Design and optimization of cache systems for small cell networks

Guilherme Iecker Ricardo

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Conception et Optimisation des Systèmes de Cache pour les Réseaux Small Cell

Guilherme IECKER RICARDO

EURECOM et Inria Sophia Antipolis - Méditerranée

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur

Dirigée par : Giovanni Neglia / Petros Elia
Co-encadrée par : Thrasyvoulos Spyropoulos
Soutenue le : 09 septembre 2021

Devant le jury, composé de :
Ilenia Tinnirello, Università di Palermo, Président
Anastasios Giovanidis, CNRS Lab LIP6, Rapporteur
Francesco De Pellegrini, Université d'Avignon, Rapporteur
Daniel Sadoc Menasché, UFRJ, Examinateur
Giovanni Neglia, Inria, Directeur
Petros Elia, EURECOM, Co-Directeur
Thrasyvoulos Spyropoulos, EURECOM, Co-Encadrant

EURECOM
Sophia Antipolis

Inria

Design and Optimization of Cache Systems for Small Cell Networks

Jury:

President		
Ilenia TINNIRELLO	Assistant Professor	Università di Palermo
Reviewers		
Anastasios GIOVANIDIS	Chargé de Recherche	Sorbonne Université - Lab LIP6
Francesco DE PELLEGRINI	Professeur	Université d'Avignon
Examiners		
Daniel SADOC MENASCHÉ	Assistant Professor	Universidade Federal do Rio de Janeiro
Director		
Giovanni NEGLIA	Directeur de Recherche	Inria
Co-Director		
Petros ELIA	Professor	EURECOM
Co-Supervisor		
Thrasyvoulos SPYROPOULOS	Assistant Professor	EURECOM

Résumé. Les techniques de mise en cache ont été largement étudiées et déployées en tant que solutions puissantes pour améliorer la performance d'une grande variété de systèmes informatiques. Motivée par les nouvelles technologies et les défis émergeant des architectures cellulaires prospectives, cette thèse propose la conception et l'analyse de nouvelles techniques de mise en cache visant l'amélioration de la qualité d'expérience des utilisateurs mobiles. Nous sommes attentifs aux réseaux dits *small cell* dotés de la technologie *CoMP Joint Transmissions*. Tout d'abord, nous étudions le scénario où le placement de contenu est effectué par une intelligence centralisée conscience de la popularité des fichiers précédemment estimée et de la topologie du réseau dans son ensemble. Le meilleur placement du contenu est obtenu en résolvant un problème d'optimisation que nous approchons via un algorithme glouton efficace. Cette solution dépend d'hypothèses strictes et peut ne pas capturer la variabilité de popularité du contenu à petite échelle. Cependant, il est utile de déterminer les limites de performance et de fournir des informations sur les compromis inhérents au problème. Ensuite, nous introduisons un cadre dynamique, dans lequel chaque cache met à jour individuellement son contenu à la volée en réponse aux demandes entrantes en se basant sur des politiques de mise en cache prédéfinies. Les politiques proposées définissent un ensemble de règles probabilistes qui prennent en compte le gain de performance global de toute opération de mise à jour du cache. Notre première politique réalise une coordination implicite entre les caches et converge asymptotiquement vers la configuration de cache optimale sous des séquences de demandes stationnaires. Nous étudions également le cas où les demandes ne sont pas stationnaires et fournissons une politique qui donne des résultats pratiques satisfaisants. Enfin, nous présentons quelques expériences basées sur des simulations numériques conçues pour capturer les attributs intrinsèques de véritables réseaux *small cell*. Les résultats empiriques confirment la tendance asymptotiquement optimale de notre première politique. Nous observons que nos politiques proposées atteignent des niveaux de performance souhaitables lorsqu'elles sont exposées à la fois à des séquences de demandes stationnaires et non stationnaires. De plus, nos politiques surpassent les autres politiques de pointe dans tous les scénarios testés.

Mots clés. Mise en cache, algorithmes distribués, optimisation, stochastique

Abstract. Caching techniques have been extensively studied and deployed as powerful solutions to performance improvement in a wide variety of computer systems. Motivated by new technologies and challenges emerging from prospective cellular architectures, this thesis proposes the design and analysis of novel caching techniques targeting the improvement of mobile users' quality of experience. We are particularly attentive to small cell networks enabled with Coordinated Multi-Point (CoMP) Joint Transmissions (JT) technology. First, we study the scenario where content placement is performed by a centralized intelligence aware of previously estimated files popularities and the whole network topology. The best content placement is obtained from solving an optimization problem that we approximate by an efficient greedy algorithm. This solution depends on strict assumptions and may fail to capture short-scale content's popularity variability. However, it is useful to determine performance bounds and to provide insights on the problem's inherent trade-offs. Then, we introduce a dynamic framework, where each cache individually updates its content on-the-fly as a response to arriving requests based on pre-defined caching policies. The proposed caching policies define a set of probabilistic rules that take into account the overall performance gain of any cache update operation. Our first policy achieves implicit coordination between caches and asymptotically converge to the optimal cache configuration under stationary request sequences. We also study the case where requests are non-stationary and provide a policy that provides satisfactory practical results. Finally, we present a set of experiments based on numerical simulations designed to capture intrinsic attributes of real small cell networks. The empirical results confirm the asymptotic convergence to an optimal solution of our first policy. We observe that both proposed policies achieve desirable performance levels when exposed to either stationary or non-stationary request sequences. Furthermore, our policies outperform other state-of-the-art policies in all tested scenarios.

Keywords. Caching, Distributed Algorithms, Optimization, Stochastics

Acknowledgements

First of all, I would like to thank my advisors, Giovanni and Akis, for their generosity and all the invaluable advice and lessons. I am really grateful to have had the chance to share my work environment with so many brilliant and kind people such as Alison, Jonas, Lorenzo, Marina, Matteo, Naser, Roya, and Thomas, from EURECOM, and Abhishek, Dimitra P., Foivos, Max, Mikhail, Natasha, and Vera, from Inria. Special thanks to Dimitra T. and Fionn, for witnessing (too) many of my near-death experiences. Also, to Chuan, for teaching me that empathy, kindness, and hard work are the best combination to thrive in academia (these do not apply to ping-pong though).

Some people keep supporting me regardless the distance. In my family, I would like to thank my grandparents, Dulce, José, Maria, and Waldemir, my godparents, Rosângela and Joselias, my cousin, Flávia, and my uncles, Armênia and Leopoldo. I am also fortunate enough to have life-long friendships and I am really grateful to them for never giving up on me, no matter what, specially Fernando, Lucas, Marcel, Maria Thereza, Matheus, and Túlio. Thank you all.

Luckily, I was able to find amazing people who also participated in my journey and helped me go through all the adversities of daily life. Thanks Sil and Lucas, for being with me since day one, and Gui, Joy, and Luigi, for being my family here whenever I needed them. I would also like to thank Ismail for showing me how fun life can be if we do things with passion and if we accept ourselves the way we are, and Harald, for having this habit of saving my life.

Finally, I would like to thank my sister, Beatriz, for being my first friend, my accomplice, and for always being there for me, and to Samuel, for bringing the joy I needed to help me go through these last months. Most importantly, I would like to dedicate this thesis to my parents, Hélio and Dulceléa, my best friends and biggest supporters. I will never be able to thank enough for all the effort and sacrifices they have made in order for me to be here. This thesis is *to them, for them, and because of them.*

Contents

List of Figures

Chapter 1

Introduction

This introductory chapter provides an overview of the main technologies and challenges that commonly motivate and shape the different problems studied in this thesis. We present a summary of these problems and list the set of goals that we wish to achieve. Then, we discuss the existing variants and solutions in a comprehensive list of related work. Finally, we outline the thesis contributions and present how they are organized in the next chapters.

1.1 Background and Technologies Overview

In this section we cover in detail the technological background of the problems we study in the next chapters. First, we provide an overview of general cache systems and, then, we discuss how they can be deployed in small-cell networks. Finally, we introduce the Coordinated Multi-Point (CoMP) technology and emphasize what are the key aspects for our models and solutions.

1.1.1 Cache Systems

Caching techniques have been studied as performance improvement solutions for a large variety of data-oriented applications; from "high-level" applications, e.g., web (browser) caching [2], to "low-level" computer systems, e.g., hierarchical memory schemes [3]. In its broadest definition, *cache* is a hardware or software component that is able to store and/or serve data at a lower cost, e.g., smaller retrieval latency, in comparison to the original data server. A fundamental characteristic of cache systems is that the available storage capacity is more limited than in the original data servers, so only a small fraction of the whole catalog of files can be cached. The performance boost essentially comes from

the fact that a selection of the most "useful" files (according to a well defined objective) can be cached and served at a lower cost. The subset of files comprising the cache current state is referred to as the *cache allocation* (or *cache configuration*).

Example 1 (Standard Web Caching)**:** Consider a classic internet-based client-server web application. The standard data flow is the following: An HTTP request is issued by the client, e.g., web browser, and go all the way through the internet to the back-end server; then, the server sends back the requested page to the client. If caches are deployed on the client side, whenever a new page needs to be retrieved from the server, the browser may opt to retain a local copy of that page at its cache. Then, we introduced a shortened data flow: Whenever a user revisits a web page, it may be rendered immediately directly from the cache, without being downloaded from the server. We illustrate, in Figure 1.1, how the data flow changes with the deployment of a cache system.

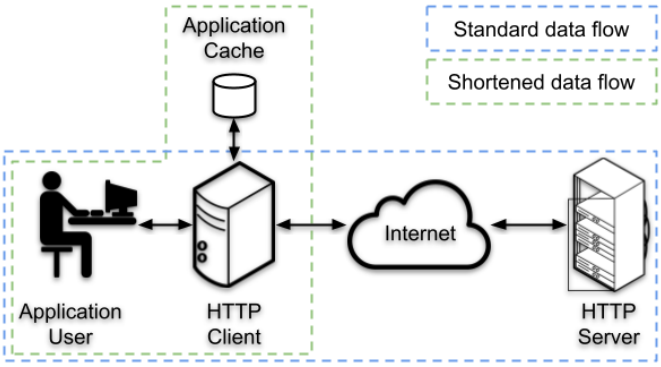
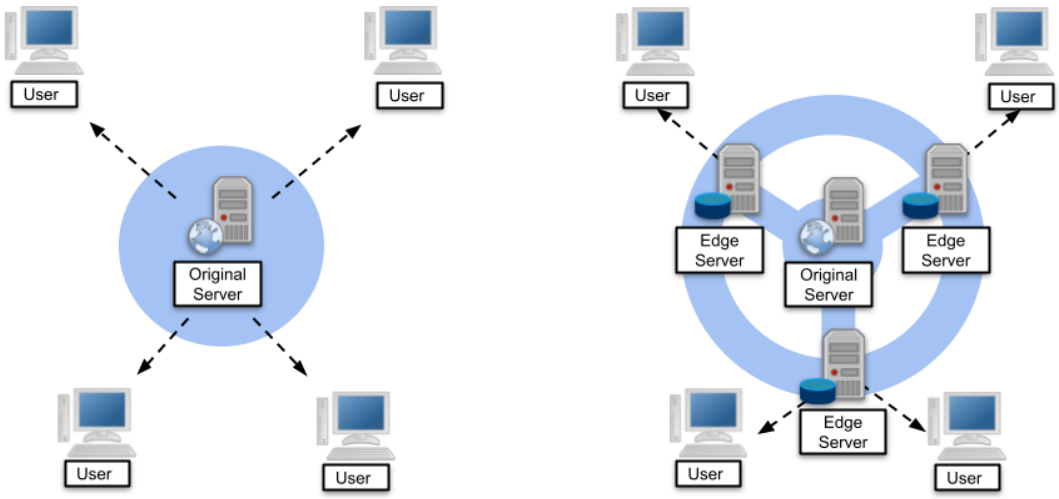


Figure 1.1 – Data flowchart for a simple web caching example.

The primary advantage of web caching, as it is the case for the majority of cache systems, is to reduce the retrieval latency and, consequently, improve overall Quality of Service (QoS) experienced by the application's clients. There are many other ways computer systems and applications can profit from cache systems. For example, additional positive effects of web caching include the ability to (i) alleviate the original servers' access traffic load, in case an excessive number of users is simultaneously querying the server [4] and (ii) reduce the traffic in the intermediate links between the users and remote servers, preventing congestion and misuse of network resources [5]. We discuss a more complex cache system in the next example.

Example 2 (CDN Caching)**:** In Content Delivery Networks (CDNs) [5], part of the content provided by the original server is replicated into many different edge servers, sometimes also referred to as Points of Presence (PoPs). Therefore, we move from a centralized

single-server architecture (Figure 1.2a) to a more distributed one, where content is placed closer the end user (Figure 1.2b). Edge servers deployment strategy depends on the CDN's business model, e.g., based on geographical and economic characteristics. Besides the advantages already discussed for the web caching example, this server replication architecture may also provide other benefits, e.g., overall internet traffic reduction, implementation of exclusive services and content to different edge servers, and so on.



(a) Classic Client-Server Architecture (b) CDN Architecture.

Figure 1.2 – Change of paradigm: From classic client-server to CDN architecture.

CDN caching can be seen as a generalization of the standard web caching technique, where the caches at the edge-servers are able to serve multiple users with web pages and other types of frequently downloaded files, such as music, videos, etc. In this context, caching decisions tend to be more complex because caches take into consideration the preference of multiple concurrent users, which is shaped by social and geographic aspects.

We discuss next how to evaluate cache solutions' performance and what we should consider in order to design efficient systems.

Performance of Cache Systems

The most intuitive way to evaluate the performance of a cache system is through the (cache) *hit rate* (or *hit ratio*) metric. The *cache hit* is the event where a requested file is found at the cache and hit rate is the number of hits relative to the total of requests. In general, a high hit rate indicates that a considerable amount of requests are being served

directly from the cache, which intuitively suggests an efficient cache system.[1]

Therefore, considering hit rate as the main performance metric, a natural goal for any caching strategy is to maximize the hit probabilities (of the lowest cost/closest) caches. A major obstacle in doing so, is that we do not generally know which file will be requested next. For this reason, some metric of file *popularity* is needed (or must be learned), that captures the overall user preferences and/or request patterns. Files popularities are central to the design and performance evaluation of cache systems, because they define the probability of each file to be requested in the future. In an alternative web caching implementation from Example 1, if the browser somehow knows before-hand an estimation for the pages popularities, it could prefetch these pages and place them into the cache and provide faster retrieval even upon a page's first access.

The hit rate is widely adopted in related literature as a very versatile performance metric. However, for some other applications, it may not be the most suitable metric to evaluate the underlying cache system's performance. For example, in the coded-caching framework [6, 7], the concept of cache hit is not enough to characterize good QoS and to capture the system's trade-offs. As we will discuss in the next chapters, we can define QoS for small-cell networks in terms of the average delay experienced by the user to retrieve the requested file. In this case, achieving a high hit rate does not necessarily mean that we are providing satisfactory QoS. Therefore, in these cases, popularities alone are not enough to decide which files should be cached; other parameters must be considered as well, e.g., the network topology.

Now, we can discuss cache solution strategies, i.e., how cache content can be managed, targeting the optimization of a desired performance metric. Until the end of this subsection, we exclusively focus on the hit rate as a performance metric, because it provides a more intuitive notion of performance.

Cache Solution Frameworks

We categorize caching strategies into two different groups: *static* and *dynamic* cache (solution) frameworks. Each cache framework establishes the general operation and constraints of the underlying cache system.

In the *static cache framework*, the entire cache allocation is updated all at once every fixed time interval. In this framework, there is a centralized oracle that is aware of the whole system topology and parameters. Usually, in this kind of approach, the cache system operation is split into two main phases:

[1] In a symmetric way, we can define the *cache miss* (i.e., the event where a requested file is not cached). In this case, an efficient cache system may be characterized by a low cache miss rate.

1. the *measurement phase*, where requests are observed and application relevant statistics, e.g., files popularities, users activity level, etc., may be estimated;

2. the *placement phase*, where files are fetched from the original data servers and effectively stored at the cache server.

We focus on the placement phase. The idea is to find the cache allocation that is able to optimize a performance metric of interest, assuming the statistics from the previous measurement phase are available.

Example 3 (CDN Hit Ratio Maximization)**:** In some related work, CDNs are studied in the static framework (see Example 2), where the two phases take place on a daily-basis. Usually, the measurement is performed during the day, when users are more active and the data traffic is more intense (generating more data and more reliable statistics estimations). Then, during the night, when the network resources are more abundant, the cache content is updated, based on the previously estimated statistics. If we wish to maximize the hit rate at one of the edge cache servers in this scenario, the solution is rather trivial: it needs to cache all the most popular files among its users. If the estimated popularities from the previous day represent the files probabilities of being requested, by caching the most popular files, we maximize the expected cache hit rate of the requests on the next day.

In the *dynamic cache framework*, the cache content is updated on-the-fly as new requests arrive. In this thesis, we consider that the cache is structured as an ordered queue and may admit files metadata or auxiliary data structures. Upon every request, the cache may perform one or more of the following operations: (i) *insert* a new file, (ii) *evict* (or remove) a cached file, or (iii) *move* a cached file from its current position in the queue to a new position. The set of applicable operations and update rules are specified by *online caching policies*, commonly referred to as just policies throughout this thesis. Caching policies are designed aiming to converge to an allocation that is able to provide a good performance on average.

One important aspect for the design of dynamic solutions is to understand the underlying request process. In most of the related literature, request processes are often modeled under the Independent Reference Model (IRM). In IRM, the request process for each file is represented as a Poisson arrival process, with a rate that is related to its popularity, and independent from the other files. We also refer to IRM-based request processes as *stationary* processes. Although real systems usually have *non-stationary* request processes, where files popularities may change drastically in short periods of time [8], IRM is still able to provide useful insights.

Example 4 (LFU Caching Policy)**:** Revisiting the CDN edge cache server hit rate maximization example, now in the dynamic framework, the classic *ideal* LFU policy is well-known to achieve optimal results under IRM. The idea behind LFU is that the cache server maintains a counter associated to every file in the catalog accounting the number of times it has been requested. Whenever a file that is not cached is requested, the least-frequently-used file, i.e., the one with the smallest counter, is evicted from the cache to make room for the new file. In this case, the LFU policy was named after its least-frequently-used eviction rule.

We illustrate LFU's operation in Figure 1.3: In step 1, file "D" is requested and its counter is incremented by 1, moving one position up in the queue (swapping places with file "C".) Then, in step 2, file "B" is requested, its counter is updated, but it does not change its position in the queue. In step 3, file "F" is requested and it is not in the cache, then the least-frequently-used file (the file at the rear of the queue), i.e., file "E", is evicted to make room for file "F" to be inserted (with a brand new counter.)

After some "warm-up" time (characterizing the transient phase), the cache converges to the optimal allocation, i.e., storing the most frequently used (or the most popular) files.

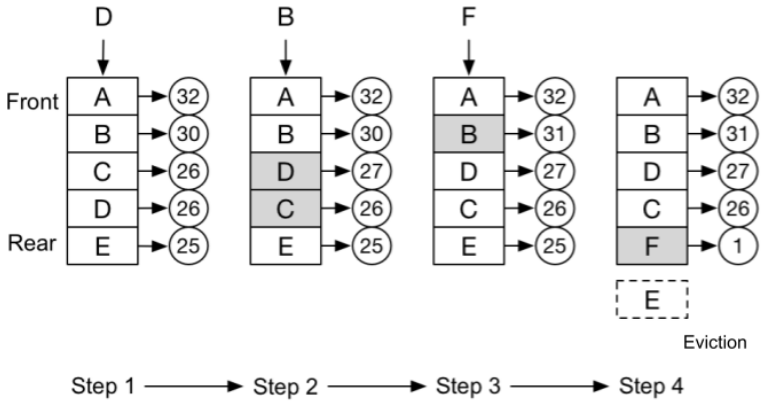


Figure 1.3 – Practical LFU Operation Example (scheme extracted from [1])

Although the ideal LFU policy is optimal under IRM, its implementation is computationally costly. For some applications, keeping track on the request frequency of cached files may be hard or infeasible.[2] Moreover, according to [9], real request processes have strong *temporal locality* property, where if a file is requested, then it is likely that

[2]It is also common to find the *practical* implementation of LFU, where caches maintain counters only for the cached files. This simplification reduces the computational resources requirements for LFU deployment, although the optimality guarantee may no longer hold.

the same file will be referenced again in the near future. There is temporal proximity between consecutive requests to the same file. In this case, it is common to make efforts to cache a copy of recently requested files to reduce the latency of subsequent requests. The LFU policy may fail at capturing this notion of time, given that the counters reflect the general request frequency and they are not sensitive to changes in popularity in a short time scale.

Example 5 (LRU Caching Policy)**:** In order to handle the issues pointed out previously, i.e., (i) to provide a lighter computational implementation and (ii) to deal with non-stationary request processes being sensitive to temporal locality, the LRU policy is a strong candidate to replace LFU in more practical systems.

The LRU implements the following rules: Whenever a non-cached file is requested, it is inserted at the front of the cache, pushing down in the queue all the other cached files. If the cache is already full, the least-recently used file, i.e., the one at the rear of the cache is evicted to make room for the new file. If a cached file is requested, it is moved from its current position to the front of the cache, also shifting the in-between files one position down in the queue.

We illustrate LRU's operation in Figure 1.4: Files from "A" to "E" are requested in sequence and inserted into the cache in that order, always pushing back older files one position at every new insertion. Then, file "F" is requested and, because the cache is full, the least-recently-used file (at the rear), i.e., file "A", is evicted and "F" is inserted at the front. Then, file "C" is requested and, since it is already in the cache, it is moved from its current position (second-to-last) to the front of the cache. Finally, file "G" is requested and is inserted at the front of the cache, causing file "B" eviction.

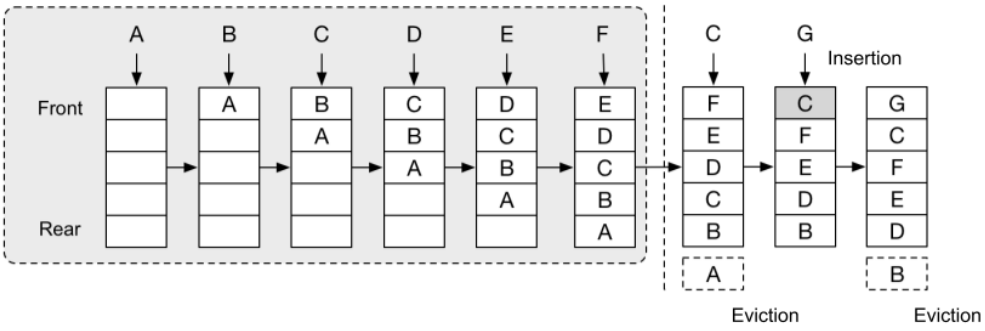


Figure 1.4 – Practical LRU Operation Example (scheme extracted from [1])

LRU does not require additional data structures and higher level computer operations, so it tends to be computationally lighter than LFU. Moreover, the least-recently-used

eviction rule keeps files in the cache while they are still likely to be requested, promising good results under real request processes. Due to these advantages, LRU and its variants, e.g., qLRU (which inserts new content with probability q), are widely deployed in real systems. Besides, our proposed policies, which we will discuss in Chapter 4, are built on top of LRU's basic operation rules.

1.1.2 Cache-Enabled Small-Cell Networks

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 [10], 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 [11]. For example, the standard 3G/4G macro-cell topology will be enriched by a large number of overlapping and often heterogeneous cells (e.g., femto, pico), in order to improve both coverage and capacity [12].

On top this architecture, *network slicing* is a technique that allows virtualization and sectorization of physical resources, e.g., routing and package switching, bandwidth, and storage capacity. Network slicing enables customized and dedicated infrastructure to specific applications and services, offering a profitable business model for network operators to be considered in the design of 5G and beyond cellular architectures [13]. For example, content providers, e.g., CDN operators and video streaming companies, may reserve their own virtual slice on a cellular network comprising storage capacity in order to empower their data distribution services with caching capabilities. By implementing cache systems closer to the mobile users, companies are able to serve content with much lower latency and, consequently, provide better QoS [14].

In this thesis, we consider a dense cellular network, where a significant fraction of users is "covered" by several base stations (BSs), whose cells are said to "overlap." BSs are connected to the back-end servers through a high-latency backhaul network (also called core network) and are able to fetch content in order to be served to mobile users. We assume that network operators and content providers interplay closely, for example via network slicing so that application-level cache systems may be deployed at every BS [15]. By doing so, BSs function as front-end servers to the application users such that cached content may be served directly from the BSs, promptly being transmitted through the wireless channel.

Figure 1.5 shows the classic cellular heterogeneous architecture: "multiple tiers (or layers) of networks of different cell sizes/footprints and/or of multiple radio access technologies" [16], leading to overlapping cells. In this case, we have a macro-cell and its

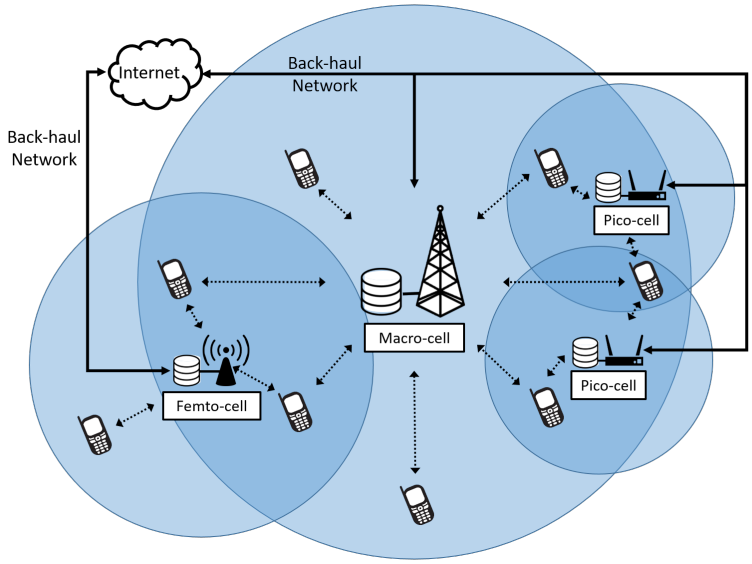


Figure 1.5 – Classic cellular heterogeneous network with macro-, pico-, and femto-cells.

subjacent smaller cells. In the same figure, we see that the different BSs are equipped with caches and are independently connected to the internet via the backhaul network. Moreover, user equipments (UEs) may be located in the overlapping coverage area of multiple BSs.

On top of a small-cell network, the cache system must be designed to optimize a performance metric of interest. Since we are interested in providing better QoS to mobile users, we focus on the average delay to serve a request (for a UE to download a desired file) as a user-centered metric. Intuitively, if the requested file is cached, it will be served faster, so, in principle, the hit ratio may still be a reasonable performance metric to be adopted. However, as we discuss in the next subsection, we add to our model a set of cooperative transmission technologies that promise to provide an even better performance (i.e., smaller delays) for content delivery in small-cell networks. We emphasize that, in this case, higher hit ratio may not necessarily imply smaller experienced delay.

1.1.3 Coordinated Multi-Point (CoMP) Technologies

In standard macro cellular networks, e.g., LTE 3G/4G architecture, UEs at the cell borders experience lower throughput than those closer to the BSs due to inter-cell interference. This issue cannot be solved by simply increasing the transmitted power, for example. Therefore, in order to provide a more even user experience throughout the whole network, BSs must be able to (i) handle inter-cell interference and (ii) reduce the gap between “cell edge” and “average” throughput.

This was the motivation behind the conception of Coordinated Multi-Point (CoMP) [17] technology. The term CoMP refers to the group of techniques where nodes (e.g., BSs) *coordinate* or *cooperate* to mitigate and/or to exploit interference of the physical layer (PHY). CoMP techniques are commonly categorized into three different groups: (i) Dynamic Point Selection (DPS), (ii) Coordinated Scheduler / Coordinated Beamforming (CS/CB), and (iii) Joint Transmissions (JT). In this thesis we focus exclusively on CoMP JT, where, in short, two or more BSs coordinate their transmissions so the combined received signal at the UE has enhanced power. The enhanced received power reflects in a better Signal-to-Noise Ratio (SNR), which produces higher transmission rates and, consequently, the UE experiences smaller delays to obtain the requested content.

Unfortunately, there are many incompatibilities between CoMP techniques and LTE-like legacy systems that make CoMP performance far from its theoretical predictions or even infeasible to be deployed on top of already existing infrastructures. However, as suggested by [18, 19], 5G and beyond cellular networks can be designed to implement the necessary infrastructure, particularly on top of heterogeneous small-cells architectures, with special effort to tackle synchronization in time and frequency, provision of accurate channel state information (CSI) to the transmitter, and user scheduling and precoding.

On top of the small-cell network discussed in Section 1.1.2, we assume a distributed implementation of CoMP JT: The network obtains channel information from the UEs over a feedback link, whereas CSI synchronization control messages are directly exchanged between BSs, e.g., via optical X2 interface [20]. By sharing the CSI related to a common UE, the neighboring BSs are able to determine how beneficial CoMP JT is in this case. Then, they dynamically self-organize into small cooperation groups (this process is called *CoMP clusterization*) in order to jointly transmit data to the UE.

1.2 Goals and Objectives

This thesis investigates how to design cache systems deployed on top of small-cell networks aiming to provide users better QoS. Our primary goal is to understand how the combinatorial structure emerging from the small-cell architecture along with the performance boost elements provided by CoMP influence caching strategies. Although we explore both solution frameworks detailed in Section 1.1, we are mostly interested in answering how efficient dynamic caching can be. In other words, we want to answer whether the implementation of distributed online policies can converge to the optimal allocation resulting from the static optimization. In order to systematically approach the overall problem, our objectives are organized as follows:

1. Development of efficient algorithms to solve the static content placement optimization problem and evaluation of how the optimal solutions are affected by the system's parameters.

2. Explore the open question regarding the existence of general (and computationally efficient) distributed strategies for small-cell network coordination, which are able to provide guarantees on global performance metrics.

In the next section, we discuss some related problems and caching solutions proposed in other works in the scientific literature.

1.3 Related Work

In this section, we provide a bibliographic review on caching solutions, with a focus on networks of caches. We cover the related work following the static and dynamic taxonomy introduced in Section 1.1.1. In parallel, we explore some other caching problem variants that will provide more insights on the big picture of how caching is being considered in different applications and domains.

1.3.1 Static Caching Solutions

The idea of statically coordinating the placement of content in a network of cache servers recently gained popularity when the authors in [21] and its extension [7] investigated such problem under the name of *FemtoCaching*. Assuming that files have known popularities and are requested according to IRM, the FemtoCaching problem is to find the content allocation that minimizes the retrieval delay. Although the FemtoCaching problem was proven to be NP-hard, the authors approached it as a submodular optimization problem and efficiently solved it via a greedy algorithm, with $\frac{1}{2}$-approximation guarantee.

To the best of our knowledge, [22] 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 heuristic with Maximum Ratio Transmission and a second approach based on Zero-Forcing BeamForming. However, both approaches lack for theoretical optimality guarantees. Introduced by [23], 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. Reference [24] considers two different CoMP techniques, i.e., joint transmission and parallel transmission, and derives formulas for the hit rate using tools from stochastic geometry.

Authors of [25] included the bandwidth costs in the formulation, and proposed an on-line algorithm for the solution of the resulting problem. This line of work has been further extended in [26], which also considers the request routing problem. In [27], the authors generalized the approach of [7, 21], providing a formulation for the joint routing and placement problem that maximizes the hit ratio. The routing-caching problem was later revisited in [28, 29]. Still in the joint optimization context, other problem variants consider different optimization metrics, e.g., energy saving [30, 31], and variables, e.g., user association [32–35]. The latter reference also considers content recommendation, which is playing an important role in nowadays applications and cache systems design.

If we look at the application layer and consider recommendation systems solutions, [36] and [37] explored the joint optimization of content placement and recommendation. The idea is that the hit rate might increase if users accept the recommendation of an already cached alternative content. This kind of problem has common elements with similarity caching [38], an emerging caching framework that considers the benefit of serving a similar cached content in exchange for some performance reduction.

Reference [39] revisited the optimal content placement problem within a stochastic geometry framework and derived an elegant analytical characterization of the optimal policy and its performance. In [40] the authors developed a few asynchronous distributed content placement algorithms with polynomial complexity and limited communication overhead (communication takes place only between overlapping cells), whose performance was shown to be very good in most of the tested scenarios.

When files have different sizes, the problem (that was already NP-Hard in its homogeneous-sizes variant) becomes significantly harder to approximate. In [41], a computationally low-cost heuristic is proposed to find good solutions, although no optimality guarantees are provided. A natural approach is to map this heterogeneous-sizes problem variant to the Submodular Multiple Knapsack Problem (SMKP) or, more generally, to the Submodular General Assignment Problem (SGAP). Specifically, for SMKP, an "unfeasible" greedy algorithm with optimality guarantees was proposed by [42] (we discuss this idea in Chapter 3). Additionally, in the context of streaming algorithms, [43, 44] proposed a computationally efficient algorithm with a satisfactory optimality guarantee that is enough to motivate its application to recommendation systems, for example.

1.3.2 Dynamic Caching Solutions

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. In any case, reliable popularity estimates over small geographical areas may be very hard to obtain [45]. 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, e.g., the celebrated Che's approximation [46, 47].

Another downside of static centralized solutions is that, in a dense cache network, having a centralized oracle aware of the entire topology may not be feasible due to the network structure complexity. Dynamic solutions with 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.

Although they are devoted to a single-cache problem, [48, 49] provided important insights for our proposed solutions, mainly when we discuss about scenarios with heterogeneous file sizes. The authors proposed an online caching policy originally designed to minimize HD-RAM systems service time, that was later extended to general utility functions. Similarly, [50, 51] proposed a simulated annealing approach, which, in turn, was adapted to an online caching policy.

Considering a dense cellular network, [52] introduced two caching policies: MULTI-LRU-ONE and MULTI-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.

Authors in [53, 54] proposed general framework to evaluate the performance of online policies in systems with multiple caches. In [55], the authors designed a distributed algorithm based on Gibbs sampling, which was shown to asymptotically converge to the optimal allocation. In [56], the authors proposed 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 a distributed online policy with provable convergence properties for the FemtoCaching setup.

Non-IRM request processes, where files have time-varying popularities, were studied for single-cache scenarios in [57]. Later, authors of [58] proposed a probabilistic approach that outperforms other adaptive policies, including kLRU, under different trace-based request processes.

In the heterogeneous-sizes problem variant, to the best of our knowledge, the current scientific literature lacks for provably efficient dynamic solutions. In the single-cache scenario, [59] proposed a caching policy that is based on the greedy criterion for hit rate maximization. As we mentioned earlier, [50,51] proposed a simulated annealing approach, which, in turn, was adapted to an online caching policy. However, when different file sizes are considered, a common work-around is to split files into chunks of equal size, as proposed in [60]. This approach is particularly suitable to video streaming applications. For example, a modification of LRU is proposed in [61] for a single-server setup and [62] introduce an MDP-based policy for networks of caches.

1.4 Contributions and Thesis Outline

Now we provide an outline of this work and we briefly summarize the contributions of each of the following chapters.

In Chapter 3, we model the content placement of the static framework as an optimization problem to minimize the average delay in small-cell networks. We prove the optimization problem is NP-Hard and propose a greedy algorithm to find an approximate solution. Assuming the network's SNRs are homogeneous, the problem can be expressed as a maximization of a monotone, submodular function subject to partition matroid constraints, which grants the greedy algorithm a $\frac{1}{2}$-approximation guarantee. We study the special case where all BSs overlap and extract important insights on the solutions characteristics. The main contributions of this chapter are:

- We provide more insight on the static problem's solution by studying a simple scenario that we call full-coverage. In this case study, we prove conditions for the optimal caching strategy to consist of replicating or diversifying contents throughout the network.
- 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 the same 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.

Part of the work included in Chapter 3 has been published/submitted in [63–65].

In Chapter 4, based on the static optimization problem defined in Chapter 3, we propose a novel general-purpose caching policy, qLRU-Δ, that asymptotically converges to an optimal allocation under IRM request process. We observe the convergence for the hit rate maximization and average delay minimization cases. In the end, with a few modifications to its operation, we show that the policy may also converge to the optimal allocation in the case where files have heterogeneous sizes. In another special case, we tackle non-stationary request processes by proposing a new caching policy 2LRU-Δ, that promises good results in practice. The main contributions of this chapter are:

- We propose a distributed online caching policy, qLRU-Δ, that, under a stationary request process, achieves an optimal configuration as the parameter q tends to 0. In this policy, BSs use only local information to update their cache states, taking a probabilistic drift towards improving the problem's objective.

