AN OPTIMIZATION APPROACH TO HETEROGENEOUS CODED CACHING

by

Alexander Michael Daniel

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

© Copyright 2017 by Alexander Michael Daniel
This thesis aims to develop an optimization framework for the development of coded caching schemes that can account for various heterogeneous aspects of practical communication systems. This framework is first used to develop an optimization-based perspective on the seminal work on the fundamental limits of caching by Maddah-Ali and Niesen, and is then extended to the case where files have non-uniform popularity and length, while users have non-uniform cache sizes. The resulting optimization problem scales exponentially in the system parameters, and so simplifications of the original problem that both perform well and scale as a polynomial function of the systems parameters are developed. By considering the aforementioned heterogeneities both individually and in conjunction with one another, insights into the influence of their interactions on optimal cache content are obtained.
Dedication

To my parents, for their “transcendental” support - it clearly cannot be expressed in a finite number of elementary operations - and for loving me despite the earlier part of this sentence.

Acknowledgements

This thesis would be incomplete without acknowledging the support of those without whom this work would not have been possible:

I must first thank my advisor, Professor Wei Yu, who has been everything I could hope for from an advisor, especially for knowing both when to give me the answer and when to have me work it out for myself. I have an accurate accounting of my debt to him, but this margin is not large enough to contain it.

I must also thank the faculty, staff, and students at the University of Toronto: to my new-old friend Ahmad Ali Khan, for long and fruitful discussions of technical and non-technical topics alike - they were both causes of and solutions to unproductive days of research; to Binbin Dai, my caching-partner-in-crime, equally for his helpful advice and for setting a standard of research quality that I felt compelled to meet; to the rest of my fellow research group members and classmates, for their companionship and commiserations throughout this process; to Professors Ravi Adve, Frank Kschischang, Stark Draper, and Alberto Leon-Garcia, for teaching me so much, and showing me how little I know; and finally, to my defense committee, Professors Josh Taylor, Stark Draper, and Ashish Khisti.

Finally, I must thank my undergraduate mentor, Professor Serguei Primak of the University of Western Ontario, for (patiently) providing me my first exposure to research, and for encouraging me to continue learning beyond the classroom. Without the experience he provided, I would have found myself with neither the desire nor the ability to complete this degree.
# Contents

Dedication iii  
Acknowledgements iii  

1 Introduction 1  
  1.1 Background and Literature Review ........................................ 1  
  1.2 Contributions ...................................................................... 3  
  1.3 Notation ............................................................................. 4  
  1.4 Thesis Format ..................................................................... 4  

2 Review 5  
  2.1 The Centralized Case ............................................................. 5  
  2.2 The Decentralized Case .......................................................... 6  
  2.3 Examples ............................................................................ 7  

3 An Optimization Approach to Coded Caching 9  

4 Homogeneous Coded Caching 12  
  4.1 Problem Analysis ................................................................. 13  

5 Heterogeneous Coded Caching 17  
  5.1 Non-Uniform File Popularity and Length .............................. 18  
    5.1.1 Background ................................................................. 18  
    5.1.2 Problem Formulations .................................................... 19  
       Popularity-First Approach .................................................... 20  
       Length-First Approach ........................................................ 21  
    5.1.3 Numerical Results .......................................................... 22  
  5.2 Non-Uniform Cache Size ...................................................... 25  
    5.2.1 Background ................................................................. 25  
    5.2.2 Problem Formulation ..................................................... 25  
    5.2.3 Numerical Results .......................................................... 28  
  5.3 Non-Uniform File Size, Popularity and Cache Size .................. 30  
    5.3.1 Background ................................................................. 30  
    5.3.2 Problem Formulation ..................................................... 30  
    5.3.3 Numerical Results .......................................................... 32
Chapter 1

Introduction

1.1 Background and Literature Review

In a technical context, the term cache refers roughly to a memory bank dedicated exclusively to the storage of frequently-accessed data. The concept of a cache dates back at least as far back as the 1960s, where caches were used to facilitate memory access in computers; in his 1968 description of the new IBM “System/360 Model 85” computer [1], J.S. Lipton characterizes the role of the cache in the novel memory architecture of the computer as follows:

“The storage hierarchy consists of a 1.04-microsecond main storage and small, fast store called a cache, which... is used to hold the contents of those portions of main storage that are currently being used. Most processor fetches can then be handled by referring to the cache, so that most of the time the processor has a short access time.”

Communications engineers have also historically found use for such devices. For instance, as computer networks expanded both in size and in the amount of data transmitted, caching proved to be useful in reducing latency while simultaneously reducing total network traffic (see e.g. the 1997 paper [2] for an overview and discussion on the role of caching for the Internet).

It is in this spirit that caching techniques have more recently been identified as a “disruptive” technology in future wireless networks [3]. And though, as discussed above, caches have long been used in wired networks, little of that research is applicable to the wireless case. In the wired case, the network topology is fixed over long periods of time, with communication typically occurring over point-to-point links with high reliability; in the wireless case however, network topology varies rapidly, with communication occurring over a shared medium whose reliability cannot be guaranteed a priori. Indeed, users in such a network may have access to more than one cache at a time, with the list of accessible caches changing over time, while caches may have multiple users to serve over the same medium simultaneously. Caching in the wireless domain therefore requires consideration in its own right.

Broadly speaking, there have been two different approaches to the study of caching in wireless networks. The first approach follows the work of Golrezaci, Shanmugan, et al. in their study of “femtocaching” [4,5], where the problem of optimizing cache content to minimize download time is considered for a certain heterogeneous wireless network where a single base station and many “femto” base station (referred to as “femtocaches” or “helpers”) are equipped with caches and serve users who are not
Chapter 1. Introduction

equipped with caches. A user associates with every helper within a certain radius, and should their desired file not be contained among the associated helpers, it can download as necessary from the macro base station, which is assumed to have a cache capable of containing all relevant files. The work considers both the case where files are stored uncoded and in their entirety, and the case where the files are fountain coded (see e.g. [6] for an overview) to enable a user to collect parts of files from different helpers.

This approach inspired significant further study from researchers seeking to extend femtocaching-style approaches to more practical network models, or optimize cache content and other network parameters with respect to a variety of metrics of interest. For instance, [7] maximizes a combination of user rates and backhaul reduction by optimizing cache content and user association strategies, while [8] maximizes cache-hit probabilities using a stochastic geometry-based analysis. In general, these kinds of studies consider a realistic model of a wireless network, and examine the effect that cache content has on the general performance of the network. This approach is not considered in this thesis, and so an exhaustive list of these works is beyond our scope; the interested reader is referred to the partial list of papers [4, 5, 7–18] and references therein.

This femtocaching-inspired approach is to be contrasted with the second approach, which involves the study of coded caching techniques. First developed by Maddah-Ali and Niesen in [19, 20], this method abstracts away many features of a realistic wireless network and examines the idea that redundant cache content across users can enable coded multicast transmissions to improve communication efficiency. Specifically, they consider a single server containing $N$ files of equal popularity and length satisfying the simultaneous requests of $K$ cache-enabled users through transmission over an error-free broadcast medium; a more detailed description is given in Chapter 2. A great deal of subsequent research has focused on designing coded caching schemes for more realistic versions of the model, including schemes for decentralized systems [21], non-uniform file popularity [22–34], non-uniform file size [35, 36], multiple user requests [37–40], non-uniform cache size [41–44], and non-uniform channel quality [45–57]. More detailed discussions of these works as they pertain to this thesis will be provided later.

The aforementioned works typically discuss either how the scheme of [19, 20] should be modified to accommodate the considered heterogeneity, or develop an entirely new scheme that enables coded multicast transmissions while accommodating the aforementioned heterogeneity. However, most of these papers consider only one type of non-uniform system parameter. While this is sensible from the viewpoint of understanding how each heterogeneity affects coded caching systems by itself, practical systems would have to account multiple types of non-uniform parameters. Moreover, we cannot in general expect the effects of these non-uniformities to be additive. It is thus important to consider combinations of heterogeneities, a concept that, to the best of our knowledge, only a few recent works have started to explore: the recent work [36] examines the achievable rate region for a system serving two users with two files, where the user cache sizes and file sizes are not necessarily uniform, while some of the aforementioned work on caching with non-uniform channel quality exploits heterogeneous cache size to rectify disparities in channel quality (see e.g. [50] and references therein). As elaborated on in the next section, this thesis represents an effort to take a considerable step forward in the study of coded caching in the presence of multiple heterogeneous system parameters.
1.2 Contributions

This thesis proposes an optimization theoretic framework to design caching schemes capable of accommodating non-uniform file length, non-uniform file popularity, and non-uniform user cache size at the same time. More specifically, we design a caching scheme that uses a generalized version of the transmission scheme of [21], paired with an optimization problem designed to yield the optimally coded content. This optimization problem, although convex, has a number of variables, constraints, and objective function terms that scale exponentially with the problem size, so subsequently this thesis develops high-quality simplifications that scale polynomially with the problem size, yet perform well compared to the original problem. The proposed optimization approach only yields numerical answers corresponding to the optimized caching schemes, but also generate practical insight into the problems considered.

Optimization approaches have been used in the past for content placement for femtocaching systems [4, 5], but its use in the coded caching context has only appeared recently: for instance, [58] uses an information-theory based optimization problem to help characterize the achievable rate region for certain numbers of users and files in the case where all other systems parameters are uniform. More related is the recent work [44] in which an optimization framework similar to the one used here is employed to develop a caching scheme in the case of non-uniform cache size. Crucially, both approaches design cache content in terms of the subsets of users who have cached the content. While we directly express our transmission scheme in terms of that cache content, the approach in [44] is to further design two sets of variables through which the transmission scheme is expressed. Moreover, the framework used in [44] does not distinguish between different files beyond designing a scheme for the worst-case scenario where users request different files. This prevents their framework from addressing file heterogeneity, whereas ours allows for it. Finally, while the optimization problem in [44] suffers from the same exponential scaling problem that the general problem here has, they do not present any tractable methods of solution like we do here.

During the preparation of this thesis, we became aware of independent work [34], which uses an optimization framework essentially the same as the one proposed here to study the case of non-uniform file popularity. While independently and simultaneously developed, a number of results in [34] are echoed in this thesis. Specifically, both works develop the same exponential-order general optimization problem; then a simplified polynomial-order optimization problem is developed for the non-uniform popularity case, although the exact formulations are different. The two different approaches will be distinguished in greater detail later in the thesis. Moreover, both works show that in the special case of uniform popularity, the caching and delivery scheme of [19, 20] is the optimal solution; however, the proof in [34] is quite involved, while a much simpler proof is presented here. Ultimately, the focus of [34] is to study the case of non-uniform popularity in great depth, while here, it is only considered as an intermediate step towards the study of the interactions of several heterogeneities at the same time. Thus, despite the similarities, both works develop many unique insights of practical significance. Indeed, the results of [44] and [34] taken jointly with the results of this thesis suggest that the optimization framework common to all three works is likely to be a useful one for coded caching.
1.3 Notation

The notation \([a : b]\) is used as shorthand for the set of consecutive integers \(\{a, a + 1, \ldots, b - 1, b\}\), and \([b]\) is used as an abbreviation of \([1 : b]\). The symbol \(\oplus\) is used to denote the bitwise "XOR" operation between two or more files (i.e. strings of bits). Both \(l\) and \(W(l)\) are used to refer to the \(l\)-th file under consideration. For an arbitrary file \(W^{(n)}\), \(|W^{(n)}|\) refers to the length of the file, and \(W^{(l)}_S\), called a "subfile" of file \(W^{(l)}\), refers to the portion of file \(l\) stored exclusively on the caches of the users in the set \(S\). For notational convenience, notation of the form \(W^{(l)}_{123}\) is used instead of \(W^{(i)}_{\{1,2,3\}}\) (for the \(S = \{1, 2, 3\}\) case in this example), returning only to the latter notation if necessary to resolve ambiguity. For a set \(S\), \(\mathcal{P}(S)\) refers to the power set of \(S\). For a real number \(t\), \(\lfloor \cdot \rfloor\) and \(\lceil \cdot \rceil\) denote the floor and ceiling functions, respectively.

We define the binomial coefficient \(^n\!\!\!C_k\) in the usual way for \(0 \leq k \leq n\), i.e. \(^n\!\!\!C_k = n!/(k!(n-k)!))\), but for \(n < 0\) or \(k > n\), we take \(^n\!\!\!C_k = 0\). Moreover, the notation of the so-called “multinomial coefficient” is used, defined as:

\[
\binom{n}{k_1, k_2, \ldots, k_m} = \frac{n!}{k_1!k_2!\ldots k_m!},
\]

where \(k_1 + k_2 + \cdots + k_m = n\). We use the notation \(\sum_{i=a}^{b} n_i\) in the usual way when \(a \leq b\), and take \(\sum_{i=a}^{b} n_i = 0\) identically if \(a > b\). More generally, a sum over an empty set of indices is taken to equal zero. Finally, if a sum over an empty set is raised to the power of zero, we take the resulting value to be 1; this is simply used for notational convenience.

1.4 Thesis Format

The organization of the rest of this thesis is as follows. The original coded caching schemes of Maddah-Ali and Niesen are presented in Chapter 2, for both the centralized [19,20] and decentralized [21] cases, in order to familiarize the reader with the concept of coded caching. The optimization framework used in this thesis is then presented in Chapter 3, and this framework is then used to analyze coded caching in the homogeneous case in Chapter 4. The various heterogeneous coded caching scenarios are considered in Chapter 5, and the thesis is concluded in Chapter 6.
Chapter 2

Review

Here we review the coded caching schemes presented by Maddah-Ali and Niesen in [19, 20] for the centralized case and [21] for the decentralized case. Consider a server containing a set of \( N \) equally popular files, \( \mathcal{F} = \{1, 2, \ldots, N\} = [N] \), of length \( F \) bits each, serving a set of \( K \) users, \( \mathcal{U} = \{1, 2, \ldots, K\} = [K] \), each of whom has a cache of size \( MF \) bits. The connection of the server to the users is over a shared, error-free link. In periods of low network usage, the caching phase occurs, where the users’ caches are filled with content by the server according to some scheme; since there is little network activity during this phase, the resources consumed here are of no concern. Then, during the next period of high network activity, the transmission phase occurs: each users makes a (simultaneous) request for one of the \( N \) files, which the server then delivers over the shared link according to some transmission scheme. The goal of the problem is to design both the caching and transmission schemes such that the number of bits, \( R \), needed to satisfy an arbitrary set of user requests during the transmission phase is minimized.

2.1 The Centralized Case

This setup is first considered by Maddah-Ali and Niesen in [19,20] for the “centralized” case, where both the total number and identity of users is known during the caching phase. Moreover, the same set of users is expected to return during the transmission phase. This latter condition makes the centralized model less appropriate for the case of a base station (BS) transmitting directly to cache-enabled mobile users because they will not reliable return for the transmission phase. Instead, the centralized model would be more useful for, say, a heterogeneous network setup where a macro BS delivers files to cache-enabled femto/pico-cell base stations serving large numbers of users who can reliably be assumed to have a request during the transmission phase. To the best of our knowledge, the effect of absent users in the transmission phase on the performance of the centralized system has not been addressed in the literature.

The centralized caching scheme itself is, for the case where \( t = KM/N \) is an integer,\(^1\) to partition each file into \( \binom{K}{t} \) equal-sized pieces; each such subfile is then labelled by one of the \( \binom{K}{t} \) subsets of \( \mathcal{U} \) of size \( t \). A user then stores a subfile on their cache if they are a member of the subset with which the

\(^1\)When \( t \) is not an integer, i.e. \( t = s\lfloor t \rfloor + (1 - s)\lceil t \rceil \) for \( s \in [0,1] \), \( sMF \) bits of cache are dedicated to do the following scheme with parameter \( \lfloor t \rfloor \), and the remaining \( (1 - s)MF \) bits are used for the scheme with parameter \( \lceil t \rceil \). Maddah-Ali and Niesen call this strategy memory sharing.
subfile is labelled. Since user \( k \) is in \( \binom{K-1}{t-1} \) subsets of size \( t \) and there are \( N \) files of length \( F \) bits, the total amount of cache memory that this requires is

\[
N \cdot \binom{K-1}{t-1} \cdot \frac{F}{(K_t)} = NF \cdot \frac{t}{K} = \frac{NFKM}{K} = MF \text{ bits,}
\]

(2.1)

and so the cache of each user is completely filled.

During the transmission phase, each user reveals their request from the set of \( N \) files; the file index requested by user \( k \) is denoted as \( d_k \). An arbitrary set of requests can be satisfied using \( \binom{K}{t+1} \) transmissions, one for each subset of \( t+1 \) users. To the subset \( S, |S| = t+1 \), the server computes and sends:

\[
\bigoplus_{k \in S} W_{S \setminus \{k\}}^{(d_k)}.
\]

(2.2)

In other words, for each user \( k \), there is a unique subfile of their desired file, \( W_{(d_k)} \), contained in the cache of every other user in the set \( S \), namely the subfile \( W_{S \setminus \{k\}}^{(d_k)} \). All \( t+1 \) of these subfiles are added together and transmitted over the shared link. Each user in \( S \) is then able to decode the transmission by using the \( t \) subfiles in their cache to obtain the subfile of the file they requested. This is repeated for every \( (t+1) \)-sized subset \( S \); user \( k \) participates in \( \binom{K-1}{t-1} \) of these, and receives a unique subfile from each one. Between the \( \binom{K-1}{t-1} \) files obtained from the transmissions and the \( \binom{K-1}{t-1} \) files already stored in their cache, then, using Pascal’s identity for binomial coefficients, we see that each user has \( \binom{K-1}{t-1} + \binom{K-1}{t} = \binom{K}{t} \) different subfiles, i.e. they can reconstruct their requested file. Since there are \( \binom{K}{t+1} \) broadcasts of size \( F/\binom{K}{t} \) bits each, the total number of bits sent is

\[
R = \binom{K}{t+1} \frac{F}{\binom{K}{t}} = \frac{K-t+1}{K} F = \frac{K(1-M/N)}{KN+1} F,
\]

(2.3)

which reduces the number of bits sent compared to the naive baseline by a factor of \( 1/(KM/N+1) \).

### 2.2 The Decentralized Case

Maddah-Ali and Niesen then extended their results to the “decentralized” case in [21]. Here, the server knows neither how many nor which users will make requests at the beginning of the transmission phase, and so the caching scheme cannot be designed as a function of the number or identity of the users. The proposed caching scheme is then for each user simply to cache \( M/N \) bits per file, uniformly at random. At the beginning of the transmission phase, the users that do make requests inform the server of both their request and their cache content, and the server then must satisfy all of the requests by using as few bits as possible.

