Improvement of Online Education Based on A3C Reinforcement Learning Edge Cache

Haichao Wang, Tingting Hou, and Jianji Ren(✉)

Henan Polytechnic University, Jiaozuo 454000, Henan, China
renjianji@hpu.edu.cn

Abstract. Online education is the complement and extension of campus education. Aiming at the situation that the mainstream network speed in online education scenarios is difficult to meet the requirements for smooth video playback, such as ultra-high-definition video resources, live broadcast, etc., this paper proposes an A3C-based online education resource caching mechanism. This mechanism uses A3C reinforcement learning-based edge caching to cache video content, which can meet the requirements of smooth video playback while reducing bandwidth consumption and improving network throughput. We use the Asynchronous Advantage Actor-Critic (A3C) technology of asynchronous advantage actors as a caching agent for network access. The agent can learn based on the content requested by the user and make a cache replacement decision. As the number of content requests increases, the hit rate of the cache agent gradually increases, and the training loss gradually decreases. The experimental comparison of LRU, LFU, and RND shows that this scheme can improve the cache hit rate.

Keywords: Edge cache · Reinforcement learning · Online education

1 Introduction

Online education, as a supplement and extension of traditional classroom education, especially during the COVID-19 epidemic, overcomes the limitations and disadvantages of traditional classroom education and can optimize the allocation of educational resources and improve the quality of teaching to a certain extent. However, with the popularity of online education and mobile Internet access devices, mobile devices (such as smartphones and tablets) are increasingly requesting multimedia services (such as audio, high-definition video, photos, etc.). The content provider's network traffic load has increased rapidly, and the quality of service (QoS) of users has decreased [1].

Online education has the following advantages: small environmental limitations, low teaching costs, and can alleviate the problem of unbalanced teaching resources. However, online education has high requirements for network infrastructure, so the biggest problem it faces is network latency. When users use online education and learning, if the network speed can not keep up, there will be a delay stuck; when transmitting data and lost packets, there will also be a picture and sound "stuck". Therefore, connectivity and networking must become the top priority for online education. However, this is a big challenge for schools with many distributed network resources.

In online education, some users will have more demand for online courses. These users may be located in more concentrated areas (urban primary and secondary schools) or multiple areas (university online education). For users in the distribution area, how to reduce the propagation flow through local short-distance communication and effectively reduce the duplicate backhaul traffic has become a hot research topic.

The emergence of 5G and edge cache can alleviate the dilemma faced by online education, which reduces latency by storing data locally, thereby obtaining a continuous and better user experience. Edge computing can disperse computing resources and move them closer to data sources. When schools use edge computing, they will prioritize connections and networks across multiple campuses to eliminate the slow speed, thereby significantly improving the online education experience for students and teachers. Therefore, it is the general trend to apply edge caching to online education.

Edge caching is a promising technology that can increase network throughput, improve energy efficiency, reduce service latency, and reduce network traffic congestion by caching cache nodes on the edge of a wireless network. Cache nodes can be placed on base stations and home gateways so that the content requested by the user is closer to the user, rather than being downloaded repeatedly from the content service provider over the network.

Therefore, it is very interesting and necessary for some effective prefetching strategies to place educational resources with high-frequency requests on edge cache nodes, thereby reducing resource access latency for end-users. We study the caching and replacement decision problems in BSs, model the content caching of edge nodes and user requests as Markov decision problems, and deploy a learning framework based on A3C for content caching of BSs. Based on the results of simulation experiments, compared with the existing LRU (least recently used), LFU (least frequently used), and RND (random number deletion) hit rates, there is a certain improvement.

The rest of this article is organized as follows: In the second part, we introduce related work; in the third part, we propose problems and algorithms; We conduct simulation experiments and give the experimental results in the fourth part; the last part is our research conclusions and future research challenges.

2 Related Work

In recent years, online education resources have grown rapidly, and the network traffic for educational applications has also increased tremendously, which has placed a heavy burden on the core network, which has ultimately led to long delays in resource access and user experience. Edge caching can help solve this problem by fetc.hing and caching new content.

