GRadient sharing A3C for Adaptive Bitrate (GRAAB): A DRL Based Approach for Adaptive Video Streaming

Mandan Naresh, Paresh Saxena and Manik Gupta
Dept. of CSIS, BITS Pilani, Hyderabad, India
{p20180420, psaxena, manik}@hyderabad.bits-pilani.ac.in

***Abstract*—Recent research has shown that the deep reinforcement learning (DRL) based asynchronous advantage actor-critic (A3C), an on-policy method, is an effective tool for generating adaptive bit rates (ABR) for video streaming and provides a high quality of experience (QoE) for end users under varying network conditions. However, vanilla A3C suffers from the following issues: (i) it has biased estimates due to the use of high entropy weights, (ii) it has a high number of computations due to the use of near-homogeneous policies while training with a single set of global parameters with multiple agents, and (iii) it lacks exploration due to the use of the on-policy method. To address the aforementioned issues, this paper presents gradient sharing A3C for adaptive bitrate (GRAAB) to generate adaptive bitrates. By sharing gradients and parameters among multiple workers, GRAAB encourages exploration and produces optimal policies without compromising long-term convergence. Using 5G traces from the Lumos 5G dataset, we highlight the advantages of GRAAB and demonstrate that it achieves a significantly higher average QoE than a number of other state-of-the-art ABR algorithms.**

***Index Terms*—DRL, ABR, QoE, and 5G.**

I. Introduction

According to the Cisco annual Internet report (2018-2023) [1], over two-thirds of the world's population will have Internet connectivity by 2023. Specifically, Internet video traffic will contribute more than any other type of Internet traffic. However, unstable network conditions may impact the quality of experience (QoE) for end users. To improve the QoE, adaptive bitrate (ABR) streaming enables rate adaptation during a video session where a server encodes source video chunks with different bitrates and a client can dynamically select the bitrate according to the current network conditions.

By adapting video bitrate based on the underlying network conditions, ABR algorithms strive to improve user quality of experience (QoE). Designing an ABR algorithm is problematic in dynamic network conditions due to sudden fluctuations in network throughput. Tradition ABR algorithms are mainly buffer-based, i.e., depending on the buffer occupancy at the client, or rate-based, i.e., depending on the estimated throughput of the network. Some examples of such ABR algorithms include FESTIVE [2], RB [3], BB [4], BOLA [5], and MPC [6]. However, they do not generalize well to a wide range of network conditions since they are based on fixed rules. Recently, deep reinforcement learning (DRL) based asynchronous advantage actor-critic (A3C) methods Pensieve [7], NANCY [8] have demonstrated several advantages over fixed rule-based ABR algorithms. However, A3C based ABR algorithms suffer from the following issues: (i) biased estimates due to the use of high entropy weights, (ii) high number of computations due to the use of near-homogeneous policies while training with a single set of global parameters with multiple agents, and (iii) lacks exploration due to the use of the on-policy method.

To address the aforementioned issues, this paper presents **gr**adient sharing **A**3C for **a**daptive **b**itrate (**GRAAB**) to generate adaptive bitrates. By sharing gradients and parameters among multiple workers, GRAAB encourages exploration and produces optimal policies without compromising long-term convergence. The main contribution of this paper is the design and implementation of GRAAB, and its evaluation with Lumos 5G network data set [9]. We have implemented GRAAB using TensorFlow [10] and carried out comparisons under a wide range of network conditions and QoE metrics. Our results demonstrate the ability of GRAAB to learn network conditions, video properties and provide a much higher QoE than a number of state-of-the-art ABR algorithms.

The remainder of the paper is organized as follows. Section II presents the relevant background on reinforcement learning and actor-critic and gradient-sharing methods. Further, Section III presents the proposed algorithm and system design. We present the experimental setup and results in Section IV. Finally, we conclude our work in Section V.

II. Background

This section presents a brief overview of reinforcement learning and the A3C methods.

