Unearthing the Impact of Structure in Data and in Topology for Caching and Computing Networks

Federico Brunero

SORBONNE UNIVERSITÉ

EURECOM
Sophia Antipolis

Unearthing the Impact of Structure in Data and in Topology for Caching and Computing Networks

Dissertation

submitted to

Sorbonne Université

in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

by

Federico Brunero

Publicly defended on December 9, 2022, before a committee composed of:

Examiner/Reviewer	**Prof. Giuseppe Caire**	Technical University of Berlin, DE
Examiner/Reviewer	**Prof. Daniela Tuninetti**	University of Illinois Chicago, US
Examiner	**Prof. Mingyue Ji**	University of Utah, US
Examiner	**Prof. Derya Malak**	EURECOM, FR
Examiner	**Prof. B. Sundar Rajan**	Indian Institute of Science, IN
Examiner	**Prof. Kai Wan**	HUST, CN
Thesis Advisor	**Prof. Petros Elia**	EURECOM, FR

The research has been conducted in the Communication Systems Departement at EURECOM (Sophia Antipolis, FR) from July 2019 to December 2022.

Manuscript compiled with pdfLaTeX on January 30, 2023.

To the people I love

Tho' much is taken, much abides; and tho'
We are not now that strength which in old days
Moved earth and heaven, that which we are, we are;
One equal temper of heroic hearts,
Made weak by time and fate, but strong in will
To strive, to seek, to find, and not to yield.

Alfred, Lord Tennyson, *Ulysses*

Acknowledgements

When it comes to writing this specific part of a PhD thesis, it really means that the PhD journey reached the end. Nevertheless, the countless feelings and emotions I experienced during my doctoral studies left such a deep mark on my soul that the PhD is probably something that will never end in my mind. Hence, because of the incredible impact that the last three and a half years had on my life, I feel the need to thank all the people who shared this journey with me, in one way or another.

I thank my advisor Prof. Petros Elia, who gave me this opportunity and supported me throughout these years. Thanks to his determined efforts at understanding the underlying meaning of each research topic we encountered together, I realized how important it is to question our beliefs and ourselves. Our different ways of working made me understand the value that lies in the art of tuning the delicate balance between diverging personalities. These instructive lessons let me mature greatly and I thank him for this.

I thank my committee members Prof. Giuseppe Caire, Prof. Daniela Tuninetti, Prof. Mingyue Ji, Prof. Derya Malak, Prof. B. Sundar Rajan, and Prof. Kai Wan for their valuable comments. Special thanks to Prof. Daniela Tuninetti, whose comforting words really helped me during a difficult moment in my PhD, and Prof. Kai Wan, with whom I deeply enjoyed our research discussions both online and in real life.

I thank all my friends. When I came back from the US, my wish was to stay in touch with all my close friends and I think that I managed to do so. Hence, many thanks to all the people in Italy who, despite the distance and the pandemic that had a dramatic impact on our lives, were always ready to meet me whenever I came back to Torino. I also thank all my friends and colleagues with whom I have spent wonderful moments here in the sunny Côte d'Azur, and so many thanks to Ali, Andrea, Elisa, Eugenio, Neli, Nik — who left his mark although his stay with us was brief — and Nino. Thanks to Emanuele, who was always available to chat when things were not so bright. Special thanks to my friend Francesco, for the many deep philosophical discussions on the beach after work and for his invaluable help

towards the end of this emotional journey. Thank you also to Lamis, a very special person who not only made me look at Nice through a different lens, but also — and most importantly — represented a genuine breath of fresh air at a very difficult time in my existence.

I thank my whole family. I thank my parents, Paola and Roberto, and my sister, Chiara, because their unconditional love and support represent a real safety net, which I often take for granted, whereas I probably should never do so. I also thank my grandmother Didi, whom I am always glad to make proud of me whenever I visit her, and my grandmother Lina, who deserves my deepest gratitude because, although she is no longer with us, I feel that every academic success I achieved is mainly thanks to her, this PhD included.

Finally, I thank Ludovica, my love, immensely. The discomfort and the frustration that I had to face were sometimes so intrusive, overwhelming and oppressive that I just owe her thousands of apologies for all the times that I dragged her — very unjustly — into a bad mood together with me. At the same time, she deserves from me the most heartfelt thanks because, although my company was often anything but joyful over these years, she never lost her clarity of thought, and she always found the unceasing strength to provide me with those lovely and comforting words that played a key role in getting me to the end of this troubled adventure. She always pushed me so far beyond my own limits that I am simply grateful to her for the better person that I am, and for the even better individual that I have yet to become. I am sure that one day I will have the chance to pay her back with all the love she offered me day after day. Now, it only remains for me to wish us both some well-deserved peace of mind, so as to live our present with lightheartedness, no matter what the future holds for us.

Nice, January 30, 2023 Federico Brunero

Abstract

Caching has shown to be an excellent expedient for the purposes of reducing the traffic load in data networks. An information-theoretic study of caching, known as *coded caching*, represented a key breakthrough in understanding how memory can be effectively transformed into data rates, where such transformation relies on carefully designed caching policies that can be employed not only to lower the volume of flowing data, but also to change the structure of the communication problem itself. Besides sparking a flurry of brilliant research works, coded caching also revealed the deep connection between caching and computing networks, which similarly show the same need for novel algorithmic solutions to reduce the traffic load.

Despite the vast literature within the aforementioned caching framework, there remain some fundamental limitations, whose resolution is critical. For instance, it is well-known that the coding gain ensured by coded caching not only is merely linear in the overall caching resources, but also turns out to be severely constrained in most practical settings. These same challenges also affect the computing counterpart, for which the speedup factor brought about by coding is severely hindered in realistic scenarios.

The goal of this thesis is that of exploring, from an information-theoretic point of view, different solutions to possibly surpass the aforementioned constrained linear coding gain, which represents a limiting linear barrier to the actual benefits of coded multicasting. In particular, this manuscript aims at improving and deepening the understanding of the key role that structure plays either in data or in topology for caching and computing networks.

In the first part of the thesis, we explore the fundamental limits of caching under some information-theoretic models that impose structure in data, where by this we mean that we assume to know in advance what data are of interest to whom. We begin with the problem of selfish caching, which can be seen as an extreme way to exploit data structure. Under a relatively broad symmetric system model, we show by means of an information-theoretic converse that selfish caching not only is an ineffective way to capitalize on file preferences, but also implies unbounded damages with respect to non-selfish policies.

Hence, as such selfish approach can fail, we wonder whether there is anything else that we can do to harness any preexisting structure in user profiles. To address this question, we propose an extremely broad (albeit symmetric) system model for the file preferences and we identify the fundamental limits of the so-called coded caching problem with tactical user profiles under uncoded prefetching. For such setting, we prove that our proposed achievable scheme, which basically corresponds to the classical coded caching scheme, is within a constant multiplicative factor of 4 from the optimal. This relevant outcome allows us to conclude that, under the considered broad model, exploiting data structure may provide just a marginal improvement, which is quantifiable by a factor of 4 even considering non-selfish caching strategies. Subsequently, further results are provided for an additional system model for the file preferences, under which we identify, again with the help of a novel converse bound, the optimal worst-case load under uncoded prefetching within a constant multiplicative factor. Finally, we investigate how data structure can be exploited also in the pertinent context of distributed computing. Under the coded distributed computing setting with structured support, which is highly inspired by the caching problem with tactical user profiles and where each output function only depends on a subset of the input files, we identify the optimal communication load within a constant multiplicative gap.

As the neat conclusion above is that structure in data does not allow for a significant improvement over the state of the art, we proceed to show in the second part of the thesis how things radically change when the structure exploited is in network topology. First, we consider the multi-access caching problem with an extremely powerful combinatorial topology, which provides astounding gains that greatly surpass the constrained linear coding gain under modest caching resources. For such system model, we identify the exact optimal worst-case load under uncoded prefetching by extending an existing achievable scheme, and by developing a novel converse bound. Afterward, we explore the topology-agnostic multi-access setting, and we provide two novel converse bounds on the optimal average worst-case load across ensembles of connectivities and under fixed uncoded prefetching, which allow us to certify the goodness of the aforementioned combinatorial topology. Finally, we conclude with the in-depth analysis of a novel multi-access computing model, which is of great interest for real-case applications. For such setting, besides identifying the optimal worst-case load and the optimal max-link load within a constant gap from the optimal, we show how a proper topology shaping yields a concurrent decrease of the communication load and increase of the speedup factor in distributing computations. This is further evidence of the impressive ramifications brought about by structure in topology, which ultimately represents a crucial ingredient to exploit in multi-access networks.

Contents

List of Figures

List of Tables

Notations

For sets we use calligraphic symbols, whereas for vectors we use bold symbols. In each chapter, we may introduce additional notation that will remain confined to the chapter in question.

B	number of bits of each file in the library of a caching problem
$H(\,\cdot\,)$	entropy of a random variable
H_n	group of circular permutations of $[n]$ for some $n \in \mathbb{N}^+$
J	download cost in distributed computing
K	number of users in a caching problem
L	communication load in distributed computing
$L_{\text{max-link}}$	max-link load in distributed computing
M	memory (in number of files) available at each user in a caching problem
N	number of files in a caching problem
R	worst-case communication load
r	computation load in distributed computing
$R_{\text{u,s}}$	worst-case communication load under uncoded and selfish placement
R_{u}	worst-case communication load under uncoded placement
S_n	group of all permutations of $[n]$ for some $n \in \mathbb{N}^+$. The elements of S_n are represented either by permutation functions or by permutation vectors depending on the context. Hence, for a permutation function $\pi \in S_n$, we denote by $\boldsymbol{\pi}^n \coloneqq (\pi(1), \ldots, \pi(n))$ the vector

	of elements from $[n]$ permuted according to the permutation function π and we let π^{-1} denote the inverse function of π. Alternatively, we may also write a permutation of the set $[n]$ directly as $\boldsymbol{u} = (u_1, \ldots, u_n) \in S_n$, where it is implied $\boldsymbol{u} = \boldsymbol{\pi}^n$ for some permutation function $\pi \in S_n$
Z	generic cache content
$[a : b]$	set $\{a, a+1, \ldots, b-1, b\}$ for $a, b \in \mathbb{N}^+$ with $a < b$
$[a : b]_q$	set $\{a \bmod q, (a+1) \bmod q, \ldots, (b-1) \bmod q, b \bmod q\}$ for some $a, b, q \in \mathbb{N}^+$
$[a]$	set $\{1, \ldots, a\}$ for $a \in \mathbb{N}^+$
$[n]_m$	set $\{\mathcal{A} : \mathcal{A} \subseteq [n], \lvert\mathcal{A}\rvert = m\}$ for some $n, m \in \mathbb{N}^+$ with $m \in [0 : n]$
$\alpha \mid \beta$	denotes that the integer α divides integer β for some $\alpha, \beta \in \mathbb{N}^+$
$\binom{n}{k}$	binomial coefficient, where $\binom{n}{k} = 0$ whenever $n < 0$, $k < 0$ or $n < k$ for some $n, k \in \mathbb{N}^+$
$\lfloor \cdot \rfloor$	denotes the floor function
$\lvert \cdot \rvert$	size or cardinality of the taken argument
$\mathbb{F}_{2^m}^n$	denotes the n-dimensional vector space over the finite field with cardinality 2^m for some $n, m \in \mathbb{N}^+$
$\mathbb{N}$	set of non-negative integers
$\mathbb{N}^+$	set of positive integers
$\mathcal{S}^N$	N-ary Cartesian product, i.e., $\mathcal{S}^N = \mathcal{S}_1 \times \cdots \times \mathcal{S}_N = \prod_{n \in [N]} \mathcal{S}_n = \{(s_1, \ldots, s_N) : s_n \in \mathcal{S}_n, n \in [N]\}$ given N sets $\{\mathcal{S}_n : n \in [N]\}$
$\oplus$	bitwise XOR operation
$m \bmod n$	modulo operation on m with integer divisor n, letting $m \bmod n = n$ when n divides m

Chapter 1

Introduction

Our lives strongly depend on our ability to communicate with each other. Even though we always take it for granted, communication is essential. It is fundamental when we teach our children a language, when we argue with our friends about politics, or when we need to express our own feelings to the people we love. Whether we realize it or not, every little facet of our lives involves the aspect of conveying our emotions, thoughts, and beliefs. After all, even Aristotle said[1] that human beings are by nature *social animals*, who are inherently inclined at socializing, at creating communities and, ultimately, at communicating.

Nevertheless, we do not express ourselves always in the same manner. Depending on the context in which we find ourselves, we often modify the way we interact with each other. For instance, when we talk to a kid who is still learning our mother tongue, it is crucial to use very simple and often redundant words; when we go to a concert with our friends, we often need to speak louder because of the background music, which interferes with our voice. Clearly, there are several factors that might change the way we relate to other people. Since each of us has their own sensitivity, we do not always want to say or hear the same things; at the same time, we often interact with the surrounding context in different ways. What we always do, though, is to adapt our communication style — maybe even unconsciously — to the circumstances, so as to communicate in the most suitable and efficient way.

Now, it is reasonable to assume that the experienced reader who is reading this manuscript is extremely familiar with the technical content that will follow this brief introduction. Hence, the two small paragraphs above are intended for the more inexperienced reader, who certainly deserves to grasp at least, at a high level, the content of this thesis. Broadly speaking, as human

[1] See *Politics*, 1253a.

beings always adapt to and sometimes mold the surrounding environment in order to convey their thoughts and words more efficiently, we can say in short that this manuscript studies how efficiently we can communicate under some well-defined circumstances. More formally, and for the more experienced reader, this thesis studies the fundamental limits of some powerful cache-aided communication models, and since such study greatly relies on the mathematical tools provided by information theory, we can safely conclude that this thesis naturally fits within the research context of information-theoretic caching.

1.1 Motivation for Caching

Novel data-hungry applications are expected to put great pressure on current and future communication networks, whether this will be due to improved video streaming platforms, to the introduction of novel cloud computing services, or to the emergence of innovative applications that will play a key role in future networking standards [1]. As the traffic growth seems inevitable in the upcoming future, one can understand at this point why there has been constant interest into looking for new efficient communication techniques.

1.1.1 Caching as a Promising Solution

Although the continuous improvement of physical devices is certainly going to provide a better infrastructure to support higher data rates, future communication systems cannot solely rely on the development of faster and more resilient electronic equipment. Hence, looking for novel algorithmic solutions becomes an urgent need to the purpose of increasing the speed of communications under the limiting constraints dictated by the physical medium. In this regard, considering that a large portion of data traffic consists of cacheable content, the idea of *caching* seems definitely a promising solution.

When we think of communicating data through a communication network, the first things that come to mind are network resources, such as spectrum frequencies to be allocated in a wireless setting, switches to be configured in a data center, or time slots to be carefully assigned in shared-medium networks. Typically, such network resources are often both expensive and very limited. Nevertheless, one thing we often forget is the concept of *memory*. On the one hand, memories are cheap and abundant; on the other hand, memories can be efficiently used with the purpose of reducing the amount of data flowing through highly saturated communication networks. Indeed, if storage devices are properly used to proactively *cache* data, then we can effectively take

advantage of storage capabilities to transform memory into data rates. This is exactly what *coded caching* achieves.

1.1.2 The Major Breakthrough of Coded Caching

Now, after pointing out the importance of taking advantage of storage capabilities, it comes natural to wonder whether there are specific caching strategies which are more effective than others, and how we can study analytically the improvements brought about by the use of memories in communication networks. In this regard, Maddah-Ali and Niesen (MAN) approached the caching problem from an information-theoretic perspective, and proposed in [2] the so-called *coded caching* framework. The seminal work in [2] represented a major breakthrough, since it revealed that carefully designed caching policies can be employed not only to lower the volume of flowing data, but also to change the structure of the communication problem itself.

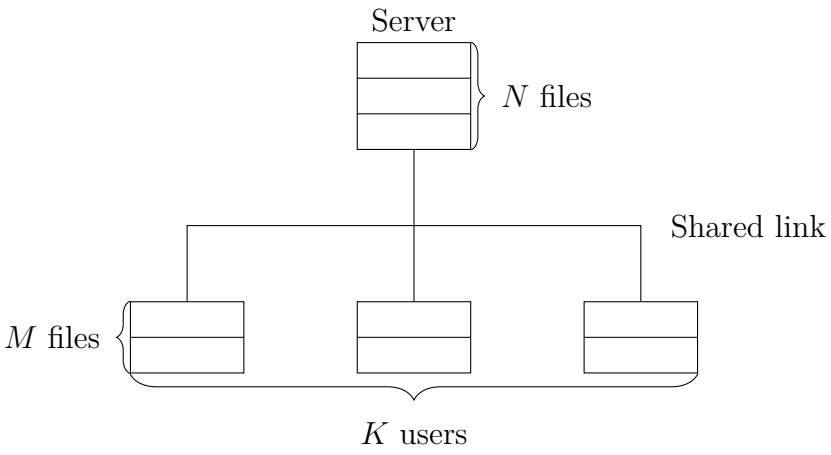

Figure 1.1: A server with access to a library of N files is connected through a shared error-free broadcast link to K users, each having a cache of size equal to M files.

The caching model considered by Maddah-Ali and Niesen in [2] consists of a central server, which has access to a library containing N files $\{W_n : n \in [N]\}$ of B bits each. The server is connected to K users through a shared error-free broadcast channel, and each user is equipped with a cache of size MB bits, or, equivalently, M files. A schematic of the system model is represented in Figure 1.1.

The caching problem consists of two sequential phases. During the first phase, which is referred to as the *placement phase*, the central server fills

the caches without any knowledge of the future requests from the users. Typically, this phase takes place during off-peak hours, when the network is not overloaded. During the second phase, which is called *delivery phase*, and which typically happens when the network is saturated and interference-limited, the requests of the users are simultaneously revealed to the central server. During delivery, the server prepares the broadcast message X, which is then sent over the error-free broadcast link, so that each user can retrieve the missing information from the received transmission. The users cancel the interference terms that appear in the broadcast transmission, and do so by means of the cached contents they have access to, eventually decoding their own messages.

In the aforementioned caching problem, the task is to design the placement-and-delivery scheme that minimizes the worst-case communication load R, which is defined as the number of bits — normalized by the file size B — transmitted during the delivery phase in the worst-case scenario. More in general, the goal is to characterize the optimal worst-case communication load $R^\star$, which is formally defined as

$$R^\star(M) \coloneqq \inf\{R : (M, R) \text{ is achievable}\} \tag{1.1}$$

where the tuple (M, R) is said to be *achievable* if there exists a caching-and-delivery scheme which guarantees, for any possible demand and a given memory M, a load R.

Before we describe the coded caching scheme in its general form, we first present in the following a small toy example, so as to show the key ideas behind coded caching.

A Toy Example

Consider the following simple caching network, where we have $N = 2$ files in the main library and there are $K = 2$ users connected to the central server. We assume that each user is equipped with a cache of size $M = 1$ file. As we mentioned above, the caching problem is split into the placement phase and the delivery phase.

Placement phase This phase takes place much before the users reveal their requests to the server. Each file in the main library is evenly divided in two parts, so that we have

$$W_1 = \{W_{1,1}, W_{1,2}\} \tag{1.2}$$
$$W_2 = \{W_{2,1}, W_{2,2}\} \tag{1.3}$$

where $|W_{n,1}| = |W_{n,2}| = B/2$ for $n \in \{1, 2\}$. Then, if we denote by Z_k the content of the cache of user k with $k \in \{1, 2\}$, the server fills the caches Z_1 and Z_2 as follows

$$Z_1 = \{W_{1,1}, W_{2,1}\} \tag{1.4}$$
$$Z_2 = \{W_{1,2}, W_{2,2}\}. \tag{1.5}$$

We can verify that this placement scheme does not exceed the available memory at the users. For example, if we focus on the cache of user 1, we can see that it contains two subfiles, i.e., subfile $W_{1,1}$ and subfile $W_{2,1}$, where each of them has size $B/2$ bits. This means that Z_1 contains a total of B bits or, equivalently, $M = 1$ file. The same holds for the cache of user 2.

Delivery phase This phase commences after the users reveal their requested files. If we assume in this example that user 1 requests W_1 and user 2 requests W_2, we can see from the content of Z_1 and Z_2 that: user 1 misses $W_{1,2}$ to correctly reconstruct W_1; user 2 misses $W_{2,1}$ to correctly reconstruct W_2. At this point, the server transmits the coded message $X = W_{1,2} \oplus W_{2,1}$ over the broadcast channel. It is relatively straightforward to check that each user can correctly decode the missing piece of the requested file from the transmission X.

- User 1 has in its cache Z_1 the term $W_{2,1}$, which can be removed from the coded message X by calculating $X \oplus W_{2,1} = W_{1,2}$, decoding in this manner the missing term $W_{1,2}$.
- Similarly, user 2 has in its cache Z_2 the term $W_{1,2}$, which can be removed from the coded message X by calculating $X \oplus W_{1,2} = W_{2,1}$, decoding in this manner the missing term $W_{2,1}$.

In conclusion, each user receives the missing piece of the requested file.

For the scheme above, we can see that the achieved communication load is equal to $R_{\text{MAN}} = 1/2$, where R_{MAN} denotes the communication load guaranteed by the MAN coded caching scheme. First, we recall that the load is defined as the number of transmitted bits, normalized by the file size B. Then, we observe that the transmitted message X is given by the bitwise XOR of two subfiles with size $B/2$, which implies that the size of X is equal to $B/2$. This means that $R_{\text{MAN}} = |X|/B = 1/2$.

Notice that, if we had decided to let the server deliver first $W_{1,2}$ to user 1, and then $W_{2,1}$ to user 2, the total number of transmitted bits would have been equal to the size of $W_{1,2}$ plus the size of $W_{2,1}$, i.e., B bits in total. Hence, this simple yet instructive example shows how the use of coding during the

delivery procedure allowed to halve the total number of (normalized) bits delivered by the central server to the users.

The Coded Caching Scheme

We can proceed now to the description of the MAN coded caching scheme in its general form. We recall that in the coded caching framework we consider a main library containing N files $\{W_n : n \in [N]\}$ of B bits each and K users that are equipped with a cache of MB bits. The users are connected to the central server through an error-free broadcast channel.

Placement phase Let $M = tN/K$, where $t \in [0 : K]$. Each file is split into $\binom{K}{t}$ non-overlapping subfiles as

$$W_n = \{W_{n,\mathcal{T}} : \mathcal{T} \subseteq [K], |\mathcal{T}| = t\} \tag{1.6}$$

for each $n \in [N]$. The cache of user k is then filled as

$$Z_k = \{W_{n,\mathcal{T}} : \mathcal{T} \subseteq [K], |\mathcal{T}| = t, k \in \mathcal{T}\} \tag{1.7}$$

for each $k \in [K]$. We can check that the total number of subfiles cached by each user is equal to $\binom{K-1}{t-1}$ for each of the N files in total. Hence, considering that each subfile has size $B/\binom{K}{t}$, we have

$$|Z_k| = N\frac{\binom{K-1}{t-1}}{\binom{K}{t}}B = MB \tag{1.8}$$

for each $k \in [K]$. Consequently, the cache memory constraint is satisfied.

Delivery phase After the demand vector $\boldsymbol{d} = (d_1, \ldots, d_K)$ is revealed, where we denote by W_{d_k} the file requested by user $k \in [K]$, the server prepares the transmission X, which is given by

$$X = \left(\bigoplus_{k \in \mathcal{S}} W_{d_k, \mathcal{S}\setminus\{k\}} : \mathcal{S} \subseteq [K], |\mathcal{S}| = t + 1 \right). \tag{1.9}$$

We recall that $d_k \in [N]$ for each $k \in [K]$.

Since the broadcast message X contains a total of $\binom{K}{t+1}$ bitwise XORs, the worst-case load R_{MAN} is given by

$$R_{\text{MAN}}(t) = \frac{|X|}{B} = \frac{\binom{K}{t+1}}{\binom{K}{t}} \tag{1.10}$$

for each $t \in [0:K]$. For non-integer values of t, a memory-sharing scheme can be employed, as described in [2]. In general, the worst-case load R_{MAN} is a piece-wise linear curve with corner points

$$(M, R_{\text{MAN}}) = \left(t\frac{N}{K}, \frac{\binom{K}{t+1}}{\binom{K}{t}} \right), \quad \forall t \in [0:K]. \tag{1.11}$$

This achievable performance in (1.11) was shown to be exactly optimal in [3], [4] under the assumption of uncoded cache placement.

Definition 1.1 (Uncoded Cache Placement)**.** A cache placement is *uncoded* if the bits of the files are simply copied within the caches of the users.

1.2 Main Contributions

Since its original information-theoretic formulation, coded caching has been rightfully credited with being able to transform memory into data rates, and has so sparked a flurry of research on a variety of topics, such as on information-theoretic converses [3], [4], the interplay between caching and PHY [5]–[15], caching and privacy [16]–[18], and the critical bottleneck of subpacketization [19]–[25]. More recently, we saw interesting findings on coded caching under file popularity considerations [26]–[30] or in scenarios with heterogeneous user preferences [31]–[36], as well as on a variety of other scenarios [37]–[46], including the recent finding that coded caching can provide astounding gains over realistic downlink systems [47].

Nevertheless, despite the several variations of coded caching that have been explored since its original formulation, there are still many open questions to address and several limitations to overcome. For instance, it is well-known that the coding gain[2] guaranteed by coded caching not only is merely linear in the overall caching resources, but also turns out to be quite modest in practice due to several reasons[3]. This phenomenon holds also in the computing setting, which is closely related to the coded caching framework and which similarly shows that the speedup factor brought about by coding is severely hindered in realistic scenarios. Hence, our goal here is that of exploring different solutions to possibly surpass such constrained linear coding gain, which represents a limiting linear barrier that dramatically hinders the actual benefits of coded

[2]With *coding gain* we refer to the multiplicative reduction in the volume of data brought about by the use of coding. In the single-stream scenario, this effectively coincides with the number of users served at a time by each coded packet in the broadcast message X.

[3]For example, the subpacketization, which is a direct consequence of the combinatorial nature of the MAN scheme, represents a relevant bottleneck in practical settings.

multicasting in practical settings. In particular, the goal of this thesis is to improve and deepen the understanding of the key role that structure plays either in data or in topology for caching and computing networks, where the underlying *structure* — whether in data or in topology — will be defined by similar combinatorial abstractions. When the structure is imposed in data, we will show that this does not help significantly toward breaking the aforementioned linear barrier — in fact, we will show that structure can even hurt, if it is exploited in the wrong way. Instead, when the structure is imposed in the topology, we will demonstrate how we can cleverly take advantage of it, providing for both the caching and the computing setting stunning gains. In the following, we outline the main contributions that will be presented in this manuscript.

1.2.1 Exploring the Impact of Structure in Data

In the first part of the thesis, we investigate the benefits that we can possibly obtain from exploiting the structure in data. As data can reveal some underlying structure in several ways, we point out that in our context what really matters is what data is desired by whom. For this reason, we will explore user preferences, where by this we imply that each user is assumed to have a file demand set (FDS) of desired files. We organize this first part of the manuscript in four distinct chapters, where the first three chapters will regard coded caching with file preferences, and the last chapter will show how some of the results developed in the context of caching can be employed in computing networks. We provide a brief summary of each chapter in the following.

A Negative Result on Selfish Caching Policies

As we mentioned above, our aim of exploiting some preexisting structure in data is equivalent to taking advantage of the fact that we know what data are desired by whom. Hence, since in a realistic scenario the users may have diverging interests which may intersect to various degrees, what happens for example if each file is of potential interest to, say, 40 % of the users and each user has potential interest in 40 % of the library? In this regard, one approach that is worthy of exploring in caching is the so-called *selfish caching* approach, which can be seen as the most aggressive way to capitalize on some previous knowledge on the user preferences.

A key ingredient in using caches has commonly been the exploitation of the fact that some contents/files are more popular than others, and so are generally to be allocated more cache space [48], [49]. This inevitably introduces the

consideration that different users may have different file preferences, which in turn brings to the fore the concept of *selfish caching* where simply users cache independently and selfishly only contents that they are interested in potentially consuming themselves [32]–[36]. In the traditional prefetching scenario, where emphasis is based heavily on bringing relevant content closer to each user, this idea of selfish caching brought about performance improvements [50], [51] in the form of higher local caching gains for each user.

Nevertheless, we have to be careful, as this selfish idea can have some serious ramifications (as it will be proved). Indeed, we recall that the main idea of coded caching is that it multicasts at any given time a linear combination of different contents desired by different users. This implies that any one receiver associated to a multicast message must be able to find in its cache all the undesired contents (subfiles) of that multicast message. This is achieved in [2] by means of a highly structured and coordinated content placement phase, where each user caches a small fraction of *every* file of a common library. This relationship between undesired and cached contents deteriorates when using selfish caching, simply because each receiver selfishly opts — based on its own preferences — to not cache some of these undesired files. However, these same undesired files may eventually appear as interference at that selfish receiver who will now not be able to "cache-out" this interference. At the same time, though, such selfish caching allows for a much more targeted placement of files such that each user can cache more of what it actually wants.

In this thesis, Chapter 2 provides novel and unified results in the context of selfish coded caching. First, we propose the symmetric (K, α, F) FDS structure, which assumes an N-file library $\{\mathcal{W}_{\mathcal{S}} : \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha\}$ to be a collection of disjoint file classes $\mathcal{W}_{\mathcal{S}} = \{W_{f,\mathcal{S}} : f \in [F]\}$, with each class $\mathcal{W}_{\mathcal{S}}$ consisting of F different files. For such setting, we then consider that each user $k \in [K]$ has an FDS given by

$$\mathcal{F}_k = \{\mathcal{W}_{\mathcal{S}} : \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha, k \in \mathcal{S}\} \tag{1.12}$$

which describes the files this user is potentially interested in. After proposing a new selfish system model, which aims to calibrate the selfishness effect by calibrating the degree of separation between the interests of the different users, we employ index coding arguments to derive an information-theoretic converse (lower bound) on the optimal worst-case communication load $R^{\star}_{\text{u,s}}$ under the assumption of uncoded and selfish placement. Such converse bound is shown to be a piecewise linear curve with corner points

$$(M, R_{\text{LB}}) = \left(t\frac{N}{K}, \frac{\binom{\alpha}{t+1} + (K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}} \right), \quad \forall t \in [0:\alpha]. \tag{1.13}$$

In the above, we recall that the parameter α denotes the number of users which each class of files is of interest to. The case $\alpha = K$ trivially corresponds to the standard MAN system model, where every file is of interest to every user; instead, the case $\alpha = 1$ corresponds to the other extreme case, where each user has its own set of files that is not of interest to any other user.

The main contribution of Chapter 2 is the information-theoretic converse above, which not only represents a novelty in terms of the combinatorial arguments exploited for its construction, but also allows us to draw the powerful conclusion that selfish coded caching can be, under some assumptions, rather detrimental. Indeed, as it will be shown in Corollary 2.1.1, we have that

$$\frac{R^{\star}_{\mathrm{u,s}}(t)}{R_{\mathrm{MAN}}(t)} \geq 1, \quad \forall t \in [0 : \alpha - 1] \tag{1.14}$$

$$\frac{R^{\star}_{\mathrm{u,s}}(t)}{R_{\mathrm{MAN}}(t)} > 1, \quad \forall t \in [\alpha - 2] \tag{1.15}$$

where this implies that unselfish coded caching strictly outperforms — in the non-trivial memory regime $t \in [\alpha - 2]$ — any conceivable implementation of selfish coded caching under the proposed symmetric FDS structure. In addition, as it will be shown in Corollary 2.1.2, the coding gain (CG) of selfish caching does not scale as the number of users K increases, but it is instead bounded as

$$\mathrm{CG} < \frac{1}{1 - \alpha/K} \tag{1.16}$$

for any fixed ratio α/K. Recalling the original premise of unbounded linear coding gain, the above implies that indeed selfish caching can be unboundedly detrimental and so one must be careful before considering selfish caching policies, as these can yield unbounded losses over non-selfish coded caching. Further details and comments will be provided in Chapter 2.

Coded Caching With Tactical User Profiles

From the results in Chapter 2, we can definitively conclude that selfish caching can fail, as it turns out to be a too extreme approach. Nevertheless, we still wonder whether there is anything else that we can do to harness some preexisting structure in user preferences. In particular, we ask ourselves again what we can do if each user is interested in the same fraction of files N'/N and each file is of interest to the fraction of users K'/K for some $N' \in [N]$ and $K' \in [K]$. While in Chapter 2 we investigated such setting under a very strict caching policy and under a very strict combinatorial structure, in Chapter 3 we take one step further and we address the aforementioned

question under the most general (albeit symmetric) structure for the user interests, for which we allow any caching policy (as long as it is uncoded) and we do not impose any specific pattern in the way the user profiles overlap. These two aspects represent a massive difference with respect to the setting previously considered in Chapter 2.

First, we propose an extremely broad system model for the user profiles, where such model follows the combinatorial structure of the so-called *tactical configurations*, a symmetric and balanced block design in combinatorial mathematics. In such setting, we assume that the N files in the system are split in G groups containing N/G files each, where this implies that the library can be partitioned as

$$\{W_1, \dots, W_N\} = \{\mathcal{W}_{\mathcal{K}_1}, \dots, \mathcal{W}_{\mathcal{K}_G}\} \tag{1.17}$$

where $\mathcal{K}_g \subseteq [K]$ is the subset of users which are interested in the set of files given by $\mathcal{W}_{\mathcal{K}_g}$ for some $g \in [G]$. Then, we assume that each user is interested into G' distinct groups of files, whereas each file is of interest to exactly K' users, namely, we assume that $|\{g : g \in [G], k \in \mathcal{K}_g\}| = G'$ for each $k \in [K]$ and that $|\mathcal{K}_g| = K'$ for each $g \in [G]$. This implies that the equation $GK' = KG'$ holds, where it is assumed that $G, G', K', K \in \mathbb{N}^+$. Further, the above implies that the FDS $\mathcal{F}_k$ of user $k \in [K]$ is given by

$$\mathcal{F}_k = \{\mathcal{W}_{\mathcal{K}_g} : g \in [G], k \in \mathcal{K}_g\}. \tag{1.18}$$

Then, referring to such problem as coded caching with tactical user profiles, we characterize the fundamental limits of caching under the proposed broad structure for the user interests. Interestingly, our proposed achievable scheme consists of the standard MAN coded scheme for any $t \leq \bar{t}$, whereas we employ a memory-sharing scheme for any $t > \bar{t}$. We show that the optimal worst-case load $R_{\mathrm{u}}^{\star}$ is upper bounded by the lower convex envelope of the following memory-load corner points

$$(M, R_{\mathrm{UB}}) = \left(t\frac{N}{K}, g(t)\right), \quad \forall t \in [0 : K'] \tag{1.19}$$

where $g(t)$ is defined as

$$g(t) \coloneqq \begin{cases} \dfrac{\binom{K}{t+1}}{\binom{K}{t}}, & \text{if } t \in [0 : \bar{t}] \\ \dfrac{\binom{K}{\bar{t}+1}}{\binom{K}{\bar{t}}} \dfrac{K' - t}{K' - \bar{t}}, & \text{if } t \in [\bar{t} + 1 : K'] \end{cases} \tag{1.20}$$

and $\bar{t} \coloneqq \lfloor K'/2 \rfloor$. Our converse bound shows instead that the optimal worst-case load $R_{\mathrm{u}}^{\star}$ is lower bounded by the lower convex envelope of the following memory-load corner points

$$(M, R_{\mathrm{LB}}) = \left(t\frac{N}{K}, \frac{G}{G'} \frac{\binom{K'}{t+1}}{\binom{K'}{t}} \right), \quad \forall t \in [0 : K']. \tag{1.21}$$

The interesting outcomes of Chapter 3 are several. First, the techniques employed for the construction of the converse represent a novelty in the way we capture the essential structural aspects of the system model considered here. Then, as it will be proved in Corollary 3.3.1, we identify one specific instance of the coded caching problem with tactical user profiles whose performance cannot be surpassed irrespective of the symmetric structure for the user profiles, as the converse bound is shown to be exactly optimal. At the same time, we prove that, despite the nature of the interests of the users, our proposed achievable scheme — which, we recall, coincides with the MAN coded scheme for $t \leq \bar{t}$ — is within a constant multiplicative factor of at most 4 from the optimal. This result is significant, we believe, because it shows, further again, the power of the combinatorial MAN coded caching scheme under a well-defined system model for the user profiles. It is worth noting that the proposed system model under tactical configurations includes some other well-known structures, such as the symmetric (K, α, F) FDS structure. Hence, this implies that the result in Chapter 3 holds also for the system model investigated in Chapter 2.

On the Optimality of Coded Caching With Heterogeneous User Profiles

This first part of the manuscript continues with Chapter 4, which focuses on a system model for the user profiles that is different from — and, so, not captured by — the generic model studied in Chapter 3. More specifically, in Chapter 4 we further explore the system model proposed in [34], where the files in the library are divided in two categories, i.e., common files and unique files. We consider that there are N_{c} common files $\{W_n^{\mathrm{c}} : n \in [N_{\mathrm{c}}]\}$, where each of them is of interest to every user in the system; then, for each group $g \in [G]$, we assume that there are N_{u} unique files $\{W_n^{\mathrm{u},g} : n \in [N_{\mathrm{u}}]\}$, where each of them is of interest to the users belonging to the group $g \in [G]$ only.

Under the aforementioned system model, and taking advantage of the genie-aided converse bound idea from [4] — similarly adopted also in Chapter 3 — we provide an information-theoretic lower bound on the optimal worst-case

communication load under uncoded prefetching, which is given by

$$R_{\mathrm{u}}^{\star} \geq \min_{\beta} \frac{1}{2} \left(\frac{\binom{K}{t_{\mathrm{c}}+1}}{\binom{K}{t_{\mathrm{c}}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}+1}}{\binom{K/G}{t_{\mathrm{u}}}} \right) \tag{1.22}$$

where $t_{\mathrm{c}} = K\beta M/N_{\mathrm{c}}$ and $t_{\mathrm{u}} = K(1-\beta)M/GN_{\mathrm{u}}$. In addition, the derived converse, together with an already existing achievable scheme from [34], allows us to characterize the memory-load tradeoff under uncoded placement within a constant multiplicative factor of 2.

Coded Distributed Computing With Structured Support

Finally, the first part of the manuscript is concluded by Chapter 5, where we present how the information-theoretic system model proposed in Chapter 3 can be employed in the context of distributed computing. In such case, we show how structure in data can be employed to improve the coded distributed computing (CDC) scheme proposed originally in [52].

More specifically, we consider the distributed computing problem with K computing nodes and N input files of F bits each. The goal of node $k \in [K]$ is to compute the output function $\phi_k \colon \mathbb{F}_{2^F}^{|\mathcal{N}_k|} \to \mathbb{F}_{2^B}$ which maps the files in $\mathcal{N}_k \subseteq \{w_1, \ldots, w_N\}$ to a bit stream $u_k = \phi_k(w_n : w_n \in \mathcal{N}_k) \in \mathbb{F}_{2^B}$ of length B bits. We assume that each output function ϕ_k with $k \in [K]$ is *decomposable*, which implies $\phi_k(w_n : w_n \in \mathcal{N}_k) = h_k(g_{k,n}(w_n) : w_n \in \mathcal{N}_k)$. Simply, for each $k \in [K]$ we have a map function $g_{k,n} \colon \mathbb{F}_{2^F} \to \mathbb{F}_{2^T}$, which maps the input file $w_n \in \mathcal{N}_k$ into an intermediate value (IV) $v_{k,n} = g_{k,n}(w_n) \in \mathbb{F}_{2^T}$ of T bits, and a reduce function $h_k \colon \mathbb{F}_{2^T}^{|\mathcal{N}_k|} \to \mathbb{F}_{2^B}$, which maps the IVs $v_{k,n} = g_{k,n}(w_n)$ for each $w_n \in \mathcal{N}_k$ into the output value $u_k = h_k(v_{k,n} : w_n \in \mathcal{N}_k) \in \mathbb{F}_{2^B}$ of B bits.

In Chapter 5, we focus on a specific structure for what concerns the sets $\mathcal{N}_k$ for each $k \in [K]$, where such structure follows the broad model from Chapter 3 based on tactical configurations. In other words, we investigate the symmetric setting where the only constraint is that each reduce function depends on the same number of intermediate values N', and each intermediate value is needed by the same number of reduce functions K'. Under such extensive system model, we characterize the communication load proposing both an achievable scheme and a converse bound, which are shown to be within a constant multiplicative factor of 6.

1.2.2 The Ramifications of Structure in Topology

After showing that structure in data, if exploited, provides marginal improvements under a relatively broad setting, we will proceed to show how things

radically change if the structure to exploit is in the topology. In particular, we will investigate the key role of network topology in multi-access networks and the related remarkable gains across two chapters. In the first chapter, we will study in depth a novel network topology in the context of multi-access coded caching. Subsequently, the second and last chapter of this second part will provide new results in the context of distributed computing, proposing a novel multi-access distributed computing model. We provide a brief summary of each chapter in the following.

Combinatorial Multi-Access Caching

The original coded caching model in [2] considered that each user has access to its own single dedicated cache. However, in several scenarios it is conceivable and perhaps more realistic that each cache serves more than one user, and that each user can connect to more than one cache. This motivated the work in [29] that introduced the so-called multi-access coded caching (MACC) model, which involved Λ users and Λ caches, and involved a topology where each user is connected to[4] $\alpha > 1$ consecutive caches in a cyclic wrap-around fashion as in Figure 1.2, such that each cache serves exactly α users.

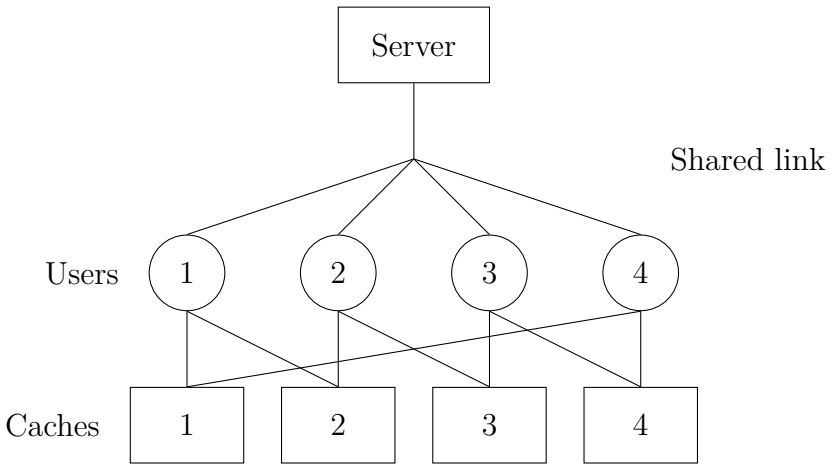

Figure 1.2: MACC model where there are $\Lambda = 4$ caches and each user is connected to $\alpha = 2$ consecutive caches following a cyclic wrap-around topology.

For various reasons, most of the works in the literature focused on the MACC model with cyclic wrap-around topology. Such topology, though,

[4]We urge the reader to note that now the parameter α means something different than what it meant in the first part of the manuscript.

never showed to be of real impact due to the modest improvement over the single dedicated cache setting. A substantial breakthrough came with the very recent work in [53], which proposed a MACC model enjoying the same amount of resources α and Λ, but where now the users and the caches are connected following the well-known combinatorial topology of combination networks [54]–[57]. Having as a starting point the combinatorial multi-access system model introduced in [53], we first propose in Chapter 6 a model extension which allows for a denser range of possible number of users K and for the coexistence of users that are connected to different numbers of caches. In particular, in our system model we assume that any one set of α caches is uniquely assigned to K_α users, and this holds for every $\alpha \in [0:\Lambda]$. This further implies that the total number of users is given by

$$K = \sum_{\alpha=0}^{\Lambda} K_\alpha \binom{\Lambda}{\alpha}. \tag{1.23}$$

Then, in addition to extending the delivery scheme presented in [53] to our generalized combinatorial system model, we prove our coded scheme to be exactly optimal under the assumption of uncoded placement by means of an information-theoretic converse that is based on index coding arguments. We show that the optimal worst-case communication load $R^\star_{\text{comb}}$ under uncoded prefetching is a piecewise linear curve with corner points

$$(M, R^\star_{\text{comb}}) = \left(t\frac{N}{\Lambda}, \sum_{\alpha=0}^{\Lambda-t} K_\alpha \frac{\binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{t}} \right), \quad \forall t \in [0:\Lambda]. \tag{1.24}$$

As a side-product of identifying the fundamental limits of performance in this broad topology, our result shows that, when there are users connected to a different number of caches α with $\alpha \in [0:\Lambda]$ in accordance to the combinatorial topology, then a MAN cache placement and a simple TDMA-like application of the scheme in [53] is enough to achieve the minimum possible load in the worst-case scenario. This implies that treating each α-setting separately is optimal and so there would not be any advantage in encoding across users connected to a different number of caches.

Subsequently, we consider a topology-agnostic scenario, which — to the best of our knowledge — has not been considered so far in the literature of multi-access caching. For such agnostic scenario, we investigate various ensembles of connectivities, including the ensemble $\mathcal{B}$ of all possible connectivities, as well as the smaller ensemble $\mathcal{B}_\alpha$ of those connectivities that simply abide by the constraint that each user is connected to the same number of α caches, without any additional structural constraint on the connectivity or

on the number of users that each cache has to treat. For such settings, we develop novel information-theoretic converse bounds on the optimal average worst-case load, where the average is taken over the ensemble of interest, and where the optimal is over an optimized fixed placement. In particular, when we consider the ensemble $\mathcal{B}_\alpha$ of multi-access coded caching problems with Λ caches and $K = K'_\alpha$ users each connected to exactly $\alpha \in [\Lambda]$ caches, we show that the optimal average worst-case communication load $R^\star_{\text{avg},\mathcal{B}_\alpha}$ is lower bounded — under the assumption of fixed uncoded cache placement and equiprobable connectivities — by $R_{\text{avg},\mathcal{B}_\alpha,\text{LB}}$ which is a piecewise linear curve with corner points

$$(M, R_{\text{avg},\mathcal{B}_\alpha,\text{LB}}) = \left(t\frac{N}{\Lambda}, \frac{K'_\alpha \binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{t}} + A_t \right), \quad \forall t \in [0 : \Lambda - \alpha + 1] \tag{1.25}$$

where A_t is defined as

$$A_t \coloneqq \frac{K'_\alpha}{|\mathcal{B}_\alpha|} \binom{\Lambda - t}{\alpha} \left(1 - \frac{1}{\binom{t+\alpha}{\alpha}} \right). \tag{1.26}$$

Then, when we consider the ensemble $\mathcal{B}$ of multi-access coded caching problems with Λ caches and K users, where any set of caches can arbitrarily serve any number of users without any constraint or structure, we show that the optimal average worst-case communication load $R^\star_{\text{avg},\mathcal{B}}$ is lower bounded — under the assumption of fixed uncoded prefetching and equiprobable connectivities — by $R_{\text{avg},\mathcal{B},\text{LB}}$ which is a piecewise linear curve with corner points

$$(M, R_{\text{avg},\mathcal{B},\text{LB}}) = \left(t\frac{N}{\Lambda}, \sum_{\alpha=0}^{\Lambda} \frac{K\binom{\Lambda}{t+\alpha}}{2^\Lambda \binom{\Lambda}{t}} + A_{t,\alpha} \right), \quad \forall t \in [0 : \Lambda] \tag{1.27}$$

where $A_{t,\alpha}$ is defined as

$$A_{t,\alpha} \coloneqq \frac{K}{|\mathcal{B}|} \binom{\Lambda - t}{\alpha} \left(1 - \frac{1}{\binom{t+\alpha}{\alpha}} \right). \tag{1.28}$$

We wish to point out that the meaning of the converses above is twofold. Indeed, under the assumption that the cache placement procedure is performed only once regardless of the connectivity, the results above are of interest not only because they lower bound the optimal average worst-case load in the agnostic scenario, but also because a careful comparison between the bounds above and the optimal performance in (1.24) allows to draw the insightful

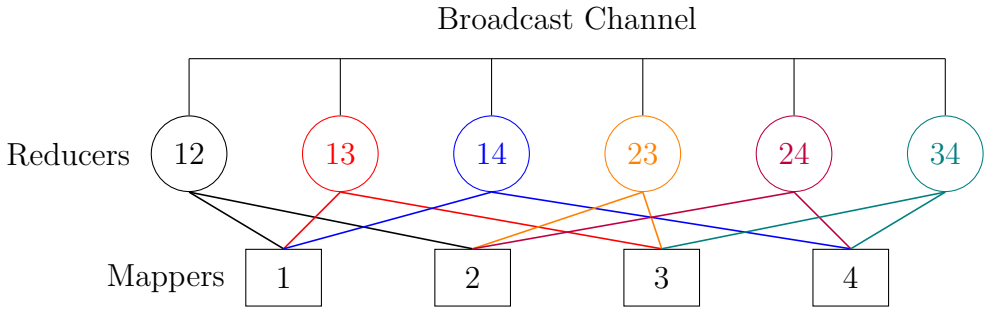

Figure 1.3: Multi-access distributed computing problem with $\Lambda = 4$ mappers and $K = 6$ reducers, where each reducer is connected exactly and uniquely to a subset of $\alpha = 2$ map nodes.

conclusion that the generalized combinatorial topology is among the good connectivities under the standard MAN cache placement. This leaves open the possibility that there might exist other good connectivities and so this brings to the fore the open question of whether, under the assumption of a MAN placement, the combinatorial topology is indeed the best possible topology. Further details will be provided in Chapter 6.

Multi-Access Distributed Computing

With the development of large-scale machine learning algorithms and applications relying heavily on large volumes of data, we are now experiencing an ever-growing need to distribute large computations across multiple computing nodes. Under the well-known MapReduce [58] framework, the overall computing process is typically split in three distinct phases, starting with the *map phase*, the *shuffle phase* and then the *reduce phase*. Nevertheless, several studies have shown that the aforementioned distributed map-shuffle-reduce approach comes with bottlenecks that may severely hinder the parallelization of computationally-intensive operations, one for all the communication bottleneck in the shuffle phase. This is why the authors in [52] introduced CDC as a novel information-theoretic framework that can yield lower communication loads during data shuffling, where this gain could be attributed to a careful and joint design of the map and the shuffle phases.

Nevertheless, the approach in [52] suffers from the already mentioned linear barrier given by the linear coding gain, which is inevitably further constrained in practical settings. Hence, we propose in Chapter 7 the new multi-access distributed computing (MADC) model, which can be considered as a non-trivial generalization of the original setting introduced in [52], and which entails *mappers* (map nodes) being connected to various *reducers* (reduce

nodes), and where these mappers and reducers are now distinct entities. More specifically, we study the MADC setting with combinatorial topology, where now mappers are connected to the reducers according to the multi-access topology already considered in Chapter 6 in the caching context (see Figure 1.3). We start our analysis of the problem by first neglecting the communication cost between mappers and reducers, and we propose a novel coded scheme that allows for efficient communication among reducers over the broadcast communication channel. In particular, we show that the optimal communication load $L^\star$ is upper bounded by L_{UB} which is a piecewise linear curve with corner points

$$(r, L_{\text{UB}}) = \left(r, \frac{\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\binom{r+\alpha}{r}-1\right)} \right), \quad \forall r \in [\Lambda - \alpha + 1]. \tag{1.29}$$

Then, we also provide for such setting an information-theoretic lower bound on the communication load, which reveals its novelty in the way we employ combinatorial arguments to build an entropy-based bound. We show that the converse is a piecewise linear curve with corner points

$$(r, L_{\text{LB}}) = \left(r, \frac{\binom{\Lambda}{r+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \right), \quad \forall r \in [\Lambda - \alpha + 1] \tag{1.30}$$

and that it is within a constant multiplicative gap of 1.5 from the achievable communication load guaranteed by the proposed coded scheme. We then proceed to also account for the download cost from mappers to reducers, and for such setting we introduce an additional mappers-to-reducers communication scheme which, together with the previous inter-reducer scheme, allows us to upper bound the optimal max-link communication load as

$$L_{\text{max-link,UB}} = \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} \frac{\tilde{a}^j_\star}{N}, \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}^j_\star}{N} \right) \tag{1.31}$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \ldots, \tilde{a}^\Lambda_\star)$ is the optimal solution to the linear program

$$\min_{\tilde{\boldsymbol{a}}_{\mathcal{M}}} \quad \frac{1}{2} \sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{1.32a}$$

$$\text{subject to} \quad \tilde{a}^j_{\mathcal{M}} \geq 0, \quad \forall j \in [\Lambda] \tag{1.32b}$$

$$\sum_{j\in[\Lambda]} \frac{\tilde{a}^j_{\mathcal{M}}}{N} = 1 \tag{1.32c}$$

$$\sum_{j\in[\Lambda]} j\frac{\tilde{a}^j_{\mathcal{M}}}{N} \leq r \tag{1.32d}$$

and where $\tilde{\boldsymbol{a}}_{\mathcal{M}} = (\tilde{a}^1_{\mathcal{M}}, \ldots, \tilde{a}^{\Lambda}_{\mathcal{M}})$ is the control variable. In addition, we also propose a novel converse bound which is given by

$$L_{\text{max-link,LB}} = \frac{1}{2}\sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_\star}{N} \tag{1.33}$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \ldots, \tilde{a}^{\Lambda}_\star)$ is the optimal solution to the linear program in (1.32). As it will be shown in Chapter 7, the achievable scheme and the converse bound above allow us to characterize the optimal max-link communication load within a constant multiplicative gap of 4.

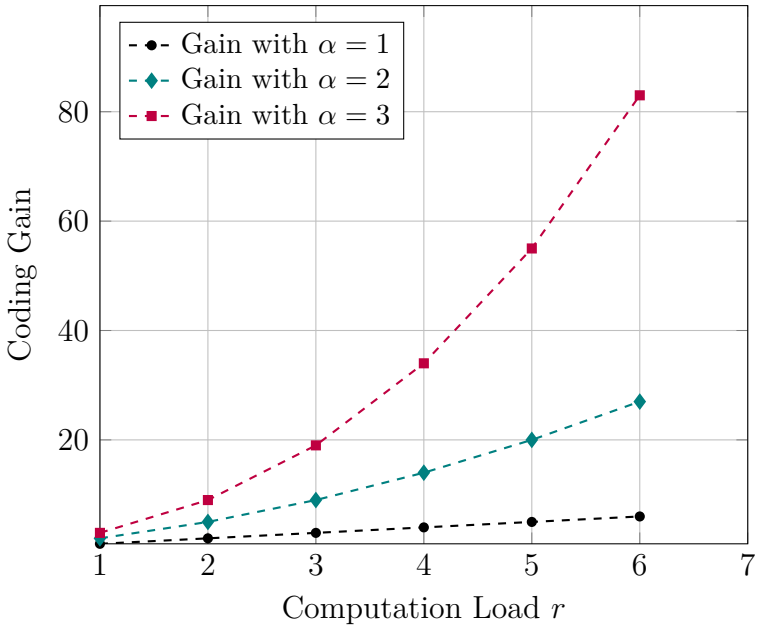

Figure 1.4: Comparison between the coding gain for different values of α as a function of the computation load r. We recall that $\alpha = 1$ corresponds to the original CDC framework.

We wish to point out that all the results above come with the realization that not only the coding gain guaranteed by the novel intra-reducer coded scheme is equal to

$$\binom{r+\alpha}{r} - 1 \tag{1.34}$$

which is much larger than the gain originally provided by the scheme in [52] (see Figure 1.4), but also that the number of reducers K is dramatically increased under fixed computational resources, where this naturally implies a massive parallelization and increased speedup factor in computing reduce functions. These observations further validate the fact that properly exploiting the structure in topology allows to provide stunning gains with respect to traditional solutions.

1.2.3 List of Publications

The work behind this thesis resulted in some publications. Concerning the first part of this manuscript, Chapter 2 is based on [J1], [C1], whereas Chapter 3 is based on unpublished results, Chapter 4 is based on [C2] and Chapter 5 is based also on unpublished results. For what concerns the second part of the thesis, Chapter 6 is based on [J2], [C3], while Chapter 7 is based on [J3]. The list of publications is provided in the following.

Journals

[J1] **F. Brunero** and P. Elia, "Unselfish coded caching can yield unbounded gains over selfish caching," *IEEE Trans. Inf. Theory*, Aug. 2022, early access. doi: 10/gqrs9z, pre-published.

[J2] **F. Brunero** and P. Elia, "Fundamental limits of combinatorial multi-access caching," *IEEE Trans. Inf. Theory*, Jul. 2022, early access. doi: 10/jbgr, pre-published.

[J3] **F. Brunero** and P. Elia, "Multi-access distributed computing," *IEEE Trans. Inf. Theory*, Jun. 2022, arXiv: 2206.12851 [cs.IT], submitted.

Conferences

[C1] **F. Brunero** and P. Elia, "Coded caching does not generally benefit from selfish caching," in *2022 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 1139–1144.

[C2] **F. Brunero** and P. Elia, "On the optimality of coded caching with heterogeneous user profiles," in *2022 IEEE Inf. Theory Workshop (ITW)*, 2022, forthcoming.

[C3] **F. Brunero** and P. Elia, "The exact load-memory tradeoff of multi-access coded caching with combinatorial topology," in *2022 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2022, pp. 1701–1706.

Part I

Exploring the Impact of Structure in Data

Chapter 2

A Negative Result on Selfish Caching Policies

In this chapter, we explore the interplay between coded caching and selfish caching. First, we propose a new selfish model which aims to calibrate the degree of separation between the interests of the different users. Then, we develop an information-theoretic converse bound on the optimal worst-case communication load under uncoded and selfish placement, where the bound proves that unselfish coded caching can far outperform selfish coded caching. As the converse offers some interesting insights on coding designs for selfish coded caching, we conclude the chapter by providing, for a class of demands, achievable schemes whose performance matches the expression of the converse.

2.1 Introduction

One key ingredient in using caches has commonly been the exploitation of the fact that some contents/files are more popular than others, and so are generally to be allocated more cache space [48], [49]. This inevitably introduces the consideration that different users may have different file preferences, which in turn brings to the fore the concept of *selfish caching* where simply users cache independently and selfishly only contents that they are interested in potentially consuming themselves [32]–[36].

In the traditional prefetching scenario, where emphasis is based heavily on bringing relevant content closer to each user, this idea of selfish caching brought about performance improvements [50], [51] in the form of higher local caching gains for each user. Nevertheless, only few works investigated the concept of selfishness in the context of coded caching.

2.1.1 Past Works on Heterogeneous User Profiles and Selfish Coded Caching

If on one hand a key ingredient in standard prefetching systems has commonly been the exploitation of the fact that some contents are more popular than others, and so are generally to be allocated more cache space, on the other we are just beginning to explore the interplay between coded caching, heterogeneous preferences and selfish caching, where by selfish caching we refer to caching policies where each user caches only contents from its own set of individual preferences.

Recent works have sought to explore this interplay. For example, in the context of coded caching with users having heterogeneous content preferences, the work in [32] analyzed the peak load of three different coded caching schemes that account for the user preferences, revealing occasional performance gains that are similarly dependent on the structure of these preferences. Related analysis appears in [33], now for the average load of the same schemes in [32]. On the other hand, the work in [35] focused on finding instances where unselfish coded caching outperforms selfish designs. This work nicely considered the performance of selfish coded caching in the context of heterogeneous file demand sets (FDSs), cleverly employing bounds to show that, for the case of $K = 2$ users and $N = 3$ files, unselfish designs strictly outperform selfish designs in terms of communication load, albeit only by a factor of up to 14 %. In addition, the notable work in [36] established, under the assumption of selfish and uncoded prefetching, the optimal average load for the case of $K = 2$ users and a variety of overlaps between the two users' profiles, also providing explicit prefetching schemes of a selfish nature. In a similar vein, the recent work in [31] considered the scenario where users are interested in a limited set of contents depending on their location. To the best of our knowledge, the above constitutes the majority of works on selfish coded caching.

2.1.2 An Adversarial Interplay Between Coded Caching and Selfish Caching

Our motivation to understand the interplay between coded caching and selfish caching comes not only from the fact that coded caching systems may indeed need to operate under some selfish legacy constraints[1], but also mainly from

[1]Here, we can think of a scenario where a server delivers via a bottleneck link content to caches, whose purpose is to bring content closer to the end user via dedicated non-interfering links. In such scenario, the delivery *to* the caches would benefit from a coded caching design, while the subsequent delivery *from* the caches would benefit from a selfish placement since the caches may target groups of users with potentially dissimilar interests.

the fact that there exists an interesting "adversarial" interplay between coded caching and selfish prefetching. To understand this a bit better, we recall that the main idea of coded caching is that it multicasts at any given time a linear combination of different contents desired by different users. This implies that any one receiver associated to a multicast message must be able to find in its cache all the undesired contents (subfiles) of that multicast message. This is achieved in [2] by means of a highly structured and coordinated content placement phase, where each user caches a small fraction of *every* file of a common library. This relationship between undesired and cached contents deteriorates when using selfish caching, simply because each receiver selfishly opts — based on its own preferences — to not cache some of these undesired files. However, these same undesired files may eventually appear as interference at that selfish receiver who will now not be able to "cache-out" this interference. At the same time, though, such selfish caching allows for a much more targeted placement of files such that each user can cache more of what it actually wants. Furthermore, such selfish scenario would correspond to a substantially smaller set of possible demands, which could conceivably be exploited to reduce the load.

To better understand the main challenge of the problem, let us consider the following simple scenario with $K = 3$ users and a library $\{A, B, C\}$ of $N = 3$ files. Let us now assume that user 1 is only interested in files from the file demand set $\mathcal{F}_1 = \{A, B\}$, user 2 only from the set $\mathcal{F}_2 = \{A, C\}$, and user 3 only from $\mathcal{F}_3 = \{B, C\}$. We know from [3], [4] that the MAN scheme in [2] guarantees the smallest worst-case load (under uncoded placement) during the delivery phase if each user caches a proper fraction M/N of *each* file, where M represents the memory (in number of files) that each user is equipped with. However, if we know in advance the sets $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$, do we need each user to cache also from files that are not of interest to minimize the number of bits to be transmitted? Is it convenient or disadvantageous to have each user k cache selfishly only from files in $\mathcal{F}_k$? What are the effects of selfish caching on coded caching?

2.1.3 Main Contributions

To understand this interplay between coded caching and selfish caching, we first propose a new selfish model which aims to calibrate the selfishness effect, by calibrating the degree of separation between the interests of the different users. Our so-called symmetric FDS structure not only aims to encapsulate this aspect of intersection of interests, but is also designed to reflect and accentuate the aforementioned adversarial relationship between the selfish placement and the ability to encode across users as one would expect in the

coded caching setting.

Subsequently, for the aforementioned symmetric FDS structure, we employ index coding arguments to derive an information-theoretic converse (lower bound) on the optimal worst-case communication load under the assumption of uncoded and selfish placement. This bound proves that **unselfish coded caching can far outperform selfish coded caching**. Indeed, the bound makes clear the fact that, while, as noted, selfish caching implies a much smaller set of possible demands (see Definition 2.2) as well as allows for a much more targeted placement of contents, these benefits will come, in our symmetric setting, at a heavy cost of fewer multicasting opportunities and a substantial loss in coding gain. The main contribution of our work is this information-theoretic converse, which allows us to draw the powerful conclusion that selfish coded caching can be, under some assumptions, rather detrimental.

This same converse offers some interesting insights on coding designs for selfish coded caching. While our converse now reveals that such designs, even if they are optimally constructed, would essentially never be able to provide good performance, these designs do pose an exceptionally interesting and challenging coding problem, which we address partially by providing, for a class of demands, achievable schemes whose performance matches the expression of the converse.

Remark 2.1. We wish to highlight that the focus of the chapter — which is to understand the effects of selfish caching policies — does not capture problems that arise from having a mismatch between the set of cached files and the set of demanded files[2]. Such problems remain, to date, open.

2.1.4 Chapter Outline

The rest of the chapter is organized as follows. The system model is presented in Section 2.2, where Section 2.2.2 offers a small motivating example that can help the reader appreciate the dynamics of selfish coded caching. Then, Section 2.3 presents the information-theoretic converse, whose proof in Section 2.4 is followed by a clarifying example. The proposed selfish coded caching placement is presented in Section 2.5 and so are the delivery designs for some sets of demands. Additional optimal schemes are presented in Section 2.6 for other sets of demands, whereas some of the proofs are relegated in Appendix A.

[2]We can have such mismatch when, for example, each user caches only from a subset of files, but then requests contents from the entire library, or when each user caches from the entire library, but only requests files from a limited subset of this library.

2.2 System Model

Similarly to the original scenario in [2], we consider the centralized caching scenario (see Figure 1.1) where one central server has access to a library containing N files of B bits each. This server is connected to K users through a shared error-free broadcast channel, and each user is equipped with a cache of size M files or, equivalently, MB bits.

