times T$ iterations to complete the training phase. Here, M denotes the mini-batch size used for each update, I represents the total number of training episodes, and T indicates the number of steps taken in each episode.

Additionally, we define N to be the size of the mini-batch used for each update, X_μ, and Y_Q as the number of layers in the critic network. Let $n^\mu_{l_\mu}$ and $n^Q_{l_Q}$ denote the number of neurons in the l_μ-th layer of the actor network and the l_Q-th layer of the critic network, respectively. Consequently, the overall computational complexity during the training phase can be expressed as:

$$Complexity = O\left(M \times I \times T\left(\sum_{l_\mu=0}^{X_\mu-1} n_{l_\mu}^{\mu} n_{l_\mu+1}^{\mu} + \sum_{l_Q=0}^{Y_Q-1} n_{l_Q}^{Q} n_{l_Q+1}^{Q}\right)\right). \tag{27}$$

In the execution phase, the agent's operations are streamlined by relying solely on the policy function derived from the trained model. Consequently, the computational complexity for each time interval is given by $O\left(\sum_{l_\mu=0}^{X_\mu-1} n_{l_\mu}^{\mu} n_{l_\mu+1}^{\mu}\right)$.

The proposed framework leverages the DDPG algorithm, which is known as notably computationally intensive however, once the model has been effectively trained, it facilitates real-time inference. Consequently, DDPG-BARMAS represents an off-policy actor-critic approach that operates with a deterministic policy. This means that upon receiving a state s_t, the framework directly computes and outputs the corresponding action a_t. Such a mechanism is particularly well-suited for time-sensitive applications where swift action computation is crucial.

5. Performance Evaluation

In this section, we present a detailed examination of our optimization framework's efficacy through a multi-faceted simulation study. First, we outline the simulation parameters and the experimental setup. Next, we perform a rigorous convergence analysis to assess the stability and efficiency of the framework. Finally, we conclude with a comparative performance evaluation, contrasting our framework with several baseline methods to underscore its advantages and effectiveness.

5.1. Simulation Configuration and Environment Setup

We configured the simulation within the Anaconda environment, utilizing PyTorch to visually illustrate our proposed DDPG-BARMAS algorithm, designed to optimize bitrate for average streamers in live video streaming services [26]. The simulation was executed on a GPU-equipped server featuring an NVIDIA GeForce RTX 3090 Ti, paired with an Intel Core i9-12900K CPU running at 3.20 GHz and 64 GB of RAM.

In the live video streaming environment, we have configured a single gNB serving $k \in \mathcal{K}$ streamers who transmit uplink signals for live video streaming on an RSMA-enabled gNB. The gNB is equipped with 4 antennas, providing simultaneous live streaming services to 10 users. We assigned a bandwidth of B = 1 MHz to assess the algorithm's performance under constrained bandwidth conditions, thereby avoiding any bias towards consistently high available rates, which could otherwise lead to uniformly maximum bitrate for every streamer. The noise variance σ^2 at the transmitter is selected to be -170 dBm/Hz, while the total power P_{ki} allocated by the gNB for each streamer is 23 dBm. Furthermore, we modeled the buffer size β as $t = 0$ at the start, indicating an empty buffer due to the absence of live video uploads on the uplink network channel. Before proceeding with the analysis of streamer popularity p_k and retention rates f_k, it was assumed that all streamers start with equal bitrates. The maximum supported video bitrate codec was assumed to be 22 *Mbps* while the minimum was set at *100 Kbps*. Based on the guidelines of the YouTube Live Streaming Bitrate Selection Procedure [27], we categorized live video streaming bitrates into 10 distinct video quality levels, including 360p, 480p, 720p, 720p @ 60fps, 1080p, 1080p @ 60fps, 1440p @ 30fps, 1440p @ 60fps, 4K @ 30fps, and 4K @ 60fps. For instance, the bitrate requirements for 720p @ 60fps and 1080p fall below 2.25 and 3 Mbps, respectively. Simulation parameters are described in Table 2.