A. Reinforcement Learning

Reinforcement learning [11] is the process of dynamic learning in which an agent has limited or no prior knowledge about the environment. For each time step $t = 0, 1, 2, 3, ...$, the agent is in a state s_t, performs action a_t, moves to state s_{t+1} and observes a reward r_t. The agent selects actions based on a policy $\pi(s_t, a_t)$, where $\pi(s_t, a_t)$ is the probability that when the agent is in a state s_t, and it takes action a_t. The agent's objective is to formulate a strategy that maximizes the expected return given by $V^*(s) = \max_{\{\pi \in \Pi\}} V^\pi(s)$ where Π is the set of possible policies and the state-value function $V^\pi(s) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, \pi\right]$ where $\gamma \leq 1$ is a discount factor.

B. DRL empowered actor-critic methods

Actor-critic [12] methods comprise an actor neural network $\pi(a_t|s_t;\theta)$ representing the policy and a critic $v(s_t;\theta_v)$ neural network, an estimate of the value function based on the policy generated by the actor. The gradient of the cumulative discounted rewards with respect to the policy parameters, θ, is given by,

$$\nabla_\theta E_{\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r_t\right] = E_{\pi_\theta}\left[\sum_{t=0}^{\infty}\nabla_\theta \log \pi_\theta(s_t,a_t)\,\psi\, A^{\pi_\theta}(s_t,a_t)\right] \tag{1}$$

where ψ is the weight parameter for the advantage function $A^{\pi_\theta}(s_t,a_t)$ and the advantage function is given by,

$$A^{\pi_\theta}(s_t,a_t) = r_t + \gamma V^{\pi_\theta}(s_{t+1};\theta_v) - V^{\pi_\theta}(s_t;\theta_v) \tag{2}$$

The policy parameter (θ) is updated as follows:

$$\Delta\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi_\theta(s_t,a_t)\,\psi\, A^{\pi_\theta}(s_t,a_t) + \beta\nabla_\theta H(.|s_t) \tag{3}$$

where $H(.)$ is the entropy of the policy to promote random actions, and α is the learning rate. The parameter β is the entropy regularization that is used to control the random actions. The update of the critic parameter (θ_v) is given by,

$$\Delta\theta_v \leftarrow \theta_v - \alpha_v \sum_t \nabla_{\theta_v}\,\psi\, A(s_t,a_t)^2 \tag{4}$$

where α_v is the critic learning rate. To further speed up the training, A3C uses multiple actors and critic networks [13]. The multiple learning agents are created, each with their copy of the environment with local actors and critics. These learning agents execute in parallel and update the parameters of a central agent asynchronously using stochastic gradient descent. At the end of an episode, the learning agent updates the central agent and then updates itself with the current state of the global agent. The parallel learning can be achieved by executing each learning agent in a separate thread.

C. Gradient sharing for A3C agents

Recently, gradient sharing is utilized by multiple A3C agents and has shown to achieve much better performance than vanilla A3C [14]. Some of the key components to enable gradient sharing in A3C are as follows:

- Gradient clipping: it enables exploration using gradient sharing operation with small learning rates on parameter difference (D) between global parameters. It is given by:

$$D = |\theta_i - \theta_j| \text{ subject to } D < |m_i - m_j| \tag{5}$$

  where θ_i, and θ_j are the global parameters, and m_i, and m_j are the index of the assigned global parameters to the workers. To support policy diversification without compromising long-term convergence, gradient sharing uses M sets of global parameters and asynchronously shares gradients among N workers. Based on Equation (5), the gradient difference among the global networks is given by,

$$G = |g_j - g_i| < |m_i - m_j|, \text{where } |g_i| < 1.0,\ |g_j| < 1.0 \tag{6}$$

  where g_i, g_j are gradients in global networks.
- Reduced entropy loss: To reduce the large entropy loss in A3C, [14] uses advantage weights with zero mean noise, which is used to reduce the bias with a limited entropy range. To reduce entropy loss, $H(.)$, in Equation (4), the zero entropy noise (ϵ) is included in the advantage function as follows:

$$A^{\pi_\theta}_{\epsilon}(s_t,a_t) = E\left[\sum_{t=0}^{T}\psi A(s_t,a_t) + \epsilon\right] \tag{7}$$

  where the policy parameter (θ) and the critic parameter (θ_v) are updated as follows:

$$\Delta\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi_\theta(s_t,a_t) A^{\pi_\theta}_{\epsilon}(s_t,a_t) + \beta\nabla_\theta H(.|s_t) \tag{8}$$

$$\Delta\theta_v \leftarrow \theta_v - \alpha_v \sum_t \nabla_{\theta_v} A_\epsilon(s_t,a_t)^2 \tag{9}$$

III. GRAAB: Design and Implementation

In this section, we present the design and implementation of the proposed GRAAB algorithm to generate ABR for video.

