Journal of Parallel and Distributed Computing 136 (2020) 1–13

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Budgeted video replacement policy in mobile crowdsensing

En Wang [a], Yongjian Yang [a,*], Jie Wu [b], Kaihao Lou [a], Wenbin Liu [a], Yuanbo Xu [a]

[a] Department of Computer Science and Technology, Jilin University, China
[b] Department of Computer and Information Sciences, Temple University, USA

ARTICLE INFO

Article history:
Received 14 August 2018
Received in revised form 28 May 2019
Accepted 7 October 2019
Available online 14 October 2019

Keywords:
Mobile video crowdsensing
Replacement policy
Budgeted
NP-hard

ABSTRACT

Mobile crowdsensing offers a new platform that recruits a suitable set of users to collectively complete an information collection/sensing task through users' equipped devices. As a special case, video crowdsensing is to collect different video segments of the same event that are taken separately by the built-in cameras of mobile devices, and then combine them into a complete video. Mobile crowdsensing has attracted considerable attention recently due to the rich information that can be provided by videos. However, because of the limited caching space, a suitable video replacement policy is necessary. In this paper, we propose a Budgeted Video replaCement policy in mobile Video crowdsensing (BVCV), which first determines a video segment's value according to its caching situation and natural attributes. Then, we formulate the video caching problem as a budgeted maximum coverage problem, which is a well-known NP-hard problem. Finally, we propose a practical greedy solution and also infer the approximate ratio, which could be regarded as the lower bound of BVCV to the optimal solution. Our experiments with the real mobility datasets (StudentLife dataset, Buffalo/phonelab-wifi dataset) show that, the proposed budgeted video replacement policy achieves a longer successfully delivered video length, compared with other general replacement policies.

1. Introduction

Mobile CrowdSensing (MCS) is a new kind of crowdsourcing method, where mobile users could utilize the rich sensors in their handheld devices to sense the surroundings and jointly finish a common sensing task [9]. The information collected by a user's device is just data, while the collected and addressed information of tremendous devices can be transformed into knowledge. The knowledge is taken from users and also serves users through a variety of applications ranging from urban dynamic mining and environment monitoring to parking space management and indoor localization [7,10,14]. The existing works in terms of MCS mainly focus on user recruitment, task allocation and incentive mechanism [11,23]. Obviously, different task requirements lead to the different data types (e.g., text, picture, or video). In this paper, we are especially interested in Video CrowdSensing (VCS) [12,30], where people shoot videos for an event with their mobile devices. Compared with the traditional MCS, VCS collects a richer information through videos and could give users a more intuitive impression for the event.

Suppose that an important event happens in the location without any surveillance video around it. The revivification of event relies on the crowdsensed videos taken by the mobile users. For example, Waze [3] collects information on road conditions and accidents shared by drivers. Then it provides a timely and efficient road traffic report. Google map applications provide users with a visual and intuitive feeling for a specific location through the videos of street cameras. All of these are VCS applications [26], which provide users an intuitional feedback for the specific event.

However, VCS also faces some difficulties along with providing users a good service experience. Among them, the most important issue is caching problem [30]. This is because, in VCS, the data sensed by mobile devices is video file which occupies a non-negligible cache space, and each mobile device usually has a limited cache space to serve the crowdsensing task. So it is really important to efficiently cache the videos (i.e., caching the more important videos) and further get a higher video crowdsensing quality (i.e., uploading a longer required video).

In this paper, in order to efficiently schedule the videos in the cache, we propose a budgeted video replacement policy in mobile video crowdsensing, which first decides a video segment's value according to its own attribute (video length) and also the copy distribution (number of copies) among the other mobile devices. Here, a video segment is part of the whole required video. We then formulate the video caching problem as a budgeted maximum coverage problem [19], which is a well-known NP-hard problem. Finally, we propose a practical greedy solution and also infer the approximate ratio between the result of the proposed algorithm and that of the optimal solution.

* Corresponding author.
E-mail address: yyj@jlu.edu.cn (Y. Yang).

https://doi.org/10.1016/j.jpdc.2019.10.003
0743-7315/© 2019 Elsevier Inc. All rights reserved.

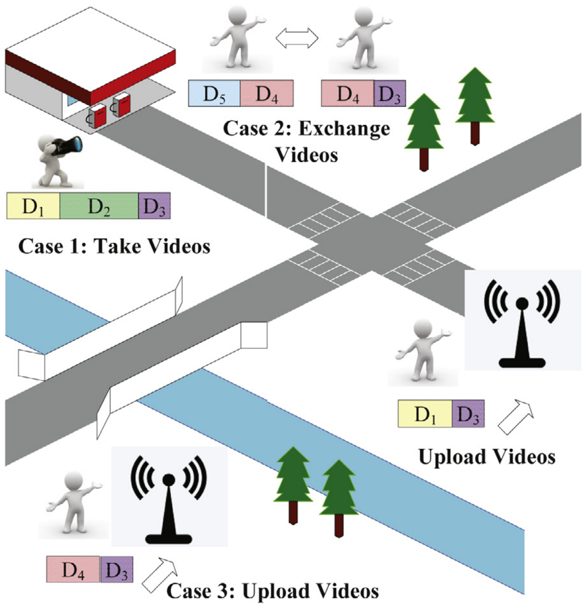


Fig. 1. The video crowdsensing description. The users take (case 1) videos for the specific task and copy (case 2) the stored videos with each other, until they access the wifi, and then they upload (case 3) all the caching videos.

As shown in Fig. 1, users walk around in a city, and they take videos for the specific event, which is required by the task. Then they cache them in their devices (i.e., D_i in Fig. 1). When they move into a wifi area, they could upload the cached videos to the cloud server. Users could predict the frequency to access the wifi area, then when users encounter each other, they could copy the videos in order to upload the videos as soon as possible. Obviously, the cache space of a user is usually not enough to store all the videos (generated by itself and collected from the other users). For uploading as long as possible nonoverlapping videos before the given deadline of task, they should decide which video segment to delete among the existing videos and the new coming one, according to the incremental value of video's utility. We assume that frequency of taking the new video from an event or copying from another neighbor user is unknown and such frequency does not follow any particular distribution. Although cache management is not a new research area, it still brings many new challenges for VCS applications.

- *Video Segment Utility*: it is difficult for us to decide an unheuristic method for ranking the importance of different videos.
- *Dependency of Videos*: the relationship among different video segments is interdependence as these video segments may contain some overlap parts of the whole required video. Hence, the new coming video's contribution could not be easily measured by its own utility, as it depends on the existing videos.
- *Variety in Video Lengths*: the videos may have the different sizes, that is to say, they consume different caching spaces. A video with a high utility may also occupy the large caching space. Hence, we could not directly decide the priority of videos according to the utility.
- *Multi-User Situation*: the multi-user situation will be more difficult because the different users' caching situations will influence each other and may also jointly influence the successfully uploaded video length. Here, we only focus on the different time to take videos for an event, while ignoring the different angles and distances.

In order to overcome the above challenges, we design a budgeted video replacement policy in mobile video crowdsensing. The main contributions of this paper are briefly summarized as follows:

- We propose an efficient way to measure the corresponding utility for each video segment, through calculating the influence on the contribution of keeping the video on user's cache.
- We propose a budgeted video replacement policy in mobile video crowdsensing, taking both the different videos' sizes and coverages into consideration, in order to maximize the total nonoverlapping length of uploaded videos.
- We formulate the video caching problem as the NP-hard problem, then we adopt a practical greedy heuristic to address the problem, and also infer the approximate ratio between the results of the proposed algorithm and the optimal solution.
- We conduct extensive simulations based on two widely-used real-world traces: StudentLife dataset and Buffalo/phonelab-wifi dataset. The results show that compared with other video replacement policies, BVCV achieves the longest successfully delivered video length.

The remainder of this paper is organized as follows: we review the related work in Section 2. The problem formulation and description are shown in Section 3. The budgeted video replacement policy proposed in this paper is described detailedly in Section 4 including the algorithms and approximate ratio. In Section 5, we evaluate the performance of the proposed replacement policy through extensive simulations. We conclude the paper in Section 6. Our proofs are presented in Appendix.

2. Related work

In this paper, we consider the related work as belonging into two aspects: (1) visual sensing; (2) resource management.

