Lifecycle-Aware Online Video Caching

Tong Li, *Student Member, IEEE,* Tristan Braud,
Yong Li, *Senior Member, IEEE,* and Pan Hui, *Fellow, IEEE*

Abstract—The current explosion of video traffic compels service providers to deploy caches at edge networks. Nowadays, most caching systems store data with a high programming voltage corresponding to the largest possible 'expiry date', typically on the order of years, which maximizes the cache damage. However, popular videos rarely exhibit lifecycles longer than a couple of months. Consequently, the programming voltage can instead be adapted to fit the lifecycle and mitigate the cache damage accordingly.
In this paper, we propose LiA-cache, a Lifecycle-Aware caching policy for online videos. LiA-cache finds both near-optimal caching retention times and cache eviction policies by optimizing traffic delivery cost and cache damage cost conjointly. We first investigate temporal patterns of video access from a real-world dataset covering 10 million online videos collected by one of the largest mobile network operators in the world. We next cluster the videos based on their access lifecycles and integrate the clustering into a general model of the caching system. Specifically, LiA-cache analyzes videos and caches them depending on their cluster label. Compared to other popular policies in real-world scenarios, LiA-cache can reduce cache damage up to 90%, while keeping a cache hit ratio close to a policy purely relying on video popularity.

Index Terms—Video caching, Video lifecycles, Caching policy, Cache damage, Edge networks.

1 Introduction

1.1 Motivation

Every second, thousands of videos are transmitted across mobile networks from servers to enormous numbers of mobile devices. According to Cisco, video traffic accounted for 75% of data traffic in 2017 and will reach 82% by 2022 [1]. This soaring demand for online video services is pushing network operators and service providers to bring content caches closer to the edge of the network, e.g., base stations, to lessen the traffic burden in the core network while reducing content delivery latency [2], [3].

Nowadays, although most network caches rely on flash memory [4], [5], the adoption of flash memory is still hindered by the high rate of wear out called cache damage, which is directly proportional to the programmed data *retention time* [6], [7]. The relationship between data retention time and cache damage can be briefly described as follows. Flash memory has multiple flash cells [8]. Data is stored in flash memory by programming the voltage of each flash cell higher than a threshold. The threshold of voltage is called the programming voltage [9]. Additionally, before a flash cell is programmed, the existing data has to be erased using an equal but opposite programming voltage. Each program/erase cycle will harm flash cells and reduce the flash lifetime [10]. We refer to the flash cell damage caused by program/erase cycles as cache damage. Generally, a high programming voltage guarantees that flash cells will retain data for a long duration, known as the data retention time. Current flash technology writes all files by applying a high programming voltage to ensure a specific data retention time, typically from one to ten years. However, such a high programming voltage will also cause severe cache damage and significantly reduce the memory lifetime [8]–[12].

On the other hand, as many new videos appear every day, the popularity of a given video decays quickly over time [13]. The average lifecycle of such videos is on the day or month level, rather than the year level. Thus, programming the cache at voltages high enough to store all video content for years is wasteful. Moreover, if flash cells are programmed with a high positive voltage, the corresponding erasing operation requires an equally high negative voltage. Hence, each program/erase cycle will cause more extensive damage to the cache and decrease the life expectancy of the memory. Even though the cost of memory itself is affordable, changing the caches scattered around the world becomes costly, when considering transportation, human labor, *etc*. A practical approach to solve this problem is to set a dynamic programmed retention time to reduce cache damage by leveraging video lifecycles.

The content retention time in caches also profoundly affects the performance of the cache system. When a user requests content, i.e., an online video, the service provider first checks whether it is available in the edge cache. If so, the edge cache transmits the content; otherwise, a cache miss happens, and the content has to be streamed from a remote data center, causing traffic delivery cost [14]. If the retention time of content is set too short for the sake of lower cache damage, many cache misses may occur even though the cache is not fully occupied. When the cached content expires, i.e., exceeds its programmed retention time, the content is 'invalid' because its integrity cannot be guaranteed due to the low residual voltage of the flash cells. In this case, severe traffic delivery and high network latency

- *T. Li and T. Braud are with the System and Media Laboratory (SyMLab), Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong.*
- *Y. Li is with Beijing National Research Center for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing, China.*
  E-mail: liyong07@tsinghua.edu.cn
- *P. Hui is with the System and Media Laboratory (SyMLab), Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong. He is also with the Department of Computer Science, University of Helsinki, Helsinki, Finland.*
  E-mail: panhui@cse.ust.hk

will arise between end devices and remote data centers from the large number of cache misses. Consequently, the optimal data retention time is the result of a trade-off between traffic delivery cost and cache damage cost.

Currently, most cache policies are user-oriented and focus on improving the user experience. For instance, the conventional cache policies, e.g., random (RND), first in first out (FIFO), and least recently used (LRU), aim at improving the cache hit ratio. State-of-the-art cache policies take popularity features into account [15]–[20], and the cached content has a high probability of being requested by users in the future, which reduces transmission delays. However, the existing research lacks a service provider viewpoint and does not reduce operating expenses by lowering the cache damage cost.

1.2 Methodology and Contributions

In this paper, we investigate the optimal cache policy from both service providers' and mobile users' perspectives to jointly minimize cache damage cost and traffic delivery cost by exploring the lifecycles of online videos. In summary, we focus on three questions. Do online videos have lifecycles? If so, what kinds of lifecycle patterns do they have? Can their lifecycle patterns provide useful information to improve the cache system and reduce cache damage?

To explore the lifecycles of online videos, we analyze a large-scale real-world dataset of online video access over two months from one of the largest network operators in the world. The dataset consists of 269 million viewing requests from 9,517,339 users watching over 9,055,188 videos. We first discover two kinds of temporal video access patterns, i.e., pulse and noise shapes, and define the concept of the lifecycles for online videos of the pulse shape. To extract the typical lifecycle of the pulse shape videos, we use the K-Medoids clustering algorithm to separate the content into two clusters: long-lived videos of *one-month* lifecycle and short-lived videos of *one-week* lifecycle. We next obtain the typical temporal pattern for each cluster.

We then model the cache system and formulate an optimization problem to jointly minimize traffic delivery cost and cache damage cost, considering nightly content eviction operations. The optimization problem is an integer programming problem. To solve this NP-Hard problem, we separate it into two sub-problems: retention time optimization and cache eviction policy design. We then devise optimal algorithms for the two sub-problems with the assistance of video lifecycle analysis and prediction.

The contributions of our work can be summarized as follows.

- *Lifecycle modeling for video contents.* We analyze the access patterns of various videos and extract two video clusters with different lifecycles, i.e., long-lived videos and short-lived videos. For these two clusters, we build an exponential temporal pattern and a power-law temporal pattern respectively and devise their closed-form expressions.
- *LiA-cache policy*. We propose a novel lifecycle-aware cache policy that jointly minimizes cache damage cost and traffic delivery cost by leveraging our modeled lifecycle patterns. To the best of our knowledge, this is the first study to introduce the consideration of cache damage into the cache system.
- *Extensive evaluations via a large-scale real-world dataset*. We evaluate the effectiveness of our LiA-cache policy in comparison with other popular policies, like FIFO, RND, LRU, and other policies relying on popularity patterns. We demonstrate that LiA-cache policy can decrease the cache damage cost by 30 times, i.e., extending cache lifetime 30 times, while providing close to an optimal cache hit ratio.

TABLE 1
Table of notations.

Notation	Description
$\mathbf{T}$	set of time slots
$\mathbf{N}$	set of contents
Ψ_n^{del}	traffic delivery cost caused by content n
Ψ_n^{dam}	cache damage cost caused by content n
$P_n(t)$	daily view temporal function of content n
$\delta_n(t)$	indicating whether content n is cached at time slot t
ε	weight of cache damage cost
t_n	time slot when content n starts to be cached
τ_n	programmed cache retention time of content n
B	cache capacity
T_n	time slot when $P_n(t)$ is at its maximum value
P_n^{MAX}	maximum value of $P_n(t)$
T_C	typical lifecycle for the contents in cluster $\mathbf{C}$
Δp	threshold of daily views to determine video lifecycles

The rest of this paper is organized as follows. In section 2, we focus on identifying and extracting the typical lifecycle patterns of online videos and describe our real-world dataset in detail. We then present the cache system model in section 3, as well as the optimization problem formulation. In section 4, we give a two-step heuristic cache policy relying on extracted lifecycle patterns. In section 5, we evaluate the caching performance of the proposed LiA-cache policy by comparing it with other popular policies. Before presenting our conclusions, we discuss previous works related to both cache policies and cache damage.

We summarize the notation and the corresponding description of the main symbols in Table 1.

2 Temporal Patterns of Online Videos

In this section, we empirically characterize online video lifecycles by leveraging a large-scale real-world dataset. We first study the temporal patterns and notice that the daily views of a video over time exhibit either a pulse or noise shape. We next investigate pulse shape patterns and measure the similarity of such patterns by lifecycles instead of popularity. We finally extract the typical lifecycles of online videos using the K-Medoids clustering algorithm and model the obtained temporal patterns.

