Proactive Retention-Aware Caching with Multi-path Routing for Wireless Edge Networks

Samta Shukla[a], Onkar Bhardwaj[b], Alhussein Abouzeid[a], Theodoros Salonidis[d], Ting He[e]

[a] Rensselaer Polytechnic Institute, USA

[b] Akamai Technologies, USA

[c] IBM T. J. Watson Research Center, USA

[d] Penn State University, USA

Abstract

We consider the problem of proactive retention aware caching in a heterogeneous wireless edge network consisting of mobile users accessing content from a server and associated to one or more edge caches. Our goal is to design a caching policy that minimizes the sum of content storage costs and server access costs over two design variables: the retention time of each cached content and the probability that a user routes content requests to each of its associated caches. We develop a model that captures multiple aspects such as cache storage costs and several capabilities of modern wireless technologies, such as server multicast/unicast transmissions, device multi-path routing, and cache access constraints. We formulate the problem of Proactive Retention Routing Optimization (PRRO) as a non-convex, non-linear mixed-integer program. We prove that it is NP-Hard under both multicast/unicast modes – even when the caches have a large capacity and storage costs are linear – and develop greedy algorithms that have provable performance bounds for the case of uncapacitated caches. Finally, we propose heuristics with low computational complexity for the capacitated cache case as well as for the case of convex storage costs. Systematic evaluations based on real-world data demonstrate the effectiveness of our approach, compared to the existing caching schemes.

Index Terms

Proactive/edge caching, heterogeneous/wireless networks, storage cost, approximation algorithms, multicast.

A preliminary version of this work appeared in [1].

I. Introduction

Wireless edge caching advocates adding storage at the wireless edge network infrastructure to address the explosive growth in mobile data demand. The main motivation and efforts have been to reduce response time and network infrastructure energy, bandwidth or monetary costs. In this paper, we explore the less explored dimension of *cache storage cost* in wireless edge networks. Cache storage cost is expected to play an increasingly important role due to the increased heterogeneity in caching services, heterogeneity in wireless infrastructure and usage of high performance flash caches. Existing edge caches focus on delivery of content such as web objects, files, videos or images. In the near future, they will host data needed by edge cloud computing services such as machine learning and classification tasks (object/face/speech recognition) or even containers/VM images that execute such services [2]. Edge cloud computing services adopt the cloud usage-based models, where running services or storing data incurs cost for the duration they are retained in the caches of edge infrastructure.

In terms of network heterogeneity, the all-IP nature of modern 4G cellular networks enables placing caches at various parts of the cellular infrastructure: from the evolved packet core (EPC) to macro/pico/femto base stations. Small cell 5G heterogeneous networks (HetNets) advocate converged cellular and Wi-Fi architectures. This convergence provides more opportunities for cache deployment in heterogeneous wireless edge, but at a higher storage cost, especially as caches are deployed closer to the wireless user. In addition to telcos, the 5G convergence may also create a competition among Internet content providers; whoever pays a higher storage cost (for renting cache space in the wireless infrastructure) gets to extend their own services closer to the wireless user. Finally, another aspect of cache storage cost has been the increasing deployment of flash caches. Retaining data in a flash cache reduces response time but also incurs a cost owing to decreasing cache lifetime per use [3].

In this paper, we address the problem of proactive retention-aware caching and request routing for wireless edge networks. Our objective is to minimize the sum of cache storage cost and server access/transmission costs. In contrast to reactive caching mechanisms such as LRU, proactive caching caches content in advance based on predictions of content popularities, user mobility patterns, etc. Proactive caching has become popular in edge wireless networks due to the availability of large amounts of user data that enable accurate predictions using machine learning models [4], [5].

Our model addresses several new aspects compared to previous work on wireless edge caching. First, it accounts for cache storage cost under a usage-based model using content retention times as optimization

variables. Second, it accounts for the general case when cache coverage areas overlap and user content requests can be routed to one or more caches. Modern mobile devices such as smartphones can utilize multiple wireless interfaces and multi-path networking technologies [6], [7]. Third, it accounts for both unicast and multicast modes of content transmission by the server (upon a cache miss). Multicast is an increasingly popular transmission mode for wireless multimedia content. It has been incorporated in 3GPP specifications for the proposed technology for LTE, the Evolved Multimedia Broadcast and Multicast Services (eMBMS) [8]. Fourth, in addition to cache capacity constraints, it captures practical cache access constraints which limit the number of users that can simultaneously request contents from a cache.

Our contribution: Our key contributions are as follows:

1) We introduce and formulate the *Proactive Retention Routing Optimization (PRRO)* problem as a non-linear, non-convex, mixed-integer program. This problem is highly challenging: in addition to the non-linearity and non-convexity of the objective, it is coupled across contents (due to cache capacities), time slots (due to multipath request routing) and caches (in case of server multicast).
2) We prove that PRRO is NP-Hard even without cache capacity constraints (Theorem 1), and contrast this result against existing results in the literature.
3) We study PRRO when server employs *multicast* mode of content transmission, storage costs are *linear* in retention time, and caches are *uncapacitated*. We propose a greedy algorithm that gives a constant-factor approximation in terms of k (where k is the maximum number of users a cache can serve) on the performance of the optimal (Algorithm 1). With retention variables, our problem turns out to be non-convex, thus we cannot apply standard rounding techniques. Instead, we analyze the performance using two principal techniques: We define a class of solutions, which we call *tight solutions* (Definition 1), and show that every optimal solution for a given instance can be transformed into a tight optimal solution without affecting the value of the objective (Theorem 2). Thus, the greedy algorithm that we develop for multicast server mode focuses only on the class of tight solutions. We also show that for any given instance, there exists another instance with a sparser user-cache association graph (Lemma 3) which has equal gap between the output of our greedy heuristic and the optimal solution as the original instance. These two results enable us to focus on analyzing tight solutions in these sparse instances to prove an upper bound of $k+1$ on the performance of the greedy heuristic for multicast (Theorem 4).
4) We next investigate PRRO when server employs *unicast* mode of content transmission, storage costs

are *linear* in retention time, and caches are *uncapacitated*. Mapping it to the prize-collecting set cover problem [9], we prove that it observes an upper bound of $\mathcal{H}(k) = \Theta(\ln k)$, where k is the maximum number of users a cache can be associated with (Theorem 4).

5) We develop a low complexity heuristic based on PRRO to create a feasible solution for both multicast and unicast cases when the caches are *capacitated* (Algorithm 2) as well as describe how to adapt it when storage cost is a *convex* function of retention duration.
6) We evaluate our algorithms based on a real-world dataset by modeling *overlapping cache coverage* for the capacitated cache case for a wide range of parameters. Through simulations, we observe that our algorithms outperform other contemporary caching policies which are storage unaware.

Paper organization: The rest of the paper is organized as follows: We begin by describing related work in Section II. We define the system model and formulate the problem of Proactive Retention Routing Optimization (PRRO) in Section III. Section III-C formulates the problem for the case of uncapacitated caches, and Section IV investigates the properties of the optimal solutions for this formulation. Sections V-V-B investigate approximation algorithms with provable guarantees for multicast and unicast server mode for uncapacitated caches. We extend this heuristic for the cache capacitated case as well as for convex storage cost in Section VII, and evaluate the performance of our algorithms on a real data-set in Section VIII. All the missing proofs are provided in the Appendix in the full version [10].

II. Related Work

Unicast proactive caching. Proactive caching in small cell networks has been addressed in [11]. Poularakis et al. addressed a similar problem for minimizing server downlink cost [12]. Dehghan et al. addressed a joint request routing and caching problem for minimizing access delay, where users can route requests to multiple caches [13]. A recent work [14] considered proactive caching with unicast. These works do not take cache storage costs (i.e. retention costs) into account and assume unicast server transmissions.

Multicast proactive caching. Several recent works study proactive caching with multicast server transmissions [3], [8], [15]–[17]. Storage is only considered in [3], [8] which aim to optimize the sum of storage and server cost. The work in [8] jointly optimizes multicast schedules with caching decisions. This storage cost model does not capture the content retention time aspect; it is a constant that depends on the binary caching decisions. In addition, it assumes that multicast scheduling is a lower layer control knob only controllable by the telecom operators. In contrast, in our work, cache retention times can be

easily implemented at higher layers by both operators and content providers. Recent work [3] optimizes for cache retention times, however, this work assumes non-overlapping cache areas and does not allow routing user requests to multiple caches.

Timer-based caching. Deciding what to cache and for how long is closely tied to the work on content cacheability. In earlier works, every proactively cached content was stored in cache for a frame of fixed duration (see [15], [18]–[20]). Another line of work considered the problem of finding the optimal timers for every content under various objectives. For example, [21], [22] studies cache hit-rate maximizing timers, [23] studies utility maximizing timers, and [24] obtains optimal timer values to monetize an on-demand caching application. Although numerous, none of the above works optimize the problem with respect to cache storage cost, multicast server transmissions or multi-path request routing.

III. Model and Formulation

In this section, we introduce the system model, state the assumptions and formulate the problem.

A. System Model

We consider a 5G HetNet architecture [25], [26] consisting of a base station instrumented with a content server housing M contents, a set of N caches and a set of I mobile users. Let $[M]$, $[N]$, $[I]$ denote the set of all contents, caches and users, respectively. Caches in our model are higher layer (logical) entities that can control one or more base stations or access points. This decouples the cache from the physical device it is deployed at and allows caches to be deployed at physical locations of the wired network infrastructure in addition to wireless devices (e.g. a cache could refer to the storage unit in a small cell base station (SBS) or in a WiFi-access point). It also allows the scenario where one wireless base station has a cache and is connected to other base stations without caches that still serve users.

