Coded Caching: Convex Optimization and Graph Theoretical Perspectives

by

Seyed Ali Saberali

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Electrical and Computer Engineering)

The University of British Columbia
(Vancouver)

October 2020

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Coded Caching: Convex Optimization and Graph Theoretical Perspectives

submitted by **Seyed Ali Saberali** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Electrical and Computer Engineering**.

Examining Committee:

Lutz Lampe, Electrical and Computer Engineering, University of British Columbia
Supervisor

Ian Blake, Electrical and Computer Engineering, University of British Columbia
Supervisory Committee Member

Stephanie van Willigenburg, Mathematics, University of British Columbia
University Examiner

Joel Friedman, Mathematics, University of British Columbia
University Examiner

Stark Draper, Electrical and Computer Engineering, University of Toronto
External Examiner

Additional Supervisory Committee Members:

Lele Wang, Electrical and Computer Engineering
Supervisory Committee Member

Abstract

Data communications is an essential building block of everyday life. A fundamental challenge in communications networks has always been the limited capacity of the links that transfer data from sources to destinations. A core technique to alleviate the load on the network links is to cache popular content in intermediate nodes between the sources and destinations to avoid redundant transmission of the same data. Although the concept of caching has been well studied in the context of computer networks, new settings are emerging in communication networks for which the conventional caching techniques are significantly inefficient and deliver far less than the full benefits that caching can provide. One such setting is that of networks in which caches are connected to the backbone communications system through broadcast links. In the recent years, substantial amount of work has been devoted to characterizing the fundamental limits of the gain of caching in such networks, and coming up with techniques to achieve those limits. At the center of these attempts has been the introduction of *coded caching*, a technique rooted in information theory that takes advantage of network coding to minimize the amount of information that needs to be communicated in the network.

This thesis is devoted to the development of coded caching techniques for three specific settings that are of significant practical interest. In particular, it adapts a convex optimization perspective to address the problem of caching in the presence of duplicate demands, and the problem of designing the optimal way to place the content in caches when different files are non-uniformly popular. The latter is a core problem in the caching literature, for which we derive the optimal placement scheme for certain settings of the problem. We further look into the problem of placement of files in caches without splitting them into sub-packets. We establish

a graph theoretical model to study this problem and explore the efficiency of coded caching under this constraint. We believe that this thesis provides fundamental insights into these problems and solutions for them.

Lay Summary

Communication networks have witnessed a tremendous increase in mobile data traffic over the last decade. Transmission of such large volumes of information from servers to the end users of the network is a challenging task, mainly due to the limited capacity of the network communications links. A viable solution for alleviating the traffic on the network links is the storage of popular content in network nodes close to the end users. While the conventional caching strategies result in a linear gain in reducing the network traffic as the cache capacities increase, network coding inspired techniques considerably increase such gains and more effectively reduce the traffic. In this thesis, we propose such coding techniques and optimize them for use in more practical scenarios. We further design variations of coded caching techniques that are easier to implement and yet effective.

Preface

This thesis presents the original work and contributions I conducted during my Ph.D. research under the supervision of Professor Lutz Lampe in the Department of Electrical and Computer Engineering of the University of British Columbia. Throughout my work, I always benefited from the technical discussions and feedback from Professor Ian Blake in the Department of Electrical and Computer Engineering of the University of British Columbia.

The contributions of this thesis have been published in several peer-reviewed articles. In particular, a version of Chapter 3 appears in

S. A. Saberali, H. Ebrahimzadeh Saffar, L. Lampe and I. Blake, "Adaptive Delivery in Caching Networks" in *IEEE Communications Letters*, vol. 20, no. 7, pp. 1405-1408, July 2016.

I am responsible for all contributions in Chapter 3. This includes the review of the literature, identifying and formulation of the problem, development of the solution, doing the computer simulations and comparing the results to the results existing in the literature. For this work, I benefited from technical discussions with my co-author, Dr. Hamidreza Ebrahimzadeh Saffar, who was at the time a Post Doctoral Fellow in our research group. He brought to my attention the works in the literature that look at the caching problem from a network coding perspective, a perspective from which I conducted my research in this thesis. Professor Lutz Lampe and Professor Ian Blake supervised this work through technical discussions and the review of the manuscript.

The work in Chapter 4 was published in

S. A. Saberali, L. Lampe and I. F. Blake, "Full Characterization of Optimal Uncoded Placement for the Structured Clique Cover Delivery of Nonuniform De-

mands," in *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 633-648, Jan. 2020.

I led the research for all the contributions in Chapter 4, including literature review, identification of the problem, formulation of the problem as an optimization problem, solving the problem and in particular linking the problem to the mathematical concepts related to the submodular set functions, and preparation of the publication manuscript. Professor Lutz Lampe and Professor Ian Blake supervised this work through technical discussions and the review of the manuscript.

Finally, the contributions presented in Chapter 5 have been published in
S. A. Saberali, L. Lampe and I. F. Blake, "Decentralized Coded Caching Without File Splitting," in IEEE Transactions on Wireless Communications, vol. 18, no. 2, pp. 1289-1303, Feb. 2019.

I am responsible for all the contributions in Chapter 5, including the review of the literature, identification of the problem, formulation of the problem, coming up with the mathematical tools to address the problem and solving it, performing all the simulations, and preparation of the publication manuscript. This work was also done under the supervision of Professor Lutz Lampe and Professor Ian Blake.

Table of Contents

List of Tables

List of Figures

List of Symbols

$[n]$	set $\{1, \ldots, n\}$
$[n]_i$	set $\{i+1, i+2, \ldots, i+n\}$
Δ	number of equal-length subfiles per file
γ	Lagrange multiplier for capacity constraint
π_s^g	probability that for a set of caches $\mathcal{S} : \lvert\mathcal{S}\rvert = s$, $g \in \mathcal{G}_s$ is the set of files in $\boldsymbol{d}_\mathcal{S}$
A_D	a delivery algorithm
$\mathcal{C}$	set of distinct demands in $\boldsymbol{d}$
$\boldsymbol{d}$	demand vector $(d_1, \ldots, d_K)$
$\boldsymbol{d}_\mathcal{S}$	subdemand vector for requests of caches in $\mathcal{S}$
d_k	index of file demanded by cache k
$\mathcal{D}$	side information digraph
$\mathcal{D}_a$	asymptotic approximation to $\mathcal{D}$ at $K/N \to 0$
$\mathcal{E}$	set of edges of $\mathcal{D}$
F	length of files (bits)
$\mathcal{G}$	side information graph

$\mathcal{G}_a$ asymptotic approximation to $\mathcal{G}$ at $K/N \to 0$

$\mathcal{G}_s$ set of all subsets of $[N]$ with cardinality less than or equal s

K number of caches

L number of distinct demands in $\boldsymbol{d}$

$\mathcal{M}$ set of cache capacities for base-cases of RMSC

M cache capacity (files)

M_l $\max\{m \in \mathcal{M} \mid m < M\}$

M_u $\min\{m \in \mathcal{M} \mid M < m\}$

N number of files

$\mathcal{P}$ a placement of files in caches

p_n request probability of file n

q M/N

$\mathcal{R}$ set of optimal rates for base-cases of RMSC

R delivery rate (files)

$\mathcal{S}$ a subset of $[K]$

Supp($\boldsymbol{w}$) support of vector $\boldsymbol{w}$

X^n set of the bits of file n

$X^n_{\mathcal{S}}$ set of bits of file n that are exclusively cached in caches in $\mathcal{S}$

$x^n_{\mathcal{S}}$ length of subfile $X^n_{\mathcal{S}}$ normalized by F bits

x^n_s $x^n_{\mathcal{S}}$ for $\mathcal{S} : |\mathcal{S}| = s$

y^n_s $\binom{K}{s} x^n_s$

$\mathcal{Y}^*$ set of optimal solutions for base-cases of RMSC

$Y_i(t)$ number of cliques of size i formed up to iteration $t-1$ by Algorithm 9 on $\mathcal{G}_a$

Glossary

CDN Content Delivery Network

CFC Coded File Caching

CSC Coded Subfile Caching

CFC-CCD Coded File Caching with greedy Clique Cover Delivery

CSC-CCD Coded Subfile Caching with greedy Clique Cover Delivery

JRSM Joint Rate and Storage Minimization

RMSC Rate Minimization with Storage Constraint

SCC Structured Clique Cover (Algorithm 4)

Chapter 1

Introduction

From the early days of the invention of the internet, network congestion deemed to be a fundamental challenge in the communication of data. Enormous amounts of information must be transmitted from servers to clients over network links with limited communication capacities. During the peak traffic periods, the limited capacity of the links becomes a bottleneck for the transmission of data, adversely affecting the quality of service provided to the users. On the other hand, the nature of the web traffic is such that the same content is often demanded by several and potentially by a large number of users, a property that can be exploited to alleviate network congestion. In particular, the traffic on the network links can be partially offloaded by the storage of highly reused content at intermediate nodes in between the server and the end-users, eliminating the need for redundant transmission of the same content over the entire path from the server to the user. This technique is known as content caching and is nowadays an indispensable component of the Content Delivery Networks (CDNS).

Caching in CDNS is a mature field and the subject has been long studied. However, the newly established connections between content caching and certain elements of information theory, as well as the need for the deployment of caching techniques in networks that are fundamentally different from the conventional caching networks, has sparked renewed interest in research on caching techniques in the span of the last six years. In particular, the index coding problem in information theory corresponds to a fundamental component of the operation

of caching systems and answers the following question: how can the content that is not stored in the caches be delivered to them most efficiently by exploiting the content that they already have available. With the information theoretic perspective to the content caching problem comes a wide range of possible designs for caching systems, each of which needs to be evaluated and optimized for the best performance. This requires the deployment of various tools that are provided by optimization theory. This thesis therefore approaches the caching problem from an optimization theory perspective as well as a graph theoretical perspective inspired by index coding. This is accomplished by providing new designs for caching systems, their evaluation and optimization of their performance.

1.1 Caching in wireless networks: A fundamentally distinct caching paradigm

A prime motive for the consideration of caching problem in an information theoretic framework is the emergence of network models that are fundamentally different from the conventional caching network models. The network model determines how the caching nodes communicate with the servers in order to receive the content. Caching in wireless communication networks is an example of a network where the cache-server communications pattern significantly deviates from the conventional caching systems. In the following, we review the practical importance of caching in wireless networks and elaborate on the factors that differentiate the caching problem in such networks from the traditional caching paradigm.

1.1.1 Practical significance

Communication networks have witnessed a tremendous increase in mobile data traffic over the last decade. Projections indicate that this growth will continue from 2017 to 2022 with a compound annual growth rate of 46%, leading to a seven-fold increase in mobile data traffic [1]. The growth in mobile traffic is mainly due to the proliferation of smart devices and the emergence of a vast array of wireless services, including multimedia streaming, social networking and web browsing. The share of mobile video streaming from the mobile traffic is expected to increase to

75% by 2020. Following a similar trend, the average traffic that a mobile-connected end-user generates in 2020 is estimated to be 3.3 GB per month, up from a 495 MB per month in 2015 [2, 3]. To meet such tremendous traffic demands, the next generation communication networks need to shift to networking paradigms that allocate resources based on awareness of content, user and location [2]. Such paradigms promote distributed and proactive designs with the ability to adapt to the varying environmental parameters.

A powerful approach to meet the traffic demands in wireless communication networks is the deployment of multi-tier architectures, where small cells, aka femtocells, underlay the macrocellular network. This architecture brings short-range and low-cost base-stations closer to the end-users, providing a considerable capacity gain over the conventional high-power macro-cellular networks. Nonetheless, the existing multi-tier networks fall short of meeting the peak traffic demands during the congestion hours. This is because of the lack of cost-effective backhaul connectivity of the small cells to the backbone communication network [4, 5].

Local content caching is a promising technique to solve the problem of limited backhaul connectivity, replacing the expensive backhaul capacity with the cheaper storage capacity. Local caching exploits storage nodes with large memory capacities in the small cells (Fig. 1.1). By storing the popular content closer to the end-users, the caches serve the users locally and offload traffic from the backhaul links of the network [5]. This is effective in scenarios with large enough portion of duplicate demands. The cache-enabled networking paradigm is also proactive, in the sense that the network can take advantage of the users' contextual information to predict the future user demands using large-scale machine learning algorithms, and leverage its predictive ability to best utilize the available resources [4].

1.1.2 Broadcasting: A key differentiator for wireless caching

Although caching in content delivery networks is a mature technique, web caching techniques are not sufficient for wireless caching as they ignore certain fundamental characteristics of wireless networks. In particular, a key differentiator between web caching and wireless caching is the broadcasting nature of the wireless channel, i.e., a message sent by a transmitter can be received by multiple clients. This

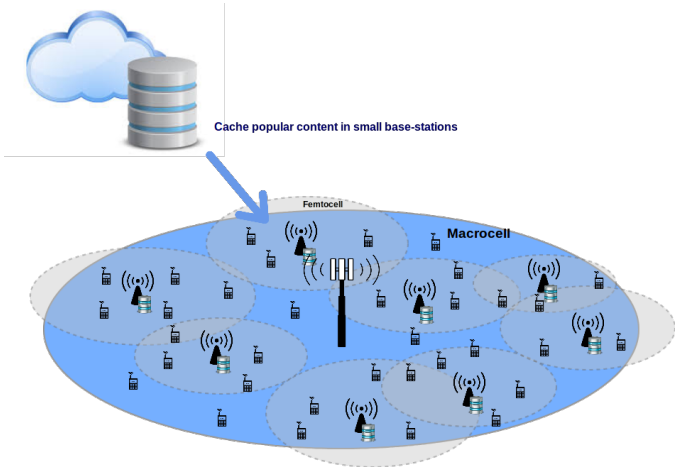


Figure 1.1: A two-tier cellular network: several caching femtocells are deployed within a macro-cell

property provides the opportunity for smarter designs of caching networks by using coding techniques from information theory, and promises notable gains in reducing the network traffic [6].

1.2 Connection to index coding

1.2.1 Overview of index coding

The broadcast nature of the wireless channel closely connects the design of caching systems to a long-investigated problem in information theory known as *index coding* or *source coding with side information* problem [7]. The index coding problem consists of a server and multiple caching clients. There exists a broadcast channel shared by all the clients, and only the server can transmit on the shared channel. Every clients makes a request for data from the server while it has some side information about the requests of the other clients. More precisely, each user has specific parts of the content requested by the other clients available in its cache. Although the clients cannot communicate that information with each other, such side information can provide opportunities for the server to send less amount of information on the channel in order to deliver the requests of the clients. Such

opportunities are referred to as coding opportunities. In the index coding problem, one aims at a smart construction of the server messages such that it maximally exploits the side information. This in turn translates to the server broadcasting the *minimal* amount of supplementary information on the shared channel such that every client is able to decode the transmitted messages for its desired information.

Solving the index coding problem is NP-hard in its general form, i.e., for an arbitrary arrangement of the side information on the clients. This results from the fact that the linear index coding problem can equivalently be formulated to the matrix min-rank problem [7] for linear codes, which is NP-hard to solve. Connections between the index coding problem and other NP-hard problems like vertex clique cover problem and graph coloring problem are also established in [8] and [9] and [10], respectively.

1.2.2 Canonical model for coded caching

To elaborate on the connection between the wireless caching problem and index coding, we lay out the canonical model for wireless coded caching problem. This model consists of a network with a central server and multiple caching nodes. The central server is connected to caches through an error-free broadcast channel.

A library of popular files is given, only a subset of which can be stored in the caches due to their limited storage capacity. The goal is to distribute the content of the files in the caches such that the amount of information transmitted on the broadcast channel to deliver the missing content is minimized. Since user demands are random, one can aim at minimizing either the peak or the average amount of information transmitted.

1.2.3 Similarities to and differences from index coding

It is evident that the index coding problem is closely related to the coded caching problem. In particular, in the caching problem, minimizing the amount of information that is transmitted by the server directly translates to requiring the server to fetch the least amount of information from the backbone communication channel, which alleviates the traffic load on the backhaul link of the network.

Despite their similarities, the index coding and caching problems have fundamental differences. In index coding problem, the core task is to construct delivery messages that best utilize the coding opportunities provided by a given side information arrangement, i.e., the focus is on creating the delivery messages. In contrast, there are two core tasks to the caching problem: the first task is the placement of the side information on the caching nodes, i.e., creating coding opportunities, and the second task is the construction of delivery messages. More precisely, the placement of content in the caches is itself a design aspect of the caching problem. Furthermore, the same placement of content in the caches is used to deliver a possibly large number of requests that are successively made by the clients, as one cannot update the content of the caches after delivering every set of client demands. The client demands are made randomly based on a probability distribution. Hence, the placement must be done such that it both creates coding opportunities and keeps the hit rate of the caches high, which eventually translate into minimizing a measure like the expected amount or maximum amount of supplementary information sent on the broadcast channel over the entire delivery interval. It becomes evident from our discussion that coded caching is a more general problem, with the index coding constituting one aspect of it. Despite the existence of a rich literature on the index coding problem, this difference has sparked the interest of the scientific community in the coded caching field since the year 2014 when the seminal work [11] on coded caching was published.

1.3 Summary of contributions and organization

This thesis investigates three research problems in the field of coded caching. Here, we provide a brief summary of each problem and our contributions. We leave a thorough review of the relevant literature to Chapter 2.

The first problem concerns the design of coded caching schemes when the files to be cached have different request probabilities. This is a significant problem as the core algorithms developed for coded caching are proposed for the case where demands for different files[1] are uniformly distributed. As will be discussed in

[1] popular files to be cached

Chapter 2, many works in the literature attempted to find the optimal way of allocating cache memory to files with different levels of popularity when the core coded message construction algorithms are used. This thesis lays out an optimization framework to formulate the problem of content placement and finds the optimal placement strategy analytically. One benefit of the analytical approach we follow is the insights that our approach provides about the optimal strategy and its connection to the placement algorithms that had been derived for uniformly popular files.

As the second problem, we explore the development of a coded caching scheme that keeps the files intact during the placement in the caches. In particular, the core coded caching algorithms require the splitting of each file into a possibly large number of chunks which will be distributed over different caching nodes. This property is inherited by the majority of the works in the coded caching literature. However, splitting files into such large number of chunks is not only difficult in practice, the theoretically promised caching gains are only valid if files are infinitely long. In this thesis, we propose a coded caching scheme where files are kept intact for placement in caches. In order to analyze the performance of the proposed method, we use a random graph model and a method of modeling the dynamics of the proposed algorithm by differential equations. We show that although the coding gain of file caching is smaller than that of subfile caching, it is still notably larger than uncoded caching.

The third problem that we investigate in this thesis is the construction of coded messages when multiple caches demand identical files. The core coded caching algorithm were not efficient in such a scenario, as they were originally developed under the assumption that caches request distinct files. We propose a method to improve the core algorithms to better suit duplicate file requests by different caches. In the chronological order, this was the earliest work completed for the sake of this thesis. Later works in the literature proposed more efficient techniques to deal with duplicate demands.

This thesis is organized as follows. In Chapter 2, we lay out the canonical model for coded caching and the general problem statement. We then briefly discuss the core coded caching algorithms that initiated the field. This is followed by a review of coded caching literature. In Chapter 3, we present our work on

coded caching with duplicate demands. Chapter 4 is devoted to coded caching with nonuniform demands. We present our work on coded caching without subpacketization of files in Chapter 5. We conclude the thesis in Chapter 6.

Chapter 2

Problem Statement and Review of Literature

2.1 Problem statement

At the core of the coded caching literature is the model proposed in [11] for a caching network which consists of a central server and multiple caching nodes. In this chapter we first present this model and then review the coded caching literature.

Consider a network with a central server that is connected to K caching nodes through a shared error-free communication link as in Fig. 2.1. There is a library of N files, each of which of size F bits. The whole database of files is available to the server. To the contrary, every cache has a limited storage capacity of MF bits. Hence, each cache can only have the equivalent of M files available in its local storage. The non-locally available content needed by a cache has to be acquired from the server via the shared broadcast communications link.

The described caching network operates in two phases. In the first phase, known as the placement phase, every cache fills its storage with parts of the content from the library, up to their storage capacity. This phase usually takes place during the off-peak hours of the network operation. We denote by $\mathcal{P}$ the placement of files in the caches. The knowledge of the file placement is equivalent to knowing a mapping from the bits of every file in the library to the storage of every cache in the network. The other phase of operation is known as delivery phase. The deliv-

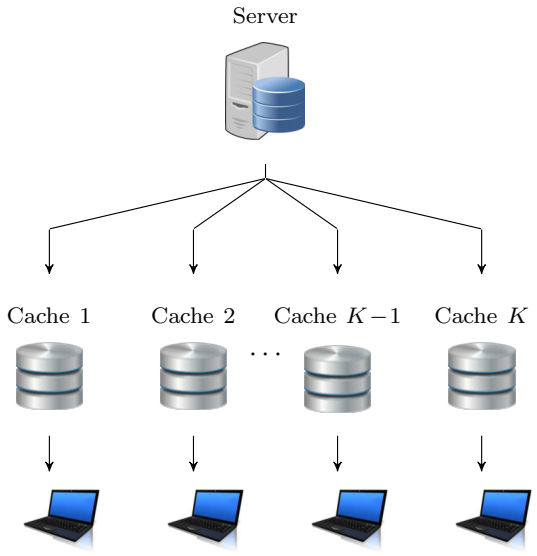


Figure 2.1: Schematic of a network of caches and a central server.

ery phase takes place during the time when the network is congested and after the placement phase. During delivery, the placement of content in the caches $\mathcal{P}$ does not change. Each cache provides its users with the parts of the requested files that it has locally available. However, the caches need to fetch the missing content from the server via the shared link.

Notation 1. *We use notation* $[n]$ *to denote the set of the first* n *positive integers* $\{1, \ldots, n\}$. *Similarly, we use* $[n]_i$ *to denote the set of the first* n *positive integers larger than* i, *i.e.,* $\{i+1, i+2, \ldots, i+n\}$.

The server is aware of the local content of all the caches. At each time instant, it is informed of the file requested by each cache. In particular, every cache $k \in [K]$ reveals one request for a file $d_k \in [N]$. We refer to $\boldsymbol{d} = [d_1, \ldots, d_K]$ as the *demand vector* which represents the demands of all caches at the current time instant.

To deliver the content requested by every cache, the server transmits a message $X_{\boldsymbol{d}}$ of size $R(\boldsymbol{d}; \mathcal{P}, A_D)F$ bits on the shared link. The quantity $R(\boldsymbol{d}; \mathcal{P}, A_D)$ is the *delivery rate* for the demand vector $\boldsymbol{d}$, given a specific placement of files $\mathcal{P}$ and a delivery algorithm used by the server denoted by A_D. To avoid notation clutter, we

drop the dependency of the rate on the placement and delivery algorithm from the notations when it is clear from the context.

Every cache needs to reconstruct the file it requested using both the content it has available locally and the message sent by the server. We say that a memory-rate pair $(M, R(\boldsymbol{d}))$ is *achievable for the demand vector* $\boldsymbol{d}$ if every cache k is able to perfectly recover its requested file d_k. Further, we say that the memory-peak-rate pair (M, R_{peak}) is *achievable* if it is achievable for all possible demand vectors $\boldsymbol{d} \in [N]^K$. For cache size M, the smallest rate R_{peak} for which (M, R_{peak}) is achievable characterizes the memory-peak-rate tradeoff. This smallest peak rate is denoted by $R^*_{\text{peak}}(M)$. Similarly, $\mathbb{E}\left(R(\boldsymbol{d}; \mathcal{P}, A_D)\right)$ is the expected delivery rate, where the expectation is over the randomness in vector $\boldsymbol{d}$, and possibly the randomness in the delivery algorithm A_D and in the placement algorithm that determines $\mathcal{P}$.

The design of a coded caching scheme requires the selection of the appropriate performance metric. In many practical scenarios, it is of interest to characterize the memory-expected-rate tradeoff $(M, \mathbb{E}\left(R(\boldsymbol{d}; \mathcal{P}, A_D)\right))$, i.e., to design the placement $\mathcal{P}$ and the delivery algorithm A_D such that for every demand vector $\boldsymbol{d}$, $R(\boldsymbol{d})$ is achievable and the expected delivery rate is minimized. Depending on the design priorities and requirements, the objective could instead be the characterization of the memory-peak-rate tradeoff. Notice that if the expected delivery rate is used as the measure of merit of a caching scheme, the knowledge of the probability distribution of the files becomes of primary significance as it directly affects the expected delivery rate.

Next in this chapter, we focus on the placement and delivery algorithms at the core of the coded caching literature, which are originally designed for uniformly distributed user demands and the peak-rate criterion as the performance measure. We then review the literature for other coded caching schemes that adapt these algorithms and their variations for more complex scenarios, as well as the works that follow fundamentally different approaches to coded-caching.

2.2 Algorithms core to coded caching

At the center of several coded caching schemes in the literature are the centralized and decentralized caching algorithms proposed by Maddah-Ali and Niesen in [11,

Algorithm 1 Placement of the centralized caching scheme of [11]

Require: $K, M, N, \{X^n\}_{n=1,\dots,N}$

$t \leftarrow KM/N$
for $k \in [K]$ **do**
  for $n \in [N]$ **do**
    split X^n into $\binom{K}{t}$ non-overlapping subfiles of equal size labeled by $\{X^n_{\mathcal{S}} : \mathcal{S} \subset [K], |\mathcal{S}| = t\}$
    Place subfiles $X^n_{\mathcal{S}} : k \in \mathcal{S}$ in the cache of user k
  end for
end for

12], respectively. Because of the conceptual significance of these algorithms and the fact that many of the techniques developed in the literature and in this thesis are closely related to them, we present these algorithms in this section.

First, we lay out the notations that we use throughout this thesis to characterize an uncoded placement of files in the caches.

Notation 2. *Let X^n be the set of the bits of file n. Also, for each $\mathcal{S} \subset [K]$, let $X^n_{\mathcal{S}} \subset X^n$ represent the bits of file n that are exclusively cached in the caches in $\mathcal{S}$.*[1] *By definition, subsets $X^n_{\mathcal{S}}$ are disjoint for different $\mathcal{S}$ and $\bigcup_{\mathcal{S}\subset[K]} X^n_{\mathcal{S}} = X^n$. Also, define $x^n_{\mathcal{S}} = |X^n_{\mathcal{S}}|/F$ as the ratio of bits of file n that are exclusively cached in the caches in $\mathcal{S}$. Then, it follows that $\sum_{\mathcal{S}\subset[K]} x^n_{\mathcal{S}} = 1$ for every $n \in [N]$. We denote by $\boldsymbol{x}$ the vector of all placement parameters $x^n_{\mathcal{S}}$.*

Centralized caching scheme of [11] The placement of [11] is presented in Algorithm 1. This placement algorithm assumes that the popularity of files is uniform, hence it allocates the same amount of cache storage to each file in the library. About this algorithm, the following features are worthy to recognize:

- For integer $t = KM/N$, each file is split into $\binom{K}{t}$ equal size subfiles. Hence, this algorithm does not keep the files intact during placement.
- With memory-rate pairs (M_1, R_1) and (M_2, R_2) for Algorithm 2 such that $t_1 = KM_1/N$ and $t_2 = KM_2/N$ are consecutive integers, any point on the

[1] In other words, bits $X^n_{\mathcal{S}} \subset X^n$ are stored in every cache in $\mathcal{S}$ and are not stored in any cache in $[K] \setminus \mathcal{S}$.

Algorithm 2 Delivery of the centralized caching scheme of [11]

Require: $\boldsymbol{d}$; $\{X^n\}_{n=1,\dots,N}$
1: $t \leftarrow KM/N$
2: **for** $\mathcal{S} \subset [K] : |\mathcal{S}| = t+1$ **do**
3: server sends $\oplus_{k\in\mathcal{S}} X^{d_k}_{\mathcal{S}\backslash k}$
4: **end for**

line segment connecting the pairs is also achievable using *memory sharing*. In particular, for $M = (1-\theta)M_1 + \theta M_2, 0 \leq \theta \leq 1$, use Algorithm 2 with cache size $M_1 F$ to delivery the first $(1-\theta)F$ bits of every file and with cache size $M_2 F$ for the rest θF bits of every file.

- If the number of caches in the network changes by the arrival of a new cache or the departure of an existing cache, quantity t could change. This requires the content of all the caches to be updated based on the new number of caches in the system. Hence, a central node needs to be aware of the global architecture of the network and to determine the content that each cache needs to store.

- This is a deterministic placement algorithm.

Once the content is placed in the caches using Algorithm 1, the missing content can be delivered to the caches by Algorithm 2. For a subset $\mathcal{S} \subset [K]$ of size $t+1$, consider a cache $l \in \mathcal{S}$. Then, $\mathcal{S}\backslash k$ is of size t and all the subfiles embedded in the message $\oplus_{k\in\mathcal{S}} X^{d_k}_{\mathcal{S}\backslash k}$[2] are available in cache l, except for $X^{d_l}_{\mathcal{S}\backslash l}$. Hence, cache l can decode the message for the part of its desired file that is not available locally. Repeating this process for all subsets $\mathcal{S}$ in the for loop, cache l can recover all the subfiles of file d_l that it has missing, and therefore, is able to fully reconstruct its requested file d_l.

It can be shown that the peak delivery rate of the centralized coded caching scheme with Algorithms 1 and 2 for placement and delivery is [11]

$$R^{\text{centralized}}_{\text{peak}}(M) = K\frac{1-M/N}{1+KM/N}F \quad \text{bits}, \tag{2.1}$$

[2]For compactness of notation, we use $\mathcal{S}\backslash k$ instead of $\mathcal{S}\backslash\{k\}$ when only one element is to be excluded from set $\mathcal{S}$.

Algorithm 3 Placement of the decentralized caching scheme of [12]

Require: $K, M, N, \{X^n\}_{n=1,\ldots,N}$

$t \leftarrow KM/N$
for $k \in [K]$ **do**
  for $n \in [N]$ **do**
    user k independently caches a subset $\frac{M}{N}F$ of bits of file n, chosen uniformly at random
  end for
end for

Algorithm 4 Delivery of the decentralized caching scheme of [12] (Structured-Clique Cover Algorithm)

Require: $\boldsymbol{d}; \{X^n\}_{n=1,\ldots,N}$

for $s = 1, \ldots, K$ **do**
  for $\mathcal{S} \subset [K] : |\mathcal{S}| = s$ **do**
    server sends $\oplus_{k \in \mathcal{S}} X^{d_k}_{\mathcal{S}\backslash k}$
  end for
end for

which provides a coding gain of $1 + KM/N$.

Decentralized caching scheme of [12] The facts that by the centralized placement algorithm of 1, the content of the caches are strongly dependent on the total number of caches, and the identity of the subfiles stored by each cache must be arranged in advance, restrict its usage in practice. The placement algorithm in Algorithm 3 resolves these issues by randomizing the placement process.

The corresponding delivery procedure in Algorithm 4 can be seen as a generalization of Algorithm 2. Here, in the for loop, the coded messages are constructed for all nonempty subsets $\mathcal{S} \subset [K]$ instead of only the subsets of size $t + 1$. We call this algorithm the Structured-Clique Cover Algorithm.

The following points characterize some important features of the placement and delivery based on Algorithms 3 and 4:

- The content that each cache stores is completely independent from the content that the other caches store. This results in a *decentralized* placement algorithm.

- The placement and delivery algorithms effectively split each file into 2^K subfiles corresponding to the subsets of $[K]$. Again, the number of subfiles grows exponentially with K. The lengths of these subfiles are not equal. In particular, the expected length of $X_{\mathcal{S}}^n$ is equal to $(\frac{M}{N})^s(1-\frac{M}{N})^{K-s}F$ bits, where $s=|\mathcal{S}|$.

The peak delivery rate of the decentralized coded caching with 3 and 4 for placement and delivery is

$$R_{\text{peak}}^{\text{decentralized}}(M) = K(1-M/N)\frac{N}{KM}\left(1-(1-M/N)^K\right)F \quad \text{bits} \qquad (2.2)$$

as $F \to +\infty$.

The decentralized caching scheme has a coding gain of $\frac{N}{KM}\left(1-(1-M/N)^K\right)$ and is proved to be order optimal, i.e., its peak delivery rate is within a constant factor of the optimal delivery rate [12].

Modified delivery algorithms of [13] In the case that multiple caches request the same file, Algorithms 2 and 4 embed the same subfiles of such a file into multiple server messages. This can be redundant and sub-optimal. Modified versions of the centralized and decentralized delivery algorithms were proposed in [13, Section IV.B and Appendix C.A], which resolve this inefficiency and are shown to achieve the optimal memory-rate tradeoff for the case of uniform demands when uncoded placement is used, for the centralized and decentralized settings [13].

More specifically, for any given demand vector $\boldsymbol{d}$, the modified algorithms first choose a set of leader caches $\mathcal{U}$ that have the property that they have all requested distinct files in $\boldsymbol{d}$. Hence, the number of leader caches is between 1 and K, depending on $\boldsymbol{d}$. In the decentralized setting, contrary to Algorithm 4, which greedily transmits the binary sums $\oplus_{k\in\mathcal{S}}X_{\mathcal{S}\backslash k}^{d_k}$ for every $\mathcal{S}\subset[K]$, the modified algorithm transmits the binary sum for $\mathcal{S}\subset[K]$ only if $\mathcal{S}\cap\mathcal{U}\neq\emptyset$. For the leader caches, decoding of the server messages is straightforward, as they have to only decode one server message as in Algorithms 2 and 4. The non-leader caches can still decode the server messages for the parts they are missing, but that requires each of them to process multiple messages. A similar result holds for delivery in the centralized setting.

