Dynamic Caching Strategy for Multi-bitrate Videos Based on Popularity Prediction

Dongyang Peng, Guyu Hu*, Rui Wang, Jiachen Zu, Tianfeng Wang
Command and Control Engineering College, Army Engineering University of PLA, Nanjing, China
**Corresponding Author.*

Abstract

With the rapid increase of video traffic in the Internet, Content Delivery Network (CDN) has been used more widely, which is considered as a conventional method to support video delivery services. Scalable video coding and edge computing provide CDN with the ability to transcode videos at the edge of the network, enabling more flexible delivery of multi-bitrate videos. However, choosing which contents from the video library to cache on the edge servers remains a challenging problem, especially when the problem is extended to dynamic scenario. In this paper, an online caching model for dynamic scenario is designed innovatively, where the popularity of videos and users' requests change over time, while the caching strategy will be adjusted accordingly. Then a two-levels video popularity prediction algorithm (global popularity and local popularity) is novelly proposed, which is adapt to deal with in the caching strategy problem. Based on the popularity prediction, a revenue calculation algorithm is further proposed to obtain the predicted caching revenue of multi-bitrate videos. Finally, we develop a hybrid greedy based genetic algorithm to optimize the caching model, and perform simulation experiments using the YouTube dataset as a simulated dataset. The experimental results show that our popularity prediction based dynamic caching algorithm (PPDA) can greatly improve the cache hit ratio and reduce the average transmission delay.

Keywords: *Dynamic caching strategy, popularity prediction, multi-bitrate video, hybrid greedy based genetic algorithm, CDN*

I. Introduction

In recent years, there is a growing demand for online video, which is gradually becoming the longest-used domain by Internet users. Cisco predicts that IP video traffic will account for 82% of all IP traffic in 2022 [1]. At the same time, users' requirements for video service quality are increasing, including not only video bitrate but also video response delay. Although video service providers are adopting new technologies to address these issues, network bandwidth and video servers are under more tremendous pressure.

CDN [2] is a traditional method used to provide video distribution services. It deploys a large number of cache servers at the edge of the network, and users can access videos through the nearby cache servers. In this way, CDN not only reduces video response delay, but also avoids a large amount of redundant traffic and eases the congestion of the backbone network.

Scalable video coding [3] is proposed to ensure the smoothness of video transmission in network with different bandwidth. Scalable video coding encodes video streams into multi-layer streams, which usually include base layer and enhancement layer. Based on the bandwidth, the appropriate video stream will be selected to transmission to the user. This also effectively solves the problem of inconsistent video clarity requirements of different users. However, SCV also increases the computational power requirements of servers and end devices. For example, the HEVC/H.265 video compression standard increases the computational power requirements for video encoders by at least four times [4].

Recently, distributed computing has been used in CDNs [5-8], which can unload the main data processing and storage at the edge nodes of the network. A source server deployed on a cloud computing node connected to several cache

servers deployed on edge computing nodes becomes a typical deployment architecture for CDNs. Although the CDN architecture mentioned above can provide users with better online video services, there is still an important challenge: which videos should be cached on cache servers?

The advantage of CDN is that they can provide services close to the user, which imposes a basic requirement that the video requested by the user must already be stored on the cache node. If the requested video is not cached, the caching server needs to request the video resource from the source server. Obviously large number of unhit requests would lead to a poor performance of CDN and users' experience. Since the cache capacity of each cache node is limited, a single edge node does not have enough capacity to cache all files in video library. Therefore, this requires us to select the appropriate and limited number of video files to store on cache nodes. And this is the caching strategy problem in multi-bitrate videos.

Since users' preferences are diverse and time-varying, and video service providers are constantly providing new video contents, the caching strategy on each caching node needs to make pointed adjustment. The objectives of developing caching strategy are also different in different scenarios. The cache hit ratio, the average delay of video transmission, the maximum bitrate of video, and user satisfaction are all perspectives that deserve to be discussed and analyzed.

Currently, most works treat the caching strategy problem as a kind of 0-1 backpack problem that we need to select a number of video contents from a video library and store them in different servers. Each video has a different value, and these values can be defined as delay savings, users' satisfaction, cache hit ratio, etc. Thus, the problem can be formulated as how to design the strategy of selection to maximize the total value of the videos in the cache server under capacity limitation. Previous works have studied this issue and achieved good results. On the basis of these studies, this problem is expended.

First, in contrast to previous works considering only the caching policy issues in static scenarios, we take the dynamic scenario into account. In our dynamic caching model, a continuous period of time is divided into several smaller time periods. During each time period, the video requests change with users' preferences, and the cache server will adjust its caching strategy with the change of requests. Thus, our goal becomes to find the caching strategy in all time periods that maximizes the sum of revenues. The dynamic scenario of the caching strategy problem can provide us with a global perspective, avoiding the frequent adjust to the cached content at each time period.

Second, the popularity of videos is predicted by analyzing the historical data of users' requests, instead of setting the popularity to obey the Zipf distribution by default as in the previous works. Besides, Yan et.al [9] indicate that the popularity of video varies by regions, which suggests that the popularity of the same video varies in different cache nodes. It is reasonable to consider that although each video has a popular trend, it is different on each cache node. Therefore, the popularity of videos can be predicted from two perspectives innovatively. On the one hand, we predict the local popularity of each video is from the users' request information received by each cache node. On the other hand, we obtain the cache information of all cache nodes from the source server and use it to predict the global popularity of each video.

Third, when considering the caching revenue of a multi-bitrate video, we should distinguish the transcoding relationship between different bitrate versions. For example, assume that there is a request for the video with the bitrate of 128kps, while only videos with the bitrate of 512kps and 1024kps are cached. Now the server can response this request by transcoding and it is wiser to select the 512kps bitrate version, which has a smaller transcoding delay. Finally, by combining the caching revenue calculation method and the predicted popularity of videos, we can find the predicted caching revenue values of videos on each cache node in each time period, and propose a hybrid greedy based genetic algorithm to maximize the overall revenue.

By conducting experiments to compare our algorithm with other caching algorithms, we show that our dynamic

model can greatly improve the cache hit ratio and reduce the average transmission delay.

In summary, the main contributions of this paper are as follows.

(1) A caching model for dynamic scenarios is proposed firstly, which differs from the previous works. This model takes into account the dynamics of users' requests and the trend of video popularity.

(2) We novelly analyze video popularity on two levels, which will be helpful in the caching strategy problem. With local popularity, the caching strategy can be adjusted in advance to better adapt to users' requests. And the global popularity can analyze the overall popularity trend and determine whether the popularity will explode in the next time period.

(3) The method to calculate the cache revenue for multi-bitrate videos is designed, which jointly takes the transcoding relationship between different bitrate versions of the same video into consideration. Each cache node is guaranteed to respond to the user's request with minimal delay based on the current cached contents.

(4) An HGGA algorithm is developed to optimize the problem, which will be used to update the cached contents. Through simulation experiments, it is verified that our dynamic model can achieve better experimental results than existing works.

The remainder of this paper is organized as follows. Section II summarizes the related work. Section III describes the system model in detail. Section IV describes our algorithm in detail. Section V performs simulation experiments and analysis. Section VI summarizes the full paper and provides an outlook for future work.

II. Related Work

2.1 Optimization objectives and constraints for the caching strategy problem

Previous works have studied the problem of caching strategies. Depending on the scenario, the focus of those studies varies, and the main differences are the optimization objectives and constraints of the caching strategy. The optimization objective proposed by Poularakis et.al [10] was the average delay of video transmission with the caching capacity of the cache server as a constraint. This was designed to prioritize the ability of users to view videos smoothly, but had the impact of generally lower video quality. In contrast, the literature [11] aimed at maximizing the bitrate of the transmitted video to provide users with a higher quality viewing experience. Combining these two views, the user satisfaction [12] was utilized as an optimization objective, which can be represented by a utility function. Such an approach considers both transmission delay and video bitrate to find a relatively balanced result. Besides, Li et.al [13] considered the caching strategy from the perspective of energy savings, where influencing factors includes the required energy of caching, transcoding, and transmissing videos.