- We show how the policy can be adapted to tackle the hit rate maximization problem and average delay minimization problem and show empirically its convergence via simulations.

- We also propose 2LRU-Δ policy that addresses the problem of non-stationary requests, offering better performance in real scenarios.

- We propose a variant of qLRU-Δ that is able to handle the cases where files have heterogeneous sizes. We call this policy qLRU-HS and prove that, when we consider the performance gain relative to the file size (i.e., its cost-benefit), it converges to an optimal allocation when q tends to 0

Part of the work included in Chapter 4 has been published/submitted in [64–66].

In Chapter 5, we first discuss the convergence of qLRU-Δ for different performance metric and under different experimental setups. Then, we evaluate the performance of qLRU-Δ's variant, qLRU-Δd and qLRU-Δh, for different network density levels, request processes, and SNRs regimes. Finally, we consider the special case where files have different sizes and observe qLRU-HS convergence and performance. Finally, we compare the policies performance with other policies from the literature, using the optimal static allocation as a baseline. The main contributions of this chapter are:

- Using an extensive set of simulations, we demonstrate qLRU-Δ's convergence properties, and we observe both its ability to outperform other state-of-the-art policies in all considered scenarios.

- We also show empirically that 2LRU-Δ outperforms other policies, including qLRU-Δ, for the case where files are requested according to a non-stationary process.
- We propose a series of experiments to study qLRU-HS's convergence in practice as well as its performance with respect with other solutions from the literature.

Part of the work included in Chapter 5 has been published/submitted in [63–66].

Chapter 2

System Model and Operation

2.1 Network Model

In the rest of this thesis, we consider a small-cell network, which is a simplification of the heterogeneous architecture introduced in Section 1.1.2 where all layers equally participate in the content delivery process. Therefore, we do not make any distinction between cells, i.e., macro, femto, and pico cells play the same role in the content provision. Therefore, we define the *CoMP-aided cache-enabled small-cell* (CCSC) network model as the small-cell network architecture deploying a cache system at the BSs level and empowered with dynamic CoMP JT technology.

A generic instance of CCSC network consists of a set $[B]$ of base stations (BSs) arbitrarily located in a given area $A \subseteq \mathbb{R}^2$, where $[n]$ denotes the set $\{1, \ldots, n\}$. Moreover, there is a set $[U]$ of user equipments (UEs) spread across area A that can connect to at least one BS.

We abstract the downlink channel, including fading effects (such as geographic barriers and secondary interference sources), by encapsulating all the physical characteristics into the Signal-to-Noise Ratio (SNR) quantity. Let $V_u^{(b)} \in \mathbb{R}^+$ be the SNR of the wireless channel between BS b and UE u. If the channel SNR is below a minimum SNR value, $V_{\min}$, we assume u and b cannot communicate, and set $V_u^{(b)} = 0$.

Because of the high density of BSs, each UE u will, in general, be within communication range of multiple BSs. We denote by $I_u = \left\{ b \in [B] : V_u^{(b)} > 0 \right\}$ 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.

The most important aspects of CCSC networks is the way UEs are connected to the BSs, which, in turn, will help describe how data should be allocated in the BSs and available to the UE. It is very useful to represent small-cell networks as a bipartite

graph. We provide an example for $B = 3$ BSs in Figure 2.1. There are two separate groups of nodes, BSs and UEs, interconnected by edges representing the actual network's neighboring relationships. It may be useful to assign weights to the edges and nodes of such graph trying to capture other system's characteristics, e.g., edge weights are expected download delays or available bandwidth, and node weights are UEs activity levels (probability to generate a request). Whenever we mention the *network topology* or *network of caches*, we are actually making a reference to the bipartite graph structure with its possible weights.

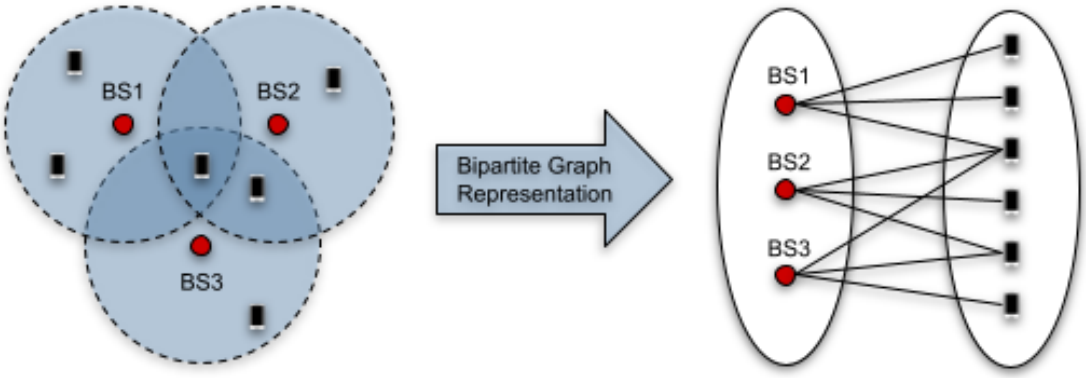
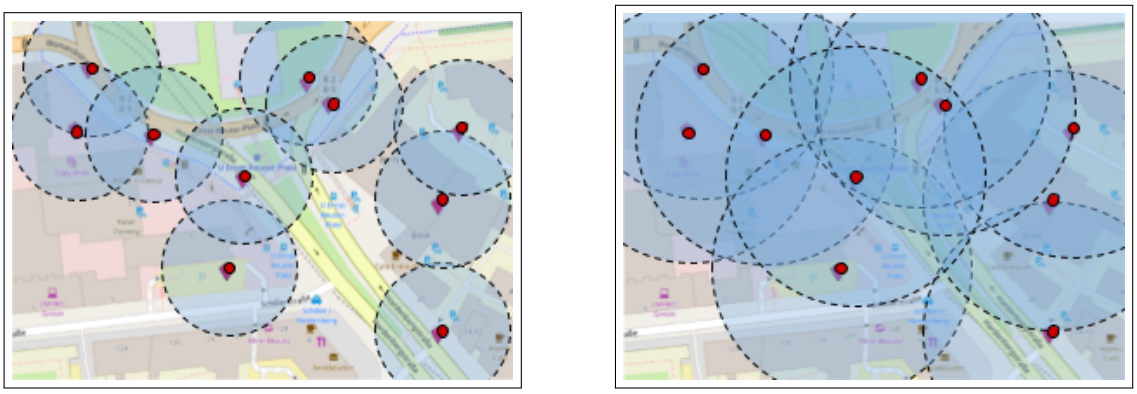


Figure 2.1 – Example of a CCSC network with $B = 3$ BSs and its bipartite graph representation.

We can control the cells sizes by changing the transmission power of each BS. Therefore, a larger transmission power defines a larger coverage area, which in turn, will provide a network with highly overlapped coverage areas. We define the *density level* ρ of a network as the average number of BSs that each UE is connected to. In highly dense networks, UEs are connected to many BSs on average and this affects (i) the number of possible sources to download files from and (ii) the number of CoMP JT opportunities (also how many BSs are available to participate in CoMP JTs). If we consider a fixed number of UEs, for different density levels, we observe the formation of *distinct user areas*, where we say that two areas are distinct if the covered UEs are in the transmission range of different sets of BSs.

2.2 The Berlin Network

As an example of a CCSC network, we show in Figure 2.2 the *Berlin* network, where $B = 10$ T-mobile BSs are located in the city of Berlin. This scheme is extracted from [67]. We also use this network in our simulations described in Chapter 5.



(a) Small coverage area – small density

(b) Large coverage area – large density

Figure 2.2 – Berlin network BSs position with different coverage area sizes.

Also in Figure 2.2, we show how much more overlap is created (i.e., how much denser the network gets) by moving from a case where BSs have small coverage areas (Figure 2.2a) to a case with larger coverage areas (Figure 2.2b). In Table 2.1, we see how Berlin network density changes and its associated number of distinct user areas as we increase the coverage area.

Coverage Radius [m]	10	25	50	100	150	200	250
Density ρ [BSs/UE]	1 (No overlap)	1.1	1.7	3.5	5.9	9.4	10 (Full overlap)
Number of distinct areas	10	17	37	78	66	17	1

Table 2.1 – Berlin network: the radius, in meters, defining the BSs circular coverage areas and their corresponding network density ρ, in BSs/UE.

2.3 Content Delivery

In CCSC networks, the content delivery operation depends on the cache configuration as well as the transmission conditions (SNRs) experienced by the UE at the moment when the request for content is posed. Because CoMP techniques are supported, all the neighboring BSs caching the file may coordinate to jointly transmit it to the UE. We describe the network operation as follows:

1. When a UE has a request, it broadcasts an inquiry message to its neighboring BSs.

2. Then, according to the current cache state, there are three possibilities:

(a) Cache miss: No cached copy of the requested content was found in the neighborhood. Then, the UE sends a direct request to one of its neighboring BSs, which will need to retrieve it from the content provider's back-end server.

(b) "Sufficient" cache hit: One or more copies of the requested content were found in the neighborhood. Then, the UE sends an explicit request to download the content to one (or more) of the neighboring BSs caching it.

(c) "Insufficient" cache hit: One or more copies of the requested content were found in the neighborhood, but for some reason, it is worthy to choose one additional BS to retrieve a copy of the content from the backhaul network and jointly transmit the file along with the BSs already caching the content.

We illustrate the network operation in the following example.

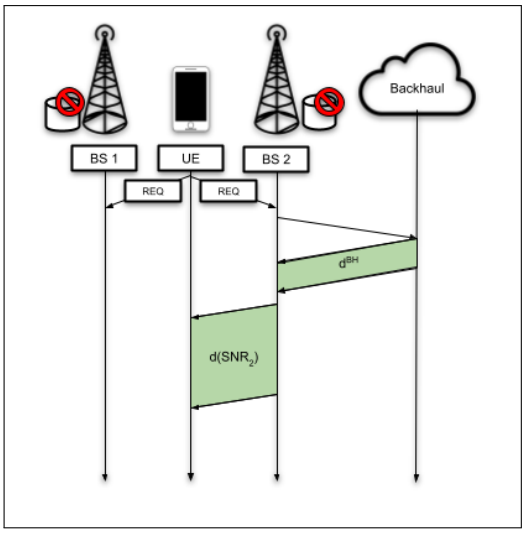
2.4 Operation Example

Consider a small scenario with a UE requesting a file to its 2 neighboring BSs where its SNR with BS 1 is much greater than the SNR of BS 2, i.e., $\text{SNR}_1 \gg \text{SNR}_2$. In this example, we describe in more detail how the network operates and delivers contents to mobile users. In particular, this operation is based on the delay $d(\text{SNR})$ the user experiences to download the requested content through the wireless channel, which is inversely proportional to log of the SNR, i.e., $d(\text{SNR}) \propto \log^{-1}(1+\text{SNR})$. In this example, we denote the delay to retrieve the content from the back-end server through the backhaul network as d^{BH}.

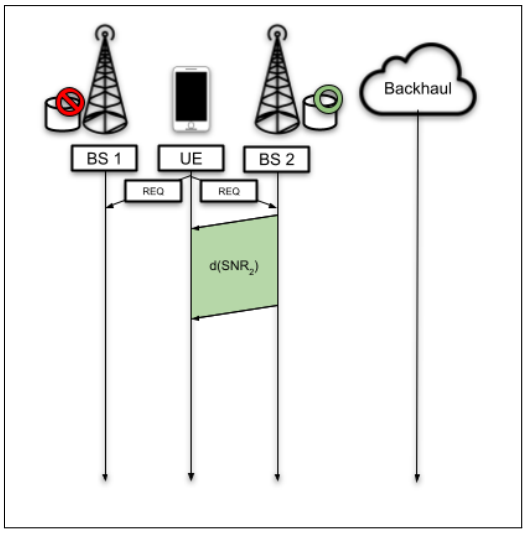
We show four possible transmission cases in Figure 2.3 and we describe them as follows:

1. Cache Miss (Figure 2.3a): If the UE's requested file is not found at the caches of its neighboring BSs, the BS with the highest SNR (BS 2) will retrieve the file from a back-end server and transmit it back to the UE. In this case, the UE will experience a service time equal to the back-end server fetching time plus the time to receive the file through the wireless channel, i.e., $d^{\text{BH}} + d(\text{SNR}_1)$.

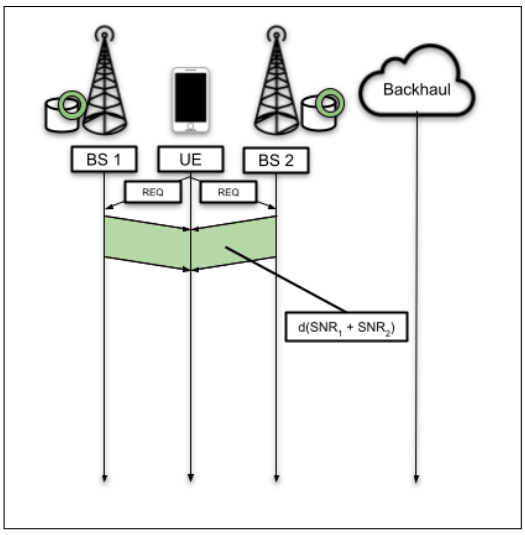
2. Cache "Sufficient" Hit (Figure 2.3b): This is the most standard case where the file is found at one of the neighboring BSs cache, say the BS with the highest SNR, and therefore it is directly transmitted without being fetched from the back-end server. In this case, the UE will experience a service time equal to $d(\text{SNR}_1)$.



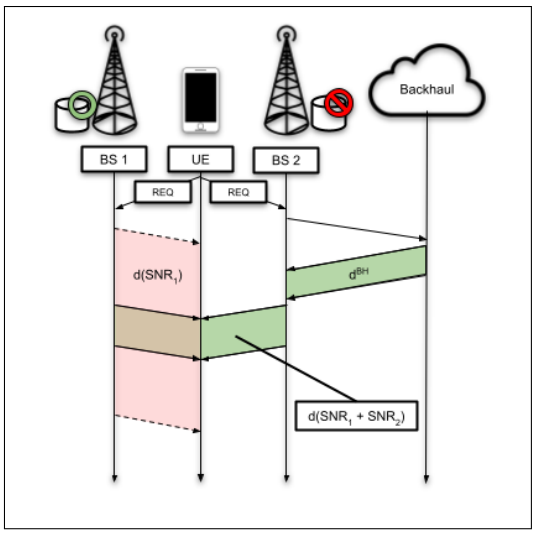
(a) Cache miss



(b) Cache (Single) “Sufficient” Hit



(c) Cache Multiple Hits



(d) Cache “Insufficient” Hit

Figure 2.3 – Example of transmission situations emerging from the CCSC architecture.

3. Multiple ("Sufficient") Hits (Figure 2.3c): Thanks to CoMP JT, if the file is cached at two or more neighboring BSs, it is jointly transmitted with a higher rate so the UE will experience an even smaller delay, that we denote by $d(\mathrm{SNR}_1 + \mathrm{SNR}_2)$.

4. Cache "Insufficient" Hit (Figure 2.3d): This is a very particular case where, even when the file is found at a neighboring cache, it is more effective to retrieve an additional copy from the back-end server and perform CoMP JT. For example, the file is cached at BS 2 that has a very weak SNR with the UE, causing a very long wireless channel transmission time, say $d(\mathrm{SNR}_2)$. Then, BS 1 may opt to download an extra copy of the requested file from the back-end server in order to jointly transmit it along with BS 2. In this case, the UE will experience the time to retrieve the file from the back-end server plus the joint transmission time through the wireless channel, i.e., $d^{\mathrm{BH}} + d(\mathrm{SNR}_1 + \mathrm{SNR}_2)$. Note that this is only the case when $d^{\mathrm{BH}} + d(\mathrm{SNR}_1 + \mathrm{SNR}_2) < d(\mathrm{SNR}_2)$. In general, 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 extra copy to the BS with highest SNR might be amortized by the better overall channel performance. We note that it is possible to have multiple "insufficient" hits, i.e., multiple BSs cache the requested file but it is still worthy to include another BS (for example, with significantly higher SNR).

Chapter 3

Static Caching Solutions

In this chapter, we discuss how to perform the static placement by solving a corresponding optimization problem, given that other system's parameters are known, e.g., files popularities and network topology. First, we introduce the retrieval delay model based on the CoMP-aided cache-enabled small-cell (CCSC) network (presented in Chapter 2). Then, we formalize the general optimization problem and discuss its properties and possible solutions. We show how the general problem may be adapted to maximize hit rate, as in related literature, and to minimize the average delay. Extending the average delay minimization case, we study in detail the particular scenario where SNRs are homogeneous for which we prove that the corresponding objective function is monotone and submodular. As a consequence, a greedy algorithm enjoys a $\frac{1}{2}$-approximation guarantee. We provide some more insight on the problem's solution by evaluating a simple scenario where BSs completely overlap. We confirm that the general delay minimization problem, for heterogeneous SNRs, does not enjoy the same optimality guarantees by providing a counter-example where submodularity condition does not hold. Finally, we discuss the nuances of the delay minimization problem variant where files have heterogeneous sizes.

3.1 System Model and Operation

We consider a general instance of CCSC network as introduced in Section 2.1. Each BS is equipped with a cache that can store up to C files from a catalog $[F] = \{1, \dots, F\}$ of files. We assume that the aggregate request process follows the IRM model: each request is for file f with probability λ_f independently from the past, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_F$ and $\sum_{f\in[F]} \lambda_f = 1$. We refer to the probability λ_f as the popularity of file f. In general, every file $f \in [F]$ has size S_f, in bytes. However, most results of this chapter are for the case where files have the same size, i.e., $S_f = S, \forall f \in [F]$, which is widely considered in the literature (e.g., [23, 51, 66, 68, 69]) as large files are often split into smaller chunks of roughly equal sizes. In Section 3.5, we discuss the general case where files have different sizes and the associated problem's complexity and possible solutions.

We characterize the cache variables using a set notation, such that the ground set is denoted by $\Omega = [B] \times [F]$, where element $(b, f) \in \Omega$ represents the placement of file f in BS b's cache and we represent a *cache allocation* set by $X \subseteq \Omega$. Let $\Omega^{(b)} = \{b\} \times [F]$ be a subset of Ω representing the possible file placements in BS b. An allocation $X \subseteq \Omega$ is *feasible* if it satisfies the caches capacity (cardinality) constraints, i.e.,

$$\left|X \cap \Omega^{(b)}\right| \leq C, \forall b \in [B]. \tag{3.1}$$

For simplicity, we consider that UEs are equally probable to generate a request, i.e., with probability $\frac{1}{U}$. Because of the high density of BSs, each UE u will, in general, be within communication range of multiple BSs. We remind that $I_u = \left\{b \in [B] : V_u^{(b)} > 0\right\}$ is 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 u's neighboring BSs, under allocation X, a subset $J_{u,f}(X) = \{b \in I_u : (b, f) \in X\}$ is actually caching f.

Retrieval Delay Model

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 *wireless channel access delay* is given by

$$t_u(\mathcal{B}) \triangleq \frac{S}{W \cdot \log_2\left(1 + \sum\limits_{b\in\mathcal{B}} V_u^{(b)}\right)}, \tag{3.2}$$

where W is the channel bandwidth and we consider $t_u(\emptyset) = +\infty$.

As discussed in Chapter 2, in order for UE u to pose a new request for file f, it broadcasts an inquiry message for file f that is received by its neighboring BSs in I_u.

Then, according to the current cache state, UE u will experience a delay that is a consequence of one of the following cases:

- If $J_{u,f}(X) = \emptyset$ (Cache Miss), the BS with the highest SNR, i.e., $b^* \triangleq \underset{b \in I_u}{\arg\max} \left\{ V_u^{(b)} \right\}$ downloads f from the back-end server and then transmits it to u. In this case, u experiences a delay of
$$d^{\mathrm{BH}} + t_u(\{b^*\}),$$
which consists of (i) the *backhaul access delay* to retrieve f from the back-end servers, i.e., d^{BH}, plus (ii) the wireless channel access delay to download from b^*, i.e., $t_u(\{b^*\})$.
- If $J_{u,f}(X) \neq \emptyset$ (Cache Hit), then:
  - If it is a "sufficient" hit, all BSs in $J_{u,f}(X)$ can jointly transmit the file so u will experience a retrieval delay of
  $$t_u(J_{u,f}(X)).$$
  - Otherwise, if it is an "insufficient" hit, 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 experienced delay in this case is
  $$d^{\mathrm{BH}} + t_u(J_{u,f}(X) \cup \{b'\}).$$

  The system will opt for the solution with the smallest delay.

Finally, we define the total *experienced delay* by UE u to download file f under allocation X as

$$d_{u,f}(X) \triangleq \min\left(t_u(J_{u,f}(X)),\, d^{\mathrm{BH}} + t_u(J_{u,f}(X) \cup \{b'\})\right), \tag{3.3}$$

where $b' \triangleq \underset{b \in I_u \setminus J_{u,f}(X)}{\arg\max} \left\{ V_u^{(b)} \right\}$. Note that Equation (3.3) also captures the delay when *misses* at all caches occur, i.e., $J_{u,f}(X) = \emptyset$. In this case, $I_u \setminus J_{u,f}(X) = I_u$, so BS $b' = b^*$ will fetch the file from the backhaul and transmit it to u.

We summarize in Table 3.1 the most important notation used throughout this chapter.

Table 3.1 – Notation Summary – Chapter 3

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
S	file size
λ_f	popularity of file f
W	channel bandwidth
d^{BH}	backhaul access delay
$V_u^{(b)}$	SNR of the wireless channel between u and b
V	SNR of all communicating pairs of BS-UE (homogeneous snr regime)
Ω	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
$t_u(\mathcal{B})$	wireless channel access 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
$g_f(X, u)$	gain function
$G_f(X)$	average gain function over all UEs
$G(X)$	average gain over all UEs and files
$\Delta s_{u,f}^{(b)}(X)$	marginal gain for u by caching f at b under X
$\mathbb{1}(e)$	indicator function for event e
$H(X)$	hit ratio under allocation X
$\bar{d}(X)$	average experienced delay under allocation X
$\bar{s}(X)$	average delay saving provided by allocation X
$\Delta s_{u,f}^{(b)}(X)$	marginal delay saving for u by caching f at b under X
S_f	size of file f (heterogeneous file sizes)
R^{BH}	Backhaul transmission rate
M	Backhaul latency
d_f^{BH}	backhaul-access delay for file f (heterogeneous file sizes)

3.2 Problem Definition

Consider a non-negative utility set function $g_f(X, u)$ representing the performance gain (under an arbitrary metric) experienced by UE u for delivering file f under allocation set X. Then, we define the average performance gain over all UEs related to file f and allocation X as

$$G_f(X) \triangleq \frac{1}{U} \sum_{u \in [U]} \lambda_f \cdot g_f(X, u).$$

Our goal is to find a feasible allocation set X, i.e., satisfying the cardinality constraints (3.1), that maximizes the total average performance gain $g_f(\cdot)$ for all $f \in [F]$:

Problem 1 (General Static Optimization – GSO)**:**

$$\begin{aligned} \text{(GSO)} \qquad & \underset{X \subseteq \Omega}{\text{maximize}} && G(X) \triangleq \sum_{f \in [F]} G_f(X) \\ & \text{subject to} && \left| X \cap \Omega^{(b)} \right| \leq C, \, \forall b \in [B]. \end{aligned} \tag{3.4}$$

For the rest of this chapter, we will focus on finding practical solutions using the submodular optimization framework [70]. In this case, it is important to define the discrete derivative $\Delta G((b, f) | \, X)$ of the gain function $G(\cdot)$ as the marginal gain for adding element $(b, f) \in \Omega$ to allocation X, i.e.,

$$\Delta G((b, f) | \, X) \triangleq G(X \cup \{(b, f)\}) - G(X), \tag{3.5}$$

where we stress that the marginal gain of an element already in solution X is null, i.e., if $(b, f) \in X$, then $\Delta G((b, f) | \, X) = 0$.

In the next sections, we present how the GSO problem may be specialized to different performance metrics.

3.3 Hit Rate Maximization

A request for file f by UE u experiences a cache hit if $J_{u,f}(X) \neq \emptyset$, i.e., when UE u request file f, it has a non-empty set of BSs caching the file under allocation X. By using the indicator function that we denote as $\mathbb{1}(\cdot)$, we represent the cache hit as $\mathbb{1}(J_{u,f}(X) \neq \emptyset)$. The gain function in this case is simply the indicator function for a cache hit, i.e.,

$$g_f(X, u) = \mathbb{1}(J_{u,f}(X) \neq \emptyset),$$

and the average gain over all UEs can be expressed as:

$$G_f(X) = H_f(X) \triangleq \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} g_f(X, u)$$
$$= \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} \mathbb{1}(J_{u,f}(X) \neq \emptyset),$$

where $H_f(\cdot)$ is the hit rate for a given file $f \in [F]$.

We define the hit rate maximization problem as follows:

Problem 2 (Hit Rate Maximization Problem)**:**

$$\text{(HRMax)} \qquad \underset{X \subseteq \Omega}{\text{maximize}} \quad G(X) = H(X) \triangleq \sum_{f \in [F]} \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} \mathbb{1}(J_{u,f}(X) \neq \emptyset) \tag{3.6}$$
$$\text{subject to} \quad \left| X \cap \Omega^{(b)} \right| \leq C, \, \forall b \in [B],$$

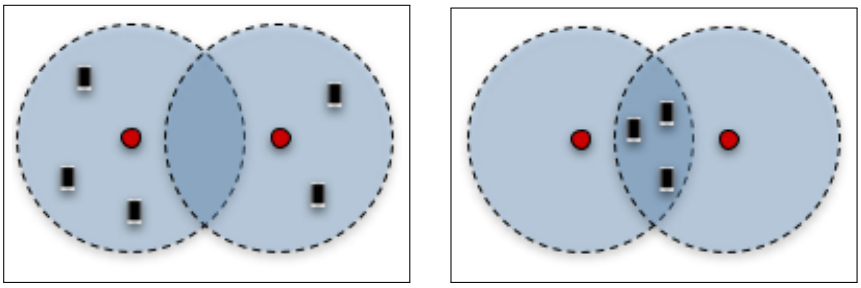
where function $H(\cdot)$ is the hit rate over all UEs and files.

In Problem 2, the solution is determined by (i) files popularities and (ii) network topology. From now on, we focus on the latter; in particular, how the UEs are distributed within the coverage areas. We show in Figure 3.1 three different UE distributions within $B = 2$ BSs and discuss how the optimal cache allocation changes as we vary the network topology.

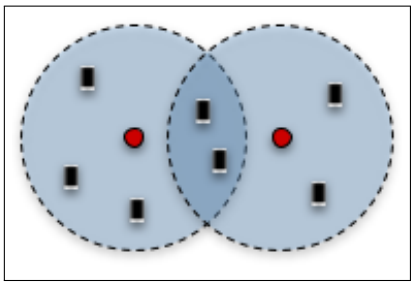
In Figure 3.1a, BSs serve disjoint sets of UEs, so the optimal cache allocation X^*_{FR} is to store the C most popular files at each BS, we call this allocation *full-replication*. In the other extreme, Figure 3.1b depicts a case where all UEs may connect to all BSs. In this case, the optimal caching strategy is to diversify the available files, such that the two BSs together cache the $B \cdot C = 2 \cdot C$ most popular files, that we call *full-diversity* allocation and denote by X^*_{FD}. The interesting cases emerge when BSs partially overlap as in Figure 3.1c. The optimal allocation X^* is non-trivially obtained by solving Problem 2.

As proved by [68, 69], Problem 2 is NP-Hard. Therefore, in order to approximate the optimal solution X^* for a general network topology, we use the greedy algorithm proposed in [68, 69], which we call GreedyHR and describe in Algorithm 1. The same authors also proved that objective (3.6) is a monotone, submodular set function and that constraints (3.1) form a partition matroid. Therefore, GreedyHR enjoys a $\frac{1}{2}$-approximation guarantee, i.e., if X^{GreedyHR} is the solution provided by GreedyHR, then $H\left(X^{\text{GreedyHR}}\right) \geq \frac{1}{2} H\left(X^*\right)$.

Note that Algorithm 1 chooses the best pair (b^*, f^*) at every iteration, i.e., the one with the largest marginal performance gain, in terms on hit rate $H(\cdot)$. As we discussed



(a) No Overlap (b) Full Overlap.



(c) Partial Overlap.

Figure 3.1 – Examples of different coverage area overlap levels for 2 BSs.

in Section 3.1, the marginal performance gain is characterized by the objective's discrete derivative, which, for the hit ratio maximization, is defined as follows

$$\begin{aligned}\Delta H((b,f)|\,X) &\triangleq H(X\cup\{(b,f)\}) - H(X) \qquad (3.7)\\ &= \sum_{f'\in[F]} \lambda_{f'}\cdot\frac{1}{U}\sum_{u\in[U]} \mathbb{1}(J_{u,f'}(X\cup\{(b,f)\})) - \sum_{f'\in[F]} \lambda_{f'}\cdot\frac{1}{U}\sum_{u\in[U]} \mathbb{1}(J_{u,f'}(X))\\ &= \sum_{f'\in[F]} \lambda_{f'}\cdot\frac{1}{U}\sum_{u\in[U]} \big(\mathbb{1}(J_{u,f'}(X\cup\{(b,f)\})) - \mathbb{1}(J_{u,f'}(X))\big)\\ &= \lambda_f\cdot\frac{1}{|\mathcal{U}(b)|}\sum_{u\in\mathcal{U}(b)} \mathbb{1}(J_{u,f}(X)=\emptyset),\end{aligned}$$

where $\mathcal{U}(b)\subseteq[B]$ is the set of UEs that are covered by BS b. We obtain the last line of the above equations by noting that for all files different from f and all users that are not covered by b their contribution to the marginal gain is null. Also, in the hit rate maximization, we only profit from making the first copy of file f available to UE u, i.e., $J_{u,f}(X)=\emptyset$. Otherwise, there is no gain because the cache hit for this request is already guaranteed at some BS.

Algorithm 1: GreedyHR

input : $[U]$, $[F]$, $[B]$, C,
$I_u, \forall u \in [U]$, $J_{u,f}(\cdot), \forall u \in [U], \forall f \in [F]$, $\lambda_f, \forall f \in [F]$, and $H(\cdot)$.
output : Allocation set X.

$X \leftarrow \emptyset$
while $\exists b \in [B] : \left|X \cap \Omega^{(b)}\right| < C$ **do**
  $(b^*, f^*) \leftarrow \underset{(b,f) \in \Omega \setminus X}{\arg\max} \{\Delta H((b,f) | X)\}$
  $X \leftarrow X \cup \{(b^*, f^*)\}$
end
return X

3.4 Average Delay Minimization

In this section we discuss cache solutions for the case where the performance metric is given by the delay experienced by the UEs to have their requests served. In this case, for a given allocation X, we define the average delay for a request over all UEs and files as follows

$$\bar{d}(X) \triangleq \sum_{f \in [F]} \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} d_{u,f}(X), \tag{3.8}$$

where $d_{u,f}(\cdot)$ is the delay defined in Equation (3.3).

We define the general average delay minimization problem as follows:

Problem 3 (Average Delay Minimization Problem)**:**

$$\text{(ADMin)} \qquad \underset{X \subseteq \Omega}{\text{minimize}} \quad \bar{d}(X) = \sum_{f \in [F]} \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} d_{u,f}(X)$$
$$\text{subject to} \quad \left|X \cap \Omega^{(b)}\right| \leq C, \forall b \in [B].$$

Now, we define the *delay saving* experienced by UE u when the requested file f is cached within allocation X as follows

$$s_{u,f}(X) \triangleq d_{u,f}(\emptyset) - d_{u,f}(X), \tag{3.9}$$

where $d_{u,f}(\emptyset) = d^{\text{BH}} + t_u(\{b^*\})$ is delay experienced by UE u as if no files were cached (also defined in Equation (3.3)).[1] Note that $d_{u,f}(\emptyset)$ is the same for all files so we can drop the subscript with respect to files and simply write it as $d_u(\emptyset)$. Then, the average

[1] The value of $d_{u,f}(\emptyset)$ can be replaced with any arbitrarily large constant as long as $s_{u,f}(X)$ is guaranteed to be non-negative for all u, f, and X.

delay saving provided by allocation X is defined as follows

$$\begin{aligned}\bar{s}(X) &\triangleq \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} s_{u,f}(X) \\ &= \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} \left(d_u(\emptyset) - d_{u,f}(X)\right).\end{aligned} \tag{3.10}$$

Now, using the notation introduced in Section 3.1, the performance gain function can be defined as the delay saving (3.9), i.e.,

$$g_f(X,u) = s_{u,f}(X).$$

Similarly, the average gain over all UEs for a given file $f \in [F]$ in the current allocation X is defined as

$$G_f(X) = \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} g_f(X,u) = \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} s_{u,f}(X),$$

such that the expected gain of allocation X is

$$\begin{aligned}G(X) &= \sum_{f\in[F]} G_f(X) \\ &= \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} s_{u,f}(X) = \bar{s}(X).\end{aligned}$$

Finally, consider the following optimization problem:

Problem 4 (Average Delay Saving Maximization Problem)**:**

$$\begin{array}{lll}\text{(DSMax)} & \underset{X\subseteq\Omega}{\text{maximize}} & G(X) = \bar{s}(X) \\ & \text{subject to} & \left|X \cap \Omega^{(b)}\right| \leq C,\ \forall b \in [B],\end{array}$$

Note that the objective function (3.10) of Problem 4 can be expressed as follows

$$\begin{aligned}\bar{s}(X) &= \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} \left(d_u(\emptyset) - d_{u,f}(X)\right) \\ &= \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} d_u(\emptyset) - \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} d_{u,f}(X)\end{aligned}$$

$$= \frac{1}{U} \sum_{u \in [U]} d_u(\emptyset) - \bar{d}(X)$$
$$= \bar{d}_0 - \bar{d}(X),$$

where $\bar{d}_0 = \frac{1}{U} \sum_{u \in [U]} d_u(\emptyset)$ is the average delay for a cache miss over all UEs and it guarantees that $\bar{s}(\cdot)$ is a non-negative function. Therefore, the Problem 4 is equivalent to Problem 3 in the sense that they share the same set of global optimizers.

We discuss the complexity of Problem 3 and its provided maximization counterpart, Problem 4, in detail later. However, it is noteworthy that, even in ideal setups such as in a full overlap topology (that would generate a full-diversity allocation in the hit rate maximization case), its solution is not always straightforward. This is the case because the optimal allocation now depends on (i) the files popularities, (ii) network topology, (iii) the quality of the wireless channel between BSs and UEs (given by the SNRs), and (iv) the backhaul-access latency. Therefore, even in the full overlap topology, the optimal solution of Problem 3 may be achieved through any cache allocation from full-replication to full-diversity.

A natural way to approach Problem 3 (and, consequently, Problem 4) is via a greedy algorithm, comparable to the one proposed for the hit rate maximization problem. In this case, we call it GreedyAD and we present a general description in Algorithm 2. The idea behind GreedyAD, similarly to GreedyHR, is to iteratively add the element that provides the largest marginal performance gain. The most important results are stated for maximization problems, then the marginal performance gain used in GreedyAD is given by the discrete derivative of objective (3.10) that we define as follows

$$\begin{aligned} \Delta\bar{s}((b,f)|\, X) &\triangleq \bar{s}(X \cup \{(b,f)\}) - \bar{s}(X) \\ &= \bar{d}_0 - \bar{d}(X \cup \{(b,f)\}) - (\bar{d}_0 - \bar{d}(X)) \\ &= \bar{d}(X) - \bar{d}(X \cup \{(b,f)\}). \end{aligned} \tag{3.11}$$

Note that the average miss delay $\bar{d}_0$ does not affect the marginal gain and, thus, does not play any role in GreedyAD.

In the next subsections, we aim to gradually build an intuition on the Problem 3's properties and solution characteristics. We start from an ideal scenario, which we call *full-coverage*, and ease the assumptions until we reach the most general case where SNRs are heterogeneous. We also study GreedyAD's efficiency at each of these steps.

Algorithm 2: GREEDYAD

input : $[U]$, $[F]$, $[B]$, C, d^{BH}, $V_u^{(b)}, \forall b \in [B], \forall u \in [U]$
$I_u, \forall u \in [U]$, $J_{u,f}(\cdot), \forall u \in [U], \forall f \in [F]$, $\lambda_f, \forall f \in [F]$, and $\bar{d}(\cdot)$.
output : Allocation set X.