When informed of the cache content of each user, the server can label an arbitrary file \( W \) as

\[
W = \bigcup_{S \in \mathcal{P}(\mathcal{U})} W_S,
\]

(2.4)

where \( \mathcal{P}(\mathcal{U}) \) represents the power set of \( \mathcal{U} \). Here, \( W_S \) represents the set of bits of \( W \) that are cached in
the caches of the users in $S$, but nowhere else, i.e. $W_{S_1} \cap W_{S_2} = \emptyset$ for all $S_1, S_2 \in \mathcal{P}(U)$. The subfile $W_0$ is the portion of $W$ not contained on the cache of any user. It is shown that under the random caching scheme, the size of a subfile $W_S$ is, with high probability as $F$ grows larger,

$$|W_S| = (M/N)^{|S|}(1 - M/N)^{K - |S|}F,$$

(2.5)

where $K$ now denotes the number of users who have made requests at the beginning of the transmission phase. The resulting transmission phase is then similar to the centralized version, but with a minor modification: the server still sends (2.2), but instead of doing it once for every user subset of size $t + 1$, it is done once for every $S \in \mathcal{P}(U) \setminus \emptyset$. Although the decentralized scheme can never perform better than the centralized scheme, the decentralized scheme sends at most 1.5 times the number of bits of the centralized scheme, and the gap between the two closes in the limit at $K \to \infty$ [59].

2.3 Examples

Consider the case where there are $K = 3$ users, $N = 3$ files, and $M = 1$ is the normalized cache size. Denote the files requested by users 1, 2, and 3 as $d_1, d_2$ and $d_3$ respectively.

Example 1. In the centralized case, we have $t = KM/N = 1$, and so the server partitions an arbitrary file $W$ as

$$W = W_1 \cup W_2 \cup W_3.$$

User 1 caches the first subfile, user 2 caches the second, and user 3 caches the third; this is then repeated for the other two files. In the transmission phase, the server sends the following:

$$\begin{align*}
W_2^{(d_1)} & \oplus W_1^{(d_2)} \\
W_3^{(d_1)} & \oplus W_1^{(d_3)} \\
W_3^{(d_2)} & \oplus W_2^{(d_3)}
\end{align*}$$

(2.6)

It is easy to check directly that each user is able to recover their desired subfile from their respective transmission, and that each user is then able to reconstruct their desired file.

Example 2. In the decentralized case, some number of users cache $MF/N$ bits of each file uniformly at random during the caching phase. At the beginning of the transmission phase, 3 users make requests, and reveal their cache contents to the server. The serve is then able to label each of the requested files as

$$\{W_\emptyset, W_1, W_2, W_3, W_{12}, W_{13}, W_{23}, W_{123}\},$$

where user 1 has $\{W_1, W_{12}, W_{13}, W_{123}\}$ cached, user 2 has $\{W_2, W_{12}, W_{23}, W_{123}\}$ cached, and user 3 has $\{W_3, W_{13}, W_{23}, W_{123}\}$ cached. For the transmission phase, the server sends:

$$\begin{align*}
W_3^{(d_3)} & \oplus W_{13}^{(d_2)} \oplus W_{12}^{(d_1)} \\
W_2^{(d_1)} & \oplus W_1^{(d_2)} \\
W_3^{(d_1)} & \oplus W_1^{(d_3)}
\end{align*}$$
Again, it is straightforward to verify that each user can decode their respective transmissions and can recover their requested file.
Chapter 3

An Optimization Approach to Coded Caching

We now consider a generalization of the scenario considered in the previous section. The transmission scenario consists of a server with a set of \( N \) files, \( F = \{1, 2, \ldots, N\} = [N] \), serving a set of \( K \) users, \( U = \{1, 2, \ldots, K\} = [K] \), over a shared, error-free link. For full generality, we allow each file \( l \) to have a distinct length of \( F_l \) bits and a distinct probability \( p_l \) of being requested, and allow each user \( k \) to have arbitrary cache size of \( M_k \) bits.

Central to the coded caching scheme of [19–21] is the partitioning of each file \( l \in F \) into subfiles \( W_{S}^{(l)} \), indexed by all subsets of \( S \subseteq U \). (No two subfiles have a non-empty intersection; together the subfiles jointly reconstruct the original file.) In the caching phase, each user \( k \) caches

\[
\bigcup_{l, S \in \mathcal{P}(U \setminus \{k\})} W_{S \cup k}^{(l)}
\]

without coding. In the content delivery phase, given the set of user requests \( \mathbf{d} = [d_1, \ldots, d_K] \), where \( d_k \) denotes the index of the file requested by user \( k \), the server transmits

\[
\bigoplus_{k \in S} W_{S \setminus \{k\}}^{(d_k)}
\]

over the shared link for each \( S \in \mathcal{P}(U) \setminus \emptyset \), so that together with uncoded content stored in each user’s local cache, all users are guaranteed to be able to reconstruct the entirety of their requested files. Note that if the constituent subfiles of (3.2) are not of the same length, i.e., the sizes of the \( W_{S \setminus \{k\}}^{(d_k)} \) are different, the shorter subfiles can be zero-padded to be the same length as the longest subfile in the transmission.

Any coded caching scheme (with coded transmissions and uncoded cache content\(^1\)) can be described using this “subset partitioning” representation [21]. Different caching strategies differ in their partitioning of the subfiles. But instead of thinking of the above as a way of labelling the cache contents, we can also use this to design the cache content, and regard the size of each subfile \( W_{S}^{(l)} \) as design variables. Consequently, instead of designing cache contents by assigning discrete bits to sets, the problem can be sim-

\(^1\)To date, there is little work comparing the benefits of coded versus uncoded cache content. Perhaps the most thorough study of coded cache content is found in [58].
plified by designing only the sizes of the subfiles: if we have $|W^{(l)}_0| = b^{(l)}_0$, $|W^{(l)}_1| = b^{(l)}_1$, ..., $|W^{(l)}_U| = b^{(l)}_U$, then the first $b^{(l)}_0 F_l$ bits of $W^{(l)}$ are assigned to $W^{(l)}_0$, the next $b^{(l)}_1 F_l$ bits to $W^{(l)}_1$, and so on, assigning the final $b^{(l)}_U F_l$ bits to $W^{(l)}_U$. (Formally, this requires that $b^{(l)}_S \in \{0, 1/F_l, \ldots, (F_l - 1)/F_l, 1\}$, but for large $F_l$, this can be relaxed to $b^{(l)}_S \in [0, 1]$ without any significant loss.)

For example, consider the centralized scenario wherein the numbers and identities of users are known in advance, so the server has the ability to design the cache content of each user. The coded caching scheme of [19, 20] (assuming uniform file length, cache size and uniform popularity) sets $|W^{(l)}_S|$ to be non-zero for only the $S$’s with $|S| = t = KM/N$. In the decentralized setting of [21], all subfiles have non-zero sizes due to the use of random cache content. More generally, $|W^{(l)}_S|$ can be explicitly designed.

The design of coded caching in this way imposes some natural constraints on the subfile sizes. First, the subfiles together must contain the entire file, i.e.

$$\sum_{S \in \mathcal{P}(U)} |W^{(l)}_S| = F_l, \quad \forall l \in \mathcal{F}. \quad (3.3)$$

Second, denoting the amount of cache dedicated to file $l$ by user $k$ as $\mu_{k,l}$, the amount of a file cached by a user is expressed as

$$\sum_{S \in \mathcal{P}(U \setminus \{k\})} |W^{(l)}_{S \cup \{k\}}| \leq \mu_{k,l}, \quad \forall k \in U, \forall l \in \mathcal{F}, \quad (3.4)$$

where

$$\sum_{l=1}^{N} \mu_{k,l} = M_k, \forall k \in [K]. \quad (3.5)$$

Finally, the subfiles cannot have a negative size:

$$|W^{(l)}_S| \geq 0, \quad \forall S \in \mathcal{P}(U), \quad \forall l \in \mathcal{F}. \quad (3.6)$$

In the transmission defined in (3.2), zero-padding is needed whenever the subfiles do not have the same length, so the length of a single transmission is determined by the largest subfile in the transmission. For a vector of user requests $d$, the number of bits sent to satisfy user requests given in $d$ is thus

$$R_d = \sum_{S \in \mathcal{P}(U \setminus \emptyset)} \max_{k \in S \setminus \{k\}} \{|W^{(d_k)}_{S \setminus \{k\}}|\} \quad (3.7)$$

The set of choices of $|W^{(l)}_S|$ define a broad family of caching schemes. To find the most efficient caching strategy among this family of schemes that minimize the expected delivery rate over all demand requests, we can formulate the following optimization problem:

$$\text{minimize} \quad \mathbb{E}[R_d] = \sum_{d \in \mathcal{D}^K} p(d) \sum_{S \in \mathcal{P}(U \setminus \emptyset)} \max_{k \in S \setminus \{k\}} \{|W^{(d_k)}_{S \setminus \{k\}}|\}$$

subject to \hspace{1cm} (3.3)-(3.6). \quad (3.8)$$

We note before continuing that the transmission scheme described by this optimization problem is the same for all possible user requests. This may result in suboptimal performance if there is repetition in the users’ requests, i.e. if the same file is requested by more than one user. For instance, if all users were to request the same file, it may be more efficient to simply transmit the entire file once than to go
through the process of transmitting the coded multicasts described by (3.2). Thus some loss is incurred by always doing the latter; however, as the following shows, it is not likely to be a significant loss. Consider the fraction of possible user requests in which all files requested are distinct. There are \( N \) ways of choosing the first file, \( N - 1 \) ways of choosing the second file, and so on, with \( N^K \) total possibilities. This gives

\[
\frac{N(N - 1) \ldots (N - K + 1)}{N^K} = \frac{N^K - c_{K-1}(K)N^{K-1} - \cdots - c_1(K)K - c_0(K)}{N^K} = 1 - \frac{c_{K-1}(K)}{N} - \cdots - \frac{c_1(K)}{N^{K-1}} - \frac{c_0(K)}{N^K},
\]

as \( N \) grows large (for a fixed \( K \)), where the \( c_i(K) \) are coefficients that are polynomial functions of \( K \). Thus as the number of files grows large, all the subtracting terms go to zero, and the fraction of file requests that contain \( K \) different files goes to one. Since we are likely to have \( N >> K \) in practice, Eq. (3.8) can be used without fear of significant loss.

Finally, note that the problem (3.8)-(3.9) is convex in the \(|W_S|\) variables, since the constraints are linear and the maximum function is a convex function of its argument. However, there are \( N^{2K} \) variables, \( N^K(2^K - 1) \) summands in the objective function, and \( N^{2K} + KN + N + K \) constraints, making this an impractical problem to solve directly. The rest of this thesis is thus dedicated to developing simplifications of (3.8)-(3.9) that allow for high quality (and even optimal) solutions while maintaining a tractable problem size.
Chapter 4

Homogeneous Coded Caching

Consider the special case of problem (3.8)-(3.9) with uniform file lengths, $F_l = F, \forall l \in [N]$, uniform file popularities, $p_l = 1/N, \forall l \in [N]$, and uniform cache sizes $M_k = M, \forall k \in [K]$; this is the same system originally considered in [19, 20]. The symmetry of the resulting problem can be exploited to reduce the computational complexity of optimization. Specifically, define $v_j$ such that $v_j = |W_S|$ for all $S$ such that $|S| = j$ and for all files $W$; this reduces the number of optimization variables from an exponential number in $K$ to a linear number in $K$. Since the file length, file popularity and cache size are all homogeneous, there is symmetry across both the users and the files, and so we would expect the solution to (3.8)-(3.9) for this uniform case to have this form, i.e., any two subfiles have the same size if their respective user sets are the same size. The summands of the objective function now simplify as

$$\max_{k \in S} \left\{|W_{S \setminus \{k\}}^{(k)}|\right\} = |W_{S \setminus \{k'\}}^{(k')}| = v_{|S|-1}$$

where $k'$ is any user in the set $S \setminus \{k\}$, because each subfile in a given transmission is the same size. Since the files are equally popular, every request vector is equally likely, and since every transmission scheme requires the same number of bits to satisfy the requests, the average does not need to be taken across all the $N^K$ possible demands as in (3.8). Finally, since $v_j F$ bits are sent to any subset of $j + 1$ users, and there are ${K \choose j}$ subsets of $j + 1$ users, the objective function becomes

$$\mathbb{E}[R_d] = \sum_{j=0}^{K-1} {K \choose j+1} v_j F,$$  \hspace{1cm} (4.1)

Note that the number of terms in the objective function now scales linearly in $K$ instead of exponentially in $K$, because each term in (4.1) accounts for a combinatorial number of transmissions.

The constraints simplify as well. Since all files are of equal length, only one file reconstruction constraint is required. Then, because there are ${K \choose j}$ subsets of size $j$, the file reconstruction constraint becomes

$$\sum_{j=0}^{K} {K \choose j} v_j = F.$$  \hspace{1cm} (4.2)

Moreover, since the cache sizes are uniform, only one cache constraint is needed, and since all file are
homogeneous, an equal amount of memory is allocated to each. The cache constraint thus simplifies to
\[ \sum_{j=1}^{K} \binom{K-1}{j-1} v_j \leq MF/N, \]  
(4.3)
because there are \( \binom{K-1}{j-1} \) subsets of size \( j \) that contain the index \( k \). As a final step, we follow [19, 20] and normalize the file length \( F = 1 \) in (4.1)-(4.3). This yields the following linear programming problem with \( K + 1 \) variables, \( K + 3 \) constraints, and \( K \) terms in the objective function:

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=0}^{K-1} \binom{K}{j+1} v_j \\
\text{subject to} & \quad \sum_{j=0}^{K} \binom{K}{j} v_j = 1, \\
& \quad \sum_{j=1}^{K} \binom{K-1}{j-1} v_j \leq M/N, \\
& \quad v_j \geq 0, \quad \forall j \in \mathcal{U}. 
\end{align*}
\]

(4.4)-(4.7)

Note that the reduction to an optimization problem that scales as a linear function of \( K \) and as a constant function of \( N \) is possible because of the symmetry created by the uniform file length, file popularity, and cache size. Later in this thesis, cases when one or more of these parameters are non-uniform are treated; but the reduction from the exponential order will no longer be without loss of generality.

Note also that the schemes of [19, 20] and [21] are all feasible points of this problem. In particular, assuming that \( t = KM/N \) is an integer, the caching scheme of [19, 20] sets \( v_t = 1/(K t) \) and \( v_j = 0 \) if \( j \neq t \), while the decentralized scheme of [21] sets the variables to be of the form \( v_j = (M/N)^j (1 - M/N)^{K-j} \) for all \( j \). For the non-integer \( t \) case, a similar scheme is also stated in [19, 20].

4.1 Problem Analysis

The following theorem shows that the caching scheme of [19, 20] is, in fact, the optimal scheme among the broad family of schemes discussed above. The theorem also works for the cases of integer or non-integer \( t \). We note that while a similar result for the integer \( t \) case exists in [34], the proof used here uses a novel reformulation over the probability simplex. It is considerably simpler and lends additional insight into the problem.

**Theorem 1.** The unique, optimal solution to (4.4)-(4.7) is:

- For \( t = KM/N \in \mathbb{Z} \):

\[
v_j^* = \begin{cases} 
1/(K t) & \text{if } j = t, \\
0 & \text{if } j \neq t. 
\end{cases}
\]

(4.8)
For \( t = KM/N \notin \mathbb{Z} \):

\[
v_j^* = \begin{cases} 
\frac{s}{\binom{K}{j}} & \text{if } j = \lfloor t \rfloor \\
\frac{(1 - s)}{\binom{K}{\lceil t \rceil}} & \text{if } j = \lceil t \rceil \\
0 & \text{else}
\end{cases}
\]  

(4.9)

where \( s = \lfloor t \rfloor - t \).

Before the proof of Theorem 1 is stated, consider the change of variables \( a_j = v_j / \binom{K}{j} \); after some mild algebra, the original problem (4.4)-(4.7) can be reformulated as:

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=0}^{K} \frac{K - j}{j + 1} a_j \\
\text{subject to} & \quad \sum_{j=0}^{K} ja_j \leq KM/N, \\
& \quad \sum_{j=0}^{K} a_j = 1, \\
& \quad a_j \geq 0, \forall j \in [0 : K].
\end{align*}
\]  

(4.10)-(4.13)

In this form, the optimization problem is more easily understood. Constraints (4.12) and (4.13) restrict the feasible space to the probability simplex. The coefficients of the objective function are a decreasing function of \( j \) with positive second differences, and the cache constraint (4.11) now has coefficients that grow linearly in the index of summation. Intuitively, the optimal solution will involve placing as much “probability mass” in high-\( j \) \( a_j \) variables to lower the objective function, but not so high as to violate the cache constraint. This reasoning will be will be explicated later in the proof of Theorem 1, wherein the optimal value of (4.10)-(4.13) will be determined directly, and then converted back to the optimal solution of (4.4)-(4.7); but first, some intermediate results are required.

**Lemma 1.** At the optimal solution \( \mathbf{a}^* \) of (4.10)-(4.13), the cache constraint (4.11) is tight.

**Proof of Lemma 1:** In the practical context, this means that the optimal caching strategy will not waste any cache space. We will show that if the cache constraint (4.11) is not tight for some feasible solution, there always exists a second feasible solution that produces a strictly lower objective function value. Specifically, consider an arbitrary feasible solution \( \mathbf{a} = [a_0, \ldots, a_K]^T \) such that

\[
KM/N - \sum_{j=0}^{K} ja_j = \epsilon > 0.
\]

Since \( \mathbf{a} \) lies on the probability simplex, there exists at least one index \( i \) for which \( a_i \neq 0 \). We will form a new solution by “shifting” some portion of \( a_i \) to \( a_{i+1} \) in order to reduce the objective function value without violating the cache constraint. Formally, there exists a new solution \( \tilde{\mathbf{a}} = [\tilde{a}_0, \ldots, \tilde{a}_K]^T \) with, for some small \( \Delta > 0 \), \( \tilde{a}_i = a_i - \Delta \), \( \tilde{a}_{i+1} = a_{i+1} + \Delta \), and \( \tilde{a}_j = a_j \) for all other indices \( j \). The
increase in the cache consumption is given by
\[ K \sum_{j=0}^{K} j \bar{a}_j - K \sum_{j=0}^{K} ja_j \]
\[ = (i(a_i - \Delta) + (i + 1)(a_{i+1} + \Delta) - (i(a_i + (i+1)a_{i+1}) - i\Delta + (i+1)\Delta) \]
\[ = \Delta, \tag{4.14} \]
so as long as \( \Delta \leq \epsilon \), the cache constraint is not violated. Since \( \Delta \) can be chosen arbitrarily small, \( \bar{a}_i \) can still be positive, and since the amount taken from \( a_i \) was given to \( a_{i+1} \), \( \bar{a} \) remains on the probability simplex, i.e. it is a feasible solution.

