Elastic FemtoCaching: Scale, Cache, and Route

Jeongho Kwak, *Member, IEEE,* Georgios Paschos, *Member, IEEE,* and George Iosifidis, *Member, IEEE*

***Abstract*—The advent of elastic Content Delivery Networks (CDNs) enable Content Providers (CPs) to lease cache capacity on demand and at different cloud and edge locations in order to enhance the quality of their services. This paper addresses key challenges in this context, namely how to invest an available budget in cache space in order to match spatio-temporal fluctuations of demand, wireless environment and storage prices. Specifically, we jointly consider *dynamic cache rental*, *content placement*, and *request-cache association* in wireless scenarios in order to provide just-in-time CDN services. The goal is to maximize the an aggregate utility metric for the CP that captures both service benefits due to caching and fairness in servicing different end users. We leverage the Lyapunov drift-minus-benefit technique and Jensen's inequality to transform our infinite horizon problem into hour-by-hour subproblems which can be solved without knowledge of future file popularity and transmission rates. For the case of non-overlapping small cells, we provide an optimal subproblem solution. However, in the general overlapping case, the subproblem becomes a mixed integer non-linear program (MINLP). In this case, we employ a randomized cache lease method to derive a scalable solution. We show that the proposed algorithm guarantees the theoretical performance bound by exploiting the submodularity property of the objective function and pick-and-compare property of the randomized cache lease method. Finally, via real dataset driven simulations, we find that the proposed algorithm achieves 154% utility compared to similar static cache storage-based algorithms in a representative urban topology.**

***Index Terms*—Elastic CDN; file caching; area-BS association; cache rental budget; Lyapunov drift-plus-penalty; submodularity; randomized cache lease method**

I. Introduction

The seminal paper of *femtocaching* [2] introduced a novel wireless edge caching architecture and proposed an efficient algorithm for proactively caching popular content files. However, a key limitation of this idea is that it considers a static deployment of caches and their long-term population with content. In practice, the dynamic (re-)scaling of the cache capacity and the frequent refreshing of their contents is imperative for coping with the time-varying file popularity and user demand intensity. In this paper, we propose the *elastic femtocaching* model which introduces a new wireless edge caching system for deciding how to scale the caches, which files to cache in each of them, and how to route the content to users.

The femtocaching architecture includes a set of edge caches deployed at small base stations (SBSs) that underlay a macro base station (MBS) in a heterogeneous wireless network. These caches are filled during off-peak hours, e.g., overnight, with popular files, which are then delivered to nearby users when demand increases. This mitigates the network congestion, as it economizes the bottleneck MBS wireless capacity and reduces the utilization of the expensive SBS backhaul links. At the same time, femtocaching improves the user experience by replacing the long-range MBS transmissions with the energy-prudent and fast SBSs-users wireless links.

The main assumption of femtocaching is that the caches have a fixed, and cheap, storage capacity and that their population with files is realized in a coarse time scale, e.g., once per day or week. However, in practice file demand changes quite fast, as users might move from one location to another, and content popularity at the various online content platforms peaks only for a few hours [3]. Moreover, installing and maintaining storage units at the edge induces operating expenditures that can render this model unsustainable. Under these conditions, the static femtocaching model can be both performance-inefficient and unnecessarily costly. This becomes particularly important today where we see the proliferation of small service (or, content) providers that have volatile demand and hence cannot afford buying or leasing large storage capacity. For these business entities it is essential to have access to elastic caching infrastructures, such as uCDN and ElastiCache, two real market elastic CDN solutions deployed by Huawei and Amazon Web Services, respectively. The importance of such flexible storage deployments is best manifested by the proliferation of solutions such as *Huawei uCDN* or *AWS ElastiCache* that allow *dynamic cache scaling*.

Motivated by the above, we revisit this fundamental caching model and propose a novel *elastic femtocaching* architecture. In this system, the caches are re-scaled and the stored files are updated in a finer time-scale in order to adapt to user needs. Moreover, the association of users to SBSs aims not simply to maximize the caching benefit, but to balance the caching benefits for the different users. These decisions are updated dynamically as new information about the expected demand becomes available, and in a way that long-term monetary budget and performance criteria are satisfied. Therefore, our goal is to develop a rigorous analytical framework for addressing these decision trade-offs in a systematic and provably-optimal fashion.

We introduce a general system model that can capture

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea Government(MSIT) (No. 2019R1F1A1062291). This work was supported by the DGIST Start-up Fund Program of the Ministry of Science and ICT(2021010006). G. Iosifidis acknowledges supprt by Science Foundations Ireland under Grant 17/CDA/4760 and by the European Commission through Grant No. 101017109 (DAEMON). The ideas and opinions expressed in this paper are of the authors, and do not represent the official position of Amazon Inc. Parts of this paper were presented at the Proceedings of *WiOpt* conference, May 2018 [1].

J. Kwak is with DGIST (Daegu Gyeongbuk Institute of Science and Technology), Daegu, 42988, South Korea (email: jeongho.kwak@dgist.ac.kr).

G. Paschos is with the Amazon Inc., 2540, Luxembourg (email: gpasxos@gmail.com).

G. Iosifidis is with Delft University of Technology, 2600, Delft, Netherlands (email: g.iosifidis@tudelft.nl).

different application scenarios. We assume that the network[1] has a certain average budget to spend, over time, for deploying the SBSs caches. The deployment cost might capture operating expenditures (e.g., energy consumed by the servers) or the leasing price when the network does not own the infrastructure, e.g., in solutions such as [4]. The system operates in a time-slotted fashion, where each slot has a duration of a few hours or less, based on the scenario. At the beginning of each slot, the network obtains information about the expected demand, and the servicing delay of the SBSs.[2]

Using this minimal-assumptions model, we formulate the elastic femtocaching problem where decisions for *(i)* cache scaling; *(ii)* file caching; and *(iii)* user-SBS association, are taken in each time slot. The objective is to satisfy the user requests with the maximum possible caching benefit, while respecting certain fairness among different subareas.[3] Our approach ensures that users experiencing unfavorable wireless conditions with the SBSs will not be exclusively served by the MBS (over long-distance, high-delay links), hence it achieves an even distribution of the edge-caching benefits. This idea is in line with the fairness criteria that have been extensively applied in wired networks (e.g., in TCP mechanisms [5]), yet have been hitherto ignored in femtocaching.

The formulated optimization problem is NP-complete, as it extends the standard femtocaching problem, and hence it is solved with an approximate solution. In each slot we use a low-complexity caching and association intra-slot policy that attains a feasible, but possibly suboptimal, operation point. Our approach leverages a lightweight greedy algorithm and exploits the submodular property of the objective function. Across different time-slots, we employ a Lyapunov-based control policy that tracks the budget over/under-spending and the QoS criteria violation as new information about demand and transmission delays become available. These signals are then used to modulate the decisions of the intra-slot policy, so as to achieve, asymptotically, the desired operation point. Our contributions can be summarized as follows:

1) We propose a wireless caching architecture, *elastic femtocaching* in order to account for time-varying file popularity and user-demand intensity.
2) We formulate the cache-scaling, content-caching, and user-association mathematical program, with a fair caching benefit maximizing objective criterion.
3) We propose a set of algorithms for solving this problem, combining a greedy algorithm, using the submodularity of objective function for the intra-slot decisions, with a Lyapunov-based control policy [6] and a randomized pick-and-compare [7] for the inter-slot decisions. The resulting algorithms achieve asymptotically a provably-optimal network operation point.

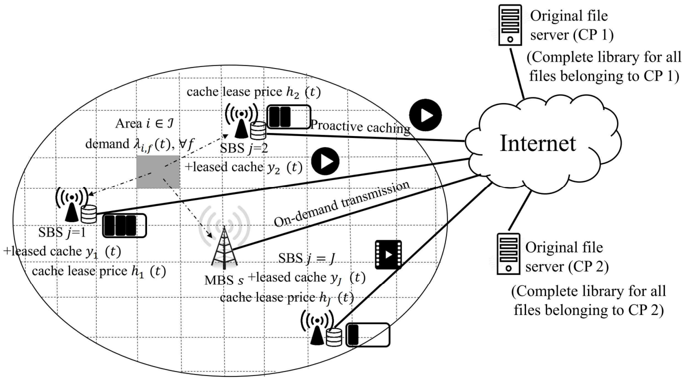

Original file
server (CP 1)
(Complete library for all
files belonging to CP 1)
cache lease price h_2 (t)
Area $i \in \mathcal{I}$
demand $\lambda_{i,f}(t)$, $\forall f$
SBS j=2
+leased cache y_2 (t)
Proactive caching
Internet
On-demand transmission
SBS j=1
+leased cache y_1 (t)
cache lease price h_1 (t)
MBS s
SBS $j = J$
+leased cache y_J (t)
cache lease price h_J (t)
Original file
server (CP 2)
(Complete library for all
files belonging to CP 2)

Fig. 1: Overview of an elastic femtocaching system.

4) We evaluate our algorithms using real datasets for demand and various SBSs deployment scenarios. Especially, we find that our *elastic* and *joint* policy attains 154% higher performance than the static femtocaching model in a typical urban network deployment scenario.

Paper Organization. The remaining of this paper is organized as follows. We discuss the related work in Sec. II and introduce the system model and the elastic femtocaching problem in Sec. III. We propose a Lyapunov-based dynamic policy in Sec. IV, and in Sec. V we discuss two different algorithms. We evaluate the proposed policies in Sec. VI and conclude in Sec. VII. All proofs can be found in the Appendix, unless otherwise stated.

II. Related Work

Wireless edge caching. The idea of wireless edge caching was introduced in [2] and further extended by several follow-up works. For example, Abedini *et al.* [8] focused on stabilizing queues of pending requests by managing jointly the link bandwidth and storage of the SBSs. On the other hand, [9] minimizes dynamically a cost criterion through both load-balancing and content replication when the SBSs have hard storage and soft link capacity constraints. The idea of creating a femtocaching network through leased caches was proposed in [10], that designed a low-complexity solution algorithm based on Lagrange relaxation. Wu *et al.* [11] aimed at minimizing long-term energy consumption while guaranteeing short-term user Quality of Service (QoS). They assumed coded caching[4] whereas *we study more challenging problem with discrete caching variables, and hence have the more challenging model of discrete caching variables.* Moreover, Ryu *et al.* [12] designed cooperative caching algorithms where multiple BSs optimize jointly the content placement without knowing the file popularity. They used a mixed-time scale model where in the long scale they retrieve files from the core network, and in the short time scale share the files among the BSs. *However, they did not consider elastic cache scaling, which is a core idea in our framework.*

[1] We use the term *network* to refer to any entity in charge for making the elastic femtocaching decisions; this can be the actual network operator as in the standard femtocaching model or, for instance, a mobile-CDN, the content provider, and so on.

[2] Such information can rely on simple statistics of the previous slot or become available through sophisticated prediction methods.

[3] Namely, instead of maximizing the aggregate caching benefits, we aim to balance the caching benefits across all users.

[4] In coded caching, a file can be split into several parts that can be stored in different caches.

TABLE I: Summary of the notations.

