Adaptive streaming algorithm based on reinforcement learning

Suliu Feng1,a, Caihong Wang1,b and Xiuhua Jiang1,*
1School of Information and Communication Engineering, Communication University of China, Beijing, China

*Corresponding author e-mail: jiangxiuhua@cuc.edu.cn, aFengSLcq@163.com, brainbow@cuc.edu.cn

Abstract. As video streaming occupies a large part of the Internet traffic, and DASH gradually becomes the dominant standard for video transmission recent years, a lot of researches about client-side bitrate adaptive algorithms are constantly emerging. This paper mainly adopts the DRL (Deep Reinforcement learning) algorithm that combines Deep Learning and Reinforcement Learning techniques to optimize the Quality of Experience (QoE) of DASH. This method learns to make decisions by observing the network environment and adjust the strategy to get more reward based on feedback, which doesn't rely on pre-programmed models or assumptions about the environment. After experiments on the real network trace datasets, we can find that the algorithm this paper adopted can obtain higher quality of user experience than the existing researches. Moreover, the experimental results show that less buffer underflow occurs and smoother video playback is guaranteed.

1. Introduction

Over the past years, video streaming has accounted for more than half of Internet traffic [1] with the diversity of video viewing devices. As a result, content providers and users are increasingly demanding high QoE (Quality of Experience) of video streaming services in a variable network environment. In this context, adaptive streaming media transmission that dynamically adjusts the request bitrate to maximize QoE in response to the network environment has emerged. Nowadays, Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [2] standard has become the research focus due to its advantages of wide coverage, good compatibility, and simple deployment. In DASH, video stream is sliced into fixed-length segments, each of which stores multiple bitrates. The client selects the optimal bitrate for the next request segment based on current network conditions and playback information using an adaptive bitrate (ABR) decision-making algorithm.

DASH standard does not specify the client-ABR algorithm, so there is plenty of research space. The overall goal of ABR algorithm is: 1) to avoid playback interruption caused by buffer overflow, i.e. rebuffering; 2) maximize the quality of video; 3) minimize video quality switching times and amplitude to ensure smooth video playback. To achieve optimal ABR algorithm has the following challenges: 1) it is difficult to achieve accurate throughput prediction in dynamic network environment; 2) ABR algorithm must balance various QoE indicators, but these indicators have inherent conflicts, such as high bit rate and heavy buffer; 3) the current bit rate decision will have cascade effects on the subsequent decisions; 4) the available decision bitrate is coarse-grained and limited to the available code rate of the video. Therefore, the research on ABR algorithm has been continuously improving.
The client-side ABR algorithm can be roughly divided into four categories: throughput-based, buffer-based, hybrid/control-based, and Markov Decision Process (MDP)-based.

Liu et al. [3] proposed a rate adaptive method that uses smooth HTTP/TCP throughput measurements as well as TFRC/tcp-like AIMD (conservative progressive upward switching and active downward switching of quality). Jiang et al. [4] proposed an improvement of rate adaptive algorithm based on three evaluation indicators of multiple commercial DASH pliers sharing bottleneck links: fairness, efficiency, and stability. The harmonic mean value of the past throughput is used as the current bandwidth estimation. Besides, stateful and delayed-update rate-selection strategy and block scheduling with random download are adopted. Huang et al. [5] proposed a pure buffer-based bitrate adaptive algorithm, and proposed a QoE model based on rebuffering, quality switching and segment quality. The proposed QoE model is expressed as non-linear stochastic optimal control problem. They designed a dynamic buffer-based PID controller which can determine the bitrate of each segment and stabilize the buffer level. Spiteri et al. [6] proposed the Buffer Occupancy Based Lyapunov Algorithm (BOLA), which is a pure buffer-based algorithm without any network bandwidth estimation. BOLA expresses the bitrate adaption as the utility maximization problem, and uses Lyapunov optimization technique to solve the optimal strategy. BOLA is now part of the experimental algorithm in dash.js.

Yin et al. [7] proposed an algorithm based on Model Predictive Control (MPC), which can optimally combine throughput and buffer occupancy feedback signals. They adopt a look-ahead horizon $h = 5$ with throughput predictions using harmonic mean of past 5 chunks and employ existing techniques like CPLEX to solve QoE optimization. Moreover, to counteract bandwidth predict error, they also proposed robust-MPC [8], whose controller is equivalent to the regular MPC taking the lower bound of throughput as input.

