Caching Policies for Delay Minimization in Small Cell Networks with Coordinated Multi-Point Joint Transmissions

Guilherme Iecker Ricardo, Alina Tuholukova, Giovanni Neglia, Thrasyvoulos Spyropoulos

Caching Policies for Delay Minimization in Small Cell Networks with Coordinated Multi-Point Joint Transmissions

Guilherme I. Ricardo[1,2], Alina Tuholukova[1,2], Giovanni Neglia[2], and Thrasyvoulos Spyropoulos[1]

[1]EURECOM, France, guilherme.ricardo@eurecom.fr, thrasyvoulos.spyropoulos@eurecom.fr
[2]Inria, Université Côte d'Azur, France, giovanni.neglia@inria.fr, tugolukovaalina@gmail.com

***Abstract*—In 5G and beyond network architectures, operators and content providers base their content distribution strategies on Heterogeneous Networks, where macro and small cells are combined to offer better Quality of Service to wireless users. On top of such networks, edge caching and Coordinated Multi-Point (CoMP) joint transmissions are used to further improve performance. In this paper, we address the average delay minimization problem by first formulating it as a static optimization problem. Even though the problem is NP-hard we are able to solve it via an efficient algorithm that guarantees a $\frac{1}{2}$-approximation ratio. We then proceed to propose two fully distributed and dynamic caching policies for the same problem. The first one asymptotically converges to the static optimal solution under the Independent Reference Model (IRM). The second one provides better results in practice under real (non-stationary) request processes. Our online policies outperform existing dynamic solutions that are PHY-unaware.**

***Index Terms*—Edge caching, CoMP, joint transmission, heterogeneous cellular networks, optimization, 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 2023 there will be 13 billion mobile connections, showing an increase of nearly 50% over 2018. 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 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 higher Quality of Service (QoS). Assuming that every small base station (BS) has a limited data storage capacity, the first technique is *caching* relevant content, e.g., the most popular content (with a higher probability of being requested). It allows users to directly access their desired content from the nearby BSs. As a consequence, the access latency can be drastically reduced as well as the backhaul congestion and main servers overload. The second technique is Coordinated Multi-Point (CoMP) *joint transmissions* [3]. The idea is that two or more BSs jointly transmit the requested file to the user. By doing so, users experience higher rates and, consequently, smaller delays to obtain the content.

Considering these techniques, in order to provide better experience for mobile users, our goal is to design strategies to minimize their experienced delay to get a request served. This problem can be solved statically, through offline solutions, or dynamically, through online caching policies.

In offline solutions, there is a centralized entity aware of the files popularities (assumed to be constant over time) and the whole network topology. With this information, it is able to decide which files should be cached at each BS, based on a given performance metric, e.g., probability of finding the requested file in the cache, bandwidth usage, original servers traffic load, etc. Normally, in this kind of approach, the content placement is split into two phases: (i) the measurement phase when requests are observed and files popularities are estimated and (ii) the file placement itself, usually performed during a low traffic load period.

However, having all this information available is a very strong assumption and is hardly satisfied in real systems. Moreover, static content placement may fail to capture short-term popularity changes. Despite its drawbacks, we use the offline approach as a comparison baseline for the online policies.

Online caching policies are distributed algorithms deployed at every BS, which specify the cache update rules, e.g., when and how to update the cache. In our proposed caching policies, the BSs, using implicit information from their local neighborhood, estimate the marginal gain (in terms of local delay savings) for keeping a copy of a cached content. The marginal gain is used to drive the (probabilistic) caching decisions for the local cache, with the end goal of maximizing *global* performance gain. Since the BSs take decisions on-the-fly, based on the request process they are exposed to, online policies turn out to be more reactive to files' popularity short-term variability. In comparison with offline solutions, each BS needs to know and exchange much less information. For these reasons, online caching policies are more appropriate to be deployed in real systems.

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.

A. Related Work

The term FemtoCaching was coined in [4], [5] to describe the framework in which overlapping small cells have some storage capacity. Assuming that files have known and static popularities and are requested according to the Independent Reference Model (IRM), the problem is to find the content placement maximizing the hit rate. The authors proved that this problem is NP-hard, so it can be efficiently approximated by a greedy heuristic, with worst-case performance guarantees.

In this framework, [6] maximizes the hit ratio by jointly optimizing content placement and user-association. The authors of [7] go even further and consider a more dynamic scenario in which network topology evolves with time, to capture users' mobility. Looking at the application layer, [8] and [9] jointly optimize content allocation and recommendation. The idea is that the hit rate might increase if users accept the recommendation of an already cached alternative content. All these variants of the FemtoCaching problem were proven to be NP-Hard and efficient heuristics were proposed.

To the best of our knowledge, [10] was one of the first papers to explore, using the idea of collaborative joint transmissions, the trade-off between hit rate and delay savings. In this context, it might be more advantageous to eliminate copies of less popular files in order to make room for multiple copies of more popular files, creating joint transmissions opportunities and, consequently, reducing the experienced delay. The authors proposed a first approach based on a randomized heuristic with Maximum Ratio Transmission and a second approach based on Zero-Forcing BeamForming. However, both approaches lack of theoretical optimality guarantees. First introduced by [11], and revisited in the present work in Section III, the average delay minimization in FemtoCaching framework under CoMP assumption can be formulated as a combinatorial optimization problem. Although this problem is NP-Hard, submodularity properties are guaranteed under specific assumptions. Then, the greedy algorithm can again be used to find a content allocation that is $\frac{1}{2}$ far from the optimal.

A common drawback of the aforementioned works is the difficulty to find the adequate timescale for popularity estimation that is long enough to provide accurate measurements and still able to capture short-term variations. Instead, online policies, such as LRU and its variants, are widely deployed because they do not require popularity estimation. Additionally, they enjoy robust analytical performance evaluation tools, like the celebrated Che's approximation [12]. Another downside of static centralized policies is that, in a dense cache network, having a centralized entity aware of the entire topology may not be feasible due to the network structure complexity. Online policies do not face the same issues because each cache takes individual decisions based on the experienced requests and possibly some limited information exchange with neighboring caches.

In [13], the authors study a hierarchy of interacting caches by considering what they called replication strategies. [14] introduces two caching policies: LRU-ONE and LRU-ALL. In the former, each user is assigned to a reference BS. When a user poses a request, its associated BS will update its cache, independently of which BS provided the file. In the latter, all neighboring BSs update their caches. The updates are based on the Least-Recently-Used (LRU) single-server policy.

In [15], the authors propose a model based on Che's and exponential approximations able to evaluate the performances of interacting caching policies in a dense cellular network. Moreover, they present the first distributed online policy with provable convergence properties for the FemtoCaching setup. In the context of online linear optimization, [16] uses a projected gradient method to decide how to split files and store chunks in the cache network. The technique cannot be applied with CoMP, because in their setting BSs store different chunks of the same file. The authors in [17] provide a general-purpose online caching policy and demonstrate its application in different study cases. They also prove its convergence to optimal allocations under IRM.

B. Paper Contribution and Organization

This work extends two previous conference papers: It consolidates the discussion on the static optimization problem introduced in [11] and it refines the caching model, case study, and experimental results from [18].

The main contributions of this paper are summarized below:

- We formalize the static delay minimization problem of allocating contents in a caching network with CoMP transmissions. We prove that the problem is NP-Hard.
- For this problem, we prove that, under homogeneous transmission conditions, the objective function is submodular, so the greedy algorithm provides a solution with $\frac{1}{2}$-approximation ratio.
- We provide more insight on the static problem's solution by studying a simple scenario that we called full-coverage. In this case study, we prove conditions for the optimal caching strategy to consist of replicating or diversifying contents throughout the network.
- In a distributed setup for the same problem, we propose two online policies, qLRU-Δd and 2LRU-Δd. In these policies, BSs use only local information to update their cache states, taking a probabilistic drift towards improving the problem's objective. qLRU-Δd is able to achieve optimal allocations under IRM when parameter q tends to 0. 2LRU-Δd addresses the problem of non-stationary requests, offering better performance in real scenarios.
- Using an extensive set of simulations, we demonstrate qLRU-Δd's convergence properties, and we observe both policies' ability to outperform other state-of-the-art policies in all considered scenarios.

The remainder of the paper is organized as follows. In Section II, we introduce the caching model and the transmission protocol. Then, in Section III, we formalize the static optimization problem, and we further discuss the full-coverage scenario in Section IV. We introduce online caching policies in Section V and evaluate their performance in Section VI. We conclude our paper with an overview of our results and possible future work in Section VII.

II. System Model and Operation

A heterogeneous network consists of "multiple tiers (or layers) of networks of different cell sizes/footprints and/or of multiple radio access technologies" [19], leading to overlapping cells. This overlap opens the possibility to use CoMP techniques, as a user is in general in the transmission range of multiple base stations (BS). While we do assume that all base stations have the same transmission power (e.g., a dense network of small cells) or equal capacity size, this is just for the sake of simplifying the presentation and providing experiments easier to interpret. Our caching policies do not require such homogeneity and their theoretical guarantees also hold when cells differ in terms of transmission power and/or cache size.

We consider in particular a set $[B]$ of base stations (BSs) arbitrarily located in a given area $A \subseteq \mathbb{R}^2$, where $[n]$ denotes the set $\{1, \ldots, n\}$. There is a set $[U]$ of user equipments (UEs) spread across area A that can connect to at least one BS. Let $G_u^{(b)} \in \mathbb{R}^+$ be the Signal-to-Noise Ratio (SNR) of the wireless channel between BS b and UE u. If the channel SNR is below a minimum SNR value, $g_{\min}$, we assume u and b cannot communicate, and set $G_u^{(b)} = 0$.

Each BS is equipped with a cache that can store up to C files from a catalog $[F]$ of files. In order to ease our exposition, we assume that all files have the same size equal to M bits. This assumption is widely considered in the literature (e.g., [4], [5], [11], [17], [20]) as large files are often split into smaller chunks of roughly equal sizes.

Let $V = [B] \times [F]$, where element $(b, f) \in V$ represents the placement of file f in BS b's cache. Let $V^{(b)} = \{b\} \times [F]$ be a subset of V representing the possible file placements in BS b. The set $X \subseteq V$ is a feasible static allocation if it satisfies the caches capacity constraints, i.e.,

$$|X \cap V^{(b)}| \leq C, \, \forall b \in [B]. \tag{1}$$

Because of the high density of BSs, each UE u will, in general, be within communication range of multiple BSs. We denote by $I_u = \left\{ b \in [B] : G_u^{(b)} > 0 \right\}$ the set of UE u's *neighboring* BSs, i.e., all BSs that have UE u within their coverage area and are able to receive requests and transmit content back to u. Among these BSs, under allocation X, a subset $J_{u,f}(X) = \{b \in I_u : (b, f) \in X\}$ is actually caching f.

Assume now that a set of BSs, $\mathcal{B} \subseteq I_u$, uses CoMP to jointly transmit the same file f to UE u. Then, the *end-to-end delay* is given by:

$$t_u(\mathcal{B}) = \frac{M}{W \log_2 \left(1 + \sum_{b \in \mathcal{B}} G_u^{(b)} \right)}, \tag{2}$$

where W is the channel bandwidth, so the denominator is the aggregate channel capacity[1] as in [22], [23]. We consider that $t_u(\emptyset) = +\infty$.

