2019

Coded caching: Information theoretic bounds and asynchronism

Hooshang Ghasemi
Iowa State University

Coded caching: Information theoretic bounds and asynchronism

by

Hooshang Ghasemi

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical Engineering (Communications and Signal Processing)

Program of Study Committee:
Aditya Ramamoorthy, Major Professor
Nicola Elia
Ahmed El-Sayed Kamal
Zhengdao Wang
Daniel Nordman

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

DEDICATION

I dedicate this thesis to my wife Sara for her endless love.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Aditya Ramamoorthy for his kind support throughout my graduation career. I learned many things from him.

I would also like to thank my committee members for their efforts and contributions to this work: Dr. Nicola Elia, Dr. Ahmed E. Kamal, Dr. Zhengdao Wang, Dr. Daniel Nordman, Dr. Ulrike Genschel, and Dr. Aditya Ramamoorthy.

The material in this work has appeared in part at the 2015, 2016 and the 2017 IEEE International Symposium on Information Theory and IEEE Transaction on Information Theory. This dissertation is supported in part by NSF grants CCF-1718470, CCF-1320416, and CCF-1149860.

ABSTRACT

Caching is often used in content delivery networks as a mechanism for reducing network traffic. Recently, the technique of coded caching was introduced whereby coding in the caches and coded transmission signals from the central server were considered. Prior results in this area demonstrate that carefully designing the placement of content in the caches and designing appropriate coded delivery signals from the server allow for a system where the delivery rates can be significantly smaller than conventional schemes.

However, matching upper and lower bounds on the transmission rate have not yet been obtained. In the first part of this dissertation we derive tighter lower bounds on the coded caching rate than were known previously. We demonstrate that this problem can equivalently be posed as a combinatorial problem of optimally labeling the leaves of a directed tree. Our proposed labeling algorithm allows for significantly improved lower bounds on the coded caching rate. Furthermore, we study certain structural properties of our algorithm that allow us to analytically quantify improvements on the rate lower bound for general values of the problem parameters. This allows us to obtain a multiplicative gap of at most four between the achievable rate and our lower bound.

The original formulation of the coded caching problem assumes that the file requests from the users are synchronized, i.e., they arrive at the server at the same time. Several subsequent contributions work under the same assumption. Furthermore, the majority of prior work does not consider a scenario where users have deadlines. In the second part of the dissertation we formulate the asynchronous coded caching problem where user requests arrive at different times. Furthermore, the users have specified deadlines. We propose a linear program for obtaining its optimal solution. However, the size of the LP (number of constraints and variables) grows rather quickly with the number of users and cache sizes. To deal with this problem, we explore a dual decomposition based

approach for solving the LP under consideration. We demonstrate that the dual function can be evaluated by equivalently solving a number of minimum cost network flow algorithms.

Moreover, we consider the asynchronous setting where the file requests are revealed to the server in an online fashion. We propose a novel online algorithm for this problem building on our prior work for the offline setting (where the server knows the request arrival times and deadlines in advance). Our simulation results demonstrate that our proposed online algorithm allows for a natural tradeoff between the feasibility of the schedule and the rate gains of coded caching.

CHAPTER 1. INTRODUCTION

Content distribution over the Internet is an important problem and is the core business of several enterprises such as YouTube, Netflix, Hulu etc. The operation of such large scale systems presents several challenges, including (but not limited to) storage of the data, ensuring reliable availability and efficient content delivery. One commonly used technique to facilitate delivery is content caching Wessels (2001). The main idea in *"conventional content caching"* is to store relatively popular content in local memory either on the desired device or in a device at the edge of the network such as an intermediate router. This local memory is referred to as the cache. Upon request, this cached content is used to serve the clients, thus reducing the number of bits transmitted from the server and thereby reducing overall network congestion. Note that even web browsers, routinely cache the content of popular websites on a local machine to speed up the loading of webpages.

The main idea of utilizing cache content for reducing data traffic in content delivery systems during peak hours can be understood from graph in Figure 1.1. The network has highest and lowest congestion at around 8 night and 4 morning respectively. In the conventional content caching the popular contents will be placed at the caches during low congestion hours (e.g. 4 morning). Therefore, the server is able to satisfy the requests during peak hours by transmitting fewer bits since some part of each request might already exist in cache content of the users.

1.1 Coded Caching

Historically, content caching algorithms have attempted to optimize the placement of content in the caches so that the average number of bits that are transmitted from the central server to the end users is minimized Meyerson et al. (2001); Korupolu et al. (1999); Borst et al. (2010); Tan and Massoulié (2013). This often requires some knowledge on the popularity of file requests Wolman

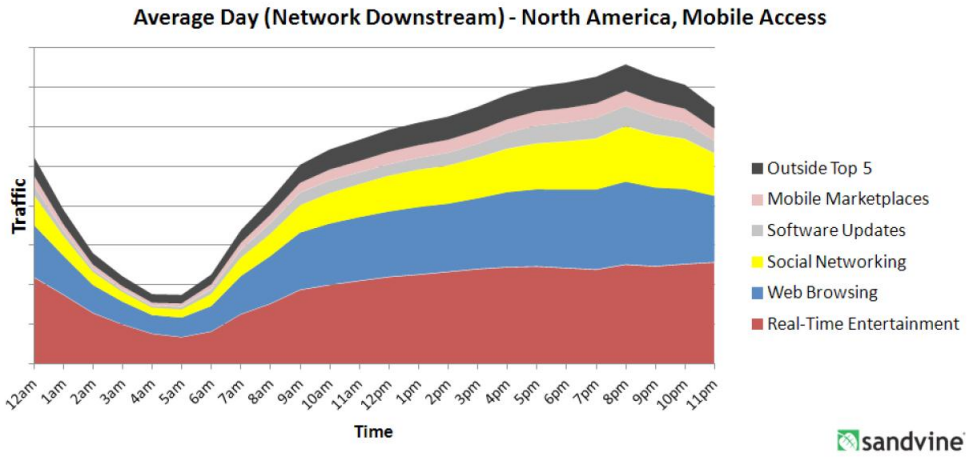


Figure 1.1: Average day traffic.

et al. (1999); Breslau et al. (1999); Applegate et al. (2010) made by the users. Moreover, the typical approach is to cache a certain fraction of the file and to obtain the remaining parts from the server when the need arises. Coding in the content of the cache and/or coding in the transmission from the server are typically not considered.

The work of Maddah-Ali and Niesen (2014) introduced the problem of coded caching, where there is a server with N files and K users each with a cache of size M. The users are connected to the server by a shared link (see Figure 1.2). In each time slot each user requests one of the N files. There are two distinct phases in coded caching.

- *Placement phase.* In this phase, the content of caches is populated. This phase should not depend on the actual user requests (which are assumed to be arbitrary). Typically, the placement phase can be executed in the *off-peak* hours where the amount of network traffic is low, e.g., around 4 a.m. in Figure 1.1.

- *Delivery phase.* In this phase, each of the K users request one of the N files. The server transmits a signal of rate R over the shared link that simultaneously serves to satisfy the demands of each of the users. This phase happens during busy hours, e.g., anytime between 10 a.m. till midnight at Figure 1.1.

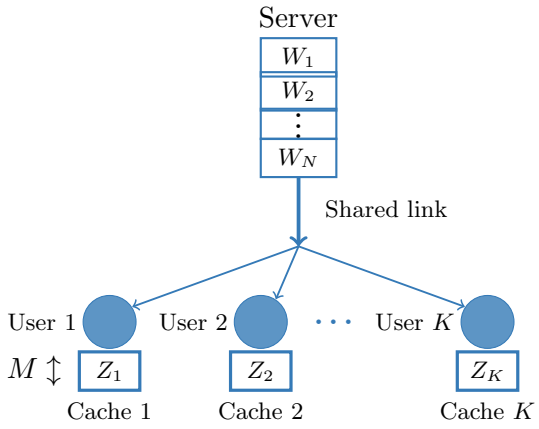


Figure 1.2: Block diagram of coded caching system.

The work of Maddah-Ali and Niesen (2014) demonstrates that a carefully designed placement scheme and a corresponding delivery scheme achieves a rate that is significantly lower than conventional caching.

To better understand the coded caching idea, we brought the following example from Maddah-Ali and Niesen (2014).

Example 1 *Consider the case $N = K = 2$, so that there are two files, say A, B, and two users each with cache memory of size $M = 1$ file.*

The caching scheme is as follows (see Figure 1.3). We split both files A and B into two subfiles of equal size, i.e., $A = (A_1, A_2)$ and $B = (B_1, B_2)$. In the placement phase, we set cache content of the user one $Z_1 = (A_1, B_1)$ and the user two $Z_2 = (A_2, B_2)$. In words, each user caches one exclusive part of each file. For the delivery phase, assume for example that user one requests file A and user two requests file B. Given that user one already has subfile A_1 of A, it only needs to obtain the missing subfile A_2, which is cached in the second user's memory Z_2. Similarly, user two only needs to obtain the missing subfile B_1, which is cached in the first user's memory Z_1. In other words, each user has one part of the file that the other user needs.

The server can in this case simply transmit $A_2 \bigoplus B_1$, where $\bigoplus$ denotes bitwise XOR. Since user one already has B_1, it can recover A_2 from $A_2 \bigoplus B_1$. Similarly, since user two already has

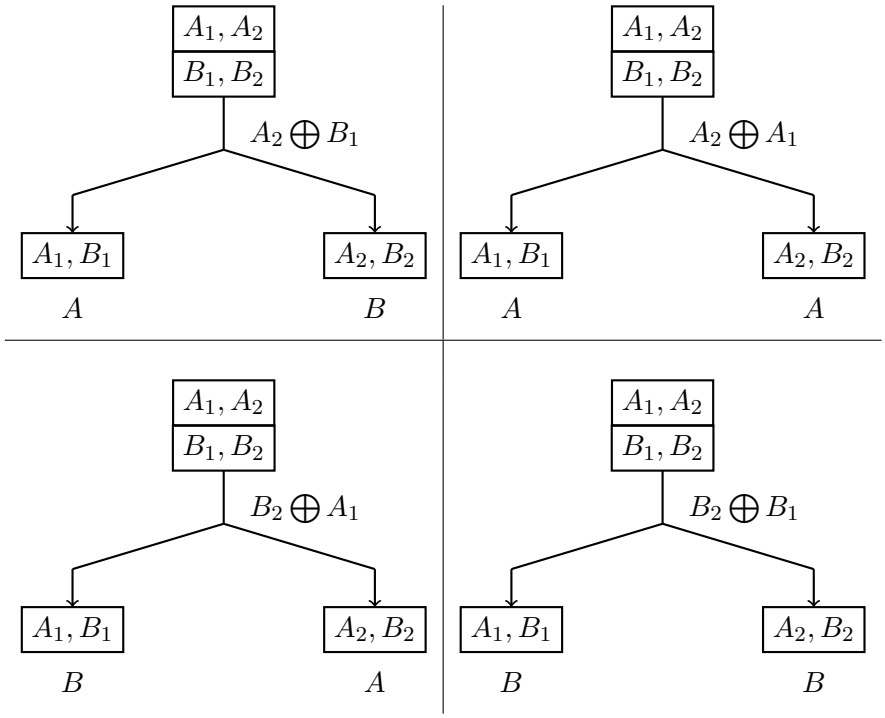


Figure 1.3: Coded Caching strategy in Maddah-Ali and Niesen (2014) for $N = 2$ files and $K = 2$ users with cache size $M = 1$ with all four possible user requests. Each file is split into two subfiles of size $1/2$, i.e., $A = (A_1, A_2)$ and $B = (B_1, B_2)$. The scheme achieves rate $R = 1/2$. Observe that, while the transmission from the server changes as a function of the user requests, the cache contents do not.

A_2, it can recover B_1 from $A_2 \bigoplus B_1$. Thus, the signal $A_2 \bigoplus B_1$ received over the shared link helps both users to effectively exchange the missing subfiles available in the cache of the other user. The signals sent over the shared link for all other requests are depicted in Figure 1.3. One can see that in all cases the signal is constructed using the same logic of exchanging the missing subfiles. It is worth pointing out that in each case the server sends a single coded multicast transmission to satisfy two (possibly different) user requests. Moreover, these coded multicasting opportunities are available simultaneously for all four possible user requests. This availability of simultaneous multicasting opportunities, enabled by careful content placement, is critical, since the placement phase has to be performed without knowledge of the actual demands in the delivery phase.

1.1.1 Lower Bounds on the Coded Caching

While coded caching promises very significant gains in transmission rates, at this point we do not have matching upper and lower bounds on the (R, M) pairs for a given N and K. In the first part of the dissertation our main contribution is in developing improved lower bounds on the required rate for the coded caching problem. We demonstrate that the computation of this lower bound can be posed as a combinatorial labeling problem on a directed tree. In particular, our method generates lower bounds on $\alpha R+\beta M$, where α and β are positive integers. We demonstrate that a careful analysis of the underlying combinatorial structure of the problem allows us to obtain significantly better lower bounds than those obtained in prior work Maddah-Ali and Niesen (2014); Sengupta et al. (2015b); Ajaykrishnan et al. (2015). In addition, our machinery allows us to show that the achievable rate of Maddah-Ali and Niesen (2014) is within a multiplicative factor of four of our proposed lower bound. Our contributions to lower bound on coded caching is presented in Chapter 2.

1.1.2 Asynchronous Coded Caching

While the coded caching scheme in Maddah-Ali and Niesen (2014) achieves a significant result, the original formulation of the coded caching problem assumes that the user requests are synchronized, i.e., all file requests from the users arrive at the server at the same time. From a practical perspective, it is important to consider the case when the requests of the users are not synchronized; we refer to this as the asynchronous coded caching problem.

In this case, a simple strategy would be to wait for the last request to arrive and then apply the scheme of Maddah-Ali and Niesen (2014). Such a strategy will be quite good in terms of the overall rate of transmission from the server. However, this may be quite bad for an end user's experience, e.g., the delay experienced by the users will essentially be dominated by the arrival time of the last request. This delay can not be tolerated for example when the end user is watching a video from the server. Our motivation to investigate this problem comes from the fact that most contents delivered over the internet are videos. From Figure 1.4 it can be seen that video traffic contains

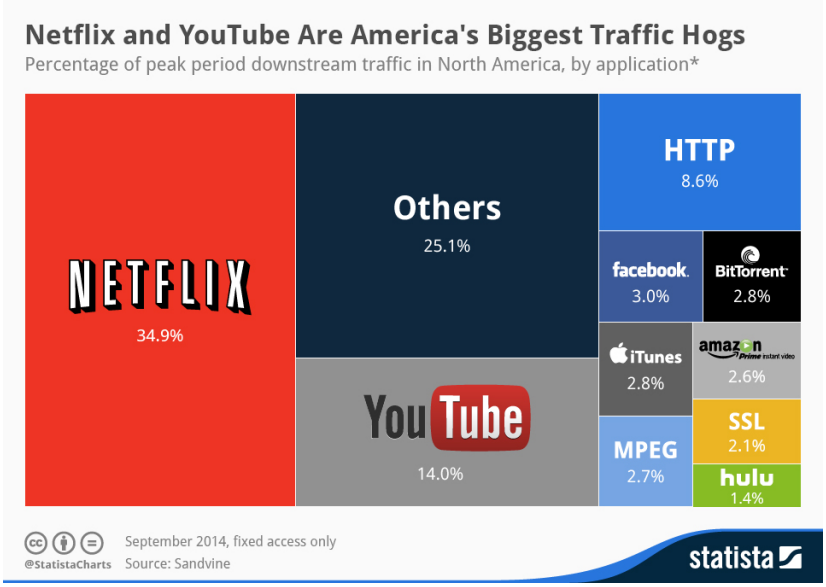


Figure 1.4: America's Biggest Traffic

more than half of the America's traffic. Therefore, it is important to study asynchronous coded caching.

In the second part of the dissertation we formulate and study the coded caching problem when the user requests arrive at different times. Moreover, each user has a specific deadline by which his/her demand needs to be satisfied. We examine both the offline and online versions of this problem. In the offline version, the server knows the arrival times and deadlines of all users before starting transmission. In the online case, the server is revealed information about the arrival times and deadlines as time progresses. In the offline scenario, where the server knows the arrival times and deadlines of each user in advance, we posed a linear programming (LP) problem which if feasible, allows the server to determine a schedule of transmissions, such that each user can be satisfied within its deadline. The size of the LP grows very quickly with the problem parameters and solving it is impractical for large scale instances. We demonstrate that we can instead work with the dual of an equivalent LP. The dual function can be evaluated by solving a set of minimum cost network flow problems. Minimum cost network flow problems have been the subject of much investigation in the optimization literature and large scale instances can be solved very quickly

Kovacs (2015). We present results that indicate that significant time savings are obtained by applying our approach. Moreover, our results indicate that the coded caching rate degrades quite gracefully in the presence of asynchronism. We also present a novel heuristic for the online version of the asynchronous coded caching problem. Our proposed algorithm is inspired by our LP formulation for the offline scenario. Roughly speaking, we solve a new offline-like LP every time a new user request comes to the server. Its solution is used to identify equations that simultaneously "benefit" multiple users. Here, the benefit to a given user takes into account the stringency of its deadline. Our simulation results demonstrate that the probability that our online algorithm is feasible is quite high and can be traded off for the rate gains of coded caching by varying a system-defined threshold.

CHAPTER 2. LOWER BOUNDS ON CODED CACHING

Content distribution over the Internet is an important problem and is the core business of several enterprises such as YouTube, Netflix, Hulu etc. The operation of such large scale systems presents several challenges, including (but not limited to) storage of the data, ensuring reliable availability and efficient content delivery. One commonly used technique to facilitate delivery is content caching Wessels (2001). The main idea in "conventional content caching" is to store relatively popular content in local memory either on the desired device or in a device at the edge of the network such as an intermediate router. This local memory is referred to as the cache. Upon request, this cached content is used to serve the clients, thus reducing the number of bits transmitted from the server and thereby reducing overall network congestion. Note that even web browsers, routinely cache the content of popular websites on a local machine to speed up the loading of webpages.

Historically, content caching algorithms have attempted to optimize the placement of content in the caches so that the average number of bits that are transmitted from the central server to the end users is minimized Meyerson et al. (2001); Korupolu et al. (1999); Borst et al. (2010); Tan and Massoulié (2013). This often requires some knowledge on the popularity of file requests Wolman et al. (1999); Breslau et al. (1999); Applegate et al. (2010) made by the users. Moreover, the typical approach is to cache a certain fraction of the file and to obtain the remaining parts from the server when the need arises. Coding in the content of the cache and/or coding in the transmission from the server are typically not considered.

The work of Maddah-Ali and Niesen (2014) introduced the problem of coded caching, where there is a server with N files and K users each with a cache of size M. The users are connected to the server by a shared link (see Figure 1.2). In each time slot each user requests one of the N files. There are two distinct phases in coded caching.

- *Placement phase.* In this phase, the content of caches is populated. This phase should not depend on the actual user requests (which are assumed to be arbitrary). Typically, the placement phase can be executed in the *off-peak* hours where the amount of network traffic is low.

- *Delivery phase.* In this phase, each of the K users request one of the N files. The server transmits a signal of rate R over the shared link that simultaneously serves to satisfy the demands of each of the users.

The work of Maddah-Ali and Niesen (2014) demonstrates that a carefully designed placement scheme and a corresponding delivery scheme achieves a rate that is significantly lower than conventional caching. While coded caching promises very significant gains in transmission rates, at this point we do not have matching upper and lower bounds on the (R, M) pairs for a given N and K.

In this chapter our main contribution is in developing improved lower bounds on the required rate for the coded caching problem. We demonstrate that the computation of this lower bound can be posed as a combinatorial labeling problem on a directed tree. In particular, our method generates lower bounds on $\alpha R + \beta M$, where α and β are positive integers. We demonstrate that a careful analysis of the underlying combinatorial structure of the problem allows us to obtain significantly better lower bounds than those obtained in prior work Maddah-Ali and Niesen (2014); Sengupta et al. (2015b); Ajaykrishnan et al. (2015). In addition, our machinery allows us to show that the achievable rate of Maddah-Ali and Niesen (2014) is within a multiplicative factor of four of our proposed lower bound. Our research in this area results to publishing a journal paper Ghasemi and Ramamoorthy (2017c) and two conference papers Ghasemi and Ramamoorthy (2015, 2016).

This section is organized as follows. Section 2.1 discusses the background, related work and summarizes the main contributions of our work. Section 2.2 presents our proposed lower bound technique. The multiplicative gap between the achievable rate and our lower bound is outlined in Section 2.3. Our proposed strategy also applies to certain variants of the coded caching problem that have been discussed in the literature; this is explained in Section 2.4. There have been some other approaches presented in the literature Maddah-Ali and Niesen (2014); Sengupta et al. (2015b);

Ajaykrishnan et al. (2015) for improving the lower bound on the coded caching rate. We present comparisons between our approach and the other approaches in Section 2.5. We conclude the section with a discussion of opportunities for future work in Section 2.6.

2.1 Background, Related Work and Summary of Contributions

In a coded caching system there is a server that contains N files, denoted $W_i, i = 1, \ldots, N$, each of size F bits. There are K users that are connected to the central server by means of a shared link. Each user has a local cache memory of size MF bits; we denote the cache content by the symbol Z_i (which is a function of $W_1, \ldots, W_N$). In each time slot, the i-th user demands the file W_{d_i} where $d_i \in \{1, \ldots, N\}$. The coded caching problem has two distinct phases. In the *placement phase*, the content of caches is populated; this phase should not depend on the actual user requests (which are assumed to be arbitrary). In the *delivery phase*, the server transmits a potentially coded signal that serves to satisfy the demands of each of the users. A pair (M, R) is said to be achievable if for every possible request pattern (there are N^K of them), every user can recover its desired file with high probability for large enough F. We let $R^\star(M)$ denote the infimum of all such achievable rates for a given M.

The coded caching problem can be formally described as follows. Let $[m] = \{1, \ldots, m\}$, where m is a positive integer. Let $\{W_n\}_{n=1}^N$ denote N independent random variables (representing the files) each uniformly distributed over $[2^F]$. The i-th user requests the file W_{d_i}, where $d_i \in [N]$. A (M, R) system consists of the following.

- K caching functions, $Z_i \triangleq \phi_i(W_1, \ldots, W_N)$ where $\phi_i : [2^F] \to [2^{\lfloor FM \rfloor}]$.
- A total of N^K encoding functions $\varphi_{d_1,\ldots,d_K}(W_1, \ldots, W_N)$, so that the delivery phase signal $X_{d_1,\ldots,d_K} \triangleq \varphi_{d_1,\ldots,d_K}(W_1, \ldots, W_N)$. Here, $\varphi_{d_1,\ldots,d_K} : [2^F]^N \to [2^{\lfloor FR \rfloor}]$.
- For each delivery phase signal and each user, we define appropriate decoding functions. There are a total of KN^K of them. For the k-th user, we define $\mu_{d_1,\ldots,d_K;k}(X_{d_1,\ldots,d_K}, Z_k)$, where

$k = 1, \ldots, K$ so that decoded file $\hat{W}_{d_1,\ldots,d_K;k} \triangleq \mu_{d_1,\ldots,d_K;k}(X_{d_1,\ldots,d_K}, Z_k)$. Here $\mu_{d_1,\ldots,d_K;k} : [2^{\lfloor RF \rfloor}] \times [2^{\lfloor FM \rfloor}] \to [2^F]$.

The probability of error is defined as

$$\max_{(d_1,\ldots,d_K)\in[N]^K} \max_{k\in[K]} P(\hat{W}_{d_1,\ldots,d_K;k} \neq W_{d_k}).$$

Definition 1 *The pair (M, R) is said to be achievable if for $\epsilon > 0$, there exists a file size F large enough so that there exists a (M, R) caching scheme with probability of error at most ϵ. We define*

$$R^\star(M) = \inf\{R : (M, R) \text{ is achievable}\}.$$

In this setting, it is not too hard to see that the best that a conventional caching system can do is to simply store an M/N fraction of each file in each of the caches. In order to satisfy the demands of the user, the server has to transmit the remaining $(1 - M/N)$ fraction of each of the requested files. Thus, the transmission rate (normalized by F) is given by

$$R_U(M) = \min(N, K)\left(1 - \frac{M}{N}\right). \tag{2.1}$$

Note that $\min(N, K)$ is the transmission rate in the absence of any caching. In Maddah-Ali and Niesen (2014), the factor $(1 - M/N)$ is referred to as the *local caching gain* as it is gain that is obtained purely from the cache, without any optimization of the transmission from the server. In the setting where we perform nontrivial coding in the cache and delivery phase encoding functions, Maddah-Ali and Niesen (2014) demonstrates that a carefully designed placement scheme and a corresponding delivery scheme achieves a rate

$$R_C(M) = K\left(1 - \frac{M}{N}\right) \cdot \min\left\{\frac{1}{1 + KM/N}, \frac{N}{K}\right\}, \tag{2.2}$$

where $M \in \{0, N/K, 2N/K, \ldots, N\}$. Other values of M are obtained by time-sharing between the solutions for integer multiples of N/K.

The factor $\frac{1}{1+KM/N}$ which definitely dominates when $N \geq K$ is referred to as the *global caching gain*. It is to be noted that the global caching gain depends on the overall cache size across all the users (owing to the term KM/N in the denominator) whereas the local caching gain only depends on the per-user cache size (owing to the term $1 - M/N$). Furthermore, they compare their achievable rate (*cf.* eq. (2.2)) to a cutset bound that can be expressed as follows.

$$R^{\star}(M) \geq \max_{s \in \{1,\ldots,\min(N,K)\}} \left(s - \frac{s}{\lfloor N/s \rfloor} M \right). \tag{2.3}$$

The work of Maddah-Ali and Niesen (2014) also shows that the rate $R_C(M)$ is within a factor of 12 of the cutset bound for all values of N, K and M.

2.1.1 Related work

Coded caching is related to but different from the index coding problem Bar-Yossef et al. (2011). In the index coding problem, there are N' sources such that i-th source has message W_i, $i = 1, \ldots, N'$. There are K terminals, each of which has some subset of $\{W_1, \ldots, W_{N'}\}$ available. In addition, each terminal requests a certain subset of the messages $\{W_1, \ldots, W_{N'}\}$. The aim in the index coding problem is to minimize the number of bits that are transmitted on the shared link so that the demands of each user are satisfied. It is well recognized that the index coding problem for arbitrary side information is a computationally hard problem where nonlinear codes may be necessary Bar-Yossef et al. (2011); Lubetzky and Stav (2009). In particular, the optimal *linear* index code corresponds to minimizing the rank of an appropriately defined matrix over a finite field. This so called minrank problem Bar-Yossef et al. (2011) is also known to be computationally hard. It can be observed that for a fixed but *uncoded* cache content and a fixed set of demands of the various users, the problem of determining the optimal delivery phase signal in the coded caching problem is equivalent to an index coding problem. Note however, that in the coded caching problem, we allow the cache content to be coded.

Since the original work of Maddah-Ali and Niesen (2014), there have been several aspects of coded caching that have been investigated. Reference Maddah-Ali and Niesen (2015) considers the

scenario of decentralized caching where the placement phase is driven by the users who randomly populate their caches with subsets of the files stored at the server. Approaches for updating the cache content are considered in Pedarsani et al. (2014) and the case of files with different popularity scores are considered in Niesen and Maddah-Ali (2016) and Ji et al. (2014a); Hachem et al. (2014a). Security issues in this domain are considered in Sengupta et al. (2015a). The work of Karamchandani et al. (2014) considers the more general case of hierarchical coded caching, where certain intermediate nodes in the network are equipped with potentially larger caches and investigates methods for minimizing the overall traffic in such networks (see also Hachem et al. (2014b)). Coded caching where each user requests multiple files was investigated in Ji et al. (2014b). The case of device-to-device (D2D) wireless networks where there is no central server was examined in Ji et al. (2013); Sengupta and Tandon (2015). Systems with files of differing sizes were examined in Zhang et al. (2015). The work of Tang and Ramamoorthy (2017, 2016b); Yan et al. (5064), considers the problem of leveraging the rate gains of coded caching with reduced subpacketization levels. Synchronization issues and the problem setting where end users have deadlines was investigated in Niesen and Maddah-Ali (2015); Ghasemi and Ramamoorthy (2017b).

In addition to these contributions, there have been other lines of work that deal with content caching. In a parallel line of work Shanmugam et al. (2013); Golrezaei et al. (2013); Ji et al. (2013) consider the problem of femtocaching in a wireless setting where in addition to a central server (or base station), there are helpers (with caches) interspersed in a cell that help the end users satisfy their demands. The goal is again to consider caching strategies that minimize the overall rate, but the solution approaches do not consider the worst case rate over all possible demand patterns; instead the popularity scores of the different files are explicitly taken into account. Moreover, while coding is considered, it is conceptually different in the sense that the coding is only restricted to parts of the same file and coding across different files is not considered. More recently, techniques inspired by coded caching have been employed for speeding up distributed computing Li et al. (2016); Lee et al. (2015).

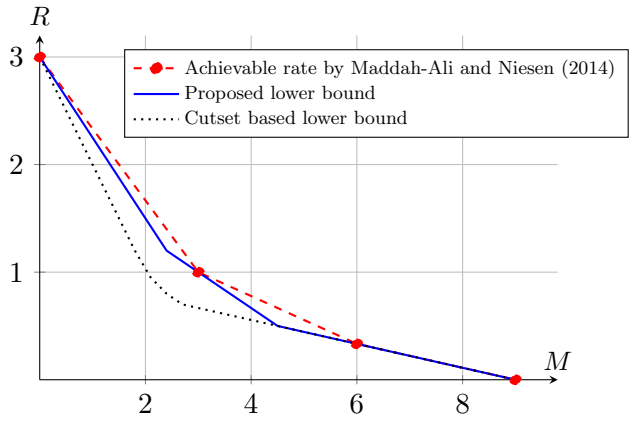


Figure 2.1: An example of a coded caching system with $N = 9$ files, $K = 3$ users. Note that the proposed lower bound is better than the cutset bound and matches the achievable rate points at multiples of N/K.

There has also been parallel work on establishing lower bounds for the coded caching problem. In Sengupta et al. (2015b), Han's inequality Cover and Thomas (2012) was leveraged to obtain an improved lower bound. A multiplicative gap of eight between their lower bound and the achievable rate in eq. (2.2) was established. The work of Ajaykrishnan et al. (2015) also presents a lower bound technique. As discussed in Section 2.5, their technique can be considered as a special case of our work. The specific case of $N = K = 3$ was considered in Tian (2015) via a computational approach. We present a detailed comparison of our technique with these other approaches in Section 2.5.

2.1.2 Summary of our contributions

In this work our main contribution is in developing improved lower bounds on the required rate for the coded caching problem. We show that the cutset based bound in eq. (2.3) is significantly loose and propose a larger class of lower bounds that are significantly tighter. Our specific contributions include the following.

- We demonstrate that the computation of our lower bound can be posed as a combinatorial labeling problem on a directed tree. Our method generates lower bounds on $\alpha R^{\star} + \beta M$, where

α, β are positive integers. While the cutset bound only optimizes over at most $\min(N, K)$ choices, our technique allows us to consider many more (α, β) pairs[1].

- We perform a careful analysis of the underlying combinatorial structure of the problem that allows us to obtain significantly better lower bounds than those obtained in prior work. For a given pair (α, β) and number of users K, it is intuitively clear that the lower bound on $\alpha R^\star + \beta M$ will be large if the number of files N is large. We define the notion of a saturated instance, which are directed trees and corresponding labelings that give the largest possible lower bound (using our technique) using as few files as possible. An analysis of saturated instances allows us to always improve on the cutset bound and in most ranges of M, our bound is strictly better.

- Our machinery allows us to show that the achievable rate of Maddah-Ali and Niesen (2014) is within a multiplicative factor of four of our proposed lower bound for all values of N and K. This is possible by analyzing some combinatorial properties of saturated instances.

- Our proposed technique also applies to other variants of coded caching problem. We discuss the application of our work to the case of D2D wireless networks and coded caching with multiple requests as well.

As an example, Figure 2.1 illustrates the tightness of the proposed lower bound for a coded caching system with a server that contains $N = 9$ files and $K = 3$ users. Specifically, our proposed bound demonstrates the optimality of the achievable scheme for values of M that are integer multiples of N/K in this specific case.

2.2 Lower Bound on $R^\star(M)$

In this section we present our proposed lower bound on $R^\star(M)$. We begin with an example that demonstrates the core idea of our approach.

[1]The cutset bound can be considered as a special case of our bound.

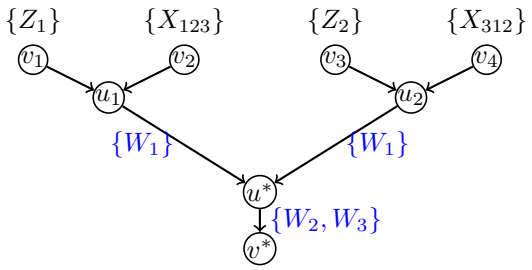


Figure 2.2: Problem instance for Example 2. For clarity of presentation, only the $W_{new}(u)$ label has been shown on the edges.

Example 2 *Consider a coded caching system with $N = K = 3$. Then, the following sequence of information theoretic inequalities hold.*

$$
\begin{aligned}
2R^\star F + 2MF &\geq H(Z_1, X_{123}) + H(Z_2, X_{312}) \\
&\overset{(a)}{=} I(W_1; Z_1, X_{123}) + H(Z_1, X_{123}|W_1) \\
&\quad + I(W_1; Z_2, X_{312}) + H(Z_2, X_{312}|W_1) \\
&= H(W_1) - H(W_1|Z_1, X_{123}) + H(Z_1, X_{123}|W_1) \\
&\quad + H(W_1) - H(W_1|Z_2, X_{312}) + H(Z_2, X_{312}|W_1) \\
&\overset{(b)}{\geq} F(1-\epsilon) + F(1-\epsilon) + H(Z_1, Z_2, X_{123}, X_{312}|W_1) \\
&= 2F(1-\epsilon) + I(W_2, W_3; Z_1, Z_2, X_{123}, X_{312}|W_1) \\
&\quad + H(Z_1, Z_2, X_{123}, X_{312}|W_1, W_2, W_3) \\
&\overset{(c)}{\geq} 2F(1-\epsilon) + 2F(1-\epsilon) = 4F(1-\epsilon),
\end{aligned}
$$

where equality (a) holds by the definition of mutual information. Inequality (b) holds by Fano's inequality since the file W_1 can be recovered with ϵ-error from the pairs (Z_1, X_{123}) and (Z_2, X_{312}) and by the fact that conditioning reduces entropy. Similarly, inequality (c) holds by Fano's inequality since the files W_2 and W_3 can be recovered with ϵ-error from $(Z_1, Z_2, X_{123}, X_{312})$. This holds for arbitrary $\epsilon > 0$ and F large enough. Dividing throughout by F we have the required result.

Thus, the key idea of the above bound is to choose the delivery phase signals in such a manner so that the various terms that are combined allow the "reuse" of the same file multiple times. For

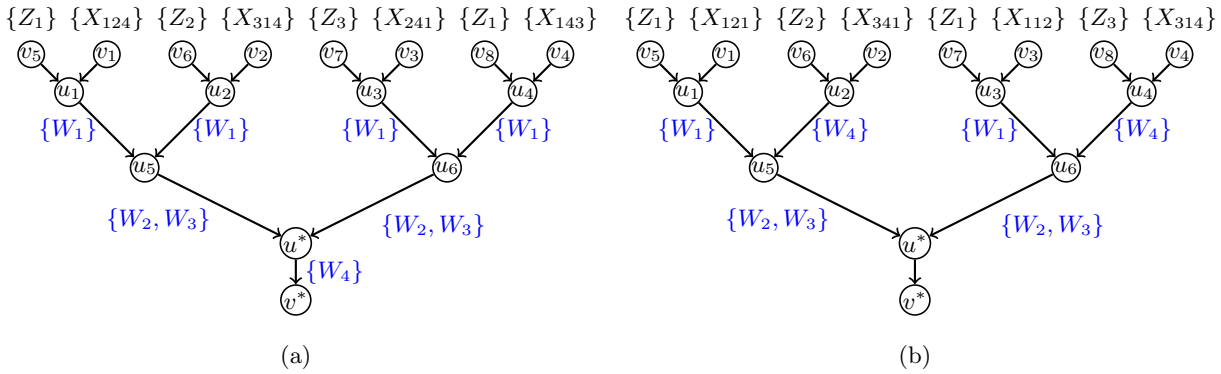


Figure 2.3: Problem instances discussed in Example 3 where $N = 4$ and $K = 3$. The instance (a) has reused more files than the corresponding cutset bound derived from instance (b).

instance, in step (a) of the above bound, we use the definition of mutual information to rewrite the terms $H(Z_1, X_{123})$ and $H(Z_2, X_{312})$. Note that both pairs (Z_1, X_{123}) and (Z_2, X_{312}) allow the recovery of the *same* file W_1, resulting in a contribution of $2F$ to the lower bound. On the other hand, the files W_2 and W_3 are recovered only once. The overall result is a lower bound of $4F$.

