Caching Policies for Delay Minimization in Small Cell Networks with Joint Transmissions

Guilherme Ricardo[1,2], Giovanni Neglia[1], and Thrasyvoulos Spyropoulos[2]

[1]Inria Université Côte d'Azur, Sophia Antipolis, France
[2]EURECOM, Communication Systems Department, Sophia Antipolis, France
Emails: {guilherme.ricardo, thrasyvoulos.spyropoulos}@eurecom.fr, giovanni.neglia@inria.fr

***Abstract*—In 5G and beyond network architectures, operators and content providers base their content distribution strategies on Heterogeneous Networks, where macro and small(er) cells are combined to offer better Quality of Service (QoS) to wireless users. On top of such networks, edge caching and Coordinated Multi-Point (CoMP) transmissions are used to further improve performance. The problem of optimally utilizing the cache space in dense and heterogeneous cell networks has been extensively studied under the name of "FemtoCaching." However, related literature usually assumes relatively simple physical layer (PHY) setups and known or stationary content popularity. In this paper, we address these issues by proposing a class of fully distributed and dynamic caching algorithms that take advantage of CoMP capabilities towards minimizing PHY-aware metrics, such as end-to-end (E2E) delay. Our policies outperform existing dynamic solutions that are PHY-unaware, under both synthetic and real (non-stationary) request processes, and converge to efficient centralized solutions, in static setups.**

***Index Terms*—Edge caching, joint transmission, CoMP, heterogeneous cellular networks, distributed algorithms.**

I. Introduction

With the ever-growing popularization of social media and on-demand video streaming, cellular data consumption has experienced an unprecedented increase. According to latest CISCO's forecast [1], by 2022 mobile data traffic will increase nearly three times compared to 2017. Network densification is considered a key strategy to cope with the traffic deluge in future networks [2]. The standard 3G/4G macro-cell topology will be enriched by a large number of overlapping and often heterogeneous small(er) cells (e.g., femto, pico), in order to improve both coverage and capacity.

On top of such a densified network, two additional techniques have been considered to provide high QoS: (i) *caching* popular files directly at the base stations, henceforth referred to as "helpers," in order to reduce access latency and backhaul congestion; and (ii) *coordinated transmissions* or CoMP [3], i.e., jointly transmitting the same file from nearby helpers, in order to improve PHY performance.

Caching at the edge to relieve (expensive) bandwidth resources has recently attracted a large amount of research work. In the popular "FemtoCaching" model [4], helpers with partially overlapping coverage can be seen as a distributed cache. In such a dense setting, requests posed by a user equipment (UE) can be satisfied by any helper in its communication range. The idea is to find the file allocation, i.e., which file to cache at each helper, that can optimize the hit rate, assuming known and stationary file popularities. The authors in [4] prove that this problem is NP-Hard. However, as the objective is submodular, there exist polynomial algorithms with approximation guarantees.

A number of follow-up works studying variations of the basic problem have appeared in the last few years, e.g. [5], [6], [7]. However, the majority of these works suffer from three key shortcomings: (i) they focus only on cache hit rate as the performance metric and do not explicitly consider CoMP capabilities; (ii) the proposed algorithms are *centralized*, requiring a central entity aware of the entire topology to take global decisions about the allocation; and (iii) popularities of all files are assumed to be *known* and *static*.

There are a few exceptions in the literature, [8], [9], [10], that address some of these issues. Targeting shortcoming (i), the femto-caching optimization framework has been studied by [8], [9], assuming CoMP capabilities. The former revisits randomized caching policies for CoMP and considers clusters in a multi-user MIMO setting. The latter proves that submodularity is preserved when the objective is E2E delay minimization and thus the problem can be efficiently approximated. However, these algorithms still assume centralized and static policies. On the other hand, the authors of [10] tackle points (ii) and (iii) above, proposing dynamic and decentralized algorithms without a priori knowledge of file popularities or caching network topology. However, they consider the standard femto-caching problem and do not exploit CoMP.

This paper proposes efficient algorithms that simultaneously address all these concerns. Our main contributions are summarized as follows:

- We propose novel distributed CoMP-aware caching policies that are executed independently by each helper, without knowledge of global network topology or file popularities.
- We show that these algorithms have desirable convergence properties to locally optimal allocations.

This work has been supported by the French government, through the EUR DS4H Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-17-EURE-0004 as well as the "5C-for-5G" JCJC project with the reference number ANR-17-CE25-0001.

