Hindawi
Wireless Communications and Mobile Computing
Volume 2019, Article ID 5038758, 15 pages
https://doi.org/10.1155/2019/5038758

Research Article

Quality Enhanced Multimedia Content Delivery for Mobile Cloud with Deep Reinforcement Learning

Muhammad Saleem,[1] **Yasir Saleem**,[1] **H. M. Shahzad Asif**,[1] **and M. Saleem Mian**[2,3]

[1]*Department of Computer Science and Engineering, University of Engineering and Technology, Lahore 54890, Pakistan*
[2]*Faculty of Electrical Engineering, University of Engineering and Technology, Lahore 54890, Pakistan*
[3]*Sharif College of Engineering and Technology, Lahore, Pakistan*

Correspondence should be addressed to Muhammad Saleem; saleemnawaz@gmail.com

Received 3 April 2019; Revised 18 May 2019; Accepted 11 June 2019; Published 18 July 2019

Academic Editor: Miguel Garcia-Pineda

The importance of multimedia streaming using mobile devices has increased considerably. The dynamic adaptive streaming over HTTP is an efficient scheme for bitrate adaptation in which video is segmented and stored in different quality levels. The multimedia streaming with limited bandwidth and varying network environment for mobile users affects the user quality of experience. We have proposed an adaptive rate control using enhanced Double Deep Q-Learning approach to improve multimedia content delivery by switching quality level according to the network, device, and environment conditions. The proposed algorithm is thoroughly evaluated against state-of-the-art heuristic and learning-based algorithms. The performance metrics such as PSNR, SSIM, quality of experience, rebuffering frequency, and quality variations are evaluated. The results are obtained using real network traces which shows that the proposed algorithm outperforms the other schemes in all considered quality metrics. The proposed algorithm provides faster convergence to the optimal solution as compared to other algorithms considered in our work.

1. Introduction

The use of tablets and smartphones is growing due to the extensive increase in mobile devices. The advanced mobile capabilities along with Wi-Fi and Long Term Evolution networks have introduced valuable user experience for mobile clients [1]. Cisco report [2] state that the world media traffic will be 78% of the mobile data traffic by 2021, and 86% of this traffic will be produced through smartphones. People are preferring mobile devices for viewing videos through streaming due to the rising popularity of media technology. The clients always demand stable media streaming services irrespective of the service provider.

The content providers use adaptive bitrate (ABR) algorithms for the optimization of video quality. The algorithms are applied on client-side to choose a bitrate for each video segment dynamically. ABR algorithms decide bitrate using throughput estimation and playback buffer occupancy. The aim of ABR algorithms is to increase user quality of experience (QoE) by adjusting video bitrate for underlying network conditions [3]. However, selecting the appropriate bitrate is challenging due to variations in network throughput [4–9] and conflicting QoE requirements such as higher bitrate, minimum rebuffering, and smoothness.

The Moving Picture Experts Group (MPEG) introduced dynamic adaptive streaming over HTTP (DASH) standard to assist deployment of this technology [10, 11]. The commercial platforms like Adobe Dynamic HTTP Streaming [11], Apple HTTP live streaming [12], Microsoft Smooth Streaming [13], and Move Networks use HTTP-based adaptive video streaming because of its popularity.

The DASH runs over the existing HTTP/TCP protocol for video transmission. The video contents are encoded in different resolutions and bitrates which are stored at the server in DASH architecture. The client can select the segment according to the bandwidth and resource availability to switch the quality level for smooth video streaming. This adaptive technique is applied independently to each client for a better quality of experience.

The study on DASH adaptations methods is still in progress. The majority of systems use simple heuristic techniques that result in disturbing quality fluctuation [14] and poor utilization of network resources. The main element in QoE performance degradation is rebuffering, which freezes the video playout temporarily [15]. The current DASH-based techniques are not effective under dynamic network conditions due to mobility and bandwidth variations for mobile devices. This results in frequent quality switches and video freezing which can decrease user QoE for video streaming. It is better to deploy rate switching approach for continuous bandwidth variations and to use relatively high aggressive rate switching for instant bandwidth hoping. The recent research concentrates on overcoming the adaptive streaming issues for dynamic network condition.

Reinforcement learning (RL) is a powerful method due to the autonomic capability of learning without feature crafting. The introduction of reinforcement learning presents an effective and viable solution for rate adaptation methods. RL selects the best strategy on the basis of previous experience through the trial-and-error method [16]. The RL for DASH methodology is based on media presentation file, dynamic wireless bandwidth, and buffer size for user player. When the rate adaptation agent is efficiently trained, the RL-based rate adaptation techniques show better QoE as compared to existing DASH-based methods [17]. The major problem in this policy is the large state space of the RL-based methods. The small state spaces are required to assure fast convergence and to develop online techniques that respond to the variations in the environment instantly.

The rate adaptation algorithms provide better video streaming services. However, these approaches face some challenges for designing and implementing the ABR algorithm as follows. (i) Network bandwidth switches over time that varies considerably across the network environment. Here, the selection of bitrate becomes complex because different network conditions need different values for input parameters. (ii) Rate adaptation algorithm is required to maintain various QoE objectives like enhancing multimedia quality, reducing rebuffering events, and minimizing quality level variations. (iii) The selection of bitrate for segment affects the status of media player significantly. The higher bitrate selection can consume the buffer occupancy and the next segment is downloaded at lower bitrate in order to reduce rebuffering events.

The RL method is applied to many disciplines but it has some limitations for nonstationary environment and large data dimensions [18]. The deep-learning method's capability to learn complex patterns also leads to classification issues [19]. Recently, the RL methods are combined with a deep neural network in order to mitigate these issues [20]. This integration can be applied in training or approximating RL functions using a deep neural network. The Deep Q-Network (DQN) [21] is a notable example of this integration that combines deep Neural Network with Q-learning. The DQN agent can easily learn the policies using RL when the inputs are high dimensional. The deep Convolutional Neural Network (CNN) overcomes the divergence and stability issues by using a target network and replay experience.

We have proposed an adaptive rate control enhanced Double Deep Q-Network (ADQ) method that employs deep reinforcement learning to choose the better quality level based on its prior experience. We have carried out our simulations using dash.js to compare our framework with existing techniques. Our algorithm maximizes the quality of experience by limiting rebuffering events and quality variations. It provides quick convergence to optimal strategies that result in fewer quality variations for better QoE for the client.

The major contributions of this research work are as follows:

(i) Enhancing video streaming quality using double Deep Q-learning that includes reward function, target network, and replay memory.

(ii) Employing HSDPA real network traces and QoE Waterloo III video dataset to evaluate proposed ADQ and other rate adaptation methods.

(iii) Reducing rebuffering events and quality variations for the smooth playback of the videos.

(iv) Improving stability and convergence time for varying network conditions.

This paper is organized as follows. Related work for adaptive multimedia content delivery schemes is discussed in Section 2. Video quality assessment methods are also given in related work. Section 3 defines the system model and problem formulation. Section 4 covers the proposed methodology and algorithms. The evaluation results are discussed in Section 5. Finally, the conclusion is given in Section 6.

2. Related Work

The recent research focuses on HTTP-based adaptive video streaming to enhance the user quality of experience. The sender-driven rate adaptation is one of the frequently explored strategies for video streaming using HTTP that uses existing CDN infrastructure. The videos are stored in the form of multiple segments at the server in DASH-based systems and the duration of a segment is a few seconds. The segments are encoded at different bitrates and a higher bitrate means a larger segment size and a higher video quality.

An interesting hybrid between standard algorithms and reinforcement learning is introduced by van der Hooft et al. [22]. They adapt the parameters of heuristic using Microsoft Smooth Streaming (MSS) to raise the performance of the system. The algorithm is not QoE aware but a similar hybrid solution could be a useful method to improve the QoE by overcoming RL drawbacks.

Buffer-Based Approach (BBA) [23] employs buffer allocation and estimation method to decide next segment bitrate. This algorithm begins with the smaller bitrate and with the rise in buffer allocation a larger bitrate is selected to smooth fluctuations caused by changing network capacity. The BBA is not suitable for short multimedia videos as it has a large buffer size. The BBA usually shows the lowest QoE even for the good network conditions. The buffer-based technique BOLA

[24] uses a Lyapunov optimization formulation to optimize a specified QoE metric.