Thus, our lower bound works with judiciously chosen labels for the delivery phase signals and combines them with the cache signals in an appropriate way such that a given file is recovered a large number of times. It turns out that doing this systematically and tractably requires the development of several new ideas. For instance, the aforementioned chain of inequalities can be equivalently represented in terms of a directed tree with appropriate labels on its leaves and edges as shown in Figure 2.2. In particular, the leaves of the tree are labeled with cache signals Z_1 and Z_2 and delivery phase signals X_{123} and X_{312}. Each internal node of the tree corresponds to the operation of combining the signals and its outgoing edge is labeled by the newly recovered file(s), e.g., at node u_1, the file W_1 is recovered. Likewise at node u^*, the files W_2 and W_3 are recovered. The lower bound can be obtained by summing the cardinalities of the edge labels. We note here that the Appendix in Maddah-Ali and Niesen (2014) considers an application of a similar bound in the specific case of $K = N = 2$.

The next example shows another crucial point that is key to our approach. Namely, one can get the same lower bound by using different number of files. It turns out that using less files to

obtain a specific lower bound can in turn be leveraged to improve the overall lower bound on the rate.

Example 3 *Consider a coded caching system with $N = 4$, $K = 3$. Suppose that we are interested in deriving a lower bound of type $4R^\star + 4M \geq L$. Using the cutset bound in (2.3) for $s = 2$ we get $2R^\star + 2M \geq 4$, which in turn yields $4R^\star + 4M \geq 8$. The corresponding information theoretical inequalities to derive such a lower bound can be equivalently presented by the directed tree and labeled leaves and edges in Figure 2.3 (b) (this is formalized in the Appendix). Note that there are no files labeled on the last edge (u^*, v^*).*

On the other hand, consider the directed tree and the corresponding labels in Figure 2.3 (a). The crucial difference is that the edge (u^*, v^*) recovers the file W_4 in Figure 2.3 (a). Summing the cardinalities of the labels allows us to obtain the inequality $4R^\star + 4M \geq 9$ which is strictly better than the cutset bound. Intuitively, this can be explained as follows. It is not too hard to see that each subtree of the original directed tree can in turn yield an inequality by itself. For instance, consider the left subtree rooted at u^*, i.e., the subtree with v_1, v_2, v_5 and v_6 as leaves and (u_5, u^*) as its last edge. This subtree allows us to lower bound $2R^* + 2M$. Summing the cardinalities of the edges of this subtree yields the value 4; crucially, this subtree only uses three files W_1, W_2 and W_3. A similar statement holds for the right subtree rooted at u^*. This allows the remaining file W_4 to be recovered on the edge (u^*, v^*).

On the other hand an examination of Figure 2.3 (b) shows that its subtrees also yield the value 4, but use four files $W_1, \ldots, W_4$. Thus, we conclude that the subtrees of Figure 2.3 (a) are more efficient in using files. This allows one more file to be recovered on the last edge (u^*, v^*) and translates into an overall better lower bound.

The key idea of our improved lower bounding technique is thus, to consider directed trees with appropriate labels that are efficient in using the number of files. We will formalize these notions in the subsequent discussion. As we have seen, there are new concepts that are needed in working with the directed trees with labeled leaves and edges. In what follows, we formally define these concepts.

Definition 2 *Directed in-tree. A directed graph $\mathcal{T} = (V, A)$, is called a directed in-tree if there is one designated node called the root such that from any other vertex $v \in V$ there is exactly one directed path from v to the root.*

The nodes in a directed in-tree that do not have any incoming edges are referred to as the leaves. The remaining nodes, excluding the leaves and the root are called internal nodes. Each node in a directed in-tree has at most one outgoing edge. We have the following definitions for a node $v \in V$.

$$out(v) = \{u \in V : (v, u) \in A\}, \text{ (outgoing neighbor) and,}$$

$$in(v) = \{u \in V : (u, v) \in A\} \text{ (incoming neighbor set).}$$

$$in - edge(v) = \{e \in A : e = (u, v)\} \text{ (incoming edge set).}$$

In this work, we exclusively work with trees which are such that the in-degree of the root equals 1. There is a natural topological order in $\mathcal{T}$ whereby for nodes $u \in \mathcal{T}$ and $v \in \mathcal{T}$, we say that $u \succ v$ if there exists a sequence of edges that can be traversed to reach v from u. This sequence of edges is denoted $path(u, v)$.

Algorithm 1 Lower Bound Algorithm

Input: $\mathcal{T} = (V, A)$ with leaves $v_1, \ldots, v_\ell$ and $\{label(v_i)\}_{i=1}^{\ell}$, such that $\mathbb{W}(v_i) = \emptyset, i = 1, \ldots, \ell$.
Initialization:

for $i \leftarrow 1, \ldots \ell$ **do**
  $W_{new}(v_i) = \Delta(v_i, v_i)$.
  $x_{(v_i, out(v_i))} = W_{new}(v_i)$.
  $y_{(v_i, out(v_i))} = |W_{new}(v_i)|$.
end for
while there exists an unlabeled edge **do**
  Pick an unlabeled node $u \in V$ such that all edges in $in - edge(u)$ are labeled.
  $\mathbb{W}(u) = \cup_{v \in in(u)} \mathbb{W}(v) \cup W_{new}(v)$.
  $\mathbb{Z}(u) = \cup_{v \in in(u)} \mathbb{Z}(v)$.
  $\mathbb{D}(u) = \cup_{v \in in(u)} \mathbb{D}(v)$.
  $W_{new}(u) = \Delta(u, u) \setminus \mathbb{W}(u)$.
  $x_{(u, out(u))} = W_{new}(u)$.
  $y_{(u, out(u))} = |W_{new}(u)|$.
end while

Output: $L = \sum_{e \in A} y_e$.

Definition 3 *Meeting point of nodes in a directed tree. Consider nodes v_1 and v_2 in a directed in-tree $\mathcal{T} = (V, A)$. We say that v_1 and v_2 meet at node u if there exist $path(v_1, u)$ and $path(v_2, u)$ in $\mathcal{T}$ such that $path(v_1, u) \cap path(v_2, u) = \emptyset$. As there exists a path from any node in $\mathcal{T}$ to the root node, it follows that the existence of node u is guaranteed.*

Let $D = \cup_{d_1 \in [N], \ldots, d_K \in [N]} \{X_{d_1, \ldots, d_K}\}$.

Definition 4 *Labeling of directed in-tree. Each node $v \in \mathcal{T}$ is assigned a label, denoted $label(v)$, which is a subset of $\{W_1, \ldots, W_N\} \cup \{Z_1, \ldots, Z_K\} \cup D$. Moreover, we also specify $\mathbb{W}(v) \subseteq \{W_1, \ldots, W_N\}$, $\mathbb{Z}(v) \subseteq \{Z_1, \ldots, Z_K\}$ and $\mathbb{D}(v) \subseteq D$ so that $label(v) = \mathbb{W}(v) \cup \mathbb{Z}(v) \cup \mathbb{D}(v)$.*

In our formulation, the leaf nodes are denoted $v_i, i = 1, \ldots, \ell$ are such that $\mathbb{W}(v_i) = \emptyset$.

Definition 5 *Recoverability. We say that a singleton source subset $\{W_i\}$ is recoverable from the pair $(Z_j, X_{d_1, \ldots, d_K})$ if $d_j = i$. Similarly, for a given set of caches $Z' \subseteq \{Z_1, \ldots, Z_K\}$ and delivery phase signals $D' \subseteq D$, we define $Rec(Z', D') \subseteq \{W_1, \ldots, W_N\}$ to be the subset of the sources that can be recovered from pairs of the form (Z_i, X_J) where $Z_i \in Z'$ and J is a multiset of cardinality K with entries from $[N]$ such that $X_J \in D'$.*

We let the entropy of a set of random variables equal the joint entropy of all the random variables in the set. We also let $[x]^+ = \max(x, 0)$.

Given a directed tree $\mathcal{T}$ with appropriate labels on its leaves we present an algorithm (see Algorithm 1) that generates an inequality of the form $\alpha R^\star + \beta M \geq L(\alpha, \beta)$. For nodes $u, v \in \mathcal{T}$, we define the following.

$$\Delta(u, v) = Rec(\mathbb{Z}(u), \mathbb{D}(v)), \text{ and}$$
$$W_{new}(u) = \Delta(u, u) \setminus \mathbb{W}(u). \tag{2.4}$$

Algorithm 1 operates as follows. It takes as input a directed in-tree $\mathcal{T}$ where each leaf $v_i, i = 1, \ldots, \ell$ has labels $\mathbb{Z}(v_i)$ and $\mathbb{D}(v_i)$ ($\mathbb{W}(v_i)$ is set to $\emptyset$). The algorithm determines the files that are recovered at each v_i and labels the corresponding outgoing edge with $W_{new}(v_i)$ and $|W_{new}(v_i)|$. Following this, the algorithm propagates the labels further down the tree in the following manner.

For a given node u whose incoming edges are labeled, we set $\mathbb{Z}(u) = \cup_{v \in in(u)} \mathbb{Z}(v)$ and $\mathbb{D}(u) = \cup_{v \in in(u)} \mathbb{D}(v)$, i.e., each of these labels is set to the union of the corresponding labels of the nodes that belong to the incoming node set of u. Next, it sets $\mathbb{W}(u) = \cup_{v \in in(u)} \mathbb{W}(v) \cup W_{new}(v)$, i.e., in addition to the $\mathbb{W}$-labels of the incoming node set, $\mathbb{W}(u)$ also contains the new files that are recovered on the incident edges. Note that at each internal node certain cache signals and delivery phase signals *meet*, e.g., Z_1 and X_{123} meet at node u_1 in Figure 2.2. The outgoing edge of an internal node is labeled by the *new* files that are recovered at the node, e.g., at u_1 the signals Z_1 and X_{123} recover the file W_1. We call a file new if it has not been recovered upstream of a given node. In a similar manner at u^* one can recover all the files $W_1, \ldots, W_3$; however only the set $\{W_2, W_3\}$ is labeled on edge (u^*, v^*) as W_1 was recovered upstream. This process is continued recursively, i.e., we label the outgoing edges with the new files that are recovered at node u, propagate the labels and continue thereafter. The algorithm continues until it labels the last outgoing edge.

It can be seen that the operation of Algorithm 1 is in one to one correspondence with the new files recovered in the sequence of inequalities in the lower bound. For example, the outgoing labels of u_1 and u_2 in Figure 2.2 correspond to step (a) in the inequalities in Example 2. We formalize this statement in the Appendix (Lemma 6) where we show that a valid lower bound is always obtained when applying Algorithm 1. The complexity of this algorithm and the other algorithms used in this paper are discussed in Appendix A.0.5.

Definition 6 *Problem Instance. Consider a given tree $\mathcal{T}$ with leaves $v_i, i = 1, \ldots, \ell$ that are labeled as discussed above. Let $\alpha = \sum_{i=1}^{\ell} |\mathbb{D}(v_i)|$ and $\beta = \sum_{i=1}^{\ell} |\mathbb{Z}(v_i)|$. Suppose that the lower bound computed by Algorithm 1 equals L. We define the associated problem instance as $P(\mathcal{T}, \alpha, \beta, L, N, K)$. We also define $\hat{\alpha} = |\cup_{i=1}^{\ell} \mathbb{D}(v_i)|$ and $\hat{\beta} = |\cup_{i=1}^{\ell} \mathbb{Z}(v_i)|$. A problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ is said to be optimal if all instances of the form $P'(\mathcal{T}', \alpha, \beta, L', N, K)$ are such that $L' \leq L$.*

It is worth emphasizing that $\hat{\alpha} \leq \alpha$ and $\hat{\beta} \leq \beta$ as some cache and delivery phase signals may be repeated.

In the subsequent discussion, we focus on understanding the characteristics of optimal problem instances. Towards this end, we shall often start with a problem instance P and modify it in

appropriate ways to arrive at another instance P'. For ease of presentation, when needed we shall refer to quantities in instance $P(P')$ by using the corresponding superscripts. For example, for a node u in P (P'), we will denote the set of new files by $W^{P}_{new}(u)$ ($W^{P'}_{new}(u)$).

It is not too hard to see that it suffices to consider directed trees whose internal nodes have an in-degree at least two. In particular, if u has in-degree equal to 1, it is evident that $W_{new}(u) = \emptyset$ and thus, $|W_{new}(u)| = 0$. In addition, we claim that w.l.o.g. it suffices to consider trees where internal nodes have in-degree at most two. Therefore, we will assume that all internal nodes have degree equal to two. More specifically, we can show the following property of problem instances (the proof appears in the Appendix).

Claim 1 *Consider a problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ such that there exists a node $u \in \mathcal{T}$ with $|in(u)| \geq 3$. Then, there exists another instance $P'(\mathcal{T}', \alpha, \beta, L', N, K)$ where $L' \geq L$ and $|in(u)| \leq 2$ for all nodes $u \in \mathcal{T}'$.*

Henceforth, we assume that all internal nodes in the problem instances under consideration have in-degree equal to two. Claim 1 can also be used to conclude that each leaf v in an instance P is such that either $|\mathbb{Z}(v)| = 1$ or $|\mathbb{D}(v)| = 1$ but not both. Indeed, if there exists a leaf v that violates this condition, we can use the modification in the proof of Claim 1 to replace v by a directed in-tree so that the condition is satisfied. If $|\mathbb{Z}(v)| = 1$, we call v a cache node; if $|\mathbb{D}(v)| = 1$ we call it a delivery phase node. In the subsequent discussion we will assume that the delivery phase nodes are labeled in an arbitrary order $v_1, \ldots, v_\alpha$ and the cache nodes from $v_{\alpha+1}, \ldots, v_{\alpha+\beta}$, where we note that $\alpha + \beta = \ell$. Moreover, we let $\mathcal{D} = \{v_1, \ldots, v_\alpha\}$ and $\mathcal{C} = \{v_{\alpha+1}, \ldots, v_{\alpha+\beta}\}$.

In the tree $\mathcal{T}$ corresponding to problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$, consider an internal node u and the edge $e = (u, v)$. In the subsequent discussion, we shall use $\mathcal{T}_u$ to refer to the subtree that has its last edge as $(u, out(u))$, i.e., the subtree that is rooted at $out(u)$. The incoming edges into u, denoted (u_l, u) and (u_r, u) are the last edges of the disjoint left and right subtrees denoted $\mathcal{T}_{u(l)}$ and $\mathcal{T}_{u(r)}$ respectively (see Figure 2.4).

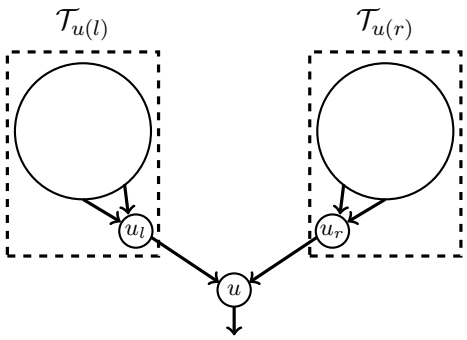


Figure 2.4: For a given node $u \in \mathcal{T}$, its in-neighbors are denoted u_l and u_r. The corresponding subtrees are denoted $\mathcal{T}_{u(l)}$ and $\mathcal{T}_{u(r)}$ and are shown enclosed in the dotted boxes.

Each of these subtrees defines a problem instance $P_l = P(\mathcal{T}_{u(l)}, \alpha_l, \beta_l, L_l, N, K)$ and $P_r = P(\mathcal{T}_{u(r)}, \alpha_r, \beta_r, L_r, N, K)$. We denote the set of delivery phase nodes and cache nodes in $\mathcal{T}_{u(r)}$ by

$$\mathcal{D}_{u(r)} = \{v \in \mathcal{D} : v \in \mathcal{T}_{u(r)}\} \text{ and}$$
$$\mathcal{C}_{u(r)} = \{v \in \mathcal{C} : v \in \mathcal{T}_{u(r)}\},$$

with similar definitions for $\mathcal{D}_{u(l)}$ and $\mathcal{C}_{u(l)}$. We also let

$$\mathcal{D}_u = \mathcal{D}_{u(l)} \cup \mathcal{D}_{u(r)}, \text{ and}$$
$$\mathcal{C}_u = \mathcal{C}_{u(l)} \cup \mathcal{C}_{u(r)}.$$

Let $\Gamma_l = \cup_{v \in \mathcal{T}_{u(l)}} W_{new}(v)$ and $\Gamma_r = \cup_{v \in \mathcal{T}_{u(r)}} W_{new}(v)$, i.e., Γ_l and Γ_r are the subsets of $\{W_1, \ldots, W_N\}$ that are used up in the problem instances P_l and P_r respectively. It can be observed that $\Gamma_l = \Delta(u_l, u_l)$ and $\Gamma_r = \Delta(u_r, u_r)$.

We shall often need to reason about the files recovered at the node u from the different subtrees. For instance, the set of cache nodes in $\mathcal{T}_{u(r)}$ and the delivery phase signals in $\mathcal{T}_{u(l)}$ meet and recover a subset of the files at u.

This set of files corresponds to those recovered from $\mathbb{Z}(u_r) \setminus \mathbb{Z}(u_l)$ and $\mathbb{D}(u_l)$, and can be informally thought of as the *files recovered when going from right to left*. Accordingly, we have the following definitions.

$$\Delta_{rl}(u) = Rec(\mathbb{Z}(u_r) \setminus \mathbb{Z}(u_l), \mathbb{D}(u_l)), \text{ and}$$
$$\Delta_{lr}(u) = Rec(\mathbb{Z}(u_l) \setminus \mathbb{Z}(u_r), \mathbb{D}(u_r)).$$

Note that by definition, we have

$$\begin{aligned}
&\Delta(u,u) = Rec(\mathbb{Z}(u), \mathbb{D}(u)) \\
&= Rec(\mathbb{Z}(u_l) \cup \mathbb{Z}(u_r), \mathbb{D}(u_l) \cup \mathbb{D}(u_r)) \\
&= Rec(\mathbb{Z}(u_l), \mathbb{D}(u_l)) \cup Rec(\mathbb{Z}(u_r), \mathbb{D}(u_r)) \\
&\quad \cup Rec(\mathbb{Z}(u_l), \mathbb{D}(u_r)) \cup Rec(\mathbb{Z}(u_r), \mathbb{D}(u_l)) \\
&\overset{(a)}{=} Rec(\mathbb{Z}(u_l), \mathbb{D}(u_l)) \cup Rec(\mathbb{Z}(u_r), \mathbb{D}(u_r)) \\
&\cup Rec(\mathbb{Z}(u_l) \setminus \mathbb{Z}(u_r), \mathbb{D}(u_r)) \cup Rec(\mathbb{Z}(u_r) \setminus \mathbb{Z}(u_l), \mathbb{D}(u_l)) \\
&= \underbrace{\Delta(\mathbb{Z}(u_l), \mathbb{D}(u_l))}_{\text{from } \mathcal{T}_{u(l)}} \cup \underbrace{\Delta(\mathbb{Z}(u_r), \mathbb{D}(u_r))}_{\text{from } \mathcal{T}_{u(r)}} \cup \Delta_{lr}(u) \cup \Delta_{rl}(u), \\
&\text{and} \qquad \mathbb{W}(u) = \Delta(\mathbb{Z}(u_l), \mathbb{D}(u_l)) \cup \Delta(\mathbb{Z}(u_r), \mathbb{D}(u_r)),
\end{aligned}$$

where (a) follows since the $Rec(\mathbb{Z}(u_l), \mathbb{D}(u_r))$ potentially contains some files that have already been recovered in $Rec(\mathbb{Z}(u_r), \mathbb{D}(u_r))$. The other equality holds because of similar reasoning. Therefore, it follows that

$$\begin{aligned}
W_{new}(u) &= \Delta(u,u) \setminus \mathbb{W}(u) \\
&= \Delta_{rl}(u) \cup \Delta_{lr}(u) \setminus \mathbb{W}(u). \qquad (2.5)
\end{aligned}$$

Note that based on Algorithm 1, we can conclude that

$$\begin{aligned}
\mathbb{W}(u) &= \cup_{v \in \{u_r, u_l\}} \mathbb{W}(v) \cup W_{new}(v) \\
&= \cup_{v \succ u} W_{new}(v) \text{ (by arguing inductively).} \qquad (2.6)
\end{aligned}$$

Algorithm 2 Computing ψ

Input: $P(\mathcal{T}, \alpha, \beta, L, N, K)$, Array $\Omega(u, \delta_u)$, where $u \in \mathcal{T}$, $\delta_u \subseteq W_{new}(u), |\delta_u| = 1$.

Initialization
  for all $u \in \mathcal{T}$, $\delta_u \subseteq W_{new}(u)$ where $|\delta_u| = 1$ **do**
    $\Omega(u, \delta_u) \leftarrow 0$,
  end for
end Initialization
for $i \leftarrow 1$ to α **do**
  for all $v' \in \mathcal{C}$ **do**
    Let u be the meeting point of v_i and v'.
    $\delta_u = \Delta(v', v_i)$.
    if $\delta_u \in W_{new}(u)$ and $\Omega(u, \delta_u) == 0$ **then**
      $\psi(v_i, v') \leftarrow 1$, and $\Omega(u, \delta_u) \leftarrow 1$.
    else
      $\psi(v_i, v') \leftarrow 0$.
    end if
  end for
end for

For the subsequent discussion, it will be useful to express the value of the lower bound L for an instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ in a functional form. In particular, we define the function $\psi : \mathcal{D} \times \mathcal{C} \rightarrow \{0, 1\}$ that allows us to express L in another way. For nodes $v_i \in \mathcal{D}, v' \in \mathcal{C}$ we can define their meeting point $u \in \mathcal{T}$. The function $\psi(v_i, v')$ is determined by means of Algorithm 2, where the sequence in which we pick the nodes $v_1, \ldots, v_\alpha$ is fixed. Each element of $W_{new}(u)$ can be recovered from multiple pairs of nodes that meet there. The array $\Omega(u, \delta_u)$ keeps track of the first time the file δ_u is encountered. The function $\psi(v_i, v')$ takes the value 1 if the file W^* recovered from the pair $(\mathbb{Z}(v'), \mathbb{D}(v_i))$ at u belongs to $W_{new}(u)$ and has not been encountered before and 0 otherwise. A formal description is given in Algorithm 2.

Claim 2 *For an instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ the following equality holds*

$$L = \sum_{i=1}^{\alpha} \sum_{v' \in \mathcal{C}} \psi(v_i, v'). \tag{2.7}$$

Proof: We first note that at the end of Algorithm 2, we have $\Omega(u, \delta_u) = 1$ for all $u \in \mathcal{T}$ and all $\delta_u \subseteq W_{new}(u)$ such that $|\delta_u| = 1$. To see this suppose that there is a $u_1 \in \mathcal{T}$ and a singleton subset δ_{u_1} of $W_{new}(u_1)$ such that $\Omega(u_1, \delta_{u_1}) = 0$. Now δ_{u_1} is recovered from some delivery phase

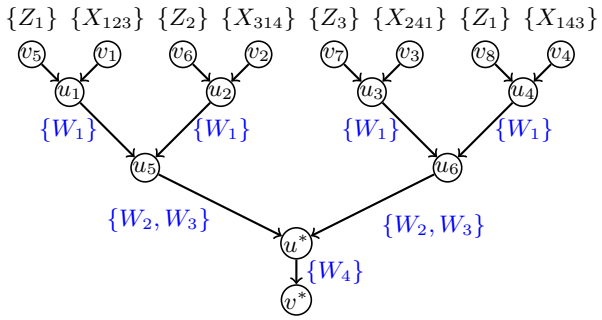


Figure 2.5: Problem instance corresponding to Example 4. There are three users and the server contains four files.

node and cache node, otherwise it would not be a subset of $W_{new}(u_1)$. As our algorithm considers all pairs of delivery phase nodes and cache nodes, at the end of the algorithm it has to be the case that $\Omega(u_1, \delta_{u_1}) = 1$.

Next, we note that for each pair (u_1, δ_{u_1}) where $u_1 \in \mathcal{T}$ and δ_{u_1} is singleton subset of $W_{new}(u_1)$, we can identify a unique pair of nodes (v_i, v') where $v_i \in \mathcal{D}$ and $v' \in \mathcal{C}$ such that $\psi(v_i, v')$ and $\Omega(u_1, \delta_{u_1})$ are set to 1 at the same step of the algorithm. The remaining pairs (v_i, v') that cannot be put in one to one correspondence with a pair (u_1, δ_{u_1}) are such that $\psi(v_i, v')$ are set to 0. Moreover as $\sum_{u \in \mathcal{T}} \sum_{\delta_u \subseteq W_{new}(u), |\delta_u|=1} \Omega(u, \delta_u) = \sum_{u \in \mathcal{T}} |W_{new}(u)| = L$, it follows that $L = \sum_{i=1}^{\alpha} \sum_{v' \in \mathcal{C}} \psi(v_i, v')$.

We now illustrate the definitions introduced above by means of the following example.

Example 4 *The problem instance in Figure 2.5 has seven internal nodes,* $\{u_1, \ldots, u_6, u^*\}$. *In the initialization step, Algorithm 2 sets* $\Omega(u_i, \{W_1\}) = 0$ *for* $1 \leq i \leq 4$, $\Omega(u_i, \{W_2\}) = \Omega(u_i, \{W_3\}) = 0$ *for* $i = 5, 6$ *and* $\Omega(u^*, \{W_4\}) = 0$. *In the next step, for node* v_1 *it sets* $\psi(v_1, v_5) = 1$, $\Omega(u_1, \{W_1\}) = 1$ *(for* $v_5 \in \mathcal{C}$*) and* $\psi(v_1, v_6) = 1$, $\Omega(u_5, \{W_2\}) = 1$ *(for* $v_6 \in \mathcal{C}$*). For* $v_7 \in \mathcal{C}$ *we have* $\delta_{u^*} = \Delta(v_7, v_1) = \{W_3\}$ *and since* $W_3 \notin W_{new}(u^*) = \{W_4\}$ *therefore* $\psi(v_1, v_7) = 0$. *By the same argument we have* $\psi(v_1, v_8) = 0$. *Thus, the contribution of* v_1 *to the lower bound, namely* $\sum_{v' \in \mathcal{C}} \psi(v_1, v') = 2$. *The complete description of the steps after the initialization, is shown in Table 2.1. The table should be read in column order from left to right. Within a column, the order of the operations is from top to bottom. Note that there are two cases,* $v_3 \in \mathcal{D}, v_6 \in \mathcal{C}$ *and* $v_4 \in \mathcal{D}, v_6 \in \mathcal{C}$ *where* $\psi(\cdot, \cdot)$ *value is set to*

0 *(since the corresponding* $\Omega(\cdot,\cdot)$ *values are already* 1*). In both cases* $\delta_{u^*} = \{W_4\}$ *and since* W_4 *is recovered already,* $\Omega(u^*, \{W_4\})$ *has already been set to* 1 *when considering* $v_2 \in \mathcal{D}, v_7 \in \mathcal{C}$*. Therefore* $\psi(v_4, v_6) = \psi(v_3, v_6) = 0$*. Another point to be noted is that delivery phase node* v_2 *contributes three files towards* L *while the other delivery nodes contribute only two files each.*

Corollary 1 *For an instance* $P(\mathcal{T}, \alpha, \beta, L, N, K)$*, we have* $L \leq \alpha \min(\beta, K)$*. Moreover, if* $N \geq \alpha \min(\beta, K)$ *there exists an instance such that* $L = \alpha \min(\beta, K)$*.*

Proof: For a node v_i, where $1 \leq i \leq \alpha$, we have

$$\begin{aligned} \sum_{v' \in \mathcal{C}} \psi(v_i, v') &\leq |\cup_{v' \in \mathcal{C}} \mathbb{Z}(v')| \\ &= \hat{\beta}, \\ &\leq \min(\beta, K). \end{aligned} \tag{2.8}$$

Let u denote the meeting point of v' and v_i. The first inequality above holds since $\psi(v_i, v') = 1$ implies that $\delta_u = \Delta(v', v_i) \subseteq W_{new}(u)$ and

$$\begin{aligned} \sum_{v' \in \mathcal{C}} \psi(v_i, v') &\leq |\cup_{v' \in \mathcal{C}} Rec(\mathbb{D}(v_i), \mathbb{Z}(v'))| \\ &= |Rec(\mathbb{D}(v_i), \cup_{v' \in \mathcal{C}} \mathbb{Z}(v'))| \leq |\cup_{v' \in \mathcal{C}} \mathbb{Z}(v')|. \end{aligned}$$

From eq. (2.8) we can conclude that $L = \sum_{i=1}^{\alpha} \sum_{v' \in \mathcal{C}} \psi(v_i, v') \leq \alpha \min(\beta, K)$. If $N \geq \alpha \min(\beta, K)$, it is easy to construct an instance with $L = \alpha \min(\beta, K)$. We simply pick any directed tree on $\alpha + \beta$ leaves. Let the cache node indices be Z_1 repeated $\beta - \min(\beta, K) + 1$ times and $Z_2, Z_3, \ldots, Z_{\min(\beta,K)-1}, Z_{\min(\beta,K)}$. Suppose that node $v \in \mathcal{D}, v' \in \mathcal{C}'$ meet at node u. We label the delivery phase leaves such that $|\cup_{(v,v') \in \mathcal{D} \times \mathcal{C}'} \Delta(v', v)| = \alpha \min(\beta, K)$. This can be done since N is large enough so that we can choose the labels such that $Rec(\mathbb{Z}(v_1'), \mathbb{D}(v_1)) \cap Rec(\mathbb{Z}(v_2'), \mathbb{D}(v_2)) = \emptyset$ for $v_1', v_2' \in \mathcal{C}'$ and $v_1, v_2 \in \mathcal{D}$. For instance, initialize $\mathbb{D}(v) = X_{1,1,\ldots,1}$ for all $v \in \mathcal{D}$ and then set $\mathbb{D}(v_i) = X_{d_1,\ldots,d_K}$, $d_j = (i-1)\min(\beta, K) + j$ for $j = 1, \ldots, \min(\beta, K)$, and $i = 1, \ldots, \alpha$. We illustrate the construction outlined above by means of the following example.

Table 2.1: The steps in Algorithm 2 after initialization when applied to Example 4. The steps flow from the leftmost to the rightmost column, and in each column from the top to the bottom row.

setting	v_1	v_2	v_3	v_4
v_5	$\delta_{u_1} = W_1$ $\psi(v_1, v_5) = 1$ $\Omega(u_1, W_1) = 1$	$\delta_{u_5} = W_3$ $\psi(v_2, v_5) = 1$ $\Omega(u_5, W_3) = 1$	$\delta_{u^*} = W_2$ $\psi(v_3, v_5) = 0$	$\delta_{u^*} = W_1$ $\psi(v_4, v_5) = 0$
v_6	$\delta_{u_5} = W_2$ $\psi(v_1, v_6) = 1$ $\Omega(u_5, W_2) = 1$	$\delta_{u_2} = W_1$ $\psi(v_2, v_6) = 1$ $\Omega(u_2, W_1) = 1$	$\delta_{u^*} = W_4$ $\psi(v_3, v_6) = 0$ $\Omega(u^*, W_4) = 1$	$\delta_{u^*} = W_4$ $\psi(v_4, v_6) = 0$ $\Omega(u^*, W_4) = 1$
v_7	$\delta_{u^*} = W_3$ $\psi(v_1, v_7) = 0$	$\delta_{u^*} = W_4$ $\psi(v_2, v_7) = 1$ $\Omega(u^*, W_4) = 1$	$\delta_{u_3} = W_1$ $\psi(v_3, v_7) = 1$ $\Omega(u_3, W_1) = 1$	$\delta_{u_6} = W_2$ $\psi(v_4, v_7) = 1$ $\Omega(u_6, W_2) = 1$
v_8	$\delta_{u^*} = W_1$ $\psi(v_1, v_8) = 0$	$\delta_{u^*} = W_3$ $\psi(v_2, v_8) = 0$	$\delta_{u_6} = W_2$ $\psi(v_3, v_8) = 1$ $\Omega(u_6, W_2) = 1$	$\delta_{u_4} = W_3$ $\psi(v_4, v_8) = 1$ $\Omega(u_4, W_3) = 1$

Example 5 *Let $\alpha = \beta = 2$, $K = 2$, and $N = 4$. We arbitrarily pick a directed tree with v_1, v_2 as delivery nodes and v_3, v_4 as cache nodes. We label $\mathbb{Z}(v_3) = Z_1$ and $\mathbb{Z}(v_3) = Z_2$, and delivery nodes as $\mathbb{D}(v_1) = X_{1,2}$ and $\mathbb{D}(v_2) = X_{3,4}$. Such a problem instance is illustrated in Figure 2.6 (a). It is evident that applying Algorithm 1 on this instance yields a lower bound of 4. However, as we will see later, this instance is not efficient in reusing files.*

At this point we have established that for a given problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$, we can always generate an inequality of the form $\alpha R^\star + \beta M \geq L$. It is natural to therefore consider *optimal* problem instances that maximize the lower bound for a given value of α, β, N and K.

Definition 7 *For given α, β, N and K, we say that a problem instance $P(\mathcal{T}^*, \alpha, \beta, L^*, N, K)$ is optimal if all problem instances $P'(\mathcal{T}, \alpha, \beta, L, N, K))$ are such that $L^* \geq L$.*

Recall that $\hat{\beta} = |\cup_{i=1}^{\ell} \mathbb{Z}(v_i)|$. For a problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$, it may be possible that $\hat{\beta} < \min(\beta, K)$. However, given such an instance, we can convert it into another instance where $\hat{\beta} = \min(\beta, K)$ without reducing the value of L. In fact, the following stronger statement holds (see Appendix A.0.2 for a proof).

Claim 3 *For a problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ suppose that there exists an internal node u^* with associated problem instance $P^* = P(\mathcal{T}_{u^*}, \alpha^*, \beta^*, L^*, N^*, K)$ such that the following condition holds.*

$$\hat{\beta}^* < \min(\beta^*, K).$$

Then, there exists another problem instance $P'(\mathcal{T}', \alpha, \beta, L', N, K)$ where $L' \geq L$ such that the above condition does not hold.

The next claim formalizes the intuitive fact that permuting the cache nodes and the delivery phase signals by the same permutation does not change the $\mathbb{W}$ labels and the lower bound of the instance.

Claim 4 *Let $P(\mathcal{T}, \alpha, \beta, L, N, K)$ to be a problem instance and let $\pi : [K] \longrightarrow [K]$ to be a permutation with inverse σ. Assume that the problem instance $P'(\mathcal{T}', \alpha, \beta, L', N, K)$ is obtained from P by the following changes for all $v \in \mathcal{D}$ and $v' \in \mathcal{C}$.*

- *Let $\mathbb{Z}^P(v') = Z_i$, then set $\mathbb{Z}^{P'}(v') = Z_{\pi(i)}$.*
- *Let $\mathbb{D}^P(v) = X_{d_1,\ldots,d_K}$, then set $\mathbb{D}^{P'}(v) = X_{d_{\sigma(1)},\ldots,d_{\sigma(K)}}$.*

Then $W^{P'}_{new}(u) = W^P_{new}(u)$, $\mathbb{W}^{P'}(u) = \mathbb{W}^P(u)$ for $u \in \mathcal{T}$, and $L' = L$.

Proof: We note that

$$\begin{aligned} Rec(Z_i, X_{d_1,\ldots,d_K}) &= W_{d_i} \\ &= W_{d_{\sigma(\pi(i))}} = Rec(Z_{\pi(i)}, X_{d_{\sigma(1)},\ldots,d_{\sigma(K)}}) \end{aligned}$$

for $i = 1, \ldots, K$. Therefore, for any $v \in \mathcal{D}$ and $v' \in \mathcal{C}$, we have $\Delta^{P'}(v', v) = \Delta^P(v', v)$ and more generally $\Delta^{P'}(u, u) = \Delta^P(u, u)$. Furthermore, $\mathbb{W}^P(u) = \Delta^P(u_l, u_l) \cup \Delta^P(u_r, u_r)$ and we have $\mathbb{W}^P(u) = \mathbb{W}^{P'}(u)$ for any $u \in \mathcal{T}$. Using eq. (2.4), we have $W^{P'}_{new}(u) = W^P_{new}(u)$ for all $u \in \mathcal{T}'$. It follows that $L' = L$.

Henceforth, we will assume w.l.o.g. that $\hat{\beta} = \min(\beta, K)$ and that Claim 3 holds. Our next lemma shows a structural property of problem instances. Namely for an instance where $L < \alpha \min(\beta, K)$, increasing the number of files allows us to increase the value of L. This lemma is a key ingredient in our proof of the main theorem (the proof appears in the Appendix).

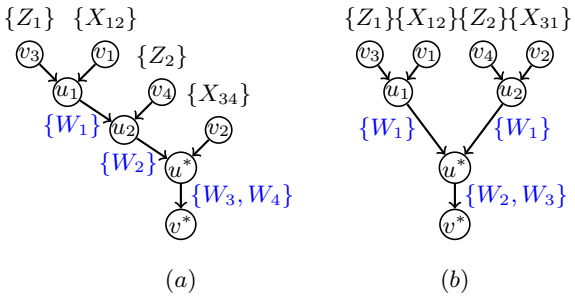


Figure 2.6: (*a*) Problem instance $P'(\mathcal{T}', \alpha, \beta, L, N', K)$, (*b*) problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ where $\alpha = 2$, $\beta = 2$ and $K = 2$. Both instances reach $L = \alpha \min(\beta, K) = 4$ with different number of files $N = 3$ and $N' = 4$.