In Markov Decision Process (MDP)-based adaptation, the video streaming process is formulated as a finite MDP to realizing making adaptive decisions under fluctuating network conditions. Claesys[9] proposed a control algorithm based on Q-learning algorithm. The client learns the dynamic characteristics of network environment through experience, selects the current optimal action according to the reward function in each state, and updates the action value Q-table. The QoE in this way is increased by 9.7% compared to the existing heuristic algorithm. Then Claesys [10] reduced the state space to two variables: the buffer level and the available bandwidth, and simplified reward. V. Martin [11,12] also proposed a control model based on Q-learning. The improvement mainly lies in the adjustment of the reward function. But Q-learning has two limitations: the state space must be discrete and the dimensionality disaster. So Deep Q-learning (DQN) which uses neural network to approximate value function appeared. I. Kim et al. [13] adopted double-DQN, which main network outputs approximate Q value of different actions and target network outputs target Q value. D-DASH [14] estimates SSIM from the segment size instead of bitrate as quality measurement, referring to A. Testolin's work [15]. Moreover, it uses mixed learning architectures including feedforward and recurrent deep neural networks with advanced strategies. Pensieve [16] employs A3C algorithm which combines policy gradient and value function based method. Compared with current ABR algorithm, the average QoE of the proposed algorithm is increased by 12%–25%.

This paper also adopts deep Reinforcement Learning which learns from past experience by trial-and-error, and gradually converges to the optimal policy which can get maximum return. Proximal Policy Optimization (PPO) [17] which combines value iteration and policy gradient algorithm is used to deal with the bitrate decision problem.

2. Related work

2.1. Dynamic Adaptive Streaming over HTTP

The MPEG organization has developed the MPEG-DASH standard for the HTTP dynamic adaptive streaming transmission system. In the DASH system, the server includes media content segments and Media Presentation Description (MPD). Each media content segment contains media content of a certain duration which each segment corresponds to different quality levels (resolution, code rate, frame rate, etc.). All media content segments are packaged in a format suitable for HTTP streaming. The MPD describes the organization of all media content segments and their respective quality levels,
while using the HTTP URL to identify the download address of the media content segments. The client can realize adaptive streaming media transmission with the information obtained by parsing MPD files.

The adaptive streaming media transmission means that the client can select media content of different quality levels for playing according to its own hardware resources and network bandwidth state. Compared with the traditional HTTP streaming media transmission system based on RTMP protocol or fixed quality level, the framework of DASH system brings intelligent management of resource requests and smooth playback of media content to the client, which ultimately leads to better quality of user experience.

2.2. Quality of Experience

QoE is a service evaluation method based on user approval. Compared with the traditional quality of service (QoS), QoE pays more attention to the subjective experience of users on the services they use in a certain objective environment, which is an important factor influencing the smooth promotion of network multimedia services.

Based on QoE indicator recommended by MPEG-DASH protocol, the factors influencing the experience quality of adaptive streaming transmission system are mainly divided into three categories: fragmented media quality, perceived experience quality fluctuation caused by media quality level switching, and perceived experience quality decline caused by rebuffing event on account of overflow of client buffer.

This paper mainly adopts the general QoE metric used by MPC, which is defined as follows:

\[
\text{QoE} = \sum R_n - \mu \sum T_n - \sum | R_{n+1} - R_n |
\]

(1)

In (1), the first part is the decision bitrate of the nth chunk, second is the penalty term of rebuffing time arises from downloading the nth chunk at bitrate Rn, while the third part penalizes changes in video quality to facilitate smoothing.

2.3. Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a new Actor-Critic (AC) framework based algorithm launched by openAI in 2017. The reason why this algorithm is chosen in this paper is that PPO combines the advantages of policy gradient algorithm and value iteration, but the previous algorithm is difficult to determine the learning step. If the difference between the old and new strategy in the training process is too large, it is equivalent to using a large learning rate, which is not conducive to learning. For this issue, PPO proposes a new objective function. After theoretical deduction, as long as the strategy parameters are updated in the direction of increasing the objective function, it can ensure that the expected return of the strategy is monotonically increasing, which solves the problem of the difficulty in determining the step size in the policy gradient algorithm.

The loss function of the original AC algorithm's policy network is shown in (2):

\[
J(\theta) = \sum_{(s,a)} [p_\theta(s,a)A^\pi(s,a)]
\]

(2)

In (2), \(p_\theta(s,a)\) represents the probability of taking action a under state s, \(A^\pi(s,a)\) is the advantage function which means how much the value obtained by taking action a in state s is better than the average value. The goal of the loss function is to increase the probability of an action that achieves a large return.