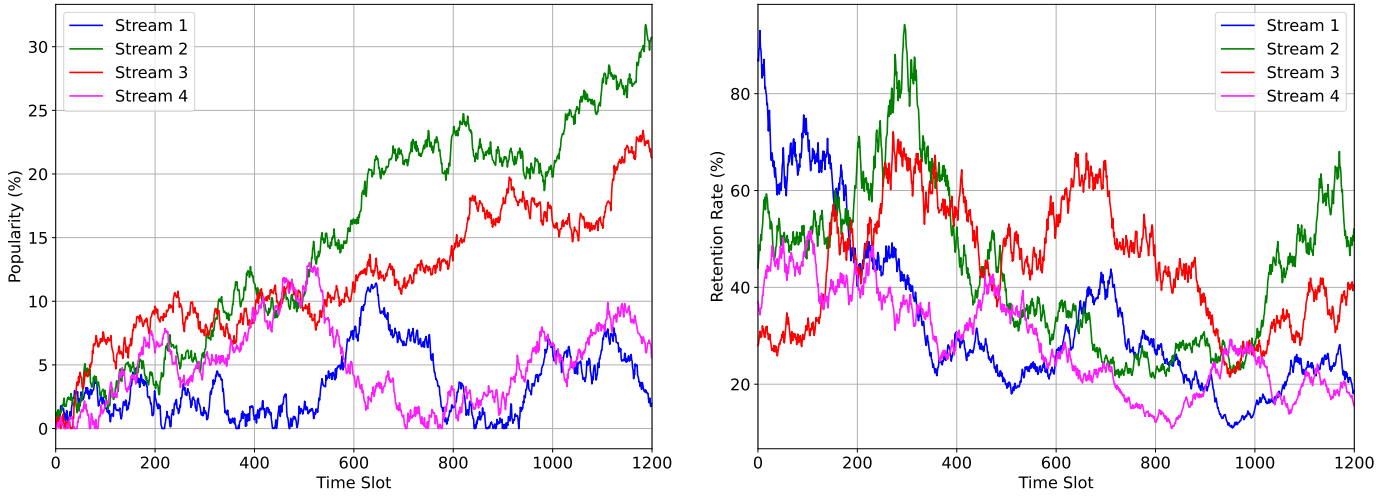
Table 2: Simulation Configuration Parameters

Parameters	Value
The number of antennas on gNB	4
The total number of streamers	10
The bandwidth B of the base station gNB	1 MHz
Uplink power p_{ki}	23 dBm
Gaussian Noise Variance σ^2	-170 dBm/Hz
SIC error rates	0.1
Timeslot duration τ	3 Seconds
Number of training episodes E	2000
Number of steps per episode S	700
Actor hidden layer nodes n_1 & n_2	1024 and 512
Critic hidden layer nodes n_1 & n_2	512 and 256

In our experiment, we anonymized user behaviors by mimicking traffic patterns of popular YouTube channels as introduced in the description of YouTube Analytics through YouTube Studio of Google Developer [28] for privacy protection. As a result, the considered dataset of live video stream traffic over mobile networks exhibits significant heterogeneity due to unpredictable user engagement and departures, which contribute substantially to network traffic. A sample capture of the popularity and retention rate of 4 streamers is illustrated in Fig. 3.

5.2. Analysis of Convergence Dynamics

We assessed and compared the efficacy of different bitrate allocation schemes through Bellman simulations over 2000 episodes. Consequently, we investigated the influence of hyperparameter iterations on our proposed algorithm through statistical analysis of training reward variations. The training regimen began with different mini-batch sizes, denoted as $\mathcal{D}$. Specifically, we experimented with three batch sizes: $\mathcal{D} = 32$, $\mathcal{D} = 64$, and $\mathcal{D} = 128$. The training with a batch size of $\mathcal{D} = 32$, achieved convergence around 800 episodes. For more rapid convergence, a larger batch size of approximately $\mathcal{D} = 128$ was used, resulting in convergence after approximately 300 episodes. However, during training, the reward for $\mathcal{D} = 128$ exhibited notable fluctuations, a consequence of the stochastic nature of the process and the lack of exploration inherent to a larger batch size. In episodes 2000 with $\mathcal{D} = 128$ was approximately 11500, which was no more than 45.7% worse compared to $D = 32$ and 46.87% worse compared to $\mathcal{D} = 64$. Furthermore, the iterative convergence for $\mathcal{D} = 64$ was relatively



(A) Popularity Index

(B) Retention Metrics

Figure 3: A capture of live stream popularity and retention rate.