With the introduction of multi-bitrate videos [14, 15], a re-estimation of the caching revenue was required and the transcoding capacity of the cache server was added as a constraint. To further investigate the impact of video transcoding on the caching strategy, literatures [16, 17] discussed the possible responses of the caching nodes when they receive a video request and analyzed which node should perform the transcoding.

2.2 Popularity prediction in caching strategy problems

In most works, the popularity of a video is a key metric, which greatly determines the probability of that video being requested by users. It is a common phenomenon that the video popularity is set to obey the Zipf distribution [18-20]. The literature [13] obtained a popularity distribution through a real dataset, but ignoring the dynamic changes of video popularity. In [21], the viability of using popularity prediction content caching scenarios was ascertained by

Famaey et.al, and their caching strategy can effectively improve cache hit rate.

There are many works from different perspectives to study the trend of video popularity. Wang et.al [22] used the burst to describe the situation of a steep rise in popularity in a short period of time followed by a rapid fall. In order to predict a burst efficiently, videos were categorized and studied by Nguyen et.al [23], who found that a burst is likely to occur in news and short-form videos [24]. Further investigated the burst and stated that the popularity of the same video varies greatly among different scenarios and different user groups. Besides, there was a positive relationship between video life cycle and popularity [25]. In addition to the burst, Su et.al also predicted the evolution of video popularity and its stages as a new exploration [26]. On the other hand, Claeys et.al [27] considered the correlation between different videos, and utilized the behavioral characteristics that users stream multiple consecutive episodes of the same series. It is worth noting that deep learning methods have also been applied to popularity prediction in caching strategy problems [28-30].

2.3 Solutions to the caching strategy problem

The common solution to the caching strategy problem is to reduce it to a 0-1 knapsack problem by simplifying some of the parameters. Zhang et.al [31] utilized the branch and cut method, which improves the hit rate by about 5%. While Zhao et.al [32] used the Lagrangian relaxation algorithm to obtain a solution method with lower algorithmic complexity. A heuristic algorithm based on a greedy strategy was proposed in [17], which also achieves good results in content access delay.

Some works considered the problem from other perspectives. In [12], Han et.al defined a user satisfaction function with respect to delay, and utilized a standard convex optimization algorithm to solve it, which greatly improves users' quality of experience (QoE). Besides, the caching strategy problem was described as a special coverage problem and a low complexity 1/2 approximation algorithm was given [33]. On the other hand, Kamiyama et.al [34] proposed a cache replacement strategy based on the hop distance to source servers, where the storage capacity of cache severs is divided into multiple virtual caches and content items are managed separately. This strategy can decrease the average link load by about 10%.

In conclusion, the 0-1 backpack problem is a class of NP-hard problems, and it is difficult for us to solve its optimal solution. Thus we can only solve the approximate solution of its optimal solution by continuously trying.

Inspired by the above research, we have the idea to combine the caching strategy problem with popularity prediction: how to appropriately adjust the caching strategy to maximize the overall revenue during the dynamic change of video popularity?

III. System Model AND Problem Formulation

In this section, we first describe our caching strategy model for dynamic scenarios, and then present the process of the caching system responding to users' requests, performing popularity prediction, and updating the caching strategy. Final, we describe how to calculate the caching revenue for multi-bitrate videos and obtain the optimization problem.

3.1 Caching model

Fig.1 illustrates a typical caching model: a source server stores all the video sources inside, and each video is encoded into different bitrate versions. Several cache servers deployed with caching and computing resources are connected to the source server, to store and transcode video files and serve users.

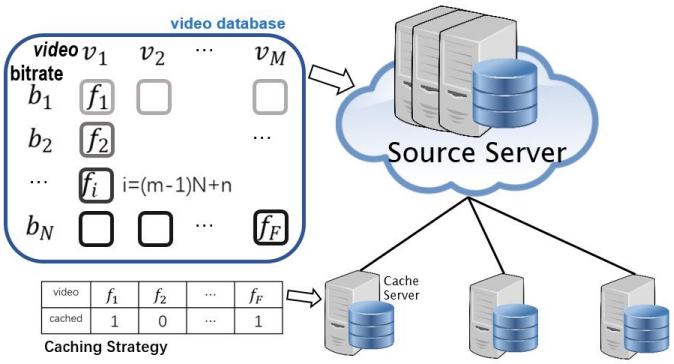

Fig 1: Caching model

Assume that each user's location is fixed and every user will acquire video services through the nearest cache server. That is, the dynamic changes in user locations is ignored and the entire network topology is considered static. Assuming that users periodically make video requests, the cache server adjusts its cached contents based those requests and the source server can obtain all request information of users to predict the popularity trend of videos.

3.2 Symbols and variables

Assume that M kinds of videos are stored in the source server, and each video has N bitrate versions. We use the set $\mathcal{M} = \{v_1, v_2, \dots, v_M\}$ is uesd to denote the different videos and let l_m denote the length of video v_m. The set $\mathcal{N} = \{b_1, b_2, \dots, b_N\}$ is used to denote the possible bitrate versions, assuming that $b_1 < b_2 < \cdots < b_N$. For simplicity, the set $\mathcal{F} = \{1,2, \dots, i, \dots, F\}$ is defined to denote the all video files and $i = (m-1)N + n$ represents the m-th video with the n-th bitrate version. The data size of file i is $s_i = l_m b_n$.

The source server is denoted as S_0 and the cache servers can be denoted as $\mathcal{K} = \{S_1, S_2, \dots, S_K\}$. Each cache server has the same caching capacity C_C and transcoding processing capacity C_T, so the delay of transcoding video file i to its lower bitrate version file j can be denoted as $d_{i,j}^T = (s_i - s_j)/C_T$. The communication bandwidth between source server S_0 and cache server is B, so the transmission delay of the video file i from S_0 to the cache server is $d_i = s_i/B$.

In the caching strategy problem, the popularity of a video is a very important factor, which greatly determines the priority of a video file to be cached. However, due to the variability between regions, the popularity of the same video may vary in different regions. To deal with this challenge, the prediction of global popularity and local popularity is designed. We use $P_{i,k}^l$ to denote the local popularity of the file f on the serverS_k and use $P_{i,k}^g$ to denote its global popularity.

For simplicity, each cache node is assumed to serve U users, and each user makes requests at the same periodicity. The list of requests on the server S_k is denoted by $Q_k = \{q_1, q_2, .., q_U | q_u \in F\}$. Note that different users may request the same video file, so the elements in the request list Q_k are repeatable. We define the variables $x_{i,k} \in \{0,1\}$, where $x_{i,k} = 1$ indicates that file i is cached in the server S_k, and $x_{i,k} = 0$, otherwise. Accordingly, the caching strategy of server S_k can be denoted by $X_k = \{x_{1,k}, x_{2,k}, \dots, x_{F,k}\}$.