$X \leftarrow \emptyset$
while $\exists b \in [B] : \left|X \cap \Omega^{(b)}\right| < C$ **do**
  $(b^*, f^*) \leftarrow \underset{(b,f) \in \Omega \setminus X}{\arg\max} \left\{\bar{d}(X) - \bar{d}(X \cup \{(b,f)\})\right\}$
  $X \leftarrow X \cup \{(b^*, f^*)\}$
end
return X

3.4.1 Homogeneous SNRs: Full-Coverage

Now, we investigate a very simple instance of the problem, that we call full-coverage scenario. It is useful to obtain more insights on the problem's properties and solutions. Moreover, we will be able to observe how the system's parameters affect the optimal cache allocations.

Assumptions and Specific Notation

The *full-coverage* scenario is based on two assumptions:

1. *Homogeneous SNR regime*, i.e., all non-zero SNRs have the same value V, i.e., $\forall b, u$, if $V_u^{(b)} > 0$, then $V_u^{(b)} = V$.

2. Each UE u can connect to all BSs ($I_u = [B], \forall u \in [U]$), so every UE has access to (the same) aggregate cache capacity of $B \cdot C$ files.

The main idea behind these assumptions is to explore the problem in its simplest form, where we eliminate the combinatorial structure of dense topologies, but retain the advantage of caching multiple copies (exploiting CoMP JT to retrieve files faster). As a matter of fact, under these assumptions, we can consider an equivalent system model with a single UE and a single BS with total cache capacity for $B \cdot C$ files. In this case, up to B copies of every file are allowed to be cached and the delay will be a function of the number of cached copies of any given file.

As a direct consequence, the wireless channel access delay (originally defined in (3.2)) can be simplified as

$$t_u(\mathcal{B}) = \frac{S}{W \cdot \log_2\left(1 + \sum\limits_{b \in \mathcal{B}} V_u^{(b)}\right)} = \frac{S}{W \cdot \log_2\left(1 + \sum\limits_{b \in \mathcal{B}} V\right)} = \frac{S}{W \cdot \log_2\left(1 + |\mathcal{B}| \cdot V\right)},$$

where we remind that $\mathcal{B}$ is a generic set of BSs. Then, we can redefine the wireless channel access delay simply as a function of an integer, representing the number of BSs in set $\mathcal{B}$, as follows

$$t(k) \triangleq \frac{S}{W \cdot \log_2(1 + V \cdot \min(k, B))}, \tag{3.12}$$

where the $\min(\cdot)$ function guarantees that the delay does not decreases below its minimum value allowed by the network topology; which enforces that the UE can not download from more than B BSs. We consider $t(0) = +\infty$.

The delay (originally defined in (3.3)) experienced by the UE to download file f under allocation X is also a function of the number of cached copies of f in its neighboring BSs, $|J_{u,f}(X)|$ (note that the subscript u is redundant in this case, but we kept it to be consistent with the general notation)

$$d(|J_{u,f}(X)|) \triangleq \min\left(t(|J_{u,f}(X)|),\, d^{\mathrm{BH}} + t(|J_{u,f}(X)| + 1)\right). \tag{3.13}$$

Problem Formulation

We define the general average delay minimization problem for the full-coverage scenario as follows:

Problem 5 (Average Delay Minimization Problem for Full-Coverage)**:**

$$\text{(ADMin-FC)} \qquad \underset{X \subseteq \Omega}{\text{minimize}} \qquad \bar{d}(X) \triangleq \sum_{f \in [F]} \lambda_f \cdot d(|J_{u,f}(X)|) \tag{3.14}$$

$$\text{subject to} \qquad \left|X \cap \Omega^{(b)}\right| \leq C,\ \forall b \in [B],$$

where function $\bar{d}(\cdot)$ is the average experienced delay over all files.

We can define the average *delay saving* for the full-coverage scenario with respect to a cache miss if a request is posed under allocation X, i.e.,

$$\bar{s}(X) \triangleq d(0) - \bar{d}(X),$$

where $d(0) = d^{\mathrm{BH}} + t(1)$ is simply the delay upon a cache miss characterized by (3.13).

Following the same reasoning from Section 3.4, the following problem is equivalent to Problem 5:

Problem 5 (Delay Saving Maximization Problem for Full-Coverage)**:**

$$\begin{aligned} \text{(DSMax-FC)} \qquad & \underset{X \subseteq \Omega}{\text{maximize}} && \bar{s}(X) = d(0) - \bar{d}(X) \\ & \text{subject to} && \left| X \cap \Omega^{(b)} \right| \leq C, \, \forall b \in [B]. \end{aligned} \tag{3.15}$$

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

Proposition 1: In the full-coverage scenario, an allocation provided by GreedyAD is optimal.

We present the proof of Proposition 1 in Appendix A.1.

Optimal Solution and Extreme Allocations

For the upcoming results, we define *locally optimal allocations* as follows:

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

First, we note that:

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

We present the proof of Lemma 2 in Appendix A.2.

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 3: In the full-coverage scenario, full-diversity is an optimal allocation if and only if

$$\lambda_1 \cdot (d(1) - d(2)) \leq \lambda_{B \cdot C} \cdot (d(0) - d(1)), \tag{3.16}$$

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

$$\lambda_{C+1} \cdot (d(0) - d(1)) \leq \lambda_C \cdot (d(B-1) - d(B)). \tag{3.17}$$

We present the proof of Proposition 3 in Appendix A.3.

As an application of the results above, Fig. 3.2 shows, for a full-coverage scenario, for which regions of the parameter space (V, d^{BH}) a full-diversity and full-replication allocations are optimal. Files popularities are synthetically generated according to Zipf law with exponent α.

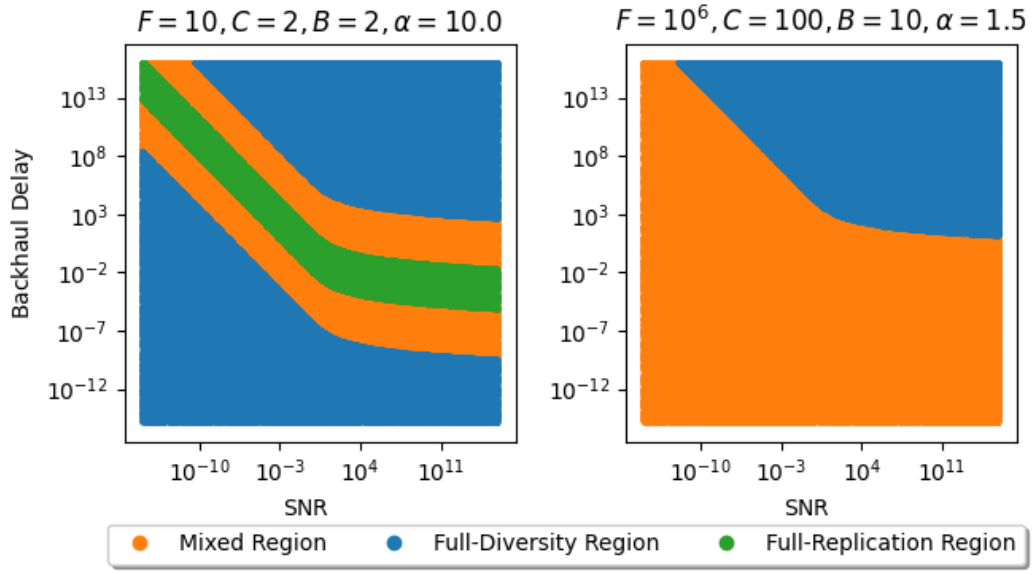


Figure 3.2 – Extreme allocations regions for different setups. Axes are in log scale.

The optimal solution depends on parameters g, d^{BH} in a complex way. Figure 3.2 (left) shows that 5 separated regions are formed even in the simple setup with $B = 2$ BSs. For a given value of g, for high values of d^{BH}, 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^{BH}, 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^{BH}, the optimal allocation moves back to full-diversity (passing again through a mixed region). This happens because, when $d^{\mathrm{BH}} \approx 0$, $d(1) = d^{\mathrm{BH}} + t(2)$ and $d(2) = t(2)$ (because $B = 2$). This makes the LHS of (3.16) approximately zero and smaller than the RHS. A more realistic setup is provided in Fig. 3.2 (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.

However, given an instance of a generic topology, it is not guaranteed that both directions of the conditions in Proposition 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 (3.16) holds. In this case (and for any topology different

from full-coverage), condition (3.16) is necessary but not sufficient for full-diversity to be the optimal allocation. Although Proposition 3 does not hold for general topologies, we can still derive new conditions:

Corollary 4: For general network topologies, assuming homogeneous SNRs, the following conditions hold: (i) Inequality (3.16) is a **necessary condition** for the full-diversity allocation to be locally optimal, and (ii) inequality (3.17) is a **sufficient condition** for the full-replication allocation to be locally optimal.

We present the proof of Corollary 4 in Appendix A.4.

Corollary 4 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.

3.4.2 Homogeneous SNRs: General Topology

Now that we have explored the basic ideas on how the parameters influence the solution of the delay minimization problem, we can move from the simple full-coverage scenario to a general topology. We investigate how it affects the problem complexity and GreedyAD's solution quality.

Assumptions and Specific Notation

In this section, we consider general network topologies, but we retain the homogeneous-SNRs assumption, i.e., all non-zero SNRs have the same value V, i.e., $\forall b \in [B], u \in [U]$, if $V_u^{(b)} > 0$, then $V_u^{(b)} = V$.

Here, we adapt the wireless channel access delay (originally defined in (3.2)) for UE u to retrieve any file from any k BSs as follows

$$t_u(k) \triangleq \frac{S}{W \cdot \log_2(1 + V \cdot \min(k, |I_u|))}. \tag{3.18}$$

The equation is similar to (3.12) defined for the full-coverage scenario, but notice now that it is also a function of the UE, since each UE u may have a different total of available BSs $|I_u|$. Consider $t_u(0) = +\infty, \forall u \in [U]$. Then, we remark that the delay (originally defined in (3.3)) 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, $|J_{u,f}(X)|$, i.e.,

$$d_u(|J_{u,f}(X)|) = \min\left(t_u(|J_{u,f}(X)|),\, d^{\mathrm{BH}} + t_u(|J_{u,f}(X)|+1)\right), \tag{3.19}$$

where, in this case, we also need to specify which UE the experienced delay is related to.

Problem Formulation and Properties

We formalize the delay minimization problem assuming homogeneous SNRs as follows:

Problem 6 (Average Delay Minimization Problem for Homogeneous SNRs)**:**

$$\begin{aligned} \text{(ADMin-HomSNR)} \qquad & \underset{X\subseteq\Omega}{\text{minimize}} && \bar{d}(X) \triangleq \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} d_u(|J_{u,f}(X)|) \\ & \text{subject to} && \left|X \cap \Omega^{(b)}\right| \leq C,\, \forall b \in [B]. \end{aligned} \tag{3.20}$$

where the objective (3.20) is the average experienced delay for a request over all UEs and files and $d_u(|J_{u,f}(X)|)$ is given by (3.19).

Differently from Problem 5, we note that it is hard to find Problem 6's exact solution:

Proposition 5: Problem 6 is NP-Hard in the homogeneous SNR regime.

We present the proof of Proposition 5 in Appendix A.5.

Optimality of GreedyAD

In order to provide approximation guarantees for GreedyAD, we need to express Problem 6 as a maximization problem. Similarly to what was done in the full-coverage scenario, we write Problem 6 as a maximization of the average delay saving as follows:

Problem 6 (Delay Saving Maximization Problem for Homogeneous SNRs)**:**

$$\begin{aligned} \text{(DSMax-HomSNR)} \qquad & \underset{X\subseteq\Omega}{\text{maximize}} && \bar{s}(X) \triangleq d^{(0)} - \bar{d}(X) \\ & \text{subject to} && |X \cap \Omega^{(b)}| \leq C,\, \forall b \in [B], \end{aligned} \tag{3.21}$$

where $d^{(0)} \triangleq d^{\mathrm{BH}} + \frac{S}{W\log_2(1+V)}$ is the delay experienced upon a cache miss (the same for all UEs in the homogeneous SNR scenario.) Function (3.21) is the average delay saving for a request under allocation X related to the cache miss delay $d^{(0)}$.

Now, consider the following lemmas:

Lemma 6: Function (3.21) is *monotone* and *submodular*.

We present the proof of Lemma 6 in Appendix A.6.

Lemma 7: Constraints (3.1) define a partition matroid.

This lemma was originally proved in [69, Lemma 2].

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

Proposition 8: In the homogeneous SNR regime, GreedyAD is a $\frac{1}{2}$-approximation algorithm for the maximization version of Problem 6, i.e.,

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

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

Proof. The maximization version of Problem 6 involves a monotone submodular set function (Lemma 6), subject to a partition matroid constraint (Lemma 7). Therefore, according to [71, Theorem 3.1], the greedy algorithm achieves a $\frac{1}{2}$-approximation ratio. □

3.4.3 Heterogeneous SNRs

We emphasize that GreedyAD can also be used to approximate Problem 3 in the general case, though it does not enjoy the same approximation guarantees as in the homogeneous SNR scenario. The reason is that the objective function may no longer be submodular when SNRs are heterogeneous. We illustrate an example below, which is consistent with the general notation introduced in Section 3.1.

Example 6: Let $X \subset X' \subset \Omega$ and $(b', f') \in \Omega \backslash X'$. 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 $V_u = [30.0, 30.0, 10.0, 100.0]$. We consider $d^{\text{BH}} = 10.0$ms, $S = 1$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'.

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

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

$$d^{\mathrm{BH}} + t_u\left(\{b_1, b_4\}\right) - t_u\left(\{b_1, b_3\}\right) = 1.1 < 1.2 = t_u\left(\{b_1, b_2\}\right) - t_u\left(\{b_1, b_2, b_3\}\right)$$

However, as shown in Section 3.4.2, if $\bar{s}(X)$ is submodular, the following 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.

Although GreedyAD does not enjoy the approximation guarantee (as in the homogeneous SNR scenario), we will see in Chapter 5 that it still provides reasonable results in practice.

3.5 Special Case: Heterogeneous File Sizes

In this section we study the delay minimization problem in the scenario where files have different sizes. We point out the main differences between such problem and its homogeneous-sizes variant. In the end, we discuss a simple greedy algorithm that is able to find a potentially non-feasible solution with approximation guarantees.

Assumptions and Specific Notation

We consider the general case where each file $f \in [F]$ has size S_f in bytes. Note that, in this case, the cache storage capacity C is given in terms of the total amount of data, also in bytes. The cache storage capacity is now represented by the following set of *knapsack* constraints

$$\sum_{(b,f) \in X \cap \Omega^{(b)}} S_f \leq C, \forall b \in [B]. \tag{3.22}$$

Some other elements from our network model must also be adapted to capture the different sizes. First, the backhaul-access delay now depends on each file f

$$d_f^{\mathrm{BH}} \triangleq M + \frac{S_f}{R^{\mathrm{BH}}}, \tag{3.23}$$

where R^{BH} is the backhaul network transmission rate and M is a constant that represents any sort of latency for accessing the back-end servers (e.g., the round-trip time in the backhaul network), henceforth generically referred to as *backhaul latency*.

For simplicity, in this section, we base the network operation on the homogeneous SNR assumption and its corresponding notation adapted from Section 3.4.2 to the

heterogeneous sizes case. Therefore, the wireless channel access delay is a function of the number of transmitting sources k but also depends on UE u and file f:

$$t_{u,f}(k) \triangleq \frac{S_f}{W \cdot \log_2\left(1 + V \cdot \min(k,\, |I_u|)\right)}. \tag{3.24}$$

The total delay experienced by UE u to retrieve file f from the cache allocation X is:

$$d_{u,f}(|J_{u,f}(X)|) \triangleq \min\left(t_{u,f}(|J_{u,f}(X)|),\, d_f^{\mathrm{BH}} + t_{u,f}\left(|J_{u,f}(X)| + 1\right)\right). \tag{3.25}$$

Problem Formulation and Solutions

Now, we consider the maximization of a generic gain function (3.4) subject to multiple knapsack (cache capacity) constraints:

Problem 7 (Heterogeneous-Sizes (HetSize) General Problem)**:**

$$\begin{aligned} \text{(HetSize)} \qquad & \underset{X \subset \Omega}{\text{maximize}} \quad G(X) = \sum_{f \in [F]} G_f(X) = \sum_{f \in [F]} \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} g_f(X, u) \\ & \text{subject to} \quad \sum_{(b,f) \in X \cap \Omega^{(b)}} S_f \leq C,\ \forall b \in [B]. \end{aligned} \tag{3.26}$$

The set of constraints (3.22) guarantees that any feasible solution meets each BS's cache capacity. Problem 7 is NP-Hard because it is a generalization of the single-cache problem with capacity ("knapsack") constraints, which is NP-Hard [72].

The following greedy algorithm may be used to find solutions for Problem 7: Starting from empty caches ($X = \emptyset$), the algorithm iteratively finds the placement (b^*, f^*) that maximizes the ratio between the delay gain and the file size given the current cache allocation X and adds a copy of f^* to b^*. Whenever the placement (b^*, f^*) makes b^*'s occupancy reach or exceed its caching capacity, b^* is considered "full" and disregarded in the upcoming iterations. The algorithm stops when all BSs are "full." We present a formal description in Algorithm 3 and, from now on, we will refer to it as *Infeasible Greedy Algorithm* (IGA), since the resulting allocation is likely to violate constraints (3.22).

Note that, IGA iteratively adds elements to the solution set based on their marginal gain related to the sizes (equivalent to the concept of cost-benefit). We express this relative marginal gain as the discrete derivative divided by the corresponding file size

$$\frac{\Delta G((b,f)|X)}{S_f} = \frac{G(X \cup \{(b,f)\}) - G(X)}{S_f}. \tag{3.27}$$

Problem 7 may be adapted to optimize any performance metric. For example, for

Algorithm 3: Infeasible Greedy Algorithm – IGA

input : $[U]$, $[F]$, $[B]$, $S_f, \forall f \in [F]$, $G(\cdot)$.
output : Solution set X and it is partitioned.

$X \leftarrow \emptyset$
while $\exists b \in [B] : \sum_{(b,f)\in X \cap \Omega^{(b)}} S_f \leq C$ **do**
  $(b^*, f^*) \leftarrow \underset{(b,f)\in\Omega\setminus X}{\arg\max} \left\{ \frac{\Delta G((b,f)|\, X)}{S_f} \right\}$
  $X \leftarrow X \cup \{(b^*, f^*)\}$
end
return X

hit ratio maximization, we would simply need to make $G(X) = H(X)$ as defined in Section 3.3. However, in the rest of this section, we will focus on the delay minimization.

Average Delay Minimization

We now formalize a specialization of Problem 7 to the average delay minimization case.

Problem 8 (Heterogeneous-Sizes (HetSize) ADMin Problem)**:**

$$\text{(ADMin-HS)} \qquad \underset{X\subset\Omega}{\text{minimize}} \quad \bar{d}_{\text{HS}}(X) \triangleq \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} d_{u,f}(|J_{u,f}(X)|) \qquad (3.28)$$

$$\text{subject to} \quad \sum_{(b,f)\in X\cap\Omega^{(b)}} S_f \leq C,\ \forall b \in [B].$$

The objective (3.28) is the average experienced delay for a request over all files and UEs and $d_{u,f}(\cdot)$ is given by (3.25). Note that minimizing $\bar{d}_{\text{HS}}(X)$ is equivalent to maximize the delay saving $\bar{s}_{\text{HS}}(X) \triangleq \bar{d}^{(0)}_{\text{HS}} - \bar{d}_{\text{HS}}(X)$, where $\bar{d}^{(0)}_{\text{HS}}$ is the maximum miss delay over all files and users.

Optimality of IGA

Consider a symmetric network topology where adding a copy of file f to any BS produces the same performance gain, e.g., the full-coverage scenario (see Section 3.4.1). We remark that Problem 8 can be directly mapped to the Submodular Multiple Knapsack Problem (SMKP) [42, 73] if the underlying network topology is symmetric:

Proposition 9: Problem 8 in the full-coverage setup is equivalent to SMKP.

We present the proof of Proposition 9 in Appendix A.7.

The SMKP objective function (3.28) was studied in Section 3.4.2 where we proved that it is monotone and submodular and this result is insensitive to the fact that files have different sizes. According to Proposition 9, instances of Problem 8 with symmetric network topology, where BSs cover equivalent groups of UEs (e.g., full-coverage scenario, see Section 3.4.1), can be directly mapped to a particular instance of SMKP. Authors from [42] proved that IGA solves SMKP with a $(1 - \frac{1}{e})$-approximation ratio.

Although it is a potentially unfeasible solution, the allocation provided by IGA achieves an average delay that is not worse than $(1 - \frac{1}{e})$ of the optimal one. Unfortunately, this approximation guarantee only holds for full-coverage setup. However, although IGA's solution is likely infeasible and the approximation guarantee is only valid for symmetric setups, it can still be used as a heuristic to approximate the minimum achievable average delay in general instances of Problem 8. We only use IGA as a comparison baseline for the dynamic solutions introduced in Chapter 4.

Chapter 4

Dynamic Caching Solutions

In static solutions, there is a centralized entity aware of the files popularities 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., hit ratio and average delay. However, having all this information available is a very strong assumption and is hardly satisfied in real systems. Moreover, static content placement is based on time-average popularity estimations, which may fail to capture short-term popularity variations. Regardless these drawbacks, we emphasize that static approaches are optimal if the underlying request process is stationary, so we will still use them as comparison baselines in our experiments in Chapter 5

In this chapter, we investigate how to overcome the aforementioned weaknesses of the static framework through dynamic solutions for networks of caches based on online policies. Initially, we propose qLRU-Δ online caching policy for asymptotic optimization of different metrics and we prove its optimality under the Independent Reference Model (IRM). Then, we show how qLRU-Δ can be specialized to solve the hit rate maximization and average delay minimization problems. We propose a second policy that promises to provide good results when the request process is not stationary. Finally, we show how we can change qLRU-Δ in order to guarantee the asymptotically optimal behavior even in the case where files have different sizes.

4.1 System Model and Additional Notation

Consider an arbitrary instance of the CoMP-aided cache-enabled small-cell (CCSC) network described in Chapter 2.[1] Because it is more convenient to describe the theoretical results, we use a slightly different notation from Chapter 3. In this chapter, we represent the allocation of a given file $f \in [F]$, all over the cache servers, by a vector of binary variables

$$\mathbf{X}_f(t) = \left(X_f^{(1)}(t), \ldots, X_f^{(B)}(t)\right),$$

where $X_f^{(b)}(t) \in \{0,1\}$ indicates whether BS b caches file f (i.e., $X_f^{(b)}(t) = 1$) or not (i.e., $X_f^{(b)}(t) = 0$). One first important difference is the dependence on time that we represent by indexing the variables with t. We denote by $\mathbf{X}(t) \in \{0,1\}^{B \times F}$ the allocation matrix containing the variables for the entire catalog of files. Any specific assignment for $\mathbf{X}_f(t)$ is generically represented by $\mathbf{x}_f$ and, consequently, a generic assignment for the allocation matrix is represented by $\mathbf{x}$. Whenever it is needed to make a reference to the static framework, which uses a set notation to represent the cache allocations, we may consider, for a given allocation set X, its matrix counterpart by simply stating that $\forall (b,f) \in \Omega$, if $(b,f) \in X$, then $X_f^{(b)}(t) = 1$, otherwise $X_f^{(b)}(t) = 0$, at time instant t.

Recall from Section 1.1 that, in the dynamic framework, caches are structured as ordered queues and are managed according to basic operations. From now on, we consider a restricted set of operations where caches may (i) *insert* new files to cache, (ii) *evict* the file at the rear, and (iii) *move* a given file from its current position to the front of the cache (we often refer to it as *move-to-the-front* (MTF)). At this point, we focus on how to represent insertions and evictions in the current matrix notation. Let $\mathbf{e}^{(b)} \in \{0,1\}^B$ be a vector that has entry b equals to 1 and all other entries equal to 0. In order to help us describe the aforementioned cache operations, we introduce the notation below:

- $\mathbf{x}_f \oplus \mathbf{e}^{(b)}$ represents a new cache allocation where a copy of file f is added at BS b if not already present (i.e., if $x_f^{(b)} = 1$, then $\mathbf{x}_f \oplus e^{(b)} = \mathbf{x}_f$). If $x_f^{(b)} = 0$, $\mathbf{x}_f \oplus \mathbf{e}^{(b)}$ may be used to represent a cache insertion.

- $\mathbf{x}_f \ominus \mathbf{e}^{(b)}$ represents a new cache allocation where there is no copy of file"f at BS b (note that, if $x_f^{(b)} = 0$, then $\mathbf{x}_f \ominus \mathbf{e}^{(b)} = \mathbf{x}_f$). If $x_f^{(b)} = 1$, $\mathbf{x}_f \ominus \mathbf{e}^{(b)}$ may be used to represent a cache eviction.

One or more actors of the system may profit from a performance gain offered by

[1] We have special interest in investigating caching solutions for CCSC such that our proposed techniques and most of our experiments are based on this architecture. However, the results of this chapter may be extended to any system consisting of network of caches.

cache deployment, which is generically denoted by the utility function $g_f(\mathbf{X}_f(t), u)$. Function $g_f(\mathbf{X}_f(t), u)$ captures the gain for delivering file f to UE u under f's allocation $\mathbf{X}_f(t)$. We assume that $g_f(\mathbf{0}, u) = 0$, i.e., if there is no cached copy of file f, the gain is zero. Later in Section 4.2.3 we discuss application examples where the utility function is the hit ratio or the average delay saving. We further define the *marginal gain* for delivering file f to UE u under a specific allocation $\mathbf{x}_f$ related to the copy of file f stored at BS b as

$$\Delta g_f^{(b)}(\mathbf{x}_f, u) \triangleq g_f(\mathbf{x}_f, u) - g_f\left(\mathbf{x}_f \ominus \mathbf{e}^{(b)}, u\right). \tag{4.1}$$

The marginal gain $\Delta g_f^{(b)}(\mathbf{x}_f, u)$ quantifies the performance improvement experienced by UE u when the system moves from allocation $\mathbf{x}_f \ominus \mathbf{e}^{(b)}$ to allocation $\mathbf{x}_f$.

The gain $g_f(\mathbf{X}_f(t), u)$ may be a random variable. For example, it may depend on the instantaneous characteristics of the wireless channels in the CCSC network, or on some user's random choice like which BS the file will be downloaded from. We assume that, conditionally on the network status $\mathbf{X}_f(t)$ and the UE u, these random variables are independent from one request to the other and are identically distributed with expected value $\mathbb{E}[g_f(\mathbf{X}_f(t), u)]$.

The last important change of notation with respect to Chapter 3 is that we represent the set of UE u's neighboring BSs that are currently caching file f simply as $J_{u,f}$. We disregard the dependence on the current allocation matrix $\mathbf{X}$ because, as we will see later, this whole information is not needed by our caching policies. Besides, the list of BSs $J_{u,f}$ may be directly provided by UE u, after inquiring its neighboring BSs.

Besides the basic notation introduced in Chapter 3, we summarize the most important notation for this chapter in Table 4.1.

4.2 Optimal Caching for Stationary Requests

We assume the request process is stationary. In particular, requests for file f are issued from UE u according to a Poisson arrival process with rate $\lambda_{f,u}$ that is independent of other files and UEs. We define the total expected gain per time unit of a given placement $\mathbf{x}_f$ as:

$$G_f(\mathbf{x}_f) = \sum_{u \in [U]} \lambda_{f,u} \cdot \mathbb{E}\left[g_f(\mathbf{x}_f, u)\right]. \tag{4.2}$$

Note that $G_f(\cdot)$ is non-negative and non-decreasing, such that, for each $\mathbf{x}_f$ and each b, we have that $G_f\left(\mathbf{x}_f \oplus \mathbf{e}^{(b)}\right) \geq G_f(\mathbf{x}_f)$.

Table 4.1 – Notation Summary – Chapter 4

Symbol	Description
$\mathbf{X}_f(t)$	vector of allocation variables for file f at time t
$\mathbf{x}_f$	instance of f's allocation; particular assignment for $\mathbf{X}_f(t)$
$g_f(\mathbf{X}_f, u)$	gain function
$\Delta g_f^{(b)}(\mathbf{x}_f, u)$	marginal gain
$G_f(\mathbf{x}_f)$	expected gain
$\Delta G_f^{(b)}(\mathbf{x}_f, u)$	marginal expected gain
I_u	set of UE u's neighboring BSs
$J_{u,f}$	set of u's neighboring BSs caching f
$\lambda_{u,f}$	rate at which file f if requested by UE u
S	file size
C	cache capacity
$p_f^{(b)}(u)$	move-to-the-front probability
β	move-to-the-front probability's normalization factor
$q_f^{(b)}(u)$	insertion probability
δ	insertion probability's normalization factor
$q^{(b)}$	insertion parameter for BS b
$T_c^{(b)}$	characteristic time at BS b
$\pi_f(\mathbf{x}_f)$	probability of file f to be cached in configuration $\mathbf{x}_f$
$T_{S,f}^{(b)}$	sojourn time of file f at BS b
$\mathbb{1}(e)$	indicator function for event e
$d_{u,f}(X)$	experienced delay by u to get f under allocation X
$\sigma_f^{(b)}(u)$	insertion probability for 2LRU-Δ
S_f	size of file f (heterogeneous file sizes)
R^{BH}	Backhaul transmission rate
M	Backhaul latency
d_f^{BH}	backhaul-access delay for file f (heterogeneous file sizes)

Finally, we define the marginal expected gain for caching a copy of file f at BS b experienced by UE u under allocation $\mathbf{x}_f$ as:

$$\Delta G_f^{(b)}(\mathbf{x}_f) \triangleq G_f(\mathbf{x}_f) - G_f\left(\mathbf{x}_f \ominus \mathbf{e}^{(b)}\right). \tag{4.3}$$

The marginal expected gain $\Delta G_f^{(b)}(\mathbf{x}_f)$ can be interpreted as the expected gain when the system moves from state $\mathbf{x}_f \ominus \mathbf{e}^{(b)}$ to state $\mathbf{x}_f$.

As discussed in Chapter 3, in the static framework, the allocation is defined once and for all and remains unchanged over time. Then, we drop the time dependence of variables $\mathbf{X}(t)$ in order to present the generic optimization problem:

Problem 1 (General Static Problem – Matrix Notation)**:**

$$\text{(GSO)} \qquad \underset{\mathbf{X}}{\text{maximize}} \qquad G(\mathbf{X}) \triangleq \sum_{f\in[F]} G_f(\mathbf{X}_f) \tag{4.4}$$

$$\text{subject to} \qquad \sum_{f\in[F]} X_f^{(b)} = C \quad \forall b \in [F], \tag{4.5}$$

$$X_f^{(b)} \in \{0,1\} \quad \forall f \in [F], \forall b \in [B].$$

We stress here that this formulation is fairly generic, as $G_f(\mathbf{X}_f)$ in Equation (4.4) can capture any objective desired by the system designer, as long as it is non-negative and non-decreasing in $\mathbf{X}_f$. Moreover, in the case where files have uniform sizes, i.e., $S_f = S, \forall f \in [F]$, because $g_f(\cdot)$ cannot decrease if a new copy of f is placed at any BS, the optimal allocation is achieved when all caches are completely full, i.e., (4.5) are equality constraints. Problem 1 is NP-hard, even in the case of the simple hit ratio metric [7]; hence, efficient algorithmic solutions are only able to approximate the problem, as discussed in Chapter 3.

In what follows, we introduce a new online caching policy (i.e., caches self-manage their contents reacting to incoming users requests). We will show that, if each cache individually deploys an instance of this policy, the entire network of caches asymptotically converges to an allocation that optimizes Problem 1. This is the case even in the absence of a priori knowledge about the request process and overall network structure.

qLRU-Δ Online Caching Policy

The qLRU-Δ online caching policy was built on top of the basic operations of plain qLRU [53], i.e., upon each request, (i) new files are inserted with probability q causing the least-recently-used one at the rear to be evicted and (ii) files found at the cache

are moved to the front. The main difference is that both operations are now performed probabilistically and the corresponding probabilities depend on their expected performance improvement, which is captured by the marginal gain, given in Equation (4.1). Upon a request from UE u for file f, qLRU-Δ functions as follows:

- Each BS b neighboring UE u with a local copy of file f (i.e., $b \in J_{u,f}$) moves the content to the front of the cache with probability proportional to the marginal gain due to its local copy, i.e.,

$$p_f^{(b)}(u) = \beta \cdot \Delta g_f^{(b)}(\mathbf{X}_f(t), u), \tag{4.6}$$

  where the constant β guarantees that $p_f^{(b)}(u) \in [0, 1]$, e.g.,

$$\beta = \left(\max_{f', u', b', \mathbf{x}_{f'}} \left\{ \Delta g_{f'}^{(b')}(\mathbf{x}_{f'}, u') \right\} \right)^{-1}. \tag{4.7}$$

- At least one of the neighboring BSs without the content (i.e., those in $I_{u,f} \setminus J_{u,f}$) will store an additional copy of f with probability

$$q_f^{(b)}(u) = q^{(b)} \cdot \delta \cdot \max \left(\Delta g_f^{(b)}(\mathbf{X}_f(t) \oplus \mathbf{e}^{(b)}, u), \epsilon \right), \tag{4.8}$$

  where $q^{(b)} \in (0, 1]$ is a dimensionless parameter, potentially different for every BS b, constant $\epsilon > 0$ guarantees that $q_f^{(b)}(u)$ is always positive, and constant δ plays the same role of β, e.g.,

$$\delta = \left(\max_{f', u', b', \mathbf{x}_{f'}} \left\{ \Delta g_f^{(b)}(\mathbf{x}_{f'} \oplus \mathbf{e}^{(b)}, u) \right\} \right)^{-1}. \tag{4.9}$$

We formalize qLRU-Δ in Algorithm 4 from the perspective of a given BS b, where we denote the least-recently-used file at the rear of the cache as f_{rear}.

Remark 1: The probability $q_f^{(b)}(u)$ in (4.8) is analogous to q in the plain qLRU policy. We note that setting $q_f^{(b)}(u) = q$, i.e., a constant value, suffices to prove the asymptotic convergence of qLRU-Δ (see Proposition 10). However, we use the more elaborate function (4.8), as this still guarantees convergence, but is able to react faster to transient dynamics, be preferentially treating a larger marginal gain $\Delta g_f^{(b)}(\mathbf{X}_f(t) \oplus \mathbf{e}^{(b)}, u)$. This can speed up the transient dynamics of the policy as will become evident in Chapter 5.

Algorithm 4: General implementation of qLRU-Δ Caching Policy (for BS b)

Input: $J_{u,f}$, $g_f(\cdot,\cdot)$.
if $b \in J_{u,f}$ **then**
  with probability $p_f^{(b)}(u)$ in (4.6) **do**
    Move-to-the-front f
  end
else
  with probability $q_f^{(b)}(u)$ in (4.8) **do**
    Evict file f_{rear} from the rear
    Insert f to the front
  end
end

Remark 2: Some information about the local neighborhood (e.g., how many additional copies of the content are stored at close-by caches also serving that user) may be needed to compute the marginal gains in (4.6) and (4.8). Such information, however, is limited, and can be piggybacked on existing messages the UE sends to query such caches, or even on channel estimates messages mobile devices regularly send to nearby BSs. In Section 4.2.3 we detail what information needs to be exchanged when the system aims to maximize the hit rate or to minimize the average delay.

We are going to prove that qLRU-Δ achieves a locally optimal configuration when the values $q^{(b)}$ tend to 0. The result relies on two approximations: (i) the characteristic time approximation (CTA) for caching policies (also known as Che's approximation) [46,47] and (ii) the exponentialization approximation (EA) for networks of interacting caches originally proposed in [56].

Proposition 10: [high-level statement] Under CTA and EA, a network of qLRU-Δ caches asymptotically achieves an optimal caching configuration when $\forall b \in [B], q^{(b)} \to 0$.

Before moving to the detailed proof, we provide some intuition about why this result holds. We observe that, as $q^{(b)}$ converges to 0, cache b exhibits two different dynamics with very different timescales: (i) the *insertion of new files* tends to happen progressively more rarely (because $q_f^{(b)}(u)$ converges to 0) while (ii) the frequency of *MTFs* for files already in the cache is unchanged ($p_f^{(b)}(u)$ does not depend on $q^{(b)}$). A file f at cache b is moved to the front with a probability proportional to $\Delta g_f^{(b)}(\mathbf{x}_f, u)$, i.e., proportional to how much the file contributes to improve the performance metric of interest. This is a very noisy signal: upon a given request, the file is moved to the front or not. At the same time, as $q^{(b)}$ converges to 0, more MTFs occur between any two file evictions.