Lemma 1 *Let $P = P(\mathcal{T}, \alpha, \beta, L, K, N)$ be an instance where $L < \alpha \min(\beta, K)$. Then, we can construct a new instance $P' = P(\mathcal{T}', \alpha, \beta, L', K, N + 1)$, where $L' = L + 1$.*

Informally, another property of optimal problem instances is that the same file is recovered as many times as possible at the same level of the tree. For instance, in Figure 2.2, W_1 is recovered in both $\mathcal{T}_{u^*(l)}$ and $\mathcal{T}_{u^*(r)}$. In fact, intuitively it is clear that the same set of files can be reused in any subtrees of an internal node. Our next claim formalizes this intuition. Recall that for a node u, $\Gamma_l = \cup_{v \in \mathcal{T}_{u(l)}} W_{new}(u)$ and $\Gamma_r = \cup_{v \in \mathcal{T}_{u(r)}} W_{new}(u)$.

Claim 5 *Consider an instance $P = P(\mathcal{T}, \alpha, \beta, L, K, N)$. For all nodes $u \in \mathcal{T}$, suppose w.l.o.g. that $|\Gamma_l| \geq |\Gamma_r|$. Suppose that there exist a node $u \in \mathcal{T}$ such that such that $\Gamma_r \nsubseteq \Gamma_l$. Then there exists another instance $P'(\mathcal{T}', \alpha, \beta, L', N', K)$ such that $N' \leq N$, $L' \geq L$, and $\Gamma_r \subseteq \Gamma_l$ for all $u \in \mathcal{T}'$.*

Next, we upper bound the maximum value of $|W_{new}(u)|$ for a node $u \in \mathcal{T}$.

Claim 6 *In instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$, consider an internal node u. Let $\rho(u) = \hat{\alpha}_l[\min(\beta_r, K - \beta_l)]^+ + \hat{\alpha}_r[\min(\beta_l, K - \beta_r)]^+$. We have*

$$|W_{new}(u)| \leq \min\left(\rho(u), [N - |\Gamma_l \cup \Gamma_r|]^+\right).$$

Proof: From eq. (2.5) it follows that

$$|W_{new}(u)| \leq |\Delta_{rl}(u) \setminus \mathbb{W}(u)| + |\Delta_{lr}(u) \setminus \mathbb{W}(u)|.$$

Next, we observe that

$$\begin{aligned}
|\Delta_{rl}(u) \setminus \mathbb{W}(u)| &= |Rec(\mathbb{Z}(u_r) \setminus \mathbb{Z}(u_l), \mathbb{D}(u_l)) \setminus \mathbb{W}(u)| \\
&\leq |\mathbb{D}(u_l)| \times |\mathbb{Z}(u_r) \setminus \mathbb{Z}(u_l)| \\
&\overset{(a)}{\leq} \hat{\alpha}_l \times \min(\hat{\beta}_r, K - \hat{\beta}_l), \\
&\overset{(b)}{=} \hat{\alpha}_l \times [\min(\beta_r, K - \beta_l)]^+,
\end{aligned}$$

where inequality (a) holds, since $|\mathbb{D}(u_l)| = \hat{\alpha}_l$ and $|\mathbb{Z}(u_r) \setminus \mathbb{Z}(u_l)| \leq \min(\hat{\beta}_r, K - \hat{\beta}_l)$. Inequality (b) holds under the conditions $\hat{\beta}_l = \min(\beta_l, K)$ and $\hat{\beta}_r = \min(\beta_r, K)$ (see Claim 10 in Appendix). We can bound $|\Delta_{lr}(u) \setminus \mathbb{W}(u)|$ in a similar manner.

To conclude the proof we note that instances P_l and P_r recover a total of $|\Gamma_l \cup \Gamma_r|$ sources. As the total number of sources is N, $|W_{new}(u)| \leq [N - |\Gamma_l \cup \Gamma_r|]^+$.

Definition 8 *Saturation number. Consider an instance $P^*(\mathcal{T}^*, \alpha, \beta, L^*, N^*, K)$, where $L^* = \alpha \min(\beta, K)$, such that for all problem instances of the form $P(\mathcal{T}, \alpha, \beta, L^*, N, K)$, we have $N^* \leq N$. We call N^* the saturation number of instances with parameters (α, β, K) and denote it by $N_{sat}(\alpha, \beta, K)$.*

In essence, for given α, β and K, saturated instances are most efficient in using the number of available files. It is easy to see that $N_{sat}(\alpha, \beta, K) \leq \alpha \min(\beta, K)$ since one can construct an instance with lower bound $\alpha \min(\beta, K)$ when $\alpha \min(\beta, K) \leq N$ (see Corollary 1).

Example 6 *Consider the two problem instances P and P' with $\alpha = 2, \beta = 2$ and $K = 2$ that are shown in Figure 2.6. The lower bound for both instances is $L = \alpha \min(\beta, K) = 4$. However, instance P uses one less file than P'. This reduction is accomplished by reusing file W_1 at both $\mathcal{T}_{u^*(l)}$ and $\mathcal{T}_{u^*(r)}$. The instance P' can be treated as a trivial instance constructed by the procedure suggested in the proof of Corollary 1 as it uses $N' = \alpha \min(\beta, K) = 4$ files. It can be verified by Algorithm 4 in Section 2.2.2 that P is one of the problem instances associated with $N_{sat}(2, 2, 2)$; therefore, $N_{sat}(2, 2, 2) = 3$.*

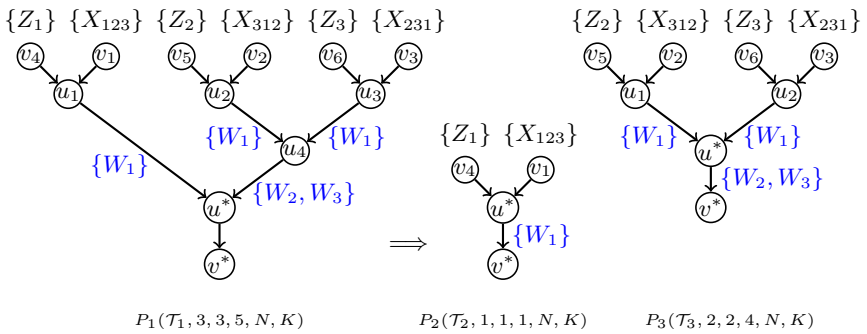


Figure 2.7: Problem instances with $N = K = 3$. Instance P_1 is non-atomic as the corresponding lower bound can be obtained by summing the lower bounds from P_2 and P_3.

Definition 9 *Atomic problem instance. For a given optimal problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ it is possible that there exist other optimal problem instances $P_i(\alpha_i, \beta_i, L_i, N, K), i = 1, \ldots, m$ with $m \geq 2$ such that $\sum_{i=1}^{m} \alpha_i = \alpha, \sum_{i=1}^{m} \beta_i = \beta$ and $\sum_{i=1}^{m} L_i = L$, i.e., the value of L follows from appropriately combining smaller problems. In this case we call the instance P non-atomic. Conversely, if such smaller problem instances do not exist, we call P an atomic problem instance.*

Example 7 *Consider the problem instance P_1 shown in Figure 2.7 with $N = K = 3$. The lower bound associated with this instance, $3R^\star + 3M \geq 5$, can be obtained by combining the lower bounds acquired by P_2 and P_3. Specifically, instance P_2 yields $R^\star + M \geq 1$ and instance P_3 yields $2R^\star + 2M \geq 4$. Note that in P_1 the last edge (u^*, v^*) is such that $W_{new}(u^*) = \emptyset$. Thus, the tree can be split into two separate instances at u^*. Thus it is non-atomic.*

It is evident that instances where no new file is recovered in the last edge are non-atomic. However, we emphasize that there are other instances that are non-atomic as well. For example, consider instance P_1', obtained from P_1 where we change the label $\mathbb{D}(v_3)$ to X_{221}. In P_1', the labels of edges (u_4, u^*) and (u^*, v^*) will change to $\{W_2\}$ and $\{W_3\}$ respectively; none of the other labels will change. Even though $W_{new}(u^*)$ is nonempty in P_1', but we still call it non-atomic since the associated lower bound does not change.

The following theorem and its corollary are the main results of our paper and can be used to identify optimal problem instances.

Theorem 1 *Suppose that there exists an optimal and atomic problem instance* $P_o(\mathcal{T} = (V, A), \alpha, \beta, L_o, N, K)$. *Then, there exists an optimal and atomic problem instance* $P^*(\mathcal{T}^* = (V^*, A^*), \alpha, \beta, L^*, N, K)$ *where* $L^* = L_o$ *with the following properties. Let us denote the last edge in* P^* *with* (u^*, v^*). *Let* $P_l^* = P(\mathcal{T}^*_{u^*(l)}, \alpha_l, \beta_l, L_l^*, |\Gamma_l|, K)$ *and* $P_r^* = P(\mathcal{T}^*_{u^*(r)}, \alpha_r, \beta_r, L_r^*, |\Gamma_r|, K)$. *Then, we have*

$$\begin{aligned} L_l^* &= \alpha_l \min(\beta_l, K), \\ L_r^* &= \alpha_r \min(\beta_r, K), \text{ and} \\ L^* &= \min\left(\alpha \min(\beta, K), L_l^* + L_r^* + [N - N_0]^+\right), \end{aligned} \tag{2.9}$$

where $N_0 = \max(N_{sat}(\alpha_l, \beta_l, K), N_{sat}(\alpha_r, \beta_r, K))$[2]. *Furthermore,* $\min(\beta_l, \beta_r) < K$.

Proof: Note that we assume that the problem instance P_o is atomic. This implies that $W^{P_o}_{new}(u^*) \neq \emptyset$ and, consequently, $N > |\Gamma_l|, |\Gamma_r|$. Using Claim 3 we can assert that $\hat{\beta}_l = \min(\beta_l, K)$ and $\hat{\beta}_r = \min(\beta_r, K)$.

We denote by (u^*, v^*), the last edge in P_o. We let $P_l = P(\mathcal{T}_{u^*(l)}, \alpha_l, \beta_l, L_l, |\Gamma_l|, K)$ and $P_r = P(\mathcal{T}_{u^*(r)}, \alpha_r, \beta_r, L_r, |\Gamma_r|, K)$. It is easy to see that $L_o = L_l + L_r + |W^{P_o}_{new}(u^*)|$. Suppose that $L_l < \alpha_l \min(\beta_l, K)$. We apply the result of Lemma 1, by noting that $|\Gamma_l| < N$, and conclude that there exists another instance $P_l^{**} = P(\mathcal{T}^{**}_{u^*(l)}, \alpha_l, \beta_l, L_l^* + 1, |\Gamma_l| + 1, K)$ that can replace P_l, where the new file is denoted W^*. We also note that in P_o, $W^* \in W^{P_o}_{new}(u^*)$. Let us denote the new instance P_o'. We emphasize that the nature of the modification in Lemma 1 is such that $\Delta^{P_o'}(u^*, u^*) = \Delta^{P_o}(u^*, u^*)$. Moreover, we note that $\mathbb{W}^{P_o'}(u^*) = \mathbb{W}^{P_o}(u^*) \cup \{W^*\}$. Thus,

$$\begin{aligned} W^{P_o'}_{new}(u^*) &= \Delta^{P_o'}(u^*, u^*) \setminus \mathbb{W}^{P_o'}(u^*) \\ &= \Delta^{P_o'}(u^*, u^*) \setminus \mathbb{W}^{P_o}(u^*) \cup \{W^*\} \\ &= W^{P_o}_{new}(u^*) \setminus \{W^*\}. \end{aligned}$$

The problem instance P_o' is also optimal since L_l is increased by one and $|W^{P_o}_{new}(u^*)|$ is decreased by one, leaving L_o unchanged. Therefore, moving files from $W^{P_o}_{new}(u^*)$ to either P_l or P_r preserves optimality. In addition, from $L_o' = L_o$ and that P_o is atomic, P_o' is atomic. Based on this argument,

[2] *As the instance is atomic, we have* $N > N_0$.

we can immediately conclude that we cannot have $L_l < \alpha_l \min(\beta_l, K)$ and $L_r < \alpha_r \min(\beta_r, K)$ as the file W^* can be used to simultaneously modify the instance P_r. Upon this modification, we can conclude that L_o can be increased by one, which contradicts the optimality of the instance P_o. Thus we assume that $L_r = \alpha_r \min(\beta_r, K)$. We can repeatedly apply the operation of moving files from $W^{P_o}_{new}(u^*)$ to P_l until we have $L^*_l = \alpha_l \min(\beta_l, K)$. It has to be the case that $|W^{P_o}_{new}(u^*)| > \alpha_l \min(\beta_l, K) - |\Gamma_l|$ so that we can repeatedly apply the operation of moving the files, for if this were not true, the instance P_o would not be atomic.

We will denote the instance that we arrive at after completing these modification by P^* which is optimal and atomic. We can also observe at this point that if we have $\beta_l \geq K$ and $\beta_r \geq K$ so that $\hat{\beta}_l = \hat{\beta}_r = K$, then $W^{P^*}_{new}(u^*) = \emptyset$ (by Claim 6) which implies that the original instance P_o is not atomic. Thus, either β_l or β_r or both have to be strictly smaller than K. In the discussion below we assume w.l.o.g. that $\beta_r < K$. It is easy to see that

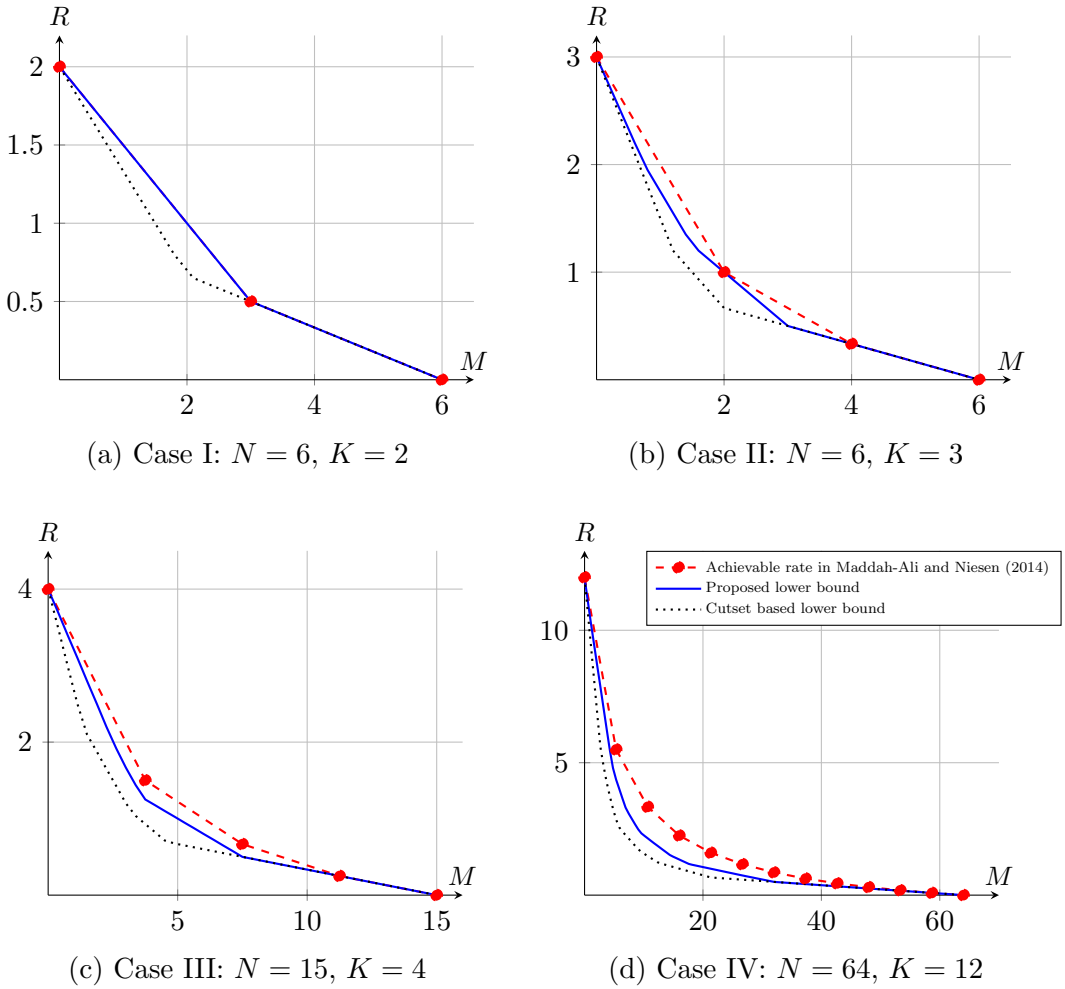
$$L^* = L^*_l + L^*_r + |W^{P^*}_{new}(u^*)|.$$

We define $\tilde{\rho}(u^*) = \alpha_l \times [\min(\beta_r, K - \beta_l)]^+ + \alpha_r \times [\min(\beta_l, K - \beta_r)]^+$ where $\tilde{\rho}(u^*) \geq \rho(u^*)$ due to the fact that $\alpha_l \geq \hat{\alpha}_l$ and $\alpha_r \geq \hat{\alpha}_r$. Using this and Claim 6, we have that

$$|W^{P^*}_{new}(u^*)| \leq \min\left(\tilde{\rho}(u^*), [N - \max(|\Gamma^*_l|, |\Gamma^*_r|)]^+\right).$$

For an optimal instance, we claim that the above inequality is met with equality. If $L^* = \alpha \min(\beta, K)$ there is nothing to prove. In this case, $|W^{P^*}_{new}(u^*)| = \alpha \min(\beta, K) - L^*_l - L^*_r = \tilde{\rho}(u^*)$ (see Claim 11 in Appendix) and the above inequality is met with equality.

Otherwise, we have $L^* < \alpha \min(\beta, K)$ which implies $\tilde{\rho}(u^*) > |W^{P^*}_{new}(u^*)|$ and $\tilde{\rho}(u^*) > N - \max(|\Gamma^*_l|, |\Gamma^*_r|)$. From the Claim 5, we can assume that either $\Gamma^*_l \subseteq \Gamma^*_r$ or $\Gamma^*_r \subseteq \Gamma^*_l$. In P^*, $N_{used} = \max(|\Gamma^*_l|, |\Gamma^*_r|) + |W^{P^*}_{new}(u^*)|$ files are used so far. Now, if $N > N_{used}$, we can use Lemma 1 to conclude that there exists a problem instance $P''(\mathcal{T}'', \alpha, \beta, L'', N'', K)$ where $N'' = N_{used} + 1 \leq N$ and $L'' = L^* + 1$. This is a contradiction since we assumed that P^* is optimal. Therefore, $N \leq N_{used}$. In addition, since the number of available files is N thus $N \geq N_{used}$. As a result, $N = N_{used} = \max(|\Gamma^*_l|, |\Gamma^*_r|) + |W^{P^*}_{new}(u^*)|$ and the inequality is met with equality. In both cases,



(a) Case I: $N = 6$, $K = 2$

(b) Case II: $N = 6$, $K = 3$

(c) Case III: $N = 15$, $K = 4$

(d) Case IV: $N = 64$, $K = 12$

Figure 2.8: Comparison of the proposed lower bound and the cutset bound.

we conclude that

$$|W_{new}^{P^*}(u^*)| = \min\left(\tilde{\rho}(u^*), [N - \max(|\Gamma_l^*|, |\Gamma_r^*|)]^+\right).$$

It follows that

$$L^* = \min\left(\alpha \min(\beta, K), L_l^* + L_r^* + [N - \max(|\Gamma_l^*|, |\Gamma_r^*|)]^+\right).$$

If $L^* = \alpha \min(\beta, K)$ the saturated instance associated with $N_{sat}(\alpha, \beta, K)$ is an optimal instance. Otherwise, $L^* < \alpha \min(\beta, K)$, and we have

$$\begin{aligned} |W_{new}^{P^*}(u^*)| &= [N - \max(|\Gamma_l^*|, |\Gamma_r^*|)]^+ \\ &\leq [N - \max(N_{sat}(\alpha_l, \beta_l, K), N_{sat}(\alpha_r, \beta_r, K))]^+ . \end{aligned} \tag{2.10}$$

We claim that for P^* to be optimal, P_l^* and P_r^* have to be such that

$$\max(|\Gamma_l^*|, |\Gamma_r^*|) = \max(N_{sat}(\alpha_l, \beta_l, K), N_{sat}(\alpha_r, \beta_r, K)).$$

To see this we proceed as follows. Note that by the definition of saturation number, there exist problem instances $P_l'(\mathcal{T}_l', \alpha_l, \beta_l, L_l', N_l', K)$ and $P_r'(\mathcal{T}_r', \alpha_r, \beta_r, L_r', N_r', K)$ such that $L_l' = L_l^*$, $L_r' = L_r^*$, $N_l' = N_{sat}(\alpha_l, \beta_l, K)$ and $N_r' = N_{sat}(\alpha_r, \beta_r, K)$. W.l.o.g. let assume $N_l' \geq N_r'$. By the Claims 3 and 5 problem instances P_l' and P_r' can be modified in such a way that $\hat{\beta}_l' = \min(\beta_l, K)$, $\hat{\beta}_r' = \min(\beta_r, K)$ and $\Gamma_l' \subseteq \Gamma_r'$. Also, by Claim 4 we can set $\cup_{v \in \mathcal{C}_l'} \mathbb{Z}(v) = \{Z_1, \ldots, Z_{\hat{\beta}_l'}\}$ and $\cup_{v \in \mathcal{C}_r'} \mathbb{Z}(v) = \{Z_{K - \hat{\beta}_r' + 1}, \ldots, Z_K\}$. This ensures that $\hat{\beta}_l = \min(\beta_l, K)$, $\hat{\beta}_r = \min(\beta_r, K)$, and $\hat{\beta} = \min(\beta, K)$ hold in the defined problem instance. Now, consider the problem instance $P' = P(\mathcal{T}', \alpha, \beta, L', N, K)$ with last edge (u', v') where P_l' and P_r' are instances corresponding to u_l' and u_r' respectively. The instance P' uses $N_l' + |W_{new}^{P'}(u')|$ files. If $N - N_l' - |W_{new}^{P'}(u')| \geq 1$, then we are able to apply Lemma 1 $N - N_l' - |W_{new}^{P'}(u')|$ times and come up with a modified version of P' so that either $L' = \alpha \min(\beta, K)$ or $N - N_l' - |W_{new}^{P'}(u')| = 0$. The first case cannot happen since by assumption P^* is optimal and $L' \leq L^* < \alpha \min(\beta, K)$. Therefore, $|W_{new}^{P'}(u')| = N - N_l'$ and $L' = L_l^* + L_r^* + N - N_l'$. Finally, as $L' \leq L^*$ and $L^* \leq L_l^* + L_r^* + N - N_l'$, we conclude that $L' = L^*$.

Corollary 2 *Suppose that there exists an optimal and atomic problem instance*

$$P_o(\mathcal{T} = (V, A), \alpha, \beta, L_o, N, K).$$

Consider problem instances $P_l'(\alpha_l', \beta_l', L_l', N, K)$ and $P_r'(\alpha_r', \beta_r', L_r', N, K)$ such that $\alpha_l' + \alpha_r' = \alpha$ and $\beta_l' + \beta_r' = \beta$ such that $N \geq N_0' = \max(N_{sat}(\alpha_l', \beta_l', K), N_{sat}(\alpha_r', \beta_r', K))$. Then we have

$$L_o \geq \min\left(\alpha \min(\beta, K), L_l' + L_r' + N - N_0'\right).$$

Proof: The result follows by applying the arguments in the proof of Theorem 1, to the problem instance where P_l^* and P_r^* are replaced by P_l' and P_r' respectively.

Lemma 2 *Consider the class of coded caching systems where $K = 3$ and $N = 3n$ for $n = 1, 2, 3, \ldots$. For this class, the achievable scheme in Maddah-Ali and Niesen (2014) for $M \in \{0, n, 2n, 3n\}$ is optimal.*

Proof: From the achievable scheme in Maddah-Ali and Niesen (2014) we have $R^\star(0) \leq 3$, $R^\star(n) \leq 1$, $R^\star(2n) \leq 1/3$, and $R^\star(3n) \leq 0$. It is easy to see that $N_{sat}(\alpha, 1, 3) = \alpha$ for any integer α. Then, the following inequalities hold,

$$\begin{aligned} 3nR^\star + M &\geq 3n, \\ nR^\star + 3M &\geq 3n, \text{ and} \\ 2nR^\star + 2M &\geq 4n. \end{aligned}$$

These inequalities are the result of Corollary 2 for $(\alpha, \beta) = (\alpha_l', \beta_l') = (3n, 1)$, $(\alpha, \beta) = (\alpha_l', \beta_l') = (n, 3)$ and $(\alpha, \beta) = (2n, 2)$ with $(\alpha_l', \beta_l') = (n, 1)$ respectively. The first two inequalities above can also be obtained by using the cutset bound while the third one cannot. Now, the second inequality for $M = 0$ implies that the achievable rate $R^\star(0) \leq 3$ is optimal. Similarly, the third inequality for $M = n$ implies that achievable rate $R^\star(n) \leq 1$ is optimal. Finally, the first inequality can be used to show that achievable rates $R^\star(2n) \leq 1/3$ and $R^\star(3n) \leq 0$ are optimal.

The following example demonstrates the effectiveness of Corollary 2.

Example 8 *Consider a system with $N = 64$, $K = 12$ and cache size $M = 16/3$. The cut-set bound for such a system provides a lower bound $R^\star(M) \geq 77/27 = 2.852$. Now, using the approach of Theorem 1 for $\alpha = 12$, $\beta = 8$, $(\alpha_l, \beta_l) = (\alpha_r, \beta_r) = (6, 4)$ yields $12R^\star + 8M \geq \min(12 \times 8, 24 + 24 + 64 - N_{sat}(6, 4, 12))$. It can be shown that $N_{sat}(6, 4, 12) = 17$ (see Algorithm 4 in Section 2.2.2). Therefore, $R^\star(M) \geq 157/36 = 4.361$. This is significantly closer to the achievable rate of 5.5 (from Maddah-Ali and Niesen (2014)).*

Theorem 1 can be leveraged effectively if it can also yield the optimal values of α_l, β_l and α_r, β_r. However, currently we do not have an algorithm for picking them in an optimal manner. Thus,

we have to use Corollary 2 with either the exact value of $N_{sat}(\alpha, \beta, K)$ or an upper bound on it. Algorithm 4 in Section 2.2.2 is an algorithm to calculate the value of $N_{sat}(\alpha, \beta, K)$. Setting $\alpha_l = \lceil \alpha/2 \rceil$, $\beta_l = \lfloor \beta/2 \rfloor$ in Theorem 1 and using the corresponding values of the saturation numbers, we can obtain the results plotted in Figure 2.8.

2.2.1 An analytic bound on the saturation number

Recall that the saturation number for a given α, β and K is the minimum value of N such that there exists a problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ with $L = \alpha \min(\beta, K)$. In particular, this implies that if we are able to construct a problem instance with N' files with a lower bound equal to $\alpha \min(\beta, K)$, then, $N_{sat}(\alpha, \beta, K) \leq N'$. In Algorithm 3, we create one such problem instance.

The basic idea of Algorithm 3 is as follows. The first part focuses on the construction of the tree, without labeling the leaves. For a given α and β, we first initialize a tree that just consists of a single edge (u^*, v^*). Following this, we partition α into two parts $\alpha_l = \lceil \alpha/2 \rceil$ and $\alpha_r = \alpha - \alpha_l$. On the other hand, β is split into $\beta_l = \lfloor \beta/2 \rfloor$ and $\beta_r = \beta - \beta_l$. The algorithm, then recursively constructs the left and right subtrees of u^*. It is important to note that the split in the (α, β) pair is done in such a manner that each subtree gets the floor and the ceiling of the one of the quantities. Moreover, the labeling of the cache node leaves is such that for a given node u, $|\mathbb{Z}(u_l) \cap \mathbb{Z}(u_r)|$ is as small as possible. The underlying reason for such a labeling is to ensure that the condition of Claim 3 doesn't hold for any $u \in \mathcal{T}$.

Following the construction of the tree, the second phase of the algorithm labels each of the delivery phase nodes, so that the computed lower bound is $L = \alpha\beta$. In this step we use $N = \alpha\beta$ files (see the procedure discussed in the proof of Corollary 1). In the third and final phase of the algorithm we modify the instance so that for any node $u \in \mathcal{T}$, we have that either $\Gamma_l \subseteq \Gamma_r$ or $\Gamma_r \subseteq \Gamma_l$; we use Claim 5 to achieve this. In the beginning all recovered files in the constructed instance are distinct so that $\Gamma(u_l) \cap \Gamma(u_r) = \emptyset$ for all nodes u. W.l.o.g. assume that $|\Gamma(u_r)| \leq |\Gamma(u_l)|$. An application of Claim 5 will thus cause a significant reduction in the number of files that are used. The following lemma quantifies this reduction.

Lemma 3 *For given α, β and K if $\beta \leq K$ then,*

$$N_{sat}(\alpha, \beta, K) \leq \left\lfloor \frac{2\alpha\beta + \alpha + \beta}{3} \right\rfloor.$$

Proof: We use Algorithm 3 to generate problem instance $P(\mathcal{T}, \alpha, \beta, L, \hat{N}_{sat}, K)$ so that $L = \alpha\beta$. By the definition of the saturation number we have $N_{sat}(\alpha, \beta, K) \leq \hat{N}_{sat}$ hence we just need to show that $\hat{N}_{sat} \leq \frac{2\alpha\beta+\alpha+\beta}{3}$.

First, we need to show that $L = \alpha\beta$. By line 32 of the algorithm the file $W_{(t-1)\beta+r}$ is recoverable in instance P_0 by the pair $(\mathbb{D}(v_t), \mathbb{Z}(v_{\alpha+r}))$ or equivalently $\Delta(v_t, v_{\alpha+r}) = W_{(t-1)\beta+r}$ for $1 \leq t \leq \alpha$ and $1 \leq r \leq \beta$. On the other hand, $\mathbb{W}(v^*) = \cup_{t=1}^{\alpha} \cup_{r=1}^{\beta} \Delta(v_t, v_{\alpha+r})$ therefore $\mathbb{W}(v^*) = \{W_1, \ldots, W_{\alpha\beta}\}$. Recall that $\mathbb{W}(v^*) = \cup_{u \in \mathcal{T}_0} W_{new}(u)$ and $L_0 = \sum_{u \in \mathcal{T}_0} |W_{new}(u)|$ so we have $L_0 \geq |\mathbb{W}(v^*)| = \alpha\beta$. But $L_0 \leq \alpha\beta$, by Corollary 2, therefore $L_0 = \alpha\beta$. In phase III of the Algorithm (Modify Delivery Phase Signals) using Claim 5, we have $L \geq L_0$ and since $L \leq \alpha\beta$ and $L_0 = \alpha\beta$ thus $L = \alpha\beta$.

W.l.o.g we set left incoming node such that $\Gamma(u_r) \subseteq \Gamma(u_l)$. Starting from the root node v^*, we let the set $\{u_0, u_1, \ldots, u_t\}$ and $\{w_0, \ldots, w_{t-1}\}$ to be the left and right incoming nodes respectively so that u_i is topologically higher than u_j for $i < j$, $u_t = u^*$ and u_0 to be a leaf. This is depicted in Figure 2.9. Recall that $\Gamma(u) = W_{new}(u) \cup \Gamma(u_l) \cup \Gamma(u_r)$ and $W_{new}(u) \cap (\Gamma(u_l) \cup \Gamma(u_r)) = \emptyset$ for any $u \in \mathcal{T}$. Therefore, recursively we have,

$$\begin{aligned} \hat{N}_{sat} &= |\Gamma(v^*)| = |\Gamma(u_t)|, \\ &= |W_{new}(u_t)| + |\Gamma(u_{t-1})|, \\ &= \sum_{i=1}^{t} |W_{new}(u_i)|, \end{aligned} \tag{2.11}$$

where we used $W_{new}(u_0) = \emptyset$ since u_0 is a leaf.

In Algorithm 3, $a(u)$ and $b(u)$ denote the number of delivery phase nodes and the number cache nodes, respectively in the subtree rooted at u. Note that by definition, we have

$$L = |W_{new}(u_t)| + \sum_{u \in \mathcal{T}_{u_{t-1}}} |W_{new}(u)| + \sum_{u \in \mathcal{T}_{w_{t-1}}} |W_{new}(u)|.$$

We conclude that $\sum_{u\in\mathcal{T}_{u_{t-1}}}|W_{new}(u)| \leq a(u_{t-1})b(u_{t-1})$ and $\sum_{u\in\mathcal{T}_{w_{t-1}}}|W_{new}(u)| \leq a(w_{t-1})b(w_{t-1})$ by using Corollary 2. Similarly, using Claim 6, we have that $|W_{new}(u_t)| \leq a(u_{t-1})b(w_{t-1}) + a(w_{t-1})b(u_{t-1})$. In fact, all these inequalities are met with equality. This can be seen as follows. An application of Claim 5 does not change the lower bound, which implies that $L = \alpha\beta = a(u_t)b(u_t)$. But, $a(u_t) = a(u_{t-1}) + a(w_{t-1})$ and $b(u_t) = b(u_{t-1}) + b(w_{t-1})$ so that

$$\begin{aligned} L = a(u_{t-1})b(w_{t-1}) + a(w_{t-1})b(u_{t-1}) \\ + a(u_{t-1})b(u_{t-1}) + a(w_{t-1})b(w_{t-1}). \end{aligned}$$

An inductive argument can be made to show a similar result for u_i, $i = 1, \ldots, t-1$.

Using these results and the equality in (2.11) yields,

$$\begin{aligned} \alpha\beta &= L, \\ &= \sum_{u\in\mathcal{T}} |W_{new}(u)|, \\ &= \sum_{i=0}^{t} |W_{new}(u_i)| + \sum_{i=0}^{t-1}\sum_{u\in\mathcal{T}_{w_i}} |W_{new}(u)|, \\ &= \hat{N}_{sat} + \sum_{i=0}^{t-1} \left(a(w_i)b(w_i)\right), \\ \Rightarrow \hat{N}_{sat} &= \alpha\beta - \sum_{i=0}^{t-1} a(w_i)b(w_i). \end{aligned} \tag{2.12}$$

Considering our setting for $a(u)$ and $b(u)$ in the line 9 of Algorithm 3 we have

$$a(u_{i+1}) = a(u_i) + a(w_i), \quad b(u_{i+1}) = b(u_i) + b(w_i), \tag{2.13}$$

for $0 \leq i \leq t-1$ and either

$$(a(u_i), b(u_i)) = (\lceil a(u_{i+1})/2\rceil, \lfloor b(u_{i+1})/2\rfloor) \quad \text{or} \quad (a(u_i), b(u_i)) = (\lfloor a(u_{i+1})/2\rfloor, \lceil b(u_{i+1})/2\rceil)\,.$$

In any case using eq. (2.13) we show in the following that $a(u_i) \leq a(w_i)+1$. By a similar argument we have $b(u_i) \leq b(w_i)+1$.

$$\begin{aligned} a(u_i) &\leq \lceil a(u_{i+1})/2 \rceil, \\ &\leq \frac{a(u_{i+1})+1}{2}, \\ &= \frac{a(u_i)+a(w_i)+1}{2}, \\ \Rightarrow a(u_i) &\leq a(w_i)+1. \end{aligned}$$

Using eq. (2.13) recursively, it is easy to see that $\alpha = a(u_0)+\sum_{i=0}^{t-1} a(w_i)$ and $\beta = b(u_0)+\sum_{i=0}^{t-1} b(w_i)$. Therefore, using eq. (2.12) and (2.11),

$$\begin{aligned} \hat{N}_{sat} &= \alpha\beta - \sum_{i=0}^{t-1} a(w_i)b(w_i), \\ &= \sum_{i=0}^{t-1} \left(a(u_i)b(w_i) + a(w_i)b(u_i)\right), \\ &\leq \sum_{i=0}^{t-1} \left([a(w_i)+1]b(w_i) + a(w_i)[b(w_i)+1]\right), \\ &\leq \sum_{i=0}^{t-1} \left(2a(w_i)b(w_i) + a(w_i) + b(w_i)\right), \\ &\leq \alpha+\beta+2\sum_{i=0}^{t-1} a(w_i)b(w_i), \\ \Rightarrow \sum_{i=0}^{t-1} a(w_i)b(w_i) &\geq \frac{\alpha\beta-\alpha-\beta}{3}. \end{aligned}$$

Finally, using the above inequality and eq. (2.12), we have

$$\begin{aligned} N_{sat}(\alpha,\beta,K) &\leq \hat{N}_{sat}, \\ &= \alpha\beta - \sum_{i=0}^{t-1} \alpha(w_i)\beta(w_i), \\ &\leq \alpha\beta - \frac{\alpha\beta-\alpha-\beta}{3} = \frac{2\alpha\beta+\alpha+\beta}{3}. \end{aligned}$$

Furthermore as $N_{sat}(\alpha,\beta,K)$ is an integer we conclude that

$$N_{sat}(\alpha,\beta,K) \leq \left\lfloor \frac{2\alpha\beta+\alpha+\beta}{3} \right\rfloor.$$

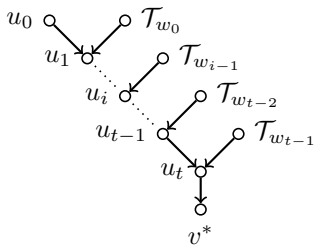


Figure 2.9: Saturation path

The aforementioned upper bound on the saturation number is tight. To see this, let consider $\beta = 1$. It is easy to see that $N_{sat}(\alpha, 1, K) = \alpha$ and using Lemma 3 we have $N_{sat} \leq \lfloor \alpha + 1/3 \rfloor = \alpha$.