2.1. Visual crowdsensing

There have been many works on visual crowdsensing. In [20], Li et al. review the current situation of crowded scene technologies. They first introduce the background and concept of crowded scene. Then, they also do a research in terms of algorithms, protocols and models in this research area. In [4], Chen et al. propose a sensing scheme to make users cover the events collaboratively. More importantly, they use crowd-powered approach to address the events. [5] uses pyramid-tree model to address the data selection problem. According to the task requirements, the most suitable pictures are selected. In [28], Wang et al. build a system called CrowdWatch, which utilizes crowd sensing technology to sense the existing obstacles and makes a timely decision and alert for distracted walkers. [6] proposes an indoor 3D modeling method with crowdsourced 2D photos, which are taken by the mobile users. In [31], Xu et al. develop a locating application in mobile devices, it extracts geometric constraints based on images crowdsourced by the mobile devices to reduce the sensing cost. PhotoCity [25] is an online game that teaches the users to take photos at targeted locations. Then, the photos could be used to build 3D models. In [21], Liu et al. design four truthful incentive mechanisms for selecting workers to form a valid team and cooperatively finish a crowdsourcing task. In [13], Hong et al. propose an algorithm to select the parameters to fit more videos of the mobile devices. [12] surveys the definitions, general methods, and specific applications of VCS, and also analyzes the future challenges and core techniques. Moreover, in [29], they present a video crowdsourcing framework, which

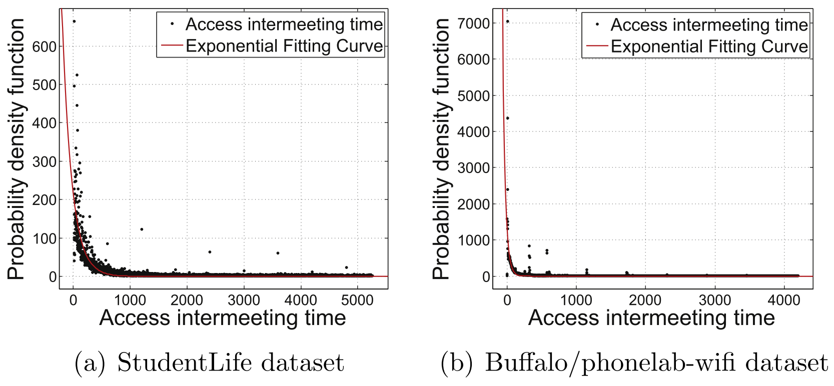


Fig. 2. The exponential probability distribution of access time intervals (entering wifi area) for the users.

utilizes the wisdom of the crowd to produce video summaries under the controlled cost of sourcing participants. [18] proposes a caption editing system, which makes the crowdsourcing an attractive task, and its interface includes game-based elements. In this way, the system achieves crowdsourced work for the useful task of video captioning.

The above researches pay attention to visual crowdsensing. However, they focus on efficient video/photo crowdsensing, while not considering the caching problem in user's device.

2.2. Resources management

There have also been some works focusing on resources management in MCS. In order to maximize the lifetime of sensing task, [1] proposes a novel method to handle the resource assignment problem so that MCS tasks are fairly allocated to the mobile users. [22] does a survey of state-of-the-art works for those focusing on reducing the resource cost and getting a high service quality. In [17], Ju et al. regard the sensing resource assignment problem as a social welfare maximization problem, and propose a primal–dual approximate algorithm to solve this problem. [15] pays attention to resource-limited crowdsensing environment, they propose an algorithm to take logical data dependencies into consideration for the purpose that application queries are answered at the central aggregation node. CARDAP [16] is proposed as a scalable, energy-efficient distributed data analytics platform for mobile crowdsensing.

The above researchers' achievements allocate the resources efficiently, but they could not be directly used in video crowdsensing because the data type they sensed is not video.

3. Problem descriptions

In this section, we describe the problem formulation from the following three aspects: network model, mobility pattern and caching scheme. Then, we further introduce the confronting problems in this paper.

3.1. Network model

There are n users (user set U) moving around a specific area, each user carries a mobile phone and freely takes videos for his/her interested events. Then, the videos are stored in the local cache. A video requester could publish his/her requirement or task to the cloud server. Then the users would like to upload the videos to the cloud server and get the corresponding reward. Because the video's size is very large in most cases, a user prefers to upload the videos to the cloud server through a free way (wifi) instead of uploading it costly (3G/4G). In this paper, we do not pay attention to the payment game or management, while we focus on the video replacement policy in each user's device. This is because the caching space serving crowdsensing tasks is usually limited, and is not enough for caching a large amount of videos. Moreover, we assume that each user knows the global information (through the cloud server) in terms of cache distributions of all users. The global information includes some two-tuples ⟨beginning time, ending time⟩ of video segments. When a user's cache overflows, then it asks the server to collect the global information from the other users. The capturing process does not need to upload the video segments, while just uploading some two-tuples in text message from the other users. Hence, we assume the uploading cost is acceptable through 3G/4G, while the uploading of videos is very expensive and needs to use the free wifi. More importantly, when the server sends global information to the user, it also adds the information of successfully uploaded video segments, which could be used to make local replacement decision. After getting the global information, a user could make an accurate replacement decision of video segments in its local cache. Each user could predict the frequency to access the wifi areas through the method in the next subsection. While we assume that frequency of getting the new video from an event or another neighbor user is unknown and such frequency does not follow any particular distribution.

3.2. Mobility pattern

In the above network environment, the users primarily utilize occasional encounters with the fixed wifi areas to upload the cached videos. Therefore, the access intermeeting time to the wifi areas will seriously influence the uploading frequency. We first define the access intermeeting time as follows:

Definition 1. Access intermeeting time is the elapsed time from the end of the previous access to a wifi area to the start of the next access.

The existing work [8] has proved that intermeeting time tails off exponentially in many mobility patterns, such as random-waypoint and random direction. In order to prove that the access intermeeting time in our traces also matches an exponential distribution, we test the distribution of access intermeeting time as shown in Fig. 2, the access intermeeting time nearly follows an exponential distribution for the two real-world traces (StudentLife and Buffalo/phonelab-wifi datasets):

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{1}$$

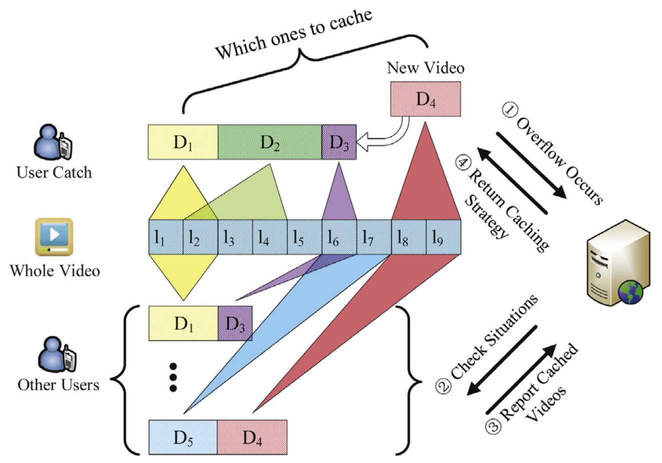


Fig. 3. The framework of budgeted video caching in video crowdsensing. The video replacement policy in a single user is determined by the stored videos of the other users' caches.

Suppose that λ matches the exponential distribution of access intermeeting time, E_t represents the mathematical average values; then $\lambda = \frac{1}{E_t}$. The parameter λ of StudentLife dataset could be achieved through Fig. 2: 0.005781, and that of Buffalo/phonelab-wifi dataset is 0.02495. Hence, the expected access intermeeting time for the above datasets are 172.98 and 40.67 time units, respectively. These are also the settings in the simulations of this paper.

3.3. Caching scheme

Each user freely takes videos and stores the videos in its local cache. The whole required video published by the task requester is equally divided into m time units. Then, a video segment (D_i) cached by a user may cover parts of the whole required video. Hence, for each user, it is difficult to decide which videos should be kept in the cache. More importantly, when the multi-user situation is considered, the problem becomes more challenging. This is because we need to take the other users' caching situations into consideration when we decide the local replacement policy.

As shown in Fig. 3, video segments D_1, D_2 and D_3 take up all the available cache of the first user. At this time, when D_4 is taken by the user, then it should decide which video segments to reserve and which to replace. This is not an easy problem because the video segments are dependent of each other. Moreover, the multi-user situation also leads to a more complex calculation process.

As mentioned in the network model, we assume that, the communication cost (not uploading cost) for a user to get parameters from cloud server could be neglected. Hence, when a user's cache overflows, it notifies the cloud server and searches for the situations of other users. Then, with the purpose of maximizing the uploaded videos' total utility, the overflowed user could efficiently manage its local cache.

3.4. Main thoughts

Facing the above challenges, we attempt to propose a budgeted video replacement policy in mobile video crowdsensing. First, we decide an efficient calculation method to measure the degree of video's importance. Then, we formulate the budgeted video caching problem as the budgeted coverage problem, which is a well-known NP-hard problem. In order to solve the NP-hard problem, we further propose a practical greedy solution. Finally, we infer the approximate ratio between the results of BVCV and the optimal solution, we also test the performance of proposed policy and prove the approximate ratio through two widely-used real-world traces.

Table 1
Main notation used throughout the paper.

Symbol	Meaning
U	The user set
B	The cache space for a user
m	The number of video units
L	The whole required video L, $L = \{l_1, l_2, \ldots, l_m\}$
V_{l_i}	Value of video unit l_i
D	The video segment set
D_i	The i_{th} kind of video segment in D: $D_i \in D$
b_i	The required cache space of D_i
P_i	Probability of uploading l_i successfully
n_i	The number of users caching video unit l_i
λ	Parameter in the exponential distribution of access intermeeting time
V_i'	The increased value of D_i

4. Budgeted video replacement policy

In this section, we first model the optimization problem for the budgeted video replacement policy in mobile video crowdsensing. Then, we prove the NP hardness of the optimization problem. Next, we propose the corresponding greedy algorithms to solve the NP-hard problem. Finally, the approximate ratio is inferred to guarantee the performance of proposed algorithm. Main notations used throughout this paper are illustrated in Table 1.