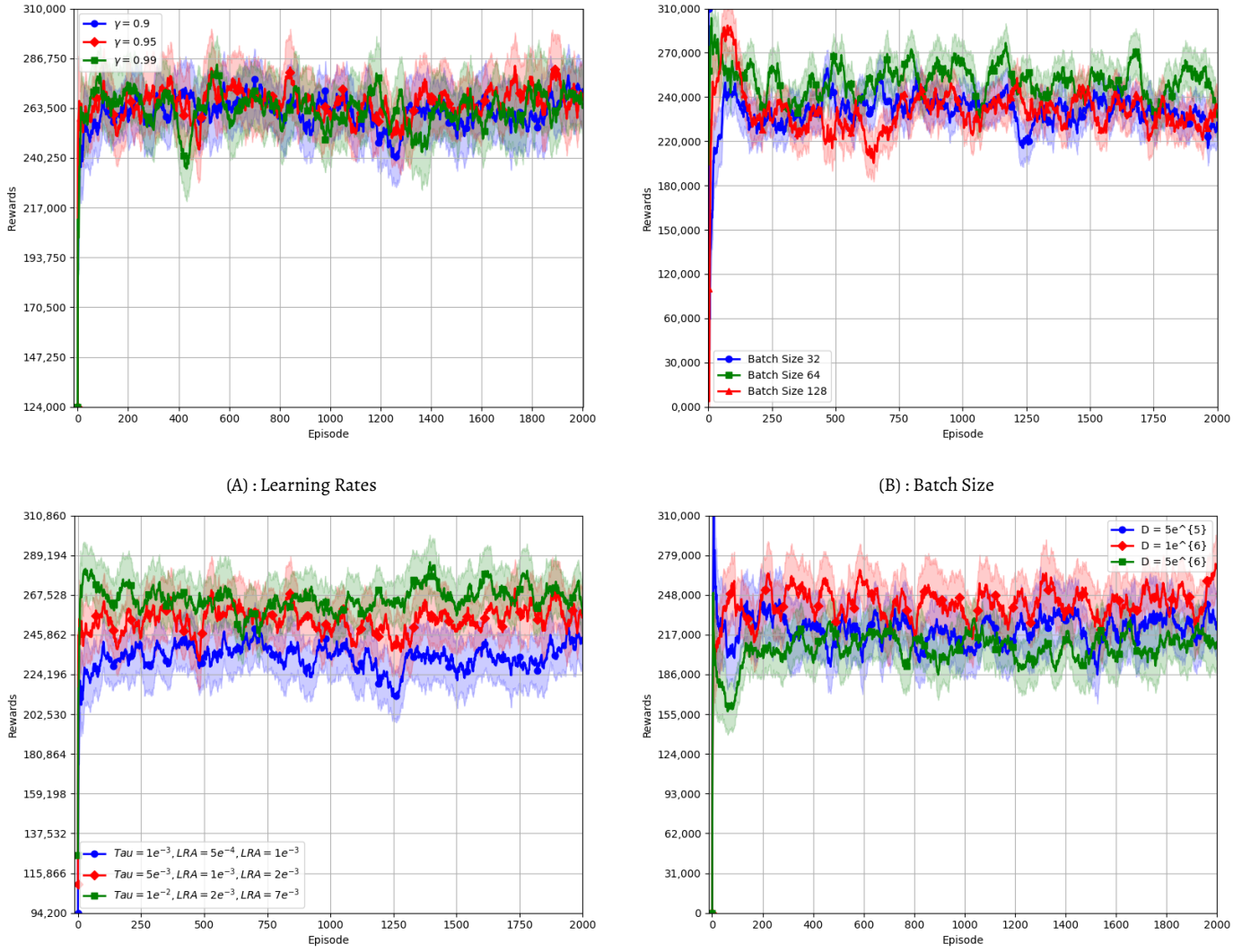
faster, with convergence occurring in around 400 episodes. Despite this, $\mathcal{D} = 64$ achieved the highest final training reward among the three batch sizes tested.

As illustrated in Fig. 4A, the learning rates for the actor network (ANc) and the critic network (CNc) are segmented into three distinct sectors within our system, specifically defined as $(ANc, CNc) = 0.9; 0.95; 0.99$, respectively. The learning rate reward demonstrates optimal performance after several hundred episodes. Although initial convergence across all scenarios exhibits similar performance, the case where $ANc, CNc) = (1e^{-4}, 5e^{-4})$ ensures stable simulation results relative to the others, achieving a steady range after approximately 400 episodes and obtaining the highest reward value among these observations.