Additive increase and multiplicative decrease (AIMD) [25] identifies bandwidth variability by applying HTTP throughput for multimedia streaming. The segment fetch time is employed to estimate throughput. A Feedback Linearization Adaptive Streaming Controller (ELASTIC) [26] reduces the network traffic fluctuations by applying feedback control theory. A QoE-aware DASH system (QDASH) [27] is a probing method which computes network bandwidth. It measures the bandwidth in varying network conditions and provides reliable values for video quality. The rate-based [28] method employs ABR controller for the MPEG-DASH standard. It selects highest available bitrate that is less than throughput predicted on mean value of last 5 segments.

The FESTIVE [29] algorithm balances fairness, efficiency, and stability across video players. The FESTIVE is implemented without randomized scheduling and consecutive segments are downloaded with an assumption of no wait time. It uses the harmonic mean of the last 5 segments based on throughput prediction to calculate the efficiency score. The stability score is calculated using bitrate switches of last 5 segments. The performance of the FESTIVE algorithm decreases for the ramp up network condition.

Bokani et al. [30] applied a Markov Decision Process using dynamic programming to balance a stable and high QoE by achieving optimal adaptation logic. The drawback of the approach is its computational complexity. The authors have provided many solutions to decrease the computational complexity but it increases the rebuffing events and requires retaining a huge quantity of computational results in the memory. The authors in [31, 32] have worked on optimization problem instead of heuristics to address RL-based DASH control strategies. A Q-learning-based approach learns the best policy which utilizes reward function to reduce rebuffering events. Although this algorithm is efficient for heuristic-based approach, yet the parameters of the model are designed for slow-varying channels. The algorithm is not suitable for dynamic network conditions. The purpose of the research is to mitigate the current flaws by using learning technique and optimal policy.

QDASH [27] overcomes the issue of sudden quality degradation by choosing the average bitrate. This quality degradation originates because of a sudden drop in bandwidth. The throughput prediction is employed instead of proxy service for estimating bandwidth without affecting the performance using harmonic mean of last 5 segments.

AppATP [33] presents an energy efficient scheme for mobile devices which effectively select the time to prefetch frequently required data and defer delay-tolerant data until better network condition. In [34], the author introduced a mathematical model to capture the relationship between time and scale for peer to peer multimedia streaming under a flash crowd scenario. The authors designed a flexible population control scheme for the flash crowd that alleviates the requirement of costly server deployment.

Yin et al. [35] introduced a Model Predictive Control (MPC) framework that takes decisions using a look-ahead approach. They have used throughput predictor to make an optimal selection of future segments. The method is comparable to dynamic programming that depends on the quality of throughput predictor for a better decision. If the predicted values are not accurate, the decisions for the next state will be suboptimal. The heuristics are applied to address this issue which leads to more conservative throughput predictions. The authors state that it consumes additional memory to store precomputed tables for real-time implementation. These heuristics do not perform well in different environments.

The RL-based adaptive video streaming has been proposed in [22, 31, 32, 36] which uses a tabular form to store and learn the value function for each state and action instead of applying function approximations. These schemes do not perform better due to limited state space. H. Mao et al. [37] proposed A3C for DASH bitrate selection using RL to achieve better performance.

The goal of decision-making and Q-Learning regarding segment selection is to enhance the quality of experience (QoE). Various factors can affect the QoE; however, all factors will have different effects. Optimization of QoE in DASH-based techniques is an open research problem. The authors in [38–40] have observed that quality level variations and the frequency of video freezing are the major factors affecting the QoE. In this research work, we consider the variations in quality level and frequency of video freezing to enhance the QoE for mobile users.

2.1. Video Quality Assessment. PSNR (Peak Signal-to-Noise Ratio) is media quality assessment method. It is computed by selecting the highest bitrate and the difference between the test and reference image [41]. Lee et al. [42] define the PSNR in

$$PSNR_{db} = 20 \log_{10} \frac{MAX_{Bitrate}}{\sqrt{EXP_{Thr} - CRT_{Thr}}} \quad (1)$$

$MAX_{Bitrate}$ represents the bitrate after video encoding. EXP_{Thr} is the expected throughput of multimedia stream over the network and CRT_{Thr} shows the current throughput of stream.

Structural similarity (SSIM) provides the structural information by employing HVS. It is computed using the original and observed image [43]. SSIM index is shown in

$$SSIM(x, y) = [l(x, y)]^{\alpha} . [c(x, y)]^{\beta} . s[(x, y)]^{\gamma} \quad (2)$$

The α, β, and γ parameters in (2) adjust three comparison functions. The general QoE metric for video streaming used by MPC [35] is defined in

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N} |q(R_n + 1) - q(R_n)| \quad (3)$$

Here N denotes the number of segments and R is the set of all possible bitrates. R_n denotes the bitrate of the segment n and q(R_n) represent the quality perceived by a user. The media player can select a video segment for downloading at bitrate $R_n \in R$. When the higher bitrate is chosen, the higher quality is perceived by the user. $q(R_n)$ is quantified and measured by

using Table 4. The rebuffering time is represented by T_n that results from downloading segment n at bitrate R_n.

Rebuffering is a stalling event caused by buffer underrun during the video playback. Rebuffering frequency is determined as the rebuffering counts per minute while playing a video. Switch frequency is the number of bitrate switches during a video playback session.

3. System Model and Problem Formulation

In this section, we will discuss the rate adaptation for DASH-based algorithms using reinforcement learning. Deep reinforcement learning (DRL) is a method in which agent and environment interact with each other using a specified set of actions. The decision-maker and learner is known as agent and the thing that interacts with an agent is called environment. The basic information regarding environmental conditions is contained in the state. The agent evaluates its actions according to assigned numerical rewards and does not need any previous information about the environment. The aim of an agent is to maximize the collective numerical reward by learning optimal action in a given environment [16]. The mapping from each state to actions is called policy π and the ultimate aim of an agent is to determine the optimal policy. The agent selects an action at time t after observing state s_t. When the action takes place, the environment transition state is updated to s_{t+1} and the agent gets a reward R_t. The goal of learning is to constantly update decision-making policy for increasing the expected discounted reward: $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R_t]$; here γ is a discount factor.

Moreover, we will discuss the state, environment, agent, and reward function for DASH-based multimedia streaming.

Table 1 illustrates the notations used in the paper.

3.1. Video Streaming. The video streaming method can be viewed as RL task. It enables the agent to learn the best action using feedback from the environment through trial and error. We define segment size as $F_m(q_m)$. The segment is indicated by m and q_m is the quality of segment m. The value of $F_m(.)$ is known to the client from the MDP file before segment m is downloaded. The mean channel occupancy experienced while downloading the segment m is represented by C_m. We can find the total downloading time [44] τ_m using

$$\tau_m = \frac{F_m(q_m)}{C_m} \tag{4}$$

The playout time and buffer time for the video segment m is denoted by T and B_m, respectively. When $\tau_m > B_m$, the playout buffer remains unoccupied before the complete downloading of the next segment resulting in rebuffering events. For $\tau_m < B_m$, the segment m is downloaded before its defined playout time and the succeeding segment downloading starts quickly that adds extra time $B_m - \tau_m$ to the segment m+1 buffer. The rebuffering time for segment m is defined in

$$\varphi_m = \max(0, \tau_m - B_m) \tag{5}$$

The next segment buffer is calculated using

$$B_{m+1} = T + \max(0, B_m - \tau_m) \tag{6}$$

We have limited the buffer size for our simulations up to 60 seconds.

3.2. State, Action, and Environment. The basic information regarding environmental condition is contained in a state s_k. The agent takes the state as input from the environment and determines the discounted reward as output in order to take action in the given environment. We require determining the state which is fed to the agent and the agent's network.

We require formulating the state transition model for ADQ-based multimedia content delivery system. The rate adaptation is selected at stage m and the next segment is decided at stage m+1. The state vector for the segment at stage m is denoted by s_m. s_m contains all the information of the network after complete downloading of segment m. The state vector consists of three parameters such as rebuffering events f_m, quality level variation Q_m, and available bandwidth Bw as shown in

$$S_m = (f_m, Q_m, \mathrm{Bw}) \tag{7}$$

The quality level of the next segment for delivery is indicated by action a_k and measured as encoding bitrate of the video segment representation. The environment in the DASH system relies on the video player, video source, and network bandwidth.

