Mobile Networks and Applications
https://doi.org/10.1007/s11036-024-02311-1

RESEARCH

DRL Empowered On-policy and Off-policy ABR for 5G Mobile Ultra-HD Video Delivery

Naresh Mandan[1] · **Paresh Saxena**[1] · **Manik Gupta**[1]

Accepted: 17 March 2024
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract
Fifth generation (5G) and beyond 5G networks support high-throughput ultra-high definition (UHD) video applications. This paper examines the use of dynamic adaptive streaming over HTTP (DASH) to deliver UHD videos from servers to 5G-capable devices. Due to the dynamic network conditions of wireless networks, it is particularly challenging to provide a high quality of experience (QoE) for UHD video delivery. Consequently, adaptive bit rate (ABR) algorithms are developed to adapt the video bit rate to the network conditions. To improve QoE, several ABR algorithms are developed, the majority of which are based on predetermined rules. Therefore, they do not apply to a broad variety of network conditions. Recent research has shown that ABR algorithms powered by deep reinforcement learning (DRL) based vanilla asynchronous advantage actor-critic (A3C) methods are more effective at generalizing to different network conditions. However, they have some limitations, such as a lag between behavior and target policies, sample inefficiency, and sensitivity to the environment's randomness. In this paper, we propose the design and implementation of two DRL-empowered ABR algorithms: (i) on-policy proximal policy optimization adaptive bit rate (PPO-ABR), and (ii) off-policy soft-actor critic adaptive bit rate (SAC-ABR). We evaluate the proposed algorithms using 5G traces from the Lumos 5G dataset and show that by utilizing specific properties of on-policy and off-policy methods, our proposed methods perform much better than vanilla A3C for different variations of QoE metrics.

Keywords Deep reinforcement learning · Actor-critic methods · Quality of experience (QoE) · Adaptive bit rates (ABR) · Video streaming

1 Introduction

5G wireless networks have several key characteristics such as high data rates, low latency, massive device connectivity, high reliability, and so on. These characteristics collectively make 5G a transformative technology that promises faster and more reliable wireless connectivity, supporting a wide range of applications across industries and improving overall user experiences. Due to the widespread use of the 5G wireless network, the volume of multimedia traffic, including video streaming, has increased significantly. As per the Ericsson Mobility Report [1], we now delve into a detailed examination of two widely used mobile applications: video streaming and web browsing. The collective usage of these applications is projected to account for over 80% of the total mobile traffic share by the year 2025. In addition, the demand for video traffic has increased with the introduction of ultra-high definition (UHD) or 4K/8K video streaming. Recently, it has been shown that 5G Standalone (SA) service, via dynamic adaptive streaming over HTTP (DASH) [2], can deliver up to 95% UHD video segments within the deadline, whereas LTE network failed to deliver more than 20% of UHD video segments within the deadline [3].

To ensure seamless video streaming, DASH uses an adaptive bitrate (ABR) module to adjust the video bitrate based on network conditions. Recently, Pensieve [4] has been proposed as a data-driven deep reinforcement learning (DRL) [5] approach to improve the ABR algorithms. It uses vanilla asynchronous advantage actor-critic (A3C) [6] with several workers and one central agent. Each worker contains an actor and a critic where the actor generates a policy, i.e., a map-

✉ Naresh Mandan
p20180420@hyderabad.bits-pilani.ac.in

Paresh Saxena
psaxena@hyderabad.bits-pilani.ac.in

Manik Gupta
manik@hyderabad.bits-pilani.ac.in

[1] Computer Science & Information Systems, BITS Pilani, Jawahar Nagar, Hyderabad 500078, Telangana, India

Published online: 28 March 2024

ping from the state to the action, and the critic identifies the effectiveness of the policy generated by the actor. Several such workers are trained in parallel to identify the best action given the current state of the environment. However, vanilla A3C suffers from several drawbacks: (i) there is a lag between each actor's behavior policy and the target policy of the central agent. As a result, the behavior and target policies become out of synchronization, resulting in suboptimal updates, (ii) there is no control over the policy parameters, and hence it leads to instability in policy updates, (iii) it relies on collecting new samples for nearly every update to the policy, making them sample inefficient, and (iv) it is sensitive to the environment's randomness because entropy maximization is not directly incorporated into the maximum expected cumulative reward. Due to these constraints, the integration of vanilla A3C for ABR generation may result in imprecise throughput prediction when there are fluctuations in the network, re-buffering at the client's device, and inaccurate bitrate selection impacting the overall QoE for the users.

This paper addresses the aforementioned issues by proposing the design and implementation of two DRL-empowered ABR algorithms: (i) on-policy proximal policy optimization adaptive bit rate (PPO-ABR), and (ii) off-policy soft-actor critic adaptive bit rate (SAC-ABR) algorithms. The aforementioned issues (i) and (ii) are addressed using an on-policy PPO-ABR for adaptive bitrate streaming and the issues (iii) and (iv) are addressed using an off-policy SAC-ABR for adaptive bitrate streaming. The PPO-ABR improves video QoE by maximizing sample efficiency using a clipped probability ratio between the new and the old policy parameters on multiple epochs of minibatch updates instead of a single epoch, and the SAC-ABR aims to achieve a better video QoE by maximizing entropy while maximizing the expected rewards and reusing past experiences using a replay buffer pool. This work is an extension of our recently published conference papers: [7] and [8]. In these papers, we present the initial sketch of the two proposed algorithms. In this paper, we present the detailed design and their evaluation over 5G traces from Lumos 5G dataset [9] focusing on the UHD video delivery. Specifically, the main contributions of this paper are as follows:

- We propose two DRL-empowered ABR algorithms: an on-policy PPO-ABR and an off-policy SAC-ABR. The experimental results show that our proposed algorithms can achieve a higher QoE over other state-of-the-art ABR algorithms. Our results show that PPO-ABR and SAC-ABR significantly improve QoE and provide up to 47.82% and 155.69% higher QoE, respectively, as compared to vanilla-A3C based Pensieve.
- To the best of our knowledge, this is the first work to propose both on-policy and off-policy DRL methods for generating ABR. By utilizing specific properties of on-policy and off-policy methods, our proposed algorithms overcome the challenges faced by vanilla A3C based ABR algorithms.
- We evaluated the proposed approaches using 5G traces from Lumos 5G datasets [10]. The dataset covers 5G throughput traces with a data rate of up to 1000 Mbps and hence supports the transmission of UHD video. We have used a 4K video for the evaluation and consider several QoE variants. Our results show the benefits of the proposed approaches over state-of-the-art ABR methods with several QoE variants.

The rest of the paper is organized as follows: Section 2 presents the relevant background on reinforcement learning, on-policy, and off-policy methods. Section 3 presents the system model and the proposed algorithms. We present the experimental setup and results in Section 4. Finally, we conclude our work in Section 5.

2 Background

This section presents the background on reinforcement learning and on-policy and off-policy RL algorithms.

2.1 Reinforcement Learning

RL is the process of dynamic learning with little or no prior information about the environment [5]. In order to maximize a long-term reward in the future, the agent makes judgments by learning from mistakes. Using a Markov decision process (MDP), the interactions between the agent and the environment are described where at time step $t = 0, 1, 2, 3, ...$, the agent is in a state s_t, performs an action $a_t \in A$, and then observes the reward $r_t = R(s_t, a_t)$. The agent's objective is to identify a policy $\pi(s, a)$ that maximizes the expected return determined by $V^*(s)$. It is given by,

$$V^*(s) = \max_{\{\pi \in \Pi\}} V_\phi^\pi(s) \tag{1}$$

where Π is a set of all possible policies and $V_\phi^\pi(s)$ is the value function with adjustable policy parameter (ϕ). This function is defined as the average discounted sum of future rewards. It is given by,

$$V_\phi^\pi(s) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, \pi_\phi\right] \tag{2}$$

where $\gamma \in [0, 1)$ is a discount factor. Similarly, the action-value function $Q_\phi^\pi(s, a)$ is defined as follows:

$$Q_\phi^\pi(s, a) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi_\phi\right] \tag{3}$$

where the optimal action-value function is given by,

$$Q^*_\phi(s,a) = \max_{\{\pi \in \Pi\}} Q^\pi_\phi(s,a) \tag{4}$$

The optimal policy $\pi^*_\phi(s)$ is directly obtained from $Q^*_\phi(s,a)$, where,

$$\pi^*_\phi(s) = \arg\max_{a \in A} Q^*_\phi(s,a) \tag{5}$$

In this paper, we focus on policy gradient methods [11] that are used to find an optimal policy by solving Eq. 5. The policy gradient methods are divided into two categories: on-policy and off-policy methods. On-policy DRL methods are used to evaluate and improve the same policy that is being used to select the actions, whereas off-policy DRL methods are used to evaluate and improve a policy different from the generated policy that is used for action selection. In the following two sections, we present the background on the on-policy and off-policy RL methods.