In case of worst-case demands, $\mathcal{U} = [K]$, hence the modified algorithms reduce to the original algorithms 2 and 4 for the centralized and decentralized settings. Hence, the original algorithms are optimal in terms of the peak delivery rate. However, Algorithms 2 and 4 are suboptimal in terms of the expected delivery rate as this case also includes demands where $|\mathcal{U}| < K$.

2.3 A survey of coded caching literature

The seminal work of Maddah-Ali and Niesen [11] in 2014 initiated the field of coded caching by providing an information theoretic formulation of caching problem and investigating its fundamental limits. In this work, the authors further proposed a certain placement of content in the caches such that symmetric coding opportunities are created among the different caches. By virtue of these symmetry structures, they offered a solution to the resulting index coding problem. The combination of these placement and delivery message construction algorithms led to unprecedented gains in reducing the amount of supplementary information that the server needs to broadcast. In the following we provide a more quantitative characterization of this gain.

A naive adaptation of the conventional uncoded caching schemes to the canonical network model in Section 1.2.2 requires every cache in the network to store M out of the N files. Since each cache is conventionally served separately by the server, the identity of files stored in each cache is irrelevant in this setting[3]. Under uniform distribution over the user requests, $M/N\,F$ bits of the content that each cache requests is available to it locally on average, and the rest $(1 - M/N)F$ bits must be fetched from the server. Hence, the total amount of supplemental information transmitted by the server on the broadcast channel, the delivery rate, will be[4]

$$R_{\text{uncoded}}^{\text{worst-case}}(M; K, N) = K \left(1 - \frac{M}{N}\right) \min\left\{1, \frac{N}{K}\right\} F \quad \text{bits.} \tag{2.3}$$

[3]This is assuming files are equally popular and user requests are independent from each other.

[4]This is assuming that at each step, the requests of the caches are distinct. In the case of duplicate requests, (2.3) is an upper bound on the transmitted information.

Using the coded caching scheme proposed in [11], the worst-case delivery rate, which is an upper bound on the average delivery rate, is derived as

$$R^{\text{worst-case}}_{\text{centralized coded}}(M;K,N) = K(1-M/N)\min\left\{\frac{1}{1+KM/N},\frac{N}{K}\right\}F \quad \text{bits},$$

which is considerably smaller than the uncoded delivery rate. In particular, the coding gain is multiplicative and is equal to $1+KM/N$. An important observation is that for a fixed storage capacity per cache, as the number of caches increases, the delivery rate of uncoded caching linearly increases, while the delivery rate of the coded caching approaches the constant value $(N-M)/M$.

In a later work [12], Maddah-Ali and Niesen proposed the decentralized caching design for coded caching as detailed in Section 2.2, which has delivery rate [12]

$$R^{\text{worst-case}}_{\text{decentralized coded}} = K(1-M/N)\min\left\{\frac{N}{KM}\left(1-(1-M/N)^K\right),\frac{N}{K}\right\}F \quad \text{bits}.$$

The decentralized nature of this method made it the building block of several caching techniques that were designed later for more complicated scenarios [14–18].

In [19, 20], it was shown that the worst-case delivery rate of the centralized scheme in [11] is not only achievable, but it is the optimal worst-case delivery rate for coded caching schemes with uncoded prefetching and uniformly distributed demands, when the number of files is larger than the number of caches ($N > K$). The exact memory rate tradeoff for coded caching with uncoded prefetching and uniform demands was derived in [13], where the authors showed that the placement schemes in [11] and [12] are optimal respectively for centralized and decentralized settings both in terms of expected delivery rate and worst-case delivery rate. This result holds regardless of the number of files and caches. Further, expressions for the optimal worst-case and the optimal expected delivery rates where derived.

The coded caching scheme of [12] requires the splitting of each file into a large number of subfiles and the storage of subfiles in the caches. The number of subfiles per each file grows exponentially with the number of caches K. This property is therefore inherited by all coded caching designs that use these two schemes as their building blocks. Several works in the literature [21–25] have investigated

coded caching schemes with lower subpacketization requirements. In particular, it was shown in [21] that the decentralized scheme of [12] has a coding gain of at most 2 even if the subpacketization levels grows exponentially with the asymptotic coding gain KM/N. The authors proposed new caching schemes that result in better coding gains for smaller subpacketization levels. In [24], the authors proposed coded caching schemes based on specific combinatorial structures designs and propose methods with substantially reduced subpacketization levels at the cost of an increase in the delivery rate. Further, construction of coded caching schemes with polynomial subpacketization levels is explored in [22] using specific types of graphs, namely Ruzsa-Szemeredi graphs. More results in [23] establish connections between code design in the finite-length file regime and problems in combinatorics. In Chapter 5 of this thesis, we also investigate the coded caching problem when files are kept intact during placement, i.e., we propose a decentralized placement of files in the caches such that a file is either cached entirely or is not cached. We propose a greedy clique-cover method for delivery of the content and derive a method to compute the expected delivery rate of the proposed scheme.

For non-uniform popularity distribution of files, the placement phase can be improved such that it provides more memory resources to the storage of the more popular files. In such a scenario, the average delivery rate is a better metric for the performance of the system [14–16, 26, 26–28]. Many of the works on coded caching with nonuniform demands are based on the decentralized caching technique and a grouping of files such that within each group the popularity of files is relatively uniform. Other schemes [26] rely on graph theoretic methods for delivery. One contribution of this thesis is to find the optimal placement of content in the caches when demands follow a non-uniform distribution and Algorithm 4 is used for delivery.

Reference [29] considered the coded caching problem when the capacities of the different caches are not identical. Reference [18] investigated the hierarchical coded caching problem, where every cache in one layer acts as a server for the caches in its lower level. Multi-sever coded caching is explored in [30] where the caching nodes are served by multiple servers. This work focuses on coding delay as the optimization metric instead of the amount of information transmitted on the broadcast channel. The problem of online coded caching is explored in [17] where

the set of files in the library evolves based on a Markov model, yet the user requests are uniformly selected from the library. For this setting, an approximate expression is derived for the long-term average delivery rate for a rule developed to update the content of caches and the algorithm for decentralized coded caching. Reference [16] extended the work of [12] to the case that each user of the network is served by $d > 1$ caches. Here, the delivery phase is performed based on the decentralized delivery of [12].

Coded caching with coded prefetching of content is considered in several works in the literature [31, 32], yet these schemes are proposed for strictly limited regimes of problem parameters. In particular, [31], proposed a coded prefetching for the case where $M = (N-1)/K$, which results in improved delivery rate when the number of caches is larger than the number of files, $K > N \geq 3$.

Unlike most of the literature on coded caching that consider zero-distortion reconstruction of files, [33] investigates the scenario with user-dependent preset average distortion requirements, for both the centralized and decentralized caching settings. Here, the objective is to minimize the delivery rate such that for every user, all possible demand combinations can be met at the specified distortion levels. Coded caching in a network with noisy erasure broadcast channel is considered in [34], where lower and upper bounds on the memory-rate tradeoff are derived. The receivers belong to two disjoint sets, weak receivers with large erasure probabilities and cache memories and strong receivers with small erasure probabilities and no caching capability. Also in [35], the authors considered coded caching in the scenario that the different links between the sever and the different caches have unequal rates instead of assuming such links are error-free.

Finally, implementation aspects of coded caching in wireless networks are explored in [36].

It is worthwhile to mention that the literature on wireless caching is not limited to coded caching techniques, and the latter is a subset of the former. In particular, many works in the literature consider models that are different from the canonical model in Section 1.2.2 and optimize objectives other than the delivery rate. References [5, 37–40] are examples of such caching schemes that do not use network coding techniques for caching. The scope of this thesis only concerns coded caching methods and we limited our review of the literature accordingly. For a

more general review of the literature, we refer the readers to the survey of caching techniques for cellular networks provided in [41], as well as [6, 42] which explore practical considerations and challenges for wireless caching.

Chapter 3

Coded Caching for Delivery of Duplicate Demands

3.1 Inefficiency of the core delivery algorithms for duplicate demands

The core delivery procedures in Algorithms 2 and 4 are efficient when the different caches request distinct files in the demand vector. However, when there exists duplicate requests in the demand vector, i.e., multiple caches request the same file, Algorithms 2 and 4 are clearly suboptimal as they ignore the redundancies in the user demands and deliver the requested content as if all demands were distinct.

The presence of redundant demands becomes more likely in several different scenarios, some of which we enumerate here. First, as the ratio of the number of caches to the number of files K/N increases, the chance of all caches requesting distinct files decreases, leading to duplicate demands. The second scenario is when there exists significant differences in the popularity of files. Duplicate demands become more likely as many caches request the highly popular files. Third, certain correlations among the requests of the different caches could make it more probable for them to demand identical content. Such correlated requests are likely in many practical scenarios. A considerable amount of multimedia requests are made through social networks like Facebook and Instagram, where users with common

social connections and interests are likely to request the same content.

In this chapter, we investigate the delivery of redundant demands using modified versions of Algorithms 2 and Algorithm 4. For placement, we use the same schemes as in Algorithms 1 and 3 for the centralized and decentralized caching scenarios, respectively. The use of such a placement scheme is justified if accurate estimates of the popularity of the files to be cached are not known during the placement.[1] In the case of correlated requests also, the marginal probabilities of requesting each file could be similar, yet the joint distribution of the demands could favor redundancy in the demand vector. Estimation of such a joint distribution can be challenging and so is designing a corresponding optimal placement of files in the caches. These are examples where a symmetric placement of files is a viable option.

For the delivery, however, we propose a new scheme based on *message selection*. Upon receiving a demand vector from the users and based on the redundancy pattern of the requests made, the server decides whether to use uncoded messages or the coded messages as in Algorithm 4 to deliver the different parts of the requested files. This distinguishes our work from [11, 12], as our proposed delivery takes the specifics of the current demand vector into account to decide on the form of the server messages. However, [11, 12] use a fixed structure to compose the server messages for all demand vectors. We also derive a lower bound on the delivery rate of redundant requests.

Subsequent to our work, [13] proposed different variants of Algorithms 1 and 3 which were proved to be optimal in terms of both the worst-case delivery rate and the expected delivery rate for the centralized and decentralized settings when demands are uniformly distributed and placement is uncoded. This means that under this setting, these algorithms handle the redundant demands optimally. We compare the performance of our proposed delivery to the optimal delivery algorithm for the completeness of the results.

[1] Clearly, in the setting where accurate estimates of the file request probabilities are available, they should be taken into account during placement. This is the investigated in Chapter 4 in detail.

3.2 Selection between coded/uncoded messages for duplicate demands

We use the caching network model in Section 2.1. To account for the redundancies in the demand vector, let L represent the number of distinct files in the demand vector, hence, $1 \leq L \leq K$. The demand vector is called *redundant* if $L < K$. Also, denote by k_i, the number of requests for the i-th most requested file in the current demand vector. Thus $k_i \geq k_j$ for $i < j$ and $i, j \in [L]$. We call $(k_1, ..., k_L)$ the redundancy pattern of the demand vector.

The delivery rate of Algorithm 4 for a non-redundant demand vector is

$$R_{\text{worst-case}} = \sum_{\substack{\mathcal{S}:\mathcal{S}\subset[K]\\ \mathcal{S}\neq\emptyset}} \max_{k\in\mathcal{S}} x^{d_k}_{\mathcal{S}\backslash k}. \tag{3.1}$$

Since the placement is symmetric w.r.t. all the files $x^n_{\mathcal{S}\backslash k}$ is the same for all files n for placement in Algorithm 1 in general and for placement in Algorithm 3 in the infinite file size regime. Hence, we can drop the superscript in $x^n_{\mathcal{S}}$ and simply use $x_{\mathcal{S}}$ instead. Similarly, both placements in Algorithms 1 and 3 are symmetric w.r.t subsets $\mathcal{S}$ as long as they have the same cardinality. Hence, $x_{\mathcal{S}}$ is only a function of $|\mathcal{S}|$. As a result, we further simplify the notation $x_{\mathcal{S}}$ to x_s, where $s = |\mathcal{S}|$. Eq. (3.1) can then be written as

$$R_{\text{worst-case}} = \sum_{s=0}^{K-1} \binom{K}{s+1} x_s. \tag{3.2}$$

If file n is requested by multiple users, including user k, Algorithm 4 embeds $X^n_{\mathcal{S}\backslash k}$ into several messages. If $|\mathcal{S}| > 1$, user k has the side information to directly decode only one of those messages. As a result, the server needs to send all the messages with $|\mathcal{S}| > 1$.[2] This is not the case for the messages with $|\mathcal{S}| = 1$, i.e., $\mathcal{S} = \{k\}$. In these cases, $\oplus_{k\in\mathcal{S}} X^{d_k}_{\mathcal{S}\backslash k} = X^{d_k}_{\emptyset}$. Such uncoded messages deliver the bits that are not stored at any cache in the system. All the users that request file

[2]Again, this is the case for the simple decoding technique that requires the decoding of exactly one server message for one subfile at each cache. If each cache can perform the extra processing of several server messages to decode each subfile, as is proposed in [13], a smaller number of messages can be sent.

n can decode $X^n_\emptyset$, so it needs to be sent only once. As a result, the load due to the uncoded messages is Lx_0 instead of Kx_0 and the delivery rate of a redundant demand will be

$$Lx_0 + \sum_{s=1}^{K-1} \binom{K}{s+1} x_s. \tag{3.3}$$

Eq. (3.3) suggests that for redundant demand vectors, the rate of Algorithm 4 can be smaller than the peak rate if the uncoded messages are sent only once for the files that are requested by multiple caches. This is the basis of our analysis in Section 3.2 where we show that depending on the redundancy pattern in the demand vector, it might be beneficial to favor the transmission of uncoded messages over coded messages.

Delivery based on message selection

To improve the delivery for demand vectors with duplicate requests, we introduce an extra step to the delivery phase, which takes place after receiving each request vector and before the transmission of the server messages to the users. In this step, the server decides whether to send each part of the requested files through the corresponding coded message in Algorithm 4 or through an uncoded message. The use of uncoded messages instead of coded messages to deliver file n can be thought of transferring bits from $X^n_\mathcal{S} : \mathcal{S} > 0$ to $X^n_\emptyset$. In other words, the cache delivers requests as if parts of its content were not available.

Let $\hat{X}^n_\mathcal{S}$ represent the subset of the bits of file n exclusively cached at $\mathcal{S}$ after the transfer is done, and $\hat{x}^n_\mathcal{S} \triangleq |\hat{X}^n_\mathcal{S}|/F$.

In our delivery method, the server first optimizes $\hat{x}^n_\mathcal{S}$. Then, it arbitrarily picks $\hat{x}^n_\mathcal{S} F$ bits of $X^n_\mathcal{S}$ to form $\hat{X}^n_\mathcal{S}$, and adds the rest of the bits to $\hat{X}^n_\emptyset$. Finally, it uses Algorithm 4 for delivery based on the resulting subsets $\hat{X}^n_\mathcal{S}$ instead of $X^n_\mathcal{S}$.

We now find the optimal lengths of the updated subsets $\hat{X}^n_\mathcal{S}$ to minimize the sum of the lengths of messages $\oplus_{k\in\mathcal{S}} \hat{X}^{d_k}_{\mathcal{S}\backslash k}$ over all $\mathcal{S} \subset [K]$. Let $\mathcal{C}$ denote the set of the distinct files requested in the current demand vector. Note that $|\mathcal{C}| = L \leq K$, and both $\mathcal{C}$ and L evolve with time. Then, the rate minimization problem is given

by

$$\begin{aligned} \underset{\hat{x}_{\mathcal{S}}^{d_k}}{\text{minimize}} \quad & \sum_{\mathcal{S}:\mathcal{S}\subset\mathcal{K}} \max_{k\in\mathcal{S}} \hat{x}_{\mathcal{S}\setminus k}^{d_k} \\ \text{subject to} \quad & \sum_{\mathcal{S}:\mathcal{S}\subset\mathcal{K}} \hat{x}_{\mathcal{S}}^{d_k} = 1, \quad \forall\, d_k \in \mathcal{C} \\ & 0 \le \hat{x}_{\mathcal{S}}^{d_k} \le x_{|\mathcal{S}|}, \quad \forall\, d_k \in \mathcal{C},\ \forall \mathcal{S} \subset \mathcal{K} : |\mathcal{S}| > 0 \\ & 0 \le \hat{x}_{\emptyset}^{d_k} \le 1, \qquad \forall\, d_k \in \mathcal{C}. \end{aligned} \tag{3.4}$$

In (3.4), $x_{|\mathcal{S}|} = |X_{\mathcal{S}}^n|/F$ are known from the placement phase. For the centralized placement algorithm of 1, the placement parameters are given by

$$x_s = \begin{cases} 1/\binom{K}{t}, & s = t \\ 0, & \text{otherwise} \end{cases} \tag{3.5a}$$

if t is an integer, and

$$x_s = \begin{cases} (\lceil t \rceil - t)/\binom{K}{\lfloor t \rfloor}, & s = \lfloor t \rfloor \\ (t - \lfloor t \rfloor)/\binom{K}{\lceil t \rceil}, & s = \lceil t \rceil \\ 0, & \text{otherwise} \end{cases} \tag{3.5b}$$

for non-integer t. The parameters of the latter case are obtained by memory-sharing [11]. For the decentralized placement and for large F, with high probability we have [12]

$$x_s^{\text{decen}} \approx q^s (1-q)^{K-s}, \quad s = 0, ..., K. \tag{3.6}$$

Quantity $\max_{k\in\mathcal{S}} \hat{x}_{\mathcal{S}\setminus k}^{d_k}$ is the length of the message $\oplus_{k\in\mathcal{S}} \hat{X}_{\mathcal{S}\setminus k}^{d_k}$. Thus, the objective function is the rate of Algorithm 4 operating based on the adjusted subsets $\hat{X}_{\mathcal{S}}^n$. The equality constraints follow the definition in Notation 2. The parameter range constraints permit the server to use uncoded messages instead of coded messages, but not vice versa.

Problem (3.4) can be posed as a linear programming problem by the standard technique of defining ancillary variables $z_{\mathcal{S}} = \max_{k\in\mathcal{S}} \hat{x}_{\mathcal{S}\setminus k}^{d_k}$, and adding the extra

Algorithm 5 Delivery algorithm based on message selection

Require: $\{X_{\mathcal{S}}^n\}_{n,\mathcal{S}}$ *// From the placement phase*

Procedure AdaptiveDelivery($d_1, ..., d_K$)
$\mathcal{C} \leftarrow$ unique($d_1, ..., d_K$) *// Set of distinct files requested*
$\{\hat{x}_{\mathcal{S}}^{*d_k}\}_{d_k \in \mathcal{C}, \mathcal{S} \subset \mathcal{K}} \leftarrow$ Solution of Problem (3.4)
for $d_k \in \mathcal{C}$ **do**
  $\hat{X}_{\emptyset}^{d_k} \leftarrow \emptyset$ *// Initialization of* $\hat{X}_{\emptyset}^{d_k}$
  for $\mathcal{S} \subset [K]$ **do**
    $\hat{X}_{\mathcal{S}}^{d_k} \leftarrow \{$first $\hat{x}_{\mathcal{S}}^{*d_k} F$ bits of $X_{\mathcal{S}}^{d_k}\}$
    $\hat{X}_{\emptyset}^{d_k} \leftarrow \hat{X}_{\emptyset}^{d_k} \cup \{$ last $(1 - \hat{x}_{\mathcal{S}}^{*d_k})F$ bits of $X_{\mathcal{S}}^{d_k}\}$
  end for
end for
Use [12, Algorithm 1] with $\{\hat{X}_{\mathcal{S}}^n\}_{n,\mathcal{S}}$ instead of $\{X_{\mathcal{S}}^n\}_{n,\mathcal{S}}$

constraints

$$z_{\mathcal{S}} \geq \hat{x}_{\mathcal{S}\setminus k}^{d_k} \tag{3.7}$$

for all $\mathcal{S} \subset [K] : |\mathcal{S}| > 0$ [43, Section 4.3], as $\hat{x}^{d_k} \geq 0$. The resulting linear programming problem can be solved numerically for $\hat{x}_{\mathcal{S}}^{*d_k}$. Algorithm 5 shows the proposed message-selection based delivery scheme.

Simplified messages selection

A simplified version of the message selection step can be formulated by only taking the number of distinct requests L into account, and ignoring the redundancy pattern of the demand vector. Then, because of the symmetry, we set $\hat{x}_{\mathcal{S}}^n = \hat{x}_s$ for all n and all $\mathcal{S} : |\mathcal{S}| = s$. This leads to

$$\begin{aligned} \underset{\hat{x}_s}{\text{minimize}} \quad & Ly_0 + \sum_{s=1}^{K-1} \binom{K}{s+1} \hat{x}_s \\ \text{subject to} \quad & \sum_{s=0}^{K} \binom{K}{s} \hat{x}_s = 1 \\ & 0 \leq \hat{x}_s \leq x_s, \ s = 1, ..., K \\ & 0 \leq \hat{x}_0 \leq 1 \end{aligned} \tag{3.8}$$

as the simplified message selection problem.

Proposition 1. *Let* $\hat{s} = \lfloor \frac{K-L}{L+1} \rfloor$. *Optimal parameters for the simplified message selection problem of (3.8) are given by*

$$\hat{x}_s = \begin{cases} \sum_{i=1,...,\hat{s}} \binom{K}{i} x_i, & s = 0 \\ 0, & s = 1, ..., \hat{s} \\ x_s, & s = \hat{s}+1, ..., K \end{cases} . \tag{3.9}$$

Proof. If we transfer bits from the subsets $X^n_{\mathcal{S}} : |\mathcal{S}| = s$ to $X^n_{\emptyset}$, the resulting change in the rate will be $L\binom{K}{s}x_s - \binom{K}{s+1}x_s$. We transfer the bits only if this difference is negative. This is the case when $s \leq \hat{s}$. This results in the parameters of (3.9). □

Algorithm 6 shows the delivery scheme based on the simplified message selection criterion.

Algorithm 6 Simplified Adaptive Delivery Algorithm

Require: $\{X^n_{\mathcal{S}}\}_{n,\mathcal{S}}$ *// From the placement phase*

Procedure SimplifiedAdaptiveDelivery($d_1, ..., d_K$)
$\mathcal{C} \leftarrow$ unique$(d_1, ..., d_K)$ *// Set of distinct files requested*
$L = |\mathcal{C}|$ *// Number of distinct requests*
$\hat{s} \leftarrow \lfloor \frac{K-L}{L+1} \rfloor$
for $d_k \in \mathcal{C}$ **do**
  $\hat{X}^{d_k}_{\emptyset} \leftarrow \cup_{\mathcal{S}:s\leq\hat{s}} X^{d_k}_{\mathcal{S}}$ *// Corresponds to the first rule of (3.9)*
  for $\mathcal{S} \subset [K] : |\mathcal{S}| > 0$ **do**
    if $|\mathcal{S}| \leq \hat{s}$ **then**
      $\hat{X}^{d_k}_{\emptyset} \leftarrow \emptyset$ *// Corresponds to the second rule of (3.9)*
    else
      $\hat{X}^{d_k}_{\mathcal{S}} \leftarrow X^{d_k}_{\mathcal{S}}$ *// Corresponds to the third rule of (3.9)*
    end if
  end for
end for
Use [12, Algorithm 1] with $\{\hat{X}^n_{\mathcal{S}}\}_{n,\mathcal{S}}$ instead of $\{X^n_{\mathcal{S}}\}_{n,\mathcal{S}}$

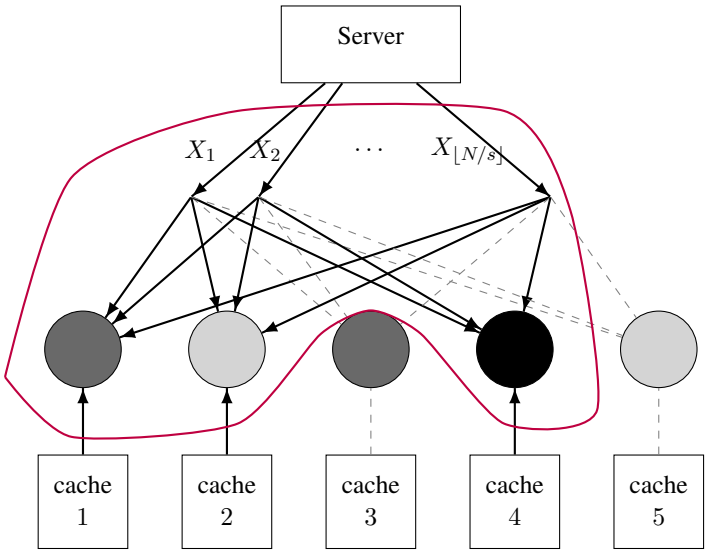


Figure 3.1: An example of a cutset separating the caches and the server from the users of the caches in $\mathcal{S} = \{1, 2, 4\}$. Solid (dashed) lines represent the information flow to the users selected (not selected) in the cutset. Here $s = 3$ and there are $\lfloor \frac{N}{s} \rfloor$ server messages. Users with the same color have identical requests. Notice that no two users with the same request are picked in $\mathcal{S}$. Here $K = 5$.

3.3 A cutset bound on delivery rate of duplicate demands

Let $R_L^*(M)$ denote the minimum rate that is achievable for every possible demand vector with L distinct requests. More specifically, let $\mathcal{A}_L$ be the set of all demands vectors with L distinct files. For a placement $\mathcal{P}_M$ that satisfies the capacity constraint, i.e., stores an equivalent of at most M files per cache, let

$$R_L(\mathcal{P}_M) = \min\{R \geq 0 | \forall \boldsymbol{d} \in \mathcal{A}_L : R(\boldsymbol{d}; \mathcal{P}_M) \leq R\},$$

where $R(\boldsymbol{d}; \mathcal{P}_M)$ is the amount of supplemental information that needs to be sent by the server over the shared link such that every cache can recover its requested file without error. Then:

$$R_L^*(M) = \min_{\mathcal{P}_M} R_L(\mathcal{P}_\mathcal{M}).$$

Proposition 2 gives a lower bound on $R_L^*(M)$.

Proposition 2 (Cutset Bound)**.** *Assume that K caches request $L \leq K$ distinct files. Then, $R_L^*(M)$ must satisfy*

$$R_L^*(M) \geq \max_{s\in\{1,\dots,L\}} \left(s - \frac{s}{\lfloor N/s \rfloor} M \right). \tag{3.10}$$

Proof. We modify the cutset bound argument of [11, Sec. VI] to bound the minimum delivery rate of the demand vectors with $L \leq K$ distinct requests.

Let $\mathcal{S}$ be a subset of caches with $|\mathcal{S}| = s$, such that there are no two caches in $\mathcal{S}$ with identical user requests. Assume that these caches have requested files $1, \dots, s$ from the library of N files. Let X_1 denote the server's input to the shared link which determines files $1, .., s$. Similarly, assume that the same users request files $(i-1)s+1, ..., is$ and the server input X_i determines the files requested. Let $i = 1, ..., \lfloor N/s \rfloor$.

Consider the cut separating $X_1, ..., X_{\lfloor N/s \rfloor}$ and the caches in $\mathcal{S}$ from the corresponding users (see Fig. 3.1). Since we assume that all files are perfectly decoded without error, the total information available to the users in the cut should be more than or equal to the total information requested by them. In other words,

$$\lfloor N/s \rfloor R_L^*(M) + sM \geq s\lfloor N/s \rfloor.$$

Since s can accept any value between 1 and L, (3.10) results. □

3.4 Numerical explorations

We now numerically explore the performance of the proposed adaptive methods and the non-adaptive method of Algorithm 4. Notice that by the rate of non-adaptive method, we refer to the rate of [12] or [11] depending on whether the decentralized or centralized placement is used, respectively. This rate is calculated by (3.3).

Fig. 3.2 shows the delivery rates of the non-adaptive and adaptive schemes, as well as the lower bound (3.10) for a network of $K = 12$ caches. The same decentralized placement is used for all cases with the parameters in (3.6). We

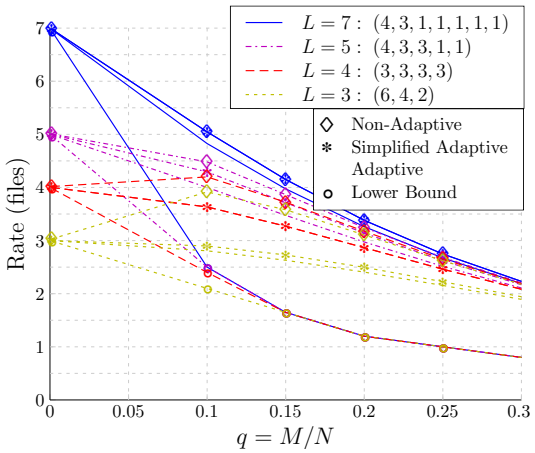


Figure 3.2: Rates of different delivery schemes for $K = 12$.

consider several redundancy patterns for the demand vector with different values of L. In Fig. 3.2, we observe a considerable improvement in the delivery rate for $M/N \leq 0.25$, when the adaptive methods are used. This improvement in the rate is more considerable when L is smaller. For instance, the performance gap to the lower bound decreases by almost 50% when $L = 3$. Notice that for a symmetric redundancy patterns like $(3, 3, 3, 3)$, both adaptive methods lead to the same delivery rate. As the pattern gets more asymmetric, the gap between the rates of the original and simplified adaptive methods increases. Also, observe that for some cases, the rate of the non-adaptive method increases with the storage capacity for small M/N. This shows the inefficiency of Algorithm 4 to deliver redundant requests.

For the second numerical example, we use the centralized placement and plot the delivery rates resulted from the different methods versus L. Fig. 3.3 shows the results. Notice that the rate of original adaptive method depends not only on L, but also on the redundancy pattern. Hence, in this example, for every value of L, the delivery rate of the original adaptive method is averaged over all the redundancy patterns with L distinct requests, assuming that the requests are independent and file popularities are uniform. Observe that the superiority of the adaptive method over the non-adaptive method of [11] is more significant for small L. In particular, we observe a sharp decrease in the adaptive delivery rate when L gets smaller than

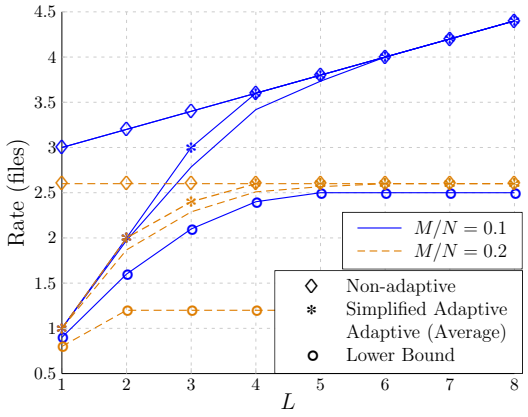


Figure 3.3: Rates of different delivery schemes for $K = 8$.

$K/2$. This suggests that the adaptive methods are considerably more effective for highly redundant requests.

We now investigate the average rates of the different delivery methods through a stochastic modelling of the dynamics of a caching network with correlated user requests. Our simulations here provide more insights on the difference between the delivery rates of the different algorithms as a function of the correlations among user demands. Consider a graph representation of the network where vertices represent the caches. An (undirected) edge between two vertices shows that the requests of the corresponding caches are correlated. To model the correlations, let $\mathcal{N}(k)$ denote the set of the last files requested by the neighbour caches of cache k. We assume that cache k requests a file, either based on its neighbours' previous requests with probability r, or independently with probability $1 - r$. In the former case, k chooses a file from $\mathcal{N}(k)$ uniformly at random. However, when choosing independently, k picks a file n from the library based on the popularity distribution of files p_n. Hence, the chance of requesting file n by cache k is

$$\hat{p}_{n,k} = \begin{cases} r/|\mathcal{N}(k)| + (1-r)p_n, & n \in \mathcal{N}(k) \\ (1-r)p_n, & \text{otherwise} \end{cases}. \tag{3.11}$$

For our simulations, we use Gibbs sampling [44, Sec. 24.2] to generate 10^4 sample vectors from the induced joint distribution of user demands. We set K=8 and

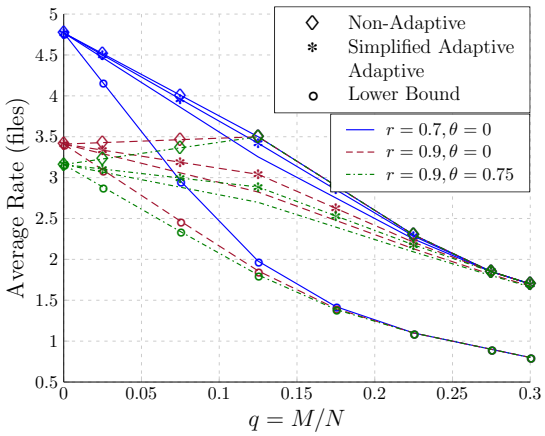


Figure 3.4: Average rates of the different delivery schemes for $K = 8$.

(r, θ)	$(0.7, 0)$	$(0.9, 0)$	$(0.9, 0.75)$
Maximum $\hat{\rho}_{ij}$	0.19	0.34	0.34
Average $\hat{\rho}_{ij}$	0.16	0.32	0.31
Average L	4.80	3.41	3.18

Table 3.1: User requests' statistics in simulations of Fig. 3.4.