1) Content request and service model: We assume that the server holds M contents of equal size. This assumption is justified in real systems which break contents in equal size chunks ([27], [28]). We consider two modes of transmitting data from the server: A downlink multicast transmission of a content from the server can be received by all the users; a downlink unicast transmission is a directed point-to-point transmission received only by the targeted user.

We consider a slotted time system, which divides a continuous period of off-peak and peak hours into T equal-length slots, referred to as a *frame*. Each slot in a frame is of duration d_T time units. We assume delay intolerant service, i.e. in a given time slot of duration d_T, for every requested content that leads to

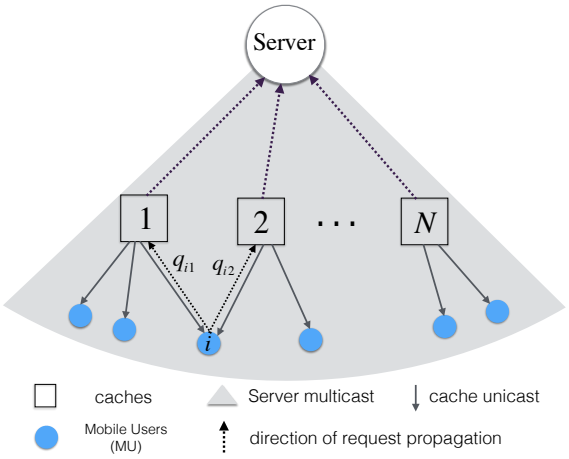


Fig. 1: Proactive caching in a 5G Hetnet.

a cache miss, the server transmits the content during the time slot. A delay-intolerant service assumption is pertinent to serving media contents (such as video) or user-generated content [19], [28]. The value of T could be determined by the periodicity in the request patterns [3], [14], [19] and/or is chosen to match the mobility footprint of users in a specific application (see Section VIII for details).

Users request content independently at every slot during the frame with each user generating at most one request for each content in a slot. Let p_{mi} denote the probability that the demand for content m is generated at user i in a given slot (assumed to be i.i.d. over slots). We assume that the request probabilities are either known or can be learned ([29], [30]).

Given the content demands at time $t = 0$ and a frame of T slots, caching decisions involve determining the retention times for every content at the beginning of the frame. Let y_{mn} denote retention time of content m at cache n, which is an integer that takes value in the set $\{0, 1, \ldots, T\}$. These caching decisions respect the individual cache capacities. We denote the storage capacity at cache n by B_n, implying that cache n can accommodate at most B_n contents at time $t = 0$.

2) User-cache association and routing: A mobile user can *associate* with the caches that are in charge of the network infrastructure of the area they are moving into. A user cache association is possible only when the transmissions from the cache can be received with sufficiently high success probability by the user (given the channel conditions, fading, etc.). Association to multiple caches means that, as a user moves, it discovers, registers to one or more caches and maintains network paths to them. A mobile user finds caches either by direct discovery, by sending discovery packets through multiple wireless interfaces, or by obtaining the IP addresses of caches in its vicinity from the server. From the list of associated caches, a user decides to activate some of the associated paths and *route* a fraction of requests to each one of these

caches. The optimal routing fraction for every content is computed by the caches, and communicated to the user. Once the routing fractions are determined, routing the optimal fraction of requests in any slot is initiated by the user. Let q_{imn} denote the *request routing probability* with which user i routes requests of content m to cache n.

The network can be represented as a three-layered hierarchy as shown in Figure 1 consisting of a server, N caches and I users. Figure 1 shows that user i is connected to and hence can request a given content from either cache 1 or cache 2, with probabilities q_{i1} and q_{i2} respectively such that the probabilities sum to one. One goal of this work is to determine the optimal routing of user requests to caches, given the user-cache association patterns, i.e. the optimal values of these routing probabilities. Our model of allowing for multiple user-cache connections captures the fact that current mobile devices come with multiple wireless network interfaces [7] and can use multi-homing technologies [31].

3) Storage cost: All the requests for a content while it is in the cache lead to a cache hit, in which case the requests are locally served by the cache. Storing a content m in cache n with retention y_{mn} incurs a *storage cost*, $\alpha h(y_{mn})$, where α is the storage cost parameter, $\alpha \geq 0$, and $h(\cdot)$ is a linear function of retention time, y_{mn}, with $h(0) := 0$. In Section VII we also propose algorithms for optimizing for convex storage cost functions. Our choice of an increasing function for storage cost can be justified briefly as follows: When storage cost is interpreted as the price paid for occupying the cache, then it is natural to see that the cost increases with the duration of keeping a content in cache. When storage cost is interpreted as memory damage then, as explained in [3], [27], [32], a higher retention time can potentially lead to a higher memory damage (or usage cost).

4) Download cost: In the event of a cache miss (which can occur either because the content is not stored or because it has expired), the request is forwarded to the server. In the case of a multicast transmission, a *multicast download cost* D_{mul} is incurred by the server for transmitting a content to all the caches/users upon miss. In the case of a unicast transmission, a *unicast download cost* D_{uni} is incurred by the server for a unicast transmission to the targeted user upon miss. *We denote D as the generic download cost (multicast or unicast) which will be clear from the context.* Download costs may reflect average CapEx or OpEx costs of the network operator in case of multicast or unicast. We assume that download cost from the server is much higher than download cost from the caches, thus, in what follows, we ignore the costs of transmissions from the caches. We also assume that the download cost from the server is no smaller than the storage cost parameter, i.e. $D \geq \alpha$, which is reasonable since transmissions from

server is more expensive than storage. Our downlink model for multicast is similar to [8], [28], wherein a single transmission from the server is received by all users. This assumption is motivated by the use of multicast transmissions in the recent work in delivering multimedia contents over cellular networks [8], [33] as well as the incorporation of multicast as a wireless standard in the upcoming 5G network (3GPP specifications) due to its superior efficiency.

We assume that the transmissions are carried over a wireless medium where the underlying PHY/MAC layer technology associated with the user is capable of mitigating the effect of interference at both the transmitter (cache) and at the receiver end. For example, a user can receive multiple concurrent transmissions from multiple caches by having multiple wireless interfaces as in [6] or by invoking multi-homing feature [31]. Interference-free transmission from caches can be achieved by synchronizing their base stations to transmit simultaneously at the same frequency band for multicast services in cellular networks (3GPP and WiMax) [8] or in WiFi networks [7].

B. Problem Formulation

To concretely formulate the problem, we define the following terms. Let $\mathbf{y}$ denote the $M \times N$ retention time matrix for M contents on N caches, where the $(m,n)^{th}$ entry corresponds to $y_{mn} \in \{0, 1, 2, \ldots, T\}$. Then, the storage cost over the frame is given by,

$$\mathrm{C_s}(\mathbf{y}, \mathrm{m}) = \sum_{m \in [M]} \sum_{n \in [N]} \alpha y_{mn}. \tag{1}$$

Let $\mathbf{q}$ denote the routing matrix of size $I \times M \times N$, with the $(i,m,n)^{th}$ entry corresponding to the routing probability $q_{imn}, i \in [I], m \in [M], n \in [N]$. Let U_n and $U(S_c)$ denote the sets of all users that are associated with cache n and a set of caches S_c, respectively. Also, let H_i and $H(S_u)$ denote the set of all caches associated with a user i and a set of users S_u, respectively.

1) Multicast download cost: In the case of a multicast server, a cache miss on content m incurs a server multicast cost D if the content is requested from *at least one of the caches* that do not have the content. Thus with multicast, the probability that the server transmits a content m in a given slot t is:

$$1 - \prod_{i \in [I]} \big((1 - p_{mi}) + \sum_{n \in H_i : y_{mn} \geq t} p_{mi} q_{imn}\big) \tag{2}$$

To explain the above equation, it represents the complement of the event that none of the caches request content m from the server in the time slot t. A cache does not request content either due to having no

incoming user requests or because the requests being served due to having the content stored at the cache. From Eqn 2, the multicast cost for downloading content m over a complete time frame of T slots is given by,

$$D \cdot \sum_{t=1}^{T} \left[1 - \prod_{i \in [I]} \left((1 - p_{mi}) + \sum_{n \in H_i : y_{mn} \geq t} p_{mi} q_{imn} \right) \right]$$

Thus the total download cost for multicast is given by

$$D \cdot \sum_{m \in [M]} \sum_{t=1}^{T} \left[1 - \prod_{i \in [I]} \left((1 - p_{mi}) + \sum_{n \in H_i : y_{mn} \geq t} p_{mi} q_{imn} \right) \right] \tag{3}$$

2) Unicast download cost: Upon a cache miss on content m, a cost of D is incurred if the content is requested from a cache that does not have the content stored. *Note that the cost of unicast transmission as well the cost of multicast transmission from the server are both denoted by D for the sake of brevity but they can be different in practice.* Thus the cost of downloading content m in the unicast mode over the frame is given by:

$$D \cdot \sum_{t=1}^{T} \sum_{i \in [I]} \sum_{n : y_{mn} < t} p_{mi} q_{imn} \tag{4}$$

In the above equation, the second summation is over all the users that request the content from cache n, the third summation is over all the caches where the content retention time has expired, denoted by the event $y_{mn} < t$, and $p_{mi} q_{imn}$ is the probability that user i requests content m from cache n in any slot. Using Eqn (4), the total unicast download cost is given by

$$D \cdot \sum_{m \in [M]} \sum_{t=1}^{T} \sum_{i \in [I]} \sum_{n : y_{mn} < t} p_{mi} q_{imn} \tag{5}$$

3) Access constraints: We assume that at most k_i users can associate with a given cache $i \in [N]$. This is especially relevant in the case of small cell base stations in HetNet settings or WiFi-access points where each cache has only a small number of users reachable from it or when it can register at most a fixed number of users due to hardware/bandwidth constraints. Thus we have $U(n) \leq k_n$ for every cache $n \in [N]$.

