Efficient Online Collaborative Caching in Cellular Networks with Multiple Base Stations

Pouya Ostovari*, Jie Wu*, and Abdallah Khreishah†
*Department of Computer & Information Sciences, Temple University, Philadelphia, PA 19122
†Department of Electrical & Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102

***Abstract*—These days we are witnessing a tremendous increase in the popularity of wireless devices, e.g. smartphones and tablets. These devices are typically connected to the Internet through cellular connections, such as LTE/4G. Because of the popularity of the wireless devices, a large portion of the traffic on the Internet goes through the cellular base stations. Caching the contents at the base stations brings the contents closer to the users, reduces the traffic on the Internet, and reduces the cost of providing the contents. In this paper, we study the problem of collaborative caching in cellular networks among a set of base stations. Motivated by the emergence of cloudlets, we consider unlimited cache space in our model, and our objective is to minimize the aggregated caching and download cost. We show that in the case of knowing the popularity of the contents, this optimization has a submodular property, and a greedy algorithm can achieve an approximation ratio of 2 for this optimization. We also provide an online algorithm that does not require any knowledge about the future requests and the content popularity. In order to evaluate our online algorithm, we compare its performance against the optimal solution through simulations.**

***Index Terms*—Collaborative caching, cellular network, wireless networks, approximation, online algorithm.**

I. Introduction

Recently, we have witnessed a rapid increase in the popularity of mobile devices, e.g. smartphones and tablet devices. These devices can be easily connected to the Internet using WiFi or cellular connections. The wireless connections and the portability of these devices provide the Internet access from everywhere. These devices are used for a variety of purposes, including but not limited to surfing the Internet, watching videos, video and voice chats, and accessing social networks.

Recent studies indicate that multimedia streaming is the dominant form of the traffic on the Internet. For instance, YouTube and Netflix produce 20-30% of the traffic on the Internet [1], [2]. Also, it is shown in [3] that video on demand services use 54% of the total Internet traffic. This percentage is predicted to grow to about 71% by 2019. As a result of the popularity of wireless mobile devices, a large portion of this traffic goes through the wireless connections of the mobile devices, typically cellular connections such as LTE/4G connections. To address this increase in the demand, the design of the cellular networks needs to be modified, and new traffic engineering methods need to be developed.

One solution to address the increasing data traffic on the Internet is to provide caching capability at the base station of the cellular networks. The dramatic decrease in the storage cost suggests that the future base stations will have the capability to store data [4]. Utilizing the storage of the base stations brings the content closer to the users, and as a result, reduces the data traffic on the backhaul links of the cellular networks and the Internet. In addition to that, caching can reduce the delay of providing the requested data to the wireless users. Providing the locally cached data to the users is much faster and cheaper than downloading from the Internet.

Caching is not a new topic, and the caching problem in general has extensively been studied by the community. However, the works related to cellular networks and base station caching are limited. In [5], [6], the authors propose using wireless helpers to cache the popular contents in order to reduce the content delivery delay. Each helper is assumed to have a limited storage and coverage area. The authors show that the problem is NP-complete, and propose an approximation algorithm for the content placement. The authors in [7] study the problem of content popularity estimation and minimize the content retrieval delay. For this purpose they use caching at the base stations, and they map the minimization problem to a knapsack problem. Hierarchical caching is studied in [8]–[10].

The mentioned works do not address the challenges in collaborative caching. Recently, in [11] collaborative caching has been studied, assuming that the contents' popularity is known. The authors propose an offline heuristic to minimize the content delivery delay. In [4], the authors use collaborative caching and propose an offline algorithm to minimize the operation cost of the base stations in the case of limited cache size. In [12], the authors study collaborative caching for minimizing inter-ISP (Internet service provider) traffic, intra-ISP traffic, and content delivery delay.

It is typical in caching problems to consider limited caches. However, because of the advances in the storage technology, the amount of available storage is growing rapidly. Moreover, the current trend in cloudlets services [13], [14], which are small clouds installed in the base stations or routers of the network, bring a high storage capacity for caching contents. As a result, it can be assumed that the available cache size is becoming unlimited. Although, similar to all cloud services, there is a cost associated with caching contents at cloudlets.

In this paper, we consider collaborative caching at the base stations from an economic perspective. In contrast with the previous works [4]–[7] which consider limited caches, we study the problem of collaborative caching in a cellular network with multiple base stations in the case of unlimited caches. In our model, we consider two types of costs. The first type of cost is the caching cost, which is paid to the cellular network providers or cloudlet service provider. The second type of cost is the cost corresponding to the traffic on the backhaul links of the cellular network. Our objective is to minimize the aggregated caching and download cost of the cellular network. For this purpose, the base stations can collaborate with each other to cache the popular contents and provide them to the users.

We show that our optimization problem is NP-complete, and it has submodular property. Therefore, a greedy algorithm can achieve approximation ratio equal to 2. Moreover, we propose an online algorithm which does not need any knowledge about the popularity of the contents and the future requests. In order to evaluate our online algorithm, we compare its total cost against the optimal solution. In order to find the optimal solution, we assume that the future requests at each base station are known, and we formulate the problem as a mixed integer and linear programming. In order to reduce the complexity of finding the optimal solution, we assume that network coding is applied on the original contents. This assumption relaxes the mixed integer and linear programming to a linear programming optimization. Our contributions in this paper can be summarized as follows:

- We formulate the problem of cache content placement as an integer programming optimization. We show that this optimization problem is NP-complete, and its objective function is submodular. We propose an offline approximation algorithm for the content placement problem, which can achieve approximation ratio equal to 2.
- Assuming that the future users' requests and the popularity of the contents are unknown by the system, we propose an online algorithm to provide a collaborative caching, which does not need a prior knowledge about the popularity of the contents.
- We evaluate our online algorithm by comparing it against the optimal solution. In order to find the optimal solution, we assume that the future request are known, and we formulate the problem as a linear programming optimization. For this purpose, we use the benefit of random linear network coding to encode the contents.