[1]For simplicity, we assume the limiting capacity provided by Shannon–Hartley theorem. However, our results are also valid for more realistic models, e.g., discrete rates dictated by predetermined Modulation and Coding Schemes (MCS) [21], as long as t_u is a non-increasing convex function (see Lemma I.1).

TABLE I
NOTATION SUMMARY

Symbol	Description
$[B]$	set of BSs $[B] = \{1, 2, \ldots, B\}$
$[U]$	set of UEs $[U] = \{1, 2, \ldots, U\}$
$[F]$	set of files $[F] = \{1, 2, \ldots, F\}$
C	cache capacity
M	file size
p_f	popularity of file f
W	channel bandwidth
d_B	backhaul access delay
$G_u^{(b)}$	SNR of the wireless channel between u and b
V	ground set of possible placements
I_u	set of UE u's neighboring BSs
$J_{u,f}(X)$	set of u's neighboring BSs caching f under allocation X
$t_u(\mathcal{B})$	end-to-end delay between u and BSs in $\mathcal{B} \subseteq I_u$
$d_{u,f}(X)$	experienced delay by u to get f under allocation X
$\bar{d}(X)$	average experienced delay
$\bar{s}(X)$	average delay saving
$\Delta d_{u,f}^{(b)}(X)$	marginal delay saving for u by caching f at b under X

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 $p_1 \geq p_2 \geq \ldots \geq p_F$. Moreover, each UE is equally likely to have generated the request. We will consider non-stationary traffic demand later in Section VI.

Upon every request, the network operates as follows: UE u broadcasts an inquiry message for file f that is received by its neighboring BSs in I_u. Then, according to the current cache state, there are two possibilities:

- If $J_{u,f}(X) = \emptyset$ (Cache Miss), the BS with the highest SNR, i.e., $b^* = \arg\max_{b \in I_u} \left\{ G_u^{(b)} \right\}$ 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 end-to-end delay to download from b^*, i.e., $t_u(\{b^*\})$.
- If $J_{u,f}(X) \neq \emptyset$ (Cache Hit), the BSs in $J_{u,f}(X)$ can jointly transmit to u, or the BS with the highest SNR in $I_u \setminus J_{u,f}(X)$, say it b', can retrieve an additional copy of f and then the BSs in $J_{u,f}(X) \cup \{b'\}$ can jointly transmit to u. The delay in the two cases is respectively $t(J_{u,f}(X))$ and $d_B + t(J_{u,f}(X \cup \{b'\}))$. The system will opt for the solution with the smallest delay.[2]

In both cases, the delay experienced by UE u to download file f under allocation X can be expressed as follows:

$$d_{u,f}(X) = \min\{t_u(J_{u,f}(X)), \, d_B + t_u(J_{u,f}(X) \cup \{b'\})\}, \tag{3}$$

where $b' = \arg\max_{b \in I_u \setminus J_{u,f}(X)} \left\{ G_u^{(b)} \right\}$.

[2]In scenarios with highly heterogeneous BSs within range of a UE, the ones currently having a cached copy might have weak SNRs, and the additional backhaul delay to fetch an additional copy to the BS with highest SNR might be amortized by the better overall channel performance.

III. Optimizing Static Cache Allocations

In this section, we assume contents are prefetched at caches during low-traffic periods (e.g., at night), as in related works [5], [6]. Files are allocated with the goal of minimizing the average experienced delay, assuming file popularities are known. This corresponds to solving the following problem:

Problem 1 (Average Delay Minimization Problem)**.**

$$\begin{aligned} \underset{X\subseteq V}{\text{minimize}} \quad & \bar{d}(X) = \sum_{u\in[U]} \frac{1}{U} \sum_{f\in[F]} p_f d_{u,f}(X) \qquad (4) \\ \text{subject to} \quad & |X \cap V^{(b)}| \leq C,\ \forall b \in [B]. \end{aligned}$$

The objective (4) is the average experienced delay for a request over all UEs and files and $d_{u,f}(X)$ is given by (3).

We start studying Problem 1 in the *homogeneous SNR regime*, where we assume that all non-zero SNRs are equal to g (i.e., $G_u^{(b)} = g$). We observe that it is hard to find Problem 1's exact solution:

Proposition III.1. *Problem 1 is NP-Hard, even in the homogeneous SNR regime.*

The proof is presented in Appendix I-B, and it relies on reducing the NP-Hard FemtoCaching problem in [5] to Problem 1.

A simple heuristic to approximate Problem 1 is through the greedy algorithm described as follows: Start from an empty solution $X = \emptyset$ and then iteratively add to X the element $(b, f) \in V$ that does not violate the constraints (1) and maximizes the marginal delay gain $\bar{d}(X) - \bar{d}(X \cup \{(b, f)\})$. We present a formal description in Algorithm 1. In the worst-case the algorithm goes through BC iterations, in which all $\mathcal{O}(BF)$ possible placements must be checked, resulting in a time complexity of $\mathcal{O}(B^2FC)$.

Algorithm 1: GreedyAD

input : files popularities $p_f, \forall f \in [F]$ and network topology $G_u^{(b)}, \forall b \in [B], \forall u \in [U]$
output: allocation set X

```
X ← ∅
while |X| < B · C do
    δ* ← 0
    for b ∈ [B] do
        if |X ∩ V^(b)| < C then
            for f ∈ [F] do
                δ ← d̄(X) − d̄(X ∪ {(b, f)})
                if δ > δ* then
                    i ← b, j ← f, δ* ← δ
    X ← X ∪ {(i, j)}
```

In order to provide approximation guarantees for GreedyAD, we need to express Problem 1 as a maximization problem. The following problem is equivalent[3] to Problem 1.

[3]Two optimization problems are "equivalent" if they have the same set of (global) optimizers [24, Section 4.1.3].

Problem 2 (Average Delay Saving Maximization Problem)**.**

$$\begin{aligned} \underset{X\subseteq V}{\text{maximize}} \quad & \bar{s}(X) = d^{(0)} - \bar{d}(X) \qquad (5) \\ \text{subject to} \quad & |X \cap V^{(b)}| \leq C,\ \forall b \in [B], \end{aligned}$$

where $d^{(0)} = d_B + \frac{M}{W \log_2(1+g)}$ is the delay experienced upon a cache miss. Function (5) can be seen as the average delay saving for a request under allocation X.

Now, consider the following lemmas:

Lemma III.2. *Function* (5) *is monotone and submodular.*

The proof is presented in Appendix I-C.

Lemma III.3. *Constraints* (1) *define a partition matroid.*

This lemma was originally proved in [5, Lemma 2].

Finally, the following theorem provides the approximation guarantees for GreedyAD in the homogeneous SNR regime.

Theorem III.4. *In the homogeneous SNR regime,* GreedyAD *is a* $1/2$*-approximation algorithm for Problem 2, i.e.,*

$$\bar{s}\left(X^{\text{GreedyAD}}\right) \geq \frac{1}{2}\bar{s}\left(X^{\text{OPT}}\right),$$

where X^{GreedyAD} *is a solution provided by* GreedyAD *and* X^{OPT} *is an optimal one.*

Proof. Problem 2 involves the maximization of a monotone submodular set function (Lemma III.2), subject to a partition matroid constraint (Lemma III.3). Therefore, according to [25, Theorem 3.1], the greedy algorithm achieves a $\frac{1}{2}$-approximation ratio. □

GreedyAD can also be used to approximate Problem 1 in the general case, though it does not enjoy the same approximation guarantees as in the homogeneous SNR regime. The reason is that the objective function is no longer submodular in the heterogeneous SNR regime as we show in an example in Appendix I-D. However, we see in Section VI that GreedyAD can still provide reasonable results.

The key characteristic of heterogeneous networks for our problem is the fact that UEs may be connected to multiple BSs simultaneously. Problem 1 imposes that allocation variables related to nearby caches are non-trivially coupled in two levels: (i) If the BS with the best SNR does not cache the requested file, it may fetch it from the back-end servers and (ii) neighboring BSs caching the requested file may jointly transmit it (with CoMP). We illustrate the trade-off created by these conflicting aspects in Section IV.

IV. Case Study: Full-Coverage Scenario

While the previous section provides a formal complexity analysis for the problem at hand, it does not provide much insight as to what the optimal allocation looks like in different scenarios. In order to fill this gap, we investigate some special instances of the problem that shed some light into the properties of the optimal cache allocation. We study the *full-coverage* scenario, which considers two assumptions: (i) SNRs are homogeneous and (ii) each UE u can connect to all BSs ($I_u = [B]$, for all u), so every UE has access to (the same) aggregate cache capacity of $B \cdot C$ files.

In this scenario, the end-to-end delay (2) is only a function of the number of copies in the system, then:

$$t(k) = \max\left\{\frac{M}{W\log_2(1+kg)}, \frac{M}{W\log_2(1+B\cdot g)}\right\}, \quad (6)$$

where we consider that we cannot have more than B copies. We consider $t(0) = +\infty$.

In this case, we can rewrite (3) as follows:

$$d(|J_{u,f}(X)|) = \min\left\{t(|J_{u,f}(X)|),\, d_B + t(|J_{u,f}(X)|+1)\right\}. \quad (7)$$

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

Proposition IV.1. *In the full-coverage scenario, an allocation provided by* GREEDYAD *is optimal.*

The proof of Proposition IV.1 is presented in Appendix II-A.

For the upcoming results, we define *locally optimal allocations* 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.*

First, we note that:

Lemma IV.2. *In the full-coverage scenario, an allocation is optimal if and only if it is locally optimal (Definition 1).*

The proof of Lemma IV.2 is in Appendix II-B.

Then, we characterize for the full-coverage scenario, the necessary and sufficient conditions for the optimal allocation to be one of the two extreme ones: *Full-diversity* (one copy of each of the $B \cdot C$ most popular files is stored in the network), and *full-replication* (the C most popular files are cached in each one of the B BSs).

Proposition IV.3. *In the full-coverage scenario, full-diversity is an optimal allocation if and only if*

$$p_1(d(1) - d(2)) \leq p_{B\cdot C}(d(0) - d(1)), \quad (8)$$

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

$$p_{C+1}(d(0) - d(1)) \leq p_C(d(B-1) - d(B)). \quad (9)$$

The proof of Proposition IV.3 is presented in Appendix II-C.

As an application of the results above, Fig. 1 shows, for a full-coverage scenario, for which regions of the parameter space (g, d_B) a full-diversity and full-replication allocations are optimal. Files popularities are generated according to Zipf law with exponent α.

The optimal solution depends on parameters g, d_B in a complex way. Fig. 1 (left) shows a simple setup, where 5 separated regions are formed. For a given value of g, for high values of d_B, we want to avoid paying the high retrieval cost for as many contents as possible, and then full-diversity is optimal. As we diminish d_B, the miss cost decreases and some form of replication is beneficial (mixed region), until we reach a full replication region. However, if we keep reducing d_B, the optimal allocation moves back to full-diversity (passing again through a mixed region). This happens because, when $d_B \approx 0$, $d(1) = d_B + t(2)$ and $d(2) = t(2)$ (because $B = 2$). This makes the LHS of (8) approximately zero and smaller than the RHS. A more realistic setup is provided in Fig. 1 (right). The fact that we cannot see the full-replication region is caused by the low difference in popularity of files C and $C+1$, for the specific values of C and α.

