BitLat: Bitrate-adaptivity and Latency-awareness Algorithm for Live Video Streaming

Chen Wang, Jianfeng Guan, Tongtong Feng, Neng Zhang, Tengfei Cao
State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China
bupt.wc@gmail.com;{jfguan;ftt;zn;caotf}@bupt.edu.cn

ABSTRACT

With the growing popularity and prosperity of living streaming applications, it is naturally confronting users' quality of experience (QoE) degradation issues especially under dynamic environments arised from nonnegligible factors such as high latency and intermittent bitrate. In this paper, we propose an efficient adaptive bitrate (ABR) algorithm called BitLat to achieve both bitrate-control and latency-control. BitLat is based on reinforcement learning to get strong adaptability for dealing with the complex and changing network conditions. More specifically, in our work, we determine the specific value of latency threshold with the help of current advanced algorithm, and design the structure of the neural network in reinforcement learning, the features used in the training process, and the corresponding reward function. Additional, we use the *Dynamic Reward Method* to further enhance the performance. Comprehensive experiments are conducted to demonstrate BitLat outperforms the state-of-the-art ABR algorithms, with improvements in average QoE of 20%-62%.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**; • **Computing methodologies** → *Reinforcement learning*; • **Computer systems organization** → *Real-time system architecture*.

KEYWORDS

Adaptive Bitrate Algorithm, Low Latency, Live Video Streaming, Reinforcement Learning

ACM Reference Format:
Chen Wang, Jianfeng Guan, Tongtong Feng, Neng Zhang, Tengfei Cao. 2019. BitLat: Bitrate-adaptivity and Latency-awareness Algorithm for Live Video Streaming. In *Proceedings of the 27th ACM International Conference on Multimedia (MM'19), Oct. 21–25, 2019, Nice, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3343031.3356069

MM '19, October 21–25, 2019, Nice, France

ACM ISBN 978-1-4503-6889-6/19/10…$15.00
https://doi.org/10.1145/3343031.3356069

1 INTRODUCTION

Recent years have witnessed a rapid increase in the live streaming applications[3], video provider (e.g., YouTube, Twitch, Vigo, Douyu), however, are troubled by how to improve users quality of experience (QoE) in live scenes, since there are few people consider ultra-low latency[14] as an important factor when they designed the ABR algorithm. In general, current outstanding ABR algorithms can be divided into two categories, rule-based and learning-based.

For the rule-based approaches, some schemes (BBA[4], BOLA[9]) rely on the buffer size when choosing next bitrate, since the core idea primarily lies in that the larger the current buffer size is, the higher bitrate tends to be selected. While other schemes (FESTIVE[5], CS2P[10]) focus on the network bandwidth to determine next bitrate, that is, the higher the current available bandwidth is, the higher bitrate tends to be selected. Besides, the rest methods such as MPC[13] both considering buffer size and available bandwidth by presenting the formula of QoE will always choose the bitrate with higher QoE.

For the learning-based approaches, Pensieve[6] is the most representative algorithm among them, which utilizes the adaptiveness of reinforcement learning[7, 12] to improve the robustness of ABR algorithm. It describes the basic training algorithm and presents the extensions that allow a single neural network model to be generalized to the videos with different properties. Moreover, another creative ABR algorithm is QARC[2] which focuses on high video quality instead of high video bitrate. For example, a low bitrate may also provide a barely satisfactory perceptual video quality when the video footage consists of darkness and few objects. Therefore, it aims to find an artful balance between video quality and video bitrate. In short, most existing ABR algorithms are facing the following problems:

i) **Adaptation-Limited problem.** Rule-based ABR algorithms usually build fixed control rules based on simple models which will be poor-performed in complex network conditions of real world.

ii) **Latency-Insensitive problem.** Learning-based algorithms choose next bitrate with caution for rebuffering event and bitrate switching event, but ignore the latency between server and client, which causes great damage to the strong interactivity under the live scenes, and reduce the users' QoE.

