Hold'em Caching: Proactive Retention-Aware Caching with Multi-path Routing for Wireless Edge Networks

Samta Shukla
Rensselaer Polytechnic Institute
shukls@rpi.edu

Onkar Bhardwaj*
Akamai Technologies
obhardwa@akamai.com

Alhussein A. Abouzeid
Rensselaer Polytechnic Institute
abouza@rpi.edu

Theodoros Salonidis
IBM Research
tsaloni@us.ibm.com

Ting He
Penn State University
tzh58@cse.psu.edu

ABSTRACT

We consider the problem of proactive retention aware caching in a heterogeneous wireless edge network consisting of mobile users connected to 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 transmissions costs over two design variables: the retention time of each cached content and the probability that a user routes content requests to 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 multipath 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 develop a greedy algorithm that has provable performance bounds. Finally, we propose a heuristic for the capacitated cache case that has low computational complexity. Systematic evaluations including real data sets demonstrate the effectiveness of our approach, compared to the existing caching schemes.

KEYWORDS

Proactive caching, Mobile users, Wireless networks, Edge caching, Multicast, Storage cost, Approximation algorithms.

1 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 has 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 [1]. 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 cache.

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 [2].

In this paper, we address the problem of proactive retention-aware caching and request routing for edge wireless networks. Our objective is to minimize the sum of cache storage cost and server access transmission costs. In contrast with 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 [3, 4].

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 multipath networking technologies [5, 6]. 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) [7]. Fourth, in addition to cache capacity constraints, it captures

*A major part of this work was done when the author was at IBM Research, Yorktown Heights, NY, USA.

practical cache access constraints which limit the number of users that can simultaneously request contents from a cache.

Our contribution: 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).

We first investigate PRRO for the more challenging case of multicast server access costs. We prove that the problem is NP-Hard even under no cache capacity constraints (Theorem 4.1). We then propose a greedy algorithm that gives a constant-factor approximation 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 4.3), 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 (Corollary 4.4). Thus our greedy algorithm focusses only on the class of tight solutions. We also show that for any given instance, there exists another instance with a much simpler network (Lemma 4.5) 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 in order to bound the performance of the greedy heuristic (Theorem 4.6).

We next investigate PRRO for the unicast server access cost case. We prove that the problem remains NP-Hard but a similar greedy approximation algorithm applies using the unicast-based optimization objective. We prove a bound on the performance of this algorithm in terms of k, where k is the maximum number of users a cache can serve (Theorem 4.8). Finally, we develop a low complexity heuristic based on PRRO to create a feasible solution for the case when the caches are capacitated (Algorithm 2).

We evaluate our algorithms based on a real-world dataset by explicitly modeling user mobility and the overlapping cache coverage for a wide range of parameters. Through simulations, we observe that the algorithms are fast to execute and perform very close to the optimal for reasonable parameter choices. Finally, we compare the results for the cache capacitated case against various other popular policies and prior works. In our simulations based on real-world data, we show that the solutions computed by our heuristics are close to the optimal solutions. Moreover, taking advantage of multiple paths in the user-cache association graph (in order to jointly optimize storage and routing) can result in significantly lower costs than making a user associate a-priori with a cache (or alternatively creating a disjoint user base for each cache) and then optimizing the cost based on locally popular contents at each cache.

Paper organization: The rest of the paper is organized as follows: We begin by describing related work in Section 2. We define the system model and formulate the problem of Proactive Retention Routing Optimization (PRRO) in Section 3. Section 4 studies the multicast mode of server transmissions wherein we derive an upper bound on the performance of a greedy heuristic using tight solutions under the large cache assumption (the case when the problem decouples over different contents). We continue the discussion with a unicast mode of server transmission in Section 4.4. We propose a heuristic for the cache capacitated case in Section 5, and evaluate the performance of our algorithms on a real data-set in Section 6.

2 RELATED WORK

Unicast proactive caching. Proactive caching in small cell networks has been addressed in [8]. Poularakis et al. addressed a similar problem for minimizing server downlink cost [9]. Dehghan et al. addressed a joint request routing and caching problem for minimizing access delay, where users can route requests to multiple caches [10]. A recent work [11] considered proactive caching without multicast. 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 [2, 12–14]. Storage is only considered in [2, 7] which aim to optimize the sum of storage and server cost. The work in [7] 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 [2] 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 [12, 15–17]). Another line of work considered the problem of finding the **optimal timers** for every content under various objectives. For example, [18, 19] studies cache hit-rate maximizing timers, [20] studies utility maximizing timers, and [21] 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.

3 MODEL AND PRELIMINARIES

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

3.1 System Model

We consider the problem of proactive retention-aware content caching in a 5G HetNet architecture [22, 23] 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 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

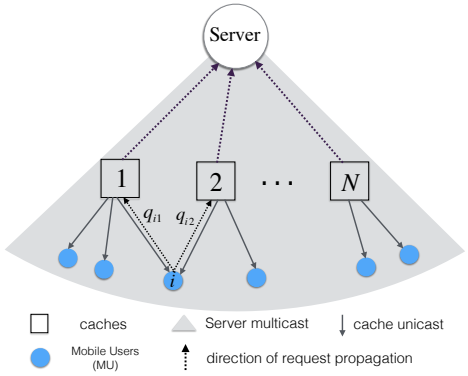

Figure 1: Proactive caching in a 5G Hetnet.