In [6], a coordinated mobile edge network that supports caching was discussed. This network structure mainly reduces the load on the front-end link, while achieving low-latency and high-rate content delivery.

Recently, researchers have used artificial intelligence (AI) technology combined with edge caching to better understand user behavior and network characteristics. In particular, neural networks placed in edge nodes learn to predict popular content and make caching decisions. For example, in [2], collaborative filtering of stack-type automatic coding, recurrent neural network, deep neural network, and recurrent neural network [3] is proposed to predict the popularity of content.

The researches in [4] and [5] show that reinforcement learning RL has great application potential in the content caching scheme design. The author of [4] proposed a cache replacement strategy based on Q-learning to reduce the traffic load in the network and used a multi-armed robber MAB to place the cache.

Aiming at complex control problems, [7] proposed a collaborative deep cache framework for mobile networks based on dual deep q networks. The framework aims to minimize content acquisition delays and does not require any prior knowledge. [8] proposed a collaborative edge cache architecture based on 5G networks, which uses mobile edge computing resources to enhance edge cache capabilities, and proposes vehicle-assisted edge caches in combination with smart car cache resources. [9] use Q-learning to design the caching mechanism, reduce the backhaul traffic load and transmission delay from the remote cloud, and propose an action selection strategy for the caching problem. Find the right cache state through reinforcement learning.

Although the replay of the experience used by the DQN algorithm can be said to solve the problem of reinforcement learning satisfying the independent and identical distribution, it uses more resources and the calculation of each interaction process, and it needs an off-policy learning algorithm to update the data generated by the old strategy, [10] put forward the concept of "executively executing multiple agents in parallel on multiple environment instances". Its advantage is that unlike traditional methods using GPUs or large-scale distributed computing, It is running on a single multi-core CPU, which is the same task. The asynchronous A3C algorithm has achieved the same or even superior results, and the cost is a shorter training time. Inspired by this, we will use A3C technology in edge caching.

3 Edge Cache Model

In the future 5G network architecture, edge nodes (EN) will become an indispensable part. To meet the low-latency and high-traffic network services in online education scenarios, this study proposes an education resource pre-cache based on the combination of EN Caching and A3C technology.

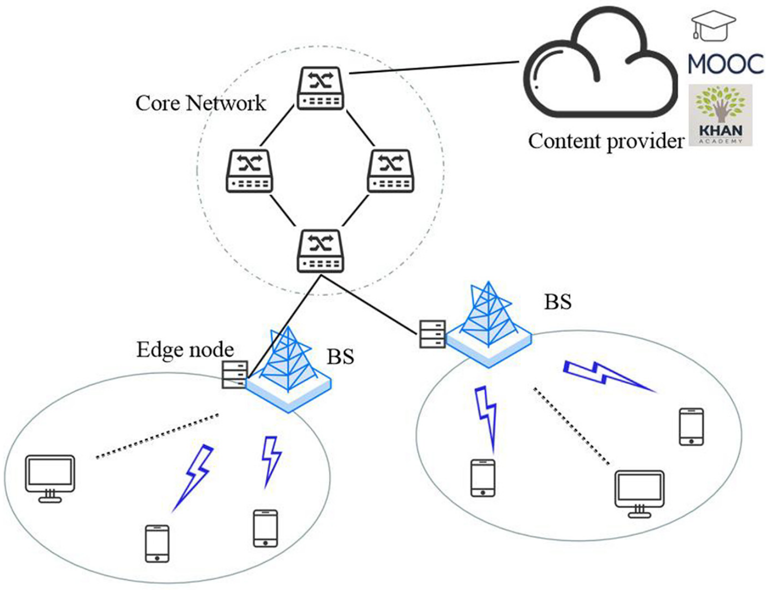

Fig. 1. Edge cache architecture.

