Reinforcing Edge-DASH: Deep Learning for Multi-Objective Streaming Optimization

Arash Bozorgchenani, *Member, IEEE,* David Naseh, *Member, IEEE,* Daniele Tarchi, *Senior Member, IEEE,* Sergio A. Salinas Monroy, *Member, IEEE,* Farshad Mashhadi, and Qiang Ni, *Senior Member, IEEE*

Abstract—With the growing demand for multimedia services, Dynamic Adaptive Streaming over HTTP (DASH) has become a key solution for delivering high-quality video content. In this work, we consider an Edge-DASH scenario and formulate a joint optimization problem that involves four critical aspects: bitrate allocation, user-to-server assignment, caching, and bandwidth allocation. Due to the complexity of the joint problem, we decompose it into sub-problems and address them separately. To solve the resulting sub-problems, we employ deep reinforcement learning, specifically the Deep Deterministic Policy Gradient (DDPG) method, for three of them, and develop a heuristic solution for the fourth. Simulation results demonstrate that our approach enhances performance across multiple metrics, including improved video delivery, reduced buffer underflow and overflow, and more efficient caching, which collectively enable greater utilization of edge resources for streaming. Moreover, we evaluated inference latency across edge and cloud hardware, confirming sub- to few-millisecond performance suitable for real-time deployment. This showcases the benefits of combining learning-based and heuristic techniques to meet the growing demand for adaptive video streaming in edge computing environments.

Index Terms—Adaptive Video Streaming, DASH, Edge Computing, Caching, Bitrate Allocation, Deep Reinforcement Learning.

1 Introduction

Mobile video streaming is estimated to account for approximately 79% of Internet traffic by 2027, consuming 7.7 million petabytes of the total 9.7 million petabytes of data [1]. Devices and connections are accelerating faster than both population and Internet users, where, according to the recent Ericsson mobility report, the number of global mobile subscriptions is projected to reach 9.5 billion by 2031, up from 8.83 billion in 2025 [2]. A large amount of research has been conducted to design solutions to guarantee Internet users' Quality of Experience (QoE) in such a high video-demanding environment.

The heterogeneity of users' demand in terms of network conditions and various hardware/software specifications (e.g., device capability or energy status) raises challenges for video service providers to provide satisfactory QoE to the users. By resorting to the Adaptive Bit Rate (ABR) streaming technology, Dynamic Adaptive Streaming over HTTP (DASH) has been developed as a solution to overcome this issue [3]. In DASH a video is divided into small-sized chunks with different pre-transcoded bitrates. This allows users to watch videos with different resolutions if they experience a low data rate or due to their preferences. Different users may have diverse preferences for video resolution metrics. For example, some users may prefer a higher resolution, whereas others may expect a smooth experience with slightly compromised video resolution. Hence, a pure network-driven strategy would not be appropriate in a mobile environment, and user preferences should be taken into account in video streaming scenarios. The rate adaptation decision can be made on the users' side, where the users can select an appropriate resolution/bitrate for each video chunk based on some factors such as screen resolution, current or predicted available bandwidth [4]. Then, based on a Bitrate Allocation (BrA) scheme, appropriate bitrates can be associated with each user considering their requested bitrate. However, BrA does not guarantee smooth streaming for the user as long as the user does not receive a sufficient data rate to download the chunks in time. Hence, a smart Bandwidth Allocation (BA) to the users is also crucial as it enables the users to download the video chunks in their buffer and prevents stalling while streaming.

Cloud computing has been widely used to support DASH services, mainly to handle the heavy computation, storage, and network demands of transcoding [5]. However, Cloud-DASH suffers from inherent drawbacks: like other cloud-based services, offloading to the cloud introduces high communication latency [6], and directing large volumes of video traffic to the core network increases congestion. Multi-access Edge Computing (MEC) brings computation and storage closer to users by co-locating edge servers with base stations [7]. DASH can benefit from MEC's ability to meet 5G's stringent low-latency requirements. Streaming from the edge also reduces backhaul load and provides more flexibility in selecting streaming sources. Nonetheless, this requires an intelligent User-to-Server Allocation (USA) mechanism to route traffic across the edge–cloud continuum, fully exploiting available resources. Similarly, effective edge streaming depends on a smart caching strategy that accounts for correlations in users' content demands.

A cache hit requires the requested chunk at the correct

- *Arash Bozorgchenani is with the School of Computer Science, University of Leeds, UK, a.bozorgchenani@leeds.ac.uk*
- *David Naseh was with the Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Italy.*
- *Daniele Tarchi is with the Department of Information Engineering, University of Florence, Firenze, Italy, daniele.tarchi@unifi.it.*
- *Sergio A. Salinas Monroy and Farshad Mashhadi are with the Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, USA.*
- *Qiang Ni is with the School of Computing and Communications, Lancaster University, UK.*

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2026.3674131

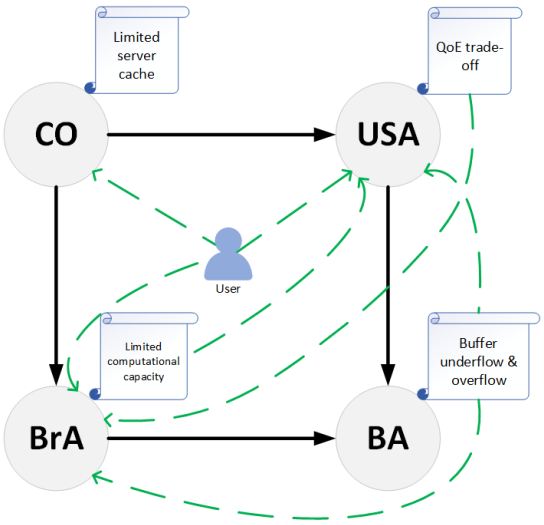

Fig. 1. Coupling relationships among the four problems (BrA, BA, USA, and CO) and their main constraints.

bitrate, making ABR-aware edge caching challenging. With ABR streaming, it is not enough for a chunk to be cached—it must be stored at the specific bitrate requested [8]. To mitigate this under limited edge storage, we enable transcoding at the BSs, reducing the need to cache all bitrate levels. However, caching multiple versions incurs high storage overhead, while real-time transcoding is computation-intensive and can quickly exhaust MEC resources when many videos are processed concurrently. Thus, an effective Caching Optimization (CO) mechanism and bitrate delivery scheme is essential to properly balance cache capacity and processing constraints.

To this aim, in this paper, we address four key and interrelated problems, namely BrA, CO, USA, and BA. Through our investigation, we observed that these problems cannot be effectively studied in isolation, as they are inherently dependent on one another. Specifically, caching is performed based on users' requests, which in turn affects the BrA and USA; the BrA is further constrained by the available computational resources required for transcoding. Moreover, BrA and USA jointly impact the BA, while the users' buffer level ultimately limits the achievable bitrate. Please refer to Fig. 1 for a more detailed view, which illustrates the coupling among the four problems and their main associated constraints.

We investigate combining the advantages of DASH and edge caching to increase the QoE of the users in a mobile wireless network. Due to the dynamicity of the environment in terms of changes in the buffer level of the users, and the diversity of videos, resolutions, and users' demands, the formulated problem is hard to solve. Hence, we decouple the joint formulated problem based on *Tammer decomposition method* into two sub-problems and prove their NP-hardness and time complexity in problem-solving. Then we exploit Deep Reinforcement Learning (DRL) algorithms to solve each of the two sub-problems. At first, DRL is considered to solve the USA problem by enabling dynamic selection of the optimal streaming source (small cell, macro cell, or cloud) for each user. Simultaneously, it addresses the BrA problem to ensure the delivery of the most appropriate bitrate, thus enhancing user QoE. Later, on a larger time scale, we address the CO problem and propose another DRL-based mechanism to update the cache of the edge servers by predicting the demanded videos and bitrate by users of a given area regularly. Finally, a BA algorithm is proposed to maintain the user buffer at an appropriate level, avoiding both underflow and overflow. The main contributions in our work can be summarized as:

1) We consider a reactive streaming approach in a 4-tier network topology, where the clients/users can stream from either the small cells, a Macro cell with a broader coverage area, or the cloud. We also study caching strategies in the small cell layer.
2) In this paper, we investigate four problems, i.e., a) USA, b) BA, c) BrA, and d) CO, offering a distinctive contribution that treats them in an integrated way while emphasizing their mutual interdependencies.
3) We formulate the joint BrA–USA and edge CO problem to optimize delivered bitrate in a multi-layer scenario. To address the high complexity of this joint problem, we decompose it into sub-problems and solve them iteratively, enabling tractable and efficient optimization.
4) We develop a Deep Deterministic Policy Gradient (DDPG)-based solution tailored to these sub-problems. DDPG is well-suited to the continuous action space inherent in this setting, and, to the best of our knowledge, has not previously been applied here, making our approach both novel and well justified.
5) We propose a BA method that leverages the outcomes of the USA and BrA sub-problems to prevent buffer underflow, overflow, and playback stalling, thereby ensuring a stable user experience.
6) The simulation results show that cache size and update frequency affect performance of the CO, transcoding and BrA improve over time, with computational capacity and user load impacting their performance in the BrA-USA problems, and the BA algorithm rapidly stabilises user buffers. Finally, we demonstrate strong overall performance when all four problems are addressed jointly.

The rest of the paper is structured as follows. Section 2 provides a review of the current state-of-the-art. The system model is explained in Section 3. Section 4 elaborates on the problem formulation. Section 5 initially breaks down the problem and subsequently details our learning solutions proposed for each issue. In Section 6, we show the simulation results. Finally, Section 7 offers concluding remarks.

2 Related Works

In this section, we present a comprehensive literature review, categorizing the works primarily into two domains: resource allocation (including computing and storage allocation, BrA, BA, USA) and Edge Caching.

2.1 Resource Allocation in Adaptive Video Streaming

Several works have coupled BrA and BA. In [4], the authors have formulated an adaptive transcoding strategy

in a mobile environment considering BrA and bandwidth adjustment. They have proposed a heuristic method for BrA after grouping the users with similar bandwidths. Mehrabi et al. [9] designed a greedy-based scheduling algorithm to periodically solve the USA and BrA problem with the goal of load balancing considering various QoE metrics. However, these works are completely network-driven and ignore the limitations that exist on the client side and try to maximize the bitrate for all clients. Wei et al. in [10] propose a quantum-inspired bitrate control method for DASH to improve streaming quality. However, it overlooks multi-tier networks and bandwidth allocation, limiting its applicability in complex environments.

Yang et al. [11] formulated a multi-dimensional edge resource allocation in a video streaming scenario. They proposed a regression-based online learning algorithm to estimate the user preferences for allocating edge caching, computing, and bandwidth resources; nevertheless, they ignored most of the important users' QoE in a video streaming scenario. Ozfatura et al. [12] studied the minimization of video streaming stalling in a DASH-based multi-user scenario with varying wireless channels. They proposed a scheduling algorithm for each time slot to select a user for video streaming. However, they assumed that edge servers can serve only one user at a time. In addition, they ignored most of the other QoE issues and did not address any of the BA or CO problems. Li et al. in [13] studied USA and BrA in an ultra-dense network. They defined the cost of the system as the delay in serving the user request. However, in a streaming scenario, minimization of the delay is not essential, while making sure that the users do not experience stalling due to underflow and overflow is important. Bokani et al. in [14] employed a Markov decision process to select the chunk quality in a dynamic channel, however, they faced a high computational complexity. Bayhan et al. in [15] studied BrA in WiFi Access Points to enable cache delivery. They considered a proactive approach for BrA while considering a tolerable difference in comparison with the bitrate requested by the users. They proposed a compositional Pareto-algebraic heuristic for the BrA and another heuristic for the BA allocation. The model used in this work has some similarities with ours; however, they have not considered transcoding techniques, which have a significant impact on users' QoE, and also no CO is performed.

Guo et al. in [16] considered that multimedia packets with similar requirements are assigned to the same QoS flow. They proposed a Deep Q-Network (DQN) to allocate storage and computing resources to each flow. The hybrid edge-cloud has been considered in some of the works. Tao et al. in [17] presented a predictive adaptive streaming with edge-cloud-assisted prediction of data rate in mobile networks. Their framework uses slow fading to get the optimal risk allocation for a long-term scheduling plan.

Recent studies have examined adaptive video streaming from various angles. Hayamizu et al. [18] proposed a QoE-aware bitrate adaptation method for MPEG-DASH over Information-Centric Networks, where bitrate selection depends on bandwidth conditions and cached segments in intermediate routers. However, higher bitrates are achieved mainly when segments are cached, limited by router capacity, and the work does not address BA or CO. Similarly, Zhong et al. [19] modeled adaptive video streaming in cache-assisted mobile networks as a multi-source multicast multi-rate problem and proposed distributed algorithms for rate adaptation. While their method optimizes user bitrates under link-capacity constraints, it assumes fixed caching and overlooks cache management, USA, and explicit bandwidth optimization.

In parallel, emulator-based approaches have emerged to evaluate streaming performance. Esper et al. [20] developed QoE-DASH, an open-source emulator extending goDASH with network topology, caching, and user-context modeling for DASH-based streaming in MEC settings. Although it enables testing of caching and recommendation strategies, it focuses on emulation rather than optimizing bitrate adaptation, BA, caching, or USA. Khan et al. [21] proposed an edge-driven multi-agent DRL framework that jointly optimizes BrA and QoE fairness among users. While effective in balancing QoE under bandwidth constraints, it does not explicitly optimize BA, caching, or USA, emphasizing edge-level bitrate control. Similarly, Lu et al. [22] introduced ABUV, a mobile video streaming framework that jointly optimizes BrA and super-resolution decisions using DRL to enhance quality and reduce energy use. However, ABUV focuses solely on client-side adaptation without addressing caching, BA, or multi-source streaming, limiting its relevance to distributed or edge-assisted systems.

In our earlier work [23], we studied a joint USA and BrA problem using a DDPG-based solution. Building on that foundation, this work significantly extends the problem scope by incorporating caching and BA, resulting in a more comprehensive and realistic system model. By jointly considering all four aspects, this work provides a more holistic and impactful solution for edge-assisted adaptive video streaming.