2.2.1 The Symmetric (K, α, F) FDS Structure

To capture the interplay between coded caching and selfish caching, we propose an FDS structure that allows us to calibrate the degree of separation between the interests of the different users. To better understand this structure and generally to better understand the concept of an FDS, let us briefly consider a simplified toy example.

Example 2.1. Consider a downlink scenario with $K = 3$ users and a library $\{A, B, C, D, E, F\}$ of $N = 6$ files[3]. Let us now assume that user 1 is only interested in potentially consuming files from the file demand set $\mathcal{F}_1 = \{A, B, C, D\}$, user 2 only from the set $\mathcal{F}_2 = \{A, B, E, F\}$, and user 3 only from $\mathcal{F}_3 = \{C, D, E, F\}$. In this setting, each user is interested in a fraction $2/3$ of the library, so for example user 1 has no interest in ever consuming the files in $\{A, B, C, D, E, F\} \setminus \mathcal{F}_1 = \{E, F\}$. Similarly, each file is of interest to the same fraction $2/3$ of users, so for example file A is only of interest to user 1 and user 2.

For such a setting, we wish to understand the performance of selfish coded caching where each user caches only contents from its own FDS. We proceed with the formal definition of the FDS structure. We note that below an FDS will be defined as a collection of file *classes*, rather than just a collection of files. This allows for more generality and we believe it also better reflects how user preferences are often categorized.

Definition 2.1 (The Symmetric (K, α, F) FDS Structure)**.** For $\alpha \in [K]$ and for $F \in \mathbb{N}^+$, the symmetric (K, α, F) FDS structure assumes an N-file library $\{\mathcal{W}_\mathcal{S} : \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha\}$ to be a collection of disjoint file classes $\mathcal{W}_\mathcal{S} = \{W_{f,\mathcal{S}} : f \in [F]\}$, with each class $\mathcal{W}_\mathcal{S}$ consisting of F different files. In this setting, each user $k \in [K]$ has a file demand set

$$\mathcal{F}_k = \{\mathcal{W}_\mathcal{S} : \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha, k \in \mathcal{S}\} \tag{2.1}$$

which describes the files this user is potentially interested in.

[3]Such files can be movies, different episodes of a TV show, YouTube videos, etc.

As the above says, the library is split into $C = \binom{K}{\alpha}$ disjoint classes of files, corresponding to a total of $N = FC = F\binom{K}{\alpha}$ files. Then, there are K FDSs, one for each user, and each file class is identified by an α-tuple $\mathcal{S}$ that tells us which α users are interested in this class[4]. The above also says that each user k is interested in its own FDS $\mathcal{F}_k$ of $|\mathcal{F}| = |\mathcal{F}_k| = F\binom{K-1}{\alpha-1}$ files, and this nicely lets us calibrate the fraction

$$\frac{|\mathcal{F}|}{N} = \frac{F\binom{K-1}{\alpha-1}}{F\binom{K}{\alpha}} = \frac{\alpha}{K} \tag{2.2}$$

of the total library that each user is interested in, where the imposed symmetry also yields a fraction $\delta \coloneqq \alpha/K$ of users interested in any one specific file. Finally, we note that $\alpha = 1$ corresponds to the trivial scenario where there is no intersection between the user interests, while $\alpha = K$ corresponds to the traditional unselfish scenario where a common library of $N = F$ files[5] is of interest to every user.

Table 2.1: Important parameters for the symmetric (K, α, F) FDS structure

Total FDSs	K
Total File Classes	$\binom{K}{\alpha}$
File Classes per FDS	$\binom{K-1}{\alpha-1}$
Fraction of Users Interested in a File	α/K
Files per Class	F
Total Files	$F\binom{K}{\alpha}$
Files per FDS	$F\binom{K-1}{\alpha-1}$
Fraction of Files of Interest to a User	$\|\mathcal{F}\|/N$

The following two examples can help familiarize the reader with the notation.

Example 2.2 (The Symmetric $(4, 2, 1)$ FDS Structure)**.** Consider the symmetric $(K, \alpha, F) = (4, 2, 1)$ structure. There are $C = \binom{K}{\alpha} = 6$ file classes, which

[4]In other words, each file belongs to α FDSs. In particular, each file in class $\mathcal{W}_\mathcal{S}$ is of interest to the α users in $\mathcal{S}$. Hence, if $k \in \mathcal{S}$, then the F files in $\mathcal{W}_\mathcal{S}$ are in $\mathcal{F}_k$ and are thus of interest to user k. Finally, under our simplifying assumption that each user has its own FDS, α also describes the number of users interested in any one specific file.

[5]In this case we assume $F \geq K$.

are given by $\mathcal{W}_{12}, \mathcal{W}_{13}, \mathcal{W}_{14}, \mathcal{W}_{23}, \mathcal{W}_{24}, \mathcal{W}_{34}$, and where[6] each class consists of $F = 1$ file. This implies a library $\{W_{1,12}, W_{1,13}, W_{1,14}, W_{1,23}, W_{1,24}, W_{1,34}\}$ of $N = 6$ files. In the above, $W_{1,12}$ simply represents the first (and, in this case, the only) file in class $\mathcal{W}_{12}$. The $K = 4$ FDSs take the form

$$\mathcal{F}_1 = \{W_{1,12}, W_{1,13}, W_{1,14}\} \tag{2.3}$$
$$\mathcal{F}_2 = \{W_{1,12}, W_{1,23}, W_{1,24}\} \tag{2.4}$$
$$\mathcal{F}_3 = \{W_{1,13}, W_{1,23}, W_{1,34}\} \tag{2.5}$$
$$\mathcal{F}_4 = \{W_{1,14}, W_{1,24}, W_{1,34}\} \tag{2.6}$$

where we recall that, for each file $W_{1,\mathcal{S}}$, the label $\mathcal{S}$ represents the FDSs the file belongs to. For example, file $W_{1,23}$ belongs to $\mathcal{F}_2$ and $\mathcal{F}_3$, and is thus of interest to user 2 and user 3. Finally we see that each user is interested in a fraction $|\mathcal{F}|/N = 0.5$ of the library, i.e., in 50 % of the library, and that each file is of interest to a fraction $\delta = \alpha/K = 0.5$ of the users.

Example 2.3 (The Symmetric $(4, 3, 2)$ FDS Structure)**.** Consider the symmetric $(K, \alpha, F) = (4, 3, 2)$ structure. There is a total of $C = \binom{K}{\alpha} = 4$ classes $\mathcal{W}_{123}, \mathcal{W}_{124}, \mathcal{W}_{134}, \mathcal{W}_{234}$ and $N = 8$ files: $W_{1,123}$ and $W_{2,123}$ from class $\mathcal{W}_{123}$, then $W_{1,124}$ and $W_{2,124}$ from class $\mathcal{W}_{124}$, and so on. The K FDSs take the form

$$\mathcal{F}_1 = \{\mathcal{W}_{123}, \mathcal{W}_{124}, \mathcal{W}_{134}\} \tag{2.7}$$
$$\mathcal{F}_2 = \{\mathcal{W}_{123}, \mathcal{W}_{124}, \mathcal{W}_{234}\} \tag{2.8}$$
$$\mathcal{F}_3 = \{\mathcal{W}_{123}, \mathcal{W}_{134}, \mathcal{W}_{234}\} \tag{2.9}$$
$$\mathcal{F}_4 = \{\mathcal{W}_{124}, \mathcal{W}_{134}, \mathcal{W}_{234}\} \tag{2.10}$$

where we see that each FDS consists of $2 \times 3 = 6$ files. For example, user 1 is interested in the files contained in the FDS $\mathcal{F}_1 = \{W_{f,\mathcal{S}} : \mathcal{S} \subseteq [4], |\mathcal{S}| = 3, 1 \in \mathcal{S}, f \in [2]\}$, user 2 is interested in the files contained in the FDS $\mathcal{F}_2 = \{W_{f,\mathcal{S}} : \mathcal{S} \subseteq [4], |\mathcal{S}| = 3, 2 \in \mathcal{S}, f \in [2]\}$, and so on. By calculating $|\mathcal{F}|/N = \alpha/K = 3/4$, we can verify that each user is interested in 75 % of the library, and each file is of interest to 75 % of the users.

Deviating from standard notation practices, we will use the double-index notation $W_{f_k,\mathcal{D}_k}$ to denote the file requested by user k. Consequently, to describe the entire demand set, we will now be needing two vectors $\boldsymbol{d} = (\mathcal{D}_1, \ldots, \mathcal{D}_K)$ and $\boldsymbol{f} = (f_1, \ldots, f_K)$. In addition, the FDS structure automatically implies restrictions in the set of possible demand vectors. For

[6]We will often omit braces and commas when indicating sets, such that for example $W_{\{1,2\}}$ may be written as W_{12}.

instance, going back to Example 2.3, any demand with $\boldsymbol{d} = (234, 123, 123, 234)$ is not valid, because $\{1\} \notin \mathcal{D}_1 = \{2, 3, 4\}$, i.e., because file $W_{f_1,234}$ is not in $\mathcal{F}_1$ and so would never be demanded by user 1. On the other hand, any demand with $\boldsymbol{d} = (124, 123, 123, 234)$ is valid because $k \in \mathcal{D}_k$ for each $k \in [K]$. The set of valid demands as well as placement constraints that define selfish coded caching are now stated below.

Definition 2.2 (Selfish Coded Caching With Uncoded Placement)**.** In selfish coded caching, a demand defined by the vectors $\boldsymbol{d} = (\mathcal{D}_1, \ldots, \mathcal{D}_K)$ and $\boldsymbol{f} = (f_1, \ldots, f_K)$ is said to be *valid* if and only if

$$k \in \mathcal{D}_k, \quad \forall k \in [K]. \tag{2.11}$$

Further, a cache placement is *selfish* when it guarantees that a subfile of $W_{f,\mathcal{S}}$ can be cached at user k only if $k \in \mathcal{S}$, whereas it is *uncoded* if it satisfies Definition 1.1.

Remark 2.2. We acknowledge that the aforementioned symmetric FDS structure can be more restrictive than the (very few) existing FDS structures in the literature. For instance, the works in [32]–[34] considered the FDS structure where each file is of interest to either groups of users or all users in the system. Nevertheless, our aim is to explore the effect of selfish caching, and what we show is that an important and general instance of selfish caching, embodied by the considered symmetric FDS structure, yields unbounded performance deterioration compared to the unconstrained (unselfish) case. Hence, this instance allows us to provide, in a unified manner, the main contribution of our work, which is the clear conclusion that indeed selfish caching can be unboundedly detrimental. The symmetric FDS structure serves as a proving step toward deriving this conclusion. At the same time, this same chosen FDS structure captures core principles of realistic file demand sets, like for example the amount of intersection between different such sets, where this intersection can be calibrated at will by the parameter α.

2.2.2 Understanding the Dynamics of Selfish Coded Caching With an Example for the $(K, \alpha, F) = (5, 4, 1)$ Structure

Let us consider a small motivating example that can help the reader appreciate the dynamics of symmetrically selfish coded caching. We will first suggest a selfish cache placement scheme that will be justified in Section 2.5, and we will then present the delivery and decoding process for a class of valid circular demands. The corresponding load that will be achieved here will in

fact be matched by the converse of the next section, consequently proving that in our example our delivery is optimal and the converse tight.

We here consider the $(K, \alpha, F) = (5, 4, 1)$ scenario, where each cache is of size $M = 2$ corresponding to the case of $t = 2$. In our scenario there are $C = \binom{K}{\alpha} = 5$ file classes $\mathcal{W}_{1234}, \mathcal{W}_{1235}, \mathcal{W}_{1245}, \mathcal{W}_{1345}, \mathcal{W}_{2345}$, and a total of $N = FC = 5$ library files. For simplicity, we will exploit the fact that $F = 1$ by slightly abusing notation such that, in this early example only, the library of $N = 5$ files will be denoted as $\{W_{1234}, W_{1235}, W_{1245}, W_{1345}, W_{2345}\}$. At this point, the 5 FDSs take the form

$$\mathcal{F}_1 = \{W_{1234}, W_{1235}, W_{1245}, W_{1345}\} \tag{2.12}$$
$$\mathcal{F}_2 = \{W_{1234}, W_{1235}, W_{1245}, W_{2345}\} \tag{2.13}$$
$$\mathcal{F}_3 = \{W_{1234}, W_{1235}, W_{1345}, W_{2345}\} \tag{2.14}$$
$$\mathcal{F}_4 = \{W_{1234}, W_{1245}, W_{1345}, W_{2345}\} \tag{2.15}$$
$$\mathcal{F}_5 = \{W_{1235}, W_{1245}, W_{1345}, W_{2345}\}. \tag{2.16}$$

Placement phase

The cache placement will follow a selfish adaptation of the MAN scheme. First each file is split into $\binom{\alpha}{t} = \binom{4}{2} = 6$ non-overlapping subfiles as

$$W_{1234} = \{W_{1234,12}, W_{1234,13}, W_{1234,14}, W_{1234,23}, W_{1234,24}, W_{1234,34}\} \tag{2.17}$$
$$W_{1235} = \{W_{1235,12}, W_{1235,13}, W_{1235,15}, W_{1235,23}, W_{1235,25}, W_{1235,35}\} \tag{2.18}$$
$$W_{1245} = \{W_{1245,12}, W_{1245,14}, W_{1245,15}, W_{1245,24}, W_{1245,25}, W_{1245,45}\} \tag{2.19}$$
$$W_{1345} = \{W_{1345,13}, W_{1345,14}, W_{1345,15}, W_{1345,34}, W_{1345,35}, W_{1345,45}\} \tag{2.20}$$
$$W_{2345} = \{W_{2345,23}, W_{2345,24}, W_{2345,25}, W_{2345,34}, W_{2345,35}, W_{2345,45}\} \tag{2.21}$$

and then the cache Z_k of each user $k \in [5]$ is filled as

$$Z_k = \{W_{\mathcal{S},\mathcal{T}} : \mathcal{S} \subseteq [5], |\mathcal{S}| = 4, \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = 2, k \in \mathcal{T}\}. \tag{2.22}$$

For example, user 1 would have to cache parts only from the files in the set $\{W_{1234}, W_{1235}, W_{1245}, W_{1345}\}$ in order to abide by the selfish constraint, and then, to abide by the cache size constraint, user 1 would cache subfiles labeled by $\{12, 13, 14\}$. Similarly, user 2 would cache only from the files in the set $\{W_{1234}, W_{1235}, W_{1245}, W_{2345}\}$, and only the subfiles labeled by $\{12, 23, 24\}$, and so on.

Delivery phase

The delivery takes place as soon as the requests of the users are revealed. Consider the demand $\boldsymbol{d}_1 = (1234, 2345, 1345, 1245, 1235)$. A schematic of this

demand is given by means of the graph in Figure 2.1. This graph, which we refer to as the *FDS request graph*, is a directed graph where each vertex is a user and where there is an edge from user k_1 to user k_2 if $W_{\mathcal{D}_{k_1}} \in \mathcal{F}_{k_2}$. This graph represents at a high level, for each given demand vector $\boldsymbol{d}$, the interplay between the users' interests.

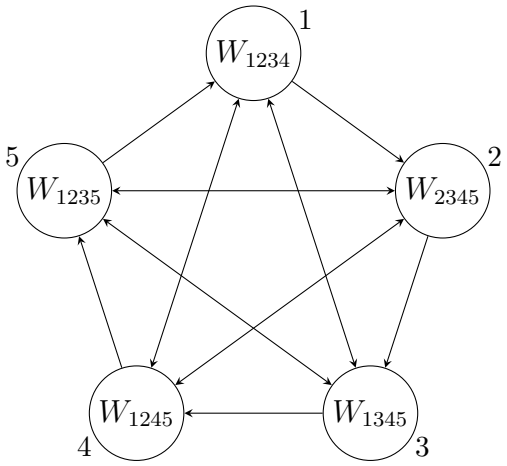

Figure 2.1: FDS request graph for the $(K, \alpha, F) = (5, 4, 1)$ FDS structure and the demand $\boldsymbol{d}_1 = (1234, 2345, 1345, 1245, 1235)$.

As a consequence of the aforementioned cache placement, each user does not cache (and consequently desires) a total of $\binom{\alpha-1}{t} = \binom{3}{2} = 3$ subfiles for its demanded file. Hence, given the demand $\boldsymbol{d}_1$, the desired subfiles are given as follows.

- User 1 desires the subfiles $W_{1234,23}$, $W_{1234,24}$ and $W_{1234,34}$.
- User 2 desires the subfiles $W_{2345,34}$, $W_{2345,35}$ and $W_{2345,45}$.
- User 3 desires the subfiles $W_{1345,14}$, $W_{1345,15}$ and $W_{1345,45}$.
- User 4 desires the subfiles $W_{1245,12}$, $W_{1245,15}$ and $W_{1245,25}$.
- User 5 desires the subfiles $W_{1235,12}$, $W_{1235,13}$ and $W_{1235,23}$.

One key aspect for achieving optimality is the utilization of specifically structured linear combinations of multicast messages, where this structure accepts the following interesting interpretation. These linear combinations effectively allow multicast messages to be used not only to deliver desired content to users, but also to deliver undesired content that can be used as side information to "bridge" the gaps left by the selfish placement. In essence,

each transmission now delivers desired content while also disseminating side information that can be used to create *cliques*. To see this, let us consider the following sequence of XORs

$$X_1 = W_{1345,14} \oplus W_{1234,24} \oplus W_{1245,12} \tag{2.23}$$
$$X_2 = W_{2345,35} \oplus W_{1235,13} \oplus W_{1345,15} \tag{2.24}$$
$$X_3 = W_{1345,14} \oplus W_{2345,35} \oplus W_{1234,23} \tag{2.25}$$
$$X_4 = W_{1234,34} \oplus W_{1245,15} \tag{2.26}$$
$$X_5 = W_{2345,45} \oplus W_{1235,12} \tag{2.27}$$
$$X_6 = W_{2345,34} \oplus W_{1245,25} \tag{2.28}$$
$$X_7 = W_{1345,45} \oplus W_{1235,23} \tag{2.29}$$

transmitted one after the other. Recalling that each file is split into 6 non-overlapping subfiles, we know that each XOR has size $|X_i| = B/6$ for each $i \in [7]$.

By using its own cache, each user can now decode its own desired content as follows.

- User 1 can recover its desired subfiles from X_1, $X_2 \oplus X_3$ and X_4.
- User 2 can recover its desired subfiles from $X_1 \oplus X_3$, X_5 and X_6.
- User 3 can recover its desired subfiles from X_2, X_3 and X_7.
- User 4 can recover its desired subfiles from X_1, X_4 and X_6.
- User 5 can recover its desired subfiles from X_2, X_5 and X_7.

For example, in the above, user 1 needs $W_{2345,35}$ to correctly decode its desired $W_{1234,23}$ from X_3, whereas user 2 needs $W_{1345,14}$ to correctly decode $W_{2345,35}$ always from X_3. The act of "passing" subfiles $W_{2345,35}$ and $W_{1345,14}$ to user 1 and user 2 with X_2 and X_1, respectively, allows the creation of a clique between user 1, user 2 and user 3. This clique is exploited by creating the XOR X_3. The corresponding communication load is equal to $R(t = 2) = |X|/B = 7/6$, which will be met by the converse.

We wish to point out that, intuitively, the set of multicast messages above is constructed around the idea that each transmission serves not only to deliver desired content, but also, at the same time, to form cliques that will be exploited in future transmissions. Indeed, in the above we chose $t + 1 = 3$ pivotal users (i.e., user 1, user 2 and user 3) which represent an *almost complete clique* in the side information graph of the induced index coding

problem. All the effort consisted then in trying to "deliver" side information[7] to "bridge" the gaps left by the selfish placement, while delivering desired content to users. This was done in X_1 and X_2, where the subfiles $W_{1345,14}$ and $W_{2345,35}$ were carefully "delivered" to user 2 and user 1, respectively, without interfering with users to which X_1 and X_2 are delivering desired information. This interpretation related to the creation of cliques is a crucial part of the dynamics of the problem that we are considering.

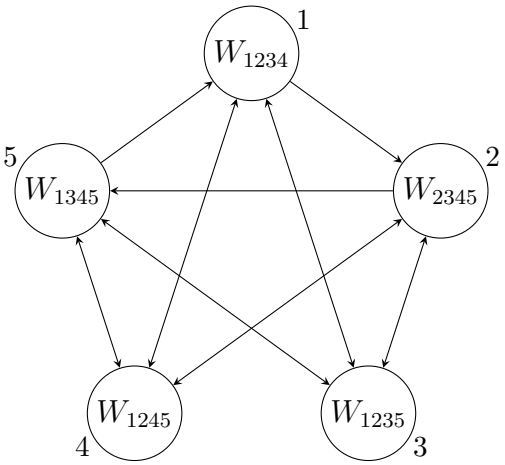

Figure 2.2: FDS request graph for the $(K, \alpha, F) = (5, 4, 1)$ FDS structure and the demand $\boldsymbol{d}_2 = (1234, 2345, 1235, 1245, 1345)$.

Now, let us consider the demand vector $\boldsymbol{d}_2 = (1234, 2345, 1235, 1245, 1345)$ with its corresponding FDS request graph that is shown in Figure 2.2. Since the graphs in Figure 2.1 and in Figure 2.2 are non-isomorphic[8], the demand $\boldsymbol{d}_2$ accepts a different delivery solution[9] than that for demand $\boldsymbol{d}_1$. Such phenomenon does not happen in the standard coded caching scenario, where indeed each demand would result in the same FDS request graph (see Figure 2.3), which is always complete[10]. In such unselfish scenario where each file

[7]Notice that we use "deliver" in quotes when referring to undesired subfiles because users do not need to actually decode subfiles which are not desired. Indeed, as explained above, it is more a matter of aligning interference, so that users can directly take linear combinations of the multicast messages to cancel out interference terms. Still, the interpretation related to the creation of cliques can give insights on how to construct multicast messages.

[8]This can be concluded by noticing that the graph in Figure 2.1 contains 5 bidirectional edges, whereas the graph in Figure 2.2 has 6 bidirectional edges.

[9]Having two non-isomorphic problems here implies that the delivery for the second problem cannot be derived from that of the first problem by a simple relabeling of the users.

[10]A complete graph is a graph where every node is connected to every other node.

is assumed to be of interest to all users, every user in the FDS request graph is connected to every other user, independently of the requested files. Hence, in the unselfish scenario, having a fixed FDS request graph for every demand allows for an identical delivery procedure for any demand. This seems to be a crucial differentiating aspect between selfish and unselfish coded caching.

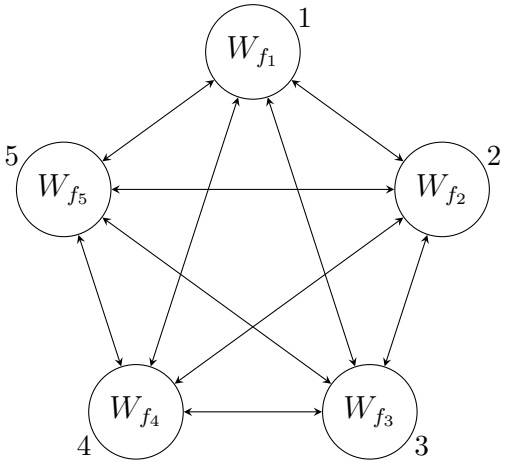

Figure 2.3: FDS request graph for any demand in the standard (unselfish) MAN scenario with $K = 5$ users and $N = 5$ files labeled as W_f with $f \in [5]$. In this case the demand is identified by the vector $\boldsymbol{f} = (f_1, f_2, f_3, f_4, f_5)$, where user $k \in [5]$ requests file W_{f_k}. This graph is complete. Hence, here the ability to create cliques of subfiles is only limited by t, and is not affected at all by the specific demand.

2.3 Main Results

Let us recall that each user is interested in its own FDS, and that each FDS only represents a fraction $|\mathcal{F}|/N$ of the library. In the general unselfish scenario, a portion $(1-|\mathcal{F}|/N)$ of each user's cache would be filled with content that would never be requested by that user. Such a non-selfish scheme would relinquish local caching gain for the benefit of being able to encode across all combinations of users. Under the basic clique-based approach in the MAN scheme, we are presented with a trade-off between local caching gain and coding gain, where the latter seems to be more desirable. Are there, though, other coding techniques that manage to harvest an abundance of coding opportunities, which are usually associated to the standard coded caching approach, exploiting the existence of a more targeted set of demands, while capitalizing on the increased local caching gain brought about by a selfish

variant? If not, then what is the amount of coding gain that can be harvested while maintaining selfish caching? These are the questions addressed by our information-theoretic converse that lower bounds the optimal worst-case load under uncoded and selfish cache placement, which will be denoted by $R^{\star}_{\mathrm{u,s}}$.

2.3.1 Theorem Statement

The converse bound employs the index coding techniques of [3] that proved the optimality of the MAN scheme under the constraint of uncoded cache placement. Our main challenge will be to account for the presence of different profiles of interest, adapting consequently the index coding approach to reflect the (K, α, F) FDS structure proposed in the previous section. The converse bound presented here shows that adding the selfish cache placement constraint implies a higher optimal communication load compared to the unselfish scenario. The result is stated in the following theorem.

Theorem 2.1 (Converse Bound for Selfish Coded Caching Under Uncoded Prefetching)**.** *Under the assumption of uncoded and selfish cache placement, and given the (K, α, F) FDS structure, the optimal worst-case communication load $R^{\star}_{\mathrm{u,s}}$ is lower bounded by R_{LB} which is a piecewise linear curve with corner points*

$$(M, R_{\mathrm{LB}}) = \left(t\frac{N}{K}, \frac{\binom{\alpha}{t+1} + (K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}} \right), \quad \forall t \in [0:\alpha] \tag{2.30}$$

corresponding to

$$R_{\mathrm{LB}} = \frac{K(1-\gamma_\alpha)}{K\gamma + 1}\Big[(K-\alpha)\gamma + 1\Big] \tag{2.31}$$

where $\gamma \coloneqq M/N$ and $\gamma_\alpha \coloneqq M/|\mathcal{F}| = \gamma K/\alpha$.

Proof. We provide the proof of the converse in Section 2.4.1. In Section 2.4.2 we also present an example that aims to help the reader better understand the construction of the outer bound. □

2.3.2 Comments on the Converse Bound

The bound reveals some interesting insights. Before discussing these insights, let us quickly recall that, in our scenario, the integer value t is upper bounded by α, since any $t \geq \alpha$ would imply zero communication load.

Comparison With MAN

The following compares, for any F, the optimal load $R^{\star}_{\mathrm{u,s}}(t)$ of selfish coded caching with that of the unselfish (MAN) scenario[11].

Corollary 2.1.1. *Given the symmetric (K, α, F) FDS structure and $\alpha \in [K-1]$, the converse reveals that*

$$\frac{R^{\star}_{\mathrm{u,s}}(t)}{R_{\mathrm{MAN}}(t)} \geq 1, \quad \forall t \in [0 : \alpha - 1] \tag{2.32}$$

$$\frac{R^{\star}_{\mathrm{u,s}}(t)}{R_{\mathrm{MAN}}(t)} > 1, \quad \forall t \in [\alpha - 2] \tag{2.33}$$

which says that, in the non-trivial range $t \in [0 : \alpha - 1]$, selfish coded caching is not better than unselfish coded caching, which instead, in the non-extremal points of t and under uncoded placement, strictly outperforms any implementation of selfish coded caching. When $\alpha = K$ and $F \geq K$, the converse expression naturally matches that of unselfish coded caching.

Proof. The proof can be found in Appendix A.1, while a graphical comparison can be found in Figure 2.4. □

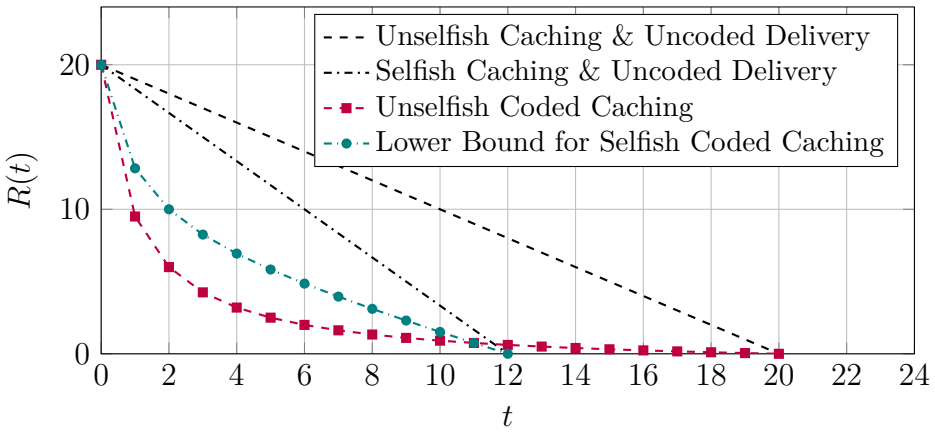

Figure 2.4: Comparison between selfish caching and unselfish caching for the $(20, 12, F)$ FDS structure.

[11]The comparison between the selfish and unselfish scenarios is made easy by the fact that the t values (i.e., the integer points corresponding to the memory-axis of the memory-load trade-off) in the two scenarios coincide.

Selfish Local Caching Gain and Coding Gain

Since selfish caching places the constraint that each user k can only cache from its own FDS $\mathcal{F}_k$, one key aspect of such selfish caching is that it brings about an increase of the effective normalized cache size for each user. Indeed, whereas in the unselfish scenario each user can cache a fraction

$$\gamma = \frac{M}{N} = \frac{t}{K} \tag{2.34}$$

of each file of possible interest, in the selfish scenario this fraction is elevated to a larger

$$\gamma_\alpha = \frac{M}{|\mathcal{F}|} = \frac{t}{\alpha} = \gamma \frac{K}{\alpha} \tag{2.35}$$

which in turn implies a larger local caching gain. Hence, in the presence of a relatively small α, selfish caching implies a sizeable increase in the effective normalized cache size $\gamma_\alpha = \gamma K/\alpha$, which in turn implies a much larger local caching gain.

On the other hand, the converse reveals that a smaller α implies a substantial reduction in the coding gain offered by selfish coded caching. To compare coding gains, we first recall that the coding gain in the original unselfish scenario takes the form

$$\frac{R^{\mathrm{u}}}{R_{\mathrm{MAN}}} = K\gamma + 1, \tag{2.36}$$

where $R^{\mathrm{u}} = K(1-\gamma)$ is the load for uncoded delivery. As previously stated, this coding gain $K\gamma + 1$ describes the speedup factor over the uncoded case. To reflect this same speedup in the selfish scenario, we must consider that the corresponding load in the uncoded scenario takes the form $R_{\mathrm{s}}^{\mathrm{u}} = K(1-\gamma_\alpha)$. With this in place, the converse reveals that the optimal coding gain of selfish coded caching is upper bounded as

$$G^\star \leq \frac{R_{\mathrm{s}}^{\mathrm{u}}}{R_{\mathrm{LB}}} = \frac{K\gamma + 1}{(K-\alpha)\gamma + 1} \tag{2.37}$$

where the value

$$D \coloneqq (K-\alpha)\gamma + 1 \tag{2.38}$$

represents the guaranteed deterioration in the coding gain when we choose to cache selfishly. Indeed, if we consider the non-trivial range $\alpha \in [2 : K-1]$, we have $D > 1$ and consequently $G < K\gamma + 1$ for $\gamma > 0$. We can see that — for fixed K and γ — this deterioration D increases with decreasing α, reflecting the fact that the closer the (K, α, F) FDS structure is to the standard MAN scenario, the smaller this deterioration D is.

An important observation though is that the coding gain of selfish coded caching does not scale with K. This is described in the following corollary.

Corollary 2.1.2. *For any fixed ratio $\delta = \alpha/K < 1$ the coding gain of selfish caching does not scale as K increases, and it is instead bounded as*

$$G^\star < \frac{1}{1-\delta}. \tag{2.39}$$

Proof. The proof can be found in Appendix A.2. □

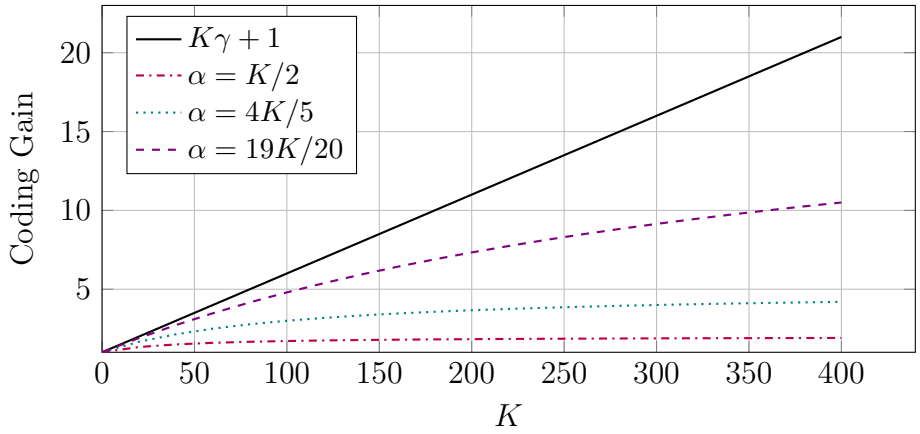

Figure 2.5: Plot of different coding gains G for varying values of K and α for the (K, α, F) FDS structure when $\gamma = 1/20$.

We can see in Figure 2.5 the comparison between different coding gains for varying values of K and α when the normalized cache size γ is fixed. As mentioned, smaller values of α correspond to much smaller coding gains. As stated in Corollary 2.1.2, each curve is upper bounded[12] by $1/(1-\delta)$.

Remark 2.3. At this point, we ought to point out that our choice of having a fully symmetric FDS structure may indeed be an overly penalizing condition. However, this choice exemplifies the mechanisms and effects that come about when selfishness is considered. This same choice nicely offers a crisp method for calibrating the intersection between the interests of the different users, taking us from a scenario where the intersection is minimal, to scenarios ever closer to the original MAN setting where the interests are identical. As suggested before, this FDS structure is a sufficient proving step for drawing, in a unified manner, the conclusion that indeed selfish coded caching *can* be quite detrimental.

[12]When $\alpha = 1$, it holds that $G^\star = 1$, since in such case uncoded delivery is optimal.

2.4 Proof of Theorem 2.1

The derivation of the converse makes extensive use of the connection between caching and index coding. This connection was made in [2] and was successfully used in [3] to derive the optimal performance of the unselfish scenario.

We quickly recall that an index coding problem [59]–[62] consists of a server wishing to deliver N' independent messages to K' users via a basic bottleneck link. Each user $k \in [K']$ has its own *desired message set* $\mathcal{M}_k \subseteq [N']$, and has knowledge of its own *side information set* $\mathcal{A}_k \subseteq [N']$. Let M_i be the message i in the set $[N']$. Then, the index coding problem is typically described by its *side information graph* in the form of a directed graph, where each vertex is a message and where there is an edge from M_i to M_j if M_i is in the side information set of the user requesting M_j. The derivation of our converse will use the following well-known result from [63, Corollary 1].

Lemma 2.1 ([63, Corollary 1])**.** *In an index coding problem with N' messages M_i for $i \in [N']$, the minimum number of transmitted bits ρ is bounded as*

$$\rho \geq \sum_{i \in \mathcal{J}} |M_i| \tag{2.40}$$

for any acyclic subgraph $\mathcal{J}$ of the problem's side information graph.

Before proceeding with the main proof, we also recall that under the (K, α, F) FDS structure we have $\{\mathcal{W}_\mathcal{S} : \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha\}$, where $\mathcal{W}_\mathcal{S} = \{W_{f,\mathcal{S}} : f \in [F]\}$ is a class of files. We further recall that there are $C = \binom{K}{\alpha}$ classes of files and $N = FC = F\binom{K}{\alpha}$ files. Additionally, we recall that the FDS of each user $k \in [K]$ is given by

$$\mathcal{F}_k = \{\mathcal{W}_S : \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha, k \in \mathcal{S}\} \tag{2.41}$$

that each file has size B bits, and that each user is equipped with a cache of size MB bits. Finally, let us remember that we are interested in the non-trivial range[13] $\alpha \in [2 : K-1]$ and in the range $t \in [0 : \alpha]$. Indeed, for $t = \alpha$ we have $M = \alpha N/K = F\binom{K-1}{\alpha-1}$, so the point $(M, R) = (\alpha N/K, 0)$ is trivially achievable as a consequence of each user being able to store the entirety of its FDS.

[13]When $\alpha = 1$ the proof is trivial, since for such case we have only two integer points corresponding to $t \in \{0, 1\}$: when $t = 0$ the load is equal to K, and when $t = 1$ each user has enough memory to cache entirely its own FDS and the load is equal to 0. Then, the case $\alpha = K$ and $F \geq K$ is equivalent to the standard (unselfish) MAN scenario, which was already considered in [3].

2.4.1 Main Proof

The first step toward the converse consists of splitting each file in a generic manner into a maximum of $2^{|\mathcal{S}|} = 2^{\alpha}$ disjoint subfiles as

$$W_{f,\mathcal{S}} = \{W_{f,\mathcal{S},\mathcal{T}} : \mathcal{T} \subseteq \mathcal{S}\} \tag{2.42}$$

for each $\mathcal{S} \subseteq [K]$ with $|\mathcal{S}| = \alpha$ and for each $f \in [F]$, where $W_{f,\mathcal{S},\mathcal{T}}$ is the subfile of $W_{f,\mathcal{S}}$ cached exactly and only by users in $\mathcal{T}$. As already mentioned in Definition 2.2, splitting each file in this way satisfies the uncoded and selfish cache placement constraint, since $\mathcal{T} \subseteq \mathcal{S}$ and $\mathcal{W}_{\mathcal{S}} \in \mathcal{F}_k$ for each $k \in \mathcal{S}$.

Constructing the Index Coding Problem

We now proceed by making the connection to index coding. Hence, let us consider the index coding problem with $K' = K$ users and $N' = K2^{\alpha-1}$ messages, such that for any demand, identified by the vectors $\boldsymbol{d} = (\mathcal{D}_1, \dots, \mathcal{D}_K)$ and $\boldsymbol{f} = (f_1, \dots, f_K)$, the desired message set and the side information set are respectively given by

$$\mathcal{M}_k = \{W_{f_k,\mathcal{D}_k,\mathcal{T}} : \mathcal{T} \subseteq \mathcal{D}_k, k \notin \mathcal{T}\} \tag{2.43}$$

$$\mathcal{A}_k = \{W_{f,\mathcal{S},\mathcal{T}} : f \in [F], \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha, \mathcal{T} \subseteq \mathcal{S}, k \in \mathcal{T}\} \tag{2.44}$$

for each user $k \in [K]$. For this setting the side information graph takes the form of a directed graph where each subfile represents a vertex, and where there is a connection from (the node corresponding to) $W_{f_{k_1},\mathcal{D}_{k_1},\mathcal{T}_1}$ to $W_{f_{k_2},\mathcal{D}_{k_2},\mathcal{T}_2}$ if and only if $W_{f_{k_1},\mathcal{D}_{k_1},\mathcal{T}_1} \in \mathcal{A}_{k_2}$, i.e., if and only if $k_2 \in \mathcal{T}_1$. To apply Lemma 2.1, we are interested in acyclic sets of vertices $\mathcal{J}$ in such side information graph. In the spirit of [3], we know that the set

$$\bigcup_{k\in[K]} \bigcup_{\mathcal{T}\subseteq(([K]\setminus\{u_1,\dots,u_k\})\cap\mathcal{D}_{u_k})} \left\{W_{f_{u_k},\mathcal{D}_{u_k},\mathcal{T}}\right\} \tag{2.45}$$

does not contain any directed cycle[14] for any demand $(\boldsymbol{d}, \boldsymbol{f})$ and any vector $\boldsymbol{u}$, where $\boldsymbol{u} = (u_1, \dots, u_K)$ is a permutation of the users in $[K]$, i.e., $\boldsymbol{u} = \boldsymbol{\pi}^K$ for some $\pi \in S_K$. Consequently, applying Lemma 2.1 yields the following lower bound

$$R^{\star}_{\mathrm{u,s}} \geq R(\boldsymbol{d}, \boldsymbol{f}, \boldsymbol{u}) \tag{2.46}$$

where $R(\boldsymbol{d}, \boldsymbol{f}, \boldsymbol{u})$ is defined as

$$R(\boldsymbol{d}, \boldsymbol{f}, \boldsymbol{u}) \coloneqq \sum_{k\in[K]} \sum_{\mathcal{T}\subseteq(([K]\setminus\{u_1,\dots,u_k\})\cap\mathcal{D}_{u_k})} \frac{\left|W_{f_{u_k},\mathcal{D}_{u_k},\mathcal{T}}\right|}{B}. \tag{2.47}$$

[14] Notice that [3, Lemma 1] considers in fact $\mathcal{T} \subseteq ([K] \setminus \{u_1, \dots, u_k\})$ and not $\mathcal{T} \subseteq (([K] \setminus \{u_1, \dots, u_k\}) \cap \mathcal{D}_{u_k})$. However, the latter is a subset of the former, and thus the lemma still holds.

Selection of Distinct Demands

Our goal is to create several bounds as the one in (2.46) and eventually average all of them to obtain a useful lower bound on the optimal worst-case load. We aim to create a bound for a set $\mathcal{C}$ of properly selected demands and for a set $\mathcal{U}_{(\boldsymbol{d},\boldsymbol{f})}$ of properly selected permutations for each demand $(\boldsymbol{d},\boldsymbol{f}) \in \mathcal{C}$. Hence, we aim to simplify the expression given by

$$\sum_{(\boldsymbol{d},\boldsymbol{f})\in\mathcal{C}} \sum_{\boldsymbol{u}\in\mathcal{U}_{(\boldsymbol{d},\boldsymbol{f})}} BR^{\star}_{\mathrm{u,s}} \geq \sum_{(\boldsymbol{d},\boldsymbol{f})\in\mathcal{C}} \sum_{\boldsymbol{u}\in\mathcal{U}_{(\boldsymbol{d},\boldsymbol{f})}} R(\boldsymbol{d},\boldsymbol{f},\boldsymbol{u}). \tag{2.48}$$

Notice that the goal of carefully selecting the demand set $\mathcal{C}$ and the permutation set $\mathcal{U}_{(\boldsymbol{d},\boldsymbol{f})}$ is twofold. The first is to provide the symmetry that will allow us to simplify (2.48) into a meaningful expression, and the second is to force the bound to be as tight as possible[15].

Remark 2.4. The careful selection of the demand set $\mathcal{C}$ and permutation set $\mathcal{U}_{(\boldsymbol{d},\boldsymbol{f})}$ for each $(\boldsymbol{d},\boldsymbol{f}) \in \mathcal{C}$ is a pivotal difference with respect to the procedure in [3]. Indeed, if one develops a converse bound following the exact same procedure in [3], then the resulting bound is looser than the one presented in this paper. Finding the good set of demands and permutations becomes the key to obtaining a tighter bound while keeping the problem analytically tractable.

For a permutation function $\pi \in H_K$, we consider the demands $(\boldsymbol{d},\boldsymbol{f})$ where $\boldsymbol{d} = (\mathcal{D}_1,\ldots,\mathcal{D}_K)$ is such that

$$\mathcal{D}_k = \{k, \pi((\pi^{-1}(k)+1) \bmod K), \ldots, \pi((\pi^{-1}(k)+\alpha-1) \bmod K)\} \tag{2.49}$$

and $f_k \in [F]$ for each $k \in [K]$. There is a total of F^K such demands. Considering that the order of H_K is $(K-1)!$ and that we take F^K demands for each $\pi \in H_K$, we consider $(K-1)!F^K$ distinct[16] demands in total, denoting by $\mathcal{C}$ the set of such demands and referring to the demands in $\mathcal{C}$ as *circular demands*. Since the vector $\boldsymbol{d}$ depends on some circular permutation $\pi \in H_K$, we will identify from now on each demand with $(\boldsymbol{d}_\pi,\boldsymbol{f}) \in \mathcal{C}$ to highlight such dependency. For a demand $(\boldsymbol{d}_\pi,\boldsymbol{f}) \in \mathcal{C}$, we let $\mathcal{U}_{(\boldsymbol{d}_\pi,\boldsymbol{f})}$ be the set containing the K circular shifts of $\boldsymbol{\pi}^K$.

The aforementioned sets $\mathcal{C}$ and $\mathcal{U}_{(\boldsymbol{d}_\pi,\boldsymbol{f})}$ for each $(\boldsymbol{d}_\pi,\boldsymbol{f}) \in \mathcal{C}$ will generally yield larger acyclic subgraphs[17] in (2.45) that can be used to increase the

[15]We wish to stress that constructing the converse using the demand set $\mathcal{C}$ and the permutation set $\mathcal{U}_{(\boldsymbol{d},\boldsymbol{f})}$ for each $(\boldsymbol{d},\boldsymbol{f}) \in \mathcal{C}$ does not mean that the converse is the tightest nor that the demands $\mathcal{C}$ are part of the worst-case demand set.

[16]Letting $\pi \in H_K$ be a circular permutation ensures that no $\boldsymbol{d}$ vector is repeated for any specific $\boldsymbol{f}$ vector.

[17]This is based on the following observation. Throughout various examples, such demands generally yielded the largest bounds compared to other classes of demands.

RHS in (2.46), and to provide a better lower bound on $R^{\star}_{\mathrm{u,s}}$. We provide in the following a clarifying example for a circular demand in $(\boldsymbol{d}_{\pi}, \boldsymbol{f}) \in \mathcal{C}$ and the corresponding set $\mathcal{U}_{(\boldsymbol{d}_{\pi},\boldsymbol{f})}$.

Example 2.4. Consider the $(4, 3, F)$ FDS structure and the permutation $\boldsymbol{\pi}^4 = (1, 2, 3, 4)$. For such permutation, we consider the demands $(\boldsymbol{d}, \boldsymbol{f})$ where $\boldsymbol{d} = (123, 234, 134, 124)$ and $\boldsymbol{f} = (f_1, f_2, f_3, f_4)$ with $f_k \in [F]$ for each $k \in [4]$. In addition, $\mathcal{U}_{(\boldsymbol{d}_{\pi},\boldsymbol{f})}$ is the set containing the 4 circular shifts of $\boldsymbol{\pi}^4$, which in this case corresponds to $\mathcal{U}_{(\boldsymbol{d}_{\pi},\boldsymbol{f})} = \{(1, 2, 3, 4), (2, 3, 4, 1), (3, 4, 1, 2), (4, 1, 2, 3)\}$.

Constructing the Optimization Problem

We will seek to simplify the expression in (2.48), and then to minimize the new simplified expression, in order to lower bound the optimal worst-case load $R^{\star}_{\mathrm{u,s}}$. Toward simplifying the expression in (2.48), we count how many times each subfile $W_{f,\mathcal{S},\mathcal{T}}$ — for any $f \in [F]$, $\mathcal{S} \subseteq [K]$ with $|\mathcal{S}| = \alpha$, $\mathcal{T} \subseteq \mathcal{S}$ and $|\mathcal{T}| = t'$ with $t' \in [0 : \alpha]$ — appears in (2.48). To this end, we need the following lemma.

Lemma 2.2. *Let $\pi \in S_K$ be a permutation of the set $[K]$. Consider $k_1, k_2 \in [K]$ such that $k_1 \neq k_2$. Consider*

$$\ell = (\pi^{-1}(k_2) - \pi^{-1}(k_1)) \bmod K. \tag{2.50}$$

Then, out of the K circular shifts of $\boldsymbol{\pi}^K$, there is a total of $(K - \ell)$ of them such that k_1 appears before k_2.

Proof. The proof is reported in Appendix A.3. □

The counting argument proceeds as follows. First, we focus on some subfile $W_{f,\mathcal{S},\mathcal{T}}$ and we assume that the file $W_{f,\mathcal{S}}$ is requested by some user $k \in (\mathcal{S} \setminus \mathcal{T})$. Next, we count the number of demands $(\boldsymbol{d}_{\pi}, \boldsymbol{f}) \in \mathcal{C}$ for which k and right-most element included from $\mathcal{T}$ are at distance ℓ in $\boldsymbol{\pi}^K$. Then, we evaluate how many times the subfile appears in the index coding bounds associated to such demands. After that, noticing that $\ell \in [t' : \alpha - 1]$, we repeat the procedure for each value of ℓ. Finally, since the same procedure can be repeated for each $k \in (\mathcal{S} \setminus \mathcal{T})$, we multiply the end result by $|\mathcal{S} \setminus \mathcal{T}| = (\alpha - t')$.

Let us focus on the subfile $W_{f,\mathcal{S},\mathcal{T}}$ for some $f \in [F]$, $\mathcal{S} \subseteq [K]$ with $|\mathcal{S}| = \alpha$, $\mathcal{T} \subseteq \mathcal{S}$ and $|\mathcal{T}| = t'$ for some $t' \in [0 : \alpha]$. For some user $k \in (\mathcal{S} \setminus \mathcal{T})$ and for $\ell \in [t' : \alpha - 1]$, there is a total of $a_\ell := t'!(\alpha - 1 - t')!(K - \alpha)!\binom{\ell-1}{t'-1}F^{K-1}$ demands $(\boldsymbol{d}_{\pi}, \boldsymbol{f}) \in \mathcal{C}$ such that the file $W_{f,\mathcal{S}}$ is requested by such user $k \in (\mathcal{S} \setminus \mathcal{T})$ and there are exactly ℓ elements in $\boldsymbol{\pi}^K$ between k and the right-most element included from $\mathcal{T}$. Let $\mathcal{C}_{k,\ell}$ be the set of such demands.

Considering how the acyclic set of vertices in (2.45) is built, the subfile $W_{f,\mathcal{S},\mathcal{T}}$ appears in the index coding bound induced by each $(\boldsymbol{d}_\pi, \boldsymbol{f}) \in \mathcal{C}_{k,\ell}$ whenever *all* the elements in $\mathcal{T}$ appear after k in $\boldsymbol{u} \in \mathcal{U}_{(\boldsymbol{d}_\pi,\boldsymbol{f})}$. Since for each demand $(\boldsymbol{d}_\pi, \boldsymbol{f}) \in \mathcal{C}_{k,\ell}$ there are exactly ℓ elements in $\boldsymbol{\pi}^K$ separating k and the right-most element from $\mathcal{T}$, we know from Lemma 2.2 that there are $(K-\ell)$ vectors $\boldsymbol{u} \in \mathcal{U}_{(\boldsymbol{d}_\pi,\boldsymbol{f})}$ where all the elements in $\mathcal{T}$ appear after k. Observing that $\ell \in [t' : \alpha - 1]$ and that such reasoning applies whenever the file $W_{f,\mathcal{S}}$ is requested by any of the $(\alpha - t')$ users in $\mathcal{S} \setminus \mathcal{T}$, the specific subfile $W_{f,\mathcal{S},\mathcal{T}}$ appears $(\alpha - t') \sum_{\ell=t'}^{\alpha-1} a_\ell (K-\ell)$ times in (2.48). Since the same reasoning applies to any other subfile, the expression in (2.48) can be rewritten as

$$R^\star_{\mathrm{u,s}} \geq \frac{1}{BK!F^K} \sum_{(\boldsymbol{d}_\pi,\boldsymbol{f})\in\mathcal{C}} \sum_{\boldsymbol{u}\in\mathcal{U}_{(\boldsymbol{d}_\pi,\boldsymbol{f})}} R(\boldsymbol{d}_\pi, \boldsymbol{f}, \boldsymbol{u}) \tag{2.51}$$

$$= \sum_{t'=0}^{\alpha} f(t') x_{t'} \tag{2.52}$$

where $f(t')$ and $x_{t'}$ are defined as

$$f(t') \coloneqq N \frac{(\alpha - t')}{F^K K!} \sum_{\ell=t'}^{\alpha-1} a_\ell (K-\ell) \tag{2.53}$$

$$0 \leq x_{t'} \coloneqq \sum_{\mathcal{S}\subseteq[K]:|\mathcal{S}|=\alpha} \sum_{\mathcal{T}\subseteq\mathcal{S}:|\mathcal{T}|=t'} \sum_{f\in[F]} \frac{|W_{f,\mathcal{S},\mathcal{T}}|}{NB}. \tag{2.54}$$

At this point, we seek to lower bound the minimum worst-case load $R^\star_{\mathrm{u,s}}$ by lower bounding the solution to the following optimization problem

$$\min_{\boldsymbol{x}} \quad \sum_{t'=0}^{\alpha} f(t') x_{t'} \tag{2.55a}$$

$$\text{subject to} \quad \sum_{t'=0}^{\alpha} x_{t'} = 1 \tag{2.55b}$$

$$\sum_{t'=0}^{\alpha} t' x_{t'} \leq \frac{KM}{N} \tag{2.55c}$$

where (2.55b) and (2.55c) correspond to the file size constraint and the cumulative cache size constraint, respectively.

Lower Bounding the Solution to the Optimization Problem

Before proceeding to lower bound the solution the optimization problem, we wish to further simplify the function $f(t')$. Indeed, this function $f(t')$ can be

rewritten as

$$f(t') = \frac{1}{\binom{\alpha}{t}} \sum_{\ell=t'}^{\alpha-1} \binom{\ell-1}{t'-1}(K-\ell). \tag{2.56}$$

Then, we can write

$$\begin{aligned} f(t') &= \frac{1}{\binom{\alpha}{t'}} \sum_{\ell=t'}^{\alpha-1} \binom{\ell-1}{t'-1}(K-\ell) && (2.57)\\ &= \frac{1}{\binom{\alpha}{t'}} \left(K \sum_{\ell=t'}^{\alpha-1} \binom{\ell-1}{t'-1} - \sum_{\ell=t'}^{\alpha-1} \ell \binom{\ell-1}{t'-1} \right) && (2.58)\\ &= \frac{1}{\binom{\alpha}{t'}} \left(K \sum_{\ell=t'-1}^{\alpha-2} \binom{\ell}{t'-1} - t' \sum_{\ell=t'}^{\alpha-1} \binom{\ell}{t'} \right) && (2.59)\\ &= \frac{K\binom{\alpha-1}{t'} - t'\binom{\alpha}{t'+1}}{\binom{\alpha}{t'}} && (2.60)\\ &= \frac{\binom{\alpha}{t'+1} + (K-\alpha)\binom{\alpha-1}{t'}}{\binom{\alpha}{t'}} && (2.61) \end{aligned}$$

where (2.60) uses the well-known hockey-stick identity, which states that

$$\sum_{i=k}^{n} \binom{i}{k} = \binom{n+1}{k+1}, \quad \forall n, k \in \mathbb{N}, \quad n \geq k. \tag{2.62}$$

Now, since we can consider $\boldsymbol{x} = (x_0, \ldots, x_\alpha)$ as a probability mass function, the optimization problem in (2.55) can be seen as the minimization of $\mathbb{E}[f(t')]$. Moreover, the following holds.

Lemma 2.3. *The function $f(t')$ is convex and is strictly decreasing for increasing values of t'.*

Proof. The proof is reported in Appendix A.4. □

Taking advantage of Lemma 2.3, we can write $\mathbb{E}[f(t')] \geq f(\mathbb{E}[t'])$ using Jensen's inequality. Then, since $f(t')$ is strictly decreasing with increasing $t' \in [0:\alpha]$, we can further write $f(\mathbb{E}[t']) \geq f(KM/N)$ taking advantage of the fact that $\mathbb{E}[t']$ is upper bounded as in (2.55c). Consequently, $\mathbb{E}[f(t')] \geq f(t)$, where $t = KM/N$. Now, when t is an integer, the bound is simply the function $f(t)$ evaluated at $t \in [0:\alpha]$. For non-integer values of t, we can follow the reasoning in [4], [39], [52] to take the lower convex envelope of the

sequence of points $\{(t, f(t)) : t \in [0:\alpha]\}$. To conclude, the converse bound is a piecewise linear curve with corner points

$$(M, R_{\text{LB}}) = \left(t\frac{N}{K}, \frac{\binom{\alpha}{t+1} + (K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}} \right), \quad \forall t \in [0:\alpha]. \tag{2.63}$$

Further, R_{LB} takes the form

$$\begin{aligned} R_{\text{LB}} &= \frac{\binom{\alpha}{t+1} + (K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}} && (2.64)\\ &= \frac{\alpha - t}{1+t} + (K-\alpha)\left(1 - \frac{t}{\alpha}\right) && (2.65)\\ &= \frac{\alpha(1-\gamma_\alpha)}{1+K\gamma} + K(1-\gamma_\alpha) - \alpha(1-\gamma_\alpha) && (2.66)\\ &= K(1-\gamma_\alpha)\left[1 + \frac{\alpha}{K(1+K\gamma)} - \frac{\alpha}{K}\right] && (2.67)\\ &= \frac{K(1-\gamma_\alpha)}{1+K\gamma}\left[1 + K\gamma + \frac{\alpha}{K} - \frac{\alpha}{K}(1+K\gamma)\right] && (2.68)\\ &= \frac{K(1-\gamma_\alpha)}{1+K\gamma}\left[(K-\alpha)\gamma + 1\right] && (2.69) \end{aligned}$$

which completes the proof. □

2.4.2 A Detailed Example for the Converse Bound

We present here in detail an example that can help the reader better understand the construction of the converse bound.

Consider the symmetric $(4, 3, 1)$ FDS structure which involves a file library $\{W_{1,\mathcal{S}} : \mathcal{S} \subseteq [4], |\mathcal{S}| = 3\}$ consisting of $C = \binom{K}{\alpha} = \binom{4}{3} = 4$ classes of files. Since there is only $F = 1$ file per class, there is a total of $N = FC = 4$ files, so in this example we make no distinction between files and classes of files. For simplicity, we will here refer to file $W_{1,\mathcal{S}}$ directly as $W_{\mathcal{S}}$, which means that each file is entirely described by a 3-tuple $\mathcal{S} \subseteq [4]$, and each demand instance is entirely defined by the $\boldsymbol{d} = (\mathcal{D}_1, \ldots, \mathcal{D}_4)$ vector only. The FDS of each user $k \in [4]$ is given by $\mathcal{F}_k = \{W_S : \mathcal{S} \subseteq [4], |\mathcal{S}| = 3, k \in \mathcal{S}\}$ and it consists of $F\binom{K-1}{\alpha-1} = 3$ files. This example considers $t \in [0:3]$, simply because having $M = tN/K = 3$ implies that the point $(M, R) = (3, 0)$ is trivially achievable being each user able to preemptively cache the entirety of its FDS.

Assuming the most general uncoded and selfish cache placement, each file $W_{\mathcal{S}}$ is split into a total of $2^\alpha = 8$ disjoint subfiles as

$$W_{\mathcal{S}} = \{W_{\mathcal{S},\mathcal{T}} : \mathcal{T} \subseteq \mathcal{S}\}, \quad \forall \mathcal{S} \subseteq [4] : |\mathcal{S}| = 3 \tag{2.70}$$

where again $W_{\mathcal{S},\mathcal{T}}$ is the subfile of $W_{\mathcal{S}}$ cached by users in $\mathcal{T}$.

Constructing the Index Coding Bound

For any given demand $\boldsymbol{d} = (\mathcal{D}_1, \dots, \mathcal{D}_4)$ where the k-th user asks for a distinct file $W_{\mathcal{D}_k}$ for each $k \in [4]$, we consider the index coding problem with $K' = K = 4$ users and $N' = K2^{\alpha-1} = 16$ messages, where each user $k \in [4]$ has a desired message set $\mathcal{M}_k = \{W_{\mathcal{D}_k,\mathcal{T}} : \mathcal{T} \subseteq \mathcal{D}_k, k \notin \mathcal{T}\}$ and a side information set $\mathcal{A}_k = \{W_{\mathcal{S},\mathcal{T}} : \mathcal{S} \subseteq [4], |\mathcal{S}| = 3, \mathcal{T} \subseteq \mathcal{S}, k \in \mathcal{T}\}$.

If we identify again in the side information graph of the problem the acyclic set of vertices as in (2.45), applying Lemma 2.1 yields the following lower bound

$$R^{\star}_{\text{u,s}} \geq R(\boldsymbol{d}, \boldsymbol{u}) \tag{2.71}$$

where now we define

$$R(\boldsymbol{d}, \boldsymbol{u}) \coloneqq \sum_{k\in[4]} \sum_{\mathcal{T}\subseteq(([4]\setminus\{u_1,\dots,u_k\})\cap\mathcal{D}_{u_k})} \frac{\left|W_{\mathcal{D}_{u_k},\mathcal{T}}\right|}{B}. \tag{2.72}$$

Recall that (2.71) holds for any demand $\boldsymbol{d}$ and any $\boldsymbol{u} = \boldsymbol{\pi}^4$ for some $\pi \in S_4$.

Constructing the Optimization Problem and Bounding its Solution

As described in Section 2.4.1, we let $\mathcal{C}$ be the set of distinct demands where we have a $\boldsymbol{d}_\pi$ vector for each circular permutation $\pi \in H_4$. The number of such permutations is $(K-1)! = 6$. For simplicity, we denote by π_i the i-th circular permutation for $i \in [6]$, where such circular permutations are given by

$$\boldsymbol{\pi}^4_1 = (1,2,3,4) \quad \boldsymbol{\pi}^4_4 = (1,3,4,2) \tag{2.73}$$
$$\boldsymbol{\pi}^4_2 = (1,2,4,3) \quad \boldsymbol{\pi}^4_5 = (1,4,2,3) \tag{2.74}$$
$$\boldsymbol{\pi}^4_3 = (1,3,2,4) \quad \boldsymbol{\pi}^4_6 = (1,4,3,2). \tag{2.75}$$

None of the above permutations can be obtained as a rotation of any of the others. Hence, this ensures to have in $\mathcal{C}$ the following distinct demand vectors

$$\boldsymbol{d}_{\pi_1} = (123, 234, 134, 124) \quad \boldsymbol{d}_{\pi_4} = (134, 123, 234, 124) \tag{2.76}$$
$$\boldsymbol{d}_{\pi_2} = (124, 234, 123, 134) \quad \boldsymbol{d}_{\pi_5} = (124, 123, 134, 234) \tag{2.77}$$
$$\boldsymbol{d}_{\pi_3} = (123, 124, 234, 134) \quad \boldsymbol{d}_{\pi_6} = (134, 124, 123, 234) \tag{2.78}$$

with $\mathcal{U}_{\boldsymbol{d}_{\pi_i}}$ containing the 4 circular shifts of $\boldsymbol{\pi}^4_i$. For instance, it is $\mathcal{U}_{\boldsymbol{d}_{\pi_1}} = \{(1,2,3,4),(2,3,4,1),(3,4,1,2),(4,1,2,3)\}$.

Now, we create a bound as in (2.71) for each $\boldsymbol{d}_{\pi_i}$ with $i \in [6]$ and for each $\boldsymbol{u} \in \mathcal{U}_{\boldsymbol{d}_{\pi_i}}$ for a given $\boldsymbol{d}_{\pi_i}$, we sum all such bounds and we obtain the expression given by

$$\sum_{i\in[6]} \sum_{\boldsymbol{u}\in\mathcal{U}_{\boldsymbol{d}_{\pi_i}}} BR^{\star}_{\mathrm{u,s}} \geq \sum_{i\in[6]} \sum_{\boldsymbol{u}\in\mathcal{U}_{\boldsymbol{d}_{\pi_i}}} R(\boldsymbol{d}_{\pi_i}, \boldsymbol{u}). \tag{2.79}$$

We simplify (2.79) by counting how many times each subfile $W_{\mathcal{S},\mathcal{T}}$ — for any $\mathcal{S} \subseteq [4]$ with $|\mathcal{S}| = 3$, $\mathcal{T} \subseteq \mathcal{S}$ and $|\mathcal{T}| = t'$ with $t' \in [0:3]$ — appears in (2.79).

For example, let us focus on the subfile $W_{123,2}$. First of all, we notice that such subfile may appear to the RHS of (2.71) whenever W_{123} is requested by any user $k \in (\mathcal{S} \setminus \mathcal{T}) = \{1, 3\}$. Assume that W_{123} is requested by user 1, which means we consider the bounds with demands $\boldsymbol{d}_{\pi_1}$ and $\boldsymbol{d}_{\pi_3}$. Denoting by ℓ the distance[18] between user 1 and user 2 in $\boldsymbol{\pi}^4_i$ for $i \in \{1, 3\}$, we notice that $\ell \in \{1, 2\}$. We see that there is only $\boldsymbol{\pi}^4_1$ for $\ell = 1$, so $W_{123,2}$ appears $(K - \ell) = 3$ times in the bounds with demand $\boldsymbol{d}_{\pi_1}$ and permutations in $\mathcal{U}_{\boldsymbol{d}_{\pi_1}}$. Similarly, there is only $\boldsymbol{\pi}^4_3$ for $\ell = 2$, so $W_{123,2}$ appears $(K - \ell) = 2$ times in the bounds with demand $\boldsymbol{d}_{\pi_3}$ and permutations in $\mathcal{U}_{\boldsymbol{d}_{\pi_3}}$. Thus, summing over the possible values of ℓ, the subfile $W_{123,2}$ appears $3 + 2 = 5$ times in the bounds with demands $\boldsymbol{d}_{\pi_1}$ and $\boldsymbol{d}_{\pi_3}$, and their relative permutations of users. The same rationale follows for the bounds built with $\boldsymbol{d}_{\pi_2}$, $\boldsymbol{d}_{\pi_6}$ and their relative permutations, i.e., the bounds where file W_{123} is requested by user 3. Hence, the subfile $W_{123,2}$ appears in (2.79) a total of $2 \times (3 + 2) = 10$ times.