4) Problem formulation: Our objective is the aggregate cost of storage and download cost. Using $C_D(\mathbf{y}, \mathbf{q})$ to denote download cost from Eqn (3) for multicast and from Eqn (5) for unicast, the aggregate

cost can be represented as:

$$L(\mathbf{y}, \mathbf{q}) = \mathrm{C_s}(\mathbf{y}) + \mathrm{C_D}(\mathbf{y}, \mathbf{q}) \tag{6}$$

We now state our problem as follows: *Given the content demands and the network topology, how long should a content be stored in the caches (i.e. determining retention times y_{mn}) and how should the user requests be routed to the caches (i.e. determining routing probabilities q_{imn}) so that the aggregate cost is minimized subject to the cache capacity and user-cache association graph?*

We denote this problem as Proactive Retention Routing Optimization (PRRO) which mathematically can be represented as:

$$\begin{aligned}
\mathbf{PRRO}: \quad & \min_{\mathbf{y},\mathbf{q}} \; L(\mathbf{y}, \mathbf{q}) \\
\text{s. t.} \quad & \sum_{m \in M} 1_{y_{mn}>0} \leq B_n, \forall n \in [N] && (7) \\
& y_{mn} \in \{0, 1, 2, \cdots, T\}, \; \forall n \in [N], m \in [M] && (8) \\
& \sum_{n \in H(i)} q_{imn} = 1, \; \forall i \in [I], m \in [M], && (9) \\
& q_{imn} \in [0, 1], \; \forall i \in [I], m \in [M], n \in [N]. && (10) \\
& U_n \leq k_n \; \forall n \in [N] && (11)
\end{aligned}$$

In Eqn (7), $1_{y_{mn}>0}$ is an indicator function which equals 1 if $y_{mn} > 0$ and 0 otherwise. The first constraint (7) in the above formulation shows that the number of contents prefetched do not exceed the individual cache capacities. Constraint (8) models that the retention times take integral values in 0 to T, with a 0 meaning that a content is not stored. Constraint (9) indicates that user i can request content m only from the caches it is associated with. Finally, constraint (10) indicates that the routing probabilities are between 0 and 1 for all possible {user, content, cache} pairs. We summarize most of the recurring notation in Table I.

C. Formulation for large caches

In this section, we describe formulation of PRRO adapted for the case when the caches are *large*, i.e. caches do not have a capacity constraint. We propose a greedy approximation algorithm to solve the problem and give theoretical bounds on its performance. We consider the cache capacitated case in

Notation	Meaning
p_{im}	Per-slot probability of request of content m for user i.
q_{imn}	Routing fraction of content m from user i to cache n.
y_{mn}	Duration for which cache n stores content m.
I, M, N	Total number of users, contents, caches respectively.
$[I], [M], [N]$	The set of all users, all contents, all caches respectively.
p_i, q_{in}, y_n	Restrictions of p_{im}, q_{imn}, y_{mn} for MIP-a setting (which considers a single content.)
$\mathbf{y}, \mathbf{q}$	The 2D/3D matrices of y_{mn}, q_{imn} respectively. (and vector/2D matrix of y_n, q_{in} for MIP-a respectively.)
α	The storage cost coefficient
D	Cost of a transmission from the server.
$L(\mathbf{y}, \mathbf{q})$	The value of the objective function for a solution $(\mathbf{y}, \mathbf{q})$.
$(\mathbf{y}^*, \mathbf{q}^*)$	An optimal solution.
$(\mathbf{y}^l, \mathbf{q}^l)$	The solution of LIN-GR heuristic.
$U_n, U(S_c)$	The set of users associated with a cache n or set of caches S_c.
$H_i, H(S_u)$	The set of caches a user i or set S_u of users associates with.
$F(\mathbf{y})$	Set of caches storing content in a tight solution $(\mathbf{y}, \mathbf{q})$.
$J(\mathbf{y})$	$\{i \in [I] : y_n = 0,\ \forall n \in H(i)\}$
k_n	Access constraint of n^{th} cache, i.e. $\lvert U_n \rvert \leq k_n$
k	$\max_n k_n$

TABLE I: Notation

Section VII. All the omitted proofs as well as the proofs for which we provide only outline can be found in the Appendix in the full version [10].

With uncapacitated caches, the problem decouples across contents, thus PRRO reduces to optimizing for each individual content. Hence, we only focus on a single content throughout the section. We redefine the variables for a single content: With a slight abuse of notation, let $\mathbf{y}$ denote the $1 \times N$ retention time vector with the n^{th} element as y_n denoting the retention time of the content at cache n. Similarly, let $\mathbf{q}$ denote the $I \times N$ routing probability matrix for the content with the $(i,n)^{th}$ element set to q_{in}. Thus the content request probability due to user i at cache n is given by $p_i q_{in}$. The cost of a solution $(\mathbf{y}, \mathbf{q})$ with multicast mode in Eqn (6) can thus be expressed as:

$$\text{Multicast:}\quad L(\mathbf{y}, \mathbf{q}) = \sum_{n \in [N]} \alpha y_n + D \cdot \sum_{t=1}^{T} \left[1 - \prod_{i \in [I]} \left[(1 - p_i) + p_i \sum_{n: y_n > t} q_{in} \right] \right] \tag{12}$$

Similarly, the cost of a solution $(\mathbf{y}, \mathbf{q})$ from Eqn (6) with unicast server mode can now be expressed as:

$$\text{Unicast:}\quad L(\mathbf{y}, \mathbf{q}) = \sum_{n \in [N]} \alpha y_n + D \cdot \sum_{t=1}^{T} \sum_{i \in [I]} p_i \sum_{n: y_n > t} q_{in} \tag{13}$$

Thus for uncapacitated caches, without loss of generality, solving PRRO is equivalent to solving indepen-

dent subproblems where in each subproblem we have been given only a single content. We refer to such a subproblem with only single content as MIP-a, which can be represented as:

$$\textbf{MIP-a}: \quad \min_{\mathbf{y},\mathbf{q}} \; L(\mathbf{y},\mathbf{q})$$

$$\text{s. t.} \quad y_n \in \{0,1,2,\cdots,T\}, \;\; \forall n \in [N] \tag{14}$$

$$\sum_{n \in H(i)} q_{in} = 1, \;\; \forall i \in [I] \tag{15}$$

$$q_{in} \in [0,1], \;\; \forall i \in [I], n \in [N] \tag{16}$$

$$U_n \leq k_n, \; \forall n \in [N] \tag{17}$$

IV. Complexity Results and Properties of Optimal Solutions

Our first result in this section is that MIP-a is NP-hard (i.e. PRRO with uncapacitated caches is NP-hard). Note that the problem is NP-Hard even *without* cache capacities, due to the overlapping cache coverage areas. This stands in contrast with two works with variations in terms of cache capacity constraints and overlapping cache coverage: First, in [3] where the authors show that poly-time solutions can be obtained *without* cache capacities for linear storage cost and disjoint cache coverage; second, in [8] where the authors prove hardness *with* cache capacity constraints but *without* overlapping cache coverage.

Theorem 1. *MIP-a is NP-hard.*

We give the full proof of Theorem 1 in Appendix A in the full version [10] where we show that for $D \gg \alpha$, set cover is a special case of MIP-a. Now we characterize properties of an optimal solution, to later employ these insights for constructing a greedy approximation algorithm. In particular, we will show that there always exists an optimal solution to PRRO such that every cache either stores the content for all T slots or does not store it all, and each user routes its requests only to one cache. Let us define a class of solutions called *tight solutions* (which are not necessarily optimal) as follows:

Definition 1. *A **tight solution** is a solution $(\mathbf{y},\mathbf{q})$ such that:*

- *(A.1)* $q_{in} \in \{0,1\}\ \forall i \in [I], n \in [N]$. *This also implies that for a user i, $q_{in} = 1$ for some $n \in [N]$ since $\sum_n q_{in} = 1$. In other words, a user requests content from exactly one cache.*
- *(A.2)* $y_n \in \{0,T\},\ \forall n \in [N]$. *In other words, a given cache either stores the content for all T slots or does not store it at all.*
- *(A.3) For a user i, if there exists a cache $n \in H(i)$ which stores the content for T slots then $q_{in} = 1$.*

The following theorem states that there always exists an optimal solution which satisfies the properties in the definition 1. This simplifies the structure of an optimal solution which will be useful later in the analysis.