3.1 Problem Formulation

The model building is illustrated in Fig. 1. In EN, there is an A3C-based caching agent. Because the agent involves a series of interactions between the agent and the environment during the edge caching process, the edge node caching issues can be modeled as a Markov decision process. The Markov process can be represented by five key elements:

A group of states S that the agent can really be in; a set of actions A performed by the agent when it moves from one state to another; transition probability, the execution of a certain behavior a, and transition from a state s, probability of going to another state s'; reward, the agent performs a certain behavior a, and transitions from one state s to another state s' to get reward probability; discount factor controls the importance of reward and future reward.

Edge caching is to strike a balance between broadband transmission costs and storage costs. Therefore, caching of popular content is very important. In general, the distribution of popular content is represented by the Zipf function distribution, that is, most users often request specific files that account for a small proportion of the requested content, while most files are rarely requested.

The following is the process by which the agent makes the cache decision:

(1) When a user's content request reaches EN, the EN first checks whether there is content requested by the user in the cache, and if so, returns the content directly to the user.

(2) If EN checks its own cache and does not find the content requested by the user, it needs to determine whether the cache is full. If the cache is not full, the EN node downloads the content requested by the user from the content provider to the local cache, and is forwarded to the user through the network.

(3) If the content requested by the user is not cached in the EN cache and the cache is full. The caching agent (based on A3C) will determine the cache content to be removed. At the same time, the agent downloads the content requested by the user from the content provider, transmits it to the user, and caches the content locally.

3.2 Asynchronous Advantage Actor-Critic

The main idea of the A3C algorithm is to learn and integrate all its experience in parallel through multiple agents. A3C can produce better accuracy than other algorithms and has good results in continuous and discrete space. The network uses multiple agents, and each agent learns in parallel with different exploration strategies in a copy of the actual environment. The experience gained by the agents is then integrated to form a global agent. Global agents are also called core networks or global networks, while other agents are called workers.

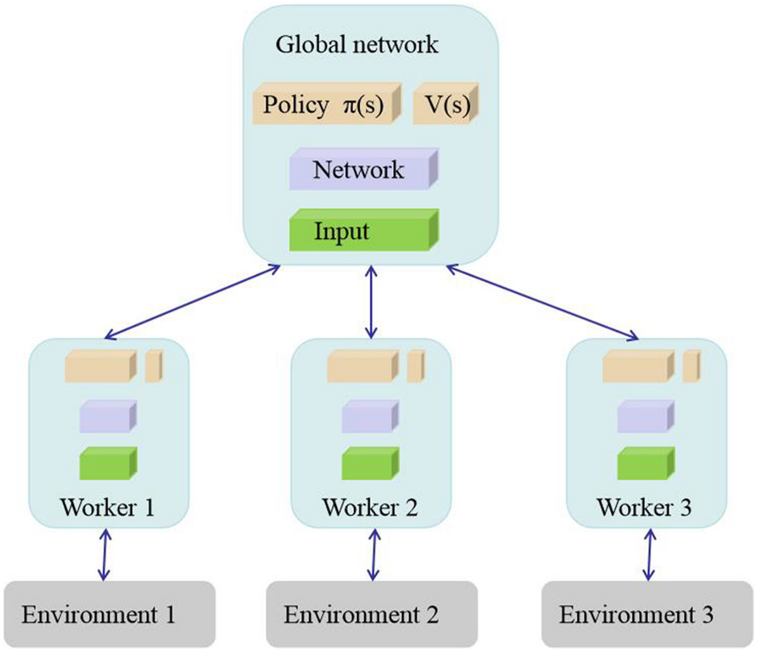

Fig. 2. A3C agent structure.

In A3C. The first ‘A’ is asynchronous. Multiple agents interact with the environment, a copy of the environment needs to be provided for each agent, and the states between the agents are asynchronous. The second ‘A’ is the advantage. The dominant function can be defined as the difference between the Q function and the value function. The Q function determines how well the behavior is in a certain state. The value function determines how good the state is the difference between Q function and value function. It means that the agent performs the behavior in the state s. The third ‘A’ is an actor-critic. There are two types of A3C neural network architecture, actors and critics. The role of the actor is to learn a strategy. The role of a critic is to evaluate the actor’s learning strategy.