The expected number of MTFs file f experiences is proportional to 1) how often it is requested ($\lambda_{f,u}$) and 2) how likely it is to be moved to the front upon a request ($p_f^{(b)}(u)$). Overall, the expected number of MTFs is proportional to $\sum_u \lambda_{f,u} \cdot \mathbb{E}\left[\Delta g_f^{(b)}(\mathbf{x}_f, u)\right]$, i.e., its contribution to the expected gain. By the law of large numbers, the random number of MTFs 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 expected gain. qLRU-Δ then behaves as a greedy algorithm that, driven by the request process, replaces the least useful file in the cache at each insertion, until the network converges to a cache allocation that provides maximum expected performance gain.

4.2.1 Modeling qLRU-Δ as a Markov Chain

In a dynamic cache system operating under qLRU-Δ, the general cache allocation changes over time as new requests arrive. If we consider $\mathbf{X}_f(t)$ as a random variable indicating the allocation of file f throughout the caches at time t, we can define the stochastic process

$$(\mathbf{X}_f(t), t \geq 0) \tag{4.10}$$

that characterizes the evolution of the caches' contents starting from time $t = 0$.

Even for very simple definitions of function $g_f(\cdot, \cdot)$, extracting useful insights from the analysis of $(\mathbf{X}_f(t), t \geq 0)$ may be an arduous and unfruitful task. In order to derive useful results, we propose a simpler representation of such process that is based on two approximations, CTA and EA, that we discuss next. Although they may appear as strong assumptions at first, as we will discuss in Chapter 5, they have little impact in practice, such that theoretical predictions will match the empirical results.

Characteristic Time Approximation

This is a standard approximation for a cache in isolation, and one of the most effective approximate approaches for analysis of cache systems. At the moment, we focus on a single cache (i.e., one BS in isolation), or equivalently on a cache b in a network of B non-overlapping cells. CTA was first introduced in [47] and revisited in [46]. It was originally proposed for LRU under the IRM request process, and it has been later extended to different caching policies and different request processes [53, 57]. Here we simply write $\Delta g_f^{(b)}(u)$ instead of $\Delta g_f^{(b)}(\mathbf{X}_f, u)$, because we are considering a single cache. Similarly, we write $\Delta G_f^{(b)}$, instead of $\Delta G_f^{(b)}(\mathbf{X}_f(t))$.

The *characteristic time* $T_c^{(b)}$ is the time a given file spends in cache b since its insertion

until its eviction in the absence of any subsequent request for it. In general, this quantity depends in a complex way on the dynamics of requests for other files. Instead, CTA assumes that $T_c^{(b)}$ is a random variable independent from other files dynamics and with an assigned probability distribution (the same for every file). This assumption makes it possible to decouple the dynamics of the different files: upon a miss for file f, the file is inserted and a timer with random value $T_c^{(b)}$ is generated. When the timer expires, the content is evicted from the cache.

Caching policies differ in (i) the distribution of $T_c^{(b)}$ and (ii) what happens to the timer upon a hit. For example, $T_c^{(b)}$ is a constant under LRU, qLRU, 2LRU, and FIFO and exponentially distributed under RANDOM. Upon a hit, the timer is renewed under LRU, qLRU, and 2LRU, but not under FIFO and RANDOM. In what follows we will only consider policies for which $T_c^{(b)}$ is a constant. Under CTA, the instantaneous cache occupancy can violate the hard buffer constraint, i.e., it is acceptable to have more files in the cache than its real capacity. The value of $T_c^{(b)}$ is obtained by imposing the expected occupancy to be equal to the buffer size

$$\sum_{f\in[F]} \pi_f^{(b)} = C, \tag{4.11}$$

where $\pi_f^{(b)}$ denotes the probability that content f is in cache b. Its expression as a function of $T_c^{(b)}$ depends on the specific caching policy [53]. Despite its simplicity, CTA was shown to provide asymptotically exact predictions for a single LRU cache under IRM as the cache size grows large [47, 74, 75].

Once inserted in the cache, a given content f will sojourn in the cache for a random amount of time $T_{S,f}^{(b)}$, independently from the dynamics of other contents. $T_{S,f}^{(b)}$ can be characterized for the different policies. In particular, if the timer is renewed upon a hit, as is the case for LRU-like policies, we have

$$\begin{aligned} T_{S,f}^{(b)} &\triangleq \sum_{k=1}^{\infty} Y_k \cdot \mathbb{1}\left(Y_1 < T_c^{(b)}, \ldots, Y_k < T_c^{(b)}\right) + T_c^{(b)} \\ &= \sum_{k=1}^{N} Y_k + T_c^{(b)}, \end{aligned} \tag{4.12}$$

where $N \in \{0, 1, \ldots\}$ is the number of consecutive hits following a miss, and Y_k is the time interval between the k-th request following a miss and the previous content request.

For a finite number of UEs, requests for content f from UE u arrive according to a Poisson process with rate $\lambda_{f,u}$. The time instants at which content f is moved to the front

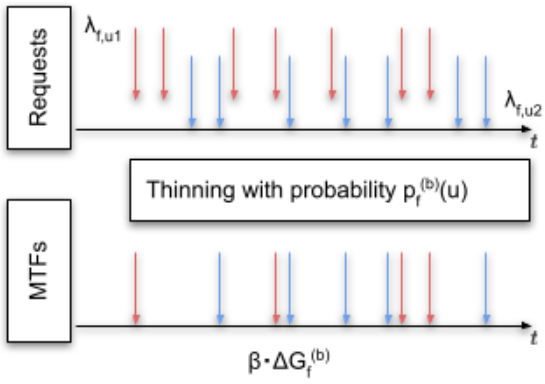


Figure 4.1 – Poisson arrival process with rate $\beta \cdot \Delta G_f^{(b)}$ representing the MTF events for file f at BS b. It is obtained from thinning the original Poisson process of arriving requests from the different UEs for file f with rate $\lambda_{f,u}$ with probability $p_f^{(b)}(u)$.

are generated by thinning this Poisson process with probability $p_f^{(b)}(u) = \beta \cdot \mathbb{E}[\Delta g_f^{(b)}(u)]$. The resulting sequence is then also a Poisson process with rate $\lambda_{f,u} \cdot \beta \cdot \mathbb{E}[\Delta g_f^{(b)}(u)]$. Finally, as we illustrate in Figure 4.1, as request processes from different UEs are independent, the aggregate cache updates with MTFs due to all UEs is a Poisson process with rate

$$\beta \sum_{u=1}^{U} \lambda_{f,u} \cdot \mathbb{E}\left[\Delta g_f^{(b)}(u)\right] = \beta \cdot \Delta G_f^{(b)}.$$

As the aggregate cache updates follow a Poisson process with rate $\beta \cdot \Delta G_f^{(b)}$, $\{Y_k\}$ are i.i.d. truncated exponential random variables with rate $\beta \cdot \Delta G_f^{(b)}$ over the interval $[0, T_c^{(b)}]$ and their expected values are given by

$$\mathbb{E}[Y_k] = \frac{1}{\beta \Delta G_f^{(b)}} - \frac{T_c^{(b)}}{e^{\beta \Delta G_f^{(b)} T_c^{(b)}} - 1}.$$

Moreover, the probability of no updates to occur during a time interval of length $T_c^{(b)}$ is $e^{-\beta \Delta G_f^{(b)} T_c^{(b)}}$. Then N is distributed as a geometric random variable defined over non-negative integer values with expected value

$$\mathbb{E}[N] = \frac{1 - e^{-\beta \Delta G_f^{(b)} T_c^{(b)}}}{e^{-\beta \Delta G_f^{(b)} T_c^{(b)}}} = e^{\beta \Delta G_f^{(b)} T_c^{(b)}} - 1.$$

We want to compute the expected value of the sojourn time $T_{S,f}^{(b)}$. Since N is clearly a stopping point for the sequence $\{Y_k\}_k$, we can then apply Wald's Lemma to

Equation (4.12), obtaining:

$$\nu_f^{(b)} \triangleq \frac{1}{\mathbb{E}[T_{S,f}^{(b)}]} = \frac{1}{\mathbb{E}[Y_1]\,\mathbb{E}[N] + T_c^{(b)}} = \frac{\beta \Delta G_f^{(b)}}{e^{\beta \Delta G_f^{(b)} T_c^{(b)}} - 1}. \tag{4.13}$$

The quantity $\nu_f^{(b)}$ may be interpreted as file f's *eviction* rate at BS b, i.e., the rate at which file f leaves the cache after its insertion at BS b.

Exponentialization Approximation

Now, we try to generalize the previous discussion to the case where up to B BSs may overlap. Notice that the sojourn time of file f inserted at time t in cache b will now depend on the whole state vector $\mathbf{X}_f(\tau)$ for $\tau \geq t$, i.e., until it is evicted, file f's allocation throughout the caches jointly dictates its permanence at any given cache. This is the case because the file's position is updated with probability (4.6) depending on the marginal gain of the copy (and then on $\mathbf{X}_f(\tau)$).

Exponentialization Approximation (EA) consists in assuming that the stochastic process $(\mathbf{X}_f(t), t \geq 0)$ for file f can be simplified as a Continuous-Time Markov Chain (CTMC). The set of states $\mathcal{S}_f$ of this CTMC is defined over all possible assignments of $\mathbf{X}_f(t)$, i.e.,

$$\mathcal{S}_f = \left\{ \mathbf{x}_f : \forall \mathbf{x}_f \in \{0,1\}^B \right\}. \tag{4.14}$$

Any transition from state $\mathbf{x}_f$ to $\mathbf{y}_f$ has a rate generically represented by $\rho[\mathbf{x}_f \to \mathbf{y}_f]$.

EA replaces then the original stochastic process with a set of CTMCs $\mathbf{X}_f(t)$, $\forall f \in [F]$, which are only coupled through the characteristic times $T_c^{(b)}$ at the BSs. Similarly to what was indicated for a single cache, we can determine the values $T_c^{(b)}$ at each cache, by imposing that:

$$\sum_{f \in [F]} \sum_{\mathbf{x}_f \in \{0,1\}^B} x_f^{(b)} \cdot \pi_f(\mathbf{x}_f) = C, \forall b \in [B], \tag{4.15}$$

where $\pi_f(\mathbf{x}_f)$ denotes the stationary probability of CTMC $\mathbf{X}_f(t)$ to be at state $\mathbf{x}_f$.

In the paper where EA was originally proposed [56], the authors showed that this CTMC representation and the related assumptions have no impact on any system metric that depends only on the stationary distribution in the following cases:

1. isolated caches,
2. caches using RANDOM policy,
3. caches using FIFO policy as far as the resulting CTMC $\mathbf{X}_f(t)$ is reversible.

Numerical results in [56] show that the approximation is practically very accurate also in

more general cases. In Chapter 5, we provide experiments suggesting that the theoretical results obtained from modeling qLRU-Δ dynamics under EA are also observed in practice, even for the case of delay minimization in CCSC networks.

In Figure 4.2, we show an example of a CTMC $\mathbf{X}_f(t)$ for a given file f in a scenario with $B = 2$ BSs. The set of states is defined as $\mathcal{S}_f = \{(0,0),(0,1),(1,0),(1,1)\}$ or simply as $\mathcal{S}_f = \{(0),(1),(2),(3)\}$. In what follows, we will discuss how to characterize the transitions and their corresponding rates. In particular, we will see how the asymptotic values of $q^{(b)}$ affect the transition rates.

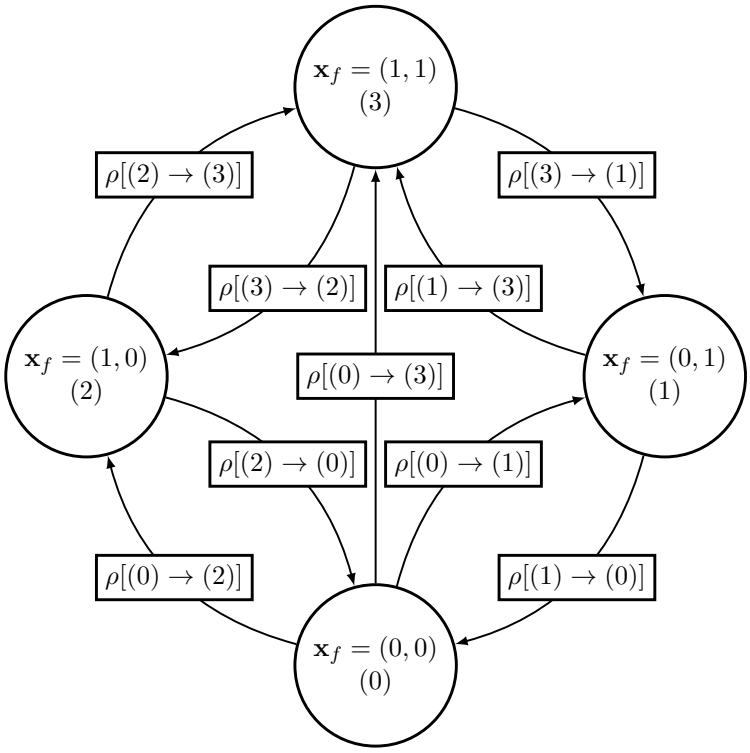


Figure 4.2 – CTMC $\mathbf{X}_f(t)$ for $B = 2$ BSs.

For a given content f, let $\mathbf{x}_f$ and $\mathbf{y}_f$ be two possible states of the CTMC $\mathbf{X}_f(t)$. We write $\mathbf{x}_f < \mathbf{y}_f$ whenever $x_f^{(b)} \leq y_f^{(b)}$ for each b and there is at least one b_0 such that $x_f^{(b_0)} < y_f^{(b_0)}$, and we say that $\mathbf{y}_f$ is an *ancestor* of $\mathbf{x}_f$, and $\mathbf{x}_f$ is a *descendant* of $\mathbf{y}_f$. For example, in Figure 4.2, state (1) is an ancestor of state (0) (which is its descendant) and (3) is an ancestor of all other states (which are all its descendants). Moreover, we define the *weight* $|\mathbf{x}_f|$ of state $\mathbf{x}_f$ as the number of copies of file f stored in state $\mathbf{x}_f$, i.e., $|\mathbf{x}_f| \triangleq \sum_b x_f^{(b)}$. If $\mathbf{x}_f < \mathbf{y}_f$ and $|\mathbf{x}_f| = |\mathbf{y}_f| - 1$, we say that $\mathbf{y}_f$ is a *parent* of $\mathbf{x}_f$ and $\mathbf{x}_f$ is a *child* of $\mathbf{y}_f$. Again, in Figure 4.2, we see that $(1),(0)$ characterizes a pair parent-child, which is not case for $(3),(0)$, for example.

Now, observe that, by construction, transition rates in the CTMC are different from 0 only between pair of states $\mathbf{x}_f$ and $\mathbf{y}_f$, such that $\mathbf{x}_f < \mathbf{y}_f$ or $\mathbf{y}_f < \mathbf{x}_f$. For example, states (1) and (2) do not have direct transitions. This is the case because there is zero probability of a file being inserted at one BS exactly at the same time as it is evicted from the other BS (when its sojourn time is over). For any pair of states $\mathbf{x}_f$ and $\mathbf{y}_f$, such that $\mathbf{x}_f < \mathbf{y}_f$, the transition $\mathbf{x}_f \to \mathbf{y}_f$ is called an *upward* transition, while $\mathbf{y}_f \to \mathbf{x}_f$ is called a *downward* transition.

A downward transition can only occur from a parent to a child ($|\mathbf{x}_f| = |\mathbf{y}_f| - 1$), because it is not possible that file f is evicted from both BSs exactly at the same time. Consider a child-parent pair $\mathbf{x}_f, \mathbf{y}_f$ and let b_0 be the index such that $x_f^{(b_0)} < y_f^{(b_0)}$. We have that the downward rate is, in fact, file f's eviction rate at BS b_0, given that the CTMC is currently at state $\mathbf{y}_f$, i.e.,

$$\rho_{[\mathbf{y}_f \to \mathbf{x}_f]} = \nu_f^{(b_0)}(\mathbf{y}_f) = \frac{\beta \cdot \Delta G_f^{(b_0)}(\mathbf{y}_f)}{e^{\beta \cdot \Delta G_f^{(b_0)}(\mathbf{y}_f) \cdot T_c^{(b_0)}} - 1}. \tag{4.16}$$

Upward transitions can occur to states that are ancestors. The exact transition rate between state $\mathbf{x}_f$ and state $\mathbf{y}_f$ with $\mathbf{x}_f < \mathbf{y}_f$ can have a quite complex expression, because it depends on the joint decisions of the BSs in $I_{u,f} \setminus J_{u,f}$. Upon a request for f, a transition $\mathbf{x}_f \to \mathbf{y}_f$ occurs, if $|\mathbf{y}_f| - |\mathbf{x}_f|$ BSs independently store, each with probability proportional to its parameter $q^{(b)}$, an additional copy of the content f in their local cache. It follows that

$$\rho_{[\mathbf{x}_f \to \mathbf{y}_f]} \propto \prod_{b \,|\, y_f^{(b)} - x_f^{(b)} = 1} q^{(b)}, \tag{4.17}$$

where we use the symbol $\propto$ to indicate that two quantities are asymptotically proportional for small $q \in \mathbb{R}$. In other words, given two functions $f(q)$ and $g(q)$, we write $f(q) \propto g(q)$ if and only if there exists a strictly positive constant a such that

$$\lim_{q \to 0} \frac{f(q)}{g(q)} = a.$$

If $a = 1$, then we write $f(q) \sim g(q)$, following Bachmann-Landau notation, which means that "$f(q)$ is in the same order of $g(q)$" or that they are asymptotically equal.

For our analysis, we are only interested in how the upward rates depend on $q^{(b)}$ when it converges to 0.

Asymptotic transition rates

Specifically, as parameters $q^{(b)}, \forall b \in [B]$ converges to 0, for every $f \in [F]$, every upward rate $\rho_{[\mathbf{x}_f \to \mathbf{y}_f]}$ tends to 0. Therefore, the characteristic time $T_C^{(b)}$ must diverge for all BSs. If it were not the case at a given BS b, none of the files would be found in this cache asymptotically, because upward rates would tend to zero, while downward rates, given by Equation (4.16), would not. This would contradict the set of constraints (4.15) imposed by the CTA. Therefore, in order to satisfy (4.15), for every BS $b \in [B]$, $T_C^{(b)}$ necessarily diverges. More precisely, we must have that

$$T_C^{(b)}\left(q^{(b)}\right) = \Theta\left(\log \frac{1}{q^{(b)}}\right), \forall b \in [B],$$

also in the Bachmann-Landau notation. In other words, there exist positive constants $a_l^{(b)}$ and $a_u^{(b)}$, such that

$$\lim_{q^{(b)} \to 0} \frac{T_C^{(b)}\left(q^{(b)}\right)}{\log \frac{1}{q^{(b)}}} \in \left[a_l^{(b)}, a_u^{(b)}\right].$$

Given that the behavior $T_C^{(b)}\left(q^{(b)}\right) / \log(1/q^{(b)})$ is expected to be "smooth," we assume that there exist positive constants $\gamma_b, \forall b \in [B]$, such that $\frac{1}{\beta \cdot \gamma_b} \in [a_l^{(b)}, a_u^{(b)}]$ and

$$T_C^{(b)}\left(q^{(b)}\right) \sim \frac{1}{\beta \cdot \gamma_b} \cdot \log\left(\frac{1}{q^{(b)}}\right).$$

The following lemma summarises the discussion of this section.

Lemma 11: Consider a pair of states $\mathbf{x}_f$ and $\mathbf{y}_f$ with $\mathbf{x}_f < \mathbf{y}_f$ and a set of positive constants $\{\gamma_b, \forall b \in [B]\}$, such that $T_c^{(b)}(q^{(b)}) \sim \frac{1}{\beta \cdot \gamma_b} \cdot \log\left(\frac{1}{q^{(b)}}\right)$. If $q^{(b)} = q^{\gamma_b}$, then upward transitions have rate

$$\rho_{[\mathbf{x}_f \to \mathbf{y}_f]} \propto q^{\boldsymbol{\gamma}^\intercal(\mathbf{y}_f - \mathbf{x}_f)},$$

and, if $\mathbf{x}_f$ and $\mathbf{y}_f$ form a pair of child-parent states, i.e., $\mathbf{x}_f = \mathbf{y}_f \ominus \mathbf{e}^{(b_0)}$, then downward transitions have rate

$$\rho_{[\mathbf{y}_f \to \mathbf{x}_f]} \propto q^{\Delta G_f^{(b_0)}(\mathbf{y}_f)}.$$

From now on, we will assume that there is a single insertion parameter q such that $q^{(b)} = q^{\gamma_b}$. For each possible transition, we define its *direct resistance* to be the exponent of the parameter q, then $r_f(\mathbf{x}_f, \mathbf{y}_f) = \boldsymbol{\gamma}^\intercal(\mathbf{y} - \mathbf{x})$, $r_f(\mathbf{y}_f, \mathbf{x}_f) = \Delta G_f^{(b_0)}(\mathbf{y}_f)$ and $r_f(\mathbf{x}_f, \mathbf{x}_f) = 0$. Observe that the higher the resistance, the less likely the corresponding transition.

Stochastically stable states

Consider the Discrete-Time Markov Chain (DTMC) $\hat{\mathbf{X}}_f(k)$, obtained sampling the CTMC $\mathbf{X}_f(t)$ with a period $\tau > 0$, i.e., $\hat{\mathbf{X}}_f(k) = \mathbf{X}_f(k \cdot \tau)$. Both $\hat{\mathbf{X}}_f(k)$ and $\mathbf{X}_f(t)$ have the same set of states $\mathcal{S}_f$ and let $\mathbf{P}_{f,q}$ denote the transition probability matrix of $\hat{\mathbf{X}}_f(k)$ for a specific value of parameter q. For $q = 0$, the set of files in the cache does not change, each state is an absorbing one and any probability distribution is stationary for $\mathbf{P}_{f,0}$.

For $q > 0$, the set of possible transitions of $\hat{\mathbf{X}}_f(k)$ is, in general, larger than the set of possible transitions of $\mathbf{X}_f(t)$, as multiple transitions of $\mathbf{X}_f(t)$ can occur during the period τ. For example, $\mathbf{X}_f(t)$ cannot move directly from $\mathbf{x}_f$ to $\mathbf{x}''_f = \mathbf{x}_f \ominus \mathbf{e}^{(b_1)} \ominus \mathbf{e}^{(b_2)}$, with $|\mathbf{x}''_f| = |\mathbf{x}_f| - 2$, but during the interval τ it could move from $\mathbf{x}_f$ to $\mathbf{x}'_f = \mathbf{x}_f \ominus \mathbf{e}^{(b_1)}$ and then from $\mathbf{x}'_f$ to $\mathbf{x}''_f$. The transition $\mathbf{x}_f \to \mathbf{x}''_f$ is then possible for $\hat{\mathbf{X}}_f(k)$.

Furthermore, for any $q > 0$, the DTMC $\hat{\mathbf{X}}_f(k)$ is finite, irreducible,[2] and aperiodic and, therefore, admits a unique stationary distribution

$$\boldsymbol{\pi}_{f,q} = (\pi_{f,q}(\mathbf{x}_f), \forall \mathbf{x}_f \in \mathcal{S}_f)\,.$$

From now on, we will focus on the asymptotic behaviour of the DTMC $\hat{\mathbf{X}}_f(k)$ when q converges to 0. For small values of τ and of q, the probability of a direct transition $\mathbf{x}_f \to \mathbf{x}'_f$ is proportional to

$$q^{r(\mathbf{x}_f,\mathbf{x}'_f)}\tau + o\left(q^{r(\mathbf{x}_f,\mathbf{x}'_f)}\right) + o(\tau).$$

On the other hand, the probability of a sequence of transitions $\mathbf{x}_f \to \mathbf{x}'_f \to \mathbf{x}''_f$ to happen within interval τ is smaller than

$$q^{r(\mathbf{x}_f,\mathbf{x}'_f)+r(\mathbf{x}'_f,\mathbf{x}''_f)}\tau^2 + o\left(q^{r(\mathbf{x}_f,\mathbf{x}'_f)}\right) + o\left(q^{r(\mathbf{x}'_f,\mathbf{x}''_f)}\right) + o(\tau).$$

These transitions may be neglected as their probabilities are $o(\tau)$ and their equivalent resistances are equal to the sum of the direct transitions they are composed by. Therefore, we can restrict ourselves to consider the same set of transitions as in $\mathbf{X}_f(t)$.[3] Each DTMC $\hat{\mathbf{X}}_f(k)$ has then transition rates proportional to a power of q, i.e.

$$P_{f,q}(\mathbf{x}_f, \mathbf{x}'_f) \propto q^{r_f(\mathbf{x}_f,\mathbf{x}'_f)},$$

where we omit, from now on, the proportionality to τ.

[2]This is guaranteed if insertion probabilities in (4.8) are positive. In some specific settings, it may be $\Delta g_f^{(b)}(\mathbf{X}_f(t) \oplus \mathbf{e}^{(b)}, u) = 0$ for each u. We can then consider $q_f^{(b)}(u) = q\gamma \max(\Delta g_f^{(b)}(\mathbf{X}_f(t) \oplus \mathbf{e}^{(b)}, u), \epsilon)$ with $\epsilon > 0$, or simply $q_f^{(b)}(u) = q$.

[3]We omit self-loops in the resulting DTMC as they do not influence the outcome of our analysis.

We end this discussion by introducing the concept of *stochastically stable* (SS) states that will be important later to characterize the solution of Problem 1.

Definition 2: A state $\mathbf{x}_f \in \mathcal{S}_f$ is called *stochastically stable* if

$$\lim_{q\to 0} \pi_{f,q}(\mathbf{x}_f) > 0.$$

These DTMCs were studied in a series of papers [76–78] by P. R. Kumar and his coauthors, because of their relation with the MCs that appear in simulated annealing problems, where $r_f(\mathbf{x}_f, \mathbf{x}'_f) = \max(C(\mathbf{x}'_f) - C(\mathbf{x}_f), 0)$ and $C(\mathbf{x}_f)$ is a cost function we want to minimize. In what follows, we will list as lemmas three results from these papers that will be useful for the optimality proof.

The Modified Balance Equations and Potential Function

Consider a weighted graph $\mathcal{G}_f$, whose nodes are the possible states $\mathbf{x}_f \in \{0,1\}^B$ and edges indicate possible direct transitions and have a weight equal to the corresponding resistance. Given an in-tree[4] $\mathcal{T}(\mathbf{x}_f)$ on $\mathcal{G}_f$ rooted at $\mathbf{x}_f$, we denote by $r_f(\mathcal{T}(\mathbf{x}_f))$ the resistance of the in-tree, i.e., the sum of all resistances of the edges of $\mathcal{T}(\mathbf{x}_f)$. We also denote by $\mathfrak{T}(\mathbf{x}_f)$ the set of all in-trees rooted at state $\mathbf{x}_f$. We show in Figure 4.3 an example of resistance graph $\mathcal{G}_f$ built over the DTMC $\hat{\mathbf{X}}_f(k)$ for $B = 2$ BSs and, in Figure 4.4, we show an example of an in-tree of $\mathcal{G}_f$ rooted at state $\mathbf{x}_f = (1, 0)$.

Finally, we denote by $r_f(\mathbf{x}_f)$ the resistance of the minimum weight in-tree in $\mathcal{G}_f$ rooted to $\mathbf{x}_f$, i.e.,

$$r_f(\mathbf{x}_f) \triangleq \min_{\mathcal{T}\in\mathfrak{T}(\mathbf{x}_f)} r_f(\mathcal{T}).$$

Intuitively, the resistance of a state is a measure of the general difficulty to reach state $\mathbf{x}_f$ from all other nodes. A consequence of the Markov Chain Tree Theorem (see for example [80]) is that

Lemma 12: [78, Lemma 1] The stationary probabilities of the DTMC $\hat{\mathbf{X}}_{f,q}(k)$ have the following expression

$$\pi_{f,q}(\mathbf{x}_f) \propto q^{r_f(\mathbf{x}_f) - \min_{\mathbf{x}'_f} r_f(\mathbf{x}'_f)}.$$

A consequence of Lemma 12 is that the stochastically stable states are those with minimal resistance.

[4] *In-trees*, also known as *anti-arborescences*, are directed rooted trees with the edges pointed towards the root node [79].

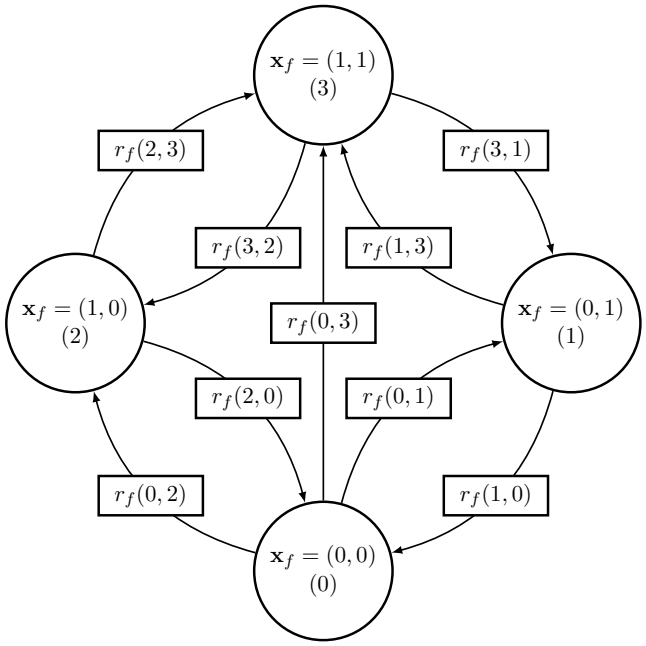


Figure 4.3 – Resistance graph $\mathcal{G}_f$ of DTMC $\hat{\mathbf{X}}_f(k)$ for $B = 2$ BSs.

Consider the following system of *modified balance equations* in the variables $\nu_f(\mathbf{x})$:

$$\begin{cases} \max\limits_{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c} \nu_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{z}_f) = \max\limits_{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c} \nu_f(\mathbf{z}_f) - r_f(\mathbf{z}_f, \mathbf{x}_f), \forall A \subset \{0,1\}^B \\ \max\limits_{\mathbf{x}_f \in \{0,1\}^B} \nu_f(\mathbf{x}_f) = \sigma. \end{cases} \tag{4.18}$$

Lemma 13: [77, Theorem 3] For each σ, the system (4.18) admits a unique solution. Solutions for different values of σ are translates of each other.

System (4.18) implicitly determines the set of stochastically stable states:

Lemma 14: [78, Theorem 4] Given $\{\nu_f(\mathbf{x}_f)\}$ the solution of system (4.18), it holds:

$$r_f(\mathbf{x}_f) - \min_{\mathbf{x}'_f} r_f(\mathbf{x}'_f) = \sigma - \nu_f(\mathbf{x}_f).$$

In particular for our system, we can prove that

Lemma 15: The *potential* function

$$\phi_f(\mathbf{x}_f) \triangleq G_f(\mathbf{x}_f) - \boldsymbol{\gamma}^\intercal \mathbf{x}_f$$

is a solution of system (4.18) (for a particular value of σ).

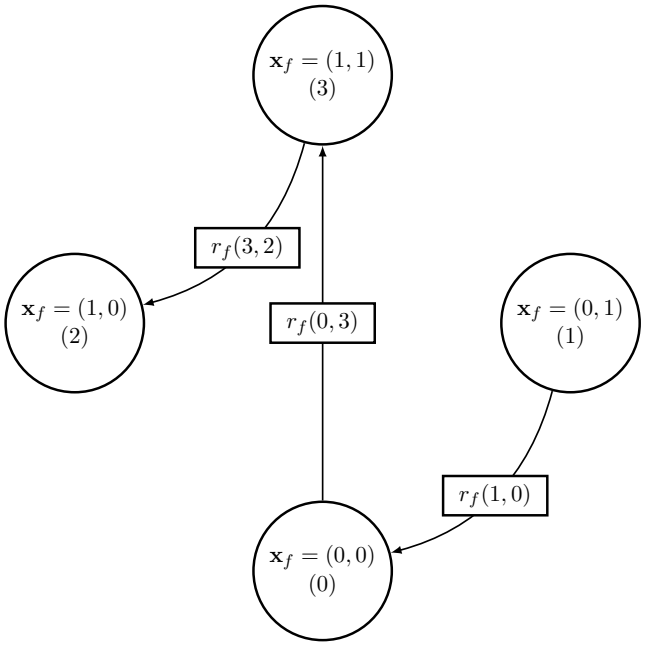


Figure 4.4 – Example of in-tree over $\mathcal{G}_f$ rooted at state 2 for $B = 2$ BSs.

Proof. Consider the function:

$$\phi_f(\mathbf{x}_f) \triangleq G_f(\mathbf{x}_f) - \boldsymbol{\gamma}^\intercal \mathbf{x}_f. \tag{4.19}$$

We show that $\{\phi_f(\mathbf{x}_f)\}$ is a solution of the system (4.18) (for a particular value of σ).

To this purpose, for a given choice of the set A, we need to evaluate

$$\max_{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c} \phi_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{z}_f).$$

We start proving that the maximum is always achieved by a pair of parent-child nodes. In particular, we show that for any two states $\hat{\mathbf{x}}_f \in A$ and $\hat{\mathbf{z}}_f \in A^c$ with $r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f) < \infty$ and $|\hat{\mathbf{z}}_f| > |\hat{\mathbf{x}}_f| + 1$, (which imply that $\hat{\mathbf{z}}_f$ is an ancestor of $\hat{\mathbf{x}}_f$), there exist two states $\mathbf{x}'_f \in A$ and $\mathbf{y}'_f \in A^c$, with $\mathbf{y}'_f$ parent of $\mathbf{x}'_f$ and

$$\phi_f(\hat{\mathbf{x}}_f) - r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f) \leq \phi_f(\mathbf{x}'_f) - r_f(\mathbf{x}'_f, \mathbf{y}'_f). \tag{4.20}$$

Consider a path from $\hat{\mathbf{z}}_f$ to $\hat{\mathbf{x}}_f$ that traverses states with strictly smaller weight (it is obtained setting progressively to zero the elements that are equal to one in $\hat{\mathbf{z}}_f$, but not in $\hat{\mathbf{x}}_f$). One of the edges of this path necessarily goes from a state in A^c to a state in A.

These two states are respectively $\mathbf{y}'_f$ and $\mathbf{x}'_f$. In fact,

$$\begin{aligned}
\phi_f(\hat{\mathbf{x}}_f) - r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f) &= G_f(\hat{\mathbf{x}}_f) - \boldsymbol{\gamma}^\intercal \hat{\mathbf{x}}_f - \boldsymbol{\gamma}^\intercal(\hat{\mathbf{z}}_f - \hat{\mathbf{x}}_f) \\
&= G_f(\hat{\mathbf{x}}_f) - \boldsymbol{\gamma}^\intercal \hat{\mathbf{z}}_f \\
&\leq G_f(\mathbf{x}'_f) - \boldsymbol{\gamma}^\intercal \hat{\mathbf{z}}_f \\
&= G_f(\mathbf{x}'_f) - \boldsymbol{\gamma}^\intercal \mathbf{x}'_f - \boldsymbol{\gamma}^\intercal(\mathbf{y}'_f - \mathbf{x}'_f) - \boldsymbol{\gamma}^\intercal(\hat{\mathbf{z}}_f - \mathbf{y}'_f) \\
&= \phi_f(\mathbf{x}'_f) - r_f(\mathbf{x}'_f, \mathbf{y}'_f) - \boldsymbol{\gamma}^\intercal(\hat{\mathbf{z}}_f - \mathbf{y}'_f) \\
&\leq \phi_f(\mathbf{x}'_f) - r_f(\mathbf{x}'_f, \mathbf{y}'_f),
\end{aligned}$$

where the first inequality follows from the monotonicity of $G_f(\cdot)$, and the second from the fact that $\mathbf{z}_f$ is an ancestor of $\mathbf{y}'_f$ and then $\hat{\mathbf{z}}_f - \mathbf{y}'_f$ is a vector with non-negative elements.

In addition note that by construction $r(\hat{\mathbf{z}}_f, \hat{\mathbf{x}}_f) = \infty$ (i.e. given two states $\mathbf{z}_f$ and $\mathbf{x}_f$ with $|\mathbf{z}_f| > |\mathbf{x}_f|$, we have $r(\mathbf{z}_f, \mathbf{x}_f) < \infty$ only if $\mathbf{x}_f$ is a child of $\mathbf{z}_f$).

