IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 21, 2018, accepted July 13, 2018, date of publication July 18, 2018, date of current version August 15, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2856913

An Efficient Online Cache Replacement Algorithm for 5G Networks

AMMAR GHARAIBEH, (Member, IEEE), ISMAIL HABABEH, (Member, IEEE), AND MUSTAFA ALSHAWAQFEH, (Member, IEEE)
School of Electrical Engineering and Information Technology, German Jordanian University, Amman 11180, Jordan
Corresponding author: Ammar Gharaibeh (ammar.gharaibeh@gju.edu.jo)

ABSTRACT In recent years, 5G cellular networks utilization has rapidly increased and is expected to grow even more in the near future. This will put the current cellular networks operators in a challenge to overcome the network's limits to satisfy the increasing mobile data traffic and the proliferation of user demands in deploying mobile applications. The deployment of cache-enabled small base stations (Femtocells) is a promising solution to reduce the backhaul traffic loading and the file-access latency and therefore decrease the cellular network operational costs. Due to the limited cache capacity when compared with the number of files that can be requested by users, in this paper, we formulate the problem of minimizing the cost paid by the cellular network while satisfying the cache capacity as an integer linear program (ILP). Due to the NP-completeness of the ILP formulation and the difficulty of obtaining the file request sequence apriori in real-life scenarios, we propose an online algorithm that decides which file to remove from cache in order to allocate capacity to the newly-requested file. The algorithm works on a per-request basis and does not require the knowledge of the file request sequence in advance. We prove that for a cache that can store up to k files, the algorithm achieves a competitive ratio of $\mathcal{O}(\log(k))$, which is the best competitive ratio achieved by any online algorithm as shown in the literature. The simulations conducted considering a single cache show that while the proposed algorithm achieves a similar hit ratio compared with widely-used replacement schemes, it can reduce the cost of the cellular network by 25%.

INDEX TERMS 5G, cache replacement, competitive ratio, femtocells, integer linear program, online algorithms.

I. INTRODUCTION

Recently, file retrieval applications such as Video on Demand and media sharing are dominating the Internet traffic. Content Delivery Networks (CDN), which are the primary source of servicing such applications, are responsible for 52% of Internet traffic in 2016, and the share is expected to rise to 70% in 2021 [1]. According to [2], Akamai company presents itself as a leading CDN service provider which handles around 30% of global web traffic. Moreover, with the wide-spread of smartphones and the proliferation of social media applications such as Facebook and WhatsApp, the mobile data traffic is expected to increase sevenfold between 2016 and 2021 [1].

This expected increase in mobile data traffic motivated adapting the operations of cellular networks, as the current cellular networks resources, such as link capacity and radio access bandwidth, cannot handle the tremendous traffic of mobile devices. To this end, cellular operators are currently deploying smaller base stations within the macrocell in order to serve the mobile users. These Small Base Stations (SBSs) or Femtocells[1] are connected to the main base station via backhaul links. However, Deploying SBSs does not entirely alleviate the challenges faced by the cellular operators. Therefore, complementary techniques are required.

New approaches based on file caching and delivery are proposed to satisfy users' demands without retransmitting the files from remote servers. This could be done by caching the files locally in storage devices located at the SBS. Caching at SBS is a fundamental aspect of 5G [3]–[5], where caching can reduce the mobile traffic on the backhaul links [6], eliminate its effects on the divergence of response time latency [7], [8], help in smoothing the traffic during peak hours [9], and improve the network's energy efficiency [10]. Thus, reliable caching techniques is of great importance.

[1] We will use the terms SBS and Femtocells interchangeably

In general, caching has been extensively studied for various network types. For example, the work in [11] study caching in Data Center networks, while Khreishah *et al.* [12] and Gharaibeh *et al.* [13] study caching for the current Internet infrastructure. In [14]–[16], caching in Content Centric Networks [17] is studied. It is pertinent to mention that the above mentioned networks have different settings than the cellular networks considered in this paper.

Due to the rapid growth of data communication over Cellular Networks(CNs), caching in CNs attracted a lot of attention and has been studied for different objectives under different settings. Caching files generated from mobile traffic is a challenge for the existing Internet infrastructures and finding the optimal caching strategy is considered NP-Hard [18], [19]. Generally, the proposed caching techniques in CNs can be broadly divided into two frameworks: (i) offline caching, and (ii) online caching. In offline caching, files' popularities are required to obtain the optimal file placement strategy.

Several offline algorithms have been proposed. For example, in order to minimize the average delay of file delivery, Golrezaei *et al.* [20] propose to utilize caching helpers and prior knowledge of files' popularities, while [21] presents a file popularity estimation-based caching algorithm. An information-theoretic framework for hierarchical caching is presented in [22] and [9], while the authors of [23]–[25] study the hierarchical caching in cellular backhaul networks. File placement in SBSs for medical applications is tackled in the optimization framework in [26]. A cooperative cell caching framework that aims at minimizing the overall users delay and the 5G network traffic load is investigated in [27]. A collaborative-based caching approach for a single macrocell with multiple small base station is proposed in [28]. In particular, the in-network caching management is handled through an offline algorithm combined with the Least Frequency Used (LFU) replacement policy. However, this proposed algorithm does not take into consideration the cost of file eviction.

While offline caching can characterize the optimal strategy, it presents several drawbacks. Firstly, solving the optimization problem presented by offline caching is generally NP-hard. Secondly, it is required to know the files' popularities in advance. All the above-mentioned works assume the prior knowledge of files' popularities, which can be hard to get in real-life scenarios.

To address the challenges mentioned above, we propose an online caching algorithm that aims to minimize the cellular network operational costs, given the cache size at each SBS. Online settings mean that files' popularities are not known, and decisions need to be made on a per-request basis (*i.e.,* a decision for a request has to be made before considering future requests). For example, the work in [29] proposes an online algorithm for caching at the main base stations in a multi-cell coordinated systems.

Since the collective size of the files that can be requested by the users is much larger than the capacity of the storage devices used to cache the requested files, cache replacement is inevitable. We assume that fetching a file from its provider incurs some cost. The definition of the cost is up to the file provider, the cellular operator, or both. The cost can be based on the amount of bandwidth or resources used to fetch the file, the monetary costs paid by the cellular operator, or the delay to deliver the file to the users. Therefore, when a request is made for a file and it is already cached, the file is served and the mobile operator does not incur any cost. On the other hand, if the file is not cached, the file has to be fetched from its provider, and thus paying additional costs. Moreover, if the cache is full, some files need to be evicted from the cache in order to allocate cache capacity for the new file. Note that a bad selection of evicted files may result in an increased total cost in the long run, especially when the evicted file is to be requested again in the future.