4.1. Optimization problem

There are n users moving around a specific area: $U = \{u_1, u_2, \ldots, u_n\}$. They freely take videos of their interested events or activities through their mobile phones, and then they store the videos in their local caches. Hence, each user could be regarded as a mobile device with limited caching space. Here, we assume that the devices have the uniform cache space B. This assumption is just used to simplify the expression of the optimization problem. In other words, the heterogeneous cache spaces will not lead to a more difficult situation, and the policy proposed in this paper is still available for the heterogeneous cache spaces.

We consider the following mobile video crowdsensing scene: a requester publishes its video sensing task in the cloud server for collecting the whole video for an event or activity. The whole required video, which is long enough to cover the task requirement is defined as follows: $L = \{l_1, l_2, \ldots, l_m\}$, which means that we equally divide the whole required video into m units. The time unit of l_i is t_{unit}, which could be achieved by the following equation.

$$m \cdot t_{unit} = t_{end} - t_{init}. \tag{2}$$

Where t_{end} is the required video's end time of the sensing task, while t_{init} is the required video's start time of the sensing task.

In order to clearly describe the modeling process, we first give the following definitions.

Definition 2. l_i is the ith video unit in the whole required video of the task. The units are obtained from the equally division of the whole required video.

Definition 3. D is the video set including all kinds of video segments taken by the mobile devices.

Definition 4. D_i is ith kind of video segment. Hence, $D = \{D_1, D_2, \ldots, D_h\}$.

Obviously, a video taken by a user may be part of the required task video. In other words, D_i may cover part of the whole required video L, which means that $D_i = L'_i \subseteq L$. The purpose of this paper is to measure the tradeoff between budgeted caching space and the video's value. Therefore, we give the following definition.

Definition 5. b_i is the required caching space for video segment D_i. Without loss of generality, we assume that the caching space of l_i is 1, then b_i is the total length of the video segment D_i.

After giving the definition of b_i, we focus on the value of video segment D_i. D_i consists of a series of l_i, and the value for l_i is given in Eq. (3). However, different from budget b_i, the value of D_i could not be directly calculated by the sum value of l_i. This is because the delivery of D_i also depends on the caching situations of other users. Hence it is a probabilistic problem for measuring the actual value of D_i.

$$\{V_{l_i}\}_{i=1}^{m} = 1 \tag{3}$$

Focusing on the probabilistic problem, we first take the mobility pattern into consideration. We attempt to measure the expected contribution of the video segment D_i. If there is only one device in the mobile video crowdsensing, then the problem becomes easy to solve. The value of video segment D_i depends on both the video length (the number of time unit l_i) and the delivery probability (the probability for a user to be in a wifi area before the task deadline). The video length is easy to obtain. And the delivery probability could be calculated through the access intermeeting time distribution, which is discussed in the previous section.

However, if there are multi-users in the crowdsensing activity, the problem becomes more difficult. This is because we need to consider the caching situations of the other users and calculate the improvement in delivery probability for storing this video in the local cache. To this end, according to the previous discussions in terms of mobility pattern, we first attempt to calculate the expected delivery probability of video unit l_i. Suppose that there are totally n_i mobile devices carrying video unit l_i in their caches at the moment. Then the access intermeeting time rate changes from λ to $n_i\lambda$. This is because we assume that all the mobile devices match the same exponential distribution. When the task deadline is T, we could achieve the probability of uploading l_i successfully before T as follows:

$$P_i = 1 - e^{-\lambda n_i T} \tag{4}$$

In the above situation, when user a needs to decide whether to cache l_i, it should judge the expected achievement of caching the video unit l_i. In this way, it could calculate the expected value of video segment D_i through the total sum of l_i covered by D_i. Suppose that, there are n_i mobile devices except user a caching video unit l_i. Then if user a decides to cache l_i, the total number of copy l_i is $n_i + 1$. If we want to measure the improvement in delivery ratio for caching l_i in user a, we just need to calculate the derivative of P_i as a function of n_i.

However, based on whether there is a user carrying video unit l_i, the results could be divided into the following two situations. When there is no user carrying l_i, i.e., $n_i = 0$, then the expected improvement of delivery ratio (successfully uploading) is $1 - e^{-\lambda T}$. If there are n_i users caching l_i, then the improvement is the derivative of P_i: $\lambda T e^{-\lambda T n_i}$. In this way, we could achieve the expected value of l_i.

$$f(n_i) = \frac{\mathrm{d}P_i}{\mathrm{d}n_i} = \begin{cases} \lambda T e^{-\lambda T n_i} & n_i \geq 1 \\ 1 - e^{-\lambda T} & n_i = 0 \end{cases} \tag{5}$$

Next, we still use the example of user a. The total video segment set is denoted by D. When the overflowing occurs in user a, we suppose that the already existing video segments and the new coming one form the video segment set $D'_i \subseteq D$. Because the caching space is not enough to cache all the video segments in D'_i, we should select a subset $D''_i \subseteq D'_i$ to be cached by user a in order to maximize the total utility of cached videos.

Then, each video segment has a caching space as well as a delivery value. So in the limited caching space, we should balance the value and caching space. This is obviously an optimization problem. Here, as previously described, the final cached video segment set is D''_i, then all the video unit sets L'' covered by D''_i are shown as follows:

$$\bigcup_{D_i \in D''_i} D_i = L'' \subseteq L \tag{6}$$

The purpose of this paper is to decide a suitable video set D''_i under the limited budget B, so as to maximize the total utility of D''_i. First, in order to make the optimization problem easy to understand, we consider an easy case, where the value of D''_i depends on the total length of video units covered by D''_i. Thus, the optimization problem turns out to be the following equation:

$$\begin{aligned} &Maximize\ V(L'') = |L''| \\ &s.t. \quad \sum_{D_i \in D''_i} b_i \leq B \end{aligned} \tag{7}$$

However, in the previous optimization problem, the value of l_i is simply regarded as 1. Actually, the value of l_i depends on many influence factors, which have been discussed in the previous subsections. The value of l_i could be achieved in a more reasonable way:

$$V_{l_i} = f(n_i), \tag{8}$$

where $f(n_i)$, which has been defined in Eq. (5), represents the expected improved value when we cache one more copy of video unit l_i.

Then, we take the value of video segment D_i into consideration, it could be calculated through the sum value of l_i covered by D_i. Then Eq. (9) is achieved.

$$V(D_i) = \sum_{\forall l_i \in D_i} f(n_i) \tag{9}$$

If D_i is the new coming segment, the above equation could measure its value accurately. However, if D_i is the video segment which already exists in the cache, then there must be a cost for replacing D_i. This is mainly because the following two reasons: (1) replacing an existing segment requires an additional caching operation; (2) the existing video segment is likely to being used by the user. Taking the above two cases into consideration, we give the modified utility of the segment D_i, which already exists in the cache.

$$V(D_i) = \sum_{\forall l_i \in D_i} f(n_i) + \alpha_i, \tag{10}$$

where α_i means the value increment compared to a new coming one. It is not difficult to find that, the above two items in Eq. (10) could be both regarded as the discount for the total length of time for the video segment D_i, so they could be added together. Moreover, if the user deletes an existing segment, the cost caused by replacing operation and reducing user's interest is presented by α_i:

$$\alpha_i = x(1 - e^{-\lambda x}) \tag{11}$$

In the above equation, x is the total length of D_i. Generally speaking, replacing a longer video segment leads to a higher deleting operation cost. Meanwhile deleting a longer video segment also leads to a higher probability to influence the user's emotion because perhaps they are watching the videos. Moreover, we consider that the reducing of user's emotion should be exponential descent function, while not proportional function. This is because, if we delete the half length of the movie, a user may have no interest to watch it. Hence, we use $x(1-e^{-\lambda x})$ to measure the value increment of existing video segment. Without loss of generality, we set λ as $\frac{10}{m}$, then, when x is approaching $\frac{m}{2}$, the interest of user is almost zero. When x is approaching 0, then there will be no influence on user's emotion.

Finally, the optimization problem turns out to be finding a suitable set of D_i'' under the constraint that $\sum_{D_i \in D_i''} b_i \leq B$, with the purpose of maximizing the total value of video units ($V(L'')$) covered by D_i''.

$$\begin{aligned} &Maximize\ V(L'') = \sum_{\forall l_i \in L''} V_{l_i} \\ &s.t. \quad \sum_{D_i \in D_i''} b_i \leq B \end{aligned} \tag{12}$$

4.2. NP-hard proof

Before solving the above optimization problem, we first attempt to prove that the budgeted caching optimization problem is NP-hard, as shown in the following theorem.

Theorem 1. *The budgeted caching optimization problem as shown in Eq. (12) is NP-hard.*

Proof. First of all, we attempt to formulate the optimization problem in Eq. (12) as a budgeted maximum coverage problem, which includes a collection of sets $X = \{X_1, X_2, \dots, X_m\}$ with the corresponding costs $c = \{c_1, c_2, \dots, c_m\}$. X_i is defined by a domain of elements $O = \{O_1, O_2, \dots, O_n\}$ with the associate weights: $w = \{w_1, w_2, \dots, w_n\}$. The purpose is to select a subcollection of X, and the total cost of the elements in selected subcollection does not exceed a given budget, while the total weight of the selected subcollection is maximized.