A. Methodology

Figure 1 presents the client-server system architecture with GRAAB utilziing parallel training among M global actor and critic parameters, as well as N local actor and critic parameters. The client requests the chunks from the server in order to download and play them. It also estimate a bit rate by utilizing the ABR module and send it to the server. Several vanilla-A3C based ABR methods are proposed however in case of vanilla A3C methods, each global network has an update method that generates near-homogeneous policies for multiple workers. Due to the generation of near-homogeneous policies, high entropy weights are required to encourage exploration, and it results in less diversity among the policies.

To overcome this issue, GRAAB uses multiple agents for training to calculate the bit rate. It relies on parallel training among M global actor and critic parameters, as well as N local actor and critic parameters. Each worker has information at the chunk level, including chunk size, chunk throughput, chunk download time, and chunk bitrate. GRAAB uses four key steps to increase the diversity among the policies. The first step is where each worker receives the chunk-level information from the current state s_t. Second, each worker computes the gradient, which is updated locally. Third, each worker updates the gradients to the global networks by locking other workers. Finally, each worker resumes training and exploration after all locks have been removed. The inputs and outputs to the GRAAB (ABR) are described below.

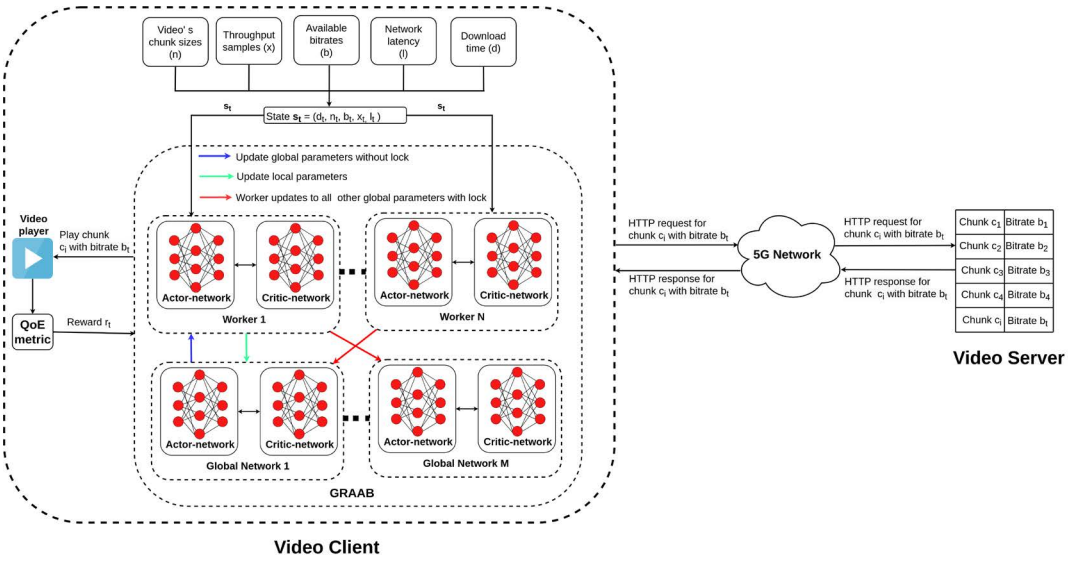


Fig. 1: Client-Server system architecture with GRAAB for ABR utilizing parallel training among M global actor and critic parameters, as well as N local actor and critic parameters.

1) Inputs: After downloading each chunk t at the client side, the GRAAB agent takes the following inputs $s_t = (x_t, d_t, n_t, b_t, l_t)$ into the neural network where x_t are the throughput measurements for each of the previous k chunks, d_t are the download times for the past k chunks, n_t is the available sizes for the next video chunk, b_t is the size of the current buffer, and l_t is the previous chunk's bitrate.

2) Outputs: Given state $s_t = (x_t, d_t, n_t, b_t, l_t)$, the agent selects a suitable bitrate for the next video chunk, i.e., a corresponding action a_t. The policy $\pi_\theta(s_t, a_t)$ is used to estimate the action a_t.