The same holds for any other subfile cached at only one user and a similar counting argument can be made for arbitrary $|\mathcal{T}| = t'$ with $t' \in [0:3]$, as described in Section 2.4.1. Thus, we can formulate an optimization problem as in (2.55), bounding its solution by means of Jensen's inequality and convexity of $f(t')$. The resulting bound is a piecewise linear curve with corner points $(M, R_{\mathrm{LB}}) = (t, (3 - t)/(1 + t) + 1 - t/3)$ for each $t \in [0:3]$.

2.5 The Exact Memory-Load Trade-Off for the α-Demands

We will here draw insights from the converse to establish a general cache placement policy, and then a delivery scheme that applies to a specific set of so-called α-Demands. For these demands and for the specific placement policy, the scheme will be proven optimal with the use of an additional converse.

[18] We remind that such distance represents the number of elements which separate index 1 and index 2, including the latter.

We start by noticing that the converse in Theorem 2.1 can be decomposed as

$$R_{\mathrm{LB}}(t) = \frac{\binom{\alpha}{t+1}}{\binom{\alpha}{t}} + \frac{(K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}} \tag{2.80}$$

with the first term $R_1(t) \coloneqq \binom{\alpha}{t+1}/\binom{\alpha}{t} = \alpha(1-\gamma_\alpha)/(1+\alpha\gamma_\alpha)$ bringing to mind a smaller MAN placement-and-delivery (unselfish) problem with α users, a common library, and normalized cache size γ_α, and with the second term $R_2(t) \coloneqq (K-\alpha)\binom{\alpha-1}{t}/\binom{\alpha}{t} = (K-\alpha)(1-\gamma_\alpha)$ bringing to mind uncoded delivery to $(K-\alpha)$ users. Let us exploit this observation to suggest a placement.

2.5.1 Cache Placement

As noted, we can think of the $R_1(t)$ term as representing the optimal load in a "smaller" α-MAN problem with α users that *are known in advance* to be interested in a common class of files and thus benefit from the corresponding α-user MAN placement. If each user — as is the case in our setting — can allocate a fraction γ_α for each file of potential interest, then a MAN placement implies that each user stores a total of $F\gamma_\alpha$ files[19]. Here, in our effort to provide a placement method, we must account for the fact that there is a total of $\binom{K}{\alpha}$ such "smaller" MAN problems, because there are $C = \binom{K}{\alpha}$ file classes. Let us now recall that each user appears in a total of $\binom{K-1}{\alpha-1}$ such smaller problems, since there are $\binom{K-1}{\alpha-1}$ file classes that each user is interested in. Our placement must account for the possibility of each user participating in any such smaller problem. This requires each user to store $F\gamma_\alpha$ files per class of interest, and thus requires a total storage capacity of $\gamma_\alpha F\binom{K-1}{\alpha-1} = M$, which, as we see, nicely satisfies the cache size constraint. This reasoning justifies the cache placement procedure that we present below.

In our proposed uncoded and selfish cache placement method, based on the same combinatorial argument of the MAN scheme, each user k proceeds to cache only from $\mathcal{F}_k$. The placement begins by splitting each file into $\binom{\alpha}{t}$ non-overlapping subfiles as

$$W_{f,\mathcal{S}} = \{W_{f,\mathcal{S},\mathcal{T}} : \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = t\} \tag{2.81}$$

for each $\mathcal{S} \subseteq [K]$ such that $|\mathcal{S}| = \alpha$ and for each $f \in [F]$, and then is completed by filling the cache Z_k of each user $k \in [K]$ as

$$Z_k = \{W_{f,\mathcal{S},\mathcal{T}} : f \in [F], \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha, \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = t, k \in \mathcal{T}\}. \tag{2.82}$$

[19]Recall that F is the total number of files in this common class of files.

Each cache stores $\binom{\alpha-1}{t-1}$ subfiles for each file in its FDS, so abiding by the cache size constraint

$$|\mathcal{F}|\binom{\alpha-1}{t-1}\frac{B}{\binom{\alpha}{t}} = F\binom{K-1}{\alpha-1}\frac{t}{\alpha}B = MB. \tag{2.83}$$

Remark 2.5. Unfortunately, the above reasoning does not immediately reflect — at least not to us — a universal delivery solution for any set of demands. To the best of our understanding, our cache placement introduces the need to resolve a large number of non-isomorphic index coding problems. Nevertheless, the careful interpretation of the converse allowed us to suggest a general selfish cache placement policy. Similarly, it also allows us to identify a well-defined class of demands, as we can see below.

2.5.2 Delivery Scheme for the Set of α-Demands

We now present the delivery method for the following class of demands.

Definition 2.3 (α-Demands)**.** Considering the (K, α, F) FDS structure with $F \geq \alpha$, the demand defined by the vectors $\boldsymbol{d} = (\mathcal{D}_1, \ldots, \mathcal{D}_K)$ and $\boldsymbol{f} = (f_1, \ldots, f_K)$ is an α-demand if and only if there exists at least one set of users $\mathcal{K} \subseteq [K]$ such that $|\mathcal{K}| = \alpha$, $f_{k_1} \neq f_{k_2}$ for any $k_1 \neq k_2$ with $k_1, k_2 \in \mathcal{K}$ and $\mathcal{D}_k = \mathcal{K}$ for all $k \in \mathcal{K}$.

Such demands can exist only if $F \geq \alpha$. Indeed, if we have at least α files per class, then we can have distinct demands where there exists at least one set of α users requesting distinct files, all belonging to the same file class.

Let $R_{\alpha,\mathrm{c}}$ denote the worst-case load when only α-demands are considered, and when the cache placement in Section 2.5.1 is adopted. We are now ready to provide the exact characterization of optimal such load $R^{\star}_{\alpha,\mathrm{c}}$.

Proposition 2.1 (The Exact Memory-Load Trade-Off for α-Demands Under the Presented Symmetric Placement)**.** *Assuming the selfish and uncoded cache placement presented in Section 2.5.1, the optimal worst-case communication load $R^{\star}_{\alpha,c}$ for the (K, α, F) FDS structure and α-Demands is a piecewise linear curve with corner points*

$$(M, R^{\star}_{\alpha,c}) = \left(t\frac{N}{K}, \frac{\binom{\alpha}{t+1} + (K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}}\right), \quad \forall t \in [0:\alpha] \tag{2.84}$$

again corresponding to

$$R^{\star}_{\alpha,c} = \frac{K(1-\gamma_\alpha)}{K\gamma+1}\Big[(K-\alpha)\gamma+1\Big]. \tag{2.85}$$

Proof. The proof of the converse is reported in Appendix A.5, whereas the proof of the achievability is reported below in Section 2.5.3. □

2.5.3 Achievability Proof of Proposition 2.1

By definition, any α-demand has at least one set of α users requesting distinct files from the same file class. If we denote by $\mathcal{K}$ one of such sets, it holds that $\mathcal{D}_k = \mathcal{K}$ for all $k \in \mathcal{K}$ and $f_{k_1} \neq f_{k_2}$ for all $k_1 \neq k_2$ with $k_1, k_2 \in \mathcal{K}$. Consider user $k \in \mathcal{K}$. According to the cache placement procedure in Section 2.5.1, this user does not have in its cache any subfile $W_{f_k,\mathcal{K},\mathcal{T}}$ where $\mathcal{T} \subseteq \mathcal{K}$, $|\mathcal{T}| = t$ and $k \notin \mathcal{T}$. If we focus on this set $\mathcal{K}$ of α users only, we can automatically construct the following sequence of multicast messages

$$X_{\mathcal{K}} = \left(\bigoplus_{k \in \mathcal{S}} W_{f_k,\mathcal{K},\mathcal{S}\setminus\{k\}} : \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| = t+1 \right). \tag{2.86}$$

For the remaining $(K-\alpha)$ users in $[K]\setminus\mathcal{K}$, we consider the following sequence

$$X_{[K]\setminus\mathcal{K}} = (W_{f_k,\mathcal{D}_k,\mathcal{T}} : k \in ([K]\setminus\mathcal{K}), k \notin \mathcal{T}) \tag{2.87}$$

of uncoded transmissions. Then, the transmitter delivers the concatenated $X = (X_{\mathcal{K}}, X_{[K]\setminus\mathcal{K}})$, inducing a load

$$\frac{|X|}{B} = \frac{|X_{\mathcal{K}}| + |X_{[K]\setminus\mathcal{K}}|}{B} = \frac{\binom{\alpha}{t+1} + (K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}} \tag{2.88}$$

which implies that $R^{\star}_{\alpha,\mathrm{c}}(t) \leq \frac{\binom{\alpha}{t+1}+(K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}}$ for all $t \in [0:\alpha]$.

Remark 2.6. As a testament to the additional usefulness of the converse developed here, we point out that the above achievable scheme is developed as a direct outcome of a meticulous inspection of the expression of the bound itself. □

2.5.4 Example of the Achievable Scheme

Consider the $(K, \alpha, F) = (5, 3, 3)$ FDS structure. We have $C = \binom{K}{\alpha} = \binom{5}{3} = 10$ classes of files with a total of $N = FC = 30$ files. The FDS structure is given by

$$\mathcal{F}_1 = \{\mathcal{W}_{123}, \mathcal{W}_{124}, \mathcal{W}_{125}, \mathcal{W}_{134}, \mathcal{W}_{135}, \mathcal{W}_{145}\} \tag{2.89}$$

$$\mathcal{F}_2 = \{\mathcal{W}_{123}, \mathcal{W}_{124}, \mathcal{W}_{125}, \mathcal{W}_{234}, \mathcal{W}_{235}, \mathcal{W}_{245}\} \tag{2.90}$$

$$\mathcal{F}_3 = \{\mathcal{W}_{123}, \mathcal{W}_{134}, \mathcal{W}_{135}, \mathcal{W}_{234}, \mathcal{W}_{235}, \mathcal{W}_{345}\} \tag{2.91}$$
$$\mathcal{F}_4 = \{\mathcal{W}_{124}, \mathcal{W}_{134}, \mathcal{W}_{145}, \mathcal{W}_{234}, \mathcal{W}_{245}, \mathcal{W}_{345}\} \tag{2.92}$$
$$\mathcal{F}_5 = \{\mathcal{W}_{125}, \mathcal{W}_{135}, \mathcal{W}_{145}, \mathcal{W}_{235}, \mathcal{W}_{245}, \mathcal{W}_{345}\} \tag{2.93}$$

where $\mathcal{W}_{\mathcal{S}} = \{W_{1,\mathcal{S}}, W_{2,\mathcal{S}}, W_{3,\mathcal{S}}\}$ for each triplet $\mathcal{S} \subseteq [5]$. Let us consider the scenario of $t = 2$. In this case, each file is split as

$$W_{f,123} = \{W_{f,123,12}, W_{f,123,13}, W_{f,123,23}\} \tag{2.94}$$
$$W_{f,124} = \{W_{f,124,12}, W_{f,124,14}, W_{f,124,24}\} \tag{2.95}$$
$$W_{f,125} = \{W_{f,125,12}, W_{f,125,15}, W_{f,125,25}\} \tag{2.96}$$
$$W_{f,134} = \{W_{f,134,13}, W_{f,134,14}, W_{f,134,34}\} \tag{2.97}$$
$$W_{f,135} = \{W_{f,135,13}, W_{f,135,15}, W_{f,135,35}\} \tag{2.98}$$
$$W_{f,145} = \{W_{f,145,14}, W_{f,145,15}, W_{f,145,45}\} \tag{2.99}$$
$$W_{f,234} = \{W_{f,234,23}, W_{f,234,24}, W_{f,234,34}\} \tag{2.100}$$
$$W_{f,235} = \{W_{f,235,23}, W_{f,235,25}, W_{f,235,35}\} \tag{2.101}$$
$$W_{f,245} = \{W_{f,245,24}, W_{f,245,25}, W_{f,245,45}\} \tag{2.102}$$
$$W_{f,345} = \{W_{f,345,34}, W_{f,345,35}, W_{f,345,45}\} \tag{2.103}$$

for all $f \in [3]$.

Consider the demand defined by the vectors $\boldsymbol{f} = (1, 2, 3, 1, 1)$ and $\boldsymbol{d} = (123, 123, 123, 124, 125)$. This is an α-demand because there exists a set $\mathcal{K}$ of α users all requesting distinct files belonging to the same file class $\mathcal{K}$. Here, this set is $\mathcal{K} = \{1, 2, 3\}$.

In accordance to the described selfish and uncoded cache placement, each user desires a total of $\binom{\alpha-1}{t}$ subfiles which are not in its cache. In this case, each user simply desires $\binom{2}{2} = 1$ subfile. The delivery of these subfiles involves the following MAN bitwise XOR

$$X_{123} = \left(\bigoplus_{k \in \mathcal{S}} W_{f_k, \mathcal{K}, \mathcal{S}\setminus\{k\}} : \mathcal{S} \subseteq \mathcal{K}, |\mathcal{S}| = t + 1 \right) \tag{2.104}$$
$$= (W_{1,123,23} \oplus W_{2,123,13} \oplus W_{3,123,12}) \tag{2.105}$$

and then the following two uncoded transmissions

$$X_{45} = (W_{1,124,12}, W_{1,125,12}) \tag{2.106}$$

that serve the users outside $\mathcal{K}$. Given that the subpacketization is $\binom{\alpha}{t} = \binom{3}{2} = 3$, the transmitted signal $X = (X_{123}, X_{45})$ induces a communication load of $R_{\alpha,\mathrm{c}}(2) = |X|/B = 1$, which matches the corresponding optimal $R^{\star}_{\alpha,\mathrm{c}}(2)$ from Proposition 2.1.

2.6 Additional Optimal Schemes for Circular Demands

We here present schemes that optimally deliver circular demands. We will do so for the $(5, 4, F)$ FDS structure with $t \in \{2, 3\}$, and for the $(6, 5, F)$ FDS structure with $t = 3$. The optimal schemes assume the selfish and uncoded cache placement described in Section 2.5.1. We prove optimality simply by showing that the load provided by the proposed achievable schemes matches the converse bound in Theorem 2.1. This suffices because, as we might recall, the construction of the converse employed only circular demands[20].

2.6.1 Circular Demands and the $(5, 4, F)$ FDS Structure

The scheme presented here is a generalization, for any circular demand, of the example in Section 2.2.2. For the considered $(K, \alpha, F) = (5, 4, F)$ structure, we know that there are $C = \binom{K}{\alpha} = 5$ file classes $\mathcal{W}_{1234}, \mathcal{W}_{1235}, \mathcal{W}_{1245}, \mathcal{W}_{1345}, \mathcal{W}_{2345}$, corresponding to $N = FC = 5F$ files. The 5 FDSs take the form

$$\mathcal{F}_1 = \{\mathcal{W}_{1234}, \mathcal{W}_{1235}, \mathcal{W}_{1245}, \mathcal{W}_{1345}\} \tag{2.107}$$
$$\mathcal{F}_2 = \{\mathcal{W}_{1234}, \mathcal{W}_{1235}, \mathcal{W}_{1245}, \mathcal{W}_{2345}\} \tag{2.108}$$
$$\mathcal{F}_3 = \{\mathcal{W}_{1234}, \mathcal{W}_{1235}, \mathcal{W}_{1345}, \mathcal{W}_{2345}\} \tag{2.109}$$
$$\mathcal{F}_4 = \{\mathcal{W}_{1234}, \mathcal{W}_{1245}, \mathcal{W}_{1345}, \mathcal{W}_{2345}\} \tag{2.110}$$
$$\mathcal{F}_5 = \{\mathcal{W}_{1235}, \mathcal{W}_{1245}, \mathcal{W}_{1345}, \mathcal{W}_{2345}\} \tag{2.111}$$

where we recall that $\mathcal{W}_{\mathcal{S}} = \{W_{f,\mathcal{S}} : f \in [F]\}$. The scheme works for any value of $F \in \mathbb{N}^+$, and for any circular demand, recalling that each circular demand is identified by a permutation vector $\hat{\boldsymbol{u}} = (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4, \hat{u}_5) \in H_5$. The FDS request graph of such generic circular demand is shown in Figure 2.6. Notice that, as expected, this is not a complete graph.

Before presenting the schemes, we wish to point out the key novelty and the general idea. As already mentioned in the example in Section 2.2.2, the crucial aspect consists of designing multicast messages that not only deliver messages themselves, but where also their linear combinations can be used by the users to correctly decode requested subfiles. Hence, the idea here is to find an efficient way to align interference, such that users can cache-out interference terms by simply XORing some of the received messages. The pivotal aspect of this approach is the ability of choosing the users that will have to take

[20]We wish to point out that the fact that the converse is matched when considering some instances of circular demands does not necessarily imply that the converse is exactly matched in general nor that circular demands are worst-case demands.

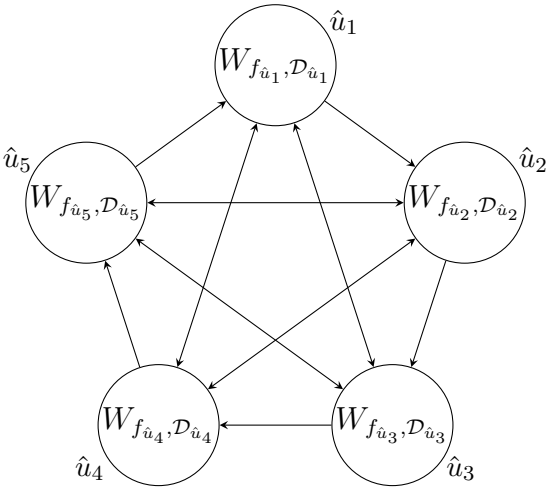

Figure 2.6: FDS request graph for a generic circular demand identified by the vector $\hat{\boldsymbol{u}} = (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4, \hat{u}_5) \in H_5$ and the $(K, \alpha, F) = (5, 4, F)$ FDS structure.

linear combinations of messages to cancel interference and correctly decode the requested subfiles.

The case of $t = 2$

According to the cache placement in Section 2.5.1, each file is split into $\binom{\alpha}{t} = \binom{4}{2} = 6$ non-overlapping subfiles as

$$W_{i,1234} = \{W_{i,1234,12}, W_{i,1234,13}, W_{i,1234,14}, W_{i,1234,23}, W_{i,1234,24}, W_{i,1234,34}\} \tag{2.112}$$

$$W_{i,1235} = \{W_{i,1235,12}, W_{i,1235,13}, W_{i,1235,15}, W_{i,1235,23}, W_{i,1235,25}, W_{i,1235,35}\} \tag{2.113}$$

$$W_{i,1245} = \{W_{i,1245,12}, W_{i,1245,14}, W_{i,1245,15}, W_{i,1245,24}, W_{i,1245,25}, W_{i,1245,45}\} \tag{2.114}$$

$$W_{i,1345} = \{W_{i,1345,13}, W_{i,1345,14}, W_{i,1345,15}, W_{i,1345,34}, W_{i,1345,35}, W_{i,1345,45}\} \tag{2.115}$$

$$W_{i,2345} = \{W_{i,2345,23}, W_{i,2345,24}, W_{i,2345,25}, W_{i,2345,34}, W_{i,2345,35}, W_{i,2345,45}\} \tag{2.116}$$

for each $f \in [F]$. Considering then that the cache content of each user $k \in [5]$ is filled as

$$Z_k = \{W_{f,\mathcal{S},\mathcal{T}} : f \in [F], \mathcal{S} \subseteq [5], |\mathcal{S}| = 4, \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = 2, k \in \mathcal{T}\} \tag{2.117}$$

it can be easily seen that each user desires a total of $\binom{\alpha-1}{t} = \binom{3}{2} = 3$ subfiles, each of size $B/6$ bits. Recalling that each circular demand is defined by a permutation vector $\hat{\boldsymbol{u}} = (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4, \hat{u}_5) \in H_5$, we conclude that the subfiles desired by each user are the following.

- User $\hat{u}_1$. The subfiles which are desired are given by $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3\}}$, $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_4\}}$ and $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_3,\hat{u}_4\}}$.
- User $\hat{u}_2$. The subfiles which are desired are given by $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4\}}$, $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_5\}}$ and $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_4,\hat{u}_5\}}$.
- User $\hat{u}_3$. The subfiles which are desired are given by $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4\}}$, $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_5\}}$ and $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_4,\hat{u}_5\}}$.
- User $\hat{u}_4$. The subfiles which are desired are given by $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2\}}$, $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_5\}}$ and $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_2,\hat{u}_5\}}$.
- User $\hat{u}_5$. The subfiles which are desired are given by $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2\}}$, $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_3\}}$ and $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_2,\hat{u}_3\}}$.

These subfiles are delivered by the following sequence of XORs

$$X_1 = W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3\}} \oplus W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_5\}} \oplus W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4\}} \tag{2.118}$$

$$X_2 = W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4\}} \oplus W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_4\}} \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2\}} \tag{2.119}$$

$$X_3 = W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_5\}} \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_3\}} \oplus W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_5\}} \tag{2.120}$$

$$X_4 = W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_3,\hat{u}_4\}} \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_5\}} \tag{2.121}$$

$$X_5 = W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4\}} \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_2,\hat{u}_5\}} \tag{2.122}$$

$$X_6 = W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2\}} \tag{2.123}$$

$$X_7 = W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_2,\hat{u}_3\}} \tag{2.124}$$

after which each user $k \in [5]$ can employ its own cache content Z_k to decode as follows.

- User $\hat{u}_1$ recovers its desired subfiles from $X_1 \oplus X_3$, X_2 and X_4.
- User $\hat{u}_2$ recovers its desired subfiles from $X_1 \oplus X_2$, X_5 and X_6.
- User $\hat{u}_3$ recovers its desired subfiles from X_1, X_3 and X_7.
- User $\hat{u}_4$ recovers its desired subfiles from X_2, X_4 and X_5.
- User $\hat{u}_5$ recovers its desired subfiles from X_3, X_6 and X_7.

Given that $|X_i| = B/6$ for each $i \in [7]$, and given that there are 7 transmissions, we have a load of $R(2) = |X|/B = 7/6$. Since then $R_{\text{LB}}(2) = 7/6$, we can conclude that the converse is tight.

The case of $t = 3$

In this case each file is split into $\binom{4}{3} = 4$ non-overlapping subfiles as

$$W_{f,1234} = \{W_{f,1234,123}, W_{f,1234,124}, W_{f,1234,134}, W_{f,1234,234}\} \tag{2.125}$$
$$W_{f,1235} = \{W_{f,1235,123}, W_{f,1235,125}, W_{f,1235,135}, W_{f,1235,235}\} \tag{2.126}$$
$$W_{f,1245} = \{W_{f,1245,124}, W_{f,1245,125}, W_{f,1245,145}, W_{f,1245,245}\} \tag{2.127}$$
$$W_{f,1345} = \{W_{f,1345,134}, W_{f,1345,135}, W_{f,1345,145}, W_{f,1345,345}\} \tag{2.128}$$
$$W_{f,2345} = \{W_{f,2345,234}, W_{f,2345,235}, W_{f,2345,245}, W_{f,2345,345}\} \tag{2.129}$$

for each $f \in [F]$. Each user then desires $\binom{3}{3} = 1$ subfile of size $B/4$. More precisely, always considering the general circular demands identified by the vector $\hat{\boldsymbol{u}} \in H_5$, the desired subfiles are given as follows.

- User $\hat{u}_1$ desires $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}}$.
- User $\hat{u}_2$ desires $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4,\hat{u}_5\}}$.
- User $\hat{u}_3$ desires $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4,\hat{u}_5\}}$.
- User $\hat{u}_4$ desires $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2,\hat{u}_5\}}$.
- User $\hat{u}_5$ desires $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2,\hat{u}_3\}}$.

After transmitting the following two XORs

$$X_1 = W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}} \oplus W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2,\hat{u}_5\}} \tag{2.130}$$
$$X_2 = W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}} \oplus W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2,\hat{u}_3\}} \tag{2.131}$$

each user can decode as follows.

- User $\hat{u}_1$ and user $\hat{u}_3$ recover their desired subfiles from X_2.
- User $\hat{u}_2$ and user $\hat{u}_4$ recover their desired subfiles from X_1.
- User $\hat{u}_5$ recovers its desired subfile from $X_1 \oplus X_2$.

Recalling that $|X_1| = |X_2| = B/4$, the 2 transmissions correspond to a communication load $R(3) = |X|/B = 1/2$ which matches the converse $R_{\text{LB}}(3) = 1/2$. This means that the scheme is optimal among all the caching-and-delivery schemes that deliver circular demands.

2.6.2 Circular Demands and the $(6, 5, F)$ FDS Structure

In this setting, we consider the $(K, \alpha, F) = (6, 5, F)$ structure, where there is a total of $C = 6$ classes of files $\mathcal{W}_{12345}, \mathcal{W}_{12346}, \mathcal{W}_{12356}, \mathcal{W}_{12456}, \mathcal{W}_{13456}, \mathcal{W}_{23456}$ and there are $N = FC = 6F$ files. The 6 FDSs take the form

$$\mathcal{F}_1 = \{\mathcal{W}_{12345}, \mathcal{W}_{12346}, \mathcal{W}_{12356}, \mathcal{W}_{12456}, \mathcal{W}_{13456}\} \tag{2.132}$$
$$\mathcal{F}_2 = \{\mathcal{W}_{12345}, \mathcal{W}_{12346}, \mathcal{W}_{12356}, \mathcal{W}_{12456}, \mathcal{W}_{23456}\} \tag{2.133}$$
$$\mathcal{F}_3 = \{\mathcal{W}_{12345}, \mathcal{W}_{12346}, \mathcal{W}_{12356}, \mathcal{W}_{13456}, \mathcal{W}_{23456}\} \tag{2.134}$$
$$\mathcal{F}_4 = \{\mathcal{W}_{12345}, \mathcal{W}_{12346}, \mathcal{W}_{12456}, \mathcal{W}_{13456}, \mathcal{W}_{23456}\} \tag{2.135}$$
$$\mathcal{F}_5 = \{\mathcal{W}_{12345}, \mathcal{W}_{12356}, \mathcal{W}_{12456}, \mathcal{W}_{13456}, \mathcal{W}_{23456}\} \tag{2.136}$$
$$\mathcal{F}_6 = \{\mathcal{W}_{12346}, \mathcal{W}_{12356}, \mathcal{W}_{12456}, \mathcal{W}_{13456}, \mathcal{W}_{23456}\} \tag{2.137}$$

where $\mathcal{W}_{\mathcal{S}} = \{W_{f,\mathcal{S}} : f \in [F]\}$. As in the previous case, we here provide a scheme for any $F \in \mathbb{N}^+$ and any circular demand. Each such circular demand is identified by a vector $\hat{\boldsymbol{u}} = (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4, \hat{u}_5, \hat{u}_6) \in H_6$, and it induces the FDS request graph in Figure 2.7. The optimal scheme is provided for the case $t = 3$.

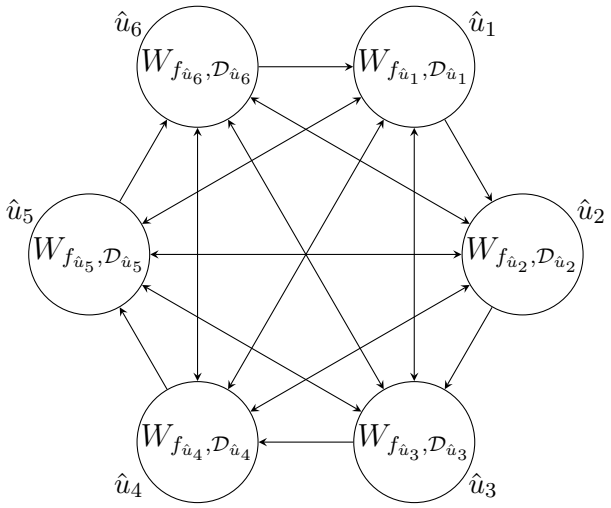

Figure 2.7: FDS request graph for a generic circular demand identified by the vector $\hat{\boldsymbol{u}} = (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4, \hat{u}_5, \hat{u}_6) \in H_6$ and the $(K, \alpha, F) = (6, 5, F)$ FDS structure.

According to the cache placement in Section 2.5.1, each file is split into $\binom{\alpha}{t} = \binom{5}{3} = 10$ non-overlapping subfiles as

$$W_{f,\mathcal{S}} = \{W_{f,\mathcal{S},\mathcal{T}} : \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = 3\} \tag{2.138}$$

for each $\mathcal{S} \subseteq [6]$ such that $|\mathcal{S}| = 5$ and for each $f \in [F]$, where each subfile has size $B/10$. For example, the file $W_{f,12345}$ is split into 10 non-overlapping subfiles $W_{f,12345,\mathcal{T}}$ for each $\mathcal{T} \in \{123, 124, 125, 134, 135, 145, 234, 235, 245, 345\}$. We recall that the set $\mathcal{T}$ represents the users which the subfile $W_{f,12345,\mathcal{T}}$ is exactly and uniquely cached at. If we consider a generic circular demand, each user misses $\binom{\alpha-1}{t} = \binom{4}{3} = 4$ subfiles given by the following.

- The subfiles $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}}$, $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_5\}}$, $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_4,\hat{u}_5\}}$ and $W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_3,\hat{u}_4,\hat{u}_5\}}$ are desired by user $\hat{u}_1$.
- The subfiles $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4,\hat{u}_5\}}$, $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4,\hat{u}_6\}}$, $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_5,\hat{u}_6\}}$ and $W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_4,\hat{u}_5,\hat{u}_6\}}$ are desired by user $\hat{u}_2$
- The subfiles $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4,\hat{u}_5\}}$, $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4,\hat{u}_6\}}$, $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_5,\hat{u}_6\}}$ and $W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_4,\hat{u}_5,\hat{u}_6\}}$ are desired by user $\hat{u}_3$.
- The subfiles $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2,\hat{u}_5\}}$, $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2,\hat{u}_6\}}$, $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_5,\hat{u}_6\}}$ and $W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_2,\hat{u}_5,\hat{u}_6\}}$ are desired by user $\hat{u}_4$.
- The subfiles $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2,\hat{u}_3\}}$, $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2,\hat{u}_6\}}$, $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_3,\hat{u}_6\}}$ and $W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_2,\hat{u}_3,\hat{u}_6\}}$ are desired by user $\hat{u}_5$.
- The subfiles $W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_1,\hat{u}_2,\hat{u}_3\}}$, $W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_1,\hat{u}_2,\hat{u}_4\}}$, $W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_1,\hat{u}_3,\hat{u}_4\}}$ and $W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}}$ are desired by user $\hat{u}_6$.

If we consider the following linear combinations of subfiles

$$\begin{aligned}
X_1 &= W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}} \oplus W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_4,\hat{u}_5,\hat{u}_6\}} \\
&\quad \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_2,\hat{u}_5,\hat{u}_6\}} && (2.139)\\
X_2 &= W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_5,\hat{u}_6\}} \oplus W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_5\}} \\
&\quad \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2,\hat{u}_3\}} && (2.140)\\
X_3 &= W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}} \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_5,\hat{u}_6\}} \\
&\quad \oplus W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_1,\hat{u}_3,\hat{u}_4\}} \oplus W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4,\hat{u}_6\}} && (2.141)\\
X_4 &= W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_5,\hat{u}_6\}} \\
&\quad \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_3,\hat{u}_6\}} && (2.142)\\
X_5 &= W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2,\hat{u}_6\}} \oplus W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4,\hat{u}_6\}} \\
&\quad \oplus W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_2,\hat{u}_3,\hat{u}_4\}} && (2.143)\\
X_6 &= W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_1,\hat{u}_2,\hat{u}_6\}} \\
&\quad \oplus W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_2,\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2,\hat{u}_5\}} && (2.144)\\
X_7 &= W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_4,\hat{u}_5,\hat{u}_6\}} \oplus W_{f_{\hat{u}_4},\mathcal{D}_{\hat{u}_4},\{\hat{u}_1,\hat{u}_2,\hat{u}_6\}} \\
&\quad \oplus W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_1,\hat{u}_2,\hat{u}_4\}} && (2.145)
\end{aligned}$$

$$X_8 = W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_1,\hat{u}_2,\hat{u}_3\}} \oplus W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_1,\hat{u}_4,\hat{u}_5\}} \oplus W_{f_{\hat{u}_1},\mathcal{D}_{\hat{u}_1},\{\hat{u}_3,\hat{u}_4,\hat{u}_5\}} \tag{2.146}$$

$$X_9 = W_{f_{\hat{u}_3},\mathcal{D}_{\hat{u}_3},\{\hat{u}_4,\hat{u}_5,\hat{u}_6\}} \oplus W_{f_{\hat{u}_6},\mathcal{D}_{\hat{u}_6},\{\hat{u}_1,\hat{u}_2,\hat{u}_3\}} \oplus W_{f_{\hat{u}_2},\mathcal{D}_{\hat{u}_2},\{\hat{u}_3,\hat{u}_5,\hat{u}_6\}} \oplus W_{f_{\hat{u}_5},\mathcal{D}_{\hat{u}_5},\{\hat{u}_2,\hat{u}_3,\hat{u}_6\}} \tag{2.147}$$

and we denote by $X = (X_i : i \in [9])$ the concatenated message sent by the central server, then each user can correctly decode its desired subfiles as follows.

- User $\hat{u}_1$ recovers its desired subfiles from X_2, X_3, $X_4 \oplus X_6$ and X_8.
- User $\hat{u}_2$ recovers its desired subfiles from X_1, X_5, X_6 and $X_7 \oplus X_9$.
- User $\hat{u}_3$ recovers its desired subfiles from $X_2 \oplus X_3$, X_4, X_8 and X_9.
- User $\hat{u}_4$ recovers its desired subfiles from X_1, X_3, $X_5 \oplus X_6$ and X_7.
- User $\hat{u}_5$ recovers its desired subfiles from X_2, X_4, X_6 and $X_8 \oplus X_9$.
- User $\hat{u}_6$ recovers its desired subfiles from $X_1 \oplus X_3$, X_5, X_7 and X_9.

The delivery procedure is slightly more involved with respect to the previous FDS structure, but is based as before on the idea of carefully aligning interference. Indeed, the messages X_i are carefully designed in such a way that also their linear combinations can be useful to some users. An equivalent interpretation of this fact is related to the previously mentioned creation of cliques. Consider for example the XOR X_3. User 1 and user 4 can directly cache-out interference to correctly decode their desired subfiles, while user 3 and user 6 miss in their cache — due to the selfish cache placement — some interfering messages appearing in the XOR X_3. Such interfering (and consequently undesired) messages are "delivered" to[21] both user 3 and user 6 by means of XORs X_2 and X_1, so allowing them to decode the desired subfiles from X_3. We can see here that with X_3 we are able serve the clique composed by user 1, user 4, user 3 and user 6, by carefully "passing" some undesired (and not cached) information to the last two users. A similar reasoning applies to the XORs X_6 and X_9, both of which are useful to 4 users simultaneously.

The communication load is equal to $R(3) = |X|/B = 9/10$ and it matches the converse $R_{\text{LB}}(3) = 9/10$. Hence, the converse here is tight.

[21]We recall that we use "deliver" in quotes when referring to undesired subfiles because users do not actually decode undesired subfiles, but rather they take linear combinations of the multicast messages to cancel out interference terms.

Chapter 3

Coded Caching With Tactical User Profiles

In this chapter, we first propose an extremely broad system model for the user profiles, where such model follows the combinatorial structure of the so-called tactical configurations*, a symmetric and balanced block design in combinatorial mathematics. Then, we characterize the fundamental limits of coded caching under uncoded prefetching for the proposed broad structure of user interests. The interesting outcome of the results presented in this chapter is that, despite the diverging interests of the users, the MAN coded scheme is still order optimal within a constant factor of* 4*. This result is significant not only because it holds for a really generic structure of user profiles — indeed, the proposed tactical configurations include some other well-known structures — but also because it shows, further again, the power of the combinatorial MAN coded caching scheme under a well-defined system model for the user profiles.*

3.1 System Model

We consider the centralized caching scenario, where a central server is connected to K users through an error-free broadcast channel and each user is equipped with a cache of size MB bits. Further, the server has access to a main catalogue of N files of B bits each.

3.1.1 Description of the System Model

We proceed now with the description of the structure for the user interests that will be studied in this chapter. First, we assume that $N \geq K$. Further, we assume that the N files in the system are split in G groups containing

N/G files each, where we naturally assume that $G \mid N$. This implies that the library can be partitioned as

$$\{W_1, \dots, W_N\} = \{\mathcal{W}_{\mathcal{K}_1}, \dots, \mathcal{W}_{\mathcal{K}_G}\} \tag{3.1}$$

where $\mathcal{K}_g \subseteq [K]$ is the subset of users which are interested in the set of files given by $\mathcal{W}_{\mathcal{K}_g}$ for some $g \in [G]$. Then, we assume that each user is interested into G' distinct groups of files, whereas each file is of interest to exactly K' users, namely, we assume that $|\{g : g \in [G], k \in \mathcal{K}_g\}| = G'$ for each $k \in [K]$ and that $|\mathcal{K}_g| = K'$ for each $g \in [G]$. This implies that the equation $GK' = KG'$ holds, where it is assumed that $G, G', K', K \in \mathbb{N}^+$. Further, the above implies that the FDS $\mathcal{F}_k$ of user $k \in [K]$ is given by

$$\mathcal{F}_k = \{\mathcal{W}_{\mathcal{K}_g} : g \in [G], k \in \mathcal{K}_g\}. \tag{3.2}$$

Notice that, since each user is interested into G' distinct groups of N/G files and each file is of interest to exactly K' users, also the equation $KN' = NK'$ holds where $N' = G'N/G$. Hence, we have

$$\frac{K'}{K} = \frac{N'}{N} = \frac{G'}{G}. \tag{3.3}$$

Remark 3.1. The aforementioned structure for the interests of the users corresponds to a *configuration* in geometry or to a t-design with $t = 1$ in combinatorial mathematics[1]. Hence, we will refer in the following to such FDS structure as tactical configuration with parameters (K, G, G', N), where $K' = KG'/G$. More in general, we will refer to such problem as coded caching with tactical user profiles with parameters (K, G, G', N). Finally, we trivially have $G' \in [G]$ and $K' \in [K]$.

In the following, we provide two clarifying examples that can help familiarize the reader with the system model introduced above. Interestingly, the examples will show how some other well-studied combinatorial structures can be seen as tactical configurations.

Example 3.1. Consider the symmetric (K, α, F) FDS structure that we analyzed in Chapter 2. For such structure, the library consists of a total of $C = \binom{K}{\alpha}$ classes of files and $N = FC = F\binom{K}{\alpha}$ files in total. Each file is of interest to α users and each user is interested into $F\binom{K-1}{\alpha-1}$ files. This structure can be seen as a tactical configuration with parameters $\left(K, G = \binom{K}{\alpha}, G' = \binom{K-1}{\alpha-1}, N = F\binom{K}{\alpha}\right)$, where $K' = K\binom{K-1}{\alpha-1}/\binom{K}{\alpha} = \alpha$.

[1]Notice that any t-design with $t \geq 2$ is also a 1-design. Hence, we are considering here all symmetric FDS structures induced by t-designs. Please, also notice that the t variable for t-designs does not coincide with the normalized cumulative cache size $t = KM/N$ used in the caching literature.

Example 3.2. Consider the setting where we have K users and $G = K$ groups of files. Then, assume that the library of N files is partitioned as

$$\{W_1, \ldots, W_N\} = \{\mathcal{W}_{\mathcal{K}_1}, \ldots, \mathcal{W}_{\mathcal{K}_K}\} \tag{3.4}$$

where $\mathcal{K}_g = [g : g+K'-1]_K$ for each $g \in [K]$ and for some $K' \in [K-1]$. We can see that there are $G = K$ groups of files, each file is of interest to K' users and each user is interested into $G' = K'$ groups of files. This structure corresponds to a tactical configuration with parameters $(K, G = K, G' = K', N)$, where $K' = KG'/G = G'$.

Remark 3.2. We wish to point out that the aforementioned system model based on tactical configurations is rather broad. Indeed, it encompasses both the symmetric FDS structure already analyzed in Chapter 2, as well as other structures commonly employed in the literature, as the cyclic structure in Example 3.2.

Deviating from standard notation practices, we will use in this chapter W_{d_k} to denote the file requested by user $k \in [K]$. Clearly, any user $k \in [K]$ will request a file W_{d_k} as long as $W_{d_k} \in \mathcal{F}_k$. In addition, we denote by $\mathcal{D}$ the set of demand vectors with distinct requested files, i.e., $W_{d_{k_1}} \neq W_{d_{k_2}}$ for each $k_1, k_2 \in [K]$ with $k_1 \neq k_2$.

Denoting by $\mathcal{X}$ the set of caching schemes with uncoded placement, the server transmits during the delivery phase a message X of $R(\boldsymbol{d}, \chi, M)B$ bits for a given demand $\boldsymbol{d} \in \mathcal{D}$, a given uncoded cache placement $\chi \in \mathcal{X}$ and some given memory M. Our goal is to characterize the optimal worst-case communication load under uncoded cache placement, namely, we aim to characterize the quantity given by

$$R^{\star}_{\mathrm{u}}(M) = \min_{\chi \in \mathcal{X}} \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{d}, \chi, M). \tag{3.5}$$

In the following, the dependency on M will be often implied for the sake of simplicity.

3.1.2 A Genie-Aided Converse Bound

We will provide our converse bound on the optimal worst-case load under uncoded prefetching using the genie-aided approach in [4]. Consider a demand vector $\boldsymbol{d} \in \mathcal{D}$ and let $\boldsymbol{u} = (u_1, \ldots, u_K) \in S_K$ be a permutation of the set $[K]$. Denoting by Z_k the cache content of user $k \in [K]$, we can construct a genie-aided user with the following cache content

$$Z' = \left(Z_{u_k} \setminus \left(\bigcup_{i \in [k-1]} Z_{u_i} \cup W_{d_{u_i}} \right) : k \in [K] \right) \tag{3.6}$$

which is enough for such genie-aided user to inductively decode all the requested files from (X, Z'). Consequently, the following

$$\begin{aligned} R(\boldsymbol{d},\chi)B &\geq H(X) && (3.7)\\ &\geq H(X \mid Z') && (3.8)\\ &\geq I\left(\left\{W_{d_{u_k}}\right\}_{k\in[K]}; X \mid Z'\right) && (3.9)\\ &= H\left(\left\{W_{d_{u_k}}\right\}_{k\in[K]} \mid Z'\right) && (3.10)\\ &= \sum_{k\in[K]} \sum_{\mathcal{T}\in([K]\setminus\{u_1,\dots,u_k\})} \left|W_{d_{u_k},\mathcal{T}}\right| && (3.11) \end{aligned}$$

holds, which means that we have the following lower bound

$$R(\boldsymbol{d},\chi) \geq \sum_{k\in[K]} \sum_{\mathcal{T}\in([K]\setminus\{u_1,\dots,u_k\})} \frac{\left|W_{d_{u_k},\mathcal{T}}\right|}{B} \tag{3.12}$$

on the communication load for a given[2] $\boldsymbol{d}\in\mathcal{D}$ and $\chi\in\mathcal{X}$. Since it will be of use later, we define the following

$$R_{\mathrm{LB}}(\boldsymbol{d},\boldsymbol{u},\chi) \coloneqq \sum_{k\in[K]} \sum_{\mathcal{T}\in([K]\setminus\{u_1,\dots,u_k\})} \frac{\left|W_{d_{u_k},\mathcal{T}}\right|}{B}. \tag{3.13}$$

For the sake of completeness, notice that the bound in (3.12) can be equivalently obtained following the index coding approach in [3], already largely employed in Chapter 2.

3.2 Main Results

In this section, we present the main results obtained for coded caching with tactical user profiles. The first result is an achievable bound, which is described in the following theorem.

Theorem 3.1 (Achievable Bound)**.** *Consider the coded caching problem with tactical user profiles with parameters (K, G, G', N). Then, the optimal worst-case load $R^{\star}_{\mathrm{u}}$ is upper bounded by the lower convex envelope of the following memory-load corner points*

$$(M, R_{\mathrm{UB}}) = \left(t\frac{N}{K}, g(t)\right), \quad \forall t\in[0:K'] \tag{3.14}$$

[2]We recall that the dependency on M is implied for the sake of simplicity.

where $g(t)$ is defined as

$$g(t) \coloneqq \begin{cases} \frac{\binom{K}{t+1}}{\binom{K}{t}}, & \text{if } t \in [0:\bar{t}] \\ \frac{\binom{K}{\bar{t}+1}}{\binom{K}{\bar{t}}} \frac{K'-t}{K'-\bar{t}}, & \text{if } t \in [\bar{t}+1:K'] \end{cases} \tag{3.15}$$

and $\bar{t} \coloneqq \lfloor K'/2 \rfloor$.

Proof. When $t \leq \bar{t}$, the achievable load is obtained by simply applying the MAN placement-and-delivery scheme. Instead, when $t > \bar{t}$, the achievable load is obtained by means of memory sharing between the point $(\bar{t}, g(\bar{t}))$ and the point $(K'N/K, 0)$. Indeed, when $t = K'$, we have that $M = K'N/K = G'N/G$ recalling that $GK' = KG'$. Hence, further recalling that each user is interested into G' groups of files containing N/G files each, the memory-load point $(G'N/G, 0)$ is trivially achievable, as each user has enough memory to store the entirety of their FDS. The proof is concluded. □

The second result is a lower bound on the optimal worst-case load under uncoded prefetching. The result is presented in the following theorem, whose proof is presented in Section 3.3.1.

Theorem 3.2 (Converse Bound)**.** *Consider the coded caching problem with tactical user profiles with parameters (K, G, G', N). Then, the optimal worst-case load $R_{\mathrm{u}}^{\star}$ is lower bounded by the lower convex envelope of the following memory-load corner points*

$$(M, R_{\mathrm{LB}}) = \left(t\frac{N}{K}, \frac{G}{G'} \frac{\binom{K'}{t+1}}{\binom{K'}{t}} \right), \quad \forall t \in [0:K']. \tag{3.16}$$

If compare the results in Theorem 3.1 and in Theorem 3.2, we obtain the order optimality result in the following theorem, whose proof is presented in Section 3.3.2.

Theorem 3.3 (Order Optimality)**.** *The achievable worst-case load in Theorem 3.1 is order optimal within a constant multiplicative factor of 4 under uncoded prefetching.*

Finally, we can provide the following corollary for the case $G' = 1$, which implies each user to be interested in only one groups of files.

Corollary 3.3.1. *When $G' = 1$, a disjoint MAN placement-and-delivery scheme is exactly optimal under uncoded prefetching.*

Proof. Recalling that the library can be partitioned as

$$\{W_1, \dots, W_N\} = \{\mathcal{W}_{\mathcal{K}_1}, \dots, \mathcal{W}_{\mathcal{K}_G}\} \tag{3.17}$$

the condition $G' = 1$ implies that $\mathcal{K}_{g_1} \cap \mathcal{K}_{g_2} = \emptyset$ for each $g_1, g_2 \in [G]$ with $g_1 \neq g_2$. Hence, we can treat each group of N' files and the associated K' users as an independent coded caching problem, for a total of G disjoint MAN caching problems with memory $M = tN'/K' = tN/K$. As a consequence, we can guarantee an achievable load R_{UB} given by the lower convex envelope of the memory-load points

$$(M, R_{\text{UB}}) = \left(t\frac{N}{K}, G\frac{\binom{K'}{t+1}}{\binom{K'}{t}} \right), \quad \forall t \in [0 : K'] \tag{3.18}$$

which coincides with the result in Theorem 3.2 when $G' = 1$. The proof is concluded. □

3.3 Collection of Proofs

3.3.1 Proof of Theorem 3.2

We recall that our goal is to lower bound the quantity

$$R_{\text{u}}^{\star} = \min_{\chi \in \mathcal{X}} \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{d}, \chi). \tag{3.19}$$

where again the dependency on M is implied to simplify the notation. We will use the following lemma to proceed.

Lemma 3.1. *Consider the coded caching problem with tactical user profiles with parameters (K, G, G', N). There exists a subset of N' demands $\mathcal{D}_{N'} \subset \mathcal{D}$ such that each user $k \in [K]$ requests each file $W_n \in \mathcal{F}_k$ in their FDS exactly once within the set $\mathcal{D}_{N'}$.*

Proof. The problem of coded caching with tactical user profiles can be represented as a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$, where $\mathcal{U} = [K]$ is the set of users, $\mathcal{V} = [N]$ is the set of file indices, and $\mathcal{E} = \{(k, n) : k \in [K], n \in [N], W_n \in \mathcal{F}_k\}$ is the set of edges. As we are considering a tactical configuration for the user profiles, we have that the degree of each vertex $k \in [K]$ is equal to N', whereas the degree of each vertex $n \in [N]$ is equal to K'. Since we know that $N/N' = K/K'$ and we further assumed that $N \geq K$, we know that the largest vertex degree is equal to $N' \geq K'$.

Now, we want to prove that there exists a subset $\mathcal{D}_{N'} \subset \mathcal{D}$ of N' demands such that any user $k \in [K]$ requests each file in $\mathcal{F}_k$ exactly once. This is equivalent to prove that the chromatic index of the bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ is equal to N', namely, we wish to prove that there exists a proper edge coloring with N' colors $\mathcal{C} = \{c_1, \ldots, c_{N'}\}$. Indeed, if such edge coloring exists, then for each color $c \in \mathcal{C}$ we can construct the demand vector $\boldsymbol{d}^c = (d_1^c, \ldots, d_K^c)$, where for each $k \in [K]$ we have $d_k^c \in [N]$ such that the edge $(k, d_k^c) \in \mathcal{E}$ is colored by c. Notice that, since we are considering an edge coloring with N' colors, then for each $k \in [K]$ we have $d_k^{c_1} \neq d_k^{c_2}$ for any $c_1, c_2 \in \mathcal{C}$ with $c_1 \neq c_2$; also, for each $c \in \mathcal{C}$ we have $d_{k_1}^c \neq d_{k_2}^c$ for any $k_1, k_2 \in [K]$ with $k_1 \neq k_2$. Hence, if we collect all such vectors in $\mathcal{D}_{N'} = \{\boldsymbol{d}^c : c \in \mathcal{C}\}$, such construction guarantees that for each user $k \in [K]$ we have exactly one demand in $\mathcal{D}_{N'}$ for which user k requests any one specific file in $\mathcal{F}_k$.

Thanks to König's theorem [64, Theorem 5.18], we know that the chromatic index of any bipartite graph is equal to its largest vertex degree, and since the largest vertex degree is equal to N' in our case, then there exists a proper edge coloring of $\mathcal{G}$ with N' colors. Hence, we can construct a proper subset $\mathcal{D}_{N'} \subset \mathcal{D}$ of N' demands — as we described above — such that any user $k \in [K]$ requests each file in $\mathcal{F}_k$ exactly once. This concludes the proof. □

By means of Lemma 3.1, we proceed to lower bound the optimal worst-case load as follows

$$
\begin{aligned}
R_{\mathrm{u}}^{\star} &= \min_{\chi \in \mathcal{X}} \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{d}, \chi) && (3.20)\\
&\geq \min_{\chi \in \mathcal{X}} \max_{\boldsymbol{d} \in \mathcal{D}_{N'}} R(\boldsymbol{d}, \chi) && (3.21)\\
&\geq \min_{\chi \in \mathcal{X}} \frac{1}{|\mathcal{D}_{N'}|} \sum_{\boldsymbol{d} \in \mathcal{D}_{N'}} R(\boldsymbol{d}, \chi) && (3.22)\\
&= \min_{\chi \in \mathcal{X}} \frac{1}{N'} \sum_{\boldsymbol{d} \in \mathcal{D}_{N'}} R(\boldsymbol{d}, \chi) && (3.23)
\end{aligned}
$$

where (3.21) holds because $\mathcal{D}_{N'} \subset \mathcal{D}$ and (3.22) follows from the fact that the maximum can be lower bounded by the average.

The Combinatorial Counting Argument

Now, as we observed in Section 3.1.2, the communication load can be lower bounded, for a given demand $\boldsymbol{d}$ and a given caching scheme χ, as in (3.12). Hence, if we construct the inequality in (3.12) for each demand $\boldsymbol{d} \in \mathcal{D}_{N'}$ and for each permutation of users $\boldsymbol{u} \in S_K$, and then we sum together all such

inequalities, we obtain

$$K! \sum_{\boldsymbol{d} \in \mathcal{D}_{N'}} R(\boldsymbol{d}, \chi) \geq \sum_{(\boldsymbol{d}, \boldsymbol{u}) \in (\mathcal{D}_{N'}, S_K)} R_{\mathrm{LB}}(\boldsymbol{d}, \boldsymbol{u}, \chi) \tag{3.24}$$

which can be further rewritten as

$$\frac{1}{N'} \sum_{\boldsymbol{d} \in \mathcal{D}_{N'}} R(\boldsymbol{d}, \chi) \geq \frac{1}{K! N'} \sum_{(\boldsymbol{d}, \boldsymbol{u}) \in (\mathcal{D}_{N'}, S_K)} R_{\mathrm{LB}}(\boldsymbol{d}, \boldsymbol{u}, \chi). \tag{3.25}$$

Now, towards simplifying the expression in (3.25), we proceed by counting how many times each subfile $W_{n,\mathcal{T}}$ — for any given $n \in [N]$ and $\mathcal{T} \subseteq [K]$ — appears in the RHS of (3.25).

First, we focus on the subfile $W_{n,\mathcal{T}_i}$ for a given $n \in [N]$ and $\mathcal{T}_i \subseteq [K]$ with $|\mathcal{T}_i| = t'$ for some $t' \in [0:K]$, where i denotes the number of users having W_n in their FDS and appearing in the set $\mathcal{T}_i$, namely, $i = |\{k : k \in \mathcal{T}_i, W_n \in \mathcal{F}_k\}|$. Then, recalling that each file is of interest to K' users in our coded caching problem with tactical user profiles, we let k be one of the $(K' - i)$ users that do not appear in $\mathcal{T}_i$. From Lemma 3.1, we know that there exists exactly one demand in $\mathcal{D}_{N'}$ for which user k requests the file W_n. In addition, for such demand all permutations $\boldsymbol{u} \in S_K$ are considered. Nevertheless, we can notice from the construction of (3.12) that the subfile $W_{n,\mathcal{T}_i}$ appears in the RHS of (3.25) only when k is located before the elements from $\mathcal{T}_i$ in the permutation vector $\boldsymbol{u}$. Since there is a total of $(K - 1 - t')!t'!\binom{K}{t'+1}$ such vectors, the subfile $W_{n,\mathcal{T}_i}$ appears $(K - 1 - t')!t'!\binom{K}{t'+1}$ times in the RHS of (3.25) when we consider the demand for which the file W_n is requested by such user k. The same reasoning holds for any of the $(K' - i)$ users not appearing in $\mathcal{T}_i$, so the subfile $W_{n,\mathcal{T}_i}$ appears $(K' - i)(K - 1 - t')!t'!\binom{K}{t'+1}$ in the RHS of (3.25). In addition, since this reasoning holds for each $n \in [N]$ and $\mathcal{T}_i \subseteq [K]$ where $i \in [\max(0, |\mathcal{T}_i| - K + K'), \min(|\mathcal{T}_i|, K')]$, we can rewrite the RHS of (3.25) as

$$\frac{N}{K! N'} \sum_{t' \in [0:K]} \sum_{i=\max(0, t'-K+K')}^{\min(t', K')} (K' - i)(K - 1 - t')!t'! \binom{K}{t'+1} x_{t',i} \tag{3.26}$$

$$= \frac{G}{G'} \sum_{t' \in [0:K]} \sum_{i=\max(0, t'-K+K')}^{\min(t', K')} \frac{K' - i}{t' + 1} x_{t',i} \tag{3.27}$$

where $x_{t',i}$ is defined as

$$0 \leq x_{t',i} \coloneqq \sum_{n \in [N]} \sum_{\substack{\mathcal{T}_i \subseteq [K] : |\mathcal{T}_i| = t', \\ |\{k : k \in \mathcal{T}_i, W_n \in \mathcal{F}_k\}| = i}} \frac{|W_{n,\mathcal{T}_i}|}{BN}. \tag{3.28}$$

We can further lower bound the RHS of (3.25) as follows

$$\frac{G}{G'}\sum_{t'\in[0:K]}\sum_{i=\max(0,t'-K+K')}^{\min(t',K')}\frac{K'-i}{t'+1}x_{t',i} \tag{3.29}$$

$$\geq\frac{G}{G'}\sum_{t'\in[0:K]}\sum_{i=\max(0,t'-K+K')}^{\min(t',K')}\frac{K'-\min(t',K')}{t'+1}x_{t',i} \tag{3.30}$$

$$=\frac{G}{G'}\sum_{t'\in[0:K]}\frac{K'-\min(t',K')}{t'+1}\sum_{i=\max(0,t'-K+K')}^{\min(t',K')}x_{t',i} \tag{3.31}$$

$$=\frac{G}{G'}\sum_{t'\in[0:K]}\frac{\binom{K'}{t'+1}}{\binom{K'}{t'}}x_{t'} \tag{3.32}$$

where $x_{t'}$ is defined as

$$0\leq x_{t'}\coloneqq\sum_{i=\max(0,t'-K+K')}^{\min(t',K')}x_{t',i}. \tag{3.33}$$

Towards the Final Lower Bound

After the passages above, we can further lower bound the optimal worst-case load as

$$R_{\mathrm{u}}^{\star}\geq\min_{\chi\in\mathcal{X}}\frac{1}{N'}\sum_{\boldsymbol{d}\in\mathcal{D}_{N'}}R(\boldsymbol{d},\chi) \tag{3.34}$$

$$\geq\min_{\chi\in\mathcal{X}}\sum_{t'\in[0:K]}f(t')x_{t'} \tag{3.35}$$

where $f(t')$ is defined as

$$f(t')\coloneqq\frac{G}{G'}\frac{\binom{K'}{t'+1}}{\binom{K'}{t'}}. \tag{3.36}$$

Moreover, for any uncoded cache placement $\chi\in\mathcal{X}$, the following

$$\sum_{t'\in[0:K]}x_{t'}=1 \tag{3.37}$$

$$\sum_{t'\in[0:K]}t'x_{t'}\leq\frac{KM}{N} \tag{3.38}$$

holds. This means that we can consider $\boldsymbol{x}=(x_0,\ldots,x_K)$ as a probability mass function with constraint in (3.38) on the first moment, where such

constraint simply represents the maximum memory that is available across the caches of all users. In light of the above, we can write

$$R_{\mathrm{u}}^{\star} \geq \min_{\chi \in \mathcal{X}} \sum_{t' \in [0:K]} f(t') x_{t'} \tag{3.39}$$

$$= \min_{\boldsymbol{x}} \mathbb{E}_{\boldsymbol{x}} \left[f(t') \right] \tag{3.40}$$

$$\geq \min_{\boldsymbol{x}} f(\mathbb{E}_{\boldsymbol{x}}[t']) \tag{3.41}$$

$$\geq \min_{\boldsymbol{x}} f(KM/N) \tag{3.42}$$

$$= f(t) \tag{3.43}$$

where $t = KM/N$. Notice that, since $f(t')$ is convex and decreasing in t', we have (3.41) and (3.42) from Jensen's inequality and the constraint on the first moment, respectively.

Now, when t is an integer, the bound is simply the function $f(t)$ evaluated[3] at $t \in [0 : K']$. For non-integer values of t, we can follow again the reasoning in [4], [39], [52] to take the lower convex envelope of the sequence of points $\{(t, f(t)) : t \in [0 : K']\}$. To conclude, the optimal worst-case load $R_{\mathrm{u}}^{\star}$ is lower bounded by the lower convex envelope of the following memory-load corner points

$$(M, R_{\mathrm{LB}}) = \left(t \frac{N}{K}, \frac{G}{G'} \frac{\binom{K'}{t+1}}{\binom{K'}{t}} \right), \quad \forall t \in [0 : K']. \tag{3.44}$$

The proof is complete. □

3.3.2 Proof of Theorem 3.3

Consider the range $t \in [0 : \bar{t}]$. Then, from Theorem 3.1 and Theorem 3.2 it holds that

$$\frac{R_{\mathrm{UB}}}{R_{\mathrm{u}}^{\star}} \leq \frac{R_{\mathrm{UB}}}{R_{\mathrm{LB}}} \tag{3.45}$$

$$= \frac{\binom{K}{t+1}}{\binom{K}{t}} \frac{\binom{K'}{t}}{\binom{K'}{t+1}} \frac{G'}{G} \tag{3.46}$$

$$= \frac{K - t}{K' - t} \frac{G'}{G} \tag{3.47}$$

$$\leq \frac{G'}{G} \frac{K - \bar{t}}{K' - \bar{t}} \tag{3.48}$$

[3]Notice that $f(t) = 0$ whenever $t \geq K'$.

where the last inequality holds since the term $(K-t)/(K'-t)$ is increasing[4] with increasing t. Now, consider the range $t \in [\bar{t}+1 : K']$. Then, it holds that

$$\frac{R_{\mathrm{UB}}}{R_{\mathrm{u}}^{\star}} \leq \frac{R_{\mathrm{UB}}}{R_{\mathrm{LB}}} \tag{3.49}$$

$$= \frac{\binom{K}{\bar{t}+1}}{\binom{K}{\bar{t}}} \frac{K'-t}{K'-\bar{t}} \frac{\binom{K'}{t}}{\binom{K'}{t+1}} \frac{G'}{G} \tag{3.50}$$

$$= \frac{G'}{G} \frac{(K-\bar{t})(1+t)}{(1+\bar{t})(K'-\bar{t})} \tag{3.51}$$

$$= \frac{G'}{G} \frac{(K-\bar{t})(1+K')}{(1+\bar{t})(K'-\bar{t})} \tag{3.52}$$

where the last inequality holds since also the term $(1+t)$ is increasing with increasing t. Now, considering that

$$\frac{G'}{G} \frac{K-\bar{t}}{K'-\bar{t}} \leq \frac{G'}{G} \frac{(K-\bar{t})(1+K')}{(1+\bar{t})(K'-\bar{t})} \tag{3.53}$$

we can conclude that over the entire range $t \in [0 : K']$ it holds that

$$\frac{R_{\mathrm{UB}}}{R_{\mathrm{u}}^{\star}} \leq \frac{G'}{G} \frac{(K-\bar{t})(1+K')}{(1+\bar{t})(K'-\bar{t})}. \tag{3.54}$$

If K' is even, we have $\bar{t} = K'/2$, and so we can see that

$$\frac{R_{\mathrm{UB}}}{R_{\mathrm{u}}^{\star}} \leq \frac{G'}{G} \frac{(K-\bar{t})(1+K')}{(1+\bar{t})(K'-\bar{t})} \tag{3.55}$$

$$\leq \frac{G'K}{G} \frac{1+K'}{(1+K'/2)K'/2} \tag{3.56}$$

$$= 4\frac{G'K}{G} \frac{1+K'}{(2+K')K'} \tag{3.57}$$

$$\leq 4\frac{G'K}{GK'} \tag{3.58}$$

$$= 4. \tag{3.59}$$

If K' is odd, we have $\bar{t} = K'/2 - 1/2$, and so we can see that

$$\frac{R_{\mathrm{UB}}}{R_{\mathrm{u}}^{\star}} \leq \frac{G'}{G} \frac{(K-\bar{t})(1+K')}{(1+\bar{t})(K'-\bar{t})} \tag{3.60}$$

[4]This can be easily checked by showing that the first derivative is non-negative.

$$\leq \frac{G'K}{G} \frac{1+K'}{(1/2+K'/2)K'/2} \tag{3.61}$$

$$= 4\frac{G'K}{GK'} \tag{3.62}$$

$$= 4. \tag{3.63}$$

We can then conclude that $R_{\mathrm{UB}}/R^{\star}_{\mathrm{u}} \leq 4$ for all $t \in [0 : K']$. This concludes the proof. □

Chapter 4

A Converse for Caching With Heterogeneous Preferences

In this chapter, we consider the coded caching scenario heterogeneous user profiles originally proposed in [32], for which the end-receiving users are divided into groups according to their file preferences. Taking advantage of the genie-aided converse bound idea from [4], we develop a novel information-theoretic converse on the worst-case communication load under uncoded cache placement. Interestingly, the developed converse bound, jointly with one of the coded schemes proposed in [32], allows us to characterize the optimal worst-case communication load under uncoded prefetching within a constant multiplicative gap of 2*. Although we restrict the caching policy to be uncoded, this result improves the previously known order optimality results for the considered caching problem.*

4.1 System Model and Related Results

We always consider the coded caching setting where there is a single server connected to K users through an error-free broadcast channel. The server has access to a central library that contains N files of B bits each and each user in the system is equipped with a cache of size MB bits.