Notation	Definition	Notation	Definition
$i \in \mathcal{I}$	subarea index	$h_j(t)$	price to lease cache storage per unit bit for j and t
$j \in \mathcal{J}$	small base station (SBS) index	$d_{ij}(t)$	average delay for serving subarea i by SBS j during t
s	macro base station (MBS) index	$d_{is}(t)$	average delay for serving subarea i by the remote server during t
$f \in \mathcal{F}$	file index	$x_{ij,f}(t)$	association probability for i, j, f, and t
B_{avg}	average budget constraint	$y_j(t)$	leased cache space at SBS j during t
$\lambda_{i,f}(t)$	demand profile for i, f and t	$z_{j,f}(t)$	file caching indicator for f, j and t
t	hour index (time slot)	$\gamma_i(t)$	auxiliary variable of subarea i at time slot t
b	size of a video file		

Dynamic caching policies. The original femtocaching model [2] presumes static content popularity and proposes a one-off proactive caching policy. Some recent studies in proactive caching, including our work, dropped the assumption of static popularity. Asheralieva *et al.* [13] exploited Lyapunov optimization for proactive content caching and delivery in cellular and device-to-device networks, aiming to minimize the time-average network cost. Similarly, Paschos *et al.* [14] addressed the issue of non-stationary content popularity using an online learning approach in the design of the routing and caching policies. *Nevertheless, the above works overlooked the possibility of cache scaling.*

On the other hand, reactive policies, such as LRU (Least Recently Used) and LFU (Least Frequently Used) [15], make dynamic caching decisions upon the arrival of each request. These policies have been originally designed for single (or, independent) caches, and have been later extended to caching networks. For instance, Giovanidis *et al.* [16] proposed a spatial multi-LRU version where each request is routed via the closest base station that has the file. This solution improves the caching hit via cooperative caching *whereas wireless latency was not taken into account.* Similarly, Leonardi *et al.* [17] proposed a q-LRU algorithm where the caching update happens depending on the cached status in multiple BSs. Chen *et al.* [18] addressed a stochastic cooperative caching in several BSs under assumption of coded and time-to-live (TTL) caching aiming at reducing content download time. Similar to our work, Carra *et al.* [[19] addressed the dynamic cache resource scaling aiming to simultaneously minimize the storage and backhaul cost. *However, unlike our work, they considered a reactive caching policy, namely time-to-live (TTL) caching, single cache and non-fairness criteria.* Moreover, Dehghan *et al.* [15] considered a utility-based objective function where the utility captures fairness between different files. They proposed utility-driven LRU and LFU algorithms aiming at maximizing the sum of utilities over all files without consideration of cache scaling. *Besides, our utility function is designed to achieve user - not file - fairness.*

Content Caching using machine learning framework. Past content caching works using machine learning, e.g., collaborative filtering, focused on accurate estimation of future content popularity [20], [21]. Recently, several studies exploited (deep) reinforcement learning (RL) to optimize the operation of caching networks. For example, Xiong *et al.* [22] addressed content caching in broadcasting systems using deep reinforcement learning. Moreover, Sadeghi *et al.* [23] used RL as a contents caching solution in a unicast system where their storage price might change with time. In their previous work, Sadeghi *et al.* [24] considered a hierarchical cloud-edge caching model where the cloud stores files according to global file popularity and the edge stores files according to local file popularity. They modeled the spatio-temporal popularity variations using a Markov chain model, and solved them with RL. Finally, Somuyiwa *et al.* [25] proposed a mobile proactive caching scheme, using again RL, where the caches are deployed at the mobile users' equipment, not at edge servers as in our model.

III. System Model and Problem Formulation

A. System Model

We consider an elastic caching network with a macro base station (MBS), denoted with s, and a set $\mathcal{J}$ of small base stations (SBSs). The set $\mathcal{J} \cup \{s\}$ of all BSs provide coverage to a geographic area and serve user requests for a catalog $\mathcal{F}$ of content files, each with size b, see Fig. 1. We partition the geographical area into $\mathcal{I}$ non-overlapping subareas where each subarea might include one or more users who share the same network characteristics (propagation delay, shadowing effects, and so on) and use $\mathcal{J}_i \subseteq \mathcal{J}$ to denote the subset of SBSs that are *reachable* by each subarea $i \in \mathcal{I}$.[5] while the MBS is reachable from all subareas. Each SBS offers *storage for lease*, which can be used to cache files so as to facilitate their delivery to the users[6].

The system operation is time slotted, where each slot represents, for instance, an hour. For each file $f \in \mathcal{F}$ in the catalog, we denote with $\lambda_{i,f}(t)$ the average number of requests for file f emanating from subarea i during slot t, and is generated by an i.i.d. stochastic process $\{\lambda_{i,f}(t)\}_t$. We also introduce the demand vector $\boldsymbol{\lambda}(t) = (\lambda_{i,f}(t) : i \in \mathcal{I}, f \in \mathcal{F})$, which is indicative of the file popularity in time and space, and therefore crucial for adjusting caching decisions. When a user requests a file, there is an associated download delay $d_{ij}(t), j \in \mathcal{J}_i \cup \{s\}$, which depends on the subarea i, where the user is located, and on whether the file is cached at SBS j

[5] The subarea model is general enough and allows the subareas to be defined very small so as to have only one user in practice.

[6] We assume that SBS edge storage and an original file server are connected with high-capacity links (e.g., optical lines). Then, our system will retrieve the updated cached files via the fast dedicated link without significant fetching costs.

or not.[7] When the file is not found in any reachable SBS, the origin server that stores the entire catalog is contacted through the MBS to obtain the file (Fig. 1), and although this ensures delivery of every file, the corresponding download delay $d_{is}(t)$ is generally large, i.e., we naturally assume that $d_{is}(t) > d_{ij}(t)$ for every slot and $i \in \mathcal{I}$, $j \in \mathcal{J}$. Hence, the perceived service quality (QoS) is improved whenever the file is retrieved from a nearby SBS cache, instead of the MBS. We denote with $\boldsymbol{d}(t) = (d_{ij}(t), d_{is}(t), \forall i \in \mathcal{I}, j \in \mathcal{J})$ the vector of all delays in slot t.

We assume that there are costs for deploying storage at the SBSs. Namely, the edge storage is leased at a time-fluctuating unit price $h_j(t)$ that can be potentially different for each SBS j. We define the respective vector $\boldsymbol{h}(t) = (h_j(t), j \in \mathcal{J})$ which is extrinsic to our system. This volatility of storage leasing prices can arise for various reasons. For instance, it can be attributed to electricity price fluctuations [27], or due to the volatility of a spot storage market that the operator uses to lease such resources.[8] This creates the need for a careful leasing strategy. To that end, we introduce the investment variables $y_j(t) \geq 0$ to denote the amount of j^{th} SBS storage that is leased for caching in slot t. These decisions are subject to an economic constraint. Specifically, we have in mind an average budget B_{avg} (dollars/hour), which must be satisfied over a long time horizon:

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{j\in\mathcal{J}} y_j(t) h_j(t) \leq B_{avg}, \qquad (1)$$

where the term $\sum_{j\in\mathcal{J}} y_j(t)h_j(t)$ represents the total investment in slot t.

In any case, measurement errors may as well impact the performance of such system. If they have a stationary zero-mean distribution, such that their effect does not impact the solution of the static problem, then they will not affect the performance of our dynamic algorithm. If, however, they do affect the static solution, then they will also impact our dynamic algorithm. Clearly, three types of measurements, i.e., file popularity, average delay and cache price jointly affect the decisions of cache scaling, content caching and routing. However, their solution would depend on the relative measurement between different files.

In this context, our goal is to address the following content provider's (CP's) question: *what is the cache scaling strategy that optimizes the average caching benefits while respecting the long-term budget constraint?* Answering this question is very challenging for the following reasons: *(i)* the CP does not know the future spatio-temporal profile of the demand, nor the storage prices that might change substantially in short time intervals; and *(ii)* the benefits of caching at an SBS change over time, and therefore even deciding the distribution of the hourly budget to leasing different caches is highly non-trivial.

To determine the average delay experienced within each slot, we must describe carefully how each file is delivered. We first introduce two more sets of variables: *(i)* file placement variable $z_{j,f}(t) \in \{0,1\}$ which takes value 1 iff file f is cached at SBS j in slot t, and *(ii)* the demand association variable $x_{ij,f}(t) \in [0,1]$ which denotes the fraction of requests for file f from location i that is served by SBS j, during slot t. Hence, the hourly end-to-end caching benefit from edge caching for each subarea $i \in \mathcal{I}$ can be expressed as:

$$D_i(t) = \sum_{j\in\mathcal{J}_i} (d_{is}(t) - d_{ij}(t)) \sum_{f\in\mathcal{F}} x_{ij,f}(t)\lambda_{i,f}(t) z_{j,f}(t), \quad (2)$$

where $x_{ij,f}(t)\lambda_{i,f}(t)$ is i's demand fraction routed to SBS j, and $(d_{is}(t) - d_{ij}(t))x_{ij,f}(t)\lambda_{i,f}(t)$ is the corresponding caching benefits (delay reduction) which is realized if the file is cached at the SBS, i.e., $z_{j,f}(t) = 1$.

Finally, in each slot the system must satisfy the following constraints. First, the entire demand of each subarea $i \in \mathcal{I}$ is routed to some of the SBSs[9], hence it holds: $\sum_{j\in\mathcal{J}_i} x_{ij,f}(t) = 1, \forall i, f, t$. Routing to an unreachable SBS is not allowed: $x_{ij,f}(t) = 0$, $\forall f$, if $j \notin \mathcal{J}_i$. Also, the cached files should not exceed the leased capacity, i.e., $\sum_{f\in\mathcal{F}} z_{j,f}(t) \leq y_j(t)/b, \forall j, t$.[10] The notation is summarized in Table I.

B. Problem Formulation

Definition 1 (Femtocaching plan)**.** *An elastic cache plan for time slot t is a selection of variables $(y_j(t), z_{j,f}(t), x_{ij,f}(t))$ such that the instantaneous constraints are satisfied:*

$$\sum_{j\in\mathcal{J}_i} x_{ij,f}(t) = 1,\ \forall i, f, t,\ x_{ij,f}(t) = 0,\ \forall f, t, i \text{ if } j \notin \mathcal{J}_i, \quad (3)$$

$$\begin{aligned} &\sum_{f\in\mathcal{F}} z_{j,f}(t) \leq y_j(t)/b, \quad y_j(t) \geq 0,\ \forall j, t \\ &z_{j,f}(t) \in \{0,1\}, \forall j, f,\ x_{ij,f}(t) \in [0,1],\ \forall i, j, f. \end{aligned} \quad (4)$$

Definition 2 (Elastic femtocaching policy)**.** *A feasible elastic femtocaching policy π at every slot observes the system state $(\boldsymbol{\lambda}(t), \boldsymbol{d}(t), \boldsymbol{h}(t))$ and chooses a femtocaching plan such that the time average budget constraint* (1) *is satisfied. We denote with Π the set of all feasible elastic CDN strategies.*

In order to improve system performance, we are clearly interested to tune our elastic femtocaching policy towards obtaining large caching benefits. Using the definition of the instantaneous caching benefit in (2), we can define the time-average caching benefit using policy π as:

$$\overline{D}_i^{\pi} \triangleq \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} D_i(\boldsymbol{x}^{\pi}(t), \boldsymbol{z}^{\pi}(t); \boldsymbol{\lambda}(t), \boldsymbol{d}(t), \boldsymbol{h}(t)),$$

[7] In practice, wireless delay can be calculated by Shannon capacity formula with wireless parameters, e.g., RSSI, CSI and interference obtained in the previous time slot [26]. Otherwise, we can use the measurement-based delay in the previous time slot. Specifically, the simple method to measure average delay is to collect the measured delay in each subarea from all users received any files from one of SBSs in the previous time slot, and take average of it. Although there exist other estimation methods for average delay, they might have a tradeoff relationship between estimation overhead and estimation accuracy.

[8] For example, storage owners sell their unused storage, and hence the price is affected by temporal ebbs and flows of traffic and storage demand.

[9] In our model, all requests are routed towards an SBS, even if they are ultimately served by the MBS; in this case, we still require a dummy selection of the $x_{ij,f}(t)$ variables. Notice that this model is slightly different from femtocaching model [2] where the association variable includes an MBS.

[10] We assume that b is the same for all files for simplicity, but we can model a heterogeneous file size scenario by dividing the different sizes of files into the same size chunks.

where $\boldsymbol{x}^{\pi}(t), \boldsymbol{z}^{\pi}(t)$ are the caching and association decisions in slot t, under policy π. A reasonable objective is to maximize the time-average total caching benefit: $\sum_i \overline{D}_i^{\pi}$. However, to achieve a *fair* caching improvement throughout the entire geographical area, we employ a general α-fairness utility function [28][11]:

$$U_i(\bar{D}_i) = \begin{cases} (1-\alpha)^{-1}\bar{D}_i^{\,1-\alpha}, & \text{if } \alpha \geq 0,\ \alpha \neq 1, \\ \log(1+\bar{D}_i), & \text{if } \alpha = 1, \end{cases} \quad (5)$$

If $\alpha = 0$, our system only considers average caching benefit maximization without accounting for fairness across the different subareas; whereas for higher α values it forces more fair distribution of the caching benefits among users in different subareas. A representative function of this α-fairness utility model is the function $\sum_i \log(1+\overline{D}_i^{\pi})$. Hereinafter we will focus on this specific function, though we mention that our results hold true for any concave function.

In summary, we would like to address the CP's question *"What is the feasible elastic femtocaching policy that achieves the highest utility?"* This question can be addressed by solving the following problem:

$$Val(\mathbf{P}) = \sup_{\pi \in \Pi} \sum_{i \in \mathcal{I}} \log(1+\overline{D}_i^{\pi}).$$

Note that **(P)** is challenging for the following reasons: *(i)* Parameters for the objective such as future traffic demand $\lambda_{i,f}(t)$, future caching gains $d_{is}(t) - d_{ij}(t)$ and cache lease price $h_j(t)$ are unknown at the time the investment decisions $y_j(\tau)$ are taken ($\tau < t$). *(ii)* Due to the time average billing constraint, a large investment $y_j(\tau)$ reduces the available budget in future slots $t > \tau$, which can be problematic in combination with the unknown future costs $h_j(t)$, delays $d_{ij}(t), d_{is}(t)$ and traffic demand $\underline{\lambda_{i,f}(t)}$. *(iii)* Due to the nonlinearity of log function, it holds $\overline{\log(x_t)} \neq \log \overline{x_t}$, and hence the objective is not decomposable to individual time slot contributions.