2.1 Dataset Overview

We extract the temporal patterns from a dataset of online videos collected by one of the largest network operators (fixed and mobile) in China. The dataset consists of online video viewing requests from November 1st to December

TABLE 2
Sample of collected real-world dataset

Time	Video ID	User ID
2014-11-01 00:00:00	6b67093ebe07a3312e1a2d6ee154194ccd081	fb1bd0cd99f1f3ff037a314e5252bf79df8dd
2014-11-01 00:00:01	26e7f3b71721bf6a31f86b80ff4ed12072bd6e72	84deafc6428de551205f5049f1375eb24e70edc9
2014-11-01 00:00:01	8e7d3cec3dab2f0e687d3c5dc67a4fe0c1b7b7	6692d1e9364a5afa175d42316b168f023c0ebf

31st, 2014. Deep packet inspection appliances at the gateways of the operator collect video viewing requests to the most extensive video content providers in China, such as Youku, Tencent, Iqiyi, *etc*. In total, our dataset contains 269 million requests from 9,517,339 devices/users who watched over 9,055,188 unique online videos. The operator has 85% of the market share for network access in mainland China, and the dataset covers most major video providers. Thus, our data provide a highly comprehensive representation of the video viewing behavior on the Internet in China. A sample of the collected real-world dataset is shown in Table 2.

We collect the resulting records approximately every half-hour. Each record consists of:

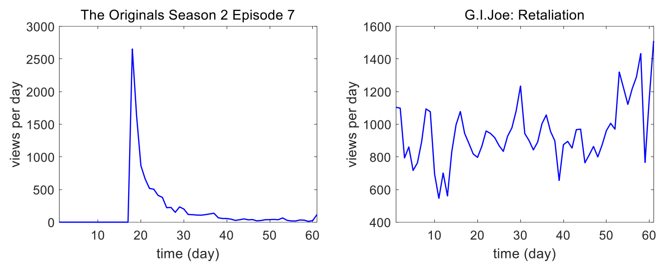
- *Time stamp*: the time at which the viewing request was issued.
- *Video content ID*: the unique identity of online video content. The operator provides a hash table to map the ID to its name.
- *User ID*: the unique identity of the user device. The operator produces user IDs based on device-level (not IP) information and anonymizes user IDs for security and privacy reasons.

2.2 Temporal Patterns of Different Shapes

We begin our analysis by visualizing the daily views of content. Similar to prior work [21], [22], we discover two kinds of temporal patterns: pulse shape patterns and noise shape patterns. Fig. 1 shows the daily views of the series episode 'The Originals Season 2 Episode 7' and the movie 'G.I. Joe: Retaliation' in our dataset. The temporal pattern of the first video belongs to the pulse shape class, while the second one belongs to the noise shape class.

For the pulse shape, the time-varying curves can be split into two phases, a sudden burst of views and a relaxation tail, making it similar to a pulse signal. Regarding the noise shape, content requests are primarily driven by fluctuations instead of bursts of activities. The time series can be viewed as a noise process. Most of the content with the noise pattern is either of overall low popularity or classic videos. Users access this content irregularly over time, and therefore, we cannot extract clear patterns.

Since noise shape patterns are irregular and do not have clear lifecycles, in this work, we only focus on pulse shape patterns and explore their lifecycles. Note that although noise-shaped videos do not have lifecycles, existing popularity-based cache policies [18], [20] can still be applied to them. In the data cleaning process, we first remove content with less than 200 cumulative views over the two months. After this step, 20,582 popular videos remain. This reduced dataset contains both noise shape patterns and pulse shape patterns. To filter out the videos with noise shape patterns, we employ an existing metric proposed by



(a) Pulse shape (b) Noise shape

Fig. 1. Examples of the two classes of video temporal patterns. On the left, a video that displays a pulse shape after the content releases. On the right, an older video with fairly random accesses over time.

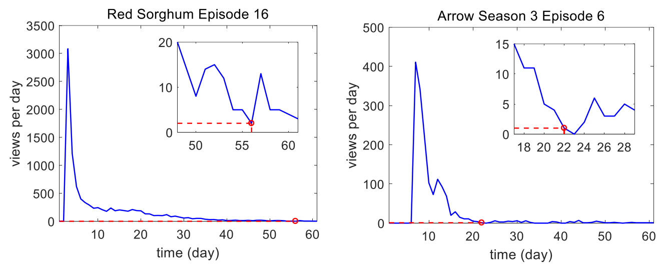


Fig. 2. Two examples of pulse shape patterns. Both videos display a similar sudden increase in popularity, but two different relaxation parts.

Crane *et al.* [21] using peak fraction. For a given video the peak fraction is the fraction of views observed on the peak day compared to the total cumulative views. When the peak fraction is higher than 0.2, the temporal pattern is regarded as pulse-shaped. Therefore, we next remove the videos, whose peak fraction is less than 0.2. We thus obtain 17,572 popular videos with pulse shape patterns. We also remove the content still active/popular on the last day of the dataset, i.e., December 31st, 2014. After all these data cleaning operations, our dataset contains 16,433 online videos. Moreover, we note that in terms of our real-world dataset, noise-shaped videos account for only 14% of the 20,582 popular videos, which is a clear minority. Therefore, although we ignore noise-shaped videos, our work is still of significant value for online video cache systems.

2.3 Measurement of Temporal Patterns' Similarity

Fig. 2 shows two examples of pulse shape temporal variation in the number of daily views. During the burst part, the number of daily views increases suddenly and rises to the maximum value within a few days, i.e., one or two days. Due to this steep increase, it is difficult to differentiate videos purely from the burst part. In other words, the relaxation part contains most of the unique features of the curve. Fig. 2 shows that two key features characterize the relaxation part: the initial value, i.e., the maximum number of daily views, and the lifecycle.

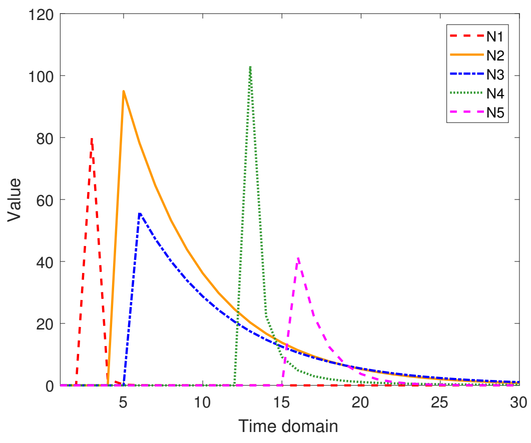


Fig. 3. Examples of time series with different lifecycles: various peak value and relaxation length.

Definition 1 (Lifecycle of Online Videos)**.** Given a small threshold Δp, a video is inactive when the number of daily views is lower than Δp. The lifecycle of a video is the period from the time slot when the number of daily views is at its maximum value to the first time slot when the video becomes inactive. Note that Δp should be an integer because it represents the number of daily views. To fully capture the relaxation part of temporal patterns, in practice, we usually set Δp as 1.

In Fig. 2, for $\Delta p = 1$, the lifecycle of videos 'Red Sorghum Episode 16' and 'Arrow Season 3 Episode 6' are 54 days and 16 days, respectively.

Currently, existing works on videos' temporal patterns mainly focus on the popularity of videos [21], [23], [24] rather than their lifecycles. In this study, we show that the lifecycle and the peak value contain enough information to estimate the temporal curve of a video.

In practice, the peak value of daily views can be determined by two methods, i.e., a proactive method and a reactive method. The proactive method means using a prediction algorithm to infer the value in advance in terms of video's attributes and has been investigated in many video popularity prediction papers. On the other hand, the reactive method is more straightforward. As the peak value is a maximum, as soon as the number of daily video views decreases, we can estimate the prior value is the peak value. However, in this case there is a one-time slot latency to determine the peak value. Because determining the peak value is not the main research problem in this work, in the following analysis, we instead concentrate on investigating the lifecycles of online videos and measuring the similarity of the lifecycles.

Various online videos have different lifecycles. However, characterizing the exact lifecycles for individual videos is difficult. Therefore, we cluster videos of similar lifecycles together to obtain the typical lifecycles. Next, we can use these typical lifecycles to approximate the exact lifecycles of videos. However, as shown in Fig. 2, temporal patterns with totally different lifecycles may appear similar. Hence, we perform the analysis in the frequency domain instead of the time domain, thanks to the Fast Fourier Transformation (FFT), which has successfully been used, for example, to analyze pulse waves in medicine [25] to differentiate the cardiac cycle.

Fig. 3 and Fig. 4 show the relationship between the lifecycle of temporal popularity curves and their Fourier Transform. Given $\Delta p = 1$, the lifecycle of $N1$ is 2 days, the lifecycle of both $N2$ and $N3$ is 21 days, and the lifecycle of $N4$ and $N5$ is 6 days. In the frequency domain, we identify three different shapes that respectively correspond to $N1$, $N2$ and $N3$, and $N4$ and $N5$. Also, we notice that the frequency domain representation of video views is 'U'-shaped for longer lifecycles and tends to a 'V'-shaped for shorter lifecycles.

In this way, we decompose the one-dimensional clustering problem in the time domain into a two-dimensional clustering problem in the frequency domain. We then cluster the lifecycle patterns with similar shapes in the frequency domain instead of the time domain.

2.4 Shape-based Clustering

As we intend to cluster videos of similar shapes, we ignore differences in absolute volumes. In other words, if two curves have a similar shape but different volumes, we still consider them to be similar and put them in the same cluster.

Before clustering, we first normalize and scale curves using the min-max normalization method. Given a series $\boldsymbol{x} = \{x_1, x_2, ..., x_i\}$, the normalization of $\boldsymbol{x}$ is given as follows,

$$x_i = \frac{x_i - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}, \tag{1}$$

where $\min(\boldsymbol{x})$ and $\max(\boldsymbol{x})$ are the minimum and maximum values in series $\boldsymbol{x}$, respectively.