As a consequence we have that $\{\phi(\mathbf{x}_f)\}$ is a solution of system (4.18), if and only if it is a solution of

$$\begin{cases}
\max\limits_{\substack{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c, \\ |\mathbf{z}_f| = |\mathbf{x}_f| \pm 1}} \nu_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{z}_f) \\
\qquad = \max\limits_{\substack{\mathbf{x}_f \in A, \mathbf{z}_f \in A^c, \\ |\mathbf{z}_f| = |\mathbf{x}_f| \pm 1}} \nu_f(\mathbf{z}_f) - r_f(\mathbf{z}_f, \mathbf{x}_f), \quad \forall A \subset \{0,1\}^B \\
\max\limits_{\mathbf{x}_f \in \{0,1\}^B} \nu_f(\mathbf{x}_f) = \sigma.
\end{cases} \tag{4.21}$$

We can then limit ourselves to check if $\phi_f(\cdot)$ satisfies the aggregate balance equations considering only the parent-child pairs. We prove a stronger relation, i.e., that for every parent-child pair, $\phi_f(\cdot)$ satisfies a pairwise balance equation. In fact, for every $\mathbf{x}_f$ and $\mathbf{y}_f$ with $\mathbf{y}_f = \mathbf{x}_f \oplus \mathbf{e}^{(b_0)}$ and parent of $\mathbf{x}_f$

$$\begin{aligned}
\phi_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{y}_f) &= G_f(\mathbf{x}_f) - \boldsymbol{\gamma}^\intercal \mathbf{x}_f - \boldsymbol{\gamma}^\intercal(\mathbf{y}_f - \mathbf{x}_f) \\
&= G_f(\mathbf{x}_f) - \boldsymbol{\gamma}^\intercal \mathbf{y}_f \\
&= G_f(\mathbf{y}_f) - \Delta G_f^{(b_0)}(\mathbf{y}_f) - \boldsymbol{\gamma}^\intercal \mathbf{y}_f \\
&= \phi_f(\mathbf{y}_f) - r_f(\mathbf{y}_f, \mathbf{x}_f).
\end{aligned}$$

It follows that $\{\phi(\mathbf{x}_f)\}$ is a solution of system (4.18). □

A consequence of Lemmas 12 – 15 is that

Corollary 16: The set of SS states is the set of global maximizers of $\phi_f(\mathbf{x}_f)$.

For each content f we are then able to characterize which configurations are stochastically stable as q converges to 0.

4.2.2 Optimality of qLRU-Δ

We define the exponential-size linear expansion of Problem 1 as follows:

Problem 9 (GSO Linear Expansion)**:**

$$\underset{a_f(\mathbf{x}_f),\forall f,\forall \mathbf{x}_f}{\text{maximize}} \quad \sum_{f\in[F]} \sum_{\mathbf{x}_f\in\{0,1\}^B} G_f(\mathbf{x}_f)\cdot a_f(\mathbf{x}_f) \tag{4.22}$$

$$\text{subject to} \quad \sum_{f\in[F]} \sum_{\mathbf{x}_f\in\{0,1\}^B} a_f(\mathbf{x}_f)\cdot x_f^{(b)} = C, \quad \forall b\in[B] \tag{4.23}$$

$$\sum_{\mathbf{x}_f\in\{0,1\}^B} a_f(\mathbf{x}_f) = 1, \qquad \forall f\in[F] \tag{4.24}$$

$$a_f(\mathbf{x}_f)\in\{0,1\}, \qquad \forall f\in[F],\forall \mathbf{x}_f\in\{0,1\}^B, \tag{4.25}$$

where the original set of variables $X_f^{(b)}, \forall b\in[B], f\in[F]$ is replaced with a new set of indicator variables. For every file $f\in[F]$ and each of its possible assignments $\mathbf{x}_f$ throughout the network of caches, the new variable $a_f(\mathbf{x}_f)\in\{0,1\}$ indicates whether assignment $\mathbf{x}_f$ is considered in the final solution (i.e., $a_f(\mathbf{x}_f)=1$), or not (i.e., $a_f(\mathbf{x}_f)=0$), which is defined in (4.25). We adapt the cache capacity constraints in (4.23) and guarantee that only one allocation assignment for each file is considered in the final solution by imposing a new constraint set (4.24). Therefore, maximizing objective (4.22) subject to constraints (4.23)–(4.25) is equivalent to solve Problem 1.

Then, we consider the continuous relaxation of Problem 9:

Problem 10 (GSO Continuous Relaxation)**:**

$$\underset{\alpha_f(\mathbf{x}_f),\forall f,\forall \mathbf{x}_f}{\text{maximize}} \quad \sum_{f\in[F]} \sum_{\mathbf{x}_f\in\{0,1\}^B} G_f(\mathbf{x}_f)\cdot \alpha_f(\mathbf{x}_f) \tag{4.26}$$

$$\text{subject to} \quad \sum_{f\in[F]} \sum_{\mathbf{x}_f\in\{0,1\}^B} \alpha_f(\mathbf{x}_f)\cdot x_f^{(b)} = C, \quad \forall b\in[B] \tag{4.27}$$

$$\sum_{\mathbf{x}_f\in\{0,1\}^B} \alpha_f(\mathbf{x}_f) = 1, \qquad \forall f\in[F] \tag{4.28}$$

$$\alpha_f(\mathbf{x}_f)\geq 0, \qquad \forall f\in[F],\forall \mathbf{x}_f\in\{0,1\}^B, \tag{4.29}$$

where we introduce the set of continuous variables $\{\alpha_f(\mathbf{x}_f)\in\mathbb{R} : \forall f\in[F],\forall \mathbf{x}_f\in\{0,1\}^B\}$ to replace the original set of integer variables $\{a_f(\mathbf{x}_f)\}$. We keep only constraints (4.27)

and (4.28) from the previous formulation and include constraints (4.29). Note that the combination of (4.28) and (4.29) enforces that feasible solutions are characterized for $\alpha_f(\mathbf{x}_f) \in [0,1], \forall f \in [F], \forall \mathbf{x}_f \in \{0,1\}^B$.

The original optimization problem (Problem 1 and its linear expansion, Problem 9) corresponds to the particular case, where we require that, for each $f \in [F]$, there exists a single state $\mathbf{x}_f$ with $\alpha_f(\mathbf{x}_f) = 1$ and $\alpha_f(\mathbf{x}'_f) = 0$ for each $\mathbf{x}'_f \neq \mathbf{x}_f$. As the feasible set of the relaxed Problem 10 includes the feasible set of Problem 1, the optimum value of Problem 10 is at least as large as the optimal value of Problem 1.

Note how the capacity constraint in Problem 10 is similar to the relaxed constraint considered by the CTA (see (4.15)). This suggests that the stationary probabilities $\pi_f(\mathbf{x}_f)$ will play the role of the coefficients $\alpha_f(\mathbf{x}_f)$.

Now we can state formally our result.

Proposition 10: Under CTA and EA, let $\{\gamma_b, b \in [B]\}$ be the constants in Lemma 11. Consider the spatial network of qLRU-Δ caches, where cache b sets parameter $q^{(b)} = q^{\gamma_b}$. As q converges to 0, the stationary probabilities

$$\left\{\pi_{f,q}(\mathbf{x}_f), f \in [F], \mathbf{x}_f \in \{0,1\}^B\right\}$$

converge to an optimal solution of Problem (4.26).

Proof. From Corollary 16 a state $\mathbf{x}_f$ is stochastically stable if and only if it is a global maximizer of $\phi_f(\cdot)$, i.e., $\lim_{q\to 0} \pi_{f,q}(\mathbf{x}_f) > 0$ if and only if $\mathbf{x}_f$ is a maximizer of $\phi_f(\cdot)$.

Let $\pi_{f,0^+}(\mathbf{x}_f) \triangleq \lim_{q\to 0} \pi_{f,q}(\mathbf{x}_f)$ denote the limit of the probability distribution. We are now going to prove that the $\{\pi_{f,0^+}(\mathbf{x}_f), f \in [F], \mathbf{x}_f \in 0,1^B\}$ is an optimal solution for Problem 10.

Problem (4.26) is a convex problem. We can consider its Lagrangian function

$$\begin{aligned} L(\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{\zeta}) = &- \sum_{f\in[F]} \sum_{\mathbf{x}_f \in \{0,1\}^B} \alpha_f(\mathbf{x}_f) G_f(\mathbf{x}_f) \\ &+ \sum_{b=1}^{B} \chi_b \left(\sum_{f\in[F]} \sum_{\mathbf{x}_f \in \{0,1\}^B} \alpha_f(\mathbf{x}_f) x_f^{(b)} - C \right) \\ &+ \sum_{f\in[F]} \zeta_f \left(\sum_{\mathbf{x}_f \in \{0,1\}^B} \alpha_f(\mathbf{x}_f) - 1 \right), \end{aligned} \tag{4.30}$$

where $\boldsymbol{\alpha}$ denotes the $F2^B$ vector of problem variables, $\boldsymbol{\chi}$ denotes the B vector of Lagrange multipliers relative to the capacity constraints, and $\boldsymbol{\zeta}$ denotes the F vector of Lagrange

multipliers relative to the total mass to assign to each file.

According to [81, Thm. 3.4.1], vector $\boldsymbol{\alpha}^*$ is a (global) maximizer of Problem 10, if there exists vectors $\boldsymbol{\chi}^*$ and $\boldsymbol{\zeta}^*$ such that

$$\begin{aligned} &1)\ \ \boldsymbol{\alpha}^* \text{ is feasible,} \\ &2)\ \ \nabla L(\boldsymbol{\alpha}^*, \boldsymbol{\chi}^*, \boldsymbol{\zeta}^*)^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) \geq 0, \ \ \forall \boldsymbol{\alpha} \geq \mathbf{0}. \end{aligned}$$

We show that the following assignments satisfy the set of conditions indicated above

$$\begin{cases} \alpha_f^*(\mathbf{x}_f) = \pi_{f,0^+}(\mathbf{x}_f), & \forall f \in [F], \mathbf{x}_f \in \{0,1\}^B, \\ \chi_b^* = \gamma_b, & \forall b \in [B], \\ \zeta_f^* = \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f), & \forall f \in [F]. \end{cases}$$

In fact, for any value q, $\sum_f \sum_{\mathbf{x}_f} x_f^{(b)} \pi_{f,q}(\mathbf{x}_f) = C$ for each b, $\sum_{\mathbf{x}_f} \pi_{f,q}(\mathbf{x}_f) = 1$ for each f, and $\pi_{f,q}(\mathbf{x}_f) \geq 0$ for each f and $\mathbf{x}_f$. The same relations are also satisfied passing to the limit when q converges to 0, then $\{\pi_{f,0^+}(\mathbf{x}_f)\}$ is a feasible solution. Finally,

$$\begin{aligned} \left.\frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{\zeta})}{\partial \alpha_f(\mathbf{x}_f)}\right|_{\substack{\boldsymbol{\alpha}=\boldsymbol{\alpha}^* \boldsymbol{\chi}=\boldsymbol{\chi}^* \\ \boldsymbol{\zeta}=\boldsymbol{\zeta}^*}} &= -G_f(\mathbf{x}_f) + \sum_{b=1}^{B} \gamma_b x_f^{(b)} + \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f) \\ &= -\phi(\mathbf{x}_f) + \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f) \\ &\begin{cases} = 0 & \text{if } \mathbf{x}_f \text{ is stochastically stable,} \\ > 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Let $\mathcal{S}_f \subset \{0,1\}^B$ denote the set of stochastically stable states for file f. It follows that

$$\begin{aligned} \nabla L(\boldsymbol{\alpha}^*, \boldsymbol{\chi}^*, \boldsymbol{\zeta}^*)^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) &= \sum_{f \in [F]} \sum_{\mathbf{x}_f \in \{0,1\}^B} \left.\frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{\zeta})}{\partial \alpha_f(\mathbf{x}_f)}\right|_{\substack{\boldsymbol{\alpha}=\boldsymbol{\alpha}^* \\ \boldsymbol{\chi}=\boldsymbol{\chi}^* \\ \boldsymbol{\zeta}=\boldsymbol{\zeta}^*}} \times (\alpha_f(\mathbf{x}_f) - \alpha_f^*(\mathbf{x}_f)) \\ &= \sum_{f \in [F]} \sum_{\mathbf{x}_f \in \mathcal{S}_f} 0 \times (\alpha_f(\mathbf{x}_f) - \pi_{f,0^+}(\mathbf{x}_f)) \\ &\quad + \sum_{f \in [F]} \sum_{\mathbf{x}_f \notin \mathcal{S}_f} \left.\frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{\zeta})}{\partial \alpha_f(\mathbf{x}_f)}\right|_{\substack{\boldsymbol{\alpha}=\boldsymbol{\alpha}^* \\ \boldsymbol{\chi}=\boldsymbol{\chi}^* \\ \boldsymbol{\zeta}=\boldsymbol{\zeta}^*}} \times (\alpha_f(\mathbf{x}_f) - 0) \\ &\geq 0. \end{aligned}$$

□

4.2.3 Application of qLRU-Δ

As we discussed, qLRU-Δ can be adapted to optimize different utility functions $G_f(\cdot)$. In this section we illustrate two specific case studies: hit rate maximization and delay minimization with CoMP techniques. We first describe what form the general qLRU-Δ assumes in these cases and then illustrate with some experiments the convergence result in Proposition 10.

Hit rate maximization

The gain is simply 1 from a hit and 0 from a miss, i.e.,

$$g_f(\mathbf{X}_f, u) = \mathbb{1}(J_{u,f} \neq \emptyset),$$

where $\mathbb{1}(\cdot)$ denotes the indicator function. According to (4.6) with $\beta = 1$, each BS b with a local copy ($b \in J_{u,f}$) moves the content to the front of the cache with probability

$$\begin{aligned} p_f^{(b)}(u) &= \beta \cdot \Delta g_f^{(b)}(\mathbf{X}_f(t), u) \\ &= \mathbb{1}(J_{u,f} \neq \emptyset) - \mathbb{1}(J_{u,f} \setminus \{b\} \neq \emptyset) \\ &= 1 - \mathbb{1}(J_{u,f} \setminus \{b\} \neq \emptyset) \\ &= \mathbb{1}(J_{u,f} \setminus \{b\} = \emptyset) = \mathbb{1}(J_{u,f} = \{b\}). \end{aligned}$$

Similarly, from (4.8), for $\delta = 1$, at least one of the BSs without the content (i.e., those in $I_{u,f} \setminus J_{u,f}$) decides to store an additional copy of f with probability

$$\begin{aligned} q_f^{(b)}(u) &= \delta \cdot q \cdot \Delta g_f^{(b)}(\mathbf{X}_f(t) \oplus \mathbf{e}^{(b)}, u) \\ &= q \cdot (\mathbb{1}(J_{u,f} \cup \{b\} \neq \emptyset) - \mathbb{1}(J_{u,f} \neq \emptyset)) \\ &= q \cdot (1 - \mathbb{1}(J_{u,f} \neq \emptyset)) = q \cdot \mathbb{1}(J_{u,f} = \emptyset). \end{aligned}$$

qLRU-Δh Online Caching Policy

Upon a miss ($J_{u,f} = \emptyset$), at least one cache decides to retrieve the content with probability q. Upon a hit ($J_{u,f} \neq \emptyset$), the cache serving the content brings it to the front if and only if no other BS is caching f (i.e., $|J_{u,f}| = 1$). We name this specialized version of qLRU-Δ to maximize the hit ratio as qLRU-Δh and we present its formal description in Algorithm 5.

Algorithm 5: qLRU-Δh Caching Policy (for BS b)

Input: I_u, $J_{u,f}$.
if $b \in J_{u,f}$ *and* $|J_{u,f}| = 1$ **then**
  Move-to-the-front f
else
  with probability q **do**
    Evict file f_{rear} from the rear;
    Insert f to the front;
  end
end

Delay minimization with CoMP

In this case, we define the performance gain function directly as the experienced delay saving related to the maximum delay possible, i.e.,

$$\begin{aligned} g_f(\mathbf{X}_f, u) &= d^{(0)} - d_{u,f}(\mathbf{X}_f) \\ &= d^{(0)} - \min\left(t_u(J_{u,f}),\, d^{\text{BH}} + t_u(J_{u,f} \cup \{b'\})\right), \end{aligned}$$

where we adapt the experienced delay originally defined in (3.3) to the matrix notation and $d^{(0)} \triangleq \max_{u',f',\mathbf{x}'_f}\{d_{u',f'}(\mathbf{x}'_f)\}$ is a bound on the retrieval time, e.g., equal to the sum of the backhaul delay and the maximum delay on the transmission channel. Note that

$$\begin{aligned} \Delta g_f^{(b)}(\mathbf{x}_f, u) &= g_f(\mathbf{x}_f, u) - g_f(\mathbf{x}_f \ominus \mathbf{e}^{(b)}, u) \\ &= d^{(0)} - d_{u,f}(\mathbf{x}_f) - \left(d^{(0)} - d_{u,f}(\mathbf{x}_f \ominus \mathbf{e}^{(b)})\right) \\ &= d_{u,f}(\mathbf{x}_f \ominus \mathbf{e}^{(b)}) - d_{u,f}(\mathbf{x}_f), \end{aligned} \tag{4.31}$$

where $d^{(0)}$ cancels out and then the choice of its value is irrelevant for the algorithm.

The total expected delay gain per request for file f in a fixed allocation $\mathbf{x}_f$ is then

$$G_f(\mathbf{x}_f) = \sum_{u\in[U]} \lambda_{f,u} \cdot \mathbb{E}\left[g_f(\mathbf{x}_f, u)\right] \tag{4.32}$$

$$= \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} (d^{(0)} - d_{u,f}(\mathbf{x}_f)), \tag{4.33}$$

where we make $\lambda_{f,u} = \lambda_f \cdot \frac{1}{U}$. Finally, we adapt our reference maximization problem, by

simply considering

$$\begin{aligned} G(\mathbf{x}) &= \sum_{f\in[F]} G_f(\mathbf{x}_f) \\ &= \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} (d^{(0)} - d_{u,f}(\mathbf{X}_f)) \\ &= d^{(0)} - \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} d_{u,f}(\mathbf{X}_f). \end{aligned}$$

When UE u requests file f, each BS b with a local copy ($b \in J_{u,f}$) moves the content to the front of the cache with probability

$$\begin{aligned} p_f^{(b)}(u) &= \beta \cdot \Delta g_f^{(b)}(\mathbf{x}_f) \\ &= \beta \cdot \left(d_{u,f}(\mathbf{x}_f \ominus \mathbf{e}^{(b)}) - d_{u,f}(\mathbf{x}_f) \right), \end{aligned} \tag{4.34}$$

where $\beta \triangleq \max_{u',f',b',\mathbf{x}'_f} \left(d_{u',f'}(\mathbf{x}'_f \ominus \mathbf{e}^{(b')}) - d_{u',f'}(\mathbf{x}'_f) \right)$ guarantees that $p_f^{(b)}(u) \in (0,1]$.

Similarly, from (4.8), at least one of the BSs without the content (i.e., all BSs in $I_{u,f} \setminus J_{u,f}$) decides if storing an additional copy of f with probability

$$\begin{aligned} q_f^{(b)}(u) &= q \cdot \delta \cdot \Delta g_f^{(b)}(\mathbf{x}_f \oplus \mathbf{e}^{(b)}) \\ &= q \cdot \delta \cdot \left(d_{u,f}(\mathbf{x}_f) - d_{u,f}(\mathbf{x}_f \oplus \mathbf{e}^{(b)}) \right), \end{aligned} \tag{4.35}$$

where $\delta \triangleq \max_{u',f',b',\mathbf{x}'_f} \left(d_{u',f'}(\mathbf{x}'_f \mathbf{e}^{(b')}) - d_{u',f'}(\mathbf{x}'_f) \right)$ has the same role as β so it guarantees that $q_f^{(b)}(u) \in (0,1]$.

qLRU-Δd Online Caching Policy

Upon a request (u, f), given the current cache allocation $\mathbf{x}_f$:

- All neighboring BSs caching f ($\forall b \in J_{u,f}$) move f to the front of the cache with probability $p_f^{(b)}(u)$, given by (4.6), otherwise they cache f.
- The remaining BSs ($\forall b \in I_u \setminus J_{u,f}$), with probability $q_f^{(b)}(u)$ given by (4.8), evict the file at the rear of the cache and insert f at the front.

We name this specialized version of qLRU-Δ to maximize the delay saving (or to minimize the average delay) as qLRU-Δd and we present its formal description in Algorithm 6 from the individual perspective of each BS b.

Algorithm 6: qLRU-Δd Caching Policy (for BS b)

Input: I_u, $J_{u,f}$, $V_u^{(b')}, \forall b' \in I_u$, and d^{BH}

if $b \in J_{u,f}$ **then**
  with prob. $p_f^{(b)}(u)$ in (4.34) **do**
    Move-to-the-front f
  end
else
  with prob. $q_f^{(b)}(u)$ in (4.35) **do**
    Evict file at the rear
    Insert f to cache
  end
end

Remark 3: From equations (3.3) and (4.31) we see that probabilities $p_{u,f}^{(b)}$ and $q_{u,f}^{(b)}$ 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., $\sum_{b' \in J_{u,f}} V_u^{(b')}$), (ii) the SNR $V_u^{(b)}$ of the local channel from b to u, and (iii) the backhaul delay d^{BH}. In cellular networks, each UE takes SNR measurements of BSs within range [82]. Thus, the information needed to set $q_{u,f}^{(b)}$ and $p_{u,f}^{(b)}$ can be obtained with negligible overhead simply being piggybacked in uplink communication from u to the BSs.

4.3 Handling Non-Stationary Requests

While qLRU-Δ converges to a local optimum for stationary popularities, the slow insertion process, required by qLRU-Δ to converge (see Proposition 10), can become problematic in practice, 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-Δ online caching policy.

In 2LRU-Δ, 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 $\mathrm{ID}(f)$. Here, we introduce the support variables $\hat{J}_{u,f}$ indicating the set of u's neighboring BSs caching file $\mathrm{ID}(f)$. The two-layers structure along with the least-recently-used eviction rule are the core of plain 2LRU. On top of these characteristics, 2LRU-Δ additionally performs insertions and moves-to-the-front (MTFs) with probability proportional to the marginal performance gain, given in Equation (4.3).

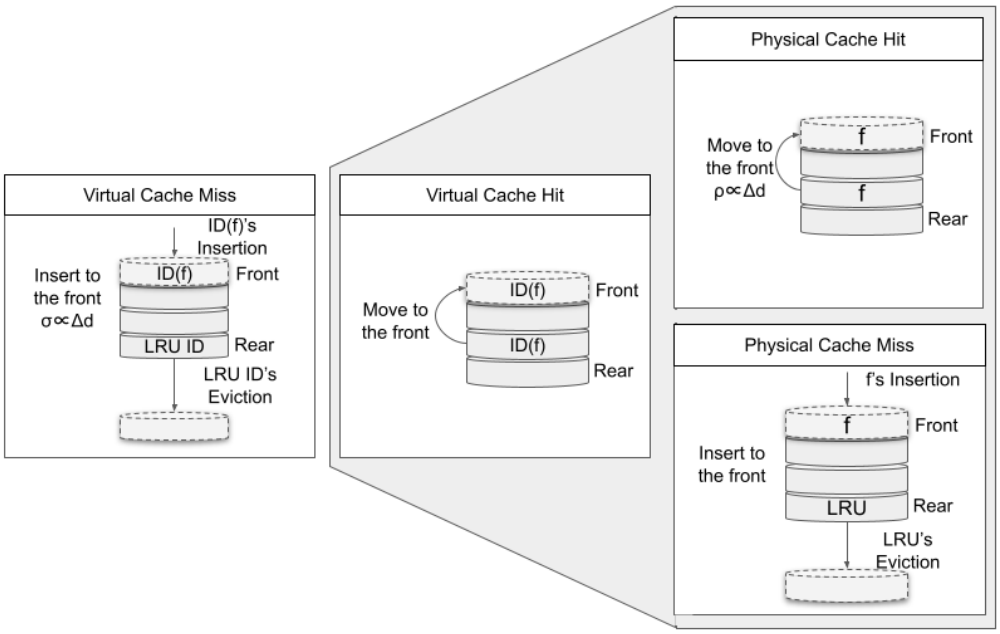


Figure 4.5 – Illustration of 2LRU-Δ operation from the perspective of a single BS b when UE u has requested file f.

2LRU-Δ Online Caching Policy

Upon a request (u, f), given the current physical $\mathbf{x}_f$:

- All neighboring BSs caching ID(f) ($\forall b \in \hat{J}_{u,f}$) move ID($f$) to the front of the virtual cache; then:
  - Each BS $b \in \hat{J}_{u,f}$ storing file f in the physical cache, i.e., $b \in J_{u,f}$, moves f to the front of the physical cache with probability

$$p_f^{(b)}(u) = \beta \cdot \Delta g_f^{(b)}(\mathbf{x}_f, u),$$

    given by Equation (4.6),
  - Each of the remaining BSs, i.e., $\forall b \in I_u \setminus \hat{J}_{u,f}$, evicts the file at the rear of the physical cache and inserts file f to the front.
- The remaining BSs ($\forall b \in \hat{J}_{u,f} \setminus J_{u,f}$), with probability

$$\sigma_f^{(b)}(u) = \delta \cdot \Delta g_f^{(b)}(\mathbf{x}_f, u),$$

  evict the ID at the rear of the virtual cache and insert ID(f) at the front.

We formalize 2LRU-Δ caching policy in Algorithm 7 from the individual perspective of each BS b. As qLRU-Δ, 2LRU-Δ has constant complexity in time and number of messages.

Algorithm 7: 2LRU-Δ Caching Policy (for BS b)

Input: I_u, $J_{u,f}$, $\hat{J}_{u,f}$, and $g_f^{(b)}(\cdot,\cdot)$

if $b \in \hat{J}_{u,f}$ **then**
  Move-to-the-front ID(f) at virtual cache
  if $b \in J_{u,f}$ **then**
    with prob. $p_f^{(b)}(u)$ in Equation (4.6) **do**
      Move-to-the-front f at physical cache
    end
  else
    Evict file at the rear of physical cache
    Insert f to physical cache
  end
else
  with prob. $\sigma_f^{(b)}(u)$ in Equation (4.3) **do**
    Evict file's ID at the rear of virtual cache
    Insert ID(f) to virtual cache
  end
end

Remark 4: We do not provide theoretical guarantees for 2LRU-Δ similar to those of qLRU-Δ. However, its two-layer structure works as a more responsive filter, which makes it easier for 2LRU-Δ to reflect short-term popularity variabilities. This fact makes 2LRU-Δ 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 [9]. 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-Δ is a strong candidate to cope with the delay minimization problem under non-stationary request processes, as we observe empirically in Chapter 5.

In what follows we discuss how to adapt 2LRU-Δ to approach hit ratio and average delay minimization cases.

2LRU-Δh Online Caching Policy

We name this specialized version of 2LRU-Δ to maximize the hit ratio as 2LRU-Δh and we present its formal description in Algorithm 8 from the individual perspective of each BS b.

In this case, the MTF probability is analogous to qLRU-Δh, i.e., $p_f^{(b)}(u) = 1$ if BS b is the only BS neighboring UE u and physically caching file f, otherwise $p_f^{(b)}(u) = 0$. Moreover, considering $\delta = 1$, the insertion probability is given by

$$\begin{aligned}\sigma_f^{(b)}(u) &= \delta \cdot \Delta g_f^{(b)}(\mathbf{x}_f \oplus \mathbf{e}^{(b)}, u) \\ &= \mathbb{1}(J_{u,f} \cup \{b\} \neq \emptyset) - \mathbb{1}(J_{u,f} \neq \emptyset) \\ &= 1 - \mathbb{1}(J_{u,f} \neq \emptyset) \\ &= \mathbb{1}(J_{u,f} = \emptyset),\end{aligned}$$

i.e., for each BSs missing file f's ID at the virtual cache, if no BSs cache ID(f), then it evicts the ID at the rear and insert ID(f) to the front, if ID(f) is not already present.

Algorithm 8: 2LRU-Δh Caching Policy (for BS b)

Input: I_u, $J_{u,f}$, and $\hat{J}_{u,f}$
if $b \in \hat{J}_{u,f}$ **then**
    Move-to-the-front ID(f) at virtual cache
    if $b \in J_{u,f}$ **then**
        if $|J_{u,f}| = 1$ **then**
            Move-to-the-front f at physical cache
        end
    else
        Evict file at the rear of physical cache
        Insert f to physical cache
    end
else
    if $|J_{u,f}| = 0$ **then**
        Evict file's ID at the rear of virtual cache
        Insert ID(f) to virtual cache
    end
end

2LRU-Δd Online Caching Policy

We name this specialized version of 2LRU-Δ to maximize the average delay saving (or to minimize the average delay) as 2LRU-Δd and we present its formal description in Algorithm 9 from the individual perspective of each BS b.

In this case, we have the MTF probability defined as

$$p_f^{(b)}(u) = \beta \cdot \Delta g_f^{(b)}(\mathbf{x}_f) = \beta \cdot \left(d_{u,f}(\mathbf{x}_f \ominus \mathbf{e}^{(b)}) - d_{u,f}(\mathbf{x}_f) \right) \tag{4.36}$$

and the insertion probability as

$$\sigma_f^{(b)}(u) = \delta \cdot \Delta g_f^{(b)}(\mathbf{x}_f \oplus \mathbf{e}^{(b)}) = \delta \cdot \left(d_{u,f}(\mathbf{x}_f) - d_{u,f}(\mathbf{x}_f \oplus \mathbf{e}^{(b)}) \right). \tag{4.37}$$

Algorithm 9: 2LRU-Δd Caching Policy (for BS b)

Input: I_u, $J_{u,f}$, $\hat{J}_{u,f}$, and $d_{u,f}(\cdot)$
if $b \in \hat{J}_{u,f}$ **then**
  Move-to-the-front ID(f) at virtual cache
  if $b \in J_{u,f}$ **then**
    with prob. $p_f^{(b)}(u)$ in Equation (4.36) **do**
      Move-to-the-front f at physical cache
    end
  else
    Evict file at the rear of physical cache
    Insert f to physical cache
  end
else
  with prob. $\sigma_f^{(b)}(u)$ in Equation (4.37) **do**
    Evict file's ID at the rear of virtual cache
    Insert ID(f) to virtual cache
  end
end

4.4 Special Case: Heterogeneous File Sizes

In this section, we consider that each file $f \in [F]$ has size S_f, given in bytes, which may be different from the other files. In this case, the cache capacity C must be given in terms of the total amount of data, also in bytes. Then, we consider the following static optimization problem

Problem 7:

$$\begin{aligned} \text{(GSOHetSize)} \qquad & \underset{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_F}{\text{maximize}} && G(\mathbf{x}) \triangleq \sum_{f\in[F]} G_f(\mathbf{x}_f) \\ & \text{subject to} && \sum_{f\in[F]} S_f \cdot x_f^{(b)} \leq C \quad \forall b \in [F], \\ & && x_f^{(b)} \in \{0,1\} \quad \forall f \in [F], \forall b \in [B]. \end{aligned}$$

We propose a caching policy that is a variant of qLRU-Δ and is able to asymptotically converge to the allocation that optimizes Problem 7. We call the policy qLRU-HS, as it is inspired by qLRU and takes explicitly into account files with heterogeneous sizes, and its operation depends on the quantity

$$\frac{\Delta g_f^{(b)}(\mathbf{X}_f(t),u)}{S_f} = \frac{g_f^{(b)}(\mathbf{X}_f(t),u) - g_f^{(b)}\left(\mathbf{X}_f(t) \ominus \mathbf{e}^{(b)},u\right)}{S_f}, \tag{4.38}$$

which is the marginal gain of performance UE u experiences thanks to the copy of file f at BS b given f's allocation in the whole network divided by file f's size (i.e., its marginal gain per byte occupied in the cache).

We describe qLRU-HS operation as follows: Upon a request (u, f), given the current allocation $\mathbf{X}_f$:

- All neighboring BSs caching f ($\forall b \in J_{u,f}$) independently move f from its current position in the queue to the front with probability

$$p_f^{(b)} \triangleq \beta \cdot \frac{\Delta g_f^{(b)}(\mathbf{X}_f(t),u)}{S_f}, \tag{4.39}$$

  where constant β ensures that $p_f^{(b)} \in (0,1]$, e.g.,

$$\beta = \min_{u',f',\mathbf{x}'_f>0} \left\{ \frac{S_{f'}}{\Delta g_{f'}^{(b')}(\mathbf{x}'_f,u')} \right\}. \tag{4.40}$$

- For the remaining BSs ($\forall b \in I_u \backslash J_{u,f}$): (i) If there is enough cache space, f is directly inserted at the front, (ii) otherwise, with probability

$$q_f^{(b)}(u) = q^{\gamma_b}, \tag{4.41}$$

  for some positive constant γ_b, each BS decided to individually evict from the rear

enough files to make room for f and insert it to the front. We refer to the file at the rear of the cache as f_{rear}.

We formalize qLRU-HS policy in Algorithm 10 from the perspective of each BS b.

Algorithm 10: qLRU-HS Caching Policy (for BS b)

Input: I_u, $J_{u,f}$, $g_f^{(b)}(\cdot)$, $S_f, \forall f \in [F]$, and $X_f^{(b)}, \forall f \in [F]$.

```
if b ∈ J_{u,f} then
    with probability p_f^{(b)} in (4.39) do
        Move-to-the-front f
    end
else
    if C − Σ_{f'∈[F]} S_{f'} · X_{f'}^{(b)} ≥ S_f then
        Insert f to the front;
    else
        with probability q_f^{(b)}(u) do
            while C − Σ_{f'∈[F]} S_{f'} · X_{f'}^{(b)} < S_f do
                Evict file f_rear from the rear;
            end
            Insert f to the front;
        end
    end
end
```

Proposition 17: Under IRM, CTA, and EA, a network of qLRU-HS caches asymptotically achieves an optimal caching configuration, when $q \to 0$, even if files have different sizes.

The optimality proof for qLRU-HS policy follows the same steps of the optimality proof for qLRU-Δ described in Section 4.2.2. We revisit each step of the proof and point out the differences.

First, under CTA and EA, a network of qLRU-HS caches may be modeled with the same group of CTMCs. However, we observe that the result in Lemma 11 is adapted to the marginal gain function $\Delta g_f^{(b)}(\mathbf{X}_f(t), u)$ related to the file size S_f, so the transition rates are defined as follows:

- Upward transition: $\rho[\mathbf{x}_f \to \mathbf{y}_f] \propto q^{\boldsymbol{\gamma}^\top(\mathbf{y}_f - \mathbf{x}_f)}$,
- Downward transition: $\rho[\mathbf{y}_f \to \mathbf{x}_f] \propto q^{\frac{\Delta G_f^{(b_0)}(\mathbf{y}_f)}{S_f}}$ (with a single different BS b_0).

Consequently, the direct resistance of upward and downward transitions are, respectively, $r_f(\mathbf{x}_f, \mathbf{y}_f) = \boldsymbol{\gamma}^\top(\mathbf{y}_f - \mathbf{x}_f)$ and $r_f(\mathbf{y}_f, \mathbf{x}_f) = \frac{\Delta G_f^{(b_0)}(\mathbf{y}_f)}{S_f}$. After these first changes, the potential function is redefined to

$$\phi_f(\mathbf{x}_f) \triangleq \frac{G_f(\mathbf{x}_f)}{S_f} - \boldsymbol{\gamma}^\top \cdot \mathbf{x}_f. \tag{4.42}$$

Note that Lemmas 12, 13, and 14 are not affected by the sizes heterogeneity. However, because the potential function has a new shape, Lemma 15 must be adapted as well. In other words, we need to show that

Lemma 18 (Equivalent of Lemma 15 for qLRU-HS)**:** $\max_{\mathbf{x}_f \in A, \mathbf{y}_f \in A^c} \{\phi_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{y}_f)\}$ is achieved by a pair of parent-child nodes in the resistance graph $\mathcal{G}_f$.