- Using a real topology and real traffic trace, we show that our policies reduce file retrieval time up to 23% in comparison to existing dynamic schemes.

II. Network Model

Consider H helpers arbitrarily located in a given region R. Each helper has a cache that can store C files from a catalog of F files. Similarly to [9], [10], in order to ease our exposition, we assume that all files have the same size equal to M bits. This is a reasonable assumption given that large files are often split into smaller chunks of equal size. It is possible to adapt our algorithms and our analysis to the heterogeneous file size case.

Let $X_f^{(h)} \in \{0,1\}$ indicate whether helper h caches a copy of file f ($X_f^{(h)} = 1$) or not ($X_f^{(h)} = 0$). Then the $H \times 1$ vector $\mathbf{X}_f = (X_f^{(1)}, \dots X_f^{(H)}) \in \{0,1\}^H$ specifies the distribution of file f across all caches, and the $H \times F$ matrix $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, \dots \mathbf{X}_F)$ the allocation of all files in the network. Let $\mathbf{e}^{(h)}$ be the $H \times 1$ vector with a 1 in position h and all other components equal to 0. We write $\mathbf{X}_f \oplus \mathbf{e}^{(h)}$ to indicate a new placement where a copy of file f is added to helper h, if not already present. Similarly, $\mathbf{X}_f \ominus \mathbf{e}^{(h)}$ indicates a new allocation where file f is removed from helper h's cache, if present.

There are U UEs spread over R, who can request files from the catalog.[1] We assume that the aggregate request process follows the IRM model: each request is for file f with probability p_f independently from the past, where $p_1 \geq p_2 \geq \dots \geq p_F$. Moreover, each UE is equally likely to have generated the request. We will also consider non-stationary traffic demand later in Section V.

Because of the high density of helpers, each UE u will, in general, be within communication range of multiple helpers. We denote by I_u the set of helpers covering UE u and by $g_{h,u}$ the signal-to-noise ratio (SNR) of the wireless channel between helper h and UE u. If multiple helpers in $A \subseteq I_u$ cooperate to transmit the same file to u, we assume they are able to achieve the aggregate channel capacity $W \log_2(1 + \sum_{h \in A} g_{h,u})$, where W is the channel bandwidth [11], [8].

We assume the following operation upon every request: UE u broadcasts an inquiry message for file f that is received by its neighboring helpers in I_u. Among these helpers, a subset $J_{u,f} \subseteq I_u$ is actually caching file f. There are then two different modalities for the transmission of file f to UE u:

- If $J_{u,f} \neq \emptyset$ (hit at some cache), the file f can be directly transmitted through the wireless channel. In particular, if $|J_{u,f}| > 1$, the helpers cooperate to jointly transmit the file and the UE experiences the radio access delay:

$$d_r(u, \mathbf{X}_f) = \frac{M}{W \log_2(1 + \sum_{h \in I_u} g_{h,u} X_f^{(h)})} \quad (1)$$

- If $J_{u,f} = \emptyset$ (miss), a helper $h^* \in I_u$, chosen according to some function (e.g., the helper with the highest SNR), downloads f from the back-end server and then transmits it to u. In this case, u experiences a delay consisting of (i) the time to fetch the file through the backhaul network, hereafter denoted by d_B, plus (ii) the time for h to transmit it back to u, referred to as the radio access delay upon a cache miss, which is defined by:

$$d_r(u, \mathbf{X}_f) = \frac{M}{W \log_2(1 + g_{h^*,u})} \quad (2)$$

The total E2E delay UE u experiences to download file f is then:

$$d_f(u, \mathbf{X}_f) = d_B \mathbb{1}_{J_{u,f} = \emptyset} + d_r(u, \mathbf{X}_f), \quad (3)$$

where $\mathbb{1}_c$ is the indicator function denoting if the condition c is satisfied or not.