B. GRAAB Training Algorithm

Algorithm 1 presents the GRAAB training and outlines the critical steps. The input to the algorithm is $s_t = (x_t, d_t, n_t, b_t, l_t)$ and it is initialized with M global parameter copies and N local learning workers. The summary of the algorithm is presented as follows:

1) Each worker $i \in N$ is assigned to a global parameter copy $m \in M$ at Line 6.
2) Locks are assigned to each global parameter $m \in M$ at Line 9.
3) Each chunk is played at a specified bitrate using the selection of the action based on the current state and the policy and to store the corresponding reward at Line 21.
4) Map the state (s_t) inputs to the closest bitrate by actor-network at Line 24.
5) Evaluate the video's bitrate by critic network using v_t at Line 25.
6) Worker i waits till all the locks L^j, $(j \in M)$ are deactivated. Once deactivated, it activates all the locks at Line 31.
7) Gradient sharing (termed as F) operation is done where the worker i updates all the other global copies ($j \in M$, $j \neq i$) using the calculated gradients at Line 32.
8) Worker i deactivates all the locks L^j, ($j \in M$) and resumes at Line 33.

The output to the algorithm is the actor-network makes the decision to play the chunk by chunk with a specified bitrate at Line 38. The critic network evaluates the video's bitrate,v_t at line 39. Finally, the actor and critic parameters are updated based at Line 40.

IV. Experimental Setup and Results

This section presents the implementation details, data sets utilized in the experiments, training methodology, performance metrics, and the outcome of the experiments.

A. Implementation Details

The experiments conducted follow the setup with $M = 4$ global actor and critic parameters, as well as $N = 8$ local actor and critic parameters. In the actor-network, the final layer uses the softmax activation function to map the input chunk size to the closest bitrate. The critic network, on the other hand, uses a linear neuron without an activation function. The purpose of this design is to allow the actor-network to make decisions based on the output of the critic network, which evaluates the quality of the decision made by the actor-network. GRAAB uses $k = 8$ past bandwidth measurements, which serve as an input to a 1D convolution neural network (CNN) layer with 128 filters of size 4 and stride 1 to generate the ABR. Similarly, past k chunk download times and subsequent chunk sizes are supplied to distinct 1D-CNNs of the same shape. These outputs are then flattened and concatenated with the remaining inputs. The results are subsequently transferred through successive layers containing 256 neurons.

Algorithm 1 Training algorithm for GRAAB utilizing the gradient sharing among multiple agents for adaptive video streaming

Input: video samples, hyperparameters;
Parameters:
Video vi; choose a video file as a input
Chunk c; select the bitrate for future chunks from video file
Initialize:
  $E^{N\leftarrow N}$ environment copies, $t \leftarrow 0$
  $A^{N\leftarrow N}$ worker threads
  $\theta_m^g \leftarrow M$ global actor and critic parameters
  $L^{M\leftarrow N}$ global locks for θ_m^g
$L^{M\leftarrow \textbf{unlock}}$, initially all locks are deactivated
$\theta_m^i \leftarrow \theta_m^g$ for $i \in N$, for $m \in M$, copy M global parameters to their respective assigned N local worker parameters
for video vi= 1,2,3...., VI **do**
  Observe initial state s_t;
  for chunk c=1,2,3...., C **do**
    while not optimal policy reached **do**
      for all worker threads A^i**,** $i \in N$
      $t \leftarrow t+1$
      $m \leftarrow$ index of the assigned global parameters for A^i
      State s_t provided by environment E^i to A^i
      Action $a_t \sim \pi(a_t, s_t)$ performed by A^i
      Environment E^i provides reward r_t and next state s_{t+1}
      $v_t \leftarrow r_t + \gamma V^{\pi_\theta}(s_{t+1})$, compute bootstrap
      Compute advantage $A(s_t, a_t)$ using Equation 7
      Map the state (s_t) inputs to the closest bitrate by actor network
      Evaluate the video's bitrate by critic network using v_t
      Update the actor network by policy parameter (θ) using Equation 8
      Update the critic network by critic parameter (θ_v) using Equation 9
      Update the assigned global parameters θ_m^g using the gradients.
      Update the local parameters of the thread $\theta_m^g \rightarrow \theta_m$
      Wait till all locks L^j with $j \in \{M-m\}$ are unlocked
      $L^j \leftarrow$ **lock** for all locks $j \in M$
      Gradient sharing operation F: Using accumulated gradients update other global parameter copies θ_j^g with $j \in \{M-m\}$
      $L^M \leftarrow$ **unlock**, deactivate all locks
    end while
  end for
end for
Output:
Actor network makes the decision to play the chunk by chunk with a specified bitrate
Critic network is evaluates the video's bitrate using v_t
Update actor and critic parameters θ, ϕ