3.3. Reward and Policy. The reward is a subjective score of a video segment and it is computed when the agent chooses the bitrate of the segment. The RL agent performs an action a_t after receiving the state s_t and this action is selected on the basis of policy. The policy performs the probability distribution over action and is denoted by π. The policy for a given state s_t and action a_t is defined as $\pi(s_t, a_t)$. In RL methods, reward function is determined as a composite of the video quality, bitrate, rebuffering events, and weighting coefficients. A reward function derives policies to increase the QoE for the user. The reward function [44] for segment t is shown in

$$\begin{aligned} R(q_{t-1}, q_t, \varphi_t, B_{t+1}) = {} & q_t - \beta \left\| q_t - q_{t-1} \right\| - \gamma \varphi_t \\ & - \delta \left[\max(0, (B_{thr} - B_{t+1}) \right]^2 \end{aligned} \tag{8}$$

The term q_t at the right side is responsible for the quality of video. The two succeeding negative terms $\beta\|q_t - q_{t-1}\|$ and $\gamma\varphi_t$ are penalty factors that account for the frames sequence and rebuffering events. The term $\delta[\max(0, (B_{thr} - B_{t+1})]^2$ is employed when buffer level is less than defined threshold B_{thr}. It is a penalty factor value which further reduces the occurrence of rebuffering events. The values of δ, β, and γ are weighting terms which add importance to the penalty terms.

3.4. Q-Learning. Q-Learning is an RL algorithm presented by Watkins [45]. In Q-table, rows represent the state s and

columns represent the action a. A Q-value Q(s; a) is stored for every state-action (s; a) pair which represents the quality of taking an action in an environmental state. Q-values are updated after action takes place in a state which results in a reward R and new state s'. Equation (9) shows the new state after taking the action.

$$Q(s,a) = (1-\alpha)\,Q(s,a) + \alpha\left[R + \gamma \max_{a'} Q\left(s', a'\right)\right] \quad (9)$$

In Equation (9), α is a learning rate which determines the agent learning from newly acquired information. The discount factor γ indicates the value of future rewards.

We will discuss the methodology of our proposed adaptive rate control using enhanced Double Deep Q-Learning (ADQ) method in Section 4. The QoE score for the entire episode of viewing video depends upon available bandwidth, resolution, buffer occupancy, and bitrate selection for next segment.

4. Adaptive Rate Control Using Enhanced Double Deep Q-Learning

The presented adaptive rate control enhanced Double DQN (ADQ) algorithm is developed on the basis of Q-learning and deep-learning approach to achieve the best policies for the DASH protocol. The extensions in DQN have been presented due to the growing popularity of deep reinforcement learning. Here, we will propose the rate adaptation algorithm and discuss the enhancements of ADQ in comparison with deep Q-learning methods.

4.1. System Architecture. We have applied ADQ algorithm to learn the best policy. The DASH-based client and server communication of our scheme is shown in Figure 1. The RL converges to the best solution in an efficient way and enhances the reward after a little training period. The client initiates connection with server to select and play the video. The client transmits an HTTP GET request to server after selecting a multimedia file. A multimedia file consists of small segments which are delivered to the client. The Media Presentation Description (MDP) file has information about adaptive streaming for the mobile client. The server stores video segments of different encodings and MPD file. The MPD file includes bit rates, resolution, timing, and URL for the video player. The client then parses the MPD file and regenerates the URLs of video levels encoded at different bitrates. The Request Handling Module (RHM) obtains and analyzes the data received from the mobile client. The media requests are processed by RHM and the requested segment is delivered to the mobile client. The HTTP Manager is responsible for handling HTTP communication between a server and mobile client. The ADQ algorithm uses parameters such as rebuffering events, quality level variation, resolution, and available bandwidth to determine optimal bitrate selection. On the basis of these parameters, the suitable video level is selected by the rate adaptation algorithm at client side. The requested video segment is then delivered to client and the process proceeds until the complete downloading of the video or video termination by the user.

A DQN is a multilayered neural network that results in a vector of action values $Q(s, \ldots, \theta)$ for a specific state s and network parameters θ. This neural network is a function from R^n *to* R^m for a state space of n-dimension and action space having m actions. Minh et al. [46] proposed experience replay and target network usage in the DQN algorithm. The target network has parameters θ' similar to the online network. The difference between target network and online network is that parameters are copied at every τ interval in the target network so that $\theta'_t = \theta_t$ and remained fixed on all remaining intervals. The target used by DQN is given in

$$Y_t^{DQN} \equiv R_{t+1} + \gamma \max_a Q\left(s_{t+1}, a; \Theta'_t\right) \quad (10)$$

An extension of the DQN is Double DQN algorithm [47] which is also a deep RL algorithm. DQN employs the same values to select an action which usually results in overestimated values. We have modified Double DQN (DDQN) [48] method in which the selection is separated from the evaluation process.

DDQN estimates and updates two Q-values for every state-action combination. As a result of these observed Q-values, the Deep Neural Network (DNN) is updated after sequence execution. The updated Q-values are used in the next execution sequence. In Double Q-learning, the experiences are assigned randomly in order to learn two value functions and update one of the two value functions which result in two sets of weights θ and θ'. The first set of weight determines the greedy policy and the second set of weight determines its value for each update. The selection and evaluation are represented in Q-learning to find the target value as shown in

$$Y_t^{Q} \equiv R_{t+1} + \gamma Q\left(s_{t+1}, \arg\max_a Q\left(s_{t+1}, a; \Theta_t\right); \Theta_t\right) \quad (11)$$

The error for Double Q-learning is given in

$$Y_t^{DoubleQ} \equiv R_{t+1} + \gamma Q\left(s_{t+1}, \arg\max_a Q\left(s_{t+1}, a; \Theta_t\right); \Theta_t{}'\right) \quad (12)$$

The selection of the action is due to the online weights θ_t in the argmax. It means the greedy policy values are still computed using the first set of weights θ_t as in Q-learning process. The second set of weights θ'_t *is* used to evaluate the value of the policy. The roles of θ and θ' can be switched to update the second set of weights.

The expected QoE score depends on the player buffer occupancy, network bandwidth, and bitrate selection for the next video segment. The agent using better adaptation scheme can make use of network resources efficiently and select the optimal bitrate to achieve highest QoE for the user. Moreover, the quality variation and rebuffering events can be reduced with the agent's optimal policy.

The introduction of ADQ approach reduces quality variations and rebuffering events and contributes to the

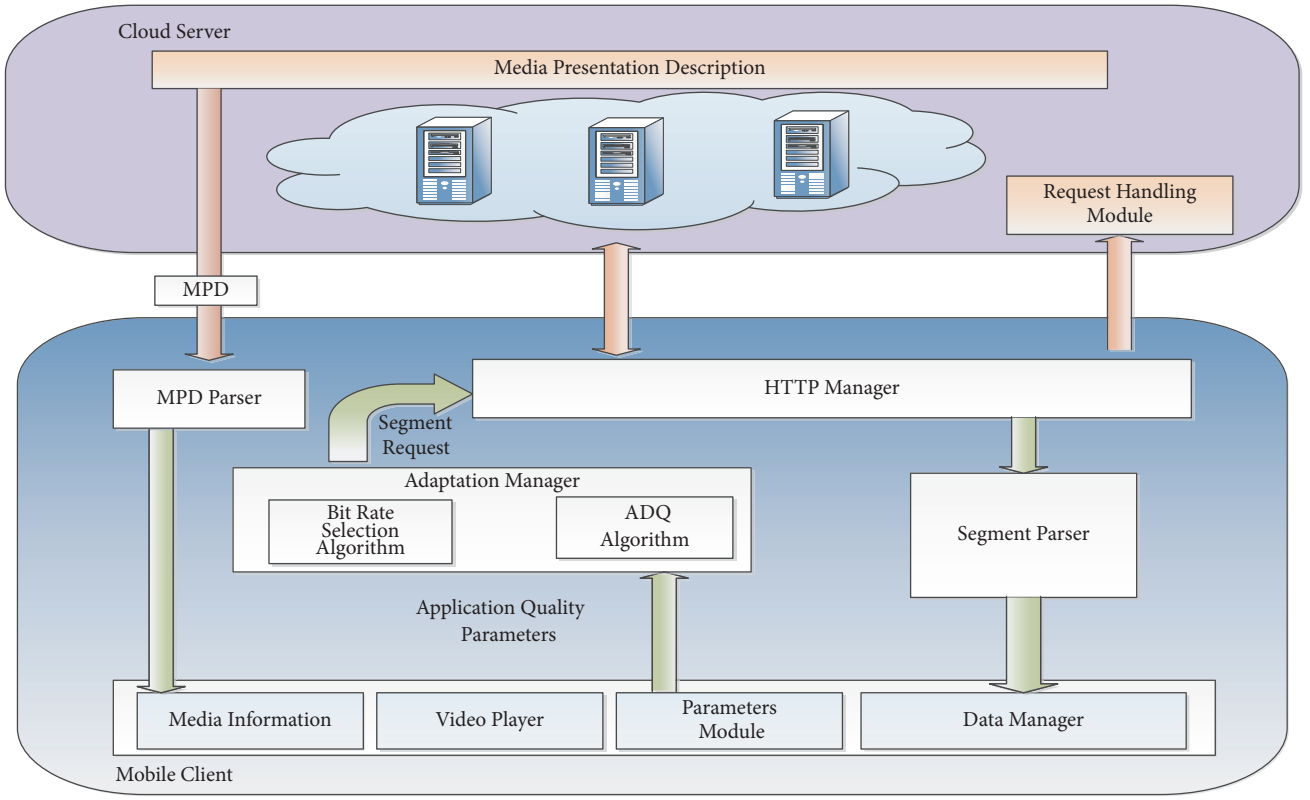

Figure 1: Rate adaptation DASH architecture for mobile client.