3.3 The revenue of cached videos

The delay savings is considered as the revenue of caching a video file. Since the delay on the link from the cache server to the user is unavoidable, we ignore the delay on this link when discussing the delay savings. To illustrate how the caching revenue is calculated, we assume the scenario where the cache server S_k has a caching strategy X_k and receives a request list Q_k. At this time the revenue of X_k will be calculated. Assuming that a element in Q_k is q_u,

corresponding to video file i, there are three possible cases for caching revenue: (a) if q_u is cached in S_k, then caching revenue (denoted as $r(X_k, q_u)$) is d_i. (b) if S_k does not cache q_u, but caches a higher bitrate version file of q_u (denoted as i'), the caching revenue is $di - d^T_{i',i}$. Note that the lowest bitrate one will be selected for transcoding when several higher bitrate version files of q_u have been cached. (c) if S_k does not cache q_u, nor does it cache any higher bitrate version file of q_u, then the caching revenue is 0.
Thus, we can obtain the revenue from the caching strategy X_k on the request list Q_k:

$$R(X_k, Q_k) = \sum_{q_u \in Q_k} r\,(X_k, q_u). \tag{1}$$

3.4 Caching strategy model for dynamic scenarios

The above calculation of $R(X_k, Q_k)$ requires the precondition that both the cache strategy and users' request are known. However, the adjustment of caching strategy is actually later than the arrival of users' requests. In other words, if the caching strategy is adjusted in the current time period, the revenue from this new caching strategy can not be calculated until the users' new requests arrive in the next time period.

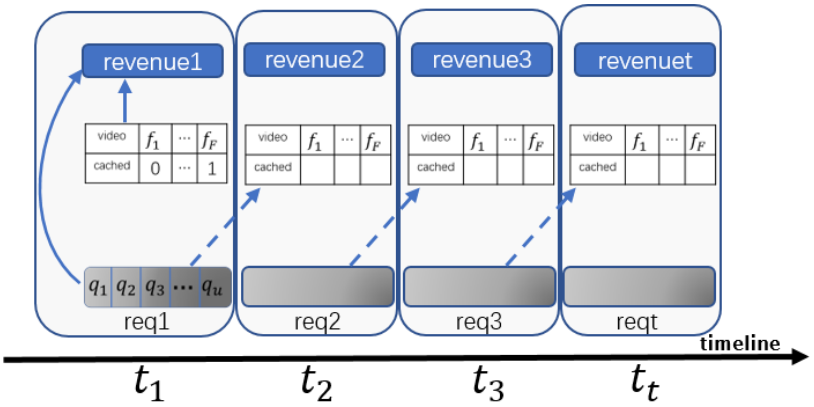

Fig 2: Dynamic changes in caching strategy

As shown in Fig.2, we divide the overall time period $\mathcal{T}$ into consecutive time periods $\mathcal{T} = \{1,2, \dots, t, \dots, T\}$. At time period t, a cache server S_k with the caching strategy $X_k(t)$ receives a list of requests $Q_k(t)$. Requests on all servers can be denoted by the set $\mathcal{Q}(t) = \{Q_1(t), Q_2(t), .., Q_K(t)\}$. The local popularity and global popularity of videos can be obtained by analyzing $Q_k(t)$ and $\mathcal{Q}(t)$ accordingly, then the cache server will transform $X_k(t)$ into a new caching strategy $X'_k(t)$.

At the arrival of the next time period $t+1$, we obtain the revenue from $X'_k(t)$, which is denoted by $R\big(X_k(t+1), Q_k(t+1)\big)$, where $X_k(t+1) = X'_k(t)$. Assume that the individual cache servers are independent of each other, ignoring the ability of the servers to work together. Thus, in time period t, the revenue from all caching strategy can be expressed as

$$R(t) = \sum_{k \in K} R\big(X_k(t), Q_k(t)\big). \tag{2}$$

The caching strategy on all servers within the overall time period $\mathcal{T}$ is denoted by the set $\mathcal{X} = \{X(1), X(2), \dots, X(T)\}$, where $\mathcal{X}^{max}$ denotes the optimal solution. $X(t)$ denotes the caching strategy on all servers at the time period t, where $X^{max}(t)$ denotes the optimal solution accordingly. In order to analyze the relationship between $X^{max}(t)$ and $\mathcal{X}^{max}$, the following lemmas are required.

Lemma 1: There is no additional delay cost when the cache server replaces cached content.

Proof: Suppose that at time period t, a adjustment is made to an item x_i in cache strategy X. If the adjustment is $x_i = 0$, it means that the file i needs to be removed from the cache space and there is no additional transfer delay obviously. If the adjustment is $x_i = 1$, it means that file i needs to be added into the server's cache space. This adjustment in caching strategy results from a request for file i. Therefore, in response to the request, file i will be stored in the temporary storage space of the cache server at time period t, whether it is transferred from the source server or transcoded from a high bitrate version file. The delay cost of this process has been taken into account in formula (1), which does not need to be recalculated in the process of adding file i to the cache space. Thus, there is no additional delay cost when the cache server replaces cached content.

Lemma 2: The optimal caching strategies in each time period are independent and $\mathcal{X}^{max} = \{X^{max}(1), X^{max}(2), \dots, X^{max}(T)\}$.

Proof: Assume that at time period t, the caching revenue on server S_k is $R\big(X_k(t), Q_k(t)\big)$, and the caching strategy change from $X_k(t)$ to $X'_k(t)$. We describe the transformation of caching strategy between time period t and time period $t+1$ as $X_k(t) \to X_k(t+1)$, where $X_k(t+1) = X'_k(t)$. From Lemma 1, it is known that there is no additional delay cost for the server to perform cache replacement. Regardless of $X_k(t)$, the process $X_k(t) \to X_k(t+1)$ has no additional delay cost and does not have an impact on the caching revenue $R\big(X_k(t+1), Q_k(t+1)\big)$. Therefore, $X_k(t)$ can be ignored when searching for the optimal caching strategy $X^{max}(t+1)$ in time period $t+1$, i.e., $X_k(t)$ and $X^{max}(t+1)$ are independent.

Therefore, it can be inferred that the optimal caching strategies in each time period are independent, and the optimal caching strategy in the overall time period $\mathcal{T}$ can be combined by the optimal caching strategies in each time period, which can be expressed as $\mathcal{X}^{max} = \{X^{max}(1), X^{max}(2), \dots, X^{max}(t), \dots, X^{max}(T)\}$. From the above discussion, the revenue from caching strategy on all servers during the overall time period $\mathcal{T}$ can be denoted as

$$R(\mathcal{T}, \mathcal{K}) = \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \sum_{q_u\in Q_k(t)} r\,(X_k(t), q_u). \tag{3}$$

Taking into account the limitations of the cache space, the optimization problem of the caching strategy model for dynamic scenarios (denoted as Q) can be formulated as

$$Q: \max_{\mathcal{X}} R(\mathcal{T}, \mathcal{K}) \tag{4}$$

$$s.t. \forall k, t, \sum_{i\in F} x_{i,k}(t) * s(i) \le C_c \tag{5}$$

$$\forall i, k, t, x_{i,k}(t) \in \{0,1\}. \tag{6}$$

IV. Proposed Algorithm

In this section, we present the approach to solving the problem Q. We first describe our overall algorithm, the popularity prediction based dynamic caching algorithm (PPDA). Then each part of the overall algorithm is explained, mainly including prediction of video popularity, prediction of caching revenue and hybrid greedy based caching strategy genetic algorithm.

4.1 Overall algorithm

PPDA is designed to solving the problem Q, which is shown in Algorithm 1. The workflow of the algorithm mainly includes the following steps: (1) In each time period, the source server receives requests from all servers so that it can analyze and predict the trend of video popularity from the time dimension. The global popularity update algorithm (GPUA) is designed to accomplish this task. (2) Each cache server S_k makes a prediction of the popularity

of the video based on the users' requests it receives, and we can use the local popularity update algorithm (LPUA) to get the local popularity of videos. (3) The predicted caching revenue can be calculated, by using the caching revenue calculation method for multi-bitrate videos and the predicted popularity. (4) A hybrid greedy based genetic algorithm (HGGA) is proposed to find the optimal caching strategy $X_k^{max}(t)$, then each cache sever will update its cached video content. Next, we will describe in detail the algorithm mentioned above.