Besides, PPO adopts importance sampling to process action distribution, and uses the samples generated by the old strategy to update the strategy parameters. So the loss function need to multiply the importance sampling coefficient.

In order to ensure that the expected return of policy is monotonically increasing, PPO restricts the difference between the old and new policy. PPO has two implementations. The first is ppo-penalty whose objective function is shown in (3), which adds KL divergence penalty term between two policies.
\[
J'(\theta) = E_{(s_t, a_t) \sim \rho_t}[P(a_t | s_t) A(\theta)(s_t, a_t) - KL(\theta, \theta')]
\]

In (3), \(\beta\) varies with KL divergence. If it is larger than target value, the penalty coefficient is increased such that the parameter is updated toward decreasing the KL divergence, and vice versa.

The second is PPO-clip, which directly clipped the first part of (3) to the preset range \((1-\varepsilon, 1+\varepsilon)\). The loss function is shown in (4).

\[
J_{\text{clip}}'(\theta) = \min_{\rho} \left\{ P(q_t | s_t), \text{clip}(\frac{P(q_t | s_t)}{\rho(q_t | s_t)(1 - \varepsilon, 1 + \varepsilon)}), d'(s_t, a_t) \right\}
\]

For purpose of encouraging exploration, probability entropy was added into the objective function, and the entropy weight decreased with the increase of training time.

In this paper, ppo-penalty and ppo-clip are both realized.

3. Adaptive bitrate algorithm based on deep RL

3.1. Adaptive bitrate algorithm implementation

3.1.1. Reinforcement learning elements. The state metrics including the throughput measurements of the past k chunks; the download time of the past k chunks which represents the interval of the throughput measurement; the available size of next chunk which corresponds to different bitrates; the current buffer level; the number of chunks that have not been downloaded; the previous decision bitrate.

The reward function refers to the linear QoE model commonly used in ABR algorithm, which including three parts: segment media quality, perceived experience quality fluctuation caused by media quality level switching, and perceived experience quality decline caused by rebuffering event on account of overflow of client buffer. In terms of quality switch, first it's need to determine whether the current buffer is less than a preset minimum threshold. If so, encourage the behavior of reducing the decision rate, and penalize the behavior of increasing the code rate, and vice versa.

\[
\text{reward} = \begin{cases} 
R_k - \alpha \cdot T_k - \beta \cdot (R_{k-1} - R_k), & \text{curbuf} > \text{min buf} \\
R_k - \alpha \cdot T_k - \beta \cdot (R_{k-1} - R_k), & \text{curbuf} < \text{min buf}
\end{cases}
\]

In (5), \(R_k\) represents the bitrate of kth chunk, \(\alpha\) and \(\beta\) respectively represent the penalty coefficients of rebuffering and quality switch. \(T_k\) is the duration from buffer underflow until the kth chunk is downloaded completely.

3.1.2. Network framework. This paper uses PPO combined with deep neural network including actor network and critic network. The network framework diagram is shown in Fig.1.

![Figure 1. The actor-critic network framework diagram.](image)

As shown in Fig.1, the input layer and hide layer settings are the same for the Actor network and the Critic network. The input layer has six state characteristics with three sequences, including throughput, sharding download completion time, and the available size list of current segment. These three
sequences pass through a one-dimensional convolution layer, the size of which is 4, and the number of filters is set as 128. Other single-value features of the input state are input a layer full connection layer which the number of hidden layer neurons is 128. All outputs of the hidden layer are combined to input a fully-connected layer and finally connected to the output layer. The output layer of the Actor network is a softmax layer, the number of neurons is the number of available bitrate, corresponding to probability distributions of different actions under input state. The output layer of the Critic network is a fully connected layer, and has one neuron, which represents the state value $V(s)$ of the input state $s$.

3.1.3. Algorithm training. In this paper, parallel training is adopted to train the algorithm, that is, several different training agents are used to collect samples. Parallel training can improve the training speed, and the sample distribution is more uniform which is more conducive to network training. Compared with the value-based method which uses experience pool to store a large number of history samples, parallel training can save the storage space greatly. In parallel training, each agent makes continuous decisions in its own environment and then obtains the decision trace sample $(s, a, r, s_{\text{next}})$. When each agent accumulates a certain amount of samples, they send the accumulated samples to the central agent, where updates the network parameters and then copies these parameters to the agent network.