III. Problem Definition

Our goal is to design an online caching policy that minimizes the expected delay. Given a file allocation $\mathbf{X}$ in the network, the expected delay for a request is:

$$\bar{d}(\mathbf{X}) = \sum_{f=1}^{F} \frac{1}{U} \sum_{u=1}^{U} p_f \cdot d_f(u, \mathbf{X}_f), \quad (4)$$

where we average over all users (uniformly) and all files (according to popularity p_f). We start by considering the static optimization problem, that can be formalized by the following integer programming problem:

$$\begin{aligned} \text{minimize} \quad & \bar{d}(\mathbf{X}) \\ \text{subject to} \quad & \sum_{f=1}^{F} X_f^{(h)} = C, \ h \in [H] \\ & X_f^{(h)} \in \{0,1\}, h \in [H], f \in [F] \end{aligned} \quad (5)$$

where $[n]$ denotes the set $\{1, 2, ..., n\}$.

This is the delay-minimization version of the femto-caching problem [4] under stationary request process and CoMP. It was studied for the first time in [9], under the *homogeneous SNR* assumption, i.e., all the SNRs are equal ($g_{h,u} = g$, for all (h,u) channels). In this scenario, the total E2E delay becomes a function of the number of copies of f available at u's neighboring caches, denoted by $k_{u,f} = 0, 1, ..., H$. Thus, we can rewrite Equation (3) as follows:

$$d(k_{u,f}) = d_B \mathbb{1}_{k_{u,f}=0} + \frac{M}{W \log_2(1 + (k_{u,f} + \mathbb{1}_{k_{u,f}=0}) g)} \quad (6)$$

Problem (5) was proven to be NP-Hard, even under this simpler homogeneous SNR assumption. At the same time, if $d(1) \leq d_B$, (5) is equivalent to maximizing a monotone and submodular function with a matroid constraint.[2] This equivalent problem can be solved using a greedy (centralized) algorithm (hereafter referred to as GreedyAD) with a guaranteed $1/2$-approximation ratio [9].

[1] We assume each request is allocated an equal amount of resource blocks; scheduling issues or other advanced interference management schemes are subject of future work.

[2] The new objective corresponds to the reduction of delay in comparison to the case when each request is a miss (caches are empty).

A. Optimal Allocation in Full-Coverage Scenario

In order to get initial insights on Problem (5), we first study the *full-coverage* scenario, which considers two additional assumptions: (i) all SNRs are equal and (ii) each UE u can connect to all helpers ($I_u = [H]$, for all u), so the network has an aggregate cache capacity of HC files.

If CoMP capabilities are not considered (i.e., classic femto-caching problem), the optimal solution is to store the HC most popular files across all caches [4]. This is not necessarily the case when joint transmissions are allowed, and the goal is to minimize the average delay (4). The network parameters, g and d_B, determine together how the average delay can be reduced by CoMP and then influence the optimal allocation.

We first observe that, in the full-coverage scenario, it is possible to compute the *optimal* allocation in polynomial time:

Proposition III.1. *In the full-coverage scenario, if $d(1) \leq d_B$, an allocation provided by the greedy algorithm is optimal.*

The proof relies on mapping problem (5) to a knapsack problem with $F \times C$ objects with unit size for which the greedy algorithm is optimal.

We then characterize, for the full-coverage scenario, the necessary and sufficient conditions (NSCs) for the optimal allocation to be one of the two extreme ones: *full-diversity* (one copy of each of the HC most popular file is stored in the network), and *full-replication* (the C most popular files are cached in each one of the H helpers).

Proposition III.2. *In the full-coverage scenario, if $d(1) \leq d_B$, full-diversity is an optimal allocation if and only if*

$$d_B \geq D_{FD} \triangleq \frac{p_1}{p_{HC}}(d(1) - d(2)), \tag{7}$$

and full-replication is an optimal allocation if and only if

$$d_B \leq D_{FR} \triangleq \frac{p_C}{p_{C+1}}(d(H-1) - d(H)). \tag{8}$$

Proof. In the full-coverage scenario, an allocation is optimal iff it is not possible to replace any file in a cache and reduce the expected delay $\bar{d}$. Let us consider first the full-diversity allocation. It is evident that it cannot be advantageous to replace one of the HC most popular files with a less popular file $j > HC$. The full-diversity allocation is then optimal iff it is not worthy to replace any file $i \in [HC]$ with an additional copy of a file $j \in [HC] \setminus \{i\}$. This is the case iff

$$p_i \cdot d_B \geq p_j(d(1) - d(2)), \forall i \in [HC], j \in [HC] \setminus \{i\},$$

i.e., the delay increase due to the cost to retrieve i through the backhaul is larger than the delay decrease due to the possibility to have two helpers jointly transmitting j. The minimum of the left-hand side of the inequality above is achieved when $i = HC$ (least popular file), and the maximum of the right-hand side is achieved when $j = 1$ (most popular file). Then, the set of inequalities above is satisfied iff

$$p_{HC} \cdot d_B \geq p_1 \left(d(1) - d(2)\right),$$

i.e., we can restrain to consider the possibility to replace the least popular of the HC files with an additional copy of the most popular file 1.