We train GRAAB and other state-of-the-art DRL-based algorithms for 100,000 iterations and selected the model with the maximum average reward. We have considered a discount factor of $\gamma = 0.99$ and learning rates for actors and critics as 0.0001 and 0.001, respectively. GRAAB uses entropy regularization factor (η) range from 5 to 0.005. In the early training iterations, GRAAB employs high entropy to promote extensive exploration. As training progresses, the GRAAB gradually reduces the high-entropy weights, ultimately favoring more stable configurations for the remainder of the iterations.

B. Datasets

We have used 5G network data sets [9] for the experiments. 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, 92 traces which consider driving-related mobility patterns and (ii) 5G-Walk, 81 traces which consider walking-related mobility patterns.

C. ABR algorithms, Performance Metrics, and Training Methodology

We compare the performance of GRAAB with the following ABR algorithms: (i) Pensieve [7]: DRL-based vanilla-A3C method to generate ABR, (ii) PPO-ABR [15]: DRL-based proximal policy optimized method to generate ABR, (iii) ALISA [16]: DRL-based importance-weighted actor-learner method to generate ABR, (iv) AL-AvgA3C [17]: DRL-based averaged-A3C method to generate ABR, (v) AL-FFEA3C [17]: DRL-based follow-then-forage method to generate ABR and (vi) SAC-ABR [18]: DRL-based soft-actor-critic based method to generate ABR.

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

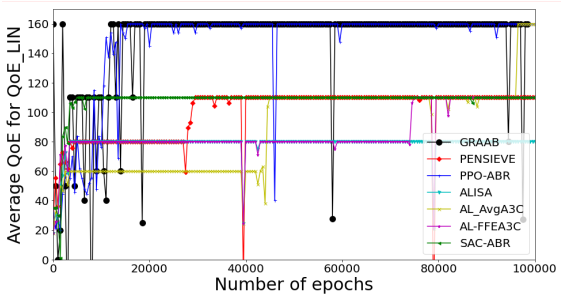
DRL-based ABR methods	5G-DRIVE Traces			5G-WALK Traces		
	QoE_{lin}	QoE_{HD}	QoE_{VR}	QoE_{lin}	QoE_{HD}	QoE_{VR}
GRAAB	**146.75**	**19.05**	46.47	**310.52**	18.08	**48.06**
PENSIEVE	99.27	17.15	48.06	118.87	16.87	20.87
PPOABR	142.87	14.15	46.24	151.11	15.77	21.82
SACABR	106.92	18.82	**48.23**	303.95	**18.85**	47.52
AL-AvgA3C	88.17	18.18	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

We compare all the ABR algorithms using QoE as a performance metric. The QoE is given as,

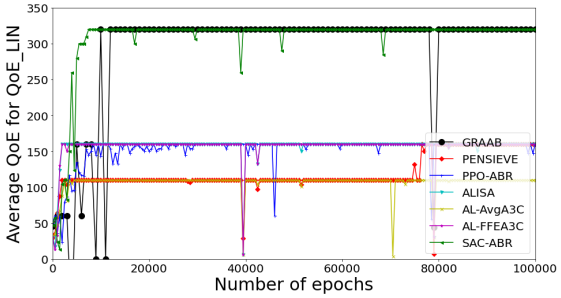
$$QoE = \sum_{n=1}^{N} q(b_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |q(b_{n+1}) - q(b_n)| \quad (10)$$

where the QoE term contains the three components: (i) the sum of all chunk's bit rates, (ii) the penalty due to re-buffering, and (iii) the smoothness of the video, measured using the difference between the bit rates used to encode consecutive chunks.