Most of the proposed algorithms in the literature works by deciding from where to fetch the requested content, then serving it to the user. Depending on the complexity and the speed of the proposed algorithm, this approach introduces additional delays when fetching the content, which may be unacceptable depending on the service requested by the user. In this paper, we look at the problem from a different view. When a content is requested and it is not available in the cache, the content is fetched and served immediately to the user. After that, our proposed algorithm decides which content to evict from the cache in order to cache the newly-requested file. Thus, content delivey is not affected by the complexity or the speed of the algorithm.

Our OCR algorithm, decides which file(s) to remove from the cache when a file that is currently not in the cache is being requested in order to minimize the total cost. We use the concept of competitive ratio in order to measure the performance of the OCR algorithm. We first formulate the problem of minimizing the total cost incurred by fetching files into the cache, while satisfying the cache capacity constraints as an optimization problem, which is presented in Section III. We then show through theoretical analysis that the cost of the solution of the OCR algorithm can be upper bounded by a factor of $\mathcal{O}(\log(k))$ times the cost of the optimal solution, where k is the cache capacity. For any online algorithm, it has been shown in [30] that the optimal competitive ratio is $\Omega(\log(k))$. Therefore, OCR is optimal in the asymptotic sense.

The contributions of this work can be summarized as follow:

- The file caching in a single-cell with multiple small base stations is formulated as an Integer Linear Program (ILP), aiming to minimize the total cost paid by the mobile operators while satisfying the cache capacity constraints.
- An online algorithm (called OCR) for cache replacement, that does not require any prior knowledge about the files' popularities, is proposed. We prove analytically that our OCR algorithm achieves the optimal competitive ration performance among online algorithms.
- Through extensive simulations, we show that OCR algorithm can reduce the cost when compared widely-used

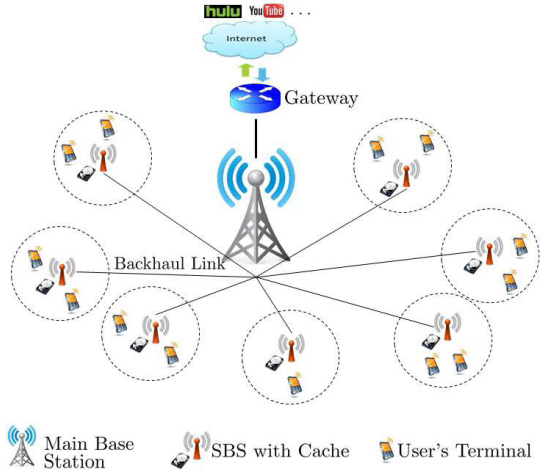

FIGURE 1. System model.

cache replacement schemes, such as Least Recently Used (LRU) and First In First Out (FIFO), by 25%.

The rest of the paper is organized as follows: In Section II we specify our settings. In Section III we present the ILP problem formulation. The online algorithm followed by proof of it competitive ration are described in Sections IV and V, respectively. Section VI presents our simulation results. We finally conclude the paper in Section VII.

II. SYSTEM MODEL

We consider a single cell consisting of a set $\mathcal{N} = \{1, 2, \ldots, n, \ldots, N\}$ of cache-capable SBSs connected to a main base station via backhaul links as shown in Figure 1. Each SBS is equipped with a cache that can store up to k files. Let f^j denote the $j-th$ file, and $\mathcal{F} = \{f^j | j = 1, 2, \ldots, J\}$ represents the set of all files that can be requested by the users in the cell. If the requested file is already cached, the file is served immediately and a cache-hit occurs. Otherwise, a cache-miss occurs and the file is retrieved from the Internet and a copy is cached. If the cache is full, an already-cached file is selected for eviction. When file f^j is evicted, it incurs a cost of c_j.

Suppose that the requested files are organized into a request sequence $R = \{f_t^j\}_{t=1}^M$, where f_t^j denotes the file that is requested at time t, and variable M represents the length of the request sequence. Therefore, a request f_t^j is served from the SBS if file f^j exists in the cache of the SBS at time t. The objective is to minimize the total eviction cost.

III. PROBLEM FORMULATION

In this section, we present the formulation of minimizing the total eviction cost. The goal of the online algorithm presented in the next section is to select which file(s) to evict if the cache is full.

Let $\hat{t}(f^j, q)$ denote the q-th time file f^j is requested, $r(f^j, t)$ represent the number of times file f^j is requested up to and including time t, and $F(t)$ be the set of files requested so far up to and including time t (*i.e.,* $F(t) = \{f^j | r(f^j, t) \geq 1\}$).

To illustrate these notions, consider the file request sequence shown in Figure 2. The request sequence is $R = \{f_1^4, f_2^8, f_3^3, f_4^2, f_5^8, f_6^1, f_7^8, f_8^9, f_9^4, f_{10}^{15}\}$. Note that file 8 (*i.e.,* f^8) is requested three times at $t = 2$, $t = 5$, and $t = 7$). Therefore, $r(f^8, 5) = 2$ since file f^8 is requested twice up to and including time $t = 5$. Similarly, $r(f^8, 10) = 3$. Additionally, $\hat{t}(f^8, 1) = 2$ since the first request of file f^8 happened at time $t = 2$. Similarly, $\hat{t}(f^8, 2) = 5$ and $\hat{t}(f^8, 3) = 7$. The set of requested files up to $t = 10$ is given by $F(10) = \{f^4, f^8, f^3, f^2, f^1, f^9, f^{15}\}$.

Now, let

$$x(f^j, q) = \begin{cases} 1 & \text{if file } f^j \text{ is evicted between the} \\ & q\text{-th request and the } (q+1)\text{-th request.} \\ 0 & \text{otherwise.} \end{cases}$$

Note that if $x(f^j, q) = 1$, then a cache miss will occur when file f^j is requested for the $(q + 1)$-th time. The cache replacement problem is represented by the following Integer Linear Program (ILP):

$$\min \sum_{j=1}^{J} \sum_{q=1}^{r(f^j,T)} x(f^j, q)c_j \tag{1}$$

$$\text{s.t.} \sum_{f^j \in F(t) \setminus \{f_t^j\}} x(f^j, r(f^j, t)) \geq |F(t)| - k, \quad \forall t \tag{2}$$

$$x(f^j, q) \in \{0, 1\}, \quad \forall t, \ \forall f^j \tag{3}$$

where the objective function is to minimize the total eviction costs. At any time t, the capacity constraints of the cache need to be satisfied. This means that from the set of files requested so far (*i.e.,* $F(t)$), at most k files can reside in the cache. Since at time t, the currently requested file f_t^j will be brought to the cache (assuming that a cache miss occurs), at most $k - 1$ files from the set $F(t) \setminus \{f_t^j\}$ can be available at the cache. This also means that at least $|F(t) \setminus \{f_t^j\}| - (k - 1) = |F(t)| - k$ files must be missing from the cache. Constraint (2) indicates that some files need to be evicted from the cache since the last time they were requested (*i.e.,* set $x(f^j, r(f^j, t)) = 1$) in order to satisfy the capacity constraints.