2.2 Edge Caching in Adaptive Video Streaming

In [24], the authors have formulated a joint optimization of QoE of users and backhaul traffic in an edge DASH scenario where the edge servers harvest from the solar panels. They proposed two heuristics for BrA and cache replacement. Similar work has also been done in [25]. However, these two approaches did not take into account MEC-enabled transcoding, which has a high impact on the quality of adaptive streaming services.

Some works studied caching problems with a Scalable Video Coding technique. Jedari et al. [26] have formulated a cache trading problem from an economic point of view, where content providers compete to cache their content on network providers' resources. They formulated a double-auction approach to maximize the profit of both sides by considering an SVC technique. However, they have not studied the QoE of video streaming from the user side, such as stalling. Zhang et al. in [27] investigated the CO problem by combining the SVC technique and information-centric wireless networks. Wu et al. [28] studied caching for downlink video streaming using SVC. They proposed a heuristic that caches both base and enhancement layers for popular videos, and only the base layer for less popular ones. However, in large-scale streaming, avoiding playback stalls requires explicit buffer management. This critical as-

pect was simplified in their work by assuming a fixed transmission deadline for each frame instead.

Guo et al. [29] studied ABR streaming under time-varying vehicular channels. They modelled BrA and BA at a small scale and CO at a larger scale, proposing a Lyapunov-based solution to maximize a weighted sum of video quality and backhaul savings. However, their model overlooks several key QoE requirements. Tran et al. [30] examined collaborative caching and request scheduling, where content is delivered either from the serving BS or via relaying. They used an LRU caching policy and proposed an online heuristic that connects each user to the BS with the highest processing resources. Yet, they did not incorporate most user QoE factors and did not introduce a new caching strategy.

Xenakis [31] investigates joint video BrA and edge caching in MEC-empowered, slice-enabled 5G networks. The study formulates a dynamic programming model to determine an optimal fixed bitrate and cache placement strategy under given rate and storage constraints, addressing the "oscillation dynamics" of traditional DASH. While effective in coordinating bitrate and caching, it does not consider dynamic BA, USA, or adaptive bitrate control across segments, and its reliance on precise network state information limits its effectiveness in dynamic environments. In [32] a solution that uses Media Cloud is proposed considering a three-way trade-off between the caching, transcoding, and bandwidth costs to minimize the total operational cost for streaming video services on demand. By considering coordination among local caches and different rate-distortion behaviour of videos, Li et al. in [33] have proposed a mobile cache placement framework for dynamic adaptive video streaming. Han et al. in [34] studied the random content placement technique in which the contents are stored according to a probability distribution and optimize the probability of successful transmission.

Some works have also studied the bitrate caching problem, which is the main focus of our study. Pederson et al. in [8] introduced a new client-driven ABR algorithm that uses the characteristics of video frames through a table for rate selection. They also proposed a heuristic to update the caching and content quality selection to increase the caching capacity. However, they only consider one cache entity, while in our work, we address caching at several BSs on a larger scale. Zhang et al. in [35] aimed to maximize QoE for mobile users due to the problem of bitrate caching. However, they have assumed that the probability of rates requested by users is known, which is not possible in a vastly varying mobile environment. They have also focused on a snapshot problem and have not considered a transcoding technique. Tran et al. in [36] addressed the optimization of the cached bitrate in a video streaming scenario. They have assumed that each streamer can cache one maximum bitrate, which should be optimized considering the cache storage and computing constraints. They proposed an online iterative greedy algorithm, which increases the cached bitrate until the computing and storage constraints allow.

ML-based solutions for caching have also been proposed in some works. In [37], the authors proposed an ML-based solution to predict the number and bitrate of the video segments requested by users. However, their caching solution executes every run, which adds a lot of overhead to the system. Raca et al. in [38] studied the prediction of future throughput using radio metrics in an adaptive video streaming scenario and proposed an ML-based solution.

TABLE 1
Comparison of the most related works on video streaming

Ref.	Problem	Video Encoding	QoE	Solution
[8]	CO	Transcoding	a, d	Heuristic
[4]	BA, BrA	Transcoding	a, b	Heuristic
[9]	USA, BrA	-	a, b, c, d, e	Heuristic
[13]	USA, BrA	Transcoding	a	Lagrangian multiplier
[15]	BrA, BA	-	a, d	Heuristic
[18]	BrA	-	a, c, d, e	Heuristic
[19]	BrA	SVC	a	Heuristic
[21]	BrA	-	a, b, d, e	DRL
[22]	BrA	-	a, d , e	DRL
[24], [25]	BrA, CO	-	a, b, c, d, e	Heuristic
[26]	CO	SVC	c	Auction
[28]	CO	SVC	d	Heuristic
[29]	BrA,BA,CO	-	a	Lyapunov opt.
[30]	CO	Transcoding	a	Heuristic
[31]	BrA, CO	-	a, d	Dynamic programming
[35]	CO	-	a	Lagrangian multiplier, heuristic
[36]	BrA, CO	Transcoding	a	Heuristic
[37]	BrA, CO	Transcoding	a	Random Forest

Tab. 1 summarizes some of the most important and relevant research activities from different points of view. In Tab. 1, we listed the QoE metrics assessed in each study, depicted with symbols: a) video bitrate, b) fairness in relation to bitrate, c) start-up delay, d) stalling frequency, and e) bitrate fluctuations. The main problems that have been addressed are BrA[1], BA, USA, and CO.

Despite the strong interdependence among the four problems and their constraints (see Fig. 1), we observed that no prior study has jointly formulated and optimized these components within a unified framework. Understanding and optimizing these coupled relationships are crucial for ensuring high user QoE and efficient edge resource utilization, where a comprehensive solution to Edge-DASH can only be achieved by jointly considering all these interdependent components, as addressing them in isolation would fail to capture the system's true complexity and operational dynamics. Our work is a holistic design that simultaneously redefines the system topology, introduces a novel joint prob-

1. BrA is sometimes also referred as bitrate selection, depending on whether it was a user or network-driven selection

lem formulation, and develops customized optimization and heuristic algorithms to address the resulting complexity. This provides a realistic and comprehensive solution that reflects the actual operational coupling in edge-DASH.

3 System Model

3.1 Network Model

We consider a large-scale wireless network covering a geographical area with multiple BSs. MBSs provide wide-area coverage and connect to the cloud through high-capacity backhaul links, each equipped with a Macrocell Edge Server (MES) for local storage and computation. For clarity, we focus on a region with a single MBS, though the model extends naturally to multiple MBSs. Each macrocell contains S SBSs, each co-located with a Small cell Edge Server (SES). The network supports on-demand video streaming for distributed edge users (ECs) who request videos at specific bitrates. SESs store selected video content in multiple bitrates and serve them over HTTP, while the MES stores all videos but only at their highest bitrate.

ECs may prefer specific bitrates for streaming due to factors like low battery power or a limited data budget. Consequently, our model employs a reactive approach: disregarding these bitrate requests could lead to an unsatisfactory QoE. This is justifiable because ECs generate their requests based on these constraints, meaning those with limitations ask for lower bitrates, while others request higher ones.

There are N ECs randomly located in the area and sending video streaming requests, where i represents the index of an EC, $\mathcal{U} = \{u_1, \ldots, u_i, \ldots, u_N\}$. We consider that ECs are connected to the nearest BS[2] (in terms of signal strength). Let us define $\mathcal{V} = \{v_1, \ldots, v_m, \ldots, v_M\}$ as the set of all videos, where m represents the index of a specific video. We envision that each video is divided into K chunks, where each chunk k can be encoded to different bitrate versions. The SESs are equipped with transcoding functionality, enabling them to transcode a video chunk with a certain bitrate to a lower bitrate. Each EC can request to watch a chunk with one of the bitrate levels $\mathcal{O} = \{o_{\min}, \ldots, o, \ldots o_{\max}\}$, from the lowest to the highest. We denote $o_{m,k}$ as the bitrate of the kth chunk of the mth video. The duration of chunks in each video is supposed to be the same while it varies across different videos. We denote L_m as the duration of any chunk in video m, which is typically between 2 to 10 seconds.

We denote $r_i^{o_{m,k}}(t)$ as the bitrate request of the ith EC for a video chunk at time instant t, which is shown with $r_i(t)$ throughout the paper for the sake of simplicity in the notation. The set of requests of all ECs is denoted as $\mathcal{R}(t) = \{r_1(t), \ldots, r_i(t), \ldots, r_N(t)\}$. The content that is later delivered to the EC is shown with $\hat{r}_i(t) \equiv \hat{r}_i^{\hat{o}_{m,k}}(t)$, which delivers the bitrate $\hat{o}_{m,k}$. Note that the bitrate delivered might be different from that requested. Moreover, we define $\hat{\mathcal{R}}(t) = [\hat{r}_i(t)]_{i=1}^N$ as the set of requests delivered. Since the encoding process is inherently variable, the chunks might have different sizes. We denote the average chunk size of a bitrate as $\Delta_{o_{m,k}}$ bits.

2. The model can be further extended to a multi-access edge scenario, where each EC can also select among the SESs for streaming, however, it is subject to future investigation as it is not the focus of this work.

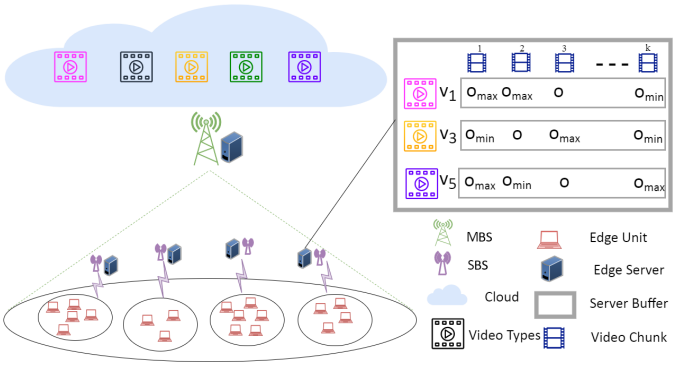

Fig. 2. The 4-tier Network Topology for Multi-access Video Streaming

The SESs provide storage and computation capabilities to enable caching and transcoding at the network edge. Each SES has a certain capacity C_s enabling it to cache a certain number of videos and chunks out of the total M videos and K chunks. The proposed 4-tier edge computing-assisted multi-access DASH system for adaptive video streaming is illustrated in Fig. 2. As seen in the figure, the caching is conducted in SESs. For instance, the SES buffer enables it to cache v_1, v_3, and v_5. Moreover, due to capacity constraints, not all video bitrates can be cached in the SES, and therefore, each chunk, if cached, is stored with a certain bitrate.

3.2 Streaming Sources and Video Encoding

Each EC at time t requests a video chunk at a target bitrate and can retrieve it from its SBS, the MBS, or the cloud. To maximize QoE, an EC first checks its SBS: if the exact bitrate is cached, it results in a direct edge hit; if only a higher-bitrate version is available, the SBS can transcode it down. When the SBS stores a lower bitrate than requested, does not cache the chunk, or when accessing the MBS yields a higher QoE, the EC streams from the MBS, which stores all videos only at the highest bitrate. If neither SBS nor MBS can satisfy the QoE requirements, or cloud access provides higher QoE, the EC retrieves the chunk from the cloud, which stores all bitrates. The objective is to optimize the delivered bitrate relative to the requested bitrate under these QoE constraints.

3.3 Time-Scales of the Operations

We define here the operational time scale for ECs and SBSs. Each video is divided into chunks, and while a single video maintains a consistent chunk duration, this duration can vary across different video titles. The ECs select the bitrate for the next chunk at intervals roughly equal to that video's chunk duration. We refer to the moment when the EC chooses the bitrate for the next chunk as the Bitrate Selection Point (BSP). Since chunk durations differ between videos, the BSP interval also varies and is specific to each video. We denote the BSP of video v_m as BSP_m. Shorter chunks allow the EC to react more quickly to changing network conditions by adjusting the bitrate more often. However, this also increases the frequency of requests and messages sent over the network. These variations in network load and channel quality are handled by the BA mechanism, which operates at a finer time scale than the EC's bitrate selection.

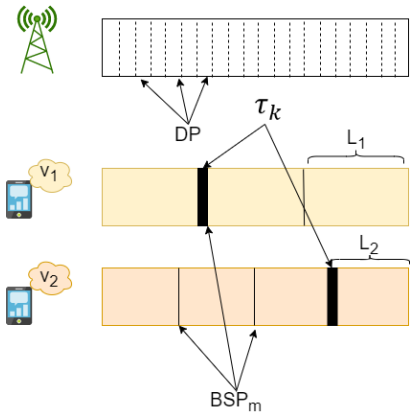

Fig. 3. Time-Scale of the operations

The SBS performs BA decisions at regular intervals known as Decision Points (DPs). It has two main responsibilities: BrA and BA. During BrA, the SBS considers the EC's requested bitrate at BSP_m and decides which bitrate to assign for the next chunk. During BA, the SBS determines how much bandwidth to allocate to the EC based on the bitrate assigned at each DP. These two processes are interconnected to ensure smooth video streaming. Importantly, bitrate selection by ECs and BA by the SBS operate on different time scales. Bitrate selection occurs less frequently (on the order of seconds), while BA must react more quickly (on the order of milliseconds). As shown in Fig. 3, the SBS performs both BA and BrA at each DP, while ECs request bitrate updates at their respective BSP_m. We use a simple scheduling scheme[3] to coordinate these operations.

4 Problem Formulation

To capture the QoE of ECs, various metrics have been considered, with video stalls significantly reducing satisfaction [39]. The video bitrate is also a key QoE factor, but it often conflicts with stall reduction: higher bitrates increase the risk of stalling. Balancing these objectives is challenging, especially when many ECs compete for limited edge resources. This work aims to maximize EC satisfaction by optimizing the delivered bitrate through efficient caching, BrA, and USA mechanisms, all while considering edge resource constraints. In this section, we introduce the key QoE parameters in our model, define our objective, and formulate the optimization problem.