The impact of the discount factor γ on the system as depicted in Fig. 4B, achieves the highest performance when γ is set to 0.9, 95, or 0.99. In contrast, when γ is excessively large or diminutive, overall performance deteriorates. This phenomenon can be attributed to the formulation of the Bellman equation, where the precise expression for the long-term average is $\gamma = 1$. As γ approaches 1, the live streaming network model for uplink prioritizes future rewards, thus accelerating their accumulation rather than focusing solely on short-term gains. Consequently, if γ is too small, the training policy prioritizes immediate rewards over long-term maximization of bitrates for average streamers. However, the performance can also degrade if $\gamma = 1 - 1e - 1.5$. This degradation occurs because a single uplink streaming channel struggles to accurately represent long-term behavior. When γ is extremely close to 1, the Q-table interprets the data from a single bitrate streaming session as long-term data, leading to poor generalization across different streamer's bitrates. Therefore, we trained our algorithm over numerous episodes with a suitable range of γ values to sustainably enhance the long-term performance of our policy. In our live streaming scenario, the discount factor γ remained largely stable and consistent throughout the training period, achieving convergence at approximately $\gamma = 0.95$. This discount factor, evaluated over roughly 400 episodes, displayed fewer oscillations in subsequent episodes compared to the initial penalty rate, ensuring robust and stable performance.

The simulation output in Fig. 4C, elucidates the evaluation of the actor-network learning rate, the critic-network learning rate, and the soft target update rates, respectively. It demonstrates the DDPG-BARMAS algorithm while considering various mini-batch sizes $\mathcal{D}$ and the learning rate of actor and critic network, θ_u and θ_Q, respectively. According to Fig. 4, we used a small mini-batch size, as our proposed algorithm does not require extended periods to reach the optimal trained policy. However, learning progress can be impeded, and local optima might be encountered if the batch size is excessively large. Although gradient evaluation with a larger batch size is more precise, it may result in slower convergence. Furthermore, the learning rate significantly influences the update of neural network parameters, determining the convergence speed and stability of our proposed algorithm. To illustrate the impact of the learning rates of the actor θ^{μ} and the critic θ^{Q}, as well as the soft target update rate on our network system architecture, we combined these factors in various ratios: (Tau, ALR, CRL) = $\left(5e^{-3}, 1e^{-3}, 2e^{-3}\right)$; $\left(1e^{-2}, 2e^{-3}, 7e^{-3}\right)$; and $\left(1e^{-3}, 5e^{-3}, 1e^{-3}\right)$, respectively. Initial observations reveal that higher learning rates accelerate the convergence of the network module. For example, with a soft target update rate Tau $= 1e^{-2}$ the model converges after approximately 250 episodes, although the training rewards at convergence are at the lower bound. In contrast, with Tau $= 1e^{-3}$ the model converges after roughly 450 episodes, achieving the highest training rewards. However, the cumulative reward decreases as training progresses. Convergence after 350 episodes, observed with Tau $= 5e^{-3}$, exhibits satisfactory results with minor fluctuations, indicating the effectiveness of this rate.

The variation in the rewards observed at different rates led us to select the moderate learning rate set $\left(5e^{-3}, 1e^{-3}, 2e^{-3}\right)$ as



(A) : Learning Rates

(B) : Batch Size

(C) : Training Episodes

(D) : Replay Buffer

Figure 4: Convergence analyses of the proposed algorithm.