IV. ONLINE ALGORITHM

In order to solve the ILP optimization problem shown in the previous section, two challenges need to be addressed.

f^4	f^8	f^3	f^2	f^8	f^1	f^8	f^9	f^4	f^{15}
$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$	$t=7$	$t=8$	$t=9$	$t=10$

FIGURE 2. Example of a file request sequence.

First, the problem is known to be NP-Complete [31], [32]. Therefore, it is unknown if an algorithm that solves the ILP problem with polynomial-time complexity exists or not. Second, to find the optimal file caching strategy, one requires to know in advance the sequence in which the files are requested. However, obtaining such knowledge is difficult in real-life situations. Therefore, we turn our attention to developing an Online Cache Replacement (OCR) algorithm that works on a per-request basis to decide which file (or files) to remove from the cache.

In the online version of the problem, which file to be requested at time t is only known at time t. Based on the information available at time t (*i.e.,* what files are currently in the cache, the set of files requested so far, ...), the online algorithm needs to decide which file to evict at time t in order to cache the newly requested file. The algorithm's decisions cannot be revoked in the future, and the decision must be made before the next request is revealed. In contrary to offline caching-based techniques, the operation of online algorithms does not require the prior knowledge of files' popularities.

A popular metric for evaluating the performance of online algorithms is the competitive ratio. For example, it has been employed to assess online routing [33] and energy efficiency [34] algorithms. In particular, the competitive ration is defined as the worst case ratio between the performance of the online algorithm and the performance of the optimal offline algorithm. Formally, let $\mathcal{W}_{off}$ stands for the optimal offline performance, and $\mathcal{W}_{on}$ denotes the performance achieved by the online algorithm, then, the competitive ration is defined as:

$$\sup_{\substack{all\ input \\ sequences}} \frac{\mathcal{W}_{on}}{\mathcal{W}_{off}}.$$

It is desirable to have a low competitive ratio, since this will mean that the worst case performance of the online algorithm is not far from the optimal offline algorithm. We show in Section V that the proposed OCR algorithm achieves the best competitive ratio achievable by any online algorithm.

A. THE ONLINE CACHE REPLACEMENT (OCR) ALGORITHM

In this section, we start with a high-level description of OCR algorithm, followed by the algorithm.

The OCR algorithm has two stages: In the first stage, we produce a fractional solution to the ILP formulation presented in Section III (where we relax the second constraint, allowing $x(f^j, q)$ to take any value between 0 and 1) that is within $\mathcal{O}(\ln(k))$ factor of the optimal solution. In the second stage, the OCR algorithm rounds the fractional solution to an integral solution (using randomized rounding techniques) that is within $\mathcal{O}(\log(M))$ factor of the fractional solution. Therefore, the overall competitive ratio of the OCR algorithm is $\mathcal{O}(\ln(k)\log(M))$.

TABLE 1. List of symbols.

Symbol	Definition
k	Cache capacity
M	Length of the request sequence
c_j	Cost of evicting file f^j
j_t	The file requested at time t
$\hat{t}(f^j, q)$	The time of the q-th request of file f^j
$r(f^j, t)$	The number of times file f^j is requested up to and including time t
$F(t)$	The set of requested files up to and including time t
$\lambda(t)$	The dual variable corresponding to constraint (2)
$\mu(f^j, q)$	The dual variable corresponding to constraint (3)

To obtain the fractional solution, we use a primal-dual approach. The primal-dual approach is based on weak duality, where any feasible dual solution is a lower bound on the optimal primal solution [35]. Specifically, let D^* denote the optimal solution to the dual problem, D_{feas} denote the value of a feasible solution to the dual problem, P^* denote the optimal primal solution, and P_{feas} denote the value of a feasible primal solution. It follows that:

$$D_{feas} \leq D^* \leq P^* \leq P_{feas} \tag{4}$$

The OCR algorithm produces a feasible primal solution P_{feas} and a feasible dual solution D_{feas}. We show through the proof of the competitive ratio presented in Section V that $P_{feas} \leq 2(1 + \ln(k))D_{feas}$. Therefore, using Equation (4), it follows that $P_{feas} \leq 2(1 + \ln(k))P^*$.

To obtain the dual program, a dual variable is associated with every primal constraint. Therefore, a variable $\lambda(t)$ the corresponds to Constraint (2) is introduced for each time t and a variable $\mu(f^j, q)$ the corresponds to Constraint (3) is introduced for each file f^j and the q-th time it is requested. Hence, the dual program of primal problem (1) is formulated as:

$$\max \sum_{t} (|F(t)| - k)\lambda(t) - \sum_{j=1}^{J} \sum_{q=1}^{r(f^j,t)} \mu(f^j, q) \tag{5}$$

such that for each file f^j and the q-th request:

$$\left(\sum_{t=\hat{t}(f^j,q)+1}^{t=\hat{t}(f^j,q+1)-1} \lambda(t) \right) - \mu(f^j, q) \leq c_j,$$

for all f^j and q:

$$\mu(f^j, q) \geq 0$$

and for all t:

$$\lambda(t) \geq 0$$

The symbols used throughout the paper and their definitions are listed in Table 1.