Then, we consider the optimization problem in this paper again. An easier situation is that all the video units have a uniform value of 1. Then the video segment D could be mapped into X, and the caching space b is mapped into c. Moreover, the value of video segment D_i is regarded as w_i. It is not difficult to find that, the simplified budgeted video caching problem is the same as the budgeted maximum coverage problem. So if we could prove that the budgeted maximum coverage problem is NP-hard, then the budgeted video caching problem is also NP-hard.

Next, we also consider an easier case of budgeted maximum coverage problem, where the cost for each X_i is uniform. Then the problem becomes selecting the number k set of X_i so that the total weight of covered elements is maximized. The uniform budgeted maximum coverage problem is obviously NP-hard, this could be inferred from the set cover problem: a task set κ is provided, a collection of subset $\{\kappa_i | 1 \leq i \leq n\}$. How to determine a k size of subcollection of $\{\kappa_i | 1 \leq i \leq n\}$ that covers as many as possible tasks in κ is obviously NP-hard. Hence, the original budgeted video caching problem in this paper is at least NP-hard. □

4.3. Algorithms and approximate ratio

In this subsection, we attempt to propose an efficient greedy algorithm to solve the NP-hard problem. The main idea is to measure the trade-off between the limited caching space and the total uploaded video length. We have proved that the budgeted video caching problem is NP-hard. Then in order to propose a greedy algorithm to solve the NP-hard problem, we should decide a measuring method to order the priority among the existing video segments and the new coming one. In this way, we could decide the suitable replacement policy in this paper.

Algorithm 1 BVCV-no bound

```
1: $M \leftarrow \emptyset$, $b(M) \leftarrow 0$, $U \leftarrow D_i'$
2: while $U \neq \emptyset$ do
3:   find $D_i \in U$ that maximize $\frac{V_i'}{b_i}$
4:   if $b(M) + b_i \leq B$ then
5:     $M \leftarrow M \cup D_i$
6:     $b(M) \leftarrow b(M) + b_i$
7:     $U \leftarrow U \backslash D_i$
8: return $M$
```

Focusing on the priority of video segment D_i, we should first consider the expected improvement on delivery performance, which is defined as the increased value of D_i: V_i'. In other words, V_i' is the expected increased value for caching D_i in its local cache, which is shown in the following equation, where M is the existing video set:

$$V_i' = V(M \bigcup D_i) - V(M) \tag{13}$$

However, the expected increased value for caching D_i could not be directly used for measuring the priority. This is because a video segment has a high expected increased value and may also occupy a large caching space. Then caching this video segment in a high priority is not necessarily the best choice. It is intuitively obvious that a video segment with the highest increased value per unit caching space should be assigned the highest priority. In other words, we should cache the video segment D_i with the highest value of $\frac{V_i'}{b_i}$ in the highest priority.

To sum up, Algorithm 1 (BVCV-no bound) is proposed to first cache the video segment with the highest value of $\frac{V_i'}{b_i}$, until the budget is not enough. This is obviously an easy solution to the NP-hard problem. Unfortunately, Algorithm 1 has an unbound approximate ratio, which could be proved as follows.

Consider a special case that, the budget is $h+1$, and there are only two video segments: D_1 and D_2 to be cached. D_1 covers video unit l_1 with value of 1 and occupies $b_1 = 1$, while D_2 covers video unit l_2 with value of h and occupies $b_2 = h+1$, where h is positive. Obviously, the optimal solution is caching D_2, and it achieves the value of h. However, the solution picked by Algorithm 1 is D_1, and it achieves the value of 1. It is worth noting that, the approximate ratio for this example is $\frac{1}{h}$, and is therefore unbounded.

Next, in order to propose a greedy algorithm within a controlled bound, we do a small modification to Algorithm 1. Then Algorithm 2 is proposed to solve the above NP-hard problem. The main idea is achieving the final solution through two ways: the first one is the same as that of Algorithm 1, the second one is the single set of D_i with the highest value: $V(D_i)$. Then, the higher value between the first and second ones is regarded as the final solution of Algorithm 2. In this way, the unbounded case in Algorithm 1 is solved. When we use Algorithm 2 to address the previous example, the first part is the same as Algorithm 1, and it selects D_1 as the solution. While the second part selects D_2 as the final solution. Obviously, when $h > 1$, the value of D_2 is higher than that of D_1, then D_2 is selected as the final solution. Otherwise, D_1 is selected as the final solution. Therefore, the performance of algorithm is bounded. Then, we give the following important theorem.

Theorem 2. *The solution of Algorithm 2 (BVCV-loose bound) achieves the approximate ratio of $\frac{1}{2}(1-\frac{1}{e})$ to the optimal solution.*

Algorithm 2 BVCV-loose bound

```
1: $M_1 \leftarrow \text{argmax}\{V(D_i), D_i \subseteq D'_i$, and $b(D_i) \leq B\}$
2: $M_2 \leftarrow \emptyset$
3: while $U \neq \emptyset$ do
4:     find $D_i \in U$ that maximize $\frac{V'_i}{b_i}$
5:     if $b(M_2) + b_i \leq B$ then
6:         $M_2 \leftarrow M_2 \cup D_i$
7:         $b(M_2) \leftarrow b(M_2) + b_i$
8:         $U \leftarrow U \backslash D_i$
9: if $V(M_1) > V(M_2)$ then
10:     return $M_1$
11: else
12:     return $M_2$
```

Algorithm 3 BVCV-tight bound

```
1: $M_1 \leftarrow \text{argmax}\{V(E), E \subseteq D'_i, |E| < k$, and $b(E) \leq B\}$
2: $M_2 \leftarrow \emptyset$
3: for $E \subseteq D'_i, |E| = k$ and $b(E) \leq B$ do
4:     $U = D'_i \backslash E$
5:     while $U \neq \emptyset$ do
6:         find $D_i \in U$ that maximize $\frac{V'_i}{b_i}$
7:         if $b(E) + b_i \leq B$ then
8:             $E \leftarrow E \cup D_i$
9:             $U \leftarrow U \backslash D_i$
10:    if $V(E) > V(M_2)$ then
11:        $M_2 \leftarrow E$
12: if $V(M_1) > V(M_2)$ then
13:     return $M_1$
14: else
15:     return $M_2$
```

Proof. The proof of Theorem 2 is shown in Appendix A, where we prove that the approximate ratio for Algorithm 2 to the optimal solution is $\frac{1}{2}(1-\frac{1}{e})$. Algorithm 2 selects the maximum value of D_i as the candidate solution, which prevents the result from unbounded approximate ratio. In this way, the approximate ratio changes from an unbounded value to $\frac{1}{2}(1-\frac{1}{e})$. □

Through Theorem 2, we propose Algorithm 2, which gets a lower bound of $\frac{1}{2}(1-\frac{1}{e})$ to the optimal solution for the budgeted video caching problem. However, the bound is still a loose bound. Although the budgeted video caching problem is not a submodule function, the bound could also come close to $1-\frac{1}{e}$. Hence, we attempt to propose the greedy algorithm with the tight bound. Then Algorithm 3 (BVCV-tight bound) is proposed.

The main thought of Algorithm 3 is shown as follows. We use enumeration method to address the NP-hard problem. k is a fixed number to record the enumeration level. Similar to Algorithm 2, we also divide the final solution into two parts. The first part consists of all the subsets of D'_i of cardinality less than k within the caching budget. The second part considers all the subset of D'_i in k items. And then, we repeat greedy solution until the budget is not enough. Finally, the higher value between the first part and second part is regarded as the final solution of Algorithm 3. According to the following important theorem, the bound of Algorithm 3 is given.

Theorem 3. *The solution of Algorithm 3 (BVCV-tight bound) achieves the lower bound of $1-\frac{1}{e}$ to the optimal solution for $k \geq 3$.*

Proof. The proof of Theorem 3 is shown in Appendix B, where we prove that the approximate ratio for Algorithm 3 to the optimal solution is $1-\frac{1}{e}$, which is the same as the algorithm that matches the submodule function. Algorithm 3 uses enumeration method to achieve a satisfactory tight bound, which changes from $\frac{1}{2}(1-\frac{1}{e})$ to $1-\frac{1}{e}$. Obviously, in the real crowdsensing system, the maximum number of video segments cached by a user is usually more than 3. Hence, the tight bound could be achieved. □

To sum up, three greedy algorithms: BVCV-no bound, BVCV-loose bound and BVCV-tight bound are proposed to address the budgeted video caching problem, which is proved to be NP-hard. The three algorithms' approximate ratios are getting tighter and tighter. Finally, in this paper, we achieve the BVCV-tight bound algorithm, whose bound comes close to $1-\frac{1}{e}$.

5. Performance evaluation

By now, we have introduced the proposed video replacement policy, and also described the corresponding algorithms to solve the NP-hard problem. Then, in this section, we attempt to do the performance evaluations to test the following two aspects: (1) BVCV actually achieves the highest performance in terms of successfully delivered video length. (2) BVCV-tight bound actually achieves the approximate ratio of $1-\frac{1}{e}$ to the optimal solution. In order to test the practicability of BVCV, we conduct simulations on two widely-used real-world traces. Two situations (ordered generated videos and randomly generated videos) are considered in the following simulations.