The rest of the paper is organized as follows. In Section II, we review the related work on caching, and we present a background on random linear network coding. We discuss the system setting in Section III. In Section IV, we formulate the problem and propose an offline algorithm. Our online method is presented in Section V. In Section VI, we propose the optimizations to find the optimal solutions. These optimal solutions are used to evaluate the performance of our online algorithm. We present our simulation results in Section VII. Section VIII concludes the paper.

II. Related Work and Background

A. Collaborative Caching

The general caching problem has extensively been studied by the research community [15]–[18], but only few works studied caching in cellar networks and base station. The authors in [5], [6] proposed a method called Femtocaching. The idea of the paper is to use some wireless caches, called helper nodes, to cache the popular contents in order to reduce the delay that the users experience. In Femtocaching, each helper has a limited storage capacity. Each wireless user might be covered with multiple helpers, and the network can be represented as a bipartite graph. The authors prove that the problem of minimizing delay is NP-complete. They show that the optimization problem has a submodular property, and they propose an approximation algorithm for the content placement problem. They later extend their work to the case of different path delays in [6]. Our work is different from these works in a sense that, we consider unlimited caches in our model. Moreover, we assume that caching incurs a cost, which makes it not beneficial to store some contents.

In [7], the problem of content popularity estimation and minimizing the content retrieval delay has been studied. The authors study hierarchical caching in [8]. Also, an information-theoretic view at hierarchical caching has been done in [9], [10]. In [11], the authors assume that the popularity of the contents are known, and they propose an offline collaborative algorithm to minimize the delivery delay.

In [19], the authors consider unlimited cache space at the base stations. They assume that the content popularity and the future requests are not known by the system. In their model, caching and downloading have costs. It is assumed that the download cost from the direct base station is less than the other base stations. Moreover, downloading from any base station results in a smaller cost than downloading from the Internet. In order to minimize the total cost, the authors propose an online algorithm. However, in contrast with our work, the authors assume that the caching cost does not depend on the duration of the time that the content is cached. As a result, once a content has been cached, it will be stored forever.

B. Network Coding

Network coding is a method used to combine and encode the raw data in new form. The authors in [20] propose network coding to solve the bottleneck problem in a multicast problem. It was shown in [21] that linear network coding achieves the capacity of a single multicast problem. The authors in [22] propose a simple method to select the coefficients of the coded packets. They suggest selecting the coefficients randomly, and show that if we select the coefficients of the linearly coded packets randomly, with a very high probability the generated coded packets will be linearly independent.

The idea in random linear network coding [22] is to generate random coefficients for the coded packets, and use these coefficients to mix the original packets. In random linear network coding, all of the operations are performed over a finite field (also know as a Galois field). Each coded packet in random linear network coding has the form of $\sum_j \epsilon_j \times P_j$, where the original packets are represented as P_j. In addition, ϵ_j represent a random coefficient from a finite field.

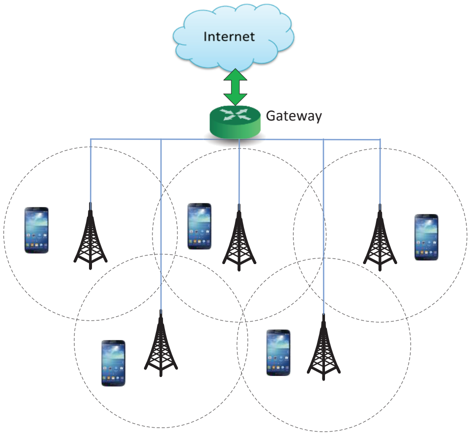


Fig. 1. The system model.

TABLE I
THE SET OF SYMBOLS USED IN THIS PAPER.

Notation	Definition
N/M	The set of BS/contents
n/m	The number of BS/contents
v_k	The size of the k-th content
W_k	The set of BS that cached content k
γ_k	The popularity of content k
g_i	The cost associated with caching at the i-th BS
d_{ij}	Cost of downloading from BS j to BS i
s_i	The capacity of the i-th cache (BS)
u_{jk}	Potential function of the j-th BS for content k
D/C	Total download/caching cost
Y	Content placement set
b	The index of the Internet
α	Rate of the reduction in the potential functions
β	Cost threshold to remove a content from a cache

In addition to provide reliability and throughput enhancement, network coding can convert some non-tractable problems to new problems that can be solved in a polynomial time. For example, many optimization problems such as content caching are hard to track [15], [16]. However, using random linear network coding, the optimal solution of these problems can be found efficiently. The intuition is that, many optimization problems have integer variables, which makes them integer or mixed integer and linear programming optimizations. However, when network coding is applied, the problem becomes similar to a flow optimization problem, and becomes linear programming. In this work, we do not use network coding in our online algorithm because of its decoding overhead for the mobile users. However, in order to find the optimal offline solution for comparison purpose, we use network coding.

III. System Setting

In this work, we consider a set of base stations, which form a cellular network. Each cellular base station covers the users in its coverage area, and is equipped which a local cache storage. In the rest of the paper, we use the terms cache and base station interchangeably. The base stations are connected to each other through backhaul links. Moreover, backhaul links connect the base stations to the Internet through a gateway. When a request comes to a base station, the base station first checks its local cache to find the content. If the content exists in its local cache, that will be send to the user directly. Otherwise, the other base stations can send the content through the backhull links. In the case that the other base stations do not have the content, that will be provided from the Internet. The system model is shown in Figure 1.