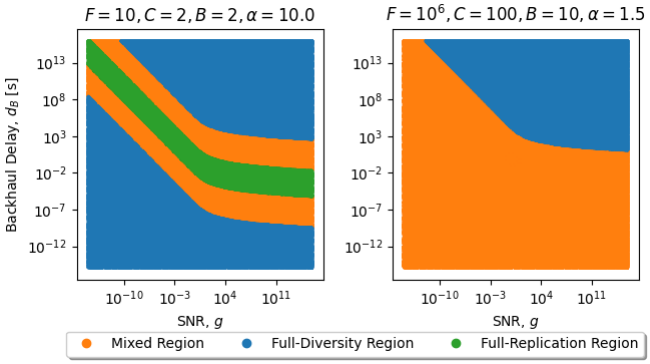

Fig. 1. Extreme allocations regions for different setups. Axes are in log scale.

However, given an instance of a generic topology, it is not guaranteed that both directions of the conditions in Proposition IV.3 are going to be satisfied. For example, consider a specific topology where BSs do not overlap at all. The optimal allocation is full-replication, even if condition (8) holds. In this case (and for any topology different from full-coverage), condition (8) is necessary but not sufficient for full-diversity to be the optimal allocation. Although Proposition IV.3 does not hold for general topologies, we can still derive new conditions:

Corollary 1. *For general network topologies, assuming homogeneous SNRs, the following conditions hold: (i) Inequality* (8) *is a* ***necessary condition*** *for the full-diversity allocation to be locally optimal, and (ii) inequality* (9) *is a* ***sufficient condition*** *for the full-replication allocation to be locally optimal.*

The proof of Corollary 1 is presented in Appendix II-D.

Corollary 1 can be used to forecast the best caching strategy for a given network. If files popularities, average SNR and backhaul access delay can be estimated, it is possible to analytically measure how close is the optimal allocation to an "extreme" one. For example, this may help develop an intuition on how beneficial CoMP joint transmissions can be for a given network setting.

Although static solutions have good performance and may eventually be optimal, they can rarely be deployed in practice because, as discussed in Section I, real systems hardly have a centralized intelligence aware of the entire network topology, transmission characteristics, and stationary files popularities [26]. However, given the tight approximation guarantees they offer, they work as a useful baseline for dynamic policies. Next section presents online solutions for the average delay minimization problem.

V. Online Caching Policies

In this section, we assume that caches are implemented as LRU-like queues with capacity for storing up to C files. The most-recently-used file is at the *front* and least-recently-used file is at the *rear* of the queue. The cache can perform three operations: (i) *Insert* a new file to the front, (ii) *evict* the file at the rear, and (iii) *move-to-the-front* a file already present.

At every request (u, f) from UE u for file f, after u's neighboring BSs serve the file, the respective caches react to that request by performing a set of operations. These operations aim to minimize the average experienced delay. Then, in what follows, we will consider the marginal delay gain experienced by UE u for having a copy of file f at BS b as follows:

$$\Delta d^{(b)}_{u,f}(X) = d_{u,f}\left(X \setminus \{(b,f)\}\right) - d_{u,f}\left(X\right). \tag{10}$$

In the next subsections, we introduce two online caching policies, qLRU-Δd and 2LRU-Δd. We use the notion of marginal delay gain $\Delta d^{(b)}_{u,f}(X)$, in (10), to drive the operations of distributed caching algorithms.

A. qLRU-Δd Caching Policy

The qLRU-Δd caching policy is inspired by qLRU-LAZY in [15], that is shown to maximize the hit ratio when parameter q converges to 0, under IRM request process. As we discuss in detail in Appendix III, we use the results in [17] to provide similar guarantees for the minimization of the average experienced delay. In summary, the policy's core idea is to perform *insertions* and *moves-to-the-front* with probabilities proportional to the gain (10). The *evictions* occur to the least-recently-used cached files.

qLRU-Δd *Policy:* Upon a request (u, f), given the current allocation X:

(i) All neighboring BSs caching f ($\forall b \in J_{u,f}(X)$) move f to the front with probability:

$$\rho^{(b)}_{u,f}(X) = \beta \cdot \Delta d^{(b)}_{u,f}(X), \tag{11}$$

where constant β guarantees that $\rho^{(b)}_{u,f}(X) \in (0, 1]$.

(ii) The remaining BSs ($\forall b \in I_u \setminus J_{u,f}(X)$) evict the file at the rear and insert f with probability:

$$q^{(b)}_{u,f}(X) = q \cdot \sigma^{(b)}_{u,f}(X), \tag{12}$$

where $q \in (0, 1]$ is a dimensionless parameter and:

$$\sigma^{(b)}_{u,f}(X) = \gamma \cdot \Delta d^{(b)}_{u,f}(X \cup \{(b,f)\}), \tag{13}$$

where γ guarantees that $\sigma^{(b)}_{u,f}(X) \in (0, 1]$.

We formalize qLRU-Δd caching policy in Algorithm 2 from the perspective of each BS b. The algorithm is a simple if-then-else clause triggered by a request message from the UE. Then, it is $\mathcal{O}(1)$ both in terms of run time and exchanged messages.

Although each BS runs the algorithm individually, the following result suggests that they manage to reach implicit coordination that asymptotically leads to an optimal allocation:

Algorithm 2: qLRU-Δd Caching Policy (for BS b)

Input: $J_{u,f}(X)$, $G_u^{(b')}, \forall b' \in I_u$, and d_B
if $b \in J_{u,f}(X)$ **then**
  with prob. $\rho^{(b)}_{u,f}(X)$ in (11) **do**
    Move-to-the-front f
else
  with prob. $q^{(b)}_{u,f}(X)$ in (12) **do**
    Evict file at the rear
    Insert f to cache

Proposition V.1. *Under IRM request traffic, Che's [12], [27], and exponentialization [15] approximations, a network of qLRU-Δd caches asymptotically achieves an optimal caching configuration, when $q \to 0$.*

Proposition V.1 follows from a more general result [17, Prop. IV.1] that we adapt in Appendix III. In fact, in this paper, we adapt the general caching policy qLRU-Δ, in [17], to solve the average delay minimization problem (4). We provide 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 ($q^{(b)}_{u,f}(X)$ converges to 0), while the frequency of *moves-to-the-front* for files already in the cache is unchanged ($\rho^{(b)}_{u,f}(X)$ does not depend on q). A file f at cache b is moved to the front with a probability proportional to $\Delta d^{(b)}_{u,f}(X)$, 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 ($\rho^{(b)}_{u,f}(X)$). Overall, the expected number of moves is proportional to $p_f \Delta d^{(b)}_{u,f}(X)$, 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 the least valuable file in the cache likely 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 global minimum.

Experiments in Section VI show that, under IRM request process, the smaller q is, the closer qLRU-Δd's delay is to GREEDYAD's delay, as it is stated by Proposition V.1.

Remark 1. The probability $q^{(b)}_{u,f}(X)$ is analogous to q in the usual qLRU policy. We note that setting $q^{(b)}_{u,f}(X) = q$, i.e., a constant value, suffices to prove the asymptotic convergence of the policy (Proposition V.1). However, we propose (12) to

make it more likely to add new copies to helpers bringing a larger marginal gain $\Delta d^{(b)}_{u,f}(X \cup \{(b,f)\})$. This can speed up the transient dynamics of the policy (see Section VI).

Remark 2. From equations (3) and (10) we see that probabilities $\rho^{(b)}_{u,f}$ and $q^{(b)}_{u,f}$ depend on the allocation of file f at nearby BSs or, more precisely, on (i) the aggregate SNR all BSs in $J_{u,f}$ can achieve when transmitting to u (i.e., $\sum_{b' \in J_{u,f}} G^{(b')}_u$), (ii) the SNR $G^{(b)}_u$ of the local channel from b to u, and (iii) the backhaul delay d_B. In cellular networks, each UE takes SNR measurements of BSs within range [28]. Thus, the information needed to set $q^{(b)}_{u,f}$ and $\rho^{(b)}_{u,f}$ can be obtained with negligible overhead and be piggybacked in uplink communications from u to the BSs.

B. 2LRU-Δd Caching 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 a short time scale: A new file gets a chance to be inserted in the cache on average by every $1/q$ requests, and by that time, its popularity may have declined. In order to gain in reactivity, we propose 2LRU-Δd.

In 2LRU-Δd, each BS maintains two storage layers: A *physical* cache and a *virtual* cache. The physical cache stores the actual files, while the virtual cache stores files' identification data. The identification for file f is denoted by ID(f). Here, we introduce the support variables $\hat{X}$ indicating the allocation of the files' IDs at virtual caches throughout the network. The two-layers structure along with the least-recently-used eviction rule are the core of plain 2LRU. On top of these characteristics, 2LRU-Δd additionally performs insertions and moves-to-the-front with probability proportional to (10).

2LRU-Δd *Policy:* Upon a request (u, f), given the current physical and virtual allocations, X and $\hat{X}$, respectively:

(i) All neighboring BSs caching ID(f) ($\forall b \in J_{u,f}(\hat{X})$) move ID($f$) to the front of the virtual cache and, if f is physically cached as well, they move f to the front of the physical cache with probability $\rho^{(b)}_{u,f}(X)$, given by (11), otherwise they cache f.

(ii) The remaining BSs ($\forall b \in I_u \backslash J_{u,f}(\hat{X})$) evict the ID at the rear and insert ID(f) at the front with probability $\sigma^{(b)}_{u,f}(X)$, given by (13).

We formalize 2LRU-Δd caching policy in Algorithm 3 from the individual perspective of each BS b. As qLRU-Δd, 2LRU-Δd has constant complexity in time and number of messages.

Remark 3. 2LRU-Δd does not enjoy the same theoretical guarantees of qLRU-Δd. However, its two-layer structure works as a more responsive filter, which makes it easier for 2LRU-Δd to reflect short-term popularity variabilities. This fact makes 2LRU-Δd more reactive than qLRU, whose insertion rate may be drastically reduced by parameter q. This feature is particularly favorable for scenarios where the request process has strong temporal locality, which is a characteristic often observed in practice [29]. Additionally, we consider an insertion probability depending on the average delay in order to tune the filter to be more selective towards files that may be supposed to reduce more the delay. Therefore, 2LRU-Δd is a strong candidate to cope with the delay minimization problem under non-stationary request processes, as we observe empirically in Section VI.

Algorithm 3: 2LRU-Δd Caching Policy (for BS b)

Input: $J_{u,f}(X)$, $J_{u,f}(\hat{X})$, $G^{(b')}_u, \forall b' \in I_u$, and d_B
if $b \in J_{u,f}(\hat{X})$ **then**
  Move-to-the-front ID(f) at virtual cache
  if $b \in J_{u,f}(X)$ **then**
    with prob. $\rho^{(b)}_{u,f}(X)$ in (11) **do**
      Move-to-the-front f at physical cache
  else
    Evict file at the rear of physical cache
    Insert f to physical cache
else
  with prob. $\sigma^{(b)}_{u,f}(X)$ in (13) **do**
    Evict file's ID at the rear of virtual cache
    Insert ID(f) to virtual cache

VI. Performance Evaluation

In this section, we first consider qLRU-Δd convergence to the optimal allocation. In particular, we show that it converges to an allocation similar to the one found by GreedyAD. Then, we investigate the proposed solutions' performance in a homogeneous SNR regime under synthetic and real request processes. Finally, we study how heterogeneous SNRs affect our policies' performance in a more realistic scenario.