C. Characterization of Achievable Performance

We characterize the performance region, denoted with $\mathcal{G}$, which contains all vectors $(\overline{D}_i^{\pi})$ of time-average caching benefits achievable by any feasible elastic femtocaching policy $\pi \in \Pi$. Once $\mathcal{G}$ is determined, $Val(\mathbf{P})$ is equivalently calculated by:

$$Val(\mathbf{P}) = \max_{(\overline{D}_i^{\pi}) \in \mathcal{G}} \sum_{i \in \mathcal{I}} \log(1+\overline{D}_i^{\pi}). \quad (6)$$

Some technical assumptions are needed about the exogenous random events. We assume that there are finite sets $\mathcal{D} = \{d_1, ..., d_{|\mathcal{D}|}\}$ (for delays), $\Lambda = \{\lambda_1, ..., \lambda_{|\Lambda|}\}$ (for traffic demand intensities), and $\mathcal{H} = \{h_1, ..., h_{|\mathcal{H}|}\}$ (for storage costs) from which a value $(\boldsymbol{\lambda}, \boldsymbol{d}, \boldsymbol{h})$ is drawn at each slot according to an unknown distribution $p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}}$. The assumption that these sets are finite facilitates the analysis while taking large cardinalities suffices to model any practical system.

Condition 1. *Let $\phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})$ denote an empirical probability distribution over femtocaching plans $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ when traffic demand profile $\boldsymbol{\lambda}$, delay profile $\boldsymbol{d}$, and cost profile $\boldsymbol{h}$ are observed. Consider the following conditions:*

$$\sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} \phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}) = 1, \quad (7)$$

$$0 \leq \phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}) \leq 1, \forall (\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}),$$

$$\sum_{(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})} p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}} \sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} \phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}) \sum_{j \in \mathcal{J}} y_j h_j \leq B_{avg}, \quad (8)$$

where every tuple $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ considered above satisfies an femtocaching plan in Definition 1.

Lemma 1. *Condition 1 is necessary for any feasible elastic femtocaching policy.*

We remark that Condition 1 characterizes a convex set of distributions of femtocaching plans. Since Condition 1 is necessary for any feasible elastic femtocaching policy, it expresses an outer bound on the performance region of our system. Therefore, one way to solve our control problem is to solve an optimization problem over $\mathcal{G}$ and select as an elastic femtocaching policy the randomized actions ϕ^* which are the solution to the optimization. However, this is impossible without knowledge of the distribution $p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}}$, and therefore in the remaining sections, we will provide a dynamic algorithm that adapts to the observed conditions.

D. Handling Non-Linear Utilities

Since the maximization of a nonlinear function of a time average cannot be decomposed into slots, we consider an alternative decomposable problem. Namely, problem **(P)**, which maximizes a nonlinear function of a time average, can be transformed into maximization of the time average of a nonlinear function using the auxiliary variable technique in [6, Chapter 5]. To this end, we introduce an auxiliary variable vector $\boldsymbol{\gamma}(t) = (\gamma_1(t), ..., \gamma_{|\mathcal{I}|}(t))$ for all t and define a function $g(t)$ as follows: $g(t) = \sum_i \log(1+\gamma_i(t)), \forall t$. Using Jensen's inequality, we can upper-bound the mean value of $g(t)$ as follows: $\overline{g}(t) \leq \sum_i \log(1+\overline{\gamma_i}(t)), \forall t$, where $\overline{g}$ and $\overline{\gamma_i}$ denote the time-average of g and γ_i, respectively. Now, let us consider the following problem. Every time slot, the CP observes $(\boldsymbol{\lambda}(t), \boldsymbol{d}(t), \boldsymbol{h}(t))$ and chooses a control action $(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t))$ and an auxiliary vector $\boldsymbol{\gamma}(t)$, where $0 \leq \gamma_i(t) \leq D_{max}$ for all i and t to solve the following **(JP)** problem:

$$\max \liminf_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i} \log(1+\gamma_i(t)), \quad (9)$$

$$s.t. \ \liminf_{T\to\infty} \frac{1}{T} \Big| \sum_{t=0}^{T-1} \Big(\gamma_i(t) - D_i(\boldsymbol{x}(t),\boldsymbol{y}(t),\boldsymbol{z}(t))\Big) \Big| \leq 0, \forall i \in \mathcal{I}, (10)$$

$$\liminf_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{j \in \mathcal{J}} y_j(t) h_j(t) \leq B_{avg}, \quad (11)$$

where $0 \leq \gamma_i(t) \leq D_{max}, \forall i, \forall t$.

[11] To capture fairness in the problem, we can use an additional constraint which guarantees the minimum delay, instead of using this α-fairness utility function.

Lemma 2. *Solving problem* **(JP)** *yields a femtocaching plan that is at least as good as the optimal solution of the problem* **(P)**.

Proof. The lemma follows from [6, Chapter 5, 5.0.5]. □

Exploiting the transformation of **(JP)**, we can decompose the average objective function into different objectives for each slot. That is, maximizing $\sum_i \log(1+\gamma_i(t))$ every slot is equivalent to maximizing the average objective function in (9). Hence, we can adopt a standard Lyapunov drift-minus-benefit technique [6].

E. Virtual Queues

To keep track of the feasibility of problem **(JP)**, we introduce virtual queues corresponding to the average budget constraint (1) and auxiliary constraint (10) whose backlogs are updated by

$$Q_B(t+1) = \Big[Q_B(t) + \textstyle\sum_{j\in\mathcal{J}} y_j(t)h_j(t) - B_{avg}\Big]^+, \quad (12)$$

$$U_i(t+1) = \Big[U_i(t) + \gamma_i(t) - D_i(t)\Big]^+, \ \ \forall i \in \mathcal{I}. \quad (13)$$

Prior work [29] shows that if the stability conditions $\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} Q_B(t) < \infty$ and $\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} U_i(t) < \infty, \forall i \in \mathcal{I}$ are satisfied, then so are constraints (1) and (10). Intuitively, the backlogs $Q_B(t)$ and $U_i(t)$ for all i count the excess budget spent and excess auxiliary variable in the previous time slots for keeping track of the average budget expenditures and average caching benefits. Then, we propose a dynamic algorithm to solve **(JP)** in the next section.

IV. Lyapunov-based Dynamic Solution

A. Slot-by-slot Problem

We consider the slot-by-slot problem without knowledge of the average traffic demands $\mathbb{E}[\lambda_{i,f}(t)]$ for all subareas and files and the average delays $\mathbb{E}[d_{ij}(t)]$ for all subareas and SBSs. Then, let us focus on slot t. The decision-maker is aware of *(i)* the traffic demand profile for the next hour $[\lambda_{i,f}(t)]_{i,f}$ [12] *(ii)* the delay profile realizations for the next hour $[d_{ij}(t), d_{is}(t)]_{i,j}$ available by measurements, and the readily available, *(iii)* prices $[h_j(t)]_j$ [13], and *(iv)* virtual queue lengths $Q_B(t)$ and $U_i(t)$ for all $i \in \mathcal{I}$, while file size b is assumed known. Therefore, the elastic femtocaching policy is applied on the state $(\boldsymbol{\lambda}(t), \boldsymbol{d}(t), \boldsymbol{h}(t), Q_B(t), [U_i(t)]_i)$. To design a policy, we employ the *Lyapunov drift-minus-benefit* framework in the following.

We first define the quadratic Lyapunov function and arising drift as follows:

$$L(t) \triangleq \tfrac{1}{2}\Big\{Q_B(t)^2 + \textstyle\sum_{i\in\mathcal{I}} U_i(t)^2\Big\},$$
$$\Delta(L(t)) \triangleq \mathbb{E}\{L(t+1) - L(t)|\mathbf{Q}(t)\},$$

where $\mathbf{Q}(t) \triangleq \{Q_B(t), U_1(t), ..., U_{|\mathcal{I}|}\}$. Since we are also interested in maximizing the time average of delay utility $\log(1 + D_i(t))$ for all subareas using feasible cache plans, we next introduce the Lyapunov drift-minus-benefit function (*DMB*):

$$DMB(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{\gamma}(t)) = \Delta(L(t)) - V \sum_{i\in\mathcal{I}} \mathbb{E}\{\log(1+\gamma_i(t))|\mathbf{Q}(t)\}, \quad (14)$$

where V is a constant parameter to balance the trade-off between two conflicting objectives: improving the budget and auxiliary variable satisfaction, or increasing the average delay utility.

Applying the queue update equations (12), (13) and Lemma 4.3 from [31], we obtain under any possible decision $(y_j(t), x_{ij,f}(t), z_{j,f}(t), \gamma_i(t))$:

$$\begin{aligned} DMB(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{\gamma}(t)) \leq P – V \sum_{i\in\mathcal{I}} \mathbb{E}\{\log(1+\gamma_i(t))|\mathbf{Q}(t)\} \\ - \mathbb{E}\Big\{\Big(B_{avg} - \sum_{j\in\mathcal{J}} y_j(t)h_j(t)\Big)Q_B(t)|\mathbf{Q}(t)\Big\} \quad (15) \\ - \sum_{i\in\mathcal{I}} \mathbb{E}\Big\{(D_i(t) - \gamma_i(t))U_i(t)|\mathbf{Q}(t)\Big\}, \end{aligned}$$

where $P = 1/2(B_{avg}^2 + |\mathcal{J}|y_{max}^2 h_{max}^2 + 2|\mathcal{I}|D_{max}^2)$ is a positive constant when y_{max}, h_{max} D_{max} denote the allowable leased cache space at an SBS during an hour, the maximum price and the maximum hourly caching benefit for a subarea, respectively. Neely [6] showed that we can uncover optimal decisions by minimizing the RHS of (15).

We propose the elastic femtocaching policy (*EFP*) which at slot t takes actions $(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t), \boldsymbol{\gamma}(t)) = (\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*, \boldsymbol{\gamma}^*)$, where

$$\boldsymbol{\gamma}^* \in \arg\max_{\boldsymbol{\gamma}^*(t)} \underbrace{V \sum_{i\in\mathcal{I}} \log(1+\gamma_i(t)) - \sum_{i\in\mathcal{I}} U_i(t)\gamma_i(t),}_{\textbf{(JP)-(a)}}$$

$$(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*) \in \arg\max_{\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t)} \underbrace{\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t)) - Q_B(t) \sum_{j\in\mathcal{J}} y_j(t)h_j(t),}_{\textbf{(JP)-(b)}}$$

with instantaneous constraints (7) and (8).

The first straightforward result is that *EFP* is a feasible elastic femtocaching policy. First, the instantaneous constraints of service (7) and storage space (8) are automatically satisfied at each slot by the design of the policy. Then, we may observe that *EFP* minimizes the RHS of (15), therefore using lemma 4.6 in [6], we can show that *EFP* also stabilizes $Q_B(t)$ and $U_i(t)$ for all subareas, and hence the billing constraint (1) and constraint with respect to an auxiliary variable (10) are asymptotically satisfied.

Additionally, by adopting similar proof methodology in [6, Chapter 5, 5.1], we obtain the following results: *(i)* *Val*(**EFP**) $\geq$ *Val*(**UBound**) $- O(1/V)$, where *Val*(**Ubound**) denotes the value of the optimization problem with the sta-

[12] In practice, it is achieved by calculating the running average of files' popularity based on the demand during the past few time slots, or by even using more sophisticated statistical or machine learning methods, cf. [30].

[13] This price information can be provided by cloud service providers,e.g., AWS [4]. If such information is available with coarser time granularity (i.e., less often), then the system can use the prices in the past few time slots.