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.

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 (e.g. [24, 25] make this assumption). A downlink multicast transmission of a content from the server can be received by all the users (as in [23, 26]); a downlink unicast transmission is a directed point-to-point transmission received only by the targeted cache. In our model, the caches can employ either multicast or unicast mode of transmissions (depending on the underlying application/technology) to locally serve the demands of the associated user-base.

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 a cache miss, the server transmits the content during the time slot. A delay-intolerant service assumption is pertinent to serving media contents, user-generated, or video content [16, 25]. The value of d_T is determined by the delay intolerance requirements, whereas the value of T is determined by the periodicity in the request patterns [2, 11, 16] and/or is chosen to suit the mobility footprint of users in a specific application (see Section 6 for details).

Users request content independently at every slot during the frame. 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 (see [27, 28]).

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 $t = 0$.

A mobile user can *associate* with the caches that are in charge of the network infrastructure of the area they are moving into. Obviously, a user cache association is possible only when 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 (infrastructure) caches, and once they are determined, routing the optimal fraction of requests in any slot can be initiated by the user. Let q_{imn} denote the *request routing probability* with which user i routes content m to cache n. The network can be expressed 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. This is a generalization of the model used in [2] wherein cache coverage areas did not overlap and a given user could only get served by a single cache. Our model of allowing for multiple user-cache connections captures the fact that current mobile devices come with multiple wireless network interfaces [6] and can use multi-homing technologies such as LISPmob [29] or multi-path TCP [5].

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 caches. Storing a content m in cache n with retention y_{mn} incurs a *storage cost*, $\alpha g(y_{mn})$, where α is the storage cost parameter, $\alpha \geq 0$, and $g(\cdot)$ is a function of retention time, y_{mn}. In this work, we assume that $g(\cdot)$ is an increasing linear function of retention times with $g(0) := 0$. The case of a convex $g(\cdot)$ is still open. Our choice of an increasing linear function 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 the longer we keep a content in cache, since this cannot be done without declining to keep other contents, owing to the cache capacity constraints. When storage cost is interpreted as memory damage then, as explained in [2, 24, 30], a higher retention time can potentially lead to a higher memory damage.

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 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 each 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

either unicast or multicast. 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 $\geq$ the storage cost parameter, i.e. $D_z \geq \alpha$ for $z \in \{\text{mul}, \text{uni}\}$, which is reasonable since transmissions over backhaul is more expensive than storage. Our downlink model for multicast is similar to [7, 25], 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 [7, 31] besides being incorporated 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 by a cache by having multiple wireless interfaces as in [5] or by invoking multi-homing feature [29]. Interference free transmission can be achieved by synchronizing base stations to transmit simultaneously at the same frequency band for multicast services in cellular networks (3GPP and WiMax) [7] or in WiFi networks [6].

3.2 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 of caching content m over the frame is given by,

$$\text{Cost}_\text{s}(\mathbf{y}) = \sum_{n \in [N]} \alpha y_{mn}. \tag{1}$$

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 $U(S)$ denote the sets of all users that are associated with cache n and a set of caches S respectively. Also, let $H(i)$ and $H(S)$ be the set of all caches associated with a user i and a set of users S respectively.

3.2.1 Multicast download cost. In case of a multicast server, a cache miss on content m incurs a server multicast cost D_{mul} 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:

$$\begin{aligned}
&P(\text{ Server transmits content } m)\\
&= 1 - P(\text{ none of the caches request } m \text{ from server})\\
&= 1 - \prod_{i \in [I]} P\Big(i \text{ does not request } m\\
&\quad \cup \text{ requests from user } i \text{ for } m \text{ are locally served}\Big)\\
&= 1 - \prod_{i \in [I]} \big((1 - p_{mi}) + \sum_{n \in H(i): y_{mn} \geq t} p_{mi} q_{imn}\big)
\end{aligned} \tag{2}$$

Thus the multicast cost for downloading content m over the frame is given by,

$$\text{Cost}_{\text{mul}}(\mathbf{y}, \mathbf{q}) = D \sum_{t=1}^{T} P(\text{ Server transmits content } m) \tag{3}$$

3.2.2 Unicast download cost. Upon a cache miss on content m, a server unicast cost D_{uni} is incurred if the content is requested from a cache that does not have the content stored. Thus the cost of unicasting content m over the frame is given by:

$$\text{Cost}_{\text{uni}}(\mathbf{y}, \mathbf{q}) = D_{\text{uni}} \sum_{t=1}^{T} \sum_{i \in [I]} \sum_{n: y_{mn} < t} p_{mi} q_{imn}, \tag{4}$$

where 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.

3.2.3 Access constraints. We consider that a cache i associates with at most k_i users. This is especially relevant in 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 $n \in [N]$.