Theorem 2. *For every instance of PRRO, there exists a tight optimal solution.*

Proof outline. We give only an outline here and complete proof can be found in the Appendix A in the full version [10]. Let us denote the optimal solution by $(\mathbf{y}^*, \mathbf{q}^*)$. To prove property (A.1), we show that for any user i if there exist two caches $j, k \in H_i$ such that $y_j^* < y_k^*$ then we have $q_{ij}^* = 0$. This implies that in an optimal solution, all the caches through which a user routes its requests must have the same retention duration. We later show that a user can change its routing fractions to select only one of these caches to route its requests without losing optimality, thus completing the part for property (A.1). Our step of proving for any user i if there exist two caches $j, k \in H_i$ such that $y_j^* < y_k^*$ then we have $q_{ij}^* = 0$ implies that if i routes its requests through a cache that stores the content for duration y_j^* then there cannot exist a cache $k \in H_i$ with $y_k^* > y_j^*$. We use this to divide users into classes corresponding to the retention durations of caches through which they route the requests. Combining this classification with linearity of storage cost, we prove that property (A.2) holds in a series of two steps. In the first step, we will prove that in $(\mathbf{y}^*, \mathbf{q}^*)$, all non-zero retention variables are equal. In the second step, we will prove that all non-zero retention variables must take value T. An optimal solution $(\mathbf{y}^*, \mathbf{q}^*)$ that satisfies property (A.1) and (A.2) must satisfy property (A.3) by virtue of optimality because otherwise it increases the download cost contradicting optimality. ■

V. Approximation Algorithm for Uncapacitated Caches With Multicast Server Mode

As a consequence of Theorem 2, there always exists a tight optimal solution for MIP-a as well since MIP-a is a special case of PRRO. With this insight, we now develop an approximation algorithm for MIP-a with multicast server mode. Since we know that every instance of MIP-a has a tight optimal solution, for the rest of this paper, we will explore the space consisting of only tight solutions for analysis as well performance evaluations unless explicitly mentioned. Thus we will assume that the properties from Definition 1 hold for every solution from here onwards unless explicitly mentioned.

For the space of tight solutions, we will say a user i's demand is *locally satisfied* when a cache associated with it stores the content (and property (A.3) of tight solutions ensures that if such a cache exists in H_i then i routes its requests only through such a cache). Given $(\mathbf{y}, \mathbf{q})$, let $J(\mathbf{y})$ denote the set

of users who do not have any cache connected with them that has stored the content in $(\mathbf{y}, \mathbf{q})$. In other words $J(\mathbf{y}) = \{i \in [I] : y_n = 0,\ \forall n \in H_i\}$. We also define $F(\mathbf{y})$ as the set of caches that store the content in a tight solution $(\mathbf{y}, \mathbf{q})$, i.e. $F(\mathbf{y}) = \{n \in [N] : y_n = T\}$. Thus the cost of a tight solution $(\mathbf{y}, \mathbf{q})$ with multicast server mode can then be expressed as

$$\text{Multicast: } L(\mathbf{y}, \mathbf{q}) = \alpha T|F(\mathbf{y})| + D \cdot T \cdot \left(1 - \prod_{i \in J(\mathbf{y})} (1 - p_i)\right) \tag{18}$$

We introduce some additional notation. Given a set of users U, let us denote by $Z(U)$ the probability that none of the users from U request the content in a slot, i.e.,

$$Z(U) = \prod_{i \in U} (1 - p_i) \tag{19}$$

Let $U(S)$ denote the users connected to a given set of caches S (See Table I). Recall that $J(\mathbf{y})$ denotes the set of users whose demands are not locally satisfied, i.e., the set of users do not have any adjacent cache storing the content. Now, for multicast mode, given a tight solution $(\mathbf{y}, \mathbf{q})$ and a set of caches S, we define $R(S, \mathbf{y})$ as follows:

$$R(S, \mathbf{y}) = Z(J(\mathbf{y}) \cap U(S)) = \prod_{i \in J(\mathbf{y}) \cap U(S)} (1 - p_i)$$

In other words, $R(S, \mathbf{y})$ is the probability that no content is requested in a slot to the server by any of the users in $J(\mathbf{y}) \cap U(S)$. Note that the download cost of a tight solution $(\mathbf{y}, \mathbf{q})$ in multicast mode can be alternatively represented as:

$$C_D(\mathbf{y}, \mathbf{q}) = D \cdot T \cdot (1 - Z(J(\mathbf{y}))) \tag{20}$$

Algorithm 1: LIN-GR

Input: The cache network

1 Start with any tight solution with $\mathbf{y} = \mathbf{0}$. Compute the cost in Equation (18).

2 For every empty cache, compute the reduction in the total cost by forming a tight solution if it stores the content.

3 Find the cache that gives maximum cost reduction and check if the cost reduction is positive. If yes, store content in that cache for T slots. Route the requests of all the users (who can get served by this cache and are not associated with any other cache) from this cache.

4 If no cache satisfies 3 then exit, or else go to step 2.

Output: A tight solution denoted by $(\mathbf{y}^l, \mathbf{q}^l)$.

Now, in Algorithm 1, we describe a greedy Algorithm LIN-GR which iterates through tight solutions.

LIN-GR *sequentially* iterates over empty caches and in every iteration it stores the content in an empty cache which provides the largest one-shot cost reduction. Note that since LIN-GR only searches through tight solutions, its final output is a tight solution. With the help of Lemma 1 below, we can show that LIN-GR can be made to execute in time complexity linear in terms of the edges in the user-cache bipartite graph (denoted by $\mathcal{E}$), and square of the number of caches. The proofs of the following two results as well as the other missing proofs in this section can be found in Appendix B in the full version [10].

Lemma 1. *LIN-GR terminates only when for every set S of empty caches, storing the content in S leads to a download cost reduction of at most $\alpha T|S|$.*

Theorem 3. *LIN-GR runs in $O(\mathcal{E} + N^2)$ complexity.*

A. Performance guarantees of LIN-GR

For any given MIP-a instance E with multicast server mode where $(\mathbf{y}^l, \mathbf{q}^l)$ denotes the solution of LIN-GR and $(\mathbf{y}^*, \mathbf{q}^*)$ denotes the optimal solution, we are interested in bounding the ratio $\gamma(E)$ as defined below:

$$\gamma(E) := \frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \tag{21}$$

Note: Since we are considering only tight solutions (with optimal solution also being tight) without loss of generality, *we will assume that $T = 1$ for this section* because T cancels out from the ratio (See Eqn (18) and (21)). We now state Lemmas 2-4 which will be useful towards bounding $\gamma(E)$. The following Lemma 2 gives us a simple upper bound on $\gamma(E)$ in terms of the number of caches storing the content in the optimal solution and the output of LIN-GR.

Lemma 2. *For an instance of MIP-a, if $(\mathbf{y}^*, \mathbf{q}^*)$ denotes the optimal solution and $(\mathbf{y}^l, \mathbf{q}^l)$ denotes the output of LIN-GR, then we have*

$$\frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \leq 1 + \frac{|F(\mathbf{y}^l)|}{|F(\mathbf{y}^*)|}$$

The following lemma says that $\gamma(E)$ of a MIP-a instance E is equal to another instance E_1 with significantly fewer edges in the user-cache association graph. Intuitively speaking, the only edges that matter for computing $\gamma(E)$ are the edges on which requests are routed in either $(\mathbf{y}^l, \mathbf{q}^l)$ or $(\mathbf{y}^*, \mathbf{q}^*)$. The following lemma comments on the structure of the reduced instance constructed in such a way.

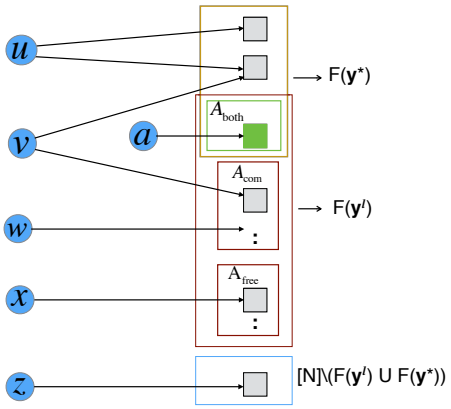


Fig. 2: As a consequence of Lemma 3, a given instance E gets transformed to a reduced instance E_1 which looks like above. Users whose demands do not get locally satisfied in either $(\mathbf{y}^*, \mathbf{q}^*)$ or $(\mathbf{y}^l, \mathbf{q}^l)$ in E look similar to z (Lemma 3(a)). Users whose demands get locally satisfied only in $(\mathbf{y}^l, \mathbf{q}^l)$ look similar to either w or z (Lemma 3(b)). Users whose demands get locally satisfied only in $(\mathbf{y}^*, \mathbf{q}^*)$ look similar to u (Lemma 3(c)). Users whose demands get locally satisfied in both $(\mathbf{y}^*, \mathbf{q}^*)$ and $(\mathbf{y}^l, \mathbf{q}^l)$ look similar to v or a (Lemma 3(d)).

Lemma 3. *Let E be an MIP-a instance. Let $(\mathbf{y}^*, \mathbf{q}^*)$ and $(\mathbf{y}^l, \mathbf{q}^l)$ denote a tight optimal solution and the tight solution of LIN-GR respectively for the instance E. Let U^* and U^l be the set of users whose demands are locally satisfied in $(\mathbf{y}^*, \mathbf{q}^*)$ and $(\mathbf{y}^l, \mathbf{q}^l)$ respectively. Then we can construct a new instance E_1 such that:*

(a) If $i \notin U^* \cup U^l$ then in E_1 user i is associated with at most one cache randomly chosen from H_i from the instance E.

(b) If $i \in U^l \setminus U^*$ then in E_1 user i is associated with exactly one cache from H_i from E, namely, the cache through which $(\mathbf{y}^l, \mathbf{q}^l)$ routes its demand in E.

(c) If $i \in U^* \setminus U^l$ then in E_1 user i is associated with exactly one cache from H_i, namely, any randomly chosen cache from H_i from E such that it stores the content in $(\mathbf{y}^*, \mathbf{q}^*)$.

(d) If $i \in U^* \cap U^l$ then in E_1 user i is associated with at most two caches as follows: If it routes its request through the same cache in$(\mathbf{y}^*, \mathbf{q}^*)$ as well as $(\mathbf{y}^l, \mathbf{q}^l)$ then i is associated with only this single cache. Otherwise it is associated with two caches, namely, the cache through which $(\mathbf{y}^l, \mathbf{q}^l)$ routes its requests and any randomly chosen cache from H_i such that it stores the content in $(\mathbf{y}^*, \mathbf{q}^*)$.