4.1 QoE Model

QoE has no single standard set of performance metrics [40], and its evaluation remains an active research topic. A detailed overview of QoE factors in adaptive streaming is provided in [41]. Common metrics in multimedia systems include stalling ratio, startup latency, bitrate switches, average bitrate, and fairness in BrA [24], [42]. In reactive streaming, fairness and bitrate-switch metrics are less relevant because ECs directly request their desired bitrate, and frequent alternation between bitrates naturally increases switch counts. Hence, these metrics do not fit well in our reactive scenario. Instead, we focus on stalling ratio alongside computing and storage constraints. Our QoE definition for DASH streaming therefore emphasizes these factors, with particular attention to caching and BrA.

We define the placement of kth chunk of mth video with bitrate $o_{m,k}$ on SES s with a binary variable $x^s_{o_{m,k}}$ equal to 1 if $o_{m,k}$ is cached and zero otherwise. Then we define the caching constraint on each SES as

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{o=1}^{O}\Delta_{o_{m,k}}\cdot x^s_{o_{m,k}} \leq C_s \quad \forall s \tag{1}$$

which shows that the capacity of each SES is limited for caching.

The playback curve specifies the cumulative data required to display the video without interruptions, which is a characteristic of the video and independent of the underlying channel [17]. During playback, the DASH algorithm selects the bitrate of the next chunk to minimize rebuffering and stalling, improving the QoE of the ECs [3]. The selection of an appropriate bitrate is mainly dependent on ECs' preferences, device capabilities, and network conditions.

To ensure smooth streaming without stalling or rebuffering, we impose underflow and overflow constraints. Let $uf(t)$ and $of(t)$ define the minimum and maximum data duration that should be sent to the EC to avoid the buffer from experiencing underflow and overflow. Since playback pauses only at chunk transitions, an EC may experience underflow or stalling only at transition points $\{0, L_m, \ldots, kL_m, \ldots, KL_m\}$, known as BSPs. We denote this stalling period or waiting time by the end of chunk k with τ_k. As shown in Fig. 3, waiting occurs at the end of chunk 1 in v_1, and chunk 3 in v_2. Hence, the total waiting time for an EC streaming K chunks is given by $\sum_{k=0}^{K}\tau_k$. Note that if a video is streamed smoothly, then $\tau_k = 0$ for all chunks.

Let us define t_k as the time that chunk k is successfully downloaded and $B_i(t)$ as the buffered video duration in seconds at the ith EC at time t. The dynamicity of the buffer occupancy evolves over downloading or displaying the chunks as

$$B_i(t+1) = B_i(t) - L_m \cdot \mathbb{1}_{\{\mathbb{P}^s_{m_k}=1\}} + L_m \cdot \mathbb{1}_{\{\mathbb{D}^s_{m_k}=1\}} \tag{2}$$

where $\mathbb{P}^i_{m_k} = 1$ indicates that the chunk is displayed, and $\mathbb{D}^i_{m_k} = 1$ indicates that if it is downloaded by ith EC. The buffer increases by L_m seconds after a chunk is downloaded at any time t_k, and decreases when a chunk with duration L_m is displayed. Note that both increase and decrease in the buffer are checked at every time instant.

For improving the QoE of the ECs, we need to ensure that the EC's buffer level does not go below zero, at each time instant. Let us assume that there have been $K(t)$ chunks downloaded up to time instant t. The buffer underflow can be avoided if

$$\sum_{k=1}^{K(t)}\left(\int_{t_{k-1}+\tau_{k-1}}^{t_k}\frac{R(t)}{F_k}dt\right) \geq t \tag{3}$$

where F_k represents the display rate in the EC, $R(t)$ is the channel data rate depending on the corresponding streaming source (either SBS, MBS or cloud) at the time instant

3. More advanced scheduling mechanisms could be considered, but are beyond the scope of this work.

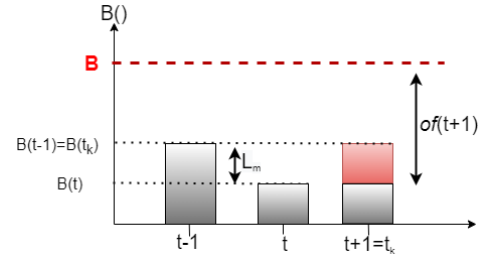

Fig. 4. The buffer level update of a typical EC at every time instant

t, and τ_k is a non-zero value only if the EC experiences a stalling after displaying chunk k. The term in parentheses shows the ratio between the downloading rate and the display rate, where a higher value of the ratio shows an increase in buffer occupancy. At each time instant, the term on the left side should be greater than t for smooth video streaming without stalling. Similarly, a buffer overflow can occur at any time instant t after a chunk is downloaded. Let us define $\mathbf{B}$ as the maximum buffer capacity, then to avoid the occurrence of the overflow, the following should hold

$$\sum_{k=1}^{K(t)} \left(\int_{t_{k-1}+\tau_{k-1}}^{t_k} \frac{R(t)}{F_k} dt \right) \leq \mathbf{B} \tag{4}$$

Fig. 4 shows the update of the buffer level for an EC. As seen in the time instant t, one chunk is displayed, which is why the buffer level decreases by an amount of L_m seconds. Moreover, in the next time instant, the chunk k is successfully downloaded and thus L_m seconds are added to the buffer. Additionally, we can see the remaining buffer level of the EC to experience the overflow.

We assume that the computing resources of the SESs for transcoding are limited. Let us denote η_o^s as the computing resource for transcoding a chunk with a cached bitrate to a lower bitrate o by an SES. Hence, the consumption of computing resources to transcode to lower bitrate levels is higher, i.e., $\eta_o^s < \eta_{o^-}^s$ if $o^- < o$. Let $\mathbb{R}_i^o = 1$ indicate if EC ith request, i.e., r_i, requires transcoding to the requested bitrate o. The constraint on transcoding computing resources is then given by:

$$\sum_{i=1}^{N_s} \sum_{o=1}^{O} \eta_o^s \cdot \mathbb{R}_i^o \leq \Omega^s \quad \forall s \tag{5}$$

where Ω^s is the maximum transcoding computing resource of an SES at a time instant and N_s is the number of ECs allocated to stream from SES s. We assume that the cloud has no limitations in terms of computation capacity. The cost of a transcoding chunk is regarded as the CPU usage on the cache servers.

4.2 Optimization Problem

Since each request can be served by the SBS, MBS, or cloud, the USA decision is made per EC to prevent stalling and satisfy QoE. USA is closely linked to BrA because the chosen server must consider whether transcoding is needed. Given edge limitations, all streaming options must be jointly optimized to assign both the bitrate and the server. In some cases, delivering a lower bitrate than requested is necessary to avoid stalling, showing the trade-off between USA and QoE. Both USA and BrA also depend on the SES cache state: better caching enables more direct hits or low-cost transcoding at the edge. Thus, caching, BrA, and USA are inherently interconnected, with caching providing the key input to the other two decisions.

Considering the cache capacity constraint of each SES, only a limited number of video chunks with a certain bitrate can be cached in each SES. In this work, MES performs CO on the SESs within its coverage, as it stores all videos and chunks at the highest bitrate. Let $\Phi^s \in \mathbb{R}^{M \times K}$ be the caching matrix of sth SES and $\phi_k^m = o_{m,k} \cdot \mathbb{1}_{\{\mathbb{I}_{m_k}^s = 1\}}$ be an element of the matrix, where $\mathbb{I}_{m_k}^s$ is an indicator function which takes 1 if the kth chunk of the mth video is cached at the sth SES. Each element of the matrix takes either 0, if the chunk of the corresponding video is not cached, or a value of $o_{m,k} \in \mathcal{O}$, otherwise. Thus, Φ^s is defined as

$$\Phi^s = \begin{pmatrix} \phi_1^1 & \dots & \phi_k^1 & \dots & \phi_K^1 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_1^m & \dots & \phi_k^m & \dots & \phi_K^m \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \phi_1^M & \dots & \phi_k^M & \dots & \phi_K^M \end{pmatrix} \tag{6}$$

which is a caching matrix showing the allocation of a bitrate to each cached video chunk.

To define the cost functions for our problem, we exploit the *Weber–Fechner* law, which in principle studies the relation between the actual change in a physical stimulus and the perceived change by humans. User satisfaction in terms of QoE has been shown in some areas, such as communication systems and multimedia applications, following logarithmic laws [35], [43]. Hence, we define the cost function of the caching problem for each request as:

$$\Lambda(\phi_k^m, r_i(t)) = \begin{cases} \alpha \log \frac{\max\left(\phi_k^m(t), r_i(t)\right)}{\min\left(\phi_k^m(t), r_i(t)\right)} & \text{if } \mathbb{I}_{m_k}^s = 1 \\ Y & \text{Otherwise} \end{cases} \tag{7}$$

where Y represents a high cost incurred when the requested chunk is not cached. This implies that a cost is associated with a mismatch between the requested and cached bitrates. The greater the difference, the higher the cost, as closer cached bitrates require fewer transcoding resources. Conversely, if the requested chunk is entirely absent from the cache, a significant cost of Y is incurred. Thereby, the overall caching cost function for an SES covering N_s ECs can be formulated as

$$\hat{\Lambda}^s(\Phi^s, \mathcal{R}^s(t)) = \sum_{i=1}^{N_s} \Lambda(\phi_k^m, r_i(t)) \tag{8}$$

where $\mathcal{R}^s(t)$ is the set of requests of the assigned ECs.

In the USA problem, on the other hand, each EC is allocated to a single server, either from SES, MES or cloud, for streaming. Let us define $a_{i,j}$ as an element that shows where u_i streams from, where j stands for the index of the SES, MES or the cloud. We define $A \in \mathbb{R}^{N \times 3}$ as the USA matrix for all ECs, where $\sum_{j=1}^{3} a_{i,j} = 1$. Given that we use a reactive approach for BrA and that BrA and USA are

performed jointly, we define their joint cost function for each EC request as

$$\Upsilon_{\phi_k^m}\left(\hat{r}_i(t), r_i(t), a_{i,j}\right) = \alpha \log \frac{\max\left(\hat{r}_i(t), r_i(t)\right)}{\min\left(\hat{r}_i(t), r_i(t)\right)} \quad (9)$$

where α is a positive constant. This implies that bitrate differences at lower levels have a greater impact on cost than at higher levels. As a result, this prioritizes minimizing losses at lower bitrates, where dissatisfaction is more noticeable. Moreover, the higher the difference between the requested and delivered bitrates, the higher the cost. Thus, when $\hat{r}_i(t) = r_i(t)$, cost is zero. Similarly, we define the overall joint BrA-USA cost function for an SES covering N_s ECs as

$$\hat{\Upsilon}^s\left(\hat{\mathcal{R}}^s(t), \mathcal{R}^s(t), A\right) = \sum_{i=1}^{N_s} \Upsilon_{\phi_k^m}\left(\hat{r}_i(t), r_i(t)\right) \quad (10)$$

We can observe that the minimization of the cost functions and the satisfaction of the QoE constraints (i.e., guaranteeing no stalling and respecting the cache and computational capacities) are interconnected. On the one hand, delivering and caching the requested bitrates minimizes the cost functions. Conversely, the computational resources for transcoding and cache capacity resources at the SES are limited to deliver all ECs' desired bitrates. Hence, there exists a trade-off between the three problems. We define the optimization problem as

$$\mathbf{P1}: \underset{\hat{\mathcal{R}}, \Phi^{\text{SES}}, A}{\text{minimize}} \Bigg\{ \sum_{t=1}^{T}\sum_{i=1}^{N} \Bigg(\Upsilon_{\phi_k^m}\left(\hat{r}_i(t), r_i(t), a_{i,j}\right) + \Lambda\left(\phi_k^m, r_i(t)\right)\Bigg)\Bigg\} \quad (11)$$

subject to

$$\mathbf{C1}: \text{Eq. (1)} \quad \forall s \quad (12)$$

$$\mathbf{C2}: \text{Eq. (3)} \quad \forall u_i \in \mathcal{U} \quad (13)$$

$$\mathbf{C3}: \text{Eq. (4)} \quad \forall u_i \in \mathcal{U} \quad (14)$$

$$\mathbf{C4}: \text{Eq. (5)} \quad \forall s \quad (15)$$

$$\mathbf{C5}: \sum_{k=0}^{K} \tau_k \leq T_{\text{W}} \quad \forall u_i \in \mathcal{U} \quad (16)$$

$$\mathbf{C6}: \sum_{j=1}^{3} a_{i,j} = 1, \quad \forall u_i \in \mathcal{U} \quad (17)$$

In (11), the objective is to minimize the deviation between the requested bitrate and the delivered bitrate and the cached bitrate for all ECs during the time horizon T. The optimization is with respect to the caching matrix, the delivered bitrate vector $\hat{\mathcal{R}}$, and the server assigned to streaming A. Constraint (12) guarantees that the caching capacity limitation is respected for each SES. Constraints (13) and (14) are the underflow and overflow conditions for each EC. The computational constraint for transcoding for each SES is shown in (15). The stalling that each EC experiences during streaming should not exceed the threshold T_{W}, and this is shown in (16). Finally, the USA condition that each EC streams either from the SBS, MBS or cloud is shown in (17).

As seen, the caching, USA and BrA problems are interconnected. Minimizing (11) with respect to $\hat{\mathcal{R}}$ ideally leads to $\hat{\mathcal{R}} = \hat{R}$, but this does not always satisfy the constraints, sometimes resulting in cost function losses. Additionally, the effectiveness of the BrA-USA policy is heavily dependent on the caching matrix, highlighting the need for an intelligent caching mechanism.

5 DRL-Based and Heuristic Approaches for the Joint Problem

5.1 Problem Decomposition and Analysis

Upon assessing the structure of $\mathbf{P1}$ and the constraints $\mathbf{C1}-\mathbf{C6}$, we can observe that by temporarily fixing the BrA vector $\hat{\mathcal{R}}$ and the USA matrix A, the original complex problem can be decomposed into two sub-problems with separated objectives and constraints by exploiting *Tammer decomposition method* [44]. Let us denote

$$\boldsymbol{\Upsilon} = \sum_{t=1}^{T}\sum_{i=1}^{N} \Big(\Upsilon_{\phi_k^m}\left(\hat{r}_i(t), r_i(t), a_{i,j}\right) + \Lambda\left(\phi_k^m, r_i(t)\right)\Big)$$

for the sake of notation simplicity. Accordingly, we first rewrite $\mathbf{P1}$ as an equivalent problem:

$$\widetilde{\mathbf{P1}}: \min_{\hat{\mathcal{R}}, A}\Big(\min_{\Phi^{\text{SES}}} \boldsymbol{\Upsilon}\Big) \quad (18)$$

subject to $\mathbf{C1}-\mathbf{C6}$.