This paper uses two parallel training methods, one is synchronous parallel, that is, when all agents have collected enough samples, the central agent will calculate the gradients to update the network parameters and distribute the parameters to each agent. The other is asynchronous parallel, that is, as long as one agent collects enough samples, the agent's samples are sent to the central agent to update its network parameters.

4. Experimental results

In this work, the test video chooses the "Envivio-Dash3" sequence of the DASH-246 JavaScript reference client. This video is decoded by the H.264 / MPEG-4 codec at bitrates $\{300, 750, 1200, 1850, 2850, 4300\}$. It has a total length of 193 seconds and is divided into 48 segments with 4 seconds per segment. The training and testing network traces adopts the 3G / HSDPA mobile dataset collected by Norway which was collected by mobile devices during the transmission process (such as by bus, train, etc.). The training set contains 127 trace files, which each file contains 300-1500 throughput measurements; the test set contains 147 trace files and each file has only 50-250 Throughput measurements. The average throughput of these trace files ranges from 0.2 to 6Mbps.

We carried out simulations under linux system, based on pycharm IDE and tensorflow software library. The experiments realized the aforementioned PPO-clip based ABR algorithm and synchronous parallel training. The mean total reward of the algorithm and the existing algorithm on the test trace set is shown in Table 1.

| Algorithm   | Mean total return |
|-------------|-------------------|
| Buffer_based| 30.04             |
| Fast_mpc    | 41.17             |
| Robust_mpc  | 42.54             |
| A3C         | 44.18             |
| PPO_sync    | 49.89             |

It can be seen from Table 1 that the algorithm of this work under the same test traces gets a higher mean total reward than the existing algorithm. Then draw the cumulative distribution curve of total reward of different algorithms on the test traces, as shown in Fig.2.
Figure 2. The cumulative distribution curve of total reward of different algorithms on test traces.

It can be seen that within the same interval range of the horizontal axis, the proportion of the algorithm this work proposed in the vertical axis is large. It indicates that PPO algorithm falls within the range of higher reward. Then select a trace at random and output the bitrate selection, buffer level and bandwidth change curve of each algorithm under the chosen network environment, as shown in Fig. 3.

Figure 3. Bitrate decision, buffer level and bandwidth curve of each algorithm change with time under the trace of car_10.

From Fig.3, the PPO algorithm can ensure the selection of a higher and stable bit rate, a lower rate switching frequency. Moreover, the buffer can be fully utilized. Even when the throughput is low, if the buffer is sufficient, a higher bit rate will still be selected, and when the buffer level is lower than the minimum threshold, the agent can decrease the decision bitrate immediately.

In order to further explore the influence of algorithm selection and training methods on experimental results, ppo-pen, ppo-clip, and synchronous and asynchronous parallel combination training were selected. The mean total reward on the test tracesets under the four settings is shown in Table 2.

| Training method                      | Mean total return |
|--------------------------------------|-------------------|
| ppo-clip, synchronous parallel training | 46.79             |
| ppo-clip, asynchronous parallel training | 40.97             |
| ppo-pen, synchronous parallel training | 45.57             |
| ppo-pen, asynchronous parallel training | 39.09             |
From the point of ppo algorithm, it shows that the total reward of ppo-clip is slightly higher than that of ppo-pen. From the way of parallel training, the total reward of synchronous parallel is higher than that of asynchronous parallel.

Analyzing it from ppo algorithm, it can be seen from (3) and (4) that ppo-clip directly restricts the ratio of the old and new policies to a certain extent, which can ensure that the distribution gap between the two updates is not too high.

From a perspective of parallel training method, asynchronous parallel could lead to the expiration gradient problem, that is, the gradient passed from the agent when updating the current network parameters may be the previous parameter, which will make the process of gradient descent become unstable. In the experiment, we find the parameters the agent uses fall behind the current central agent version by 10-16, which fully reveals the problem of expiration gradient. For this problem, some studies have proposed a solution that add delayed compensation to the gradient and use the Taylor expansion of the gradient function to effectively approximate the Hessian matrix of the loss function.

This work also conducted an analysis of the influence of history throughput sample size on the prediction of the current network environment. The throughput measurements vector length of the input state was set as 1, 4, 8, 16 respectively. Besides, ppo-clip and synchronous parallel training were adopted. Finally, output the mean total reward of these four settings on the test traces, as shown in Table 3.

| Length of throughput measurements vector | Mean total reward |
|-----------------------------------------|-------------------|
| 1                                       | 29.29             |
| 4                                       | 38.99             |
| 8                                       | 46.79             |
| 16                                      | 46.09             |

It can be seen that when the time series length increases from 1 to 4 and from 4 to 8, the mean value of total return increased greatly, but when the length increases from 8 to 16, the value doesn’t increase but slightly decreases.