performance improvement. We propose improvements in the learning mechanism of reward function and prioritized experience replay. The reward is calculated over last k steps, where k is the number of times the video quality varies in the last 1 minute. The introduction of using k steps for reward calculation reduces the quality variations as it provides a more stable estimation of the target bitrate for the next video segment. The improved target network and experience replay significantly increase the performance of the algorithm. We define [k] = {1, ⋯ , k} and target is calculated using

$$Y_t^{ADQ} \equiv \max_{i \in k} R_i + \gamma Q\left(s_{t+1}, \arg\max_a Q\left(s_{t+1}, a; \Theta_t\right); \Theta_t{}'\right) \quad (13)$$

The experience replay memory is employed to mitigate the correlation between the data and nonstationary distribution. The agent samples the data randomly from the prior experience memory to sample a minibatch of tuples. An experience is assigned with a loss factor to select samples with greater loss using distributed prioritized experience replay. The loss factor [49] is calculated using

$$l_t(\theta) = \frac{1}{2}\left(G_t - q\left(s_t, a_{t,}\theta\right)\right)^2 \quad (14)$$

G_t is the multistep return that is given by

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n q\left(s_{t+n}, \arg\max_a q\left(s_{t+n}, a, \theta\right), \theta'\right) \quad (15)$$

Figure 2 shows the state and action update using the ADQ method. The state of the system consists of available bandwidth, rebuffering events, and quality level variation. Initially, the current state s_m is input to the neural network which estimates the Q-value for all the actions in the environment. The action q_t is chosen on the basis of ε-greedy policy. While taking the action q_t, a new state s_{m+1} is updated and a new reward is calculated accordingly. This updated information is then stored in the replay memory D. The system then randomly extracts M samples from this replay memory and updates the network weights by using an optimization method. These weights are updated at every K step. The updated weights of the target network minimize the loss function value and hence select the optimal policy.

4.2. ADQ Training Algorithm. The ADQ method finds the best policy according to Algorithm 1. R_m is the reward for segment m. We considered two deep neural networks for training purpose. The first network called online network is upgraded for all new segments with weight W_t at each time step t. It is used for mapping Q values. The target network is

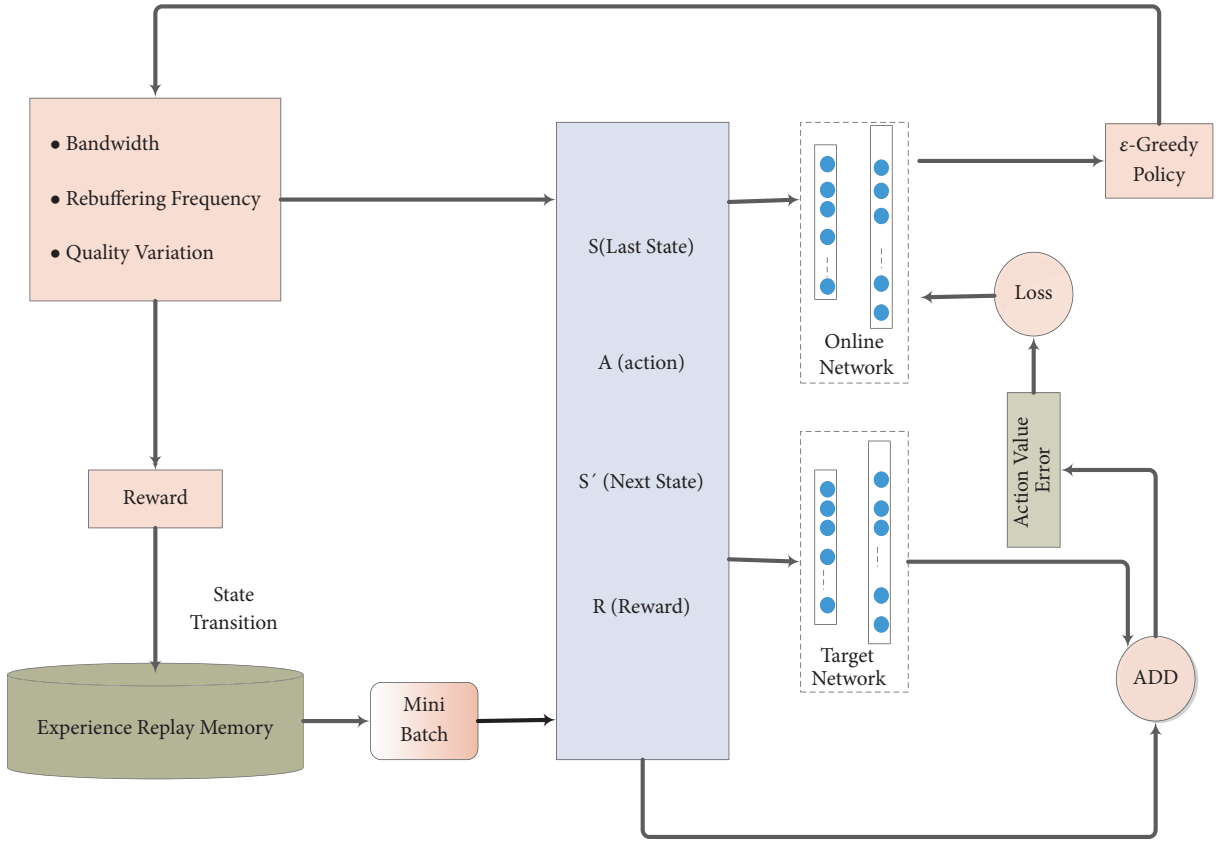

Figure 2: State update mechanism using ADQ.

Input: State (Bw, Q_m, f_m)
Output: Optimal policy to select action
Initialization: Experience Replay Memory D, Online Network Weights θ, Target Network Weights $\Theta' = \Theta$, online action value function $Q(s, a; \Theta)$, target action value function $Q(s', a', \Theta')$, k=i=0

(1) ***for*** *video-episode i=1 to E do*
(2) *Initialize state sequence for received selected video episode*
(3) ***for*** *m=1 to M do*
(4) *Select action a according to* ϵ*- greedy policy from* $Q(s, a; \Theta)$ *with probability* $\in \arg\max_a Q(\varphi(s_m), a; \Theta)$
(5) *Execute action a and observe reward* R_m
(6) *Set* $s_{m+1} = s_m, a_m$ *and preprocess the state*
(7) $\varphi_{m+1} = \varphi(s_{m+1})$
(8) *Store transition* $(\varphi_m, a_m, R_m, \varphi_{m+1})$ *in D*
(9) *Sample a mini batch of tuples* (s, a, R_m, s') *from distributed prioritized replay memory D*
(10) $a^{max} = \arg\max_{a \in A} Q(s_{m+1}, a; \Theta_m); \Theta_m{}')$ *// A is all possible set of actions*
(11) *Determine*
(12)
$$y_m^{ADQ} = \begin{cases} R_m, & \text{if episode } i \text{ terminates at iterations } m+1, \\ R_m + \gamma Q(s_{m+1}, \arg\max_a Q(s_{m+1}, a; \Theta_m); \Theta_m{}'), & \text{otherwise} \end{cases}$$
(13) *reset* $\theta' = \theta$
(14) $s = s'$
(15) ***end for***
(16) ***end for***

Algorithm 1: ADQ Training.