2.2 On-policy DRL methods

The basic on-policy method is a vanilla policy gradient method [12] where policy parameters are updated after the calculation of the total reward at the end of the episode instead of single-step. The policy gradient is given by:

$$\nabla_{\phi_k} = \sum_{t=0}^{T} \nabla_\phi \log \pi_\phi(a_t, s_t)|_{\phi_k} A_\phi(s,a) \tag{6}$$

where $A_\phi(s,a) = Q^\pi_\phi(s,a) - V^\pi_\phi(s)$ is the advantage function, ∇_ϕ is the policy optimization using a gradient operator, T is the number of steps in the episode and ϕ_k is the current policy parameters. However, the vanilla policy gradient suffers from high variance and high training time due to value estimates being calculated at the end of the episodes instead of every time step. To address these issues, actor-critic methods [12] are proposed. These methods consist of two parts: an actor represented by a policy π and a critic represented by an estimate of the action-value function. Typically, neural network function approximators are used to represent both of them. The critic, with parameters θ, estimates the current policy's value function. The main goal of this method is to reduce the variance using single-step state-value estimates. The single-step state-value estimates are derived using a temporal difference (δ), and it is given by:

$$\delta = V^\pi_\phi(s_t) + \gamma V^\pi_\phi(s_{t+1}, \phi) - V^\pi_\phi(s_t, \phi) \tag{7}$$

The policy and critic updates with respect to its parameters ϕ and θ, respectively, are defined in terms of the gradient operator ∇ as follows:

$$\Delta\phi = \phi + \alpha_p \delta \nabla \pi_\phi(s_{t+1}, a_{t+1}, \phi) \tag{8}$$

$$\Delta\theta = \theta + \alpha_c \delta \nabla V^\pi_\phi(s_t, \theta) \tag{9}$$

where α_p and α_c are the actor and critic learning rates, respectively. Furthermore, as an improvement, vanilla-A3C [6] is proposed that uses several copies of the same agent with asynchronous updates. It is more efficient than the actor-critic methods because samples for data can be parallelized using several copies of the same agent resulting in an even smaller training time. In the vanilla-A3C algorithm, the current policy parameters (ϕ_{new}) are updated based on previously collected experience with old policy parameters (ϕ) after every κ step, i.e., after every κ state-action pairs. The equation below represents the value function update for vanilla-A3C

$$\text{maximize}_\phi \; V^\pi_{\phi_{new}}(s) = \kappa \nabla V^\pi_\phi(s) + \kappa \sum_s \rho_{\pi_\phi}(s) \sum_a \pi_{\phi_{new}}(a|s) A_\phi(s,a) \tag{10}$$

where $\rho_\pi(s)$ is the distribution of state-action pairs, π_ϕ is the old policy and $\pi_{\phi_{new}}$ is the current policy. Note that $\sum_a \pi_{\phi_{new}}(a|s) A_\phi(s,a) \geq 0$ guarantees to increase the value function, however, $\sum_a \pi_{\phi_{new}}(a|s) A_\phi(s,a) < 0$ can result in a decrease in the value function resulting in increasing divergence between the old policy and the new policy.

To alleviate this issue, the on-policy trust region policy optimization (TRPO) [13] proposes Kullback-Leibler (KL) divergence [14] constraint to update the value function. The Eq. 10 is rewritten with KL divergence constraint as follows:

$$\begin{aligned} &\text{maximize}_\phi \; V^\pi_{\phi_{new}}(s) = \kappa \nabla V^\pi_\phi(s) + \kappa E_{s \sim \rho_{\pi_\phi}, a \sim \pi_\phi} \Big[r(\phi) \, A_\phi(s,a) \Big] \\ &\qquad \text{subject to } D_{KL}(\pi_{\phi_{\text{new}}} || \pi_\phi) \leq \lambda \end{aligned} \tag{11}$$

where $r(\phi) = \frac{\pi_{\phi_{new}}(s,a)}{\pi_\phi(s,a)}$ is the importance sampling ratio, $D_{KL}(\pi_{\phi_{\text{new}}} || \pi_\phi) = \sum_a \pi_{\phi_{new}}(s,a) \log \left(\frac{\pi_{\phi_{new}}(s,a)}{\pi_\phi(s,a)} \right)$ and $D_{KL}(\pi_{\phi_{\text{new}}} || \pi_\phi) \leq \lambda$ is used to constrain the divergence between the old and new policies with λ as a KL-divergence limit, $\lambda \in (0, 1]$. We can rewrite Eq. 11 to maximize only the second part, also known as the surrogate advantage objective, as follows:

$$\begin{aligned} &\text{maximize}_\phi \; \kappa E_{s \sim \rho_{\pi_\phi}, a \sim \pi_\phi} \Big[r(\phi) \, A_\phi(s,a) \Big] \\ &\qquad \text{subject to } D_{KL}(\pi_{\phi_{\text{new}}} || \pi_\phi) \leq \lambda \end{aligned} \tag{12}$$

Although TRPO provides constraints on the divergence between the old and the new policies, there is no control on the policy parameters. Hence, it would lead to instability in policy updates. To address this issue, the on-policy proximal policy optimization (PPO) algorithm [15] is proposed that uses a clipped probability ratio to constrain the divergence between the old and the new policy parameters. The objective function in PPO is derived from Eq. 12, and the maximization problem is given as:

$$\begin{aligned} &\text{maximize}_{\phi} L^{clip}(\phi_{new}) = \kappa E_t\left[\min\left(L^{CPI}(\phi), clip(r(\phi), 1-\epsilon, 1+\epsilon)A_{\phi}(s,a)\right)\right] \\ &\text{subject to } D_{KL}(\pi_{\phi_{\text{new}}} || \pi_{\phi}) \leq \lambda \end{aligned} \tag{13}$$

where ϵ is the hyperparameter for clipping and $L^{CPI}(\phi) = \kappa E_t\left[r(\phi)\, A_{\phi}(s,a)\right]$ where CPI refers to a conservative policy iteration. From Eq. 13, the first term represents the TRPO unclipped surrogate objective and the second term represents a modification of the TRPO surrogate objective using a clipped probability ratio ϵ, which removes the incentive for moving $r(\phi)$ outside of the interval $[1-\epsilon, 1+\epsilon]$. The PPO maximization considers the minimum of the clipped and unclipped objectives resulting in a smaller divergence between the new and the old policy parameters.

2.3 Off-policy RL methods

In this section, we describe state-of-the-art off-policy RL methods including DDPG [16], and SAC [17]. These methods consist of two policies: target and behavior policies [5] to generate actions. The target policy is used to select the particular action to maximize the action-value estimate, whereas the behavior policy is used to sample all possible actions. DDPG is an off-policy actor-critic method with the following actor update:

$$\pi^*_{\phi}(s) = \text{maximize}_{\phi}\, E_{b\sim s,a,r,s'}\left[Q^{\pi}_{\phi}(s, \pi_{d_{\phi}}(a))\right] \tag{14}$$

where $\pi_{d_{\phi}}$ is the target policy, and b is the behavior policy. The critic update is given by,

$$Q_{\theta}(s,a) = Q_{\theta_{old}}(s,a) + \alpha\left[r + \max_{a'} \gamma\, Q_{\theta}(s',a') - Q_{\theta_{old}}(s,a)\right] \tag{15}$$

where $Q_{\theta}(s,a)$ is the updated action-value, $Q_{\theta_{old}}(s,a)$ is the old action-value, $r + \max_{a'}\ \gamma\, Q_{\theta}(s',a')$ is the target action-value, and $\left[r + \max_{a'}\ \gamma\, Q_{\theta}(s',a') - Q_{\theta_{old}}(s,a)\right]$ is the temporal difference error. However, DDPG fails because of the overestimation of the action-value due to the use of the maximization function. To alleviate the issues with DDPG, SAC [17] uses double Q-trick [18] with entropy maximization. The double Q-trick takes a minimum of two action-values to alleviate the overestimation of the action-value function, and the entropy maximization encourages the exploration to maximize the expected return. The critic network uses soft policy evaluation to compute the soft action-value of a policy as follows:

$$\tau^{\pi} Q_{\theta}(s_t, a_t) = r(s_t, a_t) + \gamma\, E_{s_{t+1}\sim p}[V_{\theta}(s_{t+1})] \tag{16}$$

where τ the Bellman backup operator and soft state-value function $V_{\theta}(s_t)$ is given by,

$$V_{\theta}(s_t) = E_{a_t\sim\pi}\left[Q_{\theta}(s_t, a_t) + \eta\mathcal{H}(\pi_{\phi}(\cdot|s_t))\right] \tag{17}$$

where η balances the exploration-exploitation trade-off and $\mathcal{H}()$ is the entropy function that encourages the exploration of a given policy. Using the two soft action-value functions, the target update $y(r, s^{'})$ and the loss function $L(\theta)$ are given by,

$$y(r, s^{'}) = r + \gamma \min_{\{i=1,2\}} Q_{\theta_{i,\text{targ}}}(s^{'}, a^{'}(s^{'})) + \eta\mathcal{H}(\pi_{\phi}(\cdot|s_t)) \tag{18}$$

$$L(\theta) = E\left[(Q_{\theta}(s_t, a_t|\theta) - y(r, s^{'}))^2\right] \tag{19}$$

and the parameter θ is updated as follows:

$$\theta \leftarrow \theta - \alpha_c \nabla_{\theta} L(\theta) \tag{20}$$

where α_c is the critic learning rate.

The equation below represents the gradient of $L(\theta)$, which is given by,

$$\nabla_{\theta} L(\theta) = \nabla_{\theta} Q_{\theta}(s_t, a_t)[Q_{\theta}(s_t, a_t) - y(r, s^{'}))] \tag{21}$$

Furthermore, the actor uses soft policy iteration to achieve the optimal policy π^* by maximizing the expected rewards along with entropy as follows:

$$\pi^*_{\phi} = \arg\max_{\phi} \sum_{t}^{T} E_{(s,a)\sim\rho_{\pi}}\left[r(s, \tilde{a}^{\pi}_{\phi}) + \eta\mathcal{H}(\pi_{\phi}(\tilde{a}^{\pi}_{\phi}|s))\right] \tag{22}$$

Additionally, SAC also uses the reparametrization trick [19] for normalization. It allows us to rewrite the expectation over actions into an expectation over noise using a tangent hyperbolic of a Gaussian distribution, where $\tilde{a}^{\pi}_{\phi} = tanh(\mu_{\phi} + \epsilon\sigma_{\phi})$ and $\epsilon \sim N(0,1)$ is the input noise vector. The $tanh$ function ensures that actions are bounded to a

finite range. Based on the policy improvement approach of the actor's network, the loss function $L(\phi)$ is defined as:

$$L(\phi) = \arg\max_{\phi} E\left[\min_{\{i=1,2\}} Q^{\pi}_{\phi_i}(s, \tilde{a}^{\pi}_{\phi}) + \eta\mathcal{H}(\pi_{\phi}(\tilde{a}^{\pi}_{\phi}|s))\right] \tag{23}$$

where the loss function $L(\phi)$ also addresses the overestimation by using two soft action-value functions and encourages the exploration using entropy maximization. Using the gradient ascent method, the actor parameter (ϕ) is updated as follows:

$$\phi \leftarrow \phi - \alpha_p \nabla_{\phi} L(\phi) \tag{24}$$

where α_p is the learning rate of the actor-network and $\nabla_{\phi} L(\phi)$ is the gradient of $L(\phi)$, given by:

$$\nabla_{\phi} L(\phi) = \nabla_{\phi} \sum_{s,a,r,s'} \left[\min_{\{i=1,2\}} Q^{\pi}_{\phi_i}(s, \tilde{a}^{\pi}_{\phi}) + \eta\mathcal{H}(\pi_{\phi}(\tilde{a}^{\pi}_{\phi}|s))\right] \tag{25}$$

In this section, we have presented various on-policy and off-policy RL methods with advantages and disadvantages. Specifically, our proposed algorithms are developed over on-policy PPO and off-policy SAC methods. It is shown that PPO provides high sample efficiency by reducing a divergence between new and old policy parameters over vanilla-A3C and SAC seeks to optimize the trade-off between exploration and exploitation by increasing entropy while also increasing expected rewards than the vanilla-A3C method.

3 System model and proposed approaches

Figure 1 presents the system model for multimedia streaming using the DASH framework. It involves a client-server architecture where video chunks (segments) are stored on the video server and transferred to the client over a 5G network. The client establishes a connection with the video server using an HTTP request. The video server stores the video into multiple chunks, which are denoted as $[c_1, c_2 ... c_F]$, and each chunk is associated with different bitrate quality levels, which are denoted as $[q_1, q_2, ... q_F]$. On the client side, an ABR controller selects the appropriate bitrate for each video chunk. The ABR controller will make the action about the particular chunk being played at the specified quality of the bitrate based on several inputs such as estimates of throughputs, video player buffer occupancy, network statistics, number of chunks (c_t), chunk size (n_t), last chunk bitrate (b_{t-1}), size of the buffer (l_t), past chunk throughput (x_{t-1}), and past chunk download time (d_{t-1}). After the selection of the bitrate of the particular chunk, the ABR controller passes the chunk information to the throughput estimator and buffer controller for playing the next chunk bitrate.

3.1 On-policy PPO-ABR: architecture and algorithm design

The proposed PPO-ABR is an on-policy DRL method where multiple agents are trained in parallel as shown in Fig. 2. The initial sketch of the PPO-Algorithm is presented in [8]. The inputs and output to PPO-ABR are as follows.

- Input: The input to PPO-ABR is current state $s_t = (x_{t-1}, d_{t-1}, n_t, b_{t-1}, c_t, l_t)$.
- Output: The output from PPO-ABR is the selection of the chunk's bitrate from action $a_t = b_t$.

The components of PPO-ABR are explained as follows.

3.1.1 Environment

Each worker interacts with its own environment to produce the following input and output.

- Input: The input for the environment is the current state (s_t) and action (a_t).
- Output: The output of the environment is the next state (s_{t+1}) and a reward (r_t).