A. Simulation Setup

1) Cellular network: In our experimental setup, we consider files with equal size $M = 1.0$ Mbits. Each cache can store $C = 100$ files (i.e., less than 1% of the catalog, which is inline with studies about small cell caching [5], [16]). We consider the *Berlin topology*, which consists of $B = 10$ BSs located across an area A according to the positions of T-mobile BSs in Berlin extracted from [30]. We call *network density* the average number of BSs covering a UE and we assume homogeneous spatial user density. The backhaul-access delay is $d_B = 100$ ms and the channel bandwidth is $W = 5$ MHz, for all pairs BS-UE (which are values consistent with related literature [11], [17]).

2) Request Generation Mechanisms: We simulate a discrete time process, where, at every step, a UE is chosen uniformly at random to generate the next request. We consider the following mechanisms:

- *Stationary Request Process*: Simulations have a warm up phase and a measurement phase each consisting of 100 million requests. At every request, a file is chosen from a catalog of $F = 10^6$ files according to a Zipf law with exponent $\alpha = 1.2$, unless otherwise stated.

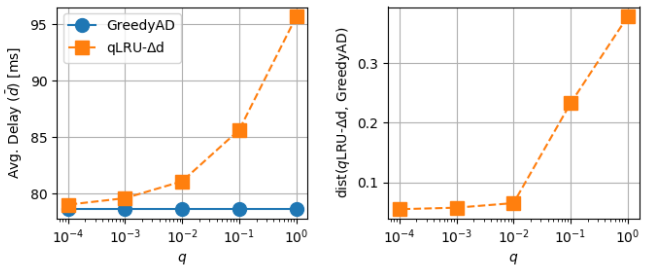

Fig. 2. Convergence analysis: Delay (left) and allocation (right) as q converges to 0, for $\alpha = 1.2$, $d_B = 100$ms, and $g = 10$dB.

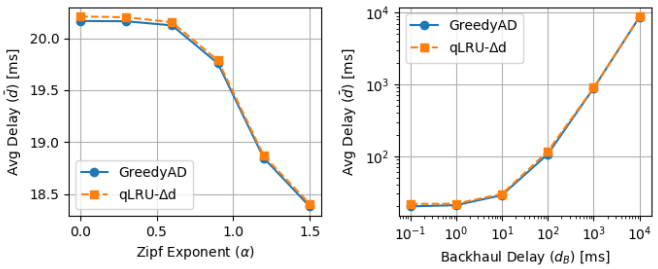

Fig. 3. Convergence analysis: α variation with $d_B = 100ms$ (left) and d_B variation with $\alpha = 1.2$ (right), for $q = 0.001$.

- *Non-stationary Request Process*: We additionally consider a request process based on a real trace from Akamai Content Delivery Network [31], consisting of 17 million requests observed during 5 consecutive days. The trace is described in [32]. Files popularities may vary over different time scales.

3) Algorithms and Policies: Besides our proposed greedy algorithm (Section III) and online policies (Section V), we consider the following solutions in our performance analysis:

- GREEDYHR: Static allocation determined by a greedy algorithm that aims to maximize the hit ratio.
- LRU-ONE and LRU-ALL [14]. Upon request (u, f), in LRU-ONE, only a fixed BS $b_u \in I_u$ updates its cache, whereas in LRU-ALL, all BSs $b \in I_u$ update their caches.
- qLRU-LAZY and 2LRU-LAZY [15]. Upon request (u, f), if it is a hit, BS $b \in I_u$ updates its cache only if $J_{u,f}(X) = \{b\}$, if it is a miss, a BS b^* is randomly chosen to serve and possibly insert the file into its cache. The updates and insertions are based on plain qLRU and 2LRU, respectively.
- LFU-ALL. Each cache keeps track of how often every file has been requested. The least-frequently-requested cached file is evicted to make room for new insertions.

B. Numerical Results

1) Convergence Analysis for qLRU-Δd

To study the convergence of qLRU-Δd, we consider the *homogeneous SNR regime* with SNR $g = 10$dB, for all connected pairs BS-UE, and the *stationary request process*. We show that, as q tends to 0, qLRU-Δd converges to the optimal static solution similar to the one provided by GREEDYAD.

Fig. 2 (left) shows the average delay of qLRU-Δd and GREEDYAD for different values of q. As q decreases, qLRU-Δd's average delay converges to GREEDYAD's one. 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[4] between their occupancy vectors $\boldsymbol{\theta}$, which are $F \times 1$ vectors containing the fraction of time every file spent on average in a cache during the measurement phase. For GREEDYAD, we can compute directly $\boldsymbol{\theta}$ from the allocation matrix. As q decreases, the distance between qLRU-Δd and GREEDYAD decreases, indicating that qLRU-Δd tends to store the same files GREEDYAD stores.

[4]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.

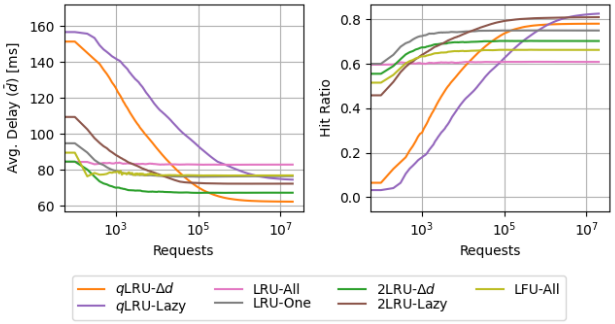

Fig. 4. Average delay (left) and hit ratio (right) of different policies after every 100 requests under IRM in a Berlin topology.

Observation 1. qLRU-Δd converges to the optimal solution provided by GREEDYAD as $q \to 0$.

Fig. 3 shows the average delay achieved by GREEDYAD and qLRU-Δd for different values of Zipf exponent α (left) and backhaul-access delay d_B (right), when $q = 0.001$. We observe that the two curves almost match for all different parameter choices, indicating that the convergence is achieved in multiple settings.

Observation 2. For sufficiently small q, qLRU-Δd achieves delays close to GREEDYAD across different network settings and popularity distributions.

Now, for a fixed network density of 9.4 and for each online policy, we show the average delay Fig. 4 (left) and the hit ratio Fig. 4 (right) after every 100 requests. In this plot, we can observe the policies convergence process. Files popularities follow a Zipf law with exponent $\alpha = 1.2$ and we also take $g = 10$dB and $d_B = 100$ms. We fix $q = 0.001$ for qLRU-Δd and qLRU-LAZY.

First, we highlight that qLRU-like policies have worse performance in the beginning due to the lower insertion rate (caused by parameter q), until the point where they stabilize

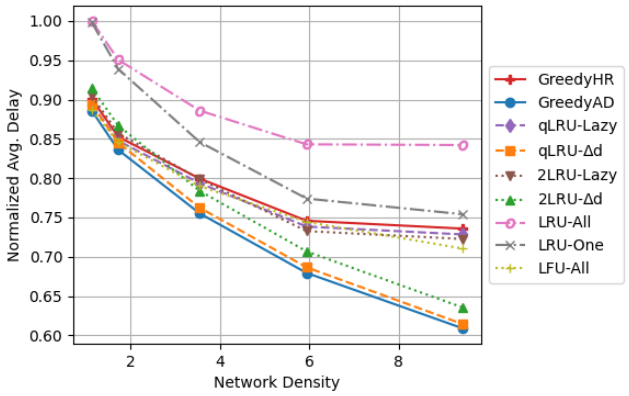

Fig. 5. Average delay of different policies versus density in Berlin topology with IRM request process ($\alpha = 1.2$).

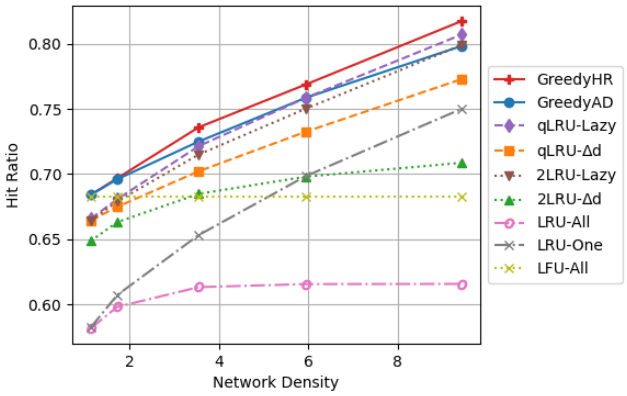

Fig. 6. Hit ratio of different policies versus density in Berlin topology with IRM request process ($\alpha = 1.2$).

and present better results (after 10^7 requests in this scenario). Despite their noticeably faster convergence, 2LRU-like policies reach performance levels close to the qLRU-like ones. This fact reveals 2LRU-Δd higher reactivity and suggests its suitability for dealing with non-stationary request processes.

Observation 3. Despite their slightly worse performance under IRM, 2LRU-like policies present faster convergence.

2) Effect of Network Density

We consider the *homogeneous SNR regime* with $g = 10$dB, for all connected pairs BS-UE. We fix the BSs positions and vary the transmission range to achieve network densities from 1.1 (almost isolated BSs) to 9.4 (highly overlapped network, with approximately 73% of UEs covered by all 10 BSs). We fix $q = 0.001$ for qLRU-Δd and qLRU-LAZY.

In the first setting, we assume a *stationary request process*. In Fig. 5, we show the normalized average delay as function of the network density, for different policies and algorithms. The qLRU-Δd result is very close to GREEDYAD one, reasserting its convergence across different densities. 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%.

Observation 4. Under stationary requests, qLRU-Δd outperforms other policies, presenting nearly optimal results.

In Fig. 6, we show the hit ratio corresponding to the experiment previously described. Policies like qLRU-LAZY and 2LRU-LAZY [15] outperform other policies as they are designed to maximize the hit ratio, even though they have inferior performance in terms of average delay (see Fig. 5).

Observation 5. As expected, policies targeting the hit ratio in general perform worse in terms of average delay.

In the second setting, we assume the *non-stationary request process*. The greedy allocation in this case was determined by estimating the files popularities over 5 days. However,

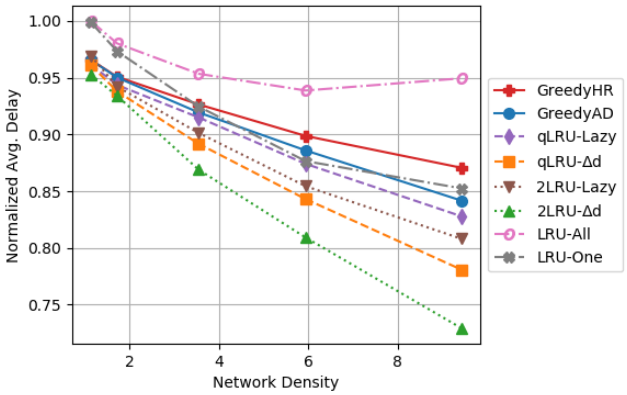

Fig. 7. Performance analysis of various policies in a real topology with Akamai trace.

real request processes exhibit strong temporal locality features. Static allocations based on time-average popularities smooth out the variability over short time scale. Consequently, we observe in Fig. 7 that GREEDYAD and GREEDYHR perform worse than most of the online policies.