Algorithm 1 Modified K-Medoids Clustering Algorithm

```
Input: Daily view series of video $n$, i.e. $P_n,\ n = 1, 2, ..., N$.
    Number of clusters $K$. Initial cluster assignments $\boldsymbol{C} = C_1, C_2, ..., C_k$.
1: $F_n = \|FFT(P_n)\|, n = 1, 2, ..., N$, where $\|\cdot\|$ is the $l_2$ norm.
2: Normalize $F_n,\ n = 1, 2, ..., N$.
3: Randomly select initial cluster medoids $\boldsymbol{\mu} = \{\mu_1, \mu_2, ..., \mu_k\}$ from $F_n,\ n = 1, 2, ..., N$.
4: repeat
5:   $\hat{\boldsymbol{\mu}} \leftarrow \boldsymbol{\mu}$.
6:   for $n = 1, 2, ..., N$ do
7:       $d(F_n, \mu_i) = \|F_n - \mu_i\|$, where $\|\cdot\|$ is the $l_2$ norm.
8:       $\lambda_n \leftarrow \arg \min_{i \in \{1,2,...,k\}} d(F_n, \mu_i)$.
9:       $C_{\lambda_n} = C_{\lambda_n} \cup \{F_n\}$.
10:  end for
11:  for $i = 1, 2, ..., k$ do
12:      $\mu_i \leftarrow \arg \min_{\mu_i \in C_i} \sum_{F_n \in C_i} d(F_n, \mu_i)$
13:  end for
14: until $\hat{\boldsymbol{\mu}} = \boldsymbol{\mu}$
Output: $\boldsymbol{C}$, $\boldsymbol{\mu}$
```

For the actual clustering step we use the K-Medoids clustering algorithm. Compared with K-means, K-Medoids is more robust to noise as it uses the real data-points in the dataset as medoids instead of the calculated centers [26].

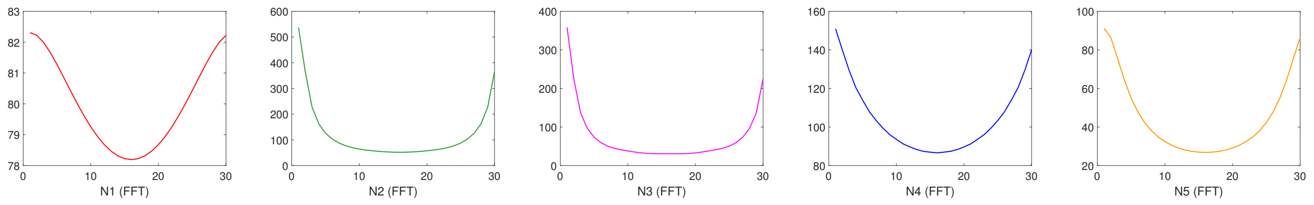


Fig. 4. The corresponding Fourier Transform of the times series in Fig. 3. The longer the lifecycle, the more U-shaped the frequency domain transformation.

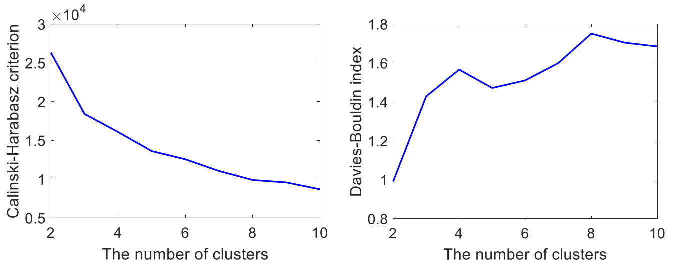


Fig. 5. Clustering quality versus the number of clusters. According to both Calinski-Harabasz criterion (higher is better) and Davies-Bouldin index (lower is better), the optimal clustering has two clusters.

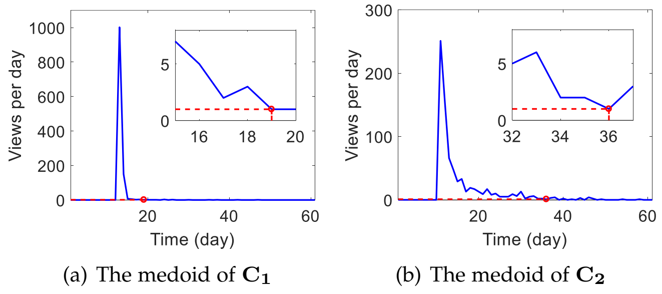


Fig. 6. The daily view curves of medoids. The medoid of $\mathbf{C_1}$ is characterized by a sudden increase and an immediate decrease, while the medoid of $\mathbf{C_2}$ has a longer relaxation part.

TABLE 3
Preference statistics of different functions

The number of preferred videos	$\mathbf{C_1}$	$\mathbf{C_2}$
Exponential function	7493 (78.5%)	1222 (17.8%)
Power-law function	2057 (21.5%)	5661 (82.2%)

Algorithm 1 summarizes the modified K-Medoids clustering algorithm. The algorithm performs clustering in three steps: initialization, assignment of objects to medoids, and updating of medoids. In the *initialization* step, we first take the Fourier Transform of daily view series and perform the min-max normalization. We then choose the medoids randomly. In the *assignment* step, we calculate the matrix of the distances between objects and medoids. We next assign each object to the nearest medoid and obtain the cluster results. In the *update* step, we find the a new (updated) medoid for each cluster, which is the object minimizing the total distance to other objects in the cluster.

2.5 Finding Typical Temporal Patterns

Like all the other variants of K-means, the K-Medoids clustering algorithm requires us to specify the number of clusters in advance. Although determining the most appropriate number of clusters remains an open issue, we can approach the problem empirically by measuring how the quality of clustering varies with the number of clusters in terms of multiple metrics.

We run the clustering algorithm with different numbers of clusters ranging from 2 to 10 and compute the Calinski-Harabasz criterion [27] and the Davies-Bouldin index [28] which evaluate both the intra-cluster similarity and the inter-cluster differences. For the Calinski-Harabasz criterion, the higher the value, the better the clustering. Conversely, the clustering quality decreases with an increase in the Davies-Bouldin index.

Fig. 5 shows the values of the two metrics relative to the number of clusters. Both metrics suggest the optimal number of clusters is two. Therefore, based on the collected large-scale and real-world dataset, we empirically discover two typical lifecycle patterns of online videos.

The daily view curves of the medoids of the two clusters are depicted in Fig. 6. $\mathbf{C_1}$ denotes the set of the first cluster and $\mathbf{C_2}$ denotes the set of the second cluster. Note that the value of the typical lifecycle is related to Δp. Generally, the larger the Δp, the shorter the lifecycle. To fully capture the relaxation part of temporal patterns, in our case, we set Δp as 1. Therefore, the typical lifecycle of cluster $\mathbf{C_1}$, i.e., the lifecycle of the medoid of $\mathbf{C_1}$, is 7 days, $T_{C_1} = 7$. The typical lifecycle of cluster $\mathbf{C_2}$ is 26 days, $T_{C_2} = 26$. Consequently, we discover two typical lifecycles of online videos, averaging around one week and one month, respectively.

According to existing studies [21], [29]–[32], human activities usually follow one of two distributions: an exponential or power-law distribution, which is consistent with the temporal patterns of the discovered medoids. The light tail in the medoid of $\mathbf{C_1}$ is characteristic of an exponential distribution, while the heavy tail in the medoid of $\mathbf{C_2}$ is characteristic of a power-law distribution. Hence, we perform curve fitting for both exponential functions and power-law functions to the online videos and compute the mean squared errors (MSE).

To determine the better distribution function for each video, we define the 'preference' of each video. If the MSE of the exponential function fitting for video n is lower than the power-law function fitting, then the 'preference' of video n is an exponential function and vice versa. We calculate the preference percentages and illustrate the values in Table 3.

In terms of Table 3, 78.5% of the videos in $\mathbf{C_1}$ prefer the exponential function, while 82.2% of videos in $\mathbf{C_2}$ prefer the power-law function. Therefore, for a video in cluster one, i.e., $\mathbf{C_1}$, with a the short lifecycle, its daily view function should be exponential. Alternatively, for a video in cluster two, i.e., $\mathbf{C_2}$, with a long lifecycle, the daily view function should follow a power-law. Indeed, for content with a short lifecycle, the temporal curve is not heavy-tailed, which matches the exponential function and vice-versa.