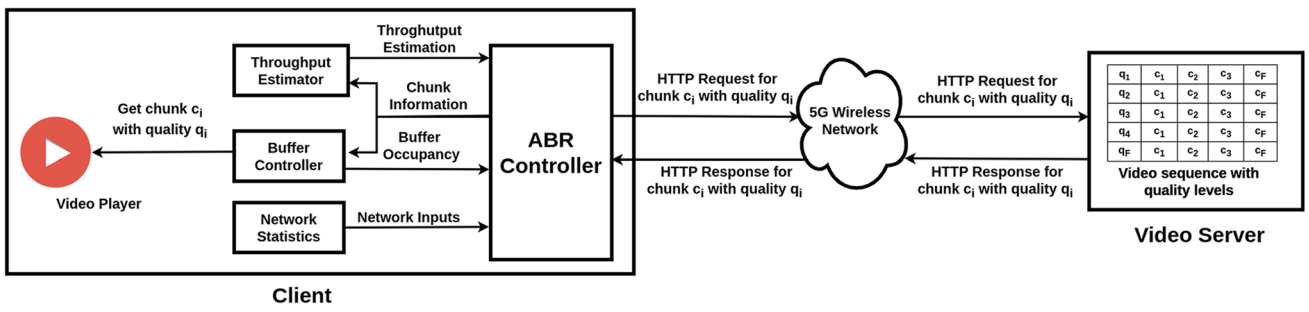


Fig. 1 System Model

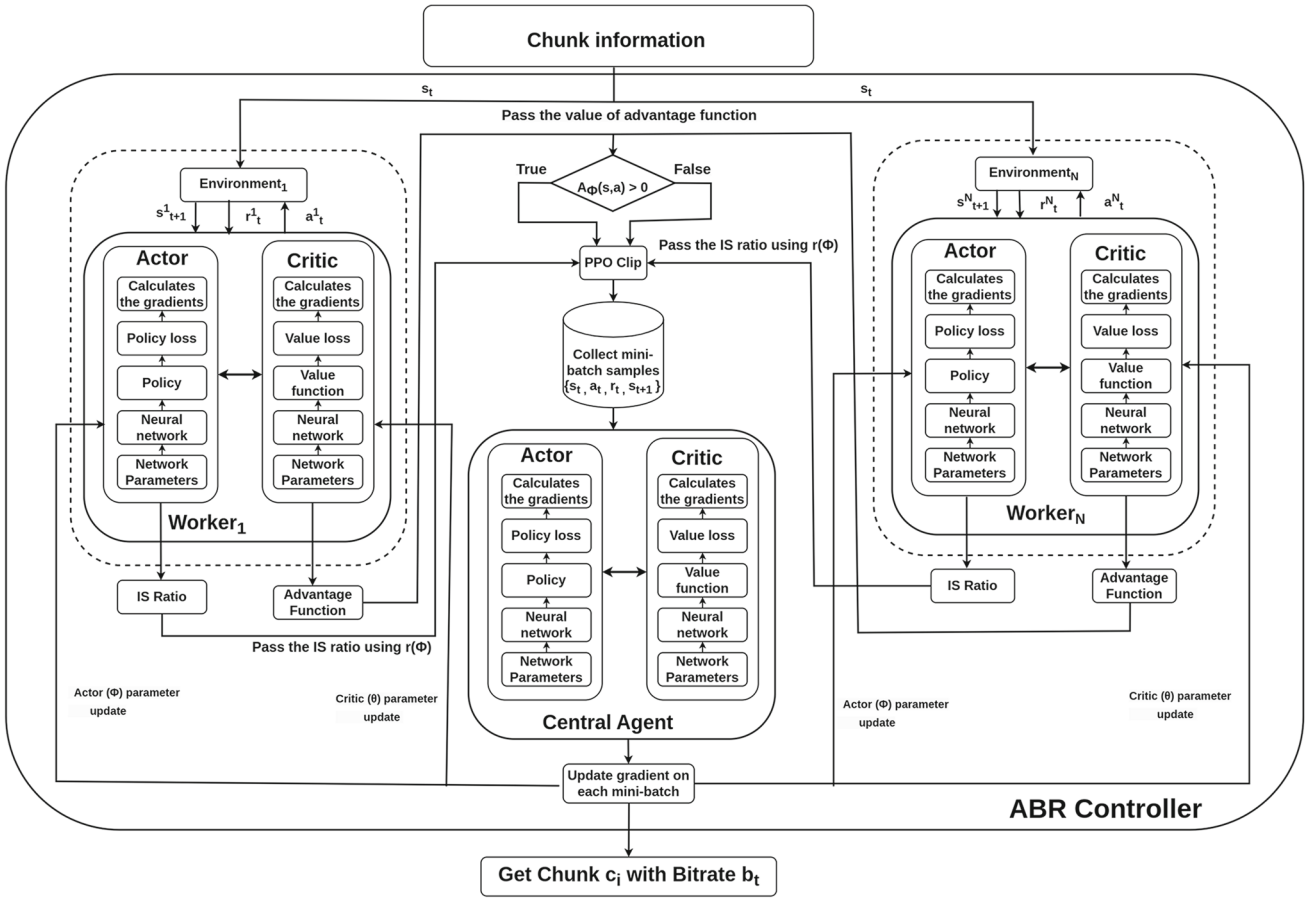


Fig. 2 Proposed Architecture for PPO-ABR

3.1.2 Workers

Each worker has the actor and critic neural networks. The actor calculates the gradients with policy parameters (ϕ) using Eq. 8, and the critic calculates the gradients with critic parameters (θ) using Eq. 9. After gathering mini-batch samples from each worker, the samples are then transmitted to the central agent.

3.1.3 IS ratio

In PPO-ABR, IS ratio estimates the chunk's bitrate by the expected value function using the probability distribution between two policies using Eq. 11.

- Input: The input for the IS ratio is the probability values of the new policy ($\pi_{\phi_{new}}$) and old policy (π_{ϕ}).
- Output: The output for IS ratio is a single scalar value based on the probability ratio between $\pi_{\phi_{new}}$, π_{ϕ}.

3.1.4 Advantage function

The advantage function estimates the advantage of taking a specific action in a given state compared to the expected value of being in that state following the current policy. If the advantage function value is greater than zero, IS ratio clips at $1+\epsilon$ otherwise IS ratio clips at $1-\epsilon$ as shown by Eq. 11.

- Input: The inputs to the advantage function are the Q-function and the value-function, which are evaluated by the critic network.
- Output: The output of the advantage function is a scalar value, representing the advantage of taking a specific action a_t in a given state s_t under the current policy.

3.1.5 PPO clip

Additionally, PPO clip constrains the divergence between the old and the new policy using Eq. 13.

- Input: The PPO clip takes three scalar values as inputs: the IS ratio and the positive and negative scalar values of the advantage function.
- Output: The output for the PPO clip is a scalar value, which is used to adjust the policy parameters.

3.1.6 Central agent

The central agent collects the experiences from all the workers, and the central agent performs asynchronous updates of the action parameters (ϕ) and the critic parameters (θ). The updated parameters are then sent to the workers. The output of the ABR controller is the bitrate to select for the next chunk, i.e., $a_t = b_t$.

3.2 Off-policy SAC-ABR: architecture and algorithm design

The proposed SAC-ABR is an off-policy DRL method where multiple agents are trained in parallel as shown in Fig. 3. The initial sketch of the PPO-Algorithm is presented in [7]. The input and output of the SAC-ABR are as follows.

- Input: The input to SAC-ABR is current state $s_t = (x_{t-1}, d_{t-1}, n_t, b_{t-1}, c_t, l_t)$.
- Output: The output from SAC-ABR is the selection of the chunk's bitrate from action $a_t = b_t$.

The mapping of the output to the input is explained with the help of the following components.

3.2.1 Environment

Each worker has an environment to act in a particular state.

- Input: The input for the environment is the current state (s_t) and action (a_t).
- Output: The output of the environment is the next state (s_{t+1}) and a reward (r_t).