We represent the set of base stations as $N = \{1, 2, .., i, .., n\}$. We use notation b to show the index of the Internet. We assume that there are m contents in the Internet, denoted as $M = \{1, 2, .., j, .., m\}$ with sizes $V = \{v_1, v_2, .., v_j, .., v_m\}$. We consider two models in this paper. In the first model, the storage size of each cache is unlimited. However, there is a cost to store the contents on the caches, which makes unpopular contents unsuitable for caching. The unit cost to store a content on base station i is represented as g_i. In our second model, the storage size of the caches are limited, and we also have a caching cost. We denote the size of the cache j as s_j. The popularity of content k is represented as γ_k. We denote the unit cost associated with downloading a content from base station j to base station i as d_{ij}. Also, d_{ib} denotes the unit cost of downloading content from the Internet to the i-th base station. In general, the downloading cost from a base station is less than that from the Internet. Also, downloading cost from the cache of the base station that covers a user is less than downloading from the other base stations, i.e. $d_{ii} < d_{ij},\ \forall j \neq i$.

Our objective is to provide a collaborative caching among multiple base stations, such that the aggregated caching and download cost is minimized. In the following sections, we first propose an offline algorithm assuming that the popularity of the contents are known a priori. We show that in this case, the objective function is minimizing a submodular function, which can be approximated using a greedy algorithm. We later propose an online algorithm which does not require priori knowledge about the popularity of the contents.

IV. Offline Algorithm

In this section, we study offline content placement on the caches, assuming that the popularity of the contents and the number of requests at each base station is known. We first formulate the problem as an optimization problem, and discuss its NP-completeness. We then show that this optimization can be modeled as a submodular function optimization.

A. Problem Formulation

We denote the number of requests at base station i for a period of time t as r_i, and the popularity of content k as γ_k. Let $(j)_i$ denote the base station with the j-th smallest downloading cost to base station i. Moreover, D_i and C_i denote the download and caching cost associated with base station i, respectively. We use variable $y_{jk} = 1$ to indicate the existence of content k on cache j; otherwise, $y_{jk} = 0$. With a similar idea in [6], we can calculate the downloading and caching cost of base station i as follows:

$$D_i = \sum_{k=1}^{m}\sum_{j=1}^{n} d_{i(j)_i} v_k \gamma_k r_i \Big[\prod_{h=1}^{j-1}(1-y_{(h)_i k})\Big] y_{(j)_i k} + \sum_{k=1}^{m} d_{ib} v_k \gamma_k r_i \Big[\prod_{h=1}^{n}(1-y_{(h)_i,k})\Big] y_{bk} \quad (1)$$

$$C_i = \sum_{k=1}^{m} g_i y_{ik} v_k t \quad (2)$$

Here, $\big[\prod_{h=1}^{j-1}(1-y_{(h)_i k})\big] y_{(j)_i k}$ is the indicator function, which means content k is in the cache of base station $(j)_i$, and it does not exist in any base station with a smaller download cost. Moreover, $\big[\prod_{h=1}^{n}(1-y_{(h)_i,k})\big] y_{bk}$ is the indicator function for the case that content k does not exist on any base station.

In the case that the popularity of the contents and the number of received requests by each base station is known, we can formulate our problem as follows:

$$\min \sum_{i=1}^{n} [D_i + C_i] \quad (3)$$

$$s.t \quad y_{ik} \in \{0,1\} \ \forall \ i,k$$

The objective function is minimizing the total cost, which is the summation of the download and caching cost at different base stations.

B. NP-completeness

It can be proved that the optimization problem is NP-complete.

Theorem 1: The optimization in (3) is NP-complete.

Proof. In order to prove the theorem, the same idea as that in [19] can be used. For this purpose, we can reduce the set cover problem to an instance of our optimization problem. The definition of the set cover problem is as follows. We have a set of elements $A = \{a_1, a_2, ..., a_{|A|}\}$, and a set of A' subsets represented as $B = \{b_1, b_2, ..., b_{|B|}\}$. Each subset b_j has a cost e_j. The objective in the set cover problem is to find a set of subsets of B such that it covers all of the elements in A and results in the minimum total cost.

The reduction from the set cover problem to (3) is as follows. We set the number of contents in our problem to one. Also, we map each element in the set cover problem to a base station requesting a content. Each subset b_j of the set B is mapped to base station j. Element $a_i \in b_j$ means that base station i can download contents from base station j. We set the caching cost of the base station j to $g_j = e_j$. Also, for all of the base stations, we set $\gamma_i = 1$. The download cost d_{ij} is set to zero if $a_i \in b_j$; otherwise it is set to e_j. It can be seen that a solution to this instance of our problem is also a solution to the set cover problem [19]. ■

C. Submodular Function Optimization

In this section, we formulate the optimization problem in (3) as the minimization of a submodular function. Modeling the optimization as the minimization of a submodular function enables us to use an approximation algorithm to solve the problem, and find an approximation ratio for the solution. We first define submodular functions.

Submodular functions: Let G be a finite ground set, and f be a set function $f : 2^G \to \mathbb{R}$. We say that f is a submodular function if for all sets $A, B \subseteq G$ we have:

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B) \quad (4)$$

Alternatively, f is a called submodular function if $f(A+l) - f(A) \geq f(B+l) - f(B)$ for $l \in G$ and $A \subseteq B$. This second definition is a useful representation, and it denotes the property of decreasing marginal utility of submodular functions. This property captures the concept of diminishing returns. As a set becomes larger, the benefit of adding an element to the set decreases. In the definition, set B is a superset of A. As a result, the benefit of adding element l to B is less than that of adding the element to A.