Algorithm 1 Popularity Prediction Based Dynamic Caching Algorithm

Input: all requests in the overall time period $\mathcal{T}$
Output: the optimal caching strategy $\mathcal{X}^{max}$
while each time period $t \in \mathcal{T}$ ***do***
GPUA: Calculate the global popularity
for each server $k \in \mathcal{K}$ ***do***
LPUA: Calculate the local popularity
Calculate predicted caching revenue
HGGA: Calculate $X_k^{max}(t)$
Update the caching contents
end for
end while

4.2 Prediction of video popularity

4.2.1 Trend analysis of videos

The article [23] points out that the popularity of movie and TV videos shows a trend of slow growth followed by rapid growth until it reaches a peak, and the change of popularity is not smooth but fluctuating.
We also analyze the YouTube dataset, which will be detailed in Section V, and the results of the analysis also validate the above conclusions.

A typical video sample is selected as a demonstration. The number of times a video is viewed per day is considered its popularity, and the number of views per day is normalized by the maximum number. Then the popularity trend of the video roughly can be obtained as shown in Fig.3.

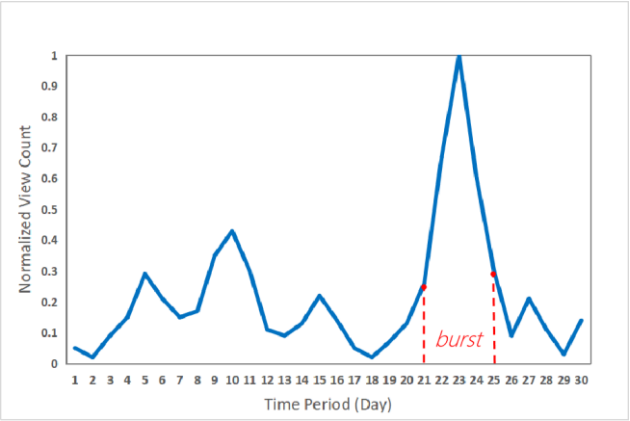

Fig 3: Popularity trend of the video

From the Fig.3, it can be observed that the popularity of the video did not grow slowly to reach its peak, but rose steeply. This phase labeled in the figure is called burst, which is defined in [22] as "a brief period of intensive activity followed by long period of nothingness". There is a burst in the evolution of video trends, which can be understood as a video becoming extremely hot for a short period of time until it peaks, and then its popularity decreases dramatically.

4.2.2 Local popularity update algorithm

The cache server can not have a comprehensive analysis of video popularity due to the lack of a global view, and it can only make predictions about local users' preferences. A common approach of popularity prediction is to make an adjustment based on the number of requests, where the value of the popularity of file i will plus one when it receives a request for file i. We name this method the linear popularity adjustment method, meaning that there is a linear relationship between the magnitude of the popularity adjustment and the number of requests. However, the analysis of Fig.3 shows that the trend of video popularity is non-linear, which does not match the linear adjustment method.

Besides, the linear way of popularity adjustment has obvious lag, if the popular phase of a video is short, it is possible that the video's popularity hasn't even been adjusted to a high value before its actual popularity starts to drop. Thus, we design a nonlinear video popularity adjustment method with the following ideas.

Suppose that the number of requested video file i in the request list $Q_k(t)$ is $C\big(i, Q_k(t)\big)$, and there are total U requests in $Q_k(t)$. We set different magnitudes for the adjustment of $P_{i,k}^l(t)$ according to the proportion of $C\big(i, Q_k(t)\big)$ to U, which is shown as follows. (a) If $C\big(i, Q_k(t)\big) \in [0,0.1U)$, $P_{i,k}^l(t) = C\big(i, Q_k(t)\big)$. (b) If $C\big(i, Q_k(t)\big) \in [0.1U, 0.3U)$, we add a weight for the part that exceeds $0.1U$, which is denoted as α. It means that the part that exceeds $0.1U$ can provide an additional contribution to the popularity, which can be formulated as $P_{i,k}^l(t) = 0.1U + a\big(C\big(i, Q_k(t)\big) - 0.1U\big)$. (c) If $C\big(i, Q_k(t)\big) \in [0.3U, U]$, this indicates that the proportion of requests for file i is high enough, and the probability that file i will get widely popular in a short time is great. So the popularity of file i will be adjusted in time, by setting $P_{i,k}^l(t) = 0.1U + 0.2\alpha U + e^{C\left(i,Q_k(t)\right)-0.3U}$.

Thus, the prediction of the local popularity of the video file i by the cache server S_k at time period t can be formulated as ($C\big(i, Q_k(t)\big)$ is abbreviated as C)

$$P_{i,k}^l(t) = \begin{cases} C, C\epsilon(0,0.1U) \\ 0.1U + \alpha(C - 0.1U), C\epsilon[0.1U, 0.3U) \\ 0.1U + 0.2\alpha U + e^{C\left(i,Q_k(t)\right)-0.3U}, C\epsilon[0.3U, U]. \end{cases} \tag{7}$$

4.2.3 Global popularity update algorithm

An accurate prediction of burst will effectively improve the efficiency of the caching strategy, especially during its rising phase [35]. But how to accurately predict it is still an urgent problem. The main difficulty is that we do not yet know when the burst will start and when it will end. Since our caching strategy aims at predicting the popularity of the video in the next time period, we only need to determine whether the burst will appear in the next time period, instead of knowing its accurate appearance time. Therefore, a simple approach is designed to determine whether a video will be in burst in the next time period.

The source server has access to request information for all users on the cache server, so we assume that the total number of times video file i is requested in time period t is denoted as $C(i,t) = \sum_{k \in K} C\big(i, Q_k(t)\big)$. Then the increment of the total number of times video file i requested in time period t is denoted as $\Delta C(i,t) = C(i,t+1) - C(i,t)$, and assume that the duration of each time period is a fixed value of Δt. So that the growth rate of the total number of requests for video file i in time period t can be denoted by $\Delta C(i,t)/\Delta t$. Through the analysis of Fig.3, we can summarize that the trend of video popularity in burst will have the following main characteristics. a) the total number of requested videos has a large value. b) the total number of requested videos i has a large growth rate. c) the total number of requested video i has a fast variation in the growth rate.

According to these three characteristics, the following constraints are used to determine whether video file i will be in burst in time period $t+1$. (a) $C(i,t) > KU/4$. (b) $\Delta C(i,t)/C(i,t) > 1$. (c) $\Delta C(i,t)/\Delta t > 1.5\Delta C(i,t-1)/\Delta t$.

If the number of requests for video file i in time period t can satisfy the above three conditions at the same time, file i is considered to be likely to be in burst in time period $t+1$, and set $P_i^g(t) = C(i,t)$, otherwise $P_i^g(t) = 0$. Note that our global popularity prediction approach is only a rough estimation and just serves as an aid to local popularity prediction.

4.2.4 Combined prediction of video popularity

Combining LPUA and GPUA, the predicted value of the popularity of file i on cache server S_k in time period t can be obtained, which is denoted as

$$P_{i,k}(t) = P_{i,k}^l(t) + P_{i,k}^g(t). \tag{8}$$

However, such a description method also has a problem that our predicted value $P_{i,k}(t)$ only reflects the popularity of the video in time period t. It can better reflect the popularity of the video in the short term, but considering the long life cycle of a video, it is likely to become popular again after a time period [36].

Therefore we consider a partial retention of the historical information on video popularity and rewrite the formula (8) as

$$P_{i,k}(t) = 0.5P_{i,k}(t-1) + 0.5\left(P_{i,k}^l(t) + P_{i,k}^g(t)\right). \tag{9}$$

4.3 Predicted caching revenue

The predicted value of caching revenue is different from the caching revenue described in Section II. The caching revenue is the actual revenue that is used to judge the goodness of our caching strategy, which could not be calculated until both the cache strategy and users' requests are known for the current time period. However, the caching revenue prediction is an estimation of the revenue from caching a video file. Next, the following will describe how to calculate the predicted caching revenue of multi-bitrate videos.