In this paper, we propose BitLat, a Bitrate-adaptivity and Latency-awareness ABR algorithm, to achieve both high bitrate and low latency. The main contributions of our work are summarized as follows:

- We propose BitLat based on reinforcement learning to get a latency-awareness ABR algorithm which is highly adaptable to various network environments.
- We take latency as the key part of the algorithm, and put forward a Dynamic Reward Method (DRM) to further improve the performance.

2 BITLAT ARCHITECTURE

2.1 PREPARATORY WORK

Figure 1 shows the basic framework of reinforcement learning. In the learning process, s_t represents the current state of the environment, a_t represents the action we choose according to s_t, which will have an impact on the environment and gain a new environment state s_{t+1}. Moreover, a reward r_{t+1} will be generated to criticize the quality of the action a_t, and will influence the next action.

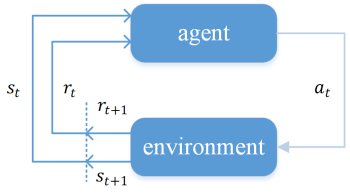

Figure 1: Reinforcement Learning Framework

However, some problems arise when applying this framework to this challenge. In this challenge[11], the designed ABR algorithm should return three parameters, which are *bitrate*, *target buffer*, *latency limit*. The *bitrate* means the next bitrate we will choose, *target buffer* controls the threshold value of fast and slow playback. *Latency limit* represents the threshold of frame skipping, e.g. frame skipping event will happen when the latency between server and client is larger than *latency limit*. Therefore, the action(a_t) here should be a triple, $a_t = (bitrate, targetbuffer, latencylimit)$. However, the value of *latency limit* is not yet specified. To clarify the specific actions in the learning process, it is necessary to set specific value for it. Moreover, in order to avoid overfitting for some special network and gain a reliable result, complex, large-scale and diverse network traces are required.

Large-scale Network Traces: In order to adapt to the complex and varied network scenarios in the real environment, it is obviously unreasonable to simply use a certain type of network (that is, strong, medium, and weak network traces provided by the challenge). We use these network data sets to randomly construct a large number of mixed networks. We cut each network set into 20 segments of the same length, and then randomly select each of these segments to generate complex data sets.

Latency Limit: With respect to determining latency limit, we turn to advanced ABR algorithms to find a solution. However, most of the classic algorithms have not yet considered the latency problem, we make some improvements, that is, making the latency limit be a part of return results. We sample the value of latency limit at intervals of 0.1 from 0.1 to 4.0, to find the approximate range of excellent value of latency limit. As Figure 2 shows, we can observe that for most of state-of-art ABR algorithms, the excellent value of latency limit should be around 1.6 to 2.0. Therefore, we choose them as the choices in the learning process, and now we specified the action triple.

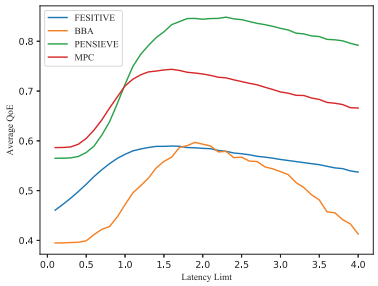

Figure 2: Advanced ABR algorithms effects for different latency limit

2.2 FRAMEWORK OVERVIEW

The process of our work can be described in Figure 3. First, we construct various network traces based on the traces data given in the challenge, and determine the latency limit with the help of current advanced algorithms. Then in the learning phase, we design the structure of neural network, the feature parameters, and the details of reward function, and we also use the constructed traces to training our model to get strong adaptability. Lastly, we propose the **Dynamic Reward Method** (DRM) to further enhance performance.