(1) Get the Device Resolution $\boldsymbol{R_d}$
(2) Start the bandwidth trace
(3) ***While*** *video all segments are not downloaded completely*
(4) Find the estimated bandwidth using $\boldsymbol{Bw_{est}^{(t)}}$ *using equation (16)*
(5) Choose the desired bitrate based on the current state
(6) Request and download the video segment according to Device resolution and desired bitrate
(7) Determine the rebuffering events and quality variations
(8) Compute QoE score for the user using equation (3)
(9) Calculate Objective quality assessment metrics score for PSNR and SSIM
(10) ***End while***

Algorithm 2: ADQ Testing.

employed for improving the stability of the system. The target network weights are upgraded at every k steps by assigning online network and remain same for further k-1 steps. Target network and online network use the value of next state s'_{m+1} for computing the optimal value $Q(s', a'; \theta)$. The target value Y^{ADQ} is calculated from the reward R and the discount factor γ. The weights of θ are updated by backpropagation of loss function values to the online network. To mitigate the stability issues, we use experience replay memory D with ADQ. A set of minibatch transactions is selected from the distributed prioritized replay memory to train the Q-network instead of using recent transitions.

The algorithm runs all episodes of the videos and for each episode, the state sequence is initialized as a default state. The inner loop downloads all segments sequentially. The action is selected based on the ϵ- greedy policy from $Q(s, a; \Theta)$. The action is executed and reward is observed for the current action. The next state is updated using the current state and action. The state transition is stored into the distributed prioritized replay memory.

4.3. Bandwidth Estimation and ADQ Testing Algorithm. We have used measurement based prediction [50] to determine estimated available bandwidth by employing the Exponentially Weighted Moving Average (EWMA). It deploys the recently perceived data and the weights of previous data in order to adjust the weights dynamically. The EWMA filter is then employed to estimate the network bandwidth as shown in

$$Bw_{est}^{(t)} = Bw^{(t-1)} + E_D{}^{(t)} \tag{16}$$

Here, $Bw_{est}^{(t)}$ represents the estimated bandwidth for t time interval, E_D represents the estimation difference, and $Bw^{(t-1)}$ indicates the bandwidth of t-1 time interval. The estimation difference E_D is calculated by adjusting the weights as shown in

$$\begin{aligned} E_D{}^{(t)} &= \left(1 - \alpha^{(t)} - \beta^{(t)}\right) E_D{}^{(t-1)} \\ &\quad + \alpha^{(t)} \left(Bw^{(t-1)} - Bw^{(t-2)}\right) + \beta^{(t)} Bw_{std} \end{aligned} \tag{17}$$

α represents the weight of moving average and β represents the weight of standard deviation.

The testing process of ADQ is described in Algorithm 2. The ADQ agent selects the optimal bitrate for next segment of a video using trained dataset in the testing phase. The algorithm is continually executed until the complete downloading of all video segments. The device resolution is determined and stored in the memory. The bandwidth trace is started to emulate the real network scenario for varying environment. When the video is selected for playback, the outer loop selects all segments sequentially. After selecting each segment, the bandwidth is computed using (16). The desired bitrate is determined according to current state of the system. The segment download is initiated according to the device resolution and desired bitrate. The algorithm counts the rebuffering events and quality variations during the video playback session. The QoE score is calculated using (3). The objective quality assessment metrics like PSNR and SSIM are employed to measure video quality.

We have compared the proposed ADQ method with FESTIVE, BBA, QDASH, MPC, Rate-based, and A3C in the testing phase. The dominant QoE factors are video quality, stability and, smoothness.

5. Results and Discussion

DASH.js is a web standard that employs the HTML5 video elements. The dash.js (version 2.9) [51] is modified to evaluate ADQ and existing ABR algorithms. It is configured to receive the video stream according to the selected bitrate. The buffer size of media player is configured for 60 seconds. The Google Chrome browser (version 71) and APACHE server are used for testing. The server machine for testing is 3.7 GHz Intel quad core processor and 16 GB RAM.

We have used Waterloo Streaming QoE Database III (SQoE-III) [52] for our implementation and result evaluation. 5 video clips are used to verify the ADQ method. Table 2 shows the frame rate (FPS), temporal information (TI), and spatial information (SI) of chosen videos. Each video is played for 300 segments which are encoded in six distinct bitrate levels. The duration of each segment is 2 seconds.

The mobile device classification according to screen resolution is shown in Table 3.

The FFmpeg [53] is employed to provide different encodings from the original video. This command line software is

Table 1: Notations.

Symbol	Description	Symbol	Description
m	Video segment	γ	Discount factor
f	Rebuffering events	D	Experience Memory
N_D	Experience Memory Capacity	B_m	Buffer time for segment m
Bw	Available bandwidth	Q_m	Quality level variation for segment m
π	Policy	s_m	State for segment m
R_d	Device Resolution	R	Current reward
f	Rebuffering events	q(R_m)	User perceived quality for segment m

Table 2: Description of Reference Videos [52].

Video ID	FPS	SI	TI	Description
BigBuckBunny	30	96	97	Animation, high motion
RushHour	30	52	20	Human, smooth motion
Ski	30	61	82	Sport, high motion
TallBuildings	30	81	13	Architecture, static
TrafficAndBuilding	30	66	15	Architecture, static

a fast encoder-decoder tool for converting videos to different sizes, formats, and bitrates.

The video is ranked using the mean opinion score as illustrated in Table 4. The quality scale for subjective testing is modified to relate six quality levels with mean opinion score.

5.1. Training Phase. The network traces of public datasets are employed for evaluation of algorithms on the basis of real network conditions. The datasets include a 3G/HSDPA mobile dataset [54] and a 4G trace dataset for different mobility patterns [55]. We use 135 throughput traces and average duration of each trace is 10 minutes. The throughput ranges from 0 to 173 Mbits/s with a granularity of one sample per second. ABR streaming is applied to RushHour video on 135 real network traces during training phase.

We have compared the training phase of our proposed ADQ method with existing RL-based ABR algorithms. The basic parameters of ABR algorithms are given in Table 5. The new state, old state, reward, and action are collected in the training phase which upgrades the Q-value network weights regularly.

Deep Q-learning is employed to select an action and to update Q-value in online phase. The DRL agent computes all actions for decision time t_k and system state s_k in the form of Q (s_k, a) using the DNN. In ε-greedy policy, each agent selects an action with the highest value of Q(s_k, a) estimated by probability 1- ε. The observed total reward $R_k(s_k, a_k)$ is used for updating the Q-value after action a_k occurs during the time interval $[t_k, t_{k+1}]$.

The greedy policy is used for the random bitrate selection in the prestages of the training phase. It trains an agent for all possible states of the environment. Training phase of our proposed method is illustrated in Algorithm 1. In training phase, the algorithm uses multistep rate selection to gain experience using distributed prioritized replay memory.

The well-trained agent can adapt the dynamic variations of network throughput for selecting the optimal bitrate. The main priority of the ADQ method is continuous video playback by using the maximum available bandwidth.

The learning machine uses an automatic mechanism that takes raw data to find the best representation for classification automatically. Deep-learning techniques use several layers of artificial neurons. The purpose of every layer is to transform the input into an abstract representation which is selected as the input of the next layer.

The parameters and values used for FESTIVE, MPC, QDASH, A3C, and proposed ADQ algorithm is given in Table 5.

The convergence speed is evaluated for RL-based and ABR methods. The video episodes experienced by the agent are set along the x-axis and the average reward value is set on the y-axis to show the convergence in Figure 3. It is clear from the graph that ADQ has higher QoE score and greater convergence speed.

The A3C and QDASH algorithms are used as a benchmark for comparison with the proposed technique. The simulations are carried out by employing greedy policy to get the highest reward after completing each video episode. The existing QDASH algorithm takes about 105 episodes to approach the A3C algorithm. QDASH obtains the lowest reward at convergence and ADQ achieves high reward after fewer video episodes.

In the initial stage of the training phase, bitrate is selected randomly using a greedy policy. It allows the agent to explore all possible and feasible states. The greedy policy is employed in the training phase for balancing exploration and utilization. The training mechanism is described in Algorithm 1. This algorithm selects the bitrate of a video segment, accumulates the multistep experience, and stores into the distributed replay memory. The network weights are updated according to error of loaded experience. The algorithm improves the performance and convergence speed. Figure 6 shows the comparison of RL-based approach with heuristic methods using the data set. The mean QoE score of ADQ

Table 3: Encoding Videos in Different Quality Levels Based On Screen Resolutions.