With formula (9), the predicted popularity $P_{i,k}(t)$ can be obtained, so the predicted value of times the video has been requested can be denoted as

$$C_{i,k}^p(t) = U\frac{P_{i,k}(t)}{\sum_{i\in F} P_{i,k}(t)}. \tag{10}$$

Based on the analysis in Subsection C of Section III, the predicted value of video caching revenue should be discussed in three cases.

Case 1: If the file i will not be cached, the the predicted caching revenue will be zero obviously.

Case 2: If the file i will be cached but its other version files will not be cached, the predicted caching revenue consists of two parts. One part is the revenue from the direct requests for file i, which is denoted by $r_1 = C_{i,k}^p(t) * d_i$. The other part is the revenue from the requests for the lower bitrate version files of i, which is denoted by a set $\omega(i)$. Then the the revenue from $\omega(i)$ can be expressed as $r_2 = \sum_{j\in\omega(i)} C_{j,k}^p(t) * \left(d_j - d_{i,j}^T\right)$. Therefore, in this case, the predicted revenue from caching video file i is $R_{i,k}^p(t) = r_1 + r_2$, i.e.

$$R_{i,k}^p(t) = C_{i,k}^p(t) * d_i + \sum_{j\in\omega(i)} C_{j,k}^p(t) * \left(d_j - d_{i,j}^T\right). \tag{11}$$

Case 3: If both file i and its other version files will be cached, the predicted value from caching file i can be calculated in the following way. Assume that a set of several other version files of i is denoted by $\varphi(i) = \{1,2,..,c\}$, $\big(1,2,..,c \in U(i)\big)$, where $U(i)$ denote the set of all version files of i. The predicted revenue from caching $\varphi(i)$ is denoted by $R^p_{\varphi(i),k}(t)$, which also consists of two parts. One part is the revenue from the direct requests in $\varphi(i)$, which can be expressed as $r_1 = \sum_{j\in\varphi(i)} C^p_{j,k}(t) * d_j$. The other part is the revenue from the requests for the lower bitrate version files of $\varphi(i)$, which can be expressed as $r_2 = \sum_{j\in\overline{\varphi}(i)} r^{max}_{(j)}$, where $\overline{\varphi}(i) = U(i) - \varphi(i)$.

Next the meaning of $r^{max}_{(j)}$ will be explained. When a user request an uncached video j, the cache server should select the file with the closest bitrate to j from the cached files to transcode it. So that the cache server will obtain the maximum revenue $r^{max}_{(j)}$, which can be calculated as:

$$r^{max}_{(j)} \begin{cases} 0, \psi(j) = \emptyset \\ \max\limits_{j'\in\psi(j)} C^p_{j',k}(t) * \left(d_j - d^T_{j',j}\right), \psi(j) \neq \emptyset \end{cases} \tag{12}$$

In formula (12), $\psi(j) = \Omega(j) \cap \varphi(j)$ denotes the set of cached files which can be transcoded to file j and $\Omega(j)$ is the set of higher bitrate version files of j.

The predicted value of revenue from caching $\varphi(i)$ is $R^p_{\varphi(i),k}(t) = r_1 + r_2$, i.e.

$$R^p_{\varphi(i),k}(t) = \sum_{j\in\varphi(i)} C^p_{j,k}(t) * d_j + \sum_{j\in\overline{\varphi}(i)} r^{max}_{(j)}. \tag{13}$$

Thus, in case 3, the predicted value from caching the video file i can be inferred as

$$R^p_{i,k}(t) = R^p_{\varphi(i)\cup\{i\},k}(t) - R^p_{\varphi(i),k}(t). \tag{14}$$

In summary, the sum of predicted caching revenue of all servers on the overall time period $\mathcal{T}$ can be calculated as

$$R^p(\mathcal{T},\mathcal{K}) = \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \sum_{q_u\in Q_k(t)} R^p_{i,k}(t). \tag{15}$$

4.4 Hybrid greedy based genetic algorithm (HGGA)

In the previous discussion, it found that the problem Q cannot be directly solved, resulting from that the caching strategy is adjusted after the arrival of users' requests. Therefore, we can only use the predicted caching revenue to determine our caching strategy, and the problem Q is reformulated as

$$Q': \max_{\mathcal{X}} R^p(\mathcal{T},\mathcal{K}) \tag{16}$$

$$s.t. \forall k,t, \sum_{i\in F} x_{i,k}(t) * s(i) \leq C_c \tag{17}$$

$$\forall i,k,t, x_{i,k}(t) \in \{0,1\} \tag{18}$$

It can be inferred that the solution of problem Q' is closer to the solution of problem Q when the predicted value of our cache revenue is closer to the actual value. Now our goal becomes to to find the optimal caching strategy denoted as $\mathcal{X}^{max}$, which has the highest $R^p(\mathcal{T},\mathcal{K})$. From Lemma 1 and Lemma 2, it can be inferred that $\mathcal{X}^{max}$ is a combination of the optimal caching strategy $X^{max}_k(t)$ in each cache server on each time period t, i.e.,$\mathcal{X}^{max} = \{X^{max}_1(1), X^{max}_1(2), \dots, X^{max}_k(t), \dots, X^{max}_K(T)\}$. $X^{max}_k(t)$ is considered as the optimal solution to the subproblem of

Q'. The subproblem Q^* can be formulated as

$$Q^*: \max_{X_k(t)} \sum_{i \epsilon Q_k(t)} R_{i,k}^p(t) \tag{19}$$

$$s.t. \forall k, t, \sum_{i \in F} x_{i,k}(t) * s(i) \leq C_c \tag{20}$$

$$\forall i, k, t, x_{i,k}(t) \in \{0,1\} \tag{21}$$

Therefore we simply require to find the solution to problem Q^* and apply it to problem Q'.

Q^* is a typical 0-1 backpack problem, and there has been various solutions. Genetic algorithm (GA) is a general heuristic algorithm, and the literature [37] points out that GA has the problem of insufficient local solving ability and does not easily jump out of local extrema. An improved genetic algorithm based on a hybrid greedy strategy is developed to solve the problem Q'.

There are two main operators in GA: the repair operator and the optimization operator. The main task of the repair operator is to remove items according to specific rules in infeasible solutions until they becomes feasible. While the main task of the optimization operator is to select suitable items to add according to specific rules in feasible solutions and stop before the infeasible solution appears again.

A genetic algorithm based on a greedy selection according to value density was proposed in [38], which means that it prefers items of the highest unit value. This strategy speeds up the convergence of the algorithm, but it also increases the risk of falling into local extremes. Therefore, we make an improvement to the algorithm by updating the population with the hybrid greedy operator and locally perturbing each individual with the local search operator.

Referring to previous works, the hybrid greedy operator is designed to use a probabilistically based choice of value density-based strategy or value-based strategy when fixing infeasible solutions. This approach can effectively avoids from getting trapped in extreme value points. The local search operator randomly flips a bit for each individual and receives modifications in a greedy way, i.e., if the flip makes the revenue increase, the new solution is accepted, otherwise the old solution is maintained. By setting the number of local searches reasonably, the possibility of the algorithm jumping out of the local extremes can be improved effectively. The pseudo code for HGGA is provided in Algorithm 2. In the input of the algorithm, $ETimes$ and $LTimes$ denote the times of population evolution and the times of local searches, respectively. The population is denoted by A, and one of the individuals is denoted by a. Each individual represents a caching strategy. LV and LD denote the list of videos in descending order of value and descending order of value density, respectively. The output is the optimal caching strategy denoted as a^{max}.