The change is in the objective function is given by
\[ K \sum_{j=0}^{K} K - j \bar{a}_j - K \sum_{j=0}^{K} ja_j \]
\[ = \left( \frac{K-i}{i+1} (a_i - \Delta) + \frac{K-i-1}{i+2} (a_{i+1} + \Delta) \right) - \left( \frac{K-i}{i+1} a_i + \frac{K-i-1}{i+2} a_{i+1} \right) \]
\[ = \Delta \left( \frac{K-i-1}{i+2} - \frac{K-i}{i+1} \right) < 0, \]
because the objective function coefficients decrease in the index \( j \), and so feasible solution without a tight cache constraint cannot be optimal. This concludes the proof of Lemma 1

**Lemma 2.** If a feasible solution \( a = [a_0, \ldots, a_K]^T \) to (4.10)-(4.13) has two non-zero variables \( a_{i_1} \neq 0 \) and \( a_{i_2} \neq 0 \) such that \( |i_2 - i_1| \geq 2 \), then \( a \) is not an optimal solution of (4.10)-(4.13).

**Proof of Lemma 2:** Like Lemma 2, we will show that there always exists a better feasible solution than \( a \), which we denote by \( \bar{a} = [\bar{a}_0, \ldots, \bar{a}_K]^T \). Without loss of generality, assume \( |i_2 - i_1| > 2 \) for convenience, although the proof technique is the same if \( |i_2 - i_1| = 2 \), and assume that \( i_2 > i_1 \). For some small \( \Delta > 0 \), set \( \bar{a}_{i_1} = a_{i_1} - \Delta, \bar{a}_{i_1+1} = a_{i_1+1} + \Delta, \bar{a}_{i_2} = a_{i_2} - \Delta, \bar{a}_{i_2-1} = a_{i_2-1} + \Delta \), and \( \bar{a}_j = a_j \) for all other \( j \) values. The only restriction on the size of \( \Delta \) is that it be small enough such that \( \bar{a}_{i_1} \) and \( \bar{a}_{i_2} \) are still positive. And since \( \Delta \) is being subtracted and added in equal measure, the point \( \bar{a} \) remains on the probability simplex.

Moreover, the cache constraint does not change: the change in the amount of cache used is given by
\[ \sum_{j=0}^{K} j \bar{a}_j - \sum_{j=0}^{K} ja_j \]
\[ = (i_1(a_{i_1} - \Delta) + (i_1 + 1)(a_{i_1+1} + \Delta) + (i_2 - 1)(a_{i_2-1} + \Delta) + i_2(a_{i_2} - \Delta)) \]
\[ - (i_1 a_{i_1} + (i_1 + 1)a_{i_1+1} + (i_2 - 1)a_{i_2-1} + i_2 a_{i_2}) \]
\[ = \Delta (-i_1 + (i_1 + 1) + i_2 - 1) \]
\[ = 0. \]
However, a similar calculation shows that the objective function changes by an amount given by

$$
\sum_{j=0}^{K} \frac{K-j}{j+1} \bar{a}_j - \sum_{j=0}^{K} \frac{K-j}{j+1} a_j = \Delta \left( \left( \frac{K-i_1-1}{i_1+2} - \frac{K-i_1}{i_1+1} \right) - \left( \frac{K-i_2}{i_2+1} - \frac{K-i_2+1}{i_2} \right) \right),
$$

(4.15)

which is less than zero: both differences in (4.15) are negative, because the objective function coefficients are decreasing functions of the index, and the first difference is larger in magnitude than the second difference, because the objective function coefficients themselves have positive second differences. Thus $\bar{a}$ yields a better objective function value than $a$, which concludes the proof.

Lemma 2 has some important consequences. Specifically, it rules out any feasible solution that has three or more non-zero $a_j$ variables, because that would necessarily require at least two of the variables to have indices greater than two apart. This implies that the optimal solution has either only one non-zero variable, or two non-zero variables that have adjacent indices, i.e. $i_2 = i_1 + 1$. The proof of Theorem 1 can now be stated.

**Proof of Theorem 1:** From Lemma 1, the cache constraint (4.11) must be satisfied with equality, and from Lemma 2, there are at most two non-zero variables, which must have adjacent indices. Thus, the constraints for the optimal solution to (4.10)-(4.13) reduce to

$$ja_j + (j+1)a_{j+1} = KM/N
$$

(4.16)

and

$$a_j + a_{j+1} = 1
$$

(4.17)

for an appropriate index $j$, and with $a_j \geq 0$ and $a_{j+1} \geq 0$. But the system of equations (4.16)-(4.17) has a unique solution that satisfies the positivity requirements on the variables. Indeed, because of (4.17) and the positivity constraints, (4.16) states that $KM/N$ is a convex combination of $j$ and $j+1$. Thus it must be the case that $j \leq KM/N$, and $j+1 \geq KM/N$. If $KM/N \not\in Z$, the only integer possibility for $j$ is $j = \lfloor KM/N \rfloor$ and $j+1 = \lceil KM/N \rceil$; representing the non-integer value $KM/N$ as $s\lfloor KM/N \rfloor + (1-s)\lceil KM/N \rceil$ for a unique $s \in (0,1)$, it becomes clear that the unique solution to (4.16)-(4.17) is $a_j = s, a_{j+1} = 1-s$ for $j = \lfloor KM/N \rfloor$. Using the change of variables $v_i = a_i/\binom{K}{i}$, this gives the optimal solution to (4.4)-(4.7) as stated by (4.9). If $KM/N \in Z$, then without loss of generality, we can set $j = KM/N$, and then the unique solution to (4.16)-(4.17) is $a_j = 1, a_{j+1} = 0$. Using the aforementioned change of variables, this gives the optimal solution to (4.4)-(4.7) described by (4.8), which concludes the proof of the Theorem.

As a final remark in this section, we acknowledge that there exist stronger results about optimal coded caching schemes than Theorem 1 in the literature, for instance, [60] shows that a slightly modified version of the scheme in [19, 20] is the optimal coded caching scheme among all schemes with uncoded cache content. We nonetheless believe the optimization theoretic perspective of the [19, 20] scheme presented here is a noteworthy one, especially given that it easily begets extensions to more practical scenarios; this is the focus of the rest of this thesis.
Chapter 5

Heterogeneous Coded Caching

We now move onto the coded caching problem with heterogeneous parameters. Although the optimization problem (3.8)-(3.9) is already capable of accounting for non-uniform file popularity, file length, and cache size, the problem size scales exponentially in system parameters, hence the optimization problem is intractable for practical system sizes. The main contributions of this and the next section are to develop simplifications to (3.8)-(3.9) that reduce the computational complexity of the problem while maintaining high-quality performance. Each non-uniformity is considered both individually and in conjunction with the others in order to gain insight into the interactions of their respective effects. We begin by examining the effect of non-uniform file popularity and non-uniform file length, but for now keep the cache sizes uniform across the users.

The procedure for each case is roughly as follows: first a new set of variables are defined that are intended to capture some structural feature of the problem, e.g. the $v_j$ variables of the simplified homogeneous problem (4.4)-(4.7). Next, certain conditions, referred to as memory inequality constraints, are imposed on the new variables (see e.g. (5.11)), which forces feasible solutions to dedicate more cache memory to certain kinds of files, e.g. more popular files. While no a priori justification for the use of these variable and constraints is given, subsequent numerical results justify their use a posteriori.

These memory inequality constraints then allow the simplification of the objective function (3.8). First, the max function can be eliminated from (3.8), since the memory inequality constraints are sufficient to determine a priori which subfiles will be the maximum given some request vector $d$. This in turn allows the expected rate to computed precisely in terms of the largest file requested, the second largest file requested, and so on, instead of in terms of the $N^K$ possible request vectors. Since there are $K$ files requested, and $N$ possibilities for each file, this ultimately reduces the scaling of the objective function from exponential to polynomial, although the objective function does not necessarily scale with $NK$ precisely.

To this end, we introduce two lemmas, the proofs of which are contained in Appendix A.

**Lemma 3** (Chu-Vandermonde Convolution). For positive integers $N, N_1, N_2,$ and $n$, with $N_1 + N_2 = N$ and $n \leq N$,

$$\binom{N}{n} = \sum_{k=0}^{n} \binom{N_1}{k} \binom{N_2}{n-k}$$

(5.1)

**Lemma 4.** Consider $K$ independent multinomial random trials with $N$ possible outcomes per trial, with probabilities $\{p_1, p_2, \ldots, p_N\}$, denoted by $Z \in [N]^K$, i.e., $Z_i$, the $i$-th element of $Z$, is the outcome
of the \(i\)-th trial. Let the random vector \(Y\) be a sorted version of \(Z\), but with index shifted by 1, so that \(Y_0\) is the smallest element of \(Z\), \(Y_1\) is the second smallest element, and so on. The probability mass function of \(Y_m\) is then given by

\[
\Pr[Y_0 = i] = \left( \sum_{l=i}^{N} p_l \right) K - \left( \sum_{l=i+1}^{N} p_l \right)^K,
\]

(5.2)

\[
\Pr[Y_1 = i] = \Pr[Y_0 = i] + K \left( \sum_{l=1}^{i-1} p_l \right) \left( \sum_{l=i}^{N} p_l \right)^{K-1} - \left( \sum_{l=1}^{i} p_l \right) \left( \sum_{l=i+1}^{N} p_l \right)^{K-1},
\]

(5.3)

and for \(m \in [2 : K-1]\),

\[
\Pr[Y_m = 1] = \sum_{k=0}^{K-m-1} \left( \begin{array}{c} K \\ m + 1 + k \end{array} \right) p_1^{m+1+k} (1 - p_1)^{K-m-1-k}
\]

(5.4)

\[
\Pr[Y_m = i] = \left( \begin{array}{c} K \\ K - m \end{array} \right) \left( \sum_{l=i}^{N} p_l \right)^{K-m} - \left( \sum_{l=i+1}^{N} p_l \right)^{K-m} \left( \sum_{l=1}^{i} p_l \right)^m
\]

\[
+ \sum_{k=0}^{K-2 \min\{m-1,K-2-k\}} \sum_{b=\max\{0,m-1-k\}}^{K-2-k-b} \left( \begin{array}{c} i-1 \\ 2+k, b, K-2-k-b \end{array} \right) p_i^{2+k} \left( \sum_{l=1}^{i} p_l \right)^b \left( \sum_{l=i+1}^{N} p_l \right)^{K-2-k-b}
\]

(5.5)

where the final expression is for \(i \in [2 : N]\).

### 5.1 Non-Uniform File Popularity and Length

#### 5.1.1 Background

The effect of non-uniform file popularity, also referred to as non-uniform demands, on coded caching has been explored in a number of papers [22–33]. In [22, 23], Niesen and Maddah-Ali modify their decentralized scheme from [21] by first grouping users according to the popularity of their respective file requests, and then transmitting to the groups sequentially using the scheme from [21]. The authors of [24–28] develop an order-optimal scheme using random caching with a graph-based algorithm to design coded multicast transmissions, with [26] focusing specifically on the application of video delivery. In [27, 28], both the popularity of the files and their request correlation are considered when designing the caching and transmission scheme, while in [29–32], a heterogeneous network structure is considered, with file popularity organized in discrete levels. Using a novel random caching-based scheme, [33] is able to show order-optimality with a constant that is independent of the popularity distribution. Finally, we repeat earlier comments that [34] uses the same optimization framework that is used to study non-uniform popularity in great depth; we nevertheless show our (similar) work for the sake of exposition.

The literature on the effects of non-uniform file length is, however, comparatively scarce. To the best of our knowledge, Zhang et al [35] provide the only scheme designed to accommodate non-uniform file length for a general number of users. A scheme is provided that uses random caching with a transmission scheme similar to the one used in this thesis, and upper and lower bounds on system performance are
derived. In particular, [35] explores a random caching scheme where files are cached with a probability proportional to size of the file. Non-uniform file size is also explored in the recent letter [36], in which the achievable rate region for both non-uniform file size and non-uniform cache size is characterized, but only for the case of $K = 2$ users and $N = 2$ files.

Note that it can be argued that if files are indeed different sizes, they can be broken up into smaller packets of a constant size $F'$ bits, and then treated as separate files. While this is a reasonable assumption while investigating other aspects of a coded caching scheme, there are two issues that need be addressed in practice. First, if a file is broken up into multiple pieces, then a user who seeks to to download the entire file must make multiple (correlated) requests to the server - a fact that should be accounted for in subsequent system design. The second practical issue comes from the fact that it is unclear how to set the common file size $F'$: efficiency demands that $F'$ be as large as possible so that any required headers represent a small proportion of the entire download, while at the same time, $F'$ should also be small enough to divide files without significant remainder.

We therefore contend that heterogeneous file length is an important parameter that a practical system must capable of accommodating in one way or another. The approach to handling non-uniform files sizes discussed above may indeed have some merit; some work has been done to analyze the case of multiple requests from users, see e.g. [37–40], and so an approach based on this technique may be viable. This thesis, however, uses a different approach: files are not broken up into smaller files of equal lengths, and so the cache content is designed to accommodate their different lengths. Indeed, there may be other approaches still, as neither approach described above may be appropriate for all cases. For instance, viral internet videos may be as short as a few seconds or as long as a few minutes. The caching approaches described above may not work well in the event of such significant disparity in length, and so the possibility of comparing the performance of different approaches in different cases is left to future work.

To the best of our knowledge, there has been no work exploring the relationship between file length and popularity and the resulting effect on cache content. In the literature discussed above, it is noted (roughly) that more popular files should be allocated more cache memory, but also that larger files should be allocated more cache memory. Given that, in general, file lengths and popularity may not have any correlation, it is not clear how these non-uniformities jointly affect optimal cache content. These interactions are explored in the following.

### 5.1.2 Problem Formulations

The first step in reducing the exponential number of variables in (3.8)-(3.9) is the definition of a new set of $(K + 1)N$ variables as

$$v_{l,j} = |W^{(l)}_S|, \forall S \text{ s.t. } |S| = j, \forall l \in [N].$$  \hspace{1cm} (5.6)

similar to the $v_j$ variables used in Section 4, except now there is a set of $v_j$ variables for each file to capture the difference in length and popularity between files. Such a reduction enforces $|W^{(l)}_S|$ to depend only on the cardinality of $S$, which is a reasonable thing to do and in fact can be proved to be without loss of generality for the special case of non-uniform file popularity alone but with uniform file length [34]. The effect of this reduction in the general case is numerically evaluated later in the section.

The simplification of the general constraints in (3.9) follows similar reasoning used to obtain the
constraints (4.2) - (4.3) in Section 4, except now there are arbitrary file lengths $F_l$ and popularities $p_l$; this gives

$$\sum_{j=0}^{K} \binom{K}{j} v_{l,j} = F_l, \forall l \in [N]$$

as the file reconstruction constraints, and

$$\sum_{j=1}^{K} \binom{K-1}{j-1} v_{l,j} \leq \mu_l, \forall l \in [N]$$

for the cache constraint. The other two constraints,

$$v_{l,j} \geq 0, \forall l \in [N], \forall j \in [0 : K],$$

and

$$\sum_{l=1}^{N} \mu_l \leq M$$

have more obvious modifications.

To express the objective function in polynomial number of terms, we now need to impose certain memory inequality conditions in order to simplify the max operator in the objective. We propose two different approaches called the popularity-first approach and the length-first approach respectively for handling the non-uniform file popularity and file length.

**Popularity-First Approach**

In the popularity-first approach, files are labelled in decreasing order of popularity, i.e. such that $p_1 \geq \cdots \geq p_N$. Then, motivated by the idea that more popular files ought to have more cache space dedicated to them, a memory inequality condition is imposed on the cache content as

$$v_{l_1,j} \geq v_{l_2,j}, \forall l_1, l_2 \in [N] \text{ s.t. } l_1 < l_2, j \in [K].$$

This memory inequality constraint is adopted to help reduce the complexity of the problem; as previously discussed, this constraint (and others like it later in the thesis) allow the max function in (3.8) to be eliminated in favour of a linear function of the variables, which in turn allows for the expected rate to be computed in a polynomial number of operations, instead of the exponential number required by (3.8).

The popularity-first approach is most appropriate for the special case of non-uniform file popularity alone, but with uniform file length. The following proposition shows explicitly the effect that (5.11) has on the objective function in this case. Note that in this special case of uniform file length, (5.11), which holds for $j = 1, \ldots, K$, becomes reversed for $j = 0$. To see this, consider two files $l_1$ and $l_2$ with $l_1 < l_2$, that satisfy (5.11); if both have length $F$, i.e. satisfy (5.7) with $F_{l_1} = F_{l_2} = F$, then $v_{l_1,0} \leq v_{l_2,0}$ because every other subfile of $l_1$ is larger than every other subfile of $l_2$.

Before stating Proposition 1, we note that, as mentioned earlier, this special case of non-uniform file popularity alone with uniform file length has already been considered in independent work [34]. In the proof of Proposition 1, the expected rate is simplified by first fixing the demand vector, and analyzing the number of bits sent to all subsets of a given size. The approach in [34] is the reverse: for subsets
of a given size, they characterize the number of bits sent across all demand vectors. This yields two
cosmetically different but mathematically equivalent expressions for the expected rate in the case of
non-uniform probability only. The expression in [34] is in fact the simpler of the two, as it only requires
the computation of what we denote as \( \Pr[Y_0 = l] \) for all files \( l \), and in fact a similar simplified problem
using this approach may be possible for the case of non-uniform file size; the development of such a
problem, however, is left to future work.

**Proposition 1.** Consider the case of non-uniform file popularity and uniform file length. Let the vari-
ables defined in (5.6) be subject to condition (5.11) with files labelled in decreasing order of popularity.
Then the objective function (3.8) simplifies exactly as

\[
E\left[ \sum_{S \in \mathcal{P}(d) \setminus \emptyset} \max_{k \leq S} \left( \frac{K-1}{j} \right) \mathbb{P}[Y_i = l] v_{l,j} \right] = \sum_{i=0}^{K-1} \sum_{l=1}^{N} \left( \frac{K-1-i}{j} \right) \mathbb{P}[Y_i = l] v_{l,j} + \sum_{i=0}^{K-1} \sum_{l=1}^{N} \mathbb{P}[Y_{K-i-1} = l] v_{l,0},
\]

(5.12)

where \( Y_i \) is the random variable representing the \((i+1)\)-th smallest index in a random request vector \( d \).