N=10^3, and assume a complete graph for the network. Further, we mainly use uniform distribution for the popularity of files. We also consider a scenario where the placement phase is performed based on a uniform popularity distribution, while the actual file popularities in the delivery phase follow a non-uniform Zipf distribution [45] with parameter θ. Note that a Zipf distribution with $\theta = 0$ is identical to the uniform distribution, and increasing θ makes the distribution more non-uniform. We use θ and r to control the popularity distribution and the dependency level of the users' requests, respectively. To characterize the resulting correlation levels among the caches' requests in our simulations, we empirically calculate the correlation coefficients $-1 \leq \hat{\rho}_{ij} \leq 1$ [46, Section 4.1] between the requests of the different caches i and j. A larger $\hat{\rho}_{ij}$ implies a higher chance that caches i and j request the same content, which leads to more redundancy in the demand vector. Table 3.1 presents the average and the maximum $\hat{\rho}_{ij}$ over all the different i and j pairs ($i \neq j$), in our simulations.

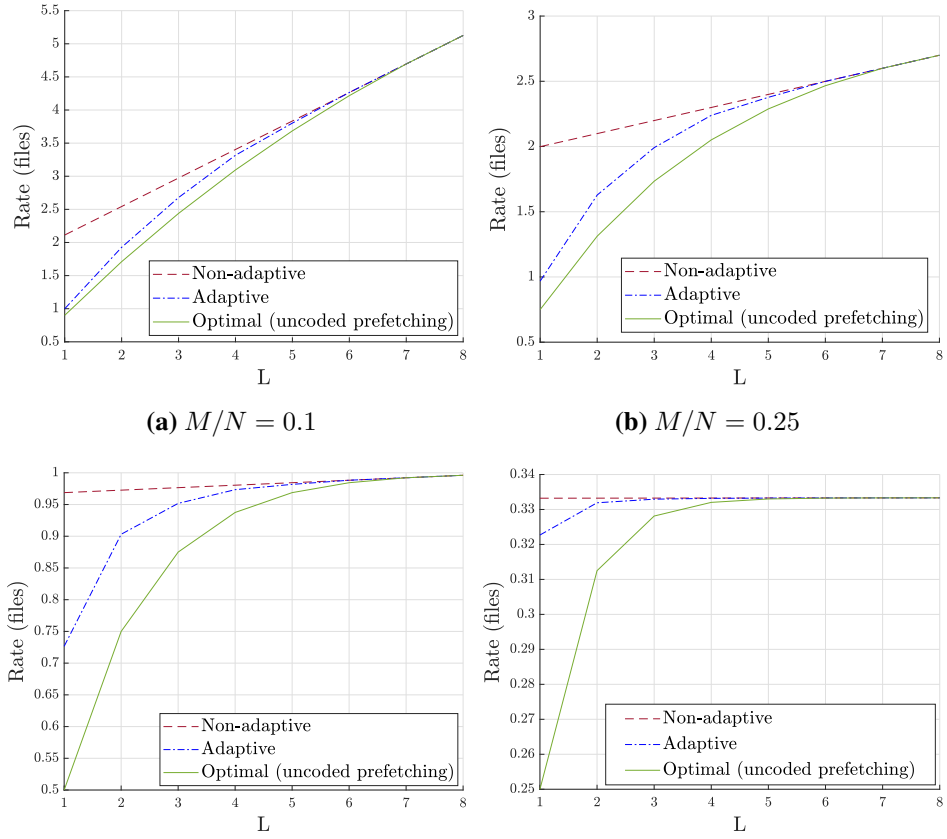
Fig. 3.4 shows the resulting average delivery rates. It also shows a lower bound

on the average rate calculated by averaging the lower bounds of (3.10) for the sample demand vectors. We observe that as requests become more correlated (larger r) and the file popularities get more non-uniform (larger θ), the adaptive method makes a larger improvement in the rate. Also, observe that the adaptive schemes are effective in decreasing the average delivery rate for $M/N < 0.25$.

3.5 Comparison to optimal coded caching with uncoded prefetching for uniform demands

In 2018, reference [13], derived the optimal decentralized and centralized coded caching schemes for the case of uniform demands and uncoded prefetching as explained in Section 2.2.

For the completeness of our discussions in this chapter, we also compare the rate of the proposed adaptive method to the rate of the optimal caching scheme with uncoded prefetching for uniform demands. In Fig. 3.5, we compare the expected rate of the non-adaptive delivery, adaptive delivery and the optimal delivery of [13]. In the hybrid coded/uncoded delivery approach, we replace the sets of coded messages whose transmission is inefficient in the presence of redundant demands with uncoded messages. Instead, in [13], a subset of coded messages are transmitted when demands are redundant, as they suffice to extract the required information at each cache, at the expense of further processing of the server messages by the caches. The latter approach benefits more from the coding opportunities and therefore results in better delivery rates. For small M/N, the rates of our proposed hybrid method and the optimal method of [13] are close. This is because the coding opportunities are limited and each message is mostly targeted towards a small number of caches. Hence, transmission of uncoded messages instead of coded messages provides more benefit when demands are redundant compared to extent of the loss of coding gain. For large M/N, however, the loss due to unused coding opportunities dominates and the gap between the hybrid coded/uncoded approach to the optimal delivery increases.



(a) $M/N = 0.1$ **(b)** $M/N = 0.25$

(c) $M/N = 0.5$ **(d)** $M/N = 0.75$

Figure 3.5: Comparison of the expected rate of the non-adaptive, adaptive and optimal delivery for uniform demands for fixed number of distinct files. Here, $K = 8$.

3.6 Concluding remarks

In this chapter, we identified an inefficiency in the core delivery algorithms of coded caching in the presence of duplicate requests in the demand vector. We proposed a method to improve the efficiency of the delivery messages by introducing s step to decide which parts of the files need to be sent by coded messages and which parts should be delivered uncoded, based on the redundancy pattern in the demand vector. We discussed the improvement that the proposed method offers

over the benchmark algorithms and also compared its performance to the optimal delivery scheme. An important contribution of this chapter, was the formulation of delivery rate in terms of placement parameters. This allows optimization of the placement parameters such that the delivery rate is minimized. A generalized version of this formulation is the foundation of our work in Chapter 4 for placement optimization.

Chapter 4

Coded Caching for Nonuniform Demands

4.1 Introduction

The seminal works [11] and [12] designed coded caching methods that minimize the peak delivery rate, i.e., the delivery rate when all demands are distinct. In the case of uniform demands, for a fixed number of caches K, the average delivery rate approaches the peak delivery rate as the the number of files increases, i.e., as $N \to \infty$. Hence, the caching schemes of [11] and [12] are also suitable for uniform demands. However, a more practical scenario concerns caching of files with different popularity levels, i.e., the probabilities of different files being requested are different.

In its most general form, not only the file request distribution can be nonuniform, it can even be different from one cache to another. This is particularly relevant if each cache is serving a small number of users and therefore the identity of popular files is closely tied to the preferences of the individual users covered by each cache. In such a scenario, the design of coded caching is challenging. On one hand, more global caching gain is achievable if different caches store files symmetrically from the same library of files, on the other hand, such a scheme cannot be sensitive to the asymmetries introduced by the different local demand distributions. A simpler scenario is the case where the user demands are nonuniform,

but the request distribution is the same for the different caches. This is reasonable assumption when each cache serves a large number of users randomly selected from the population regardless of their demographics and preferences. Even in this case, the optimal coded caching strategy is difficult to characterize because of the complex relationship between the placement strategy and the delivery rate. Intuitively, it is expected to allocate more storage to the caching of the more popular files during placement. This idea is followed in several works in the literature [14–16, 26, 47, 48]. In this chapter, we focus our attention to derive the optimal file placement for coded caching with nonuniform demands where the demand distribution is identical for different caches.

4.1.1 Related work

Two major approaches are followed for coded caching with nonuniform demands. The first approach is based on the grouping of files into different popularity groups based on their request probabilities [14–16]. With the files in each group having relatively similar request probabilities, the Structured Clique Cover (Algorithm 4) (SCC) algorithm is applied separately to the requests belonging to each group for delivery. The advantage of this method is the simplicity of the analysis of the expected rate. Its main disadvantage is that it limits the utilization of coding opportunities to the files that are within each popularity group. The design objective in this approach is to find the grouping of files that achieves the lowest expected rate. A higher number of groups provides higher degrees of freedom to assign different amounts of storage to files with different request probabilities. On the other hand, the larger the number of groups is, the more underutilized are the coding opportunities among the different groups. In [15], the library is grouped into two categories of popular and unpopular files. The requests for popular files are delivered by the SCC algorithm while the requests of unpopular files are delivered through uncoded messages. This is an extreme case of the grouping approach and its expected rate is shown to be at most a constant factor away from the information theoretic lower bound, independent of the file popularity distribution.

The second approach is followed in [47] and [48] which applies the SCC algorithm to all the user demands regardless of their request probabilities and the

amount of storage allocated to each file. For any fixed placement of content, this delivery scheme outperforms the previously discussed group-based delivery. However, optimization of the amount of memory allocated to each file is challenging because of the complicated interplay between the memory allocation parameters and the expected delivery rate. References [47] and [48] use a convex optimization formulation of the memory allocation problem which aims to minimize the expected delivery rate for a given storage capacity per cache. We refer to this problem as Rate Minimization with Storage Constraint (RMSC). Reference [48] has followed a numerical approach to solve the RMSC problem and is mainly focused on reducing the computational complexity of the numerical analysis involved. In contrast, [47] follows a theoretical approach to find structural properties in the optimal solution of the problem.

4.1.2 Our contributions

The results provided in [47] do not capture specific properties of the optimal solution which can considerably facilitate solving the memory allocation problem. In this work, we find such structures in the optimal solution and solve the RMSC problem analytically when user demands are nonuniform and the SCC procedure is used for delivery. In particular, we will show that such properties enable the derivation of the optimal solution based on a search over a finite set of points. The cardinality of this set scales linearly with the product of the number of caches and the number of files. The properties that we derive also provide a unifying interpretation of the optimal placement strategy for both uniform and nonuniform popularity distribution of files, as we will discuss in the remainder of this section.

As the first structural property, we show that for instances of the problem with cache capacities that belong to a finite set $\mathcal{M}$, the optimal placement for RMSC follows the simple pattern of splitting the library of files into two groups. One group consists of less popular files and the files in this group are not cached at all. The files in the second group are cached but are treated as if they had the same request probabilities. We call such instances of RMSC the *base-cases*. We propose an algorithm to derive $\mathcal{M}$ for any given file request distribution, number of caches and number of files.

For instances of the problem that are not among the base-cases, we prove that

the optimal solution is achieved by a convex combination of the solutions to certain base-cases of the problem. This solution is identical to the placement parameters obtained by memory sharing between the two base-cases of the RMSC problem. Memory sharing is already shown to be optimal when demands are uniform [47, Lemma 5] and [48, Theorem 1]. Hence, this result shows that memory sharing is also optimal for nonuniform demands when applied to the appropriately chosen instances of the problem. To elaborate, recall that K, N and M are the number of caches, files in the library and files that each cache can store, respectively. For optimal placement of identically popular files when SCC delivery is used, we have the following [47–49]:

- All files are treated identically during placement, in particular, the same amount of storage is allocated to the caching of each file.
- For a cache size M that corresponds to an integer value of $t = K\frac{M}{N}$, the optimal placement breaks each file into $\binom{K}{t}$ non-overlapping segments. Then, it exclusively stores each one of the segments in exactly one of the $\binom{K}{t}$ subsets of caches that have cardinality t. We refer to these cases of the problem as the base cases and denote by $\mathcal{M}$ the set of corresponding cache sizes $\{0, \frac{1}{K}N, \frac{2}{K}N, \ldots, N\}$.
- For other cache capacities, the optimal placement can be obtained by memory sharing between the optimal placements for two instances of the problem with cache capacities $M_l = \max\{m \in \mathcal{M} \mid m < M\}$ and $M_u = \min\{m \in \mathcal{M} \mid M < m\}$.[1]

We prove that a similar pattern exists in the optimal placement for nonuniform demands. In particular, we propose an algorithm with worst-case complexity of $O(K^2N^2)$ to derive the set $\mathcal{M}$ given a nonuniform popularity distribution for the files. If $M \notin \mathcal{M}$, the optimal placement is obtained by memory sharing between $M_l, M_u \in \mathcal{M}$ as it was done for uniform demands using the derived set $\mathcal{M}$. In this case, optimal placement either follows a two or a three group strategy depending on the specifics of the two corresponding base-cases used.

[1]The idea of memory sharing for uniform demands was presented in [11] as an achievable scheme when t is not an integer. References [47–49] proved that memory sharing is optimal for SCC delivery when demands are uniform.

For the optimal placement that we derive, the memory allocated to different files does not show a gradual and smooth increase as the request probability increases. Instead, for base-cases where the two-group strategy is optimal, the memory allocation exhibits a binary behavior, i.e., as the request probability increases the amount of memory allocated to the files shows an abrupt increase from zero to a new level at a certain request probability and remains at that level thereafter. A similar trend exists for non base-cases, but there might be two thresholds on request probabilities where the jumps in the memory allocated to files occur.

Finally, we find the results in [15] closely connected to the results that we derive in this paper. Reference [15] considers the setting of randomized placement algorithms and within that setting, it shows that a two-group (or occasionally a three-group) strategy guarantees a delivery rate within a constant factor of the information theoretic lower bound. In this work, we derive a deterministic placement of files in the caches which solves the RMSC problem, i.e., we analytically prove its optimality when the SCC delivery algorithm is applied to all user demands. We prove that the expected delivery rate of our optimal solution is a lower-bound on the expected delivery rates of the group-based methods that apply the SCC algorithm within each group of requested files for delivery. Given that the coded caching in [15] follows such a scheme and its rate is within a constant factor from the information-theoretic lower bound, it concludes that the delivery rate of RMSC is also within a constant factor of the optimum delivery rate. Through numerical examples we show that the grouping of files given by these two schemes and the delivery rates resulting from them can be significantly different for specific regimes of problem parameters. Further, we compare the expected rate of RMSC to the information-theoretic outer-bound on the expected rate of caching schemes with uncoded prefetching derived in [50]. This comparison suggests that the expected rate of RMSC approaches the outer-bound as the cache size increases. We provide a detailed discussion to show that the existence of this performance gap is, at least partially, due to an inefficiency in the SCC algorithm for delivery. We suggest directions for future research in order to reduce or fully close this performance gap.

The remainder of this chapter is organized as follows. The setup of the problem and formulation of the expected rate and the utilized storage in terms of place-

ment parameters are presented in Section 4.2. The RMSC problem is formulated in Section 4.3 and a duality framework is proposed for it. Structures in the optimal solution of RMSC for the base-cases are derived in Section 4.4. In Section 4.5, we propose an algorithm to identify the base-cases of the RMSC problem for any given popularity distribution of files and we derive the optimal solution of RMSC.

4.2 Problem setup and formulation of utilized storage and expected delivery rate

4.2.1 Problem Setup

Consider the canonical network of multiple caches and a central server as described in Section 2.1. We use Notation 2 to describe an uncoded placement of content in the caches. Further, similar to the demand vector $\boldsymbol{d} = [d_1, \ldots, d_K]$ which represents the demands of all caches at the current time instant, we define the subdemand vector $\boldsymbol{d}_{\mathcal{S}}$ that determines the files requested by the caches in $\mathcal{S} \subset [K]$, in the same order as in $\boldsymbol{d}$.

We assume that requests for different files are independent and the request probabilities do not change for different caches. Let $\{p_n\}_{n\in[N]}$ represent the request probabilities of the files. Here, files are sorted in the decreasing order of request probabilities, i.e., $n > m$ implies $p_n \leq p_m$. We refer to the file request probabilities as popularity distribution.

We are interested in minimizing the expected delivery rate $\mathbb{E}_{\boldsymbol{d}}\left(R(\boldsymbol{d}; \mathcal{P}, A_D)\right)$, where the expectation is over the randomness in the demand vector $\boldsymbol{d}$.

4.2.2 Delivery algorithm

In this work, we apply the SCC procedure in Algorithm 4 to all user demands for delivery regardless of their popularity levels and the memory allocated to them.

By following Algorithm 4, the server transmits messages of the form

$$\oplus_{k\in\mathcal{S}} X^{d_k}_{\mathcal{S}\setminus k} \tag{4.1}$$

for every nonempty $\mathcal{S} \subset [K]$. All the components $X_{\mathcal{S}\setminus k}$ embedded in the message

are zero-padded to the length of the largest component. Hence, the length of the message is $\max_{k\in\mathcal{S}} |X^{d_k}_{\mathcal{S}\backslash k}|$.[2]

As mentioned in Section 4.1.1, Algorithm 4 contrasts the delivery schemes in [14, 15, 51] which are also based on the SCC procedure but separately apply it to the files with *close* request probabilities. Algorithm 4 has the advantage that it allows coding among all files regardless of their request probabilities and can result in a smaller delivery rate. To elaborate, message (4.1) delivers every subset of bits in $\{X^{d_k}_{\mathcal{S}\backslash k}\}_{k\in\mathcal{S}}$ to the corresponding requesting cache in $\mathcal{S}$. Given a grouping of files into groups $l = 1, \ldots, L$, if instead of applying the SCC to the whole demand vector we applied it to subdemand vectors consisting of files in the same popularity group, the exact same subfiles delivered by (4.1) would have been delivered through the set of messages $\left\{\oplus_{k\in\hat{\mathcal{S}}_l} X^{d_k}_{\mathcal{S}\backslash k}\right\}_{l=1}^{L}$ where $\hat{\mathcal{S}}_l = \{k \in \mathcal{S} | d_k \in l\text{-th group}\}$. This message has length $\sum_{l=1}^{L} \max_{k\in\hat{\mathcal{S}}_l} |X^{d_k}_{\mathcal{S}\backslash k}|$[3] which is lower bounded by $\max_{k\in\mathcal{S}} |X^{d_k}_{\mathcal{S}\backslash k}|$ which is the length of (4.1) with SCC applied to the whole demand vector. A direct consequence of this argument is that with an optimal placement for Algorithm 4, its delivery rate would be a lower-bound on the delivery rates of caching schemes like the ones in [14, 15, 51] which apply Algorithm 4 to subdemand vectors that consist of files that are in identical popularity groups. In particular, the fact that the delivery rate of [15] is within a constant factor of the information-theoretic lower-bound [15, Section III] implies that the delivery rate of Algorithm 4 with the optimal placement that we derive here is also within a constant factor of the information-theoretic minimum rate.

4.2.3 Formulation of expected delivery rate and storage

To derive optimal placement for SCC delivery, we need to formulate the expected delivery rate and the storage used by the placement algorithm in terms of the placement parameters $x^n_{\mathcal{S}}$.

[2] From a graph theoretic perspective, this message corresponds to XORing the requested subfiles that form a clique in the side information graph [21, Section II.A] and [7, Section I.A]. Since the set of messages $\oplus_{k\in\mathcal{S}} X^{d_k}_{\mathcal{S}\backslash k}$ delivers all the missing subfiles, it covers all the vertices in the side information graph. Hence, one can see the delivery procedure of [12] as a structured way of covering the side information graph vertices with cliques.

[3] This is because the files within each group have the same placement parameters and therefore $\max_{k\in\hat{\mathcal{S}}_l} |X^{d_k}_{\mathcal{S}\backslash k}| = |X^{d_k}_{\mathcal{S}\backslash k}|, k \in \mathcal{S}$.

Expected delivery rate

For Algorithm 4 as the delivery algorithm, the delivery load is

$$R(\boldsymbol{d};\boldsymbol{x}) = \sum_{\substack{\mathcal{S}:\mathcal{S}\subset[K]\\ \mathcal{S}\neq\emptyset}} \max_{k\in\mathcal{S}} x_{\mathcal{S}\backslash k}^{d_k} \tag{4.2}$$

for a given demand vector $\boldsymbol{d}$ and placement parameters $x_{\mathcal{S}}^n$. To formulate the expected delivery rate in terms of the placement parameters, let $R_{\mathcal{S}}(\boldsymbol{d};\boldsymbol{x}) = \max_{k\in\mathcal{S}} x_{\mathcal{S}\backslash k}^{d_k}$ denote the rate required to deliver the subfiles that are exclusively stored in the subset of caches $\mathcal{S}$. Then, the expected rate with respect to randomness in user demands is

$$r(\boldsymbol{x}) \triangleq \mathbb{E}_{\boldsymbol{d}}\left(R(\boldsymbol{d};\boldsymbol{x})\right) = \sum_{\substack{\mathcal{S}:\mathcal{S}\subset\mathcal{K}\\ \mathcal{S}\neq\emptyset}} \mathbb{E}_{\boldsymbol{d}}(R_{\mathcal{S}}(\boldsymbol{d};\boldsymbol{x})).$$

We assumed that the popularity distribution of files is the same for different caches. We use this symmetry in the demand probabilities of the different caches to simplify the placement formulation by setting $x_{\mathcal{S}}^n = x_s^n$ for all $\mathcal{S} : |\mathcal{S}| = s$. In other words, for a given file, the portion of bits that is exclusively cached in any subset of caches $\mathcal{S}$ with cardinality s is the same.

Proposition 3. *The assumption $x_{\mathcal{S}}^n = x_s^n$ for all $\mathcal{S} : |\mathcal{S}| = s$ is without loss of optimality for the* RMSC *problem.*

Proof. See Appendix A.1. □

Because of the symmetric structure of the placement, $\mathbb{E}_{\boldsymbol{d}}(R_{\mathcal{S}}(\boldsymbol{d};\boldsymbol{x}))$ is the same for all $\mathcal{S} : |\mathcal{S}| = s$, and it can be denoted by $\bar{R}_s(\boldsymbol{x})$. Hence, the average rate can be written as

$$r(\boldsymbol{x}) = \sum_{s=1}^{K} \binom{K}{s} \bar{R}_s(\boldsymbol{x}).$$

Let $\mathcal{G}_s$ be the set of all subsets of $[N]$ with cardinality at most s. Let π_s^g denote the

probability that files $g \in \mathcal{G}_s$ be requested by a set of caches $\mathcal{S}$ with $|\mathcal{S}| = s$. Then,

$$\bar{R}_s(\boldsymbol{x}) = \sum_{g \in \mathcal{G}_s} \pi_s^g \max_{n \in g} x_{s-1}^n$$

and therefore, the expected delivery rate is

$$\sum_{s=1}^{K} \binom{K}{s} \sum_{g \in \mathcal{G}_s} \pi_s^g \max_{n \in g} x_{s-1}^n,$$

which can equivalently be written as

$$\sum_{s=0}^{K-1} \binom{K}{s+1} \sum_{g \in \mathcal{G}_{s+1}} \pi_{s+1}^g \max_{n \in g} x_s^n.$$

Storage used by placement

The total storage used by cache $k \in [K]$ is

$$\sum_{n=1}^{N} \sum_{\substack{\mathcal{S} \subset [K]: \\ k \in \mathcal{S}}} x_{\mathcal{S}}^n, \tag{4.3}$$

where under the symmetry condition $x_{\mathcal{S}}^n = x_s^n$ for all $\mathcal{S} : |\mathcal{S}| = s$, this quantity simplifies to

$$\sum_{n=1}^{N} \sum_{s=1}^{K} \binom{K-1}{s-1} x_s^n$$

for every cache. The inner sum is the storage that is assigned to file n in each cache, as for each file n, each cache k stores the subfiles $X_{\mathcal{S}}^n : k \in \mathcal{S}$. There are $\binom{K-1}{s-1}$ subsets of $[K]$ of cardinality s with this property for each file. The outer sum adds up the storage used for all the files in the library.

Change of variable for placement parameters

For simpler exposition of the optimization problems and better interpretability of the results, we find it useful to use the change of variable

$$y_s^n = \binom{K}{s} x_s^n. \tag{4.4}$$

Variable y_s^n is the total portion of bits of file n that is cached exclusively in all the $\binom{K}{s}$ different subsets of $[K]$ with cardinality s. As argued in Section 4.2.1, we have $\sum_{\mathcal{S}\subset[K]} x_{\mathcal{S}}^n = 1$. Given that $x_{\mathcal{S}}^n = x_{|\mathcal{S}|}^n$ and using the change of variable (4.4), it follows that

$$\sum_{s=0}^{K} y_s^n = 1, \qquad \forall n \in [N]. \tag{4.5}$$

As a result, the expected rate and storage can be formulated as functions of the new placement parameters y_s^n as

$$r(\boldsymbol{y}) = \sum_{s=0}^{K-1} \frac{K-s}{s+1} \sum_{g\in\mathcal{G}_{s+1}} \pi_{s+1}^g \max_{n\in g} y_s^n, \tag{4.6}$$

$$m(\boldsymbol{y}) = \sum_{n=1}^{N}\sum_{s=1}^{K} \frac{s}{K} y_s^n. \tag{4.7}$$

Notice that the expected rate and the amount of storage used are a convex and a linear function of the placement parameters, respectively.

4.3 Formulation of RMSC and characterization of its dual problem

4.3.1 Formulation of RMSC in terms of the placement parameters

Using (4.4)-(4.7), the problem of finding the storage parameters that minimize the expected delivery rate under the cache capacity constraint can be formulated as

$$\min_{\boldsymbol{y}} \quad \sum_{s=0}^{K-1} \frac{K-s}{s+1} \sum_{g \in \mathcal{G}_{s+1}} \pi_{s+1}^{g} \max_{n \in g} y_s^n \tag{4.8a}$$

$$\text{s.t.} \quad \sum_{n=1}^{N} \sum_{s=1}^{K} \frac{s}{K} y_s^n \leq M, \tag{4.8b}$$

$$\sum_{s=0}^{K} y_s^n = 1, \qquad n \in [N], \tag{4.8c}$$

$$y_s^n \geq 0, \qquad n \in [N],\ s = 0, \ldots, K. \tag{4.8d}$$

4.3.2 Duality framework and derivation of Joint Rate and Storage Minimization problem

Optimization problem (4.8) is convex and Slater's condition holds for it as all inequality constraints are affine [43, Section 5.2.3]. Hence, with (4.8) as primal, the duality gap between the primal and the corresponding dual problem is zero [43, Section 5.2]. To derive the dual problem, we form the Lagrangian that accounts for the capacity constraint (4.8b) as

$$L(\boldsymbol{y}, \gamma) = \sum_{s=0}^{K-1} \frac{K-s}{s+1} \sum_{g \in \mathcal{G}_{s+1}} \pi_{s+1}^{g} \max_{n \in g} y_s^n + \gamma \left(\sum_{n=1}^{N} \sum_{s=1}^{K} \frac{s}{K} y_s^n - M \right)$$

which results in the Lagrange dual function

$$\begin{aligned} g(\gamma) = \min_{\boldsymbol{y}} \quad & L(\boldsymbol{y}, \gamma) \\ \text{s.t.} \quad & \sum_{s=0}^{K} y_s^n = 1,\ n \in [N], \\ & y_s^n \geq 0, \qquad n \in [N],\ s = 0, \ldots, K. \end{aligned} \tag{4.9}$$

Then, the corresponding dual problem will be

$$\max_{\gamma \geq 0} \quad g(\gamma). \tag{4.10}$$

By dropping the terms that are independent of the placement parameters, (4.9) has the same minimizers as

$$\min_{\boldsymbol{y}} \quad \sum_{s=0}^{K-1} \frac{K-s}{s+1} \sum_{g \in \mathcal{G}_{s+1}} \pi_{s+1}^{g} \max_{n \in g} y_s^n + \gamma \sum_{n=1}^{N} \sum_{s=1}^{K} \frac{s}{K} y_s^n \tag{4.11a}$$

$$\text{s.t.} \quad \sum_{s=0}^{K} y_s^n = 1, \, n \in [N], \tag{4.11b}$$

$$y_s^n \geq 0, \qquad n \in [N], \, s = 0, \ldots, K. \tag{4.11c}$$

We call (4.11) the Joint Rate and Storage Minimization (JRSM) problem, as the objective is to minimize the total bandwidth (expected delivery rate) and storage cost of coded caching. Following the standard interpretation of the Lagrange multipliers, parameter γ can be viewed as the relative cost of storage per file. Moreover, since strong duality holds, for each storage capacity M, the optimal dual variable $\gamma^*(M)$ determines a pricing of the storage for which there exists the same minimizer to both the RMSC problem (4.8) and the Lagrangian minimization problem in (4.9) (or equivalently the JRSM problem in (4.11)). As a result, we derive the optimal solution of JRSM in Section 4.4 as an intermediate step in solving RMSC.

4.4 Optimal solution to JRSM

Finding an analytical solution to (4.11) is challenging because of the presence of the $\max$ functions that operate over overlapping sets of parameters in the objective. These parameters are tied together by constraints (4.11b) for different values of s. The interplay between the outputs of the $\max$ function applied to the overlapping groups under constraints (4.11b) makes the analysis difficult. To facilitate the analysis, we establish a connection between the nonlinear part of (4.11a) and submodular set functions. This allows us to benefit from the results in submodular function analysis to find structures in the optimal solution to JRSM.

4.4.1 An equivalent formulation of JRSM

The placement parameters corresponding to $s = 0$ are $\{y_0^n\}_{n\in[N]}$, which determine the portion of bits that are not stored in any cache for each file n. Also, each set $g \in \mathcal{G}_1$ includes exactly one file, say $g = \{i\}$. Hence, $\max_{n\in g} y_0^n = y_0^i$ and $\pi_0^g = p_i$. Thus, the objective function (4.11a) can be written as

$$\sum_{n=1}^{N} Kp_n y_0^n + \sum_{s=1}^{K-1}\left[\frac{K-s}{s+1}\sum_{g\in\mathcal{G}_{s+1}} \pi_{s+1}^g \max_{n\in g} y_s^n + \frac{s}{K}\gamma\sum_{n=1}^{N} y_s^n\right] + \gamma\sum_{n=1}^{N} y_K^n.$$

Notice that the first and last sums are in terms of parameters y_0^n and y_K^n, respectively, while the summation in the middle accounts for parameters y_s^n for $s \in [K-1]$.

Lemma 1. *At optimality,* $\sum_{n=1}^{N} Kp_n y_0^n + \gamma\sum_{n=1}^{N} y_K^n$ *can be written as* $\sum_{n=1}^{N}(Kp_n\alpha_n + \gamma(1-\alpha_n))z^n$ *where* $z^n = y_0^n + y_K^n$*, and* $\alpha_n = 1$ *if* $Kp_n < \gamma$ *and* $\alpha_n = 0$ *if* $Kp_n \geq \gamma$.

Proof. For a fixed value of z^n, we have $\sum_{n=1}^{N} Kp_n y_0^n + \gamma\sum_{n=1}^{N} y_K^n = \gamma\sum_{n=1}^{N} z^n + \sum_{n=1}^{N}(Kp_n - \gamma)y_0^n$. Hence, if $Kp_n < \gamma$, setting $y_0^n = z^n$ and $y_K^n = 0$ leads to the smallest objective and if $Kp_n \geq \gamma$, the smallest objective results for $y_0^n = 0$ and $y_K^n = z^n$. □

Corollary 1. *For some* $m \in \{0,\ldots,N\}$*, we have* $\alpha_n = 1, n > m$ *and* $\alpha_n = 0, n \leq m$.

Using Lemma 1, and the fact that $z^n = 1 - \sum_{s=1}^{K-1} y_s^n$, we get

$$\min_{\tilde{\boldsymbol{y}},\boldsymbol{\alpha}} \quad \sum_{s=1}^{K-1}\frac{K-s}{s+1}\sum_{g\in\mathcal{G}_{s+1}}\pi_s^g\max_{n\in g} y_s^n + \sum_{n=1}^{N}\sum_{s=1}^{K-1}\left[\left(\frac{s}{K} - 1 + \alpha_n\right)\gamma - Kp_n\alpha_n\right]y_s^n + l(\boldsymbol{\alpha}) \tag{4.12a}$$

$$\text{s.t.} \quad \sum_{s=1}^{K-1} y_s^n \leq 1,\ n \in [N], \tag{4.12b}$$

$$y_s^n \geq 0, \qquad n \in [N],\ s \in [K-1],, \tag{4.12c}$$

$$\alpha_n \in \{0,1\},\ n \in [N], \tag{4.12d}$$

as a problem equivalent to (4.11), where $\tilde{\boldsymbol{y}}$ is the same as $\boldsymbol{y}$, except for parameters y_0^n and y_K^n that are removed, and $l(\boldsymbol{\alpha}) = K\sum_{n=1}^{N}\alpha_n p_n + \gamma\sum_{n=1}^{N}(1-\alpha_n)$.

To find structures in the optimal vector $\tilde{\boldsymbol{y}}$, assume that the optimal parameters α_n^* are known. Then, the optimization problem for $\tilde{\boldsymbol{y}}$ becomes

$$\min_{\tilde{\boldsymbol{y}},\, t} \quad t + \sum_{n=1}^{N}\sum_{s=1}^{K-1}\Big[\Big(\frac{s}{K} - 1 + \alpha_n^*\Big)\gamma - Kp_n\alpha_n^*\Big]y_s^n \tag{4.13a}$$

$$\text{s.t.} \quad \sum_{s=1}^{K-1}\frac{K-s}{s+1}\sum_{g\in\mathcal{G}_{s+1}}\pi_s^g \max_{n\in g}|y_s^n| \le t, \tag{4.13b}$$

$$\sum_{s=1}^{K-1} y_s^n \le 1, \qquad n\in[N], \tag{4.13c}$$

$$y_s^n \ge 0, \qquad n\in[N],\ s\in[K-1]. \tag{4.13d}$$

In constraint (4.13b), we used $\max_{n\in g}|y_s^n|$, which is the l_∞-norm instead of $\max_{n\in g} y_s^n$. This does not affect the optimal solution as the two functions are equivalent in the nonnegative orthant specified by (4.13d) but makes the LHS in form of the l_∞-norm in Proposition 4 of Section 4.4.2.