Algorithm 2 Hybrid Greedy Based Genetic Algorithm (HGGA)

Input: $A, ETimes, LTimes, LD, LV$
Output: the optimal solution a^{max}
function *RepairOperator*$(list, a)$:
  for i ∈ list ***do***
    if $W > Cc$ and a_i=1 ***then***
      a_i=0
      W=W-w_i

function *OptimizationOperator*$(list, a)$:
  for $i \in list$ ***do***
    if $W < Cc$ and a_i=0 ***then***

a_i=1
W=W+w_i

function *HybridGreedyOperator*(LD, LV, p, a):
  RepairOperator(LD, a)
  Randomly generate $r \in [0,1)$
  If $r < p$ ***then***
    OptimizationOperator(LD, a)
  Else
    OptimizationOperator(LV, a)

function LocalSearchOperator $LTimes, a$:
  for $iter \in LTimes$ ***do***
    Randomly generated $j \in [0, n]$,and a[j]=1-a[j]
    b=*RepairOperator*(LD, a)
    If $V(b) > V(a)$
      a=b

function *main*:
  Randomly generate A, and repair A
  Record the current best solution a^{max}
  while Current evaluation times$< ETimes$ ***do***
    Perform crossover, mutation, selection on A
    for $a \in A$ ***do***
      Update A using *HybridGreedyOperator*
      Update A using *LocalSearchOperator*
      Update a^{max}
  return a^{max}

V. Performance Evaluation

In this section, we evaluate the performance of the proposed scheme through simulation experiments. First, the simulation configuration and data set are given. Then we present three existing algorithms as the comparison. Final, we give the experimental results and analysis.

5.1 Simulation configuration

In our simulation, three cache servers are connected to the source server, and each cache server can provide video access to 200 users at the same time. The bandwidth between the source server and the cache server is set to B=10Mb. We refer to the literature [32] for setting the parameters of the video. The source server provides 500 videos with 4 bitrate versions (64,128,512,1024kps), for a total of 2000 different video files. The length of each video is randomly generated and ranges from 1-60 minutes. Many works provide references in caching capacity and processing capacity of cache servers. In [39,40], to more visually represent the efficiency of the cache server, the caching capacity percentage was used to express the caching capacity, which is the ratio of the cache space size to the space size required to cache all video files. The default value of cache capacity percentage is set as 10%, which could range from 0% to 20%. While the processing capacity will range from 1Mbps to 20Mbps, the default value being 10Mbps.

5.2 Dataset

A real dataset obtained through the Youku open API is provided in [24], which tracks the daily visits to each newly

uploaded video on the Youku website over a 30-day period. In order to make the dataset conform to our caching model, we do some processing on the original dataset. First, most of the videos uploaded by users are relatively cold, the number of visits in 30 days does not even reach 10. Besides, some videos are officially promoted by Youku website, which has few value for using its visits to judge the popularity of the videos. Therefore, we have streamlined the dataset and selected some videos that can reflect the popularity trend. Second, the dataset does not distinguish between visits to different bitrate versions of the same video, so we performed an artificial division. The literature [41] states that the accesses of different versions follow a normal distribution, while it is also indicated in [1] that Standard-Definition video will account for two-thirds of global IP Video traffic by 2022. For implicity, we set the percentage of 4 bitrate versions as 0.1, 0.1, 0.6, 0.2 respectively. Finally, we adjust the value of visits. In our calculation model, if the value of visits is large, it may lead to an excessively large value for popularity prediction. Thus we reduce the value of visits for all videos in the ratio of 10,000:1, i.e, for every 10,000 visits, it is recorded as one visit value. Based on the above processing, we obtain a simulated dataset of users' requests, including video requests made by each user over 30 consecutive time periods.

5.3 Algorithms for comparison

We compare the performance of PPDA with the following three algorithms.

(1) LRU (last recently used): LRU is a traditional caching strategy that replaces the least frequently requested files with the most recently requested files.

(2) FIFO (first in first out): FIFO replaces the earliest cached content with the new received request content, when there is not enough cache space.

(3) PVCS (popularity based and version aware caching scheme): PVCS is proposed in [32], which uses video popularity for caching strategy decisions. However, PVCS just use the linear popularity adjustment method, which is mentioned in section IV, and it does take the global popularity into consideration.

5.4 Experimental results and analysis

The results of experiments are presented in three parts and we utilize two metrics to evaluate these algorithms, which are the hit ratio and the average delay savings. The hit ratio is the value of requests served by cache sever as a percentage of all requests.

5.4.1 Performance comparison in different time periods

Fig.4 and Fig.5 show the hit ratio and average delay savings achieved by the four algorithms at each time period when the caching capacity and processing capacity are taken as default values, respectively. As shown in Fig.4, the volatility of the cache hit ratio of the four algorithms is high, which indicates that the popularity of the videos changes significantly with the changing user preferences, and none of the four algorithms can guarantee a better result in every time period. The standard deviation of their hit ratios are calculated separately and the results are 8.9, 7.8, 6.8 and 5.8, respectively. PPDA has the highest standard deviation, which indicates that the accuracy of our prediction of video popularity is unstable. A higher accuracy will lead to higher hit ratio. Otherwise, the hit ratio will be significantly lower. However, analyzing the overall results, PPDA still outperforms other three algorithms. Fig.5 shows that PPDA has the highest average delay savings in most of the time periods. Only in few time periods, its performance is relatively poor. We analyze that the cause of this phenomenon may be the deviation in predicting popularity. Comparing the other three algorithms, it is found that PVCS is generally better, while LRU is relatively worse.

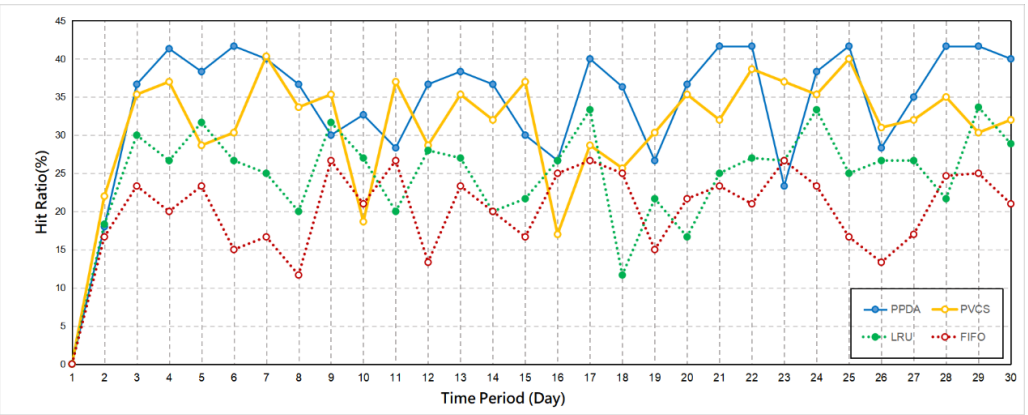

Fig 4: Performance comparison of hit ratio in different time periods

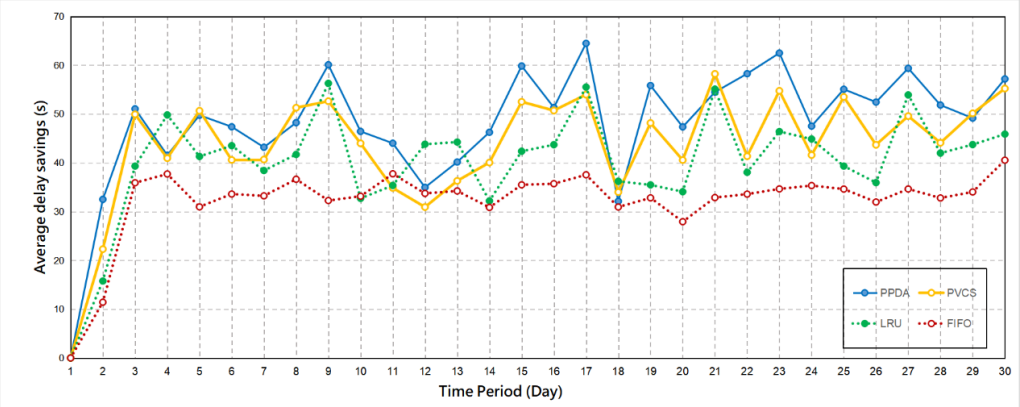

Fig 5: Performance comparison of average delay savings in different time periods

5.4.2 Performance comparison with different cache capacities

We analyze the average hit ratio and average delay savings of each algorithm when the caching capacity of the cache server is increasing. As shown in Fig.6 and Fig.7, the hit ratio of each algorithm increases with the cache capacity, and the improvement of PPDA is the most obvious. In particular, the improvement in hit ratio and delay savings of PPDA is great in the phase with weak cache capacity. This not only reflects that our PPDA can obtain higher cache revenue, but also indicates that our prediction method for video popularity achieves better results. However, as the cache capacity keeps increasing, the improvement in the effectiveness of each algorithm becomes smaller. It may be that our algorithm has already cached all the currently popular videos when the caching capacity is large enough and the extra cache space is used to cache those unpopular contents, thus the improvement in hit ratio and delay savings will not be obvious.