Theorem 2: The optimization in (3) is a submodular function.

Proof. Similar to the work in [5], [6], we show that the marginal value of adding a new content to any cache j decreases as the size of the placement set Y becomes larger. By placement set Y, we mean the set of variables y_{jk} with a value equal to 1. We represent the marginal value of adding a new element to placement Y as $Z(Y)$, which is defined as the decrease in the aggregated caching and download cost by adding the new element.

We consider two placement sets Y and Y', where $Y \subset Y'$. Assume that we add element y_{jk} to both of these placements, which does not exist on these two placements. In other words, content k is cached on base station j. For both of the placement sets, the increase in the caching cost C is the same, since the content k is placed on the same cache j. For the downloading cost, two different cases might occur:

1) This placement does not decrease the download cost for a request made on base station i. In this case, the request is served from another base station or the Internet. As a result, the marginal value $Z(Y' \cup y_{jk}) - Z(Y')$ equals zero. For placement Y, if the new placement cannot decrease the cost, we will have $Z(Y \cup y_{jk}) - Z(Y) = 0$. Otherwise, $Z(Y \cup y_{jk}) - Z(Y) > 0$.

2) The new content placement can reduce the downloading cost of the placement Y'. As a result, base station j will serve the received request by base station i. Since $Y \subset Y'$, the downloading cost of content k in placement Y' cannot be greater than Y. Consequently, the decrease in the downloading cost of content k in placement Y cannot be less than that of placement Y'. ■

Corollary 1: A greedy algorithm can achieve an approximation within a factor 2 of the optimum solution for the optimization (3).

Since our optimization problem is minimizing a submodular function, we can use a greedy algorithm to find an approximation for the optimization problem. The greedy algorithm works as follows. At each iteration, we find y_{jk} with the highest marginal value to the placement set. This process is repeated

Algorithm 1 Online Collaborative Caching (Unlimited Cache)

```
**//Initialization**
$W_k = \{\},\ R = \{\}, u_{jk} = 0, \forall\ j, k$
**//On upcoming request**
**for** each content $k$ requested on $i$-th BS **do**
    $R = R \cup \{k\}$
    Call $UP(W_k, k, i)$ to update the potentials
    $h = \arg\max_{j \in N}(u_{jk} - g_j)$ //BS with highest potential
    **if** $u_{hk} - g_h > 0$ **then**
        Cache content $k$ at BS $h$
        $W_k = W_k \cup \{h\},\ b_{hk} = 0$
        Call $CP(W, k, i)$ to calculate the new potentials
    Serve request for content $k$ from BS $h$, where
    $h = \arg\max_{j \in W} e(j, k, i)$ //Optimal BS
    $l = \arg\max_{j \in W/h} e(j, k, i)$ //Second optimal BS
    $b_{hk} = b_{hk} + e(h, k, i) - e(l, k, i)$
```

until there is no y_{jk} with a marginal value greater than zero. After this point, caching a new content on the base station increase the aggregated download and caching cost. Prior results on the approximation of submodular minimizations [23] show that a greedy algorithm achieves an objective function value within a factor 2 of the optimum.

V. Online Algorithm

In the previous section, we discussed the offline collaborative caching, in which the popularity of the contents is known. However, this information might not be available in practice. In this case, we can use an online algorithm, which determines the caching based on incoming requests. For this purpose, the first few requests will be served through the Internet, and they do not result in the content caching. The history of the requests show whether a content is popular or not. If a content is popular, that will be cached on a base station. The online algorithm needs to decide whether to serve a request through the Internet, by caching the content at a base station, or by retrieving the content from a base station that already cached the content.

In the next subsections, we first propose an online algorithm for the case of base stations with unlimited caches. We then modify our algorithm for the case of base stations with limited cache space.

A. Unlimited Cache

In our setting, the base stations do not have any predictions regarding the future request and the popularity of the contents. As a result, they can just rely on the incoming requests and their history to make a decision on caching the contents. The basic idea of our online algorithm is to define a potential function for each base station and for each content. A potential function denotes how much a base station (cache) can be useful in reducing the download cost of a content at all of the base stations. Every time a request for content k is received by a base station, all of the base stations update their potential functions for that particular content. Then, the base station with the smallest potential function for content k minus caching cost will cache the content. If there are no base stations with a potential greater that the caching cost, the content will not be cached, and the content will be served directly from the Internet.

Algorithm 2 $UP(W_k, k, i)$ (Unlimited and Limited Cache)

```
//Updating potentials on the arrival of a new content request
**for** all $j \in N$ **do**
    $u_{jk} = u_{jk} + [e(i, W_j, k) - e(i, j, k)]^+$
```

Algorithm 3 $CP(W_k, k, i)$ (Unlimited Cache)

```
//Recalculating the potentials after the arrival of a new content request
**for** all $j \in N$ **do**
    $u_{jk} = \sum_{j \in R}[e(i, W_j, k) - e(i, j, k)]^+$
```

Caching cost depends on the duration of the caching time, and storing a data for a longer time will increase the cost. As a result, content k that is not requested for a while should be removed from the cache to reduce the caching cost. For this purpose, each base station calculates the amount of help that is provided to the other base stations for content k, which is denoted as b_{jk}. In the case that the total caching cost becomes greater than or equal to a fraction of this benefit, content k will be removed from cache j.

Before discussing the details of the online algorithm, we need to define some notations:

- R: the set of requests that are received so far by the base stations.
- W_k: the set of base stations that cached content k.
- u_{jk}: the potential function of the j-th base station for content k.

Moreover, for a cache j and a content k requested at the i-th base station, we define $e(i, j, k) = d_{ij}$. In the same way, for the set of caches W_k and a content k requested at the i-th base station, $e(i, W_k, k) = \min_{j \in W_k} d_{ij}$. In the case that W_k is empty, d_{ij} becomes infinity.