There are multiple worker agents in Fig. 2. Each agent interacts with its own copy of the environment. Then the worker agent learns the strategy and calculates the gradient of the strategy loss. And update the gradient in the global network. This global network is updated by each agent. One of the advantages of A3C is that it does not use experience playback memory. Because there are multiple agents interacting with the environment and integrating their respective information into the global network. Information between agents. There is little or no correlation. Since A3C does not require memory, it can greatly reduce storage space and calculation time.

3.3 Formula Derivation

The agent tries to maximize the cumulative rewards it receives from the environment. The total amount of rewards an agent receives from the environment is called the reward. The reward function R_t is calculated as follows:

$$R_t = t_{t+1} + r_{t+2} + \cdots + r_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{1}$$

The policy function maps state to behavior as π. This represents a mapping from state to behavior. In essence, the policy function indicates what behavior is performed in each state. The ultimate goal is to find the optimal strategy for each state to specify the correct behavior, thereby maximizing the reward. The strategy function π can be expressed as:

$$\pi(s) : S- > A \tag{2}$$

State value function referred to as the value function. Determine the optimal degree of an agent in a certain state under the strategy. It is usually recorded as $V(s)$, which represents the value of the state after the policy is executed. The value function $V(s)$ is defined as follows:

$$V^{\pi}(s) = E_{\pi}[R_t | s_t = s] \tag{3}$$

Substituting formula (1) into the above R_t value, we get

$$V^{\pi}(s) = E_{\pi}[\sum_{k=0}^{\infty} r_{t+k+1}|s_t = s] \tag{4}$$

State behavior value function. Also called the Q function. Used to indicate that the agent adheres to the policy π. The optimal degree of specific actions performed in a state. The Q function is denoted as $Q(s)$, which represents the value of the behavior taken in a certain state by following the policy π. The Q function is defined as follows:

$$Q^{\pi}(s, a) = E_{\pi}[s_t = s, a_t = a] \tag{5}$$

The difference between the value function and the Q function is that the value function is the best way to determine the state. The Q function determines the optimal degree of behavior in a certain state.

Algorithm 1. A3C-based edge cache algorithm

```
1: Initialization parameters: $T = 0, \theta, \theta', \theta_v, \theta'_v$;
2: Repeat if there is a user content request;
3:     $d\theta = 0; d\theta_v = 0; \theta' = \theta; \theta'_v = \theta_v; t_{start} = t; s_t$;
4:     Repeat;
5:         Perform according to policy $\pi(a_t|s_t; \theta^t)$;
6:         Receive reward $r_t$ and new state $s_{t+1}$ ;
7:         $t = t + 1; T = T + 1$;
8:     until $s_t$ or $t - t_{start} == t_{max}$;
9:     $R = \begin{cases} 0 \\ V(s_t, \theta_v^{'}) \end{cases}$ ;
10:    For $i \in \{t - 1, \ldots, t_{start}\}$;
11:        $R = r_n + \gamma r_{n-1} + \gamma^2 r_{n-2}$;
12:        $\theta' : d\theta = d\theta + \nabla_{\theta'} log\pi(a_i|s_i; \theta')(R - V(s_i; \theta_v^{'}))$;
13:        $\theta_v^{'} : d\theta_v = d\theta_v + \sigma(R - V(s_i; \theta_v^{'}))^2/\sigma\theta_v^{'}$ ;
14: End for;
15: update $\theta$ using $d\theta$ and $\theta_v$ using $d\theta_v$;
16: Until $T > T_{max}$.
```

The A3C workflow first resets the worker agent to the global network. Then start interacting with the environment. Each worker agent learns an optimal strategy according to different exploration strategies. Next, calculate the value and strategy loss. Then calculate the loss gradient. And update the gradient in the global network. The worker agent restarts resetting the global network and repeats the counting process. The dominant function $A(s, a)$ is the difference between the Q function and the value function:

$$A(s, a) = Q(s, a) - V(s) \tag{6}$$

According to the above formula, we can get the value loss, which is the mean square difference between the discount return and the state value:

$$L_v = \sum (R - V(s)^2) \tag{7}$$

According to the value loss, the agent can train the A3C neural network, As the training increases, the value loss will gradually decrease, and the agent's reward will continue to increase.