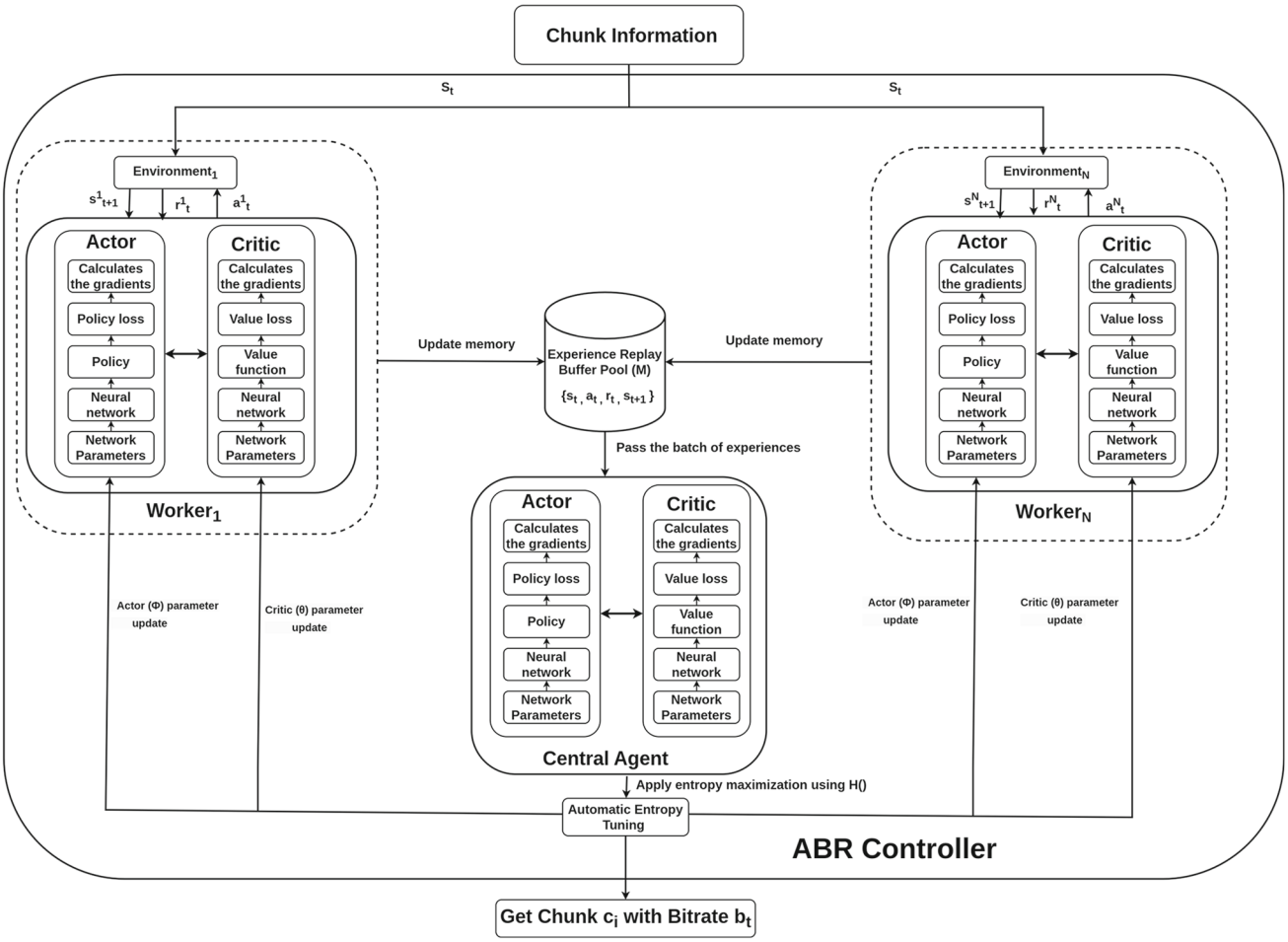


Fig. 3 Proposed Architecture for SAC-ABR

3.2.2 Workers

The SAC-ABR uses multiple workers for parallel training, as shown in Fig. 3. Each worker interacts with its own environment. Each worker consists of the actor and critic neural networks. The actor calculates the gradients based on policy loss using Eq. 23, and the critic calculates the gradients are updated based on value loss using Eq. 19. Finally, each worker sends the current information, i.e., (s_t, a_t, r_t, s_{t+1}), to experience replay buffer pool (M).

3.2.3 Experience replay buffer pool

The functionality of the experience replay buffer pool is used to collect and store the experiences (s_t, a_t, r_t, s_{t+1}) from every worker. The experience replay buffer pool helps to improve sample efficiency, as the worker can learn from past experiences without requiring new interactions with the environment for each update. After collecting experiences, the replay buffer pool randomly sends a batch of experiences to the central agent.

3.2.4 Central agent

The central agent collects the set of experiences from workers and makes asynchronous updates for the actor's parameters (ϕ) and critic's parameters (θ) using Eqs. 20 and 24, respectively. The updated parameters are then passed to the workers. While sending the parameters to the workers, the central agent use automatic entropy tuning using Eq. 17 to adjust the entropy regularization coefficient automatically.

4 Experimental details and Results

This section describes the experimental environment, including the 5G traces used in the experiments, the training methodology, the performance evaluation metrics, and the results.

4.1 5G Traces

We utilized mmWave 5G traces from the Lumos 5G dataset [10]. It consists of 173 5G throughput traces that were collected at 1-second intervals. The range of throughput between 0 and 1800 Mbps is adequate for UHD video delivery over 5G networks. Each trace file includes a timestamp (in seconds) and throughput (in megabits per second). We utilized two different classes of 5G traces: (i) 5G-Drive, which encompasses driving-related mobility patterns. There are 92 such traces and (ii) 5G-Walk, which addresses walking-related mobility patterns. There are 81 such traces.

4.2 Training methodology

We compared the proposed SAC-ABR and PPO-ABR with the following state-of-the-art DRL-based ABR algorithms.

- Pensieve [4]: DRL-based vanilla-A3C method to generate ABR.
- AL-FFEA3C [20]: DRL-based follow-then-forage method to generate ABR.
- AL-AvgA3C [20]: DRL-based averaged-A3C method to generate ABR.
- ALISA [21]: DRL-based importance-weighted actor-learner method to generate ABR.

Each model has been trained for 100,000 iterations, with a discount factor of $\gamma = 0.99$ and learning rates of 0.0001 and 0.001 for actor and critic, respectively. In order to maximize the entropy, we used an entropy regularization from 1 to 0.01 for a better exploration-exploitation tradeoff, i.e., initially, an entropy value of one is used for a few iterations, and then it is gradually decreased to 0.01. It takes approximately eight hours to complete the training for each model. Experiments utilise 4K video with a frame rate of 80 frames per second and a resolution of 4096 x 2160 pixels. The aggregate size of the video is 500MB and it consists of 158 chunks. For all our experiments, $n_{act} = 16$ agents are used. The remaining hyperparameters are identical to those used in [8].

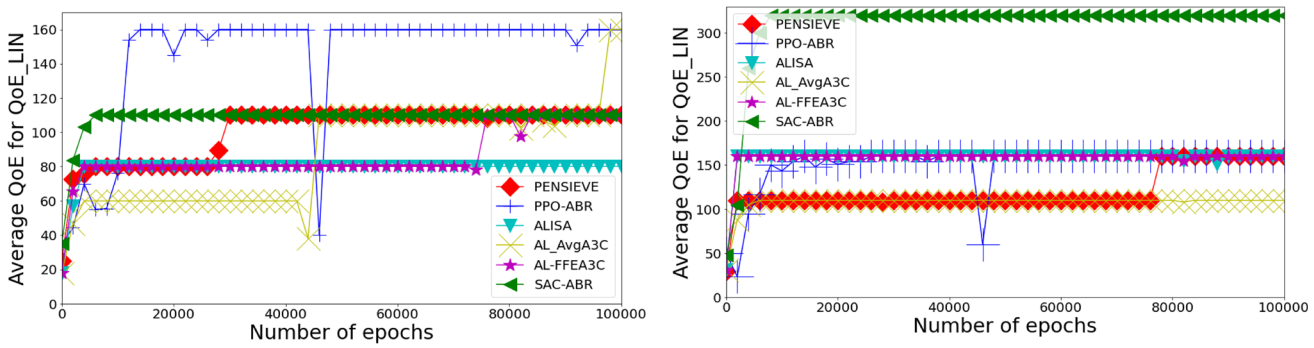
4.3 Performance metrics

We evaluate the efficacy of all ABR algorithms using QoE as a performance metric. The QoE is formulated as [4, 7, 8, 20, 21], and [22]:

$$QoE = \sum_{n=1}^{N} q(b_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |q(b_{n+1}) - q(b_n)| \quad (26)$$

where, the three components of QoE are (i) the total bit rates for all chunks, (ii) the penalty for re-buffering, and (iii) the penalty to prevent stuttering and improving smoothness. We consider three different variants of QoE:

- QoE_{LIN}: This is a generic QoE metric with the rebuffer penalty as $\mu = 160$ and $q(b_n) = b_n$.
- QoE_{HD}: This metric is designed specifically to evaluate QoE for HD videos with the rebuffer penalty as $\mu = 192$ and $q(b_n) = HD_{r_t}(b_n)$, where HD_{r_t} is assigned with the reward values of $\{1, 2, 3, 12, 15, 20\}$ when b_n is $\{20, 40, 60, 80, 110, 160\}$ Mbps, respectively.
- QoE_{VR}: This metric is designed specifically to evaluate QoE for virtual reality (VR) videos with the rebuffer penalty as $\mu = 400$ and $q(b_n) = VR_{r_t}(b_n)$, where VR_{r_t}



(a) 5G-Drive traces

(b) 5G-Walk traces

Fig. 4 Average value of QoE_{LIN} for 5G-Drive and 5G-Walk traces attained by various ABR algorithms

is assigned with reward values of {5, 10, 15, 20, 25, 50} when b_n is {20, 40, 60, 80, 110, 160} Mbps, respectively.

4.4 Results

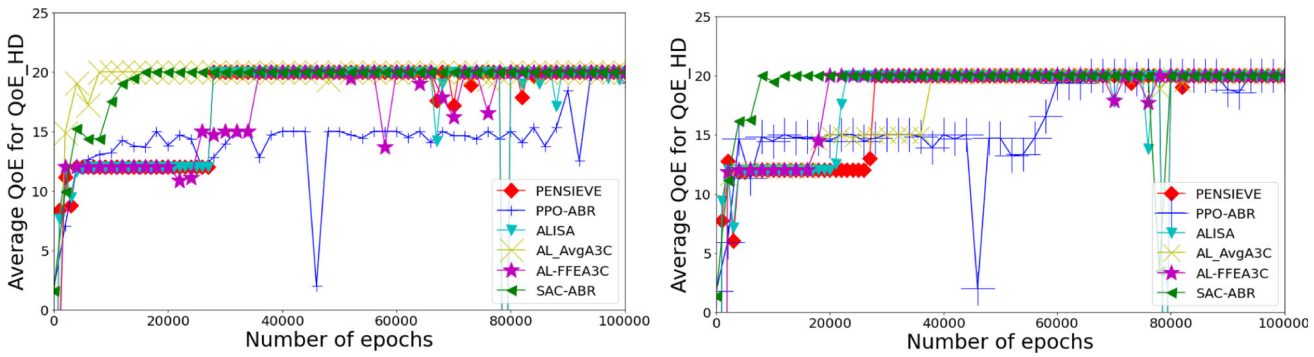
This section highlights the results of the proposed PPO-ABR and SAC-ABR algorithms and compares their performance with other state-of-the-art ABR methods. We present training results in Figs. 4a, 4b, 5a, 5b, 6a, and 6b and testing results in Figs. 7, 8, 9, 10a, 10b, 11a, 11b, and 11c.