Notice that objective function (4.13a) is linear, and both the objective function and the constraints are in terms of parameters y_s^n for $s\in[K-1]$, $n\in[N]$. For a linear objective function, if the feasible set is convex and bounded with a finite number of extreme points, then there exists an extreme point that is optimal [52, Section 2.5]. In the following, we will show that the feasible set defined by (4.13b)-(4.13d) satisfies these properties for any given value of t, and in particular for t^* at optimality, and derive structures in its extreme points. Any such structure readily implies a structure in at least one optimal solution to (4.13).

4.4.2 Review of submodular functions and relevant results

We review the definition of a submodular set function and present the results that are related to our analysis in Section 4.4. An extended discussion can be found in [53].

Definition 1. *Let $V = \{1,\dots,p\}$ be a set of p objects. For $\boldsymbol{w}\in\mathbb{R}^p$, Supp$(\boldsymbol{w})\subset V$*

denotes the support of $\boldsymbol{w}$, defined as Supp$(\boldsymbol{w}) = \{j \in V, w_j \neq 0\}$.

Definition 2. ***(Submodular function)*** *A set-function $F : 2^V \to \mathbb{R}$ is submodular if and only if, for all subsets $A, B \subset V$, we have $F(A) + F(B) \geq F(A \cup B) + F(A \cap B)$.*

Definition 3. ***(Lovász extension)*** *Given a set-function F such that $F(\emptyset) = 0$, the Lovász extension $f : \mathbb{R}^p_+ \to \mathbb{R}$ of F is defined as*

$$f(\boldsymbol{w}) = \sum_{k=1}^{p} w_{j_k} \left[F(\{j_1, \ldots, j_k\}) - F(\{j_1, \ldots, j_{k-1}\})\right],$$

where $\boldsymbol{w} \in \mathbb{R}^p_+$, $(j_1, \ldots, j_p)$ is a permutation such that $w_{j_1} \geq \ldots \geq w_{j_p}$.

Consider vector $\boldsymbol{\delta} \in \{0, 1\}^p$ as the indicator vector for subset $A \subset V$, i.e, for $i \in V$, $\delta_i = 1$ if and only if $i \in A$. Consequently, A is the support of $\boldsymbol{\delta}$. Notice that for the Lovász extension, $f(\boldsymbol{\delta}) = F(\text{Supp}(\boldsymbol{\delta}))$. Hence, f can be seen as an extension of F from vectors in $\{0, 1\}^p$ to all vectors in $\mathbb{R}^p_+$. The Lovász extension f has the following properties: 1) it is piecewise-linear, 2) when F is submodular, f is convex, and 3) minimizing F over subsets, i.e., minimizing f over $\{0, 1\}^p$, is equivalent to minimizing f over $[0, 1]^p$.

Definition 4. ***(Stable Sets)*** *A set A is stable if it cannot be augmented without increasing F, i.e., if for all sets $B \supset A$, $B \neq A$, then $F(B) > F(A)$.*

Definition 5. ***(Separable Sets)*** *A set A is separable if we can find a partition of A into $A = B_1 \cup \ldots \cup B_k$ such that $F(A) = F(B_1) + \ldots + F(B_k)$. A set A is inseparable if it is not separable.*

Proposition 4. *[53, Section 4.1] For a set of objects V and a nonnegative set-function $d(\cdot)$, let $\Omega(\boldsymbol{w}) = \sum_{G \subset V} d(G) \left\|\boldsymbol{w}_G\right\|_\infty$ for $\boldsymbol{w} \in \mathbb{R}^p$. Function $\Omega(\boldsymbol{w})$ is a norm if $\cup_{G, d(G)>0} G = V$ and it corresponds to the nondecreasing submodular function $F(A) = \sum_{G : A \cap G \neq \emptyset} d(G)$.*

Proposition 5. *[53, Proposition 2]) The extreme points of the unit ball of Ω are the vectors $\frac{1}{F(A)}\boldsymbol{v}$, with $\boldsymbol{v} \in \{-1, 0, 1\}^p$, Supp$(\boldsymbol{v}) = A$ and A a stable inseparable set.*

4.4.3 Connection to submodular functions

To find the extreme points of the region characterized by (4.13b), we establish a link to submodular functions. Let us define function

$$f_{\mathrm{c}}(\tilde{\boldsymbol{y}}) \triangleq \sum_{s=1}^{K-1} \frac{K-s}{s+1} \sum_{g \in \mathcal{G}_{s+1}} \pi_s^g \max_{n \in g} |y_s^n|.$$

The subscript c is used to highlight that this function returns the average rate due to the delivery of the bits that are cached in at least one of the caches in the system. We show that $f_{\mathrm{c}}(\tilde{\boldsymbol{y}})$ is the Lovász extension of a submodular set function. For that, consider the set $[(K-1)N]$. For each $s \in [K-1]$, objects $(s-1)N+1, \ldots, sN$ correspond to files $1, \ldots, N$, respectively. Notice that these objects belong to $[N]_{(s-1)N}$.

To define the corresponding set function, let us introduce the following for any $s \in [K-1]$ and $g \in \mathcal{G}_{s+1}$:

- Operator $u(s,g)$ that gives the set $\tilde{g} = \{(s-1)N + n \mid n \in g\}$ as output. This is basically a mapping from the files in g and set sizes s to the objects in $[(K-1)N]$. Notice that any resulting set $\tilde{g}$ is a subset of $[N]_{(s-1)N}$ for exactly one s.

- Sets $\tilde{\mathcal{G}}_{s+1} = \{u(s,g) \mid g \in \mathcal{G}_{s+1}\}$ and $\tilde{\mathcal{G}} = \bigcup_{s \in [K-1]} \tilde{\mathcal{G}}_{s+1}$.

- The inverse operators $s^{-1}(\tilde{g})$ and $g^{-1}(\tilde{g})$ that for $\tilde{g} \in \tilde{\mathcal{G}}$ return the unique s that satisfies $\tilde{g} \subset [N]_{(s-1)N}$, and the set $g = \{n \mid s^{-1}(\tilde{g})N + n \in \tilde{g}\}$, respectively.

- Weights

$$\eta_{\tilde{g}} = \frac{K - s^{-1}(\tilde{g})}{s^{-1}(\tilde{g}) + 1} \pi_{s^{-1}(\tilde{g})}^{g^{-1}(\tilde{g})} \tag{4.14}$$

  for all $\tilde{g} \in \tilde{\mathcal{G}}$. Notice that when $|\tilde{g}| = 1$, $g^{-1}(\tilde{g}) = \{i\}$ which is a singleton. In that case, $\pi_{s^{-1}(\tilde{g})}^{g^{-1}(\tilde{g})} = p_i$.

Using the operators and parameters defined above, $f_{\mathrm{c}}(\tilde{\boldsymbol{y}})$ can be written as

$$f_{\mathrm{c}}(\tilde{\boldsymbol{y}}) = \sum_{\tilde{g}\in\tilde{\mathcal{G}}} \eta_{\tilde{g}} \max_{n\in g^{-1}(\tilde{g})} |y^n_{s^{-1}(\tilde{g})}|. \tag{4.15}$$

Notice that (4.15) has the form of the norm function in Proposition 4 and as a direct consequence we have the following proposition:

Proposition 6. *Function $f_c(\tilde{\boldsymbol{y}})$ is a norm and is the Lovász extension of the submodular function*

$$F_c(A) = \sum_{\tilde{g}\in\tilde{\mathcal{G}}: A\cap\tilde{g}\neq\emptyset} \eta_{\tilde{g}}, \tag{4.16}$$

where $A \subset [(K-1)N]$.

From Proposition 6, one concludes that constraint (4.13b) characterizes a norm-ball of radius t.

For $A \subset [N]_{(s-1)N}$, let us define $P(A) = \sum_{n\in g^{-1}(A)} p_n$. Then, for the extreme points of the norm-ball, we have the following lemma.

Lemma 2. *The extreme points of the norm-ball $f_c \leq t$ are of the form*

$$\frac{t}{\frac{K-s}{s+1}\left[1-(1-P(A))^{s+1}\right]}\boldsymbol{v}, \tag{4.17}$$

where vector $\boldsymbol{v} \in \{-1,0,1\}^{KN}$, Supp$(\boldsymbol{v}) = A$, and set A is a subset of $[N]_{(s-1)N}$ for an $s \in [K-1]$.

Proof. See Appendix A.2. □

Corollary 2. *For an extreme point $\tilde{\boldsymbol{y}}$ of the norm-ball defined by (4.13b), for all $y^n_s > 0$, we have $s = \hat{s}$, for exactly one $\hat{s} \in [K-1]$.*

Theorem 1. *There is an optimal solution to the* JRSM *problem in (4.11) which is of form*

$$(y^n_s)^* = \begin{cases} 1, & s = 0, n \in [N-n^*]_{n^*}, \\ 1, & s = s^*, n \in [n^*], \\ 0, & \textit{otherwise}, \end{cases} \tag{4.18}$$

for some $s^* \in [K]$ *and some* $n^* \in \{0, \dots, N\}$.[4]

Proof. See Appendix A.3. □

Theorem 1 implies that for every $\gamma \geq 0$ there exists an optimal solution to the JRSM problem that is integral. For better illustration, we write such an optimal vector $\boldsymbol{y}$ in the matrix form Y as follows. Matrix Y has N rows corresponding to the files and $K+1$ columns corresponding to the cardinality of subsets of caches. In particular, $Y_{n,s} = y_s^n, n \in [N], s = 0, \dots, K$. Based on the structures found for y_s^n in Theorem 1, Y is of form

$$Y = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ n^*-1 \\ n^* \\ n^*+1 \\ \vdots \\ N \end{array} \begin{array}{c} \begin{array}{cccccccc} 0 & 1 & \dots & s^*-1 & s^* & s^*+1 & \dots & K \end{array} \\ \left[\begin{array}{cccccccc} \mathbf{0} & 0 & \dots & 0 & \mathbf{1} & 0 & \dots & 0 \\ \mathbf{0} & 0 & \dots & 0 & \mathbf{1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & 0 & \dots & 0 & \mathbf{1} & 0 & \dots & 0 \\ \mathbf{0} & 0 & \dots & 0 & \mathbf{1} & 0 & \dots & 0 \\ \mathbf{1} & 0 & \dots & 0 & \mathbf{0} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{1} & 0 & \dots & 0 & \mathbf{0} & 0 & \dots & 0 \end{array}\right] \end{array}, \quad (4.19)$$

i.e., i) all entries are either 0 or 1, ii) each row has exactly one entry 1, iii) at most one column with index $s \geq 1$ has nonzero entries, iv) for that column all entries are 1 for rows $1, \dots, n^*$ and 0 for rows $n^*+1, \dots, N$, for some $n^* \in [N]$. As a result, for all values of $\gamma \geq 0$ there exists an optimal solution with matrix form (4.19). Hence, we have the following corollary:

Corollary 3. *There exists a finite set* $\mathcal{Y}^*$ *of vectors that correspond to matrices of form (4.19) where* $|\mathcal{Y}^*| \leq KN + 1$ *and that set includes at least one optimal solution to the* JRSM *problem (4.11) for every* $\gamma \geq 0$.[5]

The structure of the optimal solution to JRSM in Theorem 1 has direct implications about the solution of the RMSC problem. In particular, based on the duality

[4]Notice that $n^* = 0$ corresponds to the case where for all $n \in [N]$: $y_0^n = 1$ and $y_s^n = 0, s > 0$.

[5]We show in Appendix A.5 that for specific values of γ, there are infinite number of solutions to the JRSM problem.

framework detailed in Section 4.3.2, for the optimal Lagrange multiplier γ^*, if the optimal solution to JRSM with the structure in Theorem 1 uses a storage equal to M in RMSC, this solution is also optimal to the RMSC problem. This implies that for certain storage capacities in the RMSC problem, its optimal solution has the same structure as in Theorem 1. In Section 4.5, we fully investigate the solution to the RMSC problem for the general storage capacity M and its connection to the solution of the JRSM problem.

4.5 Optimal solution to RMSC

4.5.1 Optimal solution of RMSC in terms of optimal JRSM solution

Assuming that the optimal dual parameter γ^* is known, we derived structures in the minimizers of the Lagrangian or equivalently in the optimal solution of JRSM. The derived structures limited the search space for the optimal JRSM solution to $KN + 1$ vectors specified by Theorem 1. In this section, we derive the optimal solution to RMSC by building on the results we derived for the solution of JRSM in Theorem 1. For that, let us define two sets as follows:

Definition 6. *Sets $\mathcal{M}$ and $\mathcal{R}$ are finite sets defined by storage values $\{m(\boldsymbol{y}) \mid \boldsymbol{y} \in \mathcal{Y}^*\}$ and expected rates $\{r(\boldsymbol{y}) \mid \boldsymbol{y} \in \mathcal{Y}^*\}$, respectively.*

To characterize the solution of RMSC, we take the following two steps. First, we assume that set $\mathcal{Y}^*$ and consequently set $\mathcal{M}$ are known. Based on this assumption, we derive the optimal dual parameter γ^* as a function of storage capacity M in the primal problem. Second, we use the derived γ^*-M relationship to find set $\mathcal{Y}^*$ and derive the optimal solution to RMSC.

Lemma 3. *The optimal dual parameter γ^* and the storage capacity M in the primal* RMSC *problem satisfy the following:*

1. *Parameter γ^* is non-increasing in M;*
2. *For certain storage capacities M, a continuum of dual parameters γ^* are optimal;*

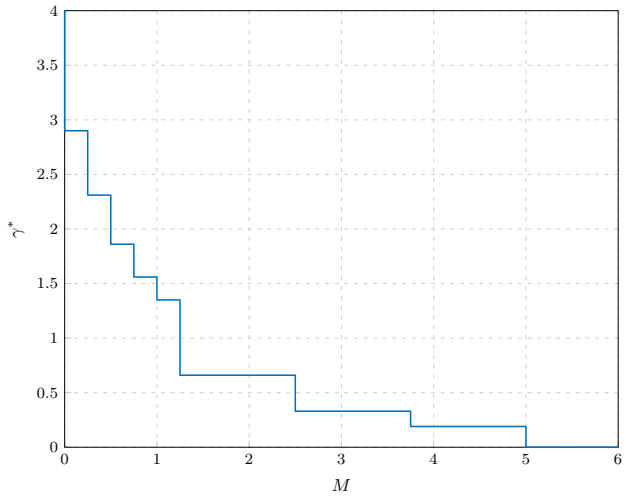


Figure 4.1: Here, $N = 5$, $K = 4$, and the popularity distribution is Zipf with parameter $\alpha = 1$.

3. *For every two consecutive values $M_1, M_2 \in \mathcal{M}, M_1 < M_2$, any $M \in [M_1, M_2]$ leads to the same dual optimal parameter γ^*.*

Proof. See Appendix A.4. □

Lemma 3 implies a stairwise relationship between the optimal dual parameter γ^* and the storage capacity M in the primal problem. An illustration of this relationship is shown in Fig. 4.1. The fact that γ^* is non-increasing in M is in agreement with the interpretation of the Lagrange multiplier γ^* as the (relative) price per unit of storage [43]: as more storage becomes available, the storage price remains the same or decreases. The second point in the lemma corresponds to the vertical line segments in Fig. 4.1. Based on Definition 6, set $\mathcal{M}$ which is derived from $\mathcal{Y}^*$ is finite and has at most $KN+1$ members. However, γ is a continuous variable and for every $\gamma \geq 0$ there is an optimal solution to JRSM in $\mathcal{Y}^*$. Hence, an interval of γ values must map to the same M. Third, a range of values of M are mapped into the same γ^*. Notice that parameter M in the primal problem can take any nonnegative value, while $\mathcal{M}$ is a set of discrete values and is of finite size. Since every $M \geq 0$ corresponds to at least one optimal dual parameter γ, then a continuum of values for M must map to the same γ^*. We show that parameter γ^* and the two solutions

$\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}^*$ that lead to the two endpoints $(m(\boldsymbol{y}_1), \gamma^*)$ and $(m(\boldsymbol{y}_2), \gamma^*)$ of the line segment are related by $\gamma^* = \frac{r(\boldsymbol{y}_1)-r(\boldsymbol{y}_2)}{m(\boldsymbol{y}_2)-m(\boldsymbol{y}_1)}$. Notice that $m(\boldsymbol{y}_1), m(\boldsymbol{y}_2) \in \mathcal{M}$ and $r(\boldsymbol{y}_1), r(\boldsymbol{y}_2) \in \mathcal{R}$. In particular, if we sort members of $\mathcal{M}$ in increasing order as $0 = M_0 < M_1 < M_2 < \ldots < M_l = N$, then rates R_i that correspond to storage values M_i follow ordering $K = R_0 > R_1 > \ldots > R_l = 0$. Hence

$$\gamma^*(M) = \begin{cases} [\frac{R_i-R_{i+1}}{M_{i+1}-M_i}, \frac{R_{i-1}-R_i}{M_i-M_{i-1}}], & M = M_i \\ \frac{R_{i-1}-R_i}{M_i-M_{i-1}}, & M_{i-1} < M < M_i \end{cases}$$

with $\gamma^*(M_0 = 0) = [\frac{K-R_1}{M_1}, +\infty]$ and $\gamma^*(M_l = N) = [0, \frac{R_{l-1}}{N-M_{l-1}}]$.

The next theorem determines the relationship between the optimal solution of RMSC and the optimal solution of JRSM which was derived in Theorem 1.

Theorem 2. *The* RMSC *problem (4.8) has an optimal solution*

$$y^*_{\text{RMSC}}(M) = \begin{cases} y^*_{\text{JRSM}}(M), & M \in \mathcal{M} \\ \frac{M_u-M}{M_u-M_l} y^*_{\text{JRSM}}(M_l) + \frac{M-M_l}{M_u-M_l} y^*_{\text{JRSM}}(M_u), & M \notin \mathcal{M} \end{cases} \tag{4.20}$$

where $y^*_{\text{JRSM}}(m)$ is the optimal solution of JRSM *of the form in Theorem 1 that uses storage M, and M_l and M_u are the largest element smaller than M and smallest element larger than M in $\mathcal{M}$, respectively.*

Proof. See Appendix A.5. □

Remark 1. *(Two or three-group placement is optimal) Given the structure of the* JRSM *solutions in Theorem 1 (and eq. (4.19)), and the connection between the* RMSC *and* JRSM *solutions in Theorem 2, it can be concluded that always a two or at most a three group placement strategy is optimal for* RMSC. *In case $M \in \mathcal{M}$, the placement follows a two-group strategy. In case $M \notin \mathcal{M}$ also two or three-group placement is optimal. This is because a linear combination of two solutions of the form in eq. (4.19) can have non-zero entries at only the first column and at most two other columns. Following the notation used in eq. (4.19), let the two* JRSM *solutions have non-zero entries as characterized by (s^*_1, n^*_1) and (s^*_2, n^*_2). Then, the linear combination of the two solutions results in a matrix where files are grouped into at most three groups (files within each group have the same placement*

parameters or equivalently the corresponding rows are the same): the ones in rows 1 to $\min(n_1^*, n_2^*)$, $\min(n_1^*, n_2^*) + 1$ *to* $\max(n_1^*, n_2^*)$ *and* $\max(n_1^*, n_2^*) + 1$ *to* N. *In case* $n_1^* = n_2^*$, *this reduces into two groups.*

Remark 2. *(Optimality of memory sharing) Theorem 2 essentially extends a result known for the optimal solution of* RMSC *for uniform demands to the general case where demands can be nonuniform. To elaborate, it has been shown that for uniform demands, if* $M \in \{1, \frac{N}{K}, 2\frac{N}{K}, \ldots, K\frac{N}{K}\}$, *then the optimal solution of* RMSC *is in the form in (4.19) for some* $s^* \in [K]$ *and* $n^* = N$ *[47–49]. In particular, for uniform demands* $\mathcal{M}_{uniform} = \{0, \frac{N}{K}, 2\frac{N}{K}, \ldots, K\frac{N}{K}\}$[6]. *For other values of* M, *the optimal solution could be obtained by memory sharing between the two values of storage in* $\mathcal{M}_{uniform}$ *closest to* M. *Theorem 2 shows that the same result is valid for nonuniform demands except for the fact that* n^* *might be any value between* 0 *and* N, *depending on the popularity distribution of files.*

As a result, we propose the following terminology:

Definition 7. *For a given number of caches and popularity distribution of files, we call set* $\mathcal{M}$ *the set of base-cases of the* RMSC *problem.*

Based on Theorem 1, for base-cases of RMSC, there exists an optimal solution which is integral. Also, from Theorem 2, for other storage capacities, the optimal solution to RMSC can be obtained by memory sharing between the solutions of two certain base-cases.

4.5.2 Algorithm to derive the set of base-case memory sizes $\mathcal{M}$

We derive an algorithm with a worst-case complexity of $O(K^2N^2)$ to find set $\mathcal{Y}^*$ and consequently $\mathcal{M}$ for any given number of caches and popularity distribution of files. With $\mathcal{M}$ determined, Theorem 2 analytically solves the RMSC problem for any cache capacity.

To find $\mathcal{Y}^*$, we need to search over the $KN + 1$ possibilities for $\boldsymbol{y}^*$ of form (4.18). For each such vector $\boldsymbol{y}$, the storage it uses can be written as a convex combination of the storage used by two other vectors $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ that satisfy $m(\boldsymbol{y}_1) \leq$

[6] In the case of $M = 0$, we have $n^* = 0$ for the optimal RMSC solution.

Algorithm 7 Procedure to Determine the Set of Base-Cases $\mathcal{M}$

Require: $K, N, \{p_n\}$

Calculate Storage and Rate for the $KN+1$ matrices of form (4.19)
$Y_0 \leftarrow \mathbf{0}_{N\times(K+1)}$, $Y_0(1:N,0) \leftarrow 1$, $M_0 \leftarrow 0$, $R_0 \leftarrow K$
for $s = 1, \ldots, K$ **do**
  for $n = 1:N$ **do**
    $i \leftarrow (s-1)N + n$
    $Y_i \leftarrow \mathbf{0}_{N\times(K+1)}$, $Y_0(n+1:N,0) \leftarrow 1$, $Y_i(1:n,s) \leftarrow 1$
    $M_i \leftarrow m(Y)$
    $R_i \leftarrow r(Y)$
  end for
end for
$(M, R, Y) \leftarrow \text{sort}_M(M, R, Y)$ # relabel (M_i, R_i, Y_i) tuples in increasing order of M_i
Build $\mathcal{M}$, $\mathcal{R}$ and $\mathcal{Y}$ by keeping solutions that outperform memory sharing between other cases
$(\mathcal{Y}_0, \mathcal{M}_0, \mathcal{R}_0) \leftarrow (\mathbf{0}_{N\times(K+1)}, 0, K)$
$c \leftarrow 0$
for $i = 1, \ldots, NK+1$ **do**
  base $\leftarrow$ True
  for $j = i+1 : NK+1$ **do**
    $R_{\text{msh}} \leftarrow \frac{M_j - M_i}{M_j - \mathcal{M}_c}\mathcal{R}_c + \frac{M_i - M_j}{M_j - \mathcal{M}_c}R_j$
    if $R_{\text{msh}} <= R_i$ **then**
      base $\leftarrow$ False # If memory-sharing results in a smaller or the same rate, (M_i, R_i, Y_i) tuple does not represent a base case
      break
    end if
  end for
  if base **then**
    $c \leftarrow c + 1$
    $(\mathcal{Y}_c, \mathcal{M}_c, \mathcal{R}_c) \leftarrow (Y_i, M_i, R_i)$
  end if
end for

$m(\boldsymbol{y})$ and $m(\boldsymbol{y}_2) \geq m(\boldsymbol{y})$. In other words, $m(\boldsymbol{y}) = m(\theta\boldsymbol{y}_1 + (1-\theta)\boldsymbol{y}_2), 0 \leq \theta \leq 1$.[7] If for such $\boldsymbol{y}_1, \boldsymbol{y}_2$, we further have $r(\boldsymbol{y}) \geq r(\theta\boldsymbol{y}_1 + (1-\theta)\boldsymbol{y}_2)$, then $\boldsymbol{y}$ does not belong to $\mathcal{Y}^*$. Hence, by removing such vectors form the $KN+1$

[7] except for the two vectors with $m(\boldsymbol{y}) \in \{0, N\}$.

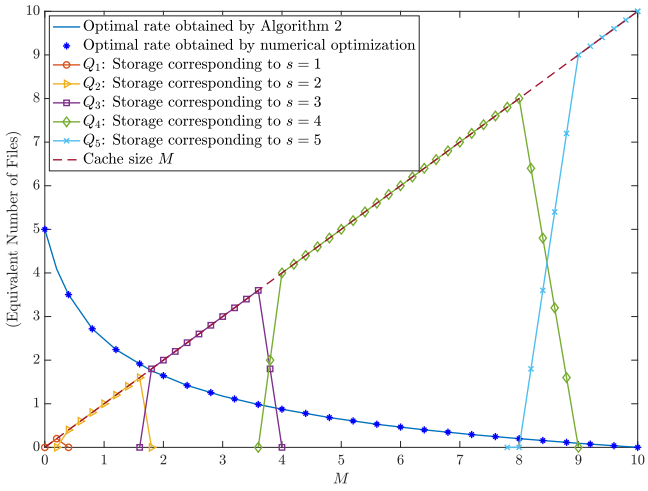


Figure 4.2: The effect of cache size on expected delivery rate and the amount of storage used to cache subsets of files that are exclusively stored in subsets of caches with cardinalities $1, \ldots, K$ for $K = 5$, $N = 10$. Here, the popularity of files follows a Zipf distribution with parameter 1.4.

possibilities, the remaining set of vectors constitutes $\mathcal{Y}^*$. The BASE procedure in Algorithm 7 implements this process by starting from Y with all entries in the first column equal to 1. This correspond to storage 0 and rate K and belongs to $\mathcal{Y}^*$. It then proceeds to the next $\boldsymbol{y}$ with the smallest storage value. It checks whether it outperforms memory sharing between the $\boldsymbol{y}$ that is already in $\mathcal{Y}^*$ with the largest storage and every remaining vector that uses more storage compared to $\boldsymbol{y}$. If that is the case, it adds the new vector to $\mathcal{Y}^*$, otherwise drops the vector and proceeds to the next vector.

Fig. 4.2 shows the expected delivery rate of the proposed method versus the cache capacity for a nonuniform distribution of files that follows a Zipf density [45] with parameter 1.4. The expected rate is once calculated based on the solution obtained by Algorithm 7 and once by solving RMSC numerically. We observe that the resulting optimal rates are in complete agreement. Fig. 4.2 also shows the amount of storage used to cache subsets of files that are exclusively stored in subsets of caches with different cardinalities $s \in [K]$. In other words, for each s,

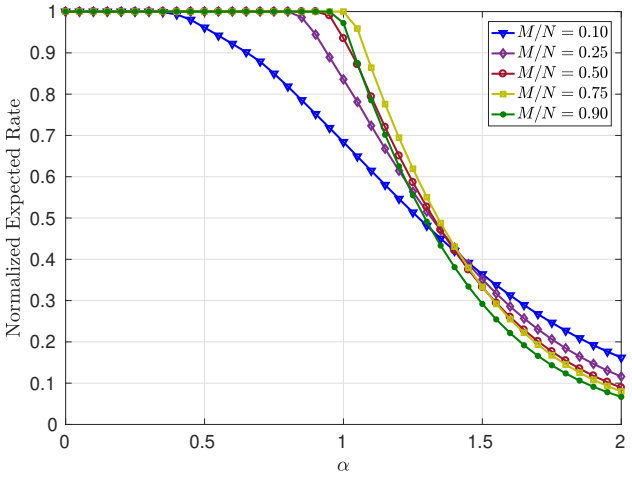
it shows $Q_s \triangleq \sum_{n=1}^{N} \frac{s}{K} y_s^n$ as a function of the cache capacity. As we expect from our analysis, either one or two values of Q_s can be positive for each choice of M.

4.5.3 Numerical exploration

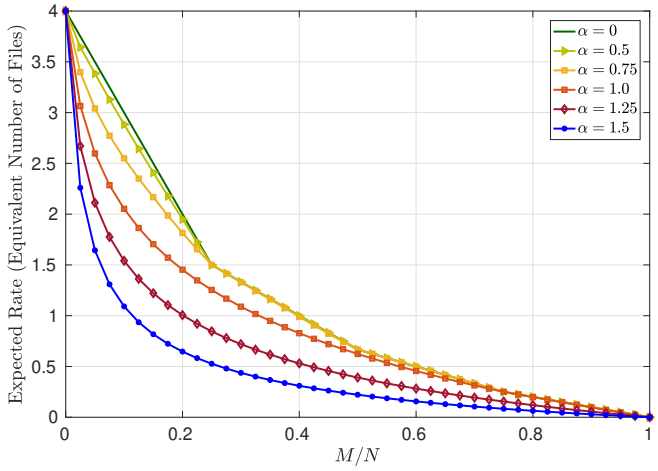
Fig. 4.3 shows the joint effect of the nonuniformity of the file request probabilities and the cache size M. The nonuniformity of the probability mass function is controlled by parameter α of Zipf distribution [45]. Fig.4.3a shows the expected rate of RMSC for $0 \leq \alpha \leq 2$ normalized by the expected rate of SCC for the corresponding value of α when the the placement for uniform demands is used. This normalization puts the curves for different cache sizes in the same scale for better visual clarity and more significantly makes the curves more interpretable. In particular, we observe that for a fixed cache size, the normalized expected rate remains almost equal to 1 as α is increased from 0 up to some threshold value. In other words, the optimal the delivery rate for a slightly nonuniform popularity distribution is almost identical to the delivery rate given by the placement that treats the files as if they were uniformly popular. The threshold value of α depends on the available cache capacity. In general, as the Zipf distribution gets more heavy-tailed (smaller α) the normalized rate gets closer to 1. Fig. 4.3b shows the expected delivery rate of RMSC versus cache size for different values of α. Consistent with our previous observation, for heavy-tailed popularity distributions, like the cases with $\alpha = 0, 0.5, 0.75$, the optimal delivery rates are close to each other and are only different when cache storage is scarce ($M/N \leq 0.25$).

Reference [15] also proposed the use of a two-group or three-group strategy for the caching of files with nonuniform demands. Fig. 4.4 compares the delivery rates obtained by RMSC to those obtained by [15]. Accordingly, Table 4.1 provides the groupings of the files made by the two techniques, where n^* and N_1 represent the index of the last popular files given by RMSC and [15], respectively.[8] Accordingly, R^* and R_1 represent the expected delivery rates of the two techniques. The superior

[8]For the caching technique in [15], it is possible for N_1 to be smaller than M. In Table 4.1, this is the case for $M = 4$ ($M/N = 0.1$). In such cases, part of the memory would remain unused if only the popular files were to be cached. In such a scenario, the authors suggest that each cache stores the entirety of the first M files, and if M is not an integer, use the remaining $M - \lfloor M \rfloor$ capacity for the partial caching of file $\lfloor M \rfloor + 1$. In this scenario, uncoded messages are used to deliver the files that are not fully cached. For more details refer to [15, Section V].