Consequently, for an online video n, we denote its peak value, i.e., the initial value of the relaxation part, as P_n^{MAX},

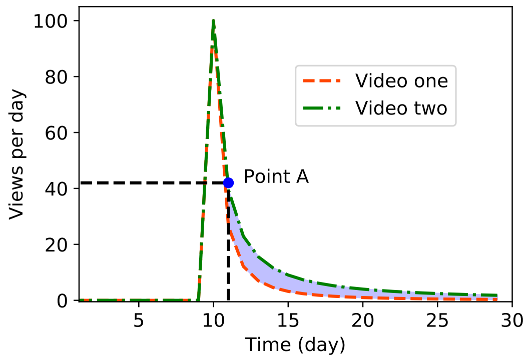


Fig. 7. A simple example to demonstrate how the cache system benefits from the information of videos' lifecycles.

and the time slot when the relaxation starts as T_n. If the video n belongs to cluster one, i.e., $n \in \mathbf{C_1}$, its daily view temporal pattern can be modeled as,

$$P_n(t) = P_n^{MAX} \cdot \exp\left(-\frac{(t-T_n)\cdot \ln\left(P_n^{MAX}/\Delta p\right)}{T_{C_1}}\right), n \in \mathbf{C_1}. \quad (2)$$

On the other hand, if the video n belongs to cluster two, i.e., $n \in \mathbf{C_2}$, its daily view temporal pattern can be modeled as,

$$P_n(t) = P_n^{MAX} \cdot (t - T_n + 1)^{-\frac{\ln\left(P_n^{MAX}/\Delta p\right)}{\ln\left(T_{C_2}+1\right)}}, n \in \mathbf{C_2}. \quad (3)$$

As K-Medoids is a hard clustering algorithm, a video n will belong to either $\mathbf{C_1}$ or $\mathbf{C_2}$. As shown in (2) and (3), in each cluster, we apply the typical lifecycle to approximate the exact lifecycle of the video n. In this study, based on our collected real-world dataset, we discover only two typical lifecycles of online videos. Nonetheless, even when new typical lifecycle patterns do appear, we can still apply a similar clustering approach to model the daily view temporal patterns of videos.

3 Cache System Model and Problem Formulation

To improve the Quality of Experience (QoE) for end-users, network operators and video service providers deploy edge caches to bring content closer to end devices. Since the cache system does not have unlimited capacity, they can only store a fraction of videos at the edge of networks. Therefore, cache operators have to make decisions regarding which videos to cache and how long to keep these videos in memory.

In this section, we define the system model and formulate a joint optimization problem to minimize the traffic delivery cost and cache damage cost subject to the cache capacity. The traffic delivery cost is caused by cache miss and refers to the amount of content which is transmitted from a remote data center instead of edge caches. The cache damage cost is directly proportional to the cache retention time of videos according to literature [6], [7], [10].

We first present a simple example to demonstrate how the cache system benefits from the information of videos' lifecycles. We assume that there are two online videos, i.e., video one and video two. Their daily view curves are shown in Fig.7. Video one and video two achieve the same peak on the tenth day but have different lifecycles. Assuming the cache size is one, if we adopt a popularity-based cache policy, on the tenth day, there is no difference between caching video one or video two because of their identical popularity. If the system caches video one on the tenth day, the cached video one will be evicted by video two at point A, causing extra traffic delivery costs, because the popularity of video two will be higher than video one. However, if we can estimate the lifecycle information in advance, then on the tenth day, we will know that caching video two can save on traffic delivery costs, as shown by the blue shaded area in Fig.7. In this way, the lifecycle information helps to improve the user experience of viewing videos.

It is worth noting that if we predict the popularity of videos instead, we can make a similar caching decision. However, predicting lifecycle and predicting popularity are two different tasks. Popularity prediction is more complicated since it is a regression problem. Also, if we want to make a similar decision, we need to predict the file's popularity over multiple future time slots, which may cause severe error accumulation. Nevertheless, the task of lifecycle prediction is a binary classification problem, thus less difficult. Moreover, lifecycle as a single parameter will not cause the issue of error accumulation for the prediction task.

Next, we start to model the cache system by considering both traffic delivery and cache damage costs. For practicality reasons, we adopt a daily caching procedure and divide the timescale into a set of time slots as follows: $\boldsymbol{T} = \{1, 2, ..., T\}$. We assume that the network consists of N videos and denote the set of videos as $\mathbf{N}$. We simplify the system by estimating these videos to be unit size. Let $P_n(t)$ denote the temporal variation of daily views for video n. We define the traffic delivery cost for video n as,

$$\Psi_n^{del} \triangleq \sum_{t=1}^{T} P_n(t)\left[1 - \delta_n(t)\right], \forall n \in \mathbf{N}, \quad (4)$$

where $\delta_n(t)$ is an 1-0 indicator variable to indicate whether video n is cached at time slot t or not. For an arbitrary video n, each cache miss leads to a unit traffic delivery cost since each video is unit sized.

Let τ_n be the cache retention time of video n. The cache damage cost is directly proportional to τ_n [6], [7], [10]. We can, therefore, define Ψ_n^{dam} as the cache damage cost for caching video n, relative to τ_n,

$$\Psi_n^{dam} \triangleq f(\tau_n), \forall n \in \mathbf{N}, \quad (5)$$

where $f(\cdot)$ is the function of cache damage cost.

Assuming the local cache is storage-limited, we formulate the following optimization problem to minimize the sum of the traffic delivery cost and weighted cache damage cost for all videos,

$$\min_{\delta_n(t)} \sum_{n \in \mathbf{N}} \left(\Psi_n^{del} + \varepsilon \cdot \Psi_n^{dam}\right). \quad (6)$$

s.t.

$$\varepsilon \geq 0, \quad (7)$$

$$\delta_n(t) = \begin{cases} 1 & t \in \{t_n, ..., t_n + \tau_n\} \\ 0 & \text{otherwise} \end{cases}, \forall n \in \mathbf{N}, \quad (8)$$

$$\tau_n \in \{0,1,2,...,T\}, \forall n \in \mathbf{N}, \tag{9}$$

$$\sum_{n\in N} \delta_n(t) \le B, \forall t \in \mathbf{T}, \tag{10}$$

ε is the weight of the cache damage cost relative to the traffic delivery cost, which is always greater than or equal to 0. With the increase in ε, the cache damage cost will take a more critical part of the optimization objective. On the other hand, if we set ε to 0, the cache system will be degraded to a conventional cache system neglecting the cache damage cost. t_n is the time slot at which video n starts to be cached. As the cache retention time of video n is τ_n, during the time slots from t_n to $t_n + \tau_n$, video n will be cached, as shown in (8). B is the cache capacity. (10) guarantees the cached videos will not exceed the capacity of caches.

4 Lifecycle-Aware Cache Policy

We model the optimization problem as an integer programming problem. The problem is NP-Hard [33]. To solve this problem, we propose a two-step heuristic algorithm called LiA-cache policy. In the first step, we relax the constraint (10) and compute the optimal retention time τ_n for each content. Second, based on the computed τ_n, we design a cache eviction policy and propose a caching algorithm to minimize the overall cost function.

4.1 Retention Time Optimization

In this step, we relax the capacity limitation of the cache and minimize the cost for each video.

$$\min_{\delta_n(t)} \Psi_n, \tag{11}$$

where Ψ_n is the sum of traffic delivery cost and weighted cache damage cost of video n. Ψ_n can be written as follows,

$$\Psi_n = \Psi_n^{del} + \varepsilon \cdot \Psi_n^{dam}. \tag{12}$$

Substituting (4) and (5), we obtain

$$\Psi_n = \sum_{t=1}^{T} P_n(t)\left[1-\delta_n(t)\right] + \varepsilon \cdot f(\tau_n). \tag{13}$$

Using the integral to replace summation, we have,

$$\Psi_n = \int_{t=1}^{T} P_n(t)\,dt - \int_{t=t_n}^{t_n+\tau_n} P_n(t)\,dt + \varepsilon \cdot f(\tau_n). \tag{14}$$

Taking the derivative of Ψ_n with respect to τ_n gives,

$$\frac{d\Psi_n}{d\tau_n} = -P_n(t_n+\tau_n) + \varepsilon \cdot \frac{df(\tau_n)}{d\tau_n}. \tag{15}$$

In terms of [8], the cache damage cost function $f(\cdot)$ is a convex increasing polynomial of degree $m, m \ge 1$ and can be given by $f(\tau_n) = a_m(\tau_n)^m + a_{m-1}(\tau_n)^{m-1} + ... + a_1\tau_n$ with coefficients $a_i \ge 0, \forall i \ge 1$. We have

$$\frac{df(\tau_n)}{d\tau_n} = \left(m \cdot a_m \cdot (\tau_n)^{m-1} + ... + a_1\right). \tag{16}$$

$\frac{df(\tau_n)}{d\tau_n}$ is a monotone increasing function. Recalling (2) and (3), $-P_n(t_n+\tau_n)$ is monotone decreasing function. Therefore, in (15), Ψ_n is at a minimum when $\frac{d\Psi_n}{d\tau_n}$ equals 0, namely

$$\varepsilon \cdot \frac{df(\tau_n)}{d\tau_n} = P_n(t_n+\tau_n). \tag{17}$$

Substituting (16), we have

$$\varepsilon \cdot \left(m \cdot a_m \cdot (\tau_n)^{m-1} + ... + a_1\right) = P_n(t_n+\tau_n). \tag{18}$$

Since the function $P_n(\cdot)$ is either exponential or a power-law function, the above equation is a transcendental equation and the closed-form expression of τ_n cannot be obtained apart from when $m = 1$.

When $m = 1$, i.e., $f(\tau_n) = a_1\tau_n$, we can get the closed-form expression of τ_n. For the video n that belongs to the cluster one ($\mathbf{C_1}$), the optimal τ_n can be expressed as

$$\tau_n = \left\lfloor T_n - t_n - \frac{\ln\left(a_1 \cdot \varepsilon \big/ P_n^{MAX}\right)}{\ln\left(P_n^{MAX} \big/ \Delta p\right)} T_{C_1} \right\rfloor, \tag{19}$$

where $\lfloor\cdot\rfloor$ is the rounding down operation.