Observation 6. Under non-stationary requests, static solutions tend to perform worse than online policies.

On the contrary, 2LRU-like policies are highly reactive and may be able to capture short-time popularity variations, promising better performance. Fig. 7 shows that 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.

Observation 7. Under non-stationary requests, 2LRU-Δd outperforms all other policies.

3) Heterogeneous SNR Regime

In the online policies simulations, at every request (u, f),

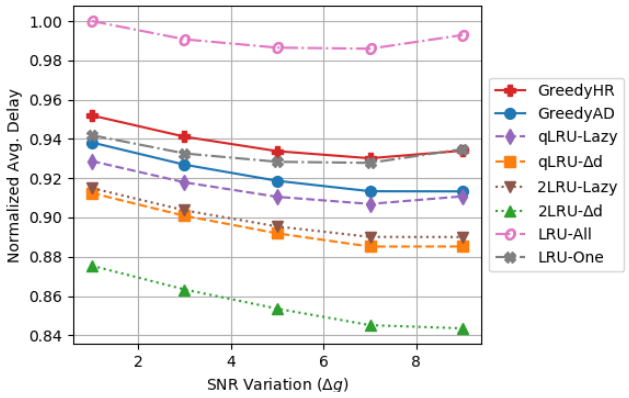

Fig. 8. Slow SNR variability regime: Berlin topology with density 5.9, $g_0 = 10.0$dB, and $d_B = 100.0$ms following trace-based request process.

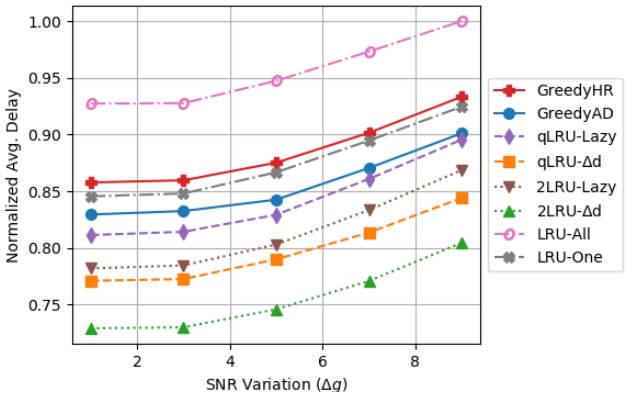

Fig. 9. Fast SNR variability regime: Berlin topology with density 5.9, $g_0 = 10.0$dB, and $d_B = 100.0$ms following trace-based request process.

the SNRs $G_u^{(b)}, \forall b \in I_u$ are chosen uniformly at random within a range, i.e., $G_u^{(b)} \in [g_0 - \Delta g, g_0 + \Delta g]$. For the static solutions, we simply calculate in advance the average experienced delay for each UE to download from $k = 0, \ldots, |I_u|$ cached copies, and apply the greedy algorithm as if SNRs were homogeneous. We consider the same Berlin topology, with density of 5.9 BSs/UE, on average, and file requests follow again the (non-stationary) Akamai trace. Moreover, we consider two different scenarios for SNR variability:

- *Slow SNR variability regime*: SNRs can be considered constant from the moment the request is posed until it is served.
- *Fast SNR variability regime*: SNRs change over the timescale corresponding to the backhaul retrieval time. As a consequence, the BS that retrieves an additional copy (b' in (3)) may not have the highest SNR by the time the copy is available.

First, in Fig. 8, we show the performance of the caching policies under slow SNR variability regime. We present the average delay as a function of the SNR variation Δg. We observe that all curves decrease for smaller values of SNR variation ($\Delta g \in [1.0, 7.0]$). The average delay tends to increase again for larger SNR variation ($\Delta g \geq 9.0$) for the hit-ratio maximization schemes. The fact that the BS with the highest SNR serves the requested file mitigates the miss cost for the delay-based schemes. Our proposed policies also outperform other schemes. The maximum observed performance gain (related to 2LRU-Δd and LRU-ALL) moderately increases with Δg, going from 13% to around 15%.

Observation 8. The proposed policies outperform other schemes and the SNR variation has low impact on the schemes' relative performance gains.

The SNR variability may be interpreted as the BSs using different transmission powers, which is a common characteristic of real heterogeneous cellular networks (e.g., in an overlay of femto, pico, and macro cells). The previous experimental result suggests that dynamic policies are resilient to different transmission conditions and may achieve satisfactory results even in these scenarios.

In a similar fashion, in Fig. 9, we show the performance of the caching policies under fast SNR variability regime. All policies presents a strictly increasing behavior. This fact is explained by Jensen's inequality, since the delay is now a convex random function: Given $g = g_0 + \Delta g$ and $g' = g_0 - \Delta g$, the delay reduction achieved with the larger g is smaller than the delay increase due to the smaller g'.

Observation 9. In a scenario with more unstable transmission conditions (fast SNR variability), the average delay strictly increases with the SNR variation.

VII. Discussion and Conclusions

In this paper we proposed static and online solutions for the experienced delay minimization problem in a FemtoCaching architecture with CoMP joint transmissions. We formulated the static optimization problem and provided a greedy algorithm with approximation guarantees under homogeneous SNR regime. In this regime, we studied the full-coverage scenario that gives us more insight on the possible solutions. Then, we introduced two novel online caching policies able to minimize delay under IRM and to provide good results under a real request process. In our experiments, we observed qLRU-Δd's convergence and evaluated all proposed cache schemes performances under different request processes and SNR regimes. We conclude that our caching policies achieve considerable performance gains with negligible additional deployment complexity.

Interesting future works include, for example, studying the problem under heterogeneous file size assumption. Besides the files popularities and network transmission conditions (SNRs and backhaul access time), the files' sizes would affect the problem solution by imposing different costs in terms of both storage and transmission time.

REFERENCES

[1] CISCO, "Cisco annual internet report (2018–2023)," CISCO, Tech. Rep., March 2020.

[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. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *2012 Proceedings IEEE INFOCOM*, March 2012, pp. 1107–1115.

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

[6] K. Poularakis, G. Iosifidis, and L. Tassiulas, "Approximation algorithms for mobile data caching in small cell networks," *IEEE Transactions on Communications*, vol. 62, no. 10, pp. 3665–3677, Oct 2014.

[7] T. Wang, L. Song, and Z. Han, "Dynamic femtocaching for mobile users," in *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, March 2015, pp. 861–865.

[8] P. Sermpezis, T. Giannakas, T. Spyropoulos, and L. Vigneri, "Soft cache hits: Improving performance through recommendation and delivery of related content," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1300–1313, 2018.

[9] M. Costantini, T. Spyropoulos, T. Giannakas, and P. Sermpezis, "Approximation guarantees for the joint optimization of caching and recommendation," in *IEEE ICC 2020*, Dublin, IRELAND, 06 2020.

[10] W. C. Ao and K. Psounis, "Distributed caching and small cell cooperation for fast content delivery," in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, ser. MobiHoc '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 127–136. [Online]. Available: https://doi.org/10.1145/2746285.2746300

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] G. S. Paschos, A. Destounis, L. Vigneri, and G. Iosifidis, "Learning to cache with no regrets," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 235–243.

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

[18] G. Ricardo, G. Neglia, and T. Spyropoulos, "Caching policies for delay minimization in small cell networks with joint transmissions," in *IEEE ICC 2020*, Dublin, IRELAND, 06 2020.

[19] H. Sun and R. Q. Hu, *Heterogeneous cellular networks*. John Wiley & Sons, 2013.

[20] G. Neglia, D. Carra, and P. Michiardi, "Cache Policies for Linear Utility Maximization," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 302–313, 2018. [Online]. Available: https://doi.org/10.1109/TNET.2017.2783623

[21] G. Arvanitakis, T. Spyropoulos, and F. Kaltenberger, "An analytical model for flow-level performance in heterogeneous wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 3, pp. 1488–1501, 2018.

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

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

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

[25] A. Krause, R. Rajagopal, A. Gupta, and C. Guestrin, "Simultaneous placement and scheduling of sensors," in *2009 International Conference on Information Processing in Sensor Networks*. IEEE, 2009, pp. 181–192.

[26] M. Leconte *et al.*, "Placing dynamic content in caches with small population," in *IEEE INFOCOM 2016*, 2016.

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

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

[29] 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.

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

[31] 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.

[32] G. Neglia, D. Carra, M. Feng, V. Janardhan, P. Michiardi, and D. Tsigkari, "Access-time-aware cache algorithms," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 2, no. 4, pp. 21:1–21:29, Nov. 2017.

[33] J. Edmonds, "Matroids and the greedy algorithm," *Mathematical programming*, vol. 1, no. 1, pp. 127–136, 1971.

Guilherme Iecker Ricardo received a B.Sc. degree in Computer and Information Engineering and a M.Sc. degree in Systems Engineering and Computer Science, both from Universidade Federal do Rio de Janeiro, Brazil, in 2016 and 2018, respectively. Currently, he is pursuing a Ph.D. degree in Université Côte d'Azur, in a collaboration between EURECOM and INRIA, Sophia Antipolis research centers in France.

Alina Tuholukova received a Master's degree from the University of Nice Sophia Antipolis, France, in 2015. Then, she worked as a research engineer at Inria, Sophia Antipolis until 2016.

Giovanni Neglia received the master's degree in electronic engineering and the Ph.D. degree in telecommunications from the University of Palermo, Italy, in 2001 and 2005, respectively. He has been a Researcher at Inria, Sophia Antipolis, France, since September 2008. In 2005, he was a Research Scholar at the University of Massachusetts, Amherst, MA, USA, visiting the Computer Networks Research Group. Before joining Inria, he was a post-doctorate at the University of Palermo and an External Scientific Advisor of Maestro Team at Inria.

Thrasyvoulos Spyropoulos received the Diploma in Electrical and Computer Engineering from the National Technical University of Athens, Greece, and a Ph.D degree in Electrical Engineering from the University of Southern California. He was a post-doctoral researcher at INRIA and then, a senior researcher with the Swiss Federal Institute of Technology (ETH) Zurich. He is currently an Assistant Professor at EURECOM, Sophia-Antipolis. He is the recipient of the best paper award in IEEE SECON 2008, and IEEE WoWMoM 2012.

APPENDIX I
ADDITIONAL CONTENT FOR SECTION III

A. Homogeneous SNR Additional Notation

In the homogeneous SNR regime, the delay experienced by UE u to download file f under allocation X is a function of the number of cached copies of f in its neighboring BSs, $|J_{u,f}(X)|$:

$$d_u(|J_{u,f}(X)|) = \min\{t_u(|J_{u,f}(X)|), \\ d_B + t_u(|J_{u,f}(X)|+1)\}. \tag{14}$$

In this case, we cannot allow two-steps transmissions when all neighboring BSs are already providing a cached copy of the requested file, i.e., $J_{u,f}(X) = I_u$. Then, to avoid this issue, we redefine t_u as follows:

$$t_u(k) = \max\left\{\frac{M}{W\log_2(1+k\cdot g)}, \frac{M}{W\log_2(1+|I_u|\cdot g)}\right\}, \tag{15}$$

where we truncate the delay at its minimum allowed value. We consider $t_u(0) = +\infty$.