2.2.2 Best lower bound for a fixed M

Theorem 1 and Corollary 2 characterize optimal and near-optimal problem instances for fixed α and β. In general, the best lower bound on the rate is obtained when we optimize over a range of choices for α and β. In our approach we restrict β to be less than $2K$, i.e., $\beta < 2K$. Our next result shows that an atomic problem instance with $\beta < 2K$ has $\alpha < 2N$. As a result, when $\beta < 2K$ the range of α, β pairs that we need to consider is limited.

Lemma 4 *Any problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ with $\beta < 2K$ and $\alpha \geq 2N$ is non-atomic.*

Proof: We let (u^*, v^*) to be the last edge in $\mathcal{T}$. If $\alpha \geq 2N$ then either $\alpha_l \geq N$ or $\alpha_r \geq N$ or both. W.l.o.g. we assume that $\alpha_l \geq N$. We note that $\beta_l < 2K$ as $\beta < 2K$. Claim 7 below shows that $N_{sat}(\alpha_l, \beta_l, K) \geq N$ for $\beta_l < 2K$. Therefore, $N \leq N_0 = \max\{N_{sat}(\alpha_l, \beta_l, K), N_{sat}(\alpha_r, \beta_r, K)\}$ and from (2.10) we have $|W_{new}(u^*)| = 0$. This implies that the problem is non-atomic.

Claim 7 *$N_{sat}(\alpha, \beta, K) \geq \alpha$ for any $\beta < 2K$.*

Proof: We use an inductive argument. Clearly, $N_{sat}(1, \beta', K) \geq 1$ for nonzero β' since at least one file must be used. Furthermore, by inspection we have $N_{sat}(\alpha', 1, K) = \alpha'$. Therefore, the base cases are established. Now, we assume that $N_{sat}(\alpha', \beta', K) \geq \alpha'$ for all $\alpha' \leq \alpha$ and $\beta' \leq \beta < 2K$.

We will first show that $N_{sat}(\alpha, \beta + 1, K) \geq \alpha$. Let $P(\mathcal{T}, \alpha, \beta + 1, L, N_s, K)$ be the problem instance associated with $N_{sat}(\alpha, \beta + 1, K)$ so that $N_s = N_{sat}(\alpha, \beta + 1, K)$ and $L = \alpha \min(\beta +$

$1, K)$. We also let (u^*, v^*) to be the last edge in $\mathcal{T}$ and P_l and P_r to be the problem instances corresponding to $\mathcal{T}_{u^*(l)}$ and $\mathcal{T}_{u^*(r)}$ respectively. By Claim 6, $|W_{new}(u^*)| \leq \rho(u^*) = \alpha_l[\min(\beta_r, K - \beta_l)]^+ + \alpha_r[\min(\beta_l, K - \beta_r)]^+$. We claim that $|W_{new}(u^*)| = \rho(u^*)$, $L_l = \alpha_l \min(\beta_l, K)$, and $L_r = \alpha_r \min(\beta_r, K)$ for problem instance P. This follows from the fact that $L = |W_{new}(u^*)| + L_l + L_r = \alpha \min(\beta + 1, K)$ and the limits on $|W_{new}(u^*)|$, L_l, and L_r discussed in Claim 6 and Corollary 1. The problem instances P_l and P_r are both saturated instances and each uses the minimum number of files. If this is not the case, replacing them in P with problem instances associated with $N_{sat}(\alpha_l, \beta_l, K)$ and $N_{sat}(\alpha_r, \beta_r, K)$ will result in a problem instance $P''(\mathcal{T}'', \alpha, \beta, L, N_s'', K)$ with $N_s'' < N_s$. But this contradicts our assumption that P is a problem instance associated with $N_{sat}(\alpha, \beta + 1, K)$. Thus we have $|\Gamma(v_l^*, v_l^*)| = N_{sat}(\alpha_l, \beta_l, K)$ and $|\Gamma(v_r^*, v_r^*)| = N_{sat}(\alpha_r, \beta_r, K)$. Then

$$\begin{aligned}
&N_{sat}(\alpha, \beta + 1, K) = |\Gamma(v^*, v^*)|, \\
&= |W_{new}(u^*) \cup \Gamma(u_l^*, u_l^*) \cup \Gamma(u_r^*, u_r^*)|, \\
&= |W_{new}(u^*)| + |\Gamma(u_l^*, u_l^*) \cup \Gamma(u_r^*, u_r^*)|, \\
&\geq |W_{new}(u^*)| + \max\left(N_{sat}(\alpha_l, \beta_l, K),\ N_{sat}(\alpha_r, \beta_r, K)\right), \\
&= \rho(u^*) + \max\left(N_{sat}(\alpha_l, \beta_l, K),\ N_{sat}(\alpha_r, \beta_r, K)\right). \qquad (2.14)
\end{aligned}$$

We note that we are guaranteed that either $[\min(\beta_r, K - \beta_l)]^+ \geq 1$ or $[\min(\beta_l, K - \beta_r)]^+ \geq 1$ or both must hold as $\beta + 1 \leq 2K$. Thus, we can assert that $\rho(u^*) \geq \min(\alpha_l, \alpha_r)$. Now, if we have $\beta_l > 0$ and $\beta_r > 0$, then using the induction hypothesis, we have $\max\left(N_{sat}(\alpha_l, \beta_l, K),\ N_{sat}(\alpha_r, \beta_r, K)\right) \geq \max(\alpha_l, \alpha_r)$ so that $N_{sat}(\alpha, \beta + 1, K) \geq \alpha$. On the other hand if w.l.o.g. $\beta_r = 0$, we have from eq. (2.14) that

$$\begin{aligned}
&N_{sat}(\alpha, \beta + 1, K) \\
&\geq (\alpha - \alpha_l) \min(\beta + 1, K) + N_{sat}(\alpha_l, \beta + 1, K), \\
&\geq \alpha - \alpha_l + N_{sat}(\alpha_l, \beta + 1, K).
\end{aligned}$$

One can argue recursively by considering the left and right branches of the instance associated with $N_{sat}(\alpha_l, \beta + 1, K)$ and arrive at the required result.

Next, we show that $N_{sat}(\alpha+1,\beta,K) \geq \alpha$. In this case, as before let (u^*,v^*) be the last node of the instance and let P_l and P_r to be the problem instances associated with $\mathcal{T}_{u^*(l)}$ and $\mathcal{T}_{u^*(r)}$ respectively. Now, if $\alpha_l > 0$ and $\alpha_r > 0$, then the induction hypothesis can be applied to conclude that $\max\left(N_{sat}(\alpha_l,\beta_l,K),\ N_{sat}(\alpha_r,\beta_r,K)\right) \geq \max(\alpha_l,\alpha_r)$ so that the result holds. On the other hand, if w.l.o.g. $\alpha_r = 0$, then we have from eq. (2.14) that

$$N_{sat}(\alpha+1,\beta,K) \geq N_{sat}(\alpha+1,\beta_l,K),$$

where $\beta_l < \beta$. One can recursively argue by examining the left and right branches of the instance associated with $N_{sat}(\alpha_l+1,\beta_l,K)$ and arrive at the required result, by using the fact that $N_{sat}(\alpha,1,K) \geq \alpha$ for any α.

The results for $N_{sat}(\alpha,\beta+1,K)$ and $N_{sat}(\alpha+1,\beta,K)$ can be used to show the corresponding result for $N_{sat}(\alpha+1,\beta+1,K)$ in a similar manner.

Thus far we have shown that the range of α is limited to $\alpha < 2N$ when β is limited to $\beta < 2K$. In fact, $\beta \geq 2K$ is a valid choice, though in our experiments it does not appear to yield any better lower bounds on the rate than the ones we have right now. If these choices of β are useful, they are likely to yield better lower bounds only in the regime when M is very small. The reason for this behavior is that for a fixed α the saturation number $N_{sat}(\alpha,\beta,K)$ takes maximum value at $\beta = K$ and starts decreasing once $\beta > K$.

Although Algorithm 3 is used to get an analytical upper bound on the saturation number, the exact saturation number $N_{sat}(\alpha,\beta,K)$ is recursively computable. It is not hard to see that the inequality in (2.14) is met with equality for a problem instance $P(\mathcal{T},\alpha,\beta,L,N_s,K)$ associated with the saturation number $N_{sat}(\alpha,\beta,K)$. This is a consequence of the fact that either $\Gamma_l \subseteq \Gamma_r$ or $\Gamma_r \subseteq \Gamma_l$. For a fixed α, β, and K, there are limited possibilities for $0 \leq \alpha_l \leq \alpha$ and $0 \leq \beta_l \leq \beta$. Corresponding to each possible (α_l,β_l) we can construct a saturated problem instance. This also includes the problem instance associated with the saturation number $N_{sat}(\alpha,\beta,K)$. Therefore, the following recurrence holds

$$N_{sat}(\alpha,\beta,K) = \min_{(\alpha_l,\beta_l)\in\mathcal{I}(\alpha,\beta)} \left\{ \rho(\alpha_l,\beta_l,\alpha,\beta) + \max\left(N_{sat}(\alpha_l,\beta_l,K),\ N_{sat}(\alpha-\alpha_l,\beta-\beta_l,K) \right) \right\},$$

where $\rho(\alpha_l, \beta_l, \alpha, \beta) = \alpha_l[\min(\beta - \beta_l, K - \beta_l)]^+ + (\alpha - \alpha_l)[\min(\beta_l, K + \beta_l - \beta)]^+$ and $\mathcal{I}(\alpha, \beta) = \{(a, b) : \ 0 \leq a \leq \alpha, \ 0 \leq b \leq \beta\} \setminus \{(0, 0), \ (\alpha, \beta)\}$. We note that $(\alpha_l, \beta_l) \in \{(0, 0), \ (\alpha, \beta)\}$ are trivial and we ignore those cases. Using this recurrence, Algorithm 4 computes the saturation number in time which is polynomial in the (α, β) pair (see the analysis in Appendix A.0.5).

Thus, the overall process of computing the lower bound on the rate for a fixed value of M proceeds as follows. We consider $1 \leq \alpha \leq 2N$ and $1 \leq \beta \leq 2K - 1$. For each (α, β) in this range, we consider all possible (α_l, β_l) and (α_r, β_r) pairs and compute the lower bound on $\alpha R^\star + \beta M$. This procedure requires us to precompute $N_{sat}(a, b, K)$ for $1 \leq a \leq 2N$ and $1 \leq b \leq 2K$. The precomputation step has time-complexity $O(N^2K^2)$ (see Appendix A.0.5). After this step, we start computing the lower bounds over all possible (α, β) pairs. For each value of (α, β), and for a specific (α_l, β_l) and (α_r, β_r) such that $\alpha_l + \alpha_r = \alpha$ and $\beta_l + \beta_r = \beta$, the complexity of computing the lower bound is $O(1)$ since we can use the characterization of Theorem 1 and the saturation numbers are precomputed. Thus, for a value of (α, β), the complexity of computing the bound is $O(\alpha\beta) \leq O(NK)$. As, we consider a total of NK values of (α, β) in total, the time-complexity of our procedure is $O(N^2K^2)$.

2.3 Multiplicative Gap Between Upper and Lower Bounds

We now show that for any set of problem parameters, our proposed lower bound and the achievable rate of Maddah-Ali and Niesen (2014) in eq. (2.2) are within a factor of four, i.e., we show the following result.

Theorem 2 *Consider a coded caching system with N files and K users each with a normalized cache size M. Then,*

$$\gamma(M) = \frac{R_c(M)}{R^\star(M)} \leq 4,$$

for $0 \leq M \leq N$.

The key idea in proving this result is to exploit the analytical upper bound on the saturation number $N_{sat}(\alpha, \beta, K)$ proposed in Section 2.2.1. For a given N and K, we consider three distinct

regions of M. For each range, an appropriate (α, β) pair allows us to obtain a lower bound on the rate that is within a factor of four of the achievable rate.

Proof:

We use Corollary 2 with the 2α and 2β, so that P_l' and P_r' have parameters α and β. This gives us the following lower bound.

$$2\alpha R^\star(M) + 2\beta M \geq \min\left(2\alpha \min(2\beta, K), 2\alpha\beta + [N - N_0]^+\right),$$

Moreover, we restrict $2\beta \leq K$ so that,

$$\begin{aligned} 2\alpha R^\star(M) + 2\beta M &\geq \min\left(4\alpha\beta, 2\alpha\beta + [N - N_0]^+\right) \\ \implies R^\star(M) &\geq \min\left(2\beta, \beta + \frac{[N - N_0]^+}{2\alpha}\right) - \frac{\beta}{\alpha} M. \end{aligned} \tag{2.15}$$

Our first observation is that for $\min(N, K) \leq 4$, the bound is easily seen to be true. Towards this end, by setting $\alpha = N, \beta = 1$ in (2.15), we obtain

$$R^\star(M) \geq 1 - \frac{M}{N}.$$

where we used $N_{sat}(N, 1, K) = N$. Furthermore, from eq. (2.2),

$$R_c(M) \leq \min(N, K)\,(1 - M/N),$$

This means that $\gamma(M) = \min(N, K) \leq 4$ for $\min(N, K) \leq 4$.

Thus, in the subsequent discussion, we only consider $\min(N, K) \geq 5$. As in Maddah-Ali and Niesen (2014), we divide the M-axis to three separated regions. For given M, we explore the space of (α, β) pairs to obtain an appropriate lower bound that allows us to show the multiplicative gap of four.

2.3.1 Region I: $0 \leq M \leq \max(1, N/K)$

First, we consider the range $0 \leq M \leq 1$. In eq. (2.15) we set $\alpha = 1, \beta = \lfloor \min(N,K)/2 \rfloor$. By such a setting we have $2\beta \leq \min(N,K) \leq K$ and $N \geq N_{sat}(1,\beta,K) = \beta$. Therefore for $M \leq 1$,

$$\begin{aligned}
R^\star(M) &\geq \min\left(2\beta, \frac{N+\beta}{2}\right) - \beta M \\
&\overset{(a)}{\geq} \min\left(\beta, \frac{N-\beta}{2}\right) \\
&\overset{(b)}{\geq} \min\left(\frac{\min(N,K)-1}{2}, \frac{N-\min(N,K)/2}{2}\right) \\
&\overset{(c)}{\geq} \min\left(\frac{\min(N,K)-1}{2}, \frac{\min(N,K)}{4}\right) \\
&\overset{(d)}{\geq} \frac{\min(N,K)}{4} \\
&\geq \frac{\min(N,K)(1-M/N)}{4} \\
&\geq R_c(M)/4.
\end{aligned}$$

Here, (a) holds since $M \leq 1$, (b) holds since $(\min(N,K)-1)/2 \leq \beta \leq \min(N,K)/2$, (c) holds since $N \geq \min(N,K)$, and (d) holds since $\min(N,K) \geq 2$.

Next, consider the range $M \in [1, N/K]$. Note that we only need to consider the scenario where $N \geq K$. The achievable rate $R_c(M)$ in this interval is upper bounded by the convex combination of the rates $R_c(0)$ and $R_c(N/K)$ so that

$$R_c(M) \leq \lambda R_c(N/K) + (1-\lambda)R_c(0) = K(1-\lambda/2) - \lambda/2, \text{ where } \lambda = KM/N.$$

Now, we set $\alpha = \lceil N/K \rceil, \beta = \lfloor K/2 \rfloor$ so that $\alpha\beta \leq (N/K+1)K/2 = N/2 + K/2 \leq N$. As, $N_{sat}(\alpha, \beta, K) \leq \alpha\beta$, this means that $N \geq N_{sat}(\alpha, \beta, K)$. In addition, note that $2\beta \leq K$. Therefore, we can use eq. (2.15) to obtain

$$\begin{aligned}
R^\star(M) &\geq \min\left\{2\beta,\ \beta + \frac{N - N_{sat}(\alpha,\beta,K)}{2\alpha}\right\} - \frac{\beta}{\alpha}M, \\
&\overset{(a)}{\geq} \min\left\{2\beta\left(1 - \frac{M}{2\alpha}\right),\ \frac{2\beta}{3} + \frac{N - 2\beta M}{2\alpha} - \frac{\beta}{6\alpha} - \frac{1}{6}\right\}, \\
&\overset{(b)}{\geq} \min\left\{(K-1)\left(1 - \frac{KM}{2N}\right),\ \frac{\beta}{2} + \frac{N - 2\beta M}{4N/K} - \frac{1}{6}\right\}, \\
&\geq \min\left\{\frac{K}{2}\left(1 - \frac{\lambda}{2}\right),\ \frac{\beta}{2}(1-\lambda) + \frac{K}{4} - \frac{1}{6}\right\}, \\
&\overset{(c)}{\geq} \min\left\{\frac{R_c(M)}{2},\ \frac{K}{2}\left(1 - \frac{\lambda}{2}\right) - \frac{(1-\lambda)}{4} - \frac{1}{6}\right\}, \\
&\overset{(d)}{\geq} \min\left\{\frac{R_c(M)}{2},\ \frac{R_c(M)}{4} + \frac{(K-3)}{4}\left(1 - \frac{\lambda}{2}\right) + \frac{1}{3}\right\}, \\
&\overset{(e)}{\geq} R_c(M)/4,
\end{aligned} \tag{2.16}$$

where in (a) we used Lemma 3 to bound $N_{sat}(\alpha, \beta, K)$, in (b) we used $N - 2\beta M \geq 0$, $1 \leq \alpha \leq N/K + 1 \leq 2N/K$, $(K-1)/2 \leq \beta$, and in (c) we used $\beta \geq (K-1)/2$, $\lambda = KM/N$ and the expression for the upper bound on $R_c(M)$ above. Next, (d) holds because of the achievable rate bound and (e) holds since $\min(N, K) \geq 5$. Therefore, $\gamma(M) \leq 4$ for $M \in [1, N/K]$ and $N \geq K$. Thus, we conclude that we have $\gamma(M) \leq 4$ for $M \in [0, \max(1, N/K)]$.

2.3.2 Region II: $\max(1, N/K) < M \leq N/2$

For M such that $\max(N/K, 1) < M \leq N/2$ we define $t_0 = \lfloor KM/N \rfloor$ so that $t_0 N/K < M \leq (t_0+1)N/K$. Since $M \geq N/K$ thus $t_0 \geq 1$. Using eq. (2.2), it turns out that,

$$\begin{aligned}
R_c(M) &\leq R_c(t_0 N/K), \\
&= \frac{K}{t_0+1} - \frac{t_0}{t_0+1}, \\
&\leq \frac{K}{KM/N} - \frac{1}{2}, \qquad \text{since } t_0 + 1 \geq KM/N \text{ and } t_0 \geq 1 \\
&= \frac{N}{M} - \frac{1}{2}.
\end{aligned}$$

Now, consider setting $\alpha = \lfloor 2M \rfloor$ and $\beta = \lfloor N/2M \rfloor$. With this setting we have $\alpha \geq 2$ (since $M \geq 1$), $\beta \geq 1$ (since $M \leq N/2$), and $\beta \leq N/2M < K/2$ (since $M > N/K$). Furthermore, since $\alpha\beta \leq 2M \times N/2M = N$ and $N_{sat}(\alpha, \beta, K) \leq \alpha\beta$ therefore $N \geq N_{sat}(\alpha, \beta, K)$. This together with $2\beta \leq K$ implies that such a setting allows the usage of (2.15). Therefore, using Lemma 3 to bound $N_{sat}(\alpha, \beta, K)$, we have

$$R^\star(M) \quad \geq \quad \min \left\{ 2\beta, \frac{2\beta}{3} + \frac{N}{2\alpha} - \frac{\beta}{6\alpha} - \frac{1}{6} \right\} - \frac{\beta}{\alpha} M.$$

We claim that $2\beta \geq 2\beta/3 + N/2\alpha - \beta/6\alpha - 1/6$ or equivalently $8\alpha\beta + \alpha + \beta \geq 3N$. This can be seen as follows. When, $N/4 < M \leq N/2$ we have $\alpha > N/2, \beta = 1$, so that this holds. On the other hand when $\max(1, N/K) < M \leq N/4$, we have $\alpha \geq 2M - 1$, $\beta \geq N/2M - 1$, so that $8\alpha\beta + \alpha + \beta \geq 8N - 7(N/2M + 2M) + 6$. It can been seen that $N/2M + 2M \leq N/2 + 2$ for $1 \leq M \leq N/4$ therefore $8\alpha\beta + \alpha + \beta \geq 9N/2 - 8 \geq 3N$ for $N \geq 6$. For $N = 5$, the claim trivially holds since $\alpha \geq 2, \beta \geq 1$ so that $8\alpha\beta + \alpha + \beta \geq 19 \geq 3 \times N = 15$.

Thus, we have

$$\begin{aligned}
R^\star(M) \quad &\geq \quad \frac{2\beta}{3} + \frac{N - 2\beta M}{2\alpha} - \frac{\beta}{6\alpha} - \frac{1}{6}, \\
&\overset{(a)}{\geq} \quad \frac{7\beta}{12} + \frac{N - 2\beta M}{4M} - \frac{1}{6}, \\
&= \quad \frac{N}{4M} + \frac{\beta}{12} - \frac{1}{6}, \\
&\overset{(b)}{\geq} \quad \frac{N}{4M} - \frac{1}{12}, \\
&\geq \quad \frac{N}{4M} - \frac{1}{8} \\
&\geq \quad \frac{R_c(M)}{4},
\end{aligned}$$

where in (a) we used $N - 2\beta M \geq 0$, $\alpha \geq 2$ and $\alpha \leq 2M$ and in (b) we used $\beta \geq 1$. Eventually, $\gamma(M) \leq 4$ for $\max(N/K, 1) \leq M \leq N/2$.

2.3.3 Region III: $N/2 < M \leq N$

Let $t_0 = \lfloor K/2 \rfloor$ so that $M \geq t_0 N/K$ for $M \in (N/2, N]$. For any $M \in (N/2, N]$ the convex combination of rate $R_c(t_0 N/K)$ and $R_c(N)$ gives us $R_c(M) \leq \lambda R_c(t_0 N/K) + (1 - \lambda) R_c(N) =$

$\lambda R_c(t_0 N/K)$ where $M = \lambda t_0 N/K + (1-\lambda)N$ or equivalently $\lambda = (1 - M/N)/(1 - t_0/K)$. According to this and eq. (2.2) we observe that,

$$\begin{aligned} R_c(M) &\leq \lambda R_c(t_0 N/K), \\ &= \frac{(1 - M/N)}{(1 - t_0/K)} \frac{(K - t_0)}{(t_0 + 1)}, \\ &= \frac{K(1 - M/N)}{(1 + t_0)}, \\ &\overset{(a)}{\leq} \frac{K(1 - M/N)}{K/2}, \\ &= 2(1 - M/N), \end{aligned}$$

where (a) holds since $1 + t_0 = 1 + \lfloor K/2 \rfloor \geq K/2$.

Now if we set $\alpha = N$ and $\beta = 1$ in (2.15) we obtain

$$\begin{aligned} R^\star(M) &\geq 1 - M/N \\ &\geq \frac{R_c(M)}{2}. \end{aligned}$$

This implies that $\gamma(M) \leq 2 \leq 4$ for $M \in [N/2, N]$ and concludes the proof.

2.4 Lower Bounds on the Other Variants of the Coded Caching Problem

In addition to the original coded caching problem there are many variants of the problem including coded caching with multiple requests Ji et al. (2014b), decentralized coded caching Maddah-Ali and Niesen (2015) and caching in device to device wireless networks Ji et al. (2013). Our proposed strategy applies with minor changes for these problems.

2.4.1 Caching in device to device wireless networks

Wireless device to device (D2D) networks where communication is limited to be single-hop are studied in Ji et al. (2013). There are K users who are the nodes of the network. Each user has a cache of size M and N files are stored across the different user caches. Thus, in this setting we necessarily have $KM \geq N$. As in the coded caching problem there are placement and delivery

phases. In the placement phase the caches are populated from a server; this phase does not depend on the user demands. The server then leaves the network. We let Z_i represent the cache content of the i-th user. In the delivery phase each user requests a file and the remaining users are informed about this request. Based on the requests, each user broadcasts a signal so that all demands can be satisfied. We denote by $X^{(i)}_{d_1,\ldots,d_K}$ the signal that is broadcasted in the delivery phase by the i-th user when the j-th user requests file $d_j \in [N]$ for $1 \leq j \leq K$. The delivery signal sent by each user is a function of its cache content so that $H(X^{(i)}_{d_1,\ldots,d_K}|Z_i) = 0$. We also denote by $X_{d_1,\ldots,d_K}$ the set of signals sent by all the users, i.e., $X_{d_1,\ldots,d_K} = \{X^{(1)}_{d_1,\ldots,d_K}, \ldots, X^{(K)}_{d_1,\ldots,d_K}\}$. The rate of the signal that the i-th user sends in the delivery phase is denoted by $R_{i,d_1,\ldots,d_K}(M)$. We are interested in lower bounding the worst case rate that denoted by $R^\star(M) = K \max_{i,d_1,\ldots,d_k} R_{i,d_1,\ldots,d_K}(M)$.

The cut-set technique and Han's inequality have been studied in Ji et al. (2013) and Sengupta and Tandon (2015) respectively to establish lower bounds on $R^\star(M)$. The multiplicative gap established in Ji et al. (2013) depends on M and is not constant, whereas Sengupta and Tandon (2015) shows a gap of at most 8.

The D2D setting is almost exactly the same as the coded caching setting studied in our work. Our technique for obtaining lower bounds is applicable here with essentially no change and we can use Theorem 1 and its corollary. Furthermore, since $H(X^{(i)}_{d_1,\ldots,d_K}|Z_i) = 0$ we can get lower bounds that are somewhat tighter. By treating $X_{d_1,\ldots,d_K}$ as the delivery signal of the original coded caching problem, we can apply our lower bound to show that the multiplicative gap between the achievable rate in Ji et al. (2013) and our proposed lower bounds is at most 4. The proof is quite similar to that of Theorem 2 and is omitted.

2.4.2 Coded caching with multiple requests

Coded caching with multiple requests is variation of the original problem in which each user requests l files from the server in the delivery phase. A straightforward achievable scheme in this setting is to apply the scheme of Maddah-Ali and Niesen (2014) l times. This problem is investigated in Ji et al. (2014b) where a new achievable scheme is proposed based on multiple groupcast index

coding. Furthermore, Ji et al. (2014b) introduces a cut-set type lower bound and shows that their scheme is within a multiplicative factor of 18 of the lower bound. In contrast, using our approach we can demonstrate a multiplicative gap of 4 for this problem as well.

In this setting the only difference with respect to the original problem is that from a cache signal Z_i and delivery signal $X_{d_1,\dots,d_K}$ one can recover up to l distinct files. Thus, d_i is a vector of size l containing information about the l files requested by i-th user. Therefore, all statements we presented for the original problem are applicable here, bearing in mind that $Rec(Z_i, X_{d_1,\dots,d_K})$ can be as large as l. For instance, an extension of eq. (2.8) gives us $L \leq l\alpha \min(\beta, K)$. Similarly, the saturation number $N_{sat}(\alpha, \beta, K, l)$ is defined as the minimum N' among all problem instances $P(\mathcal{T}, \alpha, \beta, L, N', K, l)$ with $L = l\alpha \min(K, \beta)$. It is easy to verify that $N_{sat}(\alpha, \beta, K, l) \leq l\alpha \min(\beta, K)$ in a similar way. The following claim can be shown (we omit the proof as it is very similar to the previous discussion).

Claim 8 *Consider a coded caching system with a server containing N files and K users. Each user has a cache of size M and demands l files in the delivery phase. The following lower bound holds for $N \geq N_0$ where $N_0 = N_{sat}(\alpha, \beta, K, l)$,*

$$\alpha R^\star(M) + \beta M \geq \min\Big\{2l\alpha \min(\beta, K),\ l\alpha \min(\beta, K) + (N - N_0)/2)\Big\}.$$

Similarly, an extension of the Lemma 3 holds so that $N_{sat}(\alpha, \beta, K, l) \leq l(2\alpha\beta + \alpha + \beta)/3$ for $\beta \leq K$. Exploiting this upper bound and Claim 8, we are able to show that the multiplicative gap of the straightforward achievable scheme and our lower bound is at most 4. Let $R_c^l(M) = lR_c(M)$ where $R_c(M)$ is defined in eq. (2.2).

Theorem 3 *Consider a coded caching system with a server containing N files and K users. Each user requests l files, and has a cache of size $0 \leq M \leq N$. Then*

$$\frac{R_c^l(M)}{R^\star(M)} \leq 4.$$

Proof: We divide the M axis into three regions, $0 \leq M \leq \max(l, N/K)$, $\max(l, N/K) \leq M \leq N/2$, and $N/2 \leq M \leq N$. In each region we show $R_c^l(M)/R^\star(M) \leq 4$ for any N and K. In the

following proof, $M = l$ plays the same role as $M = 1$ in proof of Theorem 2. Before embarking on the proof, we note that we only need to analyze the gap for $\min(N, lK) \geq 5$. Note that the lower bounds of the original problem are also valid here. Indeed, if each user requests the same file l times (instead of requesting l distinct files) the problem will be equivalent to the original one. Now, in (2.15) if we set $\alpha = N$ and $\beta = 1$ then we get $NR^\star + M \geq N$, or equivalently $R^\star(M) \geq (1 - M/N)$, which is applicable to the multiple request problem. Since $R_c^l(M) \leq \min(N, lK)(1 - M/N)$, therefore $R_c^l(M)/R^\star(M) \leq 4$ for $\min(N, lK) \leq 4$.

2.4.2.1 Region I: $0 \leq M \leq \max(l, N/K)$

For $0 \leq M \leq \max(l, N/K)$, we first show that the result holds for $M \leq l$. Since we separately analyze the gap for $M \geq N/2$ we assume $l \leq N/2$ so that $M \leq \max(l, N/K) \leq N/2$. We use result of the Claim 8 with setting $\alpha = 1$ and $\beta = \lfloor \min(N/2l, K/2) \rfloor$ where $\beta \geq 1$ from $l \leq N/2$. Following the exact same steps as in Section 2.3.1 for $M \leq 1$, it turns out that $R^\star(M) \geq \min(N, lK)/4 \geq R_c^l(M)/4$ for $M \leq l$.

Now, we assume that $l \leq M \leq \max(l, N/K)$ which is nonempty if $N/K \geq l$. Therefore, we only need to analyze the gap for $N \geq lK$ and $l \leq M \leq N/K$. In this range of M the convex combination of $M = 0$ and $M = N/K$ is achievable so that $R_c^l(M) \leq \lambda R_c^l(N/K) + (1 - \lambda) R_c^l(0)$. From $R_c^l(0) = lK$ and $R_c^l(N/K) = l(K - 1)/2$ we have $R_c^l(M) \leq lK(1 - \lambda/2) - l\lambda/2$ where $\lambda = KM/N$. By setting $\alpha = \lceil N/lK \rceil$ and $\beta = \lfloor K/2 \rfloor$, we have $\alpha\beta \leq \alpha K/2 \leq N/2l + K/2 \leq N/l$ (from $lK \leq N$) and that $N_{sat}(\alpha, \beta, K, l) \leq l\alpha\beta \leq N$.

This ensures that the setting is valid for using Claim 8. According to Claim 8 for such a setting we have,

$$\begin{aligned}
R^*(M) &\geq \min\left(2l\beta, l\beta + \frac{N - N_{sat}(\alpha,\beta,K,l)}{2\alpha}\right) - \frac{\beta M}{\alpha}, \\
&\overset{(a)}{\geq} \min\left(\frac{lK}{2}\left(1-\frac{\lambda}{2}\right), \frac{lK(1-\lambda/2)}{2} - \frac{l(1-\lambda)}{4} - \frac{l}{6}\right), \\
&\overset{(b)}{\geq} \min\left(\frac{R_c(M)}{2}, \frac{lK(1-\lambda/2)}{4} + \frac{l(1-\lambda/2)}{2} - \frac{l(5-3\lambda)}{12}\right), \\
&= \min\left(\frac{R_c(M)}{2}, \frac{lK(1-\lambda/2)}{4} + \frac{l}{12}\right), \\
&\geq \min\left(\frac{R_c(M)}{2}, \frac{R_c(M)}{4}\right) \geq \frac{R_c(M)}{4},
\end{aligned}$$

where inequality (a) can be obtained by making the same argument as we made in the first five lines of eq. (2.16) and (b) from $K \geq 2$.

2.4.2.2 Region II: $\max(l, N/K) \leq M \leq N/2$

In the first step, we try to get an upper bound on the achievable rate. Letting $t_0 = \lfloor KM/N \rfloor$ and following the argument we made in Section 2.3.2 gives us $R_c^l(M) \leq lR_c(M) \leq l\,(N/M - 1/2)$ for M in this range. Next, by setting $\alpha = \lfloor 2M/l \rfloor$ and $\beta = \lfloor N/2M \rfloor$ we have $N_{sat}(\alpha,\beta,K,l) \leq l\alpha\beta \leq N$ and $\beta \leq N/2M \leq K/2$ (since $M \geq N/K$) which imply that the constraints of the Claim 8 are satisfied. Therefore,

$$\begin{aligned}
R^\star &\geq \min\left(2l\beta,\ l\beta + \frac{N - N_{sat}(\alpha,\beta,K,l)}{2\alpha}\right) - \frac{\beta M}{\alpha}, \\
&\overset{(a)}{\geq} \min\left(2l\beta\left(1-\frac{M}{2l\alpha}\right),\ \frac{7l\beta}{12} + \frac{N-2\beta M}{2\alpha} - \frac{l}{6}\right), \\
&\overset{(b)}{\geq} \min\left(2l\beta\left(1-\frac{M}{2M}\right),\ \frac{7l\beta}{12} + \frac{N-2\beta M}{4M/l} - \frac{l}{6}\right), \\
&\overset{(c)}{\geq} \min\left(\frac{Nl}{4M},\ \frac{Nl}{4M} - \frac{l}{12}\right), \\
&\geq R_c^l(M)/4,
\end{aligned}$$

where in (a) we used upper bound on $N_{sat}(\alpha,\beta,K,l)$ and that $\beta/\alpha \leq \beta/2$ (from $\alpha \geq 2$), in (b) we used $N - 2\beta M \geq 0$, $\alpha \leq 2M/l$, and $\alpha \geq 2M/l - 1 \geq M/l$ (from $M \leq l$). In (c) we used $\beta \geq K/4$ (for $K \geq 2$) and $\beta \geq 1$ (from $M \leq N/2$).

2.4.2.3 Region III: $N/2 \leq M \leq N$

Using the same argument we made in Section 2.3.3 the achievable rate is bounded by $R_c^l(M) \leq lR_c(M) \leq 2l\,(1 - M/N)$. According to Claim 8 by setting $\alpha = \lfloor N/l \rfloor$ and $\beta = 1$ one may not recover all N files since $\alpha l \leq N$, but if we increase α to $\lceil N/l \rceil$ then all files will be recovered. Therefore $\alpha R^\star(M) + M \geq N$ or equivalently $R^\star(M) \geq (N - M)/\alpha$. From $N - M \geq 0$ and that $\alpha \leq N/l + 1 \leq 2N/l$ (since $l \leq N$) it turns out that $R^\star(M) \geq l(1 - M/N)/2 \geq 4R_c^l(M)$ for $N/2 \leq M \leq N$. This concludes the proof.

2.4.3 Decentralized coded caching

In the original coded caching problem the placement phase is managed by a central server. However, in many scenarios such coordinated placement phase may be impractical. Instead, a decentralized placement phase was investigated in Maddah-Ali and Niesen (2015) where the users cache random subsets of the bits of each file while respecting the cache size constraint. Even in this setting a multiplicative gap of 12 to the cut-set lower bound was obtained. Note that the lower bounds established for the centralized coded caching problem are also applicable to the decentralized case. By similar techniques to those used in proof of Theorem 2 we can establish a multiplicative gap of 4. The proof is omitted as it is quite similar.

2.5 Comparison with Existing Results

Lower bounds on the coding caching rate have been proposed in independent work as well. In this section we compare our lower bounds with other approaches.

2.5.1 Comparison with cutset bound

Our first observation is that the cutset bound in Maddah-Ali and Niesen (2014) is a special case of the bound in eq. (2.9). In particular, suppose that $\alpha = \lfloor N/s \rfloor$, $\beta = s$ for $s = 1, \ldots, \min(N, K)$. In this case, we have $\alpha\beta \leq N$. Thus, it is easy to construct a problem instance where $L = \alpha\beta$ (see Corollary 1). This also follows from observing that $N_{sat}(\alpha, \beta, K) \leq \alpha\beta$.