Our online algorithm is shown in Algorithm 1. In the initialization phase, the potentials of the base stations for all of the contents are set to 0. When a request k comes to base station i, Algorithm 2 is called to update the potentials of the base stations that do not have content k in their cache. In this Algorithm, $[e(W_j, k, i) - e(j, k, i)]^+$ is the amount of cost reduction that base station j can provide for base station i. Then, the online algorithm finds the base station with the highest $u_{hk} - g_h$. If $u_{hk} - g_h > 0$, the base station will cache the content k. Otherwise, the content will not be cached on any base station. Then, Algorithm 3 is called to recalculate the potentials. Finally, the base station which results in the minimum cost will provide the content k to base station i, which can potentially be the base station i.

The idea behind using the potentials is to find how useful is to cache a content at a base station. For this purpose, the first

Algorithm 4 Iterative Potentials Update and Cache Release (Unlimited and Limited Cache)

```
1: //Iterative potentials update
2: for all $j \in N$ do
3:     for all $k \in M$ do
4:         if Base station $j$ did not receive any request for content $j$ at the previous time slot then
5:             $u_{jk} = u_{jk} - u_{jk}/\alpha$
------------------------------------------------------------
6: //Iterative cost calculation and cache release
7: for all $j \in N$ do
8:     for all $k \in M$ cached on BS $j$ do
9:         $c_{jk} = c_{jk} + g_j v_k$
10:        if $c_{jk} > b_{jk}/\beta$ then
11:            Remove content $k$ from BS $j$
12:            $b_{jk} = 0$, $c_{jk} = 0$
```

Algorithm 5 Online Collaborative Caching (Limited Cache)

```
1: //Lines 1-6 of Algorithm 1
2: if $u_{hk} - g_h > 0$ then
3:     if remaining storage in cache $j$ is at least $v_k$ then
4:         Cache content $k$ at BS $h$
5:         $W_k = W_k \cup \{h\}$, $b_{hk} = 0$
6:         Call $CP(W, k, i)$ to calculate the new potentials
7:     else
8:         Find cache $h$ and content $l$ with maximum $u_{hk} - u_{hl}$
9:         if remaining storage in cache $h$ is at least $v_k$ then
10:            Remove content $l$ from cache $h$
11:        Cache content $k$ at BS $h$
12: // Lines 11-13 of Algorithm 1
```

few requests increase the potentials of the base stations, and when the potential of a base station reaches its caching cost, the content will be cached on the base stations. The question is: what should we do if there is a large gap between the requests for content k? In other words, how should we update the potentials for unpopular contents? It is clear that for this type of content, the caching cost might be greater than the earned benefit from caching. As a result, we iteratively reduce the potentials of the base stations for content k in the case that a request does not come to the system for content k. The details are shown in the first part of Algorithm 4.

The popularity of the content changes over a time. The online algorithm might decide to cache content k on a base station, but the popularity of the content might decrease after a while. In this case, we are paying the caching cost without getting enough benefit from the stored content. As a result, the online algorithm needs to remove this content from the cache. For this purpose, as shown in the second part of Algorithm 4, we iteratively update the caching cost of the contents, and we remove a content from a cache, when its caching cost reaches a given portion of its benefit. We update the benefit of caching content k on base station j, whenever base station j helps any base station (including itself) in providing the content. The details of the benefit update are shown in lines 13 and 14 of Algorithm 1. The benefit of a caching at base station h is computed as the cost of providing content k from base station h with the minimum cost to base station i minus the cost of retrieving the content from the base station l with the second minimum cost. The logic behind this calculation is that, the benefit of the base station h is equal to the reduction in the cost when the data is provided from base station h.

B. Limited Cache

In this subsection, we extend our online algorithm to the case in which the cache size of the base stations is limited. We represent the size of cache j as s_j. Similar to the previous subsection, we assume that the base stations do not know the popularity of the contents. Also, they do not have any prediction about the future requests of the users. Consequently, the base stations can just rely on the coming requests and their history to decide whether to cache a content or not. Similar to the case of unlimited cache size, the idea of our online algorithm is to use the potential function for each cache and content. These potential functions show how much a cache can provide benefit by reducing the download cost of a content at all of the base stations. When a request for content k arrives at a base station, the potential functions of the base stations are updated for that request (content). If all of the base stations' potential functions minus their caching costs are less than or equal to zero, the data will not be cached. Otherwise, the base station j with the largest potential function minus caching cost is selected. In contrast with the previous subsection, the selected cache might be full. In this case, we need to decide to replace a cached content on base station j, or select another base station to cache the new request.

Our proposed online algorithm with a limited cache size is shown in Algorithm 5. The first part of the algorithm is similar to lines 1 to 6 of Algorithm 1. Also, the last part is similar to lines 11-13 of Algorithm 1. For brevity, we did not repeat them in Algorithm 5. When a request for content k is received by base station i, Algorithm 2 is called to update the potentials of the base stations that did not cache the content k before. Here, $[e(W_j, k, i) - e(j, k, i)]^+$ is the amount of cost reduction that base station j can provide for base station i.

After that, the online algorithm finds the base station with the highest $u_{hk} - g_h$. If for all base stations $u_{hk} - g_h \leq 0$, the content will not be cached. Otherwise, if the cache of the base station with the greatest $u_{hk} - g_h$ is not full, the content will be cached there. In the case that the cache is full, we can replace the content with the lowest potential, or add the content to another base station. For this purpose, in line 9, we find the cache and content with maximum $u_{hk} - u_{hl}$. If cache h does not have enough storage, content l will be removed. Finally, Algorithm 3 is called to calculate the potentials, and the base station which results in the minimum download cost will provide the content k to base station i.