B. Proof of Proposition III.1

We want to show that the FemtoCaching problem [5] can be reduced to Problem 1. We use the additional notation introduced in Appendix I-A for homogeneous SNR regime. The homogeneous version of the FemtoCaching problem is:

Problem 3 (Homogeneous FemtoCaching Problem)**.**

$$\underset{X\subset V}{\text{minimize}}\ \bar{d}_F(X) = \frac{1}{U}\sum_{u,f} p_f \cdot \left(\mathbb{1}_{J_{u,f}(X)=\emptyset}\cdot d_B + t(1)\right), \tag{16}$$

subject to constraints (1), where $\bar{d}_F(X)$ is the average delay for the FemtoCaching problem and $t(1) = \frac{M}{W\log_2(1+g)}$ is given by (15) dropping subscript u, since every UE is covered by at least one BS.

Assume further that popularities can be written as rational numbers, i.e.,

$$p_f = \frac{m_f}{n}, m_f, n \in \mathbb{N}, \forall f \in [F]. \tag{17}$$

We observe that, given any two allocations X, X', such that, $\bar{d}_F(X) \neq \bar{d}_F(X')$, it holds that:

$$\left|\bar{d}_F(X) - \bar{d}_F(X')\right| = \tag{18}$$

$$= \left|\frac{1}{U}\sum_{u,f} p_f(\mathbb{1}_{J_{u,f}(X)}\cdot d_B + t(1)) \right. \\ \left. -\frac{1}{U}\sum_{u,f} p_f(\mathbb{1}_{J_{u,f}(X')}\cdot d_B + t(1))\right| \tag{19}$$

$$= \frac{1}{n\cdot U}\left|\sum_{u,f} m_f(\mathbb{1}_{J_{u,f}(X)}\cdot d_B + t(1)) \right. \\ \left. -\sum_{u,f} m_f(\mathbb{1}_{J_{u,f}(X')}\cdot d_B + t(1))\right| \tag{20}$$

$$= \frac{d_B}{n\cdot U}\left|\sum_{u,f} m_f\left(\mathbb{1}_{J_{u,f}(X)} - \mathbb{1}_{J_{u,f}(X')}\right)\right| \tag{21}$$

$$\geq \frac{d_B}{n\cdot U}. \tag{22}$$

Equality (19) is the direct application of the FemtoCaching problem's objective (16). We use the popularities' rational notation in (17) to derive (20). The absolute part of (21) always results in a positive integer, which leads to inequality (22).

We take a large enough value for the SNR g, such that the following inequality holds:

$$\frac{d_B}{n\cdot U} > t(1). \tag{23}$$

Then, Problem 1's objective can be written as:

$$\bar{d}(X) =$$

$$= \frac{1}{U}\sum_{u,f} p_f d_u(|J_{u,f}(X)|) \tag{24}$$

$$= \frac{1}{U}\sum_{u,f} p_f\left(\mathbb{1}_{J_{u,f}(X)=\emptyset}d^{(0)} + \mathbb{1}_{J_{u,f}(X)\neq\emptyset}d_u(|J_{u,f}(X)|)\right) \tag{25}$$

$$= \frac{1}{U}\sum_{u,f} p_f\left(\mathbb{1}_{J_{u,f}(X)=\emptyset}(d_B + t(1)) + \mathbb{1}_{J_{u,f}(X)\neq\emptyset}t(1) \right. \\ \left. -\mathbb{1}_{J_{u,f}(X)\neq\emptyset}t(1) + \mathbb{1}_{J_{u,f}(X)\neq\emptyset}d_u(|J_{u,f}(X)|)\right) \tag{26}$$

$$= \frac{1}{U}\sum_{u,f} p_f\left(\mathbb{1}_{J_{u,f}(X)=\emptyset}d_B + t(1)\right) \\ -\frac{1}{U}\sum_{u,f} p_f\mathbb{1}_{J_{u,f}(X)\neq\emptyset}\left(t(1) - d_u(|J_{u,f}(X)|)\right) \tag{27}$$

$$= \bar{d}_F(X) - \frac{1}{U}\sum_{u,f} p_f\mathbb{1}_{J_{u,f}(X)\neq\emptyset}(t(1) - d_u(|J_{u,f}(X)|)). \tag{28}$$

Equality (24) is an adaptation of Problem 1's objective to the homogeneous SNR regime, where we replace the general delay function (3) with (14). Equation (25) comes from the fact that, if $J_{u,f}(X) = \emptyset$ (i.e., cache miss), the delay is $d^{(0)} = d_B + t(1)$ (first defined in Problem 2) or, if $J_{u,f}(X) \neq \emptyset$ (i.e., cache hit), the delay is $d_u(|J_{u,f}(X)|)$. Note that this decomposition is at first redundant, given that definition (14) covers both miss and hit cases. We obtain (27) by simply putting the indicator functions from (26) in evidence. Finally, we observe that the first term in (27) is exactly the definition of the FemtoCaching objective function, which yields (28).

Observe that $d_u(|J_{u,f}(X)|)) < t(1)$, then the following relation holds:

$$\bar{d}_F(X) \geq \bar{d}(X) \geq \bar{d}_F(X) - t(1). \tag{29}$$

Now, we prove that, given two allocations X, X',

$$\bar{d}_F(X) < \bar{d}_F(X') \Leftrightarrow \bar{d}(X) < \bar{d}(X') \tag{30}$$

and then solving Problem 1 brings the solution to Problem 3.

First, we prove $\bar{d}_F(X) < \bar{d}_F(X') \Rightarrow \bar{d}(X) < \bar{d}(X')$:

$$\begin{aligned}
&\text{Hypothesis 1:} \quad \bar{d}_F(X) < \bar{d}_F(X') \\
&\bar{d}(X) \leq \bar{d}_F(X) && \text{by (29)} \\
&\leq \bar{d}_F(X') - \frac{d_B}{n \cdot U} && \text{by Hyp. 1 and (22)} \\
&< \bar{d}_F(X') - t(1) && \text{by (23)} \\
&\leq \bar{d}(X') && \text{by (29)}
\end{aligned}$$

Second, we prove $\bar{d}(X) < \bar{d}(X') \Rightarrow \bar{d}_F(X) < \bar{d}_F(X')$:

$$\begin{aligned}
&\text{Hypothesis 2:} \quad \bar{d}(X) < \bar{d}(X') \\
&\bar{d}_F(X') \geq \bar{d}(X') \\
&> \bar{d}(X) - t(1) \\
&\geq \bar{d}_F(X) && \text{by (29)}
\end{aligned}$$

Then, because both implications hold, (30) also holds and, therefore, Problem 3 can be reduced to Problem 1. Because Problem 3 is NP-hard [5] and we can reduce it to Problem 1, then Problem 1 is NP-hard.

C. Proof of Lemma III.2

We start proving the following lemma:

Lemma I.1. *For any u and any $k_1, k_2 \in \mathbb{Z}_+$, such that $k_1 \leq k_2$, the following inequality holds:*

$$t_u(k_1) - t_u(k_1+1) \geq t_u(k_2) - t_u(k_2+1). \tag{31}$$

Proof. Let $h(x) = \frac{M}{W \log_2(1+g \cdot x)}$. Function h can be written as a function composition $h(x) = (w \circ y)(x)$, where $w(x) = \frac{M}{W \log_2(x)} = \frac{M \ln(2)}{W} \cdot \left(\frac{1}{\ln(x)}\right)$ and $y(x) = 1 + g \cdot x$. Function y is affine. Function w first and second derivatives are, respectively, $w'(x) = \frac{M \ln(2)}{W} \cdot \left(\frac{-1}{x \ln^2(x)}\right)$ and $w''(x) = \frac{M \ln(2)}{W} \cdot \left(\frac{\ln(x)+2}{x^2 \ln^3(x)}\right)$. For $x > 1$, $w'(x) < 0$, which makes w decreasing, and $w''(x) > 0$, which makes w convex. Because h is the composition of a convex decreasing function and an affine increasing function, h is also a convex decreasing function for $x > 1$. Moreover, because t_u is the point-wise maximum between h and constant $\frac{M}{W \log_2(1+|I_u| \cdot g)}$, t_u is a non-increasing convex function. This means that the function $l(k) = t_u(k) - t_u(k+1)$ is also non-increasing. Therefore, inequality (31) holds. □

We separate the proof of Lemma III.2 in two parts, one for each desired property of $\bar{s}(X)$.

Monotonicity: Let $X \subset X' \subset V$ and consider the case where $X' = X \cup \{(b', f')\}$. We can apply this argument item-by-item to prove the case for general X and X'. By definition, the set function (5) is *monotone* if:

$$\begin{aligned}
&\bar{s}(X) \leq \bar{s}(X') \Leftrightarrow \bar{d}(X) \geq \bar{d}(X') \\
&\Leftrightarrow \sum_{u,f} p_f d_u(|J_{u,f}(X)|) \geq \sum_{u,f} p_f d_u(|J_{u,f}(X')|)
\end{aligned} \tag{32}$$

We observe that $\forall f \neq f'$, the LHS equals the RHS in (32), so we focus on cases where $f = f'$. Similarly, we consider only the set of users covered by BS b', that we call $\mathcal{U}(b')$. Then, (32) becomes:

$$\begin{aligned}
&\bar{s}(X) \leq \bar{s}(X') \Leftrightarrow \\
&\Leftrightarrow p_{f'} \sum_{u \in \mathcal{U}(b')} (d_u(|J_{u,f'}(X)|) - d_u(|J_{u,f'}(X)|+1)) \geq 0.
\end{aligned}$$

Notice that $t_u(k)$ is non-increasing, which makes $d_u(k)$ non-increasing as well (see (14)). Then, $d_u(|J_{u,f'}(X)|) - d_u(|J_{u,f'}(X)|+1) \geq 0$ and, therefore, (5) is monotone.