Our bound allows us to explore a larger range of (α, β) pairs that in turn lead to better lower bounds on $R^\star$. Suppose that for a coded caching system with N files and K users, we first apply the cutset bound with certain α_1 and β_1 such that $\alpha_1\beta_1 < N$. This would result in the inequality

$$\alpha_1 R^\star + \beta_1 M \geq \alpha_1\beta_1.$$

However, our approach can do strictly better. To see this note that $\alpha_1\beta_1 < N$ implies that $N_{sat}(\alpha_1, \beta_1, K) < N$. Now, using Corollary 2 we can instead attempt to lower bound $2\alpha_1 R^\star + 2\beta_1 M$ and obtain the following inequality.

$$\begin{aligned}
&2\alpha_1 R^\star + 2\beta_1 M \\
&\geq \min\left(4\alpha_1\beta_1,\ 2\alpha_1\beta_1 + N - N_{sat}(\alpha_1, \beta_1, K)\right) \\
&\implies \alpha_1 R^\star + \beta_1 M \\
&\geq \min\left(2\alpha_1\beta_1,\ \alpha_1\beta_1 + (N - N_{sat}(\alpha_1, \beta_1, K))/2\right),
\end{aligned}$$

which is strictly better than the cutset bound since $N - N_{sat}(\alpha_1, \beta_1, K) > 0$.

Example 9 *Consider a system containing a server with four files and three users, $N = 4$ and $K = 3$. The cutset bounds corresponding to the given system are*

$$4R^\star + M \geq 4,$$
$$2R^\star + 2M \geq 4, \text{ and}$$
$$R^\star + 3M \geq 3.$$

A simple calculation shows that if $M = 1$, the above inequalities, yield the lower bound $R^\star \geq 1$.

Now, consider the second bound, $2R^\star + 2M \geq 4$ and instead attempt to obtain a lower bound on $4R^\star + 4M$. In this case it can be verified that $N_{sat}(2, 2, 3) = 3 < N$. Using Corollary 2, this results in the lower bound $L^* \geq \min(4 \times 3, 2 \times 4 + 4 - N_{sat}(2, 2, 3)) = 9$. Thus we can conclude $R^\star + M \geq 2.25$ which is better than the cutset bound $R^\star + M \geq 2$. Moreover, this inequality also yields a better lower bound $R^\star \geq 1.25$.

2.5.2 Comparison with lower bound of Sengupta et al. (2015b)

The authors in Sengupta et al. (2015b) use Han's inequality (Cover and Thomas, 2012, Theorem 17.6.1) to establish the following lower bounds on the coded caching problem.

$$\alpha R^{\star}(M) + \beta M \geq N - \frac{\mu}{\mu+\beta}[N-\alpha\beta]^{+} - [N-\alpha K]^{+}, \tag{2.17}$$

where $\mu = \min(\lceil \frac{N-\alpha\beta}{\alpha} \rceil, K-\beta)$, $\beta \in \{1,\ldots,K\}$ and $\alpha \in \{1,\ldots,\lceil \frac{N}{\beta} \rceil\}$. This bound also provides more flexibility in the choice of α as compared to the cutset bound.

An analytical comparison between our bound and the bound in inequality (2.17) is hard, especially since a priori in all these bounds, for a given M, it is unclear which particular (α,β) pair gives the best lower bound. Thus, in the discussion below we attempt to analytically compare the bounds for given (α,β). We also present a numerical comparison in Section 2.5.5. The following conclusions can be drawn.

(a) Our bound is superior, when $1/\alpha + 1/\beta \leq 0.4$, i.e., when the values of α and β are large enough. Note that the best lower bounds on $R^{\star}(M)$ for systems with N and K reasonably large are obtained for higher values of α and β. Thus, for most parameter ranges our bounds are better.

(b) The bound in Sengupta et al. (2015b) is better when $\alpha = 1$ and $N \leq K$. This in turn means that their corresponding lower bound for small values of M is better than ours.

(c) We can demonstrate that our proposed lower bound is within a factor of four of the achievable rate, whereas Sengupta et al. (2015b) only demonstrates a multiplicative gap of eight.

In the remainder of this discussion we assume that $\alpha \geq 2$ and show these claims. Let L^{*} denote the value of our lower bound and let L_H denote the lower bound of Sengupta et al. (2015b).

Case 1: $\alpha\beta > N$.

Note that $\alpha \leq \lceil N/\beta \rceil$ in inequality (2.17). Furthermore, $\alpha \geq 2$ implies that $N \geq \beta$. Thus, we can conclude that $\alpha\beta \leq \lceil N/\beta \rceil \beta \leq 2N$.

Now, we use Corollary 2 to compare the bounds. Specifically, set $\alpha_l = \lceil \alpha/2 \rceil, \beta_l = \lfloor \beta/2 \rfloor, \alpha_r = \lfloor \alpha/2 \rfloor$ and $\beta_r = \lceil \beta/2 \rceil$. This implies that

$$\max(\alpha_l\beta_l, \alpha_r\beta_r) \leq \frac{\alpha\beta}{2} \leq N.$$

Thus, we obtain $L^* = \min\left(\alpha\beta,\ \alpha_l\beta_l + \alpha_r\beta_r + N - N_0\right)$. Note that

$$\begin{aligned} N_0 &= \max\left(N_{sat}(\alpha_l, \beta_l, K),\ N_{sat}(\alpha_r, \beta_r, K)\right), \\ &\leq \max(\alpha_l\beta_l, \alpha_r\beta_r) \leq N, \end{aligned}$$

using the arguments made above. Thus,

$$\begin{aligned} L^* &= \min\{\alpha\beta,\ \alpha_l\beta_l + \alpha_r\beta_r + N - N_0\} \\ &\geq \min\{\alpha\beta,\ \alpha_l\beta_l + \alpha_r\beta_r + N - \max\left(\alpha_l\beta_l,\ \alpha_r\beta_r\right)\} \\ &= \min\{\alpha\beta, \min\left(\alpha_l\beta_l,\ \alpha_r\beta_r\right) + N\} \\ &> N. \end{aligned}$$

On the other hand note that L_H is at most N. Thus, our bound is strictly better.

Case 2(a): $\alpha\beta \leq \alpha K \leq N$.

As $N \geq \alpha\beta \geq N_{sat}(\alpha, \beta, K)$ we use (2.15) to obtain

$$L^* = \min\left(\alpha \min(K, 2\beta),\ \alpha\beta + (N - N_0)/2\right).$$

The corresponding bound L_H is obtained by setting $\mu = K - \beta$.

$$\begin{aligned} L_H &= \alpha K - (1 - \beta/K)(N - \alpha\beta) \\ &= \alpha\beta(1 + 1/x - x) - (1 - x)N, \ \text{(where } 0 \leq x = \beta/K \leq 1) \\ &\leq \alpha\beta(2 - x), \ \text{(since, } N \geq \alpha K = \alpha\beta/x). \end{aligned}$$

Thus, we conclude that $L_H \leq \min(\alpha K, \alpha\beta(2 - x)) \leq \alpha \min(K, 2\beta)$. As a result, we only need to examine whether $\alpha\beta + (N - N_0)/2 \geq L_H$. Now, using the fact that $N_0 \leq (2\alpha\beta + \alpha + \beta)/3$, we have

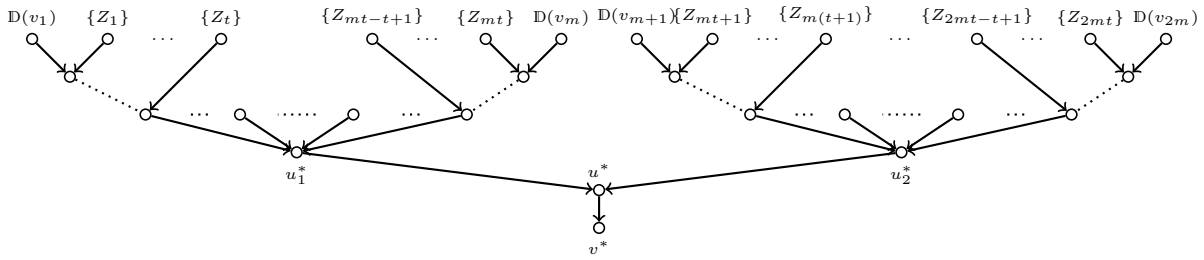


Figure 2.10: Problem instance associated with the lower bounds in Ajaykrishnan et al. (2015)

that $L^* \geq L_H$ when

$$\begin{aligned} 2\alpha\beta/3 + N/2 - (\alpha+\beta)/6 \;&\geq\; \alpha\beta(1 + 1/x - x) - (1-x)N \\ \implies (3/2 - x)N - (1/x + 1/3 - x)\alpha\beta - (\alpha+\beta)/6 \;&\geq\; 0. \end{aligned} \tag{2.18}$$

As $N \geq \alpha K = \alpha\beta/x$, inequality (2.18) certainly holds if

$$(1/2x + x - 4/3)\alpha\beta - (\alpha+\beta)/6 \geq 0.$$

It can be verified that $1/2x + x - 4/3 \geq \sqrt{2} - 4/3 \geq 1/15$ for $0 \leq x \leq 1$, so that the above inequality will definitely hold if $0.4 \geq 1/\alpha + 1/\beta$ which is the case for $\alpha, \beta \geq 5$.

Case 2(b): $\alpha\beta \leq N < \alpha K$.

In this case $\mu = \lceil N/\alpha - \beta \rceil$, so that

$$\begin{aligned} L_H &\leq N - (1 - \alpha\beta/N)(N - \alpha\beta) \\ &= \alpha\beta(2 - x') \text{ (where } 0 \leq x' = \alpha\beta/N \leq 1) \end{aligned}$$

As in the previous case, we conclude that $L^* \geq L_H$ if

$$2\alpha\beta/3 + N/2 - (\alpha+\beta)/6 \geq \alpha\beta(2 - x').$$

Upon analysis similar to the previous case, we can conclude that our bound is better when $0.4 \geq 1/\alpha + 1/\beta$.

2.5.3 Comparison with lower bound of Ajaykrishnan et al. (2015)

The work of Ajaykrishnan et al. (2015) is closest in spirit to our proposed lower bound. In particular, we show that their lower bound corresponds to specific problem instance as defined in our work. We note however that the work of Ajaykrishnan et al. (2015) does not analyze the multiplicative gaps between the achievable rates and lower bounds. The lower bounds in Ajaykrishnan et al. (2015) can be rewritten as

$$\begin{aligned} 2mR^\star + 2tmM &\geq L_0, \qquad \text{for } t \leq N,\ K \geq 2 \\ 2tmR^\star + 2mM &\geq L_0, \qquad \text{for } t \leq N,\ K \geq 2t, \end{aligned} \tag{2.19}$$

where $L_0 = \min\{4tm^2,\ 2tm^2 + N - \tilde{N}_0\}$, $\tilde{N}_0 = t(m^2 - m + 1)$, $m = n - \gamma$ and $n = \lceil (t + \sqrt{t^2 + 12t(N-t)})/6t \rceil$. Also, $\gamma = \max(0,\ \lceil n - K/2t \rceil)$ and $\gamma = \max(0,\ \lceil n - K/2 \rceil)$ in the first and second lower bounds respectively. We present these bounds using our notation so that (α, β) is equal to $(2m, 2tm)$ and $(2tm, 2m)$ in the first and second lower bounds in (2.19) respectively. Note however, that in the above bound the only free parameter is t, i.e., m itself is dependent on t. It is easy to see that $\beta \leq K$ therefore, unlike our method, this method cannot be used to obtain lower bounds when $\beta > K$.

The lower bound L_0 in eq. (2.19) above is reminiscent of our lower bound if the term $\tilde{N}_0$ is interpreted as a bound on the saturation number. In fact, for the specific setting of $(\alpha, \beta) = (m, mt)$, we can create a problem instance as described below, that is a saturated instance with exactly $t(m^2 - m + 1)$ files, so that we can infer that $N_{sat}(m, tm, K) \leq t(m^2 - m + 1)$. It turns out that this upper bound on the saturation number may be slightly stronger than the one we derived in Lemma 3 for general α and β when t and m are small. The associated problem instance of the first lower bound in (2.19) is depicted in Figure 2.10. The corresponding instance for the second lower bound in (2.19) can be derived in a similar manner. In this figure, delivery phase signals $\mathbb{D}(v_1), \ldots, \mathbb{D}(v_{2m})$ are same as the delivery phase signals defined in Ajaykrishnan et al. (2015). For this tree, it can be verified that the instance can be saturated with $t(m^2 - m + 1)$ files, so that $N_{sat}(m, tm, K) \leq t(m^2 - m + 1)$.

However, an application of Algorithm 3 will result in even better upper bound on the saturation number as shown in the example below. In particular, Algorithm 3 will generate a different tree when trying to upper bound the saturation number.

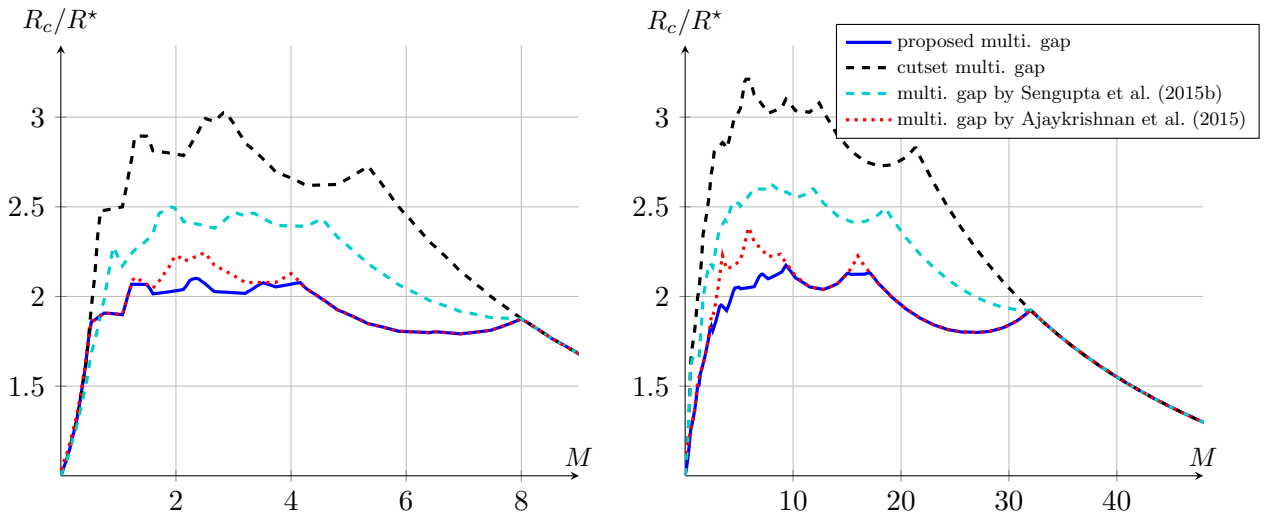
Example 10 *We consider a system with $N = 64$ files and $K = 8$ users and set $t = 2$ in eq. (2.19) so that $m = 4$ and $\tilde{N}_0 = 26$. Algorithm 4 for such a setting returns $N_{sat}(4, 8, 8) = 22$ which is smaller than $\tilde{N}_0$. On the other hand, it can be noted that in Figure 2.10, node u_1^* is such that it has $m = 4$ incoming edges which makes the corresponding lower bound looser (cf. Claim 1).*

2.5.4 Comparison with results in Tian (2015)

Reference Tian (2015) presents lower bounds for the specific case of $N = K = 3$. The inequalities are generated via a computational technique that works with the entropic region of the associated random variables. Some of the bounds presented in Tian (2015) can be obtained via our approach as well. However, the specific inequalities $3R^\star + 6M \geq 8$, $18R^\star + 12M \geq 29$ and $6R^\star + 3M \geq 8$ cannot be obtained using our approach and strictly improves our region. Note however, that it is not clear whether these inequalities can be obtained in a computationally tractable manner for the case of large N and K.

2.5.5 Numerical comparison of the various bounds

We conclude this section, by providing numerical results for two cases: (i) $N = 16, K = 30$ and (ii) $N = 64, K = 50$. In Figure 2.11 the ratio $R_c(M)/R^\star(M)$ is plotted by lower bounding $R^\star(M)$ by different methods. In case I (see Figure 2.11) we have $N = 16$ and $K = 30$. Our bound has the minimum multiplicative gap except in the small range $0 \leq M \leq 1$. Specifically, as discussed previously, the bound in Sengupta et al. (2015b) is better than ours when $K \geq N$ and $\alpha = 1$ and $0 \leq M \leq 1$. In case II, where $N > K$ our bound has minimum multiplicative gap for all range of M.



(a) Case I: $N = 16$, $K = 30$

(b) Case II: $N = 64$, $K = 50$

Figure 2.11: The plot demonstrates the multiplicative gap between the achievable rate, $R_c(M)$, in Maddah-Ali and Niesen (2014) and lower bounds $R^\star(M)$ using different lower bounding techniques. For case II our lower bound results in the least multiplicative gap. In case I, where $N \leq K$, the multiplicative gap obtained by our proposed lower bound is lower than the others for $M \geq 1$. In the range $0 \leq M \leq 1$, Sengupta et al. (2015b) provides a slightly better result.

2.6 Conclusions and Future Work

In this work we considered a coded caching system with N files, K users each with a normalized cache of size M. We demonstrated an improved lower bound on the coded caching rate $R^\star(M)$. Our approach proceeds by establishing an equivalence between a sequence of information inequalities and a combinatorial labeling problem on a directed tree. Specifically, for given positive integers α and β, we generate an inequality of the form $\alpha R^\star + \beta M \geq L$. We showed that the best L that can be obtained using our approach is closely tied to how efficiently a given number of files can be used by our proposed algorithm. Formalizing this notion, we studied certain structural properties of our algorithm that allow us to quantify the improvements that our approach affords. In particular, we show a multiplicative gap of four between our lower bound and the achievable rate. An interesting feature of our algorithm is that it is applicable for general value of N, K and M and is strictly better than all prior approaches for most parameter ranges.

There are still gaps between the currently known lower bounds and the achievable rate and an immediate open question is whether this gap can be reduced or closed. It would also be of interest to better understand coded caching rates for more general network topologies.

Algorithm 3 Instance construction for upper bounding $N_{sat}(\alpha, \beta, K)$

Input: α, β and K.

Initialization
  Let (u^*, v^*) be last edge and set $U_{new} = \{u^*\}$.
  Set $\mathbb{Z}(u^*) = \{Z_1, Z_2, \ldots, Z_{\min(\beta,K)}\}$ and $b(u^*) = \beta$, $a(u^*) = \alpha$.
  $\mathcal{C} = \emptyset$ and $\mathcal{D} = \emptyset$.
end Initialization
procedure Tree Construction & Cache nodes labeling
  while U_{new} is nonempty **do**
    Pick $u \in U_{new}$, create nodes u_l and u_r, edges (u_l, u) and (u_r, u), add them to $\mathcal{T}_0$.
    Set $a(u_l) = \lceil a(u)/2 \rceil$, $b(u_l) = \lfloor b(u)/2 \rfloor$ and $a(u_r) = a(u) - a(u_l)$, $b(u_r) = b(u) - b(u_l)$.
    Set $\mathbb{Z}(u_l)$ and $\mathbb{Z}(u_r)$ be subsets of $\mathbb{Z}(u)$ of sizes $\min(b(u_l), K)$ and $\min(b(u_r), K)$ respectively with minimum intersection.
    Remove u from U_{new}.
    if $a(u_l) + b(u_l) \geq 2$ **then**
      Add u_l to U_{new}.
    else
      If $b(u_l) == 1$ add u_l to $\mathcal{D}$ otherwise to $\mathcal{C}$.
    end if
    if $a(u_r) + b(u_r) \geq 2$ **then**
      Add u_r to U_{new}.
    else
      If $b(u_r) == 1$ add u_r to $\mathcal{D}$ otherwise to $\mathcal{C}$.
    end if
  end while
end procedure
procedure Delivery nodes labeling
  Let $\mathcal{D} = \{v_1, \ldots, v_\alpha\}$.
  for $r = 1, \ldots, \min(\beta, K)$ **do**
    Pick a node $v \in \mathcal{C}$ with $\mathbb{Z}(v) = \{Z_r\}$ and denote it by $v_{r+\alpha}$.
  end for
  Let $\mathcal{C} \setminus \{v_{\alpha+1}, \ldots, v_{\alpha+\min(\beta,K)}\} = \{v_{\alpha+\min(\beta,K)+1}, \ldots, v_\beta\}$.
  for $t = 1, \ldots, \alpha$ **do**
    for $r = 1, \ldots, \min(\beta, K)$ **do**
      $d_r = (t-1)\min(\beta, K) + r$.
    end for
    for $r = \min(\beta, K) + 1, \ldots, K$ **do**
      $d_r = 1$.
    end for
    Set $\mathbb{D}(v_t) = X_{d_1, \ldots, d_K}$
  end for
end procedure
procedure Modify Delivery phase signals
  Denote current instance by $P_0(\mathcal{T}_0, \alpha, \beta, L_0, N_0, K)$.
  Modify $P_0(\mathcal{T}_0, \alpha, \beta, L_0, N_0, K)$ by Claim 5 to obtain $P(\mathcal{T}, \alpha, \beta, L, \hat{N}_{sat}, K)$.
end procedure

Output: $\hat{N}_{sat}(\alpha, \beta, K) = |\Gamma(v^*)|$, $P(\mathcal{T}, \alpha, \beta, L, \hat{N}_{sat}, K)$.

Algorithm 4 Computing saturation number $N_{sat}(\alpha, \beta, K)$

Input: α, β and K.

Initialization:

1: For all $a \in \{0, \ldots, \alpha\}$ and $b \in \{0, \ldots, \beta\}$ set

$$\begin{aligned} N_{sat}(a, 0, K) &= 0, & N_{sat}(0, b, K) &= 0, \\ N_{sat}(a, 1, K) &= a, & N_{sat}(1, b, K) &= \min(b, K). \end{aligned}$$

Main loop:

2: **for** $a = 2$; $a \leq \alpha$; $a++$ **do**

3: **for** $b = 2$; $b \leq \beta$; $b++$ **do**

4:

$$N_{sat}(a, b, K) = \min_{(\tilde{a}, \tilde{b}) \in \mathcal{I}(a,b)} \left\{ \rho(\tilde{a}, \tilde{b}, a, b) + \max\left(N_{sat}(\tilde{a}, \tilde{b}, K),\ N_{sat}(a - \tilde{a}, b - \tilde{b}, K) \right) \right\}$$

5: **end for**

6: **end for**

Output: $N_{sat}(\alpha, \beta, K)$

CHAPTER 3. ASYNCHRONOUS CODED CACHING

Caching is a core component of solving the problem of large scale content delivery over the Internet. Conventional caching typically relies on placing popular content closer to the end users. Statistically, popular content is requested more frequently and the cache can be used to serve the user requests in this case. Contacting the central server that has all the content is not needed. This serves to reduce the induced network traffic.

In their pioneering work Maddah-Ali and Niesen (2014), Maddah-Ali and Niesen considered the usage of coding in the caching problem. In this so-called "coded caching" setting, there is a server containing N files, K users each with a cache that can store up to M files. The users are connected to the server via an error-free shared link (see Figure 3.1). The system operates in two distinct phases. In the *placement phase* the content of the caches is populated by server. This phase does not depend on the future requests of the users which are assumed to be arbitrary. In the *delivery phase* each user makes a request and the server transmits potentially coded signals to satisfy the requests of the users. The work of Maddah-Ali and Niesen (2014) demonstrated that significant reductions in the network traffic were possible as compared to conventional caching. Crucially, these gains continue to hold even if the popularity of the files is not taken into account.

While this is a significant result, the original formulation of the coded caching problem assumes that the user requests are synchronized, i.e., all file requests from the users arrive at the server at the same time. Henceforth, we refer to this as the synchronous setting. From a practical perspective, it is important to consider the asynchronous setting where user requests arrive at different times. In this case, a simple strategy would be to wait for the last request to arrive and then apply the scheme of Maddah-Ali and Niesen (2014). Such a strategy will be quite good in terms of the overall rate of transmission from the server. However, this may be quite bad for an end user's experience,

e.g., the delay experienced by the users will essentially be dominated by the arrival time of the last request.

In this chapter we formulate and study the coded caching problem when the user requests arrive at different times. Each user has a specific deadline by which his/her demand needs to be satisfied. The goal is to schedule transmission of packets so that each user is able to recover the requested file from the transmitted packets and his/her cache content within the prescribed deadline. We present algorithms for both the offline and online versions of this problem.

This chapter is organized as follows. In Section 3.1 we discuss the background and related work and overview our main contributions. The problem formulation appears in Section 3.2. Sections 3.3 and 3.4 discuss our work on the offline and the online versions of the problem, respectively. We conclude the paper with a discussion of opportunities for future work in Section 3.6.

3.1 Background, Related Work and Summary of Contributions

A coded caching system contains a server with N files, denoted W_n, $n = 1, \ldots, N$, each of size F subfiles, where a subfile is a basic unit of storage. The system also contains K users each connected to the server through an error free, broadcast shared link. Each of the users is equipped with a local cache. The i-th cache is of size $M_i F$ subfiles. We denote the cache content of user i by Z_i, where Z_i is a function of $W_1, \ldots, W_N$. Our formulation supports users with different cache sizes. A block diagram of a coded caching system is depicted in Figure 3.1.

In this work, we assume that an uncoded placement scheme is being used by the coded caching system, i.e., user $i \in [K]$ caches at most an $M_i F$-sized subset of the total number of subfiles in the server. It is well recognized that the delivery phase in this case corresponds to an index coding problem Arbabjolfaei et al. (2018). While the optimal solution for an arbitrary index coding problem is known to be hard, techniques such as clique cover on the side information graph are well-recognized to have good performance Arbabjolfaei et al. (2018). In this case each transmitted equation from the server is such that a certain number of users "benefit" from it simultaneously. Under this assumption, we formulate and study the asynchronous coded caching problem when the

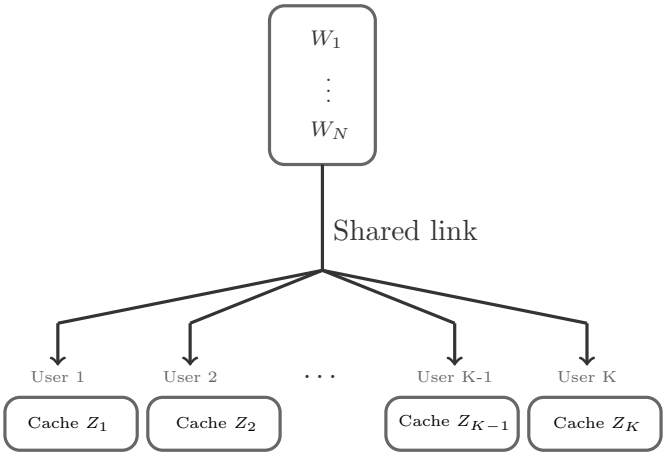


Figure 3.1: Block diagram of the coded caching system.

file requests arrive at the server at different times. Each user specifies a deadline by which he/she expects the request to be satisfied[1]. We assume that

- the delivery phase proceeds via a clique cover and
- transmitting a single packet over the shared link takes a certain number of time slots.

We study the rate gains of coded caching under this setup, i.e., among the class of strategies that allow the users to meet their deadlines, we attempt to determine those where the server transmits the fewest number of packets. Both the offline and the online versions of the problem are studied. In the offline scenario, we assume that information about all request arrival times and deadlines are known to the server before transmission, whereas in the online scenario, the arrival times and deadlines are revealed to the server as time progresses.

3.1.1 Main contributions

- *Linear programming (LP) formulation in the offline case.* We propose a LP in the offline scenario that determines a schedule for the equations transmitted from the server. If feasible,

[1]It is not too hard to see that in the absence of deadlines the server can simply wait for enough user requests to arrive before starting transmission. Thus, the deadline-free case essentially reduces to the synchronous setting.

the schedule is such that each user meets its deadlines and the rate of transmission from the server is minimized. This allows us to study the effect of asynchronism on the coded caching rate. We demonstrate that a feasible point of the LP can be interpreted as a coding solution that can be used by the server.

The computational complexity of solving this LP can be quite high for a large number of users. Accordingly, we develop a dual decomposition technique where the dual problem decouples into a set of independent minimum cost network flow problems which can be solved in an efficient manner Ahuja et al. (1993).

- *A novel online algorithm.* For the online problem we demonstrate that in general coding within subfiles of the same file is essential. Interestingly, in the synchronous case this is not needed. We propose an novel online algorithm that is inspired by recursively solving the offline LP and interpreting the corresponding output appropriately. Under certain condition, we also show that the algorithm will result in a solution that satisfies the deadline constraints with high probability.

For both scenarios we present exhaustive simulation results that corroborate our findings and demonstrate the superiority of our algorithm with respect to prior work. Overall, our work indicates that under mild asynchronism, much of the benefits of coded caching can still be leveraged.

3.1.2 Related work

The area of coded caching has seen a flurry of research activity along several dimensions in recent years. From a theoretical perspective, significant work has attempted to understand the fundamental rate limits of a coded caching system Ghasemi and Ramamoorthy (2017c); Yu et al. (2017); Yu et al. (2019). Extensions of the basic model to general networks have been examined in Tang and Ramamoorthy (2016a); Naderializadeh et al. (2017); Wan et al. (2018). Issues related to subpacketization (i.e., the number of subfiles F) have been considered in Yan et al. (2017); Tang and Ramamoorthy (2018); Lampiris and Elia (2018). A high subpacketization level can cause several issues in practical implementations.

Our work builds on our initial contributions in Ghasemi and Ramamoorthy (2017b) and Ghasemi and Ramamoorthy (2017a). In Ghasemi and Ramamoorthy (2017b,a), we presented a preliminary version of the offline formulation and the dual decomposition method. The current paper includes a more compact representation of the associated optimization problems, exhaustive simulation results and all the proofs. Moreover, the part dealing with online algorithms has not appeared in prior work.

There are relatively few prior works that have considered asynchronism within the context of coded caching. To our best knowledge, it was first studied in Niesen and Maddah-Ali (2015). They considered the decentralized coded caching model Maddah-Ali and Niesen (2015), and considered a situation where each subfile has a specific deadline. Only the online case was considered and heuristics for transmission from the server were proposed. The heuristics are found to have good performance. However, the transmission time for a packet was not considered in their formulation. Reference Lu et al. (2018) also considers the asynchronous setting; again, they do not consider the transmission time of a transmitted packet. In that sense, their setting is closer to the work of Niesen and Maddah-Ali (2015) and can be viewed as a set of rules that the server should follow in the online case. Lu et al. (2018) (Section III.C) also considers an offline setting for the centralized placement scheme of Maddah-Ali and Niesen (2014). In contrast, our LP formulation can be viewed as a bound on the possible performance of any online scheme. Our proposed online algorithm has significantly better performance than the ones presented in Niesen and Maddah-Ali (2015).

Reference Maddah-Ali and Niesen (2015) (Section V.C) also discusses the issue of asynchronism within the context of decentralized coded caching, without considering deadlines or packet transmission times. They advocate a further subpacketization of each subfile (referred to as a segment in Maddah-Ali and Niesen (2015)). It is important to note that any system will need to commit to a certain subpacketization scheme before deployment. Given this subpacketization and with user specified deadlines, the formalism of our work and our algorithms can be used to arrive at schemes that address asynchronous requests.

The work of Jiang et al. (pear) proposes an algorithm for the online scenario under the assumption of decentralized coded caching for reducing the worst-case load of fronthaul links in fog radio access networks (F-RANs); this is a different model than ours. Their work does not take transmission time into account and considers the scenario where each user has the same deadline.

The asynchronous setting has also been considered in Yang et al. (2019) for video delivery by taking into account an appropriately defined audience retention rate. Their work considers a probabilistic arrival model and presents a decentralized coded caching scheme for it.

3.2 Problem Formulation and Preliminaries

We assume that time $\tau \geq 0$ is slotted. Let $[n]$ denote the set $\{1, \ldots, n\}$ and the symbol $\oplus$ represent the XOR operation. We assume that the server contains $N \geq K$ files[2] denoted by $W_n, n = 1, \ldots, N$. The subfiles are denoted by $W_{n,f}$ so that $W_n = \{W_{n,f} : f \in [F]\}$ and the cache of user i by $Z_i \subseteq \{W_{n,f} : \ n \in [N], \ f \in [F]\}$. Z_i contains at most $M_i F$ subfiles. In the delivery phase, user i requests file W_{d_i}, where $d_i \in [N]$, from the server. We let $\Omega^{(i)}$ denote the indices of the subfiles that are not present in the i-th user's cache, i.e.,

$$\Omega^{(i)} = \{f : \ f \in [F], \ W_{d_i,f} \notin Z_i\}.$$

The equations in the delivery phase are assumed to be of the *all-but-one* type.

Definition 10 *All-but-one equation. Consider an equation E such that*

$$E = \oplus_{l=1}^{\ell} W_{d_{i_l},f_l}.$$

We say that E is of the all-but-one type if for each $l \in [\ell]$, we have $W_{d_{i_l},f_l} \notin Z_{i_l}$ and $W_{d_{i_l},f_l} \in Z_{i_k}$ for all $k \in [\ell] \setminus \{l\}$.

[2] We assume that $N \geq K$ as it corresponds to the worst case rate where each of the K users can request a different file. Furthermore, it is also the more practical scenario.

It is evident that an all-but-one equation transmitted from the server allows each of the users participating in the equation to recover a missing subfile that they need. The asynchronous coded caching problem can be formulated as follows.

Inputs.

- *User requests.* User i requests file W_{d_i}, with $d_i \in [N]$ at time T_i.
- *Deadlines.* The i-th user needs to be satisfied by time $T_i + \Delta_i$, where Δ_i is a positive integer.
- *Transmission delay.* Each subfile needs r time-slots to be transmitted over the shared link, i.e., each subfile can be treated as equivalent to r packets, where each packet can be transmitted in one time slot.

As the problem is symmetric with respect to users, w.l.o.g. we assume that $T_1 \leq T_2 \leq \ldots \leq T_K$. Let $T_{\max} = \max_i(T_i + \Delta_i)$. Note that upon sorting the set of arrival times and deadlines, i.e., $\cup_{i=1}^{K}\{T_i, T_i+\Delta_i\}$, we can divide the interval $[T_1, T_{\max})$ into *at most* $2K-1$ non-overlapping intervals. Let the integer β, where $1 \leq \beta \leq 2K-1$ denote the number of intervals. Let $\Pi_1, \ldots, \Pi_\beta$ represent the intervals where Π_i appears before Π_j if $i < j$; $|\Pi_\ell|$ denotes the length of interval Π_ℓ . The intervals are left-closed and right-open. An easy to see but very useful property of the intervals that we have defined is that for a given i, either $[T_i, T_i + \Delta_i) \cap \Pi_\ell = \Pi_\ell$ or $[T_i, T_i + \Delta_i) \cap \Pi_\ell = \emptyset$. Figure 3.2 shows an example when $K = 3$. We define $U_\ell = \{i \in [K] : \ [T_i, T_i + \Delta_i) \cap \Pi_\ell = \Pi_\ell\}$, and $D_\ell = \{d_i \in [N] : \ i \in U_\ell\}$. Thus, U_ℓ is the set of active users in time interval Π_ℓ and D_ℓ is the corresponding set of active file requests.

Outputs.

- *Transmissions at each time slot.* If the problem is feasible, the schedule specifies which equations (of the all-but-one type) need to be transmitted at each time. The schedule is such that each user can recover all its missing subfiles within its deadline. The equations transmitted at time $\tau \in \Pi_\ell$ only depend on D_ℓ.

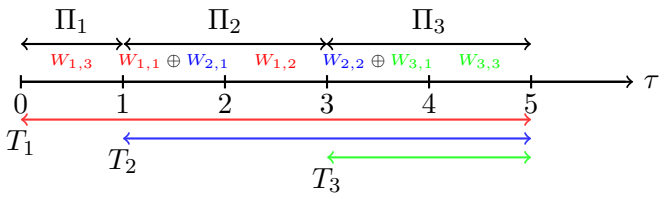


Figure 3.2: Offline solution corresponding to the Example 11. The double-headed arrows show the active time slots for each user. The transmitted equations are shown above the timeline.

We consider two versions of the above problem, namely *offline* and *online* version of the asynchronous coded caching problem.

- *Offline version.* In the offline version, we assume that the server is aware of $\{T_i, \Delta_i, d_i\}_{i=1}^{K}$ at $\tau = 0$. However, at time $\tau \in \Pi_\ell$ the transmitted equation(s) will only depend on D_ℓ, i.e., the server cannot start sending missing subfiles for a given user until its request arrives.

- *Online version.* In the online version, information about the file requests are revealed to the server as time progresses. At each time τ, the server only has information about $\{T_i, \Delta_i, d_i\}$ if $T_i \leq \tau$, i.e., the requests that have arrived by time τ.