Proof. Let $\hat{\mathbf{x}}_f$ and $\hat{\mathbf{z}}_f$ be two nodes in $\mathcal{G}_f$ such that $\hat{\mathbf{x}}_f \in A, \hat{\mathbf{z}}_f \in A^c$, and $|\hat{\mathbf{z}}_f| > |\hat{\mathbf{x}}_f| + 1$. The transition $\hat{\mathbf{x}}_f \to \hat{\mathbf{z}}_f$ has resistance $r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f)$. Now, consider a path from $\hat{\mathbf{x}}_f$ to $\hat{\mathbf{z}}_f$ that traverses nodes with strictly larger weights. By construction, there exists a pair $(\mathbf{x}'_f, \mathbf{y}'_f)$ in this path, such that $\mathbf{x}'_f \in A$ and $\mathbf{y}'_f \in A^c$. Then,

$$\begin{aligned}
&\phi_f(\hat{\mathbf{x}}_f) - r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f) \\
&\quad = \frac{G_f(\hat{\mathbf{x}}_f)}{S_f} - \gamma^\top \hat{\mathbf{x}}_f - r_f(\hat{\mathbf{x}}_f, \hat{\mathbf{z}}_f) && \text{by def. of } \phi_f(\cdot) \\
&\quad = \frac{G_f(\hat{\mathbf{x}}_f)}{S_f} - \gamma^\top \hat{\mathbf{x}}_f - \gamma^\top(\hat{\mathbf{z}}_f - \hat{\mathbf{x}}_f) && \text{by def. of } r_f(\cdot,\cdot) \\
&\quad = \frac{G_f(\hat{\mathbf{x}}_f)}{S_f} - \gamma^\top \hat{\mathbf{z}}_f \\
&\quad \leq \frac{G_f(\mathbf{x}'_f)}{S_f} - \gamma^\top \hat{\mathbf{z}}_f && \text{by monotonicty of } G_f(\cdot) \\
&\quad = \frac{G_f(\mathbf{x}'_f)}{S_f} - \gamma^\top \mathbf{x}'_f - S_f \gamma^\top(\mathbf{y}'_f - \mathbf{x}'_f) - \gamma^\top(\hat{\mathbf{z}}_f - \mathbf{y}'_f) \\
&\quad = \phi_f(\mathbf{x}'_f) - \gamma^\top(\mathbf{y}'_f - \mathbf{x}'_f) - \gamma^\top(\hat{\mathbf{z}}_f - \mathbf{y}'_f) && \text{by def. of } \phi_f(\cdot) \\
&\quad = \phi_f(\mathbf{x}'_f) - r_f(\mathbf{x}'_f, \mathbf{y}'_f) - \gamma^\top(\hat{\mathbf{z}}_f - \mathbf{y}'_f) && \text{by def. of } r_f(\cdot,\cdot) \\
&\quad \leq \phi_f(\mathbf{x}'_f) - r_f(\mathbf{x}'_f, \mathbf{y}'_f) && \hat{\mathbf{z}}_f - \mathbf{y}'_f \text{ is non-negative}
\end{aligned}$$

Moreover, transitions to ancestors are not valid in the MCs, so we set the reverse edges' resistances to infinite in $\mathcal{G}_f$, i.e., $r_f(\hat{\mathbf{z}}_f, \hat{\mathbf{x}}_f) = +\infty$. As a consequence, the system of

modified balance equations can be simplified, considering only parent-child pairs

$$\begin{cases} \max\limits_{\substack{\mathbf{x}_f \in A, \mathbf{y}_f \in A^c \\ |\mathbf{y}_f| = |\mathbf{x}_f|+1}} \nu_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{y}_f) = \\ \qquad = \max\limits_{\substack{\mathbf{x}_f \in A, \mathbf{y}_f \in A^c \\ |\mathbf{y}_f| = |\mathbf{x}_f|+1}} \nu_f(\mathbf{y}_f) - r_f(\mathbf{y}_f, \mathbf{x}_f), \ \forall A \subset \{0,1\}^B \\ \max\limits_{\mathbf{x}_f \in A} \nu_f(\mathbf{x}_f) = \sigma \end{cases} \tag{4.43}$$

Finally, we show that, for every pair parent-child , $\phi_f(\cdot)$ satisfies a pairwise balance equation. Consider a parent-child pair $\mathbf{x}_f, \mathbf{y}_f$ such that $\mathbf{y}_f = \mathbf{x}_f \oplus \mathbf{e}^{(b_0)}$. Then,

$$\begin{aligned} \phi_f(\mathbf{x}_f) - r_f(\mathbf{x}_f, \mathbf{y}_f) &= \frac{G_f(\mathbf{x}_f)}{S_f} - \gamma^\top \mathbf{x}_f - r_f(\mathbf{x}_f, \mathbf{y}_f) && \text{by def of } \phi_f(\cdot) \\ &= \frac{G_f(\mathbf{x}_f)}{S_f} - \gamma^\top \mathbf{x}_f - \gamma^\top(\mathbf{y}_f - \mathbf{x}_f) && \text{by def. of } r(\cdot,\cdot) \\ &= \frac{G_f(\mathbf{x}_f)}{S_f} - \gamma^\top \mathbf{y}_f \\ &= \frac{G_f(\mathbf{x}_f)}{S_f} - \frac{G_f(\mathbf{y}_f)}{S_f} + \frac{G_f(\mathbf{y}_f)}{S_f} - \gamma^\top \mathbf{y}_f \\ &= \frac{G_f(\mathbf{y}_f)}{S_f} - \frac{\Delta G_f^{(b_0)}(\mathbf{y}_f)}{S_f} - \gamma^\top \mathbf{y}_f && \text{by def. of } \Delta G_f^{(b_0)}(\cdot) \\ &= \phi_f(\mathbf{y}_f) - \frac{\Delta G_f^{(b_0)}(\mathbf{y}_f)}{S_f} && \text{by def. of } \phi_f(\cdot) \\ &= \phi_f(\mathbf{y}_f) - r_f(\mathbf{y}_f, \mathbf{x}_f) && \text{by def. of } r(\cdot,\cdot) \end{aligned}$$

Therefore, $\{\phi_f(\mathbf{x}_f), \forall x_f \in \{0,1\}^B\}$ is the solution of the system (4.43). □

Now, following the same reasoning used to define Problem 10, we consider the linear continuous relaxation of the original static problem for heterogeneous file sizes (Problem 7) as follows:

Problem 11 (GSOHetSize Linear Continuous Relaxation)**:**

$$\begin{aligned}
\underset{\{\alpha_f(\mathbf{x}_f)\}}{\text{maximize}} \quad & \sum_{f\in[F]}\sum_{\mathbf{x}_f\in\{0,1\}^B} \alpha_f(\mathbf{x}_f)G_f(\mathbf{x}_f) && (4.44)\\
\text{subject to} \quad & \sum_{f\in[F]}\sum_{\mathbf{x}_f\in\{0,1\}^B} \alpha_f(\mathbf{x}_f)\cdot S_f\cdot x_f^{(b)} = C, \quad \forall b\in[B]\\
& \sum_{\mathbf{x}_f\in\{0,1\}^B} \alpha_f(\mathbf{x}_f) = 1, \qquad \forall f\in[F]\\
& \alpha_f(\mathbf{x}_f)\geq 0, \qquad \forall f\in[F], \forall \mathbf{x}_f\in\{0,1\}^B.
\end{aligned}$$

Note that we can replace the inequality constraints from Problem 7 with equality constraints, since the fractional variables enforce that optimal solutions are achieved with total cache utilization. Then, the Lagrangian function of Problem 11 objective function (4.44) is

$$\begin{aligned}
L(\boldsymbol{\alpha},\boldsymbol{\chi},\boldsymbol{\zeta}) = & -\sum_{f\in[F]}\sum_{\mathbf{x}_f\in\{0,1\}^B} \alpha_f(\mathbf{x}_f)G_f(\mathbf{x}_f)\\
& + \sum_{b=1}^{B}\chi_b\left(\sum_{f\in[F]}\sum_{\mathbf{x}_f\in\{0,1\}^B} \alpha_f(\mathbf{x}_f)\cdot S_f\cdot x_f^{(b)} - C\right) && (4.45)\\
& + \sum_{f\in[F]}\zeta_f\left(\sum_{\mathbf{x}_f\in\{0,1\}^B}\alpha_f(\mathbf{x}_f) - 1\right).
\end{aligned}$$

Finally, we show that the following assignments satisfy the set of KKT conditions

$$\begin{cases}
\alpha_f^*(\mathbf{x}_f) = \pi_{f,0^+}(\mathbf{x}_f), & \forall f\in[F], \mathbf{x}_f\in\{0,1\}^B,\\
\chi_b^* = \gamma_b, & \forall b\in[B],\\
\zeta_f^* = S_f\cdot \max_{\mathbf{x}'_f\in\{0,1\}^B}\phi_f(\mathbf{x}'_f), & \forall f\in[F].
\end{cases}$$

In fact, for any value q, $\sum_f\sum_{\mathbf{x}_f} x_f^{(b)}\pi_{f,q}(\mathbf{x}_f) = C$ for each b, $\sum_{\mathbf{x}_f}\pi_{f,q}(\mathbf{x}_f) = 1$ for each f, and $\pi_{f,q}(\mathbf{x}_f)\geq 0$ for each f and $\mathbf{x}_f$. The same relations are also satisfied passing to the limit when q converges to 0, then $\{\pi_{f,0^+}(\mathbf{x}_f)\}$ is a feasible solution. Finally,

$$\begin{aligned}
\left.\frac{\partial L(\boldsymbol{\alpha},\boldsymbol{\chi},\boldsymbol{\zeta})}{\partial\alpha_f(\mathbf{x}_f)}\right|_{\substack{\boldsymbol{\alpha}=\boldsymbol{\alpha}^*\boldsymbol{\chi}=\boldsymbol{\chi}^*\\ \boldsymbol{\zeta}=\boldsymbol{\zeta}^*}} &= -G_f(\mathbf{x}_f) + \sum_{b=1}^{B}\gamma_b S_f x_f^{(b)} + S_f\cdot\max_{\mathbf{x}'_f\in\{0,1\}^B}\phi_f(\mathbf{x}'_f)\\
&= -\frac{G_f(\mathbf{x}_f)}{S_f} + \sum_{b=1}^{B}\gamma_b x_f^{(b)} + \max_{\mathbf{x}'_f\in\{0,1\}^B}\phi_f(\mathbf{x}'_f)
\end{aligned}$$

$$= -\phi(\mathbf{x}_f) + \max_{\mathbf{x}'_f \in \{0,1\}^B} \phi_f(\mathbf{x}'_f)$$

$$\begin{cases} = 0 & \text{if } \mathbf{x}_f \text{ is stochastically stable,} \\ > 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{S}_f \subset \{0,1\}^B$ denote the set of stochastically stable states for file f. It follows that

$$\begin{aligned} \nabla L(\boldsymbol{\alpha}^*, \boldsymbol{\chi}^*, \boldsymbol{\zeta}^*)^\intercal (\boldsymbol{\alpha} - \boldsymbol{\alpha}^*) &= \sum_{f \in [F]} \sum_{\mathbf{x}_f \in \{0,1\}^B} \frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{\zeta})}{\partial \alpha_f(\mathbf{x}_f)}\bigg|_{\substack{\boldsymbol{\alpha}=\boldsymbol{\alpha}^* \\ \boldsymbol{\chi}=\boldsymbol{\chi}^* \\ \boldsymbol{\zeta}=\boldsymbol{\zeta}^*}} \times (\alpha_f(\mathbf{x}_f) - \alpha^*_f(\mathbf{x}_f)) \\ &= \sum_{f \in [F]} \sum_{\mathbf{x}_f \in \mathcal{S}_f} 0 \times (\alpha_f(\mathbf{x}_f) - \pi_{f,0^+}(\mathbf{x}_f)) \\ &\quad + \sum_{f \in [F]} \sum_{\mathbf{x}_f \notin \mathcal{S}_f} \frac{\partial L(\boldsymbol{\alpha}, \boldsymbol{\chi}, \boldsymbol{\zeta})}{\partial \alpha_f(\mathbf{x}_f)}\bigg|_{\substack{\boldsymbol{\alpha}=\boldsymbol{\alpha}^* \\ \boldsymbol{\chi}=\boldsymbol{\chi}^* \\ \boldsymbol{\zeta}=\boldsymbol{\zeta}^*}} \times (\alpha_f(\mathbf{x}_f) - 0) \\ &\geq 0. \end{aligned}$$

Chapter 5

Experimental Results

In the previous chapters, we have presented static and dynamic caching solutions for a network of interacting caches. The solutions may be deployed to optimize any performance metric of interest, although we specifically show how to adapt the generic framework to maximize the hit ratio or to minimize the average experienced delay in CCSC networks. We are particularly interested in the delay minimization problem with CoMP. In this case, we were able to derive theoretical results, among which we highlight the most important ones below:

- In the static framework, a greedy algorithm, GreedyAD, is able to find an allocation whose corresponding average delay is not worse than $\frac{1}{2}$ of the optimal in the ideal scenario where SNRs are homogeneous and popularities are stationary.
- In the dynamic framework, if the request process is stationary and follow IRM, an online caching policy, qLRU-Δd, is expected to asymptotically converge to the optimal allocation as the insertion parameter q tends to 0.

The experiments in this chapter were primarily designed to investigate whether and at which level the above theoretical results are preserved empirically in more realistic setups. Some of the questions we aim to answer with our numerical simulations include, but are not limited to,

1. Given that our theoretical results were obtained under some approximations, how small should parameter q be such that the proposed specialized variants of qLRU-Δ, i.e., qLRU-Δh for hit ratio maximization and qLRU-Δd for delay minimization, are able to converge to their respective optimal allocations?
2. How much loss of performance do qLRU-Δ's variants experience when exposed to

non-stationary request processes? Are 2LRU-Δ's variants for hit ratio maximization, 2LRU-Δh, and delay minimization, 2LRU-Δd, better suitable in this case?

3. How do our proposed solutions compare with other state-of-the-art solutions under CCSC networks with different characteristics?

5.1 Experimental Setup

In this section, we first present in detail the common elements of our methodological approach and, then, we discuss the characteristics of the different experimental setups considered in the upcoming analysis.

5.1.1 Cellular Network

We simulate a cellular network based on the *Berlin topology*, presented in Chapter 2, consisting of $B = 10$ BSs geographically fixed (see Figure 2.2). There are 100 UEs randomly and statically placed, such that the *network density* ρ, i.e, the average number of BSs covering each UE, is controlled by simply adjusting the BSs coverage areas, which can be achieved in practice, for example, by tuning their transmission power. In our experiments, as discussed in Chapter 2, we focus on the distinct user areas. For simplicity, we assume that all UEs inside the same user area enjoy homogeneous transmission characteristics towards their common neighboring BSs. It is important to notice that, although the number of UEs is fixed, the number of distinct user areas may change with the network density.

We emphasize that, the important aspects for us are the underlying bipartite structure between clients and servers and the quantitative difference in performance offered by the multiple possibilities of content placement into BSs. For this reason, in order to keep our experimental setup simple, it is reasonable to assume that all BSs transmit at the same power and, therefore, have the same coverage areas.

We consider that all BSs operate in the same base frequency f_0 but use orthogonal channels with the same bandwidth $W = 5$ MHz, e.g., for BS $b \in [B]$, the channel is defined within $[f_0 + (b-1) \cdot W, f_0 + b \cdot W]$ The main experiments of this chapter consider only the scenario where files have homogeneous sizes, such that $S_f = S = 1.0$ Mbytes, $\forall f \in [F]$. Each cache can store up to 100.0 Mbytes of data that allows $C = 100$ files, i.e., less than 1% of the catalog in the stationary request scenario (consisting of $F = 10^6$ files), which is inline with studies about small cell caching [69, 83]. Since all files have the same size and assuming that the backhaul network offers uniform transmission rates to all BSs, we fix the backhaul-access delay to $d^{\text{BH}} = 100.0$ ms. Unless mentioned otherwise,

we will assume in the first scenarios homogeneous file sizes, for simplicity; we will later revisit this assumption and simulate heterogeneous files sizes as well.

5.1.2 Caching schemes

Besides the greedy algorithms presented in Chapter 3 and the online policies proposed in Chapter 4, we implemented the following solutions from related work in our simulations:

- FIFO. The standard first-in first-out: new files are inserted at the front of the cache, pushing the rest of the files closer to the rear, and evicting the last file. Note that files are not moved to the front upon a cache hit.
- qLRU. It is a variant of plain LRU (see Example 5 in Chapter 1), where the insertion of new files, which causes the eviction of the least-recently-used file at the rear, takes place with probability q.
- MULTI-LRU-ONE and MULTI-LRU-ALL [52]. In MULTI-LRU-ONE, a single neighboring BS $b_u \in I_u$ is associated to UE u in advance. Upon request (u, f), only BS b_u updates its cache. In MULTI-LRU-ALL, all UE u's neighboring BSs, $\forall b \in I_u$, update their caches. The updates and insertions are based on plain LRU.
- LFU-ALL. Each BS keeps track of how often every file has been requested within its own coverage area. Whenever a file that is not cached is requested, the least-frequently-requested cached file is evicted to make room for it. We use the *ideal* implementation, where each BS keeps data structures accounting for the request frequency of all files in the catalog (not only the cached files).

Parameter	Values	Experiments (subsection)
Bandwidth	$W = 5$ MHz	All experiments
Files size	$S = 1$ Mbytes	5.2.1-5.3.3
Minimum file size	$S_{\min} = 1$ Gbytes	5.4.1,5.4.2
File size variation	$\Delta S = 9$ Gbytes	5.4.1,5.4.2
	$\Delta S \in \{0, 13, 25, 37, 49\}$	5.4.2
Cache capacity	$C = 100$	5.2.1,5.2.2,5.2.4-5.3.3
	$C \in \{10, 30, 100, 300, 1000, 3000, 10000, 30000, 100000, 300000, 1000000\}$	5.2.3
	$C = 50$ Gbytes	5.4.1,5.4.2
	$C \in \{10, 30, 100, 300, 1000, 3000, 10000, 30000\}$ Gbytes	5.4.2
Backhaul-access delay	$d^{\mathrm{BH}} = 100$ ms	5.3.1-5.3.3
Backhaul rate	$R^{\mathrm{BH}} = 100$ Mbps	5.4.1,5.4.2
Backhaul latency	$M = 10$ ms	5.4.1
	$M \in \{3, 10, 30, 100, 300, 1000\}$ ms	5.4.2
Network density	$\rho = 5.9$ BSs/UE	5.2.1-5.2.6,5.3.3,5.4.1,5.4.2
	$\rho \in \{1.1, 1.7, 3.5, 5.9, 9.4\}$ BSs/UE	5.3.1, 5.3.2, 5.4.2
Zipf exponent	$\alpha = 1.2$	5.2.1-5.3.1,5.4.1,5.4.2
	$\alpha \in \{0.0, 0.3, 0.6, 0.9, 1.2, 1.5\}$	5.2.4
SNR	$V = 10$ dB	5.2.1-5.3.2,5.4.1,5.4.2
Base SNR	V_0 dB	5.3.3
SNR variability	$\Delta V \in \{1, 3, 5, 7, 9\}$ dB	5.3.3
Insertion parameter	$q = 0.001$	5.2.3-5.3.3, 5.4.2
	$q \in \{0.0001, 0.001, 0.01, 0.1, 1\}$	5.2.1,5.2.2,5.4.1

Table 5.1 – List of parameter values and their associated experiments

5.1.3 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 for a file. The following request processes are considered:

- *Stationary Request Process*: 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. Simulations have (i) a *warm up phase*, which should comprise the transient period where the policies are still converging, and (ii) a *measurement phase*, where we can extract statistics about the resulting allocation. In this case, each phase consists of 100 million requests.
- *Non-stationary Request Process*: The idea is to simulate a realistic request process based on a trace provided by Akamai Content Delivery Network [84], which is described in more details in [49]. The trace consists of 17 million requests generated throughout 5 consecutive days.

5.2 qLRU-Δ Convergence to an Optimal Allocation

According to Proposition 10, under stationary request processes, as q tends to 0, qLRU-Δ converges to an optimal allocation. In our first experiments, our goal is to observe this convergence in practice. We consider the Berlin topology with density of $\rho = 5.9$ BSs/UE and the stationary request process with $\alpha = 1.2$. Initially, we show empirical evidences confirming that, for qLRU-Δh and qLRU-Δd, the resulting allocations indeed converge to the optimal ones. Then, we observe that this behavior is retained in different experimental setups by changing some parameters, such as cache capacity, backhaul-access delay, and Zipf exponent. Finally, we show how additional information on the files' popularities may influence the convergence process and we finish our analysis discussing about the speed at which policies that we used in the performance evaluation (see Section 5.3) converge to an optimal allocation.

5.2.1 Convergence of qLRU-Δh – Hit Ratio

Figure 5.1a (left) shows the hit rate achieved by GreedyHR and by qLRU-Δh for different values of q. As q decreases, qLRU-Δh's hit rate converges to that of GreedyHR. The hit rate of qLRU also improves for smaller q. For a single cache, qLRU coincides with qLRU-Δh and it is then implicitly maximizing the hit rate when q converges to 0. But in a networked setting, the deployment of qLRU at each cache does not perform

as well because each cache is myopically maximizing its own hit rate without taking into account the presence of the other caches. Instead, qLRU-Δh correctly takes into account the marginal contribution the cache can bring to the whole system. Finally, FIFO achieves the lowest hit rate as the sojourn time of each content inserted in the cache is roughly the same, independently from its popularity.

We also compare how different the content allocations of qLRU-Δh, qLRU, and FIFO are from the allocation of GreedyHR. To this purpose, we define the *occupancy vector*, whose component i contains the number of copies of file i present in the network averaged during the measurement phase. We then compute the cosine distance[1] of the occupancy vectors of the specific online policy and Greedy-h. Figure 5.1a (right) shows how such distance decreases as q decreases, indicating that the files GreedyHR stores tend to be cached longer and longer under qLRU-Δh, and to a lesser extent under qLRU. The allocations of FIFO and GreedyHR are instead quite far.

Observation 1: Under stationary request process, qLRU-Δh converges to the solution provided by GreedyHR as $q \to 0$.

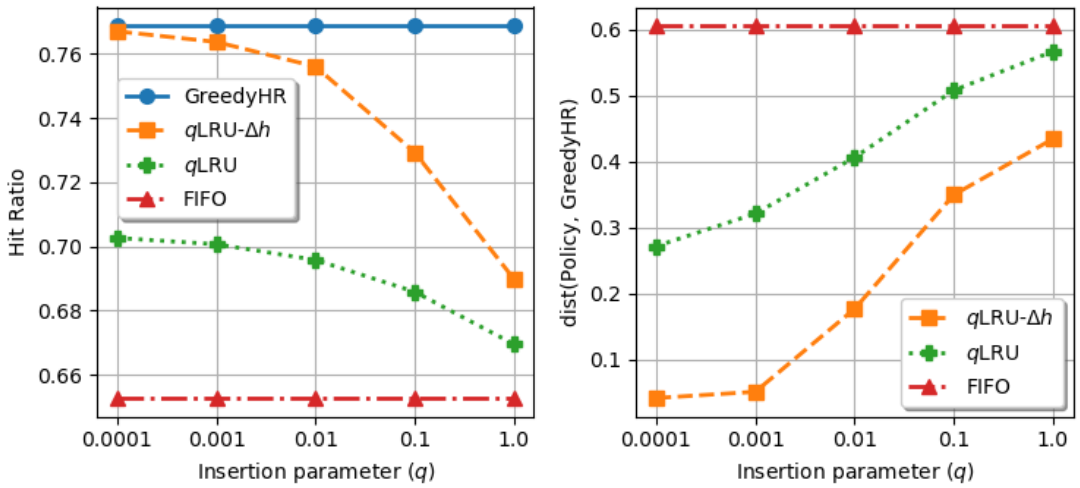
5.2.2 Convergence of qLRU-Δd – Average Delay

To investigate the convergence of qLRU-Δd, we consider the *homogeneous SNR regime* with SNR $V = 10$dB, for all connected pairs BS-UE. This assumption is particularly important because the greedy algorithm, GreedyAD, converges to the optimal allocation if SNRs are homogeneous, thus working as a more reliable comparison baseline. We aim to show empirically that, as q tends to 0, qLRU-Δd converges to a static solution similar to the one provided by GreedyAD.

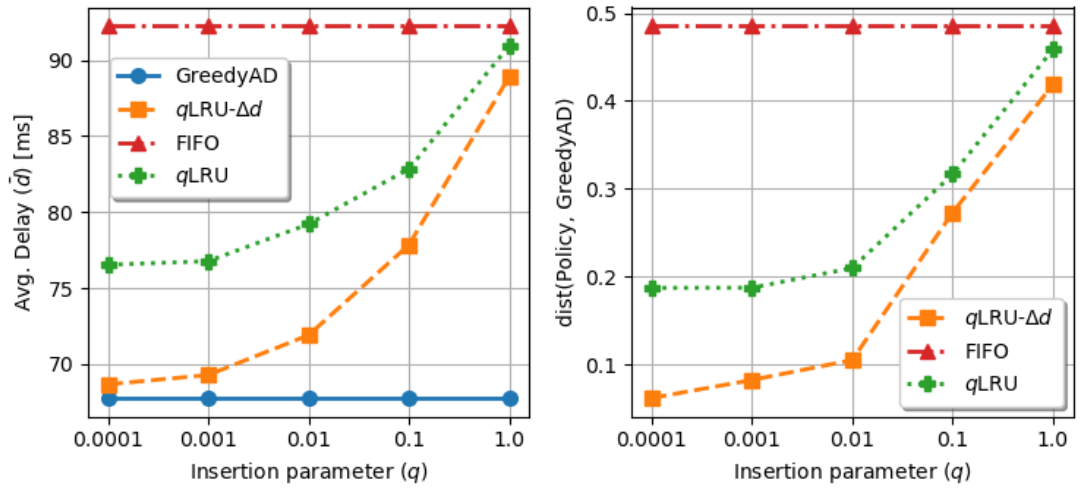
Figure 5.1b (left) shows the average delay of qLRU-Δd and GreedyAD for different values of q. As q decreases, qLRU-Δd's average delay converges to GreedyAD's one. Furthermore, Figure 5.1b (right) shows that, this gradual proximity in the average is not an accident. We observe that, as q decreases, the distance between qLRU-Δd and GreedyAD decreases, indicating that qLRU-Δd tends to store the same files GreedyAD stores.

Observation 2: Under stationary request process, qLRU-Δd converges to the solution provided by GreedyAD as $q \to 0$.

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



(a) Hit ratio maximization: Hit ratio (left) and allocation distance (right) versus insertion parameter q.



(b) Average delay minimization: Average delay (left) and allocation distance (right) versus insertion parameter q.

Figure 5.1 – Convergence analysis: (a) hit ratio and (b) average delay as q tends to 0. Setup: Berlin topology with density $\rho = 5.9$ BSs/UE, $\alpha = 1.2$, $d^{\mathrm{BH}} = 100$ ms, and $V = 10$ dB. Besides the qLRU-Δ specialized implementation and greedy algorithm corresponding to each metric, results are show for qLRU and FIFO.

5.2.3 Convergence under different cache capacities

Figures 5.2a and 5.2b show the hit ratio and average delay, respectively, of online policies and greedy algorithms as we increase the cache capacity per BS. We fix $q = 0.001$ for qLRU-Δ and qLRU. In both scenarios, qLRU-Δ outperforms all other online policies and it closely follows the result of the corresponding greedy policy. Note that the strange shape of FIFO curves is an artefact of the semi-log graph as shown by the inserts.

Observation 3: For sufficiently small q, qLRU-Δ achieves results close to the optimal ones with respect to hit ratio maximization and average delay minimization across different cache capacities.

5.2.4 Convergence under different d^{BH} and λ_f – Average Delay

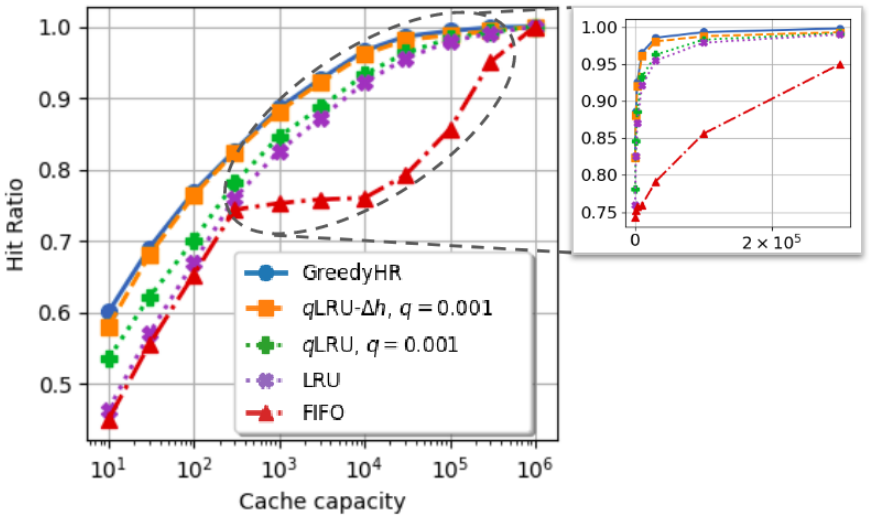
Figure 5.3 shows the average delay achieved by GreedyAD and qLRU-Δ for different values of Zipf exponent α (left) and backhaul-access delay d^{BH} (right), when $q = 0.001$. We observe that the two curves almost match for all different parameter choices, indicating that the convergence is also achieved in multiple settings.

Observation 4: For sufficiently small q, qLRU-Δd achieves delays close to GreedyAD across different backhaul-access delays and popularity distributions.

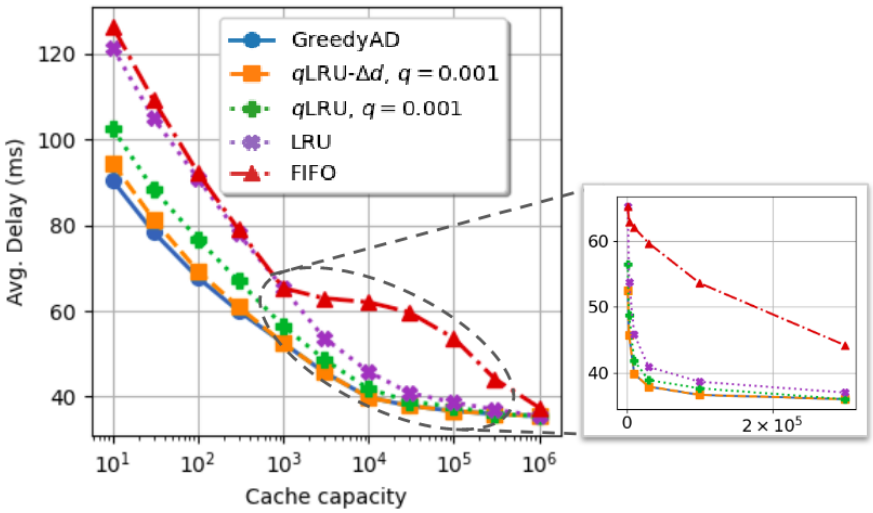
5.2.5 The role of popularities in the convergence process

If some knowledge about content popularity is available, it can be exploited to determine the initial content to allocate in the caches using the offline greedy algorithms, i.e., GreedyHR and GreedyAD, when the metric of interest is the hit ratio or the delay, respectively. We consider noisy popularity estimations in order to decide how to populate the cache initially according to different metrics (resulting from the corresponding greedy algorithm). We show through an experiment in Figure 5.4 that qLRU-Δ can modify the initial cache configuration and improve performance. The left figure considers the hit ratio as objective, the right one the delay. The ground truth popularity follows a Zipf distribution with $\alpha = 1.2$ (as in the previous experiments) and noisy popularity estimations are available: they are obtained multiplying true popularities by random values from a log-normal distribution with expected value 1.0 and variance $e^{\sigma^2} - 1$ (σ^2 is the variance of its logarithm). If $\sigma^2 = 0$, estimated popularity values coincide with the true ones, but the larger the variance σ^2, the less accurate the estimations.

The horizontal dashed lines indicate the performance of the corresponding initial cache configuration under the true request process. The solid curves show the performance



(a) Hit ratio maximization: hit ratio versus cache capacity.



(b) Average delay minimization: average delay versus cache capacity.

Figure 5.2 – Convergence analysis: (a) hit ratio and (b) average delay as cache capacity C increases. Setup: Berlin topology with density $\rho = 5.9$ BSs/UE, $\alpha = 1.2$, $d^{\mathrm{BH}} = 100$ ms, $V = 10$ dB, and $q = 0.001$. Besides qLRU-Δ and greedy algorithms corresponding to each metric, results are shown for qLRU and FIFO.

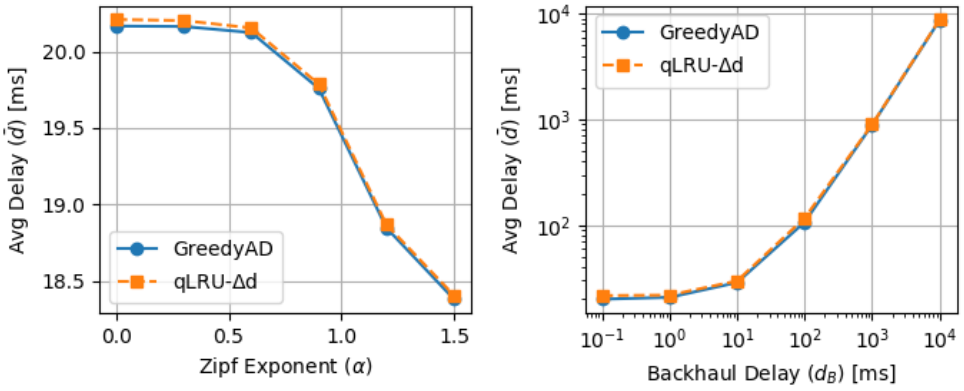


Figure 5.3 – Convergence analysis: average delay provided by qLRU-Δd in comparison with GreedyAD for increasing (left) Zipf exponent and (right) backhaul-access delay. Setup: Berlin topology with density $\rho = 5.9$ BSs/UE, $V = 10$ dB, and $q = 0.001$.

over time of qLRU-Δh (left) and qLRU-Δd (right) with $q = 10^{-3}$. We observe that the curves converge to the same value, that is slightly worse than the initial one when popularity estimations are exact ($\sigma^2 = 0$), but better in all other cases. This result shows that qLRU-Δ can effectively improve performance even when popularity estimates are available. Interestingly, one may expect that the time needed for qLRU-Δ to reach the steady state performance depends on the accuracy of the initial popularity estimates (the more accurate, the less changes would be needed to reach the final cache allocation), but the dependence, if present at all, is very small.

Observation 5: qLRU-Δ can provide performance improvements even when the initial cache allocation was statically determined by inaccurate popularity estimations.

We remark that available popularity information may also be used to tune qLRU-Δ's parameters to speed-up the transient. For example, we can modify the insertion probability, in Equation (4.8), to favor the files the greedy algorithm would have put in the cache. This change is in the same spirit of introducing the factor $\Delta g_f^{(b)}(\mathbf{X}_f(t) \oplus \mathbf{e}^{(b)}, u)$ to the insertion probability in qLRU-Δ definition (in Equation (4.8)). As we discussed at the end of Section 4.2, these changes likely improve convergence speed, but do not affect the steady-state and then qLRU-Δ's optimality guarantees.

5.2.6 Convergence speed – Average Delay

Now, for a fixed network density $\rho = 9.4$ BSs/UE and for each online policy, we show the evolution throughout the simulation of the average delay Figure 5.5 (left) and the hit

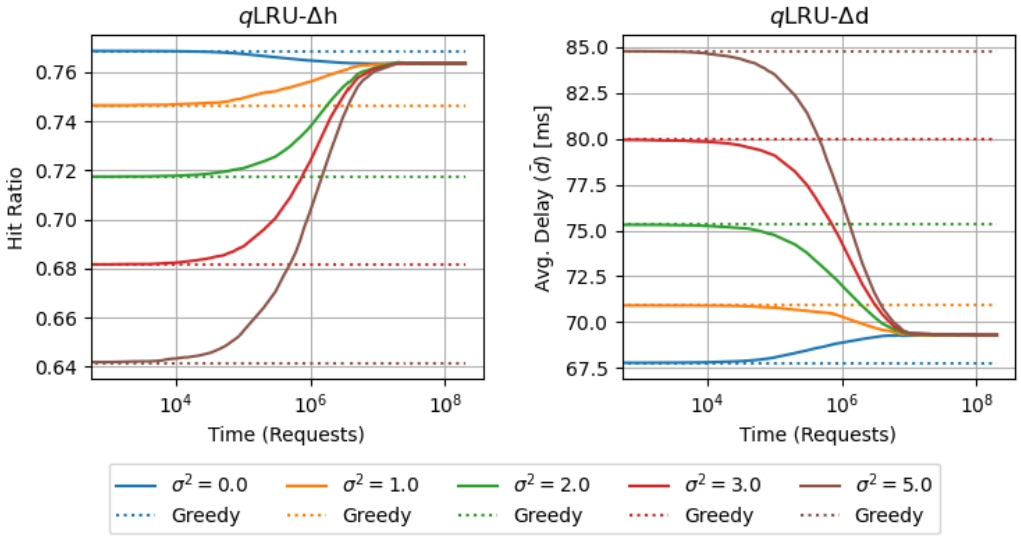


Figure 5.4 – Convergence analysis: qLRU-Δh (left) and qLRU-Δd (right) starting the simulation with the respective greedy allocation for different levels of noisy popularity estimations, represented by variance σ^2. The solid curves are the average of 100 different simulation rounds. Setup: Berlin topology with density $\rho = 5.9$ BSs/UE, $\alpha = 1.2$, $d^{\text{BH}} = 100$ ms, $V = 10$ dB, and $q = 0.001$.

ratio Figure 5.5 (right) every 100 requests. In this experiment, we wish to observe the convergence process of the policies that we will compare later in the next section. Files popularities follow a Zipf law with exponent $\alpha = 1.2$ and we also take $V = 10$ dB and $d^{\text{BH}} = 100$ ms. We fix $q = 0.001$ for qLRU-Δ variants.