(e) The value $L(\mathbf{y}^*, \mathbf{q}^*)$ of the optimum solution, the tight solution $(\mathbf{y}^l, \mathbf{q}^l)$ of LIN-GR and the value $L(\mathbf{y}^l, \mathbf{q}^l)$ remains unchanged between E and E_1. Consequently $\gamma(E_1) = \gamma(E)$.

We will call an instance E_1 as a **reduced instance** obtained from the instance E with a process described in Lemma 3. For a visual illustration of E_1, please see Figure 2 and its description. For the

reduced instance E_1 from Lemma 3, let $\mathrm{A}_{\mathrm{both}}$ denote the caches that store the content in both $(\mathbf{y}^*, \mathbf{q}^*)$ and $(\mathbf{y}^l, \mathbf{q}^l)$. Let $\mathrm{A}_{\mathrm{free}}$ denote the set of caches such that every user of every cache in this set gets his demand satisfied only in $(\mathbf{y}^l, \mathbf{q}^l)$ and not in $(\mathbf{y}^*, \mathbf{q}^*)$. Let $\mathrm{A}_{\mathrm{com}}$ denote the set $F(\mathbf{y}^l) - \mathrm{A}_{\mathrm{both}} - \mathrm{A}_{\mathrm{free}}$. Note that each cache in $\mathrm{A}_{\mathrm{com}}$ has at least one user i associated with it in E_1 such that the demand of i is satisfied by two different caches in $(\mathbf{y}^l, \mathbf{q}^l)$ and $(\mathbf{y}^*, \mathbf{q}^*)$.

The following lemma comes from a simple observation which follows from combining access constraints with the observation that in E_1 each cache in $\mathrm{A}_{\mathrm{com}}$ has at least one user associated with it which is also associated with one of the caches in $|F(\mathbf{y}^*)|$:

Lemma 4. *In the reduced instance E_1, we have*

$$\frac{|\mathrm{A}_{\mathrm{com}}| + |\mathrm{A}_{\mathrm{both}}|}{|F(\mathbf{y}^*)|} \leq k$$

We now prove the main result of this section:

Theorem 4. *For an instance E of MIP-a with multicast server mode,*

$$\gamma(E) \leq k + 1$$

***Proof**. Without loss of generality, we will consider the reduced instance E_1 obtained using Lemma 3 for proving this theorem.* We will break the analysis in two cases: $C_D(\mathbf{y}^*, \mathbf{q}^*) \leq C_D(\mathbf{y}^l, \mathbf{q}^l)$ and $C_D(\mathbf{y}^*, \mathbf{q}^*) > C_D(\mathbf{y}^l, \mathbf{q}^l)$. First let us consider the case when $C_D(\mathbf{y}^*, \mathbf{q}^*) \leq C_D(\mathbf{y}^l, \mathbf{q}^l)$. Using Eqn (20), this gives us $Z(J(\mathbf{y}^*)) \geq Z(J(\mathbf{y}^l))$. Now we prove that the set $\mathrm{A}_{\mathrm{free}} = \phi$ in this case. Suppose this was not true and $\mathrm{A}_{\mathrm{free}}$ was not empty. Now suppose that LIN-GR algorithm selects its first cache j from $\mathrm{A}_{\mathrm{free}}$ in the $(k+1)^{th}$ iteration. Denote the tight solution just prior to the selection by $(\mathbf{y}^k, \mathbf{q}^k)$. Since LIN-GR reduces download cost at every step, it follows that $C_D(\mathbf{y}^l, \mathbf{q}^l) < C_D(\mathbf{y}^k, \mathbf{q}^k)$. Thus, we have $Z(J(\mathbf{y}^l)) > Z(J(\mathbf{y}^k))$. Combining it with our earlier observation, we get

$$Z(J(\mathbf{y}^*)) \geq Z(J(\mathbf{y}^l)) > Z(J(\mathbf{y}^k)) \tag{22}$$

Since the users associated with the cache $j \in \mathrm{A}_{\mathrm{free}}$ are not associated with any other cache, after adding

content to the cache j, the download cost becomes:

$$D \cdot \left(1 - \frac{Z(J(\mathbf{y}^k))}{Z(U_j)}\right)$$

Since adding content to the cache $j \in \mathrm{A}_{\text{free}}$ reduces the overall cost in $(k+1)^{th}$ iteration for LIN-GR, we thus have

$$D \cdot \big(1 - Z(J(\mathbf{y}^k))\big) - D \cdot \left(1 - \frac{Z(J(\mathbf{y}^k))}{Z(U_j)}\right) > \alpha \quad \Rightarrow \quad D \cdot Z(J(\mathbf{y}^k)) \left(\frac{1}{Z(U_j)} - 1\right) > \alpha \tag{23}$$

From Eqn (22) and (23), we get

$$D \cdot Z(J(\mathbf{y}^*)) \left(\frac{1}{Z(U_j)} - 1\right) > \alpha \tag{24}$$

Note that since the users associated with the cache j are not associated with any other cache, the left hand expression in Eqn (24) is precisely the reduction in the download cost obtained if starting from $(\mathbf{y}^*, \mathbf{q}^*)$, we also additionally store the content in cache j. Thus Eqn (24) says that storing the content in cache j will reduce the cost of $(\mathbf{y}^*, \mathbf{q}^*)$ even further which is a contradiction to its optimality. Hence our original assumption of A_{free} being non-empty must not hold and we have $\mathrm{A}_{\text{free}} = \phi$. Thus we get $|F(\mathbf{y}^l)| = |\mathrm{A}_{\text{both}}| + |\mathrm{A}_{\text{com}}|$. Combining this with Lemma 2 and Lemma 4, we get

$$\frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \leq 1 + \frac{|\mathrm{A}_{\text{both}}| + |\mathrm{A}_{\text{com}}|}{|F(\mathbf{y}^*)|} \leq k + 1 \tag{25}$$

Thus for the case of $C_D(\mathbf{y}^*, \mathbf{q}^*) \leq C_D(\mathbf{y}^l, \mathbf{q}^l)$ we get the desired upper bound.

Now consider the second case when $C_D(\mathbf{y}^*, \mathbf{q}^*) > C_D(\mathbf{y}^l, \mathbf{q}^l)$. Thus we have $Z(J(\mathbf{y}^*)) < Z(J(\mathbf{y}^l))$. Now suppose that LIN-GR algorithm selects its first cache j from A_{free} in the $(k+1)^{th}$ iteration. Denote the tight solution just prior to the selection by $(\mathbf{y}^k, \mathbf{q}^k)$. Since LIN-GR reduces download cost at every step, it follows that $C_D(\mathbf{y}^l, \mathbf{q}^l) < C_D(\mathbf{y}^k, \mathbf{q}^k)$ which gives us $Z(J(\mathbf{y}^k)) < Z(J(\mathbf{y}^l))$. Thus we have two inequalities $Z(J(\mathbf{y}^*)) < Z(J(\mathbf{y}^l))$ and $Z(J(\mathbf{y}^k)) < Z(J(\mathbf{y}^l))$. We break the further analysis as follows:

- $Z(J(\mathbf{y}^*)) \geq Z(J(\mathbf{y}^k))$: Here if $\mathrm{A}_{\text{free}} \neq \phi$ then similar to Eqn (23) and (24) and subsequent reasoning, we can deduce that this leads to a contradiction. Thus $\mathrm{A}_{\text{free}} = \phi$ and rest of the reasoning is similar to the previous case of $C_D(\mathbf{y}^*, \mathbf{q}^*) \leq C_D(\mathbf{y}^l, \mathbf{q}^l)$ which leads to Eqn (25).

- $Z(J(\mathbf{y}^*)) < Z(J(\mathbf{y}^k))$: To begin with, notice that since LIN-GR reduces the overall cost of the solution in every step, we have $L(\mathbf{y}^l, \mathbf{q}^l) < L(\mathbf{y}^k, \mathbf{q}^k)$. Thus we get:

$$\frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \leq \frac{L(\mathbf{y}^k, \mathbf{q}^k)}{L(\mathbf{y}^*, \mathbf{q}^*)} \Rightarrow \frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \leq \frac{|F(\mathbf{y}^k)| + C_D(\mathbf{y}^k, \mathbf{q}^k)}{|F(\mathbf{y}^*)| + C_D(\mathbf{y}^*, \mathbf{q}^*)} \tag{26}$$

  Since we are considering the case when $Z(J(\mathbf{y}^*)) < Z(J(\mathbf{y}^k))$, this together with Eqn (20) implies that $C_D(\mathbf{y}^*, \mathbf{q}^*) > C_D(\mathbf{y}^k, \mathbf{q}^k)$. Substituting it in Eqn (26), we get

$$\frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \leq \frac{|F(\mathbf{y}^k)| + C_D(\mathbf{y}^*, \mathbf{q}^*)}{|F(\mathbf{y}^*)| + C_D(\mathbf{y}^*, \mathbf{q}^*)} \leq \frac{|F(\mathbf{y}^k)|}{|F(\mathbf{y}^*)|}$$

  Recall that $(\mathbf{y}^k, \mathbf{q}^k)$ was the tight solution during the execution of LIN-GR algorithm just prior to adding the first cache from the set $\mathrm{A}_{\mathrm{free}}$. Thus $|F(\mathbf{y}^k)| = |\mathrm{A}_{\mathrm{com}}| + |\mathrm{A}_{\mathrm{both}}|$. Thus the above equation becomes:

$$\frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \leq \frac{|\mathrm{A}_{\mathrm{both}}| + |\mathrm{A}_{\mathrm{com}}|}{|F(\mathbf{y}^*)|}$$

  Combining the above equation with Lemma 4 we get the desired bound.