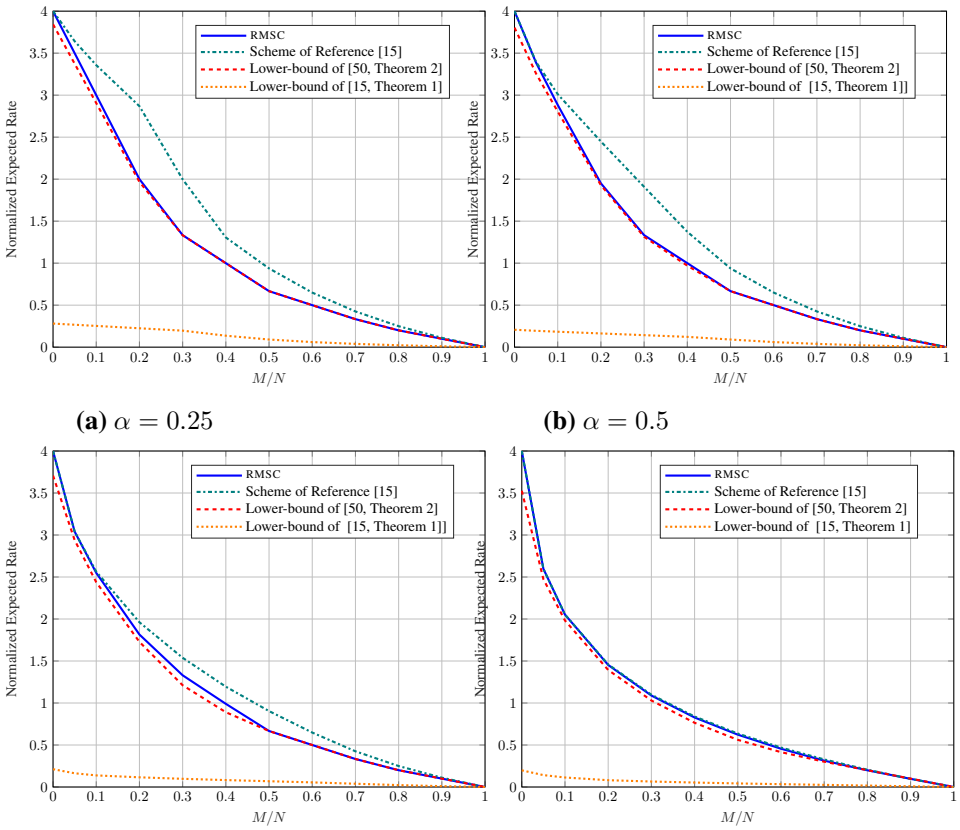
(a) Normalized expected rate versus the Zipf parameter for different cache sizes



(b) Expected rate versus the normalized cache size for different Zipf parameters

Figure 4.3: Joint effect of cache size and nonuniformity of the file request probabilities on the expected delivery rate. In (a), the expected rates are normalized by the expected rate of the delivery algorithm SCC when the placement for uniform demands is used instead of the optimal placement. Here $K = 4$ and $N = 40$.

performance of RMSC is evident in both Fig. 4.4 and Table 4.1. In Table 4.1, one observes that the set of popular files given by RMSC and [15] are identical for specific set of problem parameters but the delivery rates are different. This is because despite the identical grouping of files, the placement of the popular files and the delivery algorithms are still different for the two techniques. In particular, [15] follows a randomized placement algorithm while RMSC uses a deterministic placement technique. The randomized algorithm simplifies the placement at the expense of a higher delivery rate.



(a) $\alpha = 0.25$ **(b)** $\alpha = 0.5$

(c) $\alpha = 0.75$ **(d)** $\alpha = 1$

Figure 4.4: Comparison of the delivery rates of RMSC and method of [15] with the information theoretic lower-bounds for different parameters α of Zipf distribution. In all cases, $N = 40$.

Table 4.1: Comparison of sets of popular files resulting from RMSC and [15] and the resulting rates. In all cases $N = 40$.

α	M	M/N	n^*	N_1	R_1/R^*
0.25	4	0.1	40	0	1.1192
	16	0.4	40	40	1.3056
	32	0.7	40	40	1.2753
0.5	4	0.1	16	2	1.0424
	16	0.4	40	32	1.3732
	32	0.7	40	40	1.2753
0.75	4	0.1	8	3	1.0059
	16	0.4	40	20	1.2063
	32	0.7	40	40	1.2753
1	4	0.1	4	3	1.000
	16	0.4	22	14	1.014
	32	0.7	38	26	1.047

Comparison to the information theoretic outer bound

Also plotted in Fig. 4.4 are two lower-bounds on the expected delivery rate of nonuniform demands which are reported in [15, Theorem 1] and [50, Theorem 2]. The lower-bound in [50, Theorem 2] is over the caching schemes with uncoded placement of content. Since the SCC algorithm relies on uncoded placement of content, the bound in [50, Theorem 2] must hold for the expected delivery rate of RMSC. The gap between the lower-bound in [50, Theorem 2] and the expected rate of RMSC is small in general, suggesting that the performance of the RMSC is close to the optimal performance for coded caching with uncoded prefetching of content. More specifically, one can make the following two observations. First, for a fixed cache size, this gap increases as the file popularity distribution becomes more nonuniform (larger α). Second, for a fixed value of α, this gap shrinks and approaches zero as M/N increases. The existence of this performance gap is due to the sub-optimality of the SCC procedure as the delivery algorithm that we explore in detail in the discussion section that will follow.

For completeness of our numerical exploration, we also include the lower-bound derived in [15, Theorem 1] in Fig. 4.4. Unlike [50, Theorem 2], the bound in [15, Theorem 1] applies to all caching schemes regardless of whether coded or

uncoded prefetching is used. However, we found this bound to be loose in general. This can be seen at the extreme case of $M/N = 0$, where the minimum amount of information that can be sent over the channel is equal to the number of distinct files requested. Yet, we see that the lower-bound in [15, Theorem 1] is considerably smaller than this value, suggesting that the bound is loose. Notice that the bound in [15, Theorem 1] is derived to prove order-optimality of the caching scheme proposed in [15] and is not guaranteed to be a tight bound on the optimal rate.

4.5.4 Discussion of the performance gap to the information-theoretic outer-bound

The existence of the gap between the expected rate of RMSC and the lower-bound in [50, Theorem 2], if not fully, is at least partially caused by the insensitivity of the SCC algorithm in Algorithm 4 to the presence of duplicate requests in the demand vector. In other words, if multiple caches request identical files, it still delivers them as if the files were distinct. For instance, consider the case that $M/N = 0$ and all caches request the same file. In that case, no side information is available at the caches and all requests must be delivered by uncoded messages. The SCC algorithm delivers the requests by transmitting the requested file K times. This is clearly suboptimal, as in this case it suffices to transmit the single file requested by all caches only once. This inefficiency of the original SCC algorithm for redundant requests becomes more complex to characterize for $M > 0$, but it has been thoroughly investigated in the literature [13, 54]. In particular, a modified version of the SCC algorithm was proposed in [13, Section IV.B and Appendix C.A], which resolves this inefficiency and is shown to achieve the optimal memory-rate tradeoff for the case of uniform demands when uncoded placement is used [13]. The modified delivery algorithm was discussed in Section 2.2. This difference between the original and modified SCC algorithms can explain the behaviour of the performance gap in Fig. 4.4. In particular, as M/N increases, the portion of bits cached at subsets $\mathcal{S} \subset [K]$ with large $|\mathcal{S}|$ increases. At the same time, larger $|\mathcal{S}|$ implies a higher probability that $\mathcal{S}$ includes at least one leader cache. Hence, the amount of information transmitted similarly by the original and modified SCC algorithms increases as M/N increases. As a result, the sub-optimality of the SCC algorithm

will have a minimal effect on the delivery rate and the gap of the rate of RMSC to the lower-bound shrinks as M/N increases.[9] Similarly, for a fixed M/N ratio, as α increases, more duplicate requests for the more popular files occur in the demand vector due to their considerably higher probability of request. Effectively, this reduces the number of leader caches for the different demand vectors on average. By reducing the number of subsets $\mathcal{S} : \mathcal{S} \cap \mathcal{U} \neq \emptyset$, this directly translates into the necessity of transmission of a smaller number of coded messages, and therefore, a larger gap between the performance of the original and the modified SCC algorithms. This analysis explains why the gap between RMSC and the lower-bound increases as α increases.

4.6 Concluding remarks

In this chapter, we applied the structured clique cover delivery algorithm that was proposed for decentralized coded caching to deliver files with nonuniform request probabilities. We fully characterized the structure of the optimal placement parameters. We showed that for a finite set of cache capacities, called base-cases, the optimal placement follows the two group strategy that does not cache a subset of less popular files, but treats the other files selected for caching identically regardless of their popularities. A polynomial time procedure was also proposed to derive this set of cache capacities. We further showed that the optimal placement parameters for other storage capacities can be obtained by memory sharing between certain base cases. In this scenario, a grouping of files into two or three groups is optimal.

Motivated by our numerical results as well as the fact that for uniform demands, the modified SCC algorithm proposed in [13] results in the exact memory-rate tradeoff for caching with uncoded prefetching, it is worthwhile to explore whether an analysis similar to what we presented in this paper can characterize the optimal placement for coded caching with the modified SCC algorithm of [13] for delivery. Furthermore, it is of interest to see how the rate of such a caching scheme compares

[9] A similar trend can be seen in [13, Fig. 5a] where the gap between the expected rates of the SCC algorithm and the (optimal) modified SCC algorithm vanishes as M/N increases, for the case of uniform demands.

to the information-theoretic lower-bound on the expected delivery rate of caching with uncoded prefetching.

Chapter 5

Coded Caching at File-Level

5.1 Introduction

Most of the existing coded caching schemes require the splitting of files into a possibly large number of subfiles, i.e., they perform *coded subfile caching*. Keeping the files intact during the caching process would be appealing, broadly speaking because of its simpler implementation. However, little is known about the effectiveness of *coded file caching* in reducing the data delivery rate. In this chapter, we propose such a file caching scheme and explore its performance.

5.1.1 Motivation and related work

We consider the problem of decentralized Coded File Caching (CFC), where files are kept intact and are not broken into subfiles. Our first motivation to analyze CFC is the large size requirement of most of the existing decentralized subfile cachings. In particular, [21] has developed a finite-length analysis of the decentralized coded caching of [12] and showed that it has a multiplicative coding gain of at most 2 even when F is exponential in the asymptotic gain KM/N. In [21], the authors have also proposed a new Coded Subfile Caching (CSC) scheme that requires a file size of $\Theta(\lceil N/M \rceil^{g+1}(\log(N/M))^{g+2}(2e)^g)$ to achieve a rate of $4K/(3(g+1))$.

Another important limitation of the existing coded caching schemes is the large number of subfiles required to obtain the theoretically promised delivery rates. For instance, [12] requires 2^K subfiles per file, and the different missing subfiles of

a file requested by each cache are embedded into 2^{K-1} different server messages [12]. This exponential growth in K has adverse consequences on the practical implementation of the system and its performance. In particular, a larger number of subfiles imposes large storage and communication overheads on the network. As is pointed out in [36], during placement, the caches must also store the identity of the subfiles and the packets that are included in each subfile, as a result of the randomized placement algorithms used. This leads to a storage overhead. Similarly, during delivery, the server needs to transmit the identities of the subfiles that are embedded in each message which leads to communications overhead. Neither of these overheads are accounted for in the existing coded caching rate expressions, while they are non-negligible when the number of subfiles is relatively large. For centralized coded caching, [22] has proposed a scheme that requires a linear number of subfiles in K and yields a near-optimal delivery rate. As is pointed out in [22], the construction of the proposed scheme is not directly applicable to the real caching systems as it requires K to be quite large. However, it is interesting to see that a small number of subfiles can in theory result in a near-optimal performance. For decentralized caching, reference [55] has investigated the subfile caching problem in the finite packet per file regime and has proposed a greedy randomized algorithm, GRASP, to preserve the coding gain in this regime. Based on computer simulations, the authors show that with a relatively small number of subfiles, a considerable portion of the asymptotic coding gain of infinite file size regime can be recovered. This observation further motivates a closer look at the file caching problem and its performance analysis. Finally, since the conventional uncoded caching systems do not break files into large numbers of subfiles, it is easier for the existing caching systems to deploy coded caching schemes that require one or a small number of subfiles per file.

5.1.2 Contributions

Although CSC has been well investigated, the analysis of CFC is missing from the literature. Based on the motivations we discussed earlier, it is worthwhile to explore CFC for its potential in reducing the delivery rate. In this chapter, we investigate the decentralized CFC problem by proposing new placement and delivery

algorithms. The placement algorithm is decentralized and does not require any coordination among the caches. The first proposed delivery algorithm is in essence a greedy clique cover procedure which is applied to certain side information graphs. This algorithm is designed for the general settings of CFC. The second delivery algorithm adapts the online matching procedure of [56] and is suitable for CFC in the small cache size regime.

The online matching procedure can be seen as a special case of the message construction algorithm in [57] with the merging procedure limited to 1-level merging.

The major contribution of this work is the analysis of the expected delivery rates of the proposed algorithms for file caching. In particular, we derive an approximate random graph modeling of the side information graphs for the proposed placement and the distribution of user demands and based on this model, we express the dynamics of the delivery algorithms through systems of differential equations. The resulting systems can be solved to provide a tight approximation to the expected delivery rate of CFC with the proposed algorithms. We provide a concentration analysis for the derived approximations and further demonstrate their tightness by computer simulations.

Our results show that CFC is significantly more effective than uncoded caching in reducing the delivery rate, despite its simple implementation and structurally constrained design. It is shown that the proposed CFC schemes preserve a considerable portion of the additive coding gain of the state-of-the-art CSC schemes. We present a discussion on the extension of the proposed placement and delivery algorithms to subfile caching with an arbitrary number of subfiles per file. We show that a considerable portion of the additive gain of the existing CSC methods can be revived by using a small number of subfiles per file, which is consistent with the observations made in [55]. Hence, our proposed method provides a means to tradeoff the delivery rate with the complexity caused by the use of a larger number of subfiles.

The remainder of this chapter is organized as follows. We present our proposed CFC method in Section 5.2. Statistical analysis of the delivery rates and concentration results are provided in Section 5.3. We present numerical examples and simulation results in Section 5.4. In Section 5.5, we discuss the generalization of

Algorithm 8 Decentralized File Placement

Require: Library of N files

1: **for** $k = 1, \ldots, K$ **do**
2: Cache k stores M out of N files from the library uniformly at random
3: **end for**

the proposed placement and delivery to CSC.

5.2 Placement and delivery algorithms

We consider a system model as in Section 2.1. For the CFC problem which we consider here, each cache can only store the entirety of a file and partial storage of a file is not allowed during content placement. This is the main element that differentiates the problem setups of CFC and CSC.

In the following, we propose algorithms for the placement and delivery of CFC.

5.2.1 Placement

Algorithm 8 shows the randomized procedure that we propose for content placement. Here, each cache stores M files from the library uniformly at random. The placement is decentralized as it does not require any central coordination among the caches. In Algorithm 8, all packets of M files are entirely cached. Notice that each cache stores any particular file with probability $q = M/N$, independent from the other caches. However, the storage of the different files in a given cache are statistically dependent, as the total number of cached files must not exceed M. Notice that Algorithm 8 is different from the decentralized subfile placement of [12], as in the latter, an M/N-th portion of the packets of *every* file in the library is cached and the packets of each file are cached independently from the packets of the other files.

5.2.2 Delivery

For the proposed delivery, the server forms a certain side information graph for every demand vector that arrives, and delivers the requests by applying a message

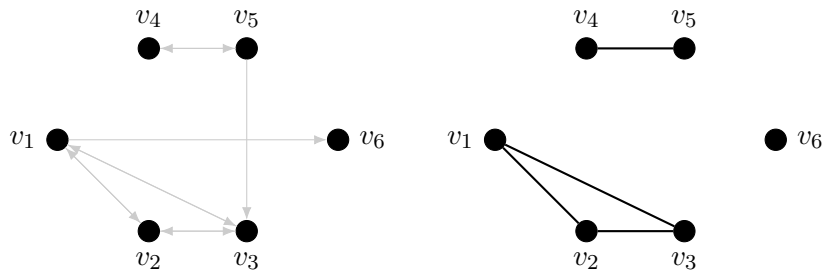


Figure 5.1: Left: an example side information digraph D. Right: the corresponding side information graph $\mathcal{G}$.

construction procedure to that graph. The graphs that we use are defined in the following.

Definition 8. *For a given placement and demand vector, the side information digraph $\mathcal{D}$ is a directed graph on K vertices. Each vertex corresponds to a cache. A vertex has a loop if and only if the file requested by the cache is available in its storage. There is a directed edge from v to w if and only if the file requested by cache w is in cache v.*

Digraph $\mathcal{D}$ was first introduced in the index coding literature and was used to characterize the minimum length of linear index codes [7]. This was done through computing the *minrank* of $\mathcal{D}$ [7, Theorem 1], which is an NP-hard problem.

Definition 9. *For a given placement and demand vector, the side information graph $\mathcal{G}$ is defined on the same set of vertices as $\mathcal{D}$. A vertex in $\mathcal{G}$ has a loop if and only if it has a loop in $\mathcal{D}$. There is an (undirected) edge between v and w if and only if both edges from v to w and from w to v are present in $\mathcal{D}$.*

For an example side information digraph $\mathcal{D}$, the corresponding side information graph $\mathcal{G}$ is shown in Fig. 5.1.

We propose two algorithms that make use of graph $\mathcal{G}$ for delivery.

Greedy clique cover delivery algorithm For delivery, the server constructs multicast messages with the property that each cache can derive its desired file from such messages using the side information that it has available.

Consider a set of unlooped vertices in graph $\mathcal{G}$ that form a clique.[1] Each vertex in that set has the file requested by the other caches available, but its own requested file is missing from its local storage. As a result, if the server transmits a message that is the XOR of the files requested by the caches in that clique, the message is decodeable by all such caches for their desired files [21, Section II.A]. Hence, to deliver the requests of every cache and minimize the delivery rate, it is of interest to cover the set of unlooped vertices of $\mathcal{G}$ with the minimum possible number of cliques and send one server message per clique to complete the delivery. Notice that the looped vertices of $\mathcal{G}$ do not need to be covered as the files requested by such caches are available to them locally.

Finding the minimum clique cover is NP-hard [58]. In this section, we adapt a polynomial time greedy clique cover procedure for the construction of the delivery messages of CFC. The proposed clique cover procedure is presented in Algorithm 9.

Notation 3. *In Algorithm 9, the content of the file requested by cache k is denoted by X_k and $\oplus$ represents the bitwise XOR operation.*

In Algorithm 9, a vertex v_t arrives at iteration (time) $t \in \{1, \cdots, K\}$. If v_t is looped, we proceed to the next iteration. Otherwise, we check if there is any *previously formed* clique of size $s \in \{1, \ldots, t-1\}$ that together with v_t forms a clique of size $s+1$. We call such a clique a *suitable clique* for v_t. If suitable cliques for v_t exist, we add v_t to the largest one of them to form a new clique. The rationale for choosing the largest suitable clique is explained in Remark 6. If there are multiple suitable cliques with the largest size, we pick one of them uniformly at random. If no suitable clique exists, v_t forms a clique of size 1. Notice that the cliques formed by Algorithm 9 are disjoint and cover the set of unlooped vertices of $\mathcal{G}$. After the clique cover procedure completes, the server sends a coded message corresponding to each clique.

The use of Algorithms 8 and 9 for placement and delivery leads to a caching scheme that we call Coded File Caching with greedy Clique Cover Delivery (CFC-CCD).

Remark 3. *In Algorithm 9, each cache only needs to listen to one server message*

[1] For an undirected simple graph (no loops and no multiple edges), a clique is a subset of vertices where every pair of vertices are adjacent.

Algorithm 9 Greedy Clique Cover Delivery

Require: $\mathcal{G}$, vertex labels $v_1, \cdots, v_K$

// Form Cliques

$C \leftarrow \emptyset$ // initialize set of cliques

for $t = 1, \ldots, K$ **do**

  if v_t has a loop **then**

    Do nothing

  else if there is a suitable clique for v_t in C **then**

    Join v_t to (one of) the largest suitable clique(s) (randomly) and update C

  else

    Add $\{v_t\}$ to C as a clique of size 1

  end if

end for

// Transmission of Messages

for $c \in C$ **do**

  Transmit $\oplus_{k \in c} X_k$

end for

to decode the file it requested, as the entirety of the file is embedded in only one message. This is in contrast to the CSC *of [12] which requires each cache to listen to* 2^{K-1} *out of the* 2^K *server messages to derive its requested content.*

Algorithm 9 has a worst-case complexity of $O(K^2)$. This is because there are at most K unlooped vertices in $\mathcal{G}$, and in each iteration of the algorithm, the adjacency of v_t with at most $t-1$ vertices must be checked for finding the largest suitable clique.

Online Matching Delivery Algorithm We now propose our second delivery algorithm, which is applicable to the small cache size regime, i.e., when q is small. In this regime, the probability of having large cliques is small. Hence, one can restrict the size of the cliques in the clique cover procedure to reduce the complexity without considerably affecting the delivery rate. As an extreme case, Algorithm 10 shows an online greedy matching algorithm adapted from [56, Algorithm 10], which restricts cliques to those of sizes 1 and 2. Notice that here graph $\mathcal{G}$ can have loops which is different from the graph model in [56, Algorithm 10]. In Algorithm 10, if v_t is looped, we remove it from the graph and proceed to the next

Algorithm 10 Online Matching Delivery

Require: $\mathcal{G}$, vertex labels $v_1, \cdots, v_K$

// Transmission of coded messages

$Q \leftarrow \emptyset$ // set of matched or looped vertices

for $t = 1, \ldots, K$ **do**

  $\mathcal{G}_t \leftarrow$ subgraph induced by $\mathcal{G}$ on vertices $\{v_1, \cdots, v_t\} \backslash Q$

  if v_t has a loop **then**

    $Q \leftarrow Q \cup \{v_t\}$

  else if v_t has a neighbor in $\mathcal{G}_t$ **then**

    Match v_t to a random neighbor $v_s \in \mathcal{G}_t$

    Transmit $X_{v_t} \oplus X_{v_s}$

    $Q \leftarrow Q \cup \{v_t, v_s\}$

  end if

end for

// Transmission of Uncoded Messages

for $v \in \{v_1, \cdots, v_K\} \backslash Q$ **do**

  Transmit X_v

end for

iteration. Otherwise, we try to match it to a previously arrived unmatched vertex. If a match found, we remove both vertices from the graph and proceed to the next iteration. Otherwise, we leave v_t for possible matching in the next iterations.

The use of Algorithms 8 and 10 for placement and delivery leads to a method that we call **cfcm!** (**CFCM!**).

5.3 Performance analsyis

In this section, we analyze the expected delivery rates of the proposed file caching schemes through a modeling of the dynamics of Algorithms 9 and 10 by differential equations.

5.3.1 Overview of Wormald's differential equations method

To analyze the performance of **CFCM!** and CFC-CCD, we use Wormald's differential equation method and in particular Wormald's theorem [59, Theorem 5.1]. A

formal statement of the theorem is provided in Appendix B.1. Here, we present a rather qualitative description of the theorem and how it can be used to analyze the performance of the greedy procedures on random graphs.

Consider a dynamic random process whose state evolves step by step in discrete time. Suppose that we are interested in the time evolution of some property P of the state of the process. Since the process is random, the state of P is a random variable. To determine how this random variable evolves with time, a general approach is to compute its expected changes per unit time at time t, and then regard time t as continuous. This way, one can write the differential equation counterparts of the expected changes per unit of time in order to model the evolution of the variable. Under certain conditions, the random steps follow the expected trends with a high probability, and as a result, the value of the random variable is concentrated around the solution of the differential equations. In this context, Wormald's theorem states that if (i) the change of property P in each step is small, (ii) the rate of changes can be stated in terms of some differentiable function, and (iii) the rate of changes does not vary too quickly with time, then the value of the random variable is sharply concentrated around the (properly scaled) solution of the differential equations at every moment [59, Section 1.1],[60, Section 3].

Notice that both Algorithms 9 and 10 can be viewed as dynamic processes on the random side information graph. In Algorithm 9, a vertex arrives at each time and potentially joins a previously formed clique. In this case, we are interested in the number of cliques at each time instant, as at the end of the process, this equals the number of coded messages that must be sent. In Algorithm 10, the arrived vertex might be matched to a previously arrived vertex. Here, the property of interest is the total number of isolated vertices plus matchings made. We apply Wormald's theorem to analyze the behavior of these random quantities and to approximate their expected values at the end of the process.

5.3.2 Statistical properties of the random graph models for side information

For the analysis of the side information graphs $\mathcal{D}$ and $\mathcal{G}$, it is required to characterize the joint distribution of the edges and the loops for each of these graphs.

In digraph $\mathcal{D}$, every directed edge or loop is present with the marginal probability of q. However, there exist dependencies in the presence of the different edges or loops in $\mathcal{D}$.

Notation 4. *We denote by e_{uv} the Bernoulli random variable representing the directed edge from vertex u to vertex v. In case of a loop, we have $u = v$. This random variable takes values 1 and 0 when the edge is present and absent, respectively. Also, we represent by $\mathcal{E}$ the set of the K^2 random variables representing all the edges and the loops of $\mathcal{D}$. Further, by f_u and $\mathcal{F}_{-u}$ we denote the random file requested by vertex u and the random set of distinct files requested by all vertices except vertex u, respectively.*[2]

Remark 4. *To characterize the dependencies among the edges and the loops of $\mathcal{D}$, consider the random variable e_{uv}. The state of the other variables $\mathcal{E} \setminus e_{uv}$ affects the distribution of e_{uv} in the following way:*

(i) The presence (absence) of a directed edge from vertex u to another vertex v implies that the file requested by v is (not) available in the cache of u. Hence, if $f_v \in \mathcal{F}_{-v}$, the values of the variables in $\mathcal{E} \setminus e_{uv}$ that correspond to the edges that originate from u fully determine e_{uv}. Otherwise, the random variables $\mathcal{E} \setminus e_{uv}$ provide information about the storage of the files $\mathcal{F}_{-v}$ in the cache of u. Since the cache size is finite, this information affects the probability of the storage of f_v in the cache of u and hence the distribution of $e_{uv}|\mathcal{E} \setminus e_{uv}$ can be different from the marginal distribution of e_{uv}.

(ii) Furthermore, the values of $\mathcal{E} \setminus e_{uv}$ can affect the probability that $f_v \in \mathcal{F}_{-v}$ (see Fig. 5.2 for an example). Based on point (i), any change in the probability of $f_v \in \mathcal{F}_{-v}$ can directly change the probability of the presence of f_v in the cache of u.

An example of point (ii) in Remark 4 is shown in Fig. 5.2 for vertices v and w. The state of the edges between these two vertices and their loops implies that $f_v \neq f_w$. Consequently, $e_{uw} = 1$ reduces the probability of the presence of f_v in the cache of u, or equivalently the probability of $e_{uv} = 1$, from the marginal probability $q = \frac{M}{N}$ to $\frac{M-1}{N-1} < q$.

[2]We have $f_u \in \mathcal{F}_{-u}$ if the file requested by vertex u is also requested by other vertices.

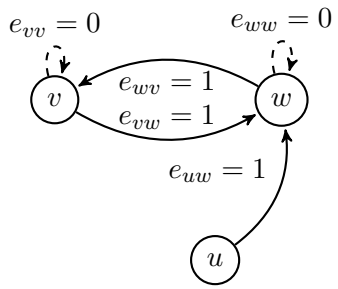


Figure 5.2: Solid (dashed) lines represents the presence (absence) of an edge or loop and no information is available about the presence of the edges and loops that are not shown. This is an illustration of the dependencies among the edges of $\mathcal{D}$. The values of e_{ij}, $i, j \in \{v, w\}$ in this example imply that files f_v and f_w are distinct, as otherwise we must have had $e_{vv} = 1$ and $e_{ww} = 1$. Further, $e_{uw} = 1$ implies that f_w is cached in u. As a result, the probability of $e_{uv} = 1$, i.e., f_v being in the cache of u, becomes $\frac{M-1}{N-1}$. This is different from the marginal probability $\frac{M}{N}$ for $e_{uv} = 1$, which shows the dependency among the edges.

Because of the dependencies among the edges of $\mathcal{D}$ and therefore the dependencies in the edges of $\mathcal{G}$, an analysis of the performance of **CFCM!** and CFC-CCD is difficult. However, we show that in the regime of $\frac{K}{N} \to 0$ such dependencies become negligible.

Definition 10. *Consider the side information digraph (graph) $\mathcal{X}$ as defined in Definition 8 (Definition 9) for a network with K caches and a library of N files from which each cache independently and uniformly at random requests a file. We say a random digraph (graph) $\mathcal{X}_a$ asymptotically approximates $\mathcal{X}$ as $K/N \to 0$ if it is defined over the same set of vertices as $\mathcal{X}$, and the joint probability distributions of the presence of edges in the two graphs satisfy $\lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{X})}(\mathcal{E}) = \lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{X}_a)}(\mathcal{E})$, where the superscripts determine the graph to which the conditional edge probability corresponds.*[3] *For simplicity, we refer to $\mathcal{X}_a$ as an asymptotic model for $\mathcal{X}$.*

Proposition 7. *For $\frac{K}{N} \to 0$, the random graph $\mathcal{D}$ is asymptotically approximated*

[3]Here again, e_{uv} represents the random variable corresponding to the presence of a directed edge from u to v if $\mathcal{X}$ is a digraph and an edge between u and v if $\mathcal{X}$ is an undirected graph. Also, $\mathcal{E}$ represents the set of random variables e_{uv} defined over all combinations of u and v.

by the graph $\mathcal{D}_a$ for which every edge and loop is present independently with probability q. Also, graph $\mathcal{G}$ is asymptotically approximated by $\mathcal{G}_a$ for which every edge and loop is present independently with probabilities q^2 and q, respectively.

Proof. See Appendix B.2. □

Based on the chain rule for probabilities and since the limits of the conditional probabilities in Definition 10 are finite,[4] a random graph $\mathcal{X}$ and an asymptotic approximation to it $\mathcal{X}_a$, have the same joint distribution over their edges as $K/N \to 0$. As a result, if a property defined over random graph $\mathcal{X}$ is solely a function of its edges, for instance the number of cliques or isolated vertices in $\mathcal{X}$, that property behaves statistically the same when defined over $\mathcal{X}_a$ as $K/N \to 0$.

5.3.3 Analysis of the expected rate of CFCM! and CFC-CCD

Algorithms 9 and 10 take a generic side information graph $\mathcal{G}$ as input and output the communication scheme comprising the delivery messages. The number of messages constructed by these algorithms is solely a function of the configuration of the edges of $\mathcal{G}$. Moreover, based on Proposition 7, graph $\mathcal{G}_a$ asymptotically approximates $\mathcal{G}$ in the $K/N \to 0$ regime. Hence, we perform the analysis of the expected delivery rate of the proposed caching schemes in the $\frac{K}{N} \to 0$ regime using the asymptotic model $\mathcal{G}_a$ for the side information graph. This approach is mathematically tractable, as unlike in $\mathcal{G}$, all the edges and loops are independent from each other in $\mathcal{G}_a$.

Assuming that the input graph to Algorithms 9 and 10 is statistically modeled by random graph $\mathcal{G}_a$, and by applying Wormald's theorem, we get the following two results for the outputs of the two algorithms.

Theorem 3. *For Algorithm 10 with input side information graphs that follow the random graph model $\mathcal{G}_a$, the number of isolated vertices plus the number of match-*

[4] In particular, $\lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{X})}(\mathcal{E}) = \lim_{\frac{K}{N}\to 0} \prod_i \mathbb{P}^{(\mathcal{X})}(e_i \mid e_1, \ldots, e_{i-1}) = \prod_i \lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{X})}(e_i \mid e_1, \ldots, e_{i-1})$, where the edge indices represent an arbitrary ordering of the set of all possible edges and loops of the random graph $\mathcal{X}$.

ings made by Algorithm 10 is

$$\frac{1}{2}\left[K(1-q)-\frac{\log(2-(1-q^2)^{K(1-q)})}{\log(1-q^2)}\right]+O(\lambda K)$$

with probability $1-O(\frac{1}{\lambda}e^{-K\lambda^3})$*, for any* $\lambda>0$*.*

Proof. See Appendix B.3. □

Theorem 4. *For Algorithm 9 with input side information graphs that follow the random graph model* $\mathcal{G}_a$*, the number of cliques that cover their vertices using Algorithm 9 is*

$$K\sum_{i=1}^{K} z_i(1;q)+O(\lambda K), \tag{5.1a}$$

with probability $1-O(\frac{K}{\lambda}e^{-K\lambda^3})$*, where functions* $z_i(x;q),\ i=1,\ldots,K$ *are given by the unique solution to the system of differential equations*

$$\begin{cases} \frac{dz_i}{dx}=(1-q)[2g_i(\boldsymbol{z})-g_{i+1}(\boldsymbol{z})+g_{i-1}(\boldsymbol{z})]\,; \\ z_i(0;q)=0, \end{cases} \tag{5.1b}$$

where

$$g_i(\boldsymbol{z})=\prod_{j=i}^{K}(1-q^{2j})^{Kz_j(x;q)} \tag{5.1c}$$

and $\boldsymbol{z}=(z_1,\ldots,z_K)$*.*

Proof. See Appendix B.4. □

One can combine Proposition 7, and Theorems 3 and 4, to get approximations to the expected delivery rates of **CFCM!** and CFC-CCD as is outlined in the following result.

Corollary 4. *For* $0\ll K\ll N$*, the expected delivery rates of* **CFCM!** *and*

CFC-CCD *can be approximated by*

$$\mathbb{E}\left(R_m\right) \approx \frac{1}{2}\left[K(1-q) - \frac{\log(2-(1-q^2)^{K(1-q)})}{\log(1-q^2)}\right] \tag{5.2}$$

and

$$\mathbb{E}(R_{cc}) \approx K\sum_{i=1}^{K} z_i(1), \tag{5.3}$$

respectively.

The approximations in (5.2) and (5.3) become tight as $K \to \infty$ and $\frac{K}{N} \to 0$. The former is required for the validity of the concentration results in Theorems 1 and 2.[5] The latter condition ensures that the asymptotic models $\mathcal{D}_a$ and $\mathcal{G}_a$ are valid statistical representations of the side information graphs $\mathcal{D}$ and $\mathcal{G}$ as per Proposition 7. Notice that this condition is satisfied in many practical scenarios where the number of files to be cached is considerably larger than the number of caches in the network.