4.4.1 Results with different QoE metrics

Table 1 presents the average QoE achieved by various ABR algorithms for different variants of QoE metrics (QoE_{LIN}, QoE_{HD}, and QoE_{VR}) across both 5G-Drive and 5G-Walk traces. The bold values in the table represent the maximum average QoE achieved by the ABR algorithm for the specific QoE metric (column-wise). Our findings indicate that PPO-ABR and SAC-ABR consistently outperform other state-of-the-art ABR algorithms in terms of QoE across most scenarios. To elaborate, PPO-ABR exhibits a noteworthy superiority, achieving a 47.82% and 27.12% higher average QoE (QoE_{LIN}) compared to Pensieve for 5G-Drive and 5G-Walk traces. Similarly, SAC-ABR demonstrates its effectiveness with a 7.70% and 155.69% higher average QoE (QoE_{LIN}) than Pensieve for 5G-Drive and 5G-Walk traces, respectively. Both PPO-ABR and SAC-ABR demonstrate better performance for other QoE metrics as well, i.e., QoE_{HD} and QoE_{VR}.

Similar advantages are observed when comparing PPO-ABR and SAC-ABR to AL-AvgA3C, AL-FFE3C, and ALISA. We also observe that SAC-ABR consistently performs better than PPO-ABR for most of the cases, which is due to the inherent advantage of using an off-policy method over an on-policy method such that both target and behavior policies are utilized for the action prediction.

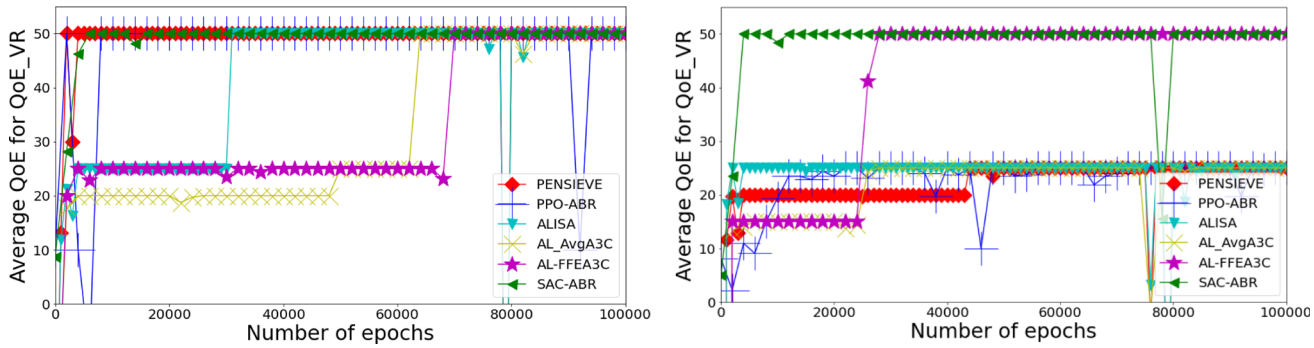
Figure 4a presents the average reward values attained by various ABR algorithms during each training epoch using the QoE_{LIN} metric on 5G-Drive traces. As the number of training epochs increases, our findings demonstrate that each algorithm exhibits distinct behaviours. Notably, the PPO-ABR achieves a high reward value early on in its training, demonstrating extraordinary performance from the start. PPO-ABR consistently outperforms other DRL-based ABR



(a) 5G-Drive traces

(b) 5G-Walk traces

Fig. 5 Average value of QoE_{HD} for 5G-Drive and 5G-Walk traces attained by various ABR algorithms



(a) 5G-Drive traces

(b) 5G-Walk traces

Fig. 6 Average value of QoE_{VR} for 5G-Drive and 5G-Walk traces attained by various ABR algorithms

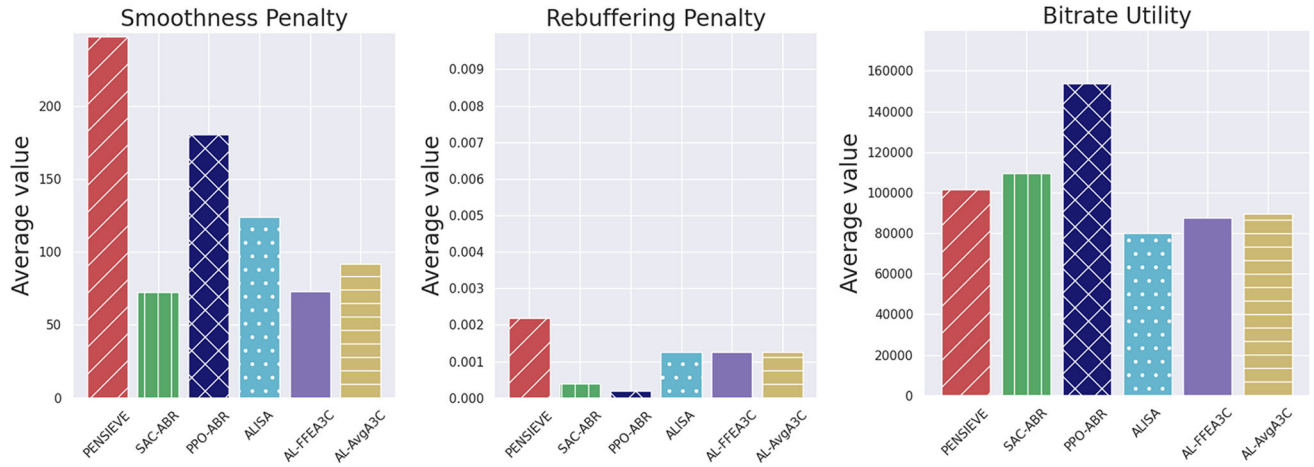


Fig. 7 Comparing PPO-ABR and SAC-ABR with existing DRL-based ABR algorithms by analyzing their performance on the individual components on the 5G traces using a QoE_{LIN} metric

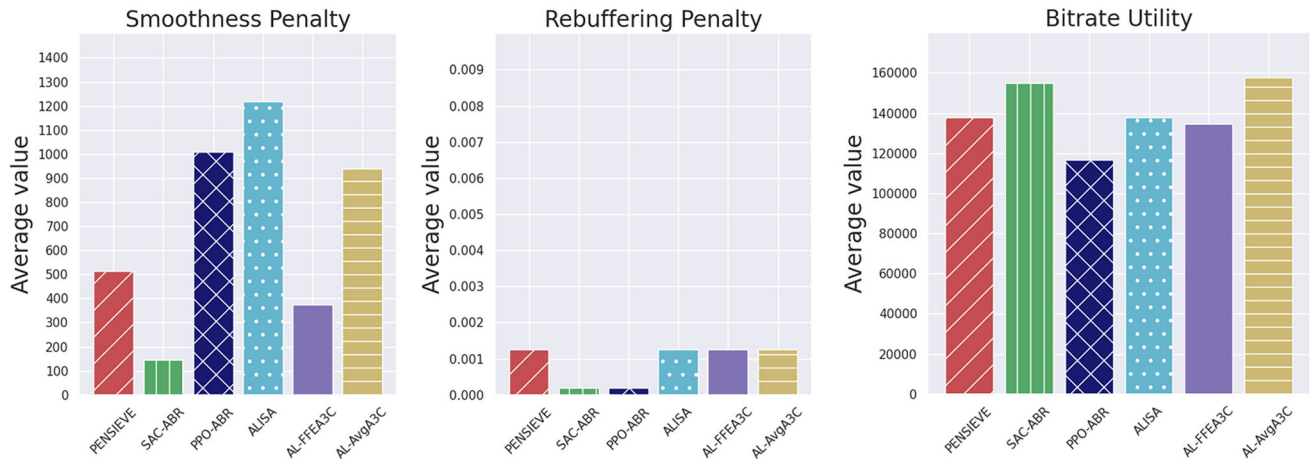


Fig. 8 Comparing PPO-ABR and SAC-ABR with existing DRL-based ABR algorithms by analyzing their performance on the individual components on the 5G traces using a QoE_{HD} metric