We can see that $\mathbf{C1}$ and other constraints are decoupled from each other; thus, solving $\widetilde{\mathbf{P1}}$ in (18) is equivalent to the joint BrA-USA subproblem ($\mathbf{P1.1}$) below, which minimizes the cost function with a given cache matrix:

$$\mathbf{P1.1}: \min_{\hat{\mathcal{R}}, A} \boldsymbol{\Upsilon}^* \quad (19)$$

subject to $\mathbf{C2}-\mathbf{C6}$, where $\boldsymbol{\Upsilon}^*$ is the optimal cost function which corresponds to the CO subproblem ($\mathbf{P1.2}$) as follows:

$$\mathbf{P1.2}: \boldsymbol{\Upsilon}^* = \min_{\Phi^{\text{SES}}} \boldsymbol{\Upsilon} \quad (20)$$

subject to $\mathbf{C1}$. Therefore, the original $\mathbf{P1}$ has been decomposed into two subproblems $\mathbf{P1.1}$ and $\mathbf{P1.2}$, namely, the joint BrA-USA and CO. It should be noted that this decomposition does not change the optimality of the solution [44], [45]. Hence, upon obtaining the solutions to both $\mathbf{P1.1}$ and $\mathbf{P1.2}$, the solution to the original problem $\mathbf{P1}$ in (11) is found.

Considering ι as the number of chunks cached in all videos, the solution space for the selection of videos and chunks for a single SES is $\binom{M\times K}{\iota}$ such that ι is in a feasible range with respect to $\mathbf{C1}$. Adding the combinations of the bitrate and USA options, the solution space for each SES becomes $\binom{M\times K}{\iota} \times N_s^{|\mathcal{O}|} \times N_s^3$. Expanding this space to all SESs, the solution space becomes $\binom{M\times K}{\iota} \times N_s^{|\mathcal{O}|} \times N_s^3 \times S$ for one specific value of ι. The solution space becomes even larger considering different values of ι. This corresponds to the number of possibilities to check for the optimal solution to $\mathbf{P1.2}$, which cannot be found in a tractable time. In the following, we prove the NP-hardness of $\mathbf{P1.2}$.

Theorem 1. The problem $\mathbf{P1.2}$ is NP-hard.

Proof. The NP-hardness of the problem $\mathbf{P1.2}$ can be proven at a given time slot via a polynomial-time reduction from *0-1 Knapsack problem* to our problem. Given n items each with x_i copies, where $x_i \in \{0, 1\}$, a *0-1 Knapsack problem* aims at maximizing the accumulated value of the selected items so that their accumulated weight does not exceed Knapsack's capacity W.

We translate the instances of *0-1 Knapsack problem* into those of our problem by mapping the set of n items and their weights to the set of $[M \times K]$ chunks and their bitrates. A higher bitrate reflects a higher cache occupancy (weight) of the respective chunk (item). In addition, the value of each item could be mapped to the negated value of the cost function in (9). Furthermore, we map the weight W to the cache capacity C_s in our problem. Besides, the mapping satisfies the constraint that at maximum one copy of each chunk can be stored in the cache. It should be noted that we aim to optimize the cache for all SESs, hence, this procedure is done for each of the cache matrices of SESs. Now, since *0-1 Knapsack problem* is NP-hard [46], our CO problem in $\mathbf{P1.2}$ is also NP-hard, (i.e., we have the reduction *0-1 Knapsack problem* $\propto \mathbf{P1.2}$). This completes the proof. □

The optimization problem in $\mathbf{P1.1}$ is a Mixed Integer Nonlinear Program (MINLP), where finding an optimal solution requires exponential time complexity [47]. The MINLP property of $\mathbf{P1.1}$ stems from the non-linearity of the objective function and the existence of both integer and non-integer constraints. Intuitively, $\mathbf{P1.1}$ can be solved by going through all combinations of the BrA vector $\hat{\mathcal{R}}$ and USA options. Nevertheless, since $\hat{\mathcal{R}}$ is an integer vector, and also considering the transcoding operation and respecting the constraints, $\mathbf{P1.1}$ can be difficult to resolve. For instance, a degree of complexity arises from the underflow and overflow constraints. Naturally, this means that streaming from the edge should be fairly allocated to the ECs such that when the buffer level is good enough for one EC, the EC can be associated with the Macro or cloud level. Balancing the buffer level among all ECs so that the underflow and overflow do not occur requires additional complexity. The same complexity can be observed in the transcoding constraint, which should be allocated to the requests among all ECs in a fair manner.

Given the large number of variables that scale with the number of ECs and SESs, our aim is to design a low-complexity, distributed solution that remains practical while offering competitive performance. To address this complexity, we propose DRL-based algorithms capable of achieving millisecond-level computation for the caching and BrA–USA decisions. Specifically, we design two feed-forward DRL-based Neural Networks (NN) that take EC demand as input and output the corresponding BrA, USA, and caching decisions. However, enforcing the $\mathbf{C2} - \mathbf{C6}$ constraints within DRL to solve $\mathbf{P1.1}$ is complex and slows down learning. Moreover, upon closer examination of $\mathbf{P1.1}$, we observe that $\mathbf{C2}$, $\mathbf{C3}$ and $\mathbf{C5}$ depend on the allocated bandwidth. This allows us to reformulate $\mathbf{P1.1}$ as:

$$\widetilde{\mathbf{P1.1}} : \min_{\hat{\mathcal{R}}, A} \boldsymbol{\Upsilon}^{*} + \zeta(\lambda_1 h_1 + \lambda_2 h_2 + \lambda_3 h_3) \tag{21}$$

subject to $\mathbf{C4}$ and $\mathbf{C6}$, where

$$h_1 = \max\left(0,\, t - \sum_{k=1}^{K(t)} \left(\int_{t_{k-1}+\tau_{k-1}}^{t_k} \frac{R(t)}{F_k}\, dt \right) \right)$$

$$h_2 = \max\left(0,\, \sum_{k=1}^{K(t)} \left(\int_{t_{k-1}+\tau_{k-1}}^{t_k} \frac{R(t)}{F_k}\, dt \right) - B \right)$$

$$h_3 = \max\left(0,\, \sum_{k=0}^{K} \tau_k - T_{\mathrm{W}} \right)$$

Here, h_1, h_2, and h_3 represent the penalty terms for violations of the constraints $\mathbf{C2}$, $\mathbf{C3}$, and $\mathbf{C5}$, respectively, while ζ serves as a weighting factor for these violations. In this formulation, DRL is used to minimize the primary objective $\boldsymbol{\Upsilon}^{*}$ without explicitly enforcing $\mathbf{C2}$, $\mathbf{C3}$, and $\mathbf{C5}$. To ensure that these constraints are satisfied, a BA algorithm is introduced, which adjusts $\hat{\mathcal{R}}$ and A to strictly enforce the feasibility. The following three subsections detail the three phases involved in solving this problem, including the heuristic BA algorithm, the proposed neural network (NN) architecture, and the training procedure.

5.2 Phase 1: Decentralized DDPG-based DRL for bitrate allocation

5.2.1 Preliminaries

We assume that each EC is associated with the SES with the highest SINR. Among all ECs in a small cell, some of them make a request in each time slot, and some others are in the playback state and have already made their request in the previous slot(s). Consider $\mathcal{U}_s^0(t)$ as the set of ECs connected to SES s and have no request, $\mathcal{U}_s^1(t)$ as the set of ECs that have a new request, and $\mathcal{U}_s(t)$ as the set of all ECs connected to SES s in slot t. Thus, we have $|\mathcal{U}_s^0(t)| + |\mathcal{U}_s^1(t)| = N_s$.

We design a distributed DDPG-based DRL algorithm with S actors located in SESs, each generating N_s BrA actions for their associated ECs. We assume that each small cell comprises an environment E, a set of possible states $\mathcal{S}$, and a set of available actions $\mathcal{A}$, with a reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$. At each time t, the SES observes the current state of the environment E, $s_t \in \mathcal{S}$, finds and executes an action $a_t \in \mathcal{A}$ according to a policy function π. Based on the action taken, the SES receives a scalar reward $r_t = r(s_t, a_t) \in \mathcal{R} \subseteq \mathbb{R}$ from the environment that is proportional to the QoE, and finds itself in the next state $s_{t+1} \in \mathcal{S}$ according to a transition probability $p(s_{t+1}|s_t, a_t)$. The actor's goal is to learn the optimal policy π^* that maximizes the expected long-term discounted reward from its actions, i.e., $R_t = \sum_{i=t}^{T} \gamma^{(i-t)} \cdot r(s_i, a_i)$, where $\gamma \in [0, 1]$ is the discount factor. In this paper, the DDPG-based DRL framework is based on the Wolpertinger Policy [48].

The actor: The actor network finds a proto-actor $\hat{a} \in \mathcal{A}$ from the set of valid actions. The obtained proto-actor is based on the decision policy of the network, which is updated after each decision. Specifically, the actor network is defined as a function parametrized by θ^{μ}, which maps the state $\mathcal{S}$ from the state space to the action space $\mathcal{A}$, according to the DDPG formulation. The mapping provides a proto-actor $\hat{a} \in \mathcal{A}$ for the current state and network parameters, $\mu(s|\theta^{\mu}) = \hat{a}$.

K-Nearest neighbours (KNN): The KNN component is used to map the action $\hat{a}$ to a set of valid actions selected from the action space $\mathcal{A}$. This is particularly useful in reducing the complexity of the DRL network in high-dimensional action spaces. The set of actions returned by the KNN component is denoted as $\mathcal{A}_k = g_k(\hat{a}_t)$ where $g_k = \arg\min_{a\in\mathcal{A}}^{k} |a - \hat{a}|_2$ is a k-nearest-neighbor mapping from a continuous space to a discrete set and returns the k actions in $\mathcal{A}$ that are closest to $\hat{a}$ by L_2 distance, i.e., $|a-\hat{a}|_2$, as stated in [48].

The critic: A critic network is used to refine the actor network actions, by evaluating the expanded actions of the KNN and choosing the action that provides the maximum Q-value, i.e., $a_t = \arg\max_{a_j\in\mathcal{A}_k} Q(s_t, a_j)$. The deterministic target policy of the critic network is defined as follows:

$$Q(s_t, a_j|\theta^Q) = \mathbb{E}_{r_t, s_{t+1}}\left[r(s_t, a_j) + \gamma Q(s_{t+1}, a_{t+1}|\theta^Q)\right],$$

where θ^Q stands for the parameters of the critic network. The critic network takes both the current state s_t and the next state s_{t+1} as input to calculate the Q value for each action in A_K. Then, the action with a maximum Q value is obtained, $a_t = \arg\max_{a_j\in\mathcal{A}_K} Q(s_t, a_j|\theta^Q)$.

5.2.2 DRL-based Solution for Decentralised BrA

In this section, we introduce a DRL-based approach, named Bitrate Neural Network (BNN), to solve the first term in $\widetilde{\mathbf{P1.1}}$. Using the DDPG algorithm, a decentralised dynamic BrA policy will be learned independently at each SES, which selects an action, i.e., bitrate allocated to the EU, upon the observation of the environment from its perspective. It must be noted that each agent (i.e., SES) has no prior knowledge of the environment, which means the number of ECs and statistics of bitrate demand and wireless bandwidth are unknown to the agent, and thus the learning process is model-free. The critic network $V(x)$ and the S actors $\pi_{\theta_i}(o_s)$, with $s = 1, 2, \ldots, S$ are parametrized by $\theta = \{\theta_c, \theta_s, \ldots, \theta_S\}$.

The NN receives requests from the N_s ECs of SBS s as input, denoted by $\mathcal{R}_s(t) = \{r_1(t), \ldots, r_{N_s}(t)\}$, and outputs the BrA decision vector $\hat{\mathcal{R}}_s(t) = \{\hat{r}_1(t), \ldots, \hat{r}_{N_s}(t)\}$. It should be noted that among N_s ECs only $|\mathcal{U}_s^1(t)|$ ECs have a new demand at slot t, and those with no new request will have an entry equal to 0 in $\mathcal{R}_s(t)$.

State Space: To reduce the overhead and make the DASH system much more scalable, we assume that the state of each agent is only determined by its local observation of the system, and each agent finds the action independently from the other agents. Full observation of the system includes the vector of queue lengths ($B(t)$), transcoding capacity, and stalling time at time t. From the perspective of the agent (SES), the state is defined as $s_{s,t} = [\bar{W}_s, \mathbf{B}_s(t), \bar{\Omega}_s, \bar{T}_w]$, where $\bar{*}$ represents the status of each parameter w.r.t. its maximum capacity threshold, and $\mathbf{B}_s(t)$ is the vector of the buffer level of the associated ECs.

Action Space: Based on the observed state of the system by each agent s, an action matrix $\bar{\mathcal{R}}_s(t) \in \mathbb{R}^{N_s\times\nu}$ is found, where ν is the total number of possibilities for streaming from all servers with or without transcoding. Since BNN addresses both BrA and USA problems, we define $\nu = 2 + |\mathcal{O}|$, where 2 refers to streaming from MBS or Cloud and the second term stands for $|\mathcal{O}|$ possible options for streaming from the SBS, i.e., a direct hit and $|\mathcal{O} - 1|$ options for transcoding. Each element of the matrix $\bar{\mathcal{R}}_s(t)$ shows the probability of assigning each EC to a bitrate available at the edge, the MES or the cloud shown in each column. Later, the matrix $\bar{\mathcal{R}}_s(t)$ is converted to vector $\hat{\mathcal{R}}_s(t) \in \mathbb{R}^{1\times N_s}$ by assigning the highest probable bitrate to each EC using a Softmax function. Since the SES is the agent and it has to make a decision for N_s ECs and for each EC there is ν number of actions, the action space is $(N_s)^\nu$. Moreover, since we are giving probabilities to these actions, the action space becomes huge and continuous. Unlike conventional DRL, which finds action from a predefined discrete action space, applying the DDPG algorithm, the BrA-USA problem can be greatly optimized in a continuous action space to minimize the first term in $\widetilde{\mathbf{P1.1}}$.