We begin by defining some relevant sets; for convenience, a tabulated list of most of the items needed in the subsequent sections can be found in Table 3.1. Consider a subset of users $U \subseteq [K]$. For each user $i \in U$ we let $\mathcal{F}_{\{i,U\}}$ denote the indices of all missing subfiles of the i-th user that have been stored in the cache of the other users in U, i.e.,

$$\mathcal{F}_{\{i,U\}} = \left\{ f \in \Omega^{(i)} : \ W_{d_i,f} \in Z_j \text{ for all } j \in U \setminus \{i\} \right\}.$$

Definition 11 *User Group. A subset $U \subseteq [K]$ is said to be a user group if $\mathcal{F}_{\{i,U\}} \neq \emptyset$ for all users $i \in U$ so that there is at least one all-but-one type equation associated with U.*

We note that for a user group U there are $\prod_{i \in U} |\mathcal{F}_{\{i,U\}}|$ different all-but-one equations. Recall that U_ℓ is the set of active users in time interval Π_ℓ and D_ℓ represents their file requests. Let $\mathcal{U}_\ell$ be

a subset of the power set of U_ℓ (i.e. the set of all subsets of U_ℓ) such that each element in $\mathcal{U}_\ell$ is an user group (*cf.* Definition 11). For any $U \subseteq [K]$, let $\mathcal{I}_U$ be the set of indices of all time intervals where the users in U are simultaneously active, i.e.,

$$\mathcal{I}_U = \left\{ \ell : \ [T_i, T_i + \Delta_i) \cap \Pi_\ell = \Pi_\ell, \ \forall \ i \in U \right\}.$$

For each missing subfile $W_{\{d_i,f\}}$ (where $f \in \Omega^{(i)}$) we let $\mathcal{U}_{\{i,f\}}$ denote the set of user groups where it can be transmitted, i.e.,

$$\mathcal{U}_{\{i,f\}} = \left\{ U \in \cup_{\ell=1}^{\beta} \mathcal{U}_\ell : \ i \in U, f \in \mathcal{F}_{\{i,U\}} \right\}.$$

We note here that for a fixed i, there are potentially multiple indices $f_1, f_2, \ldots, f_l \in \Omega^{(i)}$ such that $U \in \mathcal{U}_{\{i,f_j\}}$ for $j = 1, \ldots, l$.

Example 11 *Consider a system (shown in Figure 3.2) with $N = 3$ files, W_1, W_2, and W_3 where each file is divided into three subfiles, so that $F = 3$. There are $K = 3$ users with the following cache content, $Z_1 = \{W_{2,1}, W_{2,2}, W_{3,3}\}$, $Z_2 = \{W_{1,1}, W_{2,3}, W_{3,1}\}$, and $Z_3 = \{W_{2,2}, W_{3,2}, W_{2,1}\}$. Thus, $M_i = 1$ for $i \in [K]$. The arrival times are $T_1 = 0$, $T_2 = 1$, $T_3 = 3$, and deadlines are $\Delta_1 = 5$, $\Delta_2 = 4$, and $\Delta_3 = 2$. The i-th user requests file W_i, for $i = 1, \ldots, 3$. Therefore, $\Omega^{(1)} = \{1, 2, 3\}$, $\Omega^{(2)} = \{1, 2\}$, and $\Omega^{(3)} = \{1, 3\}$.*

In this system we have $\mathcal{F}_{\{1,\{1,2\}\}} = \{1\}$ as $W_{1,1} \in Z_2$, and $\mathcal{F}_{\{2,\{1,2\}\}} = \{1, 2\}$ as $W_{2,1}, W_{2,2} \in Z_1$. Therefore, $\{1, 2\}$ is an user group and the corresponding all-but-one equations are $W_{1,1} \oplus W_{2,1}$ and $W_{1,1} \oplus W_{2,2}$. However, $\mathcal{F}_{\{1,\{1,3\}\}} = \emptyset$ thus $\{1, 3\}$ is not an user group.

As $U = \{1, 2\}$ is an user group, we have $\mathcal{U}_2 = \{\{1\}, \{2\}, \{1, 2\}\}$. The set of time intervals where user group $\{1, 2\}$ is active is $\mathcal{I}_{\{1,2\}} = \{2, 3\}$. Finally, note that user group $U = \{2, 3\}$ is a member of $\mathcal{U}_{\{2,1\}}$ since $2 \in U$ and $1 \in \mathcal{F}_{\{2,U\}} = \{1, 2\}$. Similarly, $U \in \mathcal{U}_{\{2,2\}}$ as well since $2 \in U$ and $2 \in \mathcal{F}_{\{2,U\}}$.

3.3 Offline Asynchronous Coded Caching

In this section, we discuss the offline version of the problem where the server has the knowledge of the arrival times/deadlines of all the requests at $\tau = 0$. The offline solution of the system

in Example 11 is depicted in Figure 3.2 where the transmitted equation in each time slot appears above the timeline. It can be verified that each user can recover the missing subfiles that they need. In what follows we argue that the offline setting can be cast as a linear programming problem.

3.3.1 Linear programming formulation

For each time interval Π_ℓ with $\ell = 1, \ldots, \beta$ and for each $U \in \mathcal{U}_\ell$ we define variable $x_U(\ell) \in [0, |\Pi_\ell|]$ that represents the portion of time interval Π_ℓ that is allocated to an equation that benefits user group U. The actual equation will be determined shortly. For each missing subfile $W_{\{d_i,f\}}$ and each $U \in \mathcal{U}_{\{i,f\}}$ we define variable $y_{\{i,f\}}(U) \in [0, r]$ that represents the portion of the missing subfile $W_{\{d_i,f\}}$ transmitted within some or all of the equations associated with $x_U(\ell)$ for $\ell \in \mathcal{I}_U$. As pointed out before, for a fixed i, U can be used to transmit different missing subfiles needed by user i. However, a single equation can only help recover one missing subfile needed by i. Thus, $\sum_{\ell \in \mathcal{I}_U} x_U(\ell)$ must be shared between the appropriate $y_{\{i,f\}}(U)$'s. Accordingly, we need the following constraint for user i and a user group U which contains i.

$$\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \leq \sum_{\ell \in \mathcal{I}_U} x_U(\ell).$$

In addition, at time interval Π_ℓ at most $|\Pi_\ell|$ packets can be transmitted, so that $\sum_{U \subseteq \mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|$. To ensure that each missing subfile $W_{\{d_i,f\}}$ is transmitted in exactly r time slots we have $\sum_{U \in \mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = r$.

Table 3.1: List of variables used in the description

Variable	Description
T_i	arrival time of user i
$T_i + \Delta_i$	deadline of user i
β	number of time intervals
Π_ℓ	time interval ℓ
U_ℓ	set of the active users in time interval ℓ
$\Omega^{(i)}$	set of the indices of missing subfiles of user i
$\mathcal{U}_\ell$	set of all subsets of U_ℓ that are user groups
$\mathcal{I}_U$	set of the indices $\ell \in [\beta]$ so that $U \in \mathcal{U}_\ell$
$\mathcal{U}_{\{i,f\}}$	set of all user groups that $W_{d_i,f}$ can be transmitted within
$x_U(\ell)$	portion of Π_ℓ allocated to user group U
$y_{\{i,f\}}(U)$	portion of $W_{d_i,f}$ transmitted within user group U
$\mathcal{F}_{\{i,U\}}$	set of the indices of $f \in \Omega^{(i)}$ that can be transmitted within U

The following LP minimizes the overall rate of transmission from the server while respecting all the deadline constraints of the users.

$$\min \sum_{\ell=1}^{\beta} \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \tag{3.1}$$

$$\text{s.t.} \quad \sum_{U \in \mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|, \qquad \text{for } \ell = 1, \ldots, \beta,$$

$$\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \leq \sum_{\ell \in \mathcal{I}_U} x_U(\ell), \text{ for } i \in U,\ U \in \cup_{\ell=1}^{\beta} \mathcal{U}_\ell,$$

$$\sum_{U \in \mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = r, \qquad \text{for } f \in \Omega^{(i)},\ i \in [K],$$

$$x_U(\ell),\ y_{\{i,f\}}(U) \geq 0, \qquad \text{for } \forall i \in [K],\ \ell \in [\beta],\ U \in \cup_{\ell=1}^{\beta} \mathcal{U}_\ell.$$

Note that Niesen and Maddah-Ali (2015) considers the case when each missing subfile has a prescribed deadline. Our LP above can be modified in a straightforward manner to incorporate this aspect.

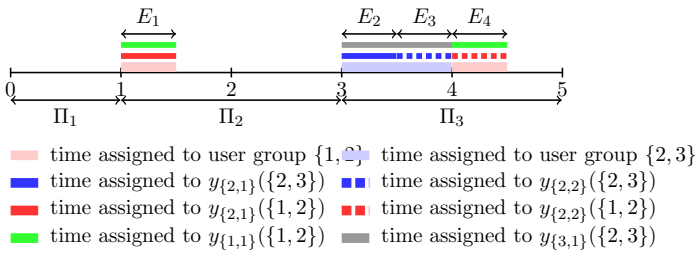


Figure 3.3: Interpretation of feasible point in (3.1) for Example 11. For readability, only equations corresponding to user groups $\{1,2\}$ and $\{2,3\}$ are depicted.

3.3.2 Interpretation of feasible point of (3.1) as a coding solution

We start by assigning time intervals to user groups. The time interval Π_ℓ, $\ell \in [\beta]$, will be arbitrarily assigned to user groups $U \in \mathcal{U}_\ell$ so that the time assigned to one user group does not overlap with another. The constraint $\sum_{U\in\mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|$ implies that such an assignment exists. For each user group U and each $i \in U$, suppose that $f_1, \ldots, f_l \in \mathcal{F}_{\{i,U\}}$ are such that $y_{\{i,f_j\}}(U) \neq 0$ for $j = 1, \ldots, l$. We assign $y_{\{i,f_j\}}(U)$ part of the total time allocated to user group U, i.e., $\sum_{\ell\in\mathcal{I}_U} x_U(\ell)$, to the missing subfile W_{d_i,f_j} for $j = 1, \ldots, l$. The constraint $\sum_{f\in\mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \leq \sum_{\ell\in\mathcal{I}_U} x_U(\ell)$ ensures that such an assignment always exists, i.e., it is possible to assign $y_{\{i,f\}}(U)$'s (for fixed i) to the available (strictly) positive $x_U(\ell)$'s, such that there is no overlap between them. This assignment is not unique in general. However, this is not a problem as any assignment can be used to determine the equations. This process is repeated for all users $i \in U$.

The equation transmitted on a particular interval is simply the XOR of the subfile indices that map to that interval. This equation is valid since the missing subfile $W_{d_i,f}$ with $f \in \mathcal{F}_{\{i,U\}}$ is in the cache of all the users in $U \setminus \{i\}$.

Finally, according to the constraint $\sum_{U\in\mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = r$, each missing subfile $W_{d_i,f}$ is transmitted in its entirety in some equation. The following example serves to illustrate the arguments above.

Example 12 *Consider again the system in Example 11. Part of a feasible solution to the LP in (3.1), corresponding to user groups* $U = \{2, 3\}$ *and* $U' = \{1, 2\}$*, is presented below.*

$$\begin{aligned} x_{\{1,2\}}(2) &= 0.5, \quad x_{\{1,2\}}(3) = 0.5, & x_{\{2,3\}}(3) &= 1, \\ y_{\{1,1\}}(\{1,2\}) &= 1, \quad y_{\{2,1\}}(\{1,2\}) = 0.5, & y_{\{2,2\}}(\{1,2\}) &= 0.5, \\ y_{\{2,1\}}(\{2,3\}) &= 0.5, \quad y_{\{2,2\}}(\{2,3\}) = 0.5, & y_{\{3,1\}}(\{2,3\}) &= 1. \end{aligned}$$

According to the solution, $x_{\{2,3\}}(3) = 1$*. Therefore, only one unit of* Π_3 *is assigned to* U *(though* $|\Pi_3| = 2$*). This is denoted by the light blue color line in Figure 3.3. For user* $3 \in U$*, there is only one missing subfile in* $\mathcal{F}_{\{3,\{2,3\}\}}$*, namely* $W_{3,1}$*. As* $y_{\{3,1\}}(\{2,3\}) = 1$ *it is assigned to* $x_{\{2,3\}}(3)$ *in its entirety. This is depicted by the gray line in Figure 3.3. For user 2 in* U *we have* $\mathcal{F}_{\{2,\{2,3\}\}} = \{1, 2\}$*. The solution specifies* $y_{\{2,1\}}(\{2,3\}) = y_{\{2,2\}}(\{2,3\}) = 0.5$*. Thus, we assign the first half of* $x_{\{2,3\}}(3)$ *to missing subfile* $W_{2,1}$ *and the second half to* $W_{2,2}$ *(see the dark blue and dotted dark blue lines in Figure 3.3). Accordingly, the server transmits equations such that the first half of the time interval assigned to user group* U *corresponds to the* $E_2 = W_{2,1} \oplus W_{3,1}$ *whereas the second half corresponds to* $E_3 = W_{2,2} \oplus W_{3,1}$*. The interpretation of the user group* U' *is similar (see Figure 3.3).*

Remark 1 *The output of the above LP will typically result in a fractional solution for the variables. A fractional solution can be interpreted by assuming that each packet that is transmitted over the shared edge can be subdivided as finely as needed. Thus, in each time slot we could transmit multiple equations that may serve potentially different subsets of users. This assumption is reasonable if the underlying subfiles and hence the packets are quite large. In any case the above LP provides a lower bound on the performance of a solution where integrality constraints are enforced.*

Remark 2 *We note that for the offline solution, within a given time interval, the user groups can be assigned in any order according to the* $x_U(\ell)'s$ *as long as they don't overlap. Moreover the assignment of* $y_{i,f}(U)'s$ *is also arbitrary as long the constraints of the LP are respected. However for the online case (cf. Section 3.4), ordering does matter since we make a best effort decision on each individual slot as we have no knowledge of future arrivals.*

Remark 3 *The complexity of our solution does not have any dependence on the arrival times T_i's and the deadlines Δ_i's. Our formulation of the LP in terms of the intervals allows us to circumvent this potential dependence. A straightforward formulation of the above problem would assign variables for each time slot which would be very expensive.*

Nevertheless, the complexity of the solving the LP does grow quite quickly (cubic) in the problem parameters. Next, we discuss a solution based on dual decomposition.

3.3.3 Dual decomposition based LP solution

As it stands, the LP in (3.1) cannot be interpreted as a network flow. Yet, intuitively one can view the missing subfiles from each user as flowing through the user groups and getting absorbed in sinks that correspond to their valid time intervals. However, the flows corresponding to different users can be shared as the all-but-one equations allow different users to benefit from the same equation. We note here that a similar sharing of flows also occurs in the problem of minimum cost multicast with network coding Lun et al. (2006); Ramamoorthy (2011). The LP in (3.1) can however be modified slightly so that the corresponding dual function is such that it can be evaluated by solving a set of *decoupled minimum cost network flow optimizations*.

3.3.3.1 Decoupling procedure

For each user $i \in U$ the variable $x_U^{(i)}(\ell)$ represents the amount of flow corresponding to user i outgoing from user group U to time interval Π_ℓ. Evidently, this amount can't be more than $x_U(\ell)$. Therefore, we have

$$x_U^{(i)}(\ell) \leq x_U(\ell),$$

which holds for all $i \in U$ and all $U \in \mathcal{U}_\ell$, $\ell \in [\beta]$. We define $\mathcal{U}_\ell^{(i)} \subseteq \mathcal{U}_\ell$ to be the subset of possible user groups at time interval Π_ℓ that include user i, i.e., $i \in U$ for all $U \in \mathcal{U}_\ell^{(i)}$.

By the flow interpretation of $x_U^{(i)}(\ell)$, we have $\sum_{f\in\mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) = \sum_{\ell\in\mathcal{I}_U} x_U^{(i)}(\ell)$ for all $U \in \cup_{\ell=1}^{\beta}\mathcal{U}_\ell^{(i)}$. For $i = 1,\ldots,K$, let $\mathcal{C}_i$ denote the following set of constraints.

$$\begin{aligned}
&\sum_{f\in\mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) = \sum_{\ell\in\mathcal{I}_U} x_U^{(i)}(\ell), \qquad \text{for } U \in \cup_{\ell=1}^{\beta}\mathcal{U}_\ell^{(i)},\\
&\sum_{U\in\mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = r, \qquad \text{for } f \in \Omega^{(i)},\\
&x_U^{(i)}(\ell),\ y_{\{i,f\}}(U) \geq 0, \qquad \text{for } U \in \mathcal{U}_\ell^{(i)},\ \ell \in [\beta],\ f \in \Omega^{(i)}.
\end{aligned}$$

Then, the original LP can be compactly rewritten as

$$\begin{aligned}
\min \quad & \sum_{\ell=1}^{\beta}\sum_{U\in\mathcal{U}_\ell} x_U(\ell) \qquad (3.2)\\
\text{s.t.} \quad & x_U^{(i)}(\ell) \leq x_U(\ell) \qquad \text{for } U \in \mathcal{U}_\ell^{(i)},\ \ell \in [\beta], i \in [K],\\
& \sum_{U\in\mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|, \qquad \text{for } \ell \in [\beta],\\
& \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K.
\end{aligned}$$

It is not too hard to see that the LPs in (3.1) and (3.2) are equivalent (see Appendix B.0.1). The only difference with respect to (3.1) is the introduction of variables $x_U^{(i)}(\ell)$ (for appropriate ranges of i, U and ℓ) such that the second set of inequality constraints in (3.1) are replaced by equality constraints. Moreover, the original constraints are maintained by setting $x_U^{(i)}(\ell) \leq x_U(\ell)$.

We proceed by considering the dual of the LP in (3.2) with respect to the constraints that involve the variables $x_U(\ell)$. The Lagrangian $\mathcal{L}(\{x_U(\ell),\ x_U^{(i)}(\ell),\ \lambda_U^{(i)}(\ell)\}_{i\in U, U\in\mathcal{U}_\ell,\ell\in[\beta]}, \{\zeta_\ell\}_{\ell\in[\beta]})$ can be expressed as

$$\mathcal{L} = \sum_{\ell=1}^{\beta}\sum_{U\in\mathcal{U}_\ell} x_U(\ell) + \sum_{\ell=1}^{\beta}\sum_{U\in\mathcal{U}_\ell}\sum_{i\in U} \lambda_U^{(i)}(\ell)\left(x_U^{(i)}(\ell) - x_U(\ell)\right) + \sum_{\ell=1}^{\beta}\zeta_\ell\left(\sum_{U\in\mathcal{U}_\ell} x_U(\ell) - |\Pi_\ell|\right)$$

where $\lambda_U^{(i)}(\ell)$'s and ζ_ℓ's are nonnegative dual variables. It turns out that minimizing the Lagrangian for fixed dual variables can be simplified by defining $\gamma_U^{(i)}(\ell) = \lambda_U^{(i)}(\ell)/(1+\zeta_\ell)$ for $i \in U$, $U \in \mathcal{U}_\ell$, and $\ell \in [\beta]$. We define $\Gamma^{(i)} = \{\gamma_U^{(i)}(\ell),\ \ell \in \mathcal{I}_U,\ U \in \mathcal{U}_\ell^{(i)}\}$, $\mathbf{x} = \{x_U(\ell),\ U \in \mathcal{U}_\ell,\ \ell \in [\beta]\}$, and

$\mathbf{x}^{(i)} = \{x_U^{(i)}(\ell),\ \ell \in \mathcal{I}_U,\ U \in \mathcal{U}_\ell^{(i)}\}$. The dual function $g(\Gamma^{(1)}, \ldots, \Gamma^{(K)}, \{\zeta_\ell\}_{\ell\in[\beta]})$ is obtained by solving for

$$\min_{\mathbf{x},\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(K)}} \mathcal{L} \qquad \text{s.t.} \quad \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K.$$

It is evident that the dual function $g(\Gamma^{(1)}, \ldots, \Gamma^{(K)}, \{\zeta_\ell\}_{\ell\in[\beta]})$ takes a nontrivial value only if

$$\sum_{i\in U} \gamma_U^{(i)}(\ell) = 1, \qquad \forall\ U \in \mathcal{U}_\ell,\ \ell \in [\beta].$$

The evaluation of $g(\Gamma^{(1)}, \ldots, \Gamma^{(K)}, \{\zeta_\ell\}_{\ell\in[\beta]})$ at a fixed set of dual variables $\Gamma^{(i)}$'s and ζ_ℓ's can therefore be written as

$$\min_{\mathbf{x}^{(1)},\ldots,\mathbf{x}^{(K)}} \sum_{\ell=1}^{\beta} \sum_{U\in\mathcal{U}_\ell} \sum_{i\in U} (1+\zeta_\ell)\gamma_U^{(i)}(\ell) x_U^{(i)}(\ell) - \sum_{\ell=1}^{\beta} \zeta_\ell |\Pi_\ell|$$
$$\text{s.t.} \quad \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K. \tag{3.3}$$

We emphasize that (3.3) is still a convex problem and that $\gamma_U^{(i)}(\ell), \zeta_\ell \geq 0$. Let $h_i(\Gamma_i, \{\zeta_\ell\}_{\ell\in[\beta]})$, $i \in [K]$ be

$$h_i(\Gamma_i, \{\zeta_\ell\}_{\ell\in[\beta]}) = \min_{\mathbf{x}^{(i)}} \sum_{\ell=1}^{\beta} \sum_{U\in\mathcal{U}_\ell^{(i)}} (1+\zeta_\ell)\gamma_U^{(i)}(\ell) x_U^{(i)}(\ell),$$
$$\text{s.t.} \quad \mathcal{C}_i. \tag{3.4}$$

Then, the dual function becomes

$$g(\Gamma^{(1)}, \ldots, \Gamma^{(K)}, \{\zeta_\ell\}_{\ell\in[\beta]}) = \sum_{i=1}^{K} h_i(\Gamma_i, \{\zeta_\ell\}_{\ell\in[\beta]}) - \sum_{\ell=1}^{\beta} \zeta_\ell |\Pi_\ell|, \tag{3.5}$$

if $\sum_{i\in U} \gamma_U^{(i)}(\ell) = 1$ for all $U \in \mathcal{U}_\ell,\ \ell \in [\beta]$. We present an approach to maximize the dual function in (3.5) shortly.

The sub-problem in (3.4) for fixed Γ_i and $\{\zeta_\ell\}_{\ell\in[\beta]}$, is a standard minimum-cost flow problem. The associated flow network corresponding to user i, $i \in [K]$, depends on Γ_i and $\{\zeta_\ell\}_{\ell\in[\beta]}$ and we denote it by $\mathcal{N}_i(\Gamma_i, \{\zeta_\ell\}_{\ell\in[\beta]})$. It contains a source node s and three intermediate layers followed by a terminal node t (see Figure 3.4 for an example). The nodes in the first, second, and third

layer correspond to missing subfiles in $\Omega^{(i)}$, user groups in $\cup_{\ell\in[\beta]}\mathcal{U}_\ell^{(i)}$, and time intervals $\{\Pi_\ell : \ \ell \in [\beta] \text{ and } i \in U_\ell\}$ respectively. The edges in $\mathcal{N}_i(\Gamma_i, \{\zeta_\ell\}_{\ell\in[\beta]})$ can be expressed as follows. There are $|\Omega^{(i)}|$ edges going from source node s to each of missing subfiles in $\Omega^{(i)}$. Also, for each $f \in \mathcal{F}_{\{i,U\}}$ there is an edge going from missing subfile node f to user group node U. Furthermore, there is an edge going from user group $U \in \cup_{\ell\in[\beta]}\mathcal{U}_\ell^{(i)}$ to time interval Π_ℓ for each $\ell \in \mathcal{I}_U$. Finally, corresponding to each time interval in $\{\Pi_\ell : \ \ell \in [\beta] \text{ and } i \in U_\ell\}$ there is an edge going from this time interval to the terminal node t.

In flow network $\mathcal{N}_i(\Gamma_i, \{\zeta_\ell\}_{\ell\in[\beta]})$, $i \in [K]$, a zero cost is assigned to all edges except those from the user group nodes to the time intervals. The cost of the edge between user group U and time interval Π_ℓ is $(1+\zeta_\ell)\gamma_U^{(i)}(\ell)$. The edge between time interval Π_ℓ and the terminal node has a capacity constraint of $|\Pi_\ell|$ and the edge between the source node and a missing subfile has a capacity constraint of r; the other edges have no capacity constraint. The variable $x_U^{(i)}(\ell)$ is the amount of flow carried by the edge from user group U to time interval Π_ℓ. The source injects a flow of value $|\Omega^{(i)}|r$ which needs to be absorbed in the terminal.

We emphasize that minimum cost network flow algorithms have been subject of much investigation Ahuja et al. (1993) within the optimization literature and large scale instances can be solved very quickly. For our work we leverage Capacity Scaling algorithms within the open-source LEMON package LEMON ().

3.3.3.2 Maximizing the dual function

The dual function in (3.5) is concave (as it can be expressed as the pointwise infimum of a family of affine functions of the dual variables Boyd and Vandenberghe (2004)). We exploit the projected subgradient method to maximize the dual function iteratively. Let $x_U^{(i)}(\ell, n-1)$ for all $i \in [K], U \in \mathcal{U}_\ell$ denote the optimal point of (3.4) when solved for $i \in [K]$ at the $n-1$ iteration. Let $\{\gamma_U^{(i)}(\ell, n-1),\ \zeta_\ell(n-1),\ \forall\ U \in \mathcal{U}_\ell^{(i)},\ \ell \in [\beta], i \in [K]\}$ denote a dual feasible point of (3.5) at the $(n-1)$-th iteration.

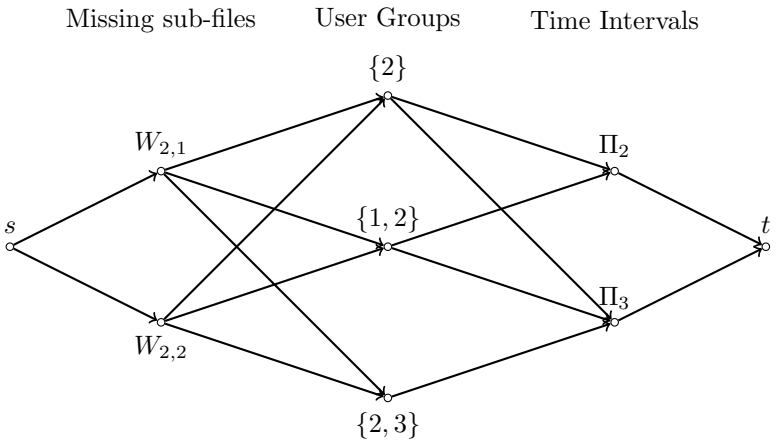


Figure 3.4: Min-cost flow network associated with subproblem (3.4) corresponding to the second user, $\mathcal{N}_2(\Gamma_2, \zeta_2, \zeta_3)$. The constraints and costs are given in the text.

According to the subgradient method for the n-th iteration we first determine the following two parameters for $i \in [K]$

$$\tilde{\gamma}_U^{(i)}(\ell, n) = \gamma_U^{(i)}(\ell, n-1) + \theta_n x_U^{(i)}(\ell, n)(1 + \zeta_\ell(n-1)),$$
$$\tilde{\zeta}_\ell(n) = \zeta_\ell(n-1) + \theta_n(\sum_{U \in \mathcal{U}_\ell} \sum_{i \in U} \gamma_U^{(i)}(\ell, n) x_U^{(i)}(\ell, n) - |\Pi_\ell|),$$

where θ_n is the step size. These intermediate variables are projected onto the feasible set and primal recovery is performed by the method of Sherali and Choi (1996). The details can be found in the Appendix B.0.2. Numerical results appear in Section 3.5.

3.4 Online Asynchronous Coded Caching

In the online scenario, at time τ only information about the already arrived requests are known to the server, i.e., it only knows T_i, d_i and Δ_i for $i \in [K]$ such that $T_i \leq \tau$. Ideally, one would want to design an online algorithm that is guaranteed to be feasible whenever the corresponding offline version is feasible. However, this appears to be a hard problem. Specifically, routinely used algorithms such as earliest-deadline-first (EDF) do not have this property (see Appendix B.0.3). In this section we discuss certain characteristics of the online solution that distinguish it from the offline solution (Section 3.4.1) and our proposed online algorithm and its properties (Section 3.4.2).

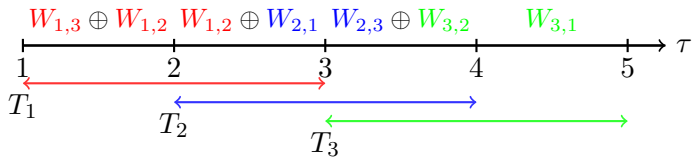


Figure 3.5: Online solution corresponding to the Example 13. Note that the server is forced to transmit $W_{1,2} \oplus W_{1,3}$ at $\tau = 1$.

3.4.1 Necessity of coding across missing subfiles of a user

Example 13 *Consider a system with $N = K = 3$ and $M_i = 1$ with $Z_i = \{W_{n,i} : n \in [N]\}$ for $i \in [K]$. The arrival times and deadlines of the users are $T_i = i$, and $\Delta_i = 2$ for $i \in [K]$ (as shown in Figure 3.5). We assume that user i is interested in files W_i for $i \in [K]$ and that transmitting a subfile takes a single time slot, i.e., $r = 1$.*

Suppose that there is an adversary that makes decisions on when a particular user request arrives. We assume that the adversary can see the decisions made by the server. Suppose that the server does not code across any user's missing subfiles. At $\tau = 1$, it has the choice to transmit either $W_{1,2}$ or $W_{1,3}$. We emphasize that it has to transmit either of these as the deadline for user 1 is $T_1 + \Delta_1 = 3$. If the server transmits $W_{1,3}$, then the adversary can force the arrival of the third user with $(T_3, \Delta_3) = (2, 2)$ and subsequently the arrival of the second user with $(T_2, \Delta_2) = (3, 2)$. In this case, the server is forced to transmit $W_{1,2}$ at $\tau = 2$, which implies that user 3 misses its deadline. In a similar manner, if the server transmits $W_{1,2}$, the adversary can easily generate an arrival pattern so that user 2 misses its deadline.

This issue can be circumvented if we transmit a linear combination of both $W_{1,2}$ and $W_{1,3}$ in the first time slot as shown in Fig 3.5. Intuitively, this is the correct strategy since transmitting $W_{1,3} \oplus W_{1,2}$ allows the server to hedge its bets against the identity of the next request arrival. This example demonstrates that coding across missing subfiles of user 1 is strictly better than the alternative. We emphasize that the synchronized model of Maddah-Ali and Niesen (2014) and the offline scenario do not require this.

Accordingly, for the online scenario we treat each missing subfile $W_{d_i,f}$ as an element of a large enough finite field $\mathbb{F}$. This allows us to consider linear combinations of the missing subfiles over $\mathbb{F}$. Note that any equation of the form

$$\bigoplus_{i\in U}\bigoplus_{f\in\mathcal{F}_{\{i,U\}}}\alpha_{\{i,f\}}W_{\{d_i,f\}},$$

where the coefficients $\alpha_{\{i,f\}}$ belong to the field $\mathbb{F}$ is also an all-but-one equation from which user i can recover $\bigoplus_{f\in\mathcal{F}_{\{i,U\}}}\alpha_{\{i,f\}}W_{\{d_i,f\}}$.

3.4.2 Recursive LP based algorithm

The online scenario differs significantly from the offline one. At time τ our only decision is to transmit an equation in the time slot $[\tau,\tau+1)$. In particular, it is possible that a request arrives at $\tau+1$ and that can change the situation drastically. It makes intuitive sense to transmit equations that benefit a large number of users. However, we also need to take into account the deadline constraints of each user. These requirements need to be balanced.

Our proposed online algorithm leverages the offline LP for enforcing the deadline constraints. We solve an LP which is similar to (3.1) each time a new user request comes into the system. This specifies a set of $x_U(\ell)$ and $y_{i,f}(U)$ variables. However, in the offline case, the ordering of the $x_U(\ell)$'s within an interval does not matter (*cf.* Remark 2). In the online case, this is no longer true. As we have no knowledge of future arrivals, it becomes important to choose the "best" user group for the time slot in which transmission needs to take place. Furthermore in each time slot, exactly one equation is transmitted, i.e., for each time slot only one user group is chosen for transmission. In contrast, in the offline LP, the fractional nature of the solution may require sharing of a time slot between multiple user groups.

Accordingly, based on the $x_U(\ell)$ variables we first decide a candidate list of feasible user groups that can be chosen for transmission at each time slot. We calculate a metric for each feasible user group U depending upon (i) the stringency of the deadlines of the users in U, and (ii) the benefit of this equation to the participating users. If this metric is above a system-defined threshold, we transmit an equation corresponding to this user group in the time slot following the user's request.

Following this we update certain variables and the process continues for each time slot thereafter. When the next user request arrives into the system, the history of the variable assignments is used to solve a new LP (similar to (3.1)), and the process continues recursively.

Consider a time $\tau = T_k$ when the request of the k-th user arrives at the server. We let $\mathcal{U}_{\text{sent}}(\tau)$ be the set of user groups associated with the previously transmitted equations. We also let $z_U(\tau)$ be the total time allocated to equations corresponding to user group U prior to time τ. Thus, if in time interval $[\tau, \tau+1)$ the server transmits an equation that exclusively benefit users in U then $z_U(\tau+1) = z_U(\tau) + 1$ otherwise $z_U(\tau+1) = z_U(\tau)$. Time intervals $\Pi_{1,k}, \ldots, \Pi_{\beta_k,k}$ are formed by the set of times in

$$\{T_k\} \cup \{T_i + \Delta_i : i \in [k],\ T_i + \Delta_i > T_k\}.$$

As in the offline case in (3.1), the sets of active users $U_{\ell,k}$, user groups $\mathcal{U}_{\ell,k}$ and $\mathcal{I}_U^{(k)}$ are defined corresponding to these time intervals, e.g., $U_{\ell,k}$ is the set of active users in $\Pi_{\ell,k}$. Moreover, $\mathcal{V}_k$ is a set of user groups that either already have been transmitted or might be transmitted after $\tau = T_k$. That is $\mathcal{V}_k = \mathcal{U}_{\text{sent}}(\tau) \cup \{\mathcal{U}_{\ell,k} :\ \ell \in [\beta_k]\}$. The variables $x_U(\ell)$'s and $y_{\{i,f\}}(U)$'s have the same interpretation as the offline case. With these variables, the server solves the following LP.

$$\min \sum_{\ell=1}^{\beta_k} \sum_{U \in \mathcal{U}_{\ell,k}} x_U(\ell) \tag{3.6}$$

$$\text{s.t.} \sum_{U \in \mathcal{U}_{\ell,k}} x_U(\ell) \le |\Pi_{\ell,k}|, \quad \text{for } \ell = 1, \ldots, \beta_k$$

$$\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \le \sum_{\ell \in \mathcal{I}_U^{(k)}} x_U(\ell) + z_U(T_k) \text{ for } i \in U, U \in \mathcal{V}_k,$$

$$\sum_{U \in \mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = r, \quad \text{for } f \in \Omega^{(i)},\ \forall\ i \in \cup_{\ell=1}^{\beta_k} U_{\ell,k},$$

$$x_U(\ell),\ y_{\{i,f\}}(U) \ge 0, \text{ for } i \in [k], \ell \in [\beta], U \in \mathcal{V}_k.$$

An important feature of time intervals $\Pi_{1,k}, \ldots, \Pi_{\beta_k,k}$ is that these time intervals end at a deadline and except the first time interval $\Pi_{1,k}$ that starts with arrival time T_k, the other time intervals start with a deadline. Thus, we have $U_{\ell+1,k} \subset U_{\ell,k}$, i.e., the set of active users in interval $\Pi_{\ell+1,k}$ is a subset of the active users in interval $\Pi_{\ell,k}$ for the range of ℓ.

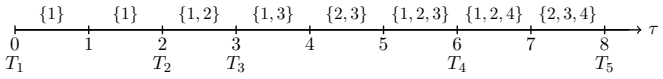


Figure 3.6: An illustration of arrival times and user groups associated with the already submitted equations upon time $\tau = 8$ in Example 14. Associated with each user group in each time slot, an equation has been submitted by the server at the same time slot.

Upon solving the LP in (3.6), the server makes a decision on the equation that will be transmitted in time slot $[T_k, T_k + 1)$. Towards this end, it creates a list of candidate user groups. Let $\{x^*_U(\ell),\ \forall\ U \in \mathcal{U}_\ell,\ \ell = 1, \ldots, \beta_k\}$ be the solution of (3.6) and let $\mathcal{X}^* = \{x^*_U(\ell) :\ x^*_U(\ell) \geq 1\}$. The elements of $\mathcal{X}^*$ are first ordered based on time intervals. Then, among the elements with the same time interval, they are ordered based on length of user group. Therefore, for two elements $x^*_U(\ell), x^*_{U'}(\ell') \in \mathcal{X}^*$ we say $x^*_U(\ell)$ is before $x^*_{U'}(\ell')$ if $\ell < \ell'$, or if $\ell = \ell'$ and $|U| \geq |U'|$. We let $\mathcal{X}^*_{\text{sorted}}$ denote the sorted version of $\mathcal{X}^*$ using this procedure. Let $v_i(\tau)$ be the number of missing packets (subfiles when $r = 1$) that have been transmitted for user i until time τ; this value is tracked in Algorithm 5.