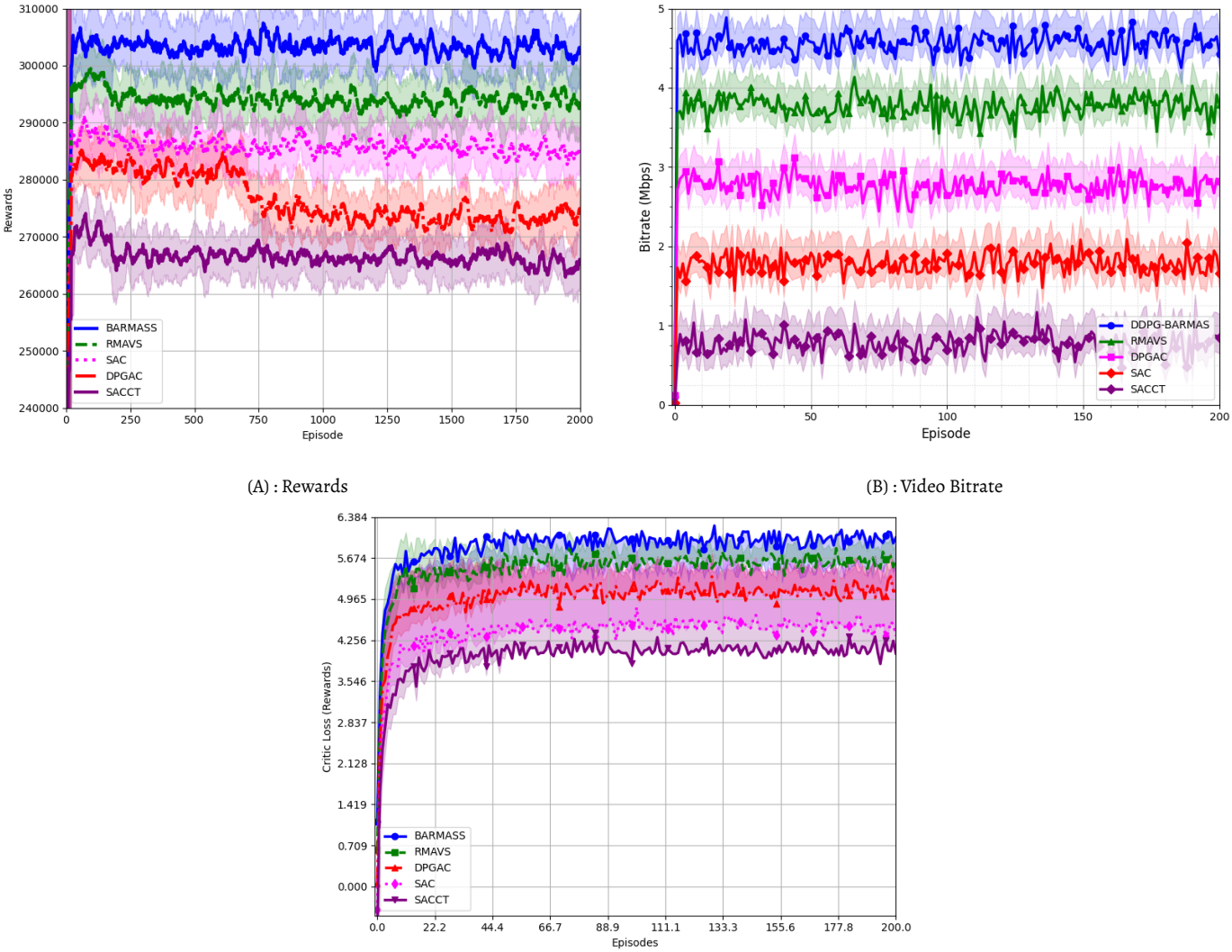
the optimal choice for our proposed algorithm. The buffer replay size was configured to 400 tuples with a batch size of 32. Throughout the training episodes, the rate of the replay buffer $\mathcal{B}$ varied periodically. As shown in Fig. 4D, the system module used buffer sizes of $\mathcal{B} = \mathcal{B} = \left\{5e^5; 1e^6; 5e^6\right\}$ to evaluate three different training rewards. The Simulation results indicate that a buffer size of $\mathcal{B} = 5e^5$most frequently led to convergence, while training rewards generally declined as iterations progressed. Furthermore, buffer sizes of $\mathcal{B} = 1e^6$ and $\mathcal{B} = 5e^6$ showed similar performance around 320 episodes. Despite the larger buffer sizes, $5e^6$ exhibited an upward trend in achieving rewards, but was the least stable, similar to $\mathcal{B} = 1e^6$.

To summarize, the optimal hyperparameters determined for our DDPG-BARMAS algorithm are as follows: convergence batch size $\mathcal{D} = 64$, cumulative discounted value $\gamma = 0.95$, learning rates Tau $= 5e^{-3}$, LRA $= 1e^3$, CRL $= 2e^{-3}$, and replay buffer size is $\mathcal{B} = 5e^6$.

5.3. *Comprehensive Performance Analysis*

In this section, we assess the efficacy of our proposed DDPG-BARMAS algorithm. The simulated scenarios assume an arbitrary number of streamers connected to the 5GMSu edge server at the gNB. To demonstrate the effectiveness of our proposed algorithm, we conducted a performance evaluation with the following components:

- DDPG on RSMA-Based Video Streaming System (DDPG-RMAVS): This actor-critic algorithm is engineered for managing QoE in Internet of Multimedia Things (IoMT) traffic management [29]. It boasts advantages such as improved video quality, reduced latency, and enhanced buffer capacity through its continuous learning action space and reward maximization.
- Soft Actor-Critic (SAC) Constrained DRL for Energy Efficiency Optimization in RSMA-Based Integrated Satellite Communication: This optimization technique addresses



(A) : Rewards

(B) : Video Bitrate

(C) : Q Loss

Figure 5: Comparative Performance Analysis of Reward, Bitrate, and Q-Loss metrics.