This completes the proof of Theorem 4. ■

B. Further Improvements Performance Guarantees of LIN-GR

The bound in Theorem 4 can be made much tighter as follows: For computing the download cost in multicast mode, the quantity that matters is the probability that at least one user requests the content in a given slot (see Eqn (18)). Thus if there are several users associated with exactly the same set of caches then these users can be aggregated into a single user whose probability of requesting the content in a slot is equal to the probability that at least one of the aggregated users request the content. Let us divide the users into equivalence classes where in an equivalence class every user is associated with exactly the same set of caches. We can replace each equivalence class by a single aggregated user as follows: If S denotes a set of users in an equivalence class then we can replace this set of users by a single user with probability p_S of requesting the content as $p_S = 1 - \prod_{i \in S}(1 - p_i)$. Let $\mathcal{K}_j$ denote the total number of equivalent classes into which users associated with cache j can be classified and let $\mathcal{K} = \max_{j \in [N]} \mathcal{K}_j$. Thus we can restate Theorem 4 as:

Theorem 5. *For an instance E of MIP-a with multicast server mode,*

$$\gamma(E) \leq \mathcal{K} + 1$$

Note that the upper bound given by the above theorem is likely to be much tighter than from Theorem 4 as in most practical scenarios $\mathcal{K}_j$ will tend to be much smaller than k_j for every cache $j \in [N]$. To improve the bound further, we can apply set cover algorithms to select a set of caches that cover the users in $U(F(\mathbf{y}))$ to see if it can lower the component of storage cost even further.

VI. Performance Guarantees for MIP-a with Unicast Server Mode

Similar to Section V, we will only consider tight solutions as candidate solutions with unicast server mode since we know searching in the space of tight solutions is a reasonable choice to make because this space always consists of an optimal solution (see Theorem 2). Similar to Eqn (18), the cost of a tight solution $(\mathbf{y}, \mathbf{q})$ for unicast server mode is given by:

$$\text{Unicast: } L(\mathbf{y}, \mathbf{q}) = \alpha T |F(\mathbf{y})| + D \cdot T \cdot \sum_{i \in J(\mathbf{y})} p_i \tag{27}$$

Let $\mathcal{E}$ denote the total number of edges in user-cache association graph and let $\mathcal{H}(x)$ denote the harmonic number associated with integer $x > 0$, i.e. $\sum_{i=1}^{x} 1/i$. For unicast mode, we have the following performance guarantee:

Theorem 6. *Given a MIP-a instance E with unicast server transmission mode, a tight solution can be obtained in $O(\mathcal{E})$ time with the following approximation factor:*

$$\gamma(E) \leq \mathcal{H}(k)$$

Proof. We can interpret MIP-a with unicast mode as an instance of the prize-collecting set cover [9] as follows: In an instance of prize-collecting set cover, we are given a set U of elements, with each element e associated with a penalty $\pi(e)$ and a family $\mathcal{S}$ of sets each of which contain some of the elements. Selecting a set $S \in \mathcal{S}$ carries a cost of $c(S)$. The objective is to select a subset $S \subseteq \mathcal{S}$ to minimize $\sum_{j \in S} c(j) + \sum_{e \notin \cup S} \pi(e)$. An instance of MIP-a can be transformed into an instance of prize-collecting set cover by creating an element e_i for each user i, setting $\pi(e_i) = DTp_i$; and for each cache j creating a set S_j consisting of elements corresponding to the users associated with cache j. The cost of selecting a set S_j corresponding to a cache j is set to αT. It can be seen that the cost of a tight solution in this

instance of prize-collecting set cover is exactly equal to the cost of a tight solution in the original MIP-a instance (See Eqn (27)).

An instance of prize-collecting set-cover can also be transformed into MIP-a unicast by applying the exact same procedure in reverse, which gives us one-one correspondence between the instances of prize-collecting set cover and MIP-a with unicast. After this, we immediately get the upper bound of $\mathcal{H}(k)$ by applying $\mathcal{H}(k)$-LMP prize collecting algorithm from Fig 1 from [9] and applying Lemma 14 from the same work. Note that for applying Lemma 14 from [9], we also make use of the fact that there exists a tight optimal solution for MIP-a with unicast (using Theorem 2), which can be mapped to an optimal solution for prize-collecting set cover due to one-one correspondence. ■

Remark: Although MIP-a with multicast is also closely related to the prize-collecting set cover problem, we cannot apply the same technique from the proof of Theorem 6 for multicast mode as the download cost with multicast does not have additive structure like unicast. However, using this similarity we will give bounds for formulations closely related to MIP-a in the following Section V-B.

VII. Heuristics for Capacity Constrained Caches and Convex Storage Costs

Recall that, Sections III-C-VI assumed large capacity caches. For capacity-constrained caches, the following Algorithm 2 is a natural extension of LIN-GR. Now instead of going through a sequence of tight solutions generated by iterating only on the caches, now we iterate on every pair of (cache, content) such that the content is not stored in that cache. In every step we select the pair that contributes to the maximum reduction in the overall cost.

Algorithm 2: Cache-Fill: Heuristic for PRRO

Input: The cache network

1 Start with a tight solution with $\mathbf{y} = \mathbf{0}$. Compute the cost in Equation (6).

2 For every (cache, content) pair such that the content is not already stored in the cache and the cache has spare capacity, compute reduction in the total cost by forming a tight solution assuming that the content is going to be stored in that cache.

3 Selected the (cache, content) that gives maximum cost reduction and check if the cost reduction is positive. If no cache satisfies 3 then exit.

4 Otherwise store the selected content in the selected cache for T slots and decrement its available capacity. Identify all the users whose requests for the selected content can now be routed through the selected cache (and such that they do not already route the requests for this content through some other cache). Route the requests for all such users for the selected content through the selected cache. Go to step 2.

Output: User-cache association $\mathbf{q}$ and retentions $\mathbf{y}$.

The heuristics LIN-GR and Cache-Fill can be extended for MIP-a and PRRO for the case of convex storage cost. For convex cost, we do not restrict ourselves to only tight solutions since Theorem 2 may not hold any more. To see it, let us take an example where there is only one cache and only one user with $p = 0.5$ (since there is only one user, this example will hold for unicast as well as multicast server mode). Let us set $\alpha = 1$, $T = 2$ and $D = 4$. If y is the number of slots for which the cache stores the content then the cost of such a solution becomes $\alpha y^2 + D(T - y)p = y^2 + 2(2 - y)$. This equation is minimized at $y = 1$ (the storage duration is assumed to be an integer). This is the unique optimal solution of this instance and it is not a tight solution.

To extend LIN-GR to MIP-a with convex storage cost, in each iteration for each empty cache (say cache j) we minimize the following convex function (which is same as the change in the total cost if we store the content in cache j for duration y):

$$\alpha h(y) + D \cdot y \cdot \prod_{i \in J(\mathbf{y})} (1 - p_i) \cdot \left(\frac{1}{\prod_{i \in J(\mathbf{y}) \cap U_j} (1 - p_i)} - 1 \right) \tag{28}$$

Using the above equation for every empty cache in each iteration, we compute the cache which leads the maximum download cost reduction in step 2 of LIN-GR (the reduction in download cost is computed by subtracting the minimum value of Eqn (28) from the download cost at the beginning of the iteration). Every other step of LIN-GR remains the same. Extending Cache-Fill to convex storage cost can be done in an analogous way, where we iterate over each (content, cache) pair.

Note that with convex storage cost with unicast mode, as Theorem 2 does not hold any more for unicast (as shown in the example in the preceding discussion), we do not have a bound similar to that from Theorem 6. Instead, for a heuristic, we apply LIN-GR and Cache-Fill verbatim except for computation we use unicast download cost instead of multicast download cost.

VIII. Performance Evaluation

In this section, we numerically evaluate the performance of our algorithms on a real dataset [34] obtained from a sporting event with thousand attendees covered by a Mobile Base Station (MBS) and several Small Cell Base Station (SBSs).

A. Evaluations setup

Our evaluations are based on the dataset from the Superbowl event, held at the New Orleans Superdome, in 2013. The stadium consists of a Macro Base Station (MBS) providing cellular coverage to the viewers

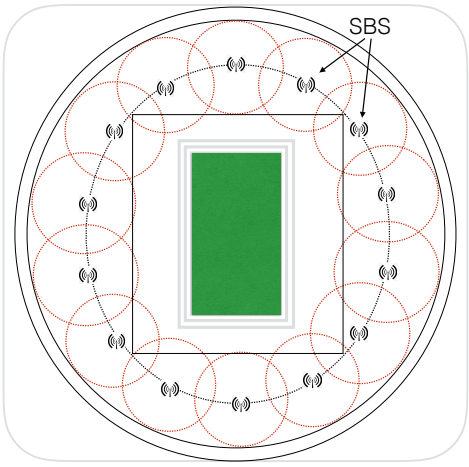
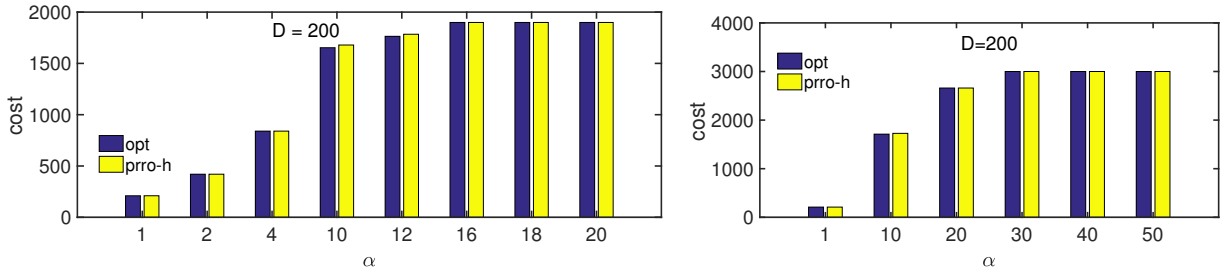


Fig. 3: Stadium consisting of 14 SBSs providing overlapping coverage to users. The requests that result in a cache miss at the SBS are served by a MBS (omitted here).