After a feasible fractional solution is obtained, randomized rounding is used to obtain an integral solution. We show in Section V that the integral solution is within $\mathcal{O}(\log(M))$

Algorithm 1 Online Cache Replacement (OCR) Algorithm

1: Initialization: $m = 0, \lambda(t) = 0 \forall t, \mu(f^j, q) = 0 \forall f^j, q, c_{total} = 0$
2: At time t when file f^j is requested
3: $m \leftarrow m + 1$
4: $\forall f^j$, keep $\lceil 2\log(m+1) \rceil$ independent random variables $\Gamma(f^j, e), 1 \leq e \leq \lceil 2\log(m+1) \rceil$, uniformly distributed in the interval [0, 1]. Define a threshold $\gamma_{fj} = \min_e \Gamma(f^j, e)$
5: For file f^j, set $x(f^j, r(f^j, t)) \leftarrow 0$.
6: **if** f^j is already in the cache, **then**
7: satisfy the request.
8: **else**
9: **if** f^j is not in the cache AND there is enough capacity to cache f^j **then**
10: cache file f^j.
11: **else**
12: Select a file for eviction as follows:
13: Increase $\lambda(t)$ at a rate 1.
14: **while** $\sum_{f^j \in F(t) \setminus \{f_t^j\}} x(f^j, r(f^j, t)) < |F(t)| - k$ **do**
15: **if** $x(f^j, q) = 1$ **then**
16: Increase $\mu(f^j, q)$ at the same rate as $\lambda(t)$.
17: **if** $x(f^j, q) = 0$ and $\left(\sum_{t=\hat{t}(f^j,q+1)-1}^{t=\hat{t}(f^j,q)+1} \lambda(t) \right) - \mu(f^j, q) = c_j$ **then**
18: $x(f^j, q) \leftarrow \frac{1}{k}$
19: **if** $\frac{1}{k} \leq x(f^j, q) < 1$ **then**
20: $x(f^j, q) \leftarrow \frac{1}{k} \exp\left(\frac{1}{c_j} \left[\left(\sum_{t=\hat{t}(f^j,q+1)-1}^{t=\hat{t}(f^j,q)+1} \lambda(t) \right) - \mu(f^j, q) - c_j \right] \right)$
21: **if** $x(f^j, q) \geq \gamma_{fj}$ **then**
22: $x(f^j, q) = 1$
23: $\forall f^j$ where $x(f^j, q)$ changed from 0 to 1, evict file f^j, $c_{total} = c_{total} + c_j$

factor of the fractional solution. The algorithm is shown in Algorithm 1.

The Algorithm works as follows: when a new request for file f^j is made at time t, the variable $x(f^j, r(f^j, t))$ is set to 0. This means that if file f^j was evicted previously, it needs to be brought in the cache. The summation $\sum_{f^j \in F(t) \setminus \{f_t^j\}} x(f^j, r(f^j, t))$ in Constraint (2) will decrease by 1 since the variable for f_t^j is excluded from the summation. If the constraint at time t is still satisfied, then there is enough space for the newly-requested file without the need to evict any file, and the algorithm will simply cache the file. Otherwise, a space must be created for the newly-requested file f_t^j by evicting another file (or files) from the cache. This is done by increasing some of the variables $x(f^j, q)$ whose value is strictly less than 1 until the constraint is satisfied.

To do this, the algorithm starts increasing the new dual variable $\lambda(t)$. The variables $x(f^j, q)$ that appear in the primal constraint and whose value is 1 corresponds to files that are already evicted from the cache. For these variables, the corresponding dual variable $\mu(f^j, q)$ is increased at the same rate as $\lambda(t)$ in order to keep the corresponding dual constraint satisfied and to maintain the feasibility of the dual solution (lines 15 and 16 in Algorithm 1). The variables $x(f^j, q)$ that are equal to 0 are increased to $\frac{1}{k}$ whenever the corresponding dual constraint becomes tight (*i.e.,* $\left(\sum_{t=\hat{t}(f^j,q+1)-1}^{t=\hat{t}(f^j,q)+1} \lambda(t) \right) - \mu(f^j, q) = c_j$). All other $x(f^j, q)$ variables have a value between $\frac{1}{k}$ and 1. These variables are increased according to the exponential function presented in line 20 of Algorithm 1.

After the fractional solution is obtained, the algorithm rounds the solution to an integral solution which is within a $\mathcal{O}(\log(M))$ factor of the fractional solution. This is done by comparing the primal variable $x(f^j, q)$ to a threshold γ_{fj} (as computed in line 4 of Algorithm 1) assigned to that variable. If the value of the variable is greater than the value of the threshold (line 21 of Algorithm 1), the variable $x(f^j, q)$ is set to 1. This randomized rounding process introduces a $\mathcal{O}(\log(M))$ factor to the competitive ratio. Therefore, the competitive ratio of the OCR algorithm is $\mathcal{O}(\ln(k)\log(M))$.

V. PERFORMANCE ANALYSIS

In this section, we prove that the competitive ratio of the OCR algorithm is $\mathcal{O}(\ln(k)\log(M))$. We start by showing that the fractional solution obtained by the OCR algorithm is $\mathcal{O}(\ln(k))$-competitive, followed by the proof that the integral solution is within $\mathcal{O}(\log(M))$ factor of the fractional solution.

To show that the fractional solution is $\mathcal{O}(\ln(k))$-competitive, we upper bound the increment of the total cost due to increasing the variables $x(f^j, q)$ from 0 to $\frac{1}{k}$, and the increment of the total cost due to increasing the variables $x(f^j, q)$ from $\frac{1}{k}$ to (at most) 1, separately. We then show that the sum of the two upper bounds is $\mathcal{O}(\ln(k))$-competitive, which implies that the total cost of the OCR algorithm is within $\mathcal{O}(\ln(k))$ factor of the total cost obtained by the optimal solution.

Theorem 1: The competitive ratio of the fractional solution of the OCR algorithm is $2(1 + \ln(k))$.

Proof: From the algorithm, we see that due to the increment of the variables $x(f^j, q)$ until the constraint at time t (*i.e.,* Constraint 2) is satisfied (line 14 in Algorithm 1), and the fact that the variables $x(f^j, q)$ are never increased beyond 1 (*i.e.,* Constraint 2 is satisfied), since whenever a variable $x(f^j, q)$ is equal to 1, its corresponding dual variable $\mu(f^j, q)$ is increased at the same rate as $\lambda(t)$, and the value of the exponential function for that variable $x(f^j, q)$ will remain fixed. Due to these reasons, the primal solution is feasible.

Now we turn our attention to the dual solution. From line 20 of Algorithm 1 and the fact that $x(f^j, q) \leq 1$, we get that:

$$\frac{1}{k}\exp\left(\frac{1}{c_j}\left[\left(\sum_{t=\hat{t}(f^j,q)+1}^{t=\hat{t}(f^j,q+1)-1}\lambda(t)\right)-\mu(f^j,q)-c_j\right]\right)\leq 1$$

$$\frac{1}{c_j}\left[\left(\sum_{t=\hat{t}(f^j,q)+1}^{t=\hat{t}(f^j,q+1)-1}\lambda(t)\right)-\mu(f^j,q)-c_j\right]\leq \ln(k)$$

$$\left(\sum_{t=\hat{t}(f^j,q)+1}^{t=\hat{t}(f^j,q+1)-1}\lambda(t)\right)-\mu(f^j,q)\leq c_j(1+\ln(k))$$

Note that the last inequality represents the dual constraint multiplied by $(1 + \ln(k))$. Therefore, by scaling the dual solution by $(1 + \ln(k))$, the dual solution becomes feasible.