4.1.1 Description of the System Model

According to the system model in [32], the K users are split in G groups, where each group consists of K/G users sharing the same interests. Furthermore, the files in the library are divided in two categories, i.e., common files and unique files.

- There are N_{c} common files $\{W_n^{\mathrm{c}} : n \in [N_{\mathrm{c}}]\}$, where each of them is of interest to every user in the system.
- Then, for each group $g \in [G]$, there are N_{u} unique files $\{W_n^{\mathrm{u},g} : n \in [N_{\mathrm{u}}]\}$, where each of them is of interest to the users belonging to the group $g \in [G]$ only.

Assuming that $\{W_n^{\mathrm{c}} : n \in [N_{\mathrm{c}}]\} \cap \{W_n^{\mathrm{u},g} : n \in [N_{\mathrm{u}}]\} = \emptyset$ for each $g \in [G]$, and that $\{W_n^{\mathrm{u},g_1} : n \in [N_{\mathrm{u}}]\} \cap \{W_n^{\mathrm{u},g_2} : n \in [N_{\mathrm{u}}]\} = \emptyset$ for each $g_1, g_2 \in [G]$ with $g_1 \neq g_2$, we have $N = N_{\mathrm{c}} + GN_{\mathrm{u}}$ files in total.

Deviating from standard notation practices, we will use in this chapter $W_{d_k}^{f_k,g(k)}$ to denote the file requested by user $k \in [K]$, where $f_k \in \{\mathrm{c}, \mathrm{u}\}$, $d_k \in [N_{f_k}]$ and $g(k)$ is an abuse of notation to denote the group which user k belongs to, i.e., $g(k) \in [G]$ for each $k \in [K]$. We further assume that $W_{d_k}^{\mathrm{c},g(k)} = W_{d_k}^{\mathrm{c}}$, since common files do not depend on the group $g \in [G]$. In addition, we let $\boldsymbol{d} = ((d_1, f_1), \ldots, (d_K, f_K))$ be the demand vector and we denote by $\mathcal{D}$ the set of all possible demand vectors with distinct requested files, i.e., $W_{d_{k_1}}^{f_{k_1},g(k_1)} \neq W_{d_{k_2}}^{f_{k_2},g(k_2)}$ for each $k_1, k_2 \in [K]$ with $k_1 \neq k_2$. Finally, we assume $N_{\mathrm{c}} \geq K$ and $N_{\mathrm{u}} \geq K/G$.

Denoting by $\mathcal{X}$ the set of caching schemes with uncoded placement, the server transmits during the delivery phase a message X of $R(\boldsymbol{d}, \chi, M)B$ bits for a given demand $\boldsymbol{d} \in \mathcal{D}$, a given uncoded cache placement $\chi \in \mathcal{X}$ and some given memory M. Our goal is to characterize the optimal worst-case communication load under uncoded cache placement, namely, we aim to characterize the quantity given by

$$R_{\mathrm{u}}^{\star}(M) = \min_{\chi \in \mathcal{X}} \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{d}, \chi, M). \tag{4.1}$$

In the following, the dependency on M will be implied for the sake of simplicity.

4.1.2 An Existing Achievable Scheme

The authors in [32] proposed for the aforementioned setting a coded scheme — referred to as Scheme 2 in [32] — which treats separately the caching and the delivery of common and unique files.

Placement phase First, the cache of each user is split in two parts for some $0 \leq \beta \leq 1$, so that βM is the part of cache that is devoted to store common files and $(1-\beta)M$ is the part of cache that is devoted to store unique files. Then, common files $\{W_n^{\mathrm{c}} : n \in [N_{\mathrm{c}}]\}$ are stored across the K users using the MAN cache placement with memory βM. Similarly, unique files $\{W_n^{\mathrm{u},g} : n \in [N_{\mathrm{u}}]\}$ are stored across the K/G users in group $g \in [G]$ using the MAN algorithm with memory $(1-\beta)M$.

Delivery phase It was shown in [32] that, when there are α users per group requesting unique files, the optimal worst-case load can be upper bounded as

$$R_{\mathrm{u}}^{\star} \leq \min_{\beta} \max_{\alpha} R(\beta, \alpha) \tag{4.2}$$

where $R(\beta, \alpha)$ is defined as

$$R(\beta, \alpha) \coloneqq \frac{\binom{K}{t_{\mathrm{c}}+1} - \binom{G\alpha}{t_{\mathrm{c}}+1}}{\binom{K}{t_{\mathrm{c}}}} + G\frac{\binom{K/G}{t_{\mathrm{u}}+1} - \binom{K/G-\alpha}{t_{\mathrm{u}}+1}}{\binom{K/G}{t_{\mathrm{u}}}} \tag{4.3}$$

with $t_{\mathrm{c}} \coloneqq K\beta M/N_{\mathrm{c}}$ and $t_{\mathrm{u}} \coloneqq K(1-\beta)M/GN_{\mathrm{u}}$.

Since the works in [32], [34] treated the variables K, G, N_{c}, N_{u} and $t = KM/N$ as continuous[1], we do the same here for the sake of simplicity. Further, we extend the Scheme 2 in [32] to the entire memory regime $0 \leq M \leq N_{\mathrm{c}} + N_{\mathrm{u}}$, using the Gamma function whenever the binomial coefficients in (4.3) have non-integer arguments.

4.1.3 A Genie-Aided Converse Bound

Similarly to what we did in Chapter 3, we will provide our converse bound on the optimal worst-case load under uncoded prefetching using the genie-aided approach in [4]. Consider a demand vector $\boldsymbol{d} \in \mathcal{D}$ and let $\boldsymbol{u} = (u_1, \ldots, u_K) \in S_K$ be a permutation of the set $[K]$. Denoting by Z_k the cache content of user $k \in [K]$, we can construct a genie-aided user with the following cache content

$$Z' = \left(Z_{u_k} \setminus \left(\bigcup_{i \in [k-1]} Z_{u_i} \cup W_{d_{u_i}}^{f_{u_i}, g(u_i)} \right) : k \in [K] \right) \tag{4.4}$$

which is enough for such genie-aided user to inductively decode all the requested files from (X, Z'). Consequently, the following

$$\begin{aligned} R(\boldsymbol{d}, \chi)B &\geq H(X) && (4.5)\\ &\geq H(X \mid Z') && (4.6)\\ &\geq I\left(\left\{ W_{d_{u_k}}^{f_{u_k}, g(u_k)} \right\}_{k \in [K]} ; X \mid Z' \right) && (4.7)\\ &= H\left(\left\{ W_{d_{u_k}}^{f_{u_k}, g(u_k)} \right\}_{k \in [K]} \mid Z' \right) && (4.8) \end{aligned}$$

[1]Indeed, if the quantities K, G, N_{c} and N_{u} are large enough, the rounding errors due to integer effects during calculations can be neglected.

$$= \sum_{k\in[K]} \sum_{\mathcal{T}\in([K]\setminus\{u_1,\dots,u_k\})} \left| W_{d_{u_k},\mathcal{T}}^{f_{u_k},g(u_k)} \right| \tag{4.9}$$

holds, which means that we have the following lower bound

$$R(\boldsymbol{d},\chi) \geq \sum_{k\in[K]} \sum_{\mathcal{T}\in([K]\setminus\{u_1,\dots,u_k\})} \frac{\left| W_{d_{u_k},\mathcal{T}}^{f_{u_k},g(u_k)} \right|}{B} \tag{4.10}$$

on the communication load for a given[2] $\boldsymbol{d}\in\mathcal{D}$ and $\chi\in\mathcal{X}$. Since it will be of use later, we define the following

$$R_{\mathrm{LB}}(\boldsymbol{d},\boldsymbol{u},\chi) \coloneqq \sum_{k\in[K]} \sum_{\mathcal{T}\in([K]\setminus\{u_1,\dots,u_k\})} \frac{\left| W_{d_{u_k},\mathcal{T}}^{f_{u_k},g(u_k)} \right|}{B}. \tag{4.11}$$

Once again, for the sake of completeness, notice that the bound in (4.10) can be equivalently obtained following the index coding approach in [3], already largely employed in Chapter 2.

4.2 Main Results

The first result provides a converse bound on the optimal worst-case load under uncoded prefetching. The proof is presented in Section 4.3.1.

Theorem 4.1. *For the coded caching problem with heterogeneous user profiles presented in Section 4.1, the optimal worst-case load under uncoded cache placement is lower bounded as*

$$R_{\mathrm{u}}^{\star} \geq \min_{\beta} \frac{1}{2}\left(\frac{\binom{K}{t_{\mathrm{c}}+1}}{\binom{K}{t_{\mathrm{c}}}} + G\frac{\binom{K/G}{t_{\mathrm{u}}+1}}{\binom{K/G}{t_{\mathrm{u}}}} \right) \tag{4.12}$$

where $t_{\mathrm{c}} = K\beta M/N_{\mathrm{c}}$ *and* $t_{\mathrm{u}} = K(1-\beta)M/GN_{\mathrm{u}}$.

If we compare the achievable performance in (4.2) with the converse in Theorem 4.1, we can provide the following optimality result, whose proof is described in Section 4.3.2.

Theorem 4.2. *The achievable load in* (4.2) *is order optimal within a multiplicative factor of* 2.

[2]We recall that the dependency on M is implied for the sake of simplicity.

Remark 4.1. The result in Theorem 4.2 improves the previously known order optimality results presented in [34]. Indeed, even though the work in [34] provided a converse bound without constraining the placement to be uncoded, the smallest gap to optimality therein was a constant factor 8 for the limited memory regime $N/K \leq M \leq N/2G$. Moreover, the achievable performance in (4.2) was shown to be within a multiplicative factor of $8 + 8K/G$ from optimal for the memory regime $G(N_{\mathrm{c}}+N_{\mathrm{u}})/K \leq M \leq N/2G$. Here, although our converse holds under the assumption of uncoded placement, we provide a gap to optimality which is a constant multiplicative factor of 2 for the entire[3] memory regime $0 \leq M \leq N_{\mathrm{c}} + N_{\mathrm{u}}$.

4.3 Collection of Proofs

In this section, we provide the proof for both Theorem 4.1 and Theorem 4.2. For the former, we recall that our goal is to lower bound the quantity

$$R_{\mathrm{u}}^{\star} = \min_{\chi \in \mathcal{X}} \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{d}, \chi). \tag{4.13}$$

where again the dependency on M is implied to simplify the notation.

4.3.1 Proof of Theorem 4.1

Denote by $\mathcal{D}_{\mathrm{c}}$ the subset of $\mathcal{D}$ that contains all demands for which users make requests only from common files, which implies $\boldsymbol{d} = ((d_1, \mathrm{c}), \ldots, (d_K, \mathrm{c}))$ for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}$. Similarly, denote by $\mathcal{D}_{\mathrm{u}}$ the subset of $\mathcal{D}$ for which users make requests only from unique files, which implies $\boldsymbol{d} = ((d_1, \mathrm{u}), \ldots, (d_K, \mathrm{u}))$ for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{u}}$. One can see that $|\mathcal{D}_{\mathrm{c}}| = \binom{N_{\mathrm{c}}}{K} K!$ and $|\mathcal{D}_{\mathrm{u}}| = \left(\binom{N_{\mathrm{u}}}{K/G}(K/G)!\right)^G$. Then, we proceed to lower bound the optimal worst-case load as follows

$$\begin{aligned}
R_{\mathrm{u}}^{\star} &= \min_{\chi \in \mathcal{X}} \max_{\boldsymbol{d} \in \mathcal{D}} R(\boldsymbol{d}, \chi) && (4.14)\\
&\geq \min_{\chi \in \mathcal{X}} \max \left(\max_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}} R(\boldsymbol{d}, \chi), \max_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{u}}} R(\boldsymbol{d}, \chi) \right) && (4.15)\\
&\geq \min_{\chi \in \mathcal{X}} \frac{1}{2} \left(\max_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}} R(\boldsymbol{d}, \chi) + \max_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{u}}} R(\boldsymbol{d}, \chi) \right) && (4.16)\\
&\geq \min_{\chi \in \mathcal{X}} \frac{1}{2} \left(\frac{1}{|\mathcal{D}_{\mathrm{c}}|} \sum_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}} R(\boldsymbol{d}, \chi) + \frac{1}{|\mathcal{D}_{\mathrm{u}}|} \sum_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{u}}} R(\boldsymbol{d}, \chi) \right) && (4.17)
\end{aligned}$$

[3]The bound in Theorem 4.1 becomes 0 only when it holds $t_{\mathrm{c}} = K$ and $t_{\mathrm{u}} = K/G$ simultaneously. This happens when $\beta = N_{\mathrm{c}}/M$ and $(1-\beta) = N_{\mathrm{u}}/M$, which implies $0 \leq M \leq N_{\mathrm{c}} + N_{\mathrm{u}}$. In addition, we recall that the Scheme 2 in [32] is extended to the entire memory regime $0 \leq M \leq N_{\mathrm{c}} + N_{\mathrm{u}}$.

$$= \min_{\chi \in \mathcal{X}} \frac{1}{2} \left(R_{\mathrm{c}}(\chi) + R_{\mathrm{u}}(\chi) \right) \tag{4.18}$$

where $R_{\mathrm{c}}(\chi)$ and $R_{\mathrm{u}}(\chi)$ are defined as

$$R_{\mathrm{c}}(\chi) \coloneqq \frac{1}{|\mathcal{D}_{\mathrm{c}}|} \sum_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}} R(\boldsymbol{d}, \chi) \tag{4.19}$$

$$R_{\mathrm{u}}(\chi) \coloneqq \frac{1}{|\mathcal{D}_{\mathrm{u}}|} \sum_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{u}}} R(\boldsymbol{d}, \chi). \tag{4.20}$$

Notice that (4.15) holds because $(\mathcal{D}_{\mathrm{c}} \cup \mathcal{D}_{\mathrm{u}}) \subset \mathcal{D}$, whereas both (4.16) and (4.17) follow from the fact that the maximum can be lower bounded by the average.

We proceed now to lower bound separately $R_{\mathrm{c}}(\chi)$ and $R_{\mathrm{u}}(\chi)$ by means of the genie-aided approach in Section 4.1.3.

Lower Bounding $R_{\mathrm{c}}(\chi)$

As we observed in Section 4.1.3, the communication load can be lower bounded, for a given demand $\boldsymbol{d}$ and a given caching scheme χ, as in (4.10). Hence, if we construct the inequality in (4.10) for each demand $\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}$ and for each permutation of users $\boldsymbol{u} \in S_K$, and then we sum together all such inequalities, we obtain

$$K! \sum_{\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}} R(\boldsymbol{d}, \chi) \geq \sum_{(\boldsymbol{d}, \boldsymbol{u}) \in (\mathcal{D}_{\mathrm{c}}, S_K)} R_{\mathrm{LB}}(\boldsymbol{d}, \boldsymbol{u}, \chi) \tag{4.21}$$

which can be further rewritten as

$$R_{\mathrm{c}}(\chi) \geq \frac{1}{K! |\mathcal{D}_{\mathrm{c}}|} \sum_{(\boldsymbol{d}, \boldsymbol{u}) \in (\mathcal{D}_{\mathrm{c}}, S_K)} R_{\mathrm{LB}}(\boldsymbol{d}, \boldsymbol{u}, \chi) \tag{4.22}$$

recalling that $\boldsymbol{d} = ((d_1, \mathrm{c}), \ldots, (d_K, \mathrm{c}))$ for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{c}}$ and that $W_n^{\mathrm{c},g} = W_n^{\mathrm{c}}$ for each $n \in [N_{\mathrm{c}}]$ and for each $g \in [G]$. Now, towards simplifying the expression in (4.22), we proceed by counting how many times each subfile $W_{n,\mathcal{T}}^{\mathrm{c}}$ — for any given $n \in [N_{\mathrm{c}}]$ and $\mathcal{T} \subseteq [K]$ — appears in the RHS of (4.22).

First, we focus on the subfile $W_{n,\mathcal{T}}^{\mathrm{c}}$ for some $n \in [N_{\mathrm{c}}]$ and $\mathcal{T} \subseteq [K]$ such that $|\mathcal{T}| = t'$ with $t' \in [0 : K]$. Next, we denote by $\mathcal{D}_{\mathrm{c},n,k}$ the subset of demands in $\mathcal{D}_{\mathrm{c}}$ for which the file W_n^{c} is requested by some specific user $k \in ([K] \setminus \mathcal{T})$. We can see that $|\mathcal{D}_{\mathrm{c},n,k}| = \binom{N_{\mathrm{c}}}{K} K!/N_{\mathrm{c}} = |\mathcal{D}_{\mathrm{c}}|/N_{\mathrm{c}}$. Then, we observe that, for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{c},n,k}$, all permutations of users $\boldsymbol{u} \in S_K$ are considered. Nevertheless, we can notice from the construction of (4.10) that, for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{c},n,k}$, the subfile $W_{n,\mathcal{T}}^{\mathrm{c}}$ appears in the RHS of (4.22) only for those permutations of users where k appears before the elements from the set $\mathcal{T}$ in the permutation vector $\boldsymbol{u}$. Since there is a total of $(K - 1 - t')! t'! \binom{K}{t'+1}$

such vectors, we can conclude that the subfile $W^{\mathrm{c}}_{n,\mathcal{T}}$ appears in the RHS of (4.22) a total of $|\mathcal{D}_{\mathrm{c}}|(K-1-t')!t'!\binom{K}{t'+1}/N_{\mathrm{c}}$ times when we consider the demands in $\mathcal{D}_{\mathrm{c},n,k}$ only. Then, since the reasoning above holds for each user $k \in ([K] \setminus \mathcal{T})$, we can conclude that the subfile $W^{\mathrm{c}}_{n,\mathcal{T}}$ appears in the RHS of (4.22) a total of $|\mathcal{D}_{\mathrm{c}}|(K-t')!t'!\binom{K}{t'+1}/N_{\mathrm{c}}$ times. Moreover, we considered a generic subfile $W^{\mathrm{c}}_{n,\mathcal{T}}$, so the above holds for any $n \in [N_{\mathrm{c}}]$ and for any $\mathcal{T} \subseteq [K]$. Therefore, we can rewrite the RHS of (4.22) as

$$\frac{1}{K!|\mathcal{D}_{\mathrm{c}}|} \sum_{t' \in [0:K]} |\mathcal{D}_{\mathrm{c}}|(K-t')!t'!\binom{K}{t'+1} x^{\mathrm{c}}_{t'} \tag{4.23}$$

where $x^{\mathrm{c}}_{t'}$ is defined as

$$0 \leq x^{\mathrm{c}}_{t'} \coloneqq \sum_{n \in [N_{\mathrm{c}}]} \sum_{\mathcal{T} \subseteq [K]:|\mathcal{T}|=t'} \frac{\left|W^{\mathrm{c}}_{n,\mathcal{T}}\right|}{BN_{\mathrm{c}}}. \tag{4.24}$$

After some algebraic manipulations, we can rewrite (4.22) as

$$R_{\mathrm{c}}(\chi) \geq \sum_{t' \in [0:K]} f_{\mathrm{c}}(t') x^{\mathrm{c}}_{t'} \tag{4.25}$$

where $f_{\mathrm{c}}(t')$ is defined as

$$f_{\mathrm{c}}(t') \coloneqq \frac{\binom{K}{t'+1}}{\binom{K}{t'}}. \tag{4.26}$$

Lower Bounding $R_{\mathrm{u}}(\chi)$

Applying as before the genie-aided approach from Section 4.1.3, we obtain the following inequality

$$R_{\mathrm{u}}(\chi) \geq \frac{1}{K!|\mathcal{D}_{\mathrm{u}}|} \sum_{(\boldsymbol{d},\boldsymbol{u}) \in (\mathcal{D}_{\mathrm{u}}, S_K)} R_{\mathrm{LB}}(\boldsymbol{d}, \boldsymbol{u}, \chi) \tag{4.27}$$

recalling that now $\boldsymbol{d} = ((d_1, \mathrm{u}), \ldots, (d_K, \mathrm{u}))$ for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{u}}$. Once again, towards simplifying the expression in (4.27), we proceed by counting how many times each subfile $W^{\mathrm{u},g}_{n,\mathcal{T}}$ — for any given $n \in [N_{\mathrm{u}}]$, $g \in [G]$ and $\mathcal{T} \subseteq [K]$ — appears in the RHS of (4.27).

First, we focus on the subfile $W^{\mathrm{u},g}_{n,\mathcal{T}_i}$ for a given $g \in [G]$, $n \in [N_{\mathrm{u}}]$ and $\mathcal{T}_i \subseteq [K]$ with $|\mathcal{T}_i| = t'$ for some $t' \in [0:K]$, where i denotes the number of users from group g that appear in the set $\mathcal{T}_i$, namely, $i = |\{k \in \mathcal{T}_i : g(k) = g\}|$. Then, we let k be one of the $(K/G - i)$ users from group

g that do not appear in $\mathcal{T}_i$ and we further assume that the file $W_n^{\mathrm{u},g}$ is requested by such user k. If we denote by $\mathcal{D}_{\mathrm{u},n,k}^g$ the subset of demands in $\mathcal{D}_{\mathrm{u}}$ for which the file $W_n^{\mathrm{u},g}$ is requested by this user k, we can see that $|\mathcal{D}_{\mathrm{u},n,k}^g| = \left(\binom{N_{\mathrm{u}}}{K/G}(K/G)!\right)^G / N_{\mathrm{u}} = |\mathcal{D}_{\mathrm{u}}|/N_{\mathrm{u}}$. In addition, for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{u},n,k}^g$, all permutations $\boldsymbol{u} \in S_K$ are considered. Nevertheless, as already observed, the subfile $W_{n,\mathcal{T}_i}^{\mathrm{u},g}$ appears in the RHS of (4.27), for each $\boldsymbol{d} \in \mathcal{D}_{\mathrm{u},n,k}^g$, only when k is located before the elements from $\mathcal{T}_i$ in the permutation vector $\boldsymbol{u}$. Since there is a total of $(K-1-t')!t'!\binom{K}{t'+1}$ such vectors, the subfile $W_{n,\mathcal{T}_i}^{\mathrm{u},g}$ appears $|\mathcal{D}_{\mathrm{u}}|(K-1-t')!t'!\binom{K}{t'+1}/N_{\mathrm{u}}$ times in the RHS of (4.27) when only the demands $\mathcal{D}_{\mathrm{u},n,k}^g$ are considered. The same reasoning holds for any of the $(K/G-i)$ users from group g not appearing in $\mathcal{T}_i$, so the subfile $W_{n,\mathcal{T}_i}^{\mathrm{u},g}$ appears $|\mathcal{D}_{\mathrm{u}}|(K/G-i)(K-1-t')!t'!\binom{K}{t'+1}/N_{\mathrm{u}}$ in the RHS of (4.27). In addition, since this reasoning holds for each $g \in [G]$, $n \in [N_{\mathrm{u}}]$ and $\mathcal{T}_i \subseteq [K]$ where $i \in [\max(0, |\mathcal{T}_i| - K + K/G), \min(|\mathcal{T}_i|, K/G)]$, after some algebraic manipulations we can rewrite the RHS of (4.27) as

$$\sum_{t' \in [0:K]} \sum_{i=\max(0,t'-K+K/G)}^{\min(t',K/G)} \frac{(K/G-i)}{t'+1} x_{t',i}^{\mathrm{u}} \tag{4.28}$$

where $x_{t',i}^{\mathrm{u}}$ is defined as

$$0 \leq x_{t',i}^{\mathrm{u}} \coloneqq \sum_{g \in [G]} \sum_{n \in [N_{\mathrm{u}}]} \sum_{\substack{\mathcal{T}_i \subseteq [K]: |\mathcal{T}_i| = t', \\ |\{k \in \mathcal{T}_i : g(k) = g\}| = i}} \frac{\left|W_{n,\mathcal{T}_i}^{\mathrm{u},g}\right|}{BN_{\mathrm{u}}}. \tag{4.29}$$

We can further lower bound the RHS of (4.27) as follows

$$\sum_{t' \in [0:K]} \sum_{i=\max(0,t'-K+K/G)}^{\min(t',K/G)} \frac{(K/G-i)}{t'+1} x_{t',i}^{\mathrm{u}} \tag{4.30}$$

$$\geq \sum_{t' \in [0:K]} \sum_{i=\max(0,t'-K+K/G)}^{\min(t',K/G)} \frac{(K/G-\min(t',K/G))}{t'+1} x_{t',i}^{\mathrm{u}} \tag{4.31}$$

$$= \sum_{t' \in [0:K]} \frac{(K/G-\min(t',K/G))}{t'+1} \sum_{i=\max(0,t'-K+K/G)}^{\min(t',K/G)} x_{t',i}^{\mathrm{u}} \tag{4.32}$$

$$= \sum_{t' \in [0:K]} G \frac{\binom{K/G}{t'+1}}{\binom{K/G}{t'}} x_{t'}^{\mathrm{u}} \tag{4.33}$$

where $x_{t'}^{\mathrm{u}}$ is defined as

$$0 \leq x_{t'}^{\mathrm{u}} \coloneqq \sum_{i=\max(0,t'-K+K/G)}^{\min(t',K/G)} \frac{x_{t',i}^{\mathrm{u}}}{G}. \tag{4.34}$$

After the passages above, we can rewrite (4.27) as

$$R_{\mathrm{u}}(\chi) \geq \sum_{t' \in [0:K]} f_{\mathrm{u}}(t') x_{t'}^{\mathrm{u}} \tag{4.35}$$

where $f_{\mathrm{u}}(t')$ is defined as

$$f_{\mathrm{u}}(t') \coloneqq G \frac{\binom{K/G}{t'+1}}{\binom{K/G}{t'}}. \tag{4.36}$$

Lower Bounding $R_{\mathbf{u}}^{\star}$

Finally, we can lower bound the optimal worst-case load $R_{\mathrm{u}}^{\star}$. Indeed, we have the following

$$R_{\mathrm{u}}^{\star} \geq \min_{\chi \in \mathcal{X}} \frac{1}{2} \left(R_{\mathrm{c}}(\chi) + R_{\mathrm{u}}(\chi) \right) \tag{4.37}$$

$$\geq \min_{\chi \in \mathcal{X}} \frac{1}{2} \left(\sum_{t' \in [0:K]} f_{\mathrm{c}}(t') x_{t'}^{\mathrm{c}} + \sum_{t' \in [0:K]} f_{\mathrm{u}}(t') x_{t'}^{\mathrm{u}} \right). \tag{4.38}$$

Moreover, for any uncoded cache placement $\chi \in \mathcal{X}$ and for some $0 \leq \beta \leq 1$, the following

$$\sum_{t' \in [0:K]} x_{t'}^{\mathrm{c}} = 1 \tag{4.39}$$

$$\sum_{t' \in [0:K]} t' x_{t'}^{\mathrm{c}} \leq \frac{K \beta M}{N_{\mathrm{c}}} \tag{4.40}$$

holds for common files, whereas we have the following

$$\sum_{t' \in [0:K]} x_{t'}^{\mathrm{u}} = 1 \tag{4.41}$$

$$\sum_{t' \in [0:K]} t' x_{t'}^{\mathrm{u}} \leq \frac{K(1-\beta) M}{G N_{\mathrm{u}}} \tag{4.42}$$

for unique files. This means that we can consider $\boldsymbol{x}^{\mathrm{c}} = (x_0^{\mathrm{c}}, \ldots, x_K^{\mathrm{c}})$ and $\boldsymbol{x}^{\mathrm{u}} = (x_0^{\mathrm{u}}, \ldots, x_K^{\mathrm{u}})$ as probability mass functions with constraints in (4.40) and (4.42) on the first moment, where such constraints simply represent the maximum memory that is available across the caches of all users for common files and unique files, respectively. In light of the above, we can write

$$R_{\mathrm{u}}^{\star} \geq \min_{\chi \in \mathcal{X}} \frac{1}{2} \left(\sum_{t' \in [0:K]} f_{\mathrm{c}}(t') x_{t'}^{\mathrm{c}} + \sum_{t' \in [0:K]} f_{\mathrm{u}}(t') x_{t'}^{\mathrm{u}} \right) \tag{4.43}$$

$$= \min_{\beta, \boldsymbol{x}^{\mathrm{c}}, \boldsymbol{x}^{\mathrm{u}}} \frac{1}{2} \left(\mathbb{E}_{\boldsymbol{x}^{\mathrm{c}}} \left[f_{\mathrm{c}}(t') \right] + \mathbb{E}_{\boldsymbol{x}^{\mathrm{u}}} \left[f_{\mathrm{u}}(t') \right] \right) \tag{4.44}$$

$$\geq \min_{\beta, \boldsymbol{x}^{\mathrm{c}}, \boldsymbol{x}^{\mathrm{u}}} \frac{1}{2} \left(f_{\mathrm{c}}(\mathbb{E}_{\boldsymbol{x}^{\mathrm{c}}} \left[t' \right]) + f_{\mathrm{u}}(\mathbb{E}_{\boldsymbol{x}^{\mathrm{u}}} \left[t' \right]) \right) \tag{4.45}$$

$$\geq \min_{\beta} \frac{1}{2} \left(f_{\mathrm{c}}(t_{\mathrm{c}}) + f_{\mathrm{u}}(t_{\mathrm{u}}) \right) \tag{4.46}$$

$$= \min_{\beta} \frac{1}{2} \left(\frac{\binom{K}{t_{\mathrm{c}}+1}}{\binom{K}{t_{\mathrm{c}}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}+1}}{\binom{K/G}{t_{\mathrm{u}}}} \right). \tag{4.47}$$

Notice that, since both $f_{\mathrm{c}}(t')$ and $f_{\mathrm{u}}(t')$ are convex and decreasing in t', we have (4.45) and (4.46) from Jensen's inequality and the constraints on the first moment, respectively. The proof is complete. □

4.3.2 Proof of Theorem 4.2

From Theorem 4.1 we have

$$R_{\mathrm{u}}^{\star} \geq \min_{\beta} \frac{1}{2} \left(\frac{\binom{K}{t_{\mathrm{c}}+1}}{\binom{K}{t_{\mathrm{c}}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}+1}}{\binom{K/G}{t_{\mathrm{u}}}} \right) \tag{4.48}$$

$$= \frac{1}{2} \left(\frac{\binom{K}{t_{\mathrm{c}}^{\star}+1}}{\binom{K}{t_{\mathrm{c}}^{\star}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}^{\star}+1}}{\binom{K/G}{t_{\mathrm{u}}^{\star}}} \right) \tag{4.49}$$

where $t_{\mathrm{c}}^{\star} = K\beta^{\star}M/N_{\mathrm{c}}$ and $t_{\mathrm{u}}^{\star} = K(1-\beta^{\star})M/GN_{\mathrm{u}}$ for some optimal $\beta^{\star}$. Further, from [32] we have

$$R_{\mathrm{u}}^{\star} \leq \min_{\beta} \max_{\alpha} R(\beta, \alpha) \tag{4.50}$$

$$\leq \max_{\alpha} R(\beta^{\star}, \alpha) \tag{4.51}$$

$$= \max_{\alpha} \frac{\binom{K}{t_{\mathrm{c}}^{\star}+1} - \binom{G\alpha}{t_{\mathrm{c}}^{\star}+1}}{\binom{K}{t_{\mathrm{c}}^{\star}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}^{\star}+1} - \binom{K/G-\alpha}{t_{\mathrm{u}}^{\star}+1}}{\binom{K/G}{t_{\mathrm{u}}^{\star}}} \tag{4.52}$$

$$\leq \frac{\binom{K}{t_{\mathrm{c}}^{\star}+1}}{\binom{K}{t_{\mathrm{c}}^{\star}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}^{\star}+1}}{\binom{K/G}{t_{\mathrm{u}}^{\star}}} \tag{4.53}$$

where the inequality in (4.51) holds since the optimal value $\beta^{\star}$, which minimizes the lower bound in Theorem 4.1, is not necessarily the optimal memory splitting for the scheme in Section 4.1.2. To conclude, we have

$$\frac{1}{2} \left(\frac{\binom{K}{t_{\mathrm{c}}^{\star}+1}}{\binom{K}{t_{\mathrm{c}}^{\star}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}^{\star}+1}}{\binom{K/G}{t_{\mathrm{u}}^{\star}}} \right) \leq R_{\mathrm{u}}^{\star} \leq \frac{\binom{K}{t_{\mathrm{c}}^{\star}+1}}{\binom{K}{t_{\mathrm{c}}^{\star}}} + G \frac{\binom{K/G}{t_{\mathrm{u}}^{\star}+1}}{\binom{K/G}{t_{\mathrm{u}}^{\star}}} \tag{4.54}$$

which implies that the coded scheme in Section 4.1.2 is order optimal within a constant multiplicative factor of 2. The proof is complete. □

Chapter 5

Coded Distributed Computing With Structured Support

The purpose of this brief and coincise chapter is that of showing how the results presented in Chapter 3 can be employed in the setting of coded distributed computing. The results presented greatly benefited from fruitful discussions with Prof. Kai Wan.

5.1 System Model and Main Results

In this section, we first start with the general formulation of the coded distributed computing problem with structured support. Then, we specialize the general system model imposing a symmetric structure that takes inspiration from the caching problem with tactical user profiles from Chapter 3. Finally, we present the main results.

5.1.1 The General Formulation

Let us consider the distributed computing problem with K computing nodes and N input files, where $K, N \in \mathbb{N}^+$. Each file $w_n \in \mathbb{F}_{2^F}$ with $n \in [N]$ has F bits for some $F \in \mathbb{N}^+$. The goal of node $k \in [K]$ is to compute the output function ϕ_k given by

$$\phi_k \colon \mathbb{F}_{2^F}^{|\mathcal{N}_k|} \to \mathbb{F}_{2^B} \tag{5.1}$$

which maps the files in $\mathcal{N}_k \subseteq \{w_1, \ldots, w_N\}$ to a bit stream $u_k = \phi_k(w_n : w_n \in \mathcal{N}_k) \in \mathbb{F}_{2^B}$ of length B bits for some $B \in \mathbb{N}^+$. We assume that each output function ϕ_k with $k \in [K]$ is *decomposable*, which implies

$$\phi_k(w_n : w_n \in \mathcal{N}_k) = h_k(g_{k,n}(w_n) : w_n \in \mathcal{N}_k). \tag{5.2}$$

Simply, for each $k \in [K]$ we have a map function $g_{k,n}\colon \mathbb{F}_{2^F} \to \mathbb{F}_{2^T}$, which maps the input file $w_n \in \mathcal{N}_k$ into an intermediate value (IV) $v_{k,n} = g_{k,n}(w_n) \in \mathbb{F}_{2^T}$ of T bits, and a reduce function $h_k\colon \mathbb{F}_{2^T}^{|\mathcal{N}_k|} \to \mathbb{F}_{2^B}$, which maps the IVs $v_{k,n} = g_{k,n}(w_n)$ for each $w_n \in \mathcal{N}_k$ into the output value $u_k = h_k(v_{k,n} : w_n \in \mathcal{N}_k) \in \mathbb{F}_{2^B}$ of B bits.

We wish to point out that when $\mathcal{N}_k = \{w_1, \ldots, w_N\}$ for each $k \in [K]$, the problem becomes the original distributed computing setting already studied in [52]. Else, we refer to the problem as the distributed computing problem with structured support.

5.1.2 The Symmetric Case

In the following, we will focus on a specific structure for what concerns the sets $\mathcal{N}_k$ for each $k \in [K]$. In particular, we first assume that the N input files are split in G groups containing N/G files each. This implies

$$\{w_1, \ldots, w_N\} = \{\mathcal{W}_{\mathcal{K}_1}, \ldots, \mathcal{W}_{\mathcal{K}_G}\} \tag{5.3}$$

where $\mathcal{W}_{\mathcal{K}_g}$ represents the g-th group containing N/G files and $\mathcal{K}_g \subseteq [K]$ with $|\mathcal{K}_g| = K'$ and $g \in [G]$. Then, we assume that the group of files $\mathcal{W}_{\mathcal{K}_g}$ is needed by the $K' < K$ output functions in $\mathcal{K}_g$ for each $g \in [G]$, and that each function $k \in [K]$ depends on $G' < G$ groups of files, i.e., $|\{\mathcal{K}_g : g \in [G], k \in \mathcal{K}_g\}| = G'$ for each $k \in [K]$. This means that we have

$$KG' = GK'. \tag{5.4}$$

The above implies $\mathcal{N}_k = \{\mathcal{W}_{\mathcal{K}_g} : g \in [G], k \in \mathcal{K}_g\}$ for each $k \in [K]$.

5.1.3 Main Results

In the following, we present the theorems for the achievable bound, the converse bound and the order optimality. The proofs of the theorems are provided in Section 5.2, Section 5.3 and Section 5.4, respectively.

Theorem 5.1 (Achievable Bound)**.** *Consider the problem of coded distributed computing with structured support. Then, the optimal communication load $L^\star$ is upper bounded by the lower convex envelope of the following corner points*

$$(r, L_{\text{UB}}) = (r, g(r)), \quad \forall r \in [K'] \tag{5.5}$$

where $g(r)$ is defined as

$$g(r) \coloneqq \begin{cases} \dfrac{G'}{G}\dfrac{1}{r}\left(1 - \dfrac{r}{K}\right), & \text{if } r \in [\bar{r}] \\ \dfrac{G'}{G}\dfrac{1}{\bar{r}}\left(1 - \dfrac{\bar{r}}{K}\right)\dfrac{K' - r}{K' - \bar{r}}, & \text{if } r \in [\bar{r} + 1 : K'] \end{cases} \tag{5.6}$$

and $\bar{r} \coloneqq \lfloor K'/2 \rfloor$.

Theorem 5.2 (Converse Bound)**.** *Consider the problem of coded distributed computing with structured support. Then, the optimal communication load* $L^\star$ *is lower bounded by the lower convex envelope of the following corner points*

$$(r, L_{\mathrm{LB}}) = \left(r, \frac{1}{K} \frac{K' - r}{r+1} \right), \quad \forall r \in [K']. \tag{5.7}$$

Theorem 5.3 (Order Optimality)**.** *Consider the problem of coded distributed computing with structured support. Then, the achievable performance in Theorem 5.1 is order optimal within a constant multiplicative factor of* 6*.*

5.2 Proof of Theorem 5.1

We present in the following a coded scheme achieving a communication load which is strictly better than the one guaranteed by the original coded distributed computing scheme with equal storage cost r. For the sake of simplicity, we assume uniform function assignment, i.e., we assume that each server computes one distinct output function. Further, we simply let server $k \in [K]$ compute the k-th output function without loss of generality.

5.2.1 Map Phase

The map phase remains essentially the same as the one presented in [52]. Assuming that $\mathcal{W}_{\mathcal{K}_g}$ contains enough files for each $g \in [G]$, we partition each group of files in $\binom{K}{r}$ batches containing $\eta \in \mathbb{N}^+$ files. Consequently, for each $g \in [G]$ we have

$$\mathcal{W}_{\mathcal{K}_g} = \bigcup_{\mathcal{T} \subseteq [K] : |\mathcal{T}| = r} \{\mathcal{B}_{\mathcal{K}_g, \mathcal{T}}\} \tag{5.8}$$

where we have a batch $\mathcal{B}_{\mathcal{K}_g, \mathcal{T}}$, which contains η files from the g-th group of files, for each $\mathcal{T} \subseteq [K]$ with $|\mathcal{T}| = r$. The above implies that the total number of files is given by $N = G\eta\binom{K}{r}$. Then, we assign server $k \in [K]$ the set of files $\mathcal{M}_k$ given by

$$\mathcal{M}_k = \{\mathcal{B}_{\mathcal{K}_g, \mathcal{T}} : g \in [G], \mathcal{T} \subseteq [K], |\mathcal{T}| = r, k \in \mathcal{T}\}. \tag{5.9}$$

Consequently, recalling that the storage load is defined as the normalized number of files that are stored across the K servers, we can verify that the storage load is given by

$$\frac{\sum_{k \in [K]} |\mathcal{M}_k|}{N} = \frac{KG\eta\binom{K-1}{r-1}}{N} = r. \tag{5.10}$$

Given the file assignment described above, server k can compute the map functions for all the files in $\mathcal{M}_k$. However, we know from the system model that each output function does not depend on all the N input files. More precisely, we know that the files in $\mathcal{W}_{\mathcal{K}_g}$ are useful only to the K' functions in $\mathcal{K}_g$. Hence, each server can compute the IVs that are really needed. Consequently, the set of IVs computed by server $k \in [K]$ is given by

$$\mathcal{V}_k = \{v_{q,n} : g \in [G], \mathcal{T} \subseteq [K], |\mathcal{T}| = r, k \in \mathcal{T}, w_n \in \mathcal{B}_{\mathcal{K}_g,\mathcal{T}}, q \in \mathcal{K}_g\}. \quad (5.11)$$

The computation load is calculated as

$$\frac{\sum_{k\in[K]} |\mathcal{V}_k|}{NK} = \frac{KK'G\eta\binom{K-1}{r-1}}{NK} = \frac{K'}{K}r \quad (5.12)$$

and this is a factor K/K' better than the computation load of the original CDC scheme.

5.2.2 Shuffle Phase and Reduce Phase

The shuffle phase proceeds exactly as described in [52]. The savings will come from the fact that each server does not need IVs for all the files in the input library.

Consider server $k \in [K]$ and let $\mathcal{S} \subseteq [K]$ where $k \in \mathcal{S}$ and $|\mathcal{S}| = r + 1$. Then, for each $s \in (\mathcal{S} \setminus \{k\})$, server k creates the symbol $U_{\mathcal{S}\setminus\{s\}} = (v_{s,n} : g \in [G], s \in \mathcal{K}_g, w_n \in \mathcal{B}_{\mathcal{K}_g,\mathcal{S}\setminus\{s\}})$, which is then split in r even segments as

$$U_{\mathcal{S}\setminus\{s\}} = (U_{\mathcal{S}\setminus\{s\},j} : j \in (\mathcal{S} \setminus \{s\})). \quad (5.13)$$

After, the following multicast message

$$X_k = \left(\bigoplus_{s\in(\mathcal{S}\setminus\{k\})} U_{\mathcal{S}\setminus\{s\},k} : \mathcal{S} \subseteq [K], |\mathcal{S}| = r + 1, k \in \mathcal{S} \right) \quad (5.14)$$

is transmitted over the broadcast channel.

Since the shuffling scheme above follows the same coded scheme in [52], we can immediately conclude that, for what concerns the reduce phase, each server can correctly obtain the missing IVs from local data and broadcast transmissions.

5.2.3 Communication Load

We can see that the communication load is given by

$$L_{\text{new}} = \frac{\sum_{k\in[K]} |X_k|}{KNT} \quad (5.15)$$

$$= \frac{K\binom{K-1}{r}\eta T G'/r}{KNT} \tag{5.16}$$

$$= \frac{G'\binom{K-1}{r}\eta}{rG\eta\binom{K}{r}} \tag{5.17}$$

$$= \frac{G'}{G}\frac{1}{r}\left(1-\frac{r}{K}\right). \tag{5.18}$$

Interestingly, since $G' < G$, the load above is strictly better than the communication load achieved by the original CDC scheme, which is given by

$$L_{\text{old}} = \frac{1}{r}\left(1-\frac{r}{K}\right). \tag{5.19}$$

However, L_{new} is suboptimal. We can see that $L_{\text{new}} = 0$ when $r = K$, even though the storage-load point $(r, L) = (K', 0)$ is achievable. Indeed, if we let server $k \in [K]$ store only the files which the output function ϕ_k depends on, we can see that in such case the storage load is given by

$$\frac{KG'\eta\binom{K}{r}}{N} = \frac{KG'}{G} = K'. \tag{5.20}$$

Hence, we propose the following scheme. Let $\bar{r} \in [K']$. When $r \leq \bar{r}$, we use the scheme presented above. When $r > \bar{r}$, we adopt a memory-sharing approach between the storage space $\bar{r}$ and K'. More precisely, the piecewise linear curve with corner points $(r, g(r))$ is achievable, where

$$g(r) = \begin{cases} \dfrac{G'}{G}\dfrac{1}{r}\left(1-\dfrac{r}{K}\right), & \text{if } r \in [\bar{r}] \\ \dfrac{G'}{G}\dfrac{1}{\bar{r}}\left(1-\dfrac{\bar{r}}{K}\right)\dfrac{K'-r}{K'-\bar{r}}, & \text{if } r \in [\bar{r}+1 : K'] \end{cases} \tag{5.21}$$

and where we arbitrarily let $\bar{r} = \lfloor K'/2 \rfloor$. Notice that the case $K' = 1$ is trivial, because we have zero load when $r = 1$. This concludes the proof. □

5.3 Proof of Theorem 5.2

5.3.1 Preliminaries

We begin the proof by introducing some useful notation. For $q \in [K]$ and $w_n \in \{w_1, \ldots, w_N\}$, we let $V_{q,n}$ be an i.i.d. random variable and we let $v_{q,n}$ be the realization of $V_{q,n}$. Then, we define

$$D_k \coloneqq \{V_{k,n} : g \in [G], k \in \mathcal{K}_g, w_n \in \mathcal{W}_{\mathcal{K}_g}\} \tag{5.22}$$

$$C_k \coloneqq \{V_{q,n} : \mathcal{T} \subseteq [K], |\mathcal{T}| = r, k \in \mathcal{T}, g \in [G], q \in \mathcal{K}_q, w_n \in \mathcal{B}_{\mathcal{K}_g,\mathcal{T}}\} \quad (5.23)$$

$$Y_k \coloneqq (D_k, C_k) \quad (5.24)$$

for each $k \in [K]$. Recalling that we denote by X_k the multicast message transmitted by node $k \in [K]$, the equation

$$H(X_k \mid C_k) = 0 \quad (5.25)$$

holds, since X_k is a deterministic function of the intermediate values computed by node k. Moreover, for any map-shuffle-reduce scheme, each node $k \in [K]$ has to be able to correctly recover all the intermediate values D_k given the transmissions of all nodes $X_{[K]} \coloneqq (X_1, \ldots, X_K)$ and given the IVs C_k computed by the node k itself. Thus, the equation

$$H(D_k \mid X_{[K]}, C_k) = 0 \quad (5.26)$$

holds for each $k \in [K]$.

5.3.2 Lower Bound on the Communication Load

For a given file assignment denoted by $\mathcal{M} \coloneqq (\mathcal{M}_1, \ldots, \mathcal{M}_K)$, we let $L_{\mathcal{M}}$ be the corresponding communication load under this assignment $\mathcal{M}$. Then, from [65, Lemma 2] we can provide the following lower bound

$$L_{\mathcal{M}} \geq \frac{1}{KNT} \sum_{k \in [K]} H(D_{u_k} \mid C_{u_k}, Y_{u_1,\ldots,u_{k-1}}) \quad (5.27)$$

for a given permutation $\boldsymbol{u} = (u_1, \ldots, u_K) \in S_K$, where we define $Y_{u_1,\ldots,u_{k-1}} \coloneqq (Y_{u_1}, \ldots, Y_{u_{k-1}})$. The inequality above can be written as

$$L_{\mathcal{M}} \geq \frac{1}{KNT} \sum_{k \in [K]} H(D_{u_k} \mid C_{u_k}, Y_{u_1,\ldots,u_{k-1}}) \quad (5.28)$$

$$= \frac{1}{KNT} \sum_{k \in [K]} \sum_{\mathcal{T} \subseteq ([K] \setminus \{u_1,\ldots,u_k\})} \sum_{g \in [G] : u_k \in \mathcal{K}_g} T a^{\mathcal{T}}_{\mathcal{K}_g} \quad (5.29)$$

where $a^{\mathcal{T}}_{\mathcal{K}_g}$ denotes the number of files from $\mathcal{W}_{\mathcal{K}_g}$ which are mapped exactly by the nodes in $\mathcal{T}$. If we build the above inequality for each $\boldsymbol{u} \in S_K$ and we sum them up, we get

$$L_{\mathcal{M}} \geq \frac{1}{KNK!} \sum_{\boldsymbol{u} \in S_K} \sum_{k \in [K]} \sum_{\mathcal{T} \subseteq ([K] \setminus \{u_1,\ldots,u_k\})} \sum_{g \in [G] : u_k \in \mathcal{K}_g} a^{\mathcal{T}}_{\mathcal{K}_g}. \quad (5.30)$$

Following a combinatorial argument similar to the one used in Chapter 3 and recalling that any file is needed by K' output functions, we can simplify the above as

$$L_{\mathcal{M}} \geq \frac{1}{KNK!} \sum_{\boldsymbol{u} \in S_K} \sum_{k \in [K]} \sum_{\mathcal{T} \subseteq ([K] \setminus \{u_1, \ldots, u_k\})} \sum_{g \in [G] : u_k \in \mathcal{K}_g} a^{\mathcal{T}}_{\mathcal{K}_g} \tag{5.31}$$

$$= \sum_{j \in [K]} \sum_{i=\max(0, j-(K-K'))}^{\min(j,K')} \sum_{g \in [G]} \sum_{\substack{\mathcal{T} \subseteq [K] : |\mathcal{T}| = j, \\ |\mathcal{T} \cap \mathcal{K}_g| = i}} \frac{(K'-i) j! (K-j-1)! \binom{K}{j+1}}{KNK!} a^{\mathcal{T}}_{\mathcal{K}_g} \tag{5.32}$$

$$\geq \frac{1}{KN} \sum_{j \in [K']} \frac{K'-j}{j+1} \underbrace{\sum_{i=\max(0, j-(K-K'))}^{j} \sum_{g \in [G]} \sum_{\substack{\mathcal{T} \subseteq [K] : |\mathcal{T}| = j, \\ |\mathcal{T} \cap \mathcal{K}_g| = i}} a^{\mathcal{T}}_{\mathcal{K}_g}}_{:= \tilde{a}^j_{\mathcal{M}}} \tag{5.33}$$

$$= \frac{1}{K} \sum_{j \in [K']} \frac{K'-j}{j+1} \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{5.34}$$

where $\tilde{a}^j_{\mathcal{M}}$ is defined as the total number of files which are mapped by exactly j map nodes under this particular file assignment $\mathcal{M}$.

For any given file assignment $\mathcal{M}$ and for any given storage load $r \in [K]$, the fact that $|\mathcal{M}_1| + \cdots + |\mathcal{M}_K| \leq rN$ also implies that $\tilde{a}^j_{\mathcal{M}} \geq 0$ for each $j \in [K]$, as well as implies that $\sum_{j \in [K]} \tilde{a}^j_{\mathcal{M}} = N$ and that $\sum_{j \in [K]} j \tilde{a}^j_{\mathcal{M}} \leq rN$. Thus, we can further lower bound the above using Jensen's inequality and the fact that $(K'-j)/(j+1)$ is convex and decreasing in j. Hence, we can write

$$L_{\mathcal{M}} \geq \frac{1}{K} \sum_{j \in [K']} \frac{K'-j}{j+1} \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{5.35}$$

$$\geq \frac{1}{K} \frac{K'-r}{r+1} \tag{5.36}$$

$$= L_{\text{LB}} \tag{5.37}$$

where (5.36) holds due to the storage constraint $\sum_{j \in [K]} j \tilde{a}^j_{\mathcal{M}} \leq rN$. Since the bound above does not depend on the file assignment $\mathcal{M}$, it holds for any possible file assignment. However, the actual bound can be further tightened using the same techniques as in [52] for non-integer values of r, and so we can finally state that the lower bound is the lower convex envelope of the points $\{(r, L_{\text{LB}}) : r \in [K']\}$. This concludes the proof. □

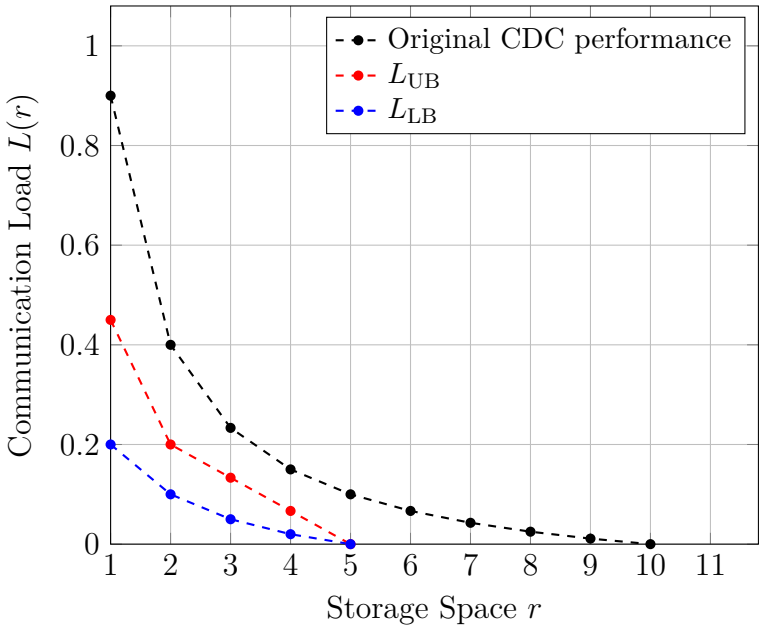

Figure 5.1: Comparison between the original CDC performance and the new scheme when there are $K = 10$ output functions and $G = 10$ groups of files, and each group of files is needed by $K' = 5$ output functions, whereas each output function depends on $G' = 5$ groups of files. The achievable scheme is in red, whereas the converse bound is in blue.

5.4 Proof of Theorem 5.3

When $r \in [\bar{r}]$, we have

$$\frac{L_{\text{UB}}}{L_{\text{LB}}} = \frac{G'}{G}\frac{1}{r}\left(1-\frac{r}{K}\right)\frac{K(r+1)}{K'-r} \tag{5.38}$$

$$\leq \frac{G'}{G}\frac{1}{r}\left(1-\frac{r}{K}\right)\frac{K(r+1)}{K'-\bar{r}} \tag{5.39}$$

$$\leq \frac{KG'}{G}\frac{r+1}{r}\frac{1}{K'-\bar{r}} \tag{5.40}$$

$$\leq \frac{2K'}{K'-\bar{r}}. \tag{5.41}$$

When K' is even, we recall that we have $\bar{r} = K'/2$. Hence, we have

$$\frac{L_{\text{UB}}}{L_{\text{LB}}} \leq \frac{2K'}{K'-\bar{r}} \tag{5.42}$$

$$= \frac{2K'}{K'-K'/2} \tag{5.43}$$

$$= 4. \tag{5.44}$$

Instead, when K' is odd, we recall that we have $\bar{r} = (K'-1)/2$. Hence, we have

$$\frac{L_{\text{UB}}}{L_{\text{LB}}} \leq \frac{2K'}{K'-\bar{r}} \tag{5.45}$$

$$= \frac{2K'}{K' - K'/2 + 1/2} \tag{5.46}$$

$$< 4. \tag{5.47}$$

Now, when $r \in [\bar{r}+1 : K']$, we have

$$\frac{L_{\text{UB}}}{L_{\text{LB}}} = \frac{G'}{G}\frac{1}{\bar{r}}\left(1 - \frac{\bar{r}}{K}\right)\frac{\cancel{K'-r}}{K'-\bar{r}}\frac{K(r+1)}{\cancel{K'-r}} \tag{5.48}$$

$$\leq \frac{KG'}{G}\frac{1}{\bar{r}}\frac{(K'+1)}{K'-\bar{r}} \tag{5.49}$$

$$= \frac{K'(K'+1)}{\bar{r}(K'-\bar{r})}. \tag{5.50}$$

When K' is even, we have

$$\frac{L_{\text{UB}}}{L_{\text{LB}}} \leq \frac{K'(K'+1)}{\bar{r}(K'-\bar{r})} \tag{5.51}$$

$$= \frac{4(K'+1)}{K'} \tag{5.52}$$

$$= 6 \tag{5.53}$$

as the smallest even K' is $K' = 2$. Instead, when K' is odd, we have

$$\frac{L_{\text{UB}}}{L_{\text{LB}}} \leq \frac{K'(K'+1)}{\bar{r}(K'-\bar{r})} \tag{5.54}$$

$$= \frac{K'(K'+1)}{(K'/2-1/2)(K'/2+1/2)} \tag{5.55}$$

$$= 4\frac{K'}{K'-1}. \tag{5.56}$$

The last inequality above is maximized when K' is minimized. Since we are considering K' odd and we exclude the trivial case $K' = 1$, the smallest odd K' is $K' = 3$. Hence, the above is upper bounded as

$$\frac{L_{\text{UB}}}{L_{\text{LB}}} \leq 4\frac{K'}{K'-1} \tag{5.57}$$

$$\leq 6. \tag{5.58}$$

In conclusion, the proposed scheme is order optimal within a factor 6 over the entire range $r \in [K']$. The proof is concluded. □

Part II

The Ramifications of Structure in Topology

Chapter 6

Combinatorial Multi-Access Caching

In this chapter, we identify the fundamental limits of multi-access coded caching (MACC) where each user is connected to multiple caches in a manner that follows a generalized combinatorial topology. First, we extend the setting and the scheme presented in [53] to a much more general topology that supports both a much denser range of users and the coexistence of users connected to different numbers of caches. For this generalized topology, we then propose a novel information-theoretic converse that establishes, together with the scheme, the exact optimal performance under uncoded placement. Subsequently, we consider different connectivity ensembles, including the very general scenario of the entire ensemble of all possible network connectivities/topologies, where any subset of caches can serve any arbitrary number of users. For these settings, we develop novel converse bounds on the optimal performance averaged over the ensemble's different connectivities. This novel analysis of topological ensembles leaves open the possibility that currently unknown topologies may yield even higher gains, a hypothesis that is part of the bigger question of which network topology yields the most caching gains.

6.1 Introduction

THE original coded caching model in [2] considered that each user has access to its own single dedicated cache. However, in several scenarios it is conceivable and perhaps more realistic that each cache serves more than one user, and that each user can connect to more than one cache. For instance, in dense cellular networks, the cache-aided access points (APs) could have overlapping coverage areas, allowing so each user to connect to more than one

AP. Even more realistically, in a wired setting where a central server wishes to communicate to multiple workers via a shared control channel, each worker could be assisted by multiple memory devices shared among the workers.

Such scenarios motivated the work in [29] that introduced the extra dimension of having users that can now have access to multiple caches. In this setting — in addition to the number of users K, the number of library files N and the cache size of M files — a new parameter α describes the number of caches that each user can access. This so-called multi-access coded caching (MACC) model introduced in [29] involved Λ users and Λ caches, and involved a topology where each user is connected to $\alpha > 1$ consecutive caches in a cyclic wrap-around fashion as in Figure 1.2, such that each cache serves exactly α users. In the same work, the authors provided a caching-and-delivery procedure which guarantees in its centralized variant[1] a worst-case load of

$$\frac{\Lambda(1-\alpha\gamma)}{\Lambda\gamma+1} = \frac{K(1-\alpha\gamma)}{K\gamma+1} \tag{6.1}$$

where we recall $\gamma = M/N$ is the fraction of the library that each cache is able to store. Such scheme takes advantage of the multi-access nature of the network and allows for an increase of the local caching gain from γ to $\alpha\gamma$, without though being able to increase the coding gain, which remains fixed at the gain $\Lambda\gamma + 1$ that only corresponds to the gain in the dedicated cache scenario where $\alpha = 1$.

6.1.1 Past Works on Multi-Access Coded Caching

Since the introduction of the aforementioned multi-access cyclic model, various works focused on the design of coding schemes that leverage the fact that each user has access to more cache space. The challenge is always to be able to achieve higher coding gains in a setting where the cache volume seen by a user must be shared among several users.

One of such works can be found in [67], which proposed a scheme that not only preserves the full local caching gain as in [29], but also achieves the topology's optimal coding gain of $\Lambda\alpha\gamma + 1$, albeit for the rather unrealistically demanding scenario[2] where $\alpha = (\Lambda - 1)/\Lambda\gamma$. For the same cyclic topology, the work in [66] designed a novel scheme for any $\alpha \geq 1$, which was — for the similarly demanding regime of $\alpha \geq \Lambda/2$ and $\Lambda\gamma \leq 2$ — proved to be at a

[1] The original work in [29] proposed the coding scheme with decentralized (stochastic) cache placement. The centralized version can be easily obtained as also mentioned in [66].

[2] Indeed, thinking about either the wired server-and-workers setting or the dense cellular network scenario previously mentioned, it is more realistic for a user to be connected to very few cache-aided devices.

factor of at most 2 from the optimal under uncoded placement. Other works include the extension of the MACC model to support privacy and secrecy constraints [68]–[70], the connection between MACC and structured index coding problems [71], the study of two-dimensional multi-access networks [72], and the application of PDA designs to the multi-access setting [73], [74].

Recently, a new MACC paradigm was presented in [75], [76], which involves previously unexplored powerful topologies that deviate from the cyclic topology originally proposed in [29]. These two works in [75], [76] drew a clever connection between coding for the MACC problem and employing a topology that is inspired by cross resolvable designs (CRDs), where these CRDs constitute a special class of designs in combinatorics. The authors provided novel placement-and-delivery schemes from CRDs achieving a coding gain equal to $(q+1)^z$, where q and z are two integer parameters such that each user is connected to $\alpha = qz$ distinct caches via a network topology that is implied by the chosen CRD. While though this gain nicely increases with α, it does not increase with the cumulative cache redundancy $\Lambda\gamma$ and so can remain relatively small as it does not capitalize on this redundancy.

A substantial breakthrough came with the very recent work in [53], which proposed a MACC model enjoying the same amount of resources α and $\Lambda\gamma$, but where now the users and the caches are connected following the well-known combinatorial topology of combination networks [54]–[57]. This was a breakthrough because it allowed for the deployment of a subsequent scheme — presented in [53] as a generalization of the original MAN scheme in [2] — that achieves an astounding coding gain $\binom{\Lambda\gamma+\alpha}{\alpha}$ far exceeding $\Lambda\gamma+1$ even for small values of α and $\Lambda\gamma$, which is the regime that really matters. A noticeable drawback of this new approach is that its performance is guaranteed only for a rather sparse range[3] of $K \gg \Lambda$, where we recall K is the number of users and Λ is the number of caches, and that it only captures the scenario where the users must all connect to an identical number of caches.

6.1.2 Main Contributions

Having as a starting point the combinatorial multi-access system model introduced in [53], we propose a model extension which allows for a denser range of possible number of users K and for the coexistence of users that are connected to different numbers of caches. For this generalized combinatorial

[3]As it will become more clear later, the work in [53] requires exactly $K = \binom{\Lambda}{\alpha}$ users for a specific value of $\alpha \in \{0, \dots, \Lambda\}$. Instead, our generalized model allows for a number of users that is equal to $K = \sum_{\alpha=0}^{\Lambda} K_\alpha \binom{\Lambda}{\alpha}$ with $K_\alpha \in \{0, 1, \dots\}$ for each $\alpha \in \{0, \dots, \Lambda\}$, which consequently makes denser the range of users K with respect to the model in [53].

system model, we extend the delivery scheme presented in [53] to support the very large coding gains. We then proceed to prove this general scheme to be exactly optimal under the assumption of uncoded placement by means of an information-theoretic converse that is based on index coding arguments. As a practical consequence of identifying the exact fundamental limits of the setting, we now know that a basic and fixed MAN placement can optimally handle any generalized combinatorial network irrespective of having unknown numbers of users connected to different numbers of caches.

Subsequently, we consider a more general scenario that involves various ensembles of connectivities, including the ensemble of all possible connectivities as well as the smaller ensemble of those connectivities that simply abide by the constraint that each user is connected to the same number of α caches, without any additional structural constraint on the connectivity or on the number of users that each cache has to treat. For these settings, we develop information-theoretic converse bounds on the optimal average worst-case load, where the average is taken over the ensemble of interest, and where the optimal is over an optimized fixed placement. In particular, this converse on the average performance assumes optimal delivery for each connectivity and assumes an optimized uncoded placement that is fixed across all connectivities. These converse bounds are then used to provide meaningful insights on the strength of the generalized combinatorial model with respect to other connectivities.

6.1.3 Chapter Outline

The rest of the chapter is organized as follows. Section 6.2 presents the system model together with some clarifying examples on the setting. Then, Section 6.3 presents the main results. Subsequently, Section 6.4 presents the coding scheme for the generalized combinatorial topology, whereas Section 6.5 presents the new matching converse bound. After this, Section 6.6 and Section 6.7 provide the proof of the converse on the optimal average worst-case load under uniformly random connectivity for the different considered ensembles. The appendices Appendix B hold all additional proofs.