5.1. Evaluation settings

To test the performances of the proposed algorithms, we first introduce the details of the real-world datasets. We adopt two widely-used real-world traces, StudentLife dataset [27] and Buffalo/phonelab-wifi dataset [24] to test the performances of the proposed video replacement policy. StudentLife is the dataset which contains all the sensor data (wifi, bluetooth, *etc*), EMA data, survey responses and educational data collected from the phones of 48 Dartmouth students over a 10-week term to assess their mental health, academic performance and behavioral trends. PhoneLab is a large scale smartphone test-bed at University at Buffalo (UB). Participants carry instrumented Nexus 5 smartphones as their primary devices. The dataset uses the PhoneLab platform to collect records, which contain the wifi scan results reported by 284 smartphones from 2014-11-07 to 2015-04-03. The PhoneLab participants are primarily UB faculty, staff and students, so most of the wifi networks reported are UB campus wifi network or each participant's home wifi network.

We first address the above two datasets by filtering out some abnormal user records (users of missing data or with abnormal behaviors). Based on the addressed users' traces, we list all the time records for a user to enter a wifi area. Then we could also get the access intermeeting time for all the users. For StudentLife dataset, the data is the participants' wifi AP scan log file, the items in concern are time and ID of wifi area. One scan log contains a set of IDs of wifi area and if the IDs in several continuous records have the intersection, we regard the time of the first record as the connection time until the ID of next record does not intersect with that of the last record. Finally we extract all the connection time records into files. For Buffalo/phonelab-wifi dataset, we first select the data of mobile users who have wifi connection records every day and the data item in concern includes the ID of users, time-stamp, connection status and ID of wifi area. If the mobile user connects the access point successfully, the connection status is true, otherwise it is false. The number of users that meet the requirement is 73. If the user connects the same access point in several continuous records, we regard the time-stamp of the first record as the connection time until the connection status of

Table 2
Simulation settings.

Parameter	Datasets	
	StudentLife	Buffalo/phonelab-wifi
User number	49	73
Cache budget	500 ∼ 1200	
Time unit (s)	10	
Task deadline	500 ∼ 1500	
Total video length	5000 ∼ 15, 000	
λ	0.005781	0.02495
Generation interval	5 ∼ 20	

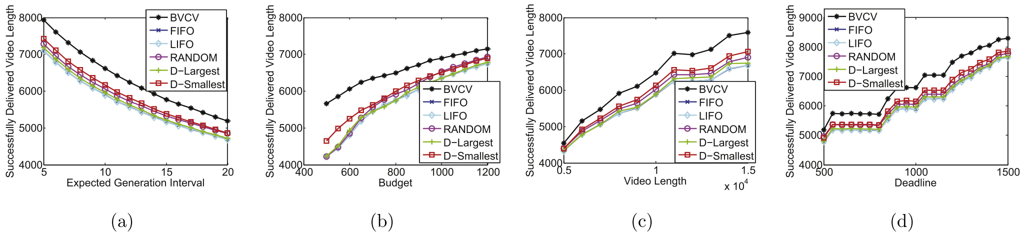


Fig. 4. The performance changing process along with the growth of different conditions under the ordered generated videos in StudentLife dataset.

record is false or the ID is another access point. Subsequently, we record the connection time of each user.

Then, we plot the curve for the distribution of access intermeeting time as shown in Fig. 2. The simulation results show that, the access intermeeting time for both the two datasets satisfies an exponential distribution. The detailed simulation parameters in this network environment are listed in Table 2. Without loss of generality, we set the generation frequency of new video segments as 5 to 20, which means that in each user, a new video segment is produced every 5 to 20 time units. In particular, time unit is equal to 10, which means that among all the items related to time, the minimal time unit is 10 s. Hence, for example, if the deadline is 500, it means that after 5000 s, the task will be closed. While the generation interval is equal to 10, which means that the video segment is generated per 100 s.

5.2. Algorithms and performances in comparison

To test the performance of the proposed video replacement policy, we divide the simulation results into the following two aspects: (1) delivery performance; (2) approximate ratio.

For the first part, we attempt to test that whether BVCV could achieve the highest successfully delivered video length compared with other replacement policies. The compared policies [2] include FIFO, LIFO, RANDOM, D-Largest and D-Smallest. The detailed information is listed as follows: FIFO first replaces the video segment which first comes in the cache. LIFO first replaces the video segment which just comes in the cache. RANDOM randomly replaces a video segment in its cache. D-Largest first replaces the video segment which occupies the largest caching space. While D-Smallest first replaces the video segment which occupies the smallest caching space.

For the second part, we attempt to prove that the proposed BVCV-tight bound could be bounded by the approximate ratio of $1 - \frac{1}{e}$ to the optimal solution. The tested algorithms include the BVCV-tight bound, which is described in Algorithm 3, and the optimal solution, which is achieved through the exhaustive method.

While a range of data is gathered from the experiments, we take the following two main performance metrics into consideration:

(1) Successfully delivered video length, which is the total uploaded nonoverlapping video length by all the mobile users before the deadline.
(2) Approximate ratio, which is the ratio between the successfully delivered video length of BVCV-tight bound to that of the optimal solution.

5.3. Simulation results under studentlife dataset

5.3.1. Simulation under the ordered generated videos

For the first part in Buffalo/phonelab-wifi dataset, the simulation settings are the same with that of StudentLife Dataset.

First of all, as shown in Fig. 4(a), six video replacement policies are tested along with the growth of expected generation intervals. It is not difficult to find that, along with the growth of x-axis, the successfully delivered video length decreases. This is easy to understand because a longer generation interval leads to a lower chance to upload the video segments. Hence the total successfully delivered video length becomes shorter. Most importantly, along with the growth of generation interval, BVCV always achieves the highest successfully delivered video length, compared with the other replacement policies.

Secondly, when the expected generation interval for the videos is set as 10, the video length is set as 10,000, and the deadline is 1000. We change the budget to see the changing curve of successfully delivered video length. As shown in Fig. 4(b), six video replacement policies are tested along with the growth of budget. Obviously, along with the growth of x-axis, the successfully delivered video length increases. This is because a larger budget results in a more powerful caching ability, which leads to a higher successfully delivered video length. It is worth noting that BVCV still achieves the highest successfully delivered video length, compared with the other replacement policies.

Thirdly, when the budget is 850, and the expected generation interval for the videos is set as 10, the deadline is still 1000. We change the video length to see how the curve of successfully delivered video length changes. As shown in Fig. 4(c), six video replacement policies are tested along with the growth of total video length. Obviously, along with the growth of x-axis, the successfully delivered video length increases. This is because a longer video length results in a longer expected successfully delivered video length. Similar to the previous results, BVCV also achieves

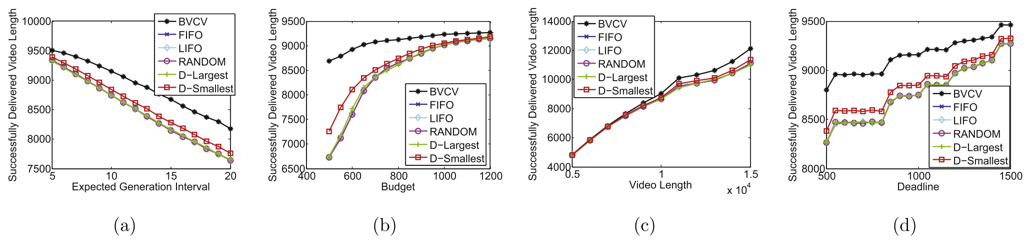


Fig. 5. The performance changing process along with the growth of different conditions under the randomly generated videos in StudentLife dataset.

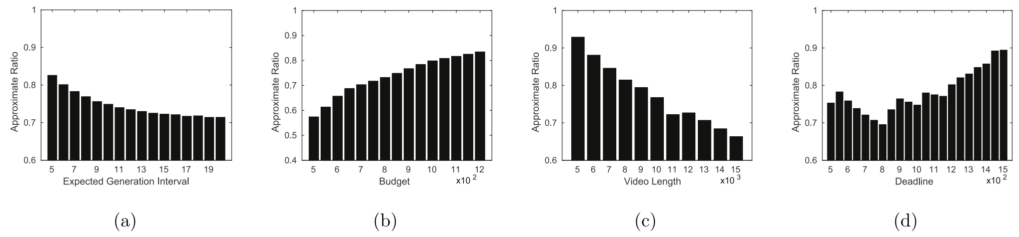


Fig. 6. The approximate ratio performance of BVCV-tight bound to the optimal solution in StudentLife dataset.

the highest successfully delivered video length, compared with the other replacement policies, along with the increase in total video length.

Finally, we set the budget as 850, and set the expected generation interval for the videos as 10, and the total video length is 10,000. We change the deadline to check the changing trend of the six different replacement policies. As shown in Fig. 4(d), among the six replacement policies, BVCV is always the one with the longest successfully delivered video length, along with the growth of deadline. Moreover, along with the growth of deadline, all the six replacement policies achieve the higher delivery performances for the required video length. This is because a longer deadline leads to the more chances to upload the videos.

In conclusion, BVCV proposed in this paper always achieves the highest successfully delivered video length, compared with the other five different replacement policies in the situation of ordered generated videos.

5.3.2. Simulation under the randomly generated videos

In this subsection, we consider the situation of randomly generating videos, which means the order of videos to access a user's cache is totally random. We also mainly consider four factors including: expected video interval, budget, video length, deadline. We fix three factors, and change one factor, to see the changing curve of the totally uploaded video length and also compare it with the previous situation.