Similar to the case with unlimited caches, we iteratively reduce the potential functions for content k in the case that no request comes to the system for it. The details are shown

Algorithm 6 $CP(W_k, k, i)$ (Limited Cache)

```
//Recalculating the potentials after the arrival of a new content request
for all $j \in N$ do
    if $j \in W_k$ then
        $u_{jk} = \sum_{j \in R}[e(i, W_j, k) - e(i, j, k)]^+$
    else $u_{jk} = u_{jk} + \sum_{j \in R}[e(i, W_j, k) - e(i, j, k)]^+$
```

in Algorithm 4, which is the same for the case of unlimited and limited caches. Algorithm 4 also handles the changes in the popularity of the contents over time. In the case that the popularity of a cached content decreases, the gained benefit through reduction in the download cost might be less than the caching cost. The received benefit through the reduction in the downloading cost is calculated upon reception of a new request. Algorithm 4 iteratively updates the caching cost of the contents on different caches. Whenever the caching cost of content k on cache j achieves $1/\beta$ of the benefit, the content will be removed from cache k.

VI. Optimal Solution

In order to check the performance of our online algorithm, we need to compare its total cost to that of the optimal cost. Assuming that we know the exact time of all of the requests, we can use mixed integer and linear programming to find the optimal solution.

A. Unlimited Cache

We use variable r_{ik}^{τ} to represent existence of a request for content k at base station i at time τ, such that $r_{ik}^{\tau} = 1$ if there is a request; otherwise, $r_{ik}^{\tau} = 0$. For each content we have two options for caching the k-th content: cached on the i-th station or not cached. In this case, the optimal solution over a time period t can be found by solving the following mixed integer and linear programming:

$$\min D + C \tag{5}$$

$$s.t \quad D \geq \sum_{i,j=1}^{n}\sum_{k=1}^{m}\sum_{\tau=1}^{\tau} d_{ij} x_{ij}^{k\tau} v_k + \sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{\tau=1}^{t} d_{ib} x_{ib}^{k\tau} v_k \tag{6}$$

$$C \geq \sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{\tau=1}^{t} g_i y_{ik}^{\tau} v_k \tag{7}$$

$$x_{ij}^{k\tau} \leq y_{jk}^{\tau}, \ \forall\ i, j, k, \tau \tag{8}$$

$$x_{ib}^{k\tau} + \sum_{j=1}^{n} x_{ij}^{k\tau} \geq r_{ik}^{\tau}, \ \forall\ i, k, \tau \tag{9}$$

$$x_{ib}^{k\tau} \geq y_{ik}^{\tau} - y_{ik}^{\tau-1}, \quad \forall i, k, \tau \tag{10}$$

$$y_{ik}^{\tau}, x_{ij}^{k\tau} \in \{0, 1\}, \ \forall\ i, j, k, \tau \tag{11}$$

The objective function is minimizing the total cost, which is the summation of download cost D and caching cost C. Constraint (6) calculates the download cost of the contents. Variable $x_{ij}^{k\tau}$ being equal to 1 means that base station j provides content k to base station i at time τ. Otherwise, $x_{ij}^{k\tau}$ equals 0. We calculate the caching cost using Constraint (7). If content k is stored on cache j at time τ, variable y_{jk}^{τ} will be equal to 1. The set of Constraints (8) ensures that the k-th content is downloaded from base station j only when the content is stored on the base station. The set of Constraints (9) ensure that all of the requests are served all the time from a base station or the Internet. In order to cache a content, that should be downloaded from the Internet, which is represented as constraint (10).

The complexity of finding the optimal solution of mixed integer and linear programming is high. If we can convert the optimization to a linear programming optimization, we can find the optimal solution in a polynomial time. For this purpose, we use random linear network coding to encode the original contents. In this way, each cache can store a portion of each content, and serve a portion of users' requests. As a result, the variables y and x will be relaxed to real numbers in the range of $[0, 1]$. In this case, the objective is the same as (5). Also, we have the set of Constraints (6)-(10). However, Constraint (11) is replaced with $y_{ik}^{\tau}, x_{ij}^{k\tau} \in [0, 1], \ \forall\ i, j, k, \tau$. This change makes the optimization linear programming, which can be solved in a polynomial time.

B. Limited Cache

We represent the cache space of base station i as s_i. In the case of limited cache space and using random linear network coding, we can formulate the problem as the following linear programming optimization:

$$\min D + C \tag{12}$$

$$s.t \quad D \geq \sum_{i,j=1}^{n}\sum_{k=1}^{m}\sum_{\tau=1}^{\tau} d_{ij} x_{ij}^{k\tau} v_k + \sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{\tau=1}^{t} d_{ib} x_{ib}^{k\tau} v_k \tag{13}$$

$$C \geq \sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{\tau=1}^{t} g_i y_{ik}^{\tau} v_k \tag{14}$$

$$x_{ij}^{k\tau} \leq y_{jk}^{\tau}, \ \forall\ i, j, k, \tau \tag{15}$$

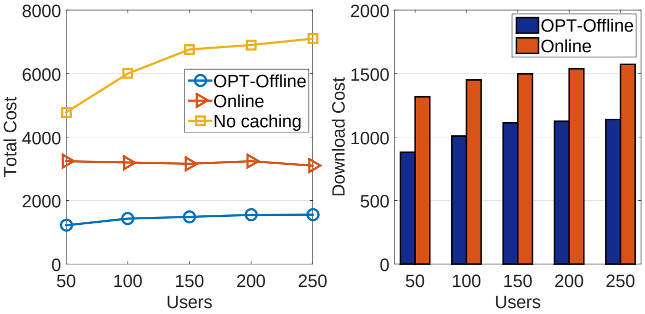
$$x_{ib}^{k\tau} + \sum_{j=1}^{n} x_{ij}^{k\tau} \geq r_{ik}^{\tau}, \ \forall\ i, k, \tau \tag{16}$$

$$\sum_{k=1}^{m} y_{ik}^{\tau} v_k \leq s_i, \ \forall i \tag{17}$$

$$x_{ib}^{k\tau} \geq y_{ik}^{\tau} - y_{ik}^{\tau-1}, \quad \forall i, k, \tau \tag{18}$$

$$y_{ik}^{\tau}, x_{ij}^{k\tau} \in \{0, 1\}, \ \forall\ i, j, k, \tau \tag{19}$$