6.2 System Model

We consider the centralized coded caching scenario where one single server has access to a library $\{W_n : n \in [N]\}$ containing N files of B bits each. The server is connected to K users through an error-free broadcast link. In the system there are Λ caches, each of size MB bits. Each user is associated (i.e.,

has full access) to a subset of these caches. We assume that the link between the server and the users is the main bottleneck, whereas we assume that each user can access its assigned caches at zero cost. As is common, we assume that $N \geq K$.

6.2.1 Description of Connectivity

Each of the K users is connected to a subset of the Λ caches. The way these connections are set up defines the network topology or connectivity. In the general case, different users are connected to a different number $\alpha \in [0 : \Lambda]$ of caches, where this value α, depending on the user, can range from $\alpha = 0$ (corresponding to users that are not assisted by any cache) up to $\alpha = \Lambda$ (corresponding to the users that happen to be connected to all caches). What we will refer to as *connectivity* will be here defined by the number of users that each α-tuple $\mathcal{U}$ of caches is *exactly* and *uniquely* connected to, where again some of these sets $\mathcal{U}$ of caches can have size $\alpha = 1$ (sets consisting of a single cache), $\alpha = 2$ (where each set is a pair of caches), and so on. Notice that, since a connectivity is defined irrespective of any permutation of users, the definition of connectivity captures the concept of the MACC problem "type". For instance, for the model in Figure 1.2 the sets of 2 caches $\{1, 2\}$, $\{2, 3\}$, $\{3, 4\}$ and $\{1, 4\}$ serve one user each. For the same model, we consider the set of 2 caches $\{1, 3\}$, $\{2, 4\}$ to be serving no user, since there is no user connected exactly and uniquely to the caches in the set $\{1, 3\}$ (i.e., there is no user connected to only cache 1 *and* to cache 3) as well as there is no user associated exactly and uniquely to the caches in the set $\{2, 4\}$ (i.e., there is no user connected to only cache 2 *and* to cache 4). Similarly, we consider also the sets $\{1\}$, $\{2\}$, $\{3\}$ and $\{4\}$ to be serving no user, since there is no user connected exactly and uniquely to one cache only. On the other hand, for the model in Figure 6.1 we can see that there are two users connected exactly and uniquely to cache $\{1\}$, there is one user associated exactly and uniquely to caches $\{1, 2, 3\}$ and there is one user connected exactly and uniquely to caches $\{2, 3, 4\}$. Hence, for such example the connectivity is defined by the number of users connected to the set of caches $\{1\}$, $\{1, 2, 3\}$ and $\{2, 3, 4\}$.

Let $\mathcal{B}$ be the set of all possible connectivites corresponding to the most general scenario, where any set of caches can arbitrarily serve any number of users without any constraint or structure. Each connectivity $b \in \mathcal{B}$ will be defined by the vector $\boldsymbol{K}_b = (K_{\mathcal{U},b} : \mathcal{U} \subseteq [\Lambda])$, where we denote by $K_{\mathcal{U},b} \in \mathbb{N}$ the number of users associated exactly and uniquely to the caches in the set $\mathcal{U}$. Naturally, it holds that

$$K = \sum_{\alpha \in [0:\Lambda]} \sum_{\mathcal{U} \subseteq [\Lambda] : |\mathcal{U}| = \alpha} K_{\mathcal{U},b}. \tag{6.2}$$

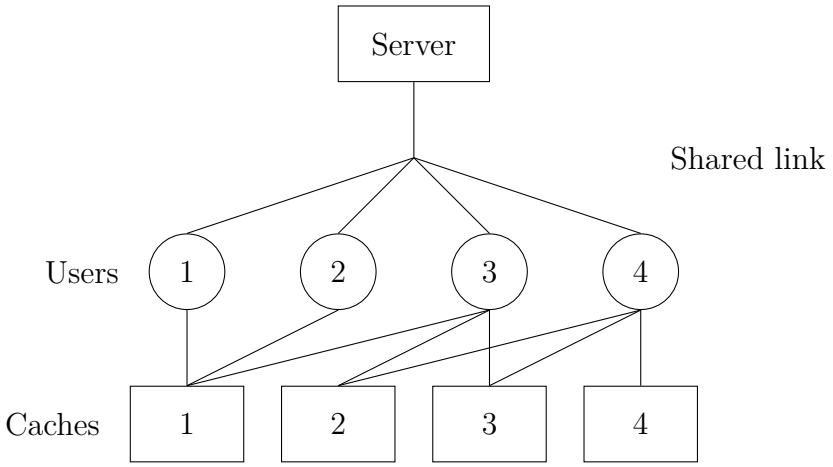

Figure 6.1: Example of connectivity for the MACC model with $\Lambda = 4$.

For a given connectivity $b \in \mathcal{B}$, and for any set $\mathcal{U} \subseteq [\Lambda]$, we denote by $\mathcal{U}_k$ the k-th user connected[4] to the $|\mathcal{U}|$ caches in $\mathcal{U}$, where naturally $k \in [K_{\mathcal{U},b}]$. If we let $\mathcal{K}_\alpha$ be the set of users which are each connected to exactly and uniquely α caches, it holds that

$$\mathcal{K} = \bigcup_{\alpha \in [0:\Lambda]} \{\mathcal{K}_\alpha\} \tag{6.3}$$

$$\mathcal{K}_\alpha = \bigcup_{\mathcal{U} \subseteq [\Lambda]:|\mathcal{U}|=\alpha} \bigcup_{k \in [K_{\mathcal{U},b}]} \{\mathcal{U}_k\} \tag{6.4}$$

where $\mathcal{K}$ denotes the entire set of K users in the system. This all holds for the most general setting corresponding to the ensemble $\mathcal{B}$, where this ensemble represents the set of all connectivities irrespective of how different these connectivities are from one another. We will revisit this general scenario later, when we will calculate the converse bound on the optimal average performance over all connectivities in $\mathcal{B}$.

A more restricted class of connectivities can be found in the ensemble $\mathcal{B}_\alpha$, which consists of all those connectivities for which each user is connected to exactly α caches for some fixed number $\alpha \in [0 : \Lambda]$. Any connectivity that

[4]Note that this reflects, as already mentioned, the fact that a connectivity is defined irrespective of any permutation of users. For example, consider $\Lambda = 2$ caches and $K = 3$ users. Then, if we have $K_{\{1\},b} = 2$ and $K_{\{2\},b} = 1$ for some connectivity $b \in \mathcal{B}$, it does not make any difference whether it is user 1 to be assigned to cache 2, and user 2 and user 3 to be assigned to cache 1; whether it is user 2 to be assigned to cache 2, and user 1 and user 3 to be assigned to cache 1; or whether it is user 3 to be assigned to cache 2, and user 1 and user 2 to be assigned to cache 1. All these scenarios correspond in fact to the same connectivity.

satisfies this constraint belongs to $\mathcal{B}_\alpha$. For example, a connectivity b belongs to $\mathcal{B}_2$ if and only if this connectivity guarantees that every user is connected to exactly $\alpha = 2$ caches. The well-known cyclic wrap-around connectivity depicted in Figure 1.2 is one of the connectivities in this class $\mathcal{B}_2$, since it guarantees that each user is connected to exactly $\alpha = 2$ caches. There exist many additional connectivities that belong to $\mathcal{B}_2$. We will revisit this class $\mathcal{B}_\alpha$ when we will calculate the average optimal performance, averaged across all its connectivities.

A broader class of connectivities is simply the $(\Lambda + 1)$-ary Cartesian product

$$\mathcal{B}^{\Lambda+1} = \prod_{\alpha=0}^{\Lambda} \mathcal{B}_\alpha = \{(b_0, \ldots, b_\Lambda) : b_\alpha \in \mathcal{B}_\alpha, \alpha \in [0:\Lambda]\} \tag{6.5}$$

which consists of all connectivities that guarantee that some K'_α users — from the $K = \sum_{\alpha\in[0:\Lambda]} K'_\alpha$ users in total — are each connected to exactly α caches for each $\alpha \in [0:\Lambda]$ and for a fixed set of integers $K'_0, K'_1, \ldots, K'_\Lambda$. For example, a connectivity b belongs to $\mathcal{B}_2 \times \mathcal{B}_3$ if and only if this connectivity guarantees that K'_2 users are connected to 2 caches, and the rest $K'_3 = K - K'_2$ users are connected to 3 caches, for any fixed pair K'_2, K'_3 such that $K'_2 + K'_3 = K$. This class is important because it captures the generalization of the aforementioned combinatorial[5] topology, for which we will calculate the optimal performance under uncoded prefetching.

To avoid heavy notation, the dependence on the connectivity b will always be suppressed and left implied. For example, while $\boldsymbol{K}_b$ is a function of the specific connectivity, this dependence will be implied when we henceforth use the notation $\boldsymbol{K}$ instead of $\boldsymbol{K}_b$. An exception to this rule will be allowed when considering the number of users $K_{\mathcal{U},b}$ associated to the caches in $\mathcal{U}$ for some $\mathcal{U} \subseteq [\Lambda]$.

In terms of file requests, we use the notation $W_{d_{\mathcal{U}_k}}$ to denote the file requested by the user identified by $\mathcal{U}_k$, which we remind the reader is simply the k-th user connected exactly and uniquely to the caches in the set $\mathcal{U}$. For the sake of simplicity, we denote by $\boldsymbol{d} = (\boldsymbol{d}_0, \ldots, \boldsymbol{d}_\Lambda)$ the demand vector containing the indices of the files requested by the users in the system, where

$$\boldsymbol{d}_\alpha \coloneqq \left(\boldsymbol{d}_{\mathcal{U},[K_{\mathcal{U},b}]} : \mathcal{U} \subseteq [\Lambda], |\mathcal{U}| = \alpha\right) \tag{6.6}$$

represents the vector containing the indices of the files requested by all the users connected to exactly α caches and where $\boldsymbol{d}_{\mathcal{U},[K_{\mathcal{U},b}]} \coloneqq (d_{\mathcal{U}_1}, \ldots, d_{\mathcal{U}_{K_{\mathcal{U},b}}})$

[5]The topology introduced in [53] follows the well-known combinatorial nature of combination networks. To avoid any confusion with the well-defined term *combination-network topology* which is prevalent in the literature of network coding, we will use the simplified term *combinatorial topology* to refer to the topology in [53].

represents the vector containing the indices of the files requested by the $K_{\mathcal{U},b}$ users connected to the caches in the set $\mathcal{U}$ for a given connectivity $b \in \mathcal{B}$. To account for the possibility that this notation is hard to follow, we offer clarifying examples later on.

6.2.2 Generalized Combinatorial Topology

Directly from the aforementioned Cartesian product class, there is a particular connectivity (topology) $b \in \mathcal{B}^{\Lambda+1}$ that is of special interest to us. This topology, which we refer to as the *generalized combinatorial topology*, guarantees that any one set of α caches is uniquely assigned to K_α users, and this holds for every $\alpha \in [0:\Lambda]$. Given the nature of the connectivity, we have that[6] $K_{\mathcal{U},b} = K_\alpha = K'_\alpha / \binom{\Lambda}{\alpha}$ for each $\mathcal{U} \subseteq [\Lambda]$ with $|\mathcal{U}| = \alpha$ and $\alpha \in [0:\Lambda]$. We also have that

$$K = \sum_{\alpha=0}^{\Lambda} K_\alpha \binom{\Lambda}{\alpha}. \tag{6.7}$$

To clarify, the term K'_α again describes the total number of users each of which is associated to exactly α caches, while the term K_α is the normalization of K'_α and it describes the total number of users uniquely served by any one set of α caches. For this generalized combinatorial topology, we collect all the K_α terms in the vector $\boldsymbol{K}_{\text{comb}} = (K_0, \ldots, K_\Lambda)$.

Example 6.1 ($\Lambda = 4, \boldsymbol{K}_{\text{comb}} = (0,0,1,0,0)$)**.** Consider the MACC problem with the original combinatorial topology in [53] and $\Lambda = 4$ caches. Since $\boldsymbol{K}_{\text{comb}} = (0,0,1,0,0)$, each set of $\alpha = 2$ caches is uniquely assigned to $K_2 = 1$ user, so there are $K = \sum_{\alpha=0}^{\Lambda} K_\alpha \binom{\Lambda}{\alpha} = \binom{4}{2} = 6$ users in total. Recalling that each user is identified by the set of the $\alpha = 2$ caches it is connected to as well as by its index $k \in [K_2]$, we write the set of users $\mathcal{K}$ as

$$\begin{aligned}
\mathcal{K} &= \bigcup_{\alpha \in [0:\Lambda]} \bigcup_{\mathcal{U} \subseteq [\Lambda]:|\mathcal{U}|=\alpha} \bigcup_{k \in [K_\alpha]} \{\mathcal{U}_k\} && (6.8)\\
&= \bigcup_{\mathcal{U} \subseteq [\Lambda]:|\mathcal{U}|=2} \bigcup_{k \in [K_2]} \{\mathcal{U}_k\} && (6.9)\\
&= \{\{1,2\}_1, \{1,3\}_1, \{1,4\}_1, \{2,3\}_1, \{2,4\}_1, \{3,4\}_1\}. && (6.10)
\end{aligned}$$

Notice that for ease of notation we will often omit braces and commas when indicating sets, so we can also write $\mathcal{K} = \{12_1, 13_1, 14_1, 23_1, 24_1, 34_1\}$.

[6] As one would expect, we assume for such combinatorial connectivity that $\binom{\Lambda}{\alpha} \mid K'_\alpha$ for each $\alpha \in [0:\Lambda]$.

These are the 6 users in the system. The demand vector is given by $\boldsymbol{d} = (\boldsymbol{d}_0, \boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \boldsymbol{d}_4) = (0, \dots, 0, \boldsymbol{d}_2, 0, \dots, 0)$, where

$$\boldsymbol{d}_2 = \left(\boldsymbol{d}_{\mathcal{U},[K_2]} : \mathcal{U} \subseteq [\Lambda], |\mathcal{U}| = 2\right) \tag{6.11}$$

$$= (d_{12_1}, d_{13_1}, d_{14_1}, d_{23_1}, d_{24_1}, d_{34_1}) \tag{6.12}$$

recalling that $\boldsymbol{d}_{\mathcal{U},[K_2]} = (d_{\mathcal{U}_1}, \dots, d_{\mathcal{U}_{K_2}})$. The setting described in this example corresponds to the model introduced in [53], as it will be pointed out also later.

Example 6.2 ($\Lambda = 4, \boldsymbol{K}_{\text{comb}} = (0, 0, 2, 2, 0)$)**.** Let us consider now the following more involved MACC problem with a generalized combinatorial topology and again $\Lambda = 4$ caches. Since $\boldsymbol{K}_{\text{comb}} = (0, 0, 2, 2, 0)$, each set of 2 caches is uniquely connected to $K_2 = 2$ users and each set of 3 caches is uniquely connected to $K_3 = 2$ users, which tells us that there are $K = \sum_{\alpha=0}^{\Lambda} K_\alpha \binom{\Lambda}{\alpha} = 2\binom{4}{2} + 2\binom{4}{3} = 20$ users in total. The set of users $\mathcal{K}$ is given by

$$\mathcal{K} = \bigcup_{\alpha \in [0:\Lambda]} \bigcup_{\mathcal{U} \subseteq [\Lambda] : |\mathcal{U}| = \alpha} \bigcup_{k \in [K_\alpha]} \{\mathcal{U}_k\} \tag{6.13}$$

$$= \bigcup_{\alpha \in [2:3]} \bigcup_{\mathcal{U} \subseteq [\Lambda] : |\mathcal{U}| = \alpha} \bigcup_{k \in [K_\alpha]} \{\mathcal{U}_k\} \tag{6.14}$$

$$= \{12_1, 12_2, 13_1, 13_2, 14_1, 14_2, 23_1, 23_2, 24_1, 24_2, 34_1, 34_2, 123_1, 123_2, 124_1, 124_2, 134_1, 134_2, 234_1, 234_2\}. \tag{6.15}$$

As a small reminder, users $12_1, 12_2$ are the first and second users connected to the pair of caches $\{1, 2\}$, while user 234_2 is the second of two users connected to the caches in the triplet $\{2, 3, 4\}$ and to no other cache. In this case, the demand vector is given by $\boldsymbol{d} = (\boldsymbol{d}_0, \boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3, \boldsymbol{d}_4) = (0, \dots, 0, \boldsymbol{d}_2, \boldsymbol{d}_3, 0, \dots, 0)$, where

$$\boldsymbol{d}_2 = \left(\boldsymbol{d}_{\mathcal{U},[K_2]} : \mathcal{U} \subseteq [\Lambda], |\mathcal{U}| = 2\right) \tag{6.16}$$

$$= (d_{12_1}, d_{12_2}, d_{13_1}, d_{13_2}, d_{14_1}, d_{14_2}, d_{23_1}, d_{23_2}, d_{24_1}, d_{24_2}, d_{34_1}, d_{34_2}) \tag{6.17}$$

$$\boldsymbol{d}_3 = \left(\boldsymbol{d}_{\mathcal{U},[K_3]} : \mathcal{U} \subseteq [\Lambda], |\mathcal{U}| = 3\right) \tag{6.18}$$

$$= (d_{123_1}, d_{123_2}, d_{124_1}, d_{124_2}, d_{134_1}, d_{134_2}, d_{234_1}, d_{234_2}) \tag{6.19}$$

recalling that $\boldsymbol{d}_{\mathcal{U},[K_2]} = (d_{\mathcal{U}_1}, \dots, d_{\mathcal{U}_{K_2}})$ and $\boldsymbol{d}_{\mathcal{U},[K_3]} = (d_{\mathcal{U}_1}, \dots, d_{\mathcal{U}_{K_3}})$.

6.2.3 Worst-Case Load and Average Worst-Case Load

As in the original coded caching scenario, the communication procedure is split into the placement phase and the delivery phase. Nevertheless, the placement phase may or may not be aware of the given topology $b \in \mathcal{B}$ that will be encountered during delivery. Both these scenarios of *topology-aware* and *topology-agnostic* cache placement will be addressed.

In the topology-aware scenario, given a unique topology $b \in \mathcal{B}$ that is known throughout placement and delivery, the worst-case communication load R_b is defined as the total number of transmitted bits, normalized by the file size B, that can guarantee the correct delivery of any K-tuple of requested files in the worst-case scenario. The optimal communication load $R_b^\star$ is consequently defined as

$$R_b^\star(M) \coloneqq \inf\{R_b : \ (M, R_b) \text{ is achievable}\} \tag{6.20}$$

where the tuple (M, R_b) is said to be *achievable* if there exists a caching-and-delivery procedure for which, for any possible demand, a load R_b can be guaranteed. This metric captures the optimal performance, optimized over all topology-aware placement-and-delivery schemes. For the particular case of the generalized combinatorial topology, this worst-case communication load will be denoted by $R_{\text{comb}}^\star$.

On the other hand, to capture the fact that topologies may vary in time often much faster than the rate with which caches can be updated, we will also consider the scenario where a fixed placement must be designed to handle an ensemble of possible topologies[7], where the ensemble of focus is known during placement. In this topology-agnostic scenario, we will employ an average metric that captures the average performance of coded caching over the ensemble of topologies. In particular, we will consider the average worst-case communication load $R_{\text{avg}} = \mathbb{E}_b[R_b]$, which is defined as the expected number of transmitted bits (averaged over a specified ensemble of connectivities, and normalized by the file size B) that can guarantee the correct delivery of all requested files irrespective of the request. When the averaging is done over the entire connectivity ensemble $\mathcal{B}$ of connectivities, the corresponding optimal average worst-case load $R_{\text{avg},\mathcal{B}}^\star$ is defined and denoted as

$$R_{\text{avg},\mathcal{B}}^\star(M) \coloneqq \inf\{R_{\text{avg},\mathcal{B}} : \ (M, R_{\text{avg},\mathcal{B}}) \text{ is achievable}\} \tag{6.21}$$

where the tuple $(M, R_{\text{avg},\mathcal{B}})$ is said to be *achievable* if there exists a joint

[7]This means that the connectivity is not known a priori and the cache placement cannot be modified whenever a new connectivity is presented.

placement-and-delivery method with an optimized fixed placement phase[8] for which an average load $R_{\text{avg},\mathcal{B}}$ can be guaranteed, where the averaging is over the connectivity ensemble of focus. Similarly, when the averaging is done over the smaller symmetric ensemble $\mathcal{B}_\alpha$, the optimal average performance will be denoted by $R^\star_{\text{avg},\mathcal{B}_\alpha}$.

6.3 Main Results

We present in this section the main results. Firstly, we identify the fundamental limits of the system model described in Section 6.2.2 corresponding to the unique generalized combinatorial topology. This will identify the optimal performance of the generalized combinatorial connectivity, optimized over all coded caching schemes under the assumption of uncoded prefetching. We will then proceed to study the performance over ensembles of connectivities. Taking into account the scenario where there are $K = K'_\alpha$ users and each of them is connected to a set of exactly α caches, we develop a converse bound on the optimal average worst-case load assuming the connectivities in $\mathcal{B}_\alpha$ to be equiprobable for a fixed $\alpha \in [\Lambda]$. Finally, such bound is further extended to consider the most general ensemble $\mathcal{B}$ of all possible connectivities.

6.3.1 Multi-Access Coded Caching With Generalized Combinatorial Topology

Our first result is obtained by extending the achievable scheme proposed in [53] and by developing a matching converse bound based on the well-known acyclic subgraph index coding method. The result is formally stated in the following theorem.

Theorem 6.1. *Consider the multi-access coded caching problem with Λ caches and the generalized combinatorial topology described in Section 6.2.2. Under the assumption of uncoded cache placement, the optimal worst-case*

[8]Here, the optimal performance is over the class of all schemes that employ a cache placement which can be chosen and optimized, but which must remain fixed for all connectivities in the ensemble. The scheme is free to employ delivery methods that are fully aware of the current topology and can adapt to it. For every choice of fixed placement, and then for every connectivity, there is a minimum amount of bits to be sent. We are interested in minimizing the average of these amounts of bits to be transmitted, averaged over all the connectivities in the ensemble of focus.

communication load $R^{\star}_{\text{comb}}$ *is a piecewise linear curve with corner points*

$$(M, R^{\star}_{\text{comb}}) = \left(t\frac{N}{\Lambda}, \sum_{\alpha=0}^{\Lambda-t} K_{\alpha} \frac{\binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{t}} \right), \quad \forall t \in [0:\Lambda]. \tag{6.22}$$

Proof. The coded caching scheme is described in Section 6.4, whereas the information-theoretic converse is presented in Section 6.5. □

Directly from the converse in Theorem 6.1, and from the application as in Section 6.4 of the scheme in [53], we now have the following corollary. To place the corollary in context, we note that typically (see for example [66]) cache placements are specifically calibrated to reflect the cache-connectivity capability of each user.

Corollary 6.1.1. *The basic* Λ*-cache MAN placement allows for the optimal performance for any instance of the generalized combinatorial topology. This means that, as long as the connectivity is from the generalized combinatorial topology, then the single MAN placement yields a uniformly optimal performance irrespective of the cache-connectivity capability of each user, i.e., irrespective of how many users are connected to how many caches.*

Remark 6.1. A further comment relates the memory point at which the load becomes 0. As one can observe from the expression in Theorem 6.1, such point varies depending on the problem instance. For example, if we consider the generalized combinatorial topology where each user is uniquely associated to a distinct cache (this corresponds to the standard MAN setting), the load becomes 0 only when each user has enough memory to store entirely the library, i.e., when $t = \Lambda$. On the other hand, for the generalized combinatorial topology where each user is connected to all caches, the load becomes 0 when $t = 1$, i.e., when the entire library is stored at least once across the caches. In general, if there is no user connected to any cache, we have $t \in [0 : \Lambda - \alpha_{\min} + 1]$, where $\alpha_{\min} \coloneqq \min\{\alpha : K_{\alpha} > 0\}$, i.e., where $\alpha_{\min}$ is the minimum value of α such that $K_{\alpha} > 0$. Clearly, if there is even just one user connected to no cache, there will be no memory point such that the load will be equal to 0.

6.3.2 Analysis of Topology Ensembles

Our second contribution is the development of a converse bound on the optimal average worst-case load for a fixed value[9] of $\alpha \in [\Lambda]$, corresponding to the ensemble $\mathcal{B}_{\alpha}$, where each connectivity in $\mathcal{B}_{\alpha}$ is assumed to be equiprobable. The result is stated in the following theorem.

[9]Clearly, the scenario $\alpha = 0$ is trivial. Indeed, if all users K are connected to no cache, there is only one connectivity which is optimally served with load equal to K.

Theorem 6.2. *Consider the ensemble $\mathcal{B}_\alpha$ of multi-access coded caching problems with Λ caches and $K = K'_\alpha$ users each connected to exactly $\alpha \in [\Lambda]$ caches. Under the assumption of fixed uncoded cache placement and equiprobable connectivities, the optimal average worst-case communication load $R^\star_{\text{avg},\mathcal{B}_\alpha}$ is lower bounded by $R_{\text{avg},\mathcal{B}_\alpha,\text{LB}}$ which is a piecewise linear curve with corner points*

$$(M, R_{\text{avg},\mathcal{B}_\alpha,\text{LB}}) = \left(t\frac{N}{\Lambda}, \frac{K'_\alpha \binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{t}} + A_t \right), \quad \forall t \in [0 : \Lambda - \alpha + 1] \tag{6.23}$$

where we define

$$A_t \coloneqq \frac{K'_\alpha}{|\mathcal{B}_\alpha|} \binom{\Lambda - t}{\alpha} \left(1 - \frac{1}{\binom{t+\alpha}{\alpha}} \right). \tag{6.24}$$

Proof. The proof is reported in Section 6.6. □

In the following we offer an interesting comparison between the results in Theorem 6.1 and in Theorem 6.2, after setting $K = K'_\alpha = K_\alpha \binom{\Lambda}{\alpha}$ to be the same in both cases.

Corollary 6.2.1. *For a fixed unique α and a fixed $K = K'_\alpha$ such that $\binom{\Lambda}{\alpha} \mid K'_\alpha$, then*

$$R_{\text{avg},\mathcal{B}_\alpha,\text{LB}} > R^\star_{\text{comb}}, \quad \forall t \in [\Lambda - \alpha]. \tag{6.25}$$

Proof. The proof follows immediately after observing that $A_t > 0$ for each $t \in [\Lambda - \alpha]$. □

The above result simply says that, for non-trivial values of t, the optimal average worst-case communication load in Theorem 6.2 is strictly larger than the optimal worst-case communication load of the combinatorial connectivity in Theorem 6.1. At this point, we can make two observations. The first is that a closer inspection reveals that the converse in Theorem 6.2 holds also when the fixed uncoded placement is restricted to be the MAN cache placement. The second observation is that — as a consequence of the assumption of equiprobable connectivities — the converse in Theorem 6.2 can be interpreted as a lower bound on the optimal arithmetic mean worst-case load of different[10] "types" of MACC problems. With these two observations in place, we can conclude that the combinatorial topology is, roughly speaking, among the

[10]As we mentioned in the introduction, since each connectivity is defined irrespective of any permutation of users, the definition of connectivity reflects somehow the notion of the "type" of a multi-access coded caching problem.

better connectivities in $\mathcal{B}_\alpha$ under the MAN cache placement assumption, in the sense that it does better than the average.

We now transition to the most general scenario where any possible connectivity is allowed, namely, where any set of caches can arbitrarily serve any number of users without any constraint or structure. This corresponds to the ensemble $\mathcal{B}$. Our third contribution is the development of a converse bound on the optimal average worst-case load assuming the connectivities in the set $\mathcal{B}$ to be equiprobable.

Theorem 6.3. *Consider the ensemble $\mathcal{B}$ of multi-access coded caching problems with Λ caches and K users, where any set of caches can arbitrarily serve any number of users without any constraint or structure. Under the assumption of fixed uncoded cache placement and equiprobable connectivities, the optimal average worst-case communication load $R^\star_{\text{avg},\mathcal{B}}$ is lower bounded by $R_{\text{avg},\mathcal{B},\text{LB}}$ which is a piecewise linear curve with corner points*

$$(M, R_{\text{avg},\mathcal{B},\text{LB}}) = \left(t\frac{N}{\Lambda}, \sum_{\alpha=0}^{\Lambda} \frac{K\binom{\Lambda}{t+\alpha}}{2^\Lambda \binom{\Lambda}{t}} + A_{t,\alpha} \right), \quad \forall t \in [0:\Lambda] \tag{6.26}$$

where we define

$$A_{t,\alpha} \coloneqq \frac{K}{|\mathcal{B}|}\binom{\Lambda - t}{\alpha}\left(1 - \frac{1}{\binom{t+\alpha}{\alpha}}\right). \tag{6.27}$$

Proof. The proof is reported in Section 6.7. □

As before, we offer the following interesting comparison between the results in Theorem 6.1 and in Theorem 6.3, after setting $K_\alpha = K/2^\Lambda$ for each $\alpha \in [0:\Lambda]$. The following corollary serves as an indication that, under fixed MAN uncoded placement, the generalized combinatorial topology is among the better connectivities in $\mathcal{B}$.

Corollary 6.3.1. *For a fixed K such that $2^\Lambda \mid K$ and for $K_\alpha = K/2^\Lambda$ for each $\alpha \in [0:\Lambda]$, then*

$$R_{\text{avg},\mathcal{B},\text{LB}} > R^\star_{\text{comb}}, \quad \forall t \in [\Lambda]. \tag{6.28}$$

Proof. The proof follows again immediately after observing that, for a given $\alpha \in [0:\Lambda-1]$, then $A_{\alpha,t} > 0$ for each $t \in [\Lambda - \alpha]$. □

Remark 6.2. The results in Corollary 6.2.1 and Corollary 6.3.1 shed light on the purpose and utility of the developed converse bounds in Theorem 6.2 and in Theorem 6.3, whose relevance appears when they are put in perspective

with the optimal performance of the MACC problem with (generalized) combinatorial topology in Theorem 6.1. As suggested above, the meaningful conclusion here is that the (generalized) combinatorial topology is a good (better than average) connectivity under the standard MAN cache placement. Nevertheless, whether it is uniformly — in the non-trivial memory regime — the best topology or not still remains a wide-open question.

6.4 Achievability Proof of Theorem 6.1

We devote this section to presenting the general placement-and-delivery scheme, which will allow us to prove that the load performance in Theorem 6.1 is indeed achievable. Recall that this is for the case of the generalized combinatorial topology presented in Section 6.2.2. As a quick reminder, under this topology, out of the total of $K = \sum_{\alpha=0}^{\Lambda} K'_\alpha$ users, there exist K'_α users each of which is associated to exactly α caches. Similarly, the normalized K_α simply describes the total number of users uniquely served by any one set of α caches. As it will be clear in a short while, we point out that the general delivery scheme here proposed is a properly calibrated orthogonal concatenation of the transmitted sequences in [53] for different values of $\alpha \in [0 : \Lambda]$.

6.4.1 Description of the General Scheme

The communication process is split into the placement phase and the delivery phase. Both phases are designed with full knowledge[11] of the topology, i.e., with knowledge of the fact that during delivery the users are connected to the caches according to the unique generalized combinatorial topology in Section 6.2.2.

Placement phase This procedure is performed by the central server without knowing the future requests of each user. Let $M = tN/\Lambda$ be the volume of data, in units of file, that each of the Λ caches can store, where $t \in [0 : \Lambda]$ is an integer value. Each file is split into $\binom{\Lambda}{t}$ equal-sized non-overlapping subfiles as follows

$$W_n = \{W_{n,\mathcal{T}} : \mathcal{T} \subseteq [\Lambda], |\mathcal{T}| = t\}, \quad \forall n \in [N] \tag{6.29}$$

[11]Indeed, there is generally some flexibility and the placement can be designed for a specific topology, when such topology is known. In fact, for the generalized combinatorial topology the basic Λ-cache MAN placement is enough to achieve the optimal performance, as mentioned in Corollary 6.1.1.

and the memory of the λ-th cache is filled as

$$Z_\lambda = \{W_{n,\mathcal{T}} : n \in [N], \mathcal{T} \subseteq [\Lambda], |\mathcal{T}| = t, \lambda \in \mathcal{T}\} \tag{6.30}$$

for each $\lambda \in [\Lambda]$. A quick calculation allows us to verify that each cache $\lambda \in [\Lambda]$ stores a total of

$$|Z_\lambda| = N\binom{\Lambda-1}{t-1}\frac{B}{\binom{\Lambda}{t}} = t\frac{N}{\Lambda}B = MB \tag{6.31}$$

bits, which guarantees that the memory size constraint is satisfied.

Delivery phase The delivery phase takes place once the file requests of the users are revealed. Let $\boldsymbol{d} = (\boldsymbol{d}_0, \ldots, \boldsymbol{d}_\Lambda)$ be the demand vector containing the indices of the files demanded by the K users in $\mathcal{K}$. Then, the server transmits $X_{\boldsymbol{d}} = (X_0, \ldots, X_{\Lambda-t})$, where $X_\alpha = (X_{\alpha,1}, \ldots, X_{\alpha,K_\alpha})$ and where

$$X_{\alpha,k} = \left(\bigoplus_{\mathcal{U}\subseteq\mathcal{S}:|\mathcal{U}|=\alpha} W_{d_{\mathcal{U}_k},\mathcal{S}\setminus\mathcal{U}} : \mathcal{S} \subseteq [\Lambda], |\mathcal{S}| = t+\alpha \right) \tag{6.32}$$

for each $k \in [K_\alpha]$ and for each $\alpha \in [0:\Lambda]$. Simply, $X_{\alpha,k}$ corresponds to the broadcast transmission which successfully delivers all the missing information to all the k-th users connected to any α caches. Notice that users connected to more than $\Lambda - t$ caches have access to the entire library, so no transmission is needed for them.

Remark 6.3. We point out that for any one fixed value of $\alpha \in [0:\Lambda]$ and $k \in [K_\alpha]$, the sequence of multicast messages in (6.32) matches the scheme in [53]. However, whereas the work in [53] considers $K_\alpha = 1$ and only one single value of $\alpha \in [0:\Lambda]$ at a time (i.e., considers that every user is connected to exactly α caches), the scheme here considers any $K_\alpha \in \mathbb{N}$ and most importantly considers all possible values of $\alpha \in [0:\Lambda]$ at the same time, so capturing a rather general scenario where the set of K users involves users with unequal cache-connectivity capabilities (i.e., users that have access to an unequal number of caches). The most interesting outcome of our generalization is the rather surprising fact that a basic TDMA-like approach of treating groups of users with different cache-connectivity capabilities is in fact optimal, as we will see soon. This is indeed surprising, because users with different cache-connectivity capabilities still maintain an abundance of common side information, which could have conceivably been exploited using joint encoding across the groups. The optimality of our scheme reveals that there is no need to encode across users with different cache-connectivity

capabilities, and that a TDMA-like approach is optimal, even though there is abundant additional opportunities to encode across different groups of users that are now treated separately.

Remark 6.4. An additional interesting consideration regards the adopted cache placement. To the best of our knowledge, the cache placement in the multi-access scenario usually changes substantially as α changes, e.g., this is what happens for the cyclic wrap-around topology (see [66]). However, as described in Corollary 6.1.1, for the combinatorial topology in [53] the same MAN cache placement holds for any value of $\alpha \in [0 : \Lambda]$. And such MAN placement not only works for any distinct value of α, but even allows to handle *optimally* all the α-instances when all such instances appear simultaneously at the same time. In other words, the MAN cache placement — which is independent of the number of caches each user is connected to, and is a function of the number of caches Λ and of the cache redundancy $\Lambda\gamma$ only — can be considered a single unified cache placement approach for which the least possible worst-case communication load is achieved under uncoded cache placement, even when each of the α-settings in [53] is taken into account simultaneously. At the same time, the MAN cache placement together with the TDMA-like delivery scheme also allows to maintain the astounding coding gain of each α-instance, since indeed each α-instance still enjoys[12] the coding gain of $\binom{\Lambda\gamma+\alpha}{\alpha}$.

Even though the proof of correctness of the delivery in (6.32), for a specific α, was reported in [53], we briefly describe for completeness how decoding is achieved in (6.32). Consider a user $\mathcal{U}'_k$ for some $\mathcal{U}' \subseteq [\Lambda]$ with $|\mathcal{U}'| = \alpha$, for some $k \in [K_\alpha]$ and for some $\alpha \in [0 : \Lambda]$. Consider a specific $\mathcal{S} \subseteq [\Lambda]$ for which $|\mathcal{S}| = t + \alpha$ and $\mathcal{U}' \subseteq \mathcal{S}$. For such set $\mathcal{S}$, the coded transmission

$$\bigoplus_{\mathcal{U}\subseteq\mathcal{S}:|\mathcal{U}|=\alpha} W_{d_{\mathcal{U}_k},\mathcal{S}\setminus\mathcal{U}} \tag{6.33}$$

is sent. Since $\mathcal{U}' \subseteq \mathcal{S}$, we can rewrite the coded transmission as

$$\bigoplus_{\mathcal{U}\subseteq\mathcal{S}:|\mathcal{U}|=\alpha} W_{d_{\mathcal{U}_k},\mathcal{S}\setminus\mathcal{U}} = W_{d_{\mathcal{U}'_k},\mathcal{S}\setminus\mathcal{U}'} \oplus \underbrace{\bigoplus_{\substack{\mathcal{U}\subseteq\mathcal{S}:|\mathcal{U}|=\alpha,\\ \mathcal{U}\neq\mathcal{U}'}} W_{d_{\mathcal{U}_k},\mathcal{S}\setminus\mathcal{U}}}_{\text{interference}}. \tag{6.34}$$

Notice that user $\mathcal{U}'_k$ can correctly decode the subfile $W_{d_{\mathcal{U}'_k},\mathcal{S}\setminus\mathcal{U}'}$. Indeed, this user has access to all subfiles in the interference term, considering that $(\mathcal{S}\setminus\mathcal{U})\cap\mathcal{U}' \neq \emptyset$ since $\mathcal{U} \neq \mathcal{U}'$. User $\mathcal{U}'_k$ can consequently decode a distinct

[12]This comes from the observation that each message in $X_{\alpha,k}$ is useful to $\binom{\Lambda\gamma+\alpha}{\alpha}$ users.

subfile for each $\mathcal{S} \subseteq [\Lambda]$ with $|\mathcal{S}| = t + \alpha$ and $\mathcal{U}' \subseteq \mathcal{S}$. Since there is a total of $\binom{\Lambda-\alpha}{t}$ such sets $\mathcal{S}$ and user $\mathcal{U}'_k$ misses a total of $\binom{\Lambda-\alpha}{t}$ subfiles, we can conclude that user $\mathcal{U}'_k$ correctly decodes all missing subfiles from the coded transmission in (6.32). Clearly, the same holds also for any other user, so showing the decodability of the scheme.

6.4.2 Performance Calculation

To evaluate the performance of the scheme proposed in Section 6.4.1, it is enough to calculate $|X_{\boldsymbol{d}}|/B$. Since it can be easily checked that

$$\frac{|X_{\alpha,k}|}{B} = \frac{\binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{t}} \tag{6.35}$$

the achievable load performance is equal to

$$\begin{aligned} R_{\text{comb,UB}} &= \frac{|X_{\boldsymbol{d}}|}{B} && (6.36)\\ &= \sum_{\alpha=0}^{\Lambda-t} \frac{|X_\alpha|}{B} && (6.37)\\ &= \sum_{\alpha=0}^{\Lambda-t} \sum_{k=1}^{K_\alpha} \frac{|X_{\alpha,k}|}{B} && (6.38)\\ &= \sum_{\alpha=0}^{\Lambda-t} K_\alpha \frac{\binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{t}} && (6.39) \end{aligned}$$

and so it holds that $R^\star_{\text{comb}} \leq R_{\text{comb,UB}}$, where this upper bound is a piecewise linear curve with corner points

$$(M, R_{\text{comb,UB}}) = \left(t\frac{N}{\Lambda}, \sum_{\alpha=0}^{\Lambda-t} K_\alpha \frac{\binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{t}} \right), \quad \forall t \in [0:\Lambda] \tag{6.40}$$

and where memory sharing is used between any two consecutive integer values of t. In Section 6.5 this performance will be shown to be optimal under the assumption of uncoded placement.

Remark 6.5. The coding scheme and its performance fully incorporate the following known scenarios.

1. Case $\boldsymbol{K}_{\text{comb}} = (K_0, 0, \ldots, 0)$. This scenario corresponds to the case where there are K_0 users and none of them is connected to any of

the Λ caches. In this case, uncoded delivery is optimal and the load performance is equal to K_0, independently of the memory value, as expected.

2. Case $\boldsymbol{K}_{\text{comb}} = (0, 1, 0, \ldots, 0)$. This scenario corresponds to the well-known dedicated-caches case where there is only one user connected to any one of the Λ caches. This implies that there are $K = \Lambda$ users in total and this case corresponds to the standard MAN setting, for which the MAN scheme was shown in [3] (see also [4]) to be optimal under uncoded cache placement.

3. Case $\boldsymbol{K}_{\text{comb}} = (0, K_1, 0, \ldots, 0)$ with $K_1 > 1$. This scenario corresponds to the shared-caches setting with uniform user-to-cache association profile, where the corresponding achievable load was also shown to be optimal (see [39]).

4. Case $\boldsymbol{K}_{\text{comb}} = (0, \ldots, 0, K_\alpha, 0, \ldots, 0)$ with $K_\alpha = 1$ for some $\alpha \in [2 : \Lambda]$. As already mentioned, this scenario corresponds to the setting considered in [53]. No optimality result was stated. □

6.5 Converse Proof of Theorem 6.1

The converse relies on the well-known acyclic subgraph index coding bound, which has been extensively used in Chapter 2 and various other settings (see for example [3], [39] to name a few) in order to derive lower bounds on the optimal worst-case load in caching under uncoded prefetching.

Proceeding along the lines of the converse derivation in Chapter 2, the first step in our converse proof consists of dividing, in the most generic manner, each file into a maximum of 2^Λ disjoint subfiles as

$$W_n = \{W_{n,\mathcal{T}} : \mathcal{T} \subseteq [\Lambda]\}, \quad \forall n \in [N] \tag{6.41}$$

where we identify with $W_{n,\mathcal{T}}$ the subfile which is exclusively stored by the caches in $\mathcal{T}$. Such placement is designated as uncoded because the bits of the library files are simply copied within the caches, according to Definition 1.1.

Constructing the Index Coding Bound

Assuming that each user requests a distinct[13] file, we consider the index coding problem with $K' = K = \sum_{\alpha=0}^{\Lambda} K_\alpha \binom{\Lambda}{\alpha}$ users and $N' = \sum_{\alpha=0}^{\Lambda} K_\alpha \binom{\Lambda}{\alpha} 2^{\Lambda-\alpha}$

[13]Notice that the set of worst-case demands may not include the set of demand vectors $\boldsymbol{d}$ with all distinct entries. However, this is not a problem, since our goal is to derive a

independent messages, where each such message represents a subfile requested by some user (who naturally does not have access to it via a cache). Recalling that $W_{d_{\mathcal{U}_k}}$ denotes the file requested by the user identified by $\mathcal{U}_k$, the desired message set and the side information set are respectively given, in their most generic form, by

$$\mathcal{M}_{\mathcal{U}_k} = \{W_{d_{\mathcal{U}_k},\mathcal{T}} : \mathcal{T} \subseteq ([\Lambda] \setminus \mathcal{U})\} \tag{6.42}$$

$$\mathcal{A}_{\mathcal{U}_k} = \{W_{n,\mathcal{T}} : n \in [N], \mathcal{T} \subseteq [\Lambda], \mathcal{T} \cap \mathcal{U} \neq \emptyset\} \tag{6.43}$$

for each user $\mathcal{U}_k$ with $k \in [K_{|\mathcal{U}|}]$ and $\mathcal{U} \subseteq [\Lambda]$. Here, the side information graph consists of a directed graph where each vertex is a subfile, and where there is an edge from the subfile $W_{d_{\mathcal{P}_{k_1}}}$ to the subfile $W_{d_{\mathcal{Q}_{k_2}}}$ if and only if $W_{d_{\mathcal{P}_{k_1}}} \in \mathcal{A}_{\mathcal{Q}_{k_2}}$ with $\mathcal{P} \subseteq [\Lambda]$, $\mathcal{Q} \subseteq [\Lambda]$, $k_1 \in [K_{|\mathcal{P}|}]$, $k_2 \in [K_{|\mathcal{Q}|}]$ and $\mathcal{P}_{k_1} \neq \mathcal{Q}_{k_2}$. Since our aim is to apply Lemma 2.1, we need to consider acyclic sets of vertices $\mathcal{J}$ in the side information graph. Toward this, we take advantage of the following lemma, which holds for any connectivity $b \in \mathcal{B}$.

Lemma 6.1. *Let $\boldsymbol{d} = (\boldsymbol{d}_0, \ldots, \boldsymbol{d}_\Lambda)$ be a demand vector and let $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ be a permutation of the Λ caches. The following set of vertices*

$$\begin{aligned} &\bigcup_{k \in [K_{\emptyset,b}]} \bigcup_{\mathcal{T} \subseteq [\Lambda]} \left\{W_{d_{\emptyset_k},\mathcal{T}}\right\} \cup \bigcup_{\alpha \in [\Lambda]} \bigcup_{i \in [\alpha:\Lambda]} \bigcup_{\substack{\mathcal{U}^i \subseteq \{c_1,\ldots,c_i\}: \\ |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i}} \\ &\bigcup_{k \in [K_{\mathcal{U}^i,b}]} \bigcup_{\mathcal{T}_i \subseteq ([\Lambda] \setminus \{c_1,\ldots,c_i\})} \left\{W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right\} \end{aligned} \tag{6.44}$$

is acyclic for any connectivity $b \in \mathcal{B}$.

Proof. The proof is reported in Appendix B.1. □

To illustrate the main idea in Lemma 6.1, we provide in the following a simple example where we provide explicitly the acyclic set of vertices for two different permutations of caches.

Example 6.3. Consider the setting where there are $\Lambda = 4$ caches and $\boldsymbol{K}_{\text{comb}} = (0, 0, 1, 1, 0)$, which implies $K = 10$ users and which further implies that each set of 2 caches is connected to $K_2 = 1$ user as well as each set of 3 caches is connected to $K_3 = 1$ user. Consider the demand vector $\boldsymbol{d} = (0, \ldots, 0, \boldsymbol{d}_2, \boldsymbol{d}_3, 0, \ldots, 0)$, where it is $\boldsymbol{d}_2 = (d_{12_1}, d_{13_1}, d_{14_1}, d_{23_1}, d_{24_1}, d_{34_1})$

converse bound on the worst-case load. Indeed, our choice of treating distinct demands yields a converse bound, which — while it does not need to be, a priori, the tightest bound — is a valid bound. In our case, the bound proves to be tight.

and $\boldsymbol{d}_3 = (d_{123_1}, d_{124_1}, d_{134_1}, d_{234_1})$. Further, assume the cache permutation $\boldsymbol{c} = (1, 4, 3, 2)$. The acyclic set of vertices from Lemma 6.1 is given by the union of the set

$$\begin{aligned}\Big\{&W_{d_{14_1},\emptyset}, W_{d_{14_1},2}, W_{d_{14_1},3}, W_{d_{14_1},23},\\ &W_{d_{13_1},\emptyset}, W_{d_{13_1},2}, W_{d_{34_1},\emptyset}, W_{d_{34_1},2},\\ &W_{d_{12_1},\emptyset}, W_{d_{24_1},\emptyset}, W_{d_{23_1},\emptyset}\Big\}\end{aligned} \tag{6.45}$$

with the set

$$\left\{W_{d_{134_1},\emptyset}, W_{d_{134_1},2}, W_{d_{124_1},\emptyset}, W_{d_{123_1},\emptyset}, W_{d_{234_1},\emptyset}\right\}. \tag{6.46}$$

Consider now the cache permutation $\boldsymbol{c} = (2, 3, 1, 4)$. In this case, the acyclic set of vertices is given by the union of the set

$$\begin{aligned}\Big\{&W_{d_{23_1},\emptyset}, W_{d_{23_1},1}, W_{d_{23_1},4}, W_{d_{23_1},14},\\ &W_{d_{12_1},\emptyset}, W_{d_{12_1},4}, W_{d_{13_1},\emptyset}, W_{d_{13_1},4},\\ &W_{d_{24_1},\emptyset}, W_{d_{34_1},\emptyset}, W_{d_{14_1},\emptyset}\Big\}\end{aligned} \tag{6.47}$$

with the set

$$\left\{W_{d_{123_1},\emptyset}, W_{d_{123_1},4}, W_{d_{234_1},\emptyset}, W_{d_{124_1},\emptyset}, W_{d_{134_1},\emptyset}\right\}. \tag{6.48}$$

Consider a demand vector $\boldsymbol{d} = (\boldsymbol{d}_0, \ldots, \boldsymbol{d}_\Lambda)$ and a permutation $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ of the set $[\Lambda]$. If we specialize the acyclic set in Lemma 6.1 to the generalized combinatorial topology, then applying Lemma 2.1 yields the following lower bound

$$\begin{aligned}BR^{\star}_{\text{comb}} \geq &\sum_{k\in[K_0]} \sum_{\mathcal{T}\subseteq[\Lambda]} \left|W_{d_{\emptyset_k},\mathcal{T}}\right| + \sum_{\alpha\in[\Lambda]} \sum_{i\in[\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha,\\ c_i\in\mathcal{U}^i}}\\ &\sum_{k\in[K_\alpha]} \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left|W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right|.\end{aligned} \tag{6.49}$$

Constructing the Optimization Problem

Now our goal is to create several bounds as the one in (6.49) considering any vector $\boldsymbol{d} \in \mathcal{D}$ and any vector $\boldsymbol{c} \in \mathcal{C}$, where we denote by $\mathcal{D}$ and $\mathcal{C}$ the

set of possible demand vectors with distinct entries and the set of possible permutation vectors of the set $[\Lambda]$, respectively. Our aim is then to average all these bounds to obtain in the end a useful lower bound on the optimal worst-case load. Considering that $|\mathcal{D}| = \binom{N}{K}K!$ and $|\mathcal{C}| = \Lambda!$, we aim to simplify the expression given by

$$\begin{aligned}\binom{N}{K}K!\Lambda!BR^{\star}_{\text{comb}} \geq \sum_{\boldsymbol{d}\in\mathcal{D}}\sum_{\boldsymbol{c}\in\mathcal{C}}\Bigg(&\sum_{k\in[K_0]}\sum_{\mathcal{T}\subseteq[\Lambda]}\left|W_{d_{\emptyset_k},\mathcal{T}}\right| \\ &+\sum_{\alpha\in[\Lambda]}\sum_{i\in[\alpha:\Lambda]}\sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha,\\ c_i\in\mathcal{U}^i}} \\ &\sum_{k\in[K_\alpha]}\sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})}\left|W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right|\Bigg).\end{aligned} \tag{6.50}$$

The next step consists of simplifying the expression in (6.50). Toward simplifying, we count how many times each subfile $W_{n,\mathcal{T}}$ — for any given $n \in [N]$, $\mathcal{T} \subseteq [\Lambda]$ and $|\mathcal{T}| = t'$ for some $t' \in [0:\Lambda]$ — appears in (6.50).

Assume that the file W_n is demanded by user $\emptyset_k$ for some $k \in [K_0]$. Out of the entire set $\mathcal{D}$ of all possible distinct demands, we find a total of $\binom{N}{K}K!/N$ distinct demands for which a file is requested by the same user. Hence, since there are $\binom{N}{K}K!/N$ distinct demands for which the file W_n is requested by user $\emptyset_k$, the subfile $W_{n,\mathcal{T}}$ is counted a total of $\Lambda!\binom{N}{K}K!/N$ times within the set of demands for which such file is requested by user $\emptyset_k$. The $\Lambda!$ term comes from the fact that the set

$$\bigcup_{k\in[K_0]}\bigcup_{\mathcal{T}\subseteq[\Lambda]}\{W_{d_{\emptyset_k},\mathcal{T}}\} \tag{6.51}$$

does not depend on the permutation vector $\boldsymbol{c}$, so the subfile $W_{n,\mathcal{T}}$ appears in such set independently of the vector $\boldsymbol{c}$. Since there are $\Lambda!$ such vectors, the subfile $W_{n,\mathcal{T}}$ is counted $\Lambda!$ times any time it is requested by the user $\emptyset_k$. The same reasoning follows for each $k \in [K_0]$, so we can conclude that the subfile $W_{n,\mathcal{T}}$ is counted

$$K_0\Lambda!\frac{\binom{N}{K}K!}{N} \tag{6.52}$$

times when we span all cases of distinct demand vectors for which the file W_n is requested by the set of users connected to 0 caches.

Assume now that the file W_n is demanded by user $\mathcal{U}_k$ for some $k \in [K_1]$ and for some $\mathcal{U} \subseteq ([\Lambda]\setminus\mathcal{T})$ with $|\mathcal{U}| = 1$. Recall also that such file is requested by user $\mathcal{U}_k$ a total of $\binom{N}{K}K!/N$ times. Then, within the set of demands for

which such user requests the file W_n, the subfile $W_{n,\mathcal{T}}$ is counted only when the elements in the set $\mathcal{U}$ appear in the vector $\boldsymbol{c}$ before[14] the elements in the set $\mathcal{T}$. Since there is a total of $t'!(\Lambda-1-t')!\binom{\Lambda}{t'+1}$ such vectors $\boldsymbol{c}$ in the set $\mathcal{C}$, the subfile $W_{n,\mathcal{T}}$ is counted a total of $t'!(\Lambda-1-t')!\binom{\Lambda}{t'+1}\binom{N}{K}K!/N$ times within the set of demands for which user $\mathcal{U}_k$ requests the file W_n. The same reasoning follows for each $k \in [K_1]$ and for each $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = 1$, so the subfile $W_{n,\mathcal{T}}$ is counted a total of

$$K_1(\Lambda-t')t'!(\Lambda-1-t')!\binom{\Lambda}{t'+1}\frac{\binom{N}{K}K!}{N} \tag{6.53}$$

times across all the demands for which the file W_n is requested by the set of users connected to 1 cache.

Let us consider now that the file W_n is demanded by user $\mathcal{U}_k$ for some $k \in [K_\alpha]$ and for some $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$. Recall also that such file is requested by user $\mathcal{U}_k$ a total of $\binom{N}{K}K!/N$ times. Then, within the set of demands for which such user requests the file W_n, the subfile $W_{n,\mathcal{T}}$ is counted only when the elements in the set $\mathcal{U}$ appear in the vector $\boldsymbol{c}$ before the elements in the set $\mathcal{T}$. Since there is a total of $\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}$ such vectors $\boldsymbol{c}$, the subfile $W_{n,\mathcal{T}}$ is counted a total of $\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}\binom{N}{K}K!/N$ times within the set of demands for which user $\mathcal{U}_k$ requests the file W_n. The same reasoning follows for each $k \in [K_\alpha]$ and for each $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$, so the subfile $W_{n,\mathcal{T}}$ is counted a total of

$$K_\alpha\binom{\Lambda-t'}{\alpha}\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}\frac{\binom{N}{K}K!}{N} \tag{6.54}$$

times within the set of demands for which the file W_n is requested by the set of users connected to α caches.

Consequently, if we consider all distinct demands in the set $\mathcal{D}$, the subfile $W_{n,\mathcal{T}}$ is counted a total of

$$\sum_{\alpha=0}^{\Lambda-t'} K_\alpha\binom{\Lambda-t'}{\alpha}\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}\frac{\binom{N}{K}K!}{N} \tag{6.55}$$

times, which gives us the number of times this same subfile appears in (6.50). The same reasoning follows for any $n \in [N]$ and for any $\mathcal{T} \subseteq [\Lambda]$ with $|\mathcal{T}| = t'$.

[14]Indeed, the subfile $W_{n,\mathcal{T}}$ appears in the acyclic graph chosen as in Lemma 6.1 for all those permutations $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ for which $\mathcal{U} \subseteq \{c_1, \ldots, c_i\}$ and $\mathcal{T} \subseteq \{c_{i+1}, \ldots, c_\Lambda\}$, i.e., this happens whenever the elements in $\mathcal{T}$ are after the elements in $\mathcal{U}$ in the permutation vector $\boldsymbol{c}$.

Thus, the expression in (6.50) can be rewritten as

$$R^{\star}_{\text{comb}} \geq \frac{1}{\binom{N}{K} K!\Lambda!} \sum_{t'=0}^{\Lambda} \sum_{\alpha=0}^{\Lambda-t'} \left(K_\alpha \binom{\Lambda - t'}{\alpha} \alpha! t'! (\Lambda - \alpha - t')! \right.$$
$$\left. \times \binom{\Lambda}{t' + \alpha} \binom{N}{K} K! x_{t'} \right) \tag{6.56}$$
$$= \sum_{t'=0}^{\Lambda} \sum_{\alpha=0}^{\Lambda-t'} K_\alpha \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} x_{t'} \tag{6.57}$$
$$= \sum_{t'=0}^{\Lambda} f(t') x_{t'} \tag{6.58}$$

where we define

$$f(t') \coloneqq \sum_{\alpha=0}^{\Lambda-t'} K_\alpha \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \tag{6.59}$$
$$0 \leq x_{t'} \coloneqq \sum_{n \in [N]} \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = t'} \frac{|W_{n,\mathcal{T}}|}{NB}. \tag{6.60}$$

At this point, we seek to lower bound the minimum worst-case load $R^{\star}_{\text{comb}}$ by lower bounding the solution to the following optimization problem

$$\min_{\boldsymbol{x}} \quad \sum_{t'=0}^{\Lambda} f(t') x_{t'} \tag{6.61a}$$
$$\text{subject to} \quad \sum_{t'=0}^{\Lambda} x_{t'} = 1 \tag{6.61b}$$
$$\sum_{t'=0}^{\Lambda} t' x_{t'} \leq \frac{\Lambda M}{N} \tag{6.61c}$$

where (6.61b) and (6.61c) correspond to the file size constraint and the cumulative cache-size constraint, respectively.

Lower Bounding the Solution to the Optimization Problem

Since the auxiliary variable $x_{t'}$ can be considered as a probability mass function, the optimization problem in (6.61) can be seen as the minimization of $\mathbb{E}[f(t')]$. Moreover, the following holds.

Lemma 6.2. *The function $f(t')$ is convex and decreasing in t'.*

Proof. The proof is reported in Appendix B.2. □

Taking advantage of Lemma 6.2, we can write $\mathbb{E}[f(t')] \geq f(\mathbb{E}[t'])$ using Jensen's inequality. Then, since $f(t')$ is decreasing with increasing $t' \in [0:\Lambda]$, we can further write $f(\mathbb{E}[t']) \geq f(\Lambda M/N)$ taking advantage of the fact that $\mathbb{E}[t']$ is upper bounded as in (6.61c). Consequently, $\mathbb{E}[f(t')] \geq f(\Lambda M/N)$, where $t = \Lambda M/N$. Now, when t is an integer, the bound is simply the function $f(t)$ evaluated at $t \in [0:\Lambda]$. For non-integer values of t, we can follow again the reasoning in [4], [39], [52] as in Section 2.4.1 to take the lower convex envelope of the sequence of points $\{(t, f(t)) : t \in [0:\Lambda]\}$. To conclude, the converse bound the optimal worst-case load $R^{\star}_{\text{comb}}$ is lower bounded by $R_{\text{comb,LB}}$ which is a piecewise linear curve with corner points

$$(M, R_{\text{comb,LB}}) = \left(t\frac{N}{\Lambda}, \sum_{\alpha=0}^{\Lambda-t} K_{\alpha} \frac{\binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{t}} \right), \quad \forall t \in [0:\Lambda]. \tag{6.62}$$

Since $R_{\text{comb,LB}} = R_{\text{comb,UB}}$, we can state that the optimal worst-case load $R^{\star}_{\text{comb}}$ under uncoded placement is a piecewise linear curve with the same corner points as in (6.62). This concludes the proof. □

6.6 Proof of Theorem 6.2

The proof follows along the lines of the index coding approach presented in Section 6.5. However, the main difference here is that we will have to develop a bound for an ensemble of connectivities, whereas the entire procedure in Section 6.5 assumed a specific connectivity. Hence, we remind that we are considering now the scenario where there are K users and each of them is connected to exactly α caches for a fixed value of $\alpha \in [\Lambda]$, where the connectivity is not constrained by further restrictions and where the connectivity is not known in advance (i.e., it is not known during placement). We remind the reader that we are interested in the connectivity ensemble $\mathcal{B}_{\alpha}$ as a whole, and that we want to develop a converse on the optimal average worst-case load $R^{\star}_{\text{avg},\mathcal{B}_{\alpha}}$ under uncoded and fixed cache placement. Furthermore, we recall that we assume each of these connectivities to be equiprobable and that for any $b \in \mathcal{B}_{\alpha}$ the number of users takes the form

$$K = K'_{\alpha} = \sum_{\mathcal{U} \subseteq [\Lambda] : |\mathcal{U}| = \alpha} K_{\mathcal{U},b}. \tag{6.63}$$

6.6.1 Constructing the Index Coding Bound

The first step of the proof consists of dividing generically — and independently of the connectivity $b \in \mathcal{B}_\alpha$ — each file into a maximum of 2^Λ non-overlapping subfiles as in (6.41), recalling that $W_{n,\mathcal{T}}$ represents the subfile which is cached uniquely and exactly by the caches in $\mathcal{T}$. Then, similarly to how we proceeded in Section 6.5, we always assume the demand vector $\boldsymbol{d} = (0, \ldots, 0, \boldsymbol{d}_\alpha, 0, \ldots, 0)$ to have distinct entries. Given a connectivity $b \in \mathcal{B}_\alpha$, we consider the index coding problem with $K' = K$ users and $N' = K2^{\Lambda-\alpha}$ independent messages. Notice that, since each user is connected to exactly α caches, the number of desired subfiles is exactly equal to $2^{\Lambda-\alpha}$, so N' is the total number of subfiles requested by the K users. Recalling that $W_{d_{\mathcal{U}_k}}$ denotes the file requested by the user identified by $\mathcal{U}_k$, the desired message set and the side information set are respectively given by

$$\mathcal{M}_{\mathcal{U}_k} = \{W_{d_{\mathcal{U}_k},\mathcal{T}} : \mathcal{T} \subseteq ([\Lambda] \setminus \mathcal{U})\} \tag{6.64}$$

$$\mathcal{A}_{\mathcal{U}_k} = \{W_{n,\mathcal{T}} : n \in [N], \mathcal{T} \subseteq [\Lambda], \mathcal{T} \cap \mathcal{U} \neq \emptyset\} \tag{6.65}$$

for each user $\mathcal{U}_k$ with $\mathcal{U} \subseteq [\Lambda]$, $|\mathcal{U}| = \alpha$ and $k \in [K_{\mathcal{U},b}]$. The side information graph consists again of a directed graph where each vertex is a subfile, and where there is a connection from the subfile $W_{d_{\mathcal{P}_{k_1}}}$ to the subfile $W_{d_{\mathcal{Q}_{k_2}}}$ if and only if $W_{d_{\mathcal{P}_{k_1}}} \in \mathcal{A}_{\mathcal{Q}_{k_2}}$ with $\mathcal{P} \subseteq [\Lambda]$, $\mathcal{Q} \subseteq [\Lambda]$, $|\mathcal{P}| = |\mathcal{Q}| = \alpha$, $k_1 \in [K_{\mathcal{P},b}]$, $k_2 \in [K_{\mathcal{Q},b}]$ and $\mathcal{P}_{k_1} \neq \mathcal{Q}_{k_2}$. Since our aim is to apply Lemma 2.1, we need to consider acyclic sets of vertices $\mathcal{J}$ in the side information graph. In the spirit of Lemma 6.1, we know that now the set

$$\bigcup_{i\in[\alpha:\Lambda]} \bigcup_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha,\\ c_i\in\mathcal{U}^i}} \bigcup_{k\in[K_{\mathcal{U}^i,b}]} \bigcup_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left\{W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right\} \tag{6.66}$$

is acyclic for any demand vector $\boldsymbol{d}$ and for any permutation of the Λ caches represented by the vector $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$. Applying Lemma 2.1 yields the following lower bound

$$BR_b \geq \sum_{i\in[\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha,\\ c_i\in\mathcal{U}^i}} \sum_{k\in[K_{\mathcal{U}^i,b}]} \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left|W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right| \tag{6.67}$$

where the term R_b represents the worst-case load given the connectivity $b \in \mathcal{B}_\alpha$.