The simulation settings for the situation of randomly generated videos is almost the same as that of the situation in the ordered generated videos. However, the simulation results are not the same. For example, as shown in Fig. 5(a), the curves for the six different replacement policies are similar to that of Fig. 4(a), and BVCV always achieves the highest successfully delivered video length. While the order of the other five replacement policies is different from that in Fig. 4(a). In Fig. 4(a), the delivery performances of FIFO and LIFO are obviously lower than that of RANDOM. After analysis, this is reasonable because in ordered generated video sequence, the users in different areas may drop the same video segment in FIFO and LIFO when the caches are overflown. In other words, the initial generated video segments in different users' caches are the same. Hence, when their caches are overflown, they will drop the same video segment. While RANDOM randomly drops video segments when the cache overflows, different users may drop different video segments. So, the total successfully uploaded video length in FIFO and LIFO is lower than that of RANDOM. However, in Fig. 5(a), the situation changes as the video segments are randomly generated for different users, then, FIFO, LIFO and RANDOM actually achieve a similar delivery performance because they almost have no difference in managing the video segments of their caches. Therefore, as shown in Fig. 5(a), the performances for FIFO, LIFO and RANDOM are very similar.

Moreover, we also test the delivery performances for the six replacement policies with different budgets, video lengths and also deadlines. The simulation results show that the variation trends for the six different replacement policies are almost same as that in Fig. 4(a). Specifically, the successfully delivered video length also decreases along with the growth of expected generation interval. Also, the successfully delivered video length always rises along with the growth of caching budget. And it increases along with the increasing of video length and deadline. All the above phenomena are in line with common logic and also make sense.

5.3.3. Simulation for the approximate ratio

In the third part of this section, we focus on testing the tight bound of BVCV. Similar to the previous situation, four factors (expected video generation interval, budget, total video length and also deadline) are considered to influence the final performance of proposed replacement policy. Hence, we test the approximate ratio between BVCV and the optimal solution along with the changes of expected video generation interval, budget, total video length and deadline.

To verify the accuracy of the inferred bound for the proposed BVCV-tight bound, we test all the delivery performances of our policy and the optimal solution. And we also calculate the approximate ratio of our policy to the optimal solution. The results are plotted in Fig. 6, where we find that, along with the growths of the four factors, the approximate ratio of BVCV-tight bound is always larger than 0.66 except the situation that the budget is lower than 550. In the simulation settings, the video segment length can be obtained ranging from 200 to 400 as a random selected way. Hence, when the budget is lower than 550, the expected number of cached video segments is less than 3. Let us look back to Algorithm 3 and Theorem 3, Algorithm 3 achieves an approximate ratio of $1-\frac{1}{e}$ to the optimal solution when $k \geq 3$. Hence, the simulation results match the theoretical results. The approximate ratio of BVCV-tight bound to the optimal solution is actually larger than $1-\frac{1}{e}$ when $k \geq 3$.

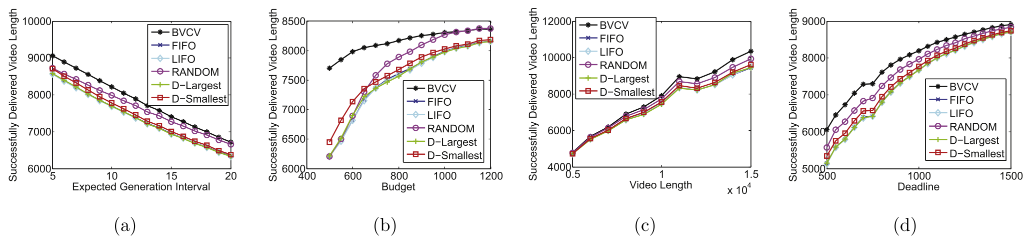


Fig. 7. The performance changing process along with the growth of different conditions under the ordered generated videos in Buffalo/phonelab-wifi dataset.

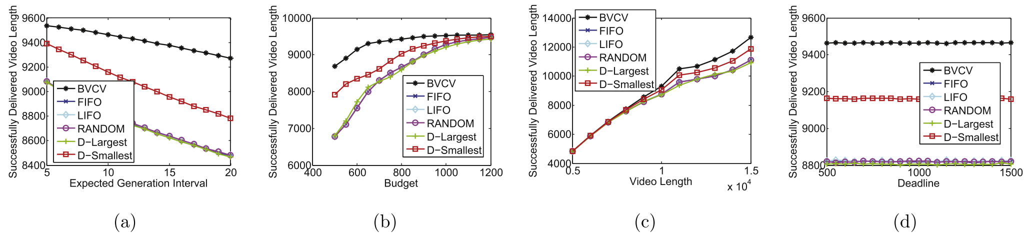


Fig. 8. The performance changing process along with the growth of different conditions under the randomly generated videos in Buffalo/phonelab-wifi dataset.

5.4. Simulation results under Buffalo-wifi

In order to further prove the feasibility of the proposed BVCV replacement policy in mobile video crowdsensing, we conduct simulations under the real-world dataset: Buffalo/phonelab-wifi.

5.4.1. Simulation under the ordered generated videos

For the first part in Buffalo/phonelab-wifi dataset, the simulation settings are the same with that of StudentLife Dataset. For the first group simulation, we set the budget as 850, the video length is 10,000, and the deadline is 1000. We change the expected generation interval to see the changing curve of delivery performance. As shown in Fig. 7(a), six video replacement policies are tested along with the growth of expected generation intervals. Obviously, along with the growth of x-axis, the successfully delivered video length also appears to be a downtrend. This makes sense because a longer generation interval leads to the less chances to upload the video segments, hence the total successfully delivered video length becomes shorter. Most importantly, along with the growth of generation interval, BVCV always achieves the highest successfully delivered video length, compared with the other replacement policies.

Secondly, in a similar way, we set the expected generation interval for the videos as 10, the video length is set as 10,000, and the deadline is still 1000, we change the budget to see the changing curve of successfully delivered video length. As shown in Fig. 7(b), six video replacement policies are tested along with the growth of budget. Similarly, along with the growth of x-axis, the successfully delivered video length increases. This is because a larger budget results in a more powerful caching ability, which leads to a longer successfully delivered video length. To sum up, BVCV still achieves the longest successfully delivered video length, compared with the other replacement policies.

Thirdly, the budget, expected generation interval and deadline are set as 850, 10 and 1000, respectively. We change the video length to see the changing curve of successfully delivered video length. As shown in Fig. 7(c), along with the growth of x-axis, the successfully delivered video length increases. This is because a longer video length results in a larger expected successfully delivered video length.

Finally, we set the budget as 850, and set the expected generation interval for the videos as 10, and the total video length is 10,000. As shown in Fig. 7(d), among the six replacement policies, BVCV is always the one with the highest value for successfully delivered video length, along with the growth of deadline.

5.4.2. Simulation under the randomly generated videos

In this subsection, we focus on the Buffalo/phonelab-wifi dataset. We still mainly consider four factors including: expected video generation interval, budget, video length, deadline. While at this time, we focus on the situation in the randomly generation videos as shown in Fig. 8. It is worth noting that, as shown in Fig. 8(d), when the deadline is 500 time units, the successfully delivered video length has reached more than 9400. And when the deadline is getting larger, the successfully delivered video length will not increase any more. This is mainly because in the randomly generated videos condition, the users in different locations will upload all kinds of videos, so the successfully delivered video length could be achieved in a short time. We omit the detailed descriptions for the simulation results because the results are similar to that of Fig. 5.

5.4.3. Approximate ratio

In the third part, we focus on proving that the tight bound of BVCV in the Buffalo/phonelab-wifi dataset could be achieved. We attempt to test the performance approximate ratio between BVCV and the optimal solution along with the changes of expected video generation interval, budget, total video length and deadline. The results are plotted in Fig. 9, where we find that, the performance of BVCV-tight bound could be bounded by $1-\frac{1}{e}$ compared with the optimal solution.

6. Conclusion

In this paper, we propose a budgeted video replacement policy (BVCV) for mobile video crowdsensing. BVCV first decides the increased value of a video segment by considering the caching situations and video segment's natural attributes. Then, we prove that the video caching problem could be regarded as a budgeted maximum coverage problem, which is at least NP-hard. Next, we propose three greedy algorithms to solve the NP-hard problem: BVCV, BVCV-loose bound, BVCV-tight bound, which are ordered by the value of lowest bound. And we prove that the approximate ratio of the BVCV-tight could be enhanced to $1-\frac{1}{e}$. Moreover, we give the strict proof about the approximate ratio. Finally, our experiments with the real mobility datasets (StudentLife dataset, Buffalo/phonelab-wifi dataset) show that, the proposed budgeted video replacement policy achieves a larger length result of successfully delivered nonoverlapping videos, compared with the other general replacement policies.