For the video n which belongs to the cluster two ($\mathbf{C_2}$), the optimal τ_n can be calculated as

$$\tau_n = \left\lfloor \exp\left(\frac{-\ln\left(a_1\varepsilon \big/ P_n^{MAX}\right)\ln\left(T_{C_2}+1\right)}{\ln\left(P_n^{MAX} \big/ \Delta p\right)}\right) + T_n - t_n - 1 \right\rfloor. \tag{20}$$

We have therefore defined the optimal retention time for caching a video. We still note that the parameter ε, as the weight of cache damage cost, will affect the optimal retention time. As shown in (19) and (20), the optimal retention time will decrease with an increase in ε. Due to the rise in ε, the cache damage cost accounts for a more significant part of the sum cost and should be strongly mitigated. Therefore, shorter retention time is preferred because cache damage is proportional to the data retention time.

4.2 Cache Eviction Policy

In this step, we take the cache capacity limitation into account and design the cache eviction policy. We consider the time-slotted caching operation with one-day long time slots, and the cache operations are performed nightly.

We first define a cache gain function for each video. For a video before time slot $\hat{t}$, it must be in one of two states, already in the cache or not yet cached. If video n is in the cache before time slot $\hat{t}$, its cache gain can be expressed as,

$$g_n = \sum_{t=\hat{t}}^{t_n+\tau_n} P_n(t) + \varepsilon \cdot \left[f\left(\hat{t}-t_n\right) - f(\tau_n)\right], \tag{21}$$

where $\sum_{t=\hat{t}}^{t_n+\tau_n} P_n(t)$ stands for the cache hit gain and $f\left(\hat{t}-t_n\right)$ is the storage gain for video n.

Alternatively, if video n has not been cached, the cache gain function is

$$g_n = \sum_{t=\hat{t}}^{\hat{t}+\tau_n} P_n(t) - \varepsilon \cdot f(\tau_n). \tag{22}$$

At the beginning of the time slot $\hat{t}$, we compute the cache gain for all videos, obtaining the gain set $\mathbf{G} = \{g_1, g_2, ..., g_n\}$, and select the top B videos in set $\mathbf{G}$ to cache.

The detailed steps and above operations are summarized in Algorithm 2. For a given time slot $\hat{t}$, we first identify cluster ID for all videos, including the newly released videos, and model their temporal patterns in terms of (2) and (3). For each video by solving (18), we obtain their optimal retention time τ. We then compute their cache gain concerning (21) and (22) and cache the top B videos with the highest gain.

Algorithm 2 LiA-cache

```
Input: The set of cached online videos $\mathbf{V_{\hat{t}}}$ before the time
    slot $\hat{t}$ and $|\mathbf{V_{\hat{t}}}| = B$.
 1: Identify cluster ID for all videos.
 2: for $n = 1, 2, ..., N$ do
 3:     Compute the optimal $\tau$ for video $n$ in terms of (18).
 4:     if the video $n$ is in $\mathbf{V_{\hat{t}}}$ then
 5:         $g_n = \sum_{t=\hat{t}}^{t_n+\tau_n} P_n(t) + \varepsilon \cdot \left[ f(\hat{t} - t_n) - f(\tau_n) \right]$
 6:     else
 7:         $g_n = \sum_{t=\hat{t}}^{\hat{t}+\tau_n} P_n(t) - \varepsilon \cdot f(\tau_n)$
 8:     end if
 9: end for
10: $\mathbf{G} = \{g_1, g_2, ..., g_n\}$, sorting $\mathbf{G}$ in descending order.
11: Put the top $B$ contents in set $\mathbf{G}$ into the set $\mathbf{V_{\hat{t}+1}}$.
Output: $\mathbf{V_{\hat{t}+1}}$
```

5 Evaluation and Discussion

In this section, we evaluate the performance of our proposed cache policy. For the sake of authenticity, we apply our cache policy to the real-world online video dataset presented in Section 2 and a synthetic video view dataset. In our case, we assume that the cache is already aware of cluster labels.

5.1 Fixed Cache Size

We first look at the performance of our LiA-cache policy, compared with other well-known policies on the real-world online video dataset. Table 4 illustrates the traffic delivery cost (TDC) and the cache damage cost (CDC) of five caching policies and the following two extreme cases, (i) unlimited cache size and (ii) zero cache size. We set the cache capacity B to 100. Thus, up to 100 videos can be in the cache simultaneously. Besides, we set the ε (see (6)) empirically to 5. We will discuss the impact of varying ε on both cache hit and cache damage in subsection 5.3.

RND is the random policy: the file is evicted from the cache uniformly at random. FIFO means the first in first out policy: the first file stored in the cache is evicted. Popularity represents the popularity-based cache policy in which at each iteration, the B most popular videos are cached. Niche is a state-of-the-art cache policy proposed by Vasilakos *et al.* [20]. Niche method adopts a dynamic popularity decay mechanism and user mobility predictions to improve cache hits by jointly exploring locality in time and space. However, due to the lack of mobility information in our real-world dataset[1], in practice, we implement Niche without mobility prediction information. Capturing only the temporal popularity in time will limit Niche's performance with respect to the cache hit ratio and probably to the cache damage as well. For RND, FIFO, Popularity, and Niche policies, we necessarily set the retention time for each caching operation to 60 days as it corresponds to the duration of the dataset.

No cache miss and Without cache are the two extreme cases, and their traffic delivery costs represent the lower and upper bounds of the system's traffic delivery cost, respectively. Note that the traffic delivery cost of No cache miss case is not equal to 0. Since we do not consider prefetch mechanisms in our system, each video has to be missed at least once before being put into the cache.

In Table 4, we notice that the performance of our proposed algorithm is better than other cache policies in terms of both traffic delivery cost and cache damage cost. That is because, apart from the popularity, our algorithm can also draw upon the useful information from videos' lifecycles. Niche benefits from its dynamic popularity decay mechanism and has a lower cache damage cost, compared to RND, FIFO, and Popularity policies. We also depict the cache damage reduction for LiA-cache compared to other policies. As our proposed algorithm is the only solution taking cache damage into account, it unsurprisingly shows ten times lower cache damage cost.

5.2 Variable Cache Size

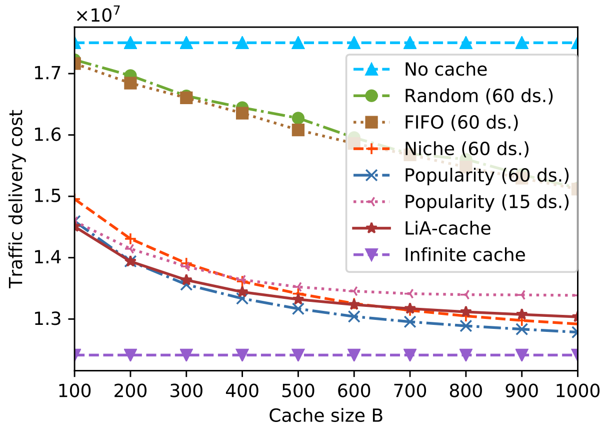
Fig. 8(a) and Fig. 8(b) respectively display how the traffic delivery cost and the cache damage cost change with the cache capacity. As illustrated in Fig. 8(a), the traffic delivery cost decreases linearly with the cache size for both RND and FIFO policies. The delivery costs for Niche, Popularity, and LiA-cache policies decrease logarithmically and quickly reach values extremely close to the lower limit. Although Niche policy has a higher traffic delivery cost when the cache size is small, its performance is similar to the Popularity policy under large cache sizes. In comparing Popularity and LiA-cache policies, although both solutions show similar results, using only Popularity as a metric shows slightly lower delivery costs. This is because the aim of our cache policy is to minimize the sum cost of traffic delivery and weighted cache damage instead of the traffic delivery cost alone. However, in the case of Popularity based caching with 15 days of retention time, the delivery cost is actually higher than LiA-cache due to the presence of videos whose lifecycle is longer than 15 days.

The cache damage cost remains close to constant over the cache capacity for LiA-cache policy. In this case, lowering the programming voltage for shorter retention times leads to a tremendous decrease in cache damage. The popularity based algorithms also display lower cache damage cost as a side effect of caching the most popular videos, which would remain longer in the cache. Moreover, compared to Popularity, Niche benefits from its dynamic popularity decay mechanism and has lower cache damage cost under small cache size scenarios. However, as cache size increases

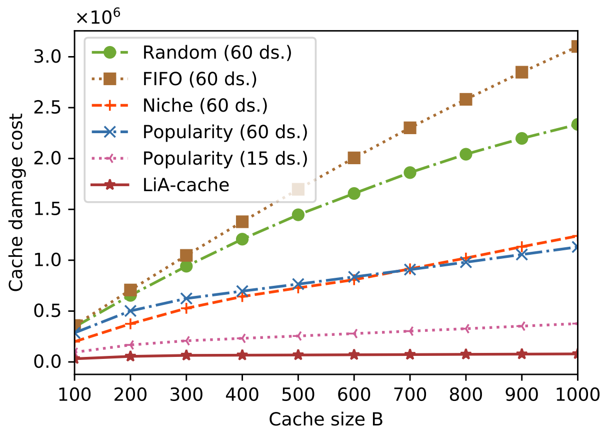
1. The synthetic traces are difficult to reveal temporal locality variations.

TABLE 4
Traffic delivery cost and Cache damage cost for caching policies in which $B = 100$ and $\varepsilon = 5$. Traffic delivery cost and Cache damage cost are computed by (4) and (5) respectively.

Cost	RND	FIFO	Popularity	Niche	LiA-cache	No cache miss	Without cache
Traffic delivery cost	17168106	17163779	14594113	14955151	14507138	12412830	17502887
Cache damage cost	344760	358080	285840	199440	30986	NA	0
Cache damage reduction (%)	91%	91%	89%	84%	NA	NA	NA



(a) Traffic delivery cost versus the cache capacity.