6.6.2 Counting the Connectivities

As it will be of use later, we proceed to count how many possible connectivities exist in $\mathcal{B}_\alpha$, recalling that such ensemble includes connectivities for which each

user connects to exactly α caches. Toward this, if we let $y_{\mathcal{U}}$ be a non-negative integer value that represents the total number of users connected to the caches in $\mathcal{U}$, the number of connectivities in $\mathcal{B}_\alpha$ is equal to the number of non-negative integer solutions of the equation

$$\sum_{\mathcal{U}\subseteq[\Lambda]:|\mathcal{U}|=\alpha} y_{\mathcal{U}} = K. \tag{6.68}$$

This is simply equal to the number of ways we can express the integer K as the sum of a sequence of $\binom{\Lambda}{\alpha}$ non-negative integers. Hence, such number corresponds to the number of $\binom{\Lambda}{\alpha}$-weak compositions [77] of the integer K and is given by

$$|\mathcal{B}_\alpha| = \binom{K + \binom{\Lambda}{\alpha} - 1}{K}. \tag{6.69}$$

This highlights the fact that a connectivity $b \in \mathcal{B}_\alpha$ is nothing but a specific way to distribute K users among $\binom{\Lambda}{\alpha}$ possible *states*, where a *state* represents a set of α caches which a user can be connected to.

6.6.3 Constructing the Optimization Problem

Our goal is to develop an information-theoretic converse on the optimal average worst-case load $R^\star_{\text{avg},\mathcal{B}_\alpha}$ over the ensemble of connectivities $\mathcal{B}_\alpha$. Hence, assuming equiprobable connectivities implies that

$$R^\star_{\text{avg},\mathcal{B}_\alpha} = \frac{1}{|\mathcal{B}_\alpha|} \sum_{b\in\mathcal{B}_\alpha} R_b. \tag{6.70}$$

To proceed, we split the ensemble of interest as $\mathcal{B}_\alpha = \mathcal{B}_{\alpha,\text{a}} \cup \mathcal{B}_{\alpha,\text{b}}$, where $\mathcal{B}_{\alpha,\text{a}}$ is the set of connectivities for which all the K users are connected to the same α caches and where $\mathcal{B}_{\alpha,\text{b}} = \mathcal{B}_\alpha \setminus \mathcal{B}_{\alpha,\text{a}}$. For each connectivity $b \in \mathcal{B}_{\alpha,\text{a}}$, we will create several bounds as the one in (6.67). To do so, we consider any vector $\boldsymbol{d} \in \mathcal{D}$ and for each such vector we employ $\Lambda!$ times the permutation vector $\boldsymbol{c}_b$ whose first α positions describe the indices of the caches to which the K users are connected[15] according to the connectivity $b \in \mathcal{B}_{\alpha,\text{a}}$. We assume that the first α elements in $\boldsymbol{c}_b$ are put in ascending order and then, similarly, the remaining $\Lambda - \alpha$ elements of the vector $\boldsymbol{c}_b$ are also placed in

[15]For example, let $\Lambda = 7$ and $\alpha = 2$. If we consider the connectivity $b_1 \in \mathcal{B}_{\alpha,\text{a}}$ for which all K users are connected to the caches $\{3, 5\}$, then we consider the permutation vector $\boldsymbol{c}_{b_1} = (3, 5, 1, 2, 4, 6, 7)$. Similarly, if we consider the connectivity $b_2 \in \mathcal{B}_{\alpha,\text{a}}$ for which all K users are connected to the caches $\{4, 7\}$, then we consider the permutation vector $\boldsymbol{c}_{b_2} = (4, 7, 1, 2, 3, 5, 6)$. For each connectivity $b \in \mathcal{B}_{\alpha,\text{a}}$, the vector $\boldsymbol{c}_b$ is employed $\Lambda!$ times for each demand vector $\boldsymbol{d} \in \mathcal{D}$.

ascending order. Instead, for each connectivity $b \in \mathcal{B}_{\alpha,\mathrm{b}}$, we will create several bounds as the one in (6.67) by considering any demand vector $\boldsymbol{d} \in \mathcal{D}$ and any permutation vector $\boldsymbol{c} \in \mathcal{C}$. Our aim is then to average all these bounds to eventually obtain a useful lower bound on the optimal average worst-case load. Considering that we have $|\mathcal{D}| = \binom{N}{K} K!$ and $|\mathcal{C}| = \Lambda!$, we aim to simplify the expression given by

$$\begin{aligned}\binom{N}{K} K!\Lambda! B \sum_{b\in\mathcal{B}_\alpha} R_b \geq & \sum_{b\in\mathcal{B}_{\alpha,\mathrm{a}}} \sum_{\boldsymbol{d}\in\mathcal{D}} \Lambda! \sum_{k\in[K]} \\ & \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_{b,1},\ldots,c_{b,\alpha}\})} \left|W_{d_{\{c_{b,1},\ldots,c_{b,\alpha}\}_k},\mathcal{T}_i}\right| \\ & + \sum_{b\in\mathcal{B}_{\alpha,\mathrm{b}}} \sum_{\boldsymbol{d}\in\mathcal{D}} \sum_{\boldsymbol{c}\in\mathcal{C}} \sum_{i\in[\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:\\ |\mathcal{U}^i|=\alpha, c_i\in\mathcal{U}^i}} \\ & \sum_{k\in[K_{\mathcal{U}^i,b}]} \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left|W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right|\end{aligned} \tag{6.71}$$

which can be rewritten by means of (6.70) as

$$\begin{aligned}R^\star_{\mathrm{avg},\mathcal{B}_\alpha} \geq & \frac{1}{\binom{N}{K} K!\Lambda! \left|\mathcal{B}_\alpha\right| B} \Bigg(\sum_{b\in\mathcal{B}_{\alpha,\mathrm{a}}} \sum_{\boldsymbol{d}\in\mathcal{D}} \Lambda! \sum_{k\in[K]} \\ & \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_{b,1},\ldots,c_{b,\alpha}\})} \left|W_{d_{\{c_{b,1},\ldots,c_{b,\alpha}\}_k},\mathcal{T}_i}\right| \\ & + \sum_{b\in\mathcal{B}_{\alpha,\mathrm{b}}} \sum_{\boldsymbol{d}\in\mathcal{D}} \sum_{\boldsymbol{c}\in\mathcal{C}} \sum_{i\in[\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:\\ |\mathcal{U}^i|=\alpha, c_i\in\mathcal{U}^i}} \\ & \sum_{k\in[K_{\mathcal{U}^i,b}]} \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left|W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right| \Bigg).\end{aligned} \tag{6.72}$$

Our next step is to simplify the expression in (6.72) and part of doing so involves counting how many times each subfile $W_{n,\mathcal{T}}$ — for any $n \in [N]$, $\mathcal{T} \subseteq [\Lambda]$ and $|\mathcal{T}| = t'$ for some $t' \in [0:\Lambda]$ — appears in (6.72).

Consider the connectivities in $\mathcal{B}_{\alpha,\mathrm{a}}$. Assume that the file W_n is demanded by the user $\mathcal{U}_k$ for some $k \in [K]$ and for some $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$. Since this corresponds to the connectivity $b \in \mathcal{B}_{\alpha,\mathrm{a}}$ for which all the users are associated to the caches in $\mathcal{U}$, we employ $\Lambda!$ consecutive times the permutation vector $\boldsymbol{c}_b$ having in the first α positions the elements in $\mathcal{U}$. Consequently, since here the elements in $\mathcal{T}$ are by construction after the elements in $\mathcal{U}$ in the vector $\boldsymbol{c}_b$, we can deduce that the subfile $W_{n,\mathcal{T}}$ is counted $\Lambda!$ times — namely,

once for each of the $\Lambda!$ times that the vector $\boldsymbol{c}_b$ is employed — whenever the file W_n is requested by user $\mathcal{U}_k$. Recalling that any file is requested by user $\mathcal{U}_k$ a total of $\binom{N}{K}K!/N$ times, the subfile $W_{n,\mathcal{T}}$ is counted $\Lambda!\binom{N}{K}K!/N$ within the set of demands for which user $\mathcal{U}_k$ requests the file W_n. The same reasoning follows for each $k \in [K]$ and for each $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$, so the subfile $W_{n,\mathcal{T}}$ is counted

$$K\binom{\Lambda - t}{\alpha}\Lambda!\frac{\binom{N}{K}K!}{N} \tag{6.73}$$

times when we focus on connectivities in $\mathcal{B}_{\alpha,\mathrm{a}}$.

Consider now a specific connectivity $b \in \mathcal{B}_{\alpha,\mathrm{b}}$. Assume that the file W_n is demanded by the user $\mathcal{U}_k$ for some $k \in [K_{\mathcal{U},b}]$ and for some $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$, and recall once more that such file is requested by user $\mathcal{U}_k$ a total of $\binom{N}{K}K!/N$ times. Within the set of demand vectors for which user $\mathcal{U}_k$ requests the file W_n, the subfile $W_{n,\mathcal{T}}$ is counted only when the elements in the set $\mathcal{U}$ appear in the vector $\boldsymbol{c}$ before the elements in the set $\mathcal{T}$. Since there is a total of $\alpha!t'!(\Lambda - \alpha - t')!\binom{\Lambda}{t'+\alpha}$ such vectors $\boldsymbol{c}$, the subfile $W_{n,\mathcal{T}}$ is counted a total of $\alpha!t'!(\Lambda - \alpha - t')!\binom{\Lambda}{t'+\alpha}\binom{N}{K}K!/N$ times within the set of distinct demands for which user $\mathcal{U}_k$ requests the file W_n. The same reasoning follows for each $k \in [K_{\mathcal{U},b}]$, so the subfile $W_{n,\mathcal{T}}$ is counted a total of $K_{\mathcal{U},b}\alpha!t'!(\Lambda - \alpha - t')!\binom{\Lambda}{t'+\alpha}\binom{N}{K}K!/N$ times within the set of distinct demands for which the file W_n is requested by the users connected to the caches in $\mathcal{U}$ for a given connectivity $b \in \mathcal{B}_{\alpha,\mathrm{b}}$. Now, considering that $\mathcal{B}_{\alpha,\mathrm{b}} = \mathcal{B} \setminus \mathcal{B}_{\alpha,\mathrm{a}}$, it holds that $K_{\mathcal{U},b} \in [0 : K-1]$. Moreover, we can easily count how many connectivities $b \in \mathcal{B}_{\alpha,\mathrm{b}}$ exist for which $K_{\mathcal{U},b} = K - i$ where $i \in [K]$. Indeed, this number of connectivities is equal to the number of non-negative integer solutions of the equation

$$\sum_{\mathcal{U}' \subseteq [\Lambda] : |\mathcal{U}'| = \alpha, \mathcal{U}' \neq \mathcal{U}} y_{\mathcal{U}'} = i \tag{6.74}$$

which is equal to the number of $\left(\binom{\Lambda}{\alpha} - 1\right)$-weak compositions of the integer i. Such number is given by

$$\binom{i + \binom{\Lambda}{\alpha} - 2}{i}. \tag{6.75}$$

At this point, when we consider all connectivities in $\mathcal{B}_{\alpha,\mathrm{b}}$, there are $\binom{\binom{\Lambda}{\alpha}-1}{1}$ connectivities with $y_{\mathcal{U}} = K-1$, there are $\binom{\binom{\Lambda}{\alpha}}{2}$ connectivities with $y_{\mathcal{U}} = K-2$, there are $\binom{\binom{\Lambda}{\alpha}+1}{3}$ connectivities with $y_{\mathcal{U}} = K-3$, and so on. In the end, if

we go over all possible connectivities in $\mathcal{B}_{\alpha,\mathrm{b}}$, the subfile $W_{n,\mathcal{T}}$ is counted a total of

$$\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}\frac{\binom{N}{K}K!}{N}\sum_{i=1}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i} \tag{6.76}$$

times within the set of demands for which the file W_n is requested by users connected to the caches in $\mathcal{U}$. The same reasoning applies for each $\mathcal{U}\subseteq([\Lambda]\backslash\mathcal{T})$ with $|\mathcal{U}|=\alpha$, so the subfile $W_{n,\mathcal{T}}$ is counted a total of

$$\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}\binom{\Lambda-t'}{\alpha}\frac{\binom{N}{K}K!}{N}\sum_{i=1}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i} \tag{6.77}$$

times when we focus on connectivities in $\mathcal{B}_{\alpha,\mathrm{b}}$.

Now, if we focus on all connectivities in $\mathcal{B}_\alpha=\mathcal{B}_{\alpha,\mathrm{a}}\cup\mathcal{B}_{\alpha,\mathrm{b}}$, we can see that subfile $W_{n,\mathcal{T}}$ appears in (6.72) a total of

$$K\binom{\Lambda-t'}{\alpha}\Lambda!\frac{\binom{N}{K}K!}{N}+\Lambda!\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\frac{\binom{N}{K}K!}{N}\sum_{i=1}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i} \tag{6.78}$$

times. The same reasoning follows for any $n\in[N]$ and for any $\mathcal{T}\subseteq[\Lambda]$ with $|\mathcal{T}|=t'$. Thus, the expression in (6.72) can be rewritten after some simplifications as

$$\begin{aligned}R^{\star}_{\mathrm{avg},\mathcal{B}_\alpha}\geq&\sum_{t'=0}^{\Lambda-\alpha+1}\frac{1}{|\mathcal{B}_\alpha|}\Bigg(K\binom{\Lambda-t'}{\alpha}\\&+\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=1}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i}\Bigg)x_{t'} &(6.79)\\=&\sum_{t'=0}^{\Lambda-\alpha+1}f(t')x_{t'} &(6.80)\end{aligned}$$

where we define

$$\begin{aligned}f(t')\coloneqq&\frac{1}{|\mathcal{B}_\alpha|}\Bigg(K\binom{\Lambda-t'}{\alpha}\\&+\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=1}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i}\Bigg) &(6.81)\\0\leq x_{t'}\coloneqq&\sum_{n\in[N]}\sum_{\mathcal{T}\subseteq[\Lambda]:|\mathcal{T}|=t'}\frac{|W_{n,\mathcal{T}}|}{NB}. &(6.82)\end{aligned}$$

At this point, we seek to lower bound the minimum worst-case load $R^{\star}_{\mathrm{avg},\mathcal{B}_\alpha}$ by lower bounding the solution to the following optimization problem

$$\min_{\boldsymbol{x}} \quad \sum_{t'=0}^{\Lambda-\alpha+1} f(t')x_{t'} \tag{6.83a}$$

$$\text{subject to} \quad \sum_{t'=0}^{\Lambda} x_{t'} = 1 \tag{6.83b}$$

$$\sum_{t'=0}^{\Lambda} t' x_{t'} \leq \frac{\Lambda M}{N} \tag{6.83c}$$

where (6.83b) and (6.83c) correspond to the file size constraint and the cumulative cache-size constraint, respectively.

6.6.4 Lower Bounding the Solution to the Optimization Problem

Before proceeding to lower bound the solution to the optimization problem, we simplify the expression $f(t)$ as follows

$$\begin{aligned}
f(t') &= \frac{K}{|\mathcal{B}_\alpha|}\binom{\Lambda - t'}{\alpha} \\
&\quad + \frac{1}{|\mathcal{B}_\alpha|}\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=1}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i} && (6.84)\\
&= \frac{K}{|\mathcal{B}_\alpha|}\binom{\Lambda - t'}{\alpha} - \frac{K}{|\mathcal{B}_\alpha|}\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \\
&\quad + \frac{1}{|\mathcal{B}_\alpha|}\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=0}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i} && (6.85)\\
&= \frac{K}{|\mathcal{B}_\alpha|}\binom{\Lambda - t'}{\alpha}\left(1 - \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda-t'}{\alpha}\binom{\Lambda}{t'}}\right) \\
&\quad + \frac{1}{|\mathcal{B}_\alpha|}\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=0}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i} && (6.86)\\
&= \frac{K}{|\mathcal{B}_\alpha|}\binom{\Lambda - t'}{\alpha}\left(1 - \frac{1}{\binom{t'+\alpha}{\alpha}}\right) \\
&\quad + \frac{1}{|\mathcal{B}_\alpha|}\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=0}^{K-1}(K-i)\binom{i+\binom{\Lambda}{\alpha}-2}{i} && (6.87)
\end{aligned}$$

$$= A_{t'} + \frac{1}{|\mathcal{B}_\alpha|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \sum_{i=0}^{K-1} (K-i) \binom{i + \binom{\Lambda}{\alpha} - 2}{i} \tag{6.88}$$

where we define

$$A_{t'} \coloneqq \frac{K}{|\mathcal{B}_\alpha|} \binom{\Lambda - t'}{\alpha} \left(1 - \frac{1}{\binom{t'+\alpha}{\alpha}}\right). \tag{6.89}$$

Now we can further simplify $f(t')$ as follows

$$f(t') = A_{t'} + \frac{1}{|\mathcal{B}_\alpha|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \sum_{i=0}^{K-1} (K-i) \binom{i + \binom{\Lambda}{\alpha} - 2}{i} \tag{6.90}$$

$$= A_{t'} + \frac{1}{|\mathcal{B}_\alpha|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \left(K \sum_{i=0}^{K-1} \binom{i + \binom{\Lambda}{\alpha} - 2}{i} - \sum_{i=0}^{K-1} i \binom{i + \binom{\Lambda}{\alpha} - 2}{i} \right) \tag{6.91}$$

$$= A_{t'} + \frac{1}{|\mathcal{B}_\alpha|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \left(K \binom{K + \binom{\Lambda}{\alpha} - 2}{K-1} - \left(\binom{\Lambda}{\alpha} - 1 \right) \sum_{i=0}^{K-2} \binom{i + \binom{\Lambda}{\alpha} - 2}{i} \right) \tag{6.92}$$

$$= A_{t'} + \frac{1}{|\mathcal{B}_\alpha|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \left(\frac{K^2}{\binom{\Lambda}{\alpha} - 1} \binom{K + \binom{\Lambda}{\alpha} - 2}{K} - \left(\binom{\Lambda}{\alpha} - 1 \right) \binom{K + \binom{\Lambda}{\alpha} - 2}{K-2} \right) \tag{6.93}$$

$$= A_{t'} + \frac{1}{|\mathcal{B}_\alpha|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \left(\frac{K^2}{\binom{\Lambda}{\alpha} - 1} \binom{K + \binom{\Lambda}{\alpha} - 2}{K} - \frac{K(K-1)}{\binom{\Lambda}{\alpha}} \binom{K + \binom{\Lambda}{\alpha} - 2}{K} \right) \tag{6.94}$$

$$= A_{t'} + \frac{1}{|\mathcal{B}_\alpha|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \frac{K}{\binom{\Lambda}{\alpha}} \binom{K + \binom{\Lambda}{\alpha} - 1}{K} \tag{6.95}$$

$$= \frac{K \binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{t'}} + A_{t'} \tag{6.96}$$

where we used in (6.92) and in (6.93) the well-known hockey-stick identity in (2.62). At this point, we can rewrite $f(t')$ as

$$f(t') = \frac{K\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{t'}} + A_{t'}. \tag{6.97}$$

Now, since the auxiliary variable $x_{t'}$ can be considered once again as a probability mass function, the optimization problem in (6.83) can be seen as the minimization of $\mathbb{E}[f(t')]$. Moreover, the function $f(t')$ can be rewritten also as

$$f(t') = K\frac{|\mathcal{B}_\alpha| - \binom{\Lambda}{\alpha}}{\binom{\Lambda}{\alpha}|\mathcal{B}_\alpha|}\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} + \frac{K}{|\mathcal{B}_\alpha|}\binom{\Lambda - t'}{\alpha}. \tag{6.98}$$

Now, from Lemma 6.2 it follows that the term $\binom{\Lambda}{t'+\alpha}/\binom{\Lambda}{t'}$ is convex and decreasing in t'; then, it can be easily checked[16] that the term $\binom{\Lambda-t'}{\alpha}$ is convex and decreasing as well; and finally, it holds that $|\mathcal{B}_\alpha| \geq \binom{\Lambda}{\alpha}$. Hence, $f(t')$ is a non-negative linear combination of convex and decreasing functions and so it is convex and strictly decreasing for increasing t'. Consequently, by applying Jensen's inequality we can write $\mathbb{E}[f(t')] \geq f(\mathbb{E}[t']) \geq f(\Lambda M/N)$ after also considering that $\mathbb{E}[t']$ is upper bounded as in (6.83c). Thus, for $t = \Lambda M/N$ and since $K = K'_\alpha$, the optimal average worst-case load $R^\star_{\text{avg},\mathcal{B}_\alpha}$ is lower bounded by $R_{\text{avg},\mathcal{B}_\alpha,\text{LB}}$ which is a piecewise linear curve with corner points

$$(M, R_{\text{avg},\mathcal{B}_\alpha,\text{LB}}) = \left(t\frac{N}{\Lambda}, \frac{K'_\alpha\binom{\Lambda}{t+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{t}} + A_t\right), \quad \forall t \in [0 : \Lambda - \alpha + 1]. \tag{6.99}$$

This concludes the proof. □

6.7 Proof of Theorem 6.3

Since the proof follows the structure of the proof in Section 6.6, we will make an effort to avoid repetitions when possible. In this setting, we focus on the ensemble $\mathcal{B}$, which captures all possible connectivities. Here, each of the K users can connect to any set of caches without any specific constraint or structure. Our interest is treating the ensemble as a whole.

[16]This can be done following the same reasoning presented in Appendix B.2, after writing down the combinatorial coefficient as a finite product and using the general Leibniz rule to show that its second derivative is non-negative.

6.7.1 Constructing the Index Coding Bound

The mapping of the caching problem to the index coding problem is identical to what we presented in Section 6.5 and in Section 6.6, so we directly apply Lemma 2.1 using the acyclic set presented in Lemma 6.1, obtaining the bound

$$BR_b \geq \sum_{k\in[K_{\emptyset,b}]} \sum_{\mathcal{T}\subseteq[\Lambda]} \left|W_{d_{\emptyset_k},\mathcal{T}}\right| + \sum_{\alpha\in[\Lambda]} \sum_{i\in[\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\dots,c_i\}:\\ |\mathcal{U}^i|=\alpha, c_i\in\mathcal{U}^i}} \sum_{k\in[K_{\mathcal{U}^i,b}]} \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\dots,c_i\})} \left|W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right| \tag{6.100}$$

for a given connectivity $b \in \mathcal{B}$.

6.7.2 Counting the Connectivities

Considering the set of connectivities $\mathcal{B}$ and recalling that there are K users, where K is naturally fixed throughout the connectivites, it consequently holds that

$$\sum_{\mathcal{U}\subseteq[\Lambda]} K_{\mathcal{U},b} = K \tag{6.101}$$

for each $b \in \mathcal{B}$. Toward evaluating the cardinality of the set $\mathcal{B}$, if we let $y_{\mathcal{U}}$ be a non-negative integer value counting the total number of users connected to the caches in $\mathcal{U}$, then the number of connectitivites in $\mathcal{B}$ is equal to the number of non-negative integer solutions of the equation

$$\sum_{\mathcal{U}\subseteq[\Lambda]} y_{\mathcal{U}} = K. \tag{6.102}$$

In this case, this is simply equal to the number of 2^Λ-weak compositions [77] of the integer K, which is given by

$$|\mathcal{B}| = \binom{K + 2^\Lambda - 1}{K}. \tag{6.103}$$

Similarly to what we already observed when counting the connectivities in $\mathcal{B}_\alpha$ for the proof of Theorem 6.2 in Section 6.6, now a connectivity $b \in \mathcal{B}$ can be seen as a way to distribute K users among 2^Λ possible states, where each state is an element of the power set of $[\Lambda]$, i.e., a set of α caches — for some $\alpha \in [0 : \Lambda]$ — which a user can be connected to.

6.7.3 Constructing the Optimization Problem

Toward building a converse bound on the optimal average worst-case load $R^{\star}_{\mathrm{avg},\mathcal{B}}$ over the connectivities in $\mathcal{B}$, we recall that as before we have

$$R^{\star}_{\mathrm{avg},\mathcal{B}} = \frac{1}{|\mathcal{B}|} \sum_{b \in \mathcal{B}} R_b. \tag{6.104}$$

Let us split again the ensemble of interest as $\mathcal{B} = \mathcal{B}_{\mathrm{a}} \cup \mathcal{B}_{\mathrm{b}}$, where $\mathcal{B}_{\mathrm{a}}$ is the set of connectivities for which all the K users are connected to the same set of caches and where $\mathcal{B}_{\mathrm{b}} = \mathcal{B} \setminus \mathcal{B}_{\mathrm{a}}$.

For each connectivity $b \in \mathcal{B}_{\mathrm{a}}$, we will create several bounds as the one in (6.100). Toward this, we consider any vector $\boldsymbol{d} \in \mathcal{D}$ and for each such vector we employ $\Lambda!$ times the permutation vector $\boldsymbol{c}_b$ whose first α positions hold the indices of the caches to which the K users are connected according to the connectivity $b \in \mathcal{B}_{\mathrm{a}}$. Also in this case, we assume that these first α positions in $\boldsymbol{c}_b$ are put in ascending order and that the same holds, separately, for the remaining $\Lambda - \alpha$ elements. Notice that, differently from the proof in Theorem 6.2, here it is the case that $\alpha \in [0 : \Lambda]$, since we are considering the most general ensemble of connectivities.

On the other hand, for each connectivity $b \in \mathcal{B}_{\mathrm{b}}$ we will create several lower bounds as the one in (6.100) by considering any demand vector $\boldsymbol{d} \in \mathcal{D}$ and any permutation vector $\boldsymbol{c} \in \mathcal{C}$. Our aim is then to take the average of all such bounds to eventually obtain a useful lower bound on the optimal average worst-case load across connectivities in $\mathcal{B}$. Noticing that it holds that $\mathcal{B}_{\mathrm{a}} = \bigcup_{\alpha=0}^{\Lambda} \mathcal{B}_{\alpha,\mathrm{a}}$ and recalling that we have $|\mathcal{D}| = \binom{N}{K} K!$ and $|\mathcal{C}| = \Lambda!$, we aim to simplify the expression given by

$$\begin{aligned}
\binom{N}{K} K! \Lambda! B \sum_{b \in \mathcal{B}} R_b \geq \sum_{\alpha=0}^{\Lambda} \sum_{b \in \mathcal{B}_{\alpha,\mathrm{a}}} \sum_{\boldsymbol{d} \in \mathcal{D}} \Lambda! \sum_{k \in [K]} \\
\sum_{\mathcal{T}_i \subseteq ([\Lambda] \setminus \{c_{b,1},\ldots,c_{b,\alpha}\})} \left| W_{d_{\{c_{b,1},\ldots,c_{b,\alpha}\}_k}, \mathcal{T}_i} \right| \\
+ \sum_{b \in \mathcal{B}_{\mathrm{b}}} \sum_{\boldsymbol{d} \in \mathcal{D}} \sum_{\boldsymbol{c} \in \mathcal{C}} \Bigg(\sum_{k \in [K_{\emptyset,b}]} \sum_{\mathcal{T} \subseteq [\Lambda]} \left| W_{d_{\emptyset_k}, \mathcal{T}} \right| \\
+ \sum_{\alpha \in [\Lambda]} \sum_{i \in [\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i \subseteq \{c_1,\ldots,c_i\}: \\ |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i}} \\
\sum_{k \in [K_{\mathcal{U}^i,b}]} \sum_{\mathcal{T}_i \subseteq ([\Lambda] \setminus \{c_1,\ldots,c_i\})} \left| W_{d_{\mathcal{U}^i_k}, \mathcal{T}_i} \right| \Bigg)
\end{aligned} \tag{6.105}$$

which can be rewritten by means of (6.104) as

$$
\begin{aligned}
R^{\star}_{\text{avg},\mathcal{B}} \geq \frac{1}{\binom{N}{K} K!\Lambda!\,|\mathcal{B}|\,B} \Bigg[& \sum_{\alpha=0}^{\Lambda} \sum_{b\in\mathcal{B}_{\alpha,\text{a}}} \sum_{\boldsymbol{d}\in\mathcal{D}} \Lambda! \sum_{k\in[K]} \\
& \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_{b,1},\ldots,c_{b,\alpha}\})} \left| W_{d_{\{c_{b,1},\ldots,c_{b,\alpha}\}_k},\mathcal{T}_i} \right| \\
& + \sum_{b\in\mathcal{B}_{\text{b}}} \sum_{\boldsymbol{d}\in\mathcal{D}} \sum_{\boldsymbol{c}\in\mathcal{C}} \Bigg(\sum_{k\in[K_{\emptyset,b}]} \sum_{\mathcal{T}\subseteq[\Lambda]} \left| W_{d_{\emptyset_k},\mathcal{T}} \right| \\
& + \sum_{\alpha\in[\Lambda]} \sum_{i\in[\alpha:\Lambda]} \sum_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:\\ |\mathcal{U}^i|=\alpha, c_i\in\mathcal{U}^i}} \sum_{k\in[K_{\mathcal{U}^i,b}]} \sum_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left| W_{d_{\mathcal{U}^i_k},\mathcal{T}_i} \right| \Bigg) \Bigg].
\end{aligned}
\tag{6.106}
$$

The next step is to simplify the expression in (6.106). Toward this, we proceed to count how many times each subfile $W_{n,\mathcal{T}}$ — for any given $n \in [N]$, $\mathcal{T} \subseteq [\Lambda]$ and $|\mathcal{T}| = t'$ for some $t' \in [0:\Lambda]$ — appears in (6.106).

Consider the connectivities in $\mathcal{B}_{\text{a}}$. Assume that the file W_n is demanded by the user $\mathcal{U}_k$ for some $k \in [K]$, for some $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$ and for some $\alpha \in [0 : \Lambda - t']$. If we follow the same reasoning as in Section 6.6 and we further consider that $\alpha \in [0 : \Lambda - t']$, we can find that the subfile $W_{n,\mathcal{T}}$ is counted

$$
\sum_{\alpha=0}^{\Lambda-t'} K \binom{\Lambda - t'}{\alpha} \Lambda! \frac{\binom{N}{K} K!}{N}
\tag{6.107}
$$

times across the set of connectivities in $\mathcal{B}_{\text{a}}$.

Let us now consider a specific connectivity $b \in \mathcal{B}_{\text{b}}$. Assume that the file W_n is demanded by the user $\mathcal{U}_k$ for some $k \in [K_{\mathcal{U},b}]$, for some $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$ and for some $\alpha \in [0 : \Lambda - t']$, recalling once again that such file is requested by user $\mathcal{U}_k$ a total of $\binom{N}{K} K!/N$ times. Once again, the procedure to follow is the same described in Section 6.6. The difference here is to count how many connectivities $b \in \mathcal{B}_{\text{b}}$ exist with $K_{\mathcal{U},b} = K - i$ where $i \in [K]$. Indeed, this number of connectivities is now equal to the number of non-negative integer solutions of the equation

$$
\sum_{\mathcal{U}'\subseteq[\Lambda]:\mathcal{U}'\neq\mathcal{U}} y_{\mathcal{U}'} = i
\tag{6.108}
$$

which is equal to the number of $\left(2^{\Lambda} - 1\right)$-weak compositions of the integer i. Such number is given by

$$
\binom{i + 2^{\Lambda} - 2}{i}.
\tag{6.109}
$$

Thus, going over all connectivities in $\mathcal{B}_{\mathrm{b}}$, the subfile $W_{n,\mathcal{T}}$ is counted a total of

$$\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}\frac{\binom{N}{K}K!}{N}\sum_{i=1}^{K-1}(K-i)\binom{i+2^{\Lambda}-2}{i} \tag{6.110}$$

times within the set of distinct demands for which the file W_n is requested by users connected to the α caches in $\mathcal{U}$. The same reasoning applies for each $\mathcal{U}\subseteq([\Lambda]\setminus\mathcal{T})$ with $|\mathcal{U}|=\alpha$ and for each $\alpha\in[0:\Lambda-t']$, and so the subfile $W_{n,\mathcal{T}}$ is counted a total of

$$\sum_{\alpha=0}^{\Lambda-t'}\alpha!t'!(\Lambda-\alpha-t')!\binom{\Lambda}{t'+\alpha}\binom{\Lambda-t'}{\alpha}\frac{\binom{N}{K}K!}{N}\sum_{i=1}^{K-1}(K-i)\binom{i+2^{\Lambda}-2}{i} \tag{6.111}$$

times when we span across connectivities in $\mathcal{B}_{\mathrm{b}}$.

Now, going through all connectivities in $\mathcal{B}=\mathcal{B}_{\mathrm{a}}\cup\mathcal{B}_{\mathrm{b}}$, we can see that the subfile $W_{n,\mathcal{T}}$ appears in (6.106) a total of

$$\begin{aligned}&\sum_{\alpha=0}^{\Lambda-t'}\Bigg(K\binom{\Lambda-t'}{\alpha}\Lambda!\frac{\binom{N}{K}K!}{N}\\&+\Lambda!\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\frac{\binom{N}{K}K!}{N}\sum_{i=1}^{K-1}(K-i)\binom{i+2^{\Lambda}-2}{i}\Bigg)\end{aligned} \tag{6.112}$$

times. The same reasoning follows for any $n\in[N]$ and for any $\mathcal{T}\subseteq[\Lambda]$ with $|\mathcal{T}|=t'$. Thus, we can rewrite (6.106) after some simplifications as

$$\begin{aligned}R^{\star}_{\mathrm{avg},\mathcal{B}}&\geq\sum_{t'=0}^{\Lambda}\sum_{\alpha=0}^{\Lambda-t'}\frac{1}{|\mathcal{B}|}\Bigg(K\binom{\Lambda-t'}{\alpha}\\&\qquad+\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=1}^{K-1}(K-i)\binom{i+2^{\Lambda}-2}{i}\Bigg)x_{t'} &(6.113)\\&=\sum_{t'=0}^{\Lambda}f(t')x_{t'} &(6.114)\end{aligned}$$

where we define

$$f(t')\coloneqq\sum_{\alpha=0}^{\Lambda-t'}\frac{1}{|\mathcal{B}|}\left(K\binom{\Lambda-t'}{\alpha}+\frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}}\sum_{i=1}^{K-1}(K-i)\binom{i+2^{\Lambda}-2}{i}\right) \tag{6.115}$$

$$0\leq x_{t'}\coloneqq\sum_{n\in[N]}\sum_{\mathcal{T}\subseteq[\Lambda]:|\mathcal{T}|=t'}\frac{|W_{n,\mathcal{T}}|}{NB}. \tag{6.116}$$

At this point, formulating the optimization problem as in (6.61) and as in (6.83), we seek to lower bound the solution to the following

$$\min_{\boldsymbol{x}} \quad \sum_{t'=0}^{\Lambda} f(t')x_{t'} \tag{6.117a}$$

$$\text{subject to} \quad \sum_{t'=0}^{\Lambda} x_{t'} = 1 \tag{6.117b}$$

$$\sum_{t'=0}^{\Lambda} t' x_{t'} \leq \frac{\Lambda M}{N}. \tag{6.117c}$$

6.7.4 Lower Bounding the Solution to the Optimization Problem

Before proceeding to lower bound the solution to the optimization problem, we aim to simplify the expression $f(t')$. Toward this, we notice that the only difference between the expression in (6.81) and the expression in (6.115) is that the latter is a summation over α of $\Lambda - t' + 1$ terms, where each of these terms is the same as the expression in (6.81) except for the considered ensemble $\mathcal{B}$ and for the term 2^{Λ} in place of $\binom{\Lambda}{\alpha}$ in the binomial coefficient which appears in the summation over i. This latter difference does not change the simplification steps presented in the previous proof for what concerns each summand in (6.115), so we can directly conclude that $f(t')$ can be simplified as

$$\begin{aligned} f(t') = \sum_{\alpha=0}^{\Lambda-t'} \frac{K}{|\mathcal{B}|} \Bigg(& \binom{\Lambda - t'}{\alpha} \\ & + \frac{1}{|\mathcal{B}|} \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \sum_{i=1}^{K-1} (K-i) \binom{i + 2^{\Lambda} - 2}{i} \Bigg) \end{aligned} \tag{6.118}$$

$$= \sum_{\alpha=0}^{\Lambda-t'} \left(\frac{K\binom{\Lambda}{t'+\alpha}}{2^{\Lambda}\binom{\Lambda}{t'}} + A_{t',\alpha} \right) \tag{6.119}$$

where we define

$$A_{t',\alpha} \coloneqq \frac{K}{|\mathcal{B}|} \binom{\Lambda - t'}{\alpha} \left(1 - \frac{1}{\binom{t'+\alpha}{\alpha}} \right). \tag{6.120}$$

Now, it can be shown that each summand

$$\frac{K\binom{\Lambda}{t'+\alpha}}{2^{\Lambda}\binom{\Lambda}{t'}} + A_{t',\alpha} \tag{6.121}$$

is convex and decrasing for fixed value of α, as we did previously in Section 6.6. Hence, we can conclude also in this case that the function $f(t')$ is a non-negative linear combination of convex and decreasing functions, and so we can lower bound the solution to the optimization problem again applying Jensen's inequality and using the memory size constraint. Consequently, for $t = \Lambda M/N$, following the same steps as in the proof of Section 6.6, we can now conclude that the optimal average worst-case load $\mathcal{B}$ is lower bounded by $R_{\text{avg},\mathcal{B},\text{LB}}$ which is a piecewise linear curve with corner points

$$(M, R_{\text{avg},\mathcal{B},\text{LB}}) = \left(t\frac{N}{\Lambda}, \sum_{\alpha=0}^{\Lambda-t} \frac{K\binom{\Lambda}{t+\alpha}}{2^{\Lambda}\binom{\Lambda}{t}} + A_{t,\alpha} \right), \quad \forall t \in [0:\Lambda] \tag{6.122}$$

This concludes the proof. □

Chapter 7

Multi-Access Distributed Computing

In this chapter, we propose multi-access distributed computing (MADC) as a novel generalization of the original coded distributed computing (CDC) model, where now mappers *(nodes in charge of the map functions) and* reducers *(nodes in charge of the reduce functions) are distinct computing nodes that are connected through a multi-access network topology. Focusing on the MADC setting with combinatorial topology, which implies Λ mappers and K reducers such that there is a unique reducer connected to any α mappers, we propose a novel coded scheme and a novel information-theoretic converse, which jointly identify the optimal inter-reducer communication load, as a function of the computation load, to within a constant gap of* 1.5*. Additionally, a modified coded scheme and converse identify the optimal max-link communication load across all existing links to within a gap of* 4*. The unparalleled coding gains reported here should not be simply credited to having access to more mapped data, but rather to the powerful role of topology in effectively aligning mapping outputs. This realization raises the open question of which multi-access network topology guarantees the best possible performance in distributed computing.*

7.1 Introduction

With the development of large-scale machine learning algorithms and applications relying heavily on large volumes of data, we are now experiencing an ever-growing need to distribute large computations across multiple computing nodes. Different computing frameworks, such as MapReduce [58] and Spark [78], have been proposed to address these needs, based

on the aforementioned simple yet powerful concept of distributing large-scale algorithms — to be executed over a set of input data files — across multiple computing machines. Under the well-known MapReduce framework, the overall process is typically split in three distinct phases, starting with the *map phase*, the *shuffle phase* and then the *reduce phase*. During the map phase, each computing node is assigned a subset of the input data files, and proceeds to apply to each locally available file certain designated map functions. The outputs of such map functions, referred to as intermediate values (IVs), are then exchanged among the computing nodes during the shuffle phase, so that each computing node can retrieve any missing, required IVs it did not compute locally. Finally, during the reduce phase, each computing node computes one (or more) output functions depending on its assigned reduce functions, each of which takes as input the IVs computed for each input file.

7.1.1 Coded Distributed Computing

Several studies have shown that the aforementioned distributed map-shuffle-reduce approach comes with bottlenecks that may severely hinder the parallelization of computationally-intensive operations. While some works [79], [80] focused on the impact of straggler nodes, other works have pointed out that the total execution time of a distributed computing application is often dominated by the shuffling process. For instance, the work in [81], having explored the behavior of several algorithms on the Amazon EC2 cluster, revealed that the communication load in the shuffle phase was in fact the dominant bottleneck in computing the above tasks in a distributed manner. Similarly, the authors in [52] observed that, for the execution of a conventional TeraSort application, more than 95 % of the overall execution time was spent for inter-node communication.

Motivated by this communication bottleneck in the shuffle phase, the authors in [52] introduced coded distributed computing (CDC) as a novel framework that can yield lower communication loads during data shuffling. This gain could be attributed to a careful and joint design of the map and the shuffle phases. Approaching the distributed computing problem from an information-theoretic perspective, the authors brought to light the interesting relationship between the computation load during the mapping phase, and the communication load of the shuffling step. In particular, the work in [52] revealed that if the computation load of the mapping phase is *carefully* increased by a factor r — which means that each input file is mapped on average by r *carefully chosen* computing nodes — then the communication load can be reduced by the same factor r by employing coding techniques during

the shuffle phase[1]. Building on the coding-based results in cache networks [2], [82], the work in [52] characterized the exact information-theoretic tradeoff between this computation and communication loads under any map-shuffle-reduce scheme with uniform mapping capabilities and uniform assignment of reduce functions.

Since its original information-theoretic inception in [52], coded distributed computing has been explored with several variations. Such variations include heterogeneity aspects where, for example, each computing node may be assigned different numbers of files to be mapped and functions to be reduced. For such settings, novel schemes, based on hypercube and hypercuboid geometric structures, were developed in [65], [83], which managed not only to compensate for the heterogeneous nature of the considered scenarios, but to also exploit these asymmetries in order to require a smaller number of input files[2], compared to the initial scheme in [52]. Regarding this problem of requiring a large number of input files, it is worth also mentioning the work in [84], where the authors proposed a system model for distributed computing, where the required number of input files was lowered dramatically under an assumption of a multi-rank wireless network.

Some additional works explored the scenario where the computing servers communicate with each other through switch networks [85] or in the presence of a randomized connectivity [86], whereas some other works further investigated distributed computing over wireless channels [87], as well as explored the interesting scenario where each computing node might have limited storage and computational resources [88], [89]. A comprehensive survey on CDC is nicely presented in [90].

7.1.2 Main Contributions

In this chapter, we propose the new multi-access distributed computing (MADC) model, which can be considered as an extension of the original setting introduced in [52], and which entails *mappers* (map nodes) being connected to various *reducers* (reduce nodes), and where these mappers and

[1]This speedup factor r is often referred to as the *coding gain*, and it reflects the number of computing servers that simultaneously benefit from a single transmission.

[2]It is worth noting here the importance of designing schemes that can work with a smaller number of input files. The coded scheme in [52], albeit achieving the information-theoretic optimal, requires a number of input files that increases exponentially with the number of computing nodes. This may entail some limitations when finite-sized data sets are considered. Finding schemes with good performance and low file-number requirements is a research direction of significant importance.

reducers are now distinct entities[3]. As is common, mappers are in charge of mapping subsets of the input files, whereas the reducers are in charge of collecting the IVs in order to compute the reduce functions. We will here focus on the so-called *combinatorial topology* which will define how the mappers are connected to the reducers. This is a widely studied topology in other settings outside of distributed computing (see the previous Chapter 6), and it will — as we will discover later on — allow for stunning gains. Under such combinatorial topology, we consider Λ map nodes and $K \geq \Lambda$ reduce nodes, where each map node maps a subset of the input files, where each reduce node is connected to α map nodes, and where there is exactly one reducer for each subset of α map nodes. Each reducer can retrieve intermediate values only from the mappers it is connected to, whereas these same reducers can exchange via an error-free shared-link broadcast channel the remaining required intermediate values. A simple schematic of the model is shown in Figure 1.3 for the case $\Lambda = 4$, $\alpha = 2$ and $K = 6$.

As discussed before, the communication load in the shuffle phase can represent a significant bottleneck of distributed computing. As a consequence, our objective is to minimize the volume of data exchanged by the reducers over the common-bus link during the shuffle phase, as well as the communication load between the mappers and the reducers. We start our analysis by first neglecting the communication cost between mappers and reducers, and we propose — for the aforementioned MADC model with combinatorial topology — a novel coded scheme that allows for efficient communication over the broadcast communication channel. For such setting, we also provide an information-theoretic lower bound on the communication load, and we show this to be within a constant multiplicative gap of 1.5 from the achievable communication load guaranteed by the proposed coded scheme. We then proceed to also account for the download cost from mappers to reducers. For such setting, our goal is to minimize the maximal (normalized) number of bits across all links in the system. To this purpose, we introduce an additional mappers-to-reducers communication scheme and a novel converse bound which, together with the previous inter-reducer scheme, allow us to characterize the optimal max-link communication load within a constant multiplicative gap of 4.

As suggested above, the newly derived fundamental limits suggest outstanding performance. While for any given computation load r the original setting in [52] accepts a maximal coding gain of r, we here show that the new

[3]This choice is reasonable if we think of mappers as computing nodes that are specialized in evaluating the map functions, and of reducers as computing nodes that are specialized in evaluating the reduce functions [91], [92].

MADC model with combinatorial topology allows for a coding gain equal to $\left(\binom{r+\alpha}{r} - 1\right)$, again for the same mapping cost r. This we believe is the first time that topology is shown to have such powerful impact in the setting of coded distributed computing.

7.1.3 Chapter Outline

The rest of the chapter is organized as follows. First, the system model is presented in Section 7.2. Next, Section 7.3 provides the main contributions. An illustrative example of the novel coded scheme for multi-access distributed computing is then described in Section 7.4. After that, the general proofs of the achievable schemes and the converse bounds are presented from Section 7.5 to Section 7.8. Some additional proofs are provided in Appendix C.

7.2 System Model

The general distributed computing problem [52] consists of computing Q output functions from N input files with $Q, N \in \mathbb{N}^+$. Each file $w_n \in \mathbb{F}_{2^F}$ with $n \in [N]$ consists of F bits for some $F \in \mathbb{N}^+$, and the q-th function is defined as

$$\phi_q \colon \mathbb{F}_{2^F}^N \to \mathbb{F}_{2^B} \tag{7.1}$$

for each $q \in [Q]$, i.e., each function maps all the N input files into a stream $u_q = \phi_q(w_1, \ldots, w_N) \in \mathbb{F}_{2^B}$ of B bits. The main assumption is that each function ϕ_q is *decomposable* and so can be written as

$$\phi_q(w_1, \ldots, w_N) = h_q(g_{q,1}(w_1), \ldots, g_{q,N}(w_N)), \quad \forall q \in [Q] \tag{7.2}$$

where there is a map function $g_{q,n} \colon \mathbb{F}_{2^F} \to \mathbb{F}_{2^T}$ for each $n \in [N]$, which maps the input file w_n into an intermediate value (IV) $v_{q,n} = g_{q,n}(w_n) \in \mathbb{F}_{2^T}$ of T bits, and a reduce function $h_q \colon \mathbb{F}_{2^T}^N \to \mathbb{F}_{2^B}$, which maps all the IVs (one per input file) into the output value $u_q = h_q(v_{q,1}, \ldots, v_{q,N}) \in \mathbb{F}_{2^B}$ of B bits.

In this chapter, we assume to have machines that are devoted to computing map functions, and machines that are devoted to computing reduce functions. Thus, a node assigned map functions is not assigned reduce functions, and vice versa. In our setting, we consider Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers, where — in accordance with the combinatorial topology of choice — there is a unique reducer connected to each subset of α mappers. Denoting by $\mathcal{U} \in [\Lambda]_\alpha$ the reducer connected to the α mappers in the set $\mathcal{U}$, before the computation begins, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ is assigned a subset $\mathcal{W}_\mathcal{U} \subseteq [Q]$ of the output functions, where here $\mathcal{W}_\mathcal{U}$ contains the indices of the functions assigned to

reducer $\mathcal{U}$. For simplicity, we assume in our setting a symmetric and uniform task assignment, which implies $|\mathcal{W}_{\mathcal{U}}| = Q/K = \eta_2$ for some $\eta_2 \in \mathbb{N}^+$ and for each $\mathcal{U} \in [\Lambda]_\alpha$, and $\mathcal{W}_{\mathcal{U}_1} \cap \mathcal{W}_{\mathcal{U}_2} = \emptyset$ for all $\mathcal{U}_1, \mathcal{U}_2 \in [\Lambda]_\alpha$ such that $\mathcal{U}_1 \neq \mathcal{U}_2$. Afterwards, the computation is performed across the set of mappers and reducers in a distributed manner following the map-shuffle-reduce paradigm.

Table 7.1: Parameters for the MADC system with combinatorial topology

Number of Mappers	Λ
Multi-Access Degree	α
Number of Reducers	$K = \binom{\Lambda}{\alpha}$
Number of Input Files	N
Computation Load	r
Communication Load	L
Download Cost	J
Max-Link Load	$L_{\text{max-link}} = \max(L, J)$

During the map phase, a set of files $\mathcal{M}_\lambda \subseteq \{w_1, \ldots, w_N\}$ is assigned to the mapper λ for each $\lambda \in [\Lambda]$. Each mapper $\lambda \in [\Lambda]$ computes the intermediate values $\mathcal{V}_\lambda = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\lambda\}$ for all the Q reduce functions using the files in $\mathcal{M}_\lambda$ which it has been assigned. Since reducer $\mathcal{U} \in [\Lambda]_\alpha$ is connected to the mappers in $\mathcal{U}$, it can access the intermediate values in the set $\mathcal{V}_{\mathcal{U}} = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_{\mathcal{U}}\}$, where $\mathcal{M}_{\mathcal{U}} = \bigcup_{\lambda \in \mathcal{U}} \mathcal{M}_\lambda$ is simply the union set of files assigned to and mapped by the map nodes in $\mathcal{U}$. When the communication cost between mappers and reducers is not neglected, we can define the *download cost* as follows.

Definition 7.1 (Download Cost)**.** The *download cost*, denoted by J, is defined as the maximal normalized number of bits transmitted across the links from the mappers to the reducers, and is given by

$$J \coloneqq \max_{\lambda \in [\Lambda]} \max_{\mathcal{U} \in [\Lambda]_\alpha : \lambda \in \mathcal{U}} \frac{R_\lambda^{\mathcal{U}}}{QNT} \tag{7.3}$$

where $R_\lambda^{\mathcal{U}}$ denotes the number of bits that are transmitted by mapper $\lambda \in [\Lambda]$ to reducer $\mathcal{U} \in [\Lambda]_\alpha$ where $\lambda \in \mathcal{U}$.

Assuming that each mapper computes all possible IVs from locally available files[4], we define the *computation load* as follows.

Definition 7.2 (Computation Load)**.** The *computation load*, denoted by r, is defined as the total number of files mapped across the Λ map nodes and normalized by the total number of files N, and it takes the form

$$r \coloneqq \frac{\sum_{\lambda\in[\Lambda]} |\mathcal{M}_\lambda|}{N}. \tag{7.4}$$

Remark 7.1. We wish to point out that the definition of computation load above reflects the overall mapping cost across the Λ map nodes. As it will become clear later, our novel system model will allow for a massive computational amelioration (in the reduce phase) — with a bounded communication overhead — at the cost of a modest mapping cost across the Λ mappers.

During the shuffle phase, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ retrieves the IVs from the mappers in $\mathcal{U}$ and creates a signal $X_\mathcal{U} \in \mathbb{F}_{2^{\ell_\mathcal{U}}}$ for some $\ell_\mathcal{U} \in \mathbb{N}^+$ and for some encoding function $\psi_\mathcal{U} \colon \mathbb{F}_{2^T}^{Q|\mathcal{M}_\mathcal{U}|} \to \mathbb{F}_{2^{\ell_\mathcal{U}}}$, where $X_\mathcal{U}$ takes the form

$$X_\mathcal{U} = \psi_\mathcal{U}(\mathcal{V}_\mathcal{U}). \tag{7.5}$$

Then, the signal $X_\mathcal{U}$ is multicasted to all other reducers via the broadcast link which connects the reducers. Since such link is assumed to be error-free, each reducer receives all the multicast transmissions without errors. The amount of information exchanged during this phase is referred to as the *communication load*, which is formally defined in the following.

Definition 7.3 (Communication Load)**.** The *communication load*, denoted by L, is defined as the total number of bits transmitted by the K reducers over the broadcast channel during the shuffle phase, and — after normalization by the number of bits of all intermediate values — this load is given by

$$L \coloneqq \frac{\sum_{\mathcal{U}\in[\Lambda]_\alpha} \ell_\mathcal{U}}{QNT}. \tag{7.6}$$

Recalling that reducer $\mathcal{U} \in [\Lambda]_\alpha$ is assigned a subset of output functions whose indices are in $\mathcal{W}_\mathcal{U}$, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ wishes to recover the IVs $\{v_{q,n} : q \in \mathcal{W}_\mathcal{U}, n \in [N]\}$ to correctly compute u_q for each $q \in \mathcal{W}_\mathcal{U}$. Thus, during the reduce phase, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ reconstructs all the needed intermediate values for each $q \in \mathcal{W}_\mathcal{U}$ using the messages communicated in the

[4]This means that each mapper $\lambda \in [\Lambda]$ computes the intermediate value $v_{q,n}$ for each $q \in [Q]$ and for each $w_n \in \mathcal{M}_\lambda$.

shuffle phase and the intermediate values $\mathcal{V}_{\mathcal{U}}$ retrieved from the mappers in $\mathcal{U}$, i.e., each reducer $\mathcal{U} \in [\Lambda]_\alpha$ computes

$$(v_{q,1}, \ldots, v_{q,N}) = \chi^q_{\mathcal{U}}(X_{\mathcal{S}} : \mathcal{S} \in [\Lambda]_\alpha, \mathcal{V}_{\mathcal{U}}) \tag{7.7}$$

for each $q \in \mathcal{W}_{\mathcal{U}}$ and for some decoding function $\chi^q_{\mathcal{U}} \colon \prod_{\mathcal{S} \in [\Lambda]_\alpha} \mathbb{F}_{2^{\ell_{\mathcal{S}}}} \times \mathbb{F}_{2^T}^{Q|\mathcal{M}_{\mathcal{U}}|} \to \mathbb{F}_{2^T}^N$. In the end, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ computes the output function $u_q = h_q(v_{q,1}, \ldots, v_{q,N})$ for each assigned $q \in \mathcal{W}_{\mathcal{U}}$.

When the download cost is neglected, our goal is to characterize the optimal tradeoff between computation and communication $L^\star(r)$. This optimal tradeoff is simply defined as

$$L^\star(r) \coloneqq \inf\{L : (r, L) \text{ is achievable}\} \tag{7.8}$$

where the tuple (r, L) is said to be *achievable* if there exists a map-shuffle-reduce procedure such that a communication load L can be guaranteed for a given computation load r. On the other hand, when we indeed jointly consider both the inter-reducer communication cost and the mapper-to-reducer download cost, then our aim will be to characterize the optimal max-link communication load $L^\star_{\text{max-link}}(r)$, which is defined as

$$L^\star_{\text{max-link}}(r) \coloneqq \inf\{L_{\text{max-link}} : (r, L_{\text{max-link}}) \text{ is achievable}\} \tag{7.9}$$

where $L_{\text{max-link}} \coloneqq \max(L, J)$ represents the maximum between the communication load and the download cost for a given computation load r. In simple words, $L_{\text{max-link}}$ represents the maximal normalized number of bits flowing across any link in the considered system model. Notice that we will assume, throughout the chapter, uniform computational capabilities across the mappers and uniform assignment of reduce functions across the reducers, as is commonly assumed (see for example the original work in [52]).

Remark 7.2. When $\alpha = 1$, there are $K = \Lambda$ mapper-reducer pairs. If we consider each pair to be a single computing server (which can automatically imply a zero download cost), the proposed system model trivially coincides with the original setting in [52]. Hence, since the results in this chapter will hold for any $\alpha \in [\Lambda]$, the proposed model can in fact be considered as a proper extension of the original coded distributed computing model.

7.3 Main Results

In this section, we will provide our main contributions. As we have already mentioned, we will first consider a setting where the download cost is neglected. Subsequently, we will provide some additional results for the more realistic scenario where the cost of delivering data from the mappers to the reducers is non-zero.

7.3.1 Characterizing the Communication Load

The first result that we provide is the achievable computation-communication tradeoff provided by the novel coded scheme that will be presented in its general form in Section 7.5. The result is formally stated in the following theorem.

Theorem 7.1 (Achievable Bound)**.** *Consider the MADC setting with combinatorial topology, where there are Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers for a fixed value of $\alpha \in [\Lambda]$. Then, the optimal communication load $L^\star$ is upper bounded by L_{UB} which is a piecewise linear curve with corner points*

$$(r, L_{\text{UB}}) = \left(r, \frac{\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\binom{r+\alpha}{r} - 1\right)} \right), \quad \forall r \in [\Lambda - \alpha + 1]. \tag{7.10}$$

Proof. The detailed proof of the scheme is reported in Section 7.5, whereas an illustrative example is instead described in Section 7.4. □

As we already mentioned in Remark 7.2, if we set $\alpha = 1$ and we consider each mapper-reducer pair as a unique computing machine, we obtain the same system model in [52]. Interestingly, we can see that, if we specialize the result in Theorem 7.1 to the case $\alpha = 1$, we obtain the same computation-communication tradeoff as in [52, Theorem 1].

Another noteworthy aspect is the following. If we fix the number of mappers Λ and the computation load r, then adding more reducers by increasing[5] the multi-access degree α will in fact entail a smaller communication load. This (perhaps surprising) outcome is most certainly not the result of each reducer requiring fewer intermediate values during the shuffle phase. Such decrease could not have compensated for the increasing K. Instead, this decrease in the communication load stems from the nature of the combinatorial topology, which allows each reducer to more efficiently use its side information to cancel interference in an accelerated manner. This is achieved because these reducers are connected to the mappers in a manner that effectively aligns the interference patterns. As one can imagine, if we increase the number of reducers and we properly connect each of them to multiple mappers, the achievable scheme in Theorem 7.1 outperforms the coded scheme in [52]. This is formally stated in the following corollary.

Corollary 7.1.1. *For fixed computation load r, the achievable communication load in Theorem 7.1 decreases for increasing α, even though K — and so*

[5]Notice that the number $K = \binom{\Lambda}{\alpha}$ of reducers is actually increased as long as $\alpha \leq \Lambda/2$. The scenario where $\alpha > \Lambda/2$ is unrealistic and is not considered here.

the corresponding speedup factor in computing reduce functions — increases substantially.

Proof. The proof is described in Appendix C.1. □

We proceed to construct an information-theoretic converse on the communication load of the MADC setting. As it will be pointed out in the general proof in Section 7.6, the construction of the converse takes inspiration from [65, Lemma 2] as well as from ideas in Chapter 6. Essentially, the bound here manages to merge the approach in [65, Lemma 2], where a converse bound is built using key properties of the entropy function, with the index coding techniques in Chapter 6, where the nodes of a side information graph are iteratively selected in a proper way to systematically identify large acyclic subgraphs that are used to develop a tight converse. The result is formally stated in the following.

Theorem 7.2 (Converse Bound)**.** *Consider the MADC setting with combinatorial topology, where there are* Λ *mappers and* $K = \binom{\Lambda}{\alpha}$ *reducers for a fixed value of* $\alpha \in [\Lambda]$*. Then, the optimal communication load* $L^\star$ *is lower bounded by* L_{LB} *which is a piecewise linear curve with corner points*

$$(r, L_{\text{LB}}) = \left(r, \frac{\binom{\Lambda}{r+\alpha}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \right), \quad \forall r \in [\Lambda - \alpha + 1]. \tag{7.11}$$

Proof. The proof is described in Section 7.6. □

Finally, from the results in Theorem 7.1 and Theorem 7.2, we can provide an order optimality guarantee for the MADC model. Indeed, comparing the achievable performance and the converse bound, we conclude that the two are within a constant multiplicative gap. We see this in the following theorem[6].

Theorem 7.3 (Order Optimality)**.** *For the MADC system with combinatorial topology,* Λ *mappers and* $K = \binom{\Lambda}{\alpha}$ *reducers for a fixed value of* $\alpha \in [2 : \Lambda]$*, the achievable performance in Theorem 7.1 is within a factor of at most* 1.5 *from the optimal.*

Proof. The proof is described in Appendix C.2. □

[6]Notice that the order optimality result in Theorem 7.3 excludes the value $\alpha = 1$. Indeed, it can be verified that for such case the achievable performance in Theorem 7.1 and the converse in Theorem 7.2 are within a factor of at most 2. However, we already know that the coded scheme in [52] is exactly optimal when $\alpha = 1$. Hence, such value is neglected when comparing the aforementioned results.

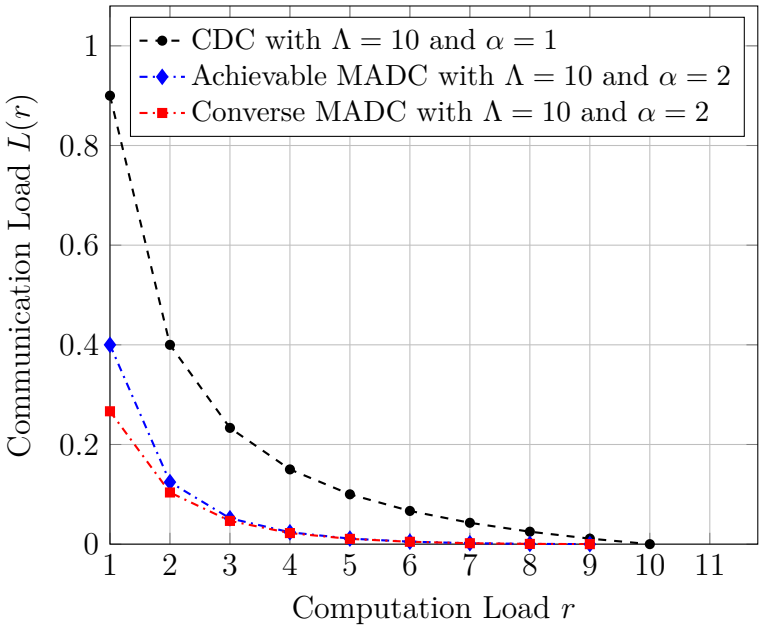

Figure 7.1: Comparison between original CDC, where there are $\Lambda = 10$ pairs of mappers and reducers, and MADC with combinatorial topology, $\Lambda = 10$ mappers and $K = 45$ reducers, where each of them is uniquely associated to $\alpha = 2$ mappers.

In Figure 7.1 we can see a comparison between the original CDC framework and the proposed MADC model. More specifically, for the first setting we consider $\Lambda = 10$ pairs of mappers and reducers, where each pair $\lambda \in [10]$ can be considered as a unique computing server having its own subset $\mathcal{M}_\lambda$ of assigned files. For the second setting we consider $\Lambda = 10$ mappers and $K = \binom{10}{2} = 45$ reducers, where there is a reducer connected to any $\alpha = 2$ mappers. According to Corollary 7.1.1, the achievable load in Theorem 7.1 decreases for increasing α and fixed computation load, as indicated by the diamond blue curve in Figure 7.1 which is well below the dot black counterpart corresponding to the original achievable scheme of coded distributed computing. Notice that the comparison in Figure 7.1 between the CDC setting with $\alpha = 1$ and the MADC setting with $\alpha > 1$ is fair for what concerns the computation-communication tradeoff: indeed, not only the computation load r remains the same as long as the number of mappers Λ stays the same, but also the number of reducers that need to communicate with each other is much larger than Λ when $\alpha > 1$.

Remark 7.3. We point out that comparing a setting where $\alpha = 1$ with a setting where $\alpha > 1$ offers noteworthy insights. Indeed, even though one could expect the communication load to be reduced when $\alpha > 1$ — as in

such case each reducer accesses more than one mapper and consequently misses less intermediate values — it is also true that the number of reducers itself increases as α increases. Consequently, there is an undeniable tension between the higher multi-access degree $\alpha > 1$ for each reducer, which implies less data needed by each reducer, but also implies a larger number of reducers in the system. For these reasons, it is interesting to notice how the network topology plays a fundamental role in resolving such tension by appropriately shaping the interference patterns. As a consequence, the communication load ultimately decreases as the number of reducers $K = \binom{\Lambda}{\alpha}$ increases as long as each of them is *properly* connected to α mappers.

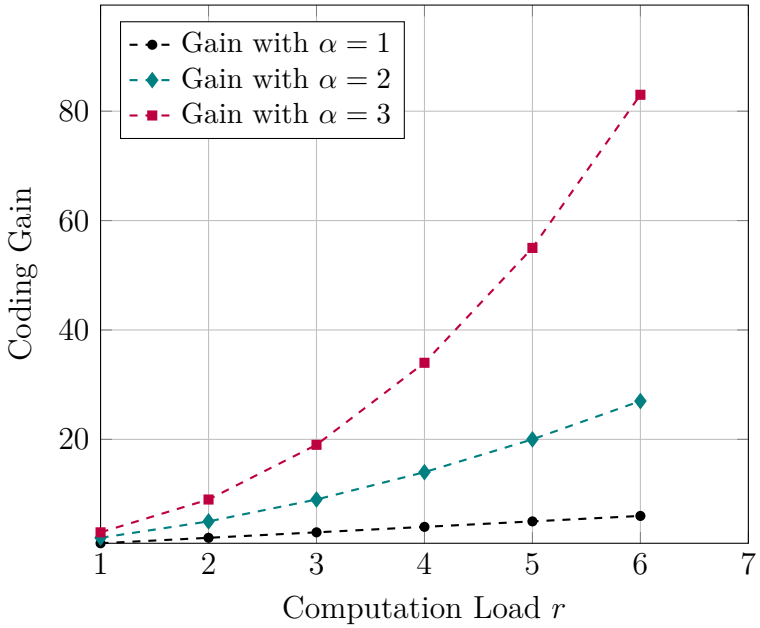

Figure 7.2: Comparison between the coding gain for different values of α as a function of the computation load r. We recall that $\alpha = 1$ corresponds to the original CDC framework.