It remains to show that the primal solution is at most twice the dual solution. Therefore, the overall competitive ratio is $2(1 + \ln(k))$.

To bound the primal solution by the dual solution, let C_1 denote the total cost due to increasing some of the variables $x(f^j, q)$ from 0 to $\frac{1}{k}$ (lines 17 and 18 in Algorithm 1). Define $\tilde{x}(f^j, q) = \min\{x(f^j, q), \frac{1}{k}\}$. We bound the term $\sum_{j=1}^{J}\sum_{q=1}^{r(f^j,t)}\tilde{x}(f^j, q)c_j$.

It follows from Algorithm 1 that:

1) If $x(f^j, q)$ is strictly greater than 0, which implies that $\tilde{x}(f^j, q) > 0$, then:

$$\left(\sum_{t=\hat{t}(f^j,q)+1}^{t=\hat{t}(f^j,q+1)-1}\lambda(t)\right)-\mu(f^j,q)=c_j \tag{6}$$

This comes from lines 17 and 18 in Algorithm 1. Moreover, $x(f^j, q)$ is strictly equal to $\frac{1}{k}$.

2) If $\mu(f^j, q) > 0$, then:

$$x(f^j,q)=1 \tag{7}$$

This follows from lines 15 and 16 in Algorithm 1.

3) At time t, let $F'(t)$ be the set of files $f^j \in F(t)$ that are already evicted from the cache since the last time they were requested (*i.e.,* $x(f^j, r(f^j, t)) = 1$). These variables are not increased any further. Therefore, the number of variables in Constraint (2) at time t is $|F(t)|-1-|F'(t)|$, where the value 1 subtracted corresponds to the newly-requested file f_t^j. By definition, for each f^j, $\tilde{x}(f^j, r(f^j, t)) \leq \frac{1}{k}$. Therefore:

$$\begin{aligned}&\sum_{f^j\in F(t)\setminus(F'(t)\cup\{f_t^j\})}\tilde{x}(f^j,r(f^j,t))\\&\leq\sum_{f^j\in F(t)\setminus(F'(t)\cup\{f_t^j\})}\frac{1}{k}\\&\leq\frac{1}{k}\left[|F(t)|-1-|F'(t)|\right]\\&\leq|F(t)|-k-|F'(t)|\end{aligned} \tag{8}$$

where the last inequality follows since Constraint (2) is still not satisfied and therefore $|F(t)| - |F'(t)| \geq k + 1$.

Equations (6), (7), and (8) imply the following:

$$\begin{aligned}&\sum_{j=1}^{J}\sum_{q=1}^{r(f^j,t)}\tilde{x}(f^j,q)c_j\\&\leq\sum_{j=1}^{J}\sum_{q=1}^{r(f^j,t)}\left(\left(\sum_{t=\hat{t}(f^j,q)+1}^{t=\hat{t}(f^j,q+1)-1}\lambda(t)\right)-\mu(f^j,q)\right)\tilde{x}(f^j,q)\\&=\sum_t\left(\sum_{f^j\in F(t)\setminus\{j_t\}}\tilde{x}(f^j,r(f^j,t))\right)\lambda(t)\\&\quad-\sum_{j=1}^{J}\sum_{q=1}^{r(f^j,t)}\mu(f^j,q)\tilde{x}(f^j,q)\\&\leq\sum_t(|F(t)|-k)\lambda(t)-\sum_{j=1}^{J}\sum_{q=1}^{r(f^j,t)}\mu(f^j,q)\end{aligned}$$

The first inequality follows from Equation (8), and the second inequality follows by changing the order of summation. The last inequality holds by taking the derivative of both sides with respect to t. The derivative of the left hand side is $\sum_{f^j\in F(t)\setminus\{f_t^j\}}\tilde{x}(f^j, r(f^j, t))$, which by Equation (8), is at most $|F(t)| - k - |F'(t)|$. The derivative of the right hand side is equal to $|F(t)| - k - |F'(t)|$, since $\lambda(t)$ is increased at a rate of 1, and $\mu(f^j, q)$ belonging to files $f^j \in F'(t)$ is increased at a rate of 1. Therefore, C_1 is $(1 + \ln(k))$-competitive.

Now let C_2 represent the total cost due to increasing the variables $x(f^j, q)$ from $\frac{1}{k}$ up to at most 1 according to the exponential function (lines 19 and 20 in Algorithm 1). Therefore:

$$C_2=\sum_{f^j\in F(t)\setminus\{f_t^j\},\frac{1}{k}\leq x(f^j,q)\leq 1}x(f^j,r(f^j,t))c_j$$

We first note that the sum of the variables $x(f^j, q)$ whose values are strictly between $\frac{1}{k}$ and 1 and the variables $x(f^j, q)$ whose values are strictly 1 is less than $|F(t)| - k$, since Constraint (2) is still not satisfied, *i.e.,*:

$$\begin{aligned}&\sum_{f^j\in F(t)\setminus\{f_t^j\},\frac{1}{k}\leq x(f^j,q)<1}x(f^j,r(f^j,t))\\&\quad+\sum_{f^j\in F(t)\setminus\{f_t^j\},x(f^j,q)=1}x(f^j,r(f^j,t))\\&<(|F(t)|-k)\end{aligned} \tag{9}$$