The reasoning for the full-replication allocation is similar: in this case we need to ensure that replacing one of the H copies of file C with (a first copy of) file $C+1$ does not reduce the expected delay, i.e.

$$p_{C+1} \cdot d_B \leq p_C(d(H-1) - d(H)).$$

□

Although Proposition III.2 does not hold for generic topologies, we can still derive new conditions for *local optimality*, defined as follows:

Definition 1. *A cache allocation $\mathbf{X}$ is locally optimal, if it does not exist another allocation $\mathbf{X}'$ such that $\bar{d}(\mathbf{X}') < \bar{d}(\mathbf{X})$, where $\mathbf{X}'$ differs from $\mathbf{X}$ only by a single file at a single cache.*

The following Corollary then holds:

Corollary 1. *Assuming homogeneous SNRs, (i) Equation (7) is a **necessary condition** for the full-diversity allocation to be locally optimal and (ii) Equation (8) is a **sufficient condition** for the full-replication allocation to be locally optimal.*

The logic of Corollary 1's proof is similar to Proposition III.2, and we omit it due to space limitations. Intuitively, as we move from a full-coverage to a less dense scenario, the optimal allocation replicates more the most popular files. Correspondingly, the NSC for full-diversity in Proposition III.2 becomes only a necessary condition, while the NSC for full-replication becomes a sufficient one.

As an application of the results above, Fig. 1 shows, for a full-coverage scenario, regions of the parameter space (g, d_B), where full-replication and full-diversity are optimal. The boundary curves, $D_{FD}(g)$ (blue dotted line) and $D_{FR}(g)$ (green dashed curve), are shown as functions of the SNR g. Any pair (g, d_B) above $D_{FD}(g)$ will result in a full-diversity allocation. However, any combination taken below $D_{FR}(g)$ will result in a full-replication allocation.

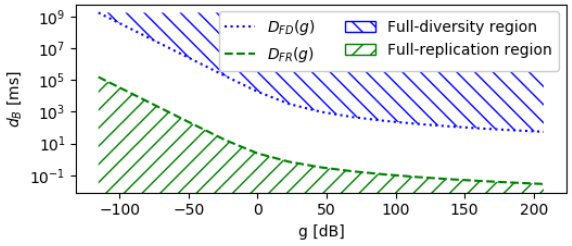

Fig. 1: Boundaries of (d_B, g) for extreme allocations: $H = 10$, $\alpha = 1.5$, $F = 10^6$, and $C = 100$. d_B axis is in log scale.

IV. Online Caching Policies

In what follows, we will consider the marginal gain for adding a copy of file f at helper h defined as:

$$\Delta d_f^{(h)}(u, \mathbf{X}_f) \triangleq d_f(u, \mathbf{X}_f \ominus \mathbf{e}^{(h)}) - d_f(u, \mathbf{X}_f) \tag{9}$$

We introduce the qLRU-Δd caching policy in Algorithm 1. It is a variant of qLRU [12] in which cache updates take into account the marginal delay reduction $\Delta d_f^{(h)}(u, \mathbf{X}_f)$. Under stationary request process, qLRU-Δd minimizes the expected delay when q converges to 0 (the formal result is stated in Proposition IV.1). The policy is inspired by qLRU-Lazy in [10], that is shown to provide similar guarantees for hit rate.

Algorithm 1 (qLRU-Δd)**.** Assume UE u has requested file f. The request is served according to the general operation described in Section II and caches update their state as follows:

- each helper $h \in J_{u,f}$ moves the file to the front of the cache with probability $\rho_f^{(h)}$ proportional to the marginal gain of keeping its local copy, i.e.
$$\rho_f^{(h)}(u, \mathbf{X}_f) = \beta \cdot \Delta d_f^{(h)}(u, \mathbf{X}_f), \tag{10}$$
where the constant β guarantees that $\rho_f^{(h)}$ is indeed a probability.
- each helper $h \in I_u \setminus J_{u,f}$ decides to store an additional copy of f with probability
$$q_f^{(h)}(u, \mathbf{X}_f) = q \cdot \sigma_f^{(h)}(u, \mathbf{X}_f), \tag{11}$$
where
$$\sigma_f^{(h)}(u, \mathbf{X}_f) = \gamma \cdot \Delta d_f^{(h)}(u, \mathbf{X}_f \oplus \mathbf{e}^{(h)}), \tag{12}$$
γ plays the same role of β above and $q \in (0, 1]$ is a dimensionless parameter.