By deploying servers with caching and computing capabilities at the edge of the network, you can solve low latency problems and ease the burden of high traffic in the cloud. Specifically, with edge computing, the edge cache can pre-cache some popular content during off-peak hours. Therefore, during peak hours, edge caching can reduce transmission delay and front-end transmission burden, and can also reduce traffic, thereby achieving the goal of improving the network environment.

In this article, we only focus on the caching strategy, which is to determine the content and timing of caching or updating the cached content, which is critical to the overall performance of the cache-enabled communication system. In this article, we choose the A3C agent with the decision-making ability to cache. A3C agent continuously improves the performance of the network experience through continuous interaction and learning with the environment at the network.

4 Experiments and Results

With the rapid development of edge computing and communication technology, traditional education is changing. Through the application of new technologies, the utilization rate of existing campus resources can be improved, and the work efficiency of students and teachers can be improved. Popular teaching content is cached on edge servers, such as communities, teaching buildings, etc. This way, the network environment, and user quality are improved.

Simulation experiments verify the effectiveness of the A3C edge cache method. This article uses the simulation experiment method to simulate the edge cache node by instantiation. For A3C agents, the actor and critic learning rate is 0.001. The experiment uses CentOS7, the processor is E5-2650, and the hard disk size is 480G SSD + 4T. Enterprise hard disk, 32G memory, code interpreter is Python, version 3.6, code dependent libraries include Tensorflow, Numpy, etc. User requests in experimental data are randomly generated by calling the Zipf function in Numpy.

The loss curve of the agent is shown in Fig. 3. The y-axis is the loss value of the edge node agent, and the x-axis is the edge node agent training cycle. 80% smoothing is done in Fig. 3 for demonstration. Every 200 times as a training cycle and record the training loss. In the first 100 cycles, the loss remained at a high level. On the one hand, the amount of training data was small, and on the other hand, the worker passed the neural network parameters to the global network for the update. After several updates, the global network was updated.

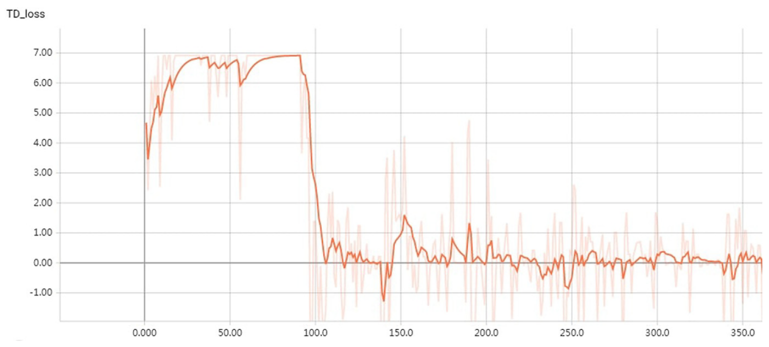

Fig. 3. A3C agent training loss.

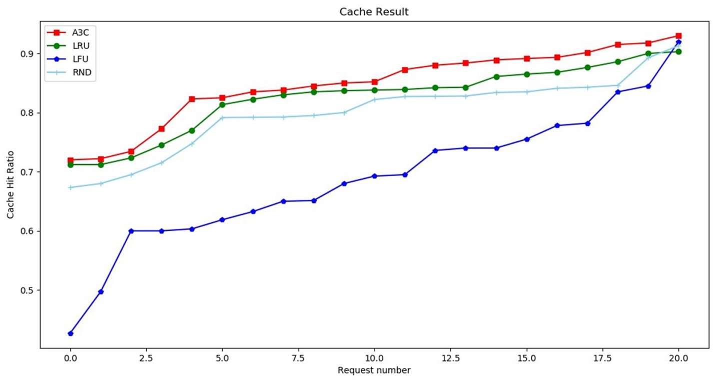

Fig. 4. A3C, LRU, LFU, RND hit ratio comparison.