We also note that since $\lambda(t)$ is increased continuously and $\mu(f^j, q)$ is increased at the same rate as $\lambda(t)$ (whenever the corresponding variable $x(f^j, q)$ is equal to 1), the rate of change of $\lambda(t)$ and $\mu(f^j, q)$ with respect to t is 1, *i.e.,*:

$$\frac{d\lambda(t)}{dt}=1,\quad\frac{d\mu(f^j,q)}{dt}=1 \tag{10}$$

To bound C_2, we bound the increment of the primal cost (due to increasing the variables $x(f^j, q)$ from $\frac{1}{k}$ up to

at most 1) in any iteration by the increment of the dual cost in the same iteration. By taking the derivative of C_2 with respect to t and comparing it to the derivative of the objective function of the dual program with respect to t, we get that:

$$\begin{aligned}\frac{dC_2}{dt} &= \sum_{f^j \in F(t)\setminus\{f_t^j\}, \frac{1}{k} \le x(f^j,q)<1} c_j \frac{dx(f^j, r(f^j,t))}{d\lambda(t)} \frac{d\lambda(t)}{dt} \\ &= \sum_{f^j \in F(t)\setminus\{f_t^j\}, \frac{1}{k} \le x(f^j,q)<1} x(f^j, r(f^j,t)) \\ &\le (|F(t)|-k) - \sum_{f^j \in F(t)\setminus\{f_t^j\}, x(f^j,q)=1} 1 \\ &= (|F(t)|-k)\frac{d\lambda(t)}{dt} - \sum_{f^j \in F(t)\setminus\{f_t^j\}, x(f^j,q)=1} \frac{d\mu(f^j,q)}{dt}\end{aligned}$$

where the second equality follows from Equation (10) and $\frac{dx(f^j,r(f^j,t))}{d\lambda(t)} = \frac{1}{c_j} x(f^j,q)$ for each $x(f^j,q)$ such that $\frac{1}{k} \le x(f^j,q) < 1$. The last inequality follows from Equation (9). The last term is the derivative of the objective function of the dual program. Therefore, the dual cost increases more than C_2. Therefore, C_2 is upper bounded by the cost of a feasible dual solution multiplied by $(1+\ln(k))$.

It follows that C_1+C_2 is at most twice the cost of a feasible dual solution multiplied by $(1+\ln(k))$, which, from weak duality, is less than the optimal primal solution multiplied by $(1+\ln(k))$. □

The proof that the integral solution of the OCR algorithm is within $\mathcal{O}(\log(M))$ of the fractional solution is shown in the following theorem:

Theorem 2: The integral solution of OCR is within $\mathcal{O}(\log(M))$ *factor of the fractional solution.*

Proof: We start by analyzing the random variables $\Gamma(f^j,e)$ obtained in line 4 in Algorithm 1. For each e, $1 \le e \le 2\log(M+1)$, the probability that $\Gamma(f^j,e) \le x(f^j,q)$ is exactly $x(f^j,q)$. Let Z denote the set of variables $x(f^j,q)$ whose values are set to 1 after the rounding process. Note that the probability that variable $x(f^j,q) \in Z$ is the probability that there exists an e such that $\Gamma(f^j,e) \le x(f^j,q)$. Let $Y(f^j,e)$ denote the indicator of the event that $\Gamma(f^j,e) \le x(f^j,q)$ (i.e., $Y(f^j,e) = 1$ if $\Gamma(f^j,e) \le x(f^j,q)$, and 0 otherwise). Since $\Gamma(f^j,e)$ is chosen uniformly at random in the range [0, 1], the probability and the expectation of the indicator $Y(f^j,e)$ is at most $x(f^j,q)$. Therefore, the expected value of the integral solution is:

$$\begin{aligned}\mathbb{E}\Big[\sum_{x(f^j,q)\in Z} x(f^j,q)c_j\Big] &\le \sum_{j=1}^{J}\sum_{q=1}^{r(f^j,T)}\sum_{e=1}^{2\log(M+1)} x(f^j,q)c_j\mathbb{E}[Y(f^j,e)] \\ &\le \sum_{j=1}^{J}\sum_{q=1}^{r(f^j,T)}\sum_{e=1}^{2\log(M+1)} x(f^j,q)c_j \\ &= 2\log(M+1)\sum_{j=1}^{J}\sum_{q=1}^{r(f^j,T)} x(f^j,q)c_j\end{aligned}$$

Therefore, the expected value of the integral solution is at most $2\log(M+1)$ times the value of the fractional solution. □

VI. EXPERIMENTAL PERFORMANCE ANALYSIS

In this section, we compare four cache replacement schemes: the OCR algorithm described in Section 3, Least Recently Used (LRU), First In First Out (FIFO), and Forgetting History and Predicting Future (FHPF) [36]. In [36], the replacement decision is based on the value of a parameter denoted by T_{est}, where the file with the largest value of T_{est} is removed from the cache to make space for the newly requested file. The value of T_{est} depends on the last time the file was accessed, in addition to the average interval of accessing the file.

In the simulations, we consider a single SBS that serves 30 users (*i.e.,* users at a coffee shop). The number of files is set to 3000. The popularity of each file is chosen according to a uniform distribution. Using the files' popularities and the number of users, a request sequence is generated and used as an input to all schemes. Then, each scheme remove one file from cache before the next request appears. The parameters for FHPF are directly taken from [36].

We compare the four schemes as the capacity of the cache (in terms of the maximum number of files the cache can store) is increased. The results are averaged over a 100 runs and are shown in Figure 3. It can be inferred from Figure 3(a) that, as the cache size increases, the total cost of the four schemes decreases, since the number of evictions decreases. Moreover, the performance of the OCR algorithm outperforms all other schemes specially when the cache capacity is low. This is because the OCR algorithm tries to select the file with the lowest cost to evict, while all other schemes do not take the cost into consideration when selecting a file for eviction.

In Figure 3(b), we compare the hit ratio of the four schemes. The figure shows that as the cache capacity increases, the hit ratio increases, since there is a higher chance of finding the requested content in the cache. Although the OCR algorithm achieves a similar lower hit ratio compared to FIFO and LRU, OCR algorithm can reduce the total cost by 25%.

In Figure 4, we measure the per demand cost savings percentage of the OCR algorithm, FHPF, and LRU with respect to FIFO. Here, we measure the costs of the OCR algorithm, FIFO, and LRU for 100 different request sequences while the cache capacity is fixed at 800. For each demand, we normalize the cost of the OCR algorithm, FIFO, and LRU with respect to the cost of FHPF, and then subtract it from 1. If we denote the costs of the OCR algorithm and FHPF for the g-th request sequence as $\mathcal{C}_{OCR}(g)$ and $\mathcal{C}_{FHPF}(g)$ respectively, then the per demand cost savings is computed as $Z_{OCR}(g) = (1 - \frac{\mathcal{C}_{OCR}(g)}{\mathcal{C}_{FHPF}(g)}) \times 100\%$. After that, the empirical CDF of the vector $[Z_{OCR}(1), Z_{OCR}(2), \ldots, Z_{OCR}(100)]$ is plotted. We do the same process for FIFO and LRU. From the figure, we observe that in 20% of the demands, the OCR algorithm achieves at least 10% additional cost savings when compared to FIFO and LRU. Moreover, compared to FHPF, our OCR algorithm