***General algorithm (GA) to solve problem* (JP)**

```
Initialization: At $t = 0$, $Q_B(t) = 0$ and $U_i(t) = 0$, $\forall i \in \mathcal{I}$.
Result: $x_{ij,f}(t), y_j(t), z_{j,f}(t), \gamma_i(t)$, $\forall i, j, f, t$
While In slot $t$, read values $Q_B(t), U_i(t), \lambda_{i,f}(t), d_{ij}(t), d_{is}(t), h_j(t)$, $\forall i, j, f$
  Step 1: Decision of auxiliary variables $\gamma_i^*(t)$ for all subareas.
  For each subarea $i \in \mathcal{I}$,
      Calculate $\gamma_i(t) = \frac{V}{U_i(t)} - 1$.
      If $\gamma_i(t) \in [0, D_{max}]$, then $\gamma_i^*(t) = \gamma_i(t)$.
      Else, $\gamma_i^*(t) = \max\big(0, V\log(1 + D_{max}) - U_i(t)D_{max}\big)$.
  End For
  Step 2: Decision of original control variables $(\boldsymbol{x}^*(t), \boldsymbol{y}^*(t), \boldsymbol{z}^*(t))$.
  Choose $\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t)$ which maximize
      $(\boldsymbol{x}^*(t), \boldsymbol{y}^*(t), \boldsymbol{z}^*(t)) \in \arg\max_{\boldsymbol{x}(t),\boldsymbol{y}(t),\boldsymbol{z}(t)} \textbf{(JP)-(b)}$,   (17)
  with $\boldsymbol{x}(t), \boldsymbol{y}(t), \boldsymbol{z}(t)$ satisfying (3), (4).
  Step 3: Update of parameters.
  Update all virtual queues based on $\big(\boldsymbol{x}^*(t), \boldsymbol{y}^*(t), \boldsymbol{z}^*(t), \boldsymbol{\gamma}^*(t)\big)$ using (12) and (13).
  Update $t \leftarrow t + 1$.
End While
```

tionary policy. *(ii)* All virtual queues can be stabilized and average queue length (sum of total average virtual queues) satisfies: $\frac{\mathbb{E}[||\mathbf{Q}(t)||]}{t} \leq \sqrt{\frac{2P+2V(D_{max}-\theta^*)}{t}}$.

Now, we provide a general algorithm (GA) to solve problem **(JP)**. In this GA, $\gamma^*(t)$ in **Step 1** can be obtained by a straightforward manner since objective (16) is a concave function with respect to the auxiliary variable for each subarea. However, $\boldsymbol{x}^*(t), \boldsymbol{y}^*(t), \boldsymbol{z}^*(t)$ in **Step 2** cannot be obtained easily due to the product of variables $x_{ij,f}(t)$ and $z_{j,f}(t)$ in $D_i(t)$. In order to implement the above GA in practice, we consider two different cases, *(i)* non-overlapping, and *(ii)* overlapping SBS coverages in the following section.

V. Intra-Slot Problem and Algorithms

In this section, we turn our attention into solving problem (17) in **Step 2** to find solutions $(\boldsymbol{x}^*(t), \boldsymbol{y}^*(t), \boldsymbol{z}^*(t))$ given cache lease budget B_{avg} and the maximum caching benefit D_{max}.

A. Non-overlapping SBS Coverage

When SBS coverage is non-overlapping, each subarea can reach a single SBS cache, which immediately simplifies routing splits $x_{ij,f}(t)$, such that $x_{ij,f}(t) = 1$, $\forall t$ if subarea i can reach SBS j and 0 otherwise, for all i, j, f. In essence, each request can be served only by the reachable cache (or the MBS when the file is not cached there). We will see that this makes our problem relatively easy to solve.

First, we note that caching file f at SBS j in slot t yields the following variable:

$$K_{j,f}(t) \triangleq \sum_i U_i(t) k_{ij,f}(t), \tag{18}$$

where $k_{ij,f}(t) = (d_{is}(t) - d_{ij}(t)) x_{ij,f}(t) \lambda_{i,f}(t)$. It is computable using known parameters $\boldsymbol{d}, \boldsymbol{x}, \boldsymbol{\lambda}$ ($\boldsymbol{x}$ is a parameter here because it is fully determined by the reachability of the cache) and independent of the decisions $\boldsymbol{y}(t), \boldsymbol{z}(t)$. Consequently, the EFP optimization problem becomes:

$$\begin{aligned} \max_{\substack{y_j(t) \geq 0 \\ z_{j,f}(t) \in \{0,1\}}} \; & \sum_{j,f} K_{j,f}(t) z_{j,f}(t) - Q_B(t) \sum_{j \in \mathcal{J}} y_j(t) h_j(t), \\ s.t. \; & \sum_{f \in \mathcal{F}} z_{j,f}(t) \leq y_j(t)/b, \quad \forall j, f. \end{aligned} \tag{19}$$

Due to its simple form, (19) can be solved by inspection. At each pair SBS-slot (j, t), we order files in decreasing values of $K_{j,f}(t)$. For an investment $y_j(t)$, the highest caching benefit is collected by caching the $y_j(t)/b$ files that rank higher in this list. This provides directly the solutions $\boldsymbol{z}(t)$ as a function of $\boldsymbol{y}(t)$, it remains now to determine the latter. With a slight abuse of notation, let us call σ the permutation of file induces that implies $K_{j,\sigma(1)}(t) \geq \cdots \geq K_{j,\sigma(|\mathcal{F}|)}(t)$ (the abuse is because we do not explicitly denote the dependence of σ on j, t to reduce clutter), then we can decompose the investment decisions per SBS, and find $y_j^*(t)$ by maximizing:

$$y_j^*(t) \in \arg\max_{y_j(t) \geq 0} \sum_{f=1}^{\lfloor y_j(t)/b \rfloor} K_{j,\sigma(f)}(t) - Q_B(t) h_j(t) y_j(t).$$

Above, $y_j^*(t)$ can be efficiently computed by listing partial sums $\sum_{f=1}^{\lfloor y_j(t)/b \rfloor} K_{j,\sigma(f)}(t)$ for $y_j(t)/b = 1, 2, \ldots$ until the difference of one partial sum from the previous one becomes smaller than $Q_B(t)h_j(t)$. Below, we provide the algorithmic steps to find $\boldsymbol{y}(t)$ and $\boldsymbol{z}(t)$ in detail.

Joint Cache Rental and File Caching Algorithm (JCC) (Step 2)

```
Result: $y_j(t), z_{j,f}(t)$, $\forall j, f, t$
Read values $Q_B(t), U_i(t), x_{ij,f}(t), \lambda_{i,f}(t), d_{ij}(t), d_{is}(t), h_j(t)$, $\forall i, j, f$
For all SBSs $j \in \mathcal{J}$,
    For all files $f \in \mathcal{F}$,
        Calculate $K_{j,f}(t)$ using (18).
    End For
    Sort $K_{j,f}(t)$ with permutation $\sigma$, such that $K_{j,\sigma(1)}(t) \geq \cdots \geq K_{j,\sigma(|\mathcal{F}|)}(t)$.
    Set partial sums $S(e) = \sum_{f=1}^{e} K_{j,\sigma(f)}(t)$, for $e = 1, 2, \ldots$ and $S(0) = 0$.
    Find $e^*$ which maximizes $S(e) - Q_B(t)h_j(t)be$.
    Choose cache lease: $y_j(t) = e^* b$
    Choose file placement:
        $z_{j,\sigma(f)}(t) = \begin{cases} 1 & \text{if } f \leq \lfloor y_j(t)/b \rfloor, \\ 0 & \text{otherwise.} \end{cases}$
    End For
End For
```

Then, the *JCC* algorithm in the non-overlapping SBS case has the following features: *(i)* Given virtual queues, association variables, demand traffic and delay profiles and storage price, the algorithm finds the amount of storage that optimizes a weighted sum of caching benefit constrained by the virtual queue stability. *(ii)* For the found storage amount that is leased, files are cached at each SBS according to which yields the highest caching benefit multiplied by virtual queue $U_i(t)$ for all subareas, until the available leased storage is filled up. *(iii)* If the average caching benefit for subarea i until time slot t becomes smaller, the virtual queue $U_i(t)$ for subarea i gets higher, which makes the probability to cache files requested by subarea i higher by Eq. (18); hence the average caching benefit for subarea i increases. Therefore, this mechanism intuitively captures fairness among all subareas.

B. General Case with Overlapping SBS Coverage

Next, we consider the general case where the coverage areas of the different SBSs can overlap. Then, the association variables $x_{ij,f}(t)$ must be jointly decided with cache rental and file placement. Recall that Eq. (17) in **Step 2** of GA determines the decisions solving:

$$\max_{\substack{y_j(t)\geq 0,\forall j\\ x_{ij,f}(t)\in[0,1]\forall i,j,f\\ z_{j,f}(t)\in\{0,1\},\forall j,f}} \quad \textbf{(JP)-(b)} \tag{20}$$

$$s.t. \quad \sum_{f\in\mathcal{F}} z_{j,f}(t) \leq \frac{y_j(t)}{b},\ \forall j, \quad \sum_{j\in\mathcal{J}_i} x_{ij,f}(t)=1, \forall i, f.$$

We note that (20) is a mixed-integer *non-linear* program (MINLP) due to the product of variables $x_{ij,f}(t)$ and $z_{j,f}(t)$ that appear in the objective. To solve this problem, we can consider two approaches:

- As explained in [2], it is possible to use MDS codes to achieve an effective "fractional file placement". In essence, each cache stores a number of linear combinations of file chunks that correspond to fractions of a file, and then each user can combine different such coded chunks to produce the original file.
- A second approach to obtain an efficient approximate solution is to apply the idea of "Low complexity scheduling" from [7]. This method assigns to the leased cache capacity by randomly selecting it for each SBS. Then, it resolves our *EFP* optimization to get a new average delay utility, and if these new values outperform previous delay utilities, the random solution is applied.

In this paper, we take the second method since *(i)* it has low computation complexity and *(ii)* it does not need to invoke additional coded caching techniques. In this context, we provide a stability guarantee for the budget queue length $Q_B(t)$ and the virtual queue lengths $U_i(t)$ for all subareas, which implies that the produced policy is asymptotically feasible. The proposed joint cache rental, greedy file caching and routing algorithm, namely *JGCA* is described as follows.

Joint Cache Rental, File Caching and Routing Algorithm (JGCA) (Step 2)

```
Result: $y_j(t), z_{j,f}(t), x_{ij,f}(t)\ \forall i,j,f,t$
In slot $t$, read values $Q_B(t), \lambda_{i,f}(t), d_{ij}(t), d_{is}(t), h_j(t)$, $\forall i,j,f$
At $t=1$, $y_j^*(1)$ is chosen as $B_{avg}/(|\mathcal{J}|h_{avg})$ for all $j\in\mathcal{J}$.
Based on the decided $y_j^*(1)$ for all $j\in\mathcal{J}$, $(\boldsymbol{x}^*(1), \boldsymbol{z}^*(1))$ are obtained using a greedy file caching and association (GFCA) policy.
For time slots $t>1$, $y_j'(t)$ is uniformly and randomly chosen among $\mathcal{U}=\{0,b,2b,...y_{max}\}$ for all $j\in\mathcal{J}$.
Based on the decided $y_j'(t)$ for all $j\in\mathcal{J}$, $(\boldsymbol{x}'(t), \boldsymbol{z}'(t))$ are obtained using the GFCA policy.
Compare
    $$\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)) - Q_B(t)\sum_{j\in\mathcal{J}} y_j'(t)h_j(t) \quad (21)$$
    and
    $$\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}^*(t-1),\boldsymbol{z}^*(t-1)) - Q_B(t)\sum_{j\in\mathcal{J}} y_j^*(t-1)h_j(t). \quad (22)$$
If Eq. (21) > Eq. (22), then $(\boldsymbol{x}^*(t)=\boldsymbol{x}'(t), \boldsymbol{y}^*(t)=\boldsymbol{y}'(t), \boldsymbol{z}^*(t)=\boldsymbol{z}'(t))$,
Else, $(\boldsymbol{x}^*(t)=\boldsymbol{x}^*(t-1), \boldsymbol{y}^*(t)=\boldsymbol{y}^*(t-1), \boldsymbol{z}^*(t)=\boldsymbol{z}^*(t-1))$.
```

The greedy file caching and association (GFCA) policy included in the JGCA can be explained as follows. First, if the total leased cache capacity $\sum_j y_j(t)$ is less than or equal to the three folds of a file size (i.e., $3b$), compare **(JP)-(b)** for all possible sets of $(\boldsymbol{x}(t), \boldsymbol{z}(t))$, and pick the biggest value. Let $(\boldsymbol{x}'(t), \boldsymbol{z}'(t))$ be the corresponding set of association variables and cached files, i.e., making the biggest **(JP)-(b)**. For a given file caching variable, the optimal subarea association can be decided by

$$x'_{ij,f}(t) = \arg\max_{x_{ij,f}(t)} \sum_{i\in\mathcal{I}} U_i(t) \times \sum_{f\in\mathcal{F}}\sum_{j\in\mathcal{J}} (d_{is}(t)-d_{ij}(t))z'_{j,f}(t)\lambda_{i,f}(t), \forall i, f.$$

Since the maximum number of cached files for all SBSs is three, the complexity of the exhaustive search of optimal $(\boldsymbol{x}(t), \boldsymbol{z}(t))$ in this case can be manageable.