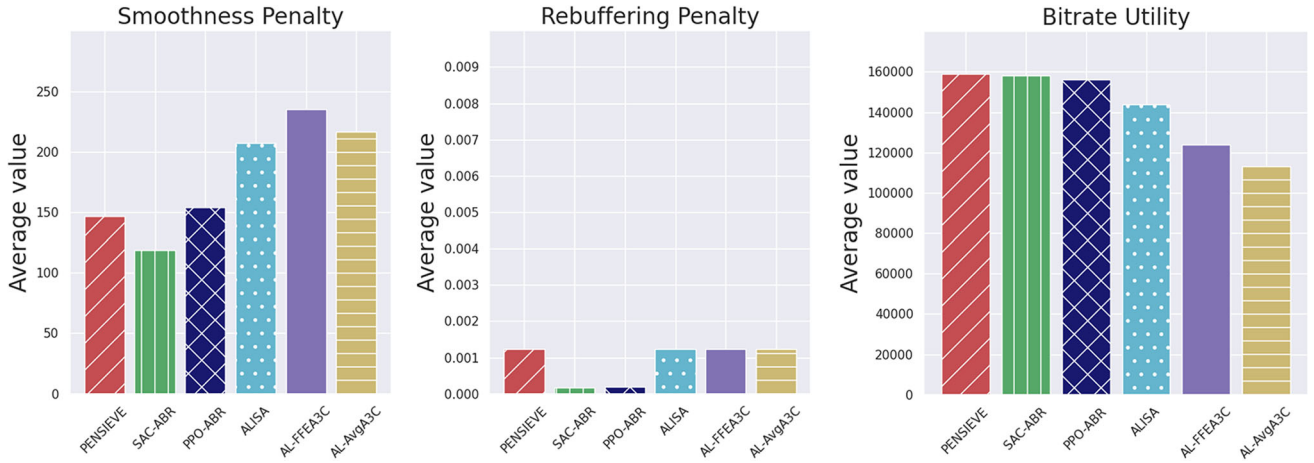
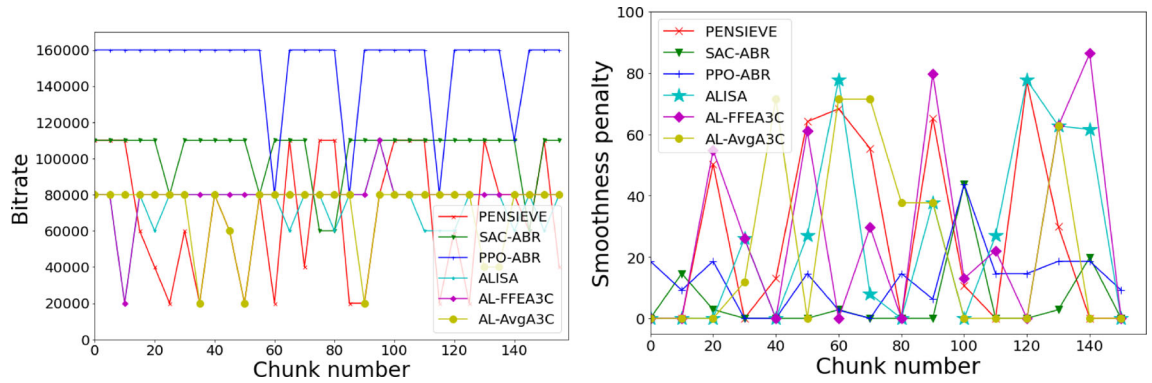
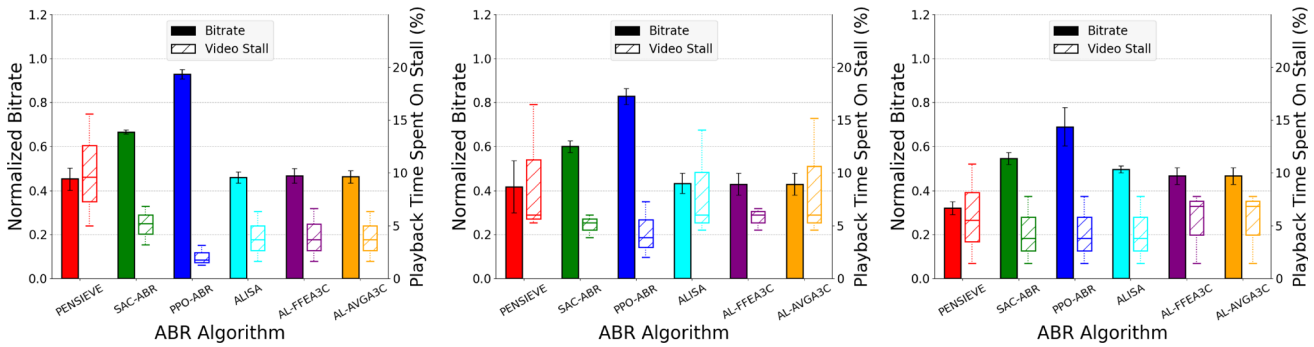


Fig. 9 Comparing PPO-ABR and SAC-ABR with existing DRL-based ABR algorithms by analyzing their performance on the individual components on the 5G traces using a QoE_{VR} metric



(a) Bitrate (b) Smoothness penalty

Fig. 10 Comparing PPO-ABR and SAC-ABR with existing DRL-based ABR algorithms by analyzing their performance based on chunk count (158), chunk duration (1 sec), chunk-wise bitrate (20 Mbps to 160 Mbps), and chunk-wise smoothness penalties (0 to 100)



(a) Chunk Duration: 1 sec (b) Chunk Duration: 2 sec (c) Chunk Duration: 4 sec

Fig. 11 Comparing PPO-ABR and SAC-ABR with existing DRL-based ABR algorithms by analyzing their performance on the bitrate and video stall for video's chunk length with 1, 2 and 4 sec

Table 1 Comparison of QoE_{lin}, QoE_{HD}, and QoE_{VR} metrics for different DRL based ABR algorithms over 5G-Drive and 5G-Walk traces

DRL-based ABR methods	5G-DRIVE Traces			5G-WALK Traces		
	QoE_{lin}	QoE_{HD}	QoE_{VR}	QoE_{lin}	QoE_{HD}	QoE_{VR}
PENSIEVE	99.27	17.15	48.06	118.87	16.87	20.87
PPO-ABR	**146.75**	14.15	46.24	151.11	15.77	21.82
SAC-ABR	106.92	18.82	**48.23**	**303.95**	**18.85**	**47.52**
AL-AvgA3C	88.17	**19.05**	30.49	106.44	16.63	20.55
ALISA	78.13	17.12	40.89	155.62	17.32	23.05
AL-FFEA3C	85.82	16.77	31.15	155.19	17.53	38.89

methods as training progresses, ultimately achieving the highest reward value. In addition, the SAC-ABR exhibits its strengths in a distinct context. In the case of 5G-Walk traces, as presented in Fig. 4b, the SAC-ABR achieves the highest average reward value in comparison to other DRL-based ABR algorithms. These findings emphasize the adaptability and efficacy of each ABR algorithm in various scenarios, highlighting the importance of selecting the most appropriate approach based on the unique characteristics of the network traces and the desired QoE outcomes. Figures 5a, and 5b presents the average reward value achieved by various ABR algorithms at each training epoch using QoE_{HD} metric over 5G-Drive and 5G-Walk traces, respectively. For these traces, the SAC-ABR outperformed compared to other competing algorithms. We also present that SAC-ABR achieves better performance compared to other ABR algorithms with QoE_{VR} metric for both 5G-Drive and 5G-Walk traces, as shown in Fig. 6a and 6b, respectively.

4.4.2 In depth analysis of different components of QoE metrics

To thoroughly comprehend and demonstrate the benefits of the proposed approaches, we execute an in-depth analysis of the QoE metrics' individual components. This comparison of ABR algorithms is based on 5G trace data, and their efficacy is evaluated using three crucial metrics: average bitrate utility, average rebuffering penalty, and average smoothness penalty. The outcomes of our analysis are illustrated in Figs. 7, 8, and 9. Compared to other ABR algorithms, SAC-ABR and PPO-ABR attain higher bitrates, reduce rebuffering occurrences, and maintain smoother video playback. These figures provide valuable insight into the overall superiority of these algorithms when operating in 5G network conditions, ultimately resulting in an improved QoE for consumers during the delivery of ultra-HD video.