Will reach a level of low loss. As the training progressed, the agent loss gradually decreased and reached around 0 after the training cycle reached 100, and then there was a large fluctuation. As the training continued, the fluctuation range became smaller and smaller, until the fluctuation range was less than plus or minus 1.

Figure 4 plots the comparative experimental results of A3C, RND, LRU, and LFU. The y-axis in the graph is the cache hit ratio, and the x-axis is the number of user requests (the unit is 1000). The cache hit rate we define is the number of cache hits divided by the sum of hits and misses. At first, due to the small number of user requests, the hit rates of the four caching strategies were very low. As user requests increase, training based on A3C caches also tends to stabilize. LFU, A3C performance is better. As the number of requests continues to increase, the cache hit rate of RND, LRU, and LFU also gradually increases, and the advantages of A3C also gradually weaken. Finally, when the number reaches the highest value, four cache methods reach similar levels. From the point of view of the caching process, a3c-based edge caching is stronger than LRU, LFU, and RND caching.

5 Conclusions and Future Work

In this paper, to alleviate the dilemma of online education facing high latency and stuttering, we propose an A3C-based edge cache online education architecture. By using resources placed in EN, content caching and hit ratio can be improved. Edge caching is an important way to improve the efficiency of 5G network content distribution. However, how to make more effective use of edge node caches in content storage and delivery still requires further exploration of computing and communication functions. Besides, the popularity of content will change with time, and how to design intelligent and adaptive caching mechanisms is a challenge in the future. We also need to further study how to motivate a large number of heterogeneous cache nodes to follow effective caching strategies, while ensuring content and network security.

References

1. Wang, X., Chen, M., Han, Z., et al.: TOSS: traffic offloading by social network service-based opportunistic sharing in mobile social networks. In: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, pp. 2346–2354. IEEE (2014)
2. Chen, M., Saad, W., Yin, C., et al.: Echo state networks for proactive caching in cloud-based radio access networks with mobile users. IEEE Trans. Wirel. Commun. **16**(6), 3520–3535 (2017)
3. Devooght, R., Bersini, H.: Collaborative filtering with recurrent neural networks. arXiv preprint arXiv:1608.07400 (2016)
4. Gu, J., Wang, W., Huang, A., et al.: Distributed cache replacement for caching-enable base stations in cellular networks. In: IEEE International Conference on Communications (ICC), 2648–2653. IEEE (2014)
5. Blasco, P., Gündüz, D.: Learning-based optimization of cache content in a small cell base station. In: 2014 IEEE International Conference on Communications (ICC), pp. 1897–1903. IEEE (2014)
6. He, S., Huang, W., Wang, J., et al.: Cache-enabled coordinated mobile edge network: opportunities and challenges. arXiv preprint arXiv:1912.11626 (2019)
7. Li, D., Han, Y., Wang, C., et al.: Deep reinforcement learning for cooperative edge caching in future mobile networks. In: IEEE Wireless Communications and Networking Conference (WCNC), pp. 1-6. IEEE (2019)
8. Zhang, K., Leng, S., He, Y., et al.: Cooperative content caching in 5G networks with mobile edge computing. IEEE Wirel. Commun. **25**(3), 80–87 (2018)
9. Chien, W.C., Weng, H.Y., Lai, C.F.: Q-learning based collaborative cache allocation in mobile edge computing. Future Gener. Comput. Syst. **102**, 603–610 (2020)
10. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: Proceedings of The 33rd International Conference on Machine Learning, vol. 48, pp. 1928–1937, 19 June 2016