Second, if the total leased cache capacity $\sum_j y_j(t)$ is greater than $3b$, the GFCA policy iteratively caches files one-by-one where an added file in each iteration is selected so as to maximize **(JP)-(b)** with the corresponding optimal subarea association $\boldsymbol{x}(t)$ until leased cache capacity $y_j(t)$ is completely filled up for all SBSs. For instance, at the first iteration, caching file #5 at SBS #1 and the second iteration, caching file #7 at SBS #5, and so on.

C. Performance Bounds

Now, we show the theoretical performance bounds of the proposed algorithms in a general scenario. First, we show the performance bound of slot-by-slot objective, i.e., **(JP)-(b)** for a given cache lease capacity $\boldsymbol{y}(t)$ using submodularity [32], [33] of **(JP)-(b)** in Lemma 3; and using the lemma and randomized scheduling policy [34], we show the performance bound of **(JP)-(b)** in each time slot. Finally, we prove that the proposed algorithms in the general case can achieve constant performance bounds of the utility and virtual queues in Theorem 1.

Definition 3. *A real-valued set function $\mathcal{A}$, defined on the subsets of finite sets $\mathbf{\Omega}$ is named a submodular set function if it satisfies the following condition for all $\mathbf{\Omega} \subseteq \mathbf{\Omega}'$, for all $f(j) \in \mathcal{F} \backslash \mathbf{\Omega}'^{j}$, and for all $j \in \mathcal{J}$:*

$$\mathcal{A}(\mathbf{\Omega} \cup \{f(j)\}) - \mathcal{A}(\mathbf{\Omega}) \geq \mathcal{A}(\mathbf{\Omega}' \cup \{f(j)\}) - \mathcal{A}(\mathbf{\Omega}'). \quad (23)$$

Lemma 3. *The objective function* **(JP)-(b)** *for a given leased cache capacity $\boldsymbol{y}$ is a non-decreasing and submodular set function with respect to file caching assuming that the optimal subarea association can be obtained for a fixed file caching state.*[14]

Proof. The submodularity of this function can be proved by searching all possible cases when a new file is cached at a particular SBS or not. □

Then, let the submodular objective function for a given leased cache capacity in Lemma 3 be

$$F(\boldsymbol{x}'(t), \boldsymbol{z}'(t)|\boldsymbol{y}(t)) = \sum_{i \in \mathcal{I}} U_i(t) D_i(\boldsymbol{x}'(t), \boldsymbol{y}(t), \boldsymbol{z}'(t)) - Q_B(t) \sum_{j \in \mathcal{J}} y_j(t) h_j(t),$$

where $\boldsymbol{y}(t)$ denotes the given leased cache capacity, $\boldsymbol{x}'(t)$ and $\boldsymbol{z}'(t)$ denote the subarea association and file caching solutions from the GFCA policy, respectively. In addition, let $\boldsymbol{x}^*(t)$ and $\boldsymbol{z}^*(t)$ be the optimal solutions of $F(\boldsymbol{x}(t), \boldsymbol{z}(t)|\boldsymbol{y}(t))$, respectively. The objective function $F(\boldsymbol{x}(t), \boldsymbol{z}(t)|\boldsymbol{y}(t))$ is convex in $\boldsymbol{x}(t)$ given $\boldsymbol{z}(t)$, but becomes a nonconvex and also discontinuous function when $\boldsymbol{z}(t)$ is considered as a variable. Thus, this problem is a challenging combinatorial problem with $O(2^{|\mathcal{F}|})$ possible cases. However, prior works, e.g., [32], [33] showed that if we can find an optimal solution of $\boldsymbol{x}(t)$ (or similar coupled variables) within a polynomial time given $\boldsymbol{z}(t)$, it is enough to prove $1 - 1/e$ performance bound of a greedy file caching algorithm using a submodularity of $F(\boldsymbol{x}(t), \boldsymbol{z}(t)|\boldsymbol{y}(t))$.

Therefore, the GFCA policy given $\boldsymbol{y}(t)$ can guarantee:

$$F(\boldsymbol{x}'(t), \boldsymbol{z}'(t)|\boldsymbol{y}(t)) \geq (1 - 1/e) F(\boldsymbol{x}^*(t), \boldsymbol{z}^*(t)|\boldsymbol{y}(t)). \quad (24)$$

According to JGCA in GA, the following lemma holds.

Lemma 4. *With uniformly and randomly picked $y_j(t)$ for all SBSs $j \in \mathcal{J}$, there exists a positive constant $0 < \rho < 1$ such that $Pr\{y_j(t) = y_j^*(t), \forall j \in \mathcal{J}\} \geq \rho$.*

Proof. The set of possible $y_j(t)$ values is finite, i.e., $|\mathcal{S}| < \infty$ where $\mathcal{S} = \{0, b, 2b, ..., y_{max}\}$. Hence, if the random selection of $y_j(t)$ distributes uniformly, there exist $y_j^*(t)$ for all SBSs $j \in \mathcal{J}$ which maximize $F(\boldsymbol{x}^*(t), \boldsymbol{z}^*(t)|\boldsymbol{y}^*(t))$ with probability $\rho \geq \frac{1}{|\mathcal{S}|^{|\mathcal{J}|}} > 0$. □

Then, we quantify the performance of the proposed GA+JGCA in the following Lemma 5 and Theorem 1.

Lemma 5. *Let $\boldsymbol{y}'(t)$ and $\boldsymbol{y}^*(t)$ be the solution of JGCA and one of the optimal solutions of the problem* **(JP)-(b)** *satisfying the femtocaching plan and the feasible elastic femtocaching policy, respectively. Then, the JGCA in GA guarantees the following performance in every time slot.*

$$\begin{aligned}\mathbb{E}\big\{F(\boldsymbol{x}'(t), \boldsymbol{z}'(t)|\boldsymbol{y}'(t))|\mathbf{Q}(t)\big\} \geq \\ (1 - 1/e)\mathbb{E}\big\{F(\boldsymbol{x}^*(t), \boldsymbol{z}^*(t)|\boldsymbol{y}^*(t))|\mathbf{Q}(t)\big\} - R,\end{aligned} \quad (25)$$

where $R = \frac{1}{\min\{p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}}\}\rho}(2|\mathcal{I}|D_{max}^2 + |\mathcal{J}|^2 y_{max}^2 h_{max}^2 + B_{avg}|\mathcal{J}|y_{max}h_{max})$.

Theorem 1. *Assume that a tuple $(\mathbf{d}, \boldsymbol{\lambda}, \boldsymbol{h})$ is i.i.d. In addition, let $\gamma_i'(t)$ and $\gamma_i^*(t)$ for all subareas $i \in \mathcal{I}$ be the solution of GA and the optimal value of* **(JP)***, respectively, and let $y_j'(t)$ and $y_j^*(t)$ for all SBSs $j \in \mathcal{J}$ be the solution of JGCA in GA and the optimal value of* **(JP)***, respectively, and let $D_i'(t)$ and $D_i^*(t)$ for all subareas $i \in \mathcal{I}$ be the caching benefits from JGCA in GA and the optimal value of* **(JP)***, respectively. Assume $\epsilon_i = (1 - 1/e)\mathbb{E}\{\gamma_i^*(t)\} - \mathbb{E}\{\gamma_i'(t)\} > 0$, $\forall i \in \mathcal{I}$. Then, the proposed GA and JGCA guarantee:*
1) The virtual queues $Q_B(t)$ and $U_i(t)$ for all $i \in \mathcal{I}$ are stable.

2) $$\frac{\mathbb{E}[||\mathbf{Q}(t)||]}{t} \leq \sqrt{\frac{2(P+R) + 2V(D_{max} - \theta^*)}{t} - \frac{1}{t^2} T|\mathcal{I}|\epsilon_{min} U_{min}}. \quad (26)$$

3) $$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i \in \mathcal{I}} \log(1 + \gamma_i'(t)) \geq \theta^* - \frac{P+R}{V}. \quad (27)$$

This solution is robust due to the comparison mechanism between the solution of the current time slot and that of the previous slot. Namely, if the budget queue increases due to the excessive investment for cache lease, it reduces the objective value, hence it forces the decision-maker to choose the solution of the previous slot. On the other hand, if the budget queue decreases due to the smaller investment for cache leasing, it increases the objective value, hence it forces the decision-maker to choose the solution of the current time slot. This mechanism stabilizes the budget queue. For the stabilization of the virtual queues for auxiliary variables, if the average caching benefit of subarea i until time t becomes smaller, the virtual queue $U_i(t)$ gets higher; thus $\gamma_i(t)$ becomes smaller by Eq. (16). It makes negative feedback for the virtual queue $U_i(t)$. Therefore, $U_i(t)$ can be stabilized.

VI. Performance Evaluation

In this section, we execute simulations to demonstrate the performance of the proposed elastic femtocaching algorithms.

A. Analysis of Demand and Price Dataset

We analyze the real datasets of traffic demand and cache lease price generated from the variation of electricity price.

Traffic demand: To generate traffic demand, we use the YouTube file request dataset in [36]. We divide the dataset

[14] Recall that the op timal subarea association can be easily obtained by a typical optimization technique [35] since **(JP)-(b)** becomes a convex function for a fixed file caching state [33].

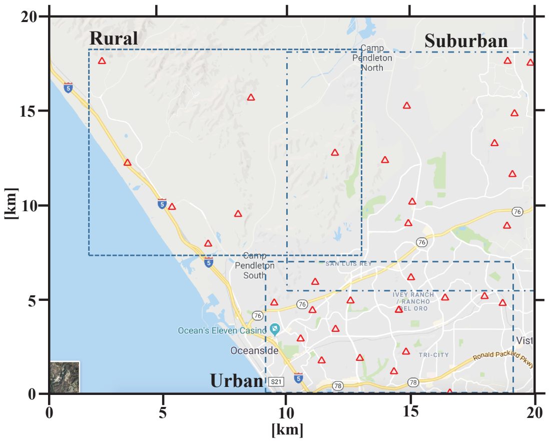


Fig. 2: Real BS topology (denoted by triangles) of a mobile operator on the west side of the US for rural, suburban and urban areas. We assume that the macro BS is located at the center of each area.

into each file for each day and each region.[15] Fig. 3(a) and 3(b) depicts the result. The number of file requests in Fig. 3(a) have daily and weekday/weekend traffic patterns. However, the traffic pattern is not similar even between two consecutive days, which makes the prediction of demand difficult. In addition, the real distribution for different regions and different time zones are fitted by Zipf distribution [37] with different parameters to show the spatio-temporal diversity of content popularity. The Zipf distribution is a well-known content popularity distribution where a higher Zipf parameter is interpreted as a higher difference of popularity among files. As shown in Fig. 3(b), the distributions of content popularity in different time zones and different regions have different Zipf parameters.

Cache prices: Currently, Amazon AWS provides constant per-hour pricing service, i.e., *ElastiCache* for leasing a unit cache resource, e.g., \$0.0309/hour/GB for cache.t2.micro CPU and US Ohio region [39]. However, since various factors such as electricity price can change this price over time, we use open traces of electricity price in the Canada Ontario region [38] to generate time-varying cache lease price with the mean value of \$0.0309/hour/GB. Fig 3(c) depicts cache lease price traces per hour and GBytes, which is not static, but highly dynamic. The analyses of Fig. 3 imply that the cache lease capacity, file caching and subarea-SBS association algorithms must be designed in an online fashion, and adapted to the varying content popularity and cache lease price to optimize the delay performance.

B. Simulation Setup

We consider two different cases in the simulations: *i)* linear BS topology with manual parameters and *ii)* real BS topology in Fig. 2 (where a macro BS is located on the center of each plane) and real parameters.

Linear BS topology case: A macro BS is located at the center of a linear line and two SBSs are located at the same distance from the macro BS. Moreover, 8 mobile users[16] are uniformly distributed. The files are requested by Zipf distribution with randomized Zipf parameters and the mean cache lease price per bit is 0.025 with randomized varying parameters where the variance of the randomness (with Gaussian distribution taking positive values) can be different.[17] Here, the order of popularity of each file in each subarea is randomly chosen. The distance between a BS and a mobile user is used to calculate the path-loss parameter. The transmission power of an SBS is 1 whereas that of an MBS is 20. The path loss is set to be $128.1 + 37.6\log_{10}(d)$ in a typical LTE system evaluation [40] where d is the distance of the BS from the center of each area, and the system bandwidth is set to be 10MHz. Moreover, fast fading is captured by randomness with different variation rates. We generate wireless transmission delay by dividing a constant file size, i.e., 1 into transmission data rate calculated by Shannon capacity formula with these path-loss parameters. In addition, delay for wired backhaul transmission is randomly chosen. Here, delay $d_{ij}(t)$ from SBS j to subarea (or mobile user) is just wireless transmission delay whereas delay $d_{is}(t)$ via MBS s to subarea (or mobile user) i is the sum of wireless transmission delay plus wired backhaul transmission since there are no files to be cached in MBS s. The average cache rental budget B_{avg} is set to be 20.