Reward Function: The goal of each agent is reward-driven, showing the importance of the reward function in the performance of the DRL algorithms. To learn adaptive BrA-USA allocation policies, we consider minimizing the objective function $\widetilde{\mathbf{P1.1}}$, while meeting the constraints **C4** and **C6**. Thus, the reward function must consider both the deviation in allocated bitrate and the penalty for transcoding computing resource constraints. Specifically, we define the reward function $r_{s,t}$ for each agent s at time t by

$$r_{s,t} = -\Bigg(w_{s,1}\cdot\sum_{i=1}^{N_s}\left(\Upsilon_{\phi_k^m}\left(\hat{r}_i(t), r_i(t), a_{i,j}\right)\right) + \sum_{i\in N_s} w_{s,2}\cdot p_i^{\text{trans}}(\hat{r}_i(t))\Bigg) \quad (22)$$

where $w_{s,*}$ is the non-negative weighted factor, and $r_{s,t}$[4] is the weighted sum of the instantaneous cost and the constraint **C4**. We define the penalty function, p_i^{trans}, for the transcoding computing resource constraint in **C4**:

$$p_i^{\text{trans}}(\hat{r}_i(t)) = \mathbb{E}_{\hat{r}_i}\left[\max\left(0, \sum_{i=1}^{N_s}\sum_{o=1}^{O}\eta_o^s\cdot\mathbb{R}_i^o - \Omega_s\right)\right] \forall i \in \mathcal{N}_s.$$

Remark 1. *The constraint* **C6** *is considered in the algorithm as a condition to be respected when performing the USA. Hence, it is not added to the above reward function.*

Remark 2. *In practical deployments, some edge units may function solely as storage nodes without local computational capability. In such cases, our framework models these storage-only nodes by setting their computation capacity to zero, allowing the BNN to avoid assigning processing tasks to them while the CO module continues to exploit their storage for caching and content delivery.*

After choosing the valid action from the action space $\mathcal{A}$ and calculating the observed reward, the critic network then estimates the value function $V(\mathbf{x})$, where $\mathbf{x}$ stands for the observation of all agents, $\mathbf{x} = \{o_1, o_2, \ldots, o_S\}$. At time t, after the actions $a_t = \{a_{1,t}, a_{2,t}, \ldots, a_{S,t}\}$ are found by the actor networks, the agents will execute the actions in the environment and send the environment to the critic. Feedback includes the reward r_t and the next instant observation $\mathbf{x}_{t+1}$. Then, the critic network calculates the temporal

4. Both terms have been normalized.

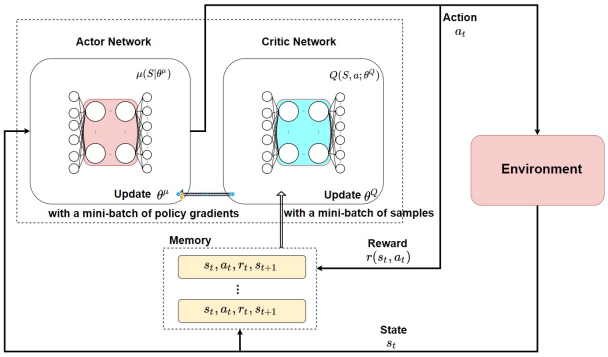

Fig. 5. DDPG-based DRL Actor/Critic Architecture.

difference (TD) error using $\delta^{\pi,\theta} = r_t + \gamma V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)$. Updating the critic network is performed by minimizing the least square TD difference, i.e., $V^* = \arg\min_V (\delta^{\pi\theta})^2$, where V^* is the optimal value function. The actor networks are updated using the policy gradient,

$$\nabla_{\theta_s} J(\theta_s) = E_{\pi\theta_s}\left[\nabla_{\theta_s} \log \pi\theta_s(o_s, a_s)\delta^{\pi\theta}\right], \quad (23)$$

where $\pi\theta_s(o_s, a_s)$ is the score of action a_s under the current policy. Then, the actor network s is updated using the gradient descent method as follows:

$$\theta_s \leftarrow \theta_s + \alpha\nabla_{\theta_s} \log \pi\theta_s(o_s, a_s)\delta^{\pi\theta}, \quad (24)$$

where $\nabla\theta_s = \alpha\nabla_{\theta_s} \log \pi\theta_s(o_s, a_s)\delta^{\pi\theta}$ is the weighted difference of parameters in actor network s, and α is the learning rate.

Fig. 5 shows the architecture of the decentralized BNNs. At each iteration, each agent (i.e., SES) observes the state and the EC's request and finds a proto-actor $\hat{a}_{s,t}$. Then, each of the proto-actors will be expanded into K actions using the KNN component, and the critic network will select an action with the highest Q value. Each possible action set can be denoted as $\mathbf{A}_h = (\hat{a_1}_K, \hat{a_2}_K, ..., \hat{a_S}_K)$, where $\hat{a_1}_K$ is the action chosen from the expanded set of SES S. In particular, we have K^S possible action combinations for all the SESs. The critic network evaluates all the combinations, and actions with the highest Q-value will be executed in the environment. Then the actors and critic networks are updated accordingly.

The detailed training stages of the BNNs are explained in Algorithm 1. To improve exploration performance, the interaction of SES agents manually starts with a random initial state s_{s+1} and ends at a predefined maximum step T_{max} for each episode. At each time step t during an episode, each agent's experience tuple $(s_{s,t}, a_{a,t}, r_{s,t}, s_{s,t+1})$ will be stored in its experience buffer $\mathcal{M}_s$. Then, the agent's actor and critic networks are updated accordingly using a mini-batch of K experience tuples $\{(s_i, a_i, r_i, s_i')\}$ randomly sampled from the replay memory $\mathcal{M}_s$. After training of E_{max} episodes, the dynamic BrA policy will be gradually and independently learned in each SES agent.

In the testing stage, each agent loads the actor-network parameters learned during training, initializes an empty buffer, interacts with a randomly initialized environment, and selects actions directly from the actor network based on the current state. To further enhance performance, *ResNet-inspired blocks* are introduced in the network to ensure efficient gradient flow, allowing deeper networks without vanishing gradients. Furthermore, *prioritized experience replay* is used to replay high-value experiences more frequently, boosting training efficiency. For optimization, an *adaptive learning rate scheduler* is employed to accelerate convergence. These techniques collectively enhance the stability and accuracy of the BNN training process [49].

Algorithm 1 Training stages of the BNNs

Input: $R(t)$ and $\Phi^{\text{SES}}(t)$, $\forall i \in N$ and $s \in S$.
for each agent $s \in S$ **do**
  Initialise the ResNet-based actor and critic networks' parameters.
  Initialize target networks $\theta_s^{\mu'} \leftarrow \theta_s^{\mu}$ and $\theta_s^{Q'} \leftarrow \theta_s^{Q}$.
  Initialize an empty experience memory $\mathcal{M}_s$.
end for
for each episode $e = 1, 2, ..., E_{max}$ **do**
  Generate an initial state $s_{s,1}$ randomly for all $s \in S$.
  for each step $t = 1, 2, ..., T_{max}$ **do**
    for each agent $s \in S$ **do**
      Determine the BrA action $a_{s,t}$ given the EC's demand, using current policy network θ_s^{μ} and exploration noise ϵ_μ.
      Execute action $a_{s,t}$ at each SES agent, receive reward $r_{s,t}$ and observe the next state $s_{s,t+1}$.
      Store the tuple $(s_{s,t}, a_{a,t}, r_{s,t}, s_{s,t+1})$ into the replay memory $\mathcal{M}_s$.
      Sample a mini-batch of K tuples from $\mathcal{M}_s$.
      Update the critic network by minimizing the loss L with the samples:
      $L = \frac{1}{K}\sum_{i=1}^{K}(r_i + \max_{a\in\mathcal{A}} Q(s_i', a|\theta_s^{Q'}) - Q(s_i, a_i|\theta_s^{Q}))^2$
      Update the actor network using the sampled policy gradient:
      $\nabla_{\theta_s^{\mu}} J = \frac{1}{K}\sum_{i=1}^{K} \nabla_a Q(s_i, a|\theta_s^{Q})|_{a=a_i} \nabla_{\theta_s^{\mu}} \mu(s_i|\theta_s^{\mu})$
    end for
    Send the BrA decision vector $\hat{\mathcal{R}_s}(t)$ to the CoNN, for all $s \in S$, and run Algorithm 2.
  end for
  if t mod $\delta = 0$ **then**
    Update the target networks by $\theta_s^{\mu'} \leftarrow \tau\theta_s^{\mu} + (1-\tau)\theta_s^{\mu'}$ and $\theta_s^{Q'} \leftarrow \tau\theta_s^{Q} + (1-\tau)\theta_s^{Q'}$.
  end if
end for

5.3 Phase 2: Heuristic-based BA Algorithm

While ABR significantly improves ECs' QoE, an EC in a dynamic environment may still experience stalling if the BA is inappropriate. For example, if ECs requesting the same chunk at different bitrates are all assigned fixed bandwidth, a high-bitrate request may suffer from insufficient downlink rate and buffer, whereas a low-bitrate request would stream smoothly. Thus, BA is a key component, alongside ABR, in maintaining high QoE. To measure the degree by which the BrA-USA decisions found by each BNN violate the constraints **C2**, **C3** and **C5** in $\mathbf{P1.1}$ (or the second term in $\widetilde{\mathbf{P1.1}}$), we define the BA cost function as

$$U(\hat{\mathcal{R}}_s) = \sum_{i\in N_s} w_{s,3} \cdot p_i^{\text{flow}}(\hat{r}_i(t)) + \sum_{i\in N_s} w_{s,4} \cdot p_i^{\text{stall}}(\hat{r}_i(t)) \quad (25)$$

which is the weighted sum of the constraints.

We define the penalty function, p_i^{flow}, to measure the degree to which BrA decisions found by the BNNs result in underflow and overflow in the ECs buffers, as shown in **C2**, **C3**. We introduce some buffer ranges $[\kappa_{*,\min}\ \kappa_{*,\max}]$ to measure the proximity of the buffer level to experience underflow or overflow. Let us define the best range[5] the

5. For instance in a $[40\%\ 60\%]$ of the buffer capacity.

buffer level can fall as $\delta_0 = \{b|\kappa_{0,\min} < b < \kappa_{0,\max}\}$, where b is the buffer level. Let us define a larger range that is more exposed to stalling by $\delta_1 = \{b|\kappa_{1,\min} < b < \kappa_{1,\max}, b \notin \delta_0\}$. Similarly we define L ranges while Lth range is $\delta_L = \{b|\kappa_{L,\min} < b < \kappa_{L,\max}, b \notin \delta_0 - \delta_{L-1}\}$, where $\kappa_{L,\min} < \kappa_{L-1,\min}$ and $\kappa_{L,\max} > \kappa_{L-1,\max}$:

$$p_i^{\text{flow}}(\hat{r}_i(t)) = \begin{cases} 0 & \text{if } B_i(t) \in \delta_0 \\ L & \text{if } B_i(t) \in \delta_L \end{cases} \tag{26}$$

where the selected bitrate has an impact on the chunk size in the downlink. Then, we define p_i^{stall} to measure the degree by which the BrA-USA decisions found by the BNNs result in stalling during streaming (constraint **C5**) as

$$p_i^{\text{stall}}(\hat{r}_i(t)) = \mathbb{E}_{\hat{r}_i}\left[\max\left(0, \sum_{k=0}^{K(t)} \tau_k(\hat{r}_i) - T_w\right)\right] \quad \forall i \in \mathcal{N}_s, \tag{27}$$

Remark 3. *Only ECs streaming from SESs receive their allocated bandwidth based on the proposed BA method, while those streaming from the MBS or cloud receive a fixed bandwidth. This highlights the dependency between the BA and USA problems.*

The proposed BA method, together with the BNN, ensures that $\mathbf{P1.1}$ is effectively addressed by handling resource allocation through DRL while enforcing constraint satisfaction via BA, achieving a balanced and feasible solution.

5.4 Phase 3: DDPG-based DRL Solution for Caching Mechanism

To optimize the placement of the videos on the SESs, we use a centralised DDPG-based DRL network placed at the MBS. The cache optimization network selects an action, i.e., the optimized set of videos with the appropriate bitrate to cache at each of the SESs, upon observation of the environment. The Caching Optimization NN (CoNN) has the same structure as the BrA network, with different state space, action space, and reward functions defined as follows.

State Space: Full observation of the environment for the cache-optimization DRL includes the matrix of videos and bitrates located at each SES (the latest update), $\bar{\Phi}^{\text{S}}$, and the storage space at the SESs, $\mathcal{C}(t)$, and is defined as $s_{c,t} = [\bar{\Phi}^{\text{S}}, \mathcal{C}(t)]$.

Action Space: Based on the agent's observed system state in the MBS, an action matrix decides which videos to cache on each SES and at what bitrate. With M videos, K chunks per videos, and upto o_{max} bitrates per chunk, the action space (all combinations of videos, chunks, and bitrates) becomes extremely large, making exploration intractable. To deal with this complexity, we consider a group of ψ chunks to be in a window represented as $Y_1 = \{v_{m,1} \dots v_{m,\psi}\}$, where $\sum_{i=1}^{\Psi} |Y_i| = K$, and Ψ is the total number of windows in a video. Therefore, we perform bitrate caching over a window instead of per chunk, reducing the action space. The problem then becomes twofold: (1) selecting which videos to cache, and (2) choosing the bitrate for each window. We now redefine the cache matrix, the agent's action, as:

$$\bar{\Phi}^{\text{S}} = \begin{pmatrix} \bar{\phi}_1^1 & \cdots & \bar{\phi}_\psi^1 & \cdots & \bar{\phi}_\Psi^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \bar{\phi}_1^M & \cdots & \bar{\phi}_\psi^M & \cdots & \bar{\phi}_\Psi^M \end{pmatrix} \tag{28}$$

where $\bar{\phi}_\psi^m = o_{m,\psi} \cdot \mathbb{1}_{\{\mathbb{I}_{m_\psi}^s = 1\}}$ where $\mathbb{I}_{m_\psi}^s$ is an indicator function which is 1 if the ψth window is cached at SES s. Thus, if a video is going to be cached, there should be a decision on the bitrate for each window.