Submodularity: Let $X \subset X' \subset V$ and $(b', f') \in V \backslash X'$. The set function (5) is *submodular* if:

$$\begin{aligned}
&\bar{s}(X \cup \{(b', f')\}) - \bar{s}(X) \geq \bar{s}(X' \cup \{(b', f')\}) - \bar{s}(X') \Leftrightarrow \\
&\Leftrightarrow \bar{d}(X) - \bar{d}(X \cup \{(b', f')\} \geq \bar{d}(X) - \bar{d}(X \cup \{(b', f')\} \\
&\Leftrightarrow \sum_{u,f} p_f \left(d_u(|J_{u,f}(X)|) - d_u(|J_{u,f}(X \cup \{(b', f')\})|)\right) \\
&\geq \sum_{u,f} p_f \left(d_u(|J_{u,f}(X')|) - d_u(|J_{u,f}(X' \cup \{(b', f')\})|)\right)
\end{aligned}$$

We observe that $\forall f \neq f'$, the LHS equals the RHS in the inequality above, so we focus on cases where $f = f'$. Similarly, we consider only the set of users covered by BS b', i.e., $\mathcal{U}(b')$. Then, the inequality above becomes:

$$\begin{aligned}
&p_{f'} \sum_{u \in \mathcal{U}(b')} d_u(|J_{u,f'}(X)|) - d_u(|J_{u,f'}(X)|+1) \\
&\geq p_{f'} \sum_{u \in \mathcal{U}(b')} d_u(|J_{u,f'}(X')|) - d_u(|J_{u,f'}(X')|+1).
\end{aligned} \tag{33}$$

We will prove (33) by showing that for each u,

$$\begin{aligned}
&d_u(|J_{u,f'}(X)|) - d_u(|J_{u,f'}(X)|+1) \\
&\quad \geq d_u(|J_{u,f'}(X')|) - d_u(|J_{u,f'}(X')|+1)
\end{aligned}$$

and, since it refers to a single file f', we can simplify the notation defining $k = |J_{u,f'}(X)|$ and $k' = |J_{u,f'}(X')|$. If we prove the inequality above for $k' = k+1$, then it will hold $\forall k' \geq k+1$. Thus, we need to show that $\forall u \in \mathcal{U}(b')$,

$$d_u(k) - d_u(k+1) \geq d_u(k+1) - d_u(k+2). \tag{34}$$

However, the delay d_u is the minimum of two functions (see (14)). We observe that $d_u(k) = t_u(k) \Rightarrow d_u(k+1) = t_u(k+1)$. In fact,

$$\begin{aligned}
&d_u(k) = t_u(k) \Rightarrow \\
&\quad \Rightarrow t_u(k) \leq d_B + t_u(k+1) \Rightarrow \\
&\quad \Rightarrow t_u(k) - t_u(k+1) \leq d_B \Rightarrow \\
&\quad \Rightarrow t_u(k+1) - t_u(k+2) \leq d_B \Rightarrow \quad \text{(by Lemma I.1)} \\
&\quad \Rightarrow d_u(k+1) = t_u(k+1).
\end{aligned}$$

Then, we need to consider only four cases:

Case (I): $d_u(k) = t_u(k)$, $d_u(k+1) = t_u(k+1)$, and $d_u(k+2) = t_u(k+2)$. Then, (34) is written as:

$$t_u(k) - t_u(k+1) \geq t_u(k+1) - t_u(k+2),$$

which is always true by Lemma I.1.

Case (II): $d_u(k) = d_B + t_u(k+1)$, $d_u(k+1) = t_u(k+1)$, and $d_u(k+2) = t_u(k+2)$. Then, (34) is written as:

$$d_B + t_u(k+1) - t_u(k+1) \geq t_u(k+1) - t_u(k+2)$$
$$d_B \geq t_u(k+1) - t_u(k+2),$$

which is true as $d_u(k+1) = t_u(k+1)$ and then $t_u(k+1) \leq d_B + t_u(k+2)$.

Case (III): $d_u(k) = d_B + t_u(k+1)$, $d_u(k+1) = d_B + t_u(k+2)$, and $d_u(k+2) = t_u(k+2)$. Then, (34) becomes:

$$d_B + t_u(k+1) - d_B + t_u(k+2) \geq d_B + t_u(k+2) - t_u(k+2) \Leftrightarrow$$
$$\Leftrightarrow d_B \leq t_u(k+1) - t_u(k+2),$$

which is true as $d_u(k+1) = d_B + t_u(k+2)$ and then $t_u(k+1) > d_B + t_u(k+2)$.

$d_u(k+1) = d_B + t_u(k+2) \Leftrightarrow t_u(k+1) > d_B + t_u(k+2)$.

Case (IV): $d_u(k) = d_B + t_u(k+1)$, $d_u(k+1) = d_B + t_u(k+2)$, $d_u(k+2) = d_B + t_u(k+3)$. This case is analogous to Case (I).

D. Problem 2 is not submodular in the heterogeneous case.

Let $X \subset X' \subset V$ and $(b', f') \in V \backslash X'$. Consider a numerical setting consisting of a catalog $[F]$ and BSs $[B] = \{1, 2, 3, 4\}$. Let $f_1 \in [F]$, $X = \{(b_1, f_1)\}$, $X' = \{(b_1, f_1), (b_2, f_1)\}$, and $(b', f') = (b_3, f_1)$. Additionally, consider that UE u is located in the region covered by all BSs simultaneously, and the power-base SNRs are $\mathbf{G}_u = [30.0, 30.0, 10.0, 100.0]$. We consider $d_B = 10.0$ms, $M = 1$Mbit, and $W = 5$MHz.

We list below the experienced delay before and after adding a copy of f_1 to BS b_3 in allocations X and X'.

$$\begin{aligned} d_{u,f_1}(X) &= d_B + t_u(\{b_1, b_4\}) &&= 38.4\text{ms} \\ d_{u,f_1}(X \cup \{(b_3, f_1)\}) &= t_u(\{b_1, b_3\}) &&= 37.3\text{ms} \\ d_{u,f_1}(X') &= t_u(\{b_1, b_2\}) &&= 33.7\text{ms} \\ d_{u,f_1}(X' \cup \{(b_3, f_1)\}) &= t_u(\{b_1, b_2, b_3\}) &&= 32.5\text{ms} \end{aligned}$$

Then, we calculate and compare the gain for making such placement in both allocations:

$$d_B + t_u(\{b_1, b_4\}) - t_u(\{b_1, b_3\}) = 1.1$$
$$< 1.2 = t_u(\{b_1, b_2\}) - t_u(\{b_1, b_2, b_3\})$$

However, as shown in Appendix I-C, if $\bar{s}(X)$ is submodular, the following inequality must hold

$$d_{u,f}(X) - d_{u,f}(X \cup \{(b', f')\}) \geq d_{u,f}(X') - d_{u,f}(X' \cup \{(b', f')\}).$$

Therefore, $\bar{s}(X)$ is not submodular in general.

APPENDIX II
ADDITIONAL CONTENT FOR SECTION IV

In this appendix we provide the proofs for the results on the Full-Coverage scenario. We use the same notation introduced in Section IV.

A. Proof of Proposition IV.1

For every file $f \in [F]$, we generate B objects $f^{(1)}, \ldots, f^{(B)}$ with weight $w(f^{(k)}) = p_f(d(k-1) - d(k)) > 0$. We gather all objects generated this way in $\mathcal{F} = \{f^{(1)}, \ldots, f^{(B)}, \forall f \in [F]\}$. The total weight of any subset $\mathcal{A} \subset \mathcal{F}$ is $w(\mathcal{A}) = \sum_{e \in \mathcal{A}} w(e)$.

We observe that any cache allocation X can be mapped to a set $\mathcal{A} \subset \mathcal{F}$ such that $w(\mathcal{A}) = \bar{s}(X)$ and $|\mathcal{A}| = B \cdot C$. In fact, let k_f be the number of copies of file f in allocation X, then $\mathcal{A} = \{f^{(i)} : 1 \leq i \leq k_f, \forall f \in [F], k_f > 0\}$ has the desired property. The opposite also holds, any set $\mathcal{A} = \{f^{(i)} : 1 \leq i \leq k_f, \forall f \in [F], k_f > 0\}$ with $|\mathcal{A}| = B \cdot C$ can be mapped to an allocation X, such that $w(\mathcal{A}) = \bar{s}(X)$. The mapping is detailed in Alg. 4.

Algorithm 4: Mapping

input : A set $\mathcal{A}$
output: Allocation set X

```
X ← ∅
i ← 0
for f ∈ [F] do
    if k_f > 0 then
        for h ∈ [k_f] do
            X ← X ∪ {((i mod B) + 1, f)}
            i ← i + 1
```

Consider the problem:

$$\begin{aligned} \underset{\mathcal{A} \subset \mathcal{F}}{\text{maximize}} \quad & w(\mathcal{A}), \\ \text{subject to} \quad & |\mathcal{A}| = B \cdot C. \end{aligned} \tag{35}$$

This is a weight maximization problem, so a greedy algorithm finds the optimal solution $\mathcal{A}^*$ (see [33]). $\mathcal{A}^*$ can be written as $\mathcal{A}^* = \{f^{(i)} : 1 \leq i \leq k_f, \forall f \in [F], k_f > 0\}$. Suppose it is not the case, i.e., $\exists f \mid f^{(k)} \in \mathcal{A}^*$ but $f^{(h)} \notin \mathcal{A}^*$, for some $h < k$. Then, there is a set $\mathcal{A}' = \mathcal{A}^* \backslash \{f^{(k)}\} \cup \{f^{(h)}\}$, such that $w(\mathcal{A}') > w(\mathcal{A}^*)$, contradicting the optimality of $\mathcal{A}^*$.

As $\mathcal{A}^* = \{f^{(i)} : 1 \leq i \leq k_f, \forall f \in [F], k_f > 0\}$, $\mathcal{A}^*$ can be mapped to an allocation X^* with $\bar{s}(X^*) = w(\mathcal{A}^*)$. We claim that X^* is an optimal solution of Problem 2. In fact, any other allocation X can be mapped to a set $\mathcal{A}$ with $\bar{s}(X) = w(\mathcal{A}) \leq w(\mathcal{A}^*) = \bar{s}(X^*)$.

Finally, consider the ordered set of choices of GREEDYAD for Problem 2, and map them to corresponding elements of $\mathcal{A}$ (the h-th choice of a copy of f by GREEDYAD corresponds to add $f^{(h)}$ to $\mathcal{A}$). These choices are possible choices for the greedy algorithm in the problem defined in (35). It follows that GREEDYAD provides an optimal solution for Problem 2.

B. Proof of Lemma IV.2

The **necessary** part is trivial: If an allocation is optimal, it provides the minimum delay among all possible allocations.

To prove the **sufficient** part, we consider a locally optimal allocation X. We prove that X is optimal by contradiction.

Consider the problem introduced in the proof of Proposition IV.1. The optimal greedy algorithm iteratively builds a solution generating the following sequence of allocations: $\mathcal{A}_0^* = \emptyset, \mathcal{A}_1^*, \mathcal{A}_2^*, \ldots, \mathcal{A}_{B\cdot C}^*$. We observe that each $\mathcal{A}_i^*$ corresponds to a valid allocation.

Let $\mathcal{A}$ be the set corresponding to X. We order the elements in $\mathcal{A}$ in decreasing order of their weights, generating the following sequence: $\mathcal{A}_0 = \emptyset, \mathcal{A}_1, \ldots, \mathcal{A}_{B\cdot C}$. We assume that $\mathcal{A}$ is not optimal, i.e., $w(\mathcal{A}^*) > w(\mathcal{A})$. Then, there is an index h, such that $w(\mathcal{A}_h^*) > w(\mathcal{A}_h)$ and $w(\mathcal{A}_m^*) = w(\mathcal{A}_m)$, for $m < h$. Let $\mathcal{A}_h = \mathcal{A}_{h-1} \cup \{\bar{f}^{(h)}\}$. Then, there is an element $\hat{f}^{(k)} \in \mathcal{A}_h^* \cap \mathcal{A}_h^c$, such that,

$$\begin{aligned} w(\hat{f}^{(k)}) &= w(\mathcal{A}_h^*) - w(\mathcal{A}_{h-1}^*) \\ &> w(\mathcal{A}_h) - w(\mathcal{A}_{h-1}) = w(\bar{f}^{(h)}). \end{aligned}$$

Moreover, $\hat{f}^{(k)} \notin \mathcal{A}$ as $w(\mathcal{A}_m) - w(\mathcal{A}_{m-1})$ is not increasing.