Remark 5. *Applying Algorithm 9 to random graph $\mathcal{G}_a$ has the property that as vertex v_t arrives, the behavior of the algorithm up to time t does not provide any information about the connectivity of vertex v_t to the previously arrived vertices. In other words, v_t is connected to any of the previously arrived vertices with probability q^2. This is because Algorithm 9 operates based on the structure of the edges in the side information graph. If the different edges are present independently, the history of the process up to time t does not change the probability of the presence of the edges between v_t and the previously arrived vertices $v_1, \ldots v_{t-1}$. This property is essential for the proof of Theorem 4.*

Remark 6 (Rationale for Choosing the Largest Clique in Algorithm 9)**.** *Algorithm 9 joins v_t to the largest suitable clique at time t. The rationale for this choice is as follows. Let $Y_i(t)$ represent the number of cliques of size i right before time t, where $i \in \{1, \cdots, K\}$. Let $\mathcal{N}_t$ be the event that there exists no suitable*

[5]This is because the asymptotics denoted by O in Theorems 3 and 4 are for $K \to +\infty$, and the probabilities in the statement of the theorems approach 1 as K increases.

clique for v_t at time t. Then,

$$\mathbb{P}(\mathcal{N}_t) = \prod_{i=1}^{K} (1 - q^{2i})^{Y_i(t)}.$$

Given $Y_i(t)$, and with the knowledge that v_t has joined a clique of size s at time t, we get $Y_i(t+1) = Y_i(t), i \notin \{s, s+1\}$, $Y_s(t+1) = Y_s(t) - 1$ and $Y_{s+1}(t+1) = Y_{s+1}(t) + 1$. Then, it is easy to show that

$$\begin{aligned} &\mathbb{P}(\mathcal{N}_{t+1} | v_t \textit{ joined a clique of size } s \textit{ at time } t, \boldsymbol{Y}(t)) \\ &\qquad = \frac{1 - q^{2(s+1)}}{1 - q^{2s}} \prod_{i=1}^{K} (1 - q^{2i})^{Y_i(t)}, \end{aligned} \tag{5.4}$$

where $\boldsymbol{Y}(t) = (Y_1(t), \cdots, Y_K(t))$. In order to minimize the expected rate, one has to make (5.4) small. Now, assume that at time t, multiple suitable cliques existed for v_t. It is straightforward to check that (5.4) would take its smallest value if the suitable clique with the largest size s was chosen at time t.

Remark 7. *In the alternative setting that the popularities of files are nonuniform, the more popular files would be cached with higher probabilities during placement. In that case, the property discussed in Remark 5 does not hold anymore for Algorithm 9. In particular, at step t, the posterior probability of v_t to connect to the previously arrived vertices was strongly dependent on the history of the process up to time t. For instance, the probability of v_t to connect to the vertices that have formed a clique with a large size is higher than the probability of v_t to connect to vertices that belong to small cliques. This is because a vertex that belongs to a large clique is more likely to correspond to a request for a more popular file. Such a file is more likely to be availabe in each cache, including the one that corresponds to vertex v_t. The modeling of the posterior probabilities of the presence of edges makes the analysis of* CFC-CCD *difficult for the side information graphs arisen from nonuniform demands.*

5.4 Performance comparison and simulations

In this section, we present numerical examples and simulation results for the application of the CFC schemes that we proposed, to demonstrate the effectiveness of CFC in reducing the delivery rate. We further show that the expressions in (5.2) and (5.3) tightly approximate the expected delivery rates of **CFCM!** and CFC-CCD, respectively.

We use the expected delivery rates of two reference schemes to comment on the performance of our proposed CFC. The reference cases are the uncoded caching and the optimal decentralized CSC scheme derived in [13]. The latter is the state-of-the-art result on CSC which provides the optimal memory-rate tradeoff over all coded cachings with uncoded placement. Although the asymptotic average delivery rate derived in [13, Theorem 2] is valid in the infinite file size regime and the underlying caching scheme requires an exponential number of subfiles in K, it provides a suitable theoretical reference for our comparisons. To compute the expected delivery rate of the optimal decentralized CSC, one needs to find the expectation in the RHS of [13, Theorem 2]. This is done in the following lemma.

Lemma 4.

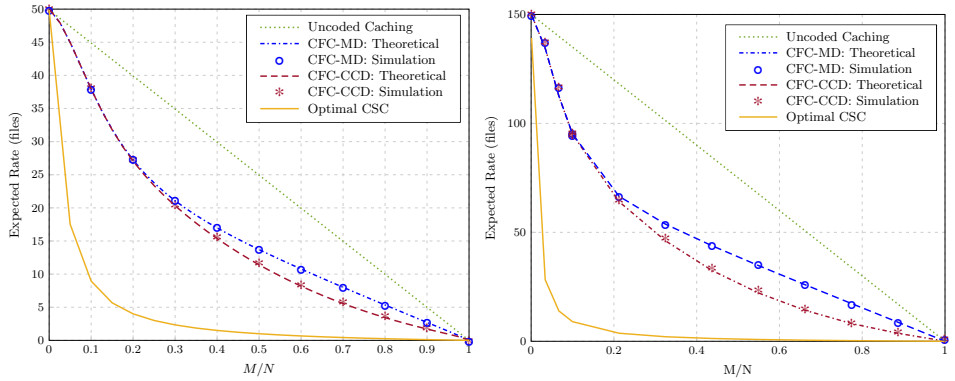
$$\mathbb{E}_{\boldsymbol{d}}(R^*_{CSC}) = \frac{N-M}{M}\left[1-\sum_{n=1}^{K}\frac{\binom{N}{n}\left\{{K \atop n}\right\}n!}{N^K}(1-M/N)^n\right], \tag{5.5}$$

where $\left\{{K \atop n}\right\}$ *represents the Stirling number of the second kind [61, Section 5.3]. Also, the expected rate of uncoded caching is derived as*

$$\begin{aligned}\mathbb{E}_{\boldsymbol{d}}(R_{uncoded}) &= N\left(1-(1-\frac{1}{N})^K\right)(1-M/N) \\ &= K(1-M/N)+O(1/N^2)\end{aligned} \tag{5.6}$$

Proof. See Appendix B.5. □

We use the approximation $\mathbb{E}_{\mathbf{d}}(R_{\text{uncoded}}) \approx K(1-M/N)$ throughout this section, as we use $N \geq 100$ in all the following examples.



(a) $K = 50$, $N = 1000$ **(b)** $K = 150$, $N = 100$

Figure 5.3: Comparison of the delivery rates of the different caching schemes.

5.4.1 Numerical Examples for Expected Rates

Fig. 5.3 shows the expected delivery rates of **CFCM!** and CFC-CCD, as well as the average rates of the optimal CSC and the uncoded caching for two networks, one with parameters $K = 50$ and $N = 1000$ and the other with parameters $K = 150$ and $N = 100$. First, we observe that CFC-CCD is notably effective in reducing the delivery rate. In particular, it reduces the delivery rate by 60 to 70 percent compared to uncoded caching. Notice that as argued in Section 5.2.2, for small cache sizes, the expected delivery rate of **CFCM!** is close to that of CFC-CCD, as only a small fraction of vertices are covered by cliques of sizes larger than 2. This is shown in Fig. 5.4. As the cache sizes get larger, the probability of formation of larger cliques increases and therefore CFC-CCD considerably outperforms **CFCM!**. Hence, in the small-cache regime, the use of **CFCM!** is practically preferred to CFC-CCD because of its lower computational complexity.

Further, we observe that the theoretical expressions in (5.2) and (5.3) are in agreement with the empirical average rates obtained by simulations. Particularly, we have $\frac{K}{N} = 0.05$ and 1.5 in Figs. 5.3a and 5.3b, respectively. The results for the latter case imply that the condition $\frac{K}{N} \to 0$ required in our theoretical analysis can be too conservative in practice.

In Fig. 5.5, the per user expected delivery rates, i.e., the expected delivery rates

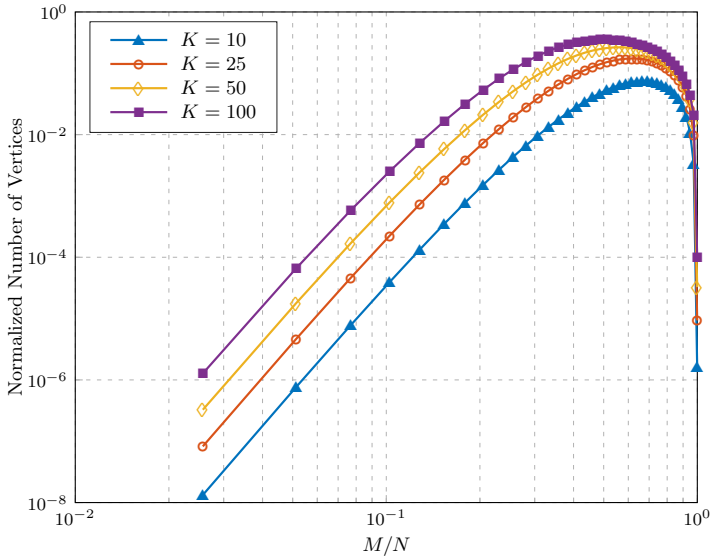


Figure 5.4: Expected number of vertices that are covered by cliques of sizes larger than 2 by Algorithm 9, normalized by K. Here, $N = 1000$.

normalized by K, are shown for different numbers of caches. In all cases, N is chosen such that $\frac{K}{N} = 0.1$. Notice that although the asymptotics in the rate expressions in Theorems 3 and 4 are for $K \to \infty$, the proposed approximations closely match the simulation results for K as small as 10. In general, our simulation explorations show that in the regime of $K \geq 20$ and $K/N \leq 0.5$, the approximations in Corollary 4 are tight.

5.4.2 Characterization of coding gain

We now look at the additive and multiplicative coding gains defined as $g_a = 1 - \frac{\mathbb{E}(R_{\text{coded}})}{\mathbb{E}(R_{\text{uncoded}})}$ and $g_m = \frac{\mathbb{E}(R_{\text{uncoded}})}{\mathbb{E}(R_{\text{coded}})}$, respectively, for CFC-CCD and **CFCM!**. Notice that $g_a = 1 - 1/g_m$. Fig. 5.6 shows g_a and g_m for different coded caching schemes. First, notice that in all cases, CFC-CCD significantly outperforms uncoded caching. Second, for CFC-CCD, g_a and g_m reach their maximum for large values of M/N, where the gap between the additive gains of CFC-CCD and optimal subfile caching shrinks significantly. The initial increase of the coding gains with M/N is due to formation of larger cliques in the side information graph. However, as $M \to N$,

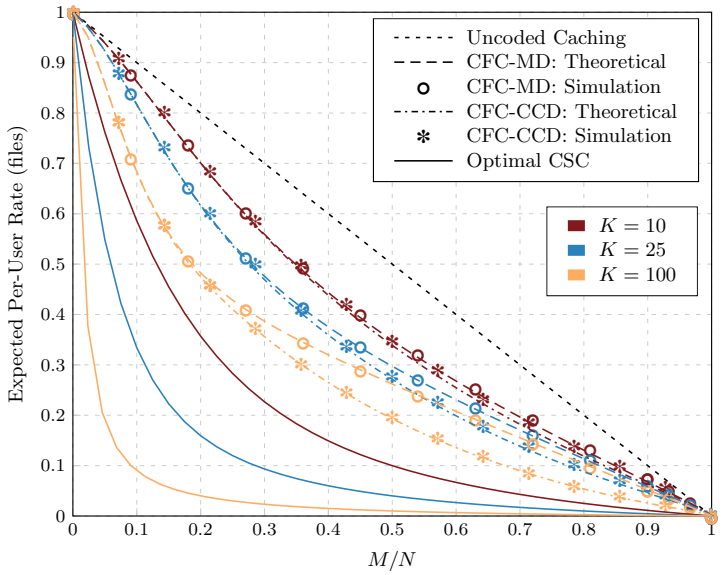
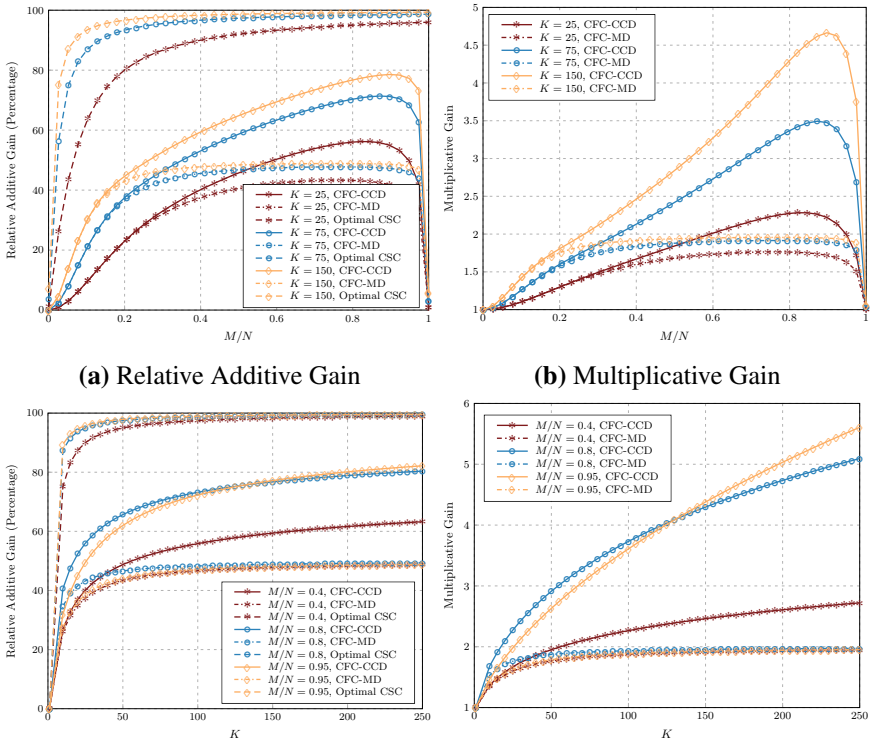


Figure 5.5: The expected per-user delivery rates for caching networks with different numbers of caches. Here, $\frac{K}{N} = 0.1$.

the gains decrease again. This behavior is due to the fact that although for large M/N most of the vertices are connected, at the same time most of the vertices are also looped and are removed by Algorithm 9. Hence, for $M \approx N$, the chance of formation of large cliques diminishes, causing g_m to approach 1 and consequently g_a to drop to 0. Third, the coding gains increase with K because of the higher number of larger cliques formed in the side information graph. For **CFCM!**, g_m is upper bounded by 2 which is the multiplicative coding gain of **CFCM!** when perfect matching occurs. Fourth, notice the gap of the curves for the optimal CSC to 100% at $M/N = 1$. This gap is equal to $\frac{1}{N(1-(1-1/N)^K)} \times 100\%$[6], which can be approximated by $\frac{1}{K} \times 100\%$ for large N.

Based on our observation in Fig. 5.6, the multiplicative gain of CFC-CCD is limited compared to the optimal CSC, as in the latter, it grows almost linearly with

[6]This is a direct consequence of eqs. (B.9) and (B.11) in which imply that $\frac{\mathbb{E}(R_{\text{uncoded}})}{\mathbb{E}(R^*_{\text{CSC}})} = \frac{N(1-(1-1/N)^K)q}{\sum_{m=1}^{K} \mathbb{P}(N_e(\boldsymbol{d})=m)(1-(1-q)^m)}$. As a result, $\lim_{q\to 1} \frac{\mathbb{E}(R_{\text{uncoded}})}{\mathbb{E}(R^*_{\text{CSC}})} = N(1-(1-1/N)^K)$ and $\lim_{q\to 1} \frac{\mathbb{E}(R_{\text{uncoded}})-\mathbb{E}(R^*_{\text{CSC}})}{\mathbb{E}(R_{\text{uncoded}})} = 1 - \frac{1}{N(1-(1-1/N)^K)}$.



(a) Relative Additive Gain **(b)** Multiplicative Gain

(c) Relative Additive Gain **(d)** Multiplicative Gain

Figure 5.6: Comparison of the additive and multiplicative coding gains obtained by the different caching schemes. Here, $N = 100$.

Kq.[7] However, we observe that a considerable portion of the additive gain of optimal CSC can be revived by CFC-CCD. Hence, it is better justified to look at the coding gains of the proposed CFC schemes as additive and not multiplicative gains.

5.5 Extension to subfile caching

We have shown that the proposed CFC is an easy to implement yet effective technique to reduce the average delivery rate of caching networks. However, there is

[7] To keep the other curves readable, the multiplicative coding gain of the optimal CSC is not shown in Figs. 5.6b and 5.6d.

a gap between the delivery rates of the proposed CFC and the optimal CSC, as the latter allows for an arbitrarily high complexity in terms of the number of subfiles used per file and also assumes infinite number of packets per file. It is of practical interest to explore the improvement in the expected delivery rate in a scenario where the system can afford a given level of complexity in terms of the number of subfiles used per file. In this section, we consider this problem and show that the placement and delivery proposed in Algorithms 8 and 9 also provide a framework for CSC when a small number of subfiles per file is used for caching. The resulting CSC scheme provides a means to tradeoff between the delivery rate and the implementation complexity, i.e., the number of subfiles used per file.

Throughout this section, performance evaluations are based on computer simulations as the theoretical results of Section 5.3 cannot be extended to subfile caching.

Placement

For subfile caching, we break each file in the library into $\Delta \geq 1$ subfiles. This leads to a library of $N\Delta$ subfiles, where each subfile is of length F/Δ. Then, we apply the placement in Algorithm 8 to the library of subfiles. Notice that the proposed subfile placement is different from the decentralized placement of [12]. In particular, here all the subfiles are of the same length. Moreover, in the placement of [12], the number of packets of each file stored in each cache was M/NF, while this number is random here. Also notice that, for each cache, the prefetching of the subfiles that belong to different files are dependent, as the total size of the cached subfiles must be at most M.

Remark 8. *The choice of Δ depends on the level of complexity that is tolerable to the system. The only restriction that the proposed placement imposes on Δ is for the ratio F/Δ to be an integer value. This condition can be easily satisfied in the finite file size regime.*

Delivery

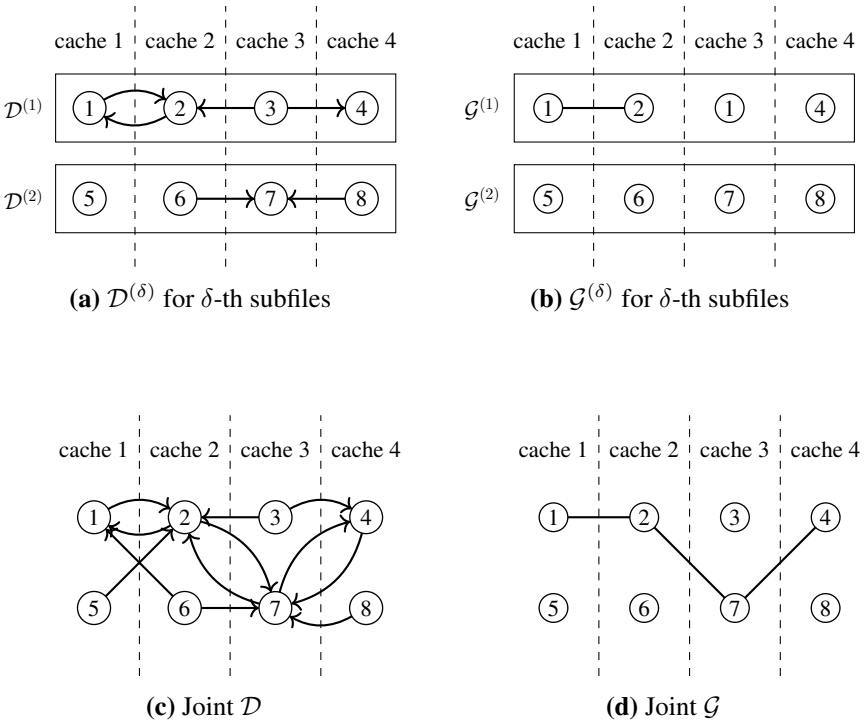
Similar to the delivery of file caching, for subfile caching the server forms the side information graphs $\mathcal{D}$ and $\mathcal{G}$ upon receiving the user demands. Then, it delivers the

requests using Algorithm 9, by treating each subfile like a file. Notice that we do not consider CSC with online matching delivery in this section, as its coding gain is limited to 2 and this bound does not improve by increasing the number of subfiles.

Side information graphs For subfile caching, both the side information graphs $\mathcal{D}$ and $\mathcal{G}$ have $K\Delta$ vertices. Similar to the approach we followed in Proposition 7, in the following analysis, we consider $\mathcal{D}_a$ and $\mathcal{G}_a$ as the asymptotic counterparts of $\mathcal{D}$ and $\mathcal{G}$ for $\frac{K\Delta}{N\Delta} \to 0$. Let $w_{k,\delta}$ represent the vertex corresponding to cache k and subfile δ. In $\mathcal{D}_a$, vertex $w_{k,\delta}$ has a loop if subfile δ of the file requested by cache k is available in cache k. The probability of a loop is q. The main structural difference between graphs $\mathcal{D}_a$ for file and subfile caching is that the presence of the edges are highly dependent in subfile caching. In particular, for two given caches k and $l \neq k$, and a given subfile δ of the file requested by cache k, there exist Δ directed edges from $w_{l,\theta}$ to $w_{k,\delta}$ for all $\theta = 1, \dots, \Delta$, if subfile δ of the file requested by cache k is available in cache l. This event has probability q. Otherwise, *all* Δ edges from $w_{l,\theta}$ to $w_{k,\delta}$ with $\theta = 1, \dots, \Delta$ are absent. Also, for subfile caching, we draw no edges from $w_{k,\delta}$ to $w_{k,\theta}$ for $\delta \neq \theta$. This does not affect the output of the delivery algorithm as if subfile θ of file X_k is present in cache k, vertex $w_{k,\theta}$ is looped and hence is ignored (not joined to any clique) by Algorithm 9. The latter two properties make the model of the random graph $\mathcal{D}_a$ for subfile caching significantly different from the model used for file caching.

Graph $\mathcal{G}_a$ is built from $\mathcal{D}_a$ in exactly the same way as for file caching. As a result, each vertex is looped with probability q. Also, any two vertices belonging to two different caches are present with marginal probability of q^2. However, the presence of the edges between the vertices of two caches are highly dependent because of the discussed structures in $\mathcal{D}_a$.

Notice that forming the joint side information graphs for the delivery of all Δ subfiles increases the coding opportunities compared to the case with Δ separate side information graphs for the disjoint delivery of the δ-th subfiles of every file. In other words, the number of edges in the joint side information graph $\mathcal{G}_a$ grows superlinearly in Δ as each of the $K\Delta$ vertices can connect to $(K-1)\Delta$ other vertices. Fig. 5.7 shows an example of separately formed and the corresponding jointly formed side information graphs.



(a) $\mathcal{D}^{(\delta)}$ for δ-th subfiles

(b) $\mathcal{G}^{(\delta)}$ for δ-th subfiles

(c) Joint $\mathcal{D}$

(d) Joint $\mathcal{G}$

Figure 5.7: $K = 4$ caches and $\Delta = 2$ subfiles per file. (a), (b): Side information graphs for separate delivery of δ-th subfiles with $\delta = 1, \ldots, \Delta$. (c), (d): Side information graphs for joint delivery of all subfiles.

Remark 9. *Coding opportunities increase by increasing Δ. Hence, in practice, Δ is determined by the highest level of complexity that is tolerable to the system in terms of the number of subfiles used.*

Delivery algorithms The delivery process is complete with the side information graph $\mathcal{G}$ inputted to Algorithm 9. We call the resulting scheme Coded Subfile Caching with greedy Clique Cover Delivery (CSC-CCD).

Simulation results

For a system with $K = 50$ caches, Fig. 5.8a shows the expected delivery rates of CSC-CCD with $\Delta = 5$ and $\Delta = 25$ as well as the expected delivery rates of CFC-CCD, the optimal decentralized CSC of [13] and uncoded caching. Notice that

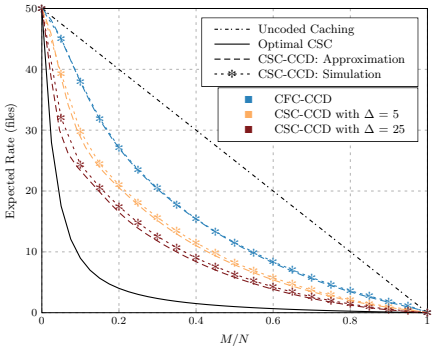
CFC-CCD is identical to CSC-CCD with $\Delta = 1$ subfile.

An important observation is that relative to the 2^K subfiles required by the optimal decentralized CSC, the use of a small number of subfiles in CSC-CCD can significantly shrink the rate gap between the delivery rates of CFC-CCD and the optimal decentralized CSC. In Fig. 5.8a, we have also shown approximate expected delivery rates based on the theoretical results in Theorem 4. More specifically, we use the approximation $R_{cc,\text{subfile}} \approx \frac{1}{\Delta} R_{cc}(K\Delta)$. As we discussed earlier, because of the structural differences between the side information graphs for file and subfile caching, such an approximation is generally prone to large errors. However, for $\Delta \ll K$ and $N\Delta \gg K\Delta$, it still results in an approximation for the expected delivery rate.

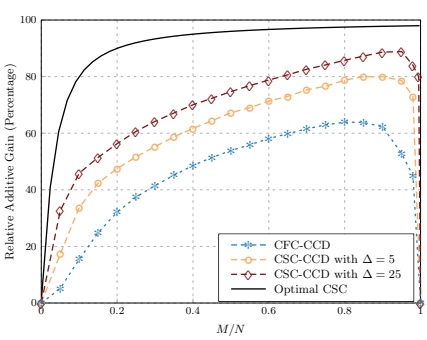
Figs. 5.8b and 5.8c show the additive and multiplicative coding gains for the proposed CSC scheme. One observes that the rate of change in the additive gain decays as the number of subfiles increases. For instance, increasing the number of subfiles from $\Delta = 1$ to $\Delta = 5$ results in a larger reduction in the delivery rate compared to the case where the number of subfiles is increased from $\Delta = 5$ to $\Delta = 25$. However, the multiplicative gain increases significantly when a larger number of subfiles is used.

5.6 Concluding remarks

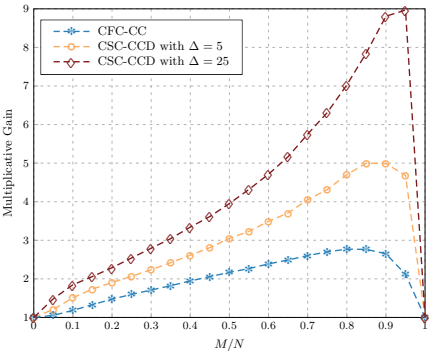
We explored the decentralized coded caching problem for the scenario where files are not broken into smaller parts (subfiles) during placement and delivery. This scenario is of practical importance because of its simpler implementation. We showed that although the requirement of caching the entirety of a file puts restrictions on creation of coding opportunities, coded file caching is still an effective way to reduce the delivery traffic of the network compared to the conventional uncoded caching. In particular, we proposed the CFC-CCD file caching scheme, which performs delivery based on a greedy clique cover algorithm operating over a certain side information graph. We further proposed a file caching scheme called **CFCM!**, which uses a simpler delivery algorithm, and is designed for the small cache size regime. We derived approximate expressions for the expected delivery rate of both schemes.



(a) Average Rates



(b) Additive Gains



(c) Multiplicative Gains

Figure 5.8: (a) shows both theoretical approximations and simulation results. (b) and (c) are based on computer simulations. Here $K = 50$ caches and $N = 1000$ files.

Because of the more restricted setup of the file caching problem, the schemes we proposed here are sub-optimal compared to the coded subfile caching schemes, but they still promise considerable gains over uncoded caching. Further, the delivery rate gap between CFC-CCD and the state-of-the-art decentralized coded subfile caching schemes shrinks considerably in the regime of large total storage capacity. These findings suggest that in scenarios where the implementation of subfile caching is difficult, it is still possible to effectively gain from network coding to improve the system performance.

Finally, we discussed the generalization of the proposed file caching schemes to subfile caching with an arbitrary number of subfiles per file. We observed that a

considerable portion of the performance loss due to the restrictions of file caching can be recovered by using a relatively small number of subfiles per file. While the construction of most of the coded subfile caching methods requires large numbers of subfiles per file, this observation gives grounds for the development of new decentralized subfile caching schemes that require a small number of subfiles without causing a considerable loss in the delivery rate.

Chapter 6

Conclusion

The field of coded caching has attracted significant attention over the last five years because of its promise in alleviation of network congestion due to the use of coding techniques. The pioneering works in the field [11, 12] mostly focused on the fundamental information-theoretic limits of coded caching, seeking the ultimate gains that it can possibly provide in reducing the server-to-cache delivery rate, under simplifying assumptions on the problem setting. Considerable amount of research has been devoted to adapt coded caching schemes to complex and realistic scenarios that are of practical importance, including extensions to nonuniform demands, finite file size regime, unequal file sizes, etc.

In this thesis, we investigated the problem of coded caching from the perspectives of optimization theory and graph theory, in an effort to address fundamental challenges for the use of coding techniques for caching. In particular, we first looked at the coded caching problem as an optimization problem. The work in Chapter 3 was among the earliest works that adapted an optimization perspective to the coded caching problem. We used a formulation in Chapter 3 for the case of uniform demands to address the problem of delivering demands that include duplicate requests. In addition to the importance of the proposed delivery algorithm in Chapter 3 for duplicate demands, that work paved our way to develop a more general formulation of the cache placement optimization problem which included nonuniform demands as well. This generalized formulation was the foundation of our work in Chapter 4.

The problem of optimal cache placement for coded caching with nonuniform demands is a central problem in the field, and several works in the literature attempted to address it. In particular, parallel to our work, some adapted an optimization formulation to solve the cache placement problem. A fundamental factor that distinguishes our work in Chapter 4 from the other works in the literature is that we followed an analytical approach to exactly solve the optimization problem, i.e., we neither left the optimization to numerical algorithms nor we used approximations to simplify the problem. What made it possible for us to address the problem analytically was the link we established between the cache placement optimization problem and the optimization of submodular set functions. We derived strong structures in the optimal placement strategy by carefully developing the connection between the two problems and aggregating the results developed for submodular function optimization into our original problem through the Lagrange duality theory. We believe our contributions in Chapter 4 present the state-of-the-art on cache allocation for coded caching with nonuniform demands and uncoded-prefetching as our comparison to the other works in the literature shows. Our proposed placement is also nearly optimal for that setting as it is suggested by our comparison to the information-theoretic lower bound.

Another major contribution of this thesis was the decentralized algorithm we proposed in Chapter 5 for coded caching under the requirement of keeping the files intact during placement. Looking at the spectrum of coded caching techniques in terms of the number of chunks they need to split every single file into during placement, this work represents one extreme of the spectrum where files remain intact. This is opposed to the other extreme in which the core coded caching algorithms represent, as they require every file to be split into a number of chunks that grows exponentially with the number of caches. In our proposed scheme, we used a randomized placement and a delivery based on the greedy clique cover algorithm, a well-known graph theoretical technique which is deployed in the index coding literature to construct coded messages. The randomized placement as well as the random nature of user demands makes the underlying side information graph random, which is the graph that the clique cover algorithm operates on. The complex random nature of the graph as well as the combinatorial nature of the clique cover algorithm makes the exact analysis of the delivery rate mathematically intractable.

A major contribution of Chapter 5 was the development of an approximate random graph model and the deployment of methods in random graph processes to derive a system of differential equations whose solution gives an approximation to the delivery rate of the proposed scheme. We argued that the approximation becomes more accurate for a larger ratio of the number of files to the number of caches. This ratio is usually large in most practical scenarios, making the analysis applicable in practice.

The coded caching problem is complex a problem, which is evident just from the fact that solving the delivery component alone is NP-hard in general. This complexity calls for the deployment of different techniques in various mathematical fields to deal with the problem. This thesis presented our effort to address coded caching under certain setups by looking at it from the optimization and graph theoretical perspectives.

Directions for future research

Our work in Chapter 4 on coded caching with nonuniform demands resulted in a nearly optimal performance for caching schemes with uncoded prefetching. There, we discussed a hypothesis on why there still exists a performance gap between the information theoretic lower bound and the rate of our proposed scheme. In particular, although the placement algorithm we proposed is optimal for the delivery algorithm in Algorithm 4, the delivery algorithm itself is suboptimal in the presence of duplicate demands as was dicussed in Chapter 3. We speculate that pairing up the modified delivery algorithm in [13] and a placement structured similar to the placement we derived in Chapter 4 might lead to a scheme that achieves the information theoretic lower bound on the rate of coded caching with uncoded prefetching for nonuniform demands. This is a promising future research direction.

Another important research direction is the estimation of the request probabilities of the files. Most of the works in the literature assume that these probabilities are known. This is a strong assumption as the prediction of future request probabilities is not a straightforward task in the context of wireless caching. This is because the cache hit ratio can be small in practice, i.e., the available data can be sparse. In that case, what would be a good learning algorithm for popularity pre-

diction? Also, if popularity prediction cannot be done reliably, how robust are the algorithms in the literature to such deviations from the assumed popularities? How can robustness to estimation errors be incorporated in designing coded caching schemes?

Another consequence of small cache hit ratio in wireless caching is that not all caches in the network would necessarily demand a file in each time step. Two relevant questions are how such a scenario can be modeled statistically, and what the expected delivery rate would be in such a scenario. Further, how would the design of caching schemes change if the set of caches active in the systems evolves over delivery period? We find these questions interesting from a theoretical point of view and of significant practical importance.