(b) Cache damage cost versus the cache capacity.

Fig. 8. Costs versus the cache capacity, with $\varepsilon = 5$. '60 ds.' indicates that the video retention time of each caching operation for that caching policy is 60 days and '15 ds.' means that the retention time is 15 days.

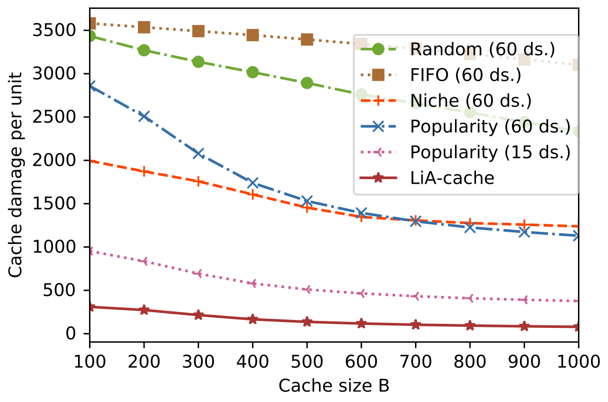


Fig. 9. Cache damage per cache unit versus cache size, where $\varepsilon = 5$.

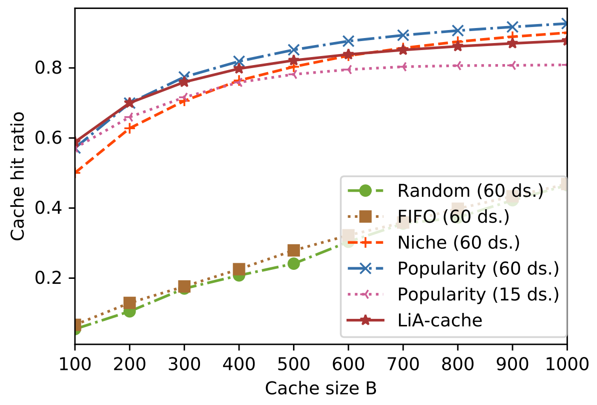


Fig. 10. Cache hit ratio versus cache size, where $\varepsilon = 5$.

the benefits brought by the dynamic popularity decay mechanism are degraded. Also, both FIFO and RND's cache damage costs increase linearly with the cache capacity. FIFO suffers from the highest cache damage cost compared to the others, due to the largest number of cache evictions. This increase in the cache damage cost is expected because the bigger the cache, the more videos are cached, and the higher the cache damage. On average, LiA-cache policy has damage costs 84% to 98% smaller than all other solutions, as shown in Table 4.

We then analyze the cache damage per cache unit as a function of cache size. The cache damage per cache unit is calculated as the entire cache damage cost divided by the cache size. The results are presented in Fig. 9. The cache damage per cache unit remains nearly constant up to a high cache size for FIFO while decreasing with cache size for both RND, Niche, and Popularity policies. Regarding video retention time, the popularity policy with 15 days retention time has overall lower cache damage than with 60 day retention time. That is because the cache damage for each program/erase operation is directly proportional to the retention time. Since we take cache damage into account in our caching decision, our algorithm's cache damage per unit also decreases although less steeply. In other words, LiA-cache policy outperforms other conventional policies, especially with lower cache capacities.

Cache hit ratio is also a critical metric in evaluating a cache policy. In Fig. 10, we notice that FIFO and RND have similar results, the hit ratio grows linearly with cache size and barely reaches about 0.5 for a cache size of 1000. LiA-cache displays cache hit ratios between 0.59 and 0.87, while the popularity based caching policy is between 0.57 and 0.92. Similar to the traffic delivery cost in Fig. 8(a), the cache hit ratio is slightly lower for LiA-cache than the popularity and Niche algorithms for large cache sizes. This is because LiA-cache policy makes a trade-off between cache damage cost and traffic delivery cost by minimizing the sum cost, referring to (6). In other words, the lower cache damage will sacrifice some cache hits. By comparing the results of Fig. 8(b) and Fig. 10, we observe that LiA-cache does not

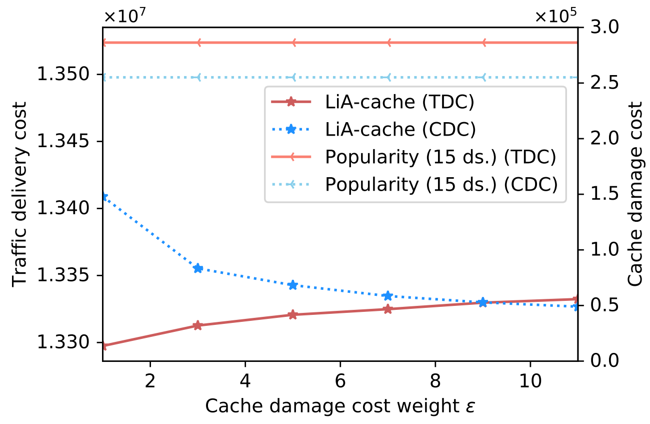


Fig. 11. Performance of traffic delivery cost (TDC) and cache damage cost (CDC) versus the cache damage weight (ε), where the cache size B is 500.

fully explore the benefits of a large cache to improve the cache hit ratio because LiA-cache keeps a low cache damage cost to achieve the minimum sum cost. It is worth noting that the importance of cache damage in the sum cost can be adjusted by parameter ϵ. The impact of ϵ will be discussed in section 5.3 in detail.

Indeed, LiA-cache tries to increase the cache hit ratio by predicting videos' future popularity using video lifecycles instead of purely using the popularity of previous time slots. This prediction is then mediated by the cache damage cost to make the final caching decision. We also note that the cache hit ratio of popularity-based caching is related to the predefined retention time. The popularity-based caching policy with a lower retention time experiences slightly a lower cache hit ratio since a 15 days retention time is too short for long-lived videos.

5.3 Influence of the Cache Damage Weight

To illustrate the impact of cache damage weight ϵ, in Fig. 11, we present the traffic delivery cost (TDC) and the cache delivery costs (CDC) of LiA-cache versus the weight given to cache damage in (6). Naturally, with a rise in ε, the traffic delivery cost increases while the cache damage cost decreases. Due to the rise in ε, the cache damage cost accounts for a more significant part of the sum cost and thus should be strongly mitigated. On the other hand, when reducing ϵ, the cache damage cost rises. Overall, setting $\epsilon = 5$ gives a good trade-off between both costs and still allows us to provide cache damage cost and traffic delivery cost significantly lower than popularity based methods.

5.4 Influence of Temporal Patterns of Videos

In previous evaluations, we evaluated our policy on the collected real-world online video dataset. However, as the view patterns of videos are fixed, we cannot explore how the temporal patterns of videos affect the performance of Li-A cache. Therefore, we generate a four-month synthetic video view dataset. In the synthetic dataset, there are 10,000 videos of pulse shape temporal patterns and three types of typical lifecycles, i.e., $T_{C_1} = 7$, $T_{C_2} = 30$, $T_{C_3} = 60$. For a video n, the video is randomly assigned to a typical lifecycle, and its exact lifecycle T_n follows a log-normal distribution,

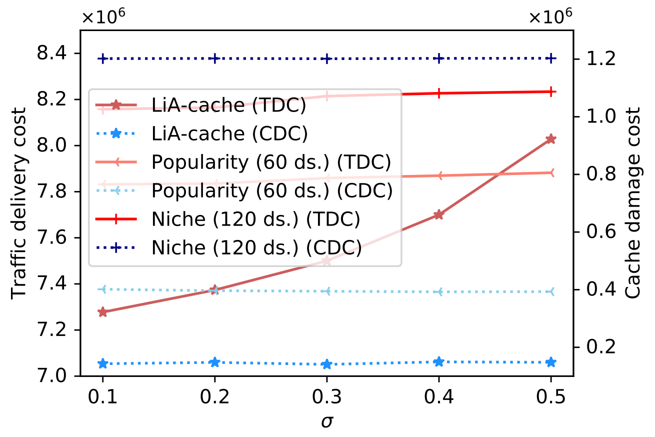


Fig. 12. Performance of traffic delivery cost (TDC) and cache damage cost (CDC) versus different σ, where the cache size B is 100, $\epsilon = 5$, and $\Delta p = 1$.

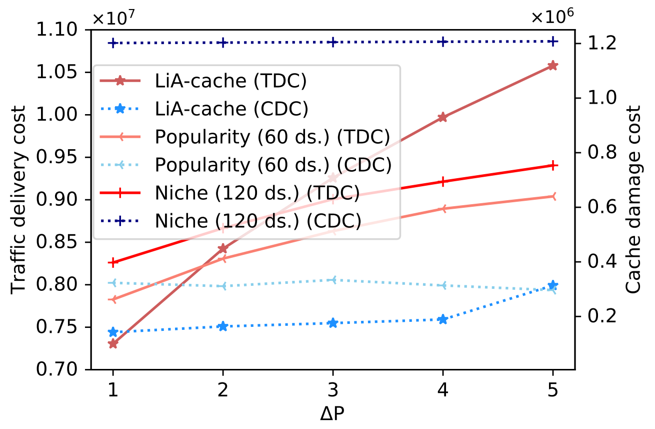


Fig. 13. Performance of traffic delivery cost (TDC) and cache damage cost (CDC) versus different Δp, where the cache size B is 100, $\epsilon = 5$, and $\sigma = 0.1$.