Specifically, we consider three different variations of the above QoE metric:

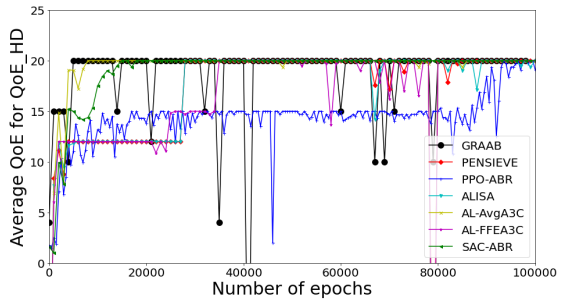


(a) 5G-Drive traces

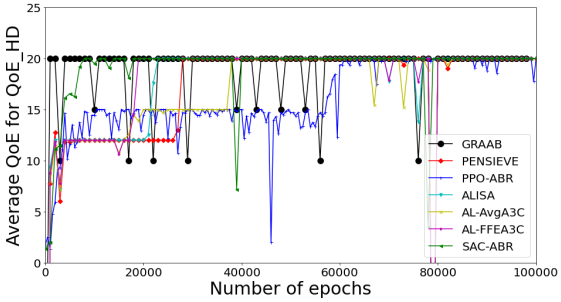


(b) 5G-Walk traces

Fig. 2: Average value of QoE_{LIN} for 5G-Drive and 5G-Walk traces attained by various ABR algorithms.



(a) 5G-Drive traces



(b) 5G-Walk traces

Fig. 3: Average value of QoE_{HD} for 5G-Drive and 5G-Walk traces attained by various ABR algorithms.

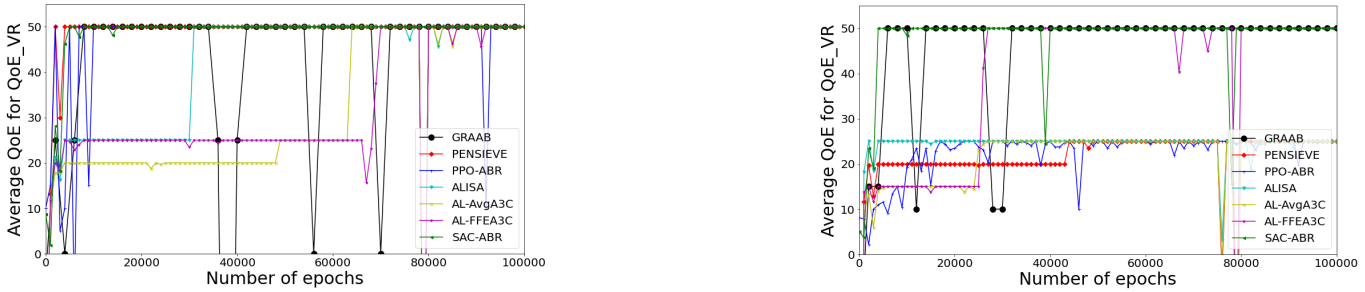
- QoE_{LIN}: This is a generic QoE metric with the rebuffer penalty as $\mu = 160$ and $q(b_n) = b_n$.
- QoE_{HD}: This metric is designed specific to evaluate QoE for HD videos with the rebuffer penalty as $\mu = 192$ and $q(b_n) = HD_{r_t}(b_n)$, where HD_{r_t} is assigned with the reward values of $\{1, 2, 3, 12, 15, 20\}$ when b_n is $\{20, 40, 60, 80, 110, 160\}$ Mbps, respectively.
- QoE_{VR}: This metric is designed specific 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} is assigned with reward values of $\{5, 10, 15, 20, 25, 50\}$ when b_n is $\{20, 40, 60, 80, 110, 160\}$ Mbps, respectively.

D. Results

Table I presents the average QoE for all the variants (QoE_{LIN} and QoE_{HD}, and QoE_{VR}) for both 5G-Drive and 5G-Walk traces. Our results show that GRAAB consistently obtains a higher QoE than other state-of-the-art ABR algorithms for almost all cases. Specifically, GRAAB achieves 47.82% and 161.22% higher average QoE (QoE_{LIN}) than Pensieve for 5G-Drive and 5G-Walk traces. Similar advantages are observed when comparing GRAAB to AL-AvgA3C, AL-FFE3C, PPO-ABR, SAC-ABR and ALISA.