Device Classes	Class 1	Class 2	Class 3	Class 4	Class 5
QL1-3840 kbps	RES<1280 × 720 > FR <30 fps>	RES<960 × 544 > FR <30 fps>	RES<800 × 448 > FR <25 fps>	RES<480 × 320 > FR <20 fps>	RES<320 × 240 > FR <20 fps>
QL2-1920 kbps	RES<800 × 448 > FR <30 fps>	RES<960 × 544 > FR <30 fps>	RES<800 × 448 > FR <25 fps>	RES<480 × 320 > FR <20 fps>	RES<320 × 240 > FR <15 fps>
QL3-960 kbps	RES<512 × 228 > FR <25 fps>	RES<592 × 366 > FR <25 fps>	RES<800 × 448 > FR <25 fps>	RES<480 × 320 > FR <20 fps>	RES<320 × 240 > FR <15 fps>
QL4-480 kbps	RES<320 × 176 > FR <20 fps>	RES<368 × 208 > FR <20 fps>	RES<480 × 272 > FR <20 fps>	RES<480 × 320 > FR <20 fps>	RES<320 × 240 > FR <15 fps>
QL5-240 kbps	RES<320 × 176 > FR <15 fps>	RES<368 × 208 > FR <15 fps>	RES<288 × 160 > FR <15 fps>	RES<300 × 200 > FR <15 fps>	RES<320 × 240 > FR <15 fps>
QL6-120 kbps	RES<320 × 176 > FR <10 fps>	RES<368 × 208 > FR <10 fps>	RES<288 × 160 > FR <10 fps>	RES<300 × 200 > FR <10 fps>	RES<320 × 240 > FR <10 fps>

Table 4: Quality Scale for Subjective Testing (ITU-T R P.911).

Score ($q(R)$)	Quality Level	Impairment
5	QL1-3840 kbps	Perceptible
4	QL2-1920 kbps	Perceptible but not annoying
3	QL3-960 kbps	Slightly annoying
2	QL4-480 kbps	Annoying
1	QL5-240 kbps	Very annoying
0.5	QL6-120 kbps	Extremely annoying

is greater than the considered methods on the validation set. The enhanced performance of ADQ is particularly due to the less rebuffering events and higher bitrate level.

5.2. Testing Phase. We have used BigBuckBunny, Ski, TallBuildings, and TrafficAndBuilding videos for the testing phase. Figure 4 shows the real bandwidth trace used during the experimental evaluation of our proposed ADQ method and existing algorithms. A bandwidth throttling module [54] is employed to simulate the bandwidth that creates the real-time scenario for testing. The testing of proposed ADQ for video playback is presented in Algorithm 2. The algorithm chooses the appropriate bitrate for downloading of a video segment. The inner loop continuously runs until all the video segments are downloaded. The throughput traces have different network scenarios to test the ADQ algorithm. The trained ADQ agent can adjust the dynamic network variations. ADQ method addresses the playback fluency by utilizing the entire bandwidth.

The results are evaluated for each FESTIVE, BBA, QDASH, MPC, Rate-based, A3C, and ADQ algorithms. The PSNR, SSIM, rebuffering frequency, total switch frequency, and QoE are measured for each method to find the video quality. The average quality metrics are shown in Table 6.

The average values of PSNR and SSIM in testing process using dynamic traces are shown in Figure 5. The PSNR value shows improvement for ADQ over other techniques. The ADQ and A3C are capable of maintaining an average SSIM higher than 0.80 for each video under consideration. The ADQ algorithm achieves larger values of SSIM as compared to FESTIVE and rate-based heuristic. It is noticed that FESTIVE shows the better SSIM in comparison with rate-based heuristic. The ADQ outperforms FESTIVE, MPC, Rate-based, BBA, A3C, and QDASH algorithms in terms of PSNR and SSIM.

Figure 6(a) shows the rebuffering frequency for each technique. The MPC shows stability in video quality but it experiences rebuffering events because of optimistic throughput prediction. MPC and FESTIVE show better video stability and lesser rebuffering events. The MPC is computationally intensive and real-time implementation needs precomputed data that results in memory consumption. The ADQ performs better than MPC and FESTIVE with very fewer rebuffering events. Figure 6(b) shows the total switch frequency for the ADQ and existing ABR algorithms. The switch frequency for FESTIVE is lower so it shows stability in video quality. The rate-based method experiences huge quality fluctuations that have a significant effect on user QoE. ADQ shows significantly better performance as compared to FESTIVE and rate-based methods for the high bandwidth fluctuations. The ADQ performs better due to low rebuffering and quick convergence using real capacity traces.

Figure 7 shows the QoE performance. The QoE is measured using (3). The QoE performance of ADQ method is greater than the existing methods for the real-time network. It uses different mobility patterns, particularly with the low network bandwidth pattern. The proposed method shows high flexibility to different network conditions for gaining better QoE while considering the available bandwidth. The existing algorithms employ fixed control laws and do not adapt to varying network conditions. A3C performance is comparable to ADQ in terms of QoE metric. It is clear

Table 5: Adaptation algorithm parameters.

Algorithm	Parameters	Values
FESTIVE	Target buffer	15 s
	Buffer Randomness	0.25 s
	α	10
MPC	Buffer Size	30 s
	γ	2
QDASH	α	0.1
	Policy	Softmax
A3C	γ	0.99
	Policy	E greedy
	Batch Size	128
	Actor learning rate	10-4
	Critic learning rate	10-3
ADQ	Hidden neurons Nh	256
	Learning rate	10-3
	Policy	Greedy Policy
	Lambda	0.9

Table 6: Video Quality Assessment results using HSDPA Real Traces.

Method	PSNR	SSIM	Rebuffering Frequency	Total Switch Frequency	QoE
FESTIVE	22.42	0.76	0.034	19	0.69
BBA	18.78	0.58	0.003	51	0.60
QDASH	21.67	0.73	0.026	69	0.71
MPC	22.69	0.79	0.031	39	0.79
Rate-based	21.12	0.72	0.035	22	0.66
A3C	22.95	0.82	0.025	32	0.81
ADQ	24.01	0.89	0.002	14	0.88

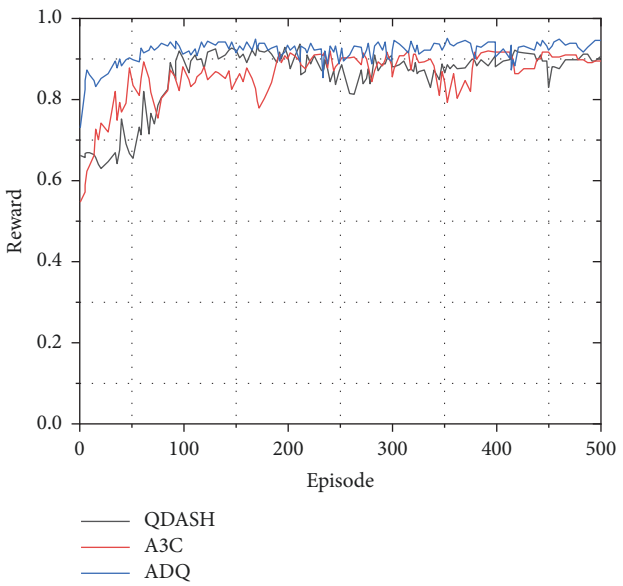

Figure 3: Reward for the A3C, QDASH, and ADQ using real bandwidth trace.

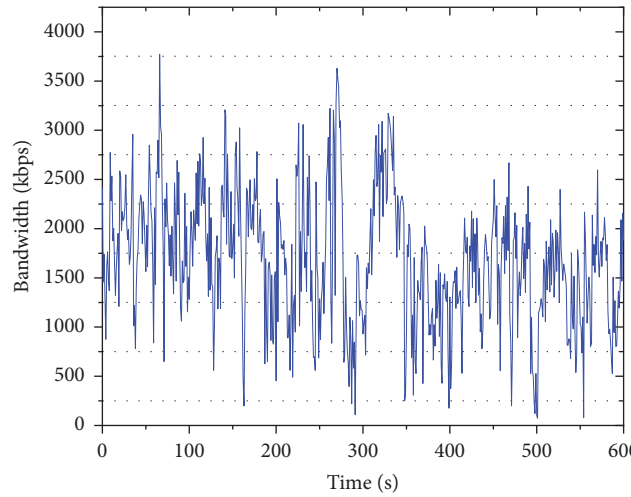

Figure 4: Dynamic bandwidth for moving car.