$\log(T_n) \sim \mathcal{N}(\log(T_C), \sigma^2)$ where T_C is the assigned typical lifecycle of video n and σ is the standard deviation of the Gaussian distribution. In terms of prior observations [34], we adopt 0.75 as the default value of the Zipf distribution exponent parameter used for video popularity.

We first investigate how the LiA-cache policy is affected by σ, the standard deviation of videos' lifecycles from typical lifecycles. In other words, σ reflects the accuracy of using typical lifecycles to approximate the exact lifecycles of videos. The higher the σ, the lower the accuracy. In Fig.12, we present the traffic delivery cost and the cache damage cost of LiA-cache, Popularity, and Niche policies. Both Popularity and Niche policies are not sensitive to changes in σ. However, for LiA-cache, with the increase in σ, its traffic delivery cost increases, and the performance is degraded lower than Popularity policy when σ is equal to 0.5. Therefore, determining the typical lifecycles of videos is critical to LiA-cache. As performed in our work, collecting real-world data and applying pattern recognition algorithms to discover typical lifecycles empirically is an applicable and effective method. Also, we still note that the cache damage cost of LiA-cache will not vary under different σ.

Next, we delve into the impact of Δp, the parameter to determine the typical lifecycles in Definition 1. In Fig.13, we present how the traffic delivery cost and the cache damage cost change with different Δp. Since we fix the typical lifecycles, increasing Δp will increase the number of video viewing requests. Therefore, the traffic delivery costs of Popularity, Niche, and LiA-cache policies increase simultaneously. Nevertheless, for LiA-cache, its growth trend is sharper because it cannot fully capture the relaxation part with large Δp, causing extra costs. Meanwhile, in terms of the cache damage cost, although LiA-cache is tolerant of a larger Δp, the performance will start to degrade when Δp is greater than 5.

5.5 Discussion

In this section, we analyzed the influence of various parameters on the cache performance, regarding traffic delivery cost (the most common cache evaluation metric) and cache damage cost (the subject of this study). We showed that keeping both metrics at an optimal level was essentially a question of a trade-off between different cache sizes as well as the weight attributed to the cache damage cost. Among the various scenarios, LiA-cache significantly outperformed them when taking cache damage into account and performed similarly to popularity based solutions in terms of delivery cost.

6 Related work

The work presented in the paper falls within the research areas of video and content request pattern analysis and edge caching systems for improving user experience. Thus, we next discuss related work in these two domains.

6.1 Content Request Patterns

Many studies have investigated content request patterns and discovered general characteristics based on real-world datasets. Breslau *et al.* [34] investigated a web page request distribution and found the requests follow a Zipf-like distribution with varying exponents. Newman [35] made a comprehensive study of power-law distributions and illustrated that power laws appear widely in web hits, copies of books sold, telephone calls, etc. Cha *et al.* [36] monitored YouTube's list of the daily 100 most popular videos and discovered the shape of the video popularity distribution is power-law. Also Finley *et al.* [37] studied the usage popularity of apps across smartphones, tablets and PCs from a USA-based user group and found that log-normal and stretched-exponential distributions were typically the best fits. However, the above studies only examined static snapshots and did not consider temporal varying in the popularity of individual content. Oka *et al.* crawled social media data from Twitter and showed that the popularity of one topic in social media undergoes stages of burst and decay. Moreover, Crane *et al.* [21] studied the time series of daily views for nearly 5 million videos on YouTube. They observed that hundreds of thousands of videos have a burst of activity following a power-law relaxation, which corresponds with our findings in section 2.

6.2 Cache at Edge

Initially, the concept of a cache comes from computer systems and was designed to fill the throughput gap between the main memory and registers [38]. The idea of caching was later introduced in mobile networks and now permeates to the edge of networks, including base stations and end devices [39], [40]. By exploring the spatiotemporal redundancy of users' requests, caching popular content at network edges can reduce transmission delays and traffic burdens [41].

Bacstug *et al.* [42] used a stochastic geometry method to model cache-enabled small cell networks and theoretically demonstrated that employing cache units in small cells indeed is beneficial in terms of average delivery rate. Li *et al.* [18] proposed a popularity-driven content cache (PopCaching) policy that learns the popularity of contents and applies the policy to determine what to store. Dernbach *et al.* [43] investigated distinct regional tastes when selecting caching content to improve the cache hit ratio and showed that the bigger the inter-region distance, the better a local caching policy can perform. Also, some scholars explored the potential for using user mobility patterns to improve cache hit ratio in edge networks. For instance, Siris *et al.* [44] combined mobility prediction and proactive caching to support seamless mobility with low transmission delays. Guan *et al.* [45] assumed that users' preferences for content and mobility patterns are prior information. They then formulated an optimization problem with the objective of maximizing the utility of caching and devised a heuristic caching strategy.

Since the popularity of content varies with time, it is critical to update caches at intervals. In [46], Blasco *et al.* divided time into periods. Within each period, there is a cache replacement phase when the content with the lowest popularity is discarded. Instead of focusing on popular content, Vasilakos *et al.* [20] proposed a distributed solution targeting less popular personalized content requests. They adopted a dynamic pricing replacement approach and an adaptable popularity decay mechanism to optimize content transmission delays. Moreover, some scholars modeled the cache replacement problem as a Markov decision process and applied machine learning algorithms, like Q-learning [47] and deep reinforcement learning [48] to obtain the optimal cache replacement strategy.

However, no prior work, including the above-mentioned work, has proposed a solution with a service provider viewpoint that enhances expected memory life while maximizing the cache hit ratio. In a nutshell, to the best of our knowledge, this is the first study to take cache damage into account and exploit the content's lifecycle pattern from a real dataset to minimize the cache damage cost while keeping a high cache hit ratio.

7 Discussion and Conclusion

This paper investigated the lifecycles of popular online videos and linked them to cache damages. Specifically, given the high voltages currently used for program/erase operations, the 'expiry date' of cached videos is on the order of years, while video lifecycles are usually less than a couple of weeks. Therefore, we proposed a novel caching policy that takes into account the cache damage caused by these high voltages and thus increases the physical memory's life expectancy while keeping a high cache hit ratio compared to other popular solutions.

Based on a large-scale and real-world dataset of online video access collected by a major operator in China over a duration of two months, we distinguished several popularity patterns for videos and classified them into two distinct clusters, short-lived videos with an exponential relaxation tail and long-lived videos whose popularity pattern follows a power-law distribution. We then proposed a heuristic algorithm using these lifecycle patterns to improve the cache hit ratio and reduce the cache damage. This algorithm matches videos to a given cluster and estimates its lifecycle to determine the retention time in which the video will be in the cache. We showed that our solution performs similarly to a pure popularity-based cache policy both regarding traffic delivery cost and cache hit ratio, while significantly reducing memory damages over time. Even popularity-based solutions with lower retention time, i.e., lower programming voltage, yield significantly higher damage to the cache, while providing a lower cache hit ratio due to the presence of videos with popularity exceeding the fixed retention time. Although our solution displays slightly higher traffic delivery costs compared to a pure popularity-based caching policy in a high retention time context, the impact is negligible considering the enormous gain in cache damage costs.

In the future, we plan to focus on the relationships between video lifecycles and their attributes, such as video type, size, quality, and release time, to quickly identify to which cluster the video content belongs. We also plan to propose some stochastic analysis of videos presenting noise-shaped popularity functions and design guidelines on how to cache them. Finally, we wish to proceed to a cross-media analysis to determine which video may experience high popularity before its release, and proactively place it in the cache as soon as it is published. For instance, analyzing the growing hype around movies or series on movie database websites would permit us to prioritize some videos based on their expected popularity.

Acknowledgment

This research has been supported in part by the project 16214817 from the Research Grants Council of Hong Kong, project FP805 from HKUST, the 5GEAR project from the Academy of Finland, the National Key Research and Development Program of China under grant 2018YFB1800804, the National Nature Science Foundation of China under grant U1836219, 61971267, 61972223, 61861136003, Beijing Natural Science Foundation under grant L182038, Beijing National Research Center for Information Science and Technology under grant 20031887521, and research fund of Tsinghua University-Tencent Joint Laboratory for Internet Innovation Technology.

References

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

[2] Elisha J Rosensweig, Daniel S Menasche, and Jim Kurose. On the steady-state of cache networks. In *2013 Proceedings IEEE INFOCOM*, pages 863–871, April 2013.

[3] M. J. Siavoshani, A. Pourmiri, and S. P. Shariatpanahi. Storage, communication, and load balancing trade-off in distributed cache networks. *IEEE Transactions on Parallel and Distributed Systems*, 29(4):943–957, April 2018.

[4] Steve Byan, James Lentini, Anshul Madan, and Luis Pabon. Mercury: Host-side flash caching for the data center. In *IEEE 28th Symposium on Mass Storage Systems and Technologies (MSST)*, pages 1–12, April 2012.

[5] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, April 2010.

[6] Liang Shi, Kaijie Wu, Mengying Zhao, Chun Jason Xue, Duo Liu, and Edwin H-M Sha. Retention trimming for lifetime improvement of flash memory storage systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(1):58–71, Jan 2016.

[7] Ren-Shuo Liu, Chia-Lin Yang, and Wei Wu. Optimizing nand flash-based ssds via retention relaxation. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, FAST'12, pages 11–11, Berkeley, CA, USA, 2012. USENIX Association.

[8] Samta Shukla and Alhussein A Abouzeid. Optimal device-aware caching. *IEEE Transactions on Mobile Computing*, 16(7):1994–2007, July 2017.