The proof of Proposition 1 is contained in Appendix B. Note that the probabilities \( \mathbb{P}[Y_i = l] \) can
be obtained directly from Lemma 4: since the files are labelled in decreasing order of popularity, the
probability that \( l \) is the \((i+1)\)-th smallest file requested in \( d \) is equivalent to the probability that \( l \) is
the \((i+1)\)-th smallest index in \( Z \). The optimization problem for the case of non-uniform file popularity,
uniform file length, and uniform cache size can now be written as

\[
\begin{align*}
\text{minimize} & \quad \sum_{i=0}^{K-1} \sum_{l=1}^{N} \left( \frac{K-1-i}{j} \right) \mathbb{P}[Y_i = l] v_{l,j} + \sum_{i=0}^{K-1} \sum_{l=1}^{N} \mathbb{P}[Y_{K-i-1} = l] v_{l,0} \\
\text{subject to} & \quad (5.7)-(5.11),
\end{align*}
\]

(5.13)

Note that this is a linear program, with a number of summands in the objective function that scales
\( K^2 N \), and exactly \( KN \) variables and \( N(K+3) + K(N(N-1))/2 + 1 \) constraints.

If the files are also of non-uniform length, further work is required, but a similar optimization problem
can nonetheless be developed. The details are omitted here both for the sake of brevity and because
numerical results suggest that the popularity-first approach does not perform as well as the length-first
approach in the general case when both popularity and file lengths are non-uniform.

**Length-First Approach**

In the length-first approach, files are labelled in decreasing order of length, i.e. such that \( F_1 \geq F_2 \geq \cdots \geq F_N \). Then, motivated by the idea that longer files ought to have more cache space dedicated to
them, the following memory inequality condition is imposed:

\[
v_{l_1,j} \geq v_{l_2,j}, \forall l_1, l_2 \in [N] \text{ s.t. } l_1 < l_2, j \in [0 : K-1].
\]

(5.15)

Using similar reasoning as Proposition 1, it is straightforward to show that

**Proposition 2.** Consider the case of non-uniform file length with either uniform or non-uniform pop-
ularity. Let the variables defined in (5.6) be subject to condition (5.15) with files labelled in decreasing
order of length. Then the objective function (3.8) simplifies exactly as
\[
\mathbb{E} \left[ \sum_{S \in \mathcal{P}(U) \setminus \emptyset} \max_{k \in S} \{W^{(d_k)}_{S(k)}\} \right] = \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{l=1}^{N} \left( K - 1 - i \right) \Pr[Y_i = l] v_{l,j} \quad (5.16)
\]
where \( Y_i \) is the random variable representing the \((i+1)\)-th smallest index in a random request vector \( d \).

The length-first optimization problem for non-uniform file popularity, non-uniform file length, and uniform cache size is obtained as
\[
\begin{align*}
\text{minimize} & \quad \sum_{j=0}^{K-1} \sum_{i=0}^{K-1} \sum_{l=1}^{N} \left( K - 1 - i \right) \Pr[Y_i = l] v_{l,j} \\
\text{subject to} & \quad (5.7)-(5.10), (5.15) \quad (5.18)
\end{align*}
\]
which is a linear program with \( K^2 N \) summands in the objective function, \( KN \) variables, and \( N(K + 3) + K(N(N - 1))/2 + 1 \) constraints.

5.1.3 Numerical Results

To evaluate the effect of the simplified problem formulation, we consider a case with \( K = 4 \) users with equal cache sizes of \( M \), and \( N = 6 \) files. When the files are of uniform length, the value \( F = 1 \) is used, and when they are of non-uniform length, the values \( \{F_1, \ldots, F_6\} = \{9/6, 8/6, 7/6, 5/6, 4/6, 3/6\} \) are used. Similarly, when the files are of uniform popularity, the value \( p_l = 1/N \) is used for all \( l \in [N] \), but when the files have non-uniform popularity, the distribution is given by a Zipf distribution with parameter \( s \), which has been observed empirically to be a reasonable model for user demands; a parameter of \( s = 0.56 \) is used in this thesis (see e.g. [5]). When both the file lengths and popularities are non-uniform, the relationship between length and popularity is specified explicitly.

Fig. 5.1 compares the rate-memory tradeoff curve for the original problem (3.8)-(3.9) and the simplified problem (5.13)-(5.14) for the non-uniform popularity and uniform length case. A baseline random caching scheme is also included for reference. This random caching scheme is essentially the decentralized scheme of [21] but with file \( n \) allocated \( \mu_n F \) bits of cache memory instead of \( MF/N \) bits; initially, the value is obtained as \( \mu_n = \min\{M p_n, 1\} \) for all \( n \in [N] \), and if \( \sum_{n=1}^{N} \mu_n < M \) after that, the remaining cache memory is allocated to each file sequentially until the remaining memory runs out.

Conversely, Fig. 5.2 compares the rate-memory tradeoff curves of (3.8)-(3.9) and the simplified problem (5.17)-(5.18) for the non-uniform length and uniform popularity case. The random caching baseline scheme used here is essentially equivalent to the one proposed in [35]. Both figures show that the performance of the general problem and the two simplified problems is identical for the respective cases considered here. Indeed, when the numerical solutions of (3.8)-(3.9) for these two cases are examined explicitly, it is clear that the memory constraint conditions are indeed satisfied, and so the optimal solutions in these cases are attainable by the respective simplified problems.

Next, Fig. 5.3 compares the performance of the original problem (3.8)-(3.9) to both the length-first and popularity-first simplified problems for the case of non-uniform file length and popularity. The specific pairings of length and popularities used and their associated labels are listed in Table 5.1. Although somewhat arbitrary, these file length and popularity combinations are intended to simulate a practical scenario where file popularity and length are relatively uncorrelated. While examining this
Figure 5.1: A comparison of the performance of the solution obtained from (3.8)-(3.9) to the solution obtained by (5.13)-(5.14), with reference to a baseline random caching scheme, for the case of non-uniform file popularity only.

Figure 5.2: A comparison of the performance of the solution obtained from (3.8)-(3.9) to the solution obtained by (5.17)-(5.18), with reference to a baseline random caching scheme, for the case of non-uniform file length only.
Figure 5.3: A comparison of the performance of the solution obtained from (3.8)-(3.9) to the solutions obtained by (5.17)-(5.18) and a popularity-first optimization problem, with reference to a baseline random caching scheme, for the case of non-uniform file length and popularity.

individual case is not sufficient for determining general patterns, it is enough to gain some important insight about the tension between file length and popularity.

Fig. 5.3 shows that in this case, the length-first scheme yields much better results than the popularity-first scheme. Indeed, the length-first scheme obtains the same performance as the original problem for all $M$ considered except $M = 1$. The reason for this divergence can be seen from Table 5.2 which shows the optimal solution to the original problem (3.8)-(3.9) in the $M = 1$ case. The value of $|W^{(S)}_S|$ is shown in the $l$-th column of the row labelled with $S$, and the files are ordered using the length-first labelling of Table 5.1. It is clear that file 1, the largest file but fifth-most popular, has been allocated less cache memory than files 2 and 3, which are the third and second most popular files respectively. Thus the memory-inequality constraint (5.15) is violated and so the length-first simplified problem cannot attain the optimal solution of (3.8)-(3.9). Despite this, the optimal value to the simplified problem is only

Table 5.1: The file labelling and length-popularity pairings used in the non-uniform file popularity and length case.

<table>
<thead>
<tr>
<th>Length-First (LF) and Popularity-First (PF) Labelling</th>
<th>Length-First (LF) and Popularity-First (PF) Labelling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LF File Index</strong></td>
<td><strong>PF File Index</strong></td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>
Table 5.2: Optimal subfile sizes and memory allocation for the general problem (3.8)-(3.9) with \( K = 4, N = 6, M = 1 \), and file lengths and popularities given in Table 5.1 (using length-first indexing), with values rounded to three decimal places.

<table>
<thead>
<tr>
<th>Subset</th>
<th>File Index</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \emptyset )</td>
<td>0.833</td>
<td>0.583</td>
<td>0.417</td>
<td>0.167</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( {1} )</td>
<td>0.167</td>
<td>0.188</td>
<td>0.188</td>
<td>0.167</td>
<td>0.167</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>( {2} )</td>
<td>0.167</td>
<td>0.188</td>
<td>0.188</td>
<td>0.167</td>
<td>0.167</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>( {3} )</td>
<td>0.167</td>
<td>0.188</td>
<td>0.188</td>
<td>0.167</td>
<td>0.167</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>( {4} )</td>
<td>0.167</td>
<td>0.188</td>
<td>0.188</td>
<td>0.167</td>
<td>0.167</td>
<td>0.125</td>
<td></td>
</tr>
<tr>
<td>( {1,2} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td></td>
</tr>
<tr>
<td>( {1,2,3,4} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Total memory:</td>
<td>0.167</td>
<td>0.188</td>
<td>0.188</td>
<td>0.167</td>
<td>0.167</td>
<td>0.125</td>
<td></td>
</tr>
</tbody>
</table>

about \( 10^{-4} \) larger than the optimal value of (3.8)-(3.9), and so the difference is not significant. It would thus appear that, while probability cannot be completely ignored in theory, a length-first approach to caching can yield very good results in practice. This insight is later used in Section 5.3.2, but first the problem of non-uniform cache size must be studied on its own first; this is done next.

## 5.2 Non-Uniform Cache Size

### 5.2.1 Background

We next consider simplifying the optimization formulation for the case with non-uniform cache sizes. For now, file popularity and file length are kept uniform; the case with all parameters being non-uniform is treated in the subsequent section. For the case of non-uniform cache size, a decentralized coded caching scheme is developed in [41] and subsequently improved upon in the \( K > N \) case by [42,43]. As previously discussed, [44] uses an optimization framework similar to the one used in this thesis to generate a scheme for the centralized case. For the sake of completion, we note again the work [36] in which the rate region for both non-uniform cache and file size is characterized for \( K = 2 \) users and \( N = 2 \) files, but the optimal scheme is not yet known for the general case.

### 5.2.2 Problem Formulation

We first consider a simple case where there are only two cache sizes, “large” and “small”, represented by \( M_L \) and \( M_S \) respectively. The variable \( K_L \) is used to represent the number of users with large caches, and \( K_S \) is used to represent the number of users with small caches, such that \( K_L + K_S = K \). Note that file lengths and popularities are fixed as uniform, i.e., \( p_1 = \cdots = p_N = 1/N \) and \( F_1 = \cdots = F_N = F = 1 \) respectively.

Since files here are equally popular and of the same size, we have symmetry across files, but now the symmetry across users is broken by the non-uniform cache size. However, certain user symmetry still exists, i.e., symmetry among members of the same cache size group. We therefore define three sets of variables for \( j \in [0 : K] \), denoted \( v_{j,S}, v_{j,L}, \) and \( v_{j,M} \), as follows. For a subset of users \( \mathcal{S} \) with a size
$|S| = j$,

$$|W_S| = \begin{cases} v_{j,S} & \text{if } S \text{ contains only small-cache users,} \\ v_{j,L} & \text{if } S \text{ contains only large-cache users,} \\ v_{j,M} & \text{otherwise,} \end{cases}$$

(5.19)

for all files $l \in [N]$.

This definition suggests some natural constraints. First, since there are only $K_S$ small-cache users, there cannot be a group of $j$ small-cache users if $j > K_S$, so we set

$$v_{j,S} = 0, \forall j > K_S. \quad (5.20)$$

Similarly, for large-cache users,

$$v_{j,L} = 0, \forall j > K_L. \quad (5.21)$$

A similar constraint is required for the $v_{1,M}$. Since there cannot be a subset of size one with both large- and small-cache users, we require

$$v_{1,M} = 0; \quad (5.22)$$

Moreover, since the $j = 0$ variables correspond to subsets of size 0, it is not particularly meaningful to discuss whether or not this corresponds to a small, large, or mixed subset. To avoid any further complications, we simply set

$$v_{0,S} = v_{0,L} = v_{0,M} = v_0, \quad (5.23)$$

To understand the file reconstruction condition, note that, for $j \geq 2$, there are $\binom{K_S}{j}$ groups of small-cache users, $\binom{K_L}{j}$ groups of large-cache users, and

$$\sum_{i=1}^{j-1} \binom{K_S}{i} \binom{K_L}{j-i}$$

mixed groups, because each mixed group must have at least one small-cache user and at least one large-cache user. By adding and subtracting the $i = 0$ and $i = j$ terms and using Lemma 3, we can rewrite this as

$$\sum_{i=0}^{j} \binom{K_S}{i} \binom{K_L}{j-i} - \binom{K_L}{j} - \binom{K_S}{j} = \binom{K}{j} - \binom{K_L}{j} - \binom{K_S}{j},$$

and so the total portion of the file cached by subsets of size $j \geq 2$ is

$$\binom{K_L}{j} (v_{j,L} - v_{j,M}) + \binom{K_S}{j} (v_{j,S} - v_{j,M}) + \binom{K}{j} v_{j,M}. \quad (5.24)$$

For $j = 1$, there are $K_L = \binom{K_L}{1}$ large users and $K_S = \binom{K_S}{1}$ small users. Moreover, since $v_{1,M} = 0$, it is easy to see that (5.24) also holds for $j = 1$. Finally, consider the $j = 0$ case. Here we only need to add $v_0$ to capture the portion of the file not cached by any user. However, note that if we set $j = 0$ in (5.24) and apply constraint (5.23), the first two terms become 0, while the last term reduces to $v_{0,M} = v_0$. 
Thus (5.24) applies for all $j$, and so we can conveniently express the file reconstruction constraint as

$$
\sum_{j=0}^{K} \left( \frac{K_L}{j} \right) (v_{j,L} - v_{j,M}) + \left( \frac{K_S}{j} \right) (v_{j,S} - v_{j,M}) + \left( \frac{K}{j} \right) v_{j,M} = 1.
$$

(5.25)

Similar reasoning is used to obtain the cache memory constraints. A small cache user caches every subfile labelled with a subset in which he is contained as a member, and so necessarily caches only subfiles of size $v_{j,S}$ and $v_{j,M}$. Specifically, a small-cache user is a member of $\binom{K_S-1}{j-1}$ small groups of size $j$, and

$$
\sum_{i=0}^{K-2} \binom{K_S-1}{i} \left( \frac{K_L}{j-1-i} \right)
$$

(5.26)
mixed groups. Using Lemma 3 once again, the cache memory constraint for the small user is obtained as

$$
\sum_{j=1}^{K} \binom{K_S-1}{j-1} (v_{j,S} - v_{j,M}) + \binom{K-1}{j-1} v_{j,M} = M_S/N.
$$

(5.27)

Note that the $j = 1$ expression reduces to $\binom{K_L-1}{0} (v_{1,S} - 0) + 0 = v_{1,S}$, as desired. Note also that each file is allocated an equal $M_S/N$ of the cache because the files are equally sized and equally popular. Using similar reasoning for large-cache users, we obtain

$$
\sum_{j=1}^{K} \binom{K_L-1}{j-1} (v_{j,L} - v_{j,M}) + \binom{K-1}{j-1} v_{j,M} = M_L/N.
$$

(5.28)

As always, it is required that all subfiles be of nonnegative size:

$$
v_{j,S} \geq 0, v_{j,L} \geq 0, v_{j,M} \geq 0, \forall j \in \{0, \ldots, K\}.
$$

(5.29)

Finally, the memory inequality constraints for this problem are introduced. In general, we would expect the subfiles that large users have cached to be longer than the ones cached by small users. This is codified in the problem explicitly with

$$
v_{j,L} \geq v_{j,M}, j \in [2 : K_L]
$$

(5.30)

$$
v_{j,M} \geq v_{j,S}, j \in [2 : K_S]
$$

(5.31)

$$
v_{j,L} \geq v_{j,S}, j \in [1 : K_L].
$$

(5.32)

Again note that there is some redundancy in these constraints, but they are nonetheless included for clarity of exposition. Note also that the first two inequalities hold from $j = 2$ to $j = K_L$ and $j = K_S$ respectively; the $j = 1$ is already constrained by (5.22), while the $j = 0$ is constrained by (5.23), and the $j > K_L$ and $j > K_S$ cases are governed by (5.21) and (5.20) respectively.

As Proposition 3 shows, the memory inequality constraints allow us to greatly simplify the original objective function (3.8).

**Proposition 3.** Assuming uniform file length and popularity but two different user cache sizes, and with variables as defined in (5.19) and satisfying (5.20)-(5.23) and (5.29)-(5.32), the objective function
(3.8) simplifies exactly as

\[
E \left[ \sum_{S \in \mathcal{P}(\mathcal{M}) \setminus \emptyset} \max_{k \in S} \{|W^{(d_k)}_{S \setminus \{k\}}|\} \right] = \sum_{j=0}^{K-1} \left( K \frac{v_{j,S} - v_{j,M}}{j+1} \right) + \left( K \frac{v_{j,M}}{j+1} \right) 
\]

\[
+ \left( K \frac{v_{j,L} - v_{j,M}}{j+1} \right) + \left( K \frac{v_{j,M}}{j+1} \right) \quad (5.33)
\]

The proof of Proposition 3 is in Appendix C. The simplified optimization problem can then be written as

\[
\text{minimize} \sum_{j=0}^{K-1} \left( K \frac{v_{j,S} - v_{j,M}}{j+1} \right) + \left( K \frac{v_{j,M}}{j+1} \right)
\]

\[
+ \left( K \frac{v_{j,L} - v_{j,M}}{j+1} \right) + \left( K \frac{v_{j,M}}{j+1} \right) \quad (5.34)
\]

subject to (5.20)-(5.23), (5.25), (5.27)-(5.32).

(5.35)\]

This is a linear program that has a number of variables, constraints, and objective function summands that scale linearly in $K$.

Finally, we note that only two cache sizes were considered in this thesis. While a practical system would likely have more than two cache sizes, it is reasonable to expect that only a small number of cache sizes will be used, e.g. cell phones with 8, 16, 32 or 64 GB of cache memory. The reasoning used here for two cache sizes could then be extended to accommodate these additional cache sizes as needed.

### 5.2.3 Numerical Results

Consider a case where there are $N = 6$ files, uniform in popularity and length, and $K = 4$ users. Define a "memory factor" $M \in [0 : N]$; there are $K_S$ small users with a cache size of $M_S = 0.8M$, and $K_L$ large users with a cache size of $M_L = 1.2M$. Fig. 5.4 compares the corresponding solution of the general problem (3.8)-(3.9) to the solution of the simplified problem (5.35)-(5.36) when there are $K_S = 2$ small users. The random caching scheme of [41] is included, but the centralized scheme of [44] (which has exponential complexity) is not. The purpose here is not to determine the best caching scheme for the heterogeneous cache case, but to, first, demonstrate the implicit performance-tractability tradeoff of using the simplified problem (5.35)-(5.36) over the general problem, and second, to demonstrate that it is worth the effort of developing and using these problems to design cache content (rather than caching randomly) when the engineering context allows for it. Nevertheless, we expect the performance of the problem in [44] to be very similar, if not identical, to the exponential problem developed here, even though the two optimization frameworks are not identical themselves. Table 5.3 also shows the optimal cache content obtained from the general problem in the $K_S = 2, M = 4$ case.