QoS requirements within a continuous action space, utilizing the SAC algorithm [30].

- Distributed DDPG for Power Allocation Control in-vehicle communication (DPGAC) [31]: This technique leverages distributed DDPG and shared DDPG approaches to address power allocation issues in in-vehicle infotainment communication within a multi-agent environment. The approach utilizes a continuous action space with actor-critic methods.

- DRL-Based Communication Transformer for Adaptive Live Streaming in Wireless Edge Networks (SACCT) [32]: This algorithm optimizes video follower engagement and QoE while reducing energy consumption through uplink transmission and edge transcoding.

To keep the fairness at most, all used DDPG models have a similar configuration. Meanwhile, the DPGAC algorithm features actor and critic networks with one input layer, one output layer, and a single hidden layer of 100 units, trained using Adam optimization, with parameters initialized from a zero-mean Gaussian distribution. In contrast, the SAC algorithm employs three fully connected hidden layers with ReLU activation and Adam optimization. The soft value network includes two hidden layers of 64 neurons each and a final layer with up to one neuron. The policy network and soft Q network in SAC also consist of hidden layers with 64 neurons.

Fig. 5A, illustrates the cumulative training rewards over 2000 episodes for various algorithms. Among the algorithms evaluated, DDPG-BARMAS exhibited superior and consistent performance, achieving an average reward of approximately 308000 per training episode. As depicted in Fig. 5, DDPG-BARMAS achieved approximately 97%, significantly outperforming DDPG-RMVAS and SAC, which achieved overall performance levels of approximately 80% and 63%, respectively. In contrast, the performance of the DPGAC initially dropped to

less than 45%, while SACCT showed notable variability, with performance oscillating dramatically to a trough of approximately 29% during episodic learning.

Fig. 5B illustrates the video bitrate maximization performance of five algorithms *DDPG-BARMAS*, RMAVS, DPGAC, SAC, and SACCT over 200 episodes, with bitrates ranging from 0 to 5 Mbps. DDPG-BARMAS maintains the highest bitrate at approximately 4.7 Mbps, which is nearly 97% with moderate fluctuations, indicating strong video quality and stability. On the other hand, RMAVS achieves a slightly lower bitrate of about 4.5 Mbps with a comparable about 80% bitrate on a slight oscillation. Whereas DPGAC operates at a midrange bitrate of 3.3 Mbps 63% but exhibits moderate fluctuations, suggesting a balance between performance and resource efficiency, SAC delivers a bitrate around 2.6 Mbps 45% with significant fluctuations, reflecting instability in maintaining quality. Lastly, the SACCT has the lowest bitrate just near 1.9 Mbps 29% but with relatively contained fluctuations, indicating optimization for constrained bandwidth environments. This comparison underscores the trade-offs between bitrate performance and stability, highlighting that higher bitrates correlate with better video quality but increased fluctuations, while lower bitrates show more significant fluctuations, impacting user experience in bandwidth-limited contexts.

In our comprehensive analysis of various Q-loss functions, depicted in Fig. 5C, our DDPG-BARMAS algorithm demonstrates significantly outstanding performance. This algorithm impressively exhibits approximately 17% lower Q loss compared to DDPG-RMAVAS, with a nearly 20% reduction relative to SAC, and an astonishing less 37% reduction when juxtaposed with both DDDPG and SACCT. The Q-loss reduction rate for DDPG-BARMAS hovers around 3%, and is close to 17% less compared to DDPG-RMAVAS. In stark contrast, SAC's Q-loss is nearly threefold higher, culminating in an overall Q-loss exceeding 45%. Moreover, both DDPG and SACCT experienced a markedly upward trajectory in Q-loss, stabilizing at a high of 29% throughout the training session. This comparative analysis underscores the remarkable efficacy of the DDPG-BARMAS algorithm. It consistently achieves a Q-loss reduction of approximately just 30% relative to any other algorithm evaluated. Hence, we can conclusively determine that DDPG-BARMAS is the most advantageous algorithm due to its consistent and substantial minimization of Q-loss values.

6. Concluding Remarks

In this study, we tackled the bitrate maximization challenge for average streamers in an RSMA-enhanced uplink live streaming system by reformulating the problem as an MDP and designing a DDPG-based BARMAS optimization framework to maximize streamer bitrates as well as improve QoE. To protect user privacy, the framework considered anonymized traffic patterns of the live stream popularities and their retention rates while user behaviors are unrevealed. While the framework exhibited superior performance in both training and evaluation across various scenarios, future research could explore the integration of heterogeneous uplink networks with multiple streamers and platforms to create a more realistic simulation environment, which should be validated by a testbed for enhanced real-world uplink live streaming support. Additionally, a thorough investigation into extending the proposed solution to existing wireless networks, such as NOMA and OFDMA, should be undertaken.