3.2.4 The formulation. Let $L_z(\mathbf{y}, \mathbf{q})$ denote the sum of storage and download costs, where $z \in \{\text{uni}, \text{mul}\}$. Then, the aggregate cost over all contents is given by:

$$L_\text{z}(\mathbf{y}, \mathbf{q}) = \sum_{m \in [M]} (\text{Cost}_\text{s}(\mathbf{y}, \mathbf{q}) + \text{Cost}_\text{z}(\mathbf{y}, \mathbf{q})) \tag{5}$$

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?* Thus, the problem of Proactive Retention Routing Optimization (PRRO) can be formulated for $z \in \{\text{uni}, \text{mul}\}$ as,

$$\begin{aligned}
\textbf{PRRO}: \quad & \min_{\mathbf{y}, \mathbf{q}} \; L_z(\mathbf{y}, \mathbf{q})\\
& \text{s. t.} \sum_{m \in M} 1_{y_{mn} > 0} \leq B_n, \forall n \in [N] && (6)\\
& y_{mn} \in \{0, 1, 2, \cdots, T\}, \; \forall n \in [N], m \in [M] && (7)\\
& \sum_{n \in H(i)} q_{imn} = 1, \; \forall i \in [I], m \in [M], && (8)\\
& q_{imn} \in [0, 1], \; \forall i \in [I], m \in [M], n \in [N] && (9)\\
& U(n) \leq k_n \; \forall n \in [N] && (10)
\end{aligned}$$

where $1_{y_{mn} > 0}$ is an indicator function which equals 1 if $y_{mn} > 0$ and 0 otherwise. The first constraint (6) in the above formulation shows that the number of contents prefetched do not exceed the individual cache capacities. Constraint (7) models that the retention times take integral values in 0 to T, with a 0 meaning that a content is not stored. Constraint (8) indicates that user i can

Notation	Meaning
p_{im}	Per-slot probab. 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 resp.
$[I], [M], [N]$	The set of all users, all contents, all caches resp.
p_i, q_{in}, y_n	Restrictions of p_{im}, q_{imn}, y_{mn} for MIP-a which considers a single content.
$\mathbf{y}, \mathbf{q}$	The 2D/3D matrices of y_{mn}, q_{imn} resp. (and vector/2D matrix of y_n, q_{in} for MIP-a resp.)
α	The storage cost coefficient
D_{mul}, D_{uni}	Server transm. cost for multicast/ unicast.
D	Represents D_{uni} or D_{mul} in the context.
$(\mathbf{y}, \mathbf{q})$	Any solution and an optimal solution resp.
$L(\mathbf{y}, \mathbf{q})$	Objective function value of a solution $(\mathbf{y}, \mathbf{q})$.
$(\mathbf{y}^*, \mathbf{q}^*)$	An optimal solution.
$(\mathbf{y}^l, \mathbf{q}^l)$	The solution of LIN-MUL heuristic.
$U_n, U(S)$	Set of users associated with cache n or set S.
$H_i, H(S)$	Set of caches user n or set S of users associates with.
$F(\mathbf{y})$	Set of caches storing content in a tight solution $(\mathbf{y}, \mathbf{q})$.
$J(\mathbf{y})$	$\{i \in [I] : y_n = 0,\ \forall n \in H(i)\}$

Table 1: Notation

request content m only from the caches it is connected to. Finally, constraint (9) 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 1.

4 ANALYTICAL RESULTS WITH LARGE CACHES

In this section, we analyze 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 Section 5.

Note: All the omitted proofs as well as the proofs for which we only provide an outline can be found in the Appendices of [32].

With uncapacitated caches, the problem decouples across contents, thus PRRO gets reduced 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 denoting the retention time, y_n, 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 server download mode in Equation (5) can thus be expressed as,

$$L_{\text{mul}}(\mathbf{y}, \mathbf{q}) = \sum_{n\in[N]} \alpha y_n + D_{mul} \sum_{t=1}^{T} \left[1 - \prod_{i\in[I]} \left[(1-p_i) + p_i \sum_{n:y_n>t} q_{in} \right] \right] \quad (11)$$

Similarly, cost of a solution $(\mathbf{y}, \mathbf{q})$ with unicast server download mode in Equation (5) can thus be expressed as

$$L_{\text{uni}}(\mathbf{y}, \mathbf{q}) = \sum_{n\in[N]} \alpha y_n + D_{uni} \sum_{t=1}^{T} \sum_{i\in[I]} p_i \sum_{n:y_n>t} q_{in} \quad (12)$$

Thus for uncapacitated caches, without loss of generality, solving PRRO is equivalent to solving independent 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.

$$\begin{aligned} \textbf{MIP-a}: \quad & \min_{\mathbf{y},\mathbf{q}} \ L_{\text{z}}(\mathbf{y}, \mathbf{q}) \\ \text{s. t.} \quad & y_n \in \{0, 1, 2, \cdots, T\}, \ \ \forall n \in [N] \\ & \sum_{n\in H(i)} q_{in} = 1, \ \ \forall i \in [I] \\ & q_{in} \in [0,1], \ \ \forall i \in [I], n \in [N] \\ & U(n) \le k_n, \ \forall n \in [N] \end{aligned}$$

THEOREM 4.1. *MIP-a is NP-hard.*

We give the full proof in [32]where we show that for $D \gg \alpha$, set cover is a special case of MIP-a. 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: (1) In [2] where the authors show that poly-time solutions can be obtained *without* cache capacities for linear storage cost and disjoint cache coverage, and (2) In [7] where the authors prove hardness *with* cache capacity constraints but *without* overlapping cache coverage.