Bibliography

[1] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022," tech. rep., Cisco, 2018. → page 2

[2] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2015-2020," tech. rep., Cisco, 2016. → page 3

[3] Ericson, "Ericsson mobility report," tech. rep., Ericson, 2015. → page 3

[4] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, vol. 52, pp. 82–89, Aug. 2014. → page 3

[5] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. Inf. Theory*, vol. 59, pp. 8402–8413, Dec. 2013. → pages 3, 19

[6] G. Paschos, E. Bastug, I. Land, G. Caire, and M. Debbah, "Wireless caching: technical misconceptions and business barriers," *IEEE Commun. Mag.*, vol. 54, pp. 16–22, Aug. 2016. → pages 4, 20

[7] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. Inf. Theory*, vol. 57, pp. 1479–1494, Mar. 2011. → pages 4, 5, 42, 71

[8] Y. Birk and T. Kol, "Coding on demand by an informed source (iscod) for efficient broadcast of different supplemental data to caching clients," *IEEE Trans. Inf. Theory*, vol. 52, pp. 2825–2830, June 2006. → page 5

[9] N. Alon, E. Lubetzky, U. Stav, A. Weinstein, and A. Hassidim, "Broadcasting with side information," in *Foundations of Computer Science, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on*, pp. 823–832, Oct 2008. → page 5

[10] K. Shanmugam, A. G. Dimakis, and M. Langberg, "Local graph coloring and index coding," in *2013 IEEE International Symposium on Information Theory*, pp. 1152–1156, July 2013. → page 5

[11] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, pp. 2856–2867, May 2014. → pages 6, 9, 11, 12, 13, 16, 17, 22, 25, 29, 30, 36, 39, 93

[12] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," *IEEE/ACM Trans. Networking*, vol. 23, pp. 1029–1040, Aug. 2015. → pages 12, 14, 15, 17, 18, 19, 22, 25, 26, 27, 29, 36, 42, 67, 68, 70, 73, 87, 93

[13] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Transactions on Information Theory*, vol. 64, pp. 1281–1296, Feb 2018. → pages 15, 17, 22, 23, 33, 64, 65, 82, 89, 95, 121, 123

[14] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, pp. 1146–1158, Feb 2017. → pages 17, 18, 37, 42

[15] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Trans. Inf. Theory*, vol. 64, pp. 349–366, Jan 2018. → pages 37, 40, 42, 60, 62, 63, 64

[16] J. Hachem, N. Karamchandani, and S. Diggavi, "Content caching and delivery over heterogeneous wireless networks," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 756–764, Apr. 2015. → pages 18, 19, 37

[17] R. Pedarsani, M. A. Maddah-Ali, and U. Niesen, "Online coded caching," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 836–845, April 2016. → page 18

[18] N. Karamchandani, U. Niesen, M. A. Maddah-Ali, and S. N. Diggavi, "Hierarchical coded caching," vol. 62, pp. 3212–3229, June 2016. → pages 17, 18

[19] Kai Wan, D. Tuninetti, and P. Piantanida, "On the optimality of uncoded cache placement," in *2016 IEEE Information Theory Workshop (ITW)*, pp. 161–165, Sep. 2016. → page 17

[20] K. Wan, D. Tuninetti, and P. Piantanida, "On caching with more users than files," in *2016 IEEE International Symposium on Information Theory (ISIT)*, pp. 135–139, July 2016. → page 17

[21] K. Shanmugam, M. Ji, A. M. Tulino, J. Llorca, and A. G. Dimakis., "Finite-length analysis of caching-aided coded multicasting," *IEEE Trans. Inf. Theory*, vol. 62, pp. 5524–5537, Oct. 2016. → pages 17, 18, 42, 67, 72

[22] K. Shanmugam, A. M. Tulino, and A. G. Dimakis, "Coded caching with linear subpacketization is possible using ruzsa-szeméredi graphs," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1237–1241, June 2017. → pages 18, 68

[23] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Transactions on Information Theory*, vol. 64, pp. 5755–5766, Aug 2018. → page 18

[24] L. Tang and A. Ramamoorthy, "Coded caching schemes with reduced subpacketization from linear block codes," *IEEE Transactions on Information Theory*, vol. 64, pp. 3099–3120, April 2018. → page 18

[25] M. Cheng, J. Jiang, Q. Wang, and Y. Yao, "A generalized grouping scheme in coded caching," *IEEE Transactions on Communications*, vol. 67, pp. 3422–3430, May 2019. → page 17

[26] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "On the average performance of caching and coded multicasting with random demands," in *Proc. 11th International Symposium on Wireless Communications Systems (ISWCS)*, pp. 922–926, Aug. 2014. → pages 18, 37

[27] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Transactions on Information Theory*, vol. 63, pp. 3923–3949, June 2017.

[28] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Transactions on Information Theory*, vol. 63, pp. 3108–3141, May 2017. → page 18

[29] M. Mohammadi Amiri, Q. Yang, and D. Gündüz, "Decentralized caching and coded delivery with distinct cache capacities," *IEEE Transactions on Communications*, vol. 65, pp. 4657–4669, Nov 2017. → page 18

[30] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *IEEE Transactions on Information Theory*, vol. 62, pp. 7253–7271, Dec 2016. → page 18

[31] M. Mohammadi Amiri and D. Gündüz, "Fundamental limits of coded caching: Improved delivery rate-cache capacity tradeoff," *IEEE Transactions on Communications*, vol. 65, pp. 806–815, Feb 2017. → page 19

[32] J. Gómez-Vilardebó, "A novel centralized coded caching scheme with coded prefetching," *IEEE Journal on Selected Areas in Communications*, vol. 36, pp. 1165–1175, June 2018. → page 19

[33] Q. Yang and D. Gündüz, "Coded caching and content delivery with heterogeneous distortion requirements," *IEEE Transactions on Information Theory*, vol. 64, pp. 4347–4364, June 2018. → page 19

[34] S. Saeedi Bidokhti, M. Wigger, and R. Timo, "Noisy broadcast networks with receiver caching," *IEEE Transactions on Information Theory*, vol. 64, pp. 6996–7016, Nov 2018. → page 19

[35] A. Tang, S. Roy, and X. Wang, "Coded caching for wireless backhaul networks with unequal link rates," *IEEE Transactions on Communications*, vol. 66, pp. 1–13, Jan 2018. → page 19

[36] Y. Fadlallah, A. M. Tulino, D. Barone, G. Vettigli, J. Llorca, and J. M. Gorce, "Coding for caching in 5g networks," *IEEE Commun. Mag.*, vol. 55, pp. 106–113, Feb. 2017. → pages 19, 68

[37] J. Liao, K. Wong, M. R. A. Khandaker, and Z. Zheng, "Optimizing cache placement for heterogeneous small cell networks," *IEEE Communications Letters*, vol. 21, pp. 120–123, Jan 2017. → page 19

[38] X. Xu and M. Tao, "Modeling, analysis, and optimization of coded caching in small-cell networks," *IEEE Transactions on Communications*, vol. 65, pp. 3415–3428, Aug 2017.

[39] E. Ozfatura and D. Gündüz, "Mobility and popularity-aware coded small-cell caching," *IEEE Communications Letters*, vol. 22, pp. 288–291, Feb 2018.

[40] W. Han, A. Liu, W. Yu, and V. K. N. Lau, "Joint frequency reuse and cache optimization in backhaul-limited small-cell wireless networks," *IEEE Transactions on Wireless Communications*, vol. 17, pp. 6917–6930, Oct 2018. → page 19

[41] L. Li, G. Zhao, and R. S. Blum, "A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 1710–1732, thirdquarter 2018. → page 20

[42] M. A. Maddah-Ali and U. Niesen, "Coding for caching: fundamental limits and practical challenges," *IEEE Commun. Mag.*, vol. 54, pp. 23–29, Aug. 2016. → page 20

[43] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004. → pages 26, 46, 55, 108

[44] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. → page 31

[45] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 126–134, Mar. 1999. → pages 32, 59, 60

[46] R. Peck, *Statistics: Learning from Data*. Cengage Learning, 2014. → page 32

[47] S. Jin, Y. Cui, H. Liu, and G. Caire, "Structural properties of uncoded placement optimization for coded delivery," *CoRR*, vol. abs/1707.07146, 2017. → pages 37, 38, 39, 57

[48] A. M. Daniel and W. Yu, "Optimization of heterogeneous coded caching," *CoRR*, vol. abs/1708.04322, 2017. → pages 37, 38, 39

[49] S. A. Saberali, H. E. Saffar, L. Lampe, and I. F. Blake, "Adaptive delivery in caching networks," *CoRR*, vol. abs/1707.09662, 2017. → pages 39, 57

[50] S. Sahraei, P. Quinton, and M. Gastpar, "The optimal memory-rate trade-off for the non-uniform centralized caching problem with two files under uncoded placement," *CoRR*, vol. abs/1808.07964, 2018. → pages 40, 62, 63, 64

[51] J. Hachem, N. Karamchandani, and S. N. Diggavi, "Effect of number of users in multi-level coded caching," in *Proc. IEEE Int. Symp. Information Theory*, pp. 1701–1705, June 2015. → page 42

[52] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*. Springer Publishing Company, Incorporated, 2015. → page 49

[53] F. R. Bach, "Structured sparsity-inducing norms through submodular functions," in *Advances in Neural Information Processing Systems*, pp. 118–126, 2010. → pages 49, 50

[54] S. A. Saberali, H. E. Saffar, L. Lampe, and I. Blake, "Adaptive delivery in caching networks," *IEEE Communications Letters*, 2016. → page 64

[55] G. Vettigli, M. Ji, A. M. Tulino, J. Llorca, and P. Festa, "An efficient coded multicasting scheme preserving the multiplicative caching gain," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 251–256, Apr. 2015. → pages 68, 69

[56] D. A. Mastin, *Analysis of approximation and uncertainty in optimization*. PhD thesis, Massachusetts Institute of Technology, 2015. Available at http://hdl.handle.net/1721.1/97761. → pages 69, 73, 117

[57] U. Niesen and M. A. Maddah-Ali, "Coded caching for delay-sensitive content," in *2015 IEEE International Conference on Communications (ICC)*, pp. 5559–5564, June 2015. → page 69

[58] R. M. Karp, *Reducibility among Combinatorial Problems*, pp. 85–103. Boston, MA: Springer US, 1972. → page 72

[59] N. C. Wormald, "The differential equation method for random graph processes and greedy algorithms," *Lectures on approximation and randomized algorithms*, pp. 73–155, 1999. → pages 74, 75, 113, 114

[60] J. Díaz and D. Mitsche, "Survey: The cook-book approach to the differential equation method," *Comput. Sci. Rev.*, vol. 4, pp. 129–151, Aug. 2010. → pages 75, 113

[61] P. J. Cameron, *Combinatorics: topics, techniques, algorithms*. Cambridge University Press, 1994. → pages 82, 122

[62] N. C. Wormald, "Differential equations for random processes and random graphs," *Ann. Appl. Probab.*, vol. 5, pp. 1217–1235, Nov. 1995. → page 113

[63] S. Yang and R. W. Yeung, "Batched sparse codes," *IEEE Trans. Inf. Theory*, vol. 60, pp. 5322–5346, Sep. 2014. → page 113

[64] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge University Press, 2008. → page 113

Appendix A

Proofs of Chapter 4

A.1 Proof of Proposition 3

Assume that there exists an optimal placement $\mathcal{P}^*$ of files for which there exist distinct subsets $\mathcal{S}_1$ and $\mathcal{S}_2 : |\mathcal{S}_1| = |\mathcal{S}_2|$ such that $x^n_{\mathcal{S}_1} \neq x^n_{\mathcal{S}_2}$. Since the popularity of files is identical for the different caches, the delivery rate remains the same if we use any permutation $\text{perm}(k)$ of the cache labels in the placement parameters $\{x^n_{\mathcal{S}}\}_{\mathcal{S}\subset[K]}$. We denote the placement over the permuted cache labels by $\mathcal{P}^*_{\text{perm}}$. More specifically, we relabel cache k to $\text{perm}(k)$ for $k \in [K]$, and use the placement parameters over the relabeled caches. In particular, if under $\mathcal{P}^*$ we have $x^n_{\mathcal{S}}\big|_{\mathcal{P}^*} = c$, then under $\mathcal{P}^*_{\text{perm}}$, we set $x^n_{\text{perm}(\mathcal{S})}\big|_{\mathcal{P}^*_{\text{perm}}} = c$, where $\text{perm}(\mathcal{S}) = \{\text{perm}(k) \mid k \in \mathcal{S}\}$. There exists $K!$ permutations of K cache labels, leading to placements $\mathcal{P}^*_{\text{perm},i}, i = 1, \ldots, K!$, all with the optimal delivery rate R^*.

Hence:

$$R^* = \frac{1}{K!}\sum_{i=1}^{K!} r(\boldsymbol{x})\big|_{\mathcal{P}^*_{\text{perm},i}} = \frac{1}{K!}\sum_{i=1}^{K!}\sum_{\substack{\mathcal{S}:\mathcal{S}\subset\mathcal{K}\\ \mathcal{S}\neq\emptyset}} \mathbb{E}_{\boldsymbol{d}}\left(\max_{k\in\mathcal{S}} x^{d_k}_{\mathcal{S}\backslash k}\big|_{\mathcal{P}^*_{\text{perm},i}}\right)$$

$$= \mathbb{E}_{\boldsymbol{d}}\left(\sum_{\substack{\mathcal{S}:\mathcal{S}\subset\mathcal{K}\\ \mathcal{S}\neq\emptyset}} \frac{1}{K!}\sum_{i=1}^{K!}\max_{k\in\mathcal{S}} x^{d_k}_{\mathcal{S}\backslash k}\big|_{\mathcal{P}^*_{\text{perm},i}}\right)$$

$$\geq \mathbb{E}_{\boldsymbol{d}}\left(\sum_{\substack{\mathcal{S}:\mathcal{S}\subset\mathcal{K}\\ \mathcal{S}\neq\emptyset}} \max_{k\in\mathcal{S}} \frac{1}{K!}\sum_{i=1}^{K!} x^{d_k}_{\mathcal{S}\backslash k}\big|_{\mathcal{P}^*_{\text{perm},i}}\right) = \mathbb{E}_{\boldsymbol{d}}\left(\sum_{\substack{\mathcal{S}:\mathcal{S}\subset\mathcal{K}\\ \mathcal{S}\neq\emptyset}} \max_{k\in\mathcal{S}} \bar{x}^{d_k}_{\mathcal{S}\backslash k}\right),$$

where we defined $\bar{x}^n_{\mathcal{S}} = \frac{1}{K!}\sum_{i=1}^{K!} x^n_{\mathcal{S}}\big|_{\mathcal{P}^*_{\text{perm},i}}$ and used the convexity of the $\max$ function. Notice that the RHS is the expected rate when for each file n and subset $\mathcal{S}$, we use the average of the corresponding placement parameters over all permutations of the optimal placement. Because of symmetry, $\bar{x}^n_{\mathcal{S}}$ has the property that $\bar{x}^n_{\mathcal{S}_1} = \bar{x}^n_{\mathcal{S}_2}$ if $|\mathcal{S}_1| = |\mathcal{S}_2|$. From the facts that i) the LHS is the optimal rate, ii) the placement parameters $\bar{x}^n_{\mathcal{S}}$ use the same amount of cache storage as $\mathcal{P}^*$ based on eq. (4.3) and iii) the sum of $\bar{x}^n_{\mathcal{S}}$ over all subsets $\mathcal{S}\subset[K]$ is 1, we conclude that the rate in the RHS must also be equal to R^*. This implies that there exists an optimal placement with the property that $x^n_{\mathcal{S}} = x^n_s$ for all $\mathcal{S} : |\mathcal{S}| = s$, which completes the proof.

A.2 Proof of Lemma 2

Based on Proposition 5, the extreme points of the unit ball $f_{\text{c}} \leq 1$ are closely connected to the set of stable inseparable subsets of $[(K-1)N]$ with regard to F_{c}. We first argue that all subsets of $[(K-1)N]$ are stable. Consider a set $A \subset [(K-1)N]$. Augment A with a new object i to get $A\cup\{i\}$. Without loss of generality, let $s^{-1}(\{i\}) = \hat{s}$. Since $\tilde{g} = \{i\}$ belongs to $\mathcal{G}_{\hat{s}+1}$ with $\eta_{\tilde{g}} > 0$ and it does not intersect with A, we have $F_{\text{c}}(A\cup\{i\}) > F_{\text{c}}(A)$. Hence, any set $A \subset [(K-1)N]$ is stable with respect to F_{c}. Consequently, every subset of $[N]_{(s-1)N}$ for $s\in[K-1]$ is also stable.

To find the inseparable sets, consider $A \subset [(K-1)N]$. Let $B_s = \{i \in A \mid s^{-1}(\{i\}) = s\}$. A necessary condition for A to be inseparable is to have only one nonempty B_s. To show this, partition A to subsets B_s, $s \in [K-1]$. Notice that each group $\tilde{g} \subset \tilde{\mathcal{G}}$ is a subset of exactly one $[N]_{(s-1)N}$, $s \in [K-1]$. Hence, if two or more subsets B_s are nonempty, then $F_{\mathrm{c}}(A) = \sum_{B_s \neq \emptyset} F_{\mathrm{c}}(B_s)$ and A is separable. Now, consider the case where only one B_s, say $B_{\hat{s}}$, is nonempty. In this case, $A \subset [N]_{(\hat{s}-1)N}$ and A can only have nonempty intersections with sets $\tilde{g} \in \tilde{\mathcal{G}}_{\hat{s}+1}$. Since $\hat{s} \geq 1$, for any partitioning of A to $P_1, \ldots, P_J$ for some J, there is at least one group $\tilde{g} \in \tilde{\mathcal{G}}_{\hat{s}+1}$ with $|\tilde{g}| \geq 2$ that intersects with both P_i and P_j for every pair $i \neq j$. Hence, $F_{\mathrm{c}}(A) < \sum_{i=1}^{J} F_{\mathrm{c}}(P_i)$. As a result, the set of all stable inseparable subsets of $[(K-1)N]$ with regard to F_{c} is $\mathcal{A} = \{A \mid A \subset [N]_{(s-1)N}, s \in [K-1]\}$.

According to Proposition 5, the support of every extreme point of the norm-ball of f_{c} belongs to $\mathcal{A}$. Further, the nonzero entries of the extreme point vector that corresponds to $A \in \mathcal{A}$ is either of $\pm 1/F_{\mathrm{c}}(A)$. Using Proposition 6:

$$F_{\mathrm{c}}(A) = \sum_{\tilde{g} \subset \tilde{\mathcal{G}}: A \cap \tilde{g} \neq \emptyset} \eta_{\tilde{g}} = \sum_{\substack{\tilde{g} \subset \tilde{\mathcal{G}}_{s^{-1}(A)}: \\ A \cap \tilde{g} \neq \emptyset}} \eta_{\tilde{g}} = \frac{K - s^{-1}(A)}{s^{-1}(A)+1} \left(1 - (1 - P(A))^{s^{-1}(A)+1}\right) \tag{A.1}$$

where we used the facts that 1) for $A \in \mathcal{A}$, all entries of A belong to only one $[N]_{(s-1)N}$, so $s^{-1}(A)$ and $g^{-1}(A)$ are well defined, 2) $\eta_{\tilde{g}} = \frac{K - s^{-1}(\tilde{g})}{s^{-1}(\tilde{g})+1} \pi^{g^{-1}(\tilde{g})}_{s^{-1}(\tilde{g})}$ and 3) $\sum_{\substack{\tilde{g} \subset \tilde{\mathcal{G}}_{s^{-1}(A)}: \\ A \cap \tilde{g} \neq \emptyset}} \eta_{\tilde{g}}$ equals the probability of having a demand vector with at most $s^{-1}(A)+1$ distinct files from $[N]$ that has at least one file in $g^{-1}(A)$. The use of $\mathcal{A}$ and (A.1) in Proposition 5 and scaling the radius of the ball from 1 to t results in (4.17).

A.3 Proof of Theorem 1

To prove Theorem 1, we first prove the following lemma.

Lemma 5. *The extreme points of the region $[f_c \leq t]^+$ defined by (4.13b) and*

(4.13d) are the origin and points of the form

$$\frac{t}{\frac{K-s}{s+1}\left[1-(1-P(A))^{s+1}\right]}\boldsymbol{v}, \tag{A.2}$$

where vector $\boldsymbol{v} \in \{0,1\}^{KN}$*, Supp*$(\boldsymbol{v}) = A$*, and set* A *is a subset of* $[N]_{(s-1)N}$ *for an* $s \in [K-1]$.

Proof. To obtain the extreme points of $[f_{\mathrm{c}} \leq t]^+$ we begin with the extreme points of the norm-ball $f_{\mathrm{c}} \leq t$ and remove the ones that have negative coordinates as they do not belong to the non-negative orthant. Further, $[f_{\mathrm{c}} \leq t]^+$ has extra extreme points that result from the intersection of $f_{\mathrm{c}} \leq t$ and planes $y_s^n = 0$. Norm-ball f_{c} is symmetric w.r.t. every plane $y_s^n = 0$ and hence the extreme point resulting from the intersection of the norm-ball and such a plane will either be an extreme point of the norm ball or the midpoint of two extreme points of the norm-ball with y_s^n coordinates of $+1$ and -1. In the latter case, the y_s^n coordinate of the extreme point of $[f_{\mathrm{c}} \leq t]^+$ will be 0. Either case, Supp$(\boldsymbol{v})$ will still be a subset of $[N]_{(s-1)N}$ for an $s \in [K-1]$. If there is no nonzero entry left in the coordinates of the extreme point of the intersection, which is the case when all planes $y_s^n = 0$ intersect, the resulting point is the origin. □

We now prove Thereom 1.

At optimality, we have $f_{\mathrm{c}}(\tilde{\boldsymbol{y}}^*) = t^*$, as otherwise the objective can be decreased by replacing t^* with $f_{\mathrm{c}}(\tilde{\boldsymbol{y}}^*)$, which contradicts the optimality of t^*.

The objective function (4.13a) calculated at an extreme point of form (A.2) with nonzero parameters for $s = s_o$ and A is $[1+\frac{\sum_{n\in g^{-1}(A)}(s_o/K-1+\alpha_n^*)\gamma - Kp_n\alpha_n^*}{(K-s_o)/(s_o+1)(1-(1-P(A))^{s_o+1}}]t$, which is a factor of t. Denote the denominator of (A.2) by $t^u(s,A)$, i.e., $t^u(s,A) = \frac{K-s}{s+1}[1-(1-P(A))^{s+1}]$. Notice that for $t = t^u(s_o,A)$, the extreme point of $[f_{\mathrm{c}} \leq t]^+$ that corresponds to s_o and A, is of form $y_s^n = 1$ for $s = s_o, n \in g^{-1}(A)$, and $y_s^n = 0$ otherwise. These parameters satisfy (4.13b)-(4.13d) and are feasible.

Hence, for any $s_o \in [K-1]$ and $A \subset [N]_{(s_o-1)N}$, we have

$$\left[1+\frac{\sum_{n\in g^{-1}(A)}(\frac{s_o}{K}-1+\alpha_n^*)\gamma - Kp_n\alpha_n^*}{\frac{K-s_o}{s_o+1}(1-(1-P(A))^{s_o+1}}\right]t^u(s_o,A)$$
$$\geq t^*+\sum_{n=1}^{N}\sum_{s=1}^{K-1}[(\frac{s}{K}-1+\alpha_n^*)\gamma - Kp_n\alpha_n^*](y_s^n)^*.$$

Also, the objective function (4.13a) finds its minimum over the set $[f_{\text{c}} \leq t^*]^+$ at one of the extreme points of $[f_{\text{c}} \leq t^*]^+$. The extreme points of $[f_{\text{c}} \leq t^*]^+$ are in the form of (A.2). Denote the extreme point with the smallest objective by $\bar{\boldsymbol{y}}$ and its corresponding s and A by s^* and A^*. Since the feasible set of problem (4.13) is a subset of $[f_{\text{c}} \leq t^*]^+$, we have

$$t^*+\sum_{n=1}^{N}\sum_{s=1}^{K-1}[(\frac{s}{K}-1+\alpha_n^*)\gamma - Kp_n\alpha_n^*](y_s^n)^*$$
$$\geq \left[1+\frac{\sum_{n\in g^{-1}(A^*)}(\frac{s^*}{K}-1+\alpha_n^*)\gamma - Kp_n\alpha_n^*}{\frac{K-s^*}{s^*+1}(1-(1-P(A^*))^{s^*+1}}\right]t^*.$$

Combining the last two equations concludes $t^u(s^*,A^*) \geq t^*$ and equivalently $\frac{t^*}{t^u(s^*,A^*)} \leq 1$. As a result, the extreme point $\bar{\boldsymbol{y}}$ also satisfies (4.13c) and is feasible to (4.13). Given that it has the smallest objective among the extreme points of $[f_{\text{c}} \leq t^*]^+$, it is optimal, i.e., $\bar{\boldsymbol{y}} = \boldsymbol{y}^*$.

Now, the objective $\left[1+\frac{\sum_{n\in g^{-1}(A^*)}(\frac{s^*}{K}-1+\alpha_n^*)\gamma - Kp_n\alpha_n^*}{\frac{K-s^*}{s^*+1}(1-(1-P(A^*))^{s^*+1}}\right]t^*$ is linear in t^*. Since $t^* \leq t^u(s^*,A^*)$ and $t^* = t^u(s^*,A^*)$ is achievable, at optimality we either have $t^* = 0$ or $t^* = t^u(s^*,A^*)$, depending on the sign of the coefficient of t^*. In the former case, we either have cached all files, i.e., $\forall n : (y_K^n)^* = 1, (y_s^n)^* = 0, s < K$, or no file is cached at all, i.e., $\forall n : (y_0^n)^* = 1, (y_s^n)^* = 0, s > 1$, as in both cases the rate f_{c} due to delivery of the content cached in at least one cache is 0.

In the case of $t^* = t^u(s^*,A^*)$, for $s \in [K-1]$ we have $(y_s^n)^* = 1, s = s^*, n \in g^{-1}(A^*)$ and $(y_s^n)^* = 0$ otherwise. Together with Lemma 1, this concludes that at optimality $(z^n)^* = 1 - \sum_{s=1}^{K-1}(y_s^n)^* \in \{0,1\}$. Hence, when $(z^n)^* = 1$ we have $(y_0^n)^* = 1$ and $(y_K^n)^* = 0$ if $Kp_n < \gamma$ and $(y_0^n)^* = 0$ and $(y_K^n)^* = 1$ if $Kp_n \geq \gamma$.

A.4 Proof of Lemma 3

To prove Lemma 3, we first show the following result:

Lemma 6. *The capacity constraint (4.8b) in* RMSC *is satisfied with equality at optimality, i.e., no storage remains unused.*

Proof. Assume that for storage capacity M, there is an optimal solution $\boldsymbol{y}^*$ with $m(\boldsymbol{y}^*) + \epsilon N = M$, where $\epsilon > 0$. Then, construct solution $\boldsymbol{y'}$ with $y'^n_s = (1 - \epsilon)y^n_s, s < K$ and $y'^n_K = (1 - \epsilon)y^n_K + \epsilon$. Essentially, $\boldsymbol{y'}$ splits every file into two parts of lengths $(1 - \epsilon)F$ and ϵF. It uses $\boldsymbol{y}^*$ for the placement of the parts of length $(1 - \epsilon)F$ and caches the other ϵF parts on every cache. This uses storage $(1 - \epsilon)m(\boldsymbol{y}^*) + \epsilon N = M - \epsilon(M - \epsilon N) < M$, which implies that the storage constraint is satisfied for $\boldsymbol{y'}$. However, $r(\boldsymbol{y'}) = (1-\epsilon)r(\boldsymbol{y}^*)+\epsilon \times 0 < r(\boldsymbol{y}^*)$. This contradicts the optimality of $\boldsymbol{y}^*$. Hence, the optimal solution of RMSC must satisfy the capacity constraint by equality. □

The first property follows from the shadow price interpretation of the Lagrange multipliers for inequality constraints [43, Section 5.6]. In particular, let denote the optimal solutions to (4.8) with storage budgets M_1 and $M_2 < M_1$ by $\boldsymbol{y}^*_1$ and $\boldsymbol{y}^*_2$. Also, let γ^*_1 and γ^*_2 be the optimal dual parameters. If $\gamma^*_1 > \gamma^*_2$, the Lagrangian minimization $\min_{\boldsymbol{y}} r(\boldsymbol{y}) + \gamma m(\boldsymbol{y})$ results in $m(\boldsymbol{y_1}) \leq m(\boldsymbol{y_2})$. Given that no storage remains unused at optimality, this contradicts the assumption $M_2 < M_1$. Hence, we should have $\gamma^*_1 > \gamma^*_2$.

The second property follows from the fact that the set $\gamma \geq 0$ in the Lagrangian minimization problem (4.9), or equivalently in JRSM, is continuous, while set $\mathcal{Y}^*$ (or $\mathcal{M}$) is finite. Hence, a range of values of γ must map to the same storage $M \in \mathcal{M}$ and they are all dual optimal.

To prove the third property consider $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}^*$ that correspond to two consecutive storage values $M_1 = m(\boldsymbol{y}_1)$ and $M_2 = m(\boldsymbol{y}_2)$. Without loss of generality assume that $M_1 < M_2$. Clearly, $M \notin \mathcal{M}$ for any $M \in (M_1, M_2)$. Now, notice that i) each capacity M must correspond to some γ^* in the dual problem, ii) for each $\gamma \geq 0$ there is an optimal solution to JRSM in $\mathcal{Y}^*$ and a corresponding storage value in $\mathcal{M}$, hence none of those solutions uses an amount of storage $M \notin \mathcal{M}$ and iii) based on Lemma 6, at optimality all the available storage must

be used, and iv) based on property 1, γ^* is nondecreasing in M. These point conclude that the optimal dual parameter for any $M \in [M_1, M_2]$ must belong to $\{\gamma^*_{M_1}, \gamma^*_{M_2}\}$, where γ^*_m represents the optimal dual parameter for capacity m and based on property 1, $\gamma^*_{M_2} \leq \gamma^*_{M_1}$. More specifically, point (iv) requires $\gamma^*_M = \gamma^*_{M_1}$ for $M \in [M_1, M']$ and $\gamma^*_M = \gamma^*_{M_2}$ for $M \in (M', M_2]$, for some $M_1 \leq M' \leq M_2$. However, we must have $\gamma^*_{M_2} = \gamma^*_{M_1}$ as otherwise any $\gamma^*_{M_2} < \gamma'' < \gamma^*_{M_1}$ corresponds to a value of $M \notin \mathcal{M}$, which contradicts point (ii). This concludes property 3, i.e., for two consecutive values $M_1, M_2 \in \mathcal{M}, M_1 < M_2$, all capacities $M_1 \leq M \leq M_2$ correspond to the same dual parameter γ^*. Further, we can derive the optimal dual parameter for $m(\boldsymbol{y}_1) \leq M \leq \tilde{m}(\boldsymbol{y}_2)$ as it satisfies $r(\boldsymbol{y}_1) + \gamma^* m(\boldsymbol{y}_1) = r(\boldsymbol{y}_2) + \gamma^* m(\boldsymbol{y}_2) = L^*$. Hence,

$$\gamma^* = \frac{r(\boldsymbol{y}_1) - r(\boldsymbol{y}_2)}{m(\boldsymbol{y}_2) - m(\boldsymbol{y}_1)}. \tag{A.3}$$

A.5 Proof of Theorem 2

To prove Theorem 2, we consider two cases:

Case I ($M \in \mathcal{M}$) This case is straightforward because of the zero duality gap in the primal-dual framework established in Section 4.3.2. In particular, vector $\boldsymbol{y}^*_{\text{JRSM}} \in \mathcal{Y}^*$ with $m(\boldsymbol{y}^*_{\text{JRSM}}) = M$ is also optimal to RMSC. [1]

Case II ($M \notin \mathcal{M}$) To derive the optimal solution of RMSC for $M \notin \mathcal{M}$, we use Lemma 3. Let $\boldsymbol{y}_1, \boldsymbol{y}_2 \in \mathcal{Y}^*$ be the solutions corresponding to the two consecutive storage values $m(\boldsymbol{y}_1), m(\boldsymbol{y}_2) \in \mathcal{M}$ such that $m(\boldsymbol{y}_1) < M < m(\boldsymbol{y}_2)$. Let γ^* and L^* be the corresponding optimal dual parameter and Lagrangian value,

[1] A direct proof for optimality of $\boldsymbol{y}^*_{\text{JRSM}}$ with $m(\boldsymbol{y}^*_{\text{JRSM}}) = M$ for RMSC is as follows. Based on Lemma 6, the optimal solution of RMSC satisfies the capacity constraint with equality. Now, assume that $\boldsymbol{y}^*_{\text{JRSM}}$ is not optimal for the RMSC problem. This means that $r(\boldsymbol{y}^*_{\text{RMSC}}) < r(\boldsymbol{y}^*_{\text{JRSM}})$. However, since $m(\boldsymbol{y}^*_{\text{RMSC}}) = m(\boldsymbol{y}^*_{\text{JRSM}}) = M$, this implies that $r(\boldsymbol{y}^*_{\text{RMSC}}) + \gamma^* m(\boldsymbol{y}^*_{\text{RMSC}}) < r(\boldsymbol{y}^*_{\text{JRSM}}) + \gamma^* m(\boldsymbol{y}^*_{\text{JRSM}})$. The last result contradicts the optimality of $\boldsymbol{y}^*_{\text{JRSM}}$ for JRSM. Hence, $\boldsymbol{y}^*_{\text{JRSM}}$ must be optimal for the RMSC problem.

respectively. Since $m(\cdot)$ is linear, for any given storage $m(\boldsymbol{y}_1) < M < m(\boldsymbol{y}_2)$, there exists a convex combination of $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ that uses storage M. We show that the same convex combination also minimizes the Lagrangian for γ^*. In that case, we are back to a case similar to Case I, and the same argument used there requires the convex combination to also optimize RMSC.