[9] Peter Desnoyers. What systems researchers need to know about nand flash. In *Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems*, HotStorage'13, pages 6–6, Berkeley, CA, USA, 2013. USENIX Association.

[10] Xavier Jimenez, David Novo, and Paolo Ienne. Wear unleveling: Improving NAND flash lifetime by balancing page endurance. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*, pages 47–59, Santa Clara, CA, 2014. USENIX.

[11] Jaeyong Jeong, Sangwook Shane Hahn, Sungjin Lee, and Jihong Kim. Lifetime improvement of NAND flash-based storage systems using dynamic program and erase scaling. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*, pages 61–74, Santa Clara, CA, 2014. USENIX.

[12] Jürgen Kaiser, Fabio Margaglia, and André Brinkmann. Extending ssd lifetime in database applications with page overwrites. In *Proceedings of the 6th International Systems and Storage Conference*, SYSTOR '13, pages 11:1–11:12, New York, NY, USA, 2013. ACM.

[13] Gabor Szabo and Bernardo A. Huberman. Predicting the popularity of online content. *Commun. ACM*, 53(8):80–88, August 2010.

[14] Muhammad Bilal and Shin-Gak Kang. A cache management scheme for efficient content eviction and replication in cache networks. *IEEE Access*, 5:1692–1701, 2017.

[15] S. Zhang, N. Zhang, X. Fang, P. Yang, and X. S. Shen. Cost-effective vehicular network planning with cache-enabled green roadside units. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, May 2017.

[16] Negin Golrezaei, Karthikeyan Shanmugam, Alexandros G Dimakis, Andreas F Molisch, and Giuseppe Caire. Femtocaching: Wireless video content delivery through distributed caching helpers. In *2012 Proceedings IEEE INFOCOM*, pages 1107–1115. IEEE, March 2012.

[17] B. N. Bharath, K. G. Nagananda, and H. V. Poor. A learning-based approach to caching in heterogenous small cell networks. *IEEE Transactions on Communications*, 64(4):1674–1686, April 2016.

[18] Suoheng Li, Jie Xu, Mihaela Van Der Schaar, and Weiping Li. Popularity-driven content caching. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, April 2016.

[19] N. Carlsson and D. Eager. Ephemeral content popularity at the edge and implications for on-demand caching. *IEEE Transactions on Parallel and Distributed Systems*, 28(6):1621–1634, June 2017.

[20] Xenofon Vasilakos, Vasilios A. Siris, and George C. Polyzos. Addressing niche demand based on joint mobility prediction and content popularity caching. *Computer Networks*, 110:306 – 323, 2016.

[21] Riley Crane and Didier Sornette. Robust dynamic classes revealed by measuring the response function of a social system. *Proceedings of the National Academy of Sciences of the United States of America*, 105(41):15649–53, 2008.

[22] Philipp Lorenz-Spreen, Bjarke Mørch Mønsted, Philipp Hövel, and Sune Lehmann. Accelerating dynamics of collective attention. *Nature communications*, 10(1):1759, 2019.
[23] Janette Lehmann, Bruno Gonçalves, José J. Ramasco, and Ciro Cattuto. Dynamical classes of collective attention in twitter. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 251–260, New York, NY, USA, 2012. ACM.
[24] Jaewon Yang and Jure Leskovec. Patterns of temporal variation in online media. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 177–186, New York, NY, USA, 2011. ACM.
[25] Zhaopeng Fan, Gong Zhang, and Simon Liao. *Pulse Wave Analysis*. InTech, 2011.
[26] Hwanjun Song, Jae-Gil Lee, and Wook-Shin Han. Pamae: Parallel k-medoids clustering with high accuracy and efficiency. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1087–1096, New York, NY, USA, 2017. ACM.
[27] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods*, 3(1):1–27, 1974.
[28] David L Davies and Donald W Bouldin. A cluster separation measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979.
[29] Albert-Laszlo Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207, 2005.
[30] R Dean Malmgren, Daniel B Stouffer, Adilson E Motter, and Luís AN Amaral. A poissonian explanation for heavy tails in e-mail communication. *Proceedings of the National Academy of Sciences of the United States of America*, 105(47):18153–18158, 2008.
[31] Marta C Gonzalez, Cesar A Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779, 2008.
[32] José Luis Iribarren and Esteban Moro. Impact of human activity patterns on the dynamics of information diffusion. *Physical review letters*, 103(3):038702, 2009.
[33] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.
[34] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, volume 1, pages 126–134 vol.1, March 1999.
[35] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.
[36] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon. Analyzing the video popularity characteristics of large-scale user generated content systems. *IEEE/ACM Transactions on Networking*, 17(5):1357–1370, Oct 2009.
[37] Benjamin Finley, Tapio Soikkeli, and Kalevi Kilkki. Mobile application usage concentration in a multidevice world. In *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications*, pages 40–51, 2016.
[38] John L Hennessy and David A Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.
[39] Ejder Bastug, Mehdi Bennis, and Mérouane Debbah. Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE Communications Magazine*, 52(8):82–89, 2014.
[40] G. Xylomenos, X. Vasilakos, C. Tsilopoulos, V. A. Siris, and G. C. Polyzos. Caching and mobility support in a publish-subscribe internet architecture. *IEEE Communications Magazine*, 50(7):52–58, July 2012.
[41] Alec Wolman, M. Voelker, Nitin Sharma, Neal Cardwell, Anna Karlin, and Henry M. Levy. On the scale and performance of cooperative web proxy caching. In *Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles*, SOSP '99, pages 16–31, New York, NY, USA, 1999. ACM.
[42] Ejder Bacstug, Mehdi Bennis, Marios Kountouris, and Mérouane Debbah. Cache-enabled small cell networks: Modeling and tradeoffs. *EURASIP Journal on Wireless Communications and Networking*, 2015(1):41, 2015.
[43] Stefan Dernbach, Nina Taft, Jim Kurose, Udi Weinsberg, Christophe Diot, and Azin Ashkan. Cache content-selection policies for streaming video services. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, April 2016.
[44] V. A. Siris, X. Vasilakos, and G. C. Polyzos. Efficient proactive caching for supporting seamless mobility. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6, June 2014.
[45] Y. Guan, Y. Xiao, H. Feng, C. Shen, and L. J. Cimini. Mobicacher: Mobility-aware content caching in small-cell networks. In *2014 IEEE Global Communications Conference*, pages 4537–4542, Dec 2014.
[46] Pol Blasco and Deniz Gündüz. Learning-based optimization of cache content in a small cell base station. In *2014 IEEE International Conference on Communications (ICC)*, pages 1897–1903. IEEE, 2014.
[47] Ejder Baştuğ, Jean-Louis Guénégo, and Mérouane Debbah. Proactive small cell networks. In *ICT 2013*, pages 1–5. IEEE, 2013.
[48] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Network*, 33(5):156–165, Sep. 2019.

Tong Li received the B.S. degree and M.S. degree in communication engineering from Hunan University, China, in 2014 and 2017. At present, he is a dual Ph.D. student at the Hong Kong University of Science and Technology and the University of Helsinki. His research interests include distributed systems, edge network, and data-driven network. He is an IEEE student member.

Tristan Braud received the M.E. degree from both Grenoble INP, Grenoble, France and Politecnico di Torino, Turin, Italy, and the Ph.D. degree from Universit Grenoble-Alpes, Grenoble, France. He is currently a postdoctoral fellow at the Hong Kong University of Science and Technology, within the System and Media Laboratory (SymLab). His major research interests are in the area of systems and networks, and future Internet Applications, and mobile computing.

Yong Li (M'09-SM'16) received the B.S. degree in electronics and information engineering from Huazhong University of Science and Technology, Wuhan, China, in 2007 and the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012. He is currently a Faculty Member of the Department of Electronic Engineering, Tsinghua University.

Dr. Li has served as General Chair, TPC Chair, SPC/TPC Member for several international workshops and conferences, and he is on the editorial board of two IEEE journals. His papers have total citations more than 6900. Among them, ten are ESI Highly Cited Papers in Computer Science, and four receive conference Best Paper (run-up) Awards. He received IEEE 2016 ComSoc Asia-Pacific Outstanding Young Researchers, Young Talent Program of China Association for Science and Technology, and the National Youth Talent Support Program.

Pan Hui (SM'14-F'18) received his Ph.D. degree from the Computer Laboratory at University of Cambridge, and both his Bachelor and MPhil degrees from the University of Hong Kong.

He is the Nokia Chair Professor in Data Science and Professor of Computer Science at the University of Helsinki. He is also the director of the HKUST-DT Systems and Media Lab at the Hong Kong University of Science and Technology. He was a senior research scientist and then a Distinguished Scientist for Telekom Innovation Laboratories (T-labs) Germany and an adjunct Professor of social computing and networking at Aalto University. His industrial profile also includes his research at Intel Research Cambridge and Thomson Research Paris. He has published more than 300 research papers and with over 17,500 citations. He has 30 granted and filed European and US patents in the areas of augmented reality, data science, and mobile computing. He has been serving on the organising and technical program committee of numerous top international conferences including ACM SIGCOMM, MobiSys, IEEE Infocom, ICNP, SECON, IJCAI, AAAI, ICWSM and WWW. He is an associate editor for the leading journals IEEE Transactions on Mobile Computing and IEEE Transactions on Cloud Computing. He is an IEEE Fellow, an ACM Distinguished Scientist, and a member of the Academia Europaea.