4.1 Properties of the optimal solution

Our aim in this section is to characterize properties of the optimal solution, to later employ these insights for constructing an approximation algorithm. In particular, we will show that there always exists an optimal solution to MIP-a such that all the caches either store the content for all T slots or do not store it all, and each user routes its requests only to one cache. In what follows, we denote the optimal solution of MIP-a as $(\mathbf{y}^*, \mathbf{q}^*)$.

THEOREM 4.2. *For every instance of MIP-a, there exists an optimal solution $(\mathbf{y}^*, \mathbf{q}^*)$ with the following properties:*

- *(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) $y^*_n \in \{0, T\}\ \forall n \in [N]$, i.e., the caches either store the content for all T slots or do 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 $q^*_{in} = 1$.*

4.2 A greedy approximation algorithm

With the insights obtained from Theorem 4.2, we now develop a simple, greedy Algorithm LIN-MUL that approximates MIP-a. We will restrict our search to the solutions which have similar properties as the properties of an optimal solution $(\mathbf{y}^*, \mathbf{q}^*)$ from

Theorem 4.2. To begin with, we define a class of solutions which has exactly the same properties as that of the optimal solution from Theorem 4.2 except that the solution may not be optimal:

Definition 4.3. A **tight solution** $(\mathbf{y}, \mathbf{q})$ is a solution where

- $q_{in} \in \{0,1\}\ \forall i \in [I], n \in [N]$. 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.
- $y_n \in \{0, T\}\ \forall n \in [N]$, i.e., the caches either store the content for all T slots or do not store it at all.
- For a user i, if there exists a cache $n \in H(i)$ which stores the content for T slots then $q_{in} = 1$.

Using Definition 4.3, we can re-state Theorem 4.2 as follows:

COROLLARY 4.4. *For every instance of MIP-a, there exists a tight optimal solution* $(\mathbf{y}^*, \mathbf{q}^*)$.

Note: 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 the purpose of developing candidate solutions**. Thus we will assume that Definition 4.3 holds for every solution $(\mathbf{y}^l, \mathbf{q}^l)$ from here onwards.

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., $F(\mathbf{y}) = \{n \in [N] : y_n = T\}$. The cost of a tight solution $(\mathbf{y}, \mathbf{q})$ can then be expressed as

$$L_{\text{mul}}(\mathbf{y}, \mathbf{q}) = \alpha T|F(\mathbf{y})| + D_{mul}T\left(1 - \prod_{i \in J(\mathbf{y})}(1 - p_i)\right) \quad (13)$$

$$L_{\text{uni}}(\mathbf{y}, \mathbf{q}) = \alpha T|F(\mathbf{y})| + D_{mul}T\sum_{i \in J(\mathbf{y})} p_i \quad (14)$$

Below we describe our greedy Algorithm LIN-MUL which iterates through tight solutions to find a heuristic solution to MIP-a. LIN-MUL *sequentially* iterates over empty caches and in every iteration it stores the content in a cache which provides the largest one-shot cost reduction if it stores the content compared to when it does not store the content. Note that since LIN-MUL only searches through tight solutions, its final output is a tight solution.
Discussion: Although LIN-MUL and other heuristics we develop for MIP-a (and PRRO) will explore only tight solutions which either store the content in a cache for the full duration of T slots or do not store it there at all, note that these heuristics and subsequent performance bounds are derived taking into account retention-aware objective.

We know that LIN-MUL is suboptimal since for $D \gg \alpha$, MIP-a has set-cover as its special case. For a specific example of suboptimality, see Example 1 in [32]. However, in the performance evaluation section, we show that LIN-MUL output stays close to the optimal solution in simulations based on real-world data.

4.3 Bounding the performance of LIN-MUL for multicast server mode

For any given MIP-a instance E (with multicast server mode), we are interested in how well LIN-MUL performs. Specifically,

Algorithm 1: LIN-MUL

Input: The cache network

1 Start with any tight solution with $\mathbf{y} = \mathbf{0}$. Compute the cost in Equation (13).
2 For every empty cache, compute the reduction in the total cost by forming a tight solution assuming that the content is going to be stored in that cache.
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. Associate all the users (who can get served by this cache and are not associated with any other cache) with this cache.
4 If no cache satisfies 3 then exit, or else go to step 2.

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

if $(\mathbf{y}^l, \mathbf{q}^l)$ denotes the solution given by LIN-MUL, then we are interested in bounding the ratio $\gamma(E)$ as defined below:

$$\gamma(E) \coloneqq \frac{L(\mathbf{y}^l, \mathbf{q}^l)}{L(\mathbf{y}^*, \mathbf{q}^*)} \quad (15)$$

Note that bounding $\gamma(E)$ is not straightforward: When D is close to α, the optimal solution is to not store the content in any of the caches. When $D \gg \alpha$, MIP-a is similar to set cover. In the next section, we describe a way to transform E into another simplified instance which enables us to bound $\gamma(E)$.