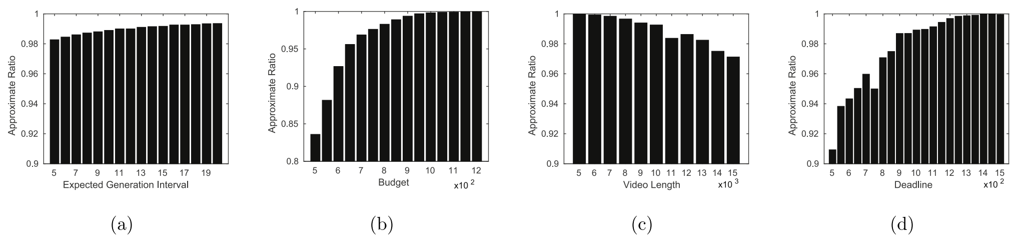


Fig. 9. The approximate ratio performance of BVCV-tight bound to the optimal solution in Buffalo/phonelab-wifi dataset.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 61772230, 61972450 and 61702215, Changchun Science and Technology Development Project No. 18DY005. China Postdoctoral Science Foundation No. 2017M611322. This work is also supported in part by NSF, USA grants CNS 1629746, CNS 1564128, CNS 1449860, CNS 1461932, CNS 1460971, and CNS 1439672.

Appendix A. Proof of Theorem 3

To prove Theorem 2, we need first give Theorems 4 and 5 to support the proof process. Then, we give the following symbols: M_{opt} denotes the optimal solution of the budgeted video caching problem. In Algorithm 2, if a video segment is selected to add to M (repeating this process), we call the process a round. Then, after i rounds, the formulated video segment set is called M_i. Without loss of generality, we reorder the formulated set so that D_i is the ith set added to M_i. And we assume that, the D_{j+1} is the first set from M_{opt} selected by Algorithm 2, while is not added to M_i, because there is no enough caching space. Then, for $M_i, i = 1, 2, \ldots, j$, D_{j+1} is the first set, which is not added into the final solution. The following theorem is achieved.

Theorem 4. *After each round i, $i = 1, 2, \ldots, j+1$, the following equation holds:*

$$V(M_i) - V(M_{i-1}) \geq \frac{b_i}{B}(V(M_{opt}) - V(M_{i-1}))$$

Proof. First of all, we consider all the video segments that have not been added to the final solution after $(i{-}1)$th round. For each video segment in $M_{opt}\backslash M_{i-1}$, the value of its increased value divided by its caching space is at most $\frac{V_i'}{b_i}$. This because D_i is selected to be the highest value of $\frac{V_i'}{b_i}$ in the remaining video segments. Then, after $i{-}1$ rounds, the total remaining caching space is no more than B, the total value of remaining video segments covered by $M_{opt} - M_{i-1}$, while not covered by M_{i-1} is no more than $B\frac{V_i'}{b_i}$. Therefore, we have,

$$V(M_{opt}) - V(M_{i-1}) \leq B\frac{V_i'}{b_i} \tag{A.1}$$

As described for V_i', it could be represented by $V(M_{opt}) - V(M_{i-1})$. Then in the above equation, we replace V_i' with $V(M_{opt}) - V(M_{i-1})$, and multiplying both sides by $\frac{b_i}{B}$. Theorem 4 is achieved. □

Theorem 5. *After each round i, $i = 1, 2, \ldots, j+1$, the following equation holds:*

$$V(M_i) \geq V(M_{opt})[1 - \prod_{k=1}^{i}(1 - \frac{b_i}{B})]$$

Proof. In order to prove the correctness of the above equation. We consider the rounds $i = 1, 2, \ldots, j+1$. When $i = 1$, then $V(M_1) = V_1'$, and we attempt to prove that $V(M_1) \geq \frac{b_1}{B}M_{opt}$. This is not difficult to achieve because $\frac{V(M_1)}{b_1}$ is maximum among all the alternative video segment sets. And the total caching space is B, so the total value of optimal solution is necessarily less than $\frac{V(M_1)}{b_1}B$. With the conclusion that $V(M_1) \geq \frac{b_1}{B}M_{opt}$, we prove that Theorem 5 holds for rounds $j = 1, 2\cdots, i-1$, then it will also hold when $j = i$ through induction. The following equation gives the detailed proof procedure.

$$\begin{aligned} V(M_i) &= V(M_{i-1}) + [V(M_i) - V(M_{i-1})] \\ &\geq V(M_{i-1}) + \frac{b_i}{B}(V(M_{opt}) - V(M_{i-1})) \\ &= (1 - \frac{b_i}{B})V(M_{i-1}) + \frac{b_i}{B}V(M_{opt}) \\ &\geq (1 - \frac{b_i}{B})(1 - \prod_{k=1}^{i-1}(1 - \frac{b_k}{B}))V(M_{opt}) + \frac{b_i}{B}V(M_{opt}) \\ &= (1 - \prod_{k=1}^{i}(1 - \frac{b_k}{B}))V(M_{opt}) \end{aligned} \tag{A.2}$$

In the above proving process, the first inequation is based on Theorem 4. And the second one is based on induction. □

Based on the above results, now we attempt to prove the correctness of Theorem 2. According to the results of Theorem 5, we have the following derivation process.

$$\begin{aligned} V(M_{j+1}) &\geq (1 - \prod_{k=1}^{j+1}(1 - \frac{b_k}{B}))V(M_{opt}) \\ &\geq (1 - (1 - \frac{1}{j+1})^{j+1})V(M_{opt}) \\ &\geq (1 - \frac{1}{e})V(M_{opt}) \end{aligned} \tag{A.3}$$

The second inequality of the above equation is due to the following reasons. There are n variables: $a_1, a_2, \ldots, a_n$, and the sum of them is shown as follows: $a_1 + a_2 + \cdots + a_n = C$. If and only if $a_1 = a_2 = \cdots = a_n = \frac{C}{n}$, then $a_1a_2\cdots a_n$ achieve the maximum value.

Finally, $V(M_{j+1})$ is equal to $V(M_j) + V_{j+1}'$. Then we consider the equation $V(M_j) + V(M_1)$, where $V(M_1)$ is the maximum value among all the alternative single video segment sets. Obviously, $V(M_1) > V_{j+1}'$, hence we have the following equation.

$$V(M_j) + V(M_1) \geq V(M_{j+1}) \geq (1 - \frac{1}{e})V(M_{opt}) \tag{A.4}$$

The solution for Algorithm 2 is achieved by the higher value between $V(M_j)$ and $V(M_1)$. Hence the value of Algorithm 2's solution is at least $\frac{1}{2}(1-\frac{1}{e})V(M_{opt})$. Theorem 2 holds.

Appendix B. Proof of Theorem 4

First of all, Algorithm 3 adopts enumeration method to address the NP-hard problem. Obviously, we could assume that $|M_{opt}| > k$. Otherwise, Algorithm 3 must find the optimal solution.

Then, we first order the optimal solution M_{opt} as the following way: the first k video segments are the ones with the highest total value. We use M_k to express the first k sets in this order, and it is not difficult to find that M_k will be also considered in Algorithm 2. Moreover, M' represents the further video segments added to M_k. After j rounds, the video segment set M_j consists of two parts: M' and M_k. Then we have $V(M_j) = V(M_k) + V(M')$.

Here we could divide the M_{opt} into the following two parts: the first one is M_k, and the other part from M_k to M_{opt} could be regarded as an application of Algorithm 2. Then we consider the $M_{opt} \backslash M_k$ as the rest optimal solution. We focus on the process from M_k to M_{opt}, the greedy process ceaselessly adds new video segments to M_k, until the first video segment from $M_{opt} \backslash M_k$, while not added to M' due to the limited budget B. Similar to the process of Algorithm 2, when the round is $j+1$, the following equation could be achieved through Eq. (A.4).

$$V(M') + V(D'_{j+1}) \geq (1-\frac{1}{e})V(M_{opt} \backslash M_k) \tag{B.1}$$

Then, because the k sets in M_k are ordered, the value of the set in M_k is at least $V(D'_{j+1})$. Otherwise, it violates to the ordering assumption. Hence, we have:

$$V(M_k) \geq kV(D'_{j+1}). \tag{B.2}$$

Now we get the following equation, Theorem 3 holds.

$$\begin{aligned} V(M_j) &= V(M_k) + V(M') \\ &\geq V(M_k) + (1-\frac{1}{e})V(M_{opt} \backslash M_k) - V(D'_{j+1}) \\ &\geq V(M_k) + (1-\frac{1}{e})V(M_{opt} \backslash M_k) - \frac{V(M_k)}{k} \\ &\geq (1-\frac{1}{k})V(M_k) + (1-\frac{1}{e})V(M_{opt}) \\ &\geq (1-\frac{1}{e})V(M_{opt}), \quad k \geq 3 \end{aligned} \tag{B.3}$$

References

[1] Luigi Atzori, Roberto Girau, Salvatore Martis, Virginia Pilloni, Marco Uras DIEE, A SIoT-aware approach to the resource management issue in mobile crowdsensing, in: Proc. of IEEE ICIN, 2017.

[2] Aruna Balasubramanian, Brian Neil Levine, Arun Venkataramani, DTN routing as a resource allocation problem, in: Proc. of ACM SIGCOMM, 2007.

[3] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, Crowdsourcing with smartphones, IEEE Internet Comput. 16 (5) (2012) 36–44.

[4] Huihui Chen, Bin Guo, Zhiwen Yu, Qi Han, Toward real-time and cooperative mobile visual sensing and sharing, in: Proc. of IEEE INFOCOM, 2016, pp. 1–9.

[5] Huihui Chen, Bin Guo, Zhiwen Yu, Qi Han, A generic framework for constraint-driven data selection in mobile crowd photographing, IEEE Internet Things J. 4 (1) (2017) 284–296.