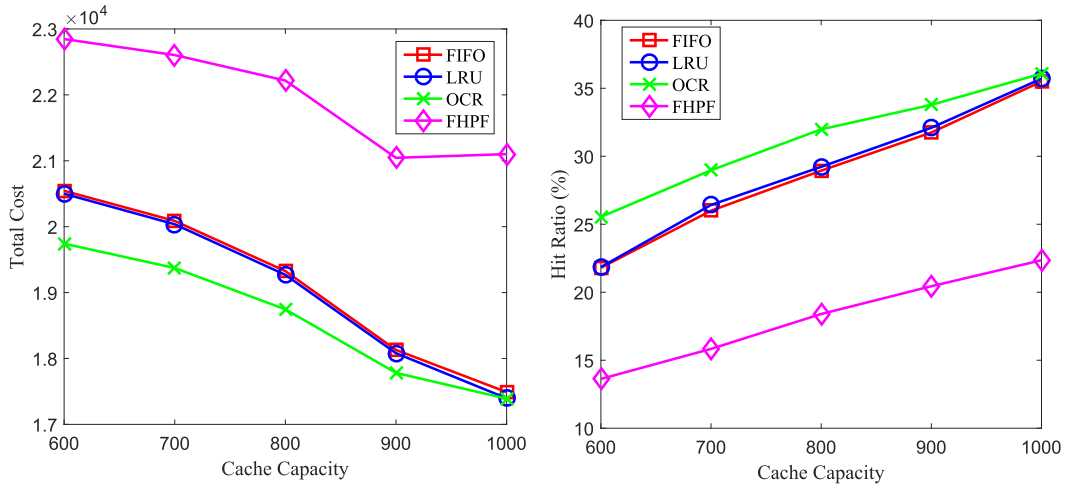

FIGURE 3. Total cost and hit ratio vs. cache capacity.

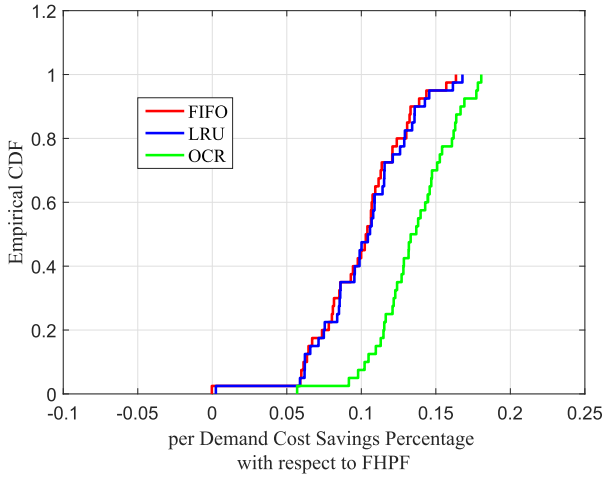

FIGURE 4. The empirical CDF of the per demand cost savings percentage with respect to FHPF.

achieves at least 10% additional cost savings in 95% of the runs.

We emphasize that the simulations are done considering only a single SBS, since the replacement decisions in the cache of any SBS are independent from other SBSs. The results show the cost savings achieved by the OCR algorithm from a single cache only. Therefore, using the OCR algorithm in multiple SBSs will result in huge savings for the cellular operator.

VII. CONCLUSION AND FUTURE WORK

In this paper, we study the problem of cache replacement in a small base station in a cellular network, with the objective of minimizing the total costs paid by the cellular operator. We formulate the problem of cache replacement as an optimization problem that knows the request sequence apriori. Since knowing the file request sequence in advance is hard in real-life scenarios, we provide an online algorithm for the problem. The online algorithm does not require any knowledge about the request sequence. Through simulations, we show that the OCR algorithm outperforms widely-used cache replacement schemes especially when the cache capacity is low. We also conclude that using the OCR algorithm will be beneficial to the cellular operators.

Possible future work includes enhancing our replacement algorithm such that the replacement decision made by a small base station is taken while considering the cache contents of the surrounding base stations (*i.e.,* a collaborative framework). Another direction for further investigation is to study the effect of changing the eviction cost over time, which might reflect how the files' popularities are changing over time.

It is worth to mention that our proposed replacement algorithm is not limited to small base stations in 5G networks. In fact, our replacement algorithm can reduce the operational costs of any network that deploy caches in their intermediate nodes, such as Content Centric Networks (CCN) [37], [38]. In Addition, our algorithm can also be adapted to be applied in any application that requires replacement, not necessarily file replacement. For example, our algorithm can be adapted to decide which network function to shut down in order to make resources available for a new function in Network Function Virtualization (NFV) [39], [40].

REFERENCES

[1] Cisco. (2017). *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf

[2] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai network: A platform for high-performance Internet applications," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 2–19, 2010.

[3] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, no. 8, pp. 82–89, Aug. 2014.

[4] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. C. M. Leung, "Cache in the air: Exploiting content caching and delivery techniques for 5G systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 131–139, Feb. 2014.

[5] J.-N. Shim, B.-Y. Min, K. Kim, J. Jang, and D. K. Kim, "Advanced femto-caching file placement technique for overlapped helper coverage," in *Proc. IEEE 79th Veh. Technol. Conf. (VTC Spring)*, May 2014, pp. 1–5.