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 Equations 13 and 15).

4.3.1 Bounding $\gamma(E)$. Our aim in the following discussion is to bound $\gamma(E)$ given a problem instance E. Towards this, we will simplify E into another instance E_1 while maintaining $\gamma(E_1) = \gamma(E)$. Recall that we are considering only tight solutions and $T = 1$ without loss of generality.

LEMMA 4.5. *Every MIP-a instance E (with multicast server mode) can be transformed into another instance E_1 with $\gamma(E_1) = \gamma(E)$ with the following additional properties:*

(a) If in E the demand of a user i is not locally satisfied in either $(\mathbf{y}, \mathbf{q})$ or $(\mathbf{y}^*, \mathbf{q}^*)$ then in E_1, i is associated with at most one cache randomly chosen from $H(i)$ from E.

(b) If in E the demand of a user i gets locally satisfied in $(\mathbf{y}, \mathbf{q})$ but not in $(\mathbf{y}^*, \mathbf{q}^*)$ then in E_1 user i is associated with exactly one cache from $H(i)$ from E, namely, the cache through which LIN-MUL routes its demand.

(c) If in E the demand of a user i gets locally satisfied in $(\mathbf{y}^*, \mathbf{q}^*)$ but not in $(\mathbf{y}, \mathbf{q})$ 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 in E the demand of a user i gets locally satisfied in both $(\mathbf{y}^*, \mathbf{q}^*)$ and $(\mathbf{y}, \mathbf{q})$, then in E_1 user i is associated with at most two caches as follows: If the cache through which LIN-MUL routes its requests also stores the content in $(\mathbf{y}^*, \mathbf{q}^*)$ then i is associated with only this single cache. Otherwise it is associated with two caches, namely, the

cache through which LIN-MUL routes its requests and any randomly chosen cache from $H(i)$ from E such that it stores the content in $(\mathbf{y}^*, \mathbf{q}^*)$.

For further illustration as to how E_1 looks like, please see Figure 2 and its description. We now state the main theorem of this section (where $g(x) := \frac{1}{1-\frac{1}{x}}$).

THEOREM 4.6. *For an instance E, define $\delta, \epsilon \geq 1$ such that $\delta := \frac{D}{L(\mathbf{y}^*,\mathbf{q}^*)}$ and $\epsilon := \frac{D}{\alpha|F(\mathbf{y}*)|}$. Let $k = \max_i\{k_i\}$. Then,*

$$\gamma(E) \leq \min\left\{\delta, \epsilon, \left(k + 1 + \frac{1}{|F(\mathbf{y}^*)|}\frac{\log g(\delta)}{\log g(|F(\mathbf{y}^*)|\epsilon)}\right)\right\}$$

Before we go to interpreting the theorem, we would like to make an interesting observation. The third term in the above theorem can be made much tighter as follows: Since the server transmission mode is multicast, for the download cost, the probability that matters that at least one user requests the content in a given slot. Thus if there are several users with same user-cache association graph 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 individual user of the aggregated users requests the content. Let us divide the users into equivalence classes where in an equivalence class every user has the same user-cache association graph. We can replace each equivalence class by a single aggregated user as described above. Thus k in the bound in Theorem 4.6 can in fact be replaced by the maximum number of equivalence classes the users of a cache can fall into, which could be much smaller than $\max_i\{k_i\}$.

The proof of Theorem 4.6 can be found in [32]. We give a brief intuitive interpretation of the result. First, if D is very close to α then $\gamma(E)$ will be small since the maximum cost of any solution is D and minimum cost of any solution which stores the content in at least one cache is α. On the other hand if D is substantially larger than α then the solutions tend to look like a set cover (i.e., storing the content in minimum number of caches to satisfy maximum number of users) in which the greedy algorithm gives a ratio of k. So let us assume that D is only fairly larger than α. For the following interpretation, we will assume that the network consists of a fairly large number of caches.

Consider the scenario when δ is small and $(\mathbf{y}^*, \mathbf{q}^*)$ is close to D, e.g., this happens when the network is mostly comprised of a large number of caches each having a small nearly-independent user base. Here, there is high aggregate demand in a slot. In this case, it is not beneficial for nearly any cache to store the content (due to their small individual user base) and instead they rely on multicast to satisfy high aggregate demand in each slot. This holds for both $(\mathbf{y}^*, \mathbf{q}^*)$ and $(\mathbf{y}, \mathbf{q})$, thus $\gamma(E)$ tends to be small.

Now consider the scenario with a large number of caches having sufficiently large and fairly non-overlapping user bases with high demands. Now many caches store the content in the optimal solution and $\alpha|F(\mathbf{y}^*)|$ will be close to D. This implies small ϵ and thus $\gamma(E)$ will also be small in such a situation. Now consider the scenario when the caches have fairly overlapping user bases with high demands. In this case, a small number of caches will end up storing the content in the optimal solution. At the end, if the multicast download cost does not form a significant part of the solution, LIN-MUL performs similar to applying the greedy algorithm for set cover (i.e., trying to minimize the number of caches storing content) thus $\gamma(E)$ is close to $k + 1$ (see [32]for a detailed explanation). However, if in this case multicast download cost forms a significant portion of the solution then the performance of the greedy algorithm could be limited by its attempt to approximate the non-linear decreasing download cost by myopically selecting the caches. Since the function is non-linear, greedy algorithms are not expected to perform well in this situation.