A further comparison is provided in Figure 7.2 which focuses on the coding gains. As we can see, the gains brought about by the multi-access setting are impressive even when the computation load is small, which is the regime of interest in practical settings.

7.3.2 Characterizing the Max-Link Load

We now consider a distributed computing scenario where the download cost is not negligible. The following describes the achievable max-link communication

load that captures both communication and download costs.

Theorem 7.4 (Achievable Bound)**.** *Consider the MADC setting with combinatorial topology, where there are Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers for a fixed value of $\alpha \in [\Lambda]$. Then, the optimal max-link communication load $L^{\star}_{\text{max-link}}$ is upper bounded by $L_{\text{max-link,UB}}$ which is given by*

$$L_{\text{max-link,UB}} = \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)}\frac{\tilde{a}^j_\star}{N}, \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}\frac{\tilde{a}^j_\star}{N}\right) \tag{7.12}$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \ldots, \tilde{a}^\Lambda_\star)$ is the optimal solution to the linear program

$$\min_{\tilde{\boldsymbol{a}}_{\mathcal{M}}} \quad \frac{1}{2}\sum_{j\in[\Lambda]}\left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}\right)\frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{7.13a}$$

$$\text{subject to} \quad \tilde{a}^j_{\mathcal{M}} \geq 0, \quad \forall j \in [\Lambda] \tag{7.13b}$$

$$\sum_{j\in[\Lambda]} \frac{\tilde{a}^j_{\mathcal{M}}}{N} = 1 \tag{7.13c}$$

$$\sum_{j\in[\Lambda]} j\frac{\tilde{a}^j_{\mathcal{M}}}{N} \leq r \tag{7.13d}$$

and where $\tilde{\boldsymbol{a}}_{\mathcal{M}} = (\tilde{a}^1_{\mathcal{M}}, \ldots, \tilde{a}^\Lambda_{\mathcal{M}})$ is the control variable.

Proof. The detailed proof of the scheme is reported in Section 7.7. □

We proceed by proposing an information-theoretic converse on the max-link communication load. The result is presented in the following theorem.

Theorem 7.5 (Converse Bound)**.** *Consider the MADC setting with combinatorial topology, where there are Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers for a fixed value of $\alpha \in [\Lambda]$. Then, the optimal max-link communication load $L^{\star}_{\text{max-link}}$ is lower bounded by $L_{\text{max-link,LB}}$ which is given by*

$$L_{\text{max-link,LB}} = \frac{1}{2}\sum_{j\in[\Lambda]}\left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}\right)\frac{\tilde{a}^j_\star}{N} \tag{7.14}$$

where the vector $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \ldots, \tilde{a}^\Lambda_\star)$ is the optimal solution to the linear program in (7.13)*.*

Proof. The proof is described in Section 7.8. □

Finally, we can compare the results in Theorem 7.4 and Theorem 7.5 to establish the gap to optimality of the achievable performance in Theorem 7.4. Notice that now we do not exclude the value $\alpha = 1$ for such comparison, since for such case there is no previously known optimality result to the best of our knowledge.

Theorem 7.6 (Order Optimality)**.** *For the MADC system with combinatorial topology, Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers for a fixed value of $\alpha \in [\Lambda]$, the achievable performance in Theorem 7.4 is within a factor of at most 4 from the optimal.*

Proof. The proof is described in Appendix C.3. □

Remark 7.4. Interestingly, the results in this section can also be derived — while keeping the same constant factor of at most 4 from the optimal — for a weighted max-link load defined as $L_{\text{max-link},\beta} \coloneqq \max(L, \beta J)$ for some $\beta \geq 0$. Notice that such metric is of particular interest whenever we want to account for a *weighted* download cost βJ, in which case the cost of communicating data from mappers to reducers is *proportional* to the maximal number of normalized bits flowing from any one mapper to any one reducer. The extreme case $\beta = 0$ corresponds to the communication load $L_{\text{max-link},0} = L$ characterized in Section 7.3.1, whereas the case $\beta = 1$ implies the max-link load $L_{\text{max-link},1} = L_{\text{max-link}}$ investigated in the current section.

7.4 Illustrative Example of the Coded Scheme

In this section, we propose an illustrative example of the coded scheme which will be later presented in its general form in Section 7.5. The example refers to the schematic in Figure 1.3. Notice that this section aims to provide an example for the achievable scheme in Theorem 7.1, consequently the download cost will be neglected.

We consider $\Lambda = 4$ mappers and $K = \binom{\Lambda}{\alpha} = 6$ reducers where $\alpha = 2$. We assume to have $Q = 12$ output functions to be computed across the ensemble of mappers and reducers, $N = 8$ input files $\{w_n : n \in [8]\}$ and computation load $r = 1$. Recalling that the computation load is defined as the normalized number of files which are mapped across the Λ map nodes, having computation load equal to $r = 1$ implies that the number of files mapped across all mappers is equal to the size of the input data set, i.e., N files. Under the uniform function assignment assumption, we assign $Q/K = 2$

output functions to each reducer[7] $\mathcal{U} \in [4]_2 = \{12, 13, 14, 23, 24, 34\}$. Thus, recalling that $\mathcal{W}_{\mathcal{U}}$ represents the indices of the reduce functions assigned to reducer $\mathcal{U} \in [4]_2$, we arbitrarily let

$$\mathcal{W}_{12} = \{1, 2\} \tag{7.15}$$
$$\mathcal{W}_{13} = \{3, 4\} \tag{7.16}$$
$$\mathcal{W}_{14} = \{5, 6\} \tag{7.17}$$
$$\mathcal{W}_{23} = \{7, 8\} \tag{7.18}$$
$$\mathcal{W}_{24} = \{9, 10\} \tag{7.19}$$
$$\mathcal{W}_{34} = \{11, 12\}. \tag{7.20}$$

7.4.1 Map Phase

This phase requires the input files to be split among mappers and so we proceed by grouping the $N = 8$ files into 4 batches $\mathcal{B}_\lambda$ for each $\lambda \in [4]$ as

$$\mathcal{B}_1 = \{w_1, w_2\} \tag{7.21}$$
$$\mathcal{B}_2 = \{w_3, w_4\} \tag{7.22}$$
$$\mathcal{B}_3 = \{w_5, w_6\} \tag{7.23}$$
$$\mathcal{B}_4 = \{w_7, w_8\}. \tag{7.24}$$

Then, recalling that $\mathcal{M}_\lambda$ represents the set of files mapped by the mapper $\lambda \in [\Lambda]$, we set here $\mathcal{M}_\lambda = \{\mathcal{B}_{\mathcal{T}_1} : \mathcal{T}_1 \subseteq [4], |\mathcal{T}_1| = 1, \lambda \in \mathcal{T}_1\} = \{\mathcal{B}_\lambda\}$.

Since each mapper is assigned 2 input files, we have that $|\mathcal{M}_\lambda| = |\mathcal{B}_\lambda| = 2$ for each $\lambda \in [4]$. Hence, we can check that such file assignment satisfies the computation load constraint $r = 1$, as indeed we have

$$\frac{\sum_{\lambda \in [4]} |\mathcal{M}_\lambda|}{N} = \frac{|\mathcal{M}_1| + |\mathcal{M}_2| + |\mathcal{M}_3| + |\mathcal{M}_4|}{8} = 1. \tag{7.25}$$

Each mapper computes $Q = 12$ intermediate values for each assigned input file. In particular, recalling that $\mathcal{V}_\lambda$ is the set of IVs computed by the mapper $\lambda \in [4]$, we have

$$\mathcal{V}_1 = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_1\} \tag{7.26}$$
$$\mathcal{V}_2 = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_2\} \tag{7.27}$$
$$\mathcal{V}_3 = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_3\} \tag{7.28}$$

[7] For the sake of simplicity, we will often omit braces and commas when indicating sets, e.g., the reducer $\{1, 2\}$, which is connected to mappers 1 and 2, can be simply denoted as 12.

$$\mathcal{V}_4 = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_4\}. \tag{7.29}$$

For example, since mapper 1 is assigned the files in $\mathcal{M}_1 = \{w_1, w_2\}$, it will compute the intermediate values $v_{q,1}$ and $v_{q,2}$ for all $q \in [12]$. Then, mapper 2 will compute the intermediate values $v_{q,3}$ and $v_{q,4}$ for all $q \in [12]$ since $\mathcal{M}_2 = \{w_3, w_4\}$. Similarly, mapper 3 and mapper 4 will compute the intermediate values $v_{q,5}$ and $v_{q,6}$, and $v_{q,7}$ and $v_{q,8}$, respectively, for all $q \in [12]$ since $\mathcal{M}_3 = \{w_5, w_6\}$ and $\mathcal{M}_4 = \{w_7, w_8\}$.

Now, considering that there is a reducer connected to any $\alpha = 2$ mappers, we know that each reducer $\mathcal{U} \in [4]_2$ can retrieve the IVs computed by the 2 mappers in $\mathcal{U}$. Recalling that $\mathcal{V}_\mathcal{U}$ denotes the union set of IVs computed by the mappers in $\mathcal{U}$, we have

$$\mathcal{V}_{12} = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_{12}\} \tag{7.30}$$
$$\mathcal{V}_{13} = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_{13}\} \tag{7.31}$$
$$\mathcal{V}_{14} = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_{14}\} \tag{7.32}$$
$$\mathcal{V}_{23} = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_{23}\} \tag{7.33}$$
$$\mathcal{V}_{24} = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_{24}\} \tag{7.34}$$
$$\mathcal{V}_{34} = \{v_{q,n} : q \in [12], w_n \in \mathcal{M}_{34}\}. \tag{7.35}$$

7.4.2 Shuffle Phase

We describe now how each reducer $\mathcal{U} \in [4]_2$ constructs its multicast message $X_\mathcal{U}$. Since the procedure is the same for each reducer, we continue our example by focusing for simplicity on reducer $\{1, 2\}$ only.

First of all, we let $\mathcal{S} \subseteq ([4] \setminus \{1, 2\})$ with $|\mathcal{S}| = 1$. Then, for each $\mathcal{R} \subseteq (\mathcal{S} \cup \{1, 2\})$ such that $|\mathcal{R}| = 2$ and $\mathcal{R} \neq \{1, 2\}$, and for $\mathcal{T}_1 = (\mathcal{S} \cup \{1, 2\}) \setminus \mathcal{R}$, reducer $\{1, 2\}$ concatenates the intermediate values $\{v_{q,n} : q \in \mathcal{W}_\mathcal{R}, w_n \in \mathcal{B}_{\mathcal{T}_1}\}$ into the symbol $U_{\mathcal{W}_\mathcal{R}, \mathcal{T}_1} = (v_{q,n} : q \in \mathcal{W}_\mathcal{R}, w_n \in \mathcal{B}_{\mathcal{T}_1})$. Notice that having $\mathcal{R} \neq \{1, 2\}$ implies that $\mathcal{T}_1 \cap \{1, 2\} \neq \emptyset$, so reducer $\{1, 2\}$ can retrieve $\mathcal{B}_{\mathcal{T}_1}$ from the mappers it is connected to and can construct the symbol $U_{\mathcal{W}_\mathcal{R}, \mathcal{T}_1}$. Subsequently, such symbol is evenly split as

$$U_{\mathcal{W}_\mathcal{R}, \mathcal{T}_1} = (U_{\mathcal{W}_\mathcal{R}, \mathcal{T}_1, \mathcal{T}_2} : \mathcal{T}_2 \subseteq (\mathcal{R} \cup \mathcal{T}_1), |\mathcal{T}_2| = 2, \mathcal{T}_2 \neq \mathcal{R}). \tag{7.36}$$

This means that when, say, $\mathcal{S} = \{3\}$, reducer $\{1, 2\}$ creates the symbols

$$U_{\mathcal{W}_{13}, 2} = (v_{q,n} : q \in \mathcal{W}_{13}, w_n \in \mathcal{B}_2) \tag{7.37}$$
$$U_{\mathcal{W}_{23}, 1} = (v_{q,n} : q \in \mathcal{W}_{23}, w_n \in \mathcal{B}_1) \tag{7.38}$$

and when $\mathcal{S} = \{4\}$, the same reducer $\{1, 2\}$ creates the symbols

$$U_{\mathcal{W}_{14}, 2} = (v_{q,n} : q \in \mathcal{W}_{14}, w_n \in \mathcal{B}_2) \tag{7.39}$$

$$U_{\mathcal{W}_{24},1} = (v_{q,n} : q \in \mathcal{W}_{24}, w_n \in \mathcal{B}_1). \tag{7.40}$$

Each of the symbols above is then evenly split in two segments as

$$U_{\mathcal{W}_{13},2} = (U_{\mathcal{W}_{13},2,12}, U_{\mathcal{W}_{13},2,23}) \tag{7.41}$$
$$U_{\mathcal{W}_{23},1} = (U_{\mathcal{W}_{23},1,12}, U_{\mathcal{W}_{23},1,13}) \tag{7.42}$$
$$U_{\mathcal{W}_{14},2} = (U_{\mathcal{W}_{14},2,12}, U_{\mathcal{W}_{14},2,24}) \tag{7.43}$$
$$U_{\mathcal{W}_{24},1} = (U_{\mathcal{W}_{24},1,12}, U_{\mathcal{W}_{24},1,14}). \tag{7.44}$$

Next, reducer $\{1,2\}$ constructs the coded message

$$\bigoplus_{\mathcal{R}\subseteq(\mathcal{S}\cup\{1,2\}):|\mathcal{R}|=2,\mathcal{R}\neq\{1,2\}} U_{\mathcal{W}_{\mathcal{R}},(\mathcal{S}\cup\{1,2\})\backslash\mathcal{R},12} \tag{7.45}$$

for each $\mathcal{S} \subseteq ([4] \setminus \{1,2\})$ with $|\mathcal{S}| = 1$, and concatenates all of them to form X_{12}, which is given by

$$X_{12} = \left(\bigoplus_{\substack{\mathcal{R}\subseteq(\mathcal{S}\cup\{1,2\}):\\ |\mathcal{R}|=2,\mathcal{R}\neq\{1,2\}}} U_{\mathcal{W}_{\mathcal{R}},(\mathcal{S}\cup\{1,2\})\backslash\mathcal{R},12} : \mathcal{S} \subseteq ([\Lambda] \setminus \{1,2\}), |\mathcal{S}| = 1 \right) \tag{7.46}$$
$$= (U_{\mathcal{W}_{13},2,12} \oplus U_{\mathcal{W}_{23},1,12}, U_{\mathcal{W}_{24},1,12} \oplus U_{\mathcal{W}_{14},2,12}). \tag{7.47}$$

Similarly, each other reducer prepares and multicasts its message following the procedure described above. In the end, the following messages

$$X_{12} = (U_{\mathcal{W}_{13},2,12} \oplus U_{\mathcal{W}_{23},1,12}, U_{\mathcal{W}_{24},1,12} \oplus U_{\mathcal{W}_{14},2,12}) \tag{7.48}$$
$$X_{13} = (U_{\mathcal{W}_{12},3,13} \oplus U_{\mathcal{W}_{23},1,13}, U_{\mathcal{W}_{14},3,13} \oplus U_{\mathcal{W}_{34},1,13}) \tag{7.49}$$
$$X_{14} = (U_{\mathcal{W}_{12},4,14} \oplus U_{\mathcal{W}_{24},1,14}, U_{\mathcal{W}_{13},4,14} \oplus U_{\mathcal{W}_{34},1,14}) \tag{7.50}$$
$$X_{23} = (U_{\mathcal{W}_{12},3,23} \oplus U_{\mathcal{W}_{13},2,23}, U_{\mathcal{W}_{24},3,23} \oplus U_{\mathcal{W}_{34},2,23}) \tag{7.51}$$
$$X_{24} = (U_{\mathcal{W}_{12},4,24} \oplus U_{\mathcal{W}_{14},2,24}, U_{\mathcal{W}_{23},4,24} \oplus U_{\mathcal{W}_{34},2,24}) \tag{7.52}$$
$$X_{34} = (U_{\mathcal{W}_{13},4,34} \oplus U_{\mathcal{W}_{14},3,34}, U_{\mathcal{W}_{23},4,34} \oplus U_{\mathcal{W}_{24},3,34}) \tag{7.53}$$

are exchanged among the reducers on the common-bus link during the shuffle phase.

7.4.3 Reduce Phase

As when describing the shuffle phase, we can again focus on reducer $\{1,2\}$ and observe how it correctly computes the reduce functions in $\mathcal{W}_{12}$ by using the set of multicast messages $\{X_{\mathcal{U}} : \mathcal{U} \in [4]_2\}$ and the set $\mathcal{V}_{12}$ of IVs which the reducer $\{1,2\}$ can access. Indeed, a similar procedure can be shown for all other reducers.

First of all, reducer $\{1,2\}$ needs the IVs $\{v_{q,n} : q \in \mathcal{W}_{12}, n \in [8]\}$ to compute the reduce functions in $\mathcal{W}_{12}$. Since such reducer has already access to the IVs in $\mathcal{V}_{12}$, it can obtain the symbols $U_{\mathcal{W}_{12},1}$ and $U_{\mathcal{W}_{12},2}$. However, it misses the intermediate values $\{v_{q,n} : q \in \mathcal{W}_{12}, w_n \notin \mathcal{M}_{12}\}$ or, similarly, it misses the symbols $U_{\mathcal{W}_{12},3} = (v_{q,n} : q \in \mathcal{W}_{12}, w_n \in \mathcal{B}_3)$ and $U_{\mathcal{W}_{12},4} = (v_{q,n} : q \in \mathcal{W}_{12}, w_n \in \mathcal{B}_4)$. We see now how these symbols can be obtained from the set of multicast messages.

During the shuffle procedure, each symbol is split in two even segments, so, consequently, symbols $U_{\mathcal{W}_{12},3}$ and $U_{\mathcal{W}_{12},4}$ are split as

$$U_{\mathcal{W}_{12},3} = (U_{\mathcal{W}_{12},3,13}, U_{\mathcal{W}_{12},3,23}) \tag{7.54}$$

$$U_{\mathcal{W}_{12},4} = (U_{\mathcal{W}_{12},4,14}, U_{\mathcal{W}_{12},4,24}). \tag{7.55}$$

Now, reducer $\{1,2\}$ can decode $U_{\mathcal{W}_{12},3,13}$ from the message X_{13}. Indeed, the term $U_{\mathcal{W}_{12},3,13} \oplus U_{\mathcal{W}_{23},1,13}$ appears in X_{13} and reducer $\{1,2\}$ can use the IVs in $\mathcal{V}_{12}$ to cancel the interference term $U_{\mathcal{W}_{23},1,13}$. Similarly, the term $U_{\mathcal{W}_{12},3,23} \oplus U_{\mathcal{W}_{13},2,23}$ appears in X_{23}, where again the interference $U_{\mathcal{W}_{13},2,23}$ can be canceled by means of the IVs retrieved by the mappers 1 and 2. Hence, reducer $\{1,2\}$ successfully decodes $U_{\mathcal{W}_{12},3,13}$ and $U_{\mathcal{W}_{12},3,23}$ from the multicasted messages X_{13} and X_{23}, reconstructing then the symbol $U_{\mathcal{W}_{12},3} = (U_{\mathcal{W}_{12},3,13}, U_{\mathcal{W}_{12},3,23})$. A similar procedure holds for reducer $\{1,2\}$ to successfully reconstruct the symbol $U_{\mathcal{W}_{12},4}$, whose two segments are decoded from messages X_{14} and X_{24}. Further, a similar procedure holds for any other reducer. Thus, we can conclude that every reducer is able to compute the assigned reduce functions after recovering the missing intermediate values from the messages multicasted by all reducers during the shuffle phase.

7.4.4 Communication Load

Recalling that the communication load is defined as the total number of bits transmitted by the K reducers during the shuffle phase (normalized by the number of bits of all intermediate values), we wish to compute for this example this load, which takes the form

$$L_{\text{UB}}(r=1) = \frac{\sum_{\mathcal{U} \in [\Lambda]_\alpha} |X_{\mathcal{U}}|}{QNT} = \frac{\sum_{\mathcal{U} \in [4]_2} |X_{\mathcal{U}}|}{96T}. \tag{7.56}$$

Since $|X_{\mathcal{U}}|$ is the same for each $\mathcal{U} \in [4]_2$, we focus again on reducer $\{1,2\}$ and its multicast transmission $X_{12} = (U_{\mathcal{W}_{13},2,12} \oplus U_{\mathcal{W}_{23},1,12}, U_{\mathcal{W}_{24},1,12} \oplus U_{\mathcal{W}_{14},2,12})$, which contains two XOR messages. Focusing on the first message $U_{\mathcal{W}_{13},2,12} \oplus U_{\mathcal{W}_{23},1,12}$, we can see that it is a XOR composed of two segments, i.e., one segment for the symbol $U_{\mathcal{W}_{13},2}$ and one segment for the symbol $U_{\mathcal{W}_{23},1}$. Since

the size of each symbol is $4T$ bits, the resulting XOR message has size $2T$ bits. Hence, given that X_{12} contains two XOR messages, we can conclude that $|X_{12}| = 4T$ bits. Consequently, the resulting achievable communication load is given by

$$L_{\text{UB}}(r=1) = \frac{\sum_{\mathcal{U}\in[4]_2} |X_{\mathcal{U}}|}{96T} = \frac{24T}{96T} = \frac{1}{4}. \tag{7.57}$$

Using the converse in Theorem 7.2, we can see that the achievable performance above is within a factor 1.5 from the optimal.

7.5 Proof of Achievable Bound in Theorem 7.1

We assume that there are Λ mappers and $K = \binom{\Lambda}{\alpha}$ reducers, and we assume the aforementioned combinatorial topology where each reducer is exactly and uniquely connected to α mappers. We then consider some arbitrary computation load $r \in [\Lambda - \alpha + 1]$ and we consider $Q = \eta_2 K$ output functions with $\eta_2 \in \mathbb{N}^+$, allowing us to separate the Q functions into K disjoint groups $\mathcal{W}_{\mathcal{U}}$ for each $\mathcal{U} \in [\Lambda]_\alpha$, so that each reducer is assigned η_2 functions, corresponding to $|\mathcal{W}_{\mathcal{U}}| = \eta_2$ for each $\mathcal{U} \in [\Lambda]_\alpha$.

7.5.1 Map Phase

First, the input database is split in $\binom{\Lambda}{r}$ disjoint batches, each containing $\eta_1 = N/\binom{\Lambda}{r}$ files, where we assume that N is large enough such that $\eta_1 \in \mathbb{N}^+$. Consequently, we have a batch of files for each $\mathcal{T}_1 \subseteq [\Lambda]$ such that $|\mathcal{T}_1| = r$, which implies

$$\{w_1, \ldots, w_N\} = \bigcup_{\mathcal{T}_1 \subseteq [\Lambda]: |\mathcal{T}_1| = r} \mathcal{B}_{\mathcal{T}_1} \tag{7.58}$$

where we denote by $\mathcal{B}_{\mathcal{T}_1}$ the batch of η_1 files associated with the label $\mathcal{T}_1$. Then, mapper $\lambda \in [\Lambda]$ is assigned all batches $\mathcal{B}_{\mathcal{T}_1}$ having $\lambda \in \mathcal{T}_1$, which means that

$$\mathcal{M}_\lambda = \{\mathcal{B}_{\mathcal{T}_1} : \mathcal{T}_1 \subseteq [\Lambda], |\mathcal{T}_1| = r, \lambda \in \mathcal{T}_1\}. \tag{7.59}$$

We can see that the computation load constraint is satisfied, since we have

$$\frac{\sum_{\lambda\in[\Lambda]} |\mathcal{M}_\lambda|}{N} = \frac{\Lambda \eta_1 \binom{\Lambda-1}{r-1}}{\eta_1 \binom{\Lambda}{r}} = r \tag{7.60}$$

Then, each mapper computes Q intermediate values for each assigned input file, so for each $\lambda \in [\Lambda]$ we have $\mathcal{V}_\lambda = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\lambda\}$. Since

then each reducer has access to α mappers, reducer $\mathcal{U} \in [\Lambda]_\alpha$ can retrieve[8] the intermediate values in $\mathcal{V}_\mathcal{U} = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\mathcal{U}\}$ recalling that $\mathcal{M}_\mathcal{U} = \cup_{\lambda \in \mathcal{U}} \mathcal{M}_\lambda$. Since $|\mathcal{V}_\mathcal{U}| = Q\eta_1 \left(\binom{\Lambda}{r} - \binom{\Lambda-\alpha}{r}\right)$ for each $\mathcal{U} \in [\Lambda]_\alpha$, we can conclude that each computing node has access to all the intermediate values when $r \geq \Lambda - \alpha + 1$. Hence, we focus on the non-trivial regime $r \in [\Lambda - \alpha + 1]$ for any given Λ and α.

7.5.2 Shuffle Phase

Consider reducer $\mathcal{U} \in [\Lambda]_\alpha$. Let $\mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{U})$ with $|\mathcal{S}| = r$. First, for each $\mathcal{R} \subseteq (\mathcal{S} \cup \mathcal{U})$ such that $|\mathcal{R}| = \alpha$ and $\mathcal{R} \neq \mathcal{U}$, and for $\mathcal{T}_1 = (\mathcal{S} \cup \mathcal{U}) \setminus \mathcal{R}$, reducer $\mathcal{U}$ concatenates the intermediate values $\{v_{q,n} : q \in \mathcal{W}_\mathcal{R}, w_n \in \mathcal{B}_{\mathcal{T}_1}\}$ into the symbol $U_{\mathcal{W}_\mathcal{R},\mathcal{T}_1} = (v_{q,n} : q \in \mathcal{W}_\mathcal{R}, w_n \in \mathcal{B}_{\mathcal{T}_1}) \in \mathbb{F}_{2^{\eta_2\eta_1 T}}$. Subsequently, such symbol is evenly split in $\left(\binom{r+\alpha}{r} - 1\right)$ segments as

$$U_{\mathcal{W}_\mathcal{R},\mathcal{T}_1} = (U_{\mathcal{W}_\mathcal{R},\mathcal{T}_1,\mathcal{T}_2} : \mathcal{T}_2 \subseteq (\mathcal{R} \cup \mathcal{T}_1), |\mathcal{T}_2| = \alpha, \mathcal{T}_2 \neq \mathcal{R}) . \tag{7.61}$$

Then, reducer $\mathcal{U}$ constructs the coded message

$$\bigoplus_{\mathcal{R} \subseteq (\mathcal{S} \cup \mathcal{U}) : |\mathcal{R}| = \alpha, \mathcal{R} \neq \mathcal{U}} U_{\mathcal{W}_\mathcal{R},(\mathcal{S}\cup\mathcal{U})\setminus\mathcal{R},\mathcal{U}} \tag{7.62}$$

for each $\mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{U})$ with $|\mathcal{S}| = r$, and finally concatenates all of them into the following message

$$X_\mathcal{U} = \left(\bigoplus_{\mathcal{R} \subseteq (\mathcal{S} \cup \mathcal{U}) : |\mathcal{R}| = \alpha, \mathcal{R} \neq \mathcal{U}} U_{\mathcal{W}_\mathcal{R},(\mathcal{S}\cup\mathcal{U})\setminus\mathcal{R},\mathcal{U}} : \mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{U}), |\mathcal{S}| = r \right) \tag{7.63}$$

which is multicasted to all other reducers via the error-free broadcast channel.

7.5.3 Reduce Phase

Consider reducer $\mathcal{U} \in [\Lambda]_\alpha$. Since such reducer is connected to α mappers, it misses a total of $\eta_2\eta_1\binom{\Lambda-\alpha}{r}$ intermediate values, i.e, it misses η_2 intermediate values for each of the η_1 files in each batch that is not assigned to the mappers in $\mathcal{U}$. More precisely, reducer $\mathcal{U}$ misses the symbol $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1}$ for each $\mathcal{T}_1 \subseteq ([\Lambda] \setminus \mathcal{U})$ with $|\mathcal{T}_1| = r$. We know that during the shuffle phase such symbol is evenly split in $\left(\binom{r+\alpha}{r} - 1\right)$ segments as

$$U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1} = (U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2} : \mathcal{T}_2 \subseteq (\mathcal{U} \cup \mathcal{T}_1), |\mathcal{T}_2| = \alpha, \mathcal{T}_2 \neq \mathcal{U}) . \tag{7.64}$$

[8]Since we are presenting here the proof of the achievable bound in Theorem 7.1, we will neglect the download cost, assuming consequently that each reducer can access the IVs without any additional communication cost.

For each $\mathcal{T}_2 \subseteq (\mathcal{U} \cup \mathcal{T}_1)$ with $|\mathcal{T}_2| = \alpha$ and $\mathcal{T}_2 \neq \mathcal{U}$, we can verify that reducer $\mathcal{U}$ can decode $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2}$ from $X_{\mathcal{T}_2}$. Indeed, there exists an $\mathcal{S} \subseteq ([\Lambda] \setminus \mathcal{T}_2)$ with $|\mathcal{S}| = r$ such that $\mathcal{S} = (\mathcal{U} \cup \mathcal{T}_1) \setminus \mathcal{T}_2$. For such $\mathcal{S}$, the corresponding coded message in $X_{\mathcal{T}_2}$ is

$$\bigoplus_{\substack{\mathcal{R}\subseteq(\mathcal{S}\cup\mathcal{T}_2):\\ |\mathcal{R}|=\alpha,\mathcal{R}\neq\mathcal{T}_2}} U_{\mathcal{W}_\mathcal{R},(\mathcal{S}\cup\mathcal{T}_2)\setminus\mathcal{R},\mathcal{T}_2} = \bigoplus_{\substack{\mathcal{R}\subseteq(\mathcal{U}\cup\mathcal{T}_1):\\ |\mathcal{R}|=\alpha,\mathcal{R}\neq\mathcal{T}_2}} U_{\mathcal{W}_\mathcal{R},(\mathcal{U}\cup\mathcal{T}_1)\setminus\mathcal{R},\mathcal{T}_2} \tag{7.65}$$

$$= U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2} \oplus \underbrace{\bigoplus_{\substack{\mathcal{R}\subseteq(\mathcal{U}\cup\mathcal{T}_1):\\ |\mathcal{R}|=\alpha,\mathcal{R}\neq\mathcal{T}_2,\mathcal{R}\neq\mathcal{U}}} U_{\mathcal{W}_\mathcal{R},(\mathcal{U}\cup\mathcal{T}_1)\setminus\mathcal{R},\mathcal{T}_2}}_{\text{interference}}. \tag{7.66}$$

Notice that reducer $\mathcal{U}$ can cancel the interference term by using the intermediate values retrieved from mappers in $\mathcal{U}$, so it can correctly decode $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1,\mathcal{T}_2}$. By following the same rationale for each $\mathcal{T}_2 \subseteq (\mathcal{U} \cup \mathcal{T}_1)$ with $|\mathcal{T}_2| = \alpha$ and $\mathcal{T}_2 \neq \mathcal{U}$, we can conclude that reducer $\mathcal{U}$ can correctly recover $U_{\mathcal{W}_\mathcal{U},\mathcal{T}_1}$ and can do so for each $\mathcal{T}_1 \subseteq ([\Lambda] \setminus \mathcal{U})$, completely recovering all the $\eta_2\eta_1\binom{\Lambda-\alpha}{r}$ missing intermediate values. The same holds for any other $\mathcal{U} \in [\Lambda]_\alpha$, and so we can conclude that each reducer is able to recover from the multicast messages of other reducers all the missing intermediate values.

7.5.4 Communication Load

The communication load guaranteed by the coded scheme described above is given by

$$L_{\text{UB}} = \frac{\sum_{\mathcal{U}\in[\Lambda]_\alpha} |X_\mathcal{U}|}{QNT} \tag{7.67}$$

$$= \frac{\binom{\Lambda}{\alpha}\eta_2\eta_1\binom{\Lambda-\alpha}{r}T/\left(\binom{r+\alpha}{r}-1\right)}{Q\eta_1\binom{\Lambda}{r}T} \tag{7.68}$$

$$= \frac{\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\binom{r+\alpha}{r}-1\right)} \tag{7.69}$$

for each $r \in [\Lambda - \alpha + 1]$. Notice that the lower convex envelope of the achievable points $\{(r, L_{\text{LB}}) : r \in [\Lambda - \alpha + 1]\}$ is achievable by adopting the memory-sharing strategy presented in [52]. The proof is concluded. □

7.6 Proof of Converse Bound in Theorem 7.2

We begin the proof by introducing some useful notation. For $q \in [Q]$ and $n \in [N]$, we let $V_{q,n}$ be an i.i.d. random variable and we let $v_{q,n}$ be the realization of $V_{q,n}$. Then, we define

$$D_{\mathcal{U}} \coloneqq \{V_{q,n} : q \in \mathcal{W}_{\mathcal{U}}, n \in [N]\} \tag{7.70}$$
$$C_{\mathcal{U}} \coloneqq \{V_{q,n} : q \in [Q], w_n \in \mathcal{M}_{\mathcal{U}}\} \tag{7.71}$$
$$Y_{\mathcal{U}} \coloneqq (D_{\mathcal{U}}, C_{\mathcal{U}}). \tag{7.72}$$

Recalling that we denote by $X_{\mathcal{U}}$ the multicast message transmitted by reducer $\mathcal{U} \in [\Lambda]_\alpha$, the equation

$$H(X_{\mathcal{U}} \mid C_{\mathcal{U}}) = 0 \tag{7.73}$$

holds, since $X_{\mathcal{U}}$ is a function of the intermediate values retrieved by reducer $\mathcal{U}$. Moreover, for any map-shuffle-reduce scheme, each reducer $\mathcal{U} \in [\Lambda]_\alpha$ has to be able to correctly recover all the intermediate values $D_{\mathcal{U}}$ given the transmissions of all reducers $X_{[\Lambda]_\alpha} \coloneqq (X_{\mathcal{U}} : \mathcal{U} \in [\Lambda]_\alpha)$ and given the IVs $C_{\mathcal{U}}$ computed by the mappers in $\mathcal{U}$. Thus, the equation

$$H(D_{\mathcal{U}} \mid X_{[\Lambda]_\alpha}, C_{\mathcal{U}}) = 0 \tag{7.74}$$

holds for each $\mathcal{U} \in [\Lambda]_\alpha$.

7.6.1 Lower Bound for a Given File Assignment

For a given file assignment denoted by $\mathcal{M} \coloneqq (\mathcal{M}_1, \ldots, \mathcal{M}_\Lambda)$, we let $L_{\mathcal{M}}$ be the corresponding communication load under this assignment $\mathcal{M}$. Then, we provide a lower bound on $L_{\mathcal{M}}$ for any given file assignment in the following lemma.

Lemma 7.1. *Consider a specific file assignment $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_\Lambda)$. Let $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ be a permutation of the set $[\Lambda]$ and define*

$$\mathcal{D}_i \coloneqq (D_{\mathcal{U}^i} : \mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}, |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i) \tag{7.75}$$
$$\mathcal{C}_i \coloneqq (C_{\mathcal{U}^i} : \mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}, |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i) \tag{7.76}$$
$$\mathcal{Y}_{i-1} \coloneqq (Y_{\mathcal{U}^j} : j \in [\alpha : i-1], \mathcal{U}^j \subseteq \{c_1, \ldots, c_j\}, |\mathcal{U}^j| = \alpha, c_j \in \mathcal{U}^j) \tag{7.77}$$

for each $i \in [\alpha : \Lambda]$. Then, the communication load is lower bounded by

$$L_{\mathcal{M}} \geq \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}). \tag{7.78}$$

Proof. The proof is described in Appendix C.4. □

It is perhaps interesting to highlight that the lemma above manages to combine relatively divergent ideas from [65, Lemma 2] and Lemma 6.1. On one hand, the proof of Lemma 7.1 is based on the iterative argument from the proof of [65, Lemma 2], where the authors built a sequence of entropy-based bounds by iteratively picking computing nodes without ordering them according to some specific permutations. On the other hand, since in our case we wish to keep into account the multi-access nature of our MADC system, the proof of Lemma 7.1 adapts the entropy-based approach from [65] by iteratively selecting the reducers according to some properly chosen permutations. The purpose of selecting reducers according to some proper permutations is that of constructing a tighter sequence of entropy-based bounds. The properly chosen permutations are inspired by Lemma 6.1, which was used to successfully develop a tight converse bound for the multi-access coded caching problem with combinatorial topology.

Now, we proceed with the proof. Denote by $\tilde{a}^{\mathcal{T}}$ the number of files which are mapped exclusively by the mappers in $\mathcal{T}$ for some $\mathcal{T} \subseteq [\Lambda]$. As each reducer $\mathcal{U} \in [\Lambda]_\alpha$ does not have access to the intermediate values of all those files that are not mapped by the mappers in $\mathcal{U}$, the term $\tilde{a}^{\mathcal{T}}$ represents the number of files whose intermediate values are required by each reducer $\mathcal{U} \in [\Lambda]_\alpha$ that does not have access to the mappers in $\mathcal{T}$, i.e., each reducer $\mathcal{U} \in [\Lambda]_\alpha$ such that $\mathcal{U} \cap \mathcal{T} = 0$ or, equivalently, each reducer $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ such that $|\mathcal{U}| = \alpha$. Taking advantage of the independence of the intermediate values and recalling that each reducer computes η_2 disjoint output functions, from Lemma 7.1 and for a given permutation $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ of the set $[\Lambda]$, we can further write

$$L_{\mathcal{M}} \geq \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{7.79}$$

$$= \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} \sum_{\mathcal{U}^i \subseteq \{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha, c_i \in \mathcal{U}^i} H(D_{\mathcal{U}^i} \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{7.80}$$

$$= \frac{1}{QNT} \sum_{i \in [\alpha:\Lambda]} \sum_{\mathcal{U}^i \subseteq \{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha, c_i \in \mathcal{U}^i} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1,\ldots,c_i\}} \tilde{a}^{\mathcal{T}} \eta_2 T \tag{7.81}$$

$$= \frac{1}{KN} \sum_{i \in [\alpha:\Lambda]} \sum_{\mathcal{U}^i \subseteq \{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha, c_i \in \mathcal{U}^i} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1,\ldots,c_i\}} \tilde{a}^{\mathcal{T}}. \tag{7.82}$$

If we build a bound as the one in Lemma 7.1 for each permutation of the set $[\Lambda]$ and we sum up all these bounds together, we obtain the expression

$$L_{\mathcal{M}} \geq \frac{1}{KN\Lambda!} \sum_{\boldsymbol{c} \in S_\Lambda} \sum_{i \in [\alpha:\Lambda]} \sum_{\mathcal{U}^i \subseteq \{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha, c_i \in \mathcal{U}^i} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1,\ldots,c_i\}} \tilde{a}^{\mathcal{T}} \tag{7.83}$$

where we recall that S_Λ represents the group of all permutations of $[\Lambda]$. Our goal now is to simplify this expression and we start doing so by counting how many times each term $\tilde{a}^{\mathcal{T}}$ appears in the RHS of (7.83) for any fixed $\mathcal{T} \subseteq [\Lambda]$ with $|\mathcal{T}| = j$ and $j \in [\Lambda]$.

First, we focus on some reducer $\mathcal{U} \subseteq ([\Lambda] \setminus \mathcal{T})$ with $|\mathcal{U}| = \alpha$. We can see that $\tilde{a}^{\mathcal{T}}$ appears in the RHS of (7.83) for all those permutations in S_Λ for which $\mathcal{U} = \mathcal{U}^i$ for some $i \in [\alpha : \Lambda]$ such that $\mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}$ with $|\mathcal{U}^i| = \alpha$ and $c_i \in \mathcal{U}^i$, and such that $\mathcal{T} \subseteq ([\Lambda] \setminus \{c_1, \ldots, c_i\})$. Denoting by $\mathcal{P}_{\mathcal{U},\mathcal{T}}$ the set of such permutations, we can see that

$$|\mathcal{P}_{\mathcal{U},\mathcal{T}}| = \alpha! j! (\Lambda - \alpha - j)! \binom{\Lambda}{\alpha + j}. \tag{7.84}$$

The same reasoning applies to any reducer $\mathcal{U} \in [\Lambda]_\alpha$ for which $\mathcal{U} \cap \mathcal{T} = \emptyset$. As a consequence, the term $\tilde{a}^{\mathcal{T}}$ appears in the RHS of (7.83) a total of

$$\sum_{\mathcal{U} \in [\Lambda]_\alpha : \mathcal{U} \cap \mathcal{T} = \emptyset} |\mathcal{P}_{\mathcal{U},\mathcal{T}}| = \binom{\Lambda - j}{\alpha} \alpha! j! (\Lambda - \alpha - j)! \binom{\Lambda}{\alpha + j} \tag{7.85}$$

times. The same rationale holds for any $\tilde{a}^{\mathcal{T}}$ where $\mathcal{T} \subseteq [\Lambda]$ and $|\mathcal{T}| = j$ with $j \in [\Lambda]$. Consequently, we can rewrite the expression in (7.83) as

$$\begin{aligned}
L_{\mathcal{M}} &\geq \frac{1}{KN\Lambda!} \sum_{\boldsymbol{c} \in S_\Lambda} \sum_{i \in [\alpha:\Lambda]} \sum_{\mathcal{U}^i \subseteq \{c_1, \ldots, c_i\} : |\mathcal{U}^i| = \alpha, c_i \in \mathcal{U}^i} \sum_{\mathcal{T} \subseteq [\Lambda] \setminus \{c_1, \ldots, c_i\}} \tilde{a}^{\mathcal{T}} && (7.86) \\
&= \frac{1}{KN\Lambda!} \sum_{j \in [\Lambda]} \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \binom{\Lambda - j}{\alpha} \alpha! j! (\Lambda - \alpha - j)! \binom{\Lambda}{\alpha + j} \tilde{a}^{\mathcal{T}} && (7.87) \\
&= \frac{1}{KN} \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{j}} \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \tilde{a}^{\mathcal{T}} && (7.88) \\
&= \frac{1}{K} \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{j}} \frac{\tilde{a}^j_{\mathcal{M}}}{N} && (7.89)
\end{aligned}$$

where $\tilde{a}^j_{\mathcal{M}} \coloneqq \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \tilde{a}^{\mathcal{T}}$ is defined as the total number of files which are mapped by exactly j map nodes under this particular file assignment $\mathcal{M}$.

For any given file assignment $\mathcal{M}$ and for any given computation load $r \in [K]$, the fact that $|\mathcal{M}_1| + \cdots + |\mathcal{M}_\Lambda| \leq rN$ also implies that $\tilde{a}^j_{\mathcal{M}} \geq 0$ for each $j \in [\Lambda]$, as well as implies that $\sum_{j \in [\Lambda]} \tilde{a}^j_{\mathcal{M}} = N$ and that $\sum_{j \in [\Lambda]} j \tilde{a}^j_{\mathcal{M}} \leq rN$. Thus, we can further lower bound the above using Jensen's inequality and

the fact that $\binom{\Lambda}{\alpha+j}/\binom{\Lambda}{j}$ is convex and decreasing[9] in j. Hence, we can write

$$L_{\mathcal{M}} \geq \frac{1}{K} \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{j}} \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{7.90}$$

$$\geq \frac{1}{K} \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{r}} \tag{7.91}$$

$$= \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \tag{7.92}$$

where (7.91) holds due to the storage constraint $\sum_{j\in[\Lambda]} j\tilde{a}^j_{\mathcal{M}} \leq rN$.

7.6.2 Lower Bound Over All Possible File Assignments

To obtain the bound in Theorem 7.2, we are looking for the smallest $L_{\mathcal{M}}$ across all file assignments $\mathcal{M}$ such that $|\mathcal{M}_1| + \cdots + |\mathcal{M}_\Lambda| \leq rN$, that is we are looking for

$$L^\star \geq \inf_{\mathcal{M}:|\mathcal{M}_1|+\cdots+|\mathcal{M}_\Lambda|\leq rN} L_{\mathcal{M}}. \tag{7.93}$$

Given that (7.92) is independent of the file assignment $\mathcal{M}$ and lower bounds $L_{\mathcal{M}}$ for any $\mathcal{M}$ such that $|\mathcal{M}_1| + \cdots + |\mathcal{M}_\Lambda| \leq rN$, we can further write

$$L^\star \geq \inf_{\mathcal{M}:|\mathcal{M}_1|+\cdots+|\mathcal{M}_\Lambda|\leq rN} L_{\mathcal{M}} \tag{7.94}$$

$$\geq \inf_{\mathcal{M}:|\mathcal{M}_1|+\cdots+|\mathcal{M}_\Lambda|\leq rN} \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \tag{7.95}$$

$$= \frac{\binom{\Lambda}{\alpha+r}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{r}} \tag{7.96}$$

$$= L_{\mathrm{LB}}. \tag{7.97}$$

Notice that the bound L_{LB} can be extended to include also the non-integer values of r as described in [52]. This concludes the proof. □

[9]This was already proved in the proof of Lemma 6.2 by writing down each combinatorial coefficient in $\binom{\Lambda}{\alpha+j}/\binom{\Lambda}{j}$ as a finite product and using then the general Leibniz rule to show that its second derivative is non-negative.

7.7 Proof of Achievable Bound in Theorem 7.4

As we mentioned in the statement of Theorem 7.4, the coded scheme depends on the solution of the linear program in (7.13). Hence, the first step is to evaluate the optimal solution[10] $\tilde{\boldsymbol{a}}_\star = (\tilde{a}^1_\star, \dots, \tilde{a}^\Lambda_\star)$. Next, we partition the input database in Λ parts, where we denote by $\mathcal{L}_j$ the j-th part, which contains $|\mathcal{L}_j| = \tilde{a}^j_\star$ files for each $j \in [\Lambda]$. Then, each part $j \in [\Lambda]$ of the database is split in $\binom{\Lambda}{j}$ batches containing η_j files each for some $\eta_j \in \mathbb{N}$, so that $\tilde{a}^j_\star = \eta_j \binom{\Lambda}{j}$ for each $j \in [\Lambda]$. This implies

$$\{w_1, \dots, w_N\} = \bigcup_{j \in [\Lambda]} \mathcal{L}_j \tag{7.98}$$

$$= \bigcup_{j \in [\Lambda]} \bigcup_{\mathcal{T}_1 \subseteq [\Lambda] : |\mathcal{T}_1| = j} \mathcal{B}_{j,\mathcal{T}_1} \tag{7.99}$$

where we denote by $\mathcal{B}_{j,\mathcal{T}_1}$ the batch containing η_j files associated with the label $\mathcal{T}_1$. Then, mapper $\lambda \in [\Lambda]$ is assigned all batches $\mathcal{B}_{j,\mathcal{T}_1}$ having $\lambda \in \mathcal{T}_1$ for each $j \in [\Lambda]$, which implies

$$\mathcal{M}_\lambda = \{\mathcal{B}_{j,\mathcal{T}_1} : j \in [\Lambda], \mathcal{T}_1 \subseteq [\Lambda], |\mathcal{T}_1| = j, \lambda \in \mathcal{T}_1\}. \tag{7.100}$$

The computation load constraint is satisfied, since we have

$$\frac{\sum_{\lambda \in [\Lambda]} |\mathcal{M}_\lambda|}{N} = \frac{\Lambda \sum_{j \in [\Lambda]} \eta_j \binom{\Lambda - 1}{j-1}}{N} \tag{7.101}$$

$$= \frac{\sum_{j \in [\Lambda]} j \eta_j \binom{\Lambda}{j}}{N} \tag{7.102}$$

$$= \frac{\sum_{j \in [\Lambda]} j \tilde{a}^j_\star}{N} \leq r \tag{7.103}$$

where the last inequality holds under the constraint in (7.13d).

Our goal is to provide an achievable scheme for the max-link communication load. Recalling that we denote by L and J the communication load and the download cost, respectively, we will have

$$L^\star_{\text{max-link}} \leq L_{\text{max-link,UB}} = \max(L, D). \tag{7.104}$$

7.7.1 Communication Load

For what concerns the communication load, we can take advantage of the achievable scheme described in Section 7.5. Simply, the scheme in Section 7.5

[10] The linear program in (7.13) is not infeasible nor unbounded. Hence, it admits an optimal solution.

is applied Λ times, one time per partition $\mathcal{L}_j$ which is considered as an independent input database. If we denote by L_j the communication load when we focus on the part $\mathcal{L}_j$, we have that L_j is given by

$$L_j = \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)}\frac{\tilde{a}_\star^j}{N} \tag{7.105}$$

for each $j \in [\Lambda]$. Hence, the overall communication load L is given by

$$L = \sum_{j\in[\Lambda]} L_j = \sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)}\frac{\tilde{a}_\star^j}{N}. \tag{7.106}$$

7.7.2 Download Cost

We remind that the download cost is defined as

$$J = \max_{\lambda\in[\Lambda]} \max_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} \frac{R_\lambda^{\mathcal{U}}}{QNT} \tag{7.107}$$

where $R_\lambda^{\mathcal{U}}$ represents the number of bits which are sent from mapper λ to reducer $\mathcal{U}$. This quantity is minimized if the number of bits transmitted over each link connecting a mapper to a reducer is the same. This can be accomplished as follows.

Consider a reducer $\mathcal{U} \in [\Lambda]_\alpha$ and a mapper $\lambda \in \mathcal{U}$. According to the file assignment above, mapper λ computes the IVs in the set $\mathcal{V}_\lambda = \{v_{q,n} : q \in [Q], w_n \in \mathcal{M}_\lambda\}$. The set $\mathcal{V}_\lambda$ can equivalently be written as follows

$$\mathcal{V}_\lambda = \{\mathcal{V}_{\lambda,\mathcal{S}} : i \in [\alpha], \mathcal{S} \subseteq (\mathcal{U}\setminus\{\lambda\}), |\mathcal{S}| = i-1\} \tag{7.108}$$

where $\mathcal{V}_{\lambda,\mathcal{S}}$ is defined as

$$\mathcal{V}_{\lambda,\mathcal{S}} \coloneqq \left\{v_{q,n} : q \in [Q], w_n \in \mathcal{M}^{\lambda\cup\mathcal{S}}\right\} \tag{7.109}$$

and where $\mathcal{M}^{\lambda\cup\mathcal{S}} \coloneqq \bigcap_{s\in(\lambda\cup\mathcal{S})} \mathcal{M}_s$. This simply says that the set $\mathcal{V}_{\lambda,\mathcal{S}}$ contains the IVs which are mapped by mapper λ and the $(i-1)$ mappers in $\mathcal{S}$. Hence, if we evenly split $\mathcal{V}_{\lambda,\mathcal{S}}$ in i segments as follows

$$\mathcal{V}_{\lambda,\mathcal{S}} = (\mathcal{V}_{\lambda,\mathcal{S},s} : s \in (\lambda\cup\mathcal{S})) \tag{7.110}$$

we simply let mapper λ send $\mathcal{V}_{\lambda,\mathcal{S},\lambda}$. This implies

$$R_\lambda^{\mathcal{U}} = \sum_{i\in[\alpha]} \sum_{\mathcal{S}\subseteq(\mathcal{U}\setminus\{\lambda\}):|\mathcal{S}|=i-1} |\mathcal{V}_{\lambda,\mathcal{S},\lambda}| \tag{7.111}$$

$$= \sum_{i\in[\alpha]} \sum_{\mathcal{S}\subseteq(\mathcal{U}\setminus\{\lambda\}):|\mathcal{S}|=i-1} \frac{|\mathcal{V}_{\lambda,\mathcal{S}}|}{i} \tag{7.112}$$

$$= \sum_{i\in[\alpha]} \sum_{\mathcal{S}\subseteq(\mathcal{U}\setminus\{\lambda\}):|\mathcal{S}|=i-1} \sum_{j\in[\Lambda]} \frac{\eta_j \binom{\Lambda-\alpha}{j-i} QT}{i} \tag{7.113}$$

$$= \sum_{j\in[\Lambda]} \sum_{i\in[\alpha]} \frac{\eta_j \binom{\Lambda-\alpha}{j-i} QT}{i} \binom{\alpha-1}{i-1} \tag{7.114}$$

for each $\lambda \in [\Lambda]$ and $\mathcal{U} \in [\Lambda]_\alpha$ with $\lambda \in \mathcal{U}$. Hence, we can further write

$$J = \max_{\lambda\in[\Lambda]} \max_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} \frac{R_\lambda^{\mathcal{U}}}{QNT} \tag{7.115}$$

$$= \frac{1}{QNT} \sum_{j\in[\Lambda]} \sum_{i\in[\alpha]} \frac{\eta_j \binom{\Lambda-\alpha}{j-i} QT}{i} \binom{\alpha-1}{i-1} \tag{7.116}$$

$$= \sum_{j\in[\Lambda]} \sum_{i\in[\alpha]} \frac{\binom{\Lambda-\alpha}{j-i}\binom{\alpha-1}{i-1}}{i\binom{\Lambda}{j}} \frac{\tilde{a}_\star^j}{N} \tag{7.117}$$

recalling that $\tilde{a}_\star^j = \eta_j \binom{\Lambda}{j}$ for each $j \in [\Lambda]$. Further, the following lemma holds.

Lemma 7.2. *For any non-negative integers Λ, α and j, we have*

$$\sum_{i\in[\alpha]} \frac{\binom{\Lambda-\alpha}{j-i}\binom{\alpha-1}{i-1}}{i\binom{\Lambda}{j}} = \frac{\binom{\Lambda}{j} - \binom{\Lambda-\alpha}{j}}{\alpha\binom{\Lambda}{j}}. \tag{7.118}$$

Proof. The proof is described in Appendix C.5. □

As a consequence, the download cost J is equivalently given by

$$J = \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{j} - \binom{\Lambda-\alpha}{j}}{\alpha\binom{\Lambda}{j}} \frac{\tilde{a}_\star^j}{N} \tag{7.119}$$

$$= \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_\star^j}{N}. \tag{7.120}$$

7.7.3 Max-Link Communication Load

Since we have now the expressions for both L and J, we can write explicitly the achievable max-link communication load as follows

$$L_{\text{max-link,UB}} = \max(L, D) \tag{7.121}$$

$$= \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} \frac{\tilde{a}_\star^j}{N}, \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_\star^j}{N} \right). \quad (7.122)$$

The expression above coincides with the achievable expression in Theorem 7.4. The proof is concluded. □

7.8 Proof of Converse Bound in Theorem 7.5

We quickly recall that $L_{\mathcal{M}}$ denotes the communication load under the file assignment $\mathcal{M} = (\mathcal{M}_1, \ldots, \mathcal{M}_\Lambda)$. Then, we let $J_{\mathcal{M}}$ and $L_{\text{max-link},\mathcal{M}} = \max\{L_{\mathcal{M}}, J_{\mathcal{M}}\}$ be the download cost and the max-link communication load, respectively, under file assignment $\mathcal{M}$.

7.8.1 Lower Bound for a Given File Assignment

From Section 7.6 we know that the inequality

$$L_{\mathcal{M}} \geq \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^j}{N} \quad (7.123)$$

holds. Thus, we can write

$$L_{\text{max-link},\mathcal{M}} = \max\{L_{\mathcal{M}}, J_{\mathcal{M}}\} \quad (7.124)$$

$$\geq \max\left\{ \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^j}{N}, J_{\mathcal{M}} \right\}. \quad (7.125)$$

In the following, we wish to develop a lower bound on $J_{\mathcal{M}}$. Starting from the definition of the download cost, we have

$$J_{\mathcal{M}} = \max_{\lambda\in[\Lambda]} \max_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} \frac{R_\lambda^{\mathcal{U}}}{QNT} \quad (7.126a)$$

$$\geq \max_{\lambda\in[\Lambda]} \frac{1}{\binom{\Lambda-1}{\alpha-1}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} R_\lambda^{\mathcal{U}} \quad (7.126b)$$

$$\geq \frac{1}{\Lambda\binom{\Lambda-1}{\alpha-1}QNT} \sum_{\lambda\in[\Lambda]} \sum_{\mathcal{U}\in[\Lambda]_\alpha:\lambda\in\mathcal{U}} R_\lambda^{\mathcal{U}} \quad (7.126c)$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha} \sum_{\lambda\in\mathcal{U}} R_\lambda^{\mathcal{U}} \quad (7.126d)$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha}QNT} \sum_{\mathcal{U}\in[\Lambda]_\alpha} R^{\mathcal{U}} \quad (7.126e)$$

where $R^{\mathcal{U}}$ is defined as

$$R^{\mathcal{U}} \coloneqq \sum_{\lambda \in \mathcal{U}} R_{\lambda}^{\mathcal{U}} \tag{7.127}$$

to represent the overall number of bits received by reducer $\mathcal{U} \in [\Lambda]_{\alpha}$. Now, since each reducer $\mathcal{U}$ is expected to receive all the IVs mapped by the mappers in $\mathcal{U}$, we have

$$R^{\mathcal{U}} \geq H(C_{\mathcal{U}}) \tag{7.128}$$

where we recall that $C_{\mathcal{U}} = \{V_{q,n} : q \in [Q], w_n \in \mathcal{M}_{\mathcal{U}}\}$ from Section 7.6. This means that we can further write

$$J_{\mathcal{M}} \geq \frac{1}{\alpha\binom{\Lambda}{\alpha} QNT} \sum_{\mathcal{U} \in [\Lambda]_{\alpha}} R^{\mathcal{U}} \tag{7.129a}$$

$$\geq \frac{1}{\alpha\binom{\Lambda}{\alpha} QNT} \sum_{\mathcal{U} \in [\Lambda]_{\alpha}} H(C_{\mathcal{U}}) \tag{7.129b}$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha} QNT} \sum_{\mathcal{U} \in [\Lambda]_{\alpha}} \sum_{\mathcal{T} \subseteq [\Lambda] : \mathcal{T} \cap \mathcal{U} \neq \emptyset} \tilde{a}^{\mathcal{T}} QT \tag{7.129c}$$

$$= \frac{1}{\alpha\binom{\Lambda}{\alpha} N} \sum_{j \in [\Lambda]} \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \left(\binom{\Lambda}{\alpha} - \binom{\Lambda - j}{\alpha} \right) \tilde{a}^{\mathcal{T}} \tag{7.129d}$$

$$= \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda - j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_{\mathcal{M}}^{j}}{N} \tag{7.129e}$$

recalling that $\tilde{a}_{\mathcal{M}}^{j} = \sum_{\mathcal{T} \subseteq [\Lambda] : |\mathcal{T}| = j} \tilde{a}^{\mathcal{T}}$. To conclude, for a given file assignment $\mathcal{M}$, the max-link communication load is lower bounded as

$$L_{\text{max-link},\mathcal{M}} \geq \max \left\{ \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha + j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^{j}}{N}, J_{\mathcal{M}} \right\} \tag{7.130}$$

$$\geq \max \left\{ \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha + j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} \frac{\tilde{a}_{\mathcal{M}}^{j}}{N}, \sum_{j \in [\Lambda]} \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda - j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}_{\mathcal{M}}^{j}}{N} \right\} \tag{7.131}$$

$$\geq \frac{1}{2} \sum_{j \in [\Lambda]} \left(\frac{\binom{\Lambda}{\alpha + j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda - j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}_{\mathcal{M}}^{j}}{N}. \tag{7.132}$$

7.8.2 Lower Bound Over All Possible File Assignments

Our aim is to develop a bound on the max-link communication load under any possible file assignment, namely, we are looking for the smallest $L_{\text{max-link},\mathcal{M}}$ across all file assignments $\mathcal{M}$ such that $|\mathcal{M}_1| + \cdots + |\mathcal{M}_{\Lambda}| \leq rN$ for a

given computation load $r \in [K]$. Since each file assignment $\mathcal{M}$ such that $|\mathcal{M}_1| + \cdots + |\mathcal{M}_\Lambda| \leq rN$ also implies that $\tilde{a}^j_{\mathcal{M}} \geq 0$ for each $j \in [\Lambda]$, as well as implies that $\sum_{j\in[\Lambda]} \tilde{a}^j_{\mathcal{M}} = N$ and that $\sum_{j\in[\Lambda]} j\tilde{a}^j_{\mathcal{M}} \leq rN$, the max-link load $L^\star_{\text{max-link}}$ is lower bounded by the solution to the following linear program

$$\min_{\tilde{\boldsymbol{a}}_{\mathcal{M}}} \quad \frac{1}{2}\sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^j_{\mathcal{M}}}{N} \tag{7.133a}$$

$$\text{subject to} \quad \tilde{a}^j_{\mathcal{M}} \geq 0, \quad \forall j \in [\Lambda] \tag{7.133b}$$

$$\sum_{j\in[\Lambda]} \frac{\tilde{a}^j_{\mathcal{M}}}{N} = 1 \tag{7.133c}$$

$$\sum_{j\in[\Lambda]} j\frac{\tilde{a}^j_{\mathcal{M}}}{N} \leq r \tag{7.133d}$$

where $\tilde{\boldsymbol{a}}_{\mathcal{M}} = (\tilde{a}^1_{\mathcal{M}}, \ldots, \tilde{a}^\Lambda_{\mathcal{M}})$ is the control variable. The proof is concluded. □

Part III

Conclusions and Appendices

Chapter 8

Conclusions and Future Directions

This chapter concludes the thesis. We will briefly summarize, chapter-by-chapter, the major contributions of the thesis, also presenting some possible future directions.

8.1 Exploring the Impact of Structure in Data

8.1.1 A Negative Result on Selfish Caching Policies

In Chapter 2, we investigated the effects that selfish caching can have on the optimal worst-case communication load in the coded caching framework. The proposed FDS structure seeks to capture the degree of intersection between the interests of the different users. While somewhat restrictive, the proposed structure was designed to bring to the fore and accentuate the adversarial relationship between coded caching and selfish caching, and by doing so, to allow us to provide insight on the nature of this adversarial relationship.

This insight is provided with the introduction of a new information-theoretic converse on the minimum worst-case communication load by means of index coding arguments. For the specific proposed broad FDS structure, the converse bound definitively resolves the question of whether selfish caching is beneficial or not. Indeed, the converse reveals that any non-zero load brought about by symmetrically selfish caching is always (with the exception of the extreme points of t) strictly worse than the optimal load guaranteed in the unselfish scenario. The rationale behind this is that, despite the sizeable increase of local caching gain brought about by the very targeted placement

of selfish caching, and despite a very restricted set of demands, the loss in multicasting opportunities is too severe. In fact, what the converse shows is that this damage is so prominent that — for any fixed (or decreasing) ratio $\delta = \alpha/K < 1$ — the coding gain does not scale with K, and is in fact bounded above by $1/(1-\delta)$. In other words, even if there is, for example, a 99 % symmetric intersection between the interests of the users (meaning, even if the users have interest in almost the same content), the coding gain will not scale as K increases. The above solidifies our conclusion that, for some powerful instances of FDS structures, selfish caching can be indeed very detrimental for the overall performance. While our FDS structure may be somewhat restrictive, the fact that this structure captures crucial elements of the selfish problem allows us to draw the conclusion that one must be cautious when considering selfish policies.

In light of the above, there are several interesting research directions. One first possibility is to explore the performance effect of FDS symmetry in selfish approaches, as indeed one of the biggest limitations of the FDS structure proposed in Chapter 2 seems its symmetric nature. Subsequently, it would be interesting to search for FDS structures that benefit well[1] from selfish coded schemes, as indeed, we recall, coded caching systems may need to operate under some selfish legacy constraints. This brings about the natural question of understanding when FDS structures can gain the most from selfish schemes. Clearly, a remaining challenge is to provide optimal schemes for any set of demands, either for the proposed or for another FDS structure.