[6] X. Peng, J.-C. Shen, J. Zhang, and K. B. Letaief, "Backhaul-aware caching placement for wireless networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.
[7] A. Vakali and G. Pallis, "Content delivery networks: status and trends," *IEEE Internet Comput.*, vol. 7, no. 6, pp. 68–74, Nov. 2003, doi: 10.1109/MIC.2003.1250586.
[8] A.-M. K. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," Grid Comput. Distrib. Syst. Lab., Univ. Melbourne, Melbourne, VIC, Australia, Tech. Rep., 2007, vol. 4.
[9] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2013, pp. 1077–1081.
[10] Z. Luo, M. LiWang, Z. Lin, L. Huang, X. Du, and M. Guizani, "Energy-efficient caching for mobile edge computing in 5g networks," *Appl. Sci.*, vol. 7, no. 6, p. 557, 2017.
[11] A. Khreishah, J. Chakareski, I. Khalil, A. Gharaibeh, and Y. Jararweh, "Joint data placement and flow control for cost-efficient data center networks," in *Proc. ICICS*, 2015, pp. 274–279.
[12] A. Khreishah, I. Khalil, A. Gharaibeh, H. B. Salameh, and R. Alasem, "Joint caching and routing for greening computer networks with renewable energy sources," in *Proc. FiCloud*, 2014, pp. 101–106.
[13] A. Gharaibeh, A. Khreishah, I. Khalil, and J. Wu, "Asymptotically-optimal incentive-based en-route caching scheme," in *Proc. IEEE 11th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2014, pp. 318–326.
[14] M. D. Ong, M. Chen, T. Taleb, X. Wang, and V. Leung, "FGPC: Fine-grained popularity-based caching design for content centric networking," in *Proc. 17th ACM Int. Conf. Modeling Anal. Simulation Wireless Mobile Syst.*, 2014, pp. 295–302.
[15] Y. Wang, Z. Li, G. Tyson, S. Uhlig, and G. Xie, "Optimal cache allocation for content-centric networking," in *Proc. 21st IEEE Int. Conf. Netw. Protocols (ICNP)*, Oct. 2013, pp. 1–10.
[16] S.-W. Lee, D. Kim, Y.-B. Ko, J.-H. Kim, and M.-W. Jang, "Cache capacity-aware CCN: Selective caching and cache-aware routing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2013, pp. 2114–2119.
[17] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. 5th Int. Conf. Emerg. Netw. Exp. Technol.*, 2009, pp. 1–12.
[18] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 8402–8413, Dec. 2013.
[19] T. Wang, L. Song, and Z. Han, "Dynamic femtocaching for mobile users," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2015, pp. 861–865.
[20] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, "FemtoCaching: Wireless video content delivery through distributed caching helpers," in *Proc. INFOCOM*, 2012, pp. 1107–1115.
[21] P. Blasco and D. Gunduz, "Learning-based optimization of cache content in a small cell base station," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 1897–1903.
[22] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. Diggavi, "Hierarchical coded caching," in *Proc. Int. Symp. Inf. Theory (ISIT)*, Jun. 2014, pp. 2142–2146.
[23] D. Pei, S. Sen, and O. Spatscheck, "To cache or not to cache: The 3G case," *IEEE Internet Comput.*, vol. 15, no. 2, pp. 27–34, Mar. 2011.
[24] H. Ahlehagh and S. Dey, "Video caching in radio access network: Impact on delay and capacity," in *Proc. Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2012, pp. 2276–2281.
[25] P. Ostovari, A. Khreishah, and J. Wu, "Cache content placement using triangular network coding," in *Proc. Wireless Commun. Netw. Conf. (WCNC)*, 2013, pp. 1375–1380.
[26] W. P. T. Leung and M. Shikh-Bahaei, "A new femtocaching file placement algorithm for telemedicine," in *Proc. 37th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2015, pp. 1377–1380.
[27] X. Wang, X. Li, V. C. Leung, and P. Nasiopoulos, "A framework of cooperative cell caching for the future mobile networks," in *Proc. 48th Hawaii Int. Conf. Syst. Sci. (HICSS)*, 2015, pp. 5404–5413.
[28] F. Pantisano, M. Bennis, W. Saad, and M. Debbah, "In-network caching and content placement in cooperative small cell networks," in *Proc. 1st Int. Conf. 5G Ubiquitous Connectivity (5GU)*, 2014, pp. 128–133.
[29] A. Gharaibeh, A. Khreishah, B. Ji, and M. Ayyash, "A provably efficient online collaborative caching algorithm for multicell-coordinated systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 10, pp. 1863–1876, Aug. 2016.
[30] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young, "Competitive paging algorithms," *J. Algorithms*, vol. 12, no. 4, pp. 685–699, 1991.
[31] S. Albers, S. Arora, and S. Khanna, "Page replacement for general caching problems," in *Proc. SODA*, vol. 99, 1999, pp. 31–40.
[32] M. Chrobak, G. J. Woeginger, K. Makino, and H. Xu, "Caching is hard—Even in the fault model," *Algorithmica*, vol. 63, no. 4, pp. 781–794, 2012.
[33] P. Jaillet and M. R. Wagner, "Generalized online routing: New competitive ratios, resource augmentation, and asymptotic analyses," *Oper. Res.*, vol. 56, no. 3, pp. 745–757, 2008.
[34] S. Albers and H. Fujiwara, "Energy-efficient algorithms for flow time minimization," *ACM Trans. Algorithms*, vol. 3, no. 4, p. 49, 2007.
[35] E. K. Chong and S. H. Zak, *An Introduction to Optimization*, vol. 76. Hoboken, NJ, USA: Wiley, 2013.
[36] Q. Ling, L. Xu, J. Yan, and Y. Zhang, "An adaptive caching algorithm suitable for time-varying user accesses in VOD systems," *Multimedia Tools Appl.*, vol. 74, no. 24, pp. 11117–11137, 2015.
[37] A. Gharaibeh, A. Khreishah, and I. Khalil, "An o (1)-competitive online caching algorithm for content centric networking," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[38] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," in *Proc. 21st Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2012, pp. 1–7.
[39] K. Joshi and T. Benson, "Network function virtualization," *IEEE Internet Comput.*, vol. 20, no. 6, pp. 7–9, Nov. 2016.
[40] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Commun. Mag.*, vol. 53, no. 2, pp. 90–97, Feb. 2015.

AMMAR GHARAIBEH received the B.S. degree (Hons.) from the Jordan University of Science and Technology in 2006, the M.S. degree in computer engineering from Texas A& M University in 2009, and the Ph.D. degree from the ECE Department, New Jersey Institute of Technology, in 2017. He is currently an Assistant Professor with the Computer Engineering Department, German Jordanian University. His research interests spans the areas of wireless networks and network caching. He has served as a TPC chair and a reviewer for several IEEE journals and conferences.

ISMAIL HABABEH received the bachelor's degree in computer science from the University of Jordan, Amman-Jordan, the master's degree in computer science from Western Michigan University, Kalamazoo, MI, USA, and the Ph.D. degree in computer science from Leeds Beckett University, Leeds, U.K. He is currently with the Faculty of Electrical Engineering and Information Technology, German Jordanian University. His research interests span the areas of cloud computing, big data security, wireless communication networks, and systems performance.

MUSTAFA ALSHAWAQFEH received the B.S. degree in communication engineering from Yarmouk University, Irbid, Jordan, in 2007, the M.S. degree in wireless communication from the Jordan University of Science and Technology, Irbid, in 2010, and the Ph.D. degree in electrical engineering from Texas A&M University, College Station, TX, USA. Since 2017, he has been an Assistant Professor with the Electrical and Communication Engineering Department, German Jordanian University. His research interests span the areas of wireless communications, signal processing, machine learning, and bioinformatics.