Remark 1. In our experiments, we take simply
$$\beta = 1/(\max_{f,h,u,\mathbf{X}_f} \Delta d_f^{(h)}(u, \mathbf{X}_f))$$
$$\gamma = 1/(\max_{f,h,u,\mathbf{X}_f} \Delta d_f^{(h)}(u, \mathbf{X}_f \oplus \mathbf{e}^{(h)})).$$

From (1), (2), and (9) we see that probabilities $\rho_f^{(h)}$ and $q_f^{(h)}$ depend on the allocation of file f at nearby helpers or, more precisely, on 1) the aggregate SNR all helpers in $J_{u,f}$ can achieve when transmitting to u (i.e. $\sum_{h' \in J_{u,f}} g_{h',u}$), 2) the SNR $g_{h,u}$ of the local channel from h to u, and 3) the backhaul delay d_B. In cellular networks, each UE takes SNR measurements of helpers within range [13]. Thus, the information needed to set $q_f^{(h)}$ and $\rho_f^{(h)}$ can be piggybacked in uplink communication from UE u to the helpers with limited overhead. Note that this exchange only provides limited information about the local (immediate) topology, which is significantly simpler than the global knowledge required by standard femto-caching schemes.

Proposition IV.1. *Under IRM request traffic, Che's [14], [15], and exponentialization [10] approximations, a network of qLRU-Δd caches asymptotically achieves a locally-optimal caching configuration in the sense of Definition 1 when $q \to 0$.*

Proposition IV.1 derives from a more general result [16, Prop. IV.1]. We provide here an intuitive explanation of why qLRU-Δd is optimal.

Intuition. We observe that, as q converges to 0, the cache exhibits two different dynamics with very different timescales: the *insertion of new files* tends to happen more and more rarely ($q_f^{(h)}$ converges to 0), while the frequency of *position updates* for files already in the cache is unchanged ($\rho_f^{(h)}$ does not depend on q). A file f at cache h is moved to the front with a probability proportional to $\Delta d_f^{(h)}(u, \mathbf{X}_f)$, i.e. proportional to how much the file contributes to reduce the delay of that specific request. This is a very noisy signal: upon a given request, the file is moved to the front or not. At the same time, as q converges to 0, more and more moves to the front occur between any two file evictions. The expected number of moves-to-the-front file f experiences is proportional to 1) how often it is requested (p_f) and 2) how likely it is to be moved to the front upon a request ($\rho_f^{(h)}$). Overall, the expected number of moves is proportional to $p_f \cdot \Delta d_f^{(h)}(\mathbf{X}_f)$, i.e. its average contribution to the decrease of the expected delay. By the law of large numbers, the random number of moves-to-the-front will be close to its expected value and it becomes likely that the least valuable file in the cache occupies the last position.

We can then think that, when a new file is inserted in the cache, it will replace the file that contributes the least to the decrease of the expected cost. qLRU-Δd then behaves as a random greedy algorithm that, driven by the request process, progressively replaces the least useful file from the cache, until it reaches a local minimum. □

In Section V, we observe that, under IRM request process, the smaller q is, the closer qLRU-Δd's delay is to the GreedyAD, as it is stated by Proposition IV.1.

Remark 2. The probability $q_f^{(h)}$ is analogous to q in the usual qLRU policy. We note that setting $q_f^{(h)} = q$, i.e. a constant value, suffices to prove the asymptotic convergence of the policy (Prop. IV.1). However, we propose (11) to make it more likely to add new copies to helpers bringing a larger marginal gain $\Delta d_f^{(h)}(u, \mathbf{X}_f \oplus \mathbf{e}^{(h)})$. This can speed up the transient dynamics of the policy.