The results show that increasing the amount of historical throughput to a certain extent can improve the rate decision performance. That's might because a certain historical throughput information is helpful to predict the current bandwidth more accurately. But when the amount of historical throughput increases to a certain point, the performance is no longer improved. This is mainly because the impact of current action on future expected return is gradually declining, the longer the previous information has the less impact on the present.

5. Conclusions

In this paper, we applied DRL algorithm combining reinforcement learning with deep neural network to DASH client-based adaptive bitrate algorithm. After testing on the real network traces datasets, we can find that the algorithm this paper adopted can obtain higher quality of user experience than the existing research and also have few rebuffering events and quality switching.

References

[1] “Cisco Visual Networking Index: Global mobile data traffic forecast update, 20152020 white paper,” Tech. Rep., 2015.
[2] “ISO/IEC 23009-1:2014: Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats,” International Organization for Standardization, Standard, May 2014.
[3] C. Liu, I. Bouazizi, and M. Gabbouj, “Rate adaptation for adaptive HTTP streaming,” in Proc. 2nd Annu. ACM Conf. Multimedia Syst. (MMSys), San Jose, CA, USA, 2011, pp. 169–174.
[4] J. Jiang, V. Sekar, and H. Zhang, “Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with FESTIVE,” in IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 22, NO. 1, FEBRUARY 2014

[5] 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 ACM SIGCOMM, Chicago, IL, USA, 2014, pp. 187–198.

[6] 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, Apr. 2016, pp. 1–9.

[7] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A controltheoretic approach for dynamic adaptive video streaming over HTTP,” in Proc. ACM Conf. Special Interest Group Data Commun. (SIGCOMM), London, U.K., 2015, pp. 325–338.

[8] X. Yin, V. Sekar, and B. Sinopoli, “Toward a principled framework to design dynamic adaptive streaming algorithms over HTTP,” in Proc. 13th ACM Workshop Hot Topics Netw. (HotNets-XIII), Los Angeles, CA, USA, 2014, pp. 1–7. [Online]. Available: https://doi.org/10.1145/2670518.2673877

[9] M. Claeys, S. Latr’e, J. Famaey, T. Wu, W. Van Leekwijk, and F. De Turck, “Design of a Q-learning-based client quality selection algorithm for HTTP adaptive video streaming,” in Adaptive and Learning Agents Workshop (ALA-2013), Saint Paul, Minnesota, USA, May 2013.

[10] M. Claeys, S. Latr’e, J. Famaey, T. Wu, W. Van Leekwijk, 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.

[11] Virginia Martín, Julián Cabrera, and Narciso García, “Q-learning based control algorithm for HTTP adaptive streaming,” in International Conference on Visual Communications and Image Processing (VCIP), Singapore, Dec 2015.

[12] Virginia Martín, Julián Cabrera, and Narciso García, “Design, optimization and evaluation of a Q-Learning HTTP Adaptive Streaming Client,” IEEE Transactions on Consumer Electronics, Vol. 62, No. 4, November 2016.

[13] Iseul Kim, Seongjun Hong, Sungwook Jung, Kyungshik Lim, “A Quality Selection Mechanism Using a Deep Q-Network for Seamless Video Streaming Services,” International Conference on Ubiquitous and Future Networks, Prague, Czech Republic, July 2018.

[14] Matteo Gadaleta, Federico Chiariotti, Michele Rossi, Andrea Zanella, “D-DASH: a Deep Q-learning Framework for DASH Video Streaming”, IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING, September, 2017.

[15] A. Testolin, M. Zanforlin, M. D. F. D. Grazia, D. Munaretto, A. Zanella, M. Zorzi, and M. Zorzi, “A machine learning approach to QoE-based video admission control and resource allocation in wireless systems,” in 13th Annual Mediterranean Ad Hoc Networking Workshop (MEDHOC-NET), Los Angeles, CA, USA, June 2014.

[16] Hongzi Mao, Ravi Netravali, Mohammad Alizadeh MIT Computer Science and Artificial Intelligence Laboratory, 2017. Neural Adaptive Video Streaming with Pensieve. In Proceedings of SIGCOMM ‘17, August 21-25, 2017, Los Angeles, CA, USA, 14 pags.

[17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov, OpenAI, Aug 2017.