Reward Function: To learn an adaptive cache optimization policy, we consider minimizing the objective function $\mathbf{P1.2}$, while meeting the constraint **C1**. Thus, the reward function must account for minimizing the difference between the requested and cached bitrate, and a penalty function on exceeding the caching capacity of the SESs. Specifically, we define the reward function $r_{c,t}$[6] for the cache optimization network at time t by

$$r_{c,t} = -\left(w_{c,1} \cdot \sum_{i=1}^{N} \left(\Lambda(\phi_k^m, r_i(t))\right) + w_{c,2} \cdot p_i^{\text{Cap}}(\hat{r}_i(t))\right) \tag{29}$$

where

$$p_i^{\text{Cap}}(\hat{r}_i(t)) = \mathbb{E}_{\hat{r}_i}\left[\max\left(0, \sum_{m=1}^{M}\sum_{i=1}^{\Psi}\sum_{o=1}^{O} \Delta_{o_{m,i}} \cdot x_{o_{m,i}}^s - C_s\right)\right]$$

To enhance the performance and stability of the CoNN, *ResNet blocks with residual connections* are introduced, allowing deeper architectures while mitigating vanishing gradient issues. Furthermore, *layer normalization* is applied at each layer to ensure stable training, and *dropout regularization* is used to prevent overfitting. The CoNN output uses a softmax function to convert the values of the last hidden layer, generating the caching decision matrix $\bar{\Phi}^{\text{S}}$. Each element of $\bar{\Phi}^{\text{S}}$, denoted as $\bar{\phi}_\psi^m$, shows the probability of caching the ψth window of video m with bitrate $o_{m,\psi}$ in SES s.

Fig. 6 shows the diagram of BNNs and CoNN working in parallel. At each iteration, the BrA decision vectors found by the BNNs, $\hat{\mathcal{R}}_s(t)$ for all $s \in S$, are sent to CoNN. To further improve the accuracy of caching decisions, the CoNN employs an *adaptive learning rate optimizer*, ensuring faster convergence [50]. Based on the observed state and the BrA decision vectors $\hat{\mathcal{R}}_s(t)$, the CoNN outputs the caching decision matrix $\bar{\Phi}^{\text{SES}}$. Then, the reward value $r_{c,t} = r(s_{c,t}, a_{a,t})$ is determined and the tuple $(s_{c,t}, a_{a,t}, r_{c,t}, s_{c,t+1})$ is saved in the CoNN memory $\mathcal{M}_c$. To avoid excessive bandwidth usage caused by constant transferring of video segments between the SESs, the caching algorithm, detailed in Algorithm 2, is executed with an optimized frequency. Specifically, CoNN cache decisions only take place every X iterations. In addition, *prioritized experience replay* is integrated into the caching algorithm to ensure that high-priority experiences are replayed more frequently, enhancing the quality of training and decision-making.

5.5 Overview of the Operations and Deployment

In this work, we address four key challenges-CO, USA, BrA, and BA-in video streaming services. Fig. 7 outlines the requirements and sequence of these operations. CO is handled at the MBS layer, while the other three are managed within each SBS. The DRL methods adopted in both BNN and CoNN are based on the DDPG algorithm, which allows continuous action control in high-dimensional environments. ECs in $\mathcal{U}_s^1(t)$ request services from their

6. Both terms have been normalized.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2026.3674131

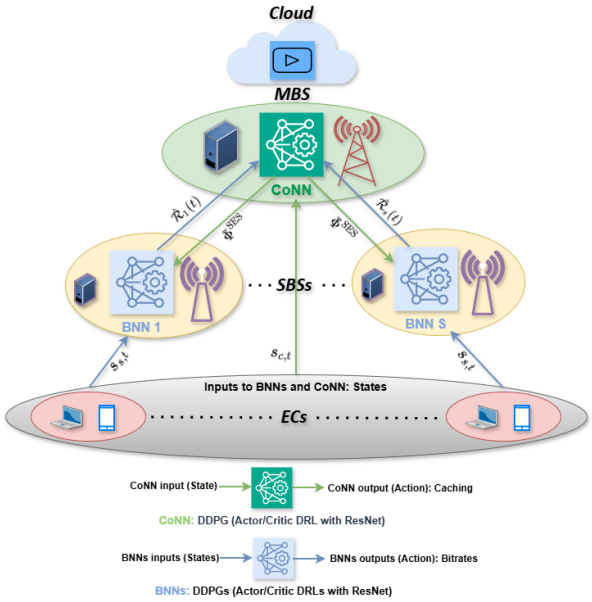

Fig. 6. Overall Architecture of CoNN-BNNs-Environment Interactions.

Algorithm 2 Training stages of the CoNN

Input: $R(t)$ for all $i \in N$, $\Phi^{\text{SES}}(t)$, for all $s \in S$, and $\hat{\mathcal{R}}_s(t)$ for all $s \in S$ received from the BNNs.

1: Initialise the ResNet-based actor and critic networks' parameters.
2: Initialize target networks $\theta_s^{\mu'} \leftarrow \theta_s^{\mu}$ and $\theta_c^{Q'} \leftarrow \theta_c^{Q}$.
3: Initialize the empty experience memory $\mathcal{M}_c$.
4: **for** each episode $e = 1, 2, ..., E_{max}$ **do**
5: Generate an initial state $s_{c,1}$ randomly.
6: **for** each step $t = 1, 2, ..., T_{max}$ **do**
7: Determine the caching allocation action $a_{c,t}$ using current policy network θ_c^{μ} and exploration noise ϵ_{μ}.
8: **if** t mod $X = 0$ **then**
9: Execute the caching allocation action and relocate videos on SESs.
10: **end if**
11: Determine the receive reward $r_{c,t}$ of action $a_{c,t}$ and observe the next state $s_{c,t+1}$.
12: Store the tuple $(s_{c,t}, a_{c,t}, r_{c,t}, s_{c,t+1})$ into the replay memory $\mathcal{M}_c$.
13: Sample a mini-batch of K tuples from $\mathcal{M}_c$.
14: Update the critic network by minimizing the loss L with the samples:
$L = \frac{1}{K}\sum_{i=1}^{K}(r_i + \max_{a\in\mathcal{A}} Q(s_i', a|\theta_c^{Q'}) - Q(s_i, a_i|\theta_c^{Q}))^2$
15: Update the actor network using the sampled policy gradient:
$\nabla_{\theta_c^{\mu}} J = \frac{1}{K}\sum_{i=1}^{K} \nabla_a Q(s_i, a|\theta_c^{Q})|_{a=a_i} \nabla_{\theta_c^{\mu}} \mu(s_i|\theta_c^{\mu})$
16: **end for**
17: **if** t mod $\delta = 0$ **then**
18: Update the target networks by $\theta_c^{\mu'} \leftarrow \tau\theta_c^{\mu} + (1-\tau)\theta_c^{\mu'}$ and $\theta_c^{Q'} \leftarrow \tau\theta_c^{Q} + (1-\tau)\theta_c^{Q'}$.
19: **end if**
20: **end for**

associated SBS, which forwards requests to its SES. The BNN then processes these using the SES cache and state space, generating the USA-BrA decision matrix. The BA module allocates bandwidth, with ECs receiving either a portion based on the proposed BA approach or a fixed share from other sources. The outputs of operations 2, 3, and 4 inform the ECs where to stream from, the available bitrate, and their bandwidth allocation. ECs update their buffers for each slot upon successful download or playback. Meanwhile, operation 1 (i.e., CO) runs a CoNN every X slots to update the SES caching matrices.

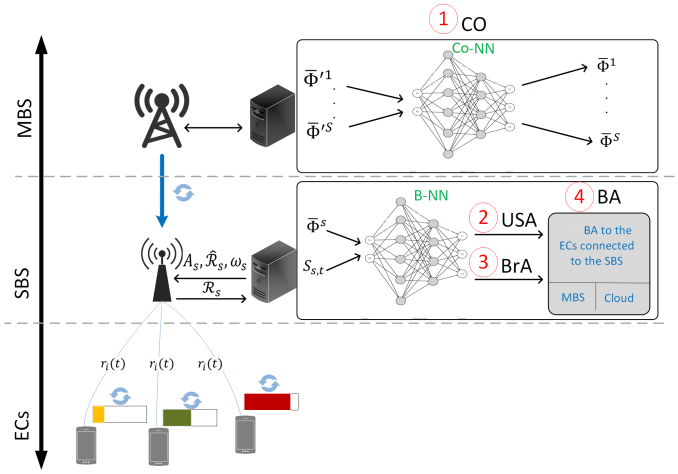

Fig. 7. DASH Architecture.

The proposed framework is designed with a modular architecture, where BA algorithm, CoNN and BNN operate autonomously within their respective layer, enabling distributed and scalable implementation (see Figs. 6 and 7). The BNN runs locally at each SES, trained on local user data and network observations, without requiring central coordination. They also run the lightweight BA algorithm, requiring no cross-layer synchronization. The CoNN, though, runs on the MES as a supervisory caching manager periodically updating cache states based on aggregated request statistics received from the SESs, similar to coordination mechanisms in Content Delivery Networks. This update frequency can be tuned to network capabilities, ensuring low communication overhead. The proposed framework can be practically deployed on current MEC and cloud infrastructures using containerized services and lightweight APIs. DRL components at the Edge and Macro layers can run as independent virtualized agents trained offline and updated asynchronously. The system requires only metadata exchange between layers, ensuring low communication overhead. The BA algorithm is handled in real time through network-slicing APIs. This distributed and service-oriented design enables incremental deployment on current 5G and edge-cloud infrastructures without significant reconfiguration.

6 Simulation Results

In this section, we present the simulation results generated using Python under different settings. A single MBS is located in the center of a circular area that covers 5 SBSs. We considered ECs to be distributed in an area with a Poisson distribution with an expected density of S/N ECs per cell.

SES cache capacity is set to $C_s = 10$ Gb, and the buffer length of each EC (i.e., $\mathbf{B}$) is considered to be 100 s [17]. The cache capacity at MES is enough to cache only the highest bitrate for all videos (each video will have approximately 4.8 Gb). We set $T_{\text{W}} = 20$ s. We consider 10 videos, each of equal length of 10 minutes [51]. We also consider that different videos have different chunk sizes in the range $[4 \quad 6]$ s. We select the 6 most popular resolutions for traditional systems as 240P, 360P, 480P, 720P, 1080P, and 1440P. The bitrate ranges are listed in Tab. 2 which are

TABLE 2
The video bitrate for each resolution

Resolution	Video bitrate range	Selected bitrate
240P	[0.3 0.7] Mbps	0.3, 0.4 Mbps
360P	[0.4 1] Mbps	0.4, 0.7, 1 Mbps
480P	[0.5 2] Mbps	0.6, 0.8, 2 Mbps
720P	[1.5 4] Mbps	1.5, 1.8, 2.75 Mbps
1080P	[3 6] Mbps	3.5 Mbps
1440P	[6 13] Mbps	8 Mbps

obtained from YouTube[7] similar to [4]. We characterize the popularity of videos using a Zipf-like distribution and sort the videos in $\mathcal{V}$ in descending order of their popularity $P_m = \{p_1, \dots p_i \dots, p_M\}, \sum_{i=1}^{M} p_i = 1$, where p_m represents the popularity of the ith rank video defined as

$$p_i = \frac{i^{-\eta}}{\sum_{j=1}^{M} j^{-\eta}} \tag{30}$$

where $\eta = 0.8$ is the popularity skewness of videos of $\mathcal{V}$. Hence, each EC selects a video based on its popularity. In a heterogeneous environment, devices have different interests in terms of bitrate selection. Hence, we consider each EC uniformly at random selects one bitrate level for its videos; however, with a 5% probability for each time slot, the selected bitrate downgrades to a lower one due to various reasons (e.g., interference). Later, after BrA, one of the bitrates in the third column of Tab. 2 is uniformly selected at random based on the requested resolution. These different bitrate values for each resolution represent different video content in different scenes. We also assume ECs with a probability of 10% start streaming from the first chunk, and with a 90% probability select a chunk randomly from the middle of the video to show real-world-like user behaviour. ECs download the chunks until the play-out buffer is full; such an aggressive approach is also exploited on YouTube [52].

The bandwidth is allocated to each EC after the USA and the allocation of the bitrates, as explained in Section 5.3. We set $W_s = 40$ MHz. The chunks are added to the buffer after they are downloaded and the buffer decreases as the chunks are being displayed. For processing capacity, we set $\Omega^s = 30$ GHz, which is the maximum number of CPU cycles per second. The number of CPU cycles per Byte has been established at 5900, based on findings from the existing literature [51]. The termination condition for an episode is when 80% of the ECs experience overflow/underflow. In the following, we show the impact of different parameters on the result. To ensure statistical reliability, all DRL experiments were conducted using 20 independent random seeds for the CoNN and 30 for the BNNs, with identical hyperparameters and variations only in network initialization, environment stochasticity, and exploration noise. The weighting parameters for (22), (25), and (29) are determined using a two-phase procedure consisting of root mean square-based scale normalization followed by coarse grid search fine-tuning with respect to the QoE metrics defined in Section 4.2. This approach ensures balanced contributions of all reward components while enabling controlled trade-offs among competing objectives.

7. https://support.google.com/youtube/answer/2853702?hl=en

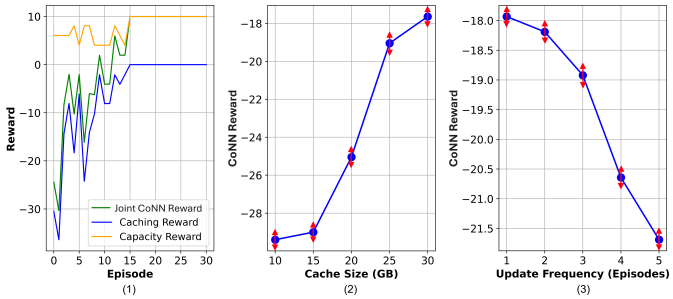

Fig. 8. (1) Caching, capacity and joint CoNN rewards versus episodes, (2) CoNN reward vs cache size, and (3) CoNN reward vs update frequency.