Fig. 5.4 shows that in the $K_S = 2$ case, while the simplified problem tracks the optimal scheme for small cache size ($M \leq 3$), it performs worse than even the random caching scheme for large $M$ values. Table 5.3 reveals why this is the case. Here, users 1 and 2 are the small users, and users 3 and 4 are the large users. The variable definitions in (5.19) specify one variable for all mixed subsets of the same size, but consider the subfile sizes for the size-three subsets: the subsets with 2 small users have smaller subfiles than the subsets with 2 large users. Thus using only one $v_{M,j}^M$ variable for these four subsets results in a loss in performance. In principle, one could introduce more variables to accommodate this,
Figure 5.4: A comparison of the performance of the solution obtained from (3.8)-(3.9) to the solution obtained by (5.35)-(5.36) for $K_S = 2$.

Table 5.3: Optimal subfile sizes and memory allocation for the general problem (3.8)-(3.9) with $K = 4, N = 6, K_S = 2, M_S = 3.2$ and $M_L = 4.8$, with values rounded to three decimal places.

<table>
<thead>
<tr>
<th>File Index</th>
<th>Subset</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\emptyset$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>{4}</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>{1, 2}</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>{1, 3}</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>{1, 4}</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>{2, 3}</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>{2, 4}</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>{3, 4}</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
<td>0.056</td>
</tr>
<tr>
<td></td>
<td>{1, 2, 3}</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
</tr>
<tr>
<td></td>
<td>{1, 2, 4}</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
<td>0.033</td>
</tr>
<tr>
<td></td>
<td>{1, 3, 4}</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
</tr>
<tr>
<td></td>
<td>{2, 3, 4}</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
</tr>
<tr>
<td></td>
<td>{1, 2, 3, 4}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Mem. (L): 0.800 0.800 0.800 0.800 0.800 0.800
Mem. (S): 0.533 0.533 0.533 0.533 0.533 0.533
albeit at a cost of a more complicated objective function.

5.3 Non-Uniform File Size, Popularity and Cache Size

5.3.1 Background

The natural final step in this program is to develop a tractable optimization problem that accommodates non-uniformity in cache size, file size, and popularity at the same time. To the best of our knowledge, there has yet to be a caching scheme proposed that handles heterogeneity in all three of these domains.

5.3.2 Problem Formulation

We begin by defining a new set of variables that, in a sense, combines the functionality of the variables defined in (5.6) and (5.19). Let

$$\left|W^{(l)}_S\right| = \begin{cases} v^S_{l,j} & \text{if } S \text{ contains only small-cache users}, \\ v^L_{l,j} & \text{if } S \text{ contains only large-cache users}, \\ v^M_{l,j} & \text{otherwise}, \end{cases} \quad (5.37)$$

for a subset of users $S$ such that $|S| = j$ and all files $l \in [N]$. The symmetry across users is broken by the heterogeneity of the user cache size, and the symmetry across files is broken by the heterogeneity of files in both length and popularity. Nevertheless, we can still exploit the symmetry across subsets of users of the same size ($j$) and type (i.e. small, large, mixed) for a particular file ($l$) to reduce the complexity of the original problem (3.8)-(3.9) while still accounting for the aforementioned heterogeneity.

Many of the constraints used in the non-uniform cache size case can be converted to their equivalents for this new case. If $j > K_S$, there cannot be a subset of $j$ small cache users, so

$$v^S_{l,j} = 0, \forall j > K_S, \forall l \in [N], \quad (5.38)$$

and similarly

$$v^L_{l,j} = 0, \forall j > K_L, \forall l \in [N]. \quad (5.39)$$

Since there cannot be a subset of size 1 containing both large and small users, the $l = 1$ variable is constrained as

$$v^M_{l,1} = 0, \forall l \in [N], \quad (5.40)$$

and for convenience, we set

$$v^S_{l,0} = v^L_{l,0} = v^M_{l,0} = v_{l,0}, \forall l \in [N]. \quad (5.41)$$

The cache size-based memory inequalities should still hold, giving, for a fixed file $l$,

$$v^L_{l,j} \geq v^M_{l,j}, \ j \in [2 : K_L], \forall l \in [N], \quad (5.42)$$

$$v^M_{l,j} \geq v^S_{l,j}, \ j \in [2 : K_S], \forall l \in [N], \quad (5.43)$$

$$v^L_{l,j} \geq v^S_{l,j}, \ j \in [1 : K_L], \forall l \in [N]. \quad (5.44)$$

We also import conditions from the non-uniform file size and popularity problem. As seen earlier, it
is better to prioritize file length rather than popularity, and so we label the files in decreasing order of file length. This gives

\[ v_{l_1,j}^L \geq v_{l_2,j}^L, \quad j \in [0 : K - 1], \forall l_1, l_2 \text{ s.t. } l_1 < l_2 \]  
\[ (5.45) \]

\[ v_{l_1,j}^M \geq v_{l_2,j}^M, \quad j \in [0 : K - 1], \forall l_1, l_2 \text{ s.t. } l_1 < l_2 \]  
\[ (5.46) \]

\[ v_{l_1,j}^S \geq v_{l_2,j}^S, \quad j \in [0 : K - 1], \forall l_1, l_2 \text{ s.t. } l_1 < l_2. \]  
\[ (5.47) \]

The remaining conditions are formed using identical reasoning to the non-uniform cache case, but occur on a file-by-file basis as needed. The file reconstruction constraint remains the same, with the minor change that the file must add up not to the common file length 1, but to \( F_l \), the actual length of the file as expressed below:

\[ F_l = \sum_{j=0}^{K-1} \left( \begin{array}{c} K_L \\ j \end{array} \right) (v_{l,j}^L - v_{l,j}^M) + \left( \begin{array}{c} K_S \\ j \end{array} \right) (v_{l,j}^S - v_{l,j}^M) + \left( \begin{array}{c} K \\ j \end{array} \right) v_{l,j}^M, \forall l \in [N] \]  
\[ (5.48) \]

The cache memory constraints are modified similarly, except instead of giving an equal amount \( M_S/N \) to each file, an amount \( \mu_l^S \) is allocated to file \( l \), yielding

\[ \mu_l^S = \sum_{j=1}^{K} \left( \begin{array}{c} K_S - 1 \\ j - 1 \end{array} \right) (v_{l,j}^S - v_{l,j}^M) + \left( \begin{array}{c} K - 1 \\ j - 1 \end{array} \right) v_{l,j}^M, \forall l \in [N], \]  
\[ (5.49) \]

where it must be the case that

\[ \sum_{l=1}^{N} \mu_l^S = M_S. \]  
\[ (5.50) \]

A similar pair of equations holds for large users:

\[ \mu_l^L = \sum_{j=1}^{K} \left( \begin{array}{c} K_L - 1 \\ j - 1 \end{array} \right) (v_{l,j}^L - v_{l,j}^M) + \left( \begin{array}{c} K - 1 \\ j - 1 \end{array} \right) v_{l,j}^M, \forall l \in [N], \]  
\[ (5.51) \]

and

\[ \sum_{l=1}^{N} \mu_l^L = M_L. \]  
\[ (5.52) \]

The subfiles are also required to be positive in size, as always:

\[ v_{l,j}^L \geq 0, \quad v_{l,j}^S \geq 0, \quad v_{l,j}^M \geq 0, \quad \forall j \in [0 : K], \quad l \in [N] \]  
\[ (5.53) \]

Finally, another set of memory inequality constraints are required to break ties between small-index, small-cache subfiles and large-index, large-cache subfiles. Leaving the justification and discussion of this choice to later sections, we develop a caching scheme under the constraints

\[ v_{l_1,j}^L \geq v_{l_2,j}^M, \quad j \in [2 : K_L], \forall l_1, l_2 \in [N] \]  
\[ (5.54) \]

\[ v_{l_1,j}^M \geq v_{l_2,j}^S, \quad j \in [2 : K_S], \forall l_1, l_2 \in [N] \]  
\[ (5.55) \]

\[ v_{l_1,j}^L \geq v_{l_2,j}^S, \quad j \in [1 : K_L], \forall l_1, l_2 \in [N]. \]  
\[ (5.56) \]
In words, this means that subfiles for any file stored on a larger cache type should be larger than the subfiles of any file stored on a smaller cache type, independent of which files are involved. The following proposition gives the objective function of the simplified optimization problem.

**Proposition 4.** Define the following functions of the integer parameters \( n, m, j \) and \( i \):

\[
\nu_1(n, m, j, i) = \frac{K_S - m}{K - n + 1} \binom{K_S - m - 1}{i} \binom{K_L - n + 1 + m}{j - i},
\]

\[
\nu_2(n, m, j, i) = \frac{K_L - n + 1 + m}{K - n + 1} \binom{K_S - m}{i} \binom{K_L - n + m}{j - i},
\]

and

\[
\nu(n, j) = \sum_{m=0}^{n-1} \left( \binom{K_S}{m} \binom{K_L}{n-1-m} \right) \left( \sum_{i=1}^{j-2} \nu_1(n, m, j, i) + \sum_{i=2}^{j-1} \nu_2(n, m, j, i) \right)
\]

Then for the variables defined in (5.37) satisfying (5.38)-(5.56), the objective function (3.8) simplifies exactly as

\[
E \left[ \sum_{S \in \mathcal{P}(U) \setminus \emptyset} \max_{k \in S} \{ W^{(d_u)}_{S, \{k\}} \} \right] = \sum_{i=0}^{K-1} \sum_{j=1}^{L-1} \sum_{l=1}^{N} \Pr[Y_i = l] v_{ij}^L + \sum_{j=1}^{K-1} \sum_{m=1}^{K_L-1} \sum_{l=1}^{N} \binom{K_L - m - 1}{m} \binom{K_L - n + m}{j - i} \Pr[Y_i^L = l | v_{ij}^L]
\]

\[
+ \sum_{j=2}^{K-1} \sum_{m=1}^{K_L-1} \sum_{l=1}^{N} \binom{K_S - 1 - m}{m} \binom{K_L - m - 1}{j - i} \Pr[Y_i^S = l | v_{ij}^S] + \sum_{j=1}^{K-1} \sum_{m=1}^{K_L-1} \sum_{l=1}^{N} \binom{K_L - m - 1}{m} \binom{K_L - n + m}{j - i} \Pr[Y_i^S = l | v_{ij}^S]
\]

\[
+ \sum_{j=2}^{K-1} \sum_{m=1}^{K_L-1} \sum_{l=1}^{N} \binom{K_S - 1 - m}{m} \binom{K_L - m - 1}{j - i} \Pr[Y_i^S = l | v_{ij}^S] + \sum_{j=3}^{K-1} \sum_{n=1}^{K-1} \sum_{l=1}^{N} \Pr[Y_{n-1} = l | v_{ij}^M]
\]

Proposition 4 is proved in Appendix D. Although visually complicated, (5.57) simplifies the original objective function (3.8) by using only \( 3(K+1)N \) variables and having a number of terms that scales with \( K^2 N \) rather than \( (2N)^K \). The number of constraints described by (5.38)-(5.56) scales with \( K N^2 \), and so the following optimization problem is a tractable method of obtaining a caching scheme that accommodates heterogeneity in cache size, file size, and file popularity:

\[
\text{minimize} \quad (5.57) \quad \text{subject to} \quad (5.38)-(5.56)
\]

5.3.3 Numerical Results

To demonstrate that this simplified optimization problem performs well when compared to the general problem (3.8)-(3.9), we again consider \( K = 4 \) users (\( K_S = 2 \) of which are small-cache users) and \( N = 6 \) files, with all of the non-uniformities used thus far: file labels and popularity/size pairs as given in Table 5.1, and cache sizes of \( M_S = 0.8M \) and \( M_L = 1.2M \) for \( M \in [0 : N] \). Fig. 5.5 compares the simplified and general problems with a naive random caching baseline.

We see that the simplified problem yields a scheme that closes mirrors, although does not match
Figure 5.5: A comparison of the performance of the solution obtained from (3.8)-(3.9) to the solution obtained by (5.58)-(5.59), with reference to a baseline random caching scheme, for the case of non-uniform file size, file popularity, and cache size.
Table 5.4: Optimal subfile sizes and memory allocation for the general problem (3.8)-(3.9) with $K = 4, N = 6, K_S = 2, M_S = 1.6, M_L = 2.4$ and file popularity/size pairs given by Table 5.1, with values rounded to three decimal places.

<table>
<thead>
<tr>
<th>Subset</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>∅</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>{1}</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.165</td>
<td>0.085</td>
</tr>
<tr>
<td>{2}</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.165</td>
<td>0.085</td>
</tr>
<tr>
<td>{3}</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.165</td>
<td>0.085</td>
</tr>
<tr>
<td>{4}</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.178</td>
<td>0.165</td>
<td>0.085</td>
</tr>
<tr>
<td>{1,2}</td>
<td>0</td>
<td>0</td>
<td>0.077</td>
<td>0.008</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>{1,3}</td>
<td>0.090</td>
<td>0.090</td>
<td>0.090</td>
<td>0.008</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>{1,4}</td>
<td>0.090</td>
<td>0.090</td>
<td>0.090</td>
<td>0.008</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>{2,3}</td>
<td>0.090</td>
<td>0.090</td>
<td>0.090</td>
<td>0.008</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>{2,4}</td>
<td>0.090</td>
<td>0.090</td>
<td>0.090</td>
<td>0.008</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>{3,4}</td>
<td>0.431</td>
<td>0.188</td>
<td>0.090</td>
<td>0.090</td>
<td>0.008</td>
<td>0</td>
</tr>
<tr>
<td>{1,2,3}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>{1,2,3,4}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mem. (L):</td>
<td>0.788</td>
<td>0.554</td>
<td>0.446</td>
<td>0.284</td>
<td>0.173</td>
<td>0.165</td>
</tr>
<tr>
<td>Mem. (S):</td>
<td>0.357</td>
<td>0.434</td>
<td>0.365</td>
<td>0.194</td>
<td>0.165</td>
<td>0.085</td>
</tr>
</tbody>
</table>

exactly, the performance of the scheme obtained from the general problem for small and intermediate $M$ values. Table 5.4 shows the optimal solution to the general problem for the $M = 2$ case; we see several violations of the memory inequality constraints that explain why the simplified problem could not achieve as good a performance as the general problem: the “true” optimal solution lies outside of its feasible space. Nevertheless, the simplified problem still achieves good performance compared to the general problem in this regime. For large $M$, we see that the expected rate of the simplified scheme does not drop as quickly as the general problem solution, and is even eclipsed by the random caching scheme. This occurs for the same reason we saw in Section 5.2 in the $K_S = 2$ case (Fig. 5.4). To compare the relative performance of the three schemes in general, Fig. 5.6 shows the percent increase in expected rate if the random caching scheme is used over the general and simplified schemes respectively. The significant increase in rate when using random caching make it clear that designing the cache content can be worthwhile when the engineering context allows for it.

5.3.4 Further Extensions

We first echo the earlier comments about heterogeneous cache sizes: we consider only two different cache sizes here, but it is possible to use the same reasoning to develop a tractable optimization problem for a practical system having more (but not many more) cache sizes.

The primary focus in this section, however, is on the memory inequality constraints (5.54)-(5.56) used in developing the simplified problem (5.58)-(5.59) of this section. Recall that these constraints require, among other things, that (roughly speaking) for a fixed index $j$, the $v^L_{i,j}$ variables for all files $l$ be larger than the $v^S_{i,j}$ variables for all files. Thus the large-user subfile for the smallest file is larger than the small-user subfile for the largest file. This restriction was required to allow us to write the simplified problem, and numerical results show that the simplified problem still performed well compared to the general problem for the considered parameters. While a full investigation of the performance of the simplified
problem across all parameter values would be laborious, it is still possible to estimate the behaviour for certain parameter regimes. We should expect (5.54)-(5.56) to result in a good simplification when the disparity between the large and small cache sizes is big, and when there are small numbers of files, because the large cache users will likely store much larger subfiles than the small cache users, irrespective of the length of the file. Conversely, we should expect the performance of (5.58)-(5.59) to be relatively poor when the cache sizes are comparable and there are large numbers of files. In this case, it may make more sense to use something like the “opposite” memory inequality constraint: a small-user subfile variable $v_{l_1,j}^S$ should be larger than any larger-user subfile variable $v_{l_2,j}^L$ if file $l_1$ is larger than file $l_2$.

While the simplified optimization problem that would result from this constraint is not explored in this thesis, it should be possible to construct such a problem using the same kind of reasoning used here. Indeed, there may also be other memory inequality constraints that prove to yield useful simplified optimization problems for other parameter sets. The appeal of the tractability of these models is that a server, knowing the relevant parameters for its system, could easily compute the performance of these schemes and choose the best among them; any discussion of the specifics of such schemes, however, is left to future work.

Figure 5.6: The percent increase in expected rate due to random cache content when compared to the simplified problem (5.58)-(5.59) and general problem (3.8)-(3.9) optimal solutions.
Chapter 6

Summary, Conclusions, and Future Directions

The two primary goals of this thesis were, one, to advance a certain optimization theoretic approach to coded caching problems, and two, to use that framework to derive both specific caching schemes and general insight for system models containing multiple heterogeneities that have yet to be considered in the literature. An exponentially-scaling optimization problem corresponding to a caching scheme capable of handling non-uniform file size, popularity, and cache size was then developed. It was shown that the original scheme of Maddah-Ali and Niesen in [19, 20] was the optimal solution of that problem for the special case of uniform file length, popularity, and cache size.

Tractable problems were then developed to handle various combinations of heterogeneous system parameters. The consideration of these special cases also permitted the observation of the effects that these non-uniformities have on the optimal cache content. It was also shown when considering non-uniform file popularity and size jointly that while popularity may in general have some influence on optimal cache allocation, file size can be a much stronger influence; indeed, very good performance was obtained in the case considered by ignoring file popularity altogether. Finally, with the insights obtained from the previously-explored special cases, we developed a tractable optimization problem corresponding to a caching scheme capable of accommodating all three of the aforementioned heterogeneities, and showed numerically that it performs well compared to the original exponentially-scaling problem.

Future work should explore and characterize the limits of the performance of the tractable caching schemes presented here. While these schemes were shown to perform well under the specific conditions in this thesis, the set of possible parameter combinations is very large. While, as discussed earlier, the tractability of the problems allows for the possibility that a server could compute the results of several different problems and use the scheme that performs best, it would be preferable to have analytic (or at least strong numerical) results that would facilitate comparison between schemes without solving the optimization problem. Nevertheless, the various insights and caching schemes obtained from the optimization framework advanced here, taken jointly with the results obtained other work using similar optimization methods, i.e. [44] and [34], provide strong evidence that such optimization frameworks will be of great utility in the design of future caching schemes and in the analysis of future caching problems. Other future directions could include the development of information theoretic outer bounds for the various heterogeneous cases considered here, and the inclusion of even more heterogeneous effects, like
non-uniform user download rates.