First, it is important to note that all policies reach convergence within the total number of requests they were exposed to during the simulation. Then, we highlight that qLRU-Δ variants have worse performance in the beginning due to the lower insertion rate (caused by small parameter q), until the point where they stabilize and present better results (after 10^7 requests in this scenario). In addition to their noticeably faster convergence, 2LRU-like policies reach performance levels close to the qLRU-Δ variants. This fact reveals 2LRU-Δ higher reactivity and suggests its suitability for dealing with non-stationary request processes.

Observation 6: Despite their slightly worse performance under stationary request processes, 2LRU-Δ variants policies present faster convergence in comparison to qLRU-Δ.

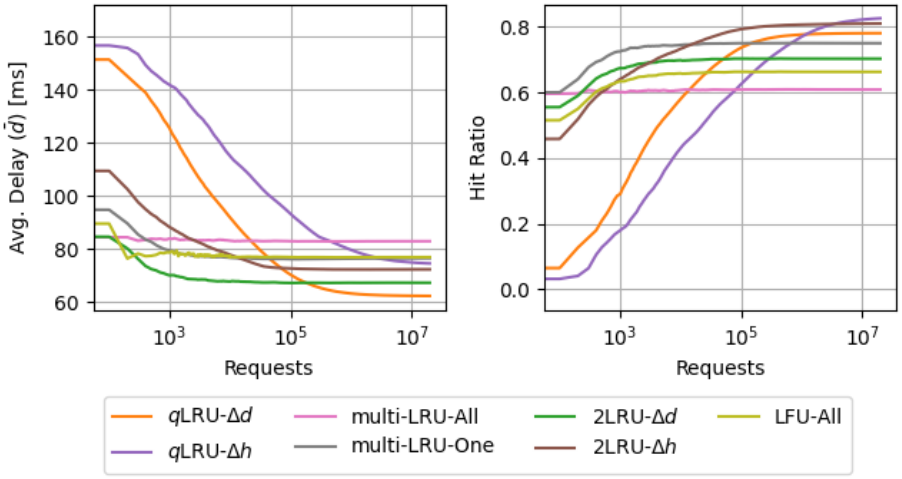


Figure 5.5 – Convergence analysis: Evolution of the average delay (left) and hit ratio (right) achieved by different policies versus the requests (plotted at every 100 requests). Setup: Berlin topology with density $\rho = 9.4$ BSs/UE, $\alpha = 1.2$, $d^{\text{BH}} = 100$ ms, $V = 10$ dB, and $q = 0.001$.

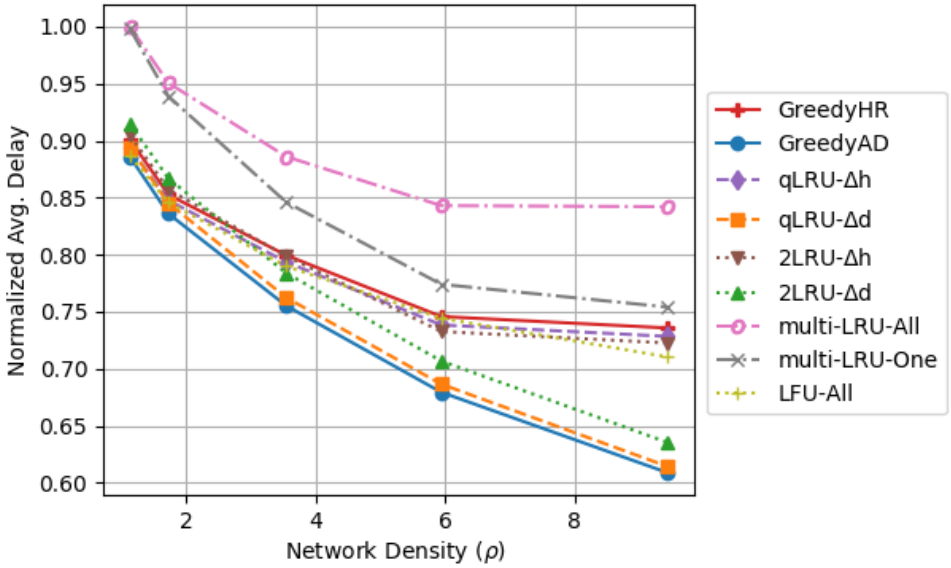
5.3 Comparison with other Caching Policies

In the first set of experiments of this section, we still assume the homogeneous SNR regime and study the policies performance over networks with different densities. We evaluate the results for stationary and non-stationary request processes. Finally, we investigate the policies in a more realistic scenario, where, BSs are exposed to a non-stationary requests process and we consider the heterogeneous SNR regime.

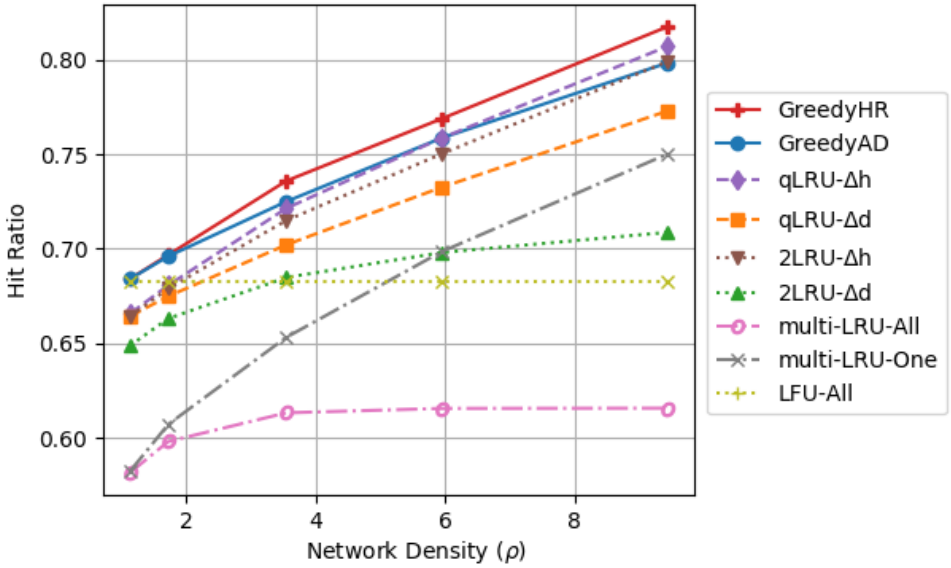
5.3.1 Effect of network density – Stationary requests

We consider the homogeneous SNR regime with $V = 10$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), see Table 2.1. We fix $q = 0.001$ for qLRU-Δ variants.

In the first setting, we assume a *stationary request process*. In Figure 5.6a, 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 the GreedyAD one, reasserting its convergence across different densities. qLRU-Δd reaches performance gains of up to 20% related to GreedyHR and other policies targeting hit ratio maximization. If compared to multi-LRU-All and multi-LRU-One, qLRU-Δd achieves gains of up to 27%.



(a) Normalized Average delay versus network density.



(b) Hit ratio versus network density.

Figure 5.6 – Performance evaluation in terms of (a) Normalized average delay and (b) hit ratio of various policies and greedy algorithms versus the network density. Setup: Berlin topology with $\alpha = 1.2$, $d^{\mathrm{BH}} = 100$ ms, $V = 10$ dB, and $q = 0.001$.

Observation 7: Under stationary requests, qLRU-Δ outperforms state-of-the-art policies, presenting nearly optimal results.

In Figure 5.6b, we show the hit ratio corresponding to the experiment previously described. Policies like qLRU-Δh and 2LRU-Δh outperform other policies as they are designed to maximize the hit ratio, even though they have inferior performance in terms of average delay (see Figure 5.6a).

Observation 8: As expected, policies targeting the hit ratio in general perform worse in terms of average delay.

5.3.2 Effect of network density – Trace-based requests

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, real request processes exhibit strong temporal locality features. Static allocations based on time-average popularities smooth out the variability over short time scales. Consequently, we see in Figure 5.7 that GreedyAD and GreedyHR perform worse than most policies.

Observation 9: Under non-stationary requests, static solutions tend to perform worse than online policies.

On the contrary, 2LRU-Δ variants are highly reactive and may be able to capture short-time popularity variations, offering better performance. Figure 5.7 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-Δh and 23% in comparison with multi-LRU-All.

Observation 10: Under non-stationary requests, 2LRU-Δ outperforms all other policies.

5.3.3 Performance under heterogeneous SNRs

In the online policies simulations, at every request (u, f), the SNRs $V_u^{(b)}, \forall b \in I_u$ are chosen uniformly at random within a range, i.e., $V_u^{(b)} \in [V_0 - \Delta V, V_0 + \Delta V]$. 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. We consider the same Berlin topology, with density of 5.9 BSs/UE, on average, and file requests follow again the Akamai trace. Moreover, we consider two different scenarios for SNR variability:

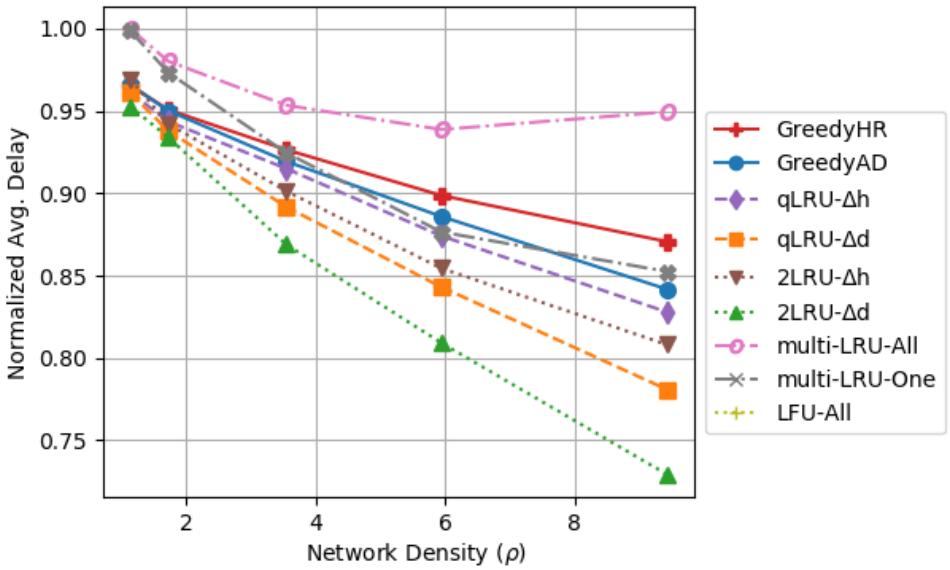
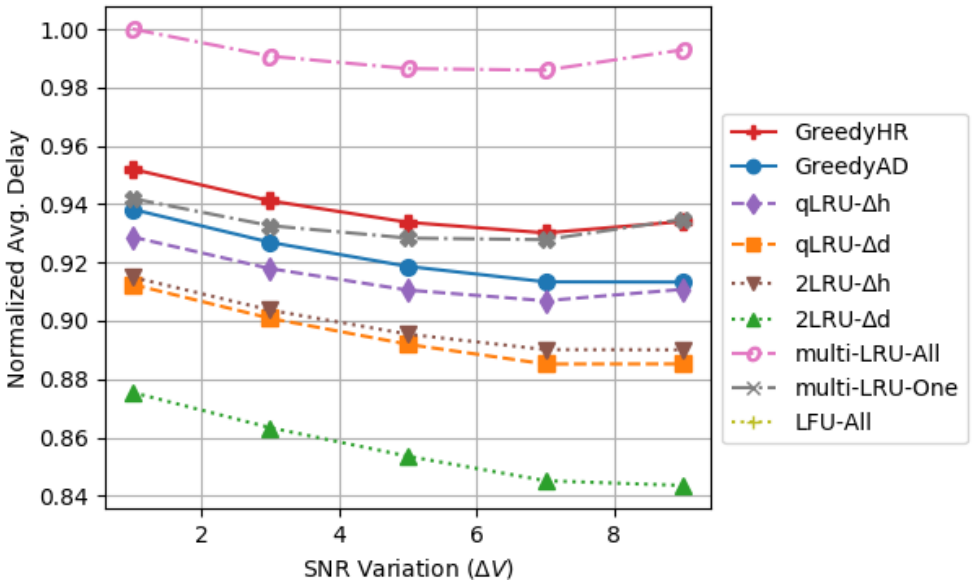


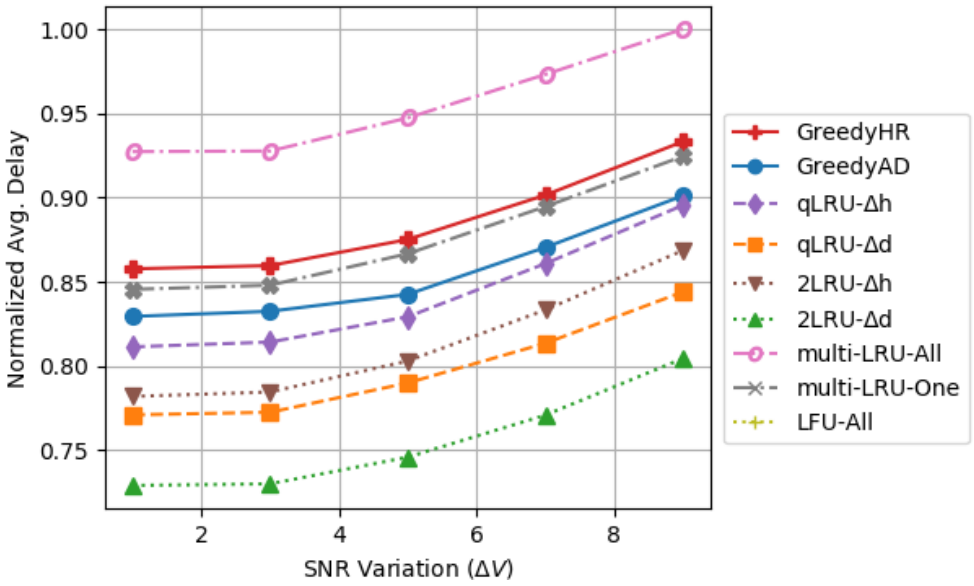
Figure 5.7 – Normalized average delay of various policies and greedy algorithms versus the network density. Setup: Berlin topology with $d^{\mathrm{BH}} = 100$ ms, $V = 10$ dB, and $q = 0.001$. The request process is based on a real trace from which requests were during 5 days.

- *Slow SNR variability regime*: the randomly generated SNRs can be considered constant from the moment the request is posed until it is served. Whenever an additional copy must be retrieved (in case of cache miss or insufficient hit), the BS with the highest SNR, i.e., b' in Equation (3.3), (jointly) transmits the file.
- *Fast SNR variability regime*: In an attempt to represent adverse transmission conditions, we consider that SNRs may change over a timescale corresponding to the backhaul retrieval time. As a consequence, the BS b' that retrieves an additional copy may not have the highest SNR by the time the copy is available. In our simulations, BS b' is chosen independently at random.

First, in Figure 5.8a, we show the performance of the caching policies under slow SNR variability regime. We present the average delay versus the SNR variation ΔV. In this setup, we fix the base SNR to $V_0 = 10$ dB and its variability ranges from $\Delta V = 1$ dB to $\Delta V = 9$ dB. We observe that all curves decrease for smaller values of SNR variation ($\Delta V \in [1.0, 7.0]$). The average delay tends to increase again for larger SNR variation ($\Delta V \geq 9.0$) 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



(a) Slow SNR variability regime: normalized average delay versus SNR variability.



(b) Fast SNR variability regime: normalized average delay versus SNR variability.

Figure 5.8 – The normalized average delay achieved by various policies versus the SNR variation in (a) slow and (b) fast SNR variability regimes. Setup: Berlin topology with density $\rho = 5.9$, $d^{\mathrm{BH}} = 100$ ms, $q = 0.001$, and base SNR $V_0 = 10$ dB.

gain (related to 2LRU-Δ and MULTI-LRU-ALL) moderately increases with ΔV, going from 13% to around 15%.

Observation 11: The proposed policies outperform other state-of-the-art solutions and the SNR variation has low impact on the techniques' 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 Figure 5.8b, we show the performance of the caching policies under fast SNR variability regime. All policies present a strictly increasing behavior. This fact is explained by Jensen's inequality, since the delay is now a convex random function: Given $V = V_0 + \Delta V$ and $V' = V_0 - \Delta V$, the delay reduction achieved with the larger V is smaller than the delay increase due to the smaller V'.

Observation 12: In a scenario with more unstable transmission conditions (fast SNR variability), the average delay strictly increases with the SNR variation.

5.4 Special Case: Heterogeneous File Sizes

In this last part, we turn our attention to heterogeneous file sizes, and the proposed algorithms, IGA and qLRU-HS, addressing the delay minimization problem stated in Problem 8. In this section, we focus on the general delay minimization problem with heterogeneous file sizes (Problem 7). First, as theoretically stated in Section 4.4, our study may suggest a theoretical analysis of qLRU-HS convergence to the optimal cache allocation when $q^{(b)}$ tends to 0. We assume that $q^{(b)} = q, \forall b \in [B]$ and adapt qLRU-HS operation to specifically minimize the average delay in the homogeneous SNR regime.

We evaluate qLRU-HS performance in different scenarios by comparing it against other policies from related literature, including:

- qLRU-Δd that we introduced in Section 4.2 and proved to by optimal under IRM if the sizes are homogeneous.
- *greedy-dual-size* [85], it aims to maximize the hit ratio in a single-cache setup, considering sizes are heterogeneous. We consider that *all* BSs run an instance of greedy-dual-size and react independently to each request in their cell. We refer to such operation as GDSIZE-ALL, where we append the suffix "All" in analogy to MULTI-LRU-ALL in [52].

- IGA greedy algorithm [42], as discussed in Section 3.5, its average delay reduction is guaranteed to be $(1 - 1/e)$ far from the optimal in symmetric setups. Thus, we use it as a lower bound for the other policies.

Once again, we consider the Berlin topology with $B = 10$ BSs, where all BSs have the same cache capacity, i.e., $C^{(b)} = C, \forall b \in [B]$, and can store up to $C = 50$ GB. Unless otherwise specified, we consider that the backhaul network is able to transmit data at $R^{\text{BH}} = 100$ Mbps with backhaul latency $M = 10$ ms. The wireless channel bandwidth is $W = 5$ MHz and all connected pairs BS-UE have fixed SNR of $V = 10$ dB.

In our simulations, we consider that, at every request, a file is chosen from a catalog of $F = 10^4$ files with probability determined by a Zipf law with exponent $\alpha = 0.8$. As indicated by [86], real file sizes may be represented by a truncated exponential distribution. We randomly generate the file sizes according to an exponential distribution within the interval $[S_{\min}, S_{\min} + \Delta S]$. Unless otherwise specified, we consider $S_{\min} = 1$ GB and $\Delta S = 9$ GB. As in the previous experiments with stationary request process, we split the simulation into warm-up and measurement phases, each having 10^7 requests.

5.4.1 Convergence Analysis

According to Proposition 17, as q tends to 0, qLRU-HS converges to an optimal allocation. In our first experiments, our goal is to observe this convergence in practice. We consider the Berlin topology with density of $\rho = 5.9$ BSs/UE.

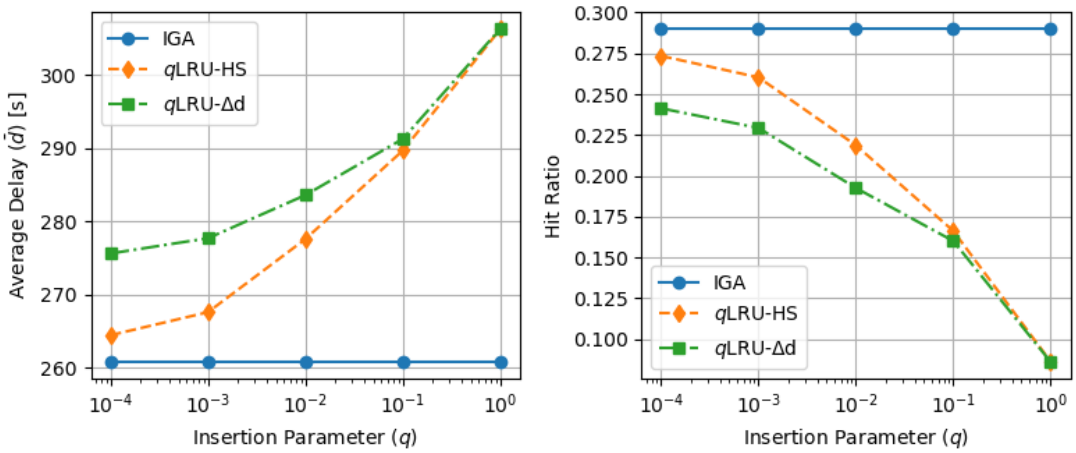


Figure 5.9 – Convergence Analysis: Average delay $\bar{d}$ (left) and hit ratio (right) versus q. Setup: Berlin topology with density $\rho = 5.9$, $R^{\text{BH}} = 100$ Mbps, $M = 10$ ms, $V = 10$ dB, $C = 50.0$ GB, $S_{\min} = 1.0$ GB, and $\Delta S = 9.0$ GB.

In Figure 5.9, we show the average delay (left) and the hit ratio (right) versus the parameter q. As a reference, we include the result of IGA for the same setup, which is independent of parameter q. We emphasize that, although IGA may be unfeasible, its delay saving is not farther than $(1 - 1/e)$ from the optimal. As we observe in Figure 5.9 (left), qLRU-HS gets closer to IGA as q decreases, suggesting its convergence to the optimal allocation. In addition to qLRU-HS results, we also plot the results for qLRU-Δd, that is also guaranteed to converge to the minimum delay as q vanishes, but only when files have all the same size [64]. However, qLRU-Δd converges to a value of average delay larger than qLRU-HS's one. This is due to fact that qLRU-Δd, while trying to minimize the delay, tends to store large files, that indeed incur large transmission delay, ignoring that they also occupy a large amount of space in the cache. In particular, given two files f_1 and f_2 with $\lambda_{f_1} > \lambda_{f_2}$ and $S_{f_2} \gg S_{f_1}$, qLRU-Δd would prefer f_2, while our caching policy qLRU-HS correctly bias its choices in favor of f_1 that leads to a larger benefit for byte occupied in the cache. From Figure 5.9 (right), we see that, for this particular scenario, better average delay is associated with a better hit ratio, which is not always necessarily the case.

In Figure 5.10, we show the average delay (left) and the hit ratio (right) versus the number of requests in the simulation. For this plot, we simulate qLRU-HS and qLRU-Δd for $q = 10^{-3}$ and $q = 10^{-4}$, and we indicate the results of IGA as reference. As we observe in Figure 5.10 (left), the average delay achieved by each policy decreases over time, and reaches its minimum value after about 10^6 requests (10^5 requests per BS).

Observation 13: For sufficiently small q, qLRU-HS shows to be sensitive to different files sizes in practice, achieving delays close to IGA and outperforming qLRU-Δd.

5.4.2 Comparison with other Caching Policies

Now, we compare the performance of qLRU-HS with other caching solutions in different scenarios. From now on, we consider $q = 10^{-3}$ for qLRU-HS and qLRU-Δd.

In Figure 5.11, we show the performance for different values of caching capacity, ranging from $C = 10$ GB to $C = 100$ TB. We present the average delay (left) and the hit ratio (right) versus the cache capacity size C. qLRU-HS provides a more efficient management of the cache, outperforming all other policies and presenting results close to the IGA ones. The difference of performance across policies is maximal for smaller values of C. In particular, for $C = 10$ GB, qLRU-HS achieves a delay about 20% smaller than GDSize-All. As expected, when the capacity increases, all policies perform better because they can store more files and also differences reduce until all policies perform equally when the cache is so large to be able to store the whole catalog.

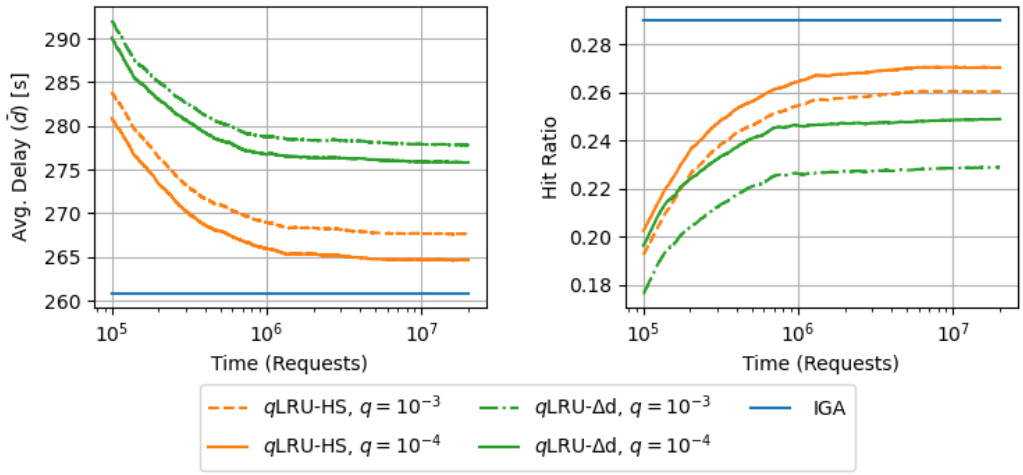


Figure 5.10 – Convergence Analysis: Average delay $\bar{d}$ (left) and hit ratio (right) versus the requests. Results of qLRU-HS and qLRU-Δd are shown for $q = 10^{-3}$ and $q = 10^{-4}$. Setup: Berlin topology with density $\rho = 5.9$, $R^{\text{BH}} = 100$ Mbps, $M = 10$ ms, $V = 10$ dB, $C = 50.0$ GB, $S_{\min} = 1.0$ GB, and $\Delta S = 9.0$ GB.

In Figure 5.12, we fix the cache capacity to $C = 30$ GB and observe the policies' performances for different levels of density, from $\rho = 1.4$ BSs/UE to $\rho = 9.1$ BSs/UE. We control the network density by simply increasing the BSs' transmission range, although we keep constant the SNR to $V = 10$ dB. In this scenario, qLRU-HS again outperforms all other policies and has results close to the IGA ones.

We observe in Figure 5.12 (left) that all policies experience a delay reduction as ρ increases. The reason is that the aggregate cache available to each UE gets larger with ρ, then more files are found in the neighboring BSs. Because of the larger aggregate cache, the difference between qLRU-HS and qLRU-Δ becomes slightly smaller as ρ increases (similarly to Figure 5.11). On the contrary, the performance gap with GDSize-All increases: the fact that all BSs in I_u react to a request from u leads to poor coordination.

Observation 14: For different aggregate storage capacities and network densities, qLRU-HS presents results close to IGA and outperforms other caching policies from the literature that promise to handle either sizes heterogeneity or cases with multiple caches.

Figure 5.13 shows the average delay $\bar{d}_{\text{P}}$ achieved by policy P normalized by the average delay $\bar{d}_{\text{IGA}}$ achieved by IGA. Results are presented for different size variability (captured by the parameter ΔS), on the left, and backhaul latency M, on the right. For these experiments, we fix the network density to $\rho = 5.9$ BSs/UE. We chose to show the results in a normalized fashion due to the large excursion of $\bar{d}_{\text{P}}$ values when both ΔS and M change.

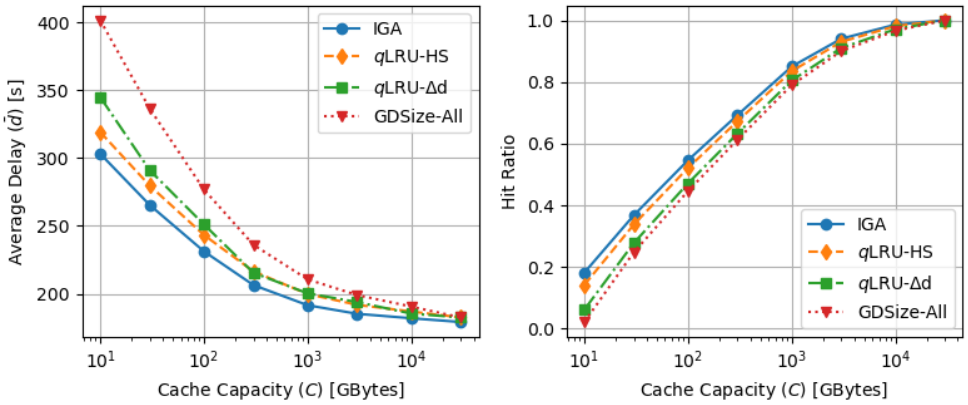


Figure 5.11 – Performance Evaluation: Average delay (left) and hit ratio (right) achieved by various policies versus increasing cache capacity. Setup: Berlin topology with density $\rho = 5.9$, $R^{\mathrm{BH}} = 100$ Mbps, $M = 10$ ms, $V = 10$ dB, $q = 10^{-3}$, $S_{\min} = 1.0$ GB, and $\Delta S = 9.0$ GB.

In Figure 5.13 (left) we evaluate $\bar{d}_{\mathrm{P}}/\bar{d}_{\mathrm{IGA}}$ for fixed $S_{\min} = 1$ GB and change the ΔS from $\Delta S = 0$ (homogeneous file sizes) to $\Delta S = 49$ GB. We first observe that qLRU-HS and qLRU-Δ both have results close to IGA in the homogeneous size case. The more heterogeneous is the catalog, in terms of size, the noisier is the convergence process. This happen because, at every cache miss, the number of insertions is not proportional to the number of evictions, leading to "asymmetric" cache updates. For example, (i) the insertion of a single large file can lead to the eviction of many other files and (ii) the insertion of a small file may cause the eviction of a large file, producing unused storage space at the cache. This fact explains why the relative performance of all dynamic policies worsens when size variability increases. Despite the increasing trend shared by all policies, we observe that qLRU-HS is always the closest to IGA. Interestingly, although GDSize-All has the worst performance, it is less sensitive to the variability of file sizes.

Observation 15: Higher size variability implies a noisier convergence process, causing the online policies to perform worse in practice.

Finally, one interesting aspect in our model is how the backhaul latency constant affects the policies operation and results. In Figure 5.13 (right), we show $\bar{d}_{\mathrm{P}}/\bar{d}_{\mathrm{IGA}}$ when the backhaul latency increases from $M = 30$ ms to $M = 1$ s. In this case, we fixed $S_{\min} = 1$ GB and the size variability to $\Delta S = 9.0$ GB. In this experiment, we also observe dynamic policies perform worse in comparison to IGA as the backhaul latency M increases. When M becomes larger, the optimal caching strategy changes from a scenario where it is convenient to store more copies of the same files across the BSs' caches (to

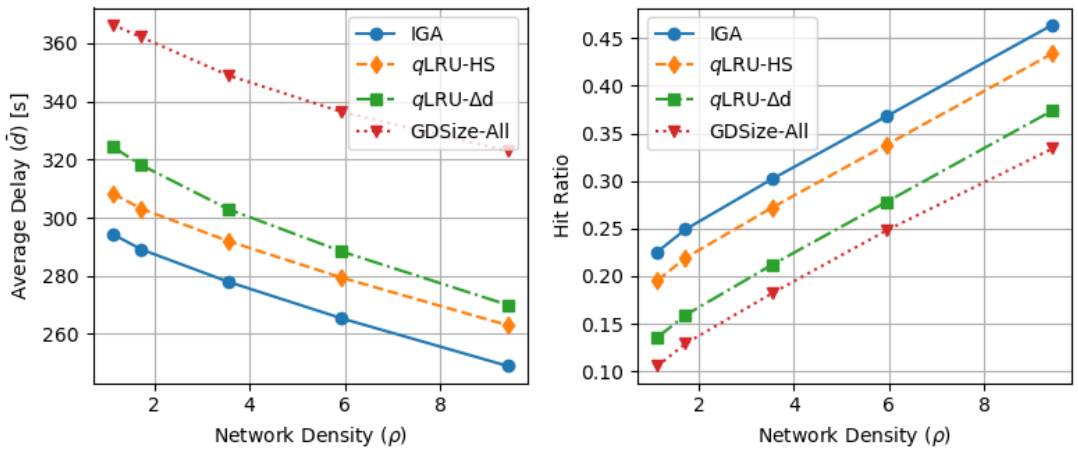


Figure 5.12 – Performance Evaluation: Average delay (left) and hit ratio (right) achieved by various policies versus increasing network density. Setup: Berlin topology with $R^{\mathrm{BH}} = 100$ Mbps, $M = 10$ ms, $V = 10$ dB, $q = 10^{-3}$, $C = 30.0$ GB, $S_{\min} = 1.0$ GB, and $\Delta S = 9.0$ GB.

create CoMP opportunities) to a scenario where file diversity across caches is preferred because it minimizes cache misses that cause the largest delay. This means that, for large enough values of backhaul latency, qLRU-Δ and qLRU-HS take an equivalent strategy, to diversify files throughout the network of caches. However, qLRU-Δ still erroneously prefer to store large files. This leads to qLRU-Δ storing on average less files, which decreases the hit probability and, in turn, worsens qLRU-Δ's performance. On the contrary, GDSize-All correctly prefer the smallest files, but, as all caches react at the same time, BSs tend to have similar cache content. This replication of files throughout the BSs is suboptimal for high latency, which explains GDSize-All's worse performance.

Observation 16: In scenarios with high miss cost, e.g., high backhaul access overhead, qLRU-HS has shown to find a good balance between files sizes and popularities in comparison with other policies.

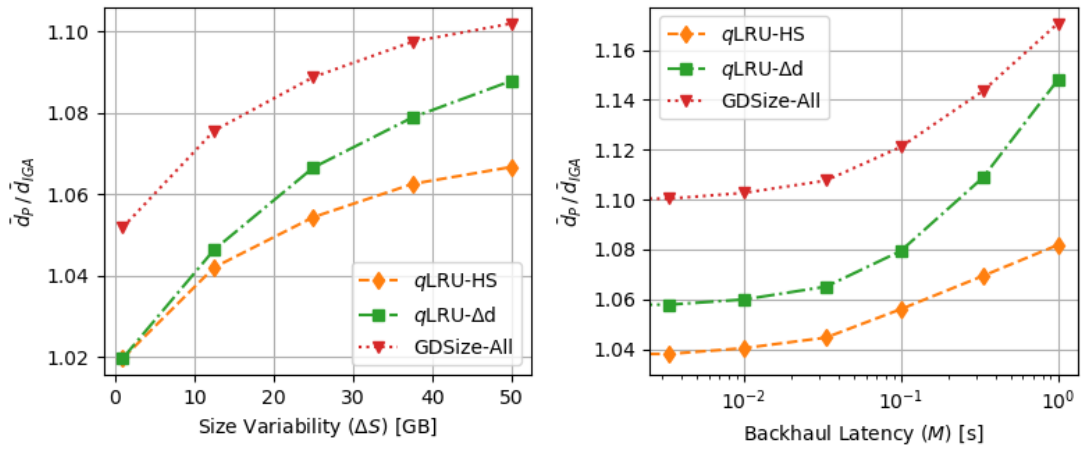


Figure 5.13 – Performance Evaluation: The ratio between the average delay achieved by various policies and IGA versus size variability (left) and backhaul-access overhead. Setup: Berlin topology with density $\rho = 5.9$ BSs/UE, $R^{\mathrm{BH}} = 100$ Mbps, $V = 10$ dB, $q = 10^{-3}$, $C = 30.0$ GB, and $S_{\min} = 1.0$ GB.

Chapter 6

Conclusion

In this thesis we proposed static and dynamic caching solutions for both hit ratio maximization and average delay minimization problems in CoMP-aided cache-enabled small-cell (CCSC) networks. We formulated a general static optimization problem that can be adapted to different metrics. Considering the delay minimization in CCSC networks, we first provided insightful theoretical results on the problem's solution using a simple scenario that we called full-coverage, where all UEs can communicate with all BSs and their respective SNRs are homogeneous.