(or users) along with $N = 14$ other SBSs. See Figure 3 for the stadium layout. We would like to emphasize that, as with all real world deployments, the coverage areas of SBSs may overlap as shown in the figure; this is in contrast with [8] where it was explicitly assumed that the cache coverage areas do not overlap. Over fifty thousand viewers attended the event generating three thousand requests for a thousand distinct contents over the span of four hours. This gives us $I = 50,000$, $M = 1000$. To model the content arrival process, we spread the 30,000 file requests over 1,000 files using a ZipF popularity distribution with an exponent of $\beta_m = 1.2$ [8]. Moreover, we further spread the requests (for various contents) over users by choosing a ZipF popularity exponent $\beta_u = 0.9$. Thus, we obtain the probability of user i requesting content m, i.e. p_{mi}, for each {user, content} pair.

We divide the four hour time duration in 16 frames of duration $T = 15$ minutes each, with each frame having a slot of duration $d_T = 1$ minute. We also vary T, d_T in our evaluation, but unless otherwise mentioned we consider $T = 15$ and $d_T = 1$ minute. We assume that users are distributed uniformly in the annular region in the stadium as shown in Figure 3. A user associates to a single cache if they lie in the non-overlapping SBS coverage region, or two caches if they lie in the overlapping coverage region (see Figure 3).

We induce mobility in users by assuming that $U_{pick} = 5\%$ of users (i.e. 2500 out of 50000 users) change their positions on every frame. The mobile users are chosen independently across frames. The changed position is sampled uniformly at random from the list of SBSs that the user is not associated with currently. We then randomly associate the user with either the chosen SBS or a pair of SBSs consisting of the chosen SBS and the one adjacent to it. (Note that SBS 1 is adjacent to SBS 14, in our stadium).



(a) Multicast (b) Unicast

Fig. 4: Comparison of the cost of the output of LIN-GR (i.e., prro-h applied to MIP-a) with the cost of the optimal solution

A random association in the mobility model can emulate user movements during the game, in the form of snack or restroom breaks, which can be a few SBSs away.

B. Comparison of PRRO output with optimal solution

In this section, we compare the cost of the output of LIN-GR against the optimal computed by brute force. We assume that all the caches have a large capacity, i.e., we consider MIP-a version of the problem. We do this since computing the optimal solution for PRRO by brute-force is computationally infeasible for capacitated caches because we need to iterate over all possibilities of how M contents can be stored in N caches. For MIP-a version of the problem, we evaluate the value of tight solution corresponding to each possible value of $\mathbf{y}$, for which MIP-a can take 2^N possible values, thus making computation of an optimal solution computationally feasible for our setting of $N = 14$. To compute the value of optimal solution, we exploit the fact that there exists a tight optimal solution (See Theorem 2) and that given a storage vector $\mathbf{y}$ the value of a corresponding tight solution is uniquely determined (see Eqn (18)).

For our experiments in this section, we compare the cost obtained from LIN-GR against the cost of the optimal solution by keeping D constant and varying α. All other parameters are assumed to take default values as mentioned in the preceding discussion unless otherwise mentioned. Figure 4 shows the results of our evaluations, from which we conclude that for our settings the output of LIN-GR is almost as good as the optimal solution in terms of cost incurred over a wide range of D and α (these results are shown for a representative D and α as the results for other values are similar).

C. Comparison of PRRO with contemporary caching policies

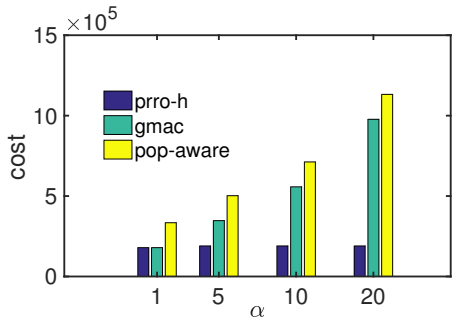
We first consider PRRO with multicast mode of server transmissions and compare its performance with the state-of-the art policies: (1) GMAC proposed in [8], which is also similar to PRAC proposed

in [3], and (2) the popularity aware caching policy. We also evaluate PRRO with unicast mode of server transmissions (with Algorithm 2 adopted to unicast mode by replacing cost computation for multicast by unicast) and compare it with femtocaching [11]. This distinction is necessary since femtocaching is configured for unicast transmissions and is not designed for multicast server transmissions. The policies used for comparison are described below:

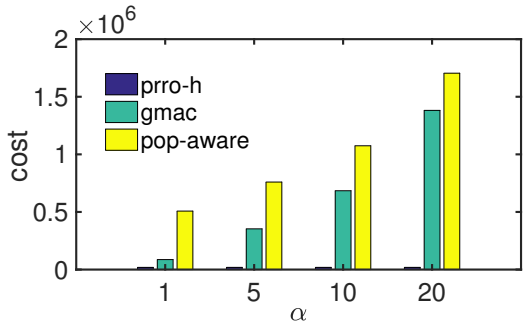
1) *prro-h:* This is the policy outlined in the algorithm Cache-fill (Algorithm 2). Recall that for linear cost the storage decisions are binary in nature, i.e., a content is stored in a cache for duration T or is not cached at all. However, for convex storage cost, the retention durations can take other values (calculated as described towards the end of Section VII).
2) *femtocaching:* This is a policy described in [11]. At each iteration, find the (content, cache) pair that minimizes the download cost when this content gets stored in the cache considered; the algorithm terminates when all caches are full. Contents are served via directed unicast transmissions. This algorithm can be viewed as solving prro-h with α set to 0.
3) *gmac:* This is a policy described in [8], also similar to the one in [3]. It is a natural variant of PRRO in that the users are associated to a single cache instead of being associated to two (or multiple) caches. This policy is executed in the same way as prro-h, except that in gmac the decision of routing and storage is not jointly done. Instead, we randomly associate users with one of the caches they can associate with and subsequently compute which content every cache should store (subject to the capacity) based on how frequently a content is requested by the users routing via the cache. Contents are served via multicast transmissions.
4) *pop-aware:* Cache the locally most popular files at each cache till all the caches are filled. Contents are served via multicast transmissions.

Femtocaching, gmac and pop-aware are widely used caching policies, however, they either do not take storage costs into account while making caching decisions (such as in femtocaching or pop-aware caching) or do not make joint caching and routing decisions (such as in gmac). We compute caching assignments as per the policy specifications, and then compute the final cost by also including the storage cost incurred by these caching assignments.

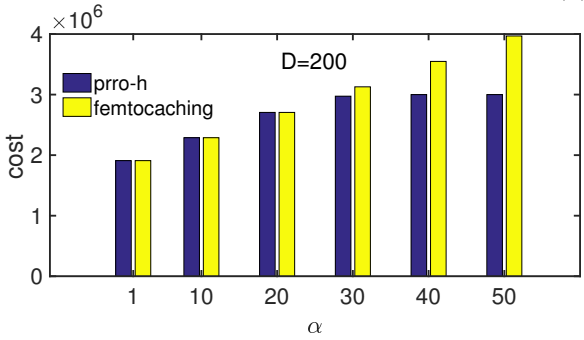
Results: For all the above caching policies, we collect the cost evolution over all frames and report the expected cost incurred per frame in the rest of this section. We study the impact of varying the following design parameters: the storage cost (α) and the cache capacity (B) at each SBS. We note that the effect of



(a) linear storage cost with multicast server mode



(b) convex storage $h(y) = y^2$ with multicast server mode

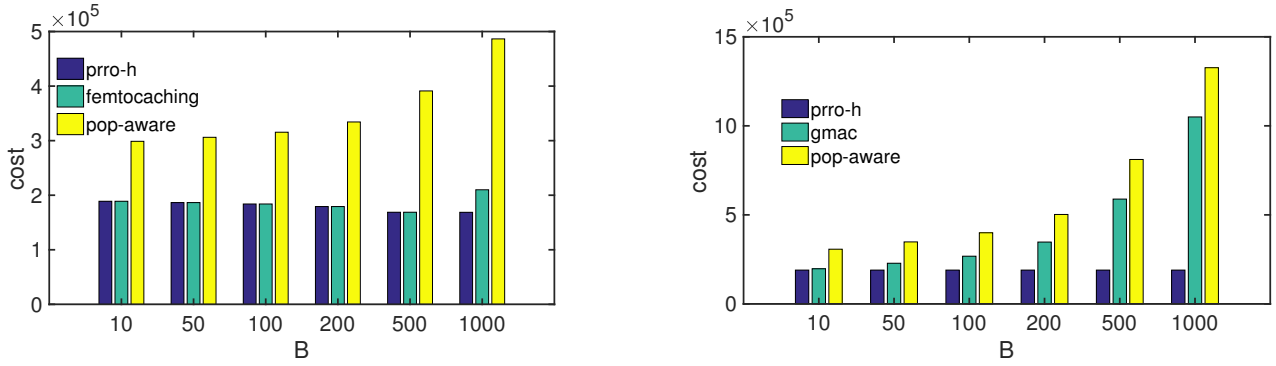


(c) linear storage cost with unicast server mode

Fig. 5: Impact of increasing α.

varying D is similar to the effect of varying α as can be intuitively expected since it effectively changes the impact of storage cost with respect to the download cost, thus to avoid repetition, we only focus on the case of changing α. Unless otherwise mentioned, we use the following parameter values: $N = 14$, $M = 1000$, $\beta_m = 1.2$, $\alpha = 5$, $D = 20$, $T = 15$, $d_T = 1$, $B = 200$.