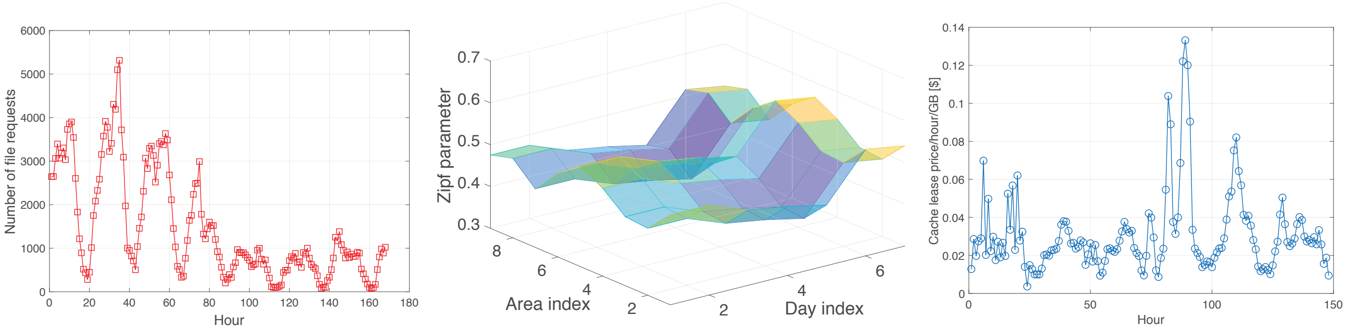
Real BS topology case: We exploit an open dataset (rural, suburban and urban areas as shown in Fig. 2), namely cellmapper [41]. We divide this topology into 9 subareas and calculate the wireless delay profile based on the center location of each subarea. A way to calculate wireless delay profile is the same with the linear BS topology case except for 100MBytes of file size. Moreover, the backhaul transmission delay is randomly picked from a dataset in [36]. Traffic demand and cache lease prices follow the real traces from [36] and [39]. The average cache rental budget is set to be enough to cache 20% of the entire file catalog, which depends on the different BS topology (rural, suburban and urban areas) and average cache lease price. For all cases, the entire catalog has 200 files, 10 simulation runs are executed and the average values are taken where the number of running time slots of each simulation is set to be 2000, and $V = 10$.

We consider the following metrics to analyze the performance of the proposed GA with JCC and JGCA: sum delay utilities of all subareas, i.e., objective value (6) in our problem formulation. In other words, maximizing this metric is exactly the same as maximizing the objective function. Additionally, we compare the proposed GA+JCC and GA+JGCA and four comparing algorithms, i.e., FCB (Fixed Cache Lease Budget), FFC (Fixed File Caching), LRU (Least Recently Used) and LFU (Least Frequently Used). The FCB algorithm uses the fixed cache lease budget for all SBSs and all-time slots, i.e., each SBS has a constant cache lease budget $B_{avg}/|\mathcal{J}|$.

[15] The reference [36] provided an open YouTube video request dataset collected every 5 minutes for 7 days in a certain university campus. The data includes individual IDs of each requested video, requested time and destination/source IP addresses, video size, and transmission data rate. We distinguish different regions with different IP addresses.

[16] Here, each user represents each area.

[17] Note that we ignore units in this first case for simplicity.



(a) Evolution of file requests. (b) Zipf distribution. (c) Cache lease price.

Fig. 3: Spatio-temporal variation of input parameters for seven days: In (a), it shows the total number of YouTube file requests in a certain university campus over time from a dataset in [36]. In (b), the real content popularity distribution for different regions and different time zones are fitted by Zipf distribution [37]. In (c), cache lease price traces per hour and GByte is depicted from a real electricity price dataset [38] and cache lease price [39].

Here, the subarea-SBS association is given by the nearest SBS association policy and file caching is based on the highest file popularity every time slot. In addition, the FFC algorithm uses the nearest SBS association policy and caches files with the order of the global content popularity for all time slots, i.e., this algorithm does not take into account the spatio-temporal variation of the content popularity. Note that the FCB and FFC algorithms capture characteristics of other existing proactive caching policies which, however, do not consider cache scaling [13], [14]. Similarly, the LRU, LFU, multi-LRU and q-LRU are representative reactive caching policies which also do not consider cache scaling [15]–[18]. Specifically, the multi-LRU policy [16] caches files at each SBS based on the LRU rule, while the routing takes place via the closest BS that caches the requested file. Besides, the q-LRU policy [17] with the "lazy" rule for $q = 1$ operates as the multi-LRU policy but updates the cache only if the file is not in any neighboring BSs. Note that the operation of multi-LRU and q-LRU are essentially similar to the standard LRU policy when the BSs have non-overlapping coverage areas. We choose the aforementioned algorithms (i.e., FCB, FFC, LFU, LRU, multi-LRU, q-LRU) as benchmarks because they are quite representative of the entire spectrum of previous works.

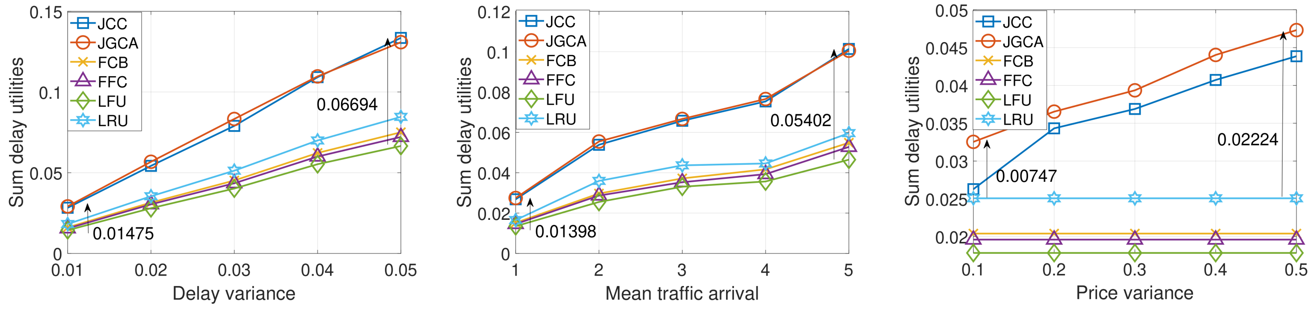
C. Simulation Results

Linear BS topology case. We first show the simulation results in the linear BS topology case to see the impact of different parameters (variance of delay, mean traffic arrival, and variance of price) on the system performance. We also consider two different scenarios under *non-overlapping SBSs* and *overlapping SBSs*. We present our results by summarizing the key observations in the following.

1) Non-overlapping SBSs scenario. In this scenario, GA with JCC is an optimal algorithm since the subarea-SBS association and content caching is uncoupled. Fig. 4 depicts the sum delay utilities for different parameters. We confirm that all budget queues and virtual queues for auxiliary variables in both GA+JCC and GA+JGCA are stable, which implies that the proposed algorithms guarantee the average budget constraint and constraint (10) (average difference between auxiliary variable and caching benefit converges to zero) are satisfied. The result shows that as the variation of input parameters becomes higher and mean traffic arrival increases, the performance gap between the proposed *elastic* cache leasing algorithms (i.e., GA+JCC and GA+JGCA) and the *static* cache leasing algorithms (FCB, FFC, LRU, LFU) increases (e.g., the sum of delay utilities is 0.01475 when delay variance is 0.01 and that is 0.06694 when delay variance is 0.05 in Fig. 4(a)). This implies that the proposed *elastic* cache leasing algorithms can attain greater performance compared to existing *static* cache leasing algorithms in the case that the network environments and pricing drastically change. Here, it is notable that the proposed GA+JGCA algorithm achieves a similar performance with the optimal GA+JCC algorithm.

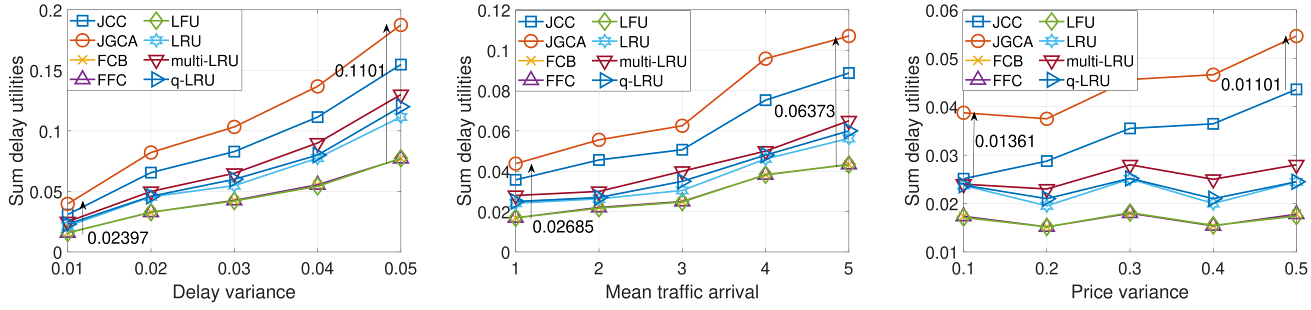
2) Overlapping SBSs scenario. In this scenario, GA+JCC is not an optimal algorithm anymore since the subarea-SBS association and content caching are tightly coupled with each other. Fig. 5 depicts the sum delay utilities for different parameters, i.e., delay variance, mean traffic arrival and price variance. Similar to the non-overlapping SBSs scenario, the proposed *elastic* cache leasing policies (GA+JCC and GA+JGCA) are much better than the existing *static* cache leasing policies (FCB, FFC, LFU, LRU, multi-LRU, q-LRU) and the gain of elastic cache leasing becomes higher as the variation of input parameters increases except for the price variation case. Here, multi-LRU [16] and q-LRU [17] with the "lazy" rule for $q = 1$ outperform the original LRU and LFU due to intelligent association rules. However, the proposed JCC and JGCA outperform these recent policies thanks to the dynamic and optimal usage of cache scaling. Moreover, the performance of the proposed *joint* cache lease, file caching and subarea-SBS association, i.e., GA+JGCA is higher (e.g., 30.3% to 48.8% higher in different price variation) than *independent* control of file caching and subarea-SBS association, i.e., GA+JCC in this overlapping SBSs scenario.

Real BS topology case. Real BSs (SBSs and MBS) are more irregularly deployed than that in the linear BS topology case; hence delay profile of each user from each BS can be



(a) Delay variance vs. sum delay utilities (b) Mean arrival vs. sum delay utilities (c) Price variance vs. sum delay utilities

Fig. 4: Sum utilities in linear BS topology case and non-overlapping SBSs scenario.



(a) Delay variance vs. sum delay utilities (b) Mean arrival vs. sum delay utilities (c) Price variance vs. sum delay utilities

Fig. 5: Sum utilities in linear BS topology case and overlapping SBSs scenario.

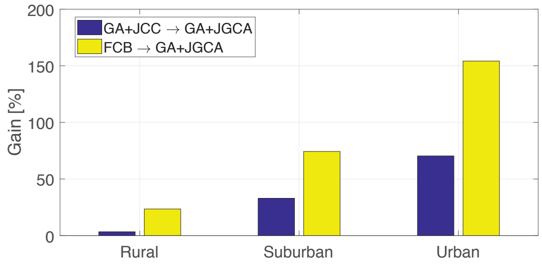


Fig. 6: Performance gain of the proposed algorithms under three real BS topologies and overlapping SBSs scenario.

significantly heterogeneous. Fig. 6 depicts the performance gain (i.e., the gain of sum delay utilities) of the proposed GA+JGCA algorithm over *static* FCB algorithm and the proposed GA+JCC algorithm which uncouples routing and caching decisions. First, since the real BS topology is more heterogeneous than the linear BS topology, the impact of the *elastic* cache leasing policy (i.e., GA+JGCA) on the system performance is higher than that in the linear BS topology case, especially in urban BS topology case. Second, as BS topology becomes denser (from rural to urban), the impact of the elastic cache leasing policy (i.e., GA+JGCA) on the system performance increases. This is because the caching benefits from backhaul transmission via the MBS to wireless transmission via the SBSs are higher in urban areas than that in rural areas. Third, *joint* control of cache leasing, file caching and routing becomes more important as BS topology becomes more dense. This interpretation can be driven from the fact that as BS topology gets denser (i.e., from rural area to urban area), the gain from the routing-caching the uncoupled solution, i.e., GA+JCC to the joint solution, i.e., GA+JGCA becomes higher.