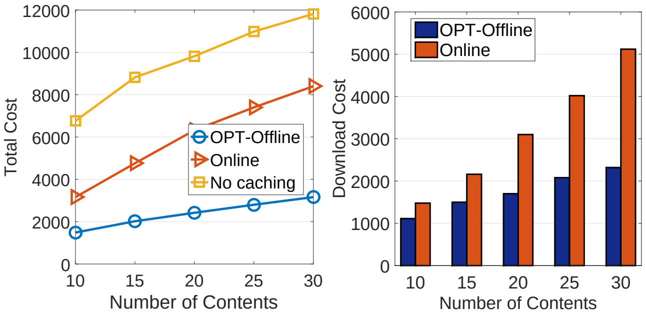
Similar to the case of unlimited cache, the objective function is minimizing the total caching and download cost. Constraint (13) calculates the download cost of the contents. The set of Constraint (14) are constraints on cache capacity. We use the set of Constraints (15) to ensure that the k-th content is downloaded from base station j only when the content is stored on the base station. The set of Constraints (16) ensure that all of the requests are served all the time. The set of Constraints (17) are the storage capacity constraints.



(a)

(b)

Fig. 2. The effect of number of users on the total cost in the case of 5 base stations, $\alpha = 5$; $\beta = 2$; (a) Total cost; (b) Download cost.



(a)

(b)

Fig. 3. The effect of number of contents on the total cost in the case of 5 base stations; $\alpha = 5$; $\beta = 2$; (a) Total cost; (b) Download cost.

VII. Evaluation Results

In this section, we evaluate our methods using numerical results. We compare the performance of our proposed online algorithm against that of the optimal solution.

A. Simulation Setting

In order to evaluate our methods, we compare the total cost of our online method against that of the optimal solution and no-caching. For this purpose, we implemented a simulator in the MATLAB environment. In order to find the optimal solution, we use the Linprog tool in the MATLAB environment to run the optimizations in Section VI. In the plots, we refer to the optimal solution as the OPT-offline. In the case of no-caching, there is no cache storage in the system, and all of the request are served directly by the Internet.

We run our simulations for 100 random runs, and the results shown in the next subsection are based on the average output of the 100 runs. Here, by random we mean that the content requests by the users are randomly selected. Also, the download and caching costs are randomly chosen from a range of values. In the following section, we study the effect of number of users, contents, and cache size on the performance of our online algorithm. In the simulations, the size of the contents are equal to 1 GB.

B. Simulation Result

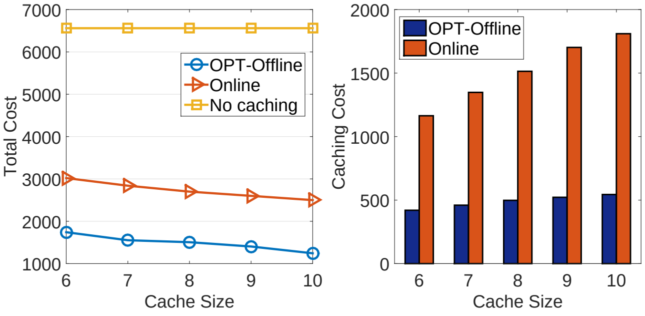
In the first simulations, we compare the total cost of our proposed online algorithm against that of the optimal offline solution and no-caching. For this purpose, we change the number of users in the network in the range of 50 and 250, and measure the total cost. In Figure 2 (a), the number of base stations is 5. Also, the costs are in the range of $g_i \in [1, 3]$, $d_{ij} \in [1, 3]$, $d_{ib} \in [7, 10]$. As the figure shows, the no-caching has the highest total cost. Moreover, the cost of the online algorithm is more than that of the optimal solution, which is due to lake of information about the popularity of the contents and the users requests. More users results in more request, which increases the cost of the no-caching method. However, Figure 2 (a) shows that the slope of the line corespondent to the optimal offline and the online methods is much lower than the no-caching method. The reason is that, caching the contents and reusing them keeps the total cost low. The figure shows that the total cost of the no-caching method can be more than twice that of the online algorithm. Moreover, the total cost of the online algorithm is at most twice of the optimal offline algorithm, which is acceptable, as in the online algorithm we do not know the future requests.

Figure 2 (b), shows the effect of number of users on the download cost of the online and the optimal offline methods. We do not show the download cost of the no-caching method, as that is similar to Figure 2 (a). As expected, more number of users increases the requests and the download cost. However, this increase is limited. As mentioned before, this can be justified by the increase in the efficiency of caching in the case of more number of users.

In the next set of experiments, we measure the effect that the number of content has on the total cost. Similar to the previous experiences, there is a gap between the total cost of our online algorithm and that of the optimal solution in Figure 3 (a). In Figure 3 (a) the popularity of the contents are selected randomly. Because of this randomness, the popularity of each content decreases as we increase the contents amount. Consequently, the caching becomes less efficient in the case of a large number of contents. Figure 3 (b) illustrates the download cost of the online and optimal offline algorithms. More number of users decreases the frequency of requesting the same content, which makes caching less efficient. The figure shows that the increase in the download cost of the optimal solution is slower than that of the online algorithm, which is due to the knowledge about the future requests.