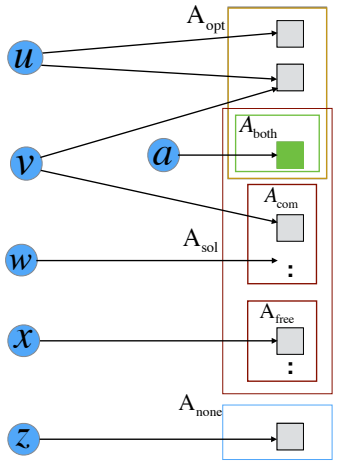

Figure 2: As a consequence of Lemma 4.5, a given instance E gets transformed to E_1 which looks like above (For the extra notation, see the discussion after the proof of Lemma 4.5 in [32]). Users whose demands do not get satisfied in either $(\mathbf{y}^*, \mathbf{q}^*)$ or $(\mathbf{y}, \mathbf{q})$ in E look similar to z (Lemma 4.5(a)). Users whose demands get satisfied only in $(\mathbf{y}, \mathbf{q})$ look similar to either w or z (Lemma 4.5(b)). Users whose demands get satisfied only in $(\mathbf{y}^*, \mathbf{q}^*)$ look similar to u (Lemma 4.5(c)). Users whose demands get satisfied in both $(\mathbf{y}^*, \mathbf{q}^*)$ and $(\mathbf{y}, \mathbf{q})$ look similar to v or a (Lemma 4.5.(d).

4.4 Unicast download

In the unicast server transmission mode, intuitively there is a higher pressure on the system to minimize the download cost even when D is close to α (unlike the case of multicast) since the unicast download cost is additive with respect to the demands which are not locally satisfied. We can form an equivalent greedy algorithm of LIN-MUL with respect to the unicast objective in Equation 14 and can also define $\gamma(E)$. We can also show that Lemma 4.5 holds. Furthermore,

LEMMA 4.7. *For unicast server mode, starting from the instance E_1, we can construct another instance E_2 such that $\mathrm{A}_{\mathrm{free}}$ and $\mathrm{A}_{\mathrm{none}}$ are empty and $\gamma(E) \leq \gamma(E_3)$.*

THEOREM 4.8. *For an instance E of MIP-a with unicast, if k is the maximum number of users a cache can reach then $\gamma(E) \leq k + 1$.*

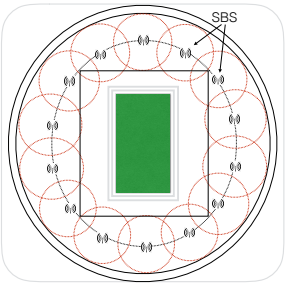

Figure 3: Stadium consisting of 14 SBSs providing overlapping coverage to users. The requests that result in a cache miss at the SBS are served by a MBS (omitted here).

5 CAPACITY CONSTRAINED CACHES

Recall that, Sections 4 and 4.4 assumed large capacity caches. For capacity-constrained caches, the following Algorithm 2 is a natural extension of LIN-MUL. Now instead of going through a sequence of tight solutions generated by iterating on only 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 (5).
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 $\mathbf{y}$.

6 PERFORMANCE EVALUATION

In this section, we numerically evaluate the performance of our algorithms on a real dataset [33] obtained from a sporting event with thousand attendees covered by a Mobile Base Station (MBS) and several Small Cell Base Station (SBSs).

6.1 Evaluations setup

Our evaluations are based on the dataset from the Superbowl event, held at the New Orleans Superdome, in 2013. The stadium

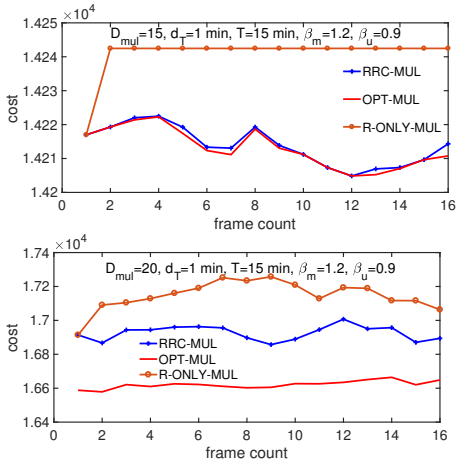

Figure 4: Impact of increasing D_{mul} from 15 to 20.

consists of a Macro Base Station (MBS) providing cellular coverage to the viewers (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 [7] 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 $\beta_m = 1.2$ [7]. 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}.

We divide the four hour time duration – i.e. 240 minutes – 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 $d_T = 1$. 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 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 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. Now we describe the results of our performance evaluations for various storage and routing policies for large as well as capacity-constrained caches.