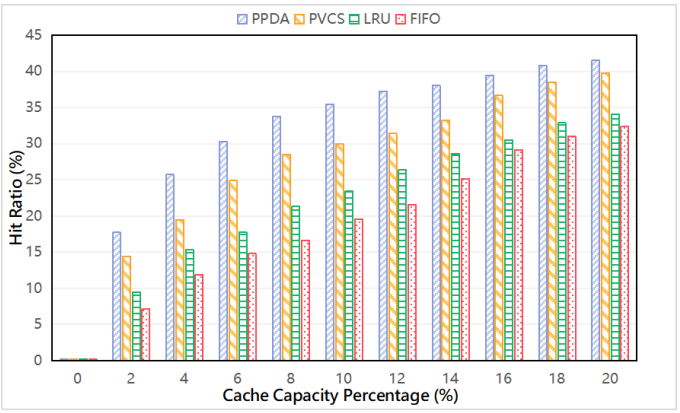

Fig 6: Performance comparison of hit ratio with different cache capacities

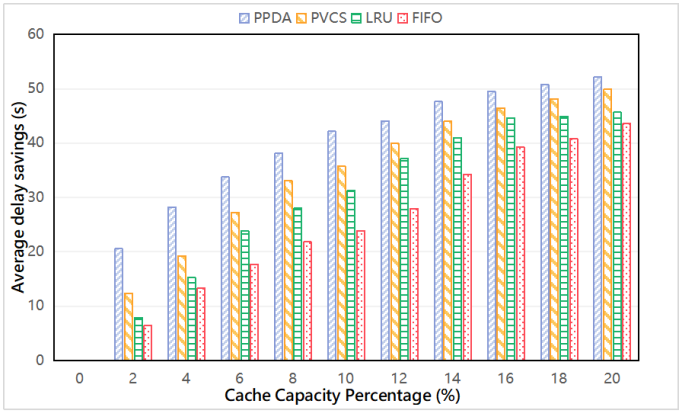

Fig 7: Performance comparison of average delay savings with different cache capacities

5.4.3 Performance comparison with different processing capacities

Fig.8 and Fig.9 show the average hit ratio and average delay saving performance of each algorithm when the processing capacity of the cache server is constantly varying. It is found that the increase in processing capacity did not have a significant impact on hit ratio and average delay. The difference between the highest and lowest values of hit ratio is less than 5%, and that of delay savings is less than 10s. We consider the following possible reasons for this result. First, FIFO and LRU select cached content simply based on the time sequence and frequency of request arrivals, so the increase in cache server processing power does not have a significant impact on them. Second, as the processing power continues to increase, the transcoding delay decreases significantly, which means that the cost of transcoding is negligible when calculating the cache revenue. Then the algorithm has a tendency to cache only the highest bitrate version file of each video when formulating the caching revenue. This leads to a smaller improvement in hit ratio and delay savings. Finally, in our simulation, the percentage of different bitrate versions is set to be fixed, which means that the number of requested videos that need to be transcoded keeps low. This leads to a limited impact of transcoding capacity improvement on the final hit ratio and delay savings.

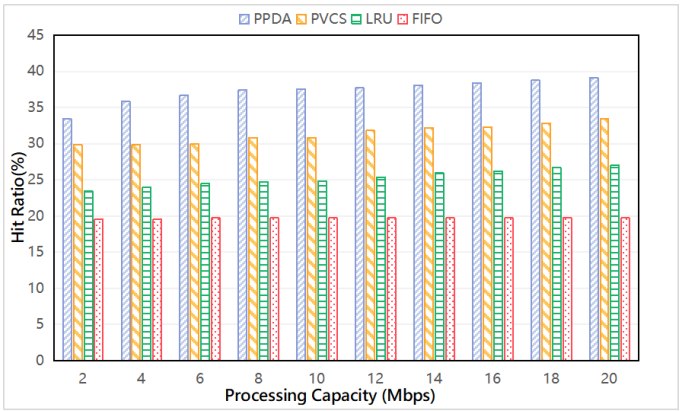

Fig 8: Performance comparison of hit ratio with different processing capacities

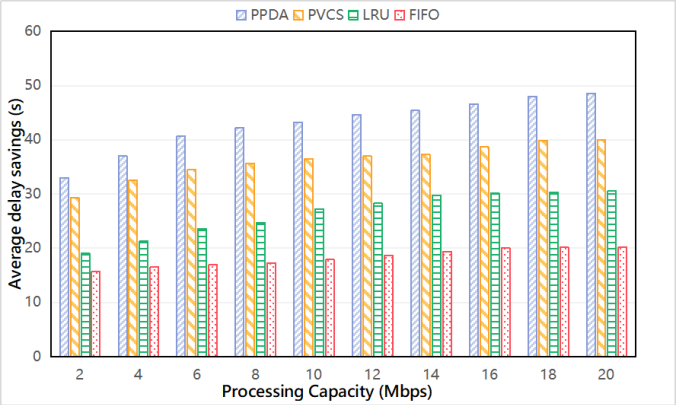

Fig 9: Performance comparison of average delay savings with different processing capacities

VI. Conclusion

In this paper, the problem of caching strategy in dynamic scenarios is studied. A dynamic caching model is designed firstly, which can help us analyze how caching strategy is decided in consecutive time periods. Then, a new popularity prediction method is proposed, in which we can investigate both global popularity and local popularity. Finally, an improved genetic algorithm is developed to solve the optimal caching strategy under each time period. Through simulation experiments, our model can achieve better results compared to other existing algorithms. In the future, our model deserves improvement in two aspects. On the one hand, the accuracy of the popularity prediction algorithm still needs to be enhanced. On the other hand, the hybrid greedy based genetic algorithm can be further improved.

References

[1] "Cisco visual networking index: Forecast and trends, 2017–2022 white paper." Website, 2019. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html.

[2] A. Vakali, G. Pallis, "Content delivery networks: status and trends," IEEE Internet Computing, vol. 7, no. 6, pp. 68-74, 2003.

[3] H. Schwarz, D. Marpe, T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," IEEE Transactions on Circuits and Systems for Video Technology, vol. 17, no. 9, pp. 1103-1120, 2007.

[4] F. Bossen, B. Bross, K. Suhring, D. Flynn, "Hevc complexity and implementation analysis," IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 12, pp. 1685-1696, 2012.

[5] F. Wang, J. Liu, M. Chen, "Calms: Cloud-assisted live media streaming for globalized demands with time/region diversities," in 2012 Proceedings IEEE INFOCOM, pp. 199-207, 2012.

[6] J. Zhu, D.S. Chan, M.S. Prabhu, P. Natarajan, H. Hu, F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 320-323, 2013.