[6] Jiang Dong, Yu Xiao, Marius Noreikis, Zhonghong Ou, Antti Yla-Jaaski, iMoon: Using smartphones for image-based indoor navigation, in: Proc. of ACM SenSys, 2015, pp. 85–97.

[7] Rong Du, Carlo Fischione, Ming Xiao, Flowing with the water: On optimal monitoring of water distribution networks by mobile sensors, in: Proc. of IEEE INFOCOM, 2016.

[8] A. Elwhishi, P. Ho, K. Naik, B. Shihada, A novel message scheduling framework for delay tolerant networks routing, IEEE Trans. Parallel Distrib. Syst. 24 (5) (2013) 871–880.

[9] Raghu K. Ganti, Fan Ye, Hui Lei, Mobile crowdsensing: Current state and future challenges, IEEE Commun. Mag. 49 (11) (2011) 32–39.

[10] Yi Gao, Wei Dong, Kai Guo, Xue Liu, Yuan Chen, Xiaojin Liu, Jiajun Bu, Chun Chen, Mosaic: A low-cost mobile sensing system for urban air quality monitoring, in: Proc. of IEEE INFOCOM, 2016.

[11] Guoju Gao, Mingjun Xiao, Jie Wu, Liusheng Huang, Chang Hu, Truthful incentive mechanism for nondeterministic crowdsensing with vehicles, IEEE Trans. Mob. Comput. (2018) http://dx.doi.org/10.1109/TMC.2018.2829506.

[12] Bin Guo, Qi Han, Huihui Chen, Longfei Shangguan, Zimu Zhou, Zhiwen Yu, The emergence of visual crowdsensing: Challenges and opportunities, IEEE Commun. Surv. Tutor. PP (99) (2017) 1–18.

[13] Hua-Jun Hong, Ching-Ling Fan, Yen-Chen Lin, Cheng-Hsin Hsu, Optimizing cloud-based video crowdsensing, IEEE Internet Things J. 3 (3) (2016) 299–313.

[14] Yidan Hu, Guojun Dai, Jin Fan, Yifan Wu, Hua Zhang, Blueaer: A fine-grained urban PM2.5 3D monitoring system using mobile sensing, in: Proc. of IEEE INFOCOM, 2016.

[15] Shaohan Hu, Shen Li, Shuochao Yao, Lu Su, Ramesh Govindan, Reginald Hobbs, Tarek F. Abdelzaher, On exploiting logical dependencies for minimizing additive cost metrics in resource-limited crowdsensing, in: Proc. of IEEE ICDCS, 2015.

[16] Prem Prakash Jayaraman, Joao Bartolo Gomes, Hai-Long Nguyen, Zahraa Said Abdallah, Shonali Krishnaswamy, Arkady Zaslavsky, Scalable energy-efficient distributed data analytics for crowdsensing applications in mobile environments, IEEE Trans. Compt. Soc. Syst. 2 (3) (2015) 109–123.

[17] Zhenyu Ju, Chuanhe Huang, Yanjiao Chen, Lin Ma, A truthful auction mechanism for resource provisioning in mobile crowdsensing, in: Proc. of IEEE IPCCC, 2017.

[18] Hernisa Kacorri, Kaoru Shinkawa, Shin Saito, Introducing game elements in crowdsourced video captioning by non-expert, in: Proc. of ACM W4A, 2014.

[19] Samir Khuller, Anna Moss, Joseph (Seffi) Naor, The budgeted maximum coverage problem, Inform. Process. Lett. 70 (1) (1999) 39–45.

[20] Teng Li, Huan Chang, Meng Wang, Bingbing Ni, Richang Hong, Crowded scene analysis: A survey, IEEE Trans. Circuits Syst. Video Technol. 25 (3) (2015) 367–386.

[21] Qing Liu, Tie Luo, Ruiming Tang, Stéphane Bressan, An efficient and truthful pricing mechanism for team formation in crowdsourcing markets, in: Proc. of IEEE ICC, 2015.

[22] Jinwei Liu, Haiying Shen, Xiang Zhang, A survey of mobile crowdsensing techniques: A critical component for the internet of things, in: Proc. of IEEE ICCCN, 2016.

[23] Francesco Restuccia, Sajal K. Das, Jamie Payton, Incentive mechanisms for participatory sensing: Survey and research challenges, ACM Trans. Sensor Netw. (2016).

[24] Jinghao Shi, Chunming Qiao, Dimitrios Koutsonikolas, Geoffrey Challen, CRAWDAD dataset buffalo/phonelab wifi (v. 2016 03 09), 2016, Downloaded from https://crawdad.org/buffalo/phonelabwifi/20160309.

[25] K. Tuite, N. Snavely, D. y. Hsiao, N. Tabing, Z. Popovic, Photocity: training experts at large-scale image acquisition through a competitive game, in: Proc. of ACM CHI, 2011, pp. 1383–1392.

[26] M. Y. S. Uddin, H. Wang, F. Saremi, G.-J. Qi, T. Abdelzaher, T. Huang, Photonet: a similarity-aware picture delivery service for situation awareness, in: Proc. of IEEE RTSS, 2011, pp. 317–226.

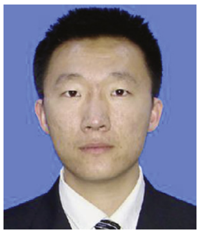
[27] Rui Wang, Fanglin Chen, Zhenyu Chen, Tianxing Li, Studentlife: Assessing mental health, academic performance and behavioral trends of college students using smartphones, in: Proc. of the ACM Conference on Ubiquitous Computing, 2014.

[28] Qianru Wang, Bin Guo, Leye Wang, Tong Xin, He Du, Huihui Chen, Zhiwen Yu, Crowdwatch: Dynamic sidewalk obstacle detection using mobile crowd sensing, IEEE Internet Things J. PP (99) (2017) 1–13.

[29] Shao-Yu Wu, Ruck Thawonmas, Kuan-Ta Chen, Video summarization via crowdsourcing, in: Proc. of ACM CHI, 2011.

[30] Yibo Wu, Yi Wang, Wenjie Hu, Guohong Cao, Smartphoto: A resource-aware crowdsourcing approach for image sensing with smartphones, IEEE Trans. Mob. Comput. 15 (5) (2016) 1249–1263.

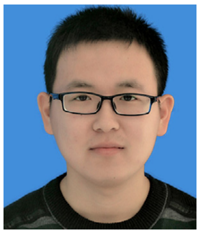
[31] Han Xu, Zheng Yang, Zimu Zhou, Longfei Shangguanand Ke Yi, Yunhao Liu, Enhancing wifi-based localization with visual clues, in: Proc. of ACM UBICOMP, 2015, pp. 963–974.



En Wang received his B.E. degree in software engineering from Jilin University, Changchun, in 2011, his M.E. degree in computer science and technology from Jilin University, Changchun, in 2013, and his Ph.D. in computer science and technology from Jilin University, Changchun, in 2016. He is currently an associate professor in the Department of Computer Science and Technology at Jilin University, Changchun. He is also a visiting scholar in the Department of Computer and Information Sciences at Temple University in Philadelphia. His current research focuses on the efficient utilization of network resources, scheduling and drop strategy in terms of buffer-management, energy-efficient communication between human-carried devices, and mobile crowdsensing.

Yongjian Yang received his B.E. degree in automatization from Jilin University of Technology, Changchun, Jilin, China, in 1983; his M.E. degree in computer communication from Beijing University of Post and Telecommunications, Beijing, China, in 1991; and his Ph.D. in software and theory of computer from Jilin University, Changchun, Jilin, China, in 2005. He is currently a professor and a Ph.D. supervisor at Jilin University, the Vice Dean of the Software College of Jilin University, Director of Key lab under the Ministry of Information Industry, Standing Director of the Communication Academy, and a member of the Computer Science Academy of Jilin Province. His research interests include: network intelligence management, wireless mobile communication and services, and wireless mobile communication.

Jie Wu is the Associate Vice Provost for International Affairs at Temple University. He also serves as Director of the Center for Networked Computing and Laura H. Carnell professor in the Department of Computer and Information Sciences. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Service Computing and the Journal of Parallel and Distributed Computing. Dr. Wu was general cochair/chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, and ACM MobiHoc 2014, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a CCF Distinguished Speaker and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.



Kaihao Lou received his B.E.degree in Software Engineering from Jilin University, Changchun, Jilin,China, in 2017. He is currently a postgraduate student in Computer System Architecture from Jilin University, Changchun, Jilin, China. His current research focuses on Mobile Crowdsensing.

Wenbin Liu received his B.S. degree in Physics from Jilin University, Changchun, China in 2012; and M.E. degree in Department of Software from Jilin University, Changchun in 2016. He is currently a Ph.D. candidate in the Department of Computer Science and Technology, Jilin University, Changchun. His current research focuses on the Mobile CrowdSensing.

Yuanbo Xu received his B.Sc. degree and M.S. degree in the College of Computer Science and Technology, Jilin University, Changchun, China. He is pursuing his Ph.D. degree in Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Jilin University, Changchun, China. He is also a visiting scholar at the Rutgers, the State University of New Jersey. His research interests include applications of data mining, recommender system, and mobile computing.