VII. Concluding Remark

Motivated by recent market developments and the potential of elastic CDNs, we proposed the new problem of dynamic cache rental, file caching and user association for wireless edge caching networks. We formulated an optimization problem for deriving these decisions in a fashion that maximizes aggregated delay savings and/or ensures servicing fairness across users, while respecting average budget constraints. A tailored dynamic algorithm was proposed to solve the problem by capturing both caching benefit and fairness among different users while ensuring long-term cache rental budget under uncertainty of file popularity and wireless channel states over time and space. Simulation results revealed that the proposed elastic cache leasing algorithm would be more important when the network environments and cache leasing price are highly volatile, which is one of the common scenarios in wireless and heterogeneous network architecture. Although we spurred a joint optimization of cache scaling, file caching and routing research, this work can be improved by addressing a few more practical aspects. First, reconfiguration cost when new files are retrieved from the original file server to the SBSs can be incorporated into our framework. A few works on Lyapunov optimization addressed the reconfiguration issue [42]. The practical approach is that the decision maker updates the corresponding file when the expected benefit (say, the objective function for newly cached file minus reconfiguration cost, e.g., bandwidth cost) is greater than the objective function for the previously cached file [14]. Second, incorporation of resource allocation decisions, e.g., transmission power control and beam/user scheduling, into this cache scaling framework would further improve the system performance. Clearly, one can achieve higher caching benefits when coordinating additional network operation parameters with the elastic caching decisions.

Appendix A
Proof of Lemma 1.

We pick an arbitrary feasible elastic CDN policy π, and prove that Condition 1 is satisfied. Note that constraints (7)-(8) are satisfied by the fact that π chooses elastic cache plans every slot (see definition of the feasible elastic CDN policy).

Then, let $\Phi_t(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})$ denote the number of slots up to t that the state was $(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})$ and policy π chose the elastic cache plan $(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$, and further, let $J_t(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})$ be the total number of slots up to t that the state was $(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})$. Then the empirical frequency of choosing this plan while in this state is simply $\Phi_t(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})/J_t(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})$. Here we adopt the standard assumption that the limits of empirical frequencies exist:[18]

$$\phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}) \triangleq \lim_{t\to\infty} \Phi_t(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})/J_t(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}).$$

Immediately, we may observe that $\phi(\cdot)$ is a (conditional) probability distribution over elastic cache plans, and therefore it satisfies all constraints (7). Finally, it remains to count the cost paid by π. This is simply

$$\frac{Total\_Cost\_Upto\_t}{t} = \sum_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}} \underbrace{\frac{J_t(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})}{t}}_{\substack{\text{fraction of time the}\\ \text{state was } (\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})}} \sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} \underbrace{\frac{\Phi_t(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})}{J_t(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})}}_{\substack{\text{frequency of plan } (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\\ \text{when state is } (\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})}} \sum_{j\in\mathcal{J}} y_j h_j, \quad (28)$$

and taking limits we arrive at the conclusion:

$$\lim_{t\to\infty} \frac{Total\_Cost\_Upto\_t}{t} = \sum_{(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})} p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}} \sum_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})} \phi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}|\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}) \sum_{j\in\mathcal{J}} y_j h_j. \quad (29)$$

Since policy π is feasible, it satisfies the constraint (1) and therefore $\lim_{t\to\infty}\frac{Total\_Cost\_Upto\_t}{t} \le B_{avg}$. It follows that condition (8) must also be satisfied.

Appendix B
Proof of Lemma 5

Recall the virtual queue dynamics in (12) and (13) for all subareas $i \in \mathcal{I}$, then we have: $U_i(t) \le U_i(t-1)+\gamma_{max}$, $\forall i \in \mathcal{I}$ and $Q_B(t) \ge Q_B(t-1) - B_{avg}$ where $\gamma_{max} = D_{max}$. Let us define $\tau_{\kappa,t} < \tau_{\kappa-1,t} < \ldots < \tau_{1,t}$ where $\boldsymbol{d}(\tau_{i,t}) = \boldsymbol{d}(t)$, $\boldsymbol{\lambda}(\tau_{i,t}) = \boldsymbol{\lambda}(t)$ and $\boldsymbol{h}(\tau_{i,t}) = \boldsymbol{h}(t)$, $\forall i \le \kappa$, $i \in \mathbb{N}^+$.

From Lemma 4, there must exist $\kappa \in \mathbb{N}^+$ such that $y'_j(\tau_{\kappa,t}) = y^*_j(\tau_{\kappa,t})$ for all SBSs $j \in \mathcal{J}$. Therefore, we have:

$$\begin{aligned}
&\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}^*(t)) - Q_B(t)\sum_{j\in\mathcal{J}} y^*_j(t)h_j(t)\\
&\le \sum_{i\in\mathcal{I}}(U_i(\tau_{\kappa,t}) + (t-\tau_{\kappa,t})D_{max})D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}^*(t))\\
&-(Q_B(\tau_{\kappa,t}) - (t-\tau_{\kappa,t})B_{avg})\sum_{j\in\mathcal{J}} y^*_j(t)h_j(t)\\
&\le \sum_{i\in\mathcal{I}} U_i(\tau_{\kappa,t})D_i(\boldsymbol{x}'(\tau_{\kappa,t}),\boldsymbol{z}'(\tau_{\kappa,t})|\boldsymbol{y}^*(\tau_{\kappa,t}))\\
&-Q_B(\tau_{\kappa,t})\sum_{j\in\mathcal{J}} y^*_j(\tau_{\kappa,t})h_j(\tau_{\kappa,t})\\
&+(t-\tau_{\kappa,t})\big[|\mathcal{I}|D^2_{max} + B_{avg}|\mathcal{J}|y_{max}h_{max}\big]. \qquad (30)
\end{aligned}$$

Moreover, from the virtual queue dynamics in (12) and (13), we have: $U_i(t) \ge U_i(t-1) - D_{max}$ for all subareas $i \in \mathcal{I}$ and $Q_B(t) \le Q_B(t-1) + |\mathcal{J}|y_{max}h_{max}$. Then, the following inequality holds:

$$\begin{aligned}
&\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}'(t)) - Q_B(t)\sum_{j\in\mathcal{J}} y'_j(t)h_j(t)\\
&\ge \sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(\tau_{\kappa,t}),\boldsymbol{z}'(\tau_{\kappa,t})|\boldsymbol{y}'(\tau_{\kappa,t})) - Q_B(t)\sum_{j\in\mathcal{J}} y'_j(\tau_{\kappa,t})h_j(t)\\
&\ge \sum_{i\in\mathcal{I}}(U_i(\tau_{\kappa,t}) - (t-\tau_{\kappa,t})D_{max})D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}'(\tau_{\kappa,t}))\\
&-(Q_B(\tau_{\kappa,t}) + |\mathcal{J}|y_{max}h_{max}(t-\tau_{\kappa,t}))\sum_{j\in\mathcal{J}} y'_j(\tau_{\kappa,t})h_j(\tau_{\kappa,t})\\
&\ge \sum_{i\in\mathcal{I}} U_i(\tau_{\kappa,t})D_i(\boldsymbol{x}'(\tau_{\kappa,t}),\boldsymbol{z}'(\tau_{\kappa,t})|\boldsymbol{y}^*(\tau_{\kappa,t})) - (t-\tau_{\kappa,t})|\mathcal{I}|D^2_{max}\\
&-Q_B(\tau_{\kappa,t})\sum_{j\in\mathcal{J}} y^*_j(\tau_{\kappa,t})h_j(\tau_{\kappa,t}) - (t-\tau_{\kappa,t})|\mathcal{J}|^2y^2_{max}h^2_{max}. \qquad (31)
\end{aligned}$$

where the first inequality in (31) comes from the comparison procedure between the previous solutions and the current solutions in JGCA. From (30) and (31), we have:

$$\begin{aligned}
&\mathbb{E}\Big\{\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}^*(t)) - Q_B(t)\sum_{j\in\mathcal{J}} y^*_j(t)h_j(t)|\mathbf{Q}(t)\Big\}\\
&\ge \mathbb{E}\Big\{\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}'(t)) - Q_B(t)\sum_{j\in\mathcal{J}} y'_j(t)h_j(t)|\mathbf{Q}(t)\Big\}\\
&+\mathbb{E}\big\{(t-\tau_{\kappa,t})|\mathbf{Q}(t)\big\}\\
&\big(|\mathcal{I}|D^2_{max}+|\mathcal{J}|^2y^2_{max}h^2_{max}+|\mathcal{I}|D^2_{max}+|\mathcal{I}|B_{avg}y_{max}h_{max}\big). \qquad (32)
\end{aligned}$$

Recall that we assume that there is unknown distribution $p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}}$ drawn from the finite states of a tuple $(\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h})$. From the fact that $\mathbb{E}\{(t-\tau_{\kappa,t})|\mathbf{Q}(t)\} \le \frac{1}{\min\{p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}}\}\rho}$[19], we have:

$$\begin{aligned}
&\mathbb{E}\{\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}^*(t)) - Q_B(t)\sum_{j\in\mathcal{J}} y^*_j(t)h_j(t)|\mathbf{Q}(t)\}\\
&-\mathbb{E}\{\sum_{i\in\mathcal{I}} U_i(t)D_i(\boldsymbol{x}'(t),\boldsymbol{z}'(t)|\boldsymbol{y}'(t)) - Q_B(t)\sum_{j\in\mathcal{J}} y'_j(t)h_j(t)|\mathbf{Q}(t)\}\\
&\le \underbrace{\frac{1}{\min\{p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}}\}\rho}\big(2|\mathcal{I}|D^2_{max}+|\mathcal{J}|^2y^2_{max}h^2_{max}+B_{avg}|\mathcal{J}|y_{max}h_{max}\big)}_{R}. \qquad (33)
\end{aligned}$$

[18] The generality of the proof requires to lift this ergodicity assumption, which can be done using the analysis of [29]. We avoid this technicality for ease of exposition.

[19] since within time $\frac{1}{\min\{p_{\boldsymbol{\lambda},\boldsymbol{d},\boldsymbol{h}}\}\rho}$, at least one optimal solution $\boldsymbol{y}^*(t)$ for a tuple $(\boldsymbol{d}(t),\boldsymbol{\lambda}(t),\boldsymbol{h}(t))$ exists.

By plugging (24) in (33), this completes the proof.

Appendix C
Proof of Theorem 1

We begin this proof from the DMB bound in Eq. (15) without the objective function when we use the JGCA and GA algorithms as follows.

$$\begin{aligned}
&\Delta(L(t)) \le P - \mathbb{E}\big\{(B_{avg} - \sum_{j\in\mathcal{J}} y'_j(t)h_j(t))Q_B(t)|\mathbf{Q}(t)\big\}\\
&-\sum_{i\in\mathcal{I}}\mathbb{E}\big\{(D'_i(t)-\gamma'_i(t))U_i(t)|\mathbf{Q}(t)\big\}\\
&\le P{+}R + (1-1/e)\mathbb{E}\big\{\sum_{j\in\mathcal{J}} y^*_j(t)Q_B(t) - \sum_{i\in\mathcal{I}} D^*_i(t)U_i(t)|\mathbf{Q}(t)\big\}\\
&-\mathbb{E}\big\{B_{avg}Q_B(t) + \sum_{i\in\mathcal{I}}\gamma'_i(t)U_i(t)|\mathbf{Q}(t)\big\}\\
&= P{+}R - \sum_{i\in\mathcal{I}}\mathbb{E}\big\{U_i(t)((1-1/e)D^*_i(t)-\gamma'_i(t))|\mathbf{Q}(t)\big\}\\
&+\mathbb{E}\big\{Q_B(t)((1-1/e)\sum_{j\in\mathcal{J}} y^*_j(t)h_j(t) - B_{avg}|\mathbf{Q}(t)\big\}. \qquad (34)
\end{aligned}$$

The second inequality comes from the result of Lemma 5. Moreover, by summing (34) over $t = \{0,1,...T-1\}$, taking expectations in both sides, and dividing both sides with T, we have:

$$\begin{aligned}
\frac{\mathbb{E}\{L(T)\} - \mathbb{E}\{L(0)\}}{T} &\le P + R - \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{I}}\epsilon_i U_i(t)\\
&\le P{+}R{-}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{I}}\epsilon_{min}U_i(t), \qquad (35)
\end{aligned}$$

where ϵ_{min} denotes the minimum value among ϵ_i for all subareas $i \in \mathcal{I}$. Using $L(0) < \infty$, we have: $\sum_{i\in\mathcal{I}}\mathbb{E}\{U_i(t)\} \le \frac{P+R}{\epsilon_{max}}$ since $U_i(t) \ge 0$ for all time slots, $\epsilon_{max} \ge 0$ and whenever $\epsilon_i > 0$, $U_i(t)$, $\forall i \in \mathcal{I}$ are stable.