Consider $\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2} \triangleq \theta\boldsymbol{y}_1 + (1-\theta)\boldsymbol{y}_2$ for $0 < \theta < 1$. For the θ for which $m(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) = M$, we need to show that $\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}$ minimizes the Lagrangian for γ^*. This is equivalent to showing that $r(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) + \gamma^* m(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) = L^*$.[2] Since $m(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) = \theta m(\boldsymbol{y}_1) + (1-\theta)m(\boldsymbol{y}_2)$, it is sufficient to show that $r(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) = \theta r(\boldsymbol{y}_1) + (1-\theta)r(\boldsymbol{y}_2)$ to prove $r(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) + \gamma^* m(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) = L^*$. To show $r(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) = \theta r(\boldsymbol{y}_1) + (1-\theta)r(\boldsymbol{y}_2)$, notice that each $\boldsymbol{y} \in \mathcal{Y}^*$ has nonzero parameters y_s^n for at most two values of s, one $s = 0$ and one $s \geq 1$. Assume that $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ have nonzero entries respectively for $s_1 > 0$ and $s_2 > 0$ and possibly for $s = 0$. Let n_1 and n_2 be the largest indexes n with nonzero $y_{s_1}^n$ and $y_{s_2}^n$ in the respective two solutions. Notice that since $\boldsymbol{y}_i \in \mathcal{Y}^*$, having $y_{s_i}^n > 0$ implies $y_{s_i}^n = 1$. We consider two cases of $s_1 \neq s_2$ and $s_1 = s_2$. In the former case, $r(\boldsymbol{y}_1) = K\sum_{n=1}^{N} p_n {y_1}_0^n + \frac{K-s_1}{s_1+1}(1-(1-\sum_{n=1}^{n_1} p_n)^{s_1+1})$, $r(\boldsymbol{y}_2) = K\sum_{n=1}^{N} p_n {y_2}_0^n + \frac{K-s_2}{s_2+1}(1-(1-\sum_{n=1}^{n_2} p_n)^{s_2+1})$ and $r(\theta\boldsymbol{y}_1 + (1-\theta)\boldsymbol{y}_2)) = K\sum_{n=1}^{N} p_n(\theta {y_1}_0^n + (1-\theta){y_2}_0^n) + \frac{K-s_1}{s_1+1}(1-(\sum_{n=1}^{n_1} p_n)^{s_1+1})\theta + \frac{K-s_2}{s_2+1}(1-(\sum_{n=1}^{n_2} p_n)^{s_2+1})(1-\theta) = \theta\tilde{r}(\boldsymbol{y}_1) + (1-\theta)r(\boldsymbol{y}_2)$. For the case of $s_1 = s_2 = s_o$,

[2]The equivalence results from the fact that if $r(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) + \gamma^* m(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) < L^*$, then $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ could not be optimal for γ^*, which is a contradiction.

since $m(\boldsymbol{y}_1) < m(\boldsymbol{y}_2)$, we must have $n_2 > n_1$. Hence, for the rates we have

$$\begin{aligned}
r(\boldsymbol{y}_1) &= K \sum_{n=1}^{N} p_n y^n_{0\,1} + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}} \pi^g_{s_o+1} \max_{n \in g} y^n_{s_o 1} \\
&= K \sum_{n=1}^{N} p_n y^n_{0\,1} + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}, g \cap [n_1] \neq \emptyset} \pi^g_{s_o+1} \\
r(\boldsymbol{y}_2) &= K \sum_{n=1}^{N} p_n y^n_{0\,2} + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}} \pi^g_{s_o+1} \max_{n \in g} y^n_{s_o 2} \\
&= K \sum_{n=1}^{N} p_n y^n_{0\,2} + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}, g \cap [n_2] \neq \emptyset} \pi^g_{s_o+1}
\end{aligned}$$

and

$$\begin{aligned}
r(\theta \boldsymbol{y}_1 + (1-\theta)\boldsymbol{y}_2) &= K \sum_{n=1}^{N} p_n (\theta y^n_{0\,1} + (1-\theta) y^n_{0\,2}) \\
&\quad + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}} \pi^g_{s_o+1} \max_{n \in g} \theta y^n_{s_o 1} + (1-\theta) y^n_{s_o 2} \\
&= K \sum_{n=1}^{N} p_n (\theta y^n_{0\,1} + (1-\theta) y^n_{0\,2}) \\
&\quad + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}, g \cap [n_1] \neq \emptyset} \pi^g_{s_o+1} (\theta + (1-\theta)) \\
&\quad + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}, g \cap [n_1] = \emptyset, g \cap [n_2 - n_1]_{n_1} \neq \emptyset} \pi^g_{s_o+1} (1-\theta) \\
&= \theta \left[K \sum_{n=1}^{N} p_n y^n_{1 0} + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}, g \cap [n_1] \neq \emptyset} \pi^g_{s_o+1} \right] \\
&\quad + (1-\theta) \left[K \sum_{n=1}^{N} p_n y^n_{2 0} + \frac{K - s_o}{s_o + 1} \sum_{g \in \mathcal{G}_{s_o+1}, g \cap [n_2] \neq \emptyset} \pi^g_{s_o+1} \right] \\
&= \theta \tilde{r}(\boldsymbol{y}_1) + (1-\theta) r(\boldsymbol{y}_2)
\end{aligned}$$

where we used the fact that if $g \cap [n_1] \neq \emptyset$, then $n_2 > n_1$ implies $g \cap [n_2] \neq \emptyset$.

This completes the proof of the third feature as we now have

$$\begin{aligned} r(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) + \gamma^* m(\boldsymbol{y}_{\theta;\boldsymbol{y}_1,\boldsymbol{y}_2}) &= \theta\tilde{r}(\boldsymbol{y}_1) + (1-\theta)r(\boldsymbol{y}_2) + \gamma^*[\theta\tilde{m}(\boldsymbol{y}_1) + (1-\theta)m(\boldsymbol{y}_2)] \\ &= \theta[r(\boldsymbol{y}_1) + \gamma^* m(\boldsymbol{y}_1)] + (1-\theta)[r(\boldsymbol{y}_2) + \gamma^* m(\boldsymbol{y}_2)] \\ &= \theta L^* + (1-\theta)L^* = L^*. \end{aligned}$$

Appendix B

Additional Material and Proofs of Chapter 5

B.1 Wormald's Theorem

In this appendix, we present Wormald's theorem [59, Theorem 5.1] and introduce the notations that we use in the proofs of Theorems 3 and 4.[1]

Consider a sequence of random processes indexed by n, $n = 1, 2, \ldots$.[2] For each n, the corresponding process is a probability space denoted by $(Q_0^{(n)}, Q_1^{(n)}, \ldots)$, where each $Q_i^{(n)}$ takes values from a set $S^{(n)}$. The elements of the space are $(q_0^{(n)}, q_1^{(n)}, \ldots)$, where $q_i^{(n)} \in S^{(n)}$. Let $H_t^{(n)}$ represent the random history of process n up to time t and $h_t^{(n)}$ represent a realization of $H_t^{(n)}$. Also, we denote by $S^{(n)+}$ the set of all $h_t^{(n)} = (q_0^{(n)}, \ldots, q_t^{(n)})$, where $q_i^{(n)} \in S^{(n)}$, $t = 0, 1, \ldots$. For simplicity of notation, the dependence on n is usually dropped in the following.

Now, consider a set of variables $W_1(t), \ldots, W_a(t)$ defined on the components of the processes. Let $w_i(h_t)$ denote the value of $W_i(t)$ for the history $h_t^{(n)}$. We are interested in analyzing the behavior of these random variables throughout the

[1] Wormald's theorem was introduced in [62, Theorem 1]. The most general setting of the theorem was established in [59, Theorem 5.1]. A simplified version of the theorem is provided in [60, Section 3], and examples of its application can be found in [59], [60, Section 4], [63] and [64, Section C.4].

[2] For instance, the different processes can be the outcomes of a procedure implemented on graphs with different numbers of vertices n. Then, each process associates with one of the graphs.

process.

Theorem 5 (Wormald's Theorem [59, THEOREM 5.1])**.** *For $1 \leq l \leq a$, where a is fixed, let $w_l : S^{(n)+} \to \mathbb{R}$ and $f_l : \mathbb{R}^{a+1} \to \mathbb{R}$, such that for some constant c_0 and all l, $|w_l(h_t)| < c_0 n$ for all $h_t \in S^{(n)+}$ for all n. Assume the following three conditions hold, where in (ii) and (iii), D is some bounded connected open set containing the closure of*

$$\{(0, z_1, \ldots, z_a) : \mathbb{P}(W_l(0) = z_l n, l = 1, \ldots, a) \neq 0\}$$

for some n, and $T_D(W_1, \ldots, W_a)$ is the minimum t such that $\left(\frac{t}{n}, \frac{W_1(t)}{n}, \ldots, \frac{W_a(t)}{n}\right) \notin D$.

(i) (Boundedness hypothesis) For some functions $\beta = \beta(n) \geq 1$ and $\gamma = \gamma(n)$, the probability that

$$\max_{l=1,\ldots,a} |W_l(t+1) - W_l(t)| \leq \beta$$

conditional upon H_t, is at least $1 - \gamma$ for $t < T_D$.

(ii) (Trend hypothesis) For some function $\lambda_1 = \lambda_1(n) = o(1)$, for all $l \leq a$

$$\left|\mathbb{E}\left(W_l(t+1) - W_l(t) | H_t\right) - f_l\left(\frac{t}{n}, \frac{W_1(t)}{n}, \ldots, \frac{W_a(t)}{n}\right)\right| \leq \lambda_1$$

for $t < T_D$.

(iii) (Lipschitz hypothesis) Each function f_l is continuous and satisfies a Lipschitz condition on $D \cap \{(t, z_1, \ldots, z_a) : t \geq 0\}$, with the same Lipschitz constant for each l.

Then, the following are true.

(a) For $(0, \hat{z}_1, \ldots, \hat{z}_a) \in D$, the system of differential equations defined by

$$\frac{dz_l}{dx} = f_l(x, z_1, \ldots, z_a), \quad l = 1, \ldots, a$$

has a unique solution that passes through $z_l(0) = \hat{z}_l,\ l = 1, \ldots, a$, which extends to points arbitrarily close to the boundary of D.

(b) Let $\lambda > \lambda_1 + c_0 n\gamma$ with $\lambda = o(1)$. For a sufficiently large constant C, with probability $1 - O(n\gamma + \frac{\beta}{\lambda} e^{-n\lambda^3/\beta^3})$

$$W_l(t) = n z_l(t/n) + O(\lambda n)$$

uniformly for all $0 \le t \le \sigma n$ and for each l, where $z_l(x)$ is the solution in (a) with $\hat{z}_l = \frac{1}{n} W_l(0)$, and $\sigma = \sigma(n)$ is the supremum of those x to which the solution can be extended before reaching within l^∞-distance $C\lambda$ of the boundary of D.

Hence, functions $z_l(x)$ model the behavior of $\frac{W_l(nx)}{n}$ for each n, and the solution to the system of differential equations provides a deterministic approximation to the dynamics of the process.

Remark 10. *In the statement of the theorem, variables W_l and time t are normalized by n. This is because in many applications, this normalization leads to only one set of differential equations for all n, instead of different systems for each n. Also, the asymptotics denoted by O are for $n \to +\infty$, and the term "uniformly" in (b) refers to the convergence implicit in the O terms.*

Remark 11. *A version of Theorem 5 also holds when a is a function of n, with the probability in (b) replaced by $1 - O(an\gamma + \frac{a\beta}{\lambda} e^{-n\lambda^3/\beta^3})$, under the condition that all functions f_l are uniformly bounded by some Lipschitz constant and f_l depends only on the variables $x, z_1, \ldots, z_l$. The latter condition is because as $n \to \infty$, one needs to solve a system of infinite number of differential equations which involves complicated technical issues. However, when f_l depends only on $x, z_1, \ldots, z_l$, one can solve the finite systems obtained for each f_l by restricting the equations to the ones that involve $x, z_1, \ldots, z_l$.*

B.2 Proof of Proposition 7

We need to prove that $\lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{X})}(\mathcal{E}) = \lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{X}_a)}(\mathcal{E})$. We have $\lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{D}_a)}(e_{uv} \mid \mathcal{E} \setminus e_{uv}) = \mathbb{P}(e_{uv}) = q$ by definition of $\mathcal{D}_a$. We prove that $\lim_{\frac{K}{N}\to 0} \mathbb{P}^{(\mathcal{D})}(e_{uv} \mid \mathcal{E} \setminus e_{uv}) = \mathbb{P}^{(\mathcal{D})}(e_{uv}) = q$ for graph $\mathcal{D}$, which implies that e_{uv} is independent of $\mathcal{E} \setminus e_{uv}$ in the $\frac{K}{N} \to 0$ regime. This concludes that the joint probability distribution

of the presence of the edges in the two graphs are also identical. In the sequel, all probabilities correspond to random graph $\mathcal{D}$. Hence, we write $\mathbb{P}$ in place of $\mathbb{P}^{(\mathcal{D})}$ to minimize notational clutter.

As per point (i) of Remark 4, the value of e_{uv} is fully determined by the state of the other edges that originate from u, if $f_v \in \mathcal{F}_{-v}$. Using the Baye's rule and the law of total probability for conditioning on $f_v \in \mathcal{F}_{-v}$, we have:

$$\mathbb{P}(e_{uv}|\mathcal{E} \setminus e_{uv}) = \frac{\mathbb{P}(e_{uv}, \mathcal{E} \setminus e_{uv})}{\mathbb{P}(\mathcal{E} \setminus e_{uv})} \tag{B.1}$$

where:

$$\begin{aligned}\mathbb{P}(e_{uv}, \mathcal{E} \setminus e_{uv}) = \mathbb{P}(e_{uv}, \mathcal{E} \setminus e_{uv}|f_v \in \mathcal{F}_{-v})\mathbb{P}(f_v \in \mathcal{F}_{-v}) \\ + \mathbb{P}(e_{uv}, \mathcal{E} \setminus e_{uv}|f_v \notin \mathcal{F}_{-v})\mathbb{P}(f_v \notin \mathcal{F}_{-v})\end{aligned}$$

and

$$\begin{aligned}\mathbb{P}(\mathcal{E} \setminus e_{uv}) = \mathbb{P}(\mathcal{E} \setminus e_{uv}|f_v \in \mathcal{F}_{-v})\mathbb{P}(f_v \in \mathcal{F}_{-v}) \\ + \mathbb{P}(\mathcal{E} \setminus e_{uv}|f_v \notin \mathcal{F}_{-v})\mathbb{P}(f_v \notin \mathcal{F}_{-v}).\end{aligned}$$

However, $|\mathcal{F}_{-v}| \leq K - 1$. As a result, $\mathbb{P}(f_v \in \mathcal{F}_{-v}) \leq \frac{K-1}{N}$ and $\lim_{\frac{K}{N}\to 0} \mathbb{P}(f_v \in \mathcal{F}_{-v}) = 1 - \lim_{\frac{K}{N}\to 0} \mathbb{P}(f_v \notin \mathcal{F}_{-v}) = 0$. Hence:

$$\begin{aligned}\lim_{\frac{K}{N}\to 0} \mathbb{P}(e_{uv}|\mathcal{E} \setminus e_{uv}) &= \frac{\lim_{\frac{K}{N}\to 0} \mathbb{P}(e_{uv}, \mathcal{E} \setminus e_{uv}|f_v \notin \mathcal{F}_{-v})}{\lim_{\frac{K}{N}\to 0} \mathbb{P}(\mathcal{E} \setminus e_{uv}|f_v \notin \mathcal{F}_{-v})} \\ &= \lim_{\frac{K}{N}\to 0} \mathbb{P}(e_{uv}|\mathcal{E} \setminus e_{uv}, f_v \notin \mathcal{F}_{-v}).\end{aligned} \tag{B.2}$$

By construction, we have $\mathbb{P}(e_{uv} \mid \mathcal{E}\setminus e_{uv}, f_v \notin \mathcal{F}_{-v}) = \mathbb{P}(f_v \in \mathcal{C}_u|\mathcal{E}\setminus e_{uv}, f_v \notin \mathcal{F}_{-v})$, where $\mathcal{C}_u$ is the set of files stored in the cache of vertex u. Since $f_v \notin \mathcal{F}_{-v}$, based on point (i) in Remark 4, $\mathcal{E} \setminus e_{uv}$ affects the probability of $f_v \in C_u$ by providing information about the amount of storage used for the files requested by the other caches. This effect is negligible when $\frac{K}{N} \to 0$. To prove this, we derive a lower and an upper bound on $\mathbb{P}(f_v \in \mathcal{C}_u \mid \mathcal{E} \setminus e_{uv}, f_v \in \mathcal{F}_{-v})$ by considering the two extreme cases where all the other edges and loops that originate from u are

present and absent, respectively. Assume that the number of distinct files requested by the vertices other than v is α. This implies $\alpha \leq K-1$. Then, the two extreme cases result in the inequalities $\frac{M-\alpha}{N-\alpha} \leq \mathbb{P}(f_v \in \mathcal{C}_u \mid \mathcal{E} \setminus e_{uv}, f_v \in \mathcal{F}_{-v}) \leq \frac{M}{N-\alpha}$. Maximizing and minimizing the upper and the lower bounds respectively over α, we bound the range of $\mathbb{P}(f_v \in \mathcal{C}_u \mid \mathcal{E} \setminus e_{uv}, f_v \in \mathcal{F}_{-v})$ over all possible demands as $\frac{M-(K-1)}{N-(K-1)} \leq \mathbb{P}(f_v \in \mathcal{C}_u \mid \mathcal{E} \setminus e_{uv}, f_v \in \mathcal{F}_{-v}) \leq \frac{M}{N-(K-1)}$. Taking the limits of the lower and the upper bounds as $\frac{K}{N} \to 0$[3] and using $M = qN$ for a fixed q, we get:

$$\lim_{\frac{K}{N}\to 0} \mathbb{P}(e_{uv} = 1 \mid \mathcal{E} \setminus e_{uv}, f_v \notin \mathcal{F}_{-v}) = \frac{M}{N} = q. \tag{B.3}$$

Eqs. (B.2) and (B.3) imply that in the $\frac{K}{N} \to 0$ regime, the presence of an edge or loop is independent of the presence of the other edges and loops in digraph $\mathcal{D}$ and each is present with probability q. This asymptotic model is denoted by $\mathcal{D}_a$. Given $\mathcal{D}_a$ as the asymptotic model of $\mathcal{D}$ for $\frac{K}{N} \to 0$, by construction, the undirected graph $\mathcal{G}$ can be asymptotically modeled by $\mathcal{G}_a$ for which every edge is present with probability q^2 and every loop is present with probability q.

B.3 Proof of Theorem 3

To prove Theorem 3, we use Wormald's theorem following an approach inspired by the approach in [56, Section 3.4.1]. Let $\mathcal{L}$ be the set of looped vertices of $\mathcal{G}_a$. Also, let $\mathcal{Q}$ and $\mathcal{U}$ represent the sets of unlooped vertices that are matched and remain unmatched by Algorithm 10, respectively. Since a looped vertex will not be matched by Algorithm 10, these three sets are disjoint and partition the vertices of $\mathcal{G}_a$.

We are interested in the number of looped vertices plus the matchings made by Algorithm 10:

$$\frac{|\mathcal{Q}|}{2} + |\mathcal{U}| = \frac{K - |\mathcal{U}| - |\mathcal{L}|}{2} + |\mathcal{U}| = \frac{1}{2}(K - |\mathcal{L}| + |\mathcal{U}|). \tag{B.4}$$

This is because one coded message is transmitted per each pair of matched vertices

[3] Since $K \geq 1$, $\frac{K}{N} \to 0$ implies $N \to \infty$.

and one uncoded message is transmitted for each unlooped and unmatched vertex.

Based on (B.4), to analyze the statistical behavior of the delivery rate one needs to analyze $|\mathcal{U}|$ and $|\mathcal{L}|$. We do this using Theorem 5. For that, we index the online matching process of Algorithm 10 by K, which is the number of vertices of $\mathcal{G}_a$. Let us define the two variables $L(t)$ and $U(t)$ on the process. Variable $L(t)$ is the number of looped vertices in $\{v_1, \cdots, v_{t-1}\}$. Variable $U(t)$ denotes the number of unlooped vertices in $\{v_1, \cdots, v_{t-1}\}$ that are not matched by Algorithm 10 in the first $t-1$ iterations. Notice that since $U(t) < K$ and $L(t) < K$, we set $c_0 = 1$ in Theorem 5.

In the following, we verify that the three conditions of Theorem 5 are satisfied for the defined variables. Both $L(t)$ and $U(t)$ satisfy the boundedness condition with $\beta = 1$ and $\gamma = 0$, as $|L(t+1) - L(t)| \leq 1$ and $|U(t+1) - U(t)| \leq 1$ always hold. For the trend hypothesis, we have

$$\begin{aligned}
\mathbb{E}(U(t+1) - U(t)|\mathcal{H}_t) = \\
&0 \times \mathbb{P}(v_t \text{ looped}) \\
&- 1 \times \mathbb{P}(v_t \text{ unlooped and matches at time} t) \\
&+ 1 \times \mathbb{P}(v_t \text{ unlooped and does not match at time} t) \\
&= -(1-q)(1-(1-q^2)^{U(t)}) + (1-q)(1-q^2)^{U(t)} \\
&= (1-q)\left[2(1-q^2)^{U(t)} - 1\right].
\end{aligned}$$

and

$$\mathbb{E}(L(t+1) - L(t)|\mathcal{H}_t) = \mathbb{P}(v_t \text{ looped }) = q,$$

where $\mathcal{H}_t$ is the history of the process up to time t. Since the derived expectations are deterministically true, the trend hypothesis holds with $\lambda_1 = 0$ and $f_1(x, z_1, z_2) = (1-q)[2(1-q^2)^{Kz_1} - 1]$ and $f_2(x, z_1, z_2) = q$, with domain D defined as $-\epsilon < x < 1+\epsilon$, $-\epsilon < z_1 < 1+\epsilon$ and $-\epsilon < z_2 < 1+\epsilon$, $\epsilon > 0$. Finally, the Lipschitz hypothesis is satisfied as f_1 and f_2 are Lipschitz continuous on $\mathbb{R}^2$, because they are differentiable everywhere and have bounded derivatives.

Since the conditions of the theorem are satisfied, the dynamics of z_1 and z_2 can

be formulated by the differential equations

$$\begin{aligned}\frac{dz_1(x)}{dx} &= (1-q)\Big[2(1-q^2)^{Kz_1(x)} - 1\Big],\ z_1(0) = 0;\\ \frac{dz_2(x)}{dx} &= q,\ z_2(0) = 0,\end{aligned}$$

where the initial conditions result from $U(0) = L(0) = 0$. Notice that the equations derived are decoupled in z_1 and z_2, and can be solved independently as

$$z_1(x) = -\frac{\log(2-(1-q^2)^{K(1-q)x})}{K\log(1-q^2)}, \qquad z_2(x) = qx.$$

Then, for $\lambda > \lambda_1 + c_0 K\gamma = 0$, with probability $1 - O(\frac{1}{\lambda}e^{-K\lambda^3})$, we have

$$\begin{aligned}U(t) = Kz_1(t/K) + O(\lambda K) =& -\frac{\log(2-(1-q^2)^{(1-q)t})}{\log(1-q^2)}\\ &+ O(\lambda K),\\ L(t) = Kz_2(t/K) + O(\lambda K) =& qt + O(\lambda K),\end{aligned}$$

uniformly for $0 \le t \le K$.

By evaluating the derived expressions for $U(t)$ and $L(t)$ at $t = K$, and their respective substitution in (B.4) for $|\mathcal{U}|$ and $|\mathcal{L}|$, Theorem 3 results.

B.4 Proof of Theorem 4

Let $Y_i(t)$ be the number of cliques of size i formed up to iteration $t-1$ when Algorithm 9 is applied to the side information graph modeled as $\mathcal{G}_a$, where $i = 1, \ldots, K$. To analyze the behavior of these variables, we use the version of Wormald's theorem that is provided in Remark 11. This is because the number of variables Y_i depends on K here.

It is straightforward to show that the boundedness hypothesis of Theorem 5 is satisfied with $\beta = 1$ and $\gamma = 0$, as in each iteration of the algorithm, the number of cliques of each size either remains unchanged, or decrements or increments by 1. Furthermore, for the trend hypothesis, we model the expected change of each

variable throughout the process as

$$\begin{aligned}
&\mathbb{E}(Y_i(t+1) - Y_i(t)|\boldsymbol{Y}(t)) \\
&\quad = 0 \times \mathbb{P}(v_t \text{ looped}) \\
&\qquad - \mathbb{P}(v_t \text{ unlooped and joins a clique of size } i \text{ at time} t) \\
&\qquad + \mathbb{P}(v_t \text{ unlooped and joins a clique of size } i-1 \text{ at time } t). \qquad \text{(B.5)}
\end{aligned}$$

Algorithm 9 operates solely based on the edges and loops of the side information graph. Hence, the output of the algorithm, i.e., the cliques formed by the algorithm up to time t, is a function of the edges and loops of the subgraph over vertices $v_1, \ldots, v_{t-1}$. Since in the asymptotic side information graph $\mathcal{G}_a$ every edge and loop is present independently, the output of the algorithm up to time t provides no information about the connectivity of vertex v_t to the previously arrived vertices. As a result, v_t is connected to any previously arrived vertex with probability q^2. Hence, with probability $(q^2)^j$, v_t is adjacent to all the vertices in a clique of size j. Similarly, $(1-q^{2j})^{Y_j(t)}$ is the probability that none of the $Y_j(t)$ cliques of size j are suitable for v_t at time t. Therefore, $\hat{g}_i(\boldsymbol{Y}) \triangleq \prod_{j=i}^{K}(1-q^{2j})^{Y_j(t)}$ is the probability that at iteration t, there exists no suitable clique of sizes equal or greater than i for v_t. Also, $\left(1-\left(1-q^{2(i-1)}\right)^{Y_{i-1}(t)}\right)\hat{g}_i(\boldsymbol{Y})$ is the probability that the largest suitable clique for v_t has size $i-1$, which implies that $Y_i(t+1)-Y_i(t)=1$. Finally, since v_t is not looped with probability $(1-q)$, we have

$$\begin{aligned}
&\mathbb{P}(v_t \text{ unlooped and joins a clique of size } i-1 \text{ at time } t) \\
&\qquad = (1-q)\left(1-\left(1-q^{2(i-1)}\right)^{Y_{i-1}(t)}\right)\hat{g}_i(\boldsymbol{Y}).
\end{aligned}$$

Using this result and by following the same line of argument for the case where Y_i

decrements at time t, (B.5) simplifies to

$$\mathbb{E}(Y_i(t+1) - Y_i(t)|\boldsymbol{Y}(t)) = (1-q)\times$$
$$\Bigg[-\left(1-\left(1-q^{2i}\right)^{Y_i(t)}\right)\prod_{j=i+1}^{K}\left(1-q^{2j}\right)^{Y_j(t)}$$
$$+\left(1-\left(1-q^{2(i-1)}\right)^{Y_{i-1}(t)}\right)\prod_{j=i}^{K}\left(1-q^{2j}\right)^{Y_j(t)}\Bigg], \tag{B.6}$$

where $\boldsymbol{Y}(t) = (Y_1(t), \ldots, Y_K(t))$.

Let $f_i(x, z_1, \ldots, z_K)$ be the right hand side of (B.6) with $Y_i(t)$ replaced by $Kz_i(x;q)$.[4] Here, we used notation $z_i(\cdot;q)$ to show that every z_i is parametrized by q. Then, the trend hypothesis of Wormald's theorem is satisfied for all variables Y_i and functions f_i with $\lambda_1 = 0$. Functions f_i are differentiable with bounded derivatives, hence they are Lipschitz continuous. Thus, based on Wormald's theorem, we get the system of differential equations in (5.1b) for z_i, where the initial conditions result from $Y_i(0) = 0$. Also, for any $\lambda > 0$, we have $Y_i(t) = Kz_i(t/K;q) + O(\lambda K)$ with probability $1 - O(\frac{K}{\lambda}e^{-K\lambda^3})$. Hence, (5.1) results.

B.5 Proof of Lemma 4

In this appendix, we derive the expressions in (5.5) and (5.6) that we use to evaluate the expected delivery rate of uncoded caching and the expected rate in [13, Theorem 2].

As in [13], let $N_{\mathrm{e}}(\boldsymbol{d})$ be the number of distinct requests in demand vector $\boldsymbol{d}$. Since $\boldsymbol{d}$ is a random vector, $N_{\mathrm{e}}(\boldsymbol{d})$ is a random number in $\{1, \ldots, K\}$. Given placement $\mathcal{P}$, if the rate of a delivery algorithm A_D only depends on $N_{\mathrm{e}}(\boldsymbol{d})$ and

[4] Notice that f_{n-l} only depends on $z_n, z_{n-1}, z_{n-(l+1)}$. Hence, by an argument similar to the one in Remark 11, one can solve for f_{n-l} by restricting the equations to the ones that involve these variables.

not the individual files requested in $\boldsymbol{d}$, then, the expected delivery rate will be

$$\mathbb{E}_{\boldsymbol{d}}(R_{A_D}) = \sum_{m=1}^{K} \mathbb{P}(N_{\mathrm{e}}(\boldsymbol{d}) = m) R(\boldsymbol{d}; \mathcal{P}, A_D \mid N_{\mathrm{e}}(\boldsymbol{d}) = m). \tag{B.7}$$

Assuming that the popularity distribution of files is uniform, we have

$$\mathbb{P}(N_{\mathrm{e}}(\boldsymbol{d}) = m) = \frac{\binom{N}{m}\left\{{K \atop m}\right\} m!}{N^K}. \tag{B.8}$$

This is because in $\binom{N}{m}$ ways one can select m out of N files. There are $\left\{{K \atop m}\right\}$ ways to partition K objects into m non-empty subsets, where $\left\{{K \atop m}\right\}$ is the Stirling number of the second kind [61, Section 5.3]. Also, there are $m!$ ways to assign one of the m selected files to each subset. Hence, there are $\binom{N}{m}\left\{{K \atop m}\right\} m!$ demand vectors of length K with m distinct files from a library of N files. Since the total number of demand vectors is N^K and they are equiprobable, (B.8) results.

Uncoded Caching Consider the cases of uncoded caching where delivery messages are uncoded. For placement every cache either stores the same set of M files out of the N files in the library, or every cache stores the first M/NF packets of every file. These correspond to uncoded file and subfile cachings. The rate required to delivery demand vector $\boldsymbol{d}$ with uncoded messages is $N_{\mathrm{e}}(\boldsymbol{d})(1 - M/N)$ files. Hence, based on (B.7) and (B.8), we get

$$\begin{aligned}\mathbb{E}_{\boldsymbol{d}}(R_{\text{uncoded}}) &= \sum_{m=1}^{K} \frac{\binom{N}{m}\left\{{K \atop m}\right\} m!}{N^K} \times m \left(1 - \frac{M}{N}\right) \\ &= N \left(1 - (1 - \frac{1}{N})^K\right) \left(1 - \frac{M}{N}\right), \end{aligned} \tag{B.9}$$

where we used

$$\sum_{m=1}^{K} m \binom{N}{m} \left\{{K \atop m}\right\} m! = N^{K+1} - N(N-1)^K. \tag{B.10}$$

The reason (B.10) holds is as follows. Consider a set of $K + 1$ objects that we want to *partition* into m subsets such that the subset containing the first object

has cardinality greater than 1, i.e., the first object is not the only object in the corresponding subset. This can be done in $m\left\{{K \atop m}\right\}$ ways as we can partition the rest of the $(K+1)-1$ objects into m subsets and then add the first object to one of the resulting m subsets. Assume that the subsets can be labeled distinctly from a set of $N \geq K$ labels. Then, there are $\sum_{m=1}^{K} m\left\{{K \atop m}\right\}\binom{N}{m}m!$ ways to partition $K+1$ objects and label them with distinct labels such that the subset containing the first object has cardinality greater than 1. This sum is equal to the total number of ways to partition $K+1$ objects and distinctly label them with the N available labels, minus the number of ways to do the same but have the first object as the only element in its corresponding subset. The former counts to N^{K+1}, and the latter can be done in $N(N-1)^K$ different ways. This proves (B.10).

Optimal CSC From (B.7), the expected rate in the RHS of [13, eq. (27)] can be written as

$$\mathbb{E}_{\boldsymbol{d}}(R^*_{\text{CSC}}) = \sum_{m=1}^{K} \mathbb{P}(N_{\text{e}}(\boldsymbol{d}) = m) \times \left[\frac{N-M}{M}(1-(1-M/N)^m)\right], \tag{B.11}$$

which by substitution of (B.8), results in (5.5).