Perhaps the most interesting possible future direction, however, would be the development of an optimization problem, like the one presented in this thesis, but for coded caching schemes with coded cache content, in addition to the coded transmissions. Such a problem would likely contain uncoded cache content in its feasible space, and so could be claimed to be to optimization problem over all possible cache content. Then, with a parameterized objective function describing, based on the choice of parameters, any possible transmission schemes, solving the optimization problem would be equivalent to determining the optimal cache content for the optimal transmission scheme. Indeed, this would constitute a viable method for determining the fundamental limits of caching. This may be a difficult task to achieve, but the work of this thesis represents a step forward towards that goal.
Appendix A

Proof of Lemmas 3 and 4

A.1 Proof of Lemma 3 (Chu-Vandermonde Convolution)

While many proofs of this classic result can readily be found online, we present a simple double counting proof. Suppose that a set of balls are labelled with the integers from 1 to \( N \), and that the first \( N_1 \) balls are red, while the remaining \( N_2 \) balls are blue; suppose further that we wish to determine the total number of collections of \( n \) balls that are unique up to the ordering of the balls in the collection. Ignoring colour, it is a basic result of combinatorics that there are \( \binom{N}{n} \) such collections. However, this value can also be calculated by noting that a collection of \( n \) balls can be comprised of \( k \) red balls and \( n-k \) blue balls. The total number of collections of size \( n \) containing \( k \) red and \( n-k \) blue balls is \( n_k = \binom{N_1}{k} \binom{N_2}{n-k} \), so to obtain the total number of collections of size \( n \), the \( n_k \) values must be added together for all possible values of \( k \).

The upper limit on \( k \) is \( \min\{N_1, n\} \) - there cannot be more than \( N_1 \) red balls in a group because there are only \( N_1 \) red balls, and there cannot be more than \( n \) red balls in a group because there are only \( n \) balls in the group. On the other hand, the lower limit on \( k \) is \( \max\{n - N_2, 0\} \) - if \( k \) is less than \( n - N_2 \), then this would require more than \( N_2 \) blue balls, and trivially, there cannot be fewer than 0 red balls in a group.

Thus it must be the case that

\[
\binom{N}{n} = \sum_{k=\max\{n-N_2,0\}}^{\min\{N_1,n\}} \binom{N_1}{k} \binom{N_2}{n-k} \quad (A.1)
\]

The desired formula is obtained when it is noted that if \( N_1 < n \), then \( \binom{N_1}{k} = 0 \) for all \( k > N_1 \), so the upper limit can simply be set to \( n \) as doing so will only add 0 to the sum. Similarly, if \( n - N_2 > 0 \), then \( \binom{N_2}{n-k} = 0 \) for all \( k < n - N_2 \), and so the lower limit can be relaxed to 0 without adding any value to the sum. We therefore obtain Eq. (5.1), as desired.

A.2 Proof of Lemma 4

We begin with the proof of the expression of \( \Pr[Y_0 = i] \), for which we use induction on \( i \) for \( i = 1, \ldots, N \). We begin first with the \( i = 1 \) case. Let \( Z \in [N]^K \) denote the sequence of outcomes from the \( N \) trials, e.g.
If for \( K = 3 \) and \( N = 4 \), trial 1 obtains outcome 2, trial 2 obtains outcome 4, and trial 3 obtains outcome 1, we have \( Z = [2, 4, 1]^T \). Let \( X_n \) denote the random variable representing the number of times the outcome \( n \) occurs in the \( K \) trials (i.e. the number of times it appears in \( Z \)), and stack the \( X_n \) variables in a vector \( X \in [K]^N \); the example above would yield \( X = [1, 1, 0, 1]^T \). Then the smallest element of \( Z \) is 1 (i.e. \( Y_0 = 1 \)) if and only if \( X_1 \geq 1 \); in other words, since 1 is the smallest possible outcome, if it occurs anywhere in \( Z \) then it is the smallest element. Thus we have

\[
\Pr[Y_0 = 1] = \Pr[X_1 \geq 1] = 1 - \Pr[X_1 = 0] = 1 - (1 - p_1)^K = \left( \frac{\sum_{l=1}^{N} p_l}{\sum_{l=2}^{N} p_l} \right)^K - \left( \frac{\sum_{l=2}^{N} p_l}{\sum_{l=2}^{N} p_l} \right)^K,
\]

which is indeed the formula (5.2) with \( i = 1 \), as desired.

For an arbitrary \( i \) such that \( 2 \leq i \leq N - 1 \), we note that if the smallest element of \( Z \) is \( i \), (i.e. \( Y_0 = i \)), then there cannot be any values smaller than \( i \), and there must be at least one \( i \) in \( Z \); in other words:

\[
\Pr[Y_0 = i] = \Pr[X_i \geq 1, X_{i-1} = \ldots = X_1 = 0] = \Pr[X_i = 0] \Pr[X_{i-1} = 0, \ldots, X_1 = 0] \Pr[X_i \geq 1 | X_{i-1} = \ldots = X_1 = 0] \]

\[
= \Pr[X_i = 0] \Pr[X_{i-1} = 0, X_{i-2} = \ldots, X_1 = 0 | X_i \geq 1] \Pr[X_1 = 0 | X_{i-1} = \ldots = X_1 = 0] \]

\[
= \left( 1 - \Pr[X_i = 0 | X_{i-1} = \ldots = X_1 = 0] \right)^K.
\]

Now, comparing the expression (A.2) for \( Y_0 = i \) to the same expression with \( Y_0 = i - 1 \), it is easy to show that

\[
\Pr[Y_0 = i] = \Pr[Y_0 = i - 1] \left( \frac{\Pr[X_{i-1} = 0 | X_{i-2} = \ldots = X_1 = 0]}{1 - \Pr[X_{i-1} = 0 | X_{i-2} = \ldots = X_1 = 0]} \right)^K \]

\[
\left( 1 - \Pr[X_i = 0 | X_{i-1} = \ldots = X_1 = 0] \right)^K.
\]

It is possible to compute these conditional probabilities directly:

\[
\Pr[X_j = 0 | X_{j-1} = \ldots = X_1 = 0] = \left( 1 - p_j^{(j)} \right)^K,
\]

where

\[
p_j^{(j)} = \frac{p_j}{\sum_{l=j}^{N} p_l}
\]

is the probability of outcome \( j \) occurring in a trial conditioned on the knowledge that outcomes 1 through \( j - 1 \) have not occurred in that trial. From the definition of \( p_j^{(j)} \), we can rewrite Eq. (A.4) as

\[
\Pr[X_j = 0 | X_{j-1} = \ldots = X_1 = 0] = \left( \frac{\sum_{l=j+1}^{N} p_l}{\sum_{l=j}^{N} p_l} \right)^K.
\]

Evaluating this expression for \( j = i \) and \( j = i - 1 \), we can obtain from Eq. (A.3), after some mild algebraic manipulation,

\[
\Pr[Y_0 = i] = \Pr[Y_0 = i - 1] \left( \frac{\sum_{l=i+1}^{N} p_l}{\sum_{l=i}^{N} p_l} \right)^K \left( \frac{\sum_{l=i-1}^{N} p_l}{\sum_{l=i}^{N} p_l} \right)^K = \left( \frac{\sum_{l=i+1}^{N} p_l}{\sum_{l=i}^{N} p_l} \right)^K - \left( \frac{\sum_{l=i-1}^{N} p_l}{\sum_{l=i}^{N} p_l} \right)^K
\]
Appendix A. Proof of Lemmas 3 and 4

\[ \Pr[Y_0 = i - 1] = \frac{\left( \sum_{l=i}^{N} p_l \right)^K}{\left( \sum_{l=i-1}^{N} p_l \right)^K} - \frac{\left( \sum_{l=i+1}^{N} p_l \right)^K}{\left( \sum_{l=i}^{N} p_l \right)^K}. \]  

(A.5)

Now by the inductive hypothesis, \( \Pr[Y_0 = i - 1] \) is precisely equal to the denominator of Eq. (A.5), and so we obtain

\[ \Pr[Y_0 = i] = \left( \sum_{l=i}^{N} p_l \right)^K - \left( \sum_{l=i+1}^{N} p_l \right)^K, \]  

(A.6)

as desired.

Although the \( \Pr[Y_0 = N] \) formula was covered in the preceding paragraph, we discuss it in further detail here because (5.2) above may not appear sensible in the \( i = N \) case. Note that if \( N \) is the smallest value in the multinomial vector \( Z \), then it must be the case that every element of \( Z \) is equal to \( N \), otherwise some element not equal to \( N \) would be the smallest value. Thus we have

\[ \Pr[Y_0 = N] = \Pr[X_N = K, X_{N-1} = 0, \ldots, X_1 = 0] \]

\[ = \frac{p_N^K K}{p_N^K K} - 0^K \]

\[ = \left( \sum_{l=N}^{N} p_l \right)^K - \left( \sum_{l=N+1}^{N} p_l \right)^K, \]

which is the formula (5.2) with \( i = N \). Note that we use the definition that \( \sum_{l=a}^{b} n_l = 0 \) when \( a > b \).

Next, we proceed to the \( m = 1 \) case. Here we will directly compute \( \Pr[Y_1 = i] \) by first deriving \( \Pr[Y_1 = i, Y_0 = j] \), and then obtaining the desired quantity from the sum

\[ \Pr[Y_1 = i] = \sum_{j=1}^{N} \Pr[Y_1 = i, Y_0 = j] \]  

(A.7)

There are three cases to consider: \( i < j, i > j \), and \( i = j \). If \( i < j \), then the second smallest element of \( Z \) is smaller than the smallest element of \( Z \), and so the probability of this occurring is 0. If \( i > j \), we write

\[ \Pr[Y_1 = i, Y_0 = j] \]

\[ = \Pr[X_i \geq 1, X_{i-1} = 0, \ldots, X_j = 1, X_{j-1} = \cdots = X_1 = 0] \]

\[ = \Pr[X_i = 0, \ldots, X_j = 1, X_{j-1} = \cdots = X_1 = 0] \Pr[X_i = 0, X_{i-1} = 0, \ldots, X_j = 1, X_{j-1} = \cdots = X_1 = 0] \]  

(A.8)

To compute the difference (A.8), note that we can form a \((K, 4)\) sequential vector of outcomes \( \tilde{Z}(m, n) \), indexed by two integers \( m \) and \( n \), from the original \((K, N)\) sequential vector of outcomes \( Z \) in the following way: for a single trial, the first outcome of \( \tilde{Z}(m, n) \) occurs if any of the first \( n-1 \) outcomes of \( Z \) occur, and so it has the probability \( \tilde{p}_1 = \sum_{l=1}^{n-1} p_l \); the second outcome of \( \tilde{Z}(m, n) \) occurs if the \( n \)-th outcome of \( Z \) occurs, and so it has a probability of \( \tilde{p}_2 = p_n \); the third outcome of \( \tilde{Z}(m, n) \)
occurs if any outcomes of $Z$ from $n + 1$ to $m$ occur, and so it has a probability of $\tilde{p}_3 = \sum_{l=n+1}^{m} p_l$; and the fourth outcome of $\tilde{Z}(m, n)$ occurs if any of the last $N - m$ outcomes of $Z$ occur, and so it has a probability $\tilde{p}_4 = \sum_{l=m+1}^{N} p_l$. If we define $\tilde{X}_j$ as the number of times outcome $j$ occurred in $\tilde{Z}(m, n)$, then we consequently have $\tilde{X}_1 = \sum_{l=1}^{n-1} X_l$, $\tilde{X}_2 = X_n$, $\tilde{X}_3 = \sum_{l=n+1}^{m} X_l$, and $\tilde{X}_4 = \sum_{l=m+1}^{N} X_l$. We can then rewrite\(^1\) (A.8) using vectors $\tilde{X}(i, j)$ and $\tilde{X}(i - 1, j)$ as

\[
\Pr[Y_1 = i, Y_0 = j] = \Pr[\tilde{X}_4 = K - 1, \tilde{X}_3 = 0, \tilde{X}_2 = 1, \tilde{X}_1 = 0] - \Pr[\tilde{X}_4 = K - 1, \tilde{X}_3 = 0, \tilde{X}_2 = 1, \tilde{X}_1 = 0]
\]

where the first term is computed with respect to $\tilde{X}(i - 1, j)$, and the second with respect to $\tilde{X}(i, j)$. Using the probabilities defined earlier, this gives

\[
\Pr[Y_1 = i, Y_0 = j] = K\tilde{p}_2(\tilde{p}_4)^{K-1} - K\tilde{p}_2(\tilde{p}_4)^{K-1} - KP_j \left( \left( \sum_{l=i}^{N} p_l \right)^{K-1} - \left( \sum_{l=i+1}^{N} p_l \right)^{K-1} \right)
\]

(A.9) (A.10)

where, again, the terms in Eq. (A.9) are computed with respect to $\tilde{X}(i - 1, j)$, and $\tilde{X}(i, j)$ respectively.

Similar reasoning yields the value of the joint probability when $i = j$:

\[
\Pr[Y_1 = i, Y_2 = i] = \Pr[X_i \geq 2, X_{i-1} = \cdots = X_1 = 0] = \Pr[X_{i-1} = \cdots = X_1 = 0] - \Pr[X_i = 0, X_{i-1} = \cdots = X_1 = 0] - \Pr[X_i = 1, X_{i-1} = \cdots = X_1 = 0] - \Pr[\tilde{X}_4 = K, \tilde{X}_3 = \tilde{X}_2 = \tilde{X}_1 = 0] - \Pr[\tilde{X}_4 = K - 1, \tilde{X}_3 = 1, \tilde{X}_2 = \tilde{X}_1 = 0].
\]

(A.11)

Here, the first term in (A.11) is computed with respect to $\tilde{X}(i - 1, i - 2)$, the second with respect to $\tilde{X}(i, i - 1)$, and the third with respect to $\tilde{X}(i, i - 1)$, although this choice of $\tilde{X}$ variables is not unique. This gives

\[
\Pr[Y_1 = i, Y_2 = i] = (\tilde{p}_4)^K - (\tilde{p}_4)^K - K\tilde{p}_2(\tilde{p}_4)^{K-1} = \left( \sum_{l=i}^{N} p_l \right)^K - \left( \sum_{l=i+1}^{N} p_l \right)^K - KP_j \left( \sum_{l=i+1}^{N} p_l \right)^{K-1}
\]

(A.12)

We can now evaluate Eq. (A.7) as

\[
\Pr[Y_1 = i] = \sum_{j=1}^{N} \Pr[Y_1 = i, Y_0 = j]
\]

\(^1\)Note that the $\tilde{X}$ variables lose the $(m, n)$ indices of the original variable $\tilde{X}(m, n)$. This is done for notational convenience, but will result in an abuse of the notation when multiple $\tilde{X}(m, n)$ are involved. We will therefore be careful to indicate which $\tilde{X}$ variables belong to which $\tilde{X}(m, n)$ vectors.
Finally, we must compute \( \Pr[Y_m = i] \) for \( m = 2, \ldots, K - 1 \). We take an approach similar to the \( \Pr[Y_1 = i] \) case, and derive \( \Pr[Y_m = i] \) using the joint probabilities \( \Pr[Y_m = i, Y_{m-1} = j] \). As before, there are three cases, \( i < j, i > j, \) and \( i = j \), and once again, the \( i < j \) is trivial: the \( m \)-th smallest element of \( Z \) cannot be smaller than the \((m-1)\)-th element of \( Z \), and so \( \Pr[Y_m = i, Y_{m-1} = j] = 0 \) if \( i < j \). If \( i > j \), we have

\[
\Pr[Y_m = i, Y_{m-1} = j] = \sum_{k=1}^{m} \Pr[X_i \geq 1, X_{i-1} = \cdots = X_{j+1} = 0, X_j = k, X_{j-1} + \cdots + X_1 = m - k]
\]

\[
= \sum_{k=1}^{m} \Pr[X_{i-1} = \cdots = X_{j+1} = 0, X_j = k, X_{j-1} + \cdots + X_1 = 0] - \Pr[X_i = \cdots = X_{j+1} = 0, X_j = k, X_{j-1} + \cdots + X_1 = m - k] - \Pr[X_{i-1} = \cdots = X_{j+1} = 0, X_j + \cdots + X_1 = m - 1] + \Pr[X_i = \cdots = X_{j+1} = 0, X_j + \cdots + X_1 = m] - \Pr[X_{i-1} = \cdots = X_{j+1} = 0, X_j + \cdots + X_1 = m] + \Pr[X_i = \cdots = X_{j+1} = 0, X_j = 0, X_{j-1} + \cdots + X_1 = m].
\]

(A.14)

Now we recast the four terms of Eq. (A.14) in terms of \( \bar{X}(i - 1, j + 1) \), \( \bar{X}(i - 1, j) \), \( \bar{X}(i, j + 1) \), and \( \bar{X}(i, j) \) respectively:

\[
\Pr[Y_m = i, Y_{m-1} = j] = \left( \Pr[\bar{X}_1 = K - m, \bar{X}_3 = 0, \bar{X}_2 = 0, \bar{X}_1 = m] - \Pr[\bar{X}_1 = K - m, \bar{X}_3 = 0, \bar{X}_2 = 0, \bar{X}_1 = m] \right)
\]

\[
+ \left( \Pr[\bar{X}_1 = K - m, \bar{X}_3 = 0, \bar{X}_2 = 0, \bar{X}_1 = m] - \Pr[\bar{X}_1 = K - m, \bar{X}_3 = 0, \bar{X}_2 = 0, \bar{X}_1 = m] \right).
\]

(A.14)
In the case where \( i = j \), there are two sub-cases to consider: \( i = j \neq 1 \) and \( i = j = 1 \). In the former sub-case, we must have \( X_i \geq 2 \), and \( X_{i-1} + \cdots + X_1 = b \leq m - 1 \). Suppose that \( X_i = 2 + k \) for some integer \( k \in \{0, \ldots, K - 2\} \). We know that \( Y_m = i \) and \( Y_{m-1} = i \), but that leaves \( k \) \( Y \) variables “adjacent” to \( Y_m \) and \( Y_{m-1} \) that must also have a value of \( i \). Let \( n_i \) denote the number of variables \( Y_{m'} \) that are equal to \( i \) and have \( m' > m \), and \( n_s \) denote the number of variables \( Y_{m'} \) that are equal to \( i \) and have \( m' < m - 1 \). Then \( n_i + n_s \) is the smallest values of \( \max \{m - 1, K - k - 2\} \).

The lower limit on \( b \) is obtained through similar reasoning, but we first note that the trivial lower limit on \( b \) is 0, which occurs when the number \( i \) constitutes (at least) the first \( m \) smallest values of \( Z \); in this case \( k \geq m - 1 \). If \( k < m - 1 \), then not all \( Y_{m'} \) with \( m' < m \) can have values of \( i \). In general, \( k + 2 \geq m + 1 \), which implies that \( b \geq m - 1 - k \). Then general lower bound on \( b \) is therefore \( b \geq \max \{0, m - 1 - k\} \).