Second, to prove the virtual queue bounds, we consider DMB bound in Eq. (15) when we use the JGCA and GA algorithms as follows.

$$\begin{aligned}
&\Delta(L(t)) - V\sum_{i\in\mathcal{I}}\mathbb{E}\big\{\log(1+\gamma^*_i(t))|\mathbf{Q}(t)\big\}\\
&\le P - V\sum_{i\in\mathcal{I}}\mathbb{E}\big\{\log(1+\gamma^*_i(t))|\mathbf{Q}(t)\big\}\\
&-\mathbb{E}\big\{(B_{avg} - \sum_{j\in\mathcal{J}} y'_j(t)h_j(t))Q_B(t)|\mathbf{Q}(t)\big\}\\
&-\sum_{i\in\mathcal{I}}\mathbb{E}\big\{(D'_i(t)-\gamma'_i(t))U_i(t)|\mathbf{Q}(t)\big\}\\
&\le P{+}R{-}V\sum_{i\in\mathcal{I}}\mathbb{E}\big\{\log(1+\gamma^*_i(t))|\mathbf{Q}(t)\big\}\\
&+(1-1/e)\mathbb{E}\big\{\sum_{j\in\mathcal{J}} y^*_j(t)Q_B(t) - \sum_{i\in\mathcal{I}} D^*_i(t)U_i(t)|\mathbf{Q}(t)\big\}\\
&-\mathbb{E}\big\{B_{avg}Q_B(t) + \sum_{i\in\mathcal{I}}\gamma'_i(t)U_i(t)|\mathbf{Q}(t)\big\}\\
&= P{+}R{-}V\sum_{i\in\mathcal{I}}\mathbb{E}\big\{\log(1+\gamma^*_i(t))|\mathbf{Q}(t)\big\}\\
&+\sum_{i\in\mathcal{I}}\mathbb{E}\big\{U_i(t)(\gamma'_i(t)-(1-1/e)D^*_i(t))|\mathbf{Q}(t)\big\}\\
&+\mathbb{E}\big\{Q_B(t)((1-1/e)\sum_{j\in\mathcal{J}} y^*_j(t)h_j(t) - B_{avg})|\mathbf{Q}(t)\big\}. \qquad (36)
\end{aligned}$$

The second inequality comes from the result of Lemma 5. Then, by summing (34) over $t \in \{0,1,...,T-1\}$ and taking expectations in both sides, we have:

$$\begin{aligned}
&\mathbb{E}[||\mathbf{Q}(t)||^2]\\
&\le 2(P+R)T + 2TV(D_{max}-\theta^*) - \sum_{t=0}^{T-1}\sum_{i\in\mathcal{I}}\epsilon_{min}U_i(t). \qquad (37)
\end{aligned}$$

Since $U_i(t)$, $\forall i \in \mathcal{I}$ are stable, $U_i(t) < \infty$, $\forall i \in \mathcal{I}$. In addition, using the fact that $\mathbb{E}[||\mathbf{Q}(T)||]^2 \le \mathbb{E}[||\mathbf{Q}(t)||^2]$, and by dividing (37) with T^2 and taking a square root, this completes the proof of queue bound. Hence, the virtual queue is also stable.

Finally, to prove the sum utility performance of JGCA and GA algorithms, we rearrange (36), divide both sides by TV, and use the fact that $L(T) \ge 0$ to have:

$$\begin{aligned}
&\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{I}}\log(1+\gamma'_i(t))\\
&\qquad \ge \theta^* - \frac{P+R}{V} + \frac{1}{TV}\sum_{t=0}^{T-1}\sum_{i\in\mathcal{I}}\epsilon_i\mathbb{E}\{U_i(t)\}. \qquad (38)
\end{aligned}$$

Whenever $\epsilon_i > 0$, $\forall i \in \mathcal{I}$, by taking $T \to \infty$, this completes the proof of utility bound.

References

[1] J. Kwak, G. Paschos, and G. Iosifidis, "Dynamic cache rental and content caching in elastic wireless CDNs," in *Prof. of WiOpt*, May 2018, pp. 1–8.

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

[3] G. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE JSAC*, vol. 36, no. 6, pp. 1111–1125, 2018.

[4] "Amazon AWS." [Online]. Available: https://aws.amazon.com

[5] S. G. Shakkottai and R. Srikant, *Network optimization and control*. Now Publishers Inc, 2008.

[6] M. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, pp. 1–211, 2010.

[7] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *Proc. of IEEE INFOCOM*, Apr. 1998, pp. 533–539.

[8] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless networks with elastic and inelastic traffic," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 864–874, 2014.

[9] K. Naveen, L. Massoulie, E. Baccelli, A. Viana, and D. Towsley, "On the interaction between content caching and request assignment in cellular cache networks," in *Proc. of the 5th Workshop on All Things Cellular (ATC)*, 2015.

[10] K. Poularakis, G. Iosifidis, I. Pefkianakis, L. Tassiulas, and M. May, "Mobile data offloading through caching in residential 802.11 wireless networks," *IEEE Trans. on Network and Service Management*, vol. 13, no. 11, pp. 71–84, 2016.

[11] X. Wu, Q. Li, V. Leung, and P. Ching, "Joint long-term cache updating and short-term content delivery in cloud-based small cell networks," *IEEE Transactions on Communications*, vol. 68, no. 5, pp. 3173–3186, May 2020.

[12] X. Lyu, C. Ren, W. Ni, H. Tian, R. Liu, and X. Tao, "Distributed online learning of cooperative caching in edge cloud," *to appear, IEEE Transactions on Mobile Computing*, pp. 1–11, 2020.

[13] A. Asheralieva and D. Niyato, "Combining contact theory and Lyapunov optimization for content sharing with edge caching and device-to-device communications," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1213–1226, Jun. 2020.

[14] G. Paschos, A. Destounis, and G. Iosifidis, "Online convex optimization for caching networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 625–638, Apr. 2020.

[15] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. Tay, "A utility optimization approach to network cache design," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1013–1027, Jun. 2019.

[16] "Spatial multi-LRU: Distributed caching for wireless networks with coverage overlap," 2016. [Online]. Available: arXiv:1612.04363.[Online] .Available:https://arxiv.org/abs/1612.04363

[17] E. Leonardi and G. Neglia, "Implicit coordination of caches in small cell networks under unknown popularity profiles," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1276–1285, Jun. 2018.

[18] L. Chen, L. Song, J. Chakareski, and J. Xu, "Collaborative content placement among wireless edge caching stations with time-to-live cache," *IEEE Transactions on Multimedia*, vol. 22, no. 2, pp. 432–444, Feb. 2020.

[19] D. Carra, G. Neglia, and P. Michiardi, "Elastic provisioning of cloud caches: A cost-aware TTL approach," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1283–1296, Jun. 2020.

[20] B. N. Bharath, K. G. Nagananda, and H. V. Poor, "A learning-based approach to caching in heterogenous small cell networks," *IEEE Trans. on Communications*, vol. 64, no. 4, 2016.

[21] J. Xu, M. Schaar, J. Liu, and H. Li, "Forecasting popularity of videos using social media," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 2, 2015.

[22] J. Xiong, Y. Fang, P. Cheng, Z. Shi, and W. Zhang, "Distributed caching in converged networks: A deep reinforcement learning approach," *to appear, IEEE Transactions on Broadcasting*, pp. 1–11, 2020.

[23] A. Sadeghi, F. Sheikholeslami, A. Marques, and G. Giannakis, "Reinforcement learning for adaptive caching with dynamic storage pricing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 10, pp. 2267–2281, Oct. 2019.

[24] A. Sadeghi, F. Sheikholeslami, and G. B. Giannakis, "Optimal and scalable caching for 5g using reinforcement learning of space-time popularities," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 180–190, Feb. 2018.

[25] S. O. Somuyiwa, A. Gyorgy, and D. Gunduz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.

[26] K. Son, H. Kim, Y. Yi, and B. Krishnamachari, "Base station operation and user association mechanisms for energy-delay tradeoffs in green cellular networks," *IEEE JSAC*, vol. 29, no. 8, pp. 1525–1536, Sep. 2011.

[27] "CAISO: California independent system operator." [Online]. Available: http://www.caiso.com/

[28] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 556–567, Oct. 2000.

[29] M. Neely, "Energy optimal control for time varying wireless networks," *IEEE Trans. on Inform. Theory*, vol. 52, no. 7, pp. 2915–2934, Jul. 2006.

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

[31] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundation and Trends in Networking*, vol. 1, no. 1, pp. 1–149, 2006.

[32] M. Dehghan *et al.*, "On the complexity of optimal routing and content caching in heterogeneous networks," in *Proc. of IEEE INFOCOM*, Apr. 2015, pp. 936–944.

[33] J. Kwak, L. Le, H. Kim, and X. Wang, "Two time-scale edge caching and BS association for power-delay tradeoff in multi-cell networks," *IEEE Transactions on Communications*, vol. 67, no. 8, pp. 5506–5519, Aug. 2019.

[34] H. Ju, J. Li, Y. Long, and X. Yang, "Adaptive cross-network cross-layer design in heterogeneous wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 14, no. 2, pp. 655–669, 2015.

[35] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[36] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network - Measurements, models, and implications," *Elsevier Computer Networks*, vol. 53, no. 4, pp. 501–514, Mar. 2009.

[37] J. Kwak, Y. Kim, L. Le, and S. Chong, "Hybrid content caching in 5G wireless networks: Cloud versus edge caching," *IEEE Transactions on Wireless Communications*, vol. 17, no. 5, pp. 3030–3045, May 2018.

[38] "IESO: Independent electricity system operator in canada." [Online]. Available: http://www.ieso.ca/en

[39] "Amazon elastic CDN service - ElastiCache." [Online]. Available: https://aws.amazon.com/elasticache/pricing

[40] "3GPP, TR 36.814, V9.0.0, Further advancements for E-UTRA physical layer aspects (Release 9)," 2010.

[41] "Crowd-sourced cellular tower and coverage mapping service." [Online]. Available: https://www.cellmapper.net

[42] C. Wang, J. Llorca, A. Tulino, and T. Javidi, "Dynamic cloud network control under reconfiguration delay and cost," *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 491–504, Apr. 2019.

Jeongho Kwak (M'15) is currently the Assistant Professor with the Department of Information and Communication Engineering, DGIST, Daegu, South Korea. Prior to joining DGIST, he was with INRS-EMT, Montreal, Canada and Trinity College Dublin, Dublin, Ireland, as a Postdoctoral Researcher and a Marie Sklodowska-Curie Fellow, respectively. His current research interests lie on learning model & resource allocation in hybrid cloud/edge network architecture, energy optimization in heterogeneous networks, and radio resource management for 5G wireless cellular networks. He received the B.S. degree (*summa cum laude*) in electrical and computer engineering from Ajou University, Suwon, South Korea, in 2008, and the M.S. and Ph.D. degrees in electrical engineering from the KAIST, Daejeon, South Korea, in 2011 and 2015, respectively.

Georgios Paschos is is a Senior Manager, Research Science at Amazon.com, leading the EU Operation Research team of Amazon Transportation Services. Previously, he worked 5 years ('14-'19) as a principal scientist at Huawei Technologies, Paris, leading the Network Control and Resource Allocation team. Dr. Paschos was at LIDS, MIT ('12-'14) and has held positions at CERTH-ITI, Greece '08-'12 (researcher), University of Thessaly, '09-'11 (adjunct lecturer) and VTT, Finland, '07-'08 (ERCIM Postdoc Fellow). He received his diploma in Electrical and Computer Engineering in 2002 from Aristotle University of Thessaloniki, and his PhD degree in Wireless Networks 2006 from ECE dept. University of Patras, Greece. Two of his papers won best paper awards in GLOBECOM 2007 and IFIP Wireless Days 2009. He has served as an associate editor for IEEE/ACM Trans. on Networking ('15-'19), IEEE Networking Letters ('18-'19), and as a TPC member of INFOCOM, WiOPT, and Netsoft. He has organized several international workshops on the topics of caching, network slicing and machine learning techniques for communication systems, and was the co-organizer and editor of the IEEE JSAC Special Issue on Caching for Comm. Systems and Networks.

George Iosifidis is an Assistant Professor with Delft University of Technology. He received a Diploma in Electronics and Telecommunications Engineering from the Greek Air Force Academy (Athens, 2000), and a PhD degree from the ECE Dep., University of Thessaly in 2012. He was a Postdoctoral researcher ('12-'14) at CERTH and at Yale University ('14-'17); and an Assistant Prof. with Trinity College Dublin ('16-'20). His research interests lie in the broad area of network optimization and economics, and his work has appeared in Nature Communications, Nature Human Behavior and PNAS. He is a co-recipient of the best paper awards in IEEE WiOPT 2013 and IEEE INFOCOM 2017, has served as a guest editor for the IEEE Journal on Selected Areas in Communications, and is currently an editor for IEEE Transactions on Communications and IEEE/ACM Transactions on Networking.