References

[1] C.-C. Chen, Y.-C. Lin, What drives live-stream usage intention? the perspectives of flow, entertainment, social interaction, and endorsement, Telematics and Informatics 35 (1) (2018) 293–303. doi:10.1016/j.tele.2017.12.003.

[2] R. Trestian, I.-S. Comsa, M. F. Tuysuz, Seamless multimedia delivery within a heterogeneous wireless networks environment: Are we there yet?, IEEE Communications Surveys & Tutorials 20 (2) (2018) 945–977. doi:10.1109/COMST.2018.2789722.

[3] M. Taha, A. Ali, J. Lloret, P. R. Gondim, A. Canovas, An automated model for the assessment of qoe of adaptive video streaming over wireless networks, Multimedia Tools and Applications 80 (17) (2021) 26833–26854. doi:10.1007/s11042-021-10934-9.

[4] M. Taha, A. Ali, Smart algorithm in wireless networks for video streaming based on adaptive quantization, Concurrency and Computation: Practice and Experience 35 (9) (2023) e7633. doi:10.1002/cpe.7633.

[5] B. Clerckx, Y. Mao, Z. Yang, M. Chen, A. Alkhateeb, L. Liu, M. Qiu, J. Yuan, V. W. Wong, J. Montojo, Multiple access techniques for intelligent and multifunctional 6g: Tutorial, survey, and outlook, Proceedings of the IEEE 112 (7) (2024) 832–879. doi:10.1109/JPROC.2024.3409428.

[6] X. Lyu, S. Aditya, B. Clerckx, Rate-splitting multiple access for overloaded multi-group multicast: A first experimental study, IEEE Transactions on Broadcasting 71 (1) (2025) 30–41. doi:10.1109/TBC.2024.3475743.

[7] S. Zhang, Y. Mao, B. Clerckx, T. Q. Quek, Interference management in space-air-ground integrated networks with fully distributed rate-splitting multiple access, IEEE Transactions on Wireless Communications 24 (1) (2025) 149–164. doi:10.1109/TWC.2024.3489219.

[8] J. Xu, O. Dizdar, B. Clerckx, Rate-splitting multiple access for short-packet uplink communications: A finite blocklength analysis, IEEE Communications Letters 27 (2) (2023) 517–521. doi:10.1109/LCOMM.2022.3226817.

[9] N.-N. Dao, D. T. Ngo, N.-T. Dinh, T. V. Phan, N. D. Vo, S. Cho, T. Braun, Hit ratio and content quality tradeoff for adaptive bitrate streaming in edge caching systems, IEEE Systems Journal 15 (4) (2021) 5094–5097. doi:10.1109/JSYST.2020.3019035.

[10] N.-N. Dao, D.-N. Vu, W. Na, T.-M. Hoang, D.-T. Do, S. Cho, Adaptive bitrate streaming in multi-user downlink noma edge caching systems with imperfect sic, Computer Networks 212 (2022) 109064. doi:10.1016/j.comnet.2022.109064.

[11] X. Ma, Q. Li, L. Zou, J. Peng, J. Zhou, J. Chai, Y. Jiang, G.-M. Muntean, Qava: Qoe-aware adaptive video bitrate aggregation for http live streaming based on smart edge computing, IEEE Transactions on Broadcasting 68 (3) (2022) 661–676. doi:10.1109/TBC.2022.3171131.

[12] J. Luo, F. R. Yu, Q. Chen, L. Tang, Adaptive video streaming with edge caching and video transcoding over software-defined mobile networks: A deep reinforcement learning approach, IEEE Transactions on Wireless Communications 19 (3) (2020) 1577–1592. doi:10.1109/TWC.2019.2955129.

[13] F. Fu, Y. Kang, Z. Zhang, F. R. Yu, T. Wu, Soft actorThfcritic drl for live transcoding and streaming in vehicular fog-computing-enabled iov, IEEE Internet of Things Journal 8 (3) (2021) 1308–1321. doi:10.1109/JIOT.2020.3003398.

[14] X. Jiang, Y. Zhang, Perceptual content-aware bitrate adaptation for http streaming using markov decision process, in: 2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall), 2021, pp. 227–231. doi:10.1109/ICISFall51598.2021.9627473.