We are therefore now in a position to write

\[
\Pr[Y_m = i, Y_{m-1} = i \neq 1] = \sum_{k=0}^{K-2 \min\{m-1, K-2-k\}} \sum_{b=\max\{0,m-1-k\}}^{K-2 \min\{m-1, K-2-k\}} \Pr[X_i = 2 + k, X_{i-1} + \cdots + X_1 = b] \\
= \sum_{k=0}^{K-2 \min\{m-1, K-2-k\}} \sum_{b=\max\{0,m-1-k\}}^{K-2 \min\{m-1, K-2-k\}} \Pr[\tilde{X}_4 = K - k - 2 - b, \tilde{X}_3 = 0, \tilde{X}_2 = 2 + k, \tilde{X}_1 = b] \tag{A.16}
\]

where the \( \tilde{X} \) variables in Eq. (A.16) are with reference to \( \tilde{X}(i, i) \). This can be computed as

\[
\Pr[\tilde{X}_4 = K - k - 2 - b, \tilde{X}_3 = 0, \tilde{X}_2 = 2 + k, \tilde{X}_1 = b] = \left( \frac{K}{K - k - 2 - b, b, 2 + k} \right) (\tilde{p}_4)^{K-k-2} (\tilde{p}_2)^{2+k} (\tilde{p}_1)^{b}
\]
Appendix A. Proof of Lemmas 3 and 4

\[
= \left( \frac{K}{K - k - 2 - b} \right) \left( \sum_{l=i+1}^{N} p_l \right)^{K-k-2-b} \left( p_i \right)^{2+k} \left( \sum_{l=1}^{i-1} p_l \right)^{b} \quad (A.17)
\]

When \( i = j = 1 \), we simply have

\[
\Pr[Y_m = 1, Y_{m-1} = 1] = \Pr[X_1 \geq m + 1] = \sum_{k=0}^{K-1-m} \Pr[X_1 = j + 1 + k] = \sum_{k=0}^{K-1-m} \left( \frac{K}{m + 1 + k} \right) (p_1)^{m+1+k} (1 - p_1)^{K-m-1-k} \quad (A.18)
\]

Finally, we compute \( \Pr[Y_m = i] \) as

\[
\Pr[Y_m = i] = \sum_{j=1}^{N} \Pr[Y_m = i, Y_{m-1} = j]
\]

\[
= \sum_{j=1}^{i-1} \Pr[Y - m = i, Y - m - 1 = j] + \Pr[Y_m = i, Y_{m-1} = i]
\]

\[
= \left( \frac{K}{K - m} \right) \left( \left( \sum_{l=1}^{N} p_l \right)^{K-m} - \left( \sum_{l=i}^{N} p_l \right)^{K-m} \right) \left( \sum_{j=1}^{i-1} \left( \sum_{l=1}^{j} p_l \right)^{m} - \left( \sum_{l=1}^{i-1} p_l \right)^{m} \right) + \Pr[Y_m = i, Y_{m-1} = i]
\]

\[
= \left( \frac{K}{K - m} \right) \left( \left( \sum_{l=i}^{N} p_l \right)^{K-m} - \left( \sum_{l=i+1}^{N} p_l \right)^{K-m} \right) \left( \sum_{l=1}^{i-1} p_l \right)^{m} + \Pr[Y_m = i, Y_{m-1} = i] \quad (A.19)
\]

Combining (A.16)-(A.19) yields the desired result. This completes the proof.
Appendix B

Proof of Proposition 1

We wish to show that

\[
E \left[ \sum_{S \in \mathcal{P}(U) \backslash \emptyset} \max_{k \in S} \{|W_{(d_k)}^{(k)}|\} \right] = \sum_{j=0}^{K-1} \sum_{i=0}^{N} \binom{K-1-i}{j} \Pr[Y_i = l] v_{i,j} + \sum_{i=0}^{K-1} \sum_{l=1}^{N} \Pr[\tilde{Y}_i = l] v_{i,0} \quad (B.1)
\]

if the memory inequality condition holds for the \(v_{i,j}\) variables. We begin with an examination of the left hand side of the equation. Inside the expectation, we sum over all subsets \(S\) of the set of users \(U\). This can be rewritten as a double summation: in the inner summation, we sum over all subsets of size \(j + 1\), and in the outer summation, we sum over all \(j\) from 0 to \(K - 1\), giving

\[
\sum_{S \in \mathcal{P}(U) \backslash \emptyset} \max_{k \in S} \{|W_{(d_k)}^{(k)}|\} = \sum_{j=0}^{K-1} \sum_{S \in \mathcal{P}(U) \backslash \emptyset: |S| = j+1} \max_{k \in S} \{|W_{(d_k)}^{(k)}|\} \quad (B.2)
\]

Replacing the \(|W_S|\) variables with the appropriate \(v_{i,j}\) variables, Eq. (B.2) becomes

\[
\sum_{j=0}^{K-1} \sum_{S \in \mathcal{P}(U) \backslash \emptyset: |S| = j+1} \max_{k \in S} \{v_{d_k,j}\} = \sum_{j=0}^{K-1} \sum_{S \in \mathcal{P}(U) \backslash \emptyset: |S| = j+1} \max_{k \in S} \{|W_{(d_k)}^{(k)}|\} \quad (B.3)
\]

For a fixed \(j \geq 1\), we note that we send one transmission to each of the \(\binom{K}{j+1}\) subsets of size \(j + 1\). For a fixed \(d\), let \(k_i\) denote the user requesting the \(i\)-th most popular file, i.e. the file \(i\)-th smallest index. Then \(k_1\) has requested the most popular file, and so by the memory inequality (5.11), \(v_{d_{k_1},j}\) is the largest variable for any transmission to a subset of which \(k_1\) is a member. Since \(k_1\) is a member of \(\binom{K-1}{j}\) subsets of size \(j + 1\) that contain \(k_1\) as a member, the inner summation of (B.3) will have \(\binom{K-1}{j}\) terms with the value \(v_{d_{k_1},j}\). Similarly, user \(k_2\) has requested the second most popular file, and so \(v_{d_{k_2},j}\) will be the largest subfile for all subsets that contain \(k_2\) but don’t contain \(k_1\). This constitutes \(\binom{K-1}{j-1}\) subsets of size \(j + 1\).

This reasoning can be extended until all subsets are characterized in terms of their maximum \(v_{i,j}\) variable. User \(k_i\) requests the \(i\)-th most popular file, and so \(v_{d_{k_i},j}\) will be the biggest element sent in any subset containing \(k_i\) but not containing \(k_1, k_2, \ldots, k_{i-1}\). Since there are \(K - i\) users who are not users \(k_1, \ldots, k_i\), and user \(k_i\) is already in the subset, there are \(\binom{K-1}{j-i}\) subsets that contain \(k_i\) but not \(k_1, k_2, \ldots, k_{i-1}\). We can therefore eliminate the \(\max\{\}\) term from the inner sum of Eq. (B.3) to obtain,
for $j = 1, \ldots, K - 1,$

$$
\sum_{i=1}^{K} \binom{K - i}{j} v_{d_{k_i},j}.
$$

(B.4)

As noted earlier, the memory inequality reverses for $j = 0$, so the least popular files take up the most memory in that case; the reasoning is the same as in the above, but we instead obtain

$$
\sum_{i=1}^{K} \binom{K - i}{0} v_{d_{k_{K+1-i}},0} = \sum_{i=1}^{K} v_{d_{k_{K+1-i}},0}.
$$

(B.5)

All that remains is to compute the expectation of these terms with respect to the demand vectors. Using the linearity of expectation and the results of Eqs. (B.2)-(B.5), the left hand side of (B.1) reduces to

$$
\sum_{j=1}^{K-1} \sum_{i=1}^{K} \binom{K - i}{j} E[v_{d_{k_i},j}] + \sum_{i=1}^{K} E[v_{d_{k_{K+1-i}},0}]
$$

(B.6)

To compute the expected value of the $v_{d_{k_i},j}$ variables ($j = 1, \ldots, K - 1$), we note that it has $N$ possible values, $v_{1,j}, v_{2,j}, \ldots, v_{N,j}$, and the probability of each outcome can be obtained from Lemma 4 in the following way. We have $v_{d_{k_i},j} = v_{l,j}$ if $l$ is the $i$-th most popular file in the request vector $d$; since the files are labelled in terms of decreasing order of popularity, the $i$-th most popular file requested is represented by the $i$-th smallest index in $d$. Thus the probability that $d_{k_i} = l$ is equivalent to the probability that $l$ is the $i$-th smallest index in $d$, and so by Lemma 4, we have

$$
E[v_{d_{k_i},j}] = \sum_{l=1}^{N} \Pr[d_{k_i} = l] v_{l,j} = \sum_{l=1}^{N} \Pr[Y_{i-1} = l] v_{l,j}.
$$

(B.7)

For $j = 0$, the size ordering is reversed, so we are concerned with the largest indices of $d$. However, Lemma (4) can also be used to compute this distribution: to compute the probability distribution of the $i$-th largest element of $d$, form an auxiliary variable $\tilde{d}$ identical to $d$ but with the labels in reverse order. Then the probability distribution of the $i$-th smallest element of $\tilde{d}$, which can be computed with Lemma 4, is equivalent to the probability distribution of the $i$-th largest element of $d$, which we denote by $\Pr[\tilde{Y}_{i+1} = l]$. We can then write

$$
E[v_{d_{k_{K+1-i}},0}] = \sum_{l=1}^{N} \Pr[d_{k_{K+1-i}} = l] v_{l,0} = \sum_{l=1}^{N} \Pr[\tilde{Y}_{i-1} = l] v_{l,0}.
$$

(B.8)

Combing Eqs. (B.6) - (B.8) yields

$$
\sum_{j=1}^{K-1} \sum_{i=1}^{K} \binom{K - i}{j} \sum_{l=1}^{N} \Pr[Y_{i-1} = l] v_{l,j} + \sum_{i=1}^{K} \sum_{l=1}^{N} \Pr[\tilde{Y}_{i-1} = l] v_{l,0}
$$

(B.9)

We complete the proof through a cosmetic change of variables $i' = i - 1$ to obtain the desired expression on the right-hand side of (B.1).
Appendix C

Proof of Proposition 3

We follow reasoning similar to what we have already seen in Section 5.2.2, namely, we divide the various subsets into subsets containing only small-cache users, subsets containing only large-cache users, and subsets containing both large- and small-cache users.

But first, we note that since the files are all the same size and length, the transmission length will be independent of the request vector $d$, and so we have

$$E \left[ \sum_{S \in \mathcal{P}(U) \setminus \emptyset} \max_{k \in S} \{|W^{(d_k)}_{S \setminus \{k\}}|\} \right] = \sum_{S \in \mathcal{P}(U) \setminus \emptyset} \max_{k \in S} \{|W^{(d_k)}_{S \setminus \{k\}}|\}.$$

As discussed above, the sum in the above expression is over all $S \in \mathcal{P}(U) \setminus \emptyset$, which we can separate into small, large, and mixed sets. For a fixed subset size of $j + 1$, there are $\binom{K_S}{j+1}$ sets of small users, $\binom{K_L}{j+1}$ sets of large users, and

$$\sum_{i=1}^{j} \binom{K_S}{i} \binom{K_L}{j+1-i} \quad \text{(C.1)}$$

groups of at least one small user and at least one large user. For a set of $j + 1$ small users, every subfile in a single coded transmission is cached by $j$ small users, and so has the size $v_{j,S}$. Similarly, for any set of $j + 1$ large users, the transmission has the size $v_{j,L}$.

For the mixed subset case, we must consider three cases. First, when there are at least 2 small users and 2 large users in the subset of $j + 1$ users, then since every subfile sent is cached on $j$ of the $j + 1$ users, there must be at least 1 small user and 1 large user among those $j$ users, and so every subfile must be of size $v_{j,M}$. However, if there is only one small user in the subset of $j + 1$ users, then the subfile requested by the small user will have been stored on the caches of $j$ large users, and so will have size $v_{j,L}$. The length of the entire transmission will therefore also be of size $v_{j,L}$. The third case occurs when there is only one large user in the subset of $j + 1$ users. Then the subfile requested by the large user will be stored on the caches of $j$ small user and so will be of size $v_{j,S}$, while every other subfile is cached on a mixed set of $j$ users and so will be of size $v_{j,M}$; the entire transmission will therefore be of length $v_{j,M}$. ¹

So, in addition to the $\binom{K_t}{j+1}$ transmissions of size $v_{j,L}$ sent for groups entirely consisting of entirely

¹In the case where a subset of size 2 contains one large user and one small user, obviously the entire transmission is of length $v_{1,L}$.
large users, there are \( \binom{K_S}{i} \binom{K_L}{j-i} \) transmissions of the same size for those mixed subsets with only one small user. The total number of transmissions of size \( v_{j,M} \) can then be simplified using Lemma 3 as

\[
\sum_{i=2}^{j} \binom{K_S}{i} \binom{K_L}{j+1-i} \\
= \sum_{i=0}^{j+1} \binom{K_S}{i} \binom{K_L}{j+1-i} - \binom{K_S}{j+1} - \binom{K_S}{j} - \binom{K_L}{j+1} \\
= \binom{K}{j+1} - \binom{K_S}{j+1} - \binom{K_S}{1} \binom{K_L}{j} - \binom{K_L}{j+1} \tag{C.2}
\]

Altogether, this gives

\[
\sum_{j=0}^{K-1} \binom{K_S}{j+1} (v_{j,S} - v_{j,M}) + \binom{K}{j+1} v_{j,M} + \left( \binom{K_L}{j+1} + \binom{K_S}{1} \binom{K_L}{j} \right) (v_{j,L} - v_{j,M}),
\]

which is what we aimed to show. We make a special note that the formula is indeed sensible for \( j = 0 \): the \( j = 0 \) term reduces to \( \binom{K}{1} v_{0,M} = K v_0 \), as needed for the individual transmissions to the \( K \) users.
Appendix D

Proof of Proposition 4

We derive the terms of Eq. (5.57) in the order that they appear. In general, we do this using the following steps. First, we identify a certain group of subsets that have similar user composition; then for that group, we determine the number of transmissions that the largest subfile will be in, the number of transmissions that the second largest subfile will be in, and so on. Finally, we compute the expected size of the maximum subfile, the second largest subfile, and so on. This approach will be familiar from previous proofs, but we nevertheless repeat it here due to the complexity of Eq. (5.57).

The groups of subsets that the seven terms of Eq. (5.57) correspond to are, in order: subsets of size one, subsets of size greater than one containing only large users, subsets of size greater than one containing only small users, mixed subsets containing more than one user but only one small user, mixed subsets containing only one large user but more than one small user, subsets containing greater than or equal to \( \max\{K_S, K_L\} + 2 \) users, and subsets containing at least two small and two large users that are less than \( \max\{K_S, K_L\} + 2 \) users. We label these sets of subsets \( S_1, \ldots, S_7 \) respectively. The following lemma shows these sets form a partition (in the loose sense of the word discussed earlier) of \( \mathcal{P}(U) \setminus \emptyset \), and so the sum over all \( S \in \mathcal{P}(U) \setminus \emptyset \) at the beginning of Eq. (5.57) can equivalently be done over all subsets in \( S_1 \), then all subsets in \( S_2 \), and so on, so that all subsets of users in \( \mathcal{P}(U) \setminus \emptyset \) will have been accounted for precisely once.

**Lemma 5.** For the sets \( S_1, \ldots, S_7 \) described above,

\[
\mathcal{P}(U) \setminus \emptyset = \bigcup_{i=1}^{7} S_i, \tag{D.1}
\]

and the \( S_i \) are mutually disjoint.

**Proof:** That \( \bigcup_{i=1}^{7} S_i \subseteq \mathcal{P}(U) \setminus \emptyset \) is trivial: for any \( i \), any set in \( S_i \) is a non-empty subset of users, and so must be contained in \( \mathcal{P}(U) \setminus \emptyset \). To show that \( \mathcal{P}(U) \setminus \emptyset \subseteq \bigcup_{i=1}^{7} S_i \), consider the number of users in an arbitrary subset of users \( S \in \mathcal{P}(U) \setminus \emptyset \): if it is one, then \( S \subseteq S_1 \) and if it is greater than or equal to \( \max\{K_S, K_L\} + 2 \), then it must be mixed because there are not enough of any one type of user to comprise the entire group, and so is must be that \( S \subseteq S_6 \). Otherwise, suppose \( 1 < |S| < \max\{K_S, K_L\} + 1 \), consider the number of small users, \( k_S \), in \( S \). If \( k_S = 0 \), then \( S \) contains only large users, and so \( S \subseteq S_2 \). If \( k_S = 1 \), then we have \( S \subseteq S_4 \). If \( 1 < k_S < |S| \), then either the number of large users is either one, or more than one; if it is one, then \( S \subseteq S_5 \), while if it is more than one, then \( S \subseteq S_7 \).
Finally, if \( k_S = |S| \), there are only small users, and so \( S \subseteq S_3 \), proving that indeed \( \mathcal{P}(U) \setminus \emptyset \subseteq \bigcup_{i=1}^{7} S_i \).

The mutual disjointedness is obvious once it is noted that a subset containing only one large/small user or no large/small users cannot exceed a size of \( \max\{K_S, K_L\} \) or \( \max\{K_S, K_L\} + 1 \) respectively. Each \( S \subseteq \mathcal{P}(U) \setminus \emptyset \) thus falls into one and only one set \( S_i \), proving the lemma.

We remark before continuing that, given the specific values of \( K_S, K_L \), some of the above subsets may be empty. As per the notation adopted in this thesis, a sum over an empty set is identically zero, and so this will not affect our subsequent calculations. In terms of the expressions below, this will correspond to binomial coefficients \( \binom{n}{k} \) with \( n < 0 \) or \( k > n \), both of which, by our notation, gives \( \binom{n}{k} = 0 \).

So per the above discussion, we can change the summation over all subsets of \( \mathcal{P}(U) \setminus \emptyset \) into seven summations over one of the \( S_i \) each:

\[
E\left[ \sum_{S \in \mathcal{P}(U) \setminus \emptyset} \max_{k \in S}\{|W_{S\setminus\{k\}}^{(d_k)}|\} \right] = \sum_{i=1}^{7} \sum_{S \in S_i} E\left[ \max_{k \in S}\{|W_{S\setminus\{k\}}^{(d_k)}|\} \right]
\]

This allows us to analyze each subset of subsets separately.