While qLRU-Δd converges to a local optimum for static popularities, in practice, the slow insertion process of qLRU-Δd, for small values of q, becomes problematic when some files are popular over short time scale: a new file gets a chance to be inserted in the cache every $1/q$ requests, and by that time, its popularity may have declined. In order to gain in reactivity, we propose 2LRU-Δd, whose operation is described in Algorithm 2. 2LRU-Δd maintains a virtual cache (with file's identification data or, simply, ID) and a physical cache (actual data). A given file is inserted in the physical cache only if its ID is already in the virtual cache. 2LRU-Δd does not enjoy the same theoretical guarantee as qLRU-Δd, but, thanks to its reactivity, it provides even smaller delays under real request processes with strong temporal locality [17].

Algorithm 2 (2LRU-Δd)**.** Assume UE u has requested file f. Let $\hat{J}_{u,f}$ be the subset of helpers storing the file's IDs in the virtual cache. The request is served according to the general

operation described in Section II and caches update their state as follows:

- If $h \in \hat{J}_{u,f}$, move f's ID to the front of h's virtual cache and,
  - if $h \in J_{u,f}$, move f to the front of h's physical cache with probability $\rho_f^{(h)}(u, \mathbf{X}_f)$;
  - else, evict the file in the physical cache's last position and insert f.
- If $h \notin \hat{J}_{u,f}$, with probability $\sigma_f^{(h)}(u, \mathbf{X}_f)$, evict the ID in h's virtual cache's last position and insert f's ID

V. Numerical Results

In our experimental approach, we first confirm qLRU-Δd policy's optimality as q tends to 0. Then, we compare it to other policies over real network topologies and for different request processes. Finally, we consider a heterogeneous SNR scenario and observe the effect of SNR variability on policies' performance.

A. qLRU-Δd Convergence Analysis

First, consider $H = 2$ helpers, whose cells partially overlap, with storage capacity for $C = 100$ files out of a catalog adding up to $F = 10^6$ files. Files popularity distribution follows a Zipf law with exponent $\alpha = 1.2$. We start considering $d_B = 100$ms and homogeneous SNR regime: every UE-helper channel has the same SNR $g = 10$dB. In all our experiments, policies' simulations have a warm up phase and a measurement phase each consisting of 10^8 requests. Moreover, we fix the files size to $M = 1.0$Mbits and the channel bandwidth to $W = 5.0$MHz. Our comparison baseline is GreedyAD, i.e., the centralized delay-minimization greedy algorithm that assumes network topology and files popularities are known [9]. Upon a miss, h^* is chosen uniformly at random among all helpers in I_u.

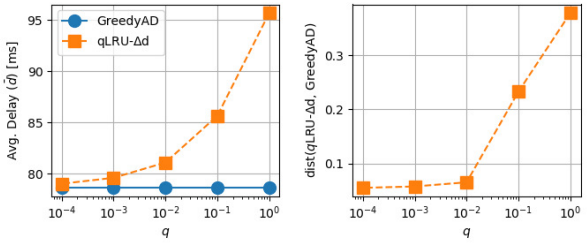

Fig. 2: Convergence analysis: delay (left) and allocation (right) convergence with q, for $\alpha = 1.2$, $d_B = 100$ms, and $g = 10$dB.

Fig. 2 (left) shows the average delay of qLRU-Δd and GreedyAD for different values of q. As q decreases, qLRU-Δd's delay converges to GreedyAD. Fig. 2 (right) shows the distance between qLRU-Δd and GreedyAD. The distance dist(P_1, P_2) between policies P_1 and P_2 is defined as the cosine distance[3] between their occupancy vectors $\boldsymbol{\theta}$, which are $F \times 1$ vectors containing the fraction of time every file spent in cache during measurement phase. For GreedyAD, we define $\boldsymbol{\theta}$ as its resulting allocation matrix. We observe a reduction of the distance between qLRU-Δd and GreedyAD, indicating that the files GreedyAD stores tend to be cached longer and longer under qLRU-Δd as q decreases.

[3]The cosine distance between vectors u and v is given by dist$(u, v) = 1 - \frac{\langle u,v \rangle}{\|u\|_2 \|v\|_2}$, where $\langle \cdot, \cdot \rangle$ denotes the inner product.