[15] T. P. Truong, N.-N. Dao, S. Cho, Hamec-rsma: Enhanced aerial computing systems with rate splitting multiple access, IEEE Access 10 (2022) 52398–52409. doi:10.1109/ACCESS.2022.3173125.

[16] J. Kim, S. Hosseinalipour, T. Kim, D. J. Love, C. G. Brinton, Multi-irs-assisted multi-cell uplink mimo communications under imperfect csi: A deep reinforcement learning approach, in: 2021 IEEE International Conference on Communications Workshops (ICC Workshops), 2021, pp. 1–7. doi:10.1109/ICCWorkshops50388.2021.9473585.

[17] Y. Yang, T. Lv, Y. Cui, P. Huang, Madrl based uplink joint resource block allocation and power control in multi-cell systems, in: 2023 IEEE Wireless Communications and Networking Conference (WCNC), 2023, pp. 1–6. doi:10.1109/WCNC55385.2023.10119106.

[18] D. Liu, J. Zhao, C. Yang, L. Hanzo, Accelerating deep reinforcement learning with the aid of partial model: Energy-efficient predictive video streaming, IEEE Transactions on Wireless Communications 20 (6) (2021) 3734–3748. doi:10.1109/TWC.2021.3053319.

[19] G. Xia, Y. Zhang, L. Ge, H. Zhou, Deep reinforcement learning based dynamic power allocation for uplink device-to-device enabled cell-free network, in: 2022 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2022, pp. 01–06. doi:10.1109/BMSB55706.2022.9828568.

[20] 3GPP TS 26.501 V18.3.1, 5G Media Streaming (5GMS); General description and architecture (September 2023).

[21] M. P. Cameron, Zipf's Law across social media, University of Waikato, NewZealand, March 2022.

[22] A.-T. Tran, N.-N. Dao, S. Cho, Bitrate adaptation for video streaming services in edge caching systems, IEEE Access 8 (2020) 135844–135852. doi:10.1109/ACCESS.2020.3011517.

[23] R. S. Sutton, A. G. Barto, et al., Reinforcement learning: An introduction, Vol. 1, MIT press Cambridge, 1998.

[24] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning (2019). arXiv:1509.02971, doi:10.48550/arXiv.1509.02971.

[25] T. P. Truong, V. D. Tuong, N.-N. Dao, S. Cho, Flyreflect: Joint flying irs trajectory and phase shift design using deep reinforcement learning, IEEE Internet of Things Journal 10 (5) (2023) 4605–4620. doi:10.1109/JIOT.2022.3218740.

[26] K. H. Lee, J. Park, S.-T. Kim, J. Y. Kwak, C. S. Cho, Design of nnef-pytorch neural network model converter, in: 2021 International Conference on Information and Communication Technology Convergence (ICTC), 2021, pp. 1710–1712. doi:10.1109/ICTC52510.2021.9621003.

[27] P. Prez, N. Garca, . Villegas, Subjective assessment of adaptive media playout for video streaming, in: 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), 2019, pp. 1–6. doi:10.1109/QoMEX.2019.8743320.

[28] Y. P. Demo, Add youtube functionality to your sites and apps, https://developers.google.com/youtube, (accessed: 01.01.2024) (2024).

[29] T.-V. Nguyen, D.-T. Hua, T. H. Huong, V. T. Hoang, N.-N. Dao, S. Cho, Intelligent qoe management for iomt streaming services in multiuser downlink rsma networks, IEEE Internet of Things Journal 11 (7) (2024) 12602–12618. doi:10.1109/JIOT.2023.3334473.

[30] Q. Zhang, L. Zhu, Y. Chen, S. Jiang, Constrained drl for energy efficiency optimization in rsma-based integrated satellite terrestrial network, Sensors 23 (18) (2023). doi:10.3390/s23187859.

[31] K. K. Nguyen, T. Q. Duong, N. A. Vien, N.-A. Le-Khac, L. D. Nguyen, Distributed deep deterministic policy gradient for power allocation control in d2d-based v2v communications, IEEE Access 7 (2019) 164533–164543. doi:0.1109/ACCESS.2019.2952411.

[32] S. Wang, S. Bi, Y.-J. A. Zhang, Deep reinforcement learning with communication transformer for adaptive live streaming in wireless edge networks, IEEE Journal on Selected Areas in Communications 40 (1) (2022) 308–322. doi:10.1109/JSAC.2021.3126062.