We next measure the effect that the cache size has on the total cost of the system. In Figure 4 (a), we set the number of base stations to 5. The size of the contents are equal to 1 GB, and the cache sizes are in the range of 6 to 10 GB. A greater cache size provides the opportunity to store a greater number of popular contents on the base stations and in more places, which can potentially reduce the total cost. As in the no-caching there is no cache and all of the request are served directly from the Internet, the total cost in the no-caching is fixed. However, the total cost of the offline and online algorithms decrease as the cache size increases.

In the last experience, we measure the effect of cache size



(a)

(b)

Fig. 4. The effect of cache size on the total cost in the case of: (a) 5 base stations; (b) 10 base stations.

on the caching cost in Figure 4 (b). The no-caching method has not been shown in the figure, since it does not use caches. The figure shows that as the cache size increases, the caching cost of the online method increases as well. The reason is that, as more caching resources are available, more popular contents can be cached in order to reduce the total cost. However, the increase in the caching cost of the optimal offline algorithm is much slower than that of the online algorithm. It might be a question why the caching cost in Figure 4 (b) increase as the cache size increase, but in Figure 4 (a) the total cost decreases. The reason is that, as the caching size increases, the popular cached contents can be reused which reduces the download cost. In other words, the decrease in the download cost is more than the increase in the caching cost.

VIII. Conclusion

Wireless devices, e.g. smartphones and tablets, can be easily connected to the Internet through cellular connections, such as LTE/4G. Since these devices are widely-used by people, a large portion of the traffic on the Internet transfers via the cellular base stations. Caching the data at the base stations reduces the traffic on the Internet. In addition to that, caching reduces the cost of providing the content to the users. In this work, we studied collaborative caching in cellular networks. In this problem, a set of base stations, collaborate to cache the more popular contents in order to minimize the total content provision cost. We showed that this problem in NP-complete, and it has a submodular property. Using this property, we proposed greedy method, which can achieve an approximation ratio equal to 2. In addition to that, we introduced an online algorithm, which does not need any knowledge about the popularity of the contents and the future request. In order to measure the performance of our online algorithm, we compared it against the optimal solution. In order to find the optimal solution, we assumed that the future requests are know and the contents are coded using random linear network coding. Using network coding make is feasible to find the optimal solution using linear programming optimization.

Acknowledgment

This research was supported in part by NSF grants CNS 1449860, CNS 1461932, CNS 1460971, CNS 1439672, CNS 1301774, and ECCS 1231461.

References

[1] A. Finamore, M. Mellia, M. Munafò, R. Torres, and S. Rao, "Youtube everywhere: impact of device and infrastructure synergies on user experience," in *ACM IMC*, 2011, pp. 345–360.

[2] C. Labovitz, S. Iekel-Johnson, D. McPherson, J. Oberheide, and F. Jahanian, "Internet inter-domain traffic," in *ACM SIGCOMM*, 2010, pp. 75–86.

[3] "Cisco visual networking index: Global mobile data traffic forecast update, 2014Ű2019 white paper," 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[4] A. Khreishah and J. Chakareski, "Collaborative caching for multicell-coordinated systems," in *CNTCV*, 2015.

[5] N. Golrezaei, K. Shanmugam, A. Dimakis, A. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helperss," in *IEEE INFOCOM*, 2012, pp. 1107–1115.

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

[7] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *IEEE ICC*, 2014, pp. 1897–1903.

[8] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, S. Sen, and O. Spatscheck, "To cache or not to cache: The 3g case," *IEEE Internet Computing*, vol. 15, no. 2, pp. 27–34, 2011.

[9] N. Karamchandani, U. Niesen, M. Maddah-Ali, and S. Diggavi, "Hierarchical coded caching," in *IEEE ISIT*, 2014, pp. 2142–2146.

[10] M. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 2, pp. 2856–2867, 2014.

[11] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5g systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[12] X. Wang, X. Li, V. Leung, and P. Nasiopoulos, "A framework of cooperative cell caching for the future mobile networks," in *HICSS*, 2015, pp. 5404–5413.

[13] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[14] D. Fesehaye, Y. Gao, K. Nahrstedt, and G. Wang, "Impact of cloudlets on interactive mobile cloud applications," in *IEEE EDOC*, 2012, pp. 123–132.

[15] S. Pawar, S. Rouayheb, H. Zhang, K. Lee, and K. Ramchandran, "Codes for a distributed caching based video-on-demand system," in *ACSSC*, 2011.

[16] H. Hao, M. Chen, A. Parekh, and K. Ramchandran, "A distributed multichannel demand-adaptive P2P VoD system with optimized caching and neighbor-selection," in *SPIE*, 2011.

[17] P. Ostovari, J. Wu, A. Khreishah, and N. B. Shroff, "Scalable video streaming with helper nodes using random linear network coding," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1574–1587, 2016.

[18] P. Ostovari, A. Khreishah, and J. Wu, "Cache content placement using triangular network coding," in *IEEE WCNC*, 2013, pp. 1375–1380.

[19] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Transactions on Mobile Computing*, vol. 15, no. 8, pp. 1863–1876, 2016.

[20] R. Ahlswede, N. Cai, S. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[21] S. Li, R. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[22] T. Ho, M. Médard, R. Koetter, D. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, 2006.

[23] G. Nemhauser, L. Wolsey, and M. Fisher, "An analysis of approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.