[7] M. Wang, P.P. Jayaraman, R. Ranjan, K. Mitra, M. Zhang, Z.E. Li, S. Khan, M. Pathan, D. Georgeakopoulos, An Overview of Cloud Based Content Delivery Networks: Research Dimensions and State-of-the-Art, vol. 9070, pp. 131-158. 03 2015.

[8] D. Tuncer, V. Sourlas, M. Charalambides, M. Claeys, J. Famaey, G. Pavlou, F. De Turck, "Scalable cache management for isp-operated content delivery services," IEEE Journal on Selected Areas in Communications, vol. 34, no. 8, pp. 2063-2076, 2016.

[9] H. Yan, J. Liu, Y. Li, D. Jin, S. Chen, "Spatial popularity and similarity of watching videos in large-scale urban environment," IEEE Transactions on Network and Service Management, vol. 15, no. 2, pp. 797-810, 2018.

[10] K. Poularakis, G. Iosifidis, A. Argyriou, I. Koutsopoulos, L. Tassiulas, "Caching and operator

cooperation policies for layered video content delivery," in IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pp. 1-9, 2016.

[11] A. Tran, N. Dao, S. Cho, "Bitrate adaptation for video streaming services in edge caching systems," IEEE Access, vol. 8, pp. 135844-135852, 2020.

[12] S. Han, H. Su, C. Yang, A.F. Molisch, "Proactive edge caching for video on demand with quality adaptation," IEEE Transactions on Wireless Communications, vol. 19, no. 1, pp. 218-234, 2020.

[13] L. Li, D. Shi, R. Hou, R. Chen, B. Lin, M. Pan, "Energy-efficient proactive caching for adaptive video streaming via data-driven optimization," IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5549-5561, 2020.

[14] Y. Wei, F. R. Yu, M. Song, Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled iot using natural actor–critic deep reinforcement learning," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2061-2073, 2019.

[15] S. Borst, V. Gupta, A. Walid, "Distributed caching algorithms for content distribution networks," in 2010 Proceedings IEEE INFOCOM, pp. 1-9, 2010.

[16] T.X. Tran and D. Pompili, "Adaptive bitrate video caching and processing in mobile-edge computing networks," IEEE Transactions on Mobile Computing, vol. 18, no. 9, pp. 1965-1978, 2019.

[17] Y. Wang, Y. Zhang, M. Sheng, K. Guo, "On the interaction of video caching and retrieving in multi-server mobile-edge computing systems," IEEE Wireless Communications Letters, vol. 8, no. 5, pp. 1444-1447, 2019.

[18] L. Cherkasova, M. Gupta, "Analysis of enterprise media server workloads: access patterns, locality, content evolution, and rates of change," IEEE/ACM Transactions on Networking, vol. 12, no. 5, pp. 781-794, 2004.

[19] M. Cha, H. Kwak, P. Rodriguez, Y.Y. Ahn, S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement, IMC '07, (New York, NY, USA), p. 1-14, Association for Computing Machinery, 2007.

[20] D. Ciullo, V. Martina, M. Garetto, E. Leonardi, "How much can large-scale video-on-demand benefit from users' cooperation?" IEEE/ACM Transactions on Networking, vol. 23, no. 6, pp. 1846-1861, 2015.

[21] J. Famaey, T. Wauters, F. De Turck, "On the merits of popularity prediction in multimedia content caching," in 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pp. 17-24, 2011.

[22] S. Wang, Z. Yan, X. Hu, P.S. Yu, Z. Li, "Burst time prediction in cascades," in Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI'15, p. 325-331, AAAI Press, 2015.

[23] J. Wu, Y. Zhou, D.M. Chiu, Z. Zhu, "Modeling dynamics of online video popularity," IEEE Transactions on Multimedia, vol. 18, no. 9, pp. 1882-1895, 2016.

[24] C. Li, J. Liu, S. Ouyang, "Analysis and prediction of content popularity for online video service: a youku case study," China Communications, vol. 13, no. 12, pp. 216-233, 2016.

[25] B. Su, Y. Wang, Y. Liu, "A new popularity prediction model based on lifetime forecast of online videos," in 2016 IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC), pp. 376-380, 2016.

[26] Q. Kong, W. Mao, G. Chen, D. Zeng, "Exploring trends and patterns of popularity stage evolution in social media," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 50, no. 10, pp. 3817-3827, 2020.

[27] M. Claeys, N. Bouten, D. De Vleeschauwer, W. Van Leekwijck, S. Latré, F. De Turck, "Cooperative announcement-based caching for video-on-demand streaming," IEEE Transactions on Network and Service Management, vol. 13, no. 2, pp. 308-321, 2016.

[28] C. Richier, R. Elazouzi, T. Jimenez, E. Altman, G. Linares, "Predicting popularity dynamics of online contents using data filtering methods," in 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 31-38, 2016.

[29] M. Nguyen, D.H. Le, T. Nakajima, M. Yoshimi, N. Thoai, "Attention-based neural network: A novel

approach for predicting the popularity of online content," in 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 329-336, 2019.

[30] N.Kumar, S.N. Swain, C. Siva Ram Murthy, "A novel distributed q-learning based resource reservation framework for facilitating d2d content access requests in lte-a networks," IEEE Transactions on Network and Service Management, vol. 15, no. 2, pp. 718-731, 2018.

[31] Z. Zhang, C. Lung, M. St-Hilaire, I. Lambadaris, "An sdn-based caching decision policy for video caching in information-centric networking," IEEE Transactions on Multimedia, vol. 22, no. 4, pp. 1069-1083, 2020.

[32] H. Zhao, Q. Wang, J. Wang, B. Wan, Z. Wu, "Popularity-based and version-aware caching scheme at edge servers for multi-version vod systems," IEEE Transactions on Circuits and Systems for Video Technology, vol. 31, no. 3, pp. 1234-1248, 2021.

[33] K. Shanmugam, N. Golrezaei, A.G. Dimakis, A.F. Molisch, G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," IEEE Transactions on Information Theory, vol. 59, no. 12, pp. 8402-8413, 2013.

[34] N. Kamiyama, Y. Nakano, K. Shiomoto, "Cache replacement based on distance to origin servers," IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 848-859, 2016.

[35] Y. Zhou, L. Chen, C. Yang, D.M. Chiu, "Video popularity dynamics and its implication for replication," IEEE Transactions on Multimedia, vol. 17, no. 8, pp. 1273-1285, 2015.

[36] C. Wang, X. Xin, J. Shang, "When to make a topic popular again? a temporal model for topic rehotting prediction in online social networks," IEEE Transactions on Signal and Information Processing over Networks, vol. 4, no. 1, pp. 202-216, 2018.

[37] K. Choi, D. Jang, S. Kang, J. Lee, T. Chung, H. Kim, "Hybrid algorithm combing genetic algorithm with evolution strategy for antenna design," IEEE Transactions on Magnetics, vol. 52, no. 3, pp. 1-4, 2016.

[38] Z.J. Lee, S.F. Su, C.Y. Lee, "Efficiently solving general weapon-target assignment problem by genetic algorithms with greedy eugenics," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 33, no. 1, pp. 113-121, 2003.

[39] W. Li, S. M.A. Oteafy, H.S. Hassanein, "Streamcache: Popularity-based caching for adaptive streaming over information-centric networks," in 2016 IEEE International Conference on Communications (ICC), pp. 1-6, 2016.

[40] S. Müller, O. Atan, M. van der Schaar, A. Klein, "Context-aware proactive content caching with service differentiation in wireless networks," IEEE Transactions on Wireless Communications, vol. 16, no. 2, pp. 1024-1036, 2017.

[41] B. Shen, S.J. Lee, S. Basu, "Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks," IEEE Transactions on Multimedia, vol. 6, no. 2, pp. 375-386, 2004.