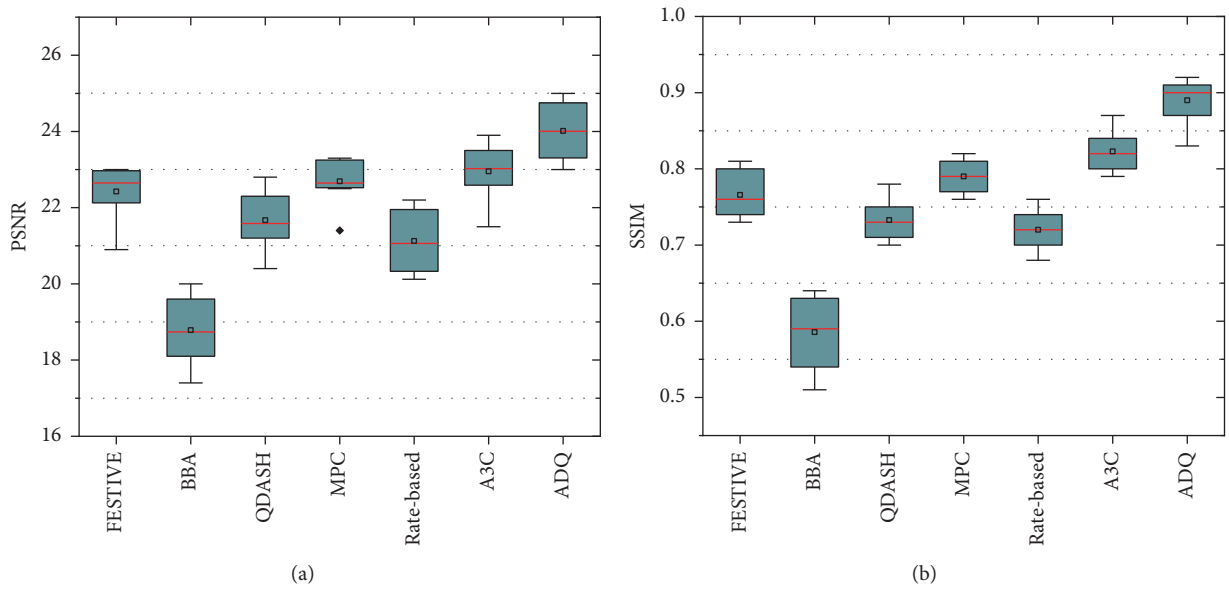

Figure 5: Boxplot of the average. (a) PSNR in test phase. (b) SSIM in test phase.

that the average QoE value of ADQ method is higher than RL based and heuristic methods. The main reason for the improvement in QoE is fewer rebuffering events, infrequent quality variations, and a higher bitrate level.

ADQ method introduces double Q-learning that increases the computational complexity. However, the benefits of ADQ restrict the cost to the training phase. Once the ADQ is trained well, it can provide better user experience by performing the adaptive rate selection at low cost.

We have used multibitrate levels and real-time HSDPA datasets to compare the proposed algorithm with existing algorithms. The advantages of rate-based and buffer-based algorithms are least quality variations and fewer rebuffering events, respectively. The MPC can perform trade-off between rebuffering frequency and total switch frequency to some extent but the implementation is hard due to the computational complexity which results in poor QoE performance. The performance of FESTIVE and QDASH algorithm is closer to our proposed method but there is still a significant gap. We notice that ADQ is able to keep the rebuffering events and quality variation length minimum throughout the video playback and maintains a higher bitrate level with low bitrate switching. Furthermore, we have achieved a higher QoE performance using real-time networks traces. The results show that our proposed algorithm ADQ outperforms BBA, QDASH, A3C, and Rate-based algorithms. Moreover, it is better than the heuristic methods such as MPC and FESTIVE.

The intelligent QoE-aware adaptation approaches have a better buffer control policy. Our algorithm uses the buffered segments for the video playback when the network bandwidth drops and increases the buffer according to the network capacity. The ADQ avoids the sudden fluctuations in playback

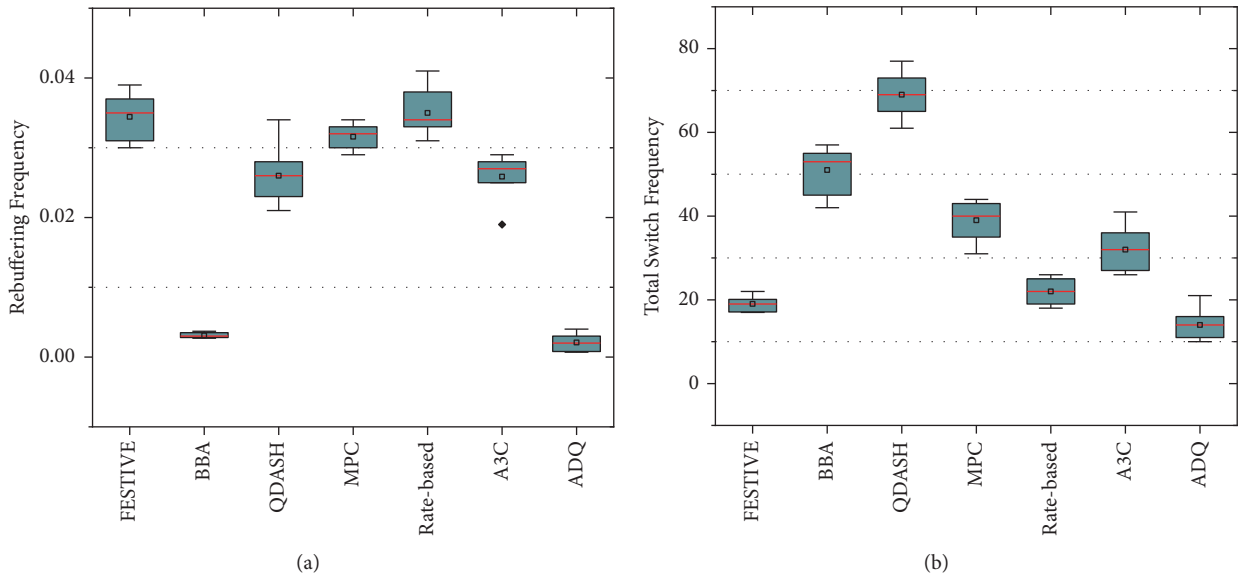

Figure 6: Boxplot of the average. (a) Rebuffering Frequency. (b) Total Switch Frequency.

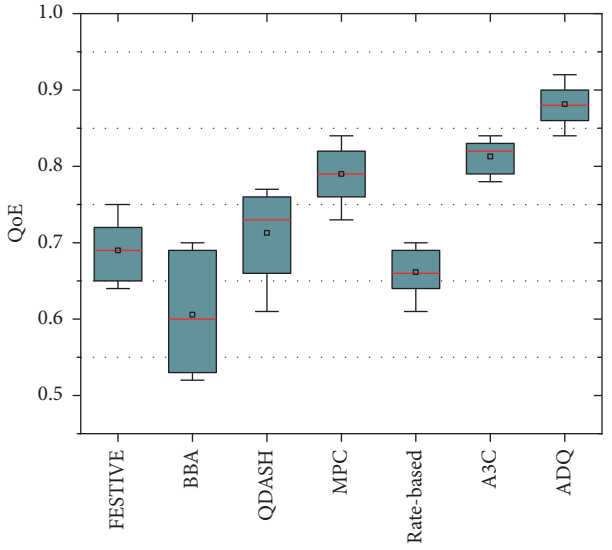

Figure 7: Boxplot of the average QoE in test phase using HSDPA real traces.

video quality with reduced rebuffering events. The reward function obtained from the HSDPA dataset assures that QoE oriented policy is well-trained rate adaptation policy; however, circumstances that affect QoE are very complex. The QoE includes three factors such as bitrate variation, rebuffering frequency, and average QoE. Stalling events and initial delay can be integrated with the learning-based schemes in the near future so that learned policy results in the higher QoE. Reinforcement learning can also be used at the base of various DASH clients during network resource allocation to make the decisions according to the available network resources. The bitrate adaption over HTTP based on network and device parameters is an attractive subject for multimedia content delivery.

6. Conclusion

We have proposed an ADQ method based on the enhanced Double Deep Q-Learning. The ADQ method introduces improvements in the replay experience and double DQN network architecture so that the client agent learns efficiently through the previous experience and converges quickly to the optimal policy. We use HSDPA dataset for evaluation of our algorithm. The proposed algorithm is implemented in dash.js and compared with other algorithms considered in the study. The ADQ method converges faster than the FESTIVE, A3C, MPC, and QDASH during the training phase. The ADQ agent converges to higher bitrate adaptation policy while experiencing a few video segments during the training phase.