6.1 CoNN Performance

Fig. 8.1 illustrates the joint CoNN reward, which consists of the caching reward and the capacity reward. The CoNN reward represents the reward obtained by the trained MBS in updating the cache of the SBSs. As shown, the reward value converges in less than 45 episodes. Fig. 8.2 demonstrates the impact of cache size on CoNN reward. With a larger cache size, more video chunks can be stored, increasing the likelihood that the requested video chunks are available in the SBS cache. This figure represents the CoNN performance over X steps. Fig. 8.3 highlights the impact of the frequency of running CoNN to update the SBS cache. Running CoNN with the highest frequency results in the highest reward per iteration, while lower frequencies, such as every 5 steps, yield a reduced reward. This decrease occurs because some requests are missed, leading to a lack of knowledge about intermediate requests, resulting in higher errors. The error bars in Figs. 8(2) and 8(3) represent the standard deviation across the 20 runs at each evaluation step.

6.2 BNN Performance

Fig. 9 illustrates the joint BNN reward and its components: the BrA reward and the transcoding reward. The developed BNNs are deployed across 5 SBSs, allowing us to observe the performance of each SBS individually and the average performance of all 5 SBSs in the final plot. As shown, all SBSs converge after several episodes, with the average convergence occurring around 150 episodes. The variation in convergence time across the SBSs is influenced by factors such as the number of ECs within each SBS and the diversity of requested videos.

Fig. 10.1 demonstrates the impact of computational capacity on the joint BNN reward across the 5 SBSs and their average reward. The increased computational capacity enables for more transcoding, allowing the delivery of more suitable bitrates to the ECs. This reduces the bitrate cost and, in turn, increases the BNN reward. Fig. 10.2 examines the impact of the number of ECs on the BNN reward. With fixed computational, caching, and bandwidth capacities, a higher number of ECs leads to a greater gap between the delivered and requested bitrates. This results in a lower joint BNN reward. Across the BNN experiments (Figs. 9 and 10), the observed standard deviations remain consistently low (all below 1% of the dynamic range), indicating strong robustness to random initialization, which we attribute to the

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2026.3674131

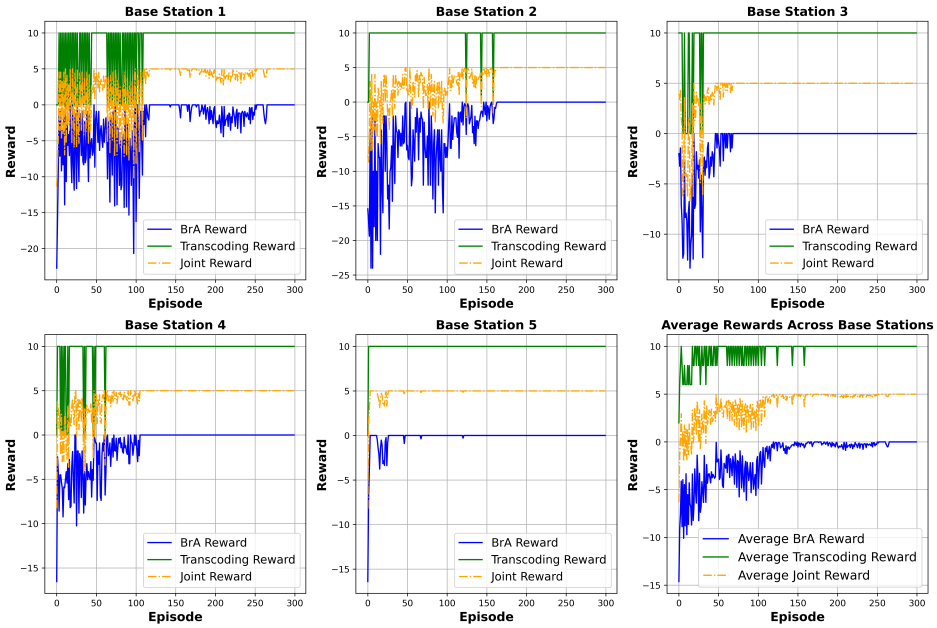

Fig. 9. Bitrate, transcoding and joint BNN rewards versus episodes.

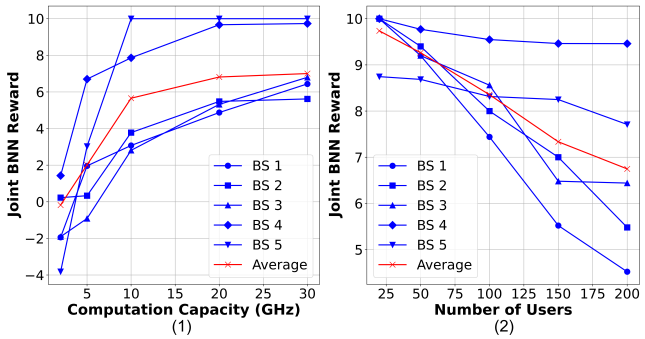

Fig. 10. Joint BNN reward versus: (1) computation capacity, and (2) number of users.

ResNet-inspired architecture, prioritized experience replay, and adaptive learning rate scheduling.

6.3 BA Performance

Fig. 11 illustrates the buffer utility in 5 SBSs, showing the average buffer utility of all ECs within each SBS and all ECs in the last plot. The figure depicts how buffer utility evolves over simulation steps for buffer lengths ranging from 100 to 300 seconds, comparing Fair Bandwidth Allocation (FBA) and the proposed BA algorithm. The buffer utility peaks when the EC buffer levels reach 40–60% of their capacity. For example, in Base Station 1, the buffer utilities for capacities of 100, 200, and 300 seconds reach their maximums at approximately 25, 40, and 50 steps, respectively. FBA, which allocates bandwidth based on requested bitrates, results in overloading, causing the buffer utility to drop back to zero. In contrast, the BA algorithm is buffer-level-aware, assigning a higher bandwidth at lower buffer levels and reducing the bandwidth as the buffers approach overload, keeping the buffer levels within the optimal 40–60% range.

Fig. 12 illustrates the stalling and the buffer utility components across 5 SBSs, with the average across all SBSs shown in the final plot. As previously described, the buffer utility starts at zero and reaches its maximum when the BA algorithm maintains the buffer level within 40–60% of its capacity (convergence time). The stalling reward, on the other hand, exhibits a decreasing trend. Initially, when the stalling constraint is not violated, the stalling reward is at its highest. However, as simulation steps progress, some ECs may experience stalling due to empty buffers caused by long download times, which in turn result from low allocated bandwidth. As the simulation continues, the BA algorithm improves performance, stabilising the buffer levels of all ECs within the 40–60% range. Consequently, stalling is eliminated, and both the buffer utility and stalling reward converge at approximately the same time.

6.4 Inference Latency

To validate the deployment feasibility and latency of the proposed DRL agents, we measured their inference performance across hardware platforms representing the full spectrum from edge devices to cloud infrastructure. As summarized in Tab. 3, both models demonstrate excellent real-time performance. The BNN achieves sub-millisecond inference times on common laptop-class CPUs (Intel i5/i7) that are representative of capable edge servers, making it highly suitable for the frequent BrA-USA decisions. The CoNN also performs well on these platforms with inference times under 5 ms, which is negligible compared to its operational time scale of minutes. On cloud GPUs/TPUs, both models achieve exceptional sub-millisecond performance, demonstrating the scalability of our approach. These results confirm that the computational overhead of our DRL solutions does not hinder their practical deployment under real-world latency constraints.

6.5 Overall Performance

To evaluate the performance of all developed algorithms (BNNs, CoNN, and the BA algorithm) on overall QoE and

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2026.3674131

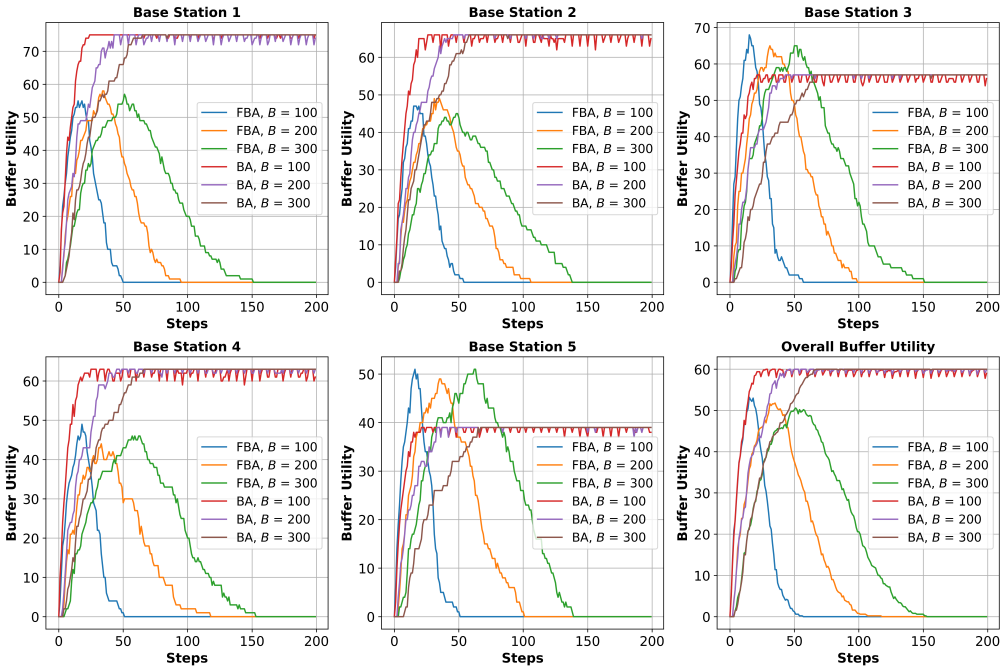

Fig. 11. Buffer utility for different values of buffer length versus steps.

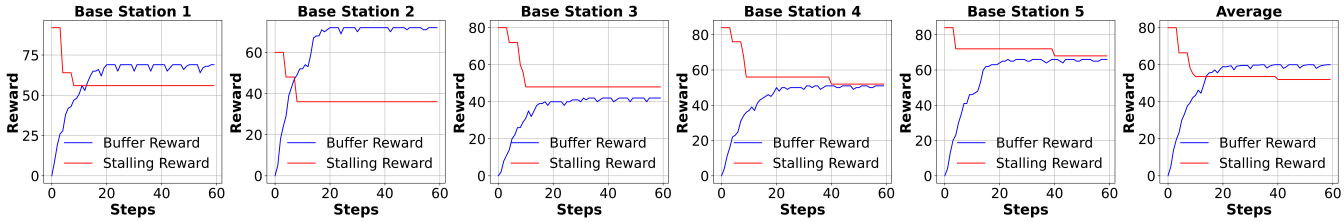

Fig. 12. Buffer and stalling rewards versus steps.

the studied KPIs, we present Fig. 13 for three NN steps: 20, 40, and 60. This figure illustrates how the QoE of ECs evolves over time. The demonstration covers a small part of the network (10 ECs served by one SBS, alongside the MBS and cloud), but similar performance is observed across all SBSs and ECs. As shown in Fig. 13.a, the BNN initially performs poorly due to errors in the delivered bitrate compared to the requested bitrate, resulting in only 50% QoS satisfaction. This improves to 70% and 100% in Fig. 13.b and Fig. 13.c, respectively. The impact of the BA algorithm is evident in the buffer levels, with satisfaction increasing from 40% to 70% and then 100% in the three figures. This demonstrates the BA algorithm's ability to dynamically allocate bandwidth, effectively avoiding underflow and overflow. CoNN's performance is reflected in the MBS caching, with QoS satisfaction increasing from 20% to 80% over the steps. Furthermore, the utilization of the streaming source improves from 50% to 80%, a direct result of the improved caching at the edge. For example, when EC 4 requests the 62nd chunk of the first video in quality 6, the absence of this chunk at the edge redirects the request to MBS, which stores all chunks at the highest quality. Although caching satisfaction is not achieved for this request, the continuum effectively delivers the required bitrate, demonstrating the efficiency of the integrated algorithms.

7 Conclusion and Future Works

In this work, we tackled the challenges of adaptive video streaming in an edge-assisted DASH framework by formulating a joint optimization problem encompassing bitrate allocation, user-to-server assignment, caching, and bandwidth allocation. Given the complexity of the overall problem, we decomposed it into manageable sub-problems and employed DDPG to solve three sub-problems with continuous action spaces (defined as BNN and CoNN), and designed a heuristic approach for bandwidth allocation.

In the simulations, we evaluated the performance of the BNN- and CoNN-based approaches from multiple perspectives, including the impact of cache size, cache update frequency, convergence behaviour, computational capacity, and the number of users. Additionally, we assessed the effectiveness of the BA algorithm. Finally, we evaluated the overall performance of the proposed solutions. As the number of decision steps increases, we observe that: (a) video chunks and bitrates are cached more efficiently based on user demand, (b) buffer underflows and overflows are minimized, (c) streaming shifts closer to the edge, and (d) a higher proportion of users receive their requested video chunks and bitrates. This demonstrates a scalable and effective solution for intelligent video delivery in Edge-DASH environments.