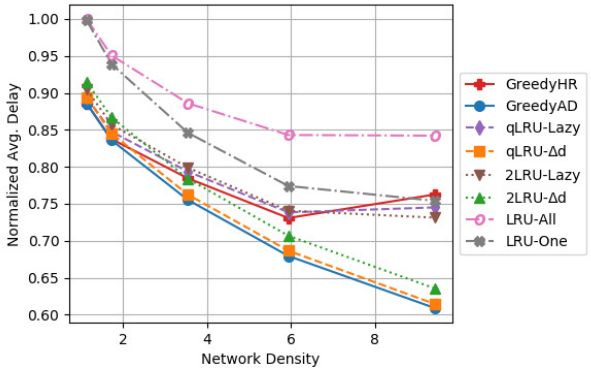

Fig. 3: Performance analysis of various policies in a real topology with IRM request process ($\alpha = 1.2$)

B. Performance Analysis

We compare the performance of our proposed policies, qLRU-Δd and 2LRU-Δd, to other policies from the literature that try to achieve coordination across caches: qLRU-Lazy and 2LRU-Lazy from [10], LRU-All and LRU-One from [18]. As a comparison reference, we provide the results from GreedyAD and the greedy hit rate maximization algorithm for classic femto-caching problem [4] (GreedyHR). We consider a topology where $H = 10$ helpers are located according to the positions of T-mobile base stations in Berlin extracted from [19]. We vary the network density by adjusting helpers' coverage area from 25m to 200m, assuming constant spatial user density, such that the expected number of helpers covering a UE is between 1.1 and 9.4. For all helpers, the SNR is $g = 10$dB and backhaul access delay is $d_B = 100$ms.

In Fig. 3, we show the average delay for an IRM request process ($\alpha = 1.2$). The qLRU-Δd results are very close to GreedyAD, confirming its convergence across different topologies. qLRU-Δd reaches performance gains of up to 20% related to GreedyHR and other hit rate maximization policies. If compared to simpler policies, such as LRU-All and LRU-One, qLRU-Δd achieves performance gains of up to 27%.

In Fig. 4, we consider a request process based on a real trace from Akamai Content Delivery Network [20]. Requests were observed for a period of 5 consecutive days. The greedy allocation in this case was determined by estimating the files popularities during this period. However, real request processes exhibit strong temporal locality features. Static allocations are based on time-average popularities that smooth out the variability over short time scale. On the contrary, 2LRU-like policies are highly reactive and may be able to capture short time popularities variations, promising smaller delays than qLRU-Δd and GreedyAD. The figure shows that

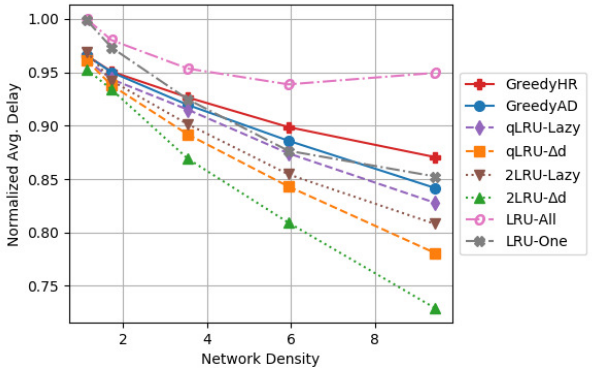

Fig. 4: Performance analysis of various policies in a real topology with Akamai trace.

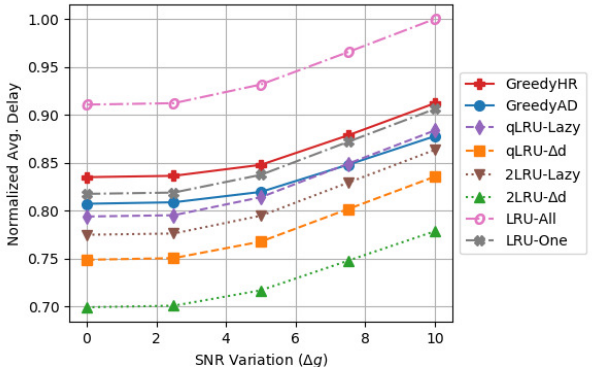

Fig. 5: Heterogeneous SNRs: Berlin topology with density 9.4, $g_0 = 10.0$dB, and $d_B = 100.0$ms with Akamai trace.

indeed 2LRU-Δd outperforms both GreedyAD and qLRU-Δd by 12% and 6%, respectively. Moreover, 2LRU-Δd provides performance gains of around 15% in comparison with 2LRU-Lazy and 23% in comparison with LRU-All.