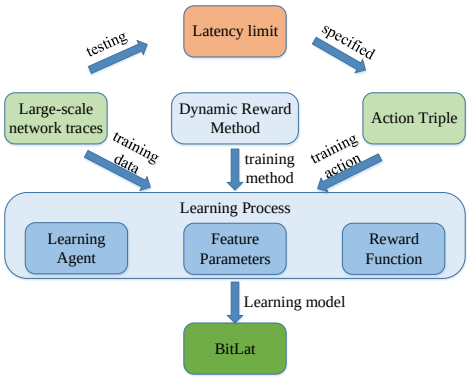

Figure 3: BitLat process overview

3 METHODOLOGIES

3.1 LEARNING AGENT

We choose A2C[8] as our reinforcement learning network model, which achieves the maximum cumulative return through the interaction between agent and environment. The A2C

model has a better scoring function than the DQN model based on strategy gradient. Different from the traditional reinforcement learning method, A2C updates parameters at each step. The gradient iteration of reinforcement learning is realized by the long-term cumulative return of the value function. Our neural network structure consists of 2 hidden layers, and each layer contains 128 units, which can balance the training speed and training effect.

3.2 FEATURE PARAMETERS

Inspired by the classic bitrate adaption theory which are designed from two aspects, i.e. buffer size and available bandwidth, we further present 16 features for bitrate selection. That is, the state s_t of the learning process, $s_t = (B_t, G_t, F_t, D_t, N_t, C_t^1, C_t^2, C_t^3, C_t^4, W_t, M_t, P_t, L_t, K_t, R_t, Q_t)$. There features are selected from three aspects.

In term of buffer size, B_t means the current buffer size (seconds). For example, the size of 10 seconds means that the content in the current buffer can still play for 10s. G_t means the current target buffer, which controls the threshold of fast and slow playback. Fast (slow) playback means that the content of 1s is actually played for 0.95 (1.05)s respectively. F_t means the current buffer status, which represents whether the video is playing or not. D_t means the difference between the threshold of fast playback and the current buffer size. N_t means the number of frames in the CDN, which ensures that we can get new frame at the next time. C_t^1, C_t^2, C_t^3, C_t^4 represents the predicted video chunk sizes (four different bitrates) which will be downloaded at the next time.

In term of available bandwidth, W_t is the available bandwidth when downloading the last frame. M_t represents the average bandwidth when downloading latest four frames. P_t means the probability of bandwidth increment at the next time.

Moreover, we also choose some features according to the calculation of QoE. L_t represents the current latency between server and client. K_t represents the total skip time during the last chunk. R_t represents the total rebuffering time during the last chunk. Q_t represents the last bitrate we chose.

3.3 REWARD FUNCTION

Since the bitrate switching can only happen at the I-frame[1, 13], the selected bitrate will affect frames between two consecutive I-frames, and these affected frames can be viewed as a chunk. We will calculate and take the QoE of all frames in one chunk as our reward. In this challenge, the specific formula of QoE calculation is as follows:

$$QoE = \sum_{n=1}^{N} T_n * Q_n - \alpha * R_n - \beta * L_n - \gamma * S_n - \mu * K_n \quad (1)$$

Where N is the total number of frames, Q_n means the chosen bitrate of frame n, T_n means the playback time of frame n, R_n represents the total rebuffering time when downloading frame n, L_n means the current latency, S_n indicates the number of switching bitrate generated when playing this frame, and K_t represents the total skipping time when downloading frame n. α, β, γ, μ indicate the weights of these factors for QoE. Specially, we called these coefficient of parameter which causes negative reward as penalty coefficient. The initial values are $\alpha = 1.85$, $\beta = 0.005$, $\gamma = 0.02$, $\mu = 0.5$.

3.4 DYNAMIC REWARD METHOD

According to different QoE calculation methods, there is an obvious conclusion that the algorithm for bitrate selection will consider different preference. More specifically, if we increase the penalty coefficient of latency significantly, the algorithm will be more inclined to avoid excessive latency when choosing next bitrate. Based on this idea, we make some improvements to the reward function, and introduce *Dynamic Reward Method*(DRM).