8.1.2 Coded Caching With Tactical User Profiles

In Chapter 3, we focused on the problem of caching with hetereogeneous user profiles under the constraint of uncoded placement only. First, we proposed an extremely broad system model for the user profiles, where such model follows the combinatorial structure of the so-called *tactical configurations*, a

[1]Considering for example the work in [32], the authors proposed therein a more lenient FDS structure, where each file is of interest to either groups of users or all users in the system. First of all, such setting represents an example of how the degree of separation among the users' interests is not concentrated in one single α value, since the model in [32] considers somehow two values α_1 and α_2, where $\alpha_1 = K$ relates to the part of library which is of interest to all users, while $\alpha_2 = 1$ relates to the subset of files which are of interest to disjoint (groups of) users. Second, for the FDS structure in [32] the authors proposed a selfish coded scheme (there referred to as Scheme 2) which can outperform the standard MAN scheme for some non-trivial memory points under some system parameters. Similarly, selfish policies were shown to be quite effective also for some instances of the system model in [31]. Hence, there exist FDS structures which can benefit from selfish policies when coded caching is adopted.

symmetric and balanced block design in combinatorial mathematics. Then, we characterized the fundamental limits of coded caching under uncoded prefetching for the proposed broad structure of user interests. The proposed achievable scheme is a simple application of the already existing MAN placement-and-delivery scheme up to an arbitrary memory value $\bar{t}$, whereas for $t \geq \bar{t}$ memory sharing is adopted. Instead, the information-theoretic converse bound is based on the genie-aided approach from [32], which we also adopted in the subsequent Chapter 4. The interesting outcome of the results presented in Chapter 3 is that, albeit the diverging interests of the users, the MAN coded scheme is still order optimal within a constant factor of 4. This result is significant not only because it holds for a really generic structure of user profiles, including the symmetric (K, α, F) FDS structure studied in Chapter 2, but also because it proves the effectiveness of the combinatorial MAN coded caching scheme under a well-defined system model for the user profiles.

There are several interesting future directions. Along the lines of what we suggested as future directions for Chapter 2, one first possibility is to study the performance effect of FDS symmetry in non-selfish approaches. Secondly, it would be interesting to understand what non-trivial FDS structures, if any, might lead to a significant performance improvement over the MAN coded caching scheme. Indeed, being the MAN scheme order optimal within a constant multiplicative factor of 4 for such a broad structure of user profiles poses a question about the effective coding gains that one can achieve if users have diverging interests. In other words, it would be interesting to understand how much one can further lower the worst-case communication load by taking advantage in general of user interests, if these are known in advance.

8.1.3 A Converse for Caching With Heterogeneous Preferences

In Chapter 4, we considered an addittional coded caching setting with heterogeneous user profiles. Under the system model originally proposed in [32], we constructed a novel information-theoretic converse on the worst-case communication load under uncoded prefetching. We developed the lower bound by taking advantage of the genie-aided approach introduced in [4]. Interestingly, the proposed converse bound, jointly with the Scheme 2 from [32], allows us to characterize the optimal worst-case load under uncoded prefetching within a constant multiplicative factor of 2. Although the converse in Theorem 4.1 holds under the constraint of uncoded placement, the result in Theorem 4.2, which provides a constant order optimality factor independent of all system

parameters, improves the previously known order optimality results in [34].

Possible extensions could include applying the converse techniques developed in Chapter 4 to other (maybe more complex) caching settings with heterogeneous user profiles, as well as establishing the exact fundamental limits of the setting considered in Chapter 4.

8.1.4 Coded Distributed Computing With Structured Support

In Chapter 5, we considered the problem of coded distributed computing with structured support. Briefly, we assumed that each computing node $k \in [K]$ has to compute a reduce function which, differently from the original CDC setting in [52], does not depend on the whole input database, rather just on a subset $\mathcal{N}_k$ of the input files. In particular, we studied the setting where the structure for the sets $\{\mathcal{N}_k : k \in [K]\}$ follows the combinatorial design of tactical configurations, already employed in Chapter 3 in the context of coded caching. For such setting, we then proposed an achievable coded scheme and a converse bound, where the two are shown to be within a constant multiplicative factor of 6.

Possible extensions could include the setting where the total number of reduce functions is equal to $Q > K$, and where each reduce function is computed by $s > 1$ computing nodes. Indeed, we believe that for such setting novel solutions would be required for what concerns the achievability, as well as we believe that the techniques employed to develop the converse bound should be modified to account for the parameters $s > 1$ and $Q > K$.

8.2 The Ramifications of Structure in Topology

8.2.1 Combinatorial Multi-Access Caching

In Chapter 6, we identified the exact fundamental limits of multi-access caching for important general topologies as well as have derived information-theoretic converses that bound the performance across ensembles of connectivities.

Our first contribution was to derive the fundamental limits of multi-access caching for the generalized combinatorial topology that stands out for the unprecedented coding gains that it allows. These gains — first recorded in [53] for a specific instance of this generalized topology — are proven in Chapter 6 to be optimal, and to hold not only for a much denser range of users, but also in the presence of a coexistence of users connected to different numbers of

caches. As a direct consequence of identifying the exact fundamental limits of the generalized combinatorial topology, we now know that a basic and fixed MAN cache placement can *optimally* handle any generalized combinatorial network regardless of having unknown numbers of users with unequal cache-connectivity capabilities.

Subsequently, we considered the optimal performance of different ensembles of connectivities, deriving a lower bound on the average performance for a large ensemble of interest as well as for the entire ensemble of all possible topologies. To the best of our knowledge, Chapter 6 is the first work that explores the fundamental limits of the average performance across ensembles of connectivities as well as the first work to consider the topology-agnostic multi-access setting. We hope this contribution applies toward better understanding the role of topology in defining the performance as well as the corresponding role of topology-awareness.

- *Practical Pertinence and Optimality of the Generalized Combinatorial Topology.* As the subpacketization constraint is one of the main bottlenecks limiting the actual use of coded caching in various real applications, the combinatorial topology is undoubtedly a promising expedient to overcome this bottleneck. Indeed, this multi-access approach allows to serve many users at a time while controlling the subpacketization parameter by carefully calibrating the number of caches Λ. This particular topology breaks the barrier of having coding gains that are close to $\Lambda\gamma + 1$, and it goes one step further by allowing the unprecedented gains that are now a polynomial power of the cache redundancy, to be achieved with a modest subpacketization, a reasonable amount of cache resources and a very modest connectivity investment. Indeed, the gains explode even when α is as small as $\alpha = 2$ and this increase — we re-emphasize — is without any increase in the caching resources $\Lambda\gamma$. Our generalized model extends even further the benefits of this topology, allowing for a denser range of users K and supporting the coexistence of users connected to different numbers of caches. As a side-product of identifying the fundamental limits of performance in this broad topology, our result shows that, when there are users connected to a different number of caches α with $\alpha \in [0 : \Lambda]$ in accordance to the combinatorial topology, then a MAN cache placement and a simple TDMA-like application of the scheme in [53] is enough to achieve the minimum possible load in the worst-case scenario. This implies that treating each α-setting separately is optimal and so there would not be any advantage in encoding across users connected to a different number of caches.

- *An Agnostic Perspective and the Search for the Best Topology.* In real scenarios the actual connectivity could change over time. In this context, under the assumption that the cache placement procedure is performed only once regardless of the connectivity, the results in Theorem 6.2 and in Theorem 6.3 are of interest not only because they lower bound the optimal average worst-case load in the agnostic scenario, but also because a careful comparison between such bounds and the optimal performance in Theorem 6.1 allows to draw the insightful conclusion that the generalized combinatorial topology is among the good connectivities under the standard MAN cache placement. This brings to the fore the open question of whether, under the assumption of a MAN placement, the combinatorial topology is indeed the best possible topology. This question is supported by the just-now mentioned specific optimization solution that leads to Theorem 6.2, as well as by the large gains that the MAN placement achieves over this topology, irrespective of the fact that different users may be connected to a different number of caches.

The results in Chapter 6 are part of a sequence of works that suggest the importance of the multi-access caching paradigm not only for its role in boosting the gains in coded caching, which had remained relatively low due to the severe subpacketization bottleneck, but also for its role in several distributed communication paradigms where communication complexity is traded off with computational complexity[2] as a function of how servers are connected to data sources. Finally, while most works on the MACC focused mainly on the cyclic wrap-around topology, our work further legitimizes the effort of investigating connectivities different from the originally proposed MACC model. The bounds on the average performance over the ensemble of connectivities leave open the possibility that there may exist another topology performing uniformly better than the powerful optimal performance of the generalized combinatorial topology identified in Theorem 6.1. Finding such a topology, if it exists, would be indeed an exciting proposition given the already large gains that are associated to the combinatorial topology.

8.2.2 Multi-Access Distributed Computing

In Chapter 7, we introduced multi-access distributed computing, a novel system model that generalizes the original CDC setting by considering mappers and reducers as distinct entities, and by considering each reducer to be connected to multiple mappers through a network topology. We focused on the MADC model with combinatorial topology, which implies Λ mappers

[2]For example, see [93] and other related works [94]–[96].

and $K = \binom{\Lambda}{\alpha}$ reducers, so that there is a reducer for any set of α mappers. Neglecting at first the download cost from mappers to reducers and so focusing only on the inter-reducer communication load, we proposed a novel coded scheme which, together with an information-theoretic converse, characterizes the optimal communication load within a constant multiplicative gap of 1.5. Subsequently, we jointly considered the setting which keeps into account the download cost and for such scenario we characterized the optimal max-link communication load within a multiplicative factor of 4. We point out that the proposed achievable shuffling scheme — which generalizes the original coded scheme in [52] (corresponding to the case $\alpha = 1$) — offers also unparalleled coding gains. As an outcome of this gain, we have the interesting occurrence that our scheme guarantees a smaller communication load when $\alpha > 1$, capitalizing on the multi-access nature of the MADC model, even though the number of reducers is increased.

Interesting future directions could include the study of the proposed MADC setting when mappers and reducers have heterogeneous computational resources. A careful study of other multi-access network topologies is also another challenging research direction. Reflecting a design freedom, the search for the best possible topology, for a given computation load, remains a very pertinent open problem in distributed computing.

Appendix A

Appendices to Chapter 2

A.1 Proof of Corollary 2.1.1

Recalling both the expression for $R_{\mathrm{MAN}}(t)$ in (1.11) and the expression for $R_{\mathrm{LB}}(t)$ in Theorem 2.1, we have that

$$\frac{R^{\star}_{\mathrm{u,s}}(t)}{R_{\mathrm{MAN}}(t)} \geq \frac{R_{\mathrm{LB}}(t)}{R_{\mathrm{MAN}}(t)} \tag{A.1}$$

$$= \frac{\binom{\alpha}{t+1} + (K-\alpha)\binom{\alpha-1}{t}}{\binom{\alpha}{t}} \frac{\binom{K}{t}}{\binom{K}{t+1}} \tag{A.2}$$

$$= \frac{\binom{\alpha}{t+1}}{\binom{\alpha}{t}} \left(1 + (K-\alpha)\frac{t+1}{\alpha}\right) \frac{\binom{K}{t}}{\binom{K}{t+1}} \tag{A.3}$$

$$= \frac{\alpha - t}{K - t} \left(\frac{K(1+t) - \alpha t}{\alpha}\right) \tag{A.4}$$

$$= 1 + \underbrace{\frac{t(K-\alpha)(\alpha - 1 - t)}{\alpha(K-t)}}_{\geq 0} \tag{A.5}$$

where the second term in the last expression is equal to 0 either when $\alpha \in [K-1]$ for $t \in \{0, \alpha - 1\}$, or when $\alpha = K$ and $f \geq K$ for any t. This concludes the proof. □

A.2 Proof of Corollary 2.1.2

We know that the optimal coding gain is upper bounded as

$$G^{\star} \leq \frac{t+1}{1 + t(K-\alpha)/K} = \bar{G}(t). \tag{A.6}$$

It can be easily verified that $\bar{G}''(t) < 0$ for $t \geq 0$, which means that $\bar{G}(t)$ is concave for positive values of t. Then, we can see that $\bar{G}(0) = 1$, whereas

$$\lim_{t\to\infty} \bar{G}(t) = \frac{K}{K-\alpha} > 1 \tag{A.7}$$

for any $\alpha \in [K-1]$. Consequently, $\bar{G}(t) < K/(K-\alpha)$, which means that $G^\star < 1/(1-\delta)$ for any $\delta = \alpha/K$. This concludes the proof. □

A.3 Proof of Lemma 2.2

Let $\pi \in S_K$ and consider the permutation vector $\boldsymbol{\pi}^K = (\pi(1), \ldots, \pi(K))$. Consider $k_1, k_2 \in [K]$ such that $k_1 \neq k_2$. Assume without loss of generality that $\pi^{-1}(k_1) < \pi^{-1}(k_2) \leq K$, in which case $\ell = \pi^{-1}(k_2) - \pi^{-1}(k_1)$. Denoting by $\mathcal{U}$ the set containing the K circular shifts of the vector $\boldsymbol{\pi}^K$, we see that there are ℓ vectors $\boldsymbol{u} \in \mathcal{U}$ such that k_2 appears before k_1 in $\boldsymbol{u}$. These cases correspond to the vectors $\boldsymbol{u} \in \mathcal{U}$ where we have in the first position of $\boldsymbol{u}$ either the element k_2, or any one of the $(\ell-1)$ elements between $\pi^{-1}(k_1)$ and $\pi^{-1}(k_2)$ in the vector $\boldsymbol{\pi}$. As a consequence, the total number of vectors $\boldsymbol{u} \in \mathcal{U}$ such that k_1 appears before k_2 is equal to $(K-\ell)$. This concludes the proof. □

A.4 Proof of Lemma 2.3

The convexity of $f(t')$ can be easily shown by verifying that the second derivative $f''(t')$ with respect to t' is strictly positive for $t' \geq 0$. Indeed, we have

$$f(t') = \frac{\binom{\alpha}{t'+1} + (K-\alpha)\binom{\alpha-1}{t'}}{\binom{\alpha}{t'}} \tag{A.8}$$

$$= \frac{\alpha - t'}{1+t'} + (K-\alpha)\left(1 - \frac{t'}{\alpha}\right) \tag{A.9}$$

$$f'(t') = -\frac{1+\alpha}{(1+t')^2} - \frac{(K-\alpha)}{\alpha} \tag{A.10}$$

$$f''(t') = \frac{2(1+\alpha)}{(1+t')^3} > 0 \tag{A.11}$$

where $f'(t')$ denotes the first derivative. Then, since $t' \in [0:\alpha]$, we can evaluate $f(0) = K$ and $f(\alpha) = 0$, showing that $f(0) > f(\alpha)$. Hence, since $f(t')$ is convex, it has to be also strictly decreasing for $t' \in [0:\alpha]$, otherwise the convexity property would be violated. This concludes the proof. □

A.5 Converse Proof of Proposition 2.1

While the achievable expression matches exactly the converse expression R_{LB}, this latter converse cannot be used to prove the optimality of $R^{\star}_{\alpha,\text{c}}$, because R_{LB} bounds the optimal *worst-case* communication load. As there is no a priori guarantee that the α-demands are part of the worst-case demands, we will here derive another bound that focuses on α-demands to prove that the achievable performance is indeed optimal.

Following the same line of reasoning as in Section 2.4.1, we apply again the index coding lower bound in Lemma 2.1, with the only difference being that now the cache placement is fixed. The corresponding index coding problem has $K' = K$ users and $N' = K\binom{\alpha-1}{t}$ messages, where $\binom{\alpha-1}{t}$ is the total number of subfiles desired by each user for a fixed value of $t \in [0:\alpha]$. The desired message set and the side information set are respectively given by

$$\mathcal{M}_k = \{W_{f_k,\mathcal{D}_k,\mathcal{T}} : \mathcal{T} \subseteq \mathcal{D}_k, |\mathcal{T}| = t, k \notin \mathcal{T}\} \tag{A.12}$$

$$\mathcal{A}_k = \{W_{i,\mathcal{S},\mathcal{T}} : i \in [f], \mathcal{S} \subseteq [K], |\mathcal{S}| = \alpha, \mathcal{T} \subseteq \mathcal{S}, |\mathcal{T}| = t, k \in \mathcal{T}\} \tag{A.13}$$

for all $k \in [K]$. In the corresponding side information graph, an edge exists from $W_{f_{k_1},\mathcal{D}_{k_1},\mathcal{T}_1}$ to $W_{f_{k_2},\mathcal{D}_{k_2},\mathcal{T}_2}$ if and only if $W_{f_{k_1},\mathcal{D}_{k_1},\mathcal{T}_1} \in \mathcal{A}_{k_2}$.

Since we are considering a converse bound on the optimal communication load under a specific cache placement and under a specific set of demands, it suffices to find a single α-demand such that $R^{\star}_{\alpha,\text{c}}(t) \geq \binom{\alpha}{t+1}/\binom{\alpha}{t} + (K-\alpha)\binom{\alpha-1}{t}/\binom{\alpha}{t}$. Toward this, consider the α-demand where $\mathcal{K}$ is a set of α users that request distinct files from the file class $\mathcal{K}$, and where the remaining users in $[K] \setminus \mathcal{K}$ request distinct files so that the set of vertices

$$\mathcal{J}_1 = \bigcup_{k \in ([K]\setminus\mathcal{K})} \bigcup_{\mathcal{T} \subseteq (\mathcal{D}_k \setminus \{k\}) : |\mathcal{T}| = t} \{W_{f_k,\mathcal{D}_k,\mathcal{T}}\} \tag{A.14}$$

is acyclic. Then, such set contains a total of $(K-\alpha)\binom{\alpha-1}{t}$ subfiles, which means that $|\mathcal{J}_1| = (K-\alpha)\binom{\alpha-1}{t}B/\binom{\alpha}{t}$. Indeed, we can see that there exist α-demands for which $\mathcal{J}_1$ is acyclic. For instance, if we assume $\mathcal{D}_k = \mathcal{S} \cup \{k\}$ for every $k \in ([K] \setminus \mathcal{K})$ for some $\mathcal{S} \subseteq \mathcal{K}$ such that $|\mathcal{S}| = \alpha - 1$, then the set $\mathcal{J}_1$ is acyclic.

Consider now the set of users in $\mathcal{K}$. Take any permutation of users $\boldsymbol{u} = (u_1, \ldots, u_\alpha)$ with $u_k \in \mathcal{K}$ for all $k \in [\alpha]$. Then, since $\mathcal{D}_k = \mathcal{K}$ for all $k \in \mathcal{K}$, the set of vertices

$$\mathcal{J}_2 = \bigcup_{k \in [\alpha]} \bigcup_{\mathcal{T} \subseteq (\mathcal{K} \setminus \{u_1,\ldots,u_k\}) : |\mathcal{T}| = t} \{W_{f_{u_k},\mathcal{K},\mathcal{T}}\} \tag{A.15}$$

is acyclic for any permutation $\boldsymbol{u}$ (see [3, Lemma 1]). It can be easily seen that such set contains a total of $\binom{\alpha-1}{t}+\binom{\alpha-2}{t}+\cdots+\binom{t}{t}=\binom{\alpha}{t+1}$ subfiles, so $|\mathcal{J}_2|=\binom{\alpha}{t+1}B/\binom{\alpha}{t}$.

Due to the fact that $\mathcal{D}_k=\mathcal{K}$ for all $k\in\mathcal{K}$, there is no edge connecting any vertex in $\mathcal{J}_2$ to any vertex in $\mathcal{J}_1$, and thus also the set $\mathcal{J}_1\cup\mathcal{J}_2$ is acyclic. At this point, applying Lemma 2.1 with respect to the acyclic set $\mathcal{J}_1\cup\mathcal{J}_2$, we get

$$\begin{aligned} BR^{\star}_{\alpha,\mathrm{c}} &\geq \sum_{k\in([K]\setminus\mathcal{K})}\sum_{\mathcal{T}\subseteq(\mathcal{D}_k\setminus\{k\}):|\mathcal{T}|=t}|W_{f_k,\mathcal{D}_k,\mathcal{T}}| \\ &\quad+\sum_{k\in[\alpha]}\sum_{\mathcal{T}\subseteq(\mathcal{K}\setminus\{u_1,\ldots,u_k\}):|\mathcal{T}|=t}\left|W_{f_{u_k},\mathcal{K},\mathcal{T}}\right| && \text{(A.16)} \\ &=|\mathcal{J}_1|+|\mathcal{J}_2| && \text{(A.17)} \\ &=(K-\alpha)\binom{\alpha-1}{t}\frac{B}{\binom{\alpha}{t}}+\binom{\alpha}{t+1}\frac{B}{\binom{\alpha}{t}} && \text{(A.18)} \end{aligned}$$

which means that $R^{\star}_{\alpha,\mathrm{c}}(t)\geq\binom{\alpha}{t+1}/\binom{\alpha}{t}+(K-\alpha)\binom{\alpha-1}{t}/\binom{\alpha}{t}$. This concludes the proof. □

Appendix B

Appendices to Chapter 6

B.1 Proof of Lemma 6.1

We show how the set described in Lemma 6.1 is guaranteed to be acyclic. Given any connectivity $b \in \mathcal{B}$, let us start by considering the set

$$\bigcup_{k\in[K_{\emptyset,b}]} \bigcup_{\mathcal{T}\subseteq[\Lambda]} \left\{W_{d_{\emptyset_k},\mathcal{T}}\right\}. \tag{B.1}$$

Since each user $\emptyset_k$ with $k \in [K_{\emptyset,b}]$ is not connected to any cache, we can conclude that there is no connection among the vertices corresponding to the subfiles $W_{d_{\emptyset_k},\mathcal{T}}$ for each $k \in [K_{\emptyset,b}]$ and $\mathcal{T} \subseteq [\Lambda]$. This directly means that the set in (B.1) is acyclic. Now consider the set of vertices

$$\begin{aligned}&\bigcup_{\alpha\in[\Lambda]} \bigcup_{i\in[\alpha:\Lambda]} \bigcup_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha,\\ c_i\in\mathcal{U}^i}} \\ &\bigcup_{k\in[K_{\mathcal{U}^i,b}]} \bigcup_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left\{W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right\}.\end{aligned} \tag{B.2}$$

Following the same reasoning as in the proof of [3, Lemma 1], for a specific permutation of caches $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ we will say that the subfile $W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}$ belongs to the i-th *level*[1], which will mean that $\mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}$ with $|\mathcal{U}^i| = \alpha$ and $c_i \in \mathcal{U}^i$, and that $\mathcal{T}_i \subseteq ([\Lambda] \setminus \{c_1, \ldots, c_i\})$ with $i \in [\alpha : \Lambda]$. As one may see, no user in the i-th level has access to the subfiles in its level, since $\mathcal{U}^i \cap \mathcal{T}_i = \emptyset$ for each $\mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}$ with $|\mathcal{U}^i| = \alpha$ and $c_i \in \mathcal{U}^i$, and for each $\mathcal{T}_i \subseteq ([\Lambda] \setminus \{c_1, \ldots, c_i\})$. Moreover, no user in the i-th level has access to the

[1]The term *level* carries the same meaning as in the proof of [3, Lemma 1], and its impact here is described mathematically in compliance with our setting.

subfiles in higher levels, since $\mathcal{U}^i \cap \mathcal{T}_j = \emptyset$ for each $\mathcal{U}^i \subseteq \{c_1, \ldots, c_i\}$ with $|\mathcal{U}^i| = \alpha$ and $c_i \in \mathcal{U}^i$, and for each $\mathcal{T}_j \subseteq ([\Lambda] \setminus \{c_1, \ldots, c_j\})$ with $j \in [i+1 : \Lambda]$. Consequently, we can conclude that the set in (B.2) is acyclic. Moreover, since by definition no user $\emptyset_k$ with $k \in [K_{\emptyset,b}]$ is connected to any cache, there cannot be any cycle between the set in (B.1) and the set in (B.2), and so the union set

$$\begin{aligned} &\bigcup_{k\in[K_{\emptyset,b}]} \bigcup_{\mathcal{T}\subseteq[\Lambda]} \left\{W_{d_{\emptyset_k},\mathcal{T}}\right\} \cup \bigcup_{\alpha\in[\Lambda]} \bigcup_{i\in[\alpha:\Lambda]} \bigcup_{\substack{\mathcal{U}^i\subseteq\{c_1,\ldots,c_i\}:\\ |\mathcal{U}^i|=\alpha, c_i\in\mathcal{U}^i}} \\ &\bigcup_{k\in[K_{\mathcal{U}^i,b}]} \bigcup_{\mathcal{T}_i\subseteq([\Lambda]\setminus\{c_1,\ldots,c_i\})} \left\{W_{d_{\mathcal{U}^i_k},\mathcal{T}_i}\right\} \end{aligned} \tag{B.3}$$

is also acyclic. This concludes the proof. □

B.2 Proof of Lemma 6.2

Let us write $f(t') = \sum_{\alpha=0}^{\Lambda-t'} f_\alpha(t') = K_0 + \sum_{\alpha=1}^{\Lambda} f_\alpha(t')$, where we define

$$f_\alpha(t') \coloneqq K_\alpha \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \tag{B.4}$$

with $f_\alpha(t') = 0$ if $t' > \Lambda - \alpha$. Regarding convexity, we will prove each $f_\alpha(t')$ to be convex for $t' \in [0 : \Lambda]$, which will then tell us that $f(t')$ is also convex, since a non-negative linear combination of functions that are convex on the same interval is also convex. Notice that, since any linear function is both convex and concave, the constant term K_0 is convex with respect to t'.

When showing the convexity of $f_\alpha(t')$, we can focus on the range $t \leq \Lambda - \alpha + 1$ since we know that $f_\alpha(t') = 0$ for any $t \geq \Lambda - \alpha + 1$. Doing so will automatically allow us to conclude that this same function $f_\alpha(t')$ is also convex over the entire[2] interval $t' \in [0 : \Lambda]$. The convexity of $f_\alpha(t')$ can be easily shown by verifying that the second derivative $f_\alpha^{(2)}(t')$ with respect to t' is non-negative for $t' \in [0 : \Lambda - \alpha + 1]$. To see this, let us first note that

$$f_\alpha(t') = K_\alpha \frac{\binom{\Lambda}{t'+\alpha}}{\binom{\Lambda}{t'}} \tag{B.5}$$

[2]Indeed, if the function $f_\alpha(t')$ is convex in the interval $t' \in [0 : \Lambda - \alpha + 1]$ and $f_\alpha(t') = 0$ for $t' \in [\Lambda - \alpha + 1 : \Lambda]$ — which implies also convexity in the interval $t' \in [\Lambda - \alpha + 1 : \Lambda]$ since any linear function is both convex and concave — then $f_\alpha(t')$ is convex also over the entire interval $t' \in [0 : \Lambda]$, since it should be self-evident that the line segment between any two points t'_1, t'_2 with $t'_1 \in [0 : \Lambda - \alpha + 1]$ and $t'_2 \in [\Lambda - \alpha + 1 : \Lambda]$ lies above the graph between such two points.

$$= K_\alpha \prod_{i=0}^{\alpha-1} \frac{\Lambda - t' - i}{t' + \alpha - i} \tag{B.6}$$

$$= K_\alpha \prod_{i=0}^{\alpha-1} f_{\alpha,i}(t') \tag{B.7}$$

where $f_{\alpha,i}(t') \coloneqq (\Lambda - t - i)/(t + \alpha - i)$. Applying the general Leibniz rule allows us to rewrite the second derivative $f_\alpha^{(2)}(t')$ as

$$f_\alpha^{(2)}(t') = \left(K_\alpha \prod_{i=0}^{\alpha-1} f_{\alpha,i}(t') \right)^{(2)} \tag{B.8}$$

$$= K_\alpha \sum_{k_0+\cdots+k_{\alpha-1}=2} \binom{2}{k_0, \ldots, k_{\alpha-1}} \prod_{i=0}^{\alpha-1} f_{\alpha,i}^{(k_i)}(t') \tag{B.9}$$

where $k_\ell \in \mathbb{Z}_0^+$ for each $\ell \in [0 : \alpha - 1]$. Now we will prove that $f_\alpha^{(2)}(t') \geq 0$ by showing that $\prod_{i=0}^{\alpha-1} f_{\alpha,i}^{(k_i)}(t') \geq 0$ for any summand in (B.9), noticing that we have a summand for every α-weak composition of 2. Such weak compositions can be divided into two classes.

- The first class includes weak compositions where $k_m = 2$ for some $m \in [0 : \alpha - 1]$ and $k_n = 0$ for each $n \in ([0 : \alpha - 1] \setminus \{m\})$. In this case, the term $\prod_{i=0}^{\alpha-1} f_{\alpha,i}^{(k_i)}(t')$ is given by the product of $\alpha - 1$ functions $f_{\alpha,n}^{(0)}(t')$ for $n \in ([0 : \alpha - 1] \setminus \{m\})$ with 1 additional function $f_{\alpha,m}^{(2)}(t')$, where $f_{\alpha,n}^{(0)}(t')$ is simply the function $f_{\alpha,n}(t')$ and where $f_{\alpha,m}^{(2)}(t')$ is the second derivative of $f_{\alpha,m}(t')$. Hence, if $f_{\alpha,n}^{(0)}(t') \geq 0$ and $f_{\alpha,m}^{(2)}(t') > 0$, then $\prod_{i=0}^{\alpha-1} f_{\alpha,i}^{(k_i)}(t') \geq 0$.

- The second class includes weak compositions where $k_{m_1} = k_{m_2} = 1$ for some $m_1, m_2 \in [0 : \alpha - 1]$ with $m_1 \neq m_2$ and $k_n = 0$ for each $n \in ([0 : \alpha - 1] \setminus \{m_1, m_2\})$. In this case, the term $\prod_{i=0}^{\alpha-1} f_{\alpha,i}^{(k_i)}(t')$ is given by the product of $\alpha - 2$ functions $f_{\alpha,n}^{(0)}(t')$ for $n \in ([0 : \alpha - 1] \setminus \{m_1, m_2\})$ with 2 additional functions $f_{\alpha,m_1}^{(1)}(t')$ and $f_{\alpha,m_2}^{(1)}(t')$, where now $f_{\alpha,n}^{(1)}(t')$ is the first derivative of $f_{\alpha,n}(t')$. Hence, if $f_{\alpha,n}^{(0)}(t') \geq 0$, $f_{\alpha,m_1}^{(1)}(t') < 0$ and $f_{\alpha,m_2}^{(1)}(t') < 0$, then $\prod_{i=0}^{\alpha-1} f_{\alpha,i}^{(k_i)}(t') \geq 0$.

At this point, in order to prove that $\prod_{i=0}^{\alpha-1} f_{\alpha,i}^{(k_i)}(t') \geq 0$ for any α-weak composition of 2, we have to prove that the following inequalities

$$f_{\alpha,i}^{(0)}(t') \geq 0 \tag{B.10}$$

$$f_{\alpha,i}^{(1)}(t') < 0 \tag{B.11}$$

$$f^{(2)}_{\alpha,i}(t') > 0 \tag{B.12}$$

hold for each $i \in [0 : \alpha - 1]$, taking into account that

$$f^{(1)}_{\alpha,i}(t') = \frac{2i - \Lambda - \alpha}{(t' + \alpha - i)^2} \tag{B.13}$$

$$f^{(2)}_{\alpha,i}(t') = -\frac{2}{t' + \alpha - i} f^{(1)}_{\alpha,i}(t'). \tag{B.14}$$

Since $i \leq \alpha - 1$ and $\alpha \leq \Lambda - t' + 1$, it holds that $f^{(0)}_{\alpha,i}(t') \geq 0$. Moreover, it holds that

$$2i - \Lambda - \alpha \leq 2(\alpha - 1) - \Lambda - \alpha \tag{B.15}$$

$$= \alpha - 2 - \Lambda \tag{B.16}$$

$$\leq -2 \tag{B.17}$$

$$< 0. \tag{B.18}$$

Consequently, we can conclude that $f^{(1)}_{\alpha,i}(t') < 0$, since the numerator is always negative and the denominator is always positive. We can also conclude that $f^{(2)}_{\alpha,i}(t') > 0$, since $f^{(2)}_{\alpha,i}(t')$ is equal to the first derivative $f^{(1)}_{\alpha,i}(t') < 0$ multiplied by a strictly negative term $-2/(t' + \alpha - i)$. Thus, given that the second derivative $f^{(2)}_{\alpha}(t')$ equals the sum of non-negative terms, we can conclude that $f^{(2)}_{\alpha}(t') \geq 0$ and that $f_{\alpha}(t')$ is convex for each $\alpha \in [\Lambda]$ and $t' \in [0 : \Lambda - \alpha + 1]$. As a consequence, $f(t')$ is convex over the entire range of interest.

The fact that $f(t')$ is decreasing in $t' \in [0 : \Lambda]$ follows from the fact that $f(0) = K \geq f(\Lambda) = K_0$ and from the aforementioned convexity of $f(t')$. This concludes the proof. □

Appendix C

Appendices to Chapter 7

C.1 Proof of Corollary 7.1.1

We wish to prove that, for fixed computation load r, the achievable performance in Theorem 7.1 decreases for increasing α. To do so, it is enough to prove that

$$s_\alpha > s_{\alpha+1} \tag{C.1}$$

where s_α is defined as

$$s_\alpha \coloneqq \frac{\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\binom{r+\alpha}{r}-1\right)}. \tag{C.2}$$

Toward this, we can see that

$$s_{\alpha+1} = \frac{\binom{\Lambda-\alpha-1}{r}}{\binom{\Lambda}{r}\left(\binom{r+\alpha+1}{r}-1\right)} \tag{C.3}$$

$$= \frac{\frac{\Lambda-\alpha-r}{\Lambda-\alpha}\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\frac{r+\alpha+1}{\alpha+1}\binom{r+\alpha}{r}-1\right)} \tag{C.4}$$

$$= \frac{\left(1-\frac{r}{\Lambda-\alpha}\right)\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\left(1+\frac{r}{\alpha+1}\right)\binom{r+\alpha}{r}-1\right)}. \tag{C.5}$$

Since $r/(\Lambda-\alpha) > 0$ and $r/(\alpha+1) > 0$, it holds that $1 - r/(\Lambda-\alpha) < 1$ and $1 + r/(\alpha+1) > 1$. Consequently, we can write

$$s_{\alpha+1} = \frac{\left(1-\frac{r}{\Lambda-\alpha}\right)\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\left(1+\frac{r}{\alpha+1}\right)\binom{r+\alpha}{r}-1\right)} \tag{C.6}$$

$$< \frac{\binom{\Lambda-\alpha}{r}}{\binom{\Lambda}{r}\left(\binom{r+\alpha}{r}-1\right)} \tag{C.7}$$

$$= s_\alpha \tag{C.8}$$

showing in this way that $s_{\alpha+1} < s_\alpha$. This concludes the proof. □

C.2 Proof of Theorem 7.3

To prove the order optimality result in Theorem 7.3, we need to upper bound the ratio $L_{\mathrm{UB}}(r)/L^\star(r)$ for each $r \in [\Lambda - \alpha + 1]$. We start by noting that the following

$$\frac{L_{\mathrm{UB}}(r)}{L^\star(r)} \leq \frac{L_{\mathrm{UB}}(r)}{L_{\mathrm{LB}}(r)} \tag{C.9}$$

$$= \frac{\binom{\Lambda-\alpha}{r}}{\cancel{\binom{\Lambda}{r}}\left(\binom{r+\alpha}{r}-1\right)} \frac{\cancel{\binom{\Lambda}{r}}\binom{\Lambda}{\alpha}}{\binom{\Lambda}{r+\alpha}} \tag{C.10}$$

$$= \frac{\binom{\Lambda-\alpha}{r}}{\left(\binom{r+\alpha}{r}-1\right)} \frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{r+\alpha}} \tag{C.11}$$

$$= \frac{\binom{r+\alpha}{r}}{\binom{r+\alpha}{r}-1} =: b_r \tag{C.12}$$

holds. Further, we notice that b_r is decreasing in r, since

$$b_{r+1} = \frac{\binom{r+1+\alpha}{r+1}}{\binom{r+1+\alpha}{r+1}-1} \tag{C.13}$$

$$= \frac{1}{1-\frac{1}{\binom{r+1+\alpha}{r+1}}} \tag{C.14}$$

$$= \frac{1}{1-\frac{r+1}{r+1+\alpha}\frac{1}{\binom{r+\alpha}{r}}} \tag{C.15}$$

$$< \frac{1}{1-\frac{1}{\binom{r+\alpha}{r}}} \tag{C.16}$$

$$= b_r \tag{C.17}$$

for each $r \in \mathbb{N}^+$. Thus, considering that $r \in [\Lambda - \alpha + 1]$, we can further write

$$\frac{L_{\mathrm{UB}}(r)}{L^\star(r)} \leq \frac{\binom{r+\alpha}{r}}{\binom{r+\alpha}{r}-1} \tag{C.18}$$

$$\leq \frac{\alpha+1}{\alpha} \tag{C.19}$$

where the last term is upper bounded when α is set to its minimum value. Now, after neglecting the value $\alpha = 1$ — in which case the corresponding achievable performance in Theorem 7.1 was already proved to be exactly optimal in [52] — we focus on the case where $\alpha \in [2:\Lambda]$, which implies that

$$\frac{L_{\mathrm{UB}}(r)}{L^{\star}(r)} \leq \frac{\alpha+1}{\alpha} \tag{C.20}$$

$$\leq \frac{3}{2}. \tag{C.21}$$

The proof is concluded. □

C.3 Proof of Theorem 7.6

From Theorem 7.4 we know that $L^{\star}_{\text{max-link}}(r)$ is upper bounded as

$$L^{\star}_{\text{max-link}}(r) \leq L_{\text{max-link,UB}}(r) \tag{C.22}$$

$$= \max\left(\sum_{j\in[\Lambda]} \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} \frac{\tilde{a}^{j}_{\star}}{N}, \sum_{j\in[\Lambda]} \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \frac{\tilde{a}^{j}_{\star}}{N} \right) \tag{C.23}$$

$$\leq \sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^{j}_{\star}}{N} \tag{C.24}$$

$$= \sum_{j\in[\Lambda]} c_j \frac{\tilde{a}^{j}_{\star}}{N} \tag{C.25}$$

where the coefficient c_j is defined as

$$c_j \coloneqq \frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}. \tag{C.26}$$

At the same time, we know from Theorem 7.5 that $L^{\star}_{\text{max-link}}(r)$ is lower bounded as

$$L^{\star}_{\text{max-link}}(r) \geq L_{\text{max-link,LB}}(r) \tag{C.27}$$

$$= \frac{1}{2} \sum_{j\in[\Lambda]} \left(\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}} \right) \frac{\tilde{a}^{j}_{\star}}{N} \tag{C.28}$$

$$= \frac{1}{2} \sum_{j \in [\Lambda]} d_j \frac{\tilde{a}_\star^j}{N} \tag{C.29}$$

where the coefficient d_j is defined as

$$d_j \coloneqq \frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha} - \binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}. \tag{C.30}$$

Hence, from the ratio $L_{\text{max-link,UB}}(r)/L_{\text{max-link,LB}}(r)$ we can evaluate the gap to optimality. In particular, we have

$$\frac{L_{\text{max-link,UB}}(r)}{L_{\text{max-link,LB}}(r)} \leq 2\frac{\sum_{j\in[\Lambda]} c_j \tilde{a}_\star^j/N}{\sum_{j\in[\Lambda]} d_j \tilde{a}_\star^j/N} \tag{C.31}$$

$$= 2\frac{\sum_{j\in[\Lambda]:\tilde{a}_\star^j>0} c_j \tilde{a}_\star^j/N}{\sum_{j\in[\Lambda]:\tilde{a}_\star^j>0} d_j \tilde{a}_\star^j/N} \tag{C.32}$$

$$\leq 2 \max_{j\in[\Lambda]:\tilde{a}_\star^j>0} \frac{c_j \tilde{a}_\star^j/N}{d_j \tilde{a}_\star^j/N} \tag{C.33}$$

$$= 2 \max_{j\in[\Lambda]:\tilde{a}_\star^j>0} \frac{c_j}{d_j} \tag{C.34}$$

$$\leq 2 \max_{j\in[\Lambda]} \frac{c_j}{d_j} \tag{C.35}$$

$$= 2 \max\left(\max_{j\in[\Lambda-\alpha]} \frac{c_j}{d_j}, \max_{j\in[\Lambda-\alpha+1:\Lambda]} \frac{c_j}{d_j} \right). \tag{C.36}$$

Now, we can see that $c_j = d_j$ when $j > \Lambda - \alpha$. Else, when $j \in [\Lambda - \alpha]$, we have

$$\frac{c_j}{d_j} = \frac{\frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}}{\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}} + \frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}} \tag{C.37}$$

$$\leq \max\left(\frac{\frac{\binom{\Lambda-\alpha}{j}}{\binom{\Lambda}{j}\left(\binom{j+\alpha}{j}-1\right)}}{\frac{\binom{\Lambda}{\alpha+j}}{\binom{\Lambda}{\alpha}\binom{\Lambda}{j}}}, \frac{\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}}{\frac{\binom{\Lambda}{\alpha}-\binom{\Lambda-j}{\alpha}}{\alpha\binom{\Lambda}{\alpha}}} \right) \tag{C.38}$$

$$= \max\left(\frac{\binom{\Lambda-\alpha}{j}}{\left(\binom{j+\alpha}{j}-1\right)} \frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{\alpha+j}}, 1 \right). \tag{C.39}$$

Since we know from Appendix C.2 that

$$\frac{\binom{\Lambda-\alpha}{j}}{\left(\binom{j+\alpha}{j}-1\right)} \frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{\alpha+j}} \leq \frac{\alpha+1}{\alpha} \tag{C.40}$$

we can further write

$$\max_{j\in[\Lambda-\alpha]} \frac{c_j}{d_j} \leq \max_{j\in[\Lambda-\alpha]} \max\left(\frac{\binom{\Lambda-\alpha}{j}}{\left(\binom{j+\alpha}{j}-1\right)} \frac{\binom{\Lambda}{\alpha}}{\binom{\Lambda}{\alpha+j}}, 1 \right) \tag{C.41}$$

$$\leq \max\left(\frac{\alpha+1}{\alpha}, 1\right). \tag{C.42}$$

Hence, we can conclude that

$$\frac{L_{\text{max-link,UB}}(r)}{L_{\text{max-link,LB}}(r)} \leq 2\max\left(\max_{j\in[\Lambda-\alpha]} \frac{c_j}{d_j}, \max_{j\in[\Lambda-\alpha+1:\Lambda]} \frac{c_j}{d_j} \right) \tag{C.43}$$

$$\leq 2\max\left(\max\left(\frac{\alpha+1}{\alpha}, 1\right), 1\right) \tag{C.44}$$

$$\leq 2\max\left(\max\left(2, 1\right), 1\right) \tag{C.45}$$

$$= 4. \tag{C.46}$$

The proof is concluded. □

C.4 Proof of Lemma 7.1

Consider a permutation $\boldsymbol{c} = (c_1, \ldots, c_\Lambda)$ of the set $[\Lambda]$. We know that $H(D_\mathcal{U} \mid X_{[\Lambda]_\alpha}, C_\mathcal{U}) = 0$ holds for any valid shuffle scheme and for each $\mathcal{U} \in [\Lambda]_\alpha$. Given this, for $\mathcal{U}^\alpha = \{c_1, \ldots, c_\alpha\}$ we can write

$$H(X_{[\Lambda]_\alpha}) \geq H(X_{[\Lambda]_\alpha} \mid C_{\mathcal{U}^\alpha}) \tag{C.47}$$

$$= H(X_{[\Lambda]_\alpha}, D_{\mathcal{U}^\alpha} \mid C_{\mathcal{U}^\alpha}) - H(D_{\mathcal{U}^\alpha} \mid X_{[\Lambda]_\alpha}, C_{\mathcal{U}^\alpha}) \tag{C.48}$$

$$= H(X_{[\Lambda]_\alpha}, D_{\mathcal{U}^\alpha} \mid C_{\mathcal{U}^\alpha}) \tag{C.49}$$

$$= H(D_{\mathcal{U}^\alpha} \mid C_{\mathcal{U}^\alpha}) + H(X_{[\Lambda]_\alpha} \mid C_{\mathcal{U}^\alpha}, D_{\mathcal{U}^\alpha}) \tag{C.50}$$

$$= H(\mathcal{D}_{\mathcal{U}^\alpha} \mid \mathcal{C}_{\mathcal{U}^\alpha}) + H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_\alpha) \tag{C.51}$$

where (C.47) follows from the fact that conditioning does not increase entropy, and where (C.49) holds because of the decodability condition $H(D_\mathcal{U} \mid X_{[\Lambda]_\alpha}, C_\mathcal{U}) = 0$ for each $\mathcal{U} \in [\Lambda]_\alpha$. Similarly, for each $i \in [\alpha+1 : \Lambda]$ we can write

$$H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_{i-1}) \geq H(X_{[\Lambda]_\alpha} \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{C.52}$$

$$= H(X_{[\Lambda]_\alpha}, \mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) - H(\mathcal{D}_i \mid X_{[\Lambda]_\alpha}, \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{C.53}$$

$$= H(X_{[\Lambda]_\alpha}, \mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{C.54}$$

$$= H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) + H(X_{[\Lambda]_\alpha} \mid \mathcal{D}_i, \mathcal{C}_i, \mathcal{Y}_{i-1}) \tag{C.55}$$

$$= H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}) + H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_i) \tag{C.56}$$

where again (C.52) is true as conditioning does not increase entropy, and where (C.54) follows because

$$H(\mathcal{D}_i \mid X_{[\Lambda]_\alpha}, \mathcal{C}_i, \mathcal{Y}_{i-1}) \leq H(\mathcal{D}_i \mid X_{[\Lambda]_\alpha}, \mathcal{C}_i) \tag{C.57}$$

$$\leq \sum_{\mathcal{U}^i \subseteq \{c_1,\ldots,c_i\}:|\mathcal{U}^i|=\alpha, c_i \in \mathcal{U}^i} H(D_{\mathcal{U}^i} \mid X_{[\Lambda]_\alpha}, C_{\mathcal{U}^i}) \tag{C.58}$$

$$= 0 \tag{C.59}$$

due to the independence of intermediate values and the decodability condition. Considering that $H(X_{[\Lambda]_\alpha} \mid \mathcal{Y}_\Lambda) = 0$, we can use iteratively the above to obtain

$$H(X_{[\Lambda]_\alpha}) \geq \sum_{i \in [\alpha:\Lambda]} H(\mathcal{D}_i \mid \mathcal{C}_i, \mathcal{Y}_{i-1}). \tag{C.60}$$

Further, we notice that $L_{\mathcal{M}} \geq H(X_{[\Lambda]_\alpha})/QNT$. This concludes the proof. □

C.5 Proof of Lemma 7.2

First, we rewrite the equality in Lemma 7.2 as

$$\sum_{i \in [\alpha]} \frac{\binom{\Lambda-\alpha}{j-i}\binom{\alpha-1}{i-1}}{i\binom{\Lambda}{j}} = \frac{\binom{\Lambda}{j} - \binom{\Lambda-\alpha}{j}}{\alpha\binom{\Lambda}{j}} \tag{C.61}$$

$$\sum_{i \in [\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j} - \binom{\Lambda-\alpha}{j} \tag{C.62}$$

$$\sum_{i \in [0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j}. \tag{C.63}$$

Thus, proving the equality in Lemma 7.2 is equivalent to showing that the following equality

$$\sum_{i \in [0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j} \tag{C.64}$$

holds. From Vandermonde's identity, we know that

$$\sum_{i \in [0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j} \tag{C.65}$$

and so it suffices to show that

$$\sum_{i \in [0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \sum_{i \in [0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i}. \tag{C.66}$$

Consider first the case $j \leq \alpha$. This means that we can write

$$\sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} + \underbrace{\sum_{i\in[j+1:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i}}_{=0} \tag{C.67}$$

$$= \sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} \tag{C.68}$$

where $\sum_{i\in[j+1:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = 0$ since we have $\binom{\Lambda-\alpha}{j-i} = 0$ for $i \in [j+1:\alpha]$. Similarly, if we consider $j \geq \alpha$, we have

$$\sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} + \underbrace{\sum_{i\in[\alpha+1:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i}}_{=0} \tag{C.69}$$

$$= \sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} \tag{C.70}$$

where $\sum_{i\in[\alpha+1:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = 0$ since we have $\binom{\alpha}{i} = 0$ for $i \in [\alpha+1:j]$. Hence, for any value of $j \in [0:\Lambda]$, we can conclude that

$$\sum_{i\in[0:\alpha]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \sum_{i\in[0:j]} \binom{\Lambda-\alpha}{j-i}\binom{\alpha}{i} = \binom{\Lambda}{j}. \tag{C.71}$$

The proof is concluded. □

Bibliography

[1] Cisco, "Cisco annual internet report (2018–2023)," White Paper, 2020.

[2] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2856–2867, May 2014.

[3] K. Wan, D. Tuninetti, and P. Piantanida, "An index coding approach to caching with uncoded cache placement," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1318–1332, Mar. 2020.

[4] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "The exact rate-memory tradeoff for caching with uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 64, no. 2, pp. 1281–1296, Feb. 2018.

[5] S. P. Shariatpanahi, S. A. Motahari, and B. H. Khalaj, "Multi-server coded caching," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 7253–7271, Dec. 2016.

[6] S. P. Shariatpanahi, G. Caire, and B. H. Khalaj, "Physical-layer schemes for wireless coded caching," *IEEE Trans. Inf. Theory*, vol. 65, no. 5, pp. 2792–2807, May 2019.

[7] Y. Cao and M. Tao, "Degrees of freedom of cache-aided wireless cellular networks," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2777–2792, May 2020.

[8] A. Tölli, S. P. Shariatpanahi, J. Kaleva, and B. H. Khalaj, "Multi-antenna interference management for coded caching," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2091–2106, Mar. 2020.

[9] S. P. Shariatpanahi and B. H. Khalaj, *On multi-server coded caching in the low memory regime*, arXiv: 1803.07655 [cs.IT], Mar. 2018.

[10] J. Zhang and P. Elia, "Fundamental limits of cache-aided wireless BC: Interplay of coded-caching and CSIT feedback," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3142–3160, May 2017.

[11] E. Lampiris, J. Zhang, and P. Elia, "Cache-aided cooperation with no CSIT," in *2017 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2960–2964.

[12] M. M. Amiri and D. Gündüz, "Cache-aided content delivery over erasure broadcast channels," *IEEE Trans. Commun.*, vol. 66, no. 1, pp. 370–381, Jan. 2018.

[13] S. Mohajer and I. Bergel, "MISO cache-aided communication with reduced subpacketization," in *ICC 2020 - 2020 IEEE Inf. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[14] I. Bergel and S. Mohajer, "Cache-aided communications with multiple antennas at finite SNR," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 8, pp. 1682–1691, Aug. 2018.

[15] N. Naderializadeh, M. A. Maddah-Ali, and A. S. Avestimehr, "Fundamental limits of cache-aided interference management," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3092–3107, May 2017.

[16] F. Engelmann and P. Elia, "A content-delivery protocol, exploiting the privacy benefits of coded caching," in *2017 15th Int. Symp. Modeling Optim. Mobile, Ad Hoc, Wireless Netw. (WiOpt)*, May 2017, pp. 1–6.

[17] Q. Yan and D. Tuninetti, "Fundamental limits of caching for demand privacy against colluding users," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 192–207, Mar. 2021.

[18] K. Wan and G. Caire, "On coded caching with private demands," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 358–372, Jan. 2021.

[19] Q. Yan, M. Cheng, X. Tang, and Q. Chen, "On the placement delivery array design for centralized coded caching scheme," *IEEE Trans. Inf. Theory*, vol. 63, no. 9, pp. 5821–5833, Sep. 2017.

[20] L. Tang and A. Ramamoorthy, "Coded caching schemes with reduced subpacketization from linear block codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3099–3120, Apr. 2018.

[21] P. Krishnan, "Coded caching via line graphs of bipartite graphs," in *2018 IEEE Inf. Theory Workshop (ITW)*, Nov. 2018, pp. 1–5.

[22] C. Shangguan, Y. Zhang, and G. Ge, "Centralized coded caching schemes: A hypergraph theoretical approach," *IEEE Trans. Inf. Theory*, vol. 64, no. 8, pp. 5755–5766, Aug. 2018.

[23] E. Lampiris and P. Elia, "Adding transmitters dramatically boosts coded-caching gains for finite file sizes," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1176–1188, Jun. 2018.

[24] H. H. S. Chittoor, P. Krishnan, K. V. S. Sree, and B. Mamillapalli, "Subexponential and linear subpacketization coded caching via projective geometry," *IEEE Trans. Inf. Theory*, vol. 67, no. 9, pp. 6193–6222, Sep. 2021.

[25] M. Cheng, J. Li, X. Tang, and R. Wei, "Linear coded caching scheme for centralized networks," *IEEE Trans. Inf. Theory*, vol. 67, no. 3, pp. 1732–1742, Mar. 2021.

[26] U. Niesen and M. A. Maddah-Ali, "Coded caching with nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 2, pp. 1146–1158, Feb. 2017.

[27] J. Zhang, X. Lin, and X. Wang, "Coded caching under arbitrary popularity distributions," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 349–366, Jan. 2018.

[28] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3923–3949, Jun. 2017.

[29] J. Hachem, N. Karamchandani, and S. Diggavi, "Coded caching for multi-level popularity and access," *IEEE Trans. Inf. Theory*, vol. 63, no. 5, pp. 3108–3141, May 2017.

[30] S. A. Saberali, L. Lampe, and I. F. Blake, "Full characterization of optimal uncoded placement for the structured clique cover delivery of nonuniform demands," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 633–648, Jan. 2020.

[31] K. Wan, M. Cheng, M. Kobayashi, and G. Caire, "On the optimal memory-load tradeoff of coded caching for location-based content," *IEEE Trans. Commun.*, vol. 70, no. 5, pp. 3047–3062, Mar. 2022.

[32] S. Wang and B. Peleato, "Coded caching with heterogeneous user profiles," in *2019 IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 2619–2623.

[33] C. Zhang and B. Peleato, "On the average rate for coded caching with heterogeneous user profiles," in *ICC 2020 - 2020 IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[34] C. Zhang, S. Wang, V. Aggarwal, and B. Peleato, "Coded caching with heterogeneous user profiles," *IEEE Trans. Inf. Theory*, Jun. 2022, early access. doi: 10/gqrtb2.

[35] C.-H. Chang and C.-C. Wang, "Coded caching with heterogeneous file demand sets — the insufficiency of selfish coded caching," in *2019 IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2019, pp. 1–5.

[36] C.-H. Chang, C.-C. Wang, and B. Peleato, "On coded caching for two users with overlapping demand sets," in *ICC 2020 - 2020 IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–6.

[37] A. M. Ibrahim, A. A. Zewail, and A. Yener, "Device-to-device coded-caching with distinct cache sizes," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2748–2762, May 2020.

[38] E. Lampiris, A. Bazco-Nogueras, and P. Elia, "Resolving the feedback bottleneck of multi-antenna coded caching," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2331–2348, Apr. 2022.

[39] E. Parrinello, A. Ünsal, and P. Elia, "Fundamental limits of coded caching with multiple antennas, shared caches and uncoded prefetching," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2252–2268, Apr. 2020.

[40] E. Lampiris and P. Elia, "Full coded caching gains for cache-less users," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7635–7651, Dec. 2020.

[41] Y.-P. Wei and S. Ulukus, "Novel decentralized coded caching through coded prefetching," in *2017 IEEE Inf. Theory Workshop (ITW)*, Nov. 2017, pp. 1–5.

[42] H. Joudeh, E. Lampiris, P. Elia, and G. Caire, "Fundamental limits of wireless caching under mixed cacheable and uncacheable traffic," *IEEE Trans. Inf. Theory*, vol. 67, no. 7, pp. 4747–4767, Jul. 2021.

[43] G. J. O. Veld and M. Gastpar, "Caching (bivariate) gaussians," *IEEE Trans. Inf. Theory*, vol. 66, no. 10, pp. 6150–6168, Oct. 2020.

[44] A. M. Daniel and W. Yu, "Optimization of heterogeneous coded caching," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1893–1919, Mar. 2020.

[45] P. Hassanzadeh, A. M. Tulino, J. Llorca, and E. Erkip, "Rate-memory trade-off for caching and delivery of correlated sources," *IEEE Trans. Inf. Theory*, vol. 66, no. 4, pp. 2219–2251, Apr. 2020.

[46] Z. Zhang and M. Tao, "Deep learning for wireless coded caching with unknown and time-variant content popularity," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1152–1163, Feb. 2021.

[47] H. Zhao, A. Bazco-Nogueras, and P. Elia, *Vector coded caching multiplicatively boosts the throughput of realistic downlink systems*, arXiv: 2202.07047 [cs.IT], Feb. 2022.

[48] G. Zhang, Y. Li, and T. Lin, "Caching in information centric networking: A survey," *Computer Networks*, vol. 57, no. 16, pp. 3128–3141, Nov. 2013.

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

[50] D. Karamshuk, N. Sastry, M. Al-Bassam, A. Secker, and J. Chandaria, "Take-away TV: Recharging work commutes with predictive preloading of catch-up TV content," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 8, pp. 2091–2101, Aug. 2016.

[51] M.-C. Lee and A. F. Molisch, "Individual preference aware caching policy design in wireless D2D networks," *IEEE Trans. Wireless Commun.*, vol. 19, no. 8, pp. 5589–5604, Aug. 2020.

[52] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, "A fundamental tradeoff between computation and communication in distributed computing," *IEEE Trans. Inf. Theory*, vol. 64, no. 1, pp. 109–128, Jan. 2018.

[53] P. N. Muralidhar, D. Katyal, and B. S. Rajan, "Maddah-Ali-Niesen scheme for multi-access coded caching," in *2021 IEEE Inf. Theory Workshop (ITW)*, Oct. 2021, pp. 1–6.

[54] K. Wan, D. Tuninetti, M. Ji, and P. Piantanida, "A novel asymmetric coded placement in combination networks with end-user caches," in *2018 Inf. Theory App. Workshop (ITA)*, Feb. 2018, pp. 1–5.

[55] M. Ji, M. F. Wong, A. M. Tulino, *et al.*, "On the fundamental limits of caching in combination networks," in *2015 IEEE 16th Int. Workshop Signal Proc. Advances Wireless Commun. (SPAWC)*, Jun. 2015, pp. 695–699.

[56] C. K. Ngai and R. Yeung, "Network coding gain of combination networks," in *Inf. Theory Workshop*, Oct. 2004, pp. 283–287.

[57] M. Xiao, M. Medard, and T. Aulin, "A binary coding approach for combination networks and general erasure networks," in *2007 IEEE Int. Symp. Inf. Theory*, Jun. 2007, pp. 786–790.

[58] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[59] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol, "Index coding with side information," *IEEE Trans. Inf. Theory*, vol. 57, no. 3, pp. 1479–1494, Mar. 2011.

[60] F. Arbabjolfaei and Y.-H. Kim, "Fundamentals of index coding," *Foundations and Trends® in Commun. and Inf. Theory*, vol. 14, no. 3-4, pp. 163–346, 2018.

[61] C. Thapa, L. Ong, and S. J. Johnson, "Interlinked cycles for index coding: Generalizing cycles and cliques," *IEEE Trans. Inf. Theory*, vol. 63, no. 6, pp. 3692–3711, Jun. 2017.

[62] M. B. Vaddi and B. S. Rajan, "Optimal index codes for a new class of interlinked cycle structure," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 684–687, Apr. 2018.

[63] F. Arbabjolfaei, B. Bandemer, Y.-H. Kim, E. Şaşoğlu, and L. Wang, "On the capacity region for index coding," in *2013 IEEE Int. Symp. Inf. Theory*, Jul. 2013, pp. 962–966.

[64] R. J. Wilson, *Introduction to graph theory*. Longman, 2010, ISBN: 9780273728894.

[65] N. Woolsey, R.-R. Chen, and M. Ji, "A combinatorial design for cascaded coded distributed computing on general networks," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5686–5700, Sep. 2021.

[66] K. S. Reddy and N. Karamchandani, "Rate-memory trade-off for multi-access coded caching with uncoded placement," *IEEE Trans. Commun.*, vol. 68, no. 6, pp. 3261–3274, Jun. 2020.

[67] B. Serbetci, E. Parrinello, and P. Elia, "Multi-access coded caching: Gains beyond cache-redundancy," in *2019 IEEE Inf. Theory Workshop (ITW)*, Aug. 2019, pp. 1–5.

[68] K. K. K. Namboodiri and B. S. Rajan, "Multi-access coded caching with secure delivery," in *2021 IEEE Inf. Theory Workshop (ITW)*, Oct. 2021, pp. 1–6.

[69] D. Liang, K. Wan, M. Cheng, and G. Caire, *Multiaccess coded caching with private demands*, arXiv: 2105.06282 [cs.IT], May 2021.

[70] K. K. K. Namboodiri and B. S. Rajan, "Multi-access coded caching with demand privacy," in *2022 IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2022.

[71] K. S. Reddy and N. Karamchandani, "Structured index coding problem and multi-access coded caching," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 4, pp. 1266–1281, Dec. 2021.

[72] M. Zhang, K. Wan, M. Cheng, and G. Caire, *Coded caching for two-dimensional multi-access networks*, arXiv: 2201.11465 [cs.IT], Jan. 2022.

[73] S. Sasi and B. S. Rajan, "Multi-access coded caching scheme with linear sub-packetization using PDAs," *IEEE Trans. Commun.*, vol. 69, no. 12, pp. 7974–7985, Dec. 2021.

[74] M. Cheng, D. Liang, K. Wan, M. Zhang, and G. Caire, "A novel transformation approach of shared-link coded caching schemes for multiaccess networks," in *2021 IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2021, pp. 849–854.

[75] D. Katyal, P. N. Muralidhar, and B. S. Rajan, "Multi-access coded caching schemes from cross resolvable designs," *IEEE Trans. Commun.*, vol. 69, no. 5, pp. 2997–3010, May 2021.

[76] P. N. Muralidhar, D. Katyal, and B. S. Rajan, "Improved multi-access coded caching schemes from cross resolvable designs," in *2021 IEEE Inf. Theory Workshop (ITW)*, Oct. 2021, pp. 1–6.

[77] R. P. Stanley, *Enumerative Combinatorics*, Second. Cambridge: Cambridge University Press, 2011, vol. 1.

[78] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Computing*, 2010.

[79] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[80] U. Kumar and J. Kumar, "A comprehensive review of straggler handling algorithms for MapReduce framework," *Int. J. Grid Distrib. Comput.*, vol. 7, no. 4, pp. 139–148, Aug. 2014.

[81] Z. Zhang, L. Cherkasova, and B. T. Loo, "Performance modeling of MapReduce jobs in heterogeneous cloud environments," in *2013 IEEE 6th Int. Conf. Cloud Computing*, Jun. 2013, pp. 839–846.

[82] M. Ji, G. Caire, and A. F. Molisch, "Fundamental limits of caching in wireless D2D networks," *IEEE Trans. Inf. Theory*, vol. 62, no. 2, pp. 849–869, Feb. 2016.

[83] N. Woolsey, R.-R. Chen, and M. Ji, "A new combinatorial coded design for heterogeneous distributed computing," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 5672–5685, Sep. 2021.

[84] E. Parrinello, E. Lampiris, and P. Elia, "Coded distributed computing with node cooperation substantially increases speedup factors," in *2018 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1291–1295.

[85] K. Wan, M. Ji, and G. Caire, "Topological coded distributed computing," in *GLOBECOM 2020 - 2020 IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.

[86] S. R. Srinivasavaradhan, L. Song, and C. Fragouli, "Distributed computing trade-offs with random connectivity," in *2018 IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2018, pp. 1281–1285.

[87] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "A scalable framework for wireless distributed computing," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 2643–2654, Oct. 2017.

[88] Q. Yan, S. Yang, and M. Wigger, "A storage-computation-communication tradeoff for distributed computing," in *2018 15th Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2018, pp. 1–5.

[89] Q. Yan, S. Yang, and M. Wigger, "Storage, computation, and communication: A fundamental tradeoff in distributed computing," in *2018 IEEE Inf. Theory Workshop (ITW)*, Nov. 2018, pp. 1–5.

[90] J. S. Ng, W. Y. B. Lim, N. C. Luong, *et al.*, "A comprehensive survey on coded distributed computing: Fundamentals, challenges, and networking applications," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 3, pp. 1800–1837, 2021.

[91] Y. Shan, B. Wang, J. Yan, Y. Wang, N. Xu, and H. Yang, "FPMR: MapReduce framework on FPGA," in *Proc. 18th Annu. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays - FPGA '10*, Feb. 2010, pp. 93–102.

[92] Y.-M. Choi and H. K.-H. So, "Map-reduce processing of k-means algorithm with FPGA-accelerated computer cluster," in *2014 IEEE 25th Int. Conf. Application-Specific Systems, Architectures and Processors*, Jun. 2014.

[93] K. Wan, H. Sun, M. Ji, and G. Caire, "On the tradeoff between computation and communication costs for distributed linearly separable computation," *IEEE Trans. Commun.*, vol. 69, no. 11, pp. 7390–7405, Nov. 2021.

[94] M. Ye and E. Abbe, "Communication-computation efficient gradient coding," *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, pp. 5610–5619, Jul. 2018.

[95] S. Dutta, V. Cadambe, and P. Grover, ""Short-dot": Computing large linear transforms distributedly using coded short dot products," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6171–6193, Oct. 2019.

[96] A. Elmahdy and S. Mohajer, "On the fundamental limits of coded data shuffling for distributed machine learning," *IEEE Trans. Inf. Theory*, vol. 66, no. 5, pp. 3098–3131, May 2020.