Figure 2a 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 GRAAB achieves a high reward value early on in its training, demonstrating better performance from the initial epochs. GRAAB consistently outperforms other DRL-based ABR methods as training progresses, ultimately achieving the highest reward value. In the case of 5G-Walk traces, as presented in Figure 2b, the GRAAB also 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. Figure 3a, and 3b 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 GRAAB outperformed compared to other competing algorithms. We also present that GRAAB achieved better performance compared to other ABR algorithms with QoE_{VR} metric for both 5G-Drive and 5G-Walk traces, as shown in Figure 4a and Figure 4b, respectively.

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 Figure 5 for the linear metric. Compared to other ABR algorithms, GRAAB attains higher bitrates, reduces rebuffering occurrences, and



(a) 5G-Drive traces

(b) 5G-Walk traces

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

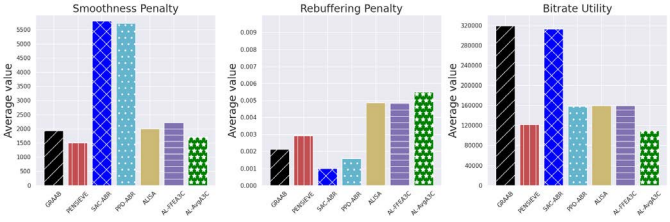


Fig. 5: Comparing PPO-ABR and SAC-ABR with existing DRL-based ABR algorithms by analyzing their performance on the individual components on the 5G traces using a QoE_{LIN} metric.

maintains smoother video playback. These figures provide valuable insight into the overall performance of GRAAB with 5G traces, ultimately resulting in an improved QoE. Similar results are obtained for QoE_{HD} and QoE_{VR} metrics as well, which are not included due to the space constraints.

V. Conclusions

In this work, we have proposed and evaluated the performance of GRAAB in comparison to other state-of-art ABR algorithms. GRAAB enables extensive exploration through gradient sharing and supports variations in policies originating from different workers and converges to better policies with fewer working agents. Future work includes the extensive study of GRAAB for edge-driven video delivery services.

Acknowledgment

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

References

[1] "Cisco annual internet report, 2018-2023." [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive." New York, NY, USA: Association for Computing Machinery, 2012.

[3] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction." Association for Computing Machinery, 2016.

[4] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service." New York, NY, USA: Association for Computing Machinery, 2014.

[5] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[6] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http." New York, NY, USA: Association for Computing Machinery, 2015.

[7] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 197–210.

[8] P. Saxena, M. Naresh, M. Gupta, A. Achanta, S. Kota, and S. Gupta, "Nancy: Neural adaptive network coding methodology for video distribution over wireless networks," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[9] A. Narayanan, X. Zhang, R. Zhu, A. Hassan, S. Jin, X. Zhu, X. Zhang, D. Rybkin, Z. Yang, Z. M. Mao, F. Qian, and Z.-L. Zhang, "A variegated look at 5g in the wild: Performance, power, and qoe implications," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 610–625. [Online]. Available: https://doi.org/10.1145/3452296.3472923

[10] M. Abadi, A. Agarwal, and P. Barham, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[12] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.

[13] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," 2016.

[14] A. B. Labao, M. A. M. Martija, and P. C. Naval, "A3c-gs: Adaptive moment gradient sharing with locks for asynchronous actor–critic agents," pp. 1162–1176, 2021.

[15] N. Mandan, S. Paresh, and G. Manik, "Ppo-abr: Proximal policy optimization based deep reinforcement learning for adaptive bitrate streaming," in *2023 International Wireless Communications and Mobile Computing (IWCMC)*, 2023, pp. 199–204.

[16] M. Naresh, P. Saxena, and M. Gupta, "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)*, 2023, pp. 97–106.

[17] M. Naresh, V. Das, P. Saxena, and M. Gupta, "Deep reinforcement learning based qoe-aware actor-learner architectures for video streaming in iot environments," *Computing*, vol. 104, 07 2022.

[18] M. Naresh, N. Gireesh, P. Saxena, and M. Gupta, "Sac-abr: Soft actor-critic based deep reinforcement learning for adaptive bitrate streaming," in *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*. IEEE, 2022, pp. 353–361.