During the whole training process, we record the penalty in the past period which is caused by rebuffering (P^{reb}) , bitrate switching (P^{sw}), latency (P^{lat}), frame skipping (P^{sk}), respectively. For simplicity, we define set $Cate$ that contains the all penalty categories(rebuffering, switching, latency, skip), and $p \in Cate$. Once the ratio of any penalty value to the total penalties exceeds a certain threshold (T_p^R), the corresponding penalty coefficient will increase slightly. **In the implementation stage, it is need to pay attention to the following three aspects of the details.**

First, in order to avoid the occurrence of extreme conditions (such as excessive rebuffering penalty caused by being in a weak network for a long time, makes the coefficient of rebuffering penalty out of balance), we just calculate the data for several frames in the past. Therefore, for a certain penalty, its total value can be expressed as follows:

$$S(P^p, K) = \sum_{i=1}^{K} P_i^p \quad (2)$$

Where K means the number of past frames, P^p records the detailed penalty data of past K frames. Furthermore, the total penalty of all factors can be expressed as:

$$T(K) = \sum_{p}^{p \in Cate} S(P^p, K) \quad (3)$$

Second, excellent strategy can eliminate the events of rebuffering and bitrate switching, but cannot erase the penalty of latency. Therefore, it is high likely that the proportion becomes too large while the true value is small. To tackle this, we set a threshold (T_p^V), and apply our strategy when the total penalty value is greater than the threshold.

Third, the proportion of penalty value affects the updating of penalty coefficient. More specifically, the larger the proportion is, the greater change will be made to coefficient. For the ratio of penalty p (referred to as R_p), we have:

$$R_p = \frac{S(P^p, K)}{T(K)} \quad (4)$$

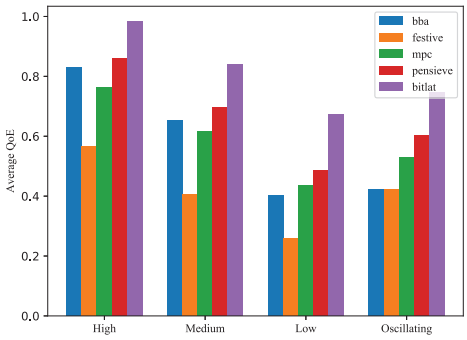

Figure 4: Comparison of QoE performance between BitLat and outstanding ABR algorithms in different network datasets

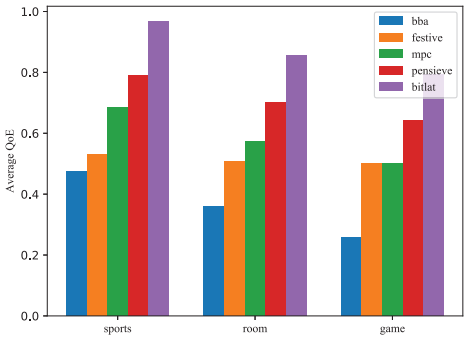

Figure 5: Comparison of QoE performance between BitLat and outstanding ABR algorithms in different video datasets

Table 1: Detailed Performance of DRM in large-scale network

	bitrate utility	rebuffer penalty	switching penalty	latency penalty	skipping penalty	total
no DRM	71170.31	8511.13	124.65	9950.98	1020.78	51562.77
DRM	70675.91	5631.42	161.82	8203.07	604.47	56075.19
GAIN	-0.69%	33.83%	-29.81%	17.56%	40.78%	8.75%

Then we set some constant values (C_p) for each penalty factor, which are used to update corresponding coefficient. For Coe_p, the update method is as follows:

$$U(Coe_p) = \begin{cases} Coe_p & \text{if } R_p <= T_p^r \\ Coe_p + \frac{R_p - T_p^r}{1 - T_p^r} * C_p & \text{else} \end{cases} \quad (5)$$

In conclusion, the overall algorithmic process of DRM is shown as follows:

Algorithm 1: Training process with DRM

```
for n = 1 to N do
    /* calculate state information for each state */;
    /* get P^reb, P^sw, P^lat, P^sk */;
    Calculate single penalty value via Eq. 2;
    Calculate total penalty value via Eq. 3;
    for all p such that p ∈ (reb, sw, lat, sk) do
        Calculate the ratio R_p via Eq. 4;
        if R_p ≥ T_p^V then
            Update Coe_p via Eq. 5
        end if
    end for
end for
```

4 EXPERIMENTS

In this stage, we demonstrate BitLat outperforms the state-of-the-art ABR algorithms, and show the benefits of DRM.