4.4.3 QoE analysis per Chunk

In order to comprehend the characteristics of transitions in video as it shifts between different bitrates, we have provided a study of the QoE metric in terms of two components: bitrate and smoothness penalty. It is evident that the majority of ABR algorithms prioritise stability, aiming to minimise the amount of bitrate transitions, as shown in Fig. 10a. Both of our proposed algorithms, PPO-ABR and SAC-ABR, have switched bitrates only 4-5 times during transmitting 158 chunks. Consequently, they achieve a lower penalty in terms of smoothness compared to other ABR methods, as shown in 10b.

4.4.4 Impact of chunk duration on QoE

To demonstrate the scalability of the proposed algorithm across varying chunk duration, we followed the methodology proposed in [23], [24], and [25] for adjusting the quality of the video based on factors such as the chunk duration, chunk counts, number of video quality switches, and video stalls (rebuffering time). For our experiments, we considered three different 4K videos [9] with chunk durations of 1 second, 2 seconds, and 4 seconds. The first video has 158 chunks with each chunk of 1 second duration, the second video has 80 chunks with each chunk of 2 seconds duration, and the third video has 40 chunks with each chunk of 4 seconds duration.

Figures 11a, 11b, and 11c present the performance of ABR algorithms for chunk duration 1, 2, and 4 seconds, respectively. We observe that PPO-ABR and SAC-ABR provide better performance than other ABR algorithms irrespective of the chunk duration. Nevertheless, the normalized bitrate declines as the chunk duration increases in order to minimise the likelihood of transitioning to various bit rates, hence minimising video rebuffering and enhancing playback smoothness.

5 Conclusions and future work

In this paper, we have effectively demonstrated how employing on-policy and off-policy based multi-agent DRL methods can significantly enhance the performance of ABR algorithms. Our proposed methods, namely PPO-ABR and SAC-ABR, effectively resolve the limitations of limited sample

size, exploration difficulties, policy inefficiency, and susceptibility to random seed and hyperparameter variations. Extensive experiments have demonstrated that PPO-ABR and SAC-ABR consistently outperform the existing Pensieve by as much as 47.82% and 155.69% percent, respectively. These evaluations are conducted under 5G throughput traces, indicating that our approaches are applicable and effective in modern network conditions. As part of our future efforts, we plan to examine the efficacy of PPO-ABR and SAC-ABR in edge-driven video delivery services to broaden the scope of our research. This study will investigate the potential benefits and optimisations that these methods can provide in the context of edge computing and its effect on the quality of experience for video streaming.

Author Contributions Mandan Naresh: Conceptualization, Methodology, Validation, Writing - original draft. Paresh Saxena: Conceptualization, Writing - original draft, Writing - review and editing, Supervision. Manik Gupta: Conceptualization, Writing - original draft, Writing - review and editing, Supervision.

Funding This work has been supported by TCS Foundation, India under the TCS research scholar program, 2019-2023.

Data Availability The data supporting the findings of this study are available upon reasonable request.

Declarations

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

References

1. Ericsson Mobility Report (2020). https://www.ericsson.com/49da93/assets/local/mobility-report/documents/2020/june2020-ericsson-mobility-report.pdf
2. Dash.js. https://github.com/Dash-Industry-Forum/dash.js/
3. Arunruangsirilert K, Bo W, Hang S, Katto J (2022) Performance evaluation of low-latency live streaming of mpeg-dash uhd video over commercial 5g nsa/sa network. In: 2022 International conference on computer communications and networks (ICCCN), IEEE, pp 1–6
4. Mao H, Netravali R, Alizadeh M (2017) Neural adaptive video streaming with pensieve. In: Proceedings of the conference of the acm special interest group on data communication. SIGCOMM '17, pp 197–210. Assoc Comput Mach, New York, NY, USA. https://doi.org/10.1145/3098822.3098843
5. Sutton RS, Barto AG (2018) Reinforcement Learning: An Introduction. A Bradford Book, Cambridge, MA, USA
6. Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. CoRR **abs/1602.01783** arXiv:1602.01783
7. Naresh M, Gireesh N, Saxena P, Gupta M (2022) Sac-abr: Soft actor-critic based deep reinforcement learning for adaptive bitrate streaming. In: 2022 14th International conference on communication systems & networks (COMSNETS), IEEE, pp 353–361
8. Naresh M, Saxena P, Gupta M (2023) Ppo-abr: Proximal policy optimization based deep reinforcement learning for adaptive bitrate streaming. In: 2023 International wireless communications and mobile computing (IWCMC), pp 199–204. https://doi.org/10.1109/IWCMC58020.2023.10182379
9. Narayanan A, Zhang X, Zhu R, Hassan A, Jin S, Zhu X, Zhang X, Rybkin D, Yang Z, Mao ZM, Qian F, Zhang Z-L (2021) A variegated look at 5g in the wild: Performance, power, and qoe implications. In: Proceedings of the 2021 ACM SIGCOMM 2021 conference. SIGCOMM '21, pp 610–625, New York, NY, USA. https://doi.org/10.1145/3452296.3472923
10. Narayanan A, Ramadan E, Mehta R, Hu X, Liu Q, Fezeu RAK, Dayalan UK, Verma S, Ji P, Li T, Qian F, Zhang Z-L (2020) Lumos5g: Mapping and predicting commercial mmwave 5g throughput. In: Proceedings of the ACM internet measurement conference. IMC '20, pp 176–193. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3419394.3423629
11. Grondman I, Busoniu L, Lopes GAD, Babuska R (2012) A survey of actor-critic reinforcement learning: Standard and natural policy gradients. IEEE Trans Syst Man Cybern, Part C (Applications and Reviews) 42(6):1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595
12. Konda VR, Tsitsiklis JN (2000) Actor-critic algorithms. In: Advances in neural information processing systems, pp 1008–1014
13. Schulman J, Levine S, Moritz P, Jordan MI, Abbeel P (2015) Trust region policy optimization. CoRR **abs/1502.05477** arXiv:1502.05477
14. Filippi S, Cappé O, Garivier A (2010) Optimism in reinforcement learning and kullback-leibler divergence. 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp 115–122
15. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov (2017) Proximal policy optimization algorithms. CoRR **abs/1707.06347** arXiv:1707.06347
16. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D ICLR (2016) Continuous control with deep reinforcement learning. In: Bengio Y, LeCun Y (eds.). http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15
17. Haarnoja T, Zhou A, Hartikainen K, Tucker G, Ha S, Tan J, Kumar V, Zhu H, Gupta A, Abbeel P, Levine S (2018) Soft actor-critic algorithms and applications. CoRR **abs/1812.05905** https://arxiv.org/abs/1812.059051812.05905
18. Van Hasselt H (2010) Double q-learning, pp 2613–2621
19. Jankowiak M, Obermeyer F (2018) Pathwise derivatives beyond the reparameterization trick. In: Dy J, Krause A (eds.) Proceedings of the 35th international conference on machine learning. proceedings of machine learning research, vol 80, pp 2235–2244. https://proceedings.mlr.press/v80/jankowiak18a.html
20. Naresh M, Das V, Saxena P, Gupta M (2022) Deep reinforcement learning based qoe-aware actor-learner architectures for video streaming in iot environments. Computing 104. https://doi.org/10.1007/s00607-021-01046-1
21. Naresh M, Saxena P, Gupta M (2023) Deep reinforcement learning with importance weighted a3c for qoe enhancement in video delivery services. In: 2023 IEEE 24th International symposium on a world of wireless, mobile and multimedia networks (WoWMoM), pp 97–106. https://doi.org/10.1109/WoWMoM57956.2023.00024
22. Yin X, Jindal A, Sekar V, Sinopoli B (2015) A control-theoretic approach for dynamic adaptive video streaming over http. SIGCOMM Comput Commun Rev 45(4):325–338. https://doi.org/10.1145/2829988.2787486
23. Taha M, Ali A, Lloret J, Gondim PRL, Canovas A (2021) An automated model for the assessment of qoe of adaptive video streaming

over wireless networks. Multimedia Tools Appl 80(17):26833–26854. https://doi.org/10.1007/s11042-021-10934-9
24. Taha M, Lloret J, Ali A, García L (2018) Adaptive video streaming testbed design for performance study and assessment of qoe. Int J Commun Syst 31:3551. https://doi.org/10.1002/dac.3551
25. Taha M, García L, Jiménez JM, Lloret J (2017) Sdn-based throughput allocation in wireless networks for heterogeneous adaptive video streaming applications. 2017 13th International wireless communications and mobile computing conference (IWCMC), pp 963–968

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.