C. Heterogeneous SNR

We use the same Berlin topology (with density of 9.4 helpers/UE, in average) but now, at every request (u, f), the SNRs $g_{h,u}, \forall h \in I_u$ are chosen uniformly at random within a range, i.e., $g_{h,u} \in [g_0 - \Delta g, g_0 + \Delta g]$, where g_0 is the base SNR and Δg its variation. We fix $g_0 = 10$dB and $d_B = 100$ms and observe the average delay of various policies for different values of Δg. Files are also requested according to Akamai's trace.

In Fig. 5, we observe that, for all tested policies, the average delay tends to increase. This fact is explained by Jensen's inequality, since the delay is now a convex random function: given that $g = g_0 + \Delta g$ and $g' = g_0 - \Delta g$, the delay reduction achieved with the larger g is smaller than the delay increase due to g'. The relative performance gains between the policies are almost constant across the different SNR variations and similar to the previous experiment (Fig. 4).

VI. Conclusions

In this paper we have presented two novel distributed caching policies able to optimize the average delay under unknown and dynamic request process. We study how the transmission parameters influence the optimal allocations in delay-minimization problem and how they differ from the traditional femto-caching solutions. In our experiments, we observe qLRU-Δd's convergence and evaluate both proposed policies performances under different request processes and SNR regimes. We show that such policies achieve considerable performance gains with negligible additional deployment complexity.

References

[1] CISCO, "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021 white paper," CISCO, Tech. Rep., Feb 2017.

[2] N. Bhushan *et al.*, "Network densification: the dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, Feb. 2014.

[3] D. Lee, H. Seo, B. Clerckx, E. Hardouin, D. Mazzarese, S. Nagata, and K. Sayana, "Coordinated multipoint transmission and reception in LTE-advanced: deployment scenarios and operational challenges," *IEEE Communications Magazine*, vol. 50, no. 2, pp. 148–155, Feb. 2012.

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

[5] T. Wang, L. Song, and Z. Han, "Dynamic femtocaching for mobile users," in *2015 IEEE WCNC*. IEEE, 2015, pp. 861–865.

[6] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 142–149, April 2013.

[7] P. Sermpezis, T. Spyropoulos, L. Vigneri, and T. Giannakas, "Femto-caching with soft cache hits: Improving performance with related content recommendation," in *IEEE GLOBECOM 2017*. IEEE, 2017, pp. 1–7.

[8] W. C. Ao and K. Psounis, "Distributed caching and small cell cooperation for fast content delivery," in *MobiHoc*. ACM, 2015, pp. 127–136.

[9] A. Tuholukova, G. Neglia, and T. Spyropoulos, "Optimal cache allocation for Femto helpers with joint transmission capabilities," in *IEEE ICC 2017, 21-25 May 2017, Paris, France*, Paris, FRANCE, 05 2017.

[10] 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, June 2018.

[11] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[12] M. Garetto, E. Leonardi, and V. Martina, "A unified approach to the performance analysis of caching systems," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, no. 3, pp. 12:1–12:28, May 2016.

[13] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*. Wiley, 2011.

[14] H. Che, Y. Tung, and Z. Wang, "Hierarchical Web caching systems: modeling, design and experimental results," *Selected Areas in Communications, IEEE Journal on*, vol. 20, no. 7, pp. 1305–1314, Sep 2002.

[15] R. Fagin, "Asymptotic miss ratios over independent references," *Journal of Computer and System Sciences*, vol. 14, no. 2, pp. 222 – 250, 1977.

[16] G. Neglia, E. Leonardi, G. Iecker, and T. Spyropoulos, "A Swiss Army Knife for Dynamic Caching in Small Cell Networks," 2019, arXiv:1912.10149.

[17] S. Traverso *et al.*, "Temporal Locality in Today's Content Caching: Why It Matters and How to Model It," *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 5, pp. 5–12, Nov. 2013.

[18] A. Giovanidis and A. Avranas, "Spatial multi-lru caching for wireless networks with coverage overlaps," *SIGMETRICS Perform. Eval. Rev.*, vol. 44, no. 1, pp. 403–405, Jun. 2016.

[19] "Openmobilenetwork." [Online]. Available: openmobilenetwork.org/

[20] E. Nygren, R. K. Sitaraman, and J. Sun, "The Akamai Network: A Platform for High-performance Internet Applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 2–19, Aug. 2010.