These improvements occur due to the changes in the Q-value network architecture and learning process. The evaluation results depict that ADQ performs better than the considered rate adaptation techniques in terms of video quality, QoE, rebuffering frequency, and total switch frequency.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] F. Liu, P. Shu, H. Jin et al., "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 14–22, 2013.

[2] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update. Cisco, San Jose, Calif, USA, 2021-2016.

[3] K. R. Smith, H. Liu, L.-T. Hsieh, X. de Foy, and R. Gazda, "Wireless adaptive video streaming with edge cloud," *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 1061807, 13 pages, 2018.

[4] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, timid, and unstable: picking a video streaming rate is hard," in *Proceedings of the 2012 ACM conference on Internet measurement conference - IMC '12*, p. 225, Boston, Mass, USA, 2012.

[5] Y. Sun, X. Yiny, J. Jiangy et al., "CS2P: improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference - SIGCOMM '16*, pp. 272–285, Florianopolis, Brazil, 2016.

[6] K. Winstein, A. Sivaraman, and H. Balakrishnan, Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks, p. 13.

[7] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*, pp. 509–522, London, UK, 2015.

[8] X. K. Zou, J. Erman, V. Gopalakrishnan et al., "Can accurate predictions improve video streaming in cellular networks?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications - HotMobile '15*, pp. 57–62, Santa Fe, NMo, USA, 2015.

[9] S. Q. Jabbar, D. J. Kadhim, and Y. Li, "Developing a video buffer framework for video streaming in cellular networks," *Wireless Communications and Mobile Computing*, vol. 2018, 13 pages, 2018.

[10] D. J. Vergados, A. Michalas, A. Sgora, and D. D. Vergados, "A fuzzy controller for rate adaptation in MPEG-DASH clients," in *Proceedings of the 2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*, pp. 2008–2012, Washington, DC, USA, 2014.

[11] J. Kua, G. Armitage, and P. Branch, "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1842–1866, 2017.

[12] Apple Inc. Available: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/Introduction/Introduction.html, 2018.

[13] A. Zambelli, "Smooth Streaming Technical Overview," p. 17.

[14] T. Hossfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming," in *Proceedings of the 2014 Sixth International Workshop on Quality of Multimedia Experience (QoMEX)*, pp. 111–116, Singapore, Singapore, 2014.

[15] Z. Li, A. C. Begen, J. Gahm, Y. Shan, B. Osler, and D. Oran, "Streaming video over HTTP with consistent quality," in *Proceedings of the 5th ACM Multimedia Systems Conference on - MMSys '14*, pp. 248–258, Singapore, Singapore, 2014.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2017.

[17] M. Claeys, S. Latre, J. Famaey, and F. De Turck, "Design and evaluation of a self-learning HTTP adaptive video streaming client," *IEEE Communications Letters*, vol. 18, no. 4, pp. 716–719, 2014.

[18] F. Woergoetter and B. Porr, "Reinforcement learning," *Scholarpedia*, vol. 3, no. 3, p. 1448, 2008.

[19] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: high confidence predictions for unrecognizable images," https://arxiv.org/abs/1412.1897, 2014.

[20] L. Zou, R. Trestian, and G.-M. Muntean, "A utility-based priority scheduling scheme for multimedia delivery over LTE networks," in *Proceedings of the 2013 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB '13)*, pp. 1–7, London, UK, 2013.

[21] Y. Li, "Deep reinforcement learning: an overview," https://arxiv.org/abs/1701.07274, 2017.

[22] J. van der Hooft, S. Petrangeli, M. Claeys, J. Famaey, and F. De Turck, "A learning-based algorithm for improved bandwidth-awareness of adaptive streaming clients," in *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 131–138, Ottawa, Canada, 2015.

[23] 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," in *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14*, pp. 187–198, Chicago, Ill, USA, 2014.

[24] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proceedings of the IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, San Francisco, Calif, USA, 2016.

[25] C. Liu, I. Bouazizi, and M. Gabbouj, "Rate adaptation for adaptive HTTP streaming," in *Proceedings of the second annual ACM conference on Multimedia systems - MMSys '11*, p. 169, San Jose, Calif, USA, 2011.

[26] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," in *Proceedings of the 2013 20th International Packet Video Workshop*, pp. 1–8, San Jose, Calif, USA, 2013.

[27] R. K. P. Mok, X. Luo, E. W. W. Chan, and R. K. C. Chang, "QDASH: A QoE-aware DASH system," p. 12.

[28] DASH Industry Forum — Catalyzing the adoption of MPEG-DASH." [Online]. Available: https://dashif.org/. [Accessed: 30-Dec-2018].

[29] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 326–340, 2014.

[30] A. Bokani, M. Hassan, and S. Kanhere, "HTTP-based adaptive streaming for mobile clients using markov decision process," in *Proceedings of the 2013 20th International Packet Video Workshop*, pp. 1–8, San Jose, Calif, USA, 2013.

[31] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, "Design and optimisation of a (FA)Q-learning-based HTTP adaptive streaming client," *Connection Science*, vol. 26, no. 1, pp. 25–43, 2014.

[32] M. Claeys, S. Latré, J. Famaey, T. Wu, W. Van Leekwijck, and F. De Turck, Design of a Q-Learning-based Client Quality Selection Algorithm for HTTP Adaptive Video Streaming, p. 9.

[33] F. Liu, P. Shu, and J. C. S. Lui, "AppATP: An Energy Conserving Adaptive Mobile-Cloud Transmission Protocol," *IEEE Transactions on Computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[34] F. Liu, B. Li, L. Zhong, H. Jin, and X. Liao, "Flash crowd in P2P live streaming systems: fundamental characteristics and design implications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 7, pp. 1227–1239, 2012.

[35] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*, pp. 325–338, London, UK, 2015.

[36] F. Chiariotti, S. D'Aronco, L. Toni, and P. Frossard, "Online learning adaptation strategy for DASH clients," in *Proceedings of the 7th International Conference on Multimedia Systems - MMSys '16*, pp. 1–12, Klagenfurt, Austria, 2016.

[37] 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 - SIGCOMM '17*, pp. 197–210, Los Angeles, Calif, USA, 2017.

[38] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of HTTP video streaming," in *Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pp. 485–492, Dublin, Ireland, 2011.

[39] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," *IEEE/ACM Transactions on Networking*, vol. 21, no. 6, pp. 2001–2014, 2013.

[40] D. Ghadiyaram, A. C. Bovik, H. Yeganeh, R. Kordasiewicz, and M. Gallant, "Study of the effects of stalling events on the quality of experience of mobile streaming videos," in *Proceedings of the 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 989–993, Atlanta, Ga, USA, 2014.

[41] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[42] S.-B. Lee, G.-M. Muntean, and A. F. Smeaton, "Performance-aware replication of distributed pre-recorded IPTV content," *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 516–526, 2009.

[43] Z. Wang, E. P. Simoncelli, and A. C. Bovik, "Multiscale structural similarity for image quality assessment," in *Proceedings of the Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, pp. 1398–1402, Pacific Grove , Calif, USA, 2003.

[44] M. Gadaleta, F. Chiariotti, M. Rossi, and A. Zanella, "D-DASH: A Deep Q-Learning Framework for DASH Video Streaming," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 703–718, 2017.

[45] C. J. C. H. Watkins, *Learning from Delayed Rewards*, University of Cambridge, England, UK, 1989.

[46] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[47] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," https://arxiv.org/abs/1509.06461, 2015.

[48] H. V. Hasselt, Double Q-learning, p. 9.

[49] D. Horgan et al., "Distributed Prioritized Experience Replay," https://arxiv.org/abs/1803.00933, 2018.

[50] C. Lai, H. Wang, H. Chao, and G. Nan, "A Network and Device Aware QoS Approach for Cloud-Based Mobile Streaming," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 747–757, 2013.

[51] Akamai. 2016. dash.js. Dash Industry Forum, 2018.

[52] Z. Duanmu, A. Rehman, and Z. Wang, "A Quality-of-Experience Database for Adaptive Video Streaming," *IEEE Transactions on Broadcasting*, vol. 64, no. 2, pp. 474–487, 2018.

[53] FFmpeg. Available: https://www.ffmpeg.org/, Accessed: 16-Jan-2019.

[54] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications, p. 5.

[55] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: a 4G LTE dataset with channel and context metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference on - MMSys '18*, pp. 460–465, Amsterdam, Netherlands, 2018.