Consider $h' = \max\{l | \bar{f}^{(l)} \in \mathcal{A}\} \geq h$. It holds that $w(\bar{f}^{(h')}) \leq w(\bar{f}^{(h)}) < w(\hat{f}^{(k)})$. Also, $k' = \min\{l | \hat{f}^{(l)} \notin \mathcal{A}\} \leq k$. It holds $w(\hat{f}^{(k')}) \geq w(\hat{f}^{(k)})$.

If $\tilde{\mathcal{A}} = \mathcal{A} \setminus \{\bar{f}^{(h')}\} \cup \{\hat{f}^{(k')}\}$, then $w(\tilde{\mathcal{A}}) > w(\mathcal{A})$ and $\tilde{\mathcal{A}}$ has been obtained from $\mathcal{A}$ replacing a single element, which contradicts the fact that X is locally optimal.

C. Proof of Proposition IV.3

As provided by Lemma IV.2, 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 $B \cdot C$ most popular files with a less popular file $j > B \cdot C$. The full-diversity allocation is then optimal iff it is not worthy to replace any file $i \in [B \cdot C]$ with an additional copy of a file $j \in [B \cdot C] \setminus \{i\}$. This is the case iff:

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

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 BSs jointly transmitting j. The minimum of the left-hand side of the inequality above is achieved when $i = B \cdot C$ (the least popular file in cache), 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

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

i.e., we can restrain to consider the possibility to replace the least popular of the $B \cdot C$ 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 B copies of file C with (a first copy of) file $C + 1$ does not reduce the expected delay, i.e.,

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

D. Proof of Corollary 1

We prove each part of the corollary separately:

First, we want to show that, for general topologies, if full-diversity is locally optimal, then (8) holds. Let X be a full-diversity allocation and X' be an allocation that differs from X by a single file, i.e., and $X' = (X \setminus \{(b, f_1)\}) \cup \{(b, f_2)\}$, for any $b \in [B]$ and $f_1, f_2 \in [F]$, such that $(b, f_1) \in X$, and $(b, f_2) \notin X$. Let $k_{u,f} = |J_{u,f}(X)|$ and $k'_{u,f} = |J_{u,f}(X')|$. If full-diversity is locally optimal, then:

$$\begin{aligned} &\bar{d}(X) \leq \bar{d}(X') \Leftrightarrow \\ \Leftrightarrow &\sum_{u\in[U]} \frac{1}{U} \sum_{f\in[F]} p_f d(k_{u,f}) \leq \sum_{u\in[U]} \frac{1}{U} \sum_{f\in[F]} p_f d(k'_{u,f}). \end{aligned}$$

We denote by $\mathcal{U}(b)$ the set of users covered by BS b. Notice that, $\forall u \notin \mathcal{U}(b), d(k_{u,f}) = d(k'_{u,f})$ so their contributions to the LHS and RHS of the inequality above cancel out. Similarly, all files different from f_1 and f_2 will have equal contributions on both sides, also being canceled out. Then, we can write:

$$\begin{aligned} &\bar{d}(X) \leq \bar{d}(X') \Leftrightarrow \\ &\Leftrightarrow \frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} \left(p_{f_1} d(k_{u,f_1}) + p_{f_2} d(k_{u,f_2})\right) \\ &\quad \leq \frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} \left(p_{f_1} d(k'_{u,f_1}) + p_{f_2} d(k'_{u,f_2})\right) \\ &\Leftrightarrow \frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} p_{f_2} \left(d(k_{u,f_2}) - d(k'_{u,f_2})\right) \\ &\quad \leq \frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} p_{f_1} \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right). \end{aligned} \tag{36}$$

Observe that $\forall u \in \mathcal{U}(b)$, $2 \geq k'_{u,f_2} > k_{u,f_2} \geq 0$. Then, it holds that:

$$p_{f_2}(d(1) - d(2)) \leq \frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} p_{f_2} \left(d(k_{u,f_2}) - d(k'_{u,f_2})\right). \tag{37}$$

Similarly, $\forall u \in \mathcal{U}(b)$, $k_{u,f_1} = 1$ and $k'_{u,f_1} = 0$. Then:

$$\frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} p_{f_1} \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right) = p_{f_1}(d(0) - d(1)). \tag{38}$$

Putting together (37) and (38) with (36), we obtain:

$$\begin{aligned} p_{f_2}(d(1) - d(2)) &\leq \frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} p_{f_2} \left(d(k_{u,f_2}) - d(k'_{u,f_2})\right) \\ &\leq \frac{1}{|\mathcal{U}(b)|} \sum_{u\in\mathcal{U}(b)} p_{f_1} \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right) \\ &= p_{f_1}(d(0) - d(1)). \end{aligned} \tag{39}$$

Then, we have that:

$$\begin{aligned}
&\bar{d}(X) \leq \bar{d}(X') \Rightarrow \\
&\Rightarrow \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} p_{f_2} \left(d(k_{u,f_2}) - d(k'_{u,f_2}) \right) \\
&\leq \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} p_{f_1} \left(d(k'_{u,f_1}) - d(k_{u,f_1}) \right) \Rightarrow \\
&\Rightarrow p_{f_2}(d(1) - d(2)) \leq p_{f_1}(d(0) - d(1)).
\end{aligned}$$

In particular, we can take $f_1 = B \cdot C$, $f_2 = 1$, and b such that $(b, 1) \notin X$ and we obtain:

$$p_1(d(1) - d(2)) \leq p_{B \cdot C}(d(0) - d(1)).$$

Therefore, if full-diversity is locally optimal, then (8) holds.

Second, we want to show that, for general topologies, if (9) holds, then full-replication is locally optimal. Equivalently, we prove that if full-replication is not locally optimal, then (9) does not hold. If a full-replication allocation Y is not locally optimal, then there exists b, f_1, f_2, with $(b, f_1) \in Y$, $(b, f_2) \notin Y$, such that $(Y \setminus \{(b, f_1)\}) \cup \{(b, f_2)\}$ has a smaller delay than Y. Note that every file $C < f \leq f_2$ leads to an even larger reduction to the delay, so we consider $f_2 = C + 1$ and $Y' = (Y \setminus \{(b, f_1)\}) \cup \{(b, C+1)\}$. Let $k_{u,f} = |J_{u,f}(Y)|$ and $k'_{u,f} = |J_{u,f}(Y')|$. Using a similar reasoning to the first part of the proof, we have that:

$$\begin{aligned}
&\frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} p_{C+1} \left(d(k_{u,C+1}) - d(k'_{u,C+1}) \right) \\
&\qquad > \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} p_{f_1} \left(d(k'_{u,f_1}) - d(k_{u,f_1}) \right).
\end{aligned} \tag{40}$$

Observe that $\forall u \in \mathcal{U}(b)$, $k_{u,C+1} = 0$ and $k'_{u,C+1} = 1$. Then, it holds that:

$$\begin{aligned}
&p_{C+1} \left(d(0) - d(1) \right) = \\
&\quad = \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} p_{C+1} \left(d(k_{u,C+1}) - d(k'_{u,C+1}) \right).
\end{aligned} \tag{41}$$

Also, $\forall u \in \mathcal{U}(b)$, $k_{u,f_1} = B$ and $k'_{u,f_1} = k_{u,f_1} - 1$. Then:

$$\begin{aligned}
&\frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} p_{f_1} \left(d(k'_{u,f_1}) - d(k_{u,f_1}) \right) = \\
&\qquad = p_{f_1}(d(B-1) - d(B)).
\end{aligned} \tag{42}$$

Putting together (41) and (42) with (40), we obtain:

$$\begin{aligned}
p_{C+1} \left(d(0) - d(1) \right) &> p_{f_1}(d(B-1) - d(B)) \\
&\geq p_C(d(B-1) - d(B))
\end{aligned}$$

that contradicts (9).

Therefore, if (9) holds, then full-replication is locally optimal.

Appendix III
Adaptation of Framework [17] to the Delay Minimization Problem

The idea of this appendix is to derive qLRU-Δd from the general policy qLRU-Δ proposed in [17].

The first important adaptation, is regarding the *gain function* $\sum_{u \in [U]} \sum_{f \in [F]} \lambda_{u,f} \gamma_{u,f}(X)$, where $\lambda_{u,f}$ is the request rate of file f from user u. In our setting we can consider $\lambda_{u,f} = p_f / U$ and

$$\gamma_{u,f}(X) = d_{u,f}(\emptyset) - d_{u,f}(X), \tag{43}$$

which is analogous to the "delay saving" function introduced in Section III.

We define the *marginal gain* related to the performance improvement experienced by UE u for retaining file f in the cache of BS b given an allocation X:

$$\begin{aligned}
&\Delta\gamma^{(b)}_{u,f}(X) = \\
&= \gamma_{u,f}(X) - \gamma_{u,f}(X \setminus \{(b,f)\}) \\
&= d_{u,f}(\emptyset) - d_{u,f}(X) - (d_{u,f}(\emptyset) - d_{u,f}(X \setminus \{(b,f)\})) \\
&= d_{u,f}(X \setminus \{(b,f)\}) - d_{u,f}(X) \\
&= \Delta d^{(b)}_{u,f}(X),
\end{aligned} \tag{44}$$

where $\Delta d^{(b)}_{u,f}(X)$ matches the definition of marginal delay gain (10) introduced in Section V. Since we are interested in maximizing the gain, the policy is going to behave as a random greedy algorithm that considers the marginal gain to take probabilistic caching decisions.

The policy qLRU-Δ performs move-to-the-front with probability:

$$\begin{aligned}
p^{(b)}_f(u) &= \beta \cdot \Delta\gamma^{(b)}_{u,f}(X) && \text{by [17, Eq. (6)]} \\
&= \beta \cdot \Delta d^{(b)}_{u,f}(X) && \text{by (44)} \\
&= \rho^{(b)}_{u,f}(X), && \text{by def. (11)}
\end{aligned}$$

which is the move-to-the-front probability used in qLRU-Δd, if the normalization factor is $\beta = \left(\max_{u,f,b,\mathbf{x}_f} \Delta d^{(b)}_{u,f}(X) \right)^{-1}$.

Similarly, upon a miss, qLRU-Δ inserts the file with probability:

$$\begin{aligned}
q^{(b)}_f(u) &= q^{(b)} \cdot \delta \cdot \Delta\gamma^{(b)}_{u,f}(X \cup \{(b,f)\}) && \text{by [17, Eq. (7)]} \\
&= q \cdot \delta \cdot \Delta d^{(b)}_{u,f}(\mathbf{x}_f \cup \{(b,f)\}) && \text{by (44)} \\
&= q^{(b)}_{u,f}(\mathbf{x}_f), && \text{by def. (12)}
\end{aligned}$$

which is the insertion probability used in qLRU-Δd, if we set $\delta = \left(\max_{u,f,b,\mathbf{x}_f} \Delta d^{(b)}_{u,f}(X \cup \{(b,f)\}) \right)^{-1}$ and $q^{(b)} = q, \forall b \in [B]$.

Since we are able to map all the quantities of our problem to the ones in [17], we conclude that qLRU-Δd is a special instance of qLRU-Δ which targets the delay minimization and, therefore, it enjoys the same optimality guarantees.