Firstly, in order to verify the strong adaptability of BitLat, we put it under different network (video) data sets and compare it with current classical ABR algorithms. It is worth mentioned that since the latency is not considered in these algorithms, we will set a fixed value for them in the experiment and guarantee that it can be viewed as an optimal value. The results can be seen in Figure 4 and Figure 5. Experiments show that BitLat improves the user QoE by 20%-62%.

Then, we conducted a detailed tests of the DRM in terms of bitrate utility, rebuffering penalty, smoothness penalty, latency penalty, and frame skipping penalty. The detailed results can be seen in Table 1. It has been proved by experiments that the penalty of rebuffering and latency accounts for the vast majority of all penalties. While the penalty of bitrate switching is very small in any cases which is consistent with the fact that the switching penalty coefficient does not change substantially during the experiments. In a large number of experimental results, DRM has improved the effect by 5% - 10%.

5 CONCLUSION

In this paper, we introduce our method for the Live Video Streaming Challenge. We design the BitLat based on reinforcement learning to overcome the Adaptation-Limited problem brought by complex and variable networks. We also consider latency as the key point of our algorithm to solve the Latency-Insensitive problem. Additionally, we propose DRM to further improve the effect of BitLat. Experiments demonstrate BitLat outperforms the state-of-the-art ABR algorithms.

ACKNOWLEDGMENT

This work was supported by the Natural Science Foundation of China under Grants Nos. 61871048, Nos. 61872253, and in part by the National Basic Research Program of China (973) under Grant 2013CB329102.

REFERENCES

[1] 2017. H.264/MPEG-4 AVC. (2017). http://handle.itu.int/11.1002/1000/6312.

[2] 2018. QARC: Video Quality Aware Rate Control for Real-Time Video Streaming based on Deep Reinforcement Learning. *CoRR* abs/1805.02482 (2018). arXiv:1805.02482 http://arxiv.org/abs/1805.02482 Withdrawn.

[3] Cisco. 2018. Cisco Visual Networking Index: Forecast and Methodology, 2017–2022. *White Paper Cisco Systems Inc* (2018).

[4] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. *SIGCOMM Comput. Commun. Rev.* 44, 4 (Aug. 2014), 187–198. https://doi.org/10.1145/2740070.2626296

[5] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2014. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. *IEEE/ACM Trans. Netw.* 22, 1 (Feb. 2014), 326–340. https://doi.org/10.1109/TNET.2013.2291681

[6] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[7] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. *CoRR* abs/1602.01783 (2016). arXiv:1602.01783 http://arxiv.org/abs/1602.01783

[8] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. *CoRR* abs/1602.01783 (2016). arXiv:1602.01783 http://arxiv.org/abs/1602.01783

[9] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9. https://doi.org/10.1109/INFOCOM.2016.7524428

[10] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16)*. ACM, New York, NY, USA, 272–285. https://doi.org/10.1145/2934872.2934898

[11] Tsinghua University. 2019. Lived Video Streaming, ACM Multimedia 2019 Grand Challenge. (2019). https://www.aitrans.online/MMGC/.

[12] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger B. Grosse, and Jimmy Ba. 2017. Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. *CoRR* abs/1708.05144 (2017). arXiv:1708.05144 http://arxiv.org/abs/1708.05144

[13] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. 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)*. ACM, New York, NY, USA, 325–338. https://doi.org/10.1145/2785956.2787486

[14] Wenxiao Zhang, Bo Han, and Pan Hui. 2018. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. In *Proceedings of the 26th ACM International Conference on Multimedia (MM '18)*. ACM, New York, NY, USA, 355–363. https://doi.org/10.1145/3240508.3240561