Future extensions of this work could explore a collabora-

TABLE 3
Inference latency of the DRL models across computing platforms

Hardware Category	Platform Specification	Model	Avg. Inference Time (ms)	95th Percentile (ms)	Deployment
CPUs	Intel Core i5-1135G7	BNN	0.85	1.3	Typical SES/MES performance
	Intel Core i5-1135G7	CoNN	3.8	5.2	MES performance
	Intel Core i7-1265U	BNN	0.62	0.95	High-end SES performance
	Intel Core i7-1265U	CoNN	2.9	4.1	High-end MES performance
Cloud GPUs	NVIDIA T4 GPU	BNN	0.25	0.38	Cloud/MES with accelerator
	NVIDIA T4 GPU	CoNN	1.1	1.6	Cloud/MES with accelerator
	NVIDIA A100 GPU	BNN	0.08	0.12	High-performance cloud
	NVIDIA A100 GPU	CoNN	0.35	0.52	High-performance cloud
	NVIDIA L4 GPU	BNN	0.15	0.22	Modern cloud inference
	NVIDIA L4 GPU	CoNN	0.68	0.95	Modern cloud inference
Cloud TPUs	Google v5e-1 TPU	BNN	0.12	0.18	Specialized AI accelerator
	Google v5e-1 TPU	CoNN	0.52	0.75	Specialized AI accelerator
	Google v6e-1 TPU	BNN	0.09	0.14	Next-gen AI accelerator
	Google v6e-1 TPU	CoNN	0.41	0.61	Next-gen AI accelerator

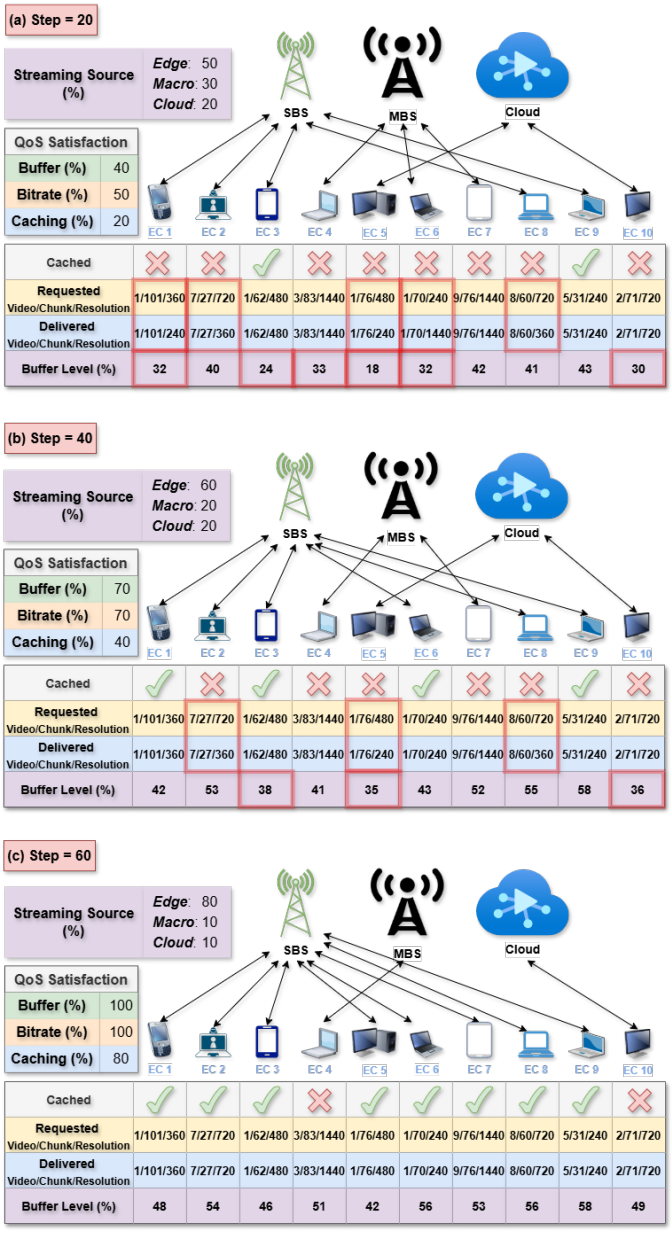

Fig. 13. Overall Performance after: (a) 20, (b) 40, and (c) 60 steps.

tive multi-agent reinforcement learning framework, where SESs can share experiences to accelerate learning and improve global resource utilization. Furthermore, incorporating user mobility into the environment model would enable a more realistic evaluation of caching and delivery strategies under dynamic network conditions, particularly in scenarios involving frequent handovers. Finally, implementing and testing the proposed methods using real-world network traces or on edge computing testbeds would offer valuable insights into their practical feasibility and performance in real deployments.

REFERENCES

[1] PwC, "Global telecom outlook 2023–2027," 2023, accessed: Jan. 11, 2025. [Online]. Available: https://www.pwc.com/gx/en/industries/tmt/telecommunications/outlook-2024-2028.html

[2] Ericsson, "Ericsson mobility report: Key figures," 2023, accessed: Jan. 11, 2025. [Online]. Available: https://www.ericsson.com/en/reports-and-papers/mobility-report/reports/november-2025

[3] S. Kumar, D. S. Vineeth, and A. F. A, "Edge Assisted DASH Video Caching Mechanism for Multi-access Edge Computing," in *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2018, pp. 1–6.

[4] D. Wang, Y. Peng, X. Ma, W. Ding, H. Jiang, F. Chen, and J. Liu, "Adaptive wireless video streaming based on edge computing: Opportunities and approaches," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 685–697, 2019.

[5] X. Li, M. A. Salehi, M. Bayoumi, and R. Buyya, "CVSS: A cost-efficient and QoS-aware video streaming using cloud services," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, 2016, pp. 106–115.

[6] A. Bozorgchenani, F. Mashhadi, D. Tarchi, and S. A. Salinas Monroy, "Multi-objective computation sharing in energy and delay constrained mobile edge computing environments," *IEEE Trans. Mobile Comput.*, vol. 20, no. 10, pp. 2992–3005, 2021.

[7] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "Centralized and distributed architectures for energy and delay efficient fog network-based edge computing services," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 1, pp. 250–263, 2019.

[8] H. A. Pedersen and S. Dey, "Enhancing mobile video capacity and quality using rate adaptation, RAN caching and processing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 2, pp. 996–1010, 2016.

[9] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Edge computing assisted adaptive mobile video streaming," *IEEE Trans. Mobile Comput.*, vol. 18, no. 4, pp. 787–800, 2019.

[10] B. Wei, H. Song, M. Nakamura, K. Kimura, N. Togawa, and J. Katto, "QuDASH: Quantum-inspired rate adaptation approach for DASH video streaming," *IEEE Access*, vol. 11, pp. 118 462–118 473, 2023.

[11] P. Yang, N. Zhang, S. Zhang, F. Lyu, L. Yu, and X. Shen, "Asymptotic optimal edge resource allocation for video streaming via user preference prediction," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[12] E. Ozfatura, O. Ercetin, and H. Inaltekin, "Optimal network-assisted multiuser DASH video streaming," *IEEE Trans. Broadcast.*, vol. 64, no. 2, pp. 247–265, 2018.

[13] L. Li, R. Hou, R. Li, H. Li, M. Pan, and Z. Han, "Delay-aware adaptive wireless video streaming in edge computing assisted ultra-dense networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[14] A. Bokani, M. Hassan, S. Kanhere, and X. Zhu, "Optimizing HTTP-based adaptive streaming in vehicular environment using markov decision process," *IEEE Trans. Multimedia*, vol. 17, no. 12, pp. 2297–2309, 2015.

[15] S. Bayhan, S. Maghsudi, and A. Zubow, "EdgeDASH: Exploiting network-assisted adaptive video streaming for edge caching," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1732–1745, 2021.

[16] B. Guo, X. Zhang, Y. Wang, and H. Yang, "Deep-Q-network-based multimedia multi-service QoS optimization for mobile edge computing systems," *IEEE Access*, vol. 7, pp. 160 961–160 972, 2019.

[17] L. Tao, Y. Gong, S. Jin, and J. Zhao, "Energy-efficient predictive HTTP adaptive streaming in mobile cellular networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 069–11 083, 2018.

[18] Y. Hayamizu, K. Goto, M. Bandai, and M. Yamamoto, "QoE-Aware Bitrate Selection in Cooperation With In-Network Caching for Information-Centric Networking," *IEEE Access*, vol. 9, pp. 165 059–165 071, 2021.

[19] L. Zhong, M. Wang, C. Xu, S. Yang, and G.-M. Muntean, "Decentralized Optimization for Multicast Adaptive Video Streaming in Edge Cache-Assisted Networks," *IEEE Trans. Broadcast.*, vol. 69, no. 3, pp. 812–822, 2023.

[20] J. P. Esper, A. Claudia Bastos Loureiro Monção, K. B. Chaves Rodrigues, C. Bonato Both, S. L. Corrêa, and K. Vieira Cardoso, "QoE-DASH: DASH QoE Performance Evaluation Tool for Edge-Cache and Recommendation," in *ICC 2022 - IEEE International Conference on Communications*, 2022, pp. 757–762.

[21] F. E. Subhan, A. Yaqoob, C. H. Muntean, and G.-M. Muntean, "EDQD: An Edge-Driven Multi-Agent DRL Solution for Improving Joint QoE in DASH-based Rich Media Content Delivery," in *2024 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, 2024, pp. 1–7.

[22] Y. Lu, J. Qi, S. Zhang, G. Luo, A. Zhu, J. Wu, and Z. Qian, "ABUV: Adaptive bitrate and upsampling for video streaming on mobile devices," *Computer Networks*, vol. 257, p. 110994, 2025.

[23] D. Naseh, A. Bozorgchenani, and D. Tarchi, "Deep Reinforcement Learning for Edge-DASH-Based Dynamic Video Streaming," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–6.

[24] A. Mehrabi, M. Siekkinen, and A. Ylä-Jääski, "Energy-aware QoE and backhaul traffic optimization in green edge adaptive mobile video streaming," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 828–839, 2019.

[25] 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.

[26] B. Jedari and M. D. Francesco, "Auction-based cache trading for scalable videos in multi-provider heterogeneous networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 1864–1872.

[27] Z. Zhang, J. Dai, M. Zeng, D. Liu, and S. Mao, "Scalable video caching for information centric wireless networks," *IEEE Access*, vol. 8, pp. 77 272–77 284, 2020.

[28] L. Wu and W. Zhang, "Caching-based scalable video transmission over cellular networks," *IEEE Commun. Lett.*, vol. 20, no. 6, pp. 1156–1159, 2016.

[29] Y. Guo, Q. Yang, F. R. Yu, and V. C. M. Leung, "Cache-enabled adaptive video streaming over vehicular networks: A dynamic approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 6, pp. 5445–5459, 2018.

[30] T. X. Tran, P. Pandey, A. Hajisami, and D. Pompili, "Collaborative multi-bitrate video caching and processing in mobile-edge computing networks," in *2017 13th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, 2017, pp. 165–172.

[31] D. Xenakis, "To DASH, or Not to DASH? Optimal Video Bitrate Selection and Edge Network Caching in MEC-Empowered Slice-Enabled Networks," *IEEE Trans. Veh. Technol.*, vol. 73, no. 4, pp. 5556–5571, 2024.

[32] Y. Jin, Y. Wen, and C. Westphal, "Optimal transcoding and caching for adaptive streaming in media cloud: an analytical approach," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 12, pp. 1914–1925, 2015.

[33] C. Li, L. Toni, J. Zou, H. Xiong, and P. Frossard, "QoE-driven mobile edge caching placement for adaptive video streaming," *IEEE Trans. Multimedia*, vol. 20, no. 4, pp. 965–984, 2018.

[34] Y. Han, R. Wang, and J. Wu, "Random caching optimization in large-scale cache-enabled Internet of Things networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 385–397, 2020.

[35] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1431–1445, 2013.

[36] A. Tran, N. Dao, and S. Cho, "Bitrate adaptation for video streaming services in edge caching systems," *IEEE Access*, vol. 8, pp. 135 844–135 852, 2020.

[37] R. Behravesh, D. F. Perez-Ramirez, A. Rao, D. Harutyunyan, R. Riggio, and R. Steinert, "ML-driven DASH content pre-fetching in MEC-enabled mobile networks," in *2020 16th International Conference on Network and Service Management (CNSM)*, 2020, pp. 1–7.

[38] D. Raca, A. H. Zahran, C. J. Sreenan, R. K. Sinha, E. Halepovic, R. Jana, V. Gopalakrishnan, B. Bathula, and M. Varvello, "Empowering video players in cellular: Throughput prediction from radio network measurements," in *Proceedings of the 10th ACM Multimedia Systems Conference*, ser. MMSys '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 201–212.

[39] S. Petrangeli, J. V. D. Hooft, T. Wauters, and F. D. Turck, "Quality of experience-centric management of adaptive video streaming services: Status and challenges," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 2s, pp. 1–29, May 2018.

[40] A. Balachandran, V. Sekar, A. Akella, S. Seshan, I. Stoica, and H. Zhang, "Developing a predictive model of quality of experience for internet video," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 339–350, Aug. 2013.

[41] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of HTTP adaptive streaming," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 469–492, 2015.

[42] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A tutorial on video quality assessment," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 1126–1165, 2015.

[43] P. Reichl, B. Tuffin, and R. Schatz, "Logarithmic laws in service quality perception: where microeconomics meets psychophysics and quality of experience," *Telecommunication Systems*, vol. 52, no. 2, pp. 587–600, 2013.

[44] K. Tammer, "The application of parametric optimization and imbedding to the foundation and realization of a generalized primal decomposition approach," in *Parametric Optimization and Related Topics*, ser. Mathematical Research, J. Guddat, H. T. Jongen, B. Kummer, and F. Nožička, Eds. Akademie-Verlag Berlin, 1987, vol. 35, p. 376–386.

[45] D. P. Palomar and M. Chiang, "A tutorial on decomposition methods for network utility maximization," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 8, pp. 1439–1451, 2006.

[46] J. Jiang, S. Zhang, B. Li, and B. Li, "Maximized cellular traffic offloading via device-to-device content sharing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 82–91, 2016.

[47] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer Science & Business Media, 2006.

[48] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[49] D. Naseh, M. Abdollahpour, and D. Tarchi, "Real-world implementation and performance analysis of distributed learning frameworks for 6G IoT applications," *Information*, vol. 15, no. 4, p. 190, 2024.

[50] L. Ridolfi, D. Naseh, S. S. Shinde, and D. Tarchi, "Implementation and evaluation of a federated learning framework on raspberry PI platforms for IoT 6G applications," *Future Internet*, vol. 15, no. 11, 2023.

[51] S. Rezvani, S. Parsaeefard, N. Mokari, M. R. Javan, and H. Yanikomeroglu, "Cooperative multi-bitrate video caching and transcoding in multicarrier NOMA-assisted heterogeneous virtualized MEC networks," *IEEE Access*, vol. 7, pp. 93 511–93 536, 2019.

[52] H. W. Barz and G. A. Bassett, *Multimedia Networks: Protocols, Design and Applications*. Wiley, 2016.