For evaluations, we study the problem across content caching schemes, content routing policies, and the modes of content transmission, and draw comparisons across various policies in terms of how they differ across these regimes. Although we started with retention time taking integral values in $[0, T]$, our caching decisions with linear storage cost become binary (see Theorem 4.2). Thus, we only need to decide whether to cache a content for duration T or to not cache it at all. We refer to the routing as *greedy* if all the requests from a user are routed to a single cache containing the content. If no cache has the content, then the requests are routed arbitrarily to any cache the user is associated with. We study both multicast (MUL) and unicast (UNI) modes of transmission by the server.

6.2 Algorithms on a large cache

In this part, we assume that all the caches have a large capacity. We study the performance trade-offs by varying various parameters for the following policies:

(1) *Retention-Routing Caching (RRC)*: The joint caching-routing decisions are made by solving MIP-a at the beginning of every frame by using LIN-MUL (see Algorithm 1) for the multicast case, and a similar algorithm LIN-UNI for the unicast case.
(2) *Routing-only Caching (R-only)*: While RRC solved the joint caching-routing optimization in the beginning of *every* frame, here caching decision is only made during the first frame. In all the subsequent frames, even though some of the users have moved, we only compute the optimal routing (by forming a tight solution) but we do not change the storage. This case is useful for checking how much a small amount of mobility can impact the cost of the solutions on short term basis.
(3) *Optimal Caching (OPT)*: Here we compute the optimal solution, i.e. the optimal retentions and routing by brute force. We exploit the properties of the optimal solution proved in Theorem 4.2, by virtue of which we only need to consider 2^N tight solutions for every possibility of content storage assignment to the N caches.

We now study the impact of various parameters used in our evaluations. We first increase D w.r.t. α and observe the following: From Figure 4, the solutions computed by LIN-MUL (i.e., RRC-MUL) perform nearly as well as the optimal solution. Also, as D increases, the cost of the optimal solution as well as the solution computed by LIN-MUL (i.e., RRC-MUL) increases. This is natural to expect as increasing D implies higher download cost for the user demands which are not locally satisfied. Also, the solutions computed by re-computing only the routing (R-ONLY-MUL) naturally perform worse than the RRC-MUL. Similar conclusions apply for the unicast case (see [32]).

To study the impact of the length of frame duration on the cost, we consider a frame of duration $T = 20$ minutes with slot duration $d_T = 2$ minutes, thus having a total of 12 frames. We plot the results in Figure 5 for the multicast case to contrast the results against those obtained with $T = 15$ min and $d_T = 1$ minute from Figures 4. A similar figure contrasting the results from for the unicast case can be found in [32]. We observe that a larger frame duration reduces the cost, which is natural since users are moving fewer number of times during the duration of interest.

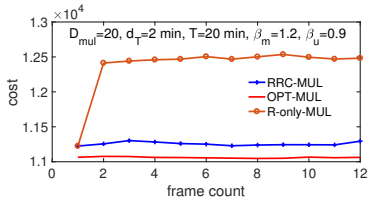

Figure 5: Impact of increasing frame length to $T = 20$ min and slot length to $d_T = 2$ min. for the multicast case.

Binning user movements into large frame durations could be lossy as it may fail to capture the real user movements which can potentially occur more often. In general, the problem of choosing the optimal values of T is a trade-off between computation and performance: Smaller values of T would model user movement very well with high granularity, however, at the cost of high computation overhead per frame and vice versa. A large d_T indicates an increased delay tolerance in users (for getting the requested content served) whereas a small value of d_T guarantees a fast content delivery to users. Thus, choosing an appropriate value of d_T requires domain knowledge about user and wireless network characteristics.

We vary the user demand spreading ZipF coefficient, β_u, from 0.1 to 3.0. A higher β_u models the skewness in user-demands, implying that only a few users request more contents compared to most of the other users. A $\beta_u = 0$ means that content demands are distributed uniformly across users. We notice similar cost trends across the policies as before (for both multicast and unicast) for every β_u with the difference that as β_u increases the cost incurred for each policy reduces. This is reasonable since, with a high user skewness, transmitting and storing only a few files can potentially satisfy demands from a large user pool.

6.3 Algorithms on a finite cache

We next consider the cache-capacitated variant of the problem for the following policies:

(1) *Greedy heuristic for PRRO*: We consider the heuristic described in Algorithm 2 to assign content to caches, and refer to it as PRRO-heuristic. Note that PRRO, with overlapping cache coverage, can offer multi-path routing option to users, under specific wireless technologies.
(2) *Greedy Multicast Aware Caching (GMAC)*: GMAC 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, except for the difference 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. Many prior works, such as [2, 7] use this model.

Benefit of multiple paths for routing. We evaluate the cost of the solutions using the greedy heuristics for PRRO and for GMAC. Note that an increase in cache capacity while making joint

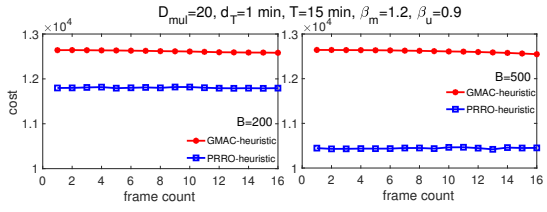

Figure 6: Impact of increasing B relative to M.

storage-routing decision makes multipath routing significantly more advantageous than simple single-path GMAC strategy. For our experiments (see Figure 6, we observed that when $B = 20\%$ of the total number of contents (i.e. $B = 200$ and $M = 1000$) we get only 7% cost savings with PRRO heuristic, but for $B = 50\%$ of M (i.e., 500 contents), we get substantial 22% cost savings.