Next, we compute a metric $\eta_U(\tau)$ for each $U \in \mathcal{X}^*_{\text{sorted}}$ that measures the overall benefit of transmitting an equation corresponding to user group U. If a user group U is chosen for transmission, it may in general benefit different users differently. For instance, if U has been used for transmission in the past, then the current transmission may be less beneficial to some of the users or of no benefit. We demonstrate this by means of the following example.

Example 14 *Consider a system $N = K = 5$, $M_i = 2$ for all users $i \in [K]$, and $r = 1$. The placement scheme is the same as Maddah-Ali and Niesen (2014) so that each file is divided to $F = 10$ subfiles and each user misses 6 subfiles. The cache content and missing subfiles are specified below.*

$$Z_1 = \{W_{\{n,f\}},\ n \in [5],\ f = 1,2,3,4\}, \qquad \Omega^{(1)} = \{5,6,7,8,9,10\}$$
$$Z_2 = \{W_{\{n,f\}},\ n \in [5],\ f = 1,5,6,7\}, \qquad \Omega^{(2)} = \{2,3,4,8,9,10\}$$
$$Z_3 = \{W_{\{n,f\}},\ n \in [5],\ f = 2,5,8,9\}, \qquad \Omega^{(3)} = \{1,3,4,6,7,10\}$$
$$Z_4 = \{W_{\{n,f\}},\ n \in [5],\ f = 3,6,8,10\}, \qquad \Omega^{(4)} = \{1,2,4,5,7,9\}$$
$$Z_5 = \{W_{\{n,f\}},\ n \in [5],\ f = 4,7,9,10\}, \qquad \Omega^{(5)} = \{1,2,3,5,6,8\}.$$

We assume that the current time is $\tau = 8$ *and that the request of users* $1, \ldots, 4$ *have arrived to the server. More specifically, we have* $T_1 = 0, T_2 = 2, T_3 = 3, T_4 = 6$ *with deadlines* $\Delta_i = 15$ *for all users* $i \in [K]$*. The server has already transmitted eight equations so that*

$$\mathcal{U}_{sent}(\tau) = \{\{1\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}, \{1,2,4\}, \{2,3,4\}\}.$$

At $\tau = 8$ *the server solves problem (3.6) with earlier parameters set to* $z_{\{1\}}(\tau) = 2$, $z_U(\tau) = 1$ *for all user groups* $U \in \mathcal{U}_{sent}(\tau) \setminus \{\{1\}\}$, *and* $z_U(\tau) = 0$ *otherwise (see Figure 3.6). Then, solving (3.6) for* $k = 5$ *yields the following* $x_U(\ell)$ *variables.*

$$\begin{aligned} &x_{\{2,3,5\}}(1) = 1, \quad x_{\{2,4,5\}}(1) = 1, \quad x_{\{3,4,5\}}(1) = 1, \\ &x_{\{3,4\}}(1) = 1, \quad x_{\{4,5\}}(1) = 1, \quad x_{\{5\}} = 2. \end{aligned} \tag{3.7}$$

Suppose that the server uses this solution as follows. It schedules user groups $\{2,3,5\}$, $\{2,4,5\}$, *and* $\{3,4,5\}$ *at time slots beginning on* $\tau = 8$, $\tau = 9$, *and* $\tau = 10$ *respectively. Before* $\tau = 8$, *the third user has benefitted from user groups*

$$\{1,3\}, \{2,3\}, \{1,2,3\}, \text{ and } \{2,3,4\}.$$

The third user can recover missing subfiles $W_{\{d_3,1\}}$ *and* $W_{\{d_3,6\}}$ *from transmitted equations associated with user groups* $\{1,2,3\}$ *and* $\{2,3,4\}$ *respectively. Moreover, it can obtain a linear combination of subfiles* $\{W_{\{d_3,f\}}, f = 1,6,7\}$ *and* $\{W_{\{d_3,f\}}, f = 1,3,4\}$ *from the equations associated with the user groups* $\{2,3\}$ *and* $\{1,3\}$ *respectively. It is easy to see that subfiles* $W_{\{d_3,1\}}$, $W_{\{d_3,6\}}$, *and* $W_{\{d_3,7\}}$, *along with a linear combination of subfiles* $\{W_{\{d_3,f\}}, f = 3,4\}$ *can be recovered from these equations. Now, at time slot* $\tau = 8$ *the server transmits an equation associated with user group* $\{2,3,5\}$*. Clearly, the fifth user benefits from this equation since this user can recover missing subfile* $W_{\{d_5,5\}}$ *as* $\mathcal{F}_{\{5,\{2,3,5\}\}} = \{5\}$*. Similarly, since* $\mathcal{F}_{\{3,\{2,3,5\}\}} = \{7\}$ *the third user can recover only the missing subfile* $W_{\{d_3,7\}}$ *from this equation. However this subfile has been recovered from the earlier equations. Therefore, this equation and user group is not beneficial for the third user. Similarly, one can show that the second user also benefits nothing from this equation and that this*

equation is only useful for the fifth user. Therefore, the users in a user group might benefit partially or not benefit from the transmitted equation associated with the user group.

Thus, we need a measure of how useful a given U is to a user i when U is used for transmission at a given time slot. This can naturally be described in terms of a related LP that we now describe. For each element $x^*_U(\ell) \in \mathcal{X}^*_{\text{sorted}}$ let $w_{\{i,U\}}(\tau)$ denote the maximum number of missing packets that are recovered by user i if user group U was chosen for transmission at time τ. Let us assume that the server chooses $\hat{U}$ to transmit an equation at the next time slot and let $\tilde{\mathcal{U}}_{\text{sent}}(\tau) = \mathcal{U}_{\text{sent}}(\tau) \cup \{\hat{U}\}$. Under this assumption we let $\tilde{z}_U(\tau)$, for $U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)$, be such that $\tilde{z}_U(\tau) = z_U(\tau)$ for $U \in \mathcal{U}_{\text{sent}}(\tau) \setminus \{\hat{U}\}$ and if $\hat{U} \in \mathcal{U}_{\text{sent}}(\tau)$ then $\tilde{z}_{\hat{U}}(\tau) = z_{\hat{U}}(\tau) + 1$; otherwise $\tilde{z}_{\hat{U}}(\tau) = 1$. Consider the set of all user groups in $\tilde{\mathcal{U}}_{\text{sent}}(\tau)$. For each user group U in this set there are $\tilde{z}_U(\tau)$ time slots available. To compute $w_{\{i,\hat{U}\}}(\tau)$, we need to find an assignment of the missing subfiles in $\Omega^{(i)}$ to each of these time slots so that number of the recovered missing subfiles of the i-th user is maximized. We let $\tilde{y}_{\{i,f\}}(U)$, for $U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)$ with $U \ni i$, to have the same interpretation as $y_{\{i,f\}}(U)$ in (3.1). This is equivalent to finding $\tilde{y}_{\{i,f\}}(U)$'s that maximize $\sum_{U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)\ U \ni i} \sum_{f \in \mathcal{F}_{\{i,U\}}} \tilde{y}_{\{i,f\}}(U)$ under the following constraints. Since for each user group $U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau)$ there are $\tilde{z}_U(\tau)$ time slots available, therefore we have $\sum_{f \in \mathcal{F}_{\{i,U\}}} \tilde{y}_{\{i,f\}}(U) \le \tilde{z}_U(\tau)$. Each missing subfile in $\Omega^{(i)}$ needs r time slots but not all of them can be recoverable. Therefore, we have $\sum_{U \in \mathcal{U}_{\{i,f\}} \cap \tilde{\mathcal{U}}_{\text{sent}}(\tau)} \tilde{y}_{\{i,f\}}(U) \le r$. Thus, $w_{\{i,\hat{U}\}}(\tau)$ can be obtained as the objective function of the following LP.

$$
\begin{aligned}
\max \quad & \sum_{U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau), U \ni i} \ \sum_{f \in \mathcal{F}_{\{i,U\}}} \tilde{y}_{\{i,f\}}(U) \qquad (3.8)\\
\text{s.t.} \quad & \sum_{f \in \mathcal{F}_{\{i,U\}}} \tilde{y}_{\{i,f\}}(U) \le \tilde{z}_U(\tau) \text{ for } \ U \in \tilde{\mathcal{U}}_{\text{sent}}(\tau), U \ni i \\
& \sum_{U \in \mathcal{U}_{\{i,f\}} \cap \tilde{\mathcal{U}}_{\text{sent}}(\tau)} \tilde{y}_{\{i,f\}}(U) \le r \ \text{ for } f \in \Omega^{(i)}, \\
& \tilde{y}_{\{i,f\}}(U) \ge 0.
\end{aligned}
$$

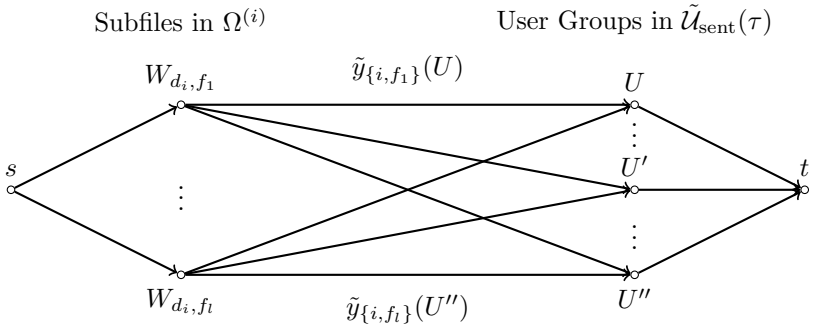


Figure 3.7: Max flow network associated with LP in (3.8).

Remark 4 *The LP in (3.8) can also be expressed as a maximum flow problem. The associated flow network consists of a source node s, a node for each $f \in \Omega^{(i)}$, a node for each user group $U \in \tilde{\mathcal{U}}_{sent}(\tau)$, and a terminal node t. There are edges with capacity r going from s to each $f \in \Omega^{(i)}$ and edges from $f \in \Omega^{(i)}$ to node $U \in \tilde{\mathcal{U}}_{sent}(\tau)$ if $f \in \mathcal{F}_{\{i,U\}}$. The flow on such an edge is $\tilde{y}_{\{i,f\}}(U)$. Moreover, from each node $U \in \tilde{\mathcal{U}}_{sent}(\tau)$ to t there exist an edge of capacity $\tilde{z}_U(\tau)$. These capacity constraints model the first two inequality constraints in (3.8). Figure 3.7 illustrates an example of this network. It is well-known that if all capacities in a flow network are integers, there exists an integral maximum flow (Kleinberg and Tardos (2006), Chapter 7). Therefore, there exists an integral solution for $\tilde{y}_{\{i,f\}}(U)$'s in (3.8) if $\tilde{z}_U(\tau)$'s are integers.*

Example 15 *In this example we show how the LP in (3.8) addresses the issue highlighted in Example 14. We consider the setting of this example and solve (3.8) for the user groups in (3.7). For simplicity, we only discuss the results for user groups $\{2,3,5\}$ and $\{3,4,5\}$. The other user groups follow the same rule. Before proceeding, we note that $v_2(\tau) = 5$, $v_3(\tau) = 4$, $v_4(\tau) = 2$ and $v_5(\tau) = 0$. Our goal is to see how these numbers change if the server decides to transmit an equation associated with either of user groups $\{2,3,5\}$ or $\{3,4,5\}$. If the server chooses $\hat{U} = \{2,3,5\}$ then a nonzero solution for the $\tilde{y}_{\{i,f\}}(\hat{U})$'s for the corresponding users is*

$$\tilde{y}_{\{2,9\}}(\{2,3\}) = 1,\ \tilde{y}_{\{2,2\}}(\{1,2,3\}) = 1,\ \tilde{y}_{\{2,8\}}(\{2,3,4\}) = 1,$$
$$\tilde{y}_{\{2,4\}}(\{1,2\}) = 1,\ \tilde{y}_{\{2,3\}}(\{1,2,4\}) = 1,\ \tilde{y}_{\{3,3\}}(\{1,3\}) = 0.0,$$
$$\tilde{y}_{\{3,4\}}(\{1,3\}) = 1.0,\ \tilde{y}_{\{3,7\}}(\{2,3\}) = 1,\ \tilde{y}_{\{3,1\}}(\{1,2,3\}) = 1,$$
$$\tilde{y}_{\{3,6\}}(\{2,3,4\}) = 1,\ \tilde{y}_{\{5,5\}}(\{2,3,5\}) = 1.$$

This solution results in $w_{\{2,\{2,3,5\}\}}(\tau)=5$, $w_{\{3,\{2,3,5\}\}}(\tau)=4$, *and* $w_{\{5,\{2,3,5\}\}}(\tau)=1$. *Therefore,* $w_{\{i,\{2,3,5\}\}}(\tau)-v_i(\tau)$ *is zero for* $i=2,3$ *and one for* $i=5$. *This implies that* $w_{\{i,\{2,3,5\}\}}(\tau)$ *correctly captures the benefits of transmitting an equation corresponding to each user in* $\{2,3,5\}$.

Now, we repeat the same analysis for user group $\{3,4,5\}$. *After solving (3.8) for this user group and user* 3, *the only change comparing to the solution of this user for the user group* $\{2,3,5\}$ *is that* $\tilde{y}_{\{3,10\}}(\{3,4,5\})=1$ *and thus* $w_{\{3,\{3,4,5\}\}}(\tau)=5$. *For the other users we have* $w_{\{4,\{3,4,5\}\}}(\tau)=3$ *and* $w_{\{5,\{3,4,5\}\}}(\tau)=1$. *Therefore,* $w_{\{i,\{3,4,5\}\}}(\tau)-v_i(\tau)=1$ *for all* $i\in\{3,4,5\}$ *and more users benefit from this user group than* $\{2,3,5\}$.

Note that user i needs to recover $r|\Omega^{(i)}|-v_i(\tau)$ missing packets and it has $T_i+\Delta_i-\tau$ time slots to obtain them. Thus, the ratio of these quantities is a measure of the stringency of the deadline of user i. Furthermore, based on the above discussion $w_{\{i,\hat{U}\}}(\tau)-v_i(\tau)$ indicates the number of packets that are useful to user i if $\hat{U}$ was chosen for transmission. Therefore the metric $\eta_U(\tau)$ is obtained by the following weighted sum.

$$\eta_U(\tau)=\sum_{i\in U}\frac{\big(r|\Omega^{(i)}|-v_i(\tau)\big)}{T_i+\Delta_i-\tau}(w_{\{i,U\}}(\tau)-v_i(\tau)).$$

At time $\tau=T_k$, the server picks the first element $x^*_U(\ell)\in\mathcal{X}^*_{\text{sorted}}$ such that $\eta_U(\tau)\geq\eta_0$ for some threshold η_0 and transmits an equation corresponding to it. Unlike the synchronous case, we choose a random linear combination of all missing subfiles of user i that can be transmitted by user group U.

When $r>1$, we subdivide a missing subfile into r packets that are denoted $W_{\{d_i,f,j\}}$ for $j=1,\ldots,r$. Thus, the server transmits

$$\bigoplus_{i\in U}\bigoplus_{f\in\mathcal{F}_{\{i,U\}}}\bigoplus_{j=1}^{r}\alpha_{\{i,f,j,m\}}W_{\{d_i,f,j\}},$$

at time interval $[\tau,\tau+1)$ where m denotes the m-th equation transmitted by the server and $\alpha_{\{i,f,j,m\}}$ are chosen independently and uniformly at random from the finite field $\mathbb{F}$. If none of the elements in $x^*_U(\ell)\in\mathcal{X}^*_{\text{sorted}}$ satisfy $\eta_U(\ell)\geq\eta_0$ then nothing will be transmitted at this time interval.

If a new user request does not come at time $\tau+1$, then the server updates the user group values and then solves (3.8) again to decide the user group for the time slot $[\tau+1,\tau+2)$. The process

continues this way until the next user request comes when the LP in (3.6) is solved. The complete details are provided in Algorithm 5.

Algorithm 5 Recursive LP Algorithm

Input: Caches Z_i for $i \in [K]$, η_0, $\{T_i, \Delta_i\}$, for $i \in [K]$.

Initialization:
set $\mathcal{U}_{\text{sent}}(0) \leftarrow \emptyset$, $\mathcal{X}_{\text{off}} \leftarrow \emptyset$, $\ell_{\text{off}} \leftarrow 0$, $m \leftarrow 1$ and $k \leftarrow 1$.
set $\mathcal{M}_i = \emptyset$, and $v_i(0) = 0$ for $i = 1, \ldots, K$.
for $\tau = 0, 1, 2, \ldots, T_{\max}$ **do**
  if $\tau = T_i + \Delta_i$ and $v_i(\tau) < r|\Omega^{(i)}|$ for some i **then**
    return `INFEASIBLE`.
  end if
  if $\tau = T_i$ (a new user makes request) for some i **then**
    $k = \arg\max_{i \in [K]} \tau = T_i$
    Solve LP (3.6). Form $\mathcal{X}^*$ and then $\mathcal{X}^*_{\text{sorted}}$.
  end if
  If $\tau = T_i$ or $\tau = T_i + \Delta_i$ for some i then $\ell_{\text{off}} \leftarrow \ell_{\text{off}} + 1$.
  if $\mathcal{X}^*_{\text{sorted}} \neq \emptyset$ **then**
    Pick first in order $x^*_{U^*}(\ell) \in \mathcal{X}^*_{\text{sorted}}$ with $\eta_{U^*}(\tau) \geq \eta_0$.
    Randomly select $\alpha_{\{i,f,j,m\}}$'s from $\mathbb{F}$ and send

$$\bigoplus_{i \in U} \bigoplus_{f \in \mathcal{F}_{\{i,U\}}} \bigoplus_{j=1}^{r} \alpha_{\{i,f,j,m\}} W_{\{d_i,f,j\}},$$

    If $U^* \in \mathcal{U}_{\text{sent}}(\tau)$ then $z_{U^*}(\tau+1) \leftarrow z_{U^*}(\tau)+1$, otherwise $z_{U^*}(\tau+1) = 1$ and $\mathcal{U}_{\text{sent}}(\tau+1) \leftarrow \mathcal{U}_{\text{sent}}(\tau) \cup \{U^*\}$
    If $\tilde{x}_{U^*}(\ell_{\text{off}}) \in \mathcal{X}_{\text{off}}$ then $\tilde{x}_{U^*}(\ell_{\text{off}}) \leftarrow \tilde{x}_{U^*}(\ell_{\text{off}}) + 1$, otherwise $\tilde{x}_{U^*}(\ell_{\text{off}}) = 1$ and $\mathcal{X}_{\text{off}} \leftarrow \mathcal{X}_{\text{off}} \cup \{\tilde{x}_{U^*}(\ell_{\text{off}})\}$
    set $x^*_{U^*}(\ell) \leftarrow x^*_{U^*}(\ell) - 1$, if $x^*_{U^*}(\ell) < 1$ remove it from $\mathcal{X}^*_{\text{sorted}}$.
    For all $i \in U^*$, set $v_i(\tau + 1) \leftarrow w_{\{i,U^*\}}(\tau)$, set $\mathcal{M}_i \leftarrow \mathcal{M}_i \cup \{m\}$, then $m \leftarrow m + 1$
    for all $U \in \mathcal{U}_{\text{sent}}(\tau) \setminus \{U^*\}$ set $z_U(\tau + 1) \leftarrow z_U(\tau)$
    for all $i \in [K] \setminus U^*$ set $v_i(\tau + 1) \leftarrow v_i(\tau)$.
  end if
end for

In general, there is no guarantee that Algorithm 5 will return a feasible schedule if the corresponding offline schedule is feasible. In that sense, Algorithm 5 can be viewed as a heuristic with good experimental performance. However, if Algorithm 5 does not return "`INFEASIBLE`", we can show that a feasible solution for the corresponding offline LP can be identified. This fact coupled with

a usage of the Schwartz-Zippel Lemma allows us to conclude that our algorithm works with high probability if it does not return "INFEASIBLE".

Claim 9 *For user requests, $\{T_i, \Delta_i, d_i\}$, where $i \in [K]$, if Algorithm 5 does not return "INFEASIBLE" then there exists a feasible integral solution for the offline LP in (3.1).*

Proof: For simplicity, we prove the claim for $r = 1$ and the proof for the general case follows directly. We will construct $x_U(\ell)$ and $y_{\{i,f\}}(U)$ variables for the offline LP from the decisions made in Algorithm 5. Note that we update the set $\mathcal{X}_{\text{off}}$ with the user groups chosen in Algorithm 5. It is not difficult to verify that for any $\tilde{x}_U(\ell) \in \mathcal{X}_{\text{off}}$ user group U is a member of $\mathcal{U}_\ell$. Moreover, the algorithm assigns integer values to $\tilde{x}_U(\ell)$. Now, for any $U \in \mathcal{U}_\ell$ in (3.1), we set $x_U(\ell) = \tilde{x}_U(\ell)$ if $\tilde{x}_U(\ell) \in \mathcal{X}_{\text{off}}$ and $x_U(\ell) = 0$ otherwise. Therefore, $x_U(\ell)$'s take integer values. Since at each time only one equation is transmitted in Algorithm 5, the first condition $\sum_{U \in \mathcal{U}_\ell} x_U(\ell) \leq |\Pi_\ell|$ holds for all $\ell \in [\beta]$.

For each $i \in [K]$ we define $[\tau_i, \tau_i + 1)$ to be the last time slot that user i benefits from the equation transmitted by the server. Clearly we have that $v_i(\tau_i + 1) \geq |\Omega^{(i)}|$ otherwise Algorithm 5 will be infeasible at $\tau = T_i + \Delta_i$. We let $U_{i,\text{last}}$ to be the user group associated with this equation where $i \in U_{i,\text{last}}$.

Note that Algorithm 5 tracks a set $\mathcal{U}_{\text{sent}}(\tau)$ that contains all the user groups that have been used by the algorithm before time τ. We let $\tilde{y}_{\{i,f\}}(U)$, $f \in \mathcal{F}_{\{i,U\}}$ and $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ with $U \ni i$ be the solution of (3.8) when solving it for $w_{\{i,U_{i,\text{last}}\}}(\tau_i)$. Then, for each $U \in \cup_{\ell=1}^{\beta} \mathcal{U}_\ell$ with $U \ni i$ and for each $f \in \mathcal{F}_{\{i,U\}}$ we assign $y_{\{i,f\}}(U) = \tilde{y}_{\{i,f\}}(U)$ if $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ and $y_{\{i,f\}}(U) = 0$ otherwise. We apply this assignment for all $i \in [K]$. Algorithm 5 assigns integer values to $z_U(\tau)$'s. From Remark 4 it follows that there exists an integral solution for $\tilde{y}_{\{i,f\}}(U)$'s and consequently the $y_{\{i,f\}}(U)$'s as well. With these assignments, we now demonstrate that the second and third conditions in (3.1) hold.

For the second condition we note that if $U \notin \mathcal{U}_{\text{sent}}(\tau_i)$ then $y_{\{i,f\}}(U) = 0$ and we have nothing to show. For $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ we have that $y_{\{i,f\}}(U) = \tilde{y}_{\{i,f\}}(U)$. Recall that $\tilde{y}_{\{i,f\}}(U)$ is the solution of (3.8) at time $\tau = \tau_i$. By the way that $z_U(\tau)$ has been updated in Algorithm 5, we have

$z_U(\tau) \leq z_U(T_{\max})$. Therefore, we have $z_U(\tau_i+1) \leq z_U(T_{\max}) = \sum_{\ell \in \mathcal{I}_U} \tilde{x}_U(\ell)$ and from (3.8) for $w_{\{i,U_{i,\text{last}}\}}(\tau_i)$,

$$\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) = \sum_{f \in \mathcal{F}_{\{i,U\}}} \tilde{y}_{\{i,f\}}(U) \leq \tilde{z}_U(\tau_i) = z_U(\tau_i+1)$$
$$\leq \sum_{\ell \in \mathcal{I}_U} \tilde{x}_U(\ell) = \sum_{\ell \in \mathcal{I}_U} x_U(\ell).$$

For the third condition, consider any user $i \in [K]$ and any $f \in \Omega^{(i)}$. Recalling the definition of τ_i and $w_{\{i,U_{i,\text{last}}\}}(\tau_i)$, we know that $w_{\{i,U_{i,\text{last}}\}}(\tau_i) = v_i(\tau_i+1) \geq |\Omega^i|$ which implies that in (3.8), we have

$$|\Omega^{(i)}| \leq \sum_{U \in \mathcal{U}_{\text{sent}}(\tau_i), U \ni i} \sum_{f \in \mathcal{F}_{\{i,U\}}} \tilde{y}_{\{i,f\}}(U)$$
$$= \sum_{f \in \Omega^{(i)}} \sum_{U \in \mathcal{U}_{\{i,f\}} \cap \mathcal{U}_{\text{sent}}(\tau_i)} \tilde{y}_{\{i,f\}}(U) \leq \sum_{f \in \Omega^{(i)}} 1 = |\Omega^{(i)}|,$$

where the last inequality comes from the second constraint in (3.8). The middle equality holds by counting arguments for missing subfiles $f \in \Omega^{(i)}$ and user groups in $U \in \mathcal{U}_{\text{sent}}(\tau_i)$. To verify this, consider a bipartite graph in which the left and right nodes correspond to $f \in \Omega^{(i)}$ and $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ with $U \ni i$ respectively. There is an edge between nodes corresponding to f and U if and only if $f \in \mathcal{F}_{\{i,U\}}$. We let $\tilde{y}_{\{i,f\}}(U)$ to be the label of this edge. By the definition of $\mathcal{U}_{\{i,f\}}$ we know that $f \in \mathcal{F}_{\{i,U\}}$ implies $U \in \mathcal{U}_{\{i,f\}}$. Therefore, outgoing edges from the node corresponding to f are the edges between f and the nodes $U \in \mathcal{U}_{\{i,f\}} \cap \mathcal{U}_{\text{sent}}(\tau_i)$. Similarly, the outgoing edges between node $U \in \mathcal{U}_{\text{sent}}(\tau_i)$ with $U \ni i$ are the edges between U and $f \in \mathcal{F}_{\{i,U\}}$. By counting $\tilde{y}_{\{i,f\}}(U)$ in two ways, from the left and right nodes, we have the required equality. Therefore, we have that $\sum_{U \in \mathcal{U}_{\{i,f\}} \cap \mathcal{U}_{\text{sent}}(\tau_i)} \tilde{y}_{\{i,f\}}(U) = 1$ for any $f \in \Omega^{(i)}$. This further implies that $\sum_{U \in \mathcal{U}_{\{i,f\}}} y_{\{i,f\}}(U) = 1$ for all $f \in \Omega^{(i)}$ and ends the proof.

The following lemma shows that if Algorithm 5 does not return "`INFEASIBLE`" then with high probability each user recovers its missing subfiles from the transmitted equations.

Lemma 5 *If Algorithm 5 does not return "`INFEASIBLE`" then with probability at least* $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{rKF}$ *all requests will be satisfied within their deadline.*

Proof: For simplicity, in the discussion below we assume that $r = 1$. The proof for $r > 1$ follows in straightforward manner. By the way that $\mathcal{M}_i$ and $v_i(\tau)$ are updated in Algorithm 5, we have $|\mathcal{M}_i| = v_i(\tau)$ at each time τ. Furthermore, $v_i(T_{\max}) = |\Omega^{(i)}|$ for all $i \in [K]$. Therefore, each user $i \in [K]$ benefits from $|\Omega^{(i)}|$ equations. For a $m \in \mathcal{M}_i$, let $\bigoplus_{i \in U} \bigoplus_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$ represent the m-th equation (the dependence on index j is suppressed since we assume that $r = 1$). User $i \in U$ can recover $\bigoplus_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$ from this equation since the missing subfiles $W_{\{d_j,f'\}}$, for $f' \in \mathcal{F}_{\{j,U\}}$ and $j \in U \setminus \{i\}$, exist in the cache of user i.

For each user $i \in [K]$ we define matrix $\mathbf{B}_i \in \mathbb{F}^{|\Omega^{(i)}| \times |\Omega^{(i)}|}$ whose rows and columns correspond to equation numbers in $\mathcal{M}_i$ and missing subfiles in $\Omega^{(i)}$ respectively. For $m \in \mathcal{M}_i$, assume that m-th equation is associated with user group U, where $i \in U$. Then, the entry of $\mathbf{B}_i$ for the row and column corresponding to $m \in \mathcal{M}_i$ and $f \in \Omega^{(i)}$ is $\alpha_{\{i,f,m\}}$ if $f \in \mathcal{F}_{\{i,U\}}$ and zero otherwise. Therefore, if matrix $\mathbf{B}_i$ is invertible then user i can recover all the missing subfiles $W_{\{d_i,f\}}$, for $f \in \Omega^{(i)}$, from equations $\sum_{f \in \mathcal{F}_{\{i,U\}}} \alpha_{\{i,f,m\}} W_{\{d_i,f\}}$ for $m \in \mathcal{M}_i$. Thus, we need to show that the determinant of $\mathbf{B}_i$ is nonzero for all $i \in [K]$ with high probability.

Towards this end, let $h_i(\{\alpha_{\{i,f,m\}},\ f \in \Omega^{(i)},\ m \in \mathcal{M}_i\})$ denote the determinant of $\mathbf{B}_i$; we treat the $\{\alpha_{\{i,f,m\}},\ f \in \Omega^{(i)},\ m \in \mathcal{M}_i\}$ as indeterminates at this point. Note that since Algorithm 5 did not return "`INFEASIBLE`", we have a feasible integral solution for the corresponding offline LP (*cf.* Claim 9). Thus, there exists an interpretation of this solution (*cf.* Section 3.3.2) such that in each time slot, only one equation is transmitted, i.e., unlike a fractional solution, we do not need to potentially transmit multiple equations in the same time slot. This in turn implies that there is a setting for coefficients $\alpha_{\{i,f,m\}}$ with $\alpha_{\{i,f,m\}} \in \{0, 1\}$ such that the multivariate polynomial h_i evaluates to a non-zero value over $\mathbb{F}$, i.e., h_i is not identically zero. This further implies that $h = \prod_{i \in [K]} h_i$ is not identically zero. Now, since each $\alpha_{\{i,f,m\}}$ appears only once in $\mathbf{B}_i$ thus its degree in polynomial h_i is one. Also, h_i is a polynomial of degree $|\Omega^{(i)}| \leq F$ thus h is a polynomial of degree at most KF. Therefore, we can use Lemma 4 in Ho et al. (2006) to show that by choosing $\alpha_{\{i,f,m\}}$'s independently and uniformly at random from $\mathbb{F}$, the determinants of $\mathbf{B}_i$'s, $i \in [K]$, are nonzero with probability at least $\left(1 - \frac{1}{|\mathbb{F}|}\right)^{KF}$.

Table 3.2: Execution time for solving the LP using our approach; we run 1000 iterations of subgradient ascent. Columns 2 & 3 indicate the size of the associated flow network. The table is ordered by the number of nodes in the flow network.

(K, t)	No. of nodes	No. of edges	Execution time (minutes)	Execution time Orig. (minutes)
$(100, 2)$	$986,161$	$17,643,986$	$8,026$	—
$(20, 4)$	$178,542$	$1,778,703$	1065	—
$(40, 2)$	$61,959$	$567,780$	63	—
$(20, 2)$	$7,542$	$43,507$	1.9	21.9
$(10, 4)$	$3,917$	$29,369$	0.8	5.33
$(10, 2)$	915	$3,866$	0.08	0.03

When $r > 1$ we will need to split a missing subfile $W_{\{d_i,f\}}$ into r sub-packets and code over these as well. Thus, the corresponding system of equations will be of size $\mathbb{F}^{r|\Omega^{(i)}|\times r|\Omega^{(i)}|}$ leading to the bound $\left(1-\frac{1}{|\mathbb{F}|}\right)^{rKF}$. Thus, by choosing $|\mathbb{F}|$ large enough, we can make the probability of success as large as we want.

3.5 Simulation Results and Comparisons with Prior Work

In this section we present simulation results for both the proposed offline and the online algorithms. Prior work in this area is primarily the work of Niesen and Maddah-Ali (2015) that presents heuristics for the online scenario. However, we note that Niesen and Maddah-Ali (2015) works with deadlines for subfiles and does not take into account the time take to transmit a packet. It uses intuitively plausible rules to decide the equations transmitted by the server depending on the deadlines of the users.

For both scenarios, the request arrival times $\{T_i,\ i \in [K]\}$ are generated according to a Poisson process with parameter λF. The arrival time is quantized to the nearest time slot. The deadlines Δ_i, $i \in [K]$ are generated uniformly at random from the range $[\Delta_{\min}, \Delta_{\max}]$ (these values will be specified for each setting below).

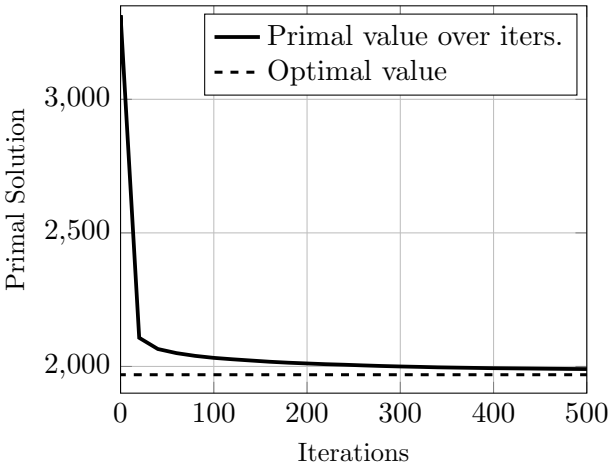


Figure 3.8: Convergence of primal recovery to the optimal solution for a system with $N = K = 20$, $r = 1$, and $t = 2$. Dashed line is the optimal value obtained by solving (3.1).

3.5.1 Offline scenario simulation

In the first set of simulations we examine the execution time of our approach for various values of (K, t) where $t = KM/N$ is an integer; the placement scheme in Maddah-Ali and Niesen (2014) was used. In these simulations we set $r = 1$, $\lambda = 0.4$, $F = \binom{K}{t}$, $\Delta_{\min} = \binom{K-1}{t}$, and $\Delta_{\max} = \binom{K}{t+1}$. Table 3.2 shows the details of the overall execution time and the size of the corresponding flow networks for the various instances. The last column of the table corresponds to the execution time (in MATLAB) of the LP in (3.1), while the second-last column corresponds to the execution time of the proposed approach above. It is evident that the proposed approach is significantly faster. In fact, memory requirements make it infeasible to even formulate the problems corresponding to the first three rows in MATLAB. Figure 3.8 shows the convergence of the primal recovery procedure to the actual rate for a system with $N = K = 20$, $t = 2$, and $r = 1$. It can be observed that there is a clear convergence of the solution to the optimal value.

3.5.2 Online scenario simulation

For the online scenario we consider both centralized Maddah-Ali and Niesen (2014) and decentralized Maddah-Ali and Niesen (2015) placement schemes for a system with $N = K = 6$ and $M = 2$ with $\Delta_{\min} = (KM/N)F$ and $\Delta_{\max} = KF$. For each experiment we run 200 trials for generating the

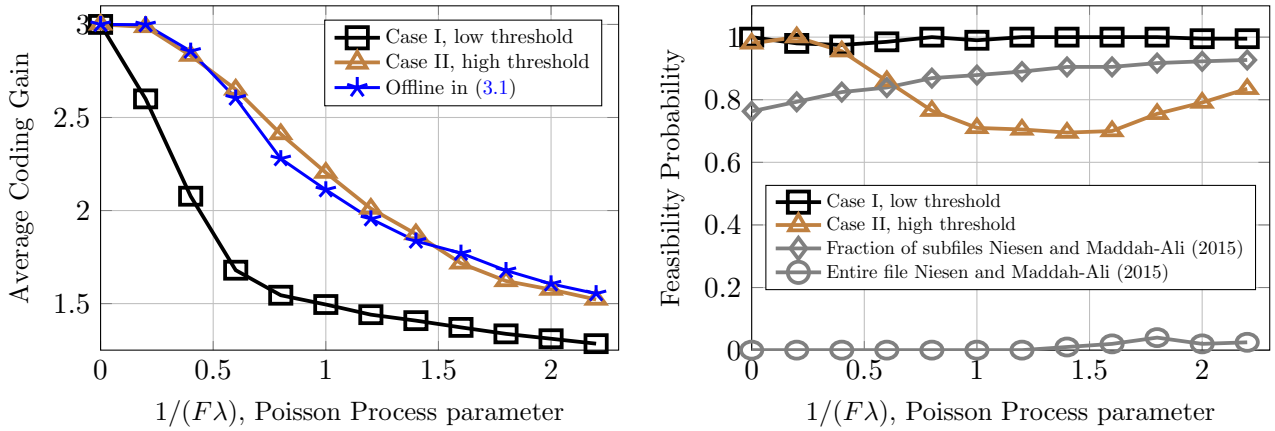


Figure 3.9: Centralized Placement in Maddah-Ali and Niesen (2014): (a) average coding gain over all feasible offline problem instances, (b) feasibility probability of the online algorithm conditioned on feasibility of the offline problem. The placement has been fixed for all trials and at each trial a new arrival time and deadline is generated. In this simulation, we set $\eta_0 = 0.4 - \frac{0.5}{\lambda}$ and $\eta_0 = 0.8 - \frac{0.2}{\lambda}$ in Case I and II respectively.

arrivals. For the centralized case, we use the placement scheme of Maddah-Ali and Niesen (2014) and the placement is fixed during each experiment. In the decentralized scheme, at each trial the cache content of each user is independently and uniformly chosen as well.