We begin with the analysis of \( S_1 \), i.e. to broadcasts of individual users. Since each transmission is to only one person, we get

\[
\sum_{S \in S_1} E\left[ \max_{k \in S}\{|W_{\emptyset}^{(d_k)}|\} \right] = \sum_{k=1}^{K} E\left[ |W_{\emptyset}^{(d_k)}|\right] = \sum_{k=1}^{K} E\left[ v_{d_k,0} \right]
\]

The above sum is over all users from \( k = 1 \) to \( k = K \), i.e. in lexicographic order. But we can instead sum over all users by adding the user requesting the largest subfile, then the user requesting the second largest subfile, and so on. Using the index \( i \) to indicate the user requesting the \((i+1)\)-th biggest subfile, we can write the expectation \( E[v_{d_k,0}] \) in terms of the random variable \( Y_i \) as defined in Lemma 4 to obtain

\[
\sum_{k=1}^{K} E\left[ v_{d_k,0} \right] = \sum_{i=0}^{K-1} E\left[ v_{f_d(i+1),0} \right] = \sum_{i=0}^{K-1} \sum_{l=1}^{N} \Pr[Y_i = l]v_{l,0}.
\]

which is the first term of Eq. (5.57), with \( v_{l,0} = v_{l,0}^{(M)} \) as per constraint (5.41). Here, \( f_d(i) \) denotes the index of the \( i \)-th biggest file in the request vector \( d \) (recall that \( f(i) \) was used earlier to denote the \( i \)-th biggest file in the set of all files).

We next consider \( S_2 \), the set of user subsets with more than one user containing only large-cache users. There are \( \binom{K_L}{j} \) user subsets of size \( j + 1 \) in \( S_2 \), for \( j \) values ranging from 1 to \( K_L - 1 \); we cannot have a subset of only large users that contains more members than there are large users. Since there are only large users in these subsets, the subfiles sent will stored on \( j \) large users caches, and so only subfiles of size \( v_{j,l,i}^{(L)} \) are sent. As we saw in earlier proofs, the largest subfile requested (i.e. corresponding to the file with the smallest index), will be sent to \( \binom{K_L-1}{j} \) subsets, the second largest subfile is the largest subfile for \( \binom{K_L-2}{j} \) subsets, and in general, the \( i \)-th largest subfile sent will be sent in \( \binom{K_L-i}{j} \) subsets, giving

\[
\sum_{S \in S_2} E\left[ \max_{k \in S}\{|W_{S\setminus\{k\}}^{(d_k)}|\} \right]
\]
Appendix D. Proof of Proposition 4

\[ = \sum_{j=1}^{K_L-1} \sum_{i=1}^{K_L} (K_L - i) \mathbb{E} \left[ v_{f_j^{S}(i),j}^L \right] \]

\[ = \sum_{j=1}^{K_L-1} \sum_{i=1}^{K_L} (K_L - i) \frac{N}{i} \sum_{l=1}^{N} \mathbb{P}(Y_{i-l}^L = l)v_{i,j}^L \]

\[ = \sum_{j=1}^{K_L-1} \sum_{i=0}^{K_L-1} \sum_{l=1}^{N} (K_L - i + 1) \mathbb{P}(Y_{i}^L = l)v_{i,j}^L, \tag{D.4} \]

where the last line is obtained by rearranging the terms and using a minor change of variable for the index of summation \(i\). This is the second term of Eq. (5.57). Here we use \(f_j^{S}(i)\) to refer to the \(i\)-th biggest file requested within the set of large users, and by \(\mathbb{P}(Y_{i}^L = l)\), we mean the probability that file \(l\) is the \(i+1\)th biggest file requested within the set of large users. We can compute \(\mathbb{P}(Y_{i}^L = l)\) using Lemma 4 with \(N\) files (outcomes) and \(K_L\) users (trials). The change from the second to third lines above then follows immediately from the definition of expectation (see Appendix B).

Using identical reasoning for \(S_3\), the set of user subsets of size greater than 1 with only small users, we can obtain

\[ \sum_{S \in S_3} \mathbb{E} \left[ \max_{k \in S} \{ |W_{S \{k\}}(d_k)| \} \right] = \sum_{j=1}^{K_S-1} \sum_{i=0}^{K_S-1} \sum_{l=1}^{N} (K_S - i + 1) \mathbb{P}(Y_{i}^S = l)v_{i,j}^S \tag{D.5} \]

which is the third term of Eq. (5.57). Here, \(\mathbb{P}(Y_{i}^S = l)\) is the probability that file \(l\) is the \(i+1\)-th biggest file requested among all small users. This can also be computed using Lemma 4, but with \(N\) outcomes and \(K_S\) trials.

Next, we consider \(S_4\), the set of mixed subsets containing more than one user but only one small user. We saw in the non-uniform cache memory case that the coded transmissions to these kinds of groups will consist almost entirely of subfiles whose size is described by mixed variables \(v_{i,j}^M\), because the subfiles are stored on a mixed subset of users’ caches, save for one subfile whose size is described by a large variable \(v_{i,j}^L\), because that subfile is stored only on large user caches. Due the memory inequality constraints (5.54)-(5.56) that prioritize cache size over file size, the one large variable (corresponding to the file requested by the one small user) will necessarily be the maximum value.

The size of the transmissions sent to these kinds of subsets will therefore depend on what files are requested by the small-cache users. Each small cache user will be in \(\binom{K_L}{j}\) many of these subsets for a subset size of \(j+1\), where \(j\) takes values from 1 to \(K_L\); if \(j\) was any larger, there would have to be more than one small user. We therefore have

\[ \sum_{S \in S_4} \mathbb{E} \left[ \max_{k \in S} \{ |W_{S \{k\}}(d_k)| \} \right] = \sum_{j=1}^{K_L} \sum_{i=1}^{K_S} \binom{K_L}{j} \mathbb{E} \left[ v_{f_j^{S}(i),j}^L \right] \tag{D.6} \]

We use \(f_j^{S}(i)\) to denote the index of the \(i\)-th biggest file requested by a small-user. Consequently, the second sum in Eq. (D.6) is over all small cache users, in decreasing order of the file size they requested.
This allows us to compute the expectation in Eq. (D.6) using the $Y^S_i$ variables in the following way:

$$
\sum_{S \in S_4} \mathbb{E} \left[ \max_{k \in S} \left\{ |W^{(d_i)}_{S \backslash \{k\}}| \right\} \right] = \sum_{j=1}^{K_S} \sum_{i=1}^{K_L} \binom{K_L}{j} \sum_{l=1}^{N} \Pr[Y^S_i = l] v^L_{i,j} = \sum_{j=1}^{K_S} \sum_{i=0}^{K_S-1} \sum_{l=1}^{N} \binom{K_L}{j} \Pr[Y^S_i = l] v^L_{i,j}.
$$

(D.7)

The final step is once again attained with a rearranging of terms and a change of variable for the $i$ index of summation. This gives the fourth term in Eq. (5.57).

The fifth term is obtained using similar reasoning. This term corresponds to $S_5$, the set of mixed user subsets containing exactly one large user and more than one small user. Here, the transmitted subfiles will all be stored on the caches of a mixed subset of users, except for the subfile requested by the large user, which will be stored on the caches of every other user in the subset, i.e. all small users. The large user’s requested subfile will have a size described by a small variable $v^S_{l,j}$, and so due to the memory inequality constraints (5.54)-(5.56), will never be the largest subfile transmitted; once again, it is the small-cache user requests that determine the largest subfile. The largest subfile requested by a small user will be transmitted to $\binom{K_S-1}{j-1}\binom{K_L}{1}$ subsets of size $j + 1$; the second largest subfile requested among small users will be the largest subfile transmitted when the largest subfile requested is not also being transmitted to that subset, and so will be transmitted $\binom{K_S-2}{j-1}\binom{K_L}{1}$ times. In general, the $i$-th largest subfile requested among small users will be transmitted only when the previous $i-1$ largest subfiles are not also being transmitted, and so will be sent $\binom{K_S-i}{j-1}\binom{K_L}{1}$ times.

There are subsets of size 3 through $K_S + 1$ in $S_5$, so indexing subset size with $j + 1$ yields

$$
\sum_{S \in S_5} \mathbb{E} \left[ \max_{k \in S} \left\{ |W^{(d_k)}_{S \backslash \{k\}}| \right\} \right]
= \sum_{j=2}^{K_S} \sum_{i=1}^{K_S-1} \binom{K_S-i}{j-1} \binom{K_L}{1} \mathbb{E}[v^M_{l,j}(i)]
= \sum_{j=2}^{K_S} \sum_{i=1}^{K_S-1} \binom{K_S-i}{j-1} \binom{K_L}{1} \sum_{l=1}^{N} \Pr[Y^S_{i-1} = l] v^M_{l,j}
= \sum_{j=2}^{K_S} \sum_{i=0}^{K_S-2} \sum_{l=1}^{N} \binom{K_S-1-i}{j-1} \binom{K_L}{1} \Pr[Y^S_i = l] v^M_{l,j}.
$$

(D.8)

The last line is once again obtained through rearranging terms and doing a change of variables for the index of summation $i$. This is the fifth term of Eq. (5.57).

The sixth term of Eq. (5.57) contains the terms for $S_6$, the set of user subsets with more than $\max\{K_S, K_L\} + 1$ users. These subsets are precisely large enough that they are all mixed and have at least two of each user type in them. Thus only subfiles stored on the caches of mixed subsets of users will be sent, and so they will have a size given by a mixed variable $v^M_{l,j}$. The only factor that determines the largest subfile for a given subset will therefore be the file index. There are no restrictions on subset composition, and so we find ourselves in a familiar situation: the largest subfile requested will be the largest file sent for $\binom{K-1}{j}$ subsets, the second largest subfile requested will be the largest subfile sent for $\binom{K-2}{j}$ subsets, and so on, such that the $i$-th biggest subfile is the largest subfile sent for $\binom{K-i}{j}$ subsets.
This gives

\[
\sum_{S \in \mathcal{S}_7} \mathbb{E} \left[ \max_{k \in S} \{|W_{d_k}^{(d_k)}|\} \right] = \sum_{j=\max\{K_S, K_L\}+1}^{K-1} \sum_{i=1}^{K} \left( K - 1 - i \right) \mathbb{E} \left[ v_{d(i),j}^M \right]
\]

the sixth term of Eq. (5.57).

The seventh and final term of Eq. (5.57) is by far the most complicated term. It corresponds to \( \mathcal{S}_7 \), the set of subsets with at least two large-cache and two small-cache users, but less than \( \max\{K_S, K_L\} + 2 \) users. Here, every subfile sent will be stored on the caches of a mixed subset of users and so will have a size given by a mixed variable \( v_{l,j}^M \). The difficulty arises when we try to characterize the number of transmissions for each of the largest subfile requested, second largest subfile requested, and so on - these numbers are dependent on whether the file was requested by a large user or by a small user. An example will illustrate this fact: consider the third largest file requested among all users and transmissions to subsets of 5 users. If a large-cache user has requested the largest subfile, and another large-cache user requests the second largest subfile, then the number of transmissions where the third largest subfile requested is the largest subfile transmitted is \( \sum_{n=1}^{\binom{K_S}{4-n}} (K_{L-3}^n) \sum_{n=1}^{\binom{K_S-1}{4-n}} (K_{L-2}^n) \) if the subfile is requested by a large-cache user, and \( \sum_{n=2}^{3} (K_{L-2}^n) \sum_{n=0}^{K-1} \sum_{n=0}^{K-1} \mathbb{E} \left[ R_{d(j),n}^{S_7} \right] \) if it is requested by a small-cache user. These two number are clearly not equal in general, and so the cache size of the user making the request matters.

Nevertheless, it is possible to compute the expected rate with a number of terms that scales as a polynomial function of \( N \) and \( K \). To this end, let \( R_{d(j),n}^{S_7} \) denote the number of bits sent to subsets in \( S_7 \) of size \( j+1 \) as part of the transmissions required to satisfy the request \( d \), when the \( n \)-th largest file in \( d \) is the largest subfile transmitted in that subset. By this definition, we have \( R_d = \sum_{j=1}^{7} \sum_{n=0}^{K-1} \sum_{n=1}^{K-1} R_{d(j),n}^{S_7} \). These values have been implicitly computed for \( S_1 \) through \( S_6 \) earlier in this appendix; we only introduce this opaque notation now because the complexity of the accounting done for \( S_7 \) demands it.

The \( R_{d(j),n}^{S_7} \) quantity can be further decomposed: the number of bits sent in this case is equal to the product of the number of transmission sent in this case, denoted by \( T(n) \), and the number of bits transmitted per transmission, which is given buy the appropriate \( v_{l,j}^M \) variable. We can then compute the \( S_7 \) term of Eq. (5.57) using conditional expectation in the following way:

\[
\sum_{S \in \mathcal{S}_7} \mathbb{E} \left[ \max_{k \in S} \{|W_{d_k}^{(d_k)}|\} \right] = \mathbb{E} \left[ \max\{K_S, K_L\} \sum_{j=3}^{K} \sum_{n=1}^{\binom{K_S}{4-n}} \sum_{n=1}^{\binom{K_S-1}{4-n}} R_{d(j),n}^{S_7} \right]
\]

For a fixed \( j \) value, we compute the expectation conditional on the fact that the \( n \)-th largest file is file
Next we consider \( \text{Pr}[S(n) = m] \), the probability that there are \( m \) small-cache users in the set of users who have the \( n-1 \) largest files among all files requested. Since all users have the same preferences, these probabilities are simply determined by the relative numbers of large and small users. Indeed \( S(n) \) has a hypergeometric distribution: the probability that \( m \) of the \( n-1 \) largest files requested are requested by small users (and thus \( n-1-m \) of these files are requested by large users) is given by

\[
\text{Pr}[S(n) = m] = \frac{{K_S \choose m}{K_L \choose n-1-m}}{{K \choose n-1}}.
\] (D.14)

Next we consider \( \text{Pr}[D_S(n) = r|S(n) = m] \), which is obtained with similar reasoning. Once again, since

\[
\sum_{n=1}^{K} \mathbb{E} \left[ R_{d,j}^{S}(n) \right] = \sum_{n=1}^{K} \sum_{l=1}^{N} \mathbb{E} \left[ R_{d,j}^{S}(n) | Y_{n-1} = l \right] \text{Pr}[Y_{n-1} = l]
\]

\[
= \sum_{n=1}^{K} \sum_{l=1}^{N} \mathbb{E} \left[ v_{f_a(n),j}^{M} T(n) | Y_{n-1} = l \right] \text{Pr}[Y_{n-1} = l]
\]

\[
= \sum_{n=1}^{K} \sum_{l=1}^{N} v_{f_a(n),j}^{M} \mathbb{E} \left[ T(n) | Y_{n-1} = l \right] \text{Pr}[Y_{n-1} = l].
\] (D.11)

The last line (D.11) is obtained because, given that \( Y_{n-1} = l \), it follows immediately that \( f_a(n) = l \), and so we have \( v_{f_a(n),j}^{M} = v_{l,j}^{M} \), which is no longer a random quantity.

Comparing Eq. (D.11) to the form of Eq. (5.57) in the statement of the proposition, we see that all that remains is to show that \( \mathbb{E} \left[ T(n) | Y_{n-1} = l \right] = \nu(n,j) \). First, we note that \( \mathbb{E} \left[ T(n) | Y_{n-1} = l \right] \mathbb{E} \left[ T(n) \right] \), since the number of transmissions in which the \( n \)-th largest subfile requested is the largest subfile sent to the subset depends only on the index \( n \) but not the identity of the \( n \)-th largest subfile. Next, letting \( S(n) \) denote the number of small users in the set of users who requested the \( n-1 \) largest files we further decompose the expectation using conditional expectation:

\[
\mathbb{E} \left[ T(n) \right] = \sum_{m=0}^{n-1} \mathbb{E} \left[ T(n) | S(n) = m \right] \text{Pr}[S(n) = m].
\] (D.12)

And further, if \( D_S(n) = 1 \) represents the event that a small user requested the \( n \)-th largest file and \( D_S(n) = 0 \) representing the event that a large user did it, we have

\[
\mathbb{E} \left[ T(n) \right]
\]

\[
= \sum_{m=0}^{n-1} \mathbb{E} \left[ T(n) | S(n) = m \right] \text{Pr}[S(n) = m]
\]

\[
= \sum_{m=0}^{n-1} \sum_{r=0}^{1} \mathbb{E} \left[ T(n) | S(n) = m, D_S(n) = r \right] \text{Pr}[D_S(n) = r | S(n) = m] \text{Pr}[S(n) = m]
\] (D.13)

Now with Eqs. (D.11)-(D.13), we have finally expressed the original expectation in Eq. (D.10) in terms of quantities that can be computed directly.

We begin with \( \text{Pr}[S(n) = m] \), the probability that there are \( m \) small-cache users in the set of users who have the \( n-1 \) largest files among all files requested. Since all users have the same preferences, these probabilities are simply determined by the relative numbers of large and small users. Indeed \( S(n) \) has a hypergeometric distribution: the probability that \( m \) of the \( n-1 \) largest files requested are requested by small users (and thus \( n-1-m \) of these files are requested by large users) is given by

\[
\text{Pr}[S(n) = m] = \frac{{K_S \choose m}{K_L \choose n-1-m}}{{K \choose n-1}}.
\] (D.14)
the large and small users have the same preferences, only their relative numbers will determine the probabilities. Since \( n - 1 \) users have already been accounted for, there are \( K - (n - 1) \) users left to choose from, and if \( m \) of them are small users, there are \( K_S - m \) small users left and \( K_L - (n - 1 - m) \) large users left. This gives

\[
\Pr[D_S(n) = 1|S(n) = m] = \frac{K_S - m}{K - n + 1} \tag{D.15}
\]

and

\[
\Pr[D_S(n) = 0|S(n) = m] = \frac{K_L - n + 1 + m}{K - n + 1} \tag{D.16}
\]

Finally, we compute \( \mathbb{E}[T(n)|S(n) = m, D_S(n) = r] \); the number of transmissions \( T(n) \) is deterministic given the values of \( S(n) \) and \( D_S(n) \), so no probabilities will be involved in the calculation. First, if \( r = 1 \), i.e. a small user has the \( n \)-th largest file request. In this case, the corresponding subfile is the largest subfile transmitted for any transmission to a subset with at least one other small user and two large users, but not the users responsible for the \( n - 1 \) larger requested files. This number is obtained as

\[
\mathbb{E}[T(n)|S(n) = m, D_S(n) = 1] = \sum_{i=1}^{j-2} \binom{K_S - m - 1}{i} \binom{K_L - n + 1 + m}{j - i} \tag{D.17}
\]

for a subset size of \( j + 1 \). The equivalent number for \( r = 0 \), i.e. a large user has the \( n \)-th largest file request, is

\[
\mathbb{E}[T(n)|S(n) = m, D_S(n) = 0] = \sum_{i=2}^{j-1} \binom{K_S - m}{i} \binom{K_L - n + m}{j - i} \tag{D.18}
\]

Substituting the expressions in Eqs. (D.14)-(D.18) into the appropriate places in Eqs. (D.11) - (D.13) yields the desired term, i.e. the seventh and final term of Eq. (5.57), which concludes the proof.
Bibliography