7 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 showed 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 evaluations, including real data sets, 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 on work supported in part by the National Science Foundation under Grant No. 1422153, and by a Tekes FiDiPro award to A. Abouzeid from University of Oulu, Finland.

REFERENCES

[1] I-Hong Hou 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*, pages 291–300. ACM, 2016.

[2] Samta Shukla and Alhussein A. Abouzeid. Proactive Retention Aware Caching. *IEEE INFOCOM*, 2017.

[3] Pol Blasco and Deniz Gündüz. Learning-Based Optimization of Cache Content in a Small Cell Base Station. *CoRR, 2014*.

[4] BN Bharath, KG Nagananda, and H Vincent Poor. A Learning-based Approach to Caching in Heterogenous Small Cell Networks. *IEEE Transactions on Communications, 2016*.

[5] Yeon-sup Lim, Yung-Chih Chen, Erich M Nahum, Don Towsley, and Richard J Gibbens. Improving Energy Efficiency of MPTCP for Mobile Devices. *arXiv preprint arXiv:1406.4463*, 2014.

[6] Ezzeldin Hamed, Hariharan Rahul, Mohammed A. Abdelghany, and Dina Katabi. Real-time Distributed MIMO Systems. In *ACM SIGCOMM*, 2016.

[7] Konstantinos Poularakis, George Iosifidis, Vasilis Sourlas, and Leandros Tassiulas. Exploiting Caching and Multicast for 5G Wireless Networks. *IEEE Transactions on Wireless Communications*, 2016.

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

[9] Konstantinos Poularakis, George Iosifidis, and Leandros Tassiulas. Approximation Algorithms for Mobile Data Caching in Small Cell Networks. *IEEE Transactions on Communications*, 2014.

[10] Mostafa Dehghan et al. On the Complexity of Optimal Routing and Content Caching in Heterogeneous Networks. *CoRR, 2015*.

[11] Ejder Baştuğ et al. Big Data Meets Telcos: A Proactive Caching Perspective. 2015.

[12] John Tadrous, Atilla Eryilmaz, and Hesham El Gamal. Proactive Resource Allocation: Harnessing the Diversity and Multicast Gains. *CoRR*, 2011.

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

[14] 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.

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

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

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

[18] N. Choungmo Fofack, M. Dehghan, D. Towsley, M. Badov, and D. L. Goeckel. On the Performance of General Cache Networks. In *ACM VALUETOOLS, 2014*.

[19] Nicaise Éric Choungmo Fofack. *On Models for Performance Analysis of a Core Cache Network and Power Save of a Wireless Access Network*. PhD thesis, Université Nice Sophia Antipolis, 2014.

[20] Ran Bi, Yingshu Li, and Xu Zheng. An Optimal Content Caching Framework for Utility Maximization. *Tsinghua Science and Technology*, 2016.

[21] Richard TB Ma and Don Towsley. Cashing in on Caching: On-demand Contract Design with Linear Pricing. *CoNext, December*, 2015.

[22] 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.

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

[24] Samta Shukla and Alhussein A Abouzeid. On Designing Optimal Memory Damag Aware Caching Policies for Content-Centric Networks. In *IEEE WiOpt, 2016*.

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

[26] B. Zhou, Y. Cui, and M. Tao. Stochastic Content-Centric Multicast Scheduling for Cache-Enabled Heterogeneous Cellular Networks. *IEEE Transactions on Wireless Communications*, 2016.

[27] Mathieu Leconte, Georgios Paschos, Lazaros Gkatzikis, Moez Draief, Spyridon Vassilaras, and Symeon Chouvardas. Placing Dynamic Content in Caches with Small Population. *CoRR*, 2016.

[28] Stefan Dernbach, Nina Taft, Jim Kurose, Udi Weinsberg, Christophe Diot, and Azin Ashkan. Cache Content-Selection Policies for Streaming Video Services. *IEEE INFOCOM*, 2016.

[29] Luca Stornaiuolo and Paolo 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.

[30] Samta Shukla and Alhussein A. Abouzeid. Optimal Device-Aware Caching. *IEEE Transaction on Mobile Computing*, 2016.

[31] Özgü Alay, Thanasis Korakis, Yao Wang, Elza Erkip, and Shivendra S Panwar. Layered Wireless Video Multicast Using Relays. *IEEE Transactions on Circuits and Systems for Video Technology*, 2010.

[32] Technical-Report. Hold'em Caching: Proactive Retention-Aware Caching with Multi-path Routing for Wireless Edge Networks. https://www.dropbox.com/s/9dukx0wmnckiv03/mainmobihoc17.pdf?dl=0, 2015.

[33] Jeffrey Erman and Kadangode K Ramakrishnan. Understanding the Supersized Traffic of the Super Bowl. In *Internet Measurement conference, ACM*, 2013.