Then, we discussed that, if we consider a general topology while still assuming homogeneous SNRs, the problem becomes hard to solve but a greedy algorithm is able to provide solutions with a $\frac{1}{2}$-approximation guarantee. Unfortunately, we concluded that the same guarantee does not hold for the general case where SNRs are heterogeneous. Finally, we discussed about the heterogeneous file sizes case and suggested a static solution based on greedy algorithm that enjoys an approximation guarantee. Although the provided solution is potentially infeasible, it may still be used as a comparison baseline in our experiments.

Then, we moved to the dynamic framework and started to discuss caching solutions based on online policies. We started assuming a stationary request process based on IRM and proposed the general-purpose caching policy qLRU-Δ. We proved that, under CTA and EA, a network of qLRU-Δ caches converge to an optimal allocation when parameter q tends to 0. We study two different cases where qLRU-Δ may be adapted to solve either the hit ratio maximization problem or the average delay minimization problem. The policy qLRU-Δ has a simple implementation, is trivially enabled by current mobile networks, and demands limited information on the local cache neighborhood.

Nevertheless, the reduced insertion rate caused by small values of q makes qLRU-Δ less reactive, and then less suitable to systems with realistic request processes where files

popularities are subject to strong temporal locality. In order to handle this scenario, we proposed 2LRU-Δ policy, which may also be adapted to different performance metrics. Although 2LRU-Δ does not provide optimality guarantees, it is more reactive than qLRU-Δ and promises to have better performance in practice, as we confirmed later in our experiments. Our last contribution regarding dynamic solutions was a novel caching policy, qLRU-HS, that adapts qLRU-Δ to the heterogeneous file sizes case. We proved that a network of qLRU-HS caches asymptotically converges to an optimal allocation if q tends to 0.

In the last part, we investigated our proposed algorithms and policies' performance experimentally through numerical simulations. First, we focused on observing qLRU-Δ's and its variants' convergence to an optimal allocation under stationary request processes in practice. We gradually reduced the value of q and were able to verify the convergence, not only in terms of the measured hit ratio and average delay, but also in terms of the decreasing distance between the resulting cache allocations and the optimal ones (provided by greedy algorithms). This optimal trend was also observed in different experimental setups for sufficiently small q. It is also noteworthy that qLRU-Δ achieved better performance in comparison with other state-of-the-art policies. This was also the case for qLRU-HS, when the files have different sizes. In the end, under a non-stationary request process based on Akamai's trace, 2LRU-Δ outperforms other policies in all tested scenarios.

We conclude that our proposed algorithms and policies provide desirable performance in practice and constitute a set of very simple and versatile techniques. For these reasons, we believe they can be attractive to many applications and even be considered potential candidates as de facto caching solutions for CDNs and future mobile network architectures. In the future, we consider investigating an extension of our dynamic policies in which caching and routing decisions are jointly designed to provide better QoS. In other words, our policies would delineate rules to decide which files to store at which cache as well as *how* the request should be served, still considering collaborative transmissions as a potential performance booster.

Appendices

Appendix A

Proofs for Chapter 3

A.1 Proof of Proposition 1

Proposition 1: In the full-coverage scenario, an allocation provided by GreedyAD is optimal.

Proof. For every file $f \in [F]$, we generate B objects $f^{(1)}, \ldots, f^{(B)}$ with weight $w(f^{(k)}) = \lambda_f(d(k-1) - d(k)) > 0$. In fact, each object $f^{(k)}$ represents the k-th copy of file f that is cached among the BSs. We gather in $\mathcal{F} = \{f^{(1)}, \ldots, f^{(B)}, \forall f \in [F]\}$ all objects generated this way. 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 Algorithm 11.

Consider the problem:

$$\begin{aligned} \underset{\mathcal{A} \subset \mathcal{F}}{\text{maximize}} \quad & w(\mathcal{A}), \\ \text{subject to} \quad & |\mathcal{A}| = B \cdot C. \end{aligned} \tag{A.1}$$

This is a weight maximization problem, so a greedy algorithm finds the optimal solution $\mathcal{A}^*$ if, and only if, the constraints form a matroid (see [87]). $\mathcal{A}^*$ 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 $\mathcal{A}' = \mathcal{A}^* \setminus \{f^{(k)}\} \cup \{f^{(h)}\}$, 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^*

Algorithm 11: 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
        end
    end
end
```

with $\bar{s}(X^*) = w(\mathcal{A}^*)$. We claim that X^* is an optimal solution of Problem 5. 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 5, 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 (A.1). It follows that GreedyAD provides an optimal solution for Problem 5. □

A.2 Proof of Lemma 2

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

Proof. 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 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. □

A.3 Proof of Proposition 3

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

$$\lambda_1 \cdot (d(1) - d(2)) \leq \lambda_{B \cdot C} \cdot (d(0) - d(1)),$$

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

$$\lambda_{C+1} \cdot (d(0) - d(1)) \leq \lambda_C \cdot (d(B-1) - d(B)).$$

Proof. As provided by Lemma 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 if and only if:

$$\lambda_i \cdot d^{\mathrm{BH}} \geq \lambda_j \cdot (d(1) - d(2)), \forall i \in [B \cdot C], j \in [B \cdot C] \setminus \{i\},$$

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 if and only if

$$\lambda_{B \cdot C} \cdot d^{\mathrm{BH}} \geq \lambda_1 \cdot (d(1) - d(2)),$$

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.,

$$\lambda_{C+1} \cdot d^{\mathrm{BH}} \leq \lambda_C \cdot (d(B-1) - d(B)).$$

□

A.4 Proof of Corollary 4

Corollary 4: For general network topologies, assuming homogeneous SNRs, the following conditions hold: (i) Inequality (3.16) is a **necessary condition** for the full-diversity allocation to be locally optimal, and (ii) inequality (3.17) is a **sufficient condition** for the full-replication allocation to be locally optimal.

Proof. We prove each part of the corollary separately:

First, we want to show that, for general topologies, if full-diversity is locally optimal, then (3.16) 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:

$$\bar{d}(X) \leq \bar{d}(X') \Leftrightarrow \sum_{u \in [U]} \frac{1}{U} \sum_{f \in [F]} \lambda_f \cdot d(k_{u,f}) \leq \sum_{u \in [U]} \frac{1}{U} \sum_{f \in [F]} \lambda_f \cdot d(k'_{u,f}).$$

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 cancelled out. Then, we can write:

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

$$\begin{aligned} &\Leftrightarrow \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_2} \cdot \left(d(k_{u,f_2}) - d(k'_{u,f_2})\right) \\ &\quad \leq \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_1} \cdot \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right). \end{aligned} \tag{A.2}$$

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

$$\lambda_{f_2} \cdot (d(1) - d(2)) \leq \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_2} \cdot \left(d(k_{u,f_2}) - d(k'_{u,f_2})\right). \tag{A.3}$$

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

$$\frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_1} \cdot \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right) = \lambda_{f_1} \cdot (d(0) - d(1)). \tag{A.4}$$

Putting together (A.3) and (A.4) with (A.2), we obtain:

$$\begin{aligned} \lambda_{f_2} \cdot (d(1) - d(2)) &\leq \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_2} \cdot \left(d(k_{u,f_2}) - d(k'_{u,f_2})\right) \\ &\leq \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_1} \cdot \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right) \\ &= \lambda_{f_1} \cdot (d(0) - d(1)). \end{aligned} \tag{A.5}$$

Then, we have that:

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

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

$$\lambda_1 (d(1) - d(2)) \leq \lambda_{B \cdot C} \cdot (d(0) - d(1)).$$

Therefore, if full-diversity is locally optimal, then (3.16) holds.

Second, we want to show that, for general topologies, if (3.17) holds, then full-replication is locally optimal. Equivalently, we prove that if full-replication is not locally optimal, then (3.17) 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:

$$\begin{aligned} &\frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{C+1} \cdot \left(d(k_{u,C+1}) - d(k'_{u,C+1}\right) \\ &\qquad > \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_1} \cdot \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right). \end{aligned} \tag{A.6}$$

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

$$\begin{aligned} &\lambda_{C+1} \left(d(0) - d(1)\right) = \\ &\quad = \frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{C+1} \cdot \left(d(k_{u,C+1}) - d(k'_{u,C+1}\right). \end{aligned} \tag{A.7}$$

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

$$\begin{aligned} &\frac{1}{|\mathcal{U}(b)|} \sum_{u \in \mathcal{U}(b)} \lambda_{f_1} \cdot \left(d(k'_{u,f_1}) - d(k_{u,f_1})\right) = \\ &\qquad = \lambda_{f_1} \cdot (d(B-1) - d(B)). \end{aligned} \tag{A.8}$$

Putting together (A.7) and (A.8) with (A.6), we obtain:

$$\lambda_{C+1} \cdot (d(0) - d(1)) > \lambda_{f_1} \cdot (d(B-1) - d(B)) \geq \lambda_C \cdot (d(B-1) - d(B))$$

that contradicts (3.17).

Therefore, if (3.17) holds, then full-replication is locally optimal. □

A.5 Proof of Proposition 5

Proposition 5: Problem 6 is NP-Hard in the homogeneous SNR regime.

Proof. We want to show that the FemtoCaching problem [69] can be reduced to Problem 6. The Homogeneous-SNR version of the FemtoCaching problem has the following objective:

$$\underset{X \subseteq \Omega}{\text{minimize}}\ \bar{d}_F(X) = \frac{1}{U} \sum_{u,f} \lambda_f \cdot \left(\mathbb{1}(J_{u,f}(X) = \emptyset) \cdot d^{\text{BH}} + t(1)\right), \tag{A.9}$$

that is subject to the capacity constraints (3.1), where $\bar{d}_F(X)$ is the average delay for the FemtoCaching problem and $t(1) = \frac{S}{*}W \cdot \log_2(1+V)$ is given by (3.18) 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.,

$$\lambda_f = \frac{m_f}{n}, m_f, n \in \mathbb{N}, \forall f \in [F]. \tag{A.10}$$

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

$$\begin{aligned}
&\left|\bar{d}_F(X) - \bar{d}_F(X')\right| = \\
&= \left|\frac{1}{U}\sum_{u,f} \lambda_f \cdot (\mathbb{1}(J_{u,f}(X)) \cdot d^{\mathrm{BH}} + t(1)) - \frac{1}{U}\sum_{u,f} \lambda_f \cdot (\mathbb{1}(J_{u,f}(X')) \cdot d^{\mathrm{BH}} + t(1))\right| && \text{(A.11)}\\
&= \frac{1}{n \cdot U}\left|\sum_{u,f} m_f \cdot ((\mathbb{1}(J_{u,f}(X)) \cdot d^{\mathrm{BH}} + t(1)) - (\mathbb{1}(J_{u,f}(X')) \cdot d^{\mathrm{BH}} + t(1)))\right| && \text{(A.12)}\\
&= \frac{d^{\mathrm{BH}}}{n \cdot U}\left|\sum_{u,f} m_f \cdot \left(\mathbb{1}(J_{u,f}(X)) - \mathbb{1}(J_{u,f}(X'))\right)\right| && \text{(A.13)}\\
&\geq \frac{d^{\mathrm{BH}}}{n \cdot U}. && \text{(A.14)}
\end{aligned}$$

Equality (A.11) is the direct application of the FemtoCaching problem's objective (A.9). We use the popularities' rational notation in (A.10) to derive (A.12). The absolute part of (A.13) always results in a positive integer, which leads to inequality (A.14).

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

$$\frac{d^{\mathrm{BH}}}{n \cdot U} > t(1). \tag{A.15}$$

Then, Problem 6's objective can be written as:

$$\begin{aligned}
\bar{d}(X) &= \frac{1}{U}\sum_{u,f} \lambda_f \cdot d_u(|J_{u,f}(X)|) && \text{(A.16)}\\
&= \frac{1}{U}\sum_{u,f} \lambda_f \cdot (\mathbb{1}(J_{u,f}(X) = \emptyset) \cdot d_u(0) + \mathbb{1}(J_{u,f}(X) \neq \emptyset) \cdot d_u(|J_{u,f}(X)|)) && \text{(A.17)}\\
&= \frac{1}{U}\sum_{u,f} \lambda_f \cdot \big(\mathbb{1}(J_{u,f}(X) = \emptyset) \cdot (d^{\mathrm{BH}} + t(1)) + \mathbb{1}(J_{u,f}(X) \neq \emptyset) \cdot t(1)
\end{aligned}$$

$$-\mathbb{1}(J_{u,f}(X) \neq \emptyset) \cdot t(1) + \mathbb{1}(J_{u,f}(X) \neq \emptyset) \cdot d_u(|J_{u,f}(X)|)) \tag{A.18}$$

$$= \frac{1}{U} \sum_{u,f} \lambda_f \cdot \big(\mathbb{1}(J_{u,f}(X) = \emptyset) \cdot d^{\mathrm{BH}} + t(1)\big)$$

$$- \frac{1}{U} \sum_{u,f} \lambda_f \cdot \mathbb{1}(J_{u,f}(X) \neq \emptyset) \, (t(1) - d_u(|J_{u,f}(X)|)) \tag{A.19}$$

$$= \bar{d}_F(X) - \frac{1}{U} \sum_{u,f} \lambda_f \cdot \mathbb{1}(J_{u,f}(X) \neq \emptyset) \cdot (t(1) - d_u(|J_{u,f}(X)|)). \tag{A.20}$$

Equality (A.16) is an adaptation of Problem 6's objective to the homogeneous SNR regime, where we replace the general delay function (3.3) with (3.19). Equation (A.17) comes from the fact that, if $J_{u,f}(X) = \emptyset$ (i.e., cache miss), the delay is $d_u(0) = d^{\mathrm{BH}} + t(1)$ 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 (3.19) covers both miss and hit cases. We obtain (A.19) by simply putting the indicator functions from (A.18) in evidence. Finally, we observe that the first term in (A.19) is exactly the definition of the FemtoCaching objective function, which yields (A.20).

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

$$\bar{d}_F(X) \geq \bar{d}(X) \geq \bar{d}_F(X) - t(1). \tag{A.21}$$

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

$$\bar{d}_F(X) < \bar{d}_F(X') \Leftrightarrow \bar{d}(X) < \bar{d}(X') \tag{A.22}$$

and then solving Problem 6 brings the solution to the Homogeneous-SNR FemtoCaching Problem.

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

Hypothesis 1: $\bar{d}_F(X) < \bar{d}_F(X')$

$$\begin{aligned}
\bar{d}(X) &\leq \bar{d}_F(X) && \text{by (A.21)} \\
&\leq \bar{d}_F(X') - \frac{d^{\mathrm{BH}}}{n \cdot U} && \text{by Hyp. 1 and (A.14)} \\
&< \bar{d}_F(X') - t(1) && \text{by (A.15)} \\
&\leq \bar{d}(X') && \text{by (A.21)}
\end{aligned}$$

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

$$\begin{aligned} \text{Hypothesis 2:} \quad & \bar{d}(X) < \bar{d}(X') \\ & \bar{d}_F(X') \geq \bar{d}(X') \\ & > \bar{d}(X) - t(1) \\ & \geq \bar{d}_F(X) \qquad \text{by (A.21)} \end{aligned}$$

Then, because both implications hold, (A.22) also holds and, therefore, the Homogeneous-SNR FemtoCaching Problem can be reduced to Problem 6. Because the Homogeneous-SNR FemtoCaching Problem is NP-hard [69] and we can reduce it to Problem 6, then Problem 6 is NP-hard. □

A.6 Proof of Lemma 6

The following lemma will assist in the proof Lemma 6:

Lemma 19: For any u and any $k_1, k_2 \in \mathbb{Z}_+$, such that $k_1 \leq k_2$, the following inequality holds:

$$t_u(k_1) - t_u(k_1 + 1) \geq t_u(k_2) - t_u(k_2 + 1). \tag{A.23}$$

Proof. Let $h(x) = \frac{S}{W \log_2(1+g \cdot x)}$. Function h can be written as a function composition $h(x) = (w \circ y)(x)$, where $w(x) = \frac{S}{W \log_2(x)} = \frac{S \ln(2)}{W} \cdot \left(\frac{1}{\ln(x)}\right)$ and $y(x) = 1 + g \cdot x$. Function y is affine. Function w first and second derivatives are, respectively, $w'(x) = \frac{S \ln(2)}{W} \cdot \left(\frac{-1}{x \ln^2(x)}\right)$ and $w''(x) = \frac{S \ln(2)}{W} \cdot \left(\frac{\ln(x)+2}{x^2 \ln^3(x)}\right)$. 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 $\frac{S}{W \log_2(1+|I_u| \cdot g)}$, 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 (A.23) holds. □

Now, we remind Lemma 6 below and proceed with its proof:

Lemma 6: The objective function in Equation (3.21) is *monotone* and *submodular*.

Proof. We separate the proof of Lemma 6 in two parts, one for each property of $\bar{s}(X)$.

Monotonicity: Let $X \subset X' \subset \Omega$ 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 (3.21) is *monotone* if:

$$\begin{aligned}\bar{s}(X) \leq \bar{s}(X') &\Leftrightarrow \bar{d}(X) \geq \bar{d}(X') \\ &\Leftrightarrow \sum_{u,f} \lambda_f \cdot d_u(|J_{u,f}(X)|) \geq \sum_{u,f} \lambda_f \cdot d_u(|J_{u,f}(X')|)\end{aligned} \tag{A.24}$$

We observe that $\forall f \neq f'$, the LHS equals the RHS in (A.24), so we focus on cases where $f = f'$. Similarly, we consider only the set of UEs covered by BS b', that we call $\mathcal{U}(b')$. Then, (A.24) becomes:

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

Notice that $t_u(k)$ is non-increasing, which makes $d_u(k)$ non-increasing as well (see (3.19)). Then, $d_u(|J_{u,f'}(X)|) - d_u(|J_{u,f'}(X)|+1) \geq 0$ and, therefore, (3.21) is monotone.

Submodularity: Let $X \subset X' \subset \Omega$ and $(b', f') \in \Omega \backslash X'$. The set function (3.21) is *submodular* if:

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

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:

$$\begin{aligned}&\lambda_{f'} \sum_{u \in \mathcal{U}(b')} d_u(|J_{u,f'}(X)|) - d_u(|J_{u,f'}(X)|+1) \\ &\geq \lambda_{f'} \sum_{u \in \mathcal{U}(b')} d_u(|J_{u,f'}(X')|) - d_u(|J_{u,f'}(X')|+1).\end{aligned} \tag{A.25}$$

We will prove (A.25) by showing that for each u, it holds that

$$d_u(|J_{u,f'}(X)|) - d_u(|J_{u,f'}(X)|+1) \geq d_u(|J_{u,f'}(X')|) - d_u(|J_{u,f'}(X')|+1)$$

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 $\forall u \in \mathcal{U}(b')$,

$$d_u(k) - d_u(k+1) \geq d_u(k+1) - d_u(k+2). \tag{A.26}$$

However, the delay d_u is the minimum of two functions (see (3.19)). We observe that $d_u(k) = t_u(k) \Rightarrow d_u(k+1) = t_u(k+1)$. In fact,

$$\begin{aligned}
&d_u(k) = t_u(k) \Rightarrow \\
&\quad \Rightarrow t_u(k) \leq d^{\mathrm{BH}} + t_u(k+1) \Rightarrow \\
&\quad \Rightarrow t_u(k) - t_u(k+1) \leq d^{\mathrm{BH}} \Rightarrow \\
&\quad \Rightarrow t_u(k+1) - t_u(k+2) \leq d^{\mathrm{BH}} \Rightarrow \qquad \text{(by Lemma 19)} \\
&\quad \Rightarrow d_u(k+1) = t_u(k+1).
\end{aligned}$$

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, (A.26) 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 19.

Case (II): $d_u(k) = d^{\mathrm{BH}} + t_u(k+1)$, $d_u(k+1) = t_u(k+1)$, and $d_u(k+2) = t_u(k+2)$. Then, (A.26) is written as:

$$\begin{aligned}
d^{\mathrm{BH}} + t_u(k+1) - t_u(k+1) &\geq t_u(k+1) - t_u(k+2) \\
d^{\mathrm{BH}} &\geq t_u(k+1) - t_u(k+2),
\end{aligned}$$

which is true as $d_u(k+1) = t_u(k+1)$ and then $t_u(k+1) \leq d^{\mathrm{BH}} + t_u(k+2)$.

Case (III): $d_u(k) = d^{\mathrm{BH}} + t_u(k+1)$, $d_u(k+1) = d^{\mathrm{BH}} + t_u(k+2)$, and $d_u(k+2) = t_u(k+2)$. Then, (A.26) becomes:

$$\begin{aligned}
&d^{\mathrm{BH}} + t_u(k+1) - d^{\mathrm{BH}} + t_u(k+2) \\
&\qquad \geq d^{\mathrm{BH}} + t_u(k+2) - t_u(k+2) \Leftrightarrow \\
&\Leftrightarrow d^{\mathrm{BH}} \leq t_u(k+1) - t_u(k+2),
\end{aligned}$$

which is true as $d_u(k+1) = d^{\mathrm{BH}} + t_u(k+2)$ and then $t_u(k+1) > d^{\mathrm{BH}} + t_u(k+2)$.

$$d_u(k+1) = d^{\mathrm{BH}} + t_u(k+2) \Leftrightarrow t_u(k+1) > d^{\mathrm{BH}} + t_u(k+2).$$

Case (IV): $d_u(k) = d^{\mathrm{BH}} + t_u(k+1)$, $d_u(k+1) = d^{\mathrm{BH}} + t_u(k+2)$, $d_u(k+2) = d^{\mathrm{BH}} + t_u(k+3)$. This case is analogous to Case (I). □

A.7 Proof of Proposition 9

Proposition 9: Problem 7 in the full-coverage setup is equivalent to SMKP.

Proof. Consider an instance of SMKP where $\Omega = \{f^{(k)} : \forall (f,k) \in [F] \times [B]\}$ is the ground set. The abstract element $f^{(k)}$ represents the k-th copy of file f in the cache network. We consider a set of bins (knapsacks) $[B]$, where each bin has capacity C. We represent the solution set by $X \subseteq \Omega$, which is partitioned according to the set of bins, i.e., $X = \left(X^{(1)}, \ldots, X^{(B)}\right)$. For any feasible solution X, its elements may be arbitrarily placed into the available bins as long as the knapsack capacity constraints are satisfied:

$$\sum_{f^{(k)} \in X^{(b)}} S_f \leq C, \forall b \in [B]. \tag{A.27}$$

Now, we define a profit function $d : \Omega \to \mathbb{R}_+$ as follows:

$$d(f^{(k)}) \triangleq \lambda_f \cdot \frac{1}{U} \sum_{u \in [U]} \left(d_{u,f}(k-1) - d_{u,f}(k)\right) = \lambda_f \cdot \left(d_f(k-1) - d_f(k)\right), \tag{A.28}$$

where we can just drop references to multiple UEs, such that $d_f(k) = d_{u,f}(k), \forall u \in [U]$ and consider a single UE due to the network symmetry.

Now we define the number of copies of file f in a given solution X as

$$k_{\max}^{(f)}(X) \triangleq \arg\max_k \{f^{(k)} : \forall f^{(k)} \in X\}, \tag{A.29}$$

where we consider $k_{\max}^{(f)}(X) = 0$ if there is no element associated to file f in X.

Then, the goal is to find a feasible solution X, i.e., satisfying knapsack capacity constraints, that maximizes the total profit:

$$D(X) \triangleq \sum_{f^{(k)} \in X} d(f^{(k)}) \tag{A.30}$$

$$= \sum_{f\in[F]} \sum_{k'=1}^{k^{(f)}_{\max}(X)} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} (d_{u,f}(k-1) - d_{u,f}(k)) \tag{A.31}$$

$$= \sum_{f\in[F]} \lambda_f \cdot \frac{1}{U} \sum_{u\in[U]} \left(d_{u,f}(0) - d_{u,f}(k^{(f)}_{\max}(X))\right) \tag{A.32}$$

$$= \sum_{f\in[F]} \lambda_f \cdot \left(d_f(0) - d_f(k^{(f)}_{\max}(X))\right) \tag{A.33}$$

$$= \sum_{f\in[F]} \lambda_f \cdot d_f(0) - \sum_{f\in[F]} \lambda_f \cdot d_f(k^{(f)}_{\max}(X)) \tag{A.34}$$

$$= \sum_{f\in[F]} \lambda_f \cdot d_f(0) - \sum_{f\in[F]} \lambda_f \cdot d_f(|J_{u,f}(X)|) \tag{A.35}$$

$$= \bar{d}^{(0)}_{\mathrm{HS}} - \bar{d}(X) \tag{A.36}$$

$$= \bar{s}_{\mathrm{HS}}(X) \tag{A.37}$$

We start replacing the definition of the profit function and pointing out that we can cover all elements in X by replacing the sum over the elements of X with a double sum, one for the files and another for the number of copies. Then, we use the telescopic sum to eliminate the sum over k', obtaining (A.32). We remove the reference to UEs (A.33), since it is a symmetric scenario and they can all be represented as a single UE. In (A.34), we split the equation for the case with 0 copies and the case with $k^{(f)}_{\max}(X)$ copies. In (A.35), we note that $k^{(f)}_{\max}(X)$ is actually equivalent to the number of copies of f available to the UE under allocation X, i.e., $|J_{u,f}(X)|$. We note that the second term of (A.35) actually characterizes the average delay over all (heterogeneous-size) files, as we adapt from Problem 5. Finally, in line (A.36). Therefore, solving this instance of SMKP is equivalent of maximizing the delay saving under the full-coverage scenario. □

Bibliography

[1] "MS Windows NT kernel description," https://xuri.me/2016/08/13/lru-and-lfu-cache-algorithms.html, accessed: 2021-06-01.

[2] M. Rabinovich and O. Spatscheck, *Web caching and replication*. Addison-Wesley Boston, USA, 2002, vol. 67.

[3] S. A. Przybylski, *Cache and memory hierarchy design: a performance directed approach*. Morgan Kaufmann, 1990.

[4] C. Kopparapu, *Load balancing servers, firewalls, and caches*. John Wiley & Sons, 2002.

[5] M. Hofmann and L. R. Beaumont, *Content networking: architecture, protocols, and practice*. Elsevier, 2005.

[6] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014. [Online]. Available: https://doi.org/10.1109/TIT.2014.2306938

[7] K. Shanmugam *et al.*, "Femtocaching: Wireless video content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[8] E. Leonardi and G. L. Torrisi, "Modeling least recently used caches with shot noise request processes," *SIAM Journal on Applied Mathematics*, vol. 77, no. 2, pp. 361–383, 2017.

[9] 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.

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

[11] 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.

[12] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. T. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: the dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, 2014.

[13] H. Zhang, N. Liu, X. Chu, K. Long, A.-H. Aghvami, and V. C. Leung, "Network slicing based 5g and future mobile networks: mobility, resource management, and challenges," *IEEE communications magazine*, vol. 55, no. 8, pp. 138–145, 2017.

[14] K. Poularakis, G. Iosifidis, A. Argyriou, and L. Tassiulas, "Video delivery over heterogeneous cellular networks: Optimizing cost and performance," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 1078–1086.

[15] C. Tsai and M. Moh, "Cache management for 5g cloud radio access networks," in *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, ser. IMCOM '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3164541.3164559

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

[17] 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.

[18] R. Irmer, H. Droste, P. Marsch, M. Grieger, G. Fettweis, S. Brueck, H.-P. Mayer, L. Thiele, and V. Jungnickel, "Coordinated multipoint: Concepts, performance, and field trial results," *IEEE Communications Magazine*, vol. 49, no. 2, pp. 102–111, 2011.

[19] V. Jungnickel, K. Manolakis, W. Zirwas, B. Panzner, V. Braun, M. Lossow, M. Sternad, R. Apelfröjd, and T. Svensson, "The role of small cells, coordinated multipoint, and massive mimo in 5g," *IEEE communications magazine*, vol. 52, no. 5, pp. 44–51, 2014.

[20] K. Manolakis, "Impairments in coordinated cellular networks: Analysis, impact on performance and mitigation," Ph.D. dissertation, TU Berlim, 11 2014.

[21] N. Golrezaei *et al.*, "Femtocaching: Wireless video content delivery through distributed caching helpers," in *IEEE INFOCOM 2012*, March 2012, pp. 1107–1115.

[22] 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

[23] 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.

[24] Z. Chen, J. Lee, T. Q. S. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 3401–3415, May 2017.

[25] K. Naveen *et al.*, "On the interaction between content caching and request assignment in cellular cache networks," in *5th Workshop on All Things Cellular: Oper., Applic, and Challenges*. ACM, 2015.

[26] Y. M. Saputra, H. T. Dinh, D. Nguyen, and E. Dutkiewicz, "A novel mobile edge network architecture with joint caching-delivering and horizontal cooperation," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2019.

[27] 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.

[28] Y. Cui and D. Jiang, "Analysis and optimization of caching and multicasting in large-scale cache-enabled heterogeneous wireless networks," *IEEE transactions on Wireless Communications*, vol. 16, no. 1, pp. 250–264, 2016.

[29] Z. Chen, J. Lee, T. Q. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 3401–3415, 2017.

[30] F. Guo, H. Zhang, X. Li, H. Ji, and V. C. Leung, "Joint optimization of caching and association in energy-harvesting-powered small-cell networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 7, pp. 6469–6480, 2018.

[31] H. Wu, H. Lu, F. Wu, and C. W. Chen, "Energy and delay optimization for cache-enabled dense small cell networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7663–7678, 2020.

[32] B. Dai and W. Yu, "Joint user association and content placement for cache-enabled wireless access networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 3521–3525.

[33] Y. Wang, X. Tao, X. Zhang, and G. Mao, "Joint caching placement and user association for minimizing user download delay," *Ieee Access*, vol. 4, pp. 8625–8633, 2016.

[34] W. Jing, X. Wen, Z. Lu, and H. Zhang, "User-centric delay-aware joint caching and user association optimization in cache-enabled wireless networks," *IEEE Access*, vol. 7, pp. 74 961–74 972, 2019.

[35] L. E. Chatzieleftheriou, G. Darzanos, M. Karaliopoulos, and I. Koutsopoulos, "Joint user association, content caching and recommendations in wireless edge networks," *ACM SIGMETRICS Performance Evaluation Review*, vol. 46, no. 3, pp. 12–17, 2019.

[36] 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.

[37] 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.

[38] A. Sabnis, T. S. Salem, G. Neglia, M. Garetto, E. Leonardi, and R. K. Sitaraman, "Grades: Gradient descent for similarity caching," in *IEEE Conference on Computer Communications (INFOCOM)*, 2021.

[39] B. Blaszczyszyn and A. Giovanidis, "Optimal geographic caching in cellular networks," in *IEEE ICC 2015*, June 2015, pp. 3358–3363.

[40] K. Avrachenkov, J. Goseling, and B. Serbetci, "A low-complexity approach to distributed cooperative caching with geographic constraints," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, pp. 27:1–27:25, Jun. 2017.

[41] T. M. Ayenew, D. Xenakis, N. Passas, and L. Merakos, "A novel content placement strategy for heterogeneous cellular networks with small cells," *IEEE Networking Letters*, vol. 2, no. 1, pp. 10–13, 2019.

[42] X. Sun, J. Zhang, and Z. Zhang, "Deterministic algorithms for the submodular multiple knapsack problem," 2020, arXiv:2003.11450.

[43] M. Garetto, E. Leonardi, and G. Neglia, "Similarity caching: Theory and algorithms," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 526–535.

[44] Q. Yu, L. Xu, and S. Cui, "Streaming algorithms for news and scientific literature recommendation: Monotone submodular maximization with a d-knapsack constraint," *IEEE Access*, vol. 6, pp. 53 736–53 747, 2018.

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

[46] 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.

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

[48] G. Neglia *et al.*, "Access-time aware cache algorithms," in *ITC-28*, September 2016.

[49] 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.

[50] G. Neglia, D. Carra, and P. Michiardi, "Cache policies for linear utility maximization," RR-9010, Inria, Tech. Rep., January 2017.

[51] ——, "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

[52] 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.

[53] 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.

[54] ——, "A unified approach to the performance analysis of caching systems," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 1, no. 3, pp. 1–28, 2016.

[55] A. Chattopadhyay, B. Błaszczyszyn, and H. P. Keeler, "Gibbsian on-line distributed content caching strategy for cellular networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 2, pp. 969–981, Feb 2018.

[56] 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.

[57] M. Garetto, E. Leonardi, and S. Traverso, "Efficient analysis of caching strategies under dynamic content popularity," in *IEEE INFOCOM 2015*, April 2015, pp. 2263–2271.

[58] S. Tarnoi, W. Kumwilaisak, V. Suppakitpaisarn, K. Fukuda, and Y. Ji, "Adaptive probabilistic caching technique for caching networks with dynamic content popularity," *Computer Communications*, vol. 139, pp. 1–15, 2019.

[59] P. Cao and S. Irani, "Cost-aware www proxy caching algorithms," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, ser. USITS'97. USA: USENIX Association, 1997, p. 18.

[60] G. Rossini, D. Rossi, M. Garetto, and E. Leonardi, "Multi-terabyte and multi-gbps information centric routers," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, 2014, pp. 181–189.

[61] E. Friedlander and V. Aggarwal, "Generalization of lru cache replacement policy with applications to video streaming," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 4, no. 3, pp. 1–22, 2019.

[62] M. Choi, A. No, M. Ji, and J. Kim, "Markov decision policies for dynamic video delivery in wireless caching networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 12, pp. 5705–5718, 2019.

[63] G. I. 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.

[64] G. I. Ricardo, A. Tuholukova, G. Neglia, and T. Spyropoulos, "Caching policies for delay minimization in small cell networks with coordinated multi-point joint transmissions," *IEEE/ACM Transactions on Networking*, 2021.

[65] G. I. Ricardo, G. Neglia, and T. Spyropoulos, "Caching heterogeneous size content in small cell networks with comp joint transmissions," *Technical Report*, 2021. [Online]. Available: http://www-sop.inria.fr/members/Guilherme.Iecker-Ricardo/papers/qlruhs.pdf

[66] G. Neglia, E. Leonardi, G. I. Ricardo, and T. Spyropoulos, "A Swiss Army Knife for Dynamic Caching in Small Cell Networks," 2021, to appear.

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

[68] 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.

[69] 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.

[70] A. Krause and D. Golovin, "Submodular function maximization." *Tractability*, vol. 3, pp. 71–104, 2014.

[71] 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.

[72] M. Chrobak, G. J. Woeginger, K. Makino, and H. Xu, "Caching is hard—even in the fault model," *Algorithmica*, vol. 63, no. 4, pp. 781–794, 2012.

[73] Q. Yu, E. L. Xu, and S. Cui, "Submodular maximization with multi-knapsack constraints and its applications in scientific literature recommendations," in *2016 IEEE global conference on signal and information processing (GlobalSIP)*. IEEE, 2016, pp. 1295–1299.

[74] P. R. Jelenkovic, "Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities," *The Annals of Applied Probability*, vol. 9, no. 2, pp. 430–464, 1999.

[75] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proceedings of the 24th International Teletraffic Congress*, 2012, p. 8.

[76] D. P. Connors and P. R. Kumar, "Balance of recurrence order in time-inhomogenous markov chains with application to simulated annealing," *Probability in the Engineering and Informational Sciences*, vol. 2, no. 2, pp. 157–184, 1988.

[77] ——, "Simulated annealing type markov chains and their order balance equations," *SIAM Journal on Control and Optimization*, vol. 27, no. 6, pp. 1440–1461, 1989.

[78] M. Desai, S. Kumar, and P. R. Kumar, "Quasi-Statically Cooled Markov Chains," *Probability in the Engineering and Informational Sciences*, vol. 8, no. 1, pp. 1–19, 1994.

[79] K. Mehlhorn and P. Sanders, *Algorithms and data structures: The basic toolbox*. Springer Science & Business Media, 2008.

[80] V. Anantharam and P. Tsoucas, "A proof of the markov chain tree theorem," *Statistics & Probability Letters*, vol. 8, no. 2, pp. 189 – 192, 1989.

[81] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, Mass: Athena Scientific, Sep. 1999.

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

[83] 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.

[84] 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.

[85] Shudong Jin and A. Bestavros, "Popularity-aware greedy dual-size web proxy caching algorithms," in *Proceedings 20th IEEE International Conference on Distributed Computing Systems*, 2000, pp. 254–261.

[86] T. Mihretu Ayenew, D. Xenakis, N. Passas, and L. Merakos, "A novel content placement strategy for heterogeneous cellular networks with small cells," *IEEE Networking Letters*, vol. 2, no. 1, pp. 10–13, 2020.

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