1) Impact of increasing α: For multicast server mode, we vary α in the set $\{1, 5, 10, 20\}$ for both linear and convex storage costs as shown in Figure 5. We observe that when storage cost is linear in retention time, then gmac performs equally well as prro-h for $\alpha = 1$, however, as α increases, gmac starts getting worse. This is because when $\alpha = 1$ (i.e., storage is relatively inexpensive), both prro-h and gmac store maximum number of contents (respecting their cache capacities) to minimize the total cost. When α increases, prro-h makes joint storage and routing decisions unlike gmac, thus gmac incurs progressively higher total cost. Note that, pop-aware caching is always worse than prro-h as well as gmac for all parameters considered. This is because it makes decisions based on local file popularity instead, which can make it replicate content storage more than necessary. With convex storage cost we always get better gains than linear as expected, since other storage-unaware policies incur more storage cost in this case. For unicast server mode as well, we observe prro-h performing better than femtocaching policy on the basis of overall cost especially for larger values of α.



(a) $\alpha = 1$ (b) $\alpha = 5$

Fig. 6: Impact of increasing B.

2) Impact of increasing B: We vary the cache capacity from 1% to 100% of M and plot the results in Figure 6 when $\alpha = 1$ and when $\alpha = 5$. When $\alpha = 1$, then both prro-h and gmac perform similar by storing the important contents since storage cost is less. When $\alpha = 5$, however, we observe that, larger cache capacities can provide up to $6 - 7$ fold cost reduction with prro-h when compared to that with femtocaching and pop-aware caching. This is because prro-h makes smart caching decisions by optimizing over both storage and download cost thus maintaining the overall cost almost constant with increasing B. This can be seen as follows: With increasing B, we intuitively expect prro-h to decrease overall cost, since prro-h chooses to utilize additional storage space only if storing some content brings the overall cost further down.

IX. Conclusion

We considered proactive retention aware caching on a heterogeneous network with mobile users associated with possibly multiple caches. Our goal is to design a caching policy that minimizes the sum of costs of content storage and content downloads. We proved that this problem is NP-Hard even when the caches have a large capacity. We developed a simple greedy algorithm and assessed its efficiency for large caches by means of investigating a core family of solutions which we call tight solutions. By means of constructing a sparse instance which bounds the performance for each instance, we derived upper bounds on its performance which work well over multiple regimes. Finally, we propose a heuristic for the capacitated cache case. Systematic performance evaluations show that the our greedy algorithms perform closely to the optimal solutions as well as demonstrate the effectiveness of our approach as compared to the existing caching schemes.

A natural extension of our work would be to consider a general network with a combination of storage and delivery cost where the content can be stored in the network at various edge nodes to facilitate quicker delivery. Since our problem turns NP-hard quickly with the introduction of cache capacities or overlapping user base, it would be interesting to relax these constraints to see if it remains tractable for some practical topologies, for example, the hierarchical Internet structure. Another interesting direction could be to investigate the setting when the caches are allowed to set prices for routing the content to users.

ACKNOWLEDGEMENT

This material is based upon work supported in part by the National Science Foundation under Grant No. 1422153, and by a Tekes FiDiPro award to A. Abouzeid through University of Oulu, Finland.
This research was partly sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copy-right notation hereon.

REFERENCES

[1] S. Shukla, O. Bhardwaj, A. A. Abouzeid, T. Salonidis, and T. He, "Hold'em caching: Proactive retention-aware caching with multi-path routing for wireless edge networks," in *Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. Mobihoc '17, 2017.

[2] I.-H. Hou, T. Zhao, S. Wang, K. Chan *et al.*, "Asymptotically optimal algorithm for online reconfiguration of edge-clouds," in *Proceedings of the 17th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2016, pp. 291–300.

[3] S. Shukla and A. A. Abouzeid, "Proactive Retention Aware Caching," *IEEE INFOCOM*, 2017.

[4] P. Blasco and D. Gündüz, "Learning-Based Optimization of Cache Content in a Small Cell Base Station," *CoRR, 2014*.

[5] B. Bharath, K. Nagananda, and H. V. Poor, "A Learning-based Approach to Caching in Heterogenous Small Cell Networks," *IEEE Transactions on Communications, 2016*.

[6] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Improving Energy Efficiency of MPTCP for Mobile Devices," *arXiv preprint arXiv:1406.4463*, 2014.

[7] E. Hamed, H. Rahul, M. A. Abdelghany, and D. Katabi, "Real-time Distributed MIMO Systems," in *ACM SIGCOMM 2016*, 2016.

[8] K. Poularakis, G. Iosifidis, V. Sourlas, and L. Tassiulas, "Exploiting Caching and Multicast for 5G Wireless Networks," *IEEE Transactions on Wireless Communications*, 2016.

[9] J. Könemann, O. Parekh, and D. Segev, "A unified approach to approximating partial covering problems," *Algorithmica*, vol. 59, no. 4, pp. 489–509, 2011.

[10] Technical-Report, "Proactive retention-aware caching with multi-path routing for wireless edge networks," https://www.dropbox.com/sh/7crye4sl6muvmac/AAAOu1Eb-8JMw-HfqzshKVTXa?dl=0.

[11] K. S. et al, "FemtoCaching: Wireless Content Delivery Through Distributed Caching Helpers," *IEEE Transactions on Information Theory*, 2013.

[12] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation Algorithms for Mobile Data Caching in Small Cell Networks," *IEEE Transactions on Communications*, 2014.

[13] M. D. et al., "On the Complexity of Optimal Routing and Content Caching in Heterogeneous Networks," *CoRR, 2015*.

[14] E. B. et al., "Big Data Meets Telcos: A Proactive Caching Perspective," 2015.

[15] J. Tadrous, A. Eryilmaz, and H. E. Gamal, "Proactive Resource Allocation: Harnessing the Diversity and Multicast Gains," *CoRR*, 2011.

[16] U. Niesen and M. A. Maddah-Ali, "Coded Caching for Delay-Sensitive Content," in *2015 IEEE International Conference on Communications, ICC*, 2015.

[17] B. Zhou, Y. Cui, and M. Tao, "Optimal Dynamic Multicast Scheduling for Cache-Enabled Content-Centric Wireless Networks," in *2015 IEEE International Symposium on Information Theory (ISIT)*, 2015.

[18] J. Tadrous, A. Eryilmaz, and H. E. Gamal, "Joint Smart Pricing and Proactive Content Caching for Mobile Services," *IEEE/ACM Transactions on Networking*, 2016.

[19] J. Tadrous and A. Eryilmaz, "On Optimal Proactive Caching for Mobile Networks With Demand Uncertainties," *IEEE/ACM Transactions on Networking*, 2016.

[20] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive Content Download and User Demand Shaping for Data Networks," *IEEE/ACM Transactions on Networking*, 2014.

[21] N. C. Fofack, M. Dehghan, D. Towsley, M. Badov, and D. L. Goeckel, "On the Performance of General Cache Networks," in *ACM VALUETOOLS, 2014*.

[22] N. É. C. Fofack, "On Models for Performance Analysis of a Core Cache Network and Power Save of a Wireless Access Network," Ph.D. dissertation, Université Nice Sophia Antipolis, 2014.

[23] R. Bi, Y. Li, and X. Zheng, "An Optimal Content Caching Framework for Utility Maximization," *Tsinghua Science and Technology*, 2016.

[24] R. T. Ma and D. Towsley, "Cashing in on Caching: On-demand Contract Design with Linear Pricing," *CoNext, December*, 2015.

[25] 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 Communications Magazine*, 2014.

[26] K. P. et al, "Exploiting Caching and Multicast for 5G Wireless Networks," *IEEE Transactions on Wireless Communications*, 2016.

[27] S. Shukla and A. A. Abouzeid, "On Designing Optimal Memory Damag Aware Caching Policies for Content-Centric Networks," in *IEEE WiOpt, 2016*.

[28] N. Abedini and S. Shakkottai, "Content Caching and Scheduling in Wireless Networks With Elastic and Inelastic Traffic," *IEEE/ACM Transactions on Networking*, 2014.

[29] M. Leconte, G. Paschos, L. Gkatzikis, M. Draief, S. Vassilaras, and S. Chouvardas, "Placing Dynamic Content in Caches with Small Population," *CoRR*, 2016.

[30] S. Dernbach, N. Taft, J. Kurose, U. Weinsberg, C. Diot, and A. Ashkan, "Cache Content-Selection Policies for Streaming Video Services," *IEEE INFOCOM*, 2016.

[31] L. Stornaiuolo and P. Bellavista, "State-of-the-art Multihoming Solutions for Android: A Quantitative Evaluation and Experience Report," in *International Conference on Network and Service Management (CNSM)*, 2015.

[32] S. Shukla and A. A. Abouzeid, "Optimal Device-Aware Caching," *IEEE Transaction on Mobile Computing*, 2016.

[33] Ö. Alay, T. Korakis, Y. Wang, E. Erkip, and S. S. Panwar, "Layered Wireless Video Multicast Using Relays," *IEEE Transactions on Circuits and Systems for Video Technology*, 2010.

[34] J. Erman and K. K. Ramakrishnan, "Understanding the Super-sized Traffic of the Super Bowl," in *Internet Measurement conference, ACM*, 2013.