For each set of generated arrivals, we first run the offline LP to check whether it is feasible. The online algorithm is run only if the offline LP is feasible. The online algorithm requires a threshold η_0 (see Section 3.4.2). We run simulations with a low threshold (case I) and a high threshold (case II). The coding gain is defined as the ratio of the uncoded rate[3] to the rate achieved by the system. Figure 3.9 (a) and Figure 3.10 (a) depict plots of the coding gain vs. $1/(F\lambda)$ in centralized and decentralized cases, respectively. As λ decreases, the arrivals are spaced further apart on average, and the coding gain of any scheme is expected to reduce. The coding gain is computed by taking an average overall all instances where a given scheme is feasible. For the offline scheme this means that we take the average of all instances where it is feasible. For the case II of the online algorithm, some of arrival patterns may result in infeasibility; these instances were not taken into account when computing the average coding gain. This explains why the coding gain of case II sometimes

[3]The uncoded rate is simply the total number of missing subfiles of the all users normalized by F.

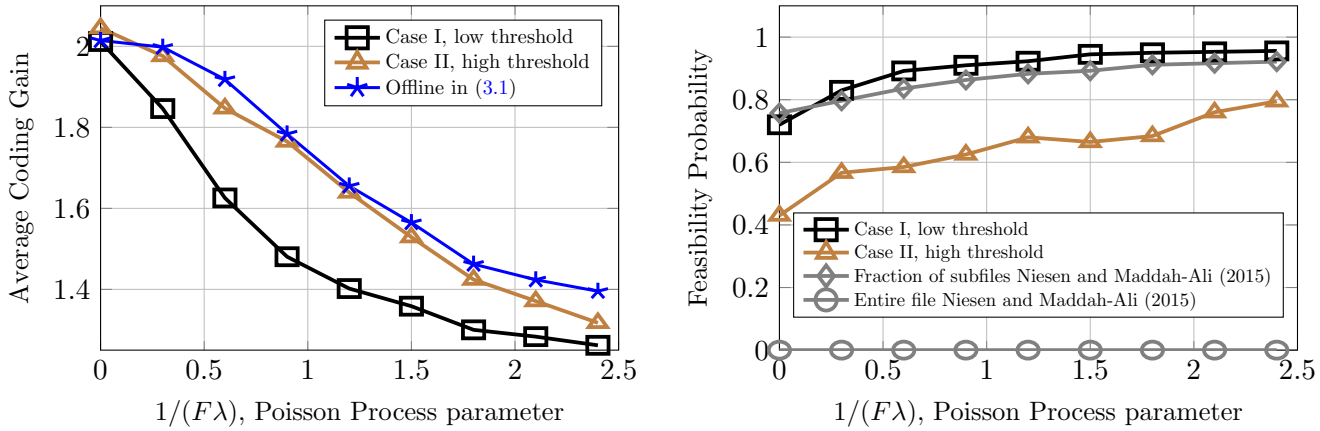


Figure 3.10: Decentralized placement scheme for $N = K = 6$, $M = 2$, and $F = 100$: (a) average coding gain over all feasible offline problem instances, (b) feasibility probability of the online algorithm conditioned on feasibility of the offline problem. At each trial cache content of each user is placed randomly and uniformly. In this simulation, we set $\eta_0 = 0.4 - \frac{0.5}{\lambda}$ and $\eta_0 = 0.8 - \frac{0.2}{\lambda}$ in Case I and Case II respectively.

appears to be higher than the offline algorithm. However, the coding gain of case I is significantly lower, because of its low threshold.

The feasibility probability of a scheme vs. the arrival rate is plotted in Figure 3.9 (b) and Figure 3.10 (b) for the centralized and decentralized placement schemes respectively. As expected the low threshold online algorithm has a very high feasibility probability ≈ 1 for a range of arrival parameters, while the high threshold algorithm has a lower feasibility probability.

For both plots, we also include the results of Niesen and Maddah-Ali (2015). In this scheme feasibility and coding gain can be traded off by setting a threshold for the defined misfit function (Section III in Niesen and Maddah-Ali (2015)). We use this scheme by setting the threshold to zero; this is the so-called First-Fit Rule in Niesen and Maddah-Ali (2015). The First-Fit rule prefers feasibility over coding gain. The setting in Niesen and Maddah-Ali (2015) considers a scenario where each subfile has a deadline. We have adapted their algorithm for our case. It can be observed that the feasibility probability of Niesen and Maddah-Ali (2015) is quite poor. Accordingly we also plot the fraction of subfiles that meet the deadline; this is somewhat better. The coding gain numbers

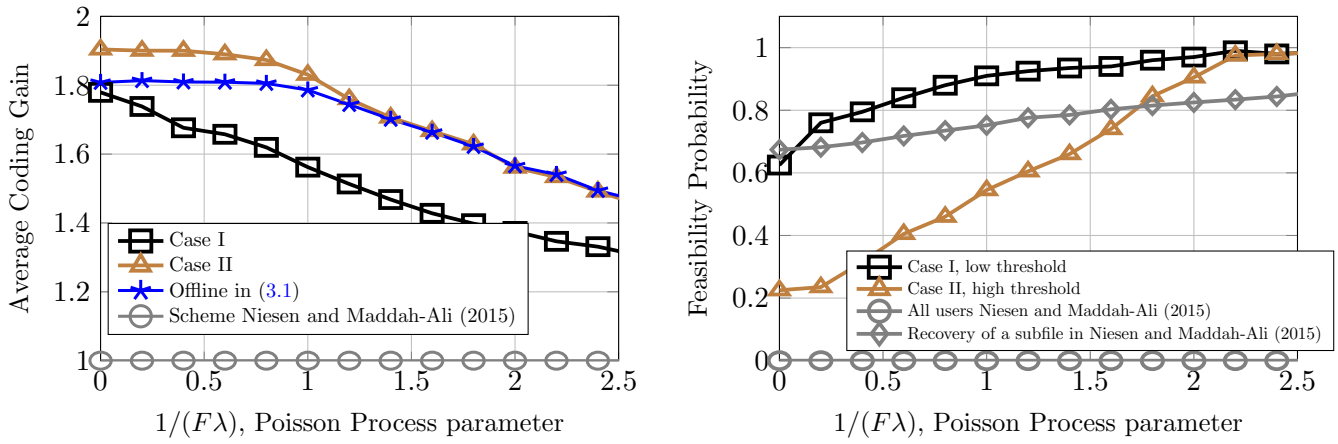


Figure 3.11: Decentralized placement scheme with single chunk request for $K = N = 6$, $M = 2$, and $F = 20$: (a) average coding gain over all feasible offline problem instances, (b) feasibility probability of the online algorithm conditioned on feasibility of the offline problem. For the scheme in Niesen and Maddah-Ali (2015) two probabilities are reported. The first one is the probability that all requests are satisfied, and the second one is the probability that a fixed request is satisfied (lines with circle and diamond marks respectively). At each trial, the cache content of each user is populated randomly and uniformly. In this simulation, we set $\eta_0 = 0.4 - \frac{0.5}{\lambda}$ and $\eta_0 = 0.8 - \frac{0.2}{\lambda}$ in Case I and Case II respectively.

for Niesen and Maddah-Ali (2015) are also quite unreliable as the algorithm is infeasible in most cases. Thus, we do not plot it.

3.5.3 Scenario where individual subfiles have deadlines

The work of Niesen and Maddah-Ali (2015) considers a situation where each subfile has its own deadline. This is inspired by applications such as video delivery over the Internet. We emphasize that this setting can be captured by our techniques. In particular, suppose that each user requests a set of subfiles from the server where the subfile requests arrive at different times and each subfile has a different deadline. In this case we can treat each subfile request of user $i \in [K]$ as corresponding to a distinct virtual user whose cache content is the same as user i. However, the requests of the users are different. In this situation, each virtual user has precisely one missing subfile. Thus, the issue of coding over the corresponding subfiles does not arise.

Our setting is again one where $K = N = 6$, $M = 2$. Each file is subdivided into $F = 20$ subfiles. Arrival times and deadlines are generated similar to the previous simulations with Poisson parameters $1/(\lambda F)$ and the deadlines are randomly chosen uniformly from $[\Delta_{\min}, \Delta_{\max}]$ with $\Delta_{\min} = KMF/N$ and $\Delta_{\max} = KF$. Similar to the previous experiments we run 200 trials and at each trial, the cache content of each user is populated randomly and uniformly among all placement schemes with cache of size MF subfiles. Thus, different users might request different number of chunks from the server. The only difference is that here each requested chunk has its own arrival time and deadline. The results are illustrated in Figure 3.11. It can be observed that our proposed approach provides significantly superior coding gain and feasibility probability as compared to the work of Niesen and Maddah-Ali (2015).

3.6 Conclusions and Future Work

In this work we considered the asynchronous coded caching problem where user requests (with deadlines) arrive at the main server at different times. We considered both offline and online versions of this problem. We demonstrated that under the assumption of all but one equations, the offline scenario can be solved by a linear program (LP). Moreover, we presented a low-complexity solution to this LP based on dual decomposition. In contrast to the synchronous case and the offline scenario, we show that the online scenario requires coding across missing subfiles of a given user. Furthermore, we present an online algorithm that leverages the offline LP in a recursive fashion. Extensive simulation results indicate that our proposed algorithm significantly outperforms prior algorithms.

Our online algorithm considers the situation where there is no knowledge about future request arrival times and file identities; this corresponds to a worst-case scenario. It would be interesting to consider cases where there is statistical information available on the arrival times and file popularity and investigate how this knowledge can be used to further improve the performance of the algorithm. Variants of the problem, with soft deadline constraints may also be of interest.

REFERENCES

Ahuja, R. K., Maganti, T. L., and Orlin, J. B. (1993). *Network Flows:Theory, Algorithms and Applications*. Prentice-Hall.

Ajaykrishnan, N., Prem, N. S., Prabhakaran, V. M., and Vaze, R. (2015). Critical database size for effective caching. In *IEEE 2015 Twenty First National Conf. on Comm.*, pages 1–6.

Applegate, D., Archer, A., Gopalakrishnan, V., Lee, S., and Ramakrishnan, K. K. (2010). Optimal content placement for a large-scale vod system. In *Proc. ACM 6th Intl. Conf. on Emerging Networking Experiments and Technologies (Co-NEXT)*.

Arbabjolfaei, F., Kim, Y.-H., et al. (2018). Fundamentals of index coding. *Foundations and Trends® in Communications and Information Theory*, 14(3-4):163–346.

Bar-Yossef, Z., Birk, Y., Jayram, T., and Kol, T. (2011). Index coding with side information. *IEEE Trans. on Info. Th.*, 57(3):1479–1494.

Borst, S. C., Gupta, V., and Walid, A. (2010). Distributed caching algorithms for content distribution networks. In *Proc. IEEE INFOCOM*, pages 1478–1486.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1999). Web caching and zipf-like distributions: evidence and implications. In *Proc. IEEE INFOCOM*, pages 126–134.

Cover, T. M. and Thomas, J. A. (2012). *Elements of information theory*. John Wiley & Sons.

Ghasemi, H. and Ramamoorthy, A. (2015). Improved lower bounds for coded caching. In *IEEE Intl. Symp. on Info. Th.*, pages 1696–1700.

Ghasemi, H. and Ramamoorthy, A. (2016). Further results on lower bounds for coded caching. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 2319–2323.

Ghasemi, H. and Ramamoorthy, A. (2017a). Algorithms for asynchronous coded caching. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 636–640.

Ghasemi, H. and Ramamoorthy, A. (2017b). Asynchronous coded caching. *IEEE Intl. Symp. on Info. Th.*, pages 2438–2442.

Ghasemi, H. and Ramamoorthy, A. (2017c). Improved lower bounds for coded caching. *IEEE Trans. on Info. Th.*, 63(7):4388–4413.

Golrezaei, N., Molisch, A., Dimakis, A., and Caire, G. (2013). Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution. *IEEE Comm. Magazine*, 51(4):142–149.

Hachem, J., Karamchandani, N., and Diggavi, S. (2014a). Multi-level coded caching. In *IEEE Intl. Symp. on Info. Th.*, pages 56–60.

Hachem, J., Karamchandani, N., and Diggavi, S. N. (2014b). Coded caching for heterogeneous wireless networks with multi-level access.

Ho, T., Medard, M., Koetter, R., Karger, D. R., Effros, M., Shi, J., and Leong, B. (2006). A random linear network coding approach to multicast. *IEEE Trans. on Info. Th.*, 52(10):4413–4430.

Ji, M., Caire, G., and Molisch, A. F. (2013). Fundamental limits of distributed caching in d2d wireless networks. In *IEEE Info. Th. Workshop*, pages 1–5.

Ji, M., Tulino, A. M., Llorca, J., and Caire, G. (2014a). Order optimal coded caching-aided multicast under zipf demand distributions. In *The 11th Intl. Symp. on Wireless Comm. Sys.*

Ji, M., Tulino, A. M., Llorca, J., and Caire, G. (2014b). Order optimal coded delivery and caching: Multiple groupcast index coding.

Jiang, Y., Huang, W., Bennis, M., and Zheng, F. (2019 (to appear)). Decentralized asynchronous coded caching design and performance analysis in fog radio access networks. *IEEE Trans. on Mob. Comput.*

Karamchandani, N., Niesen, U., Maddah-Ali, M., and Diggavi, S. (2014). Hierarchical coded caching. In *IEEE Intl. Symp. on Info. Th.*, pages 2142–2146.

Kleinberg, J. and Tardos, E. (2006). *Algorithm design*.

Korupolu, M. R., Plaxton, C. G., and Rajaraman, R. (1999). Placement algorithms for hierarchical cooperative caching. In *Proc. ACM-SIAM SODA*, pages 586–595.

Kovacs, P. (2015). Minimum-cost flow algorithms: an experimental evaluation. *Optimization Methods and Software*, 30(1):94–127.

Lampiris, E. and Elia, P. (2018). Adding transmitters dramatically boosts coded-caching gains for finite file sizes. *IEEE J. Select. Areas Comm.*, 36(6):1176–1188.

Lee, K., Lam, M., Pedarsani, R., Papailiopoulos, D., and Ramchandran, K. (2015). Speeding up distributed machine learning using codes.

LEMON. Library for efficient modeling and optimization in networks.

Li, S., Maddah-Ali, M. A., Yu, Q., and Avestimehr, A. S. (2016). A fundamental tradeoff between computation and communication in distributed computing.

Lu, C., Stankovic, J. A., Tao, G., and Son, S. H. (1999). Design and evaluation of a feedback control edf scheduling algorithm. In *Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No. 99CB37054)*, pages 56–67. IEEE.

Lu, Y., Chen, W., and Poor, H. V. (2018). Coded joint pushing and caching with asynchronous user requests. *IEEE J. Select. Areas Comm.*, 36(8):1843–1856.

Lubetzky, E. and Stav, U. (2009). Nonlinear index coding outperforming the linear optimum. *IEEE Trans. on Info. Th.*, 55(8):3544–3551.

Lun, D. S., Ratnakar, N., Médard, M., Koetter, R., Karger, D. R., Ho, T., Ahmed, E., and Zhao, F. (2006). Minimum-cost multicast over coded packet networks. *IEEE Trans. on Info. Th.*, 52(6):2608–2623.

Maddah-Ali, M. and Niesen, U. (2014). Fundamental limits of caching. *IEEE Trans. on Info. Th.*, 60(5):2856–2867.

Maddah-Ali, M. A. and Niesen, U. (2015). Decentralized coded caching attains order-optimal memory-rate tradeoff. *IEEE/ACM Trans. Netw.*, 23(4):1029–1040.

Meyerson, A., Munagala, K., and Plotkin, S. (2001). Web caching using access statistics. In *Proc. ACM-SIAM SODA*, pages 354–363.

Naderializadeh, N., Maddah-Ali, M. A., and Avestimehr, A. S. (2017). On the optimality of separation between caching and delivery in general cache networks. In *IEEE Intl. Symp. on Info. Th.*, pages 1232–1236.

Niesen, U. and Maddah-Ali, M. (2016). Coded caching with nonuniform demands. *IEEE Trans. on Info. Th.*, 63(2):1146–1158.

Niesen, U. and Maddah-Ali, M. A. (2015). Coded caching for delay-sensitive content. In *IEEE Intl. Conf. Comm.*, pages 5559–5564.

Pedarsani, R., Maddah-Ali, M., and Niesen, U. (2014). Online coded caching. In *IEEE Intl. Conf. Comm.*, pages 1878–1883.

Ramamoorthy, A. (2011). Minimum cost distributed source coding over a network. *IEEE Trans. on Info. Th.*, 57(1):461–475.

Ramamritham, K., Stankovic, J. A., and Shiah, P.-F. (1990). Efficient scheduling algorithms for real-time multiprocessor systems. *IEEE Transactions on Parallel and Distributed systems*, 1(2):184–194.

Sengupta, A. and Tandon, R. (2015). Beyond cut-set bounds-the approximate capacity of d2d networks. In *IEEE Info. Th. Workshop*, pages 78–83.

Sengupta, A., Tandon, R., and Clancy, T. C. (2015a). Fundamental limits of caching with secure delivery. *IEEE Trans. on Info. Forensics and Security*, 10(2):355–370.

Sengupta, A., Tandon, R., and Clancy, T. C. (2015b). Improved approximation of storage-rate tradeoff for caching via new outer bounds. In *IEEE Intl. Symp. on Info. Th.*, pages 1691–1695. IEEE.

Shanmugam, K., Golrezaei, N., Dimakis, A., Molisch, A., and Caire, G. (2013). Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE Trans. on Info. Th.*, 59(12):8402–8413.

Sherali, H. D. and Choi, G. (1996). Recovery of primal solutions when using subgradient optimization methods to solve Lagrangian duals of linear programs. *Operations Research Letters*, 19(3):105–113.

Spuri, M. and Buttazzo, G. C. (1994). Efficient aperiodic service under earliest deadline scheduling. In *RTSS*, pages 2–11.

Tan, B. and Massoulié, L. (2013). Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM Trans. on Netw.*, 21(2):566–579.

Tang, L. and Ramamoorthy, A. (2016a). Coded caching for networks with the resolvability property. In *IEEE Intl. Symp. on Info. Th.*

Tang, L. and Ramamoorthy, A. (2016b). Coded Caching with Low Subpacketization Levels. In *Workshop on Network Coding (NetCod)*.

Tang, L. and Ramamoorthy, A. (2017). Low Subpacketization Level Schemes for Coded Caching. In *IEEE Intl. Symp. on Info. Th.*

Tang, L. and Ramamoorthy, A. (2018). Coded caching schemes with reduced subpacketization from linear block codes. *IEEE Trans. on Info. Th.*, 64(4):3099–3120.

Tian, C. (2015). A note on the fundamental limits of coded caching.

Wan, K., Ji, M., Piantanida, P., and Tuninetti, D. (2018). Caching in combination networks: Novel multicast message generation and delivery by leveraging the network topology. In *IEEE Intl. Conf. Comm.*, pages 1–6.

Wessels, D. (2001). *Web Caching*. O' Reilly.

Wolman, A., Voelker, M., Sharma, N., Cardwell, N., Karlin, A., and Levy, H. M. (1999). On the scale and performance of cooperative web proxy caching. *ACM SIGOPS*, 33(5):16–31.

Yan, Q., Cheng, M., Tang, X., and Chen, Q. (2015 [Online] Available: http://arxiv.org/abs/1510.05064). On the placement delivery array design in centralized coded caching scheme.

Yan, Q., Cheng, M., Tang, X., and Chen, Q. (2017). On the placement delivery array design for centralized coded caching scheme. *IEEE Trans. on Info. Th.*, 63(9):5821–5833.

Yang, Q., Amiri, M. M., and Gündüz, D. (2019). Audience-retention-rate-aware caching and coded video delivery with asynchronous demands.

Yu, Q., Maddah-Ali, M. A., and Avestimehr, A. S. (2017). The exact rate-memory tradeoff for caching with uncoded prefetching. *IEEE Trans. on Info. Th.*, 64(2):1281–1296.

Yu, Q., Maddah-Ali, M. A., and Avestimehr, A. S. (2019). Characterizing the rate-memory tradeoff in cache networks within a factor of 2. *IEEE Trans. on Info. Th.*, 65(1):647–663.

Zhang, J., Lin, X., Wang, C.-C., and Wang, X. (2015). Coded caching for files with distinct file sizes. In *IEEE Intl. Symp. on Info. Th.*, pages 1686–1690.

APPENDIX A. PROOFS FOR LOWER BOUNDS

Lemma 6 *Algorithm 1 always provides a valid lower bound on* $\alpha R^\star + \beta M$ *where* $\alpha = \sum_{i=1}^{\ell} |\mathbb{D}(v_i)|$ *and* $\beta = \sum_{i=1}^{\ell} |\mathbb{Z}(v_i)|$.

Proof: Consider any internal node $v \in \mathcal{T}$. We have

$$\begin{aligned}
&\sum_{u \in in(v)} H(\mathbb{Z}(u) \cup \mathbb{D}(u) | \mathbb{W}(u) \cup W_{new}(u)), \\
&\overset{(a)}{\geq} \sum_{u \in in(v)} H(\mathbb{Z}(u) \cup \mathbb{D}(u) | \mathbb{W}(v)), \\
&\overset{(b)}{\geq} H(\mathbb{Z}(v) \cup \mathbb{D}(v) | \mathbb{W}(v)), \\
&\overset{(c)}{=} I(W_{new}(v); \mathbb{Z}(v) \cup \mathbb{D}(v) | \mathbb{W}(v)) \\
&+ H(\mathbb{Z}(v) \cup \mathbb{D}(v) | \mathbb{W}(v) \cup W_{new}(v)),
\end{aligned}$$

where inequality in (a) holds since $\mathbb{W}(u) \cup W_{new}(u) \subseteq \mathbb{W}(v)$ and conditioning reduces entropy, (b) holds since $\cup_{u \in in(v)} \mathbb{Z}(u) = \mathbb{Z}(v)$ and $\cup_{u \in in(v)} \mathbb{D}(u) = \mathbb{D}(v)$ and (c) holds by the definition of mutual information. Let V_{int} denote the set of internal nodes in $\mathcal{T}$. Let v^* denote the root and (u^*, v^*) denote its incoming edge. Then,

$$\begin{aligned}
&\sum_{v \in V_{int}} \sum_{u \in in(v)} H(\mathbb{Z}(u) \cup \mathbb{D}(u) | \mathbb{W}(u) \cup W_{new}(u)) \geq \\
&\sum_{v \in V_{int}} y_{(v, out(v))} + \sum_{v \in V_{int}} H(\mathbb{Z}(v) \cup \mathbb{D}(v) | \mathbb{W}(v) \cup W_{new}(v)),
\end{aligned}$$

where we have ignored the infinitesimal terms introduced due to Fano's inequality (for convenience of presentation). Note that the RHS of the inequality above contains terms of the form $H(\mathbb{Z}(v) \cup \mathbb{D}(v) | \mathbb{W}(v) \cup W_{new}(v))$ for all nodes $v \in V_{int}$ (including u^*).

On the other hand the LHS contains terms of a similar form for all nodes including the leaf nodes but excluding the node u^*. Canceling the common terms, we obtain,

$$\sum_{i=1}^{\ell} H(\mathbb{Z}(v_i) \cup \mathbb{D}(v_i) | W_{new}(v_i)) \geq$$
$$\left(\sum_{v \in V_i} y_{(v,out(v))} \right) + H(Z \cup \mathbb{D}(u^*) | \mathbb{W}(u^*), W_{new}(u^*)),$$

since $\mathbb{W}(v_i) = \phi$ for $i = 1, \ldots, \ell$. We can therefore conclude that

$$\sum_{i=1}^{\ell} H(\mathbb{Z}(v_i), \mathbb{D}(v_i)) \geq \sum_{v \in V} y_{(v,out(v))} \tag{A.1}$$

$$\implies \sum_{i=1}^{\ell} H(\mathbb{Z}(v_i)) + \sum_{i=1}^{\ell} H(\mathbb{D}(v_i)) \geq \sum_{v \in V} y_{(v,out(v))} \tag{A.2}$$

Noting that $M \geq H(\mathbb{Z}(v_i))$ and $R^\star \geq H(\mathbb{D}(v_i))$ we have the required result.

A.0.1 Proof of Claim 1

Proof: We iteratively modify the problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ to arrive at an instance where every node has in-degree at most two. Towards this end, we first identify a node u with in-degree $\delta \geq 3$ such that no other node is topologically higher than it (such a node may not be unique).

We modify the instance P by replacing u with a directed in-tree where each node has in-degree exactly two. Specifically, arbitrarily number the nodes in $in(u)$ from $v'_1, \ldots, v'_\delta$. We replace the node u with a directed in-tree $\mathcal{T}_u$ with leaves $v'_1, \ldots, v'_\delta$ and root u. $\mathcal{T}_u$ has $\delta - 2$ internal nodes numbered $u'_1, \ldots, u'_{\delta-2}$ such that $in(u'_i) = \{u'_{i-1}, v'_{i+1}\}$ where $u'_0 = v'_1$ (see Fig. A.1). Let us denote the new instance by $P_o = P_o(\mathcal{T}_o, \alpha, \beta, L_o, N, K)$. We claim that $L_o \geq L$. To see this, suppose that $W^* \in W^P_{new}(u)$. We show that $W^* \in \cup_{u' \in \mathcal{T}_u} W^{P_o}_{new}(u')$. This ensures that $L_o \geq L$. To see this we note that

$$\mathbb{Z}^P(u) = \mathbb{Z}^{P_o}(u)$$
$$\mathbb{D}^P(u) = \mathbb{D}^{P_o}(u), \text{ and thus,}$$
$$\Delta^P(u,u) = \Delta^{P_o}(u,u).$$

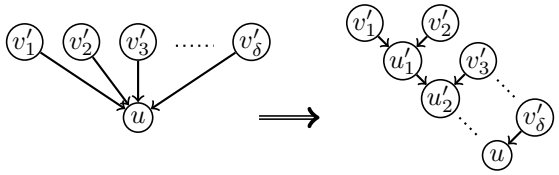


Figure A.1: Tree modification example

Thus, if $W^* \in W^P_{new}(u)$, there exists an internal node $u'_i \in \mathcal{T}_u$ with the smallest index $i \in \{1, \ldots, \delta - 2\}$ such that $W^* \in \Delta^{P_o}(u'_i, u'_i)$. Note that if $i > 1$, we have $W^* \in W^{P_o}_{new}(u'_i)$ since $W^* \notin \Delta^{P_o}(u'_{i-1}, u'_{i-1})$ which in turn implies that $W^* \notin \mathbb{W}^{P_o}(u'_i)$. On the other hand if $i = 1$, then a similar argument holds since it is easy to see that $W^* \notin \mathbb{W}^{P_o}(u'_1)$.

Note that the modification in the instance P can only affect nodes that are downstream of u. Now consider u' such that $u \in in(u')$. It is evident that $\mathbb{Z}^{P_o}(u') = \mathbb{Z}^P(u')$ and $\mathbb{D}^{P_o}(u') = \mathbb{D}^P(u')$. Moreover $\mathbb{W}^{P_o}(u') = \cup_{v \in in(u')} \mathbb{W}^{P_o}(v) \cup W^{P_o}_{new}(v)$. Now for $v \neq u$, $\mathbb{W}^{P_o}(v) = \mathbb{W}^P(v)$ and $W^{P_o}_{new}(v) = W^P_{new}(v)$ as there are no changes in the corresponding subtrees. Moreover, as $\Delta^P(u, u) = \Delta^{P_o}(u, u)$, we have that $\mathbb{W}^{P_o}(u) \cup W^{P_o}_{new}(u) = \mathbb{W}^P(u) \cup W^P_{new}(u)$. This implies that $\mathbb{W}^{P_o}(u') = \mathbb{W}^P(u')$. Thus, we can conclude that $W^{P_o}_{new}(u') = W^P_{new}(u')$. Applying an inductive argument we can conclude that the $W^{P_o}_{new}(u') = W^P_{new}(u')$ for all u' such that $u \succ u'$.

The above process can iteratively be applied to every node in the instance that is of degree at least three. Thus, we have the required result.

A.0.2 Proof of Claim 3

Proof: We identify the set $\mathcal{U}$ as the set of all nodes in $\mathcal{T}$ such that the specified condition in the claim holds. Let $\mathcal{U}^* \subset \mathcal{U}$ denote the set of nodes that are highest in the topological ordering. We modify the instance in a way such that a node $u^* \in \mathcal{U}^*$ can be removed from $\mathcal{U}$, i.e., the specified condition no longer holds for it. Moreover, our modification procedure is such that a node $u \succ u^*$ cannot enter $\mathcal{U}$ at the end of the procedure.

We now discuss the modification procedure. In the discussion below, for a given node u, we can consider the instance obtained with tree $\mathcal{T}_u$. We let β_u denote the number of cache nodes in

this instance. Note that for u^*, the condition $\hat{\beta}^* < \min(\beta^*, K)$ holds. This implies that there is a set of cache leaves in $\mathcal{T}_{u^*}$ denoted $\{v_{i_1}, \ldots, v_{i_m}\}$ such that $\mathbb{Z}(v_{i_1}) = \cdots = \mathbb{Z}(v_{i_m}) = \{Z_j\}$. Let $\Lambda = \{u \in \mathcal{T}_{u^*} : (v_{i_a}, v_{i_b}) \text{ meet at } u, \text{ for all distinct } v_{i_a}, v_{i_b} \in \{v_{i_1}, \ldots, v_{i_m}\}\}$. We identify $u_0 \in \Lambda$ such that no element of Λ is topologically higher than u_0 (note that u_0 may not be unique) and let $v^*_{i_a}$ and $v^*_{i_b}$ be one pair of the corresponding nodes in $\{v_{i_1}, \ldots, v_{i_m}\}$ that meet at u_0. W.l.o.g we assume that $v^*_{i_b} \in \mathcal{T}_{u_0(r)}$ and $v^*_{i_a} \in \mathcal{T}_{u_0(l)}$.

We claim that $u_0 = u^*$. Assume that this is not the case. Since $u_0 \in \mathcal{T}_{u^*}$ we have $u_0 \succeq u^*$. Using this and the fact that $u_0 \notin \mathcal{U}$ we have $|\cup_{v \in \mathcal{C}_{u_0}} \mathbb{Z}(v)| = \min(|\mathcal{C}_{u_0}|, K)$. Now, from $v^*_{i_a}, v^*_{i_b} \in \mathcal{C}_{u_0}$ and that $\mathbb{Z}(v^*_{i_a}) = \mathbb{Z}(v^*_{i_b})$ we conclude that $\min(|\mathcal{C}_{u_0}|, K) = K$. Moreover, as $\cup_{u \in \mathcal{T}_{u_0}} \mathbb{Z}(u) \subseteq \cup_{u \in \mathcal{T}_{u^*}} \mathbb{Z}(u)$ we have $\hat{\beta} = K$ which contradicts $\hat{\beta} < \min(\beta, K)$. Therefore $u_0 = u^*$.

We construct instance P' (with lower bound L') as follows. Choose a member of $\{Z_1, \ldots, Z_K\} \setminus \{\mathbb{Z}(v') : v' \in \mathcal{C}_{u^*}\}$ and denote it by Z_k. We set $\mathbb{Z}^{P'}(v^*_{i_b}) = \{Z_k\}$. Also, for any $u \in \mathcal{D}_{u_0(r)}$ and $\mathbb{D}^P(u) = X_{d_1,\ldots,d_K}$ we set $\mathbb{D}^{P'}(u) = X_{d'_1,\ldots,d'_K}$ such that $d'_j = d_k$ and $d'_k = d_j$ and $d'_i = d_i$ for $i \notin \{j, k\}$, i.e., we interchange the j-th and k-th labels and keep the other labels the same. With this modification, it can be seen that $\hat{\beta}^* = \min(\beta^*, K)$.

For nodes $u \succ u^*$, the change we applied to cache nodes in $\mathcal{C}_{u^*}$ to get P' is such that $\hat{\beta}_u$ continues to equal $\min(\beta_u, K)$ since Z_k is chosen from $\{Z_1, \ldots, Z_K\} \setminus \{\mathbb{Z}(v') : v' \in \mathcal{C}_{u^*}\}$

We now show that $L' \geq L$. In particular, for $u \in \mathcal{T}_{u_0(l)}$, we have $W^{P'}_{new}(u) = W^P_{new}(u)$, as there are no changes in the corresponding labels. Also we claim that $W^{P'}_{new}(u) = W^P_{new}(u)$ for $u \in \mathcal{T}_{u_0(r)}$. To see this, note that for $v \in \mathcal{D}_{u_0(r)}$ and $v' \in \mathcal{C}_{u_0(r)}$ we have $\Delta^{P'}(v', v) = \Delta^P(v', v)$ if $\mathbb{Z}(v') \notin \{Z_j, Z_k\}$. If $\mathbb{Z}^{P'}(v') = \{Z_k\}$ and $\mathbb{D}^{P'}(v) = X_{d'_1,\ldots,d'_K}$ then,

$$\begin{aligned}
\Delta^{P'}(v', v) &= Rec(\{Z_k\}, \{X_{d'_1,\ldots,d'_K}\}) \\
&= \{W_{d'_k}\} = \{W_{d_j}\} \\
&= Rec(\{Z_j\}, \{X_{d_1,\ldots,d_K}\}) \\
&= \Delta^P(v', v).
\end{aligned}$$

Furthermore, note that there does not exist any $v' \in \mathcal{C}_{u_0(r)}$ such that $\mathbb{Z}(v') = \{Z_j\}$ since we picked u_0 such that no element of Λ is topologically higher than u_0. From eq. (2.5) and (2.6), it is not hard to see that this in turn implies that $W^{P'}_{new}(u) = W^{P}_{new}(u)$ for $u \in \mathcal{T}_{u_0(r)}$.

It follows therefore that $\mathbb{W}^{P'}(u_0) = \mathbb{W}^{P}(u_0)$ (from eq. (2.6)). Let us now consider the other nodes. As the changes are applied only to $\mathcal{T}_{u_0(r)}$ so $label(u)$ changes only for nodes u such that $u_0 \succ u$. Consider the subset of internal nodes $U = \{u_0, u_1, \ldots, u_t\}$ such that (u_i, u_{i+1}) is an edge, i.e., the set of internal nodes including u_0 and all nodes downstream of u_0 such that u_t is the last internal node. W.l.o.g we assume that $u_{i-1} \in \mathcal{T}_{u_i(l)}$ for $i \geq 1$. We now show that $\cup_{u\in U} W^{P}_{new}(u) \subseteq \cup_{u\in U} W^{P'}_{new}(u)$. Towards this end we have the following observations for $u \in U$.

$$\mathbb{Z}^{P'}(u) = \mathbb{Z}^{P}(u) \cup \{Z_k\} \text{ (from the construction of } P')$$
$$\Delta^{P'}(u,u) = \cup_{v\in\mathcal{D}_u} \Delta^{P'}(u,v).$$

Now, for $v \notin \mathcal{D}_{u_0(r)}$ we have $\mathbb{D}^{P'}(v) = \mathbb{D}^{P}(v)$ so that

$$\begin{aligned}\Delta^{P'}(u,v) &= Rec(\mathbb{Z}^{P'}(u), \mathbb{D}^{P'}(v)) \\ &= Rec(\mathbb{Z}^{P'}(u), \mathbb{D}^{P}(v)) \\ &\supseteq \Delta^{P}(u,v) (\text{ since } \mathbb{Z}^{P'}(u) \supseteq \mathbb{Z}^{P}(u)).\end{aligned}$$

Conversely, for $v \in \mathcal{D}_{u_0(r)}$ we have

$$Rec\left(\{Z_j, Z_k\}, \mathbb{D}^{P'}(v)\right) = Rec\left(\{Z_j, Z_k\}, \mathbb{D}^{P}(v)\right),$$

and

$$Rec\left(\{Z_i\}, \mathbb{D}^{P'}(v)\right) = Rec\left(\{Z_i\}, \mathbb{D}^{P}(v)\right) \text{ (for } Z_i \notin \{Z_j, Z_k\}).$$

Now, note that $\{Z_k, Z_j\} \subseteq \mathbb{Z}^{P'}(u)$ so that

$$\begin{aligned}\Delta^{P'}(u,v) &= Rec\left(\mathbb{Z}^{P'}(u), \mathbb{D}^{P'}(v)\right) \\ &= Rec\left(\mathbb{Z}^{P'}(u), \mathbb{D}^{P}(v)\right), \\ &\supseteq Rec\left(\mathbb{Z}^{P}(u), \mathbb{D}^{P}(v)\right) = \Delta^{P}(u,v),\end{aligned}$$

since $\mathbb{Z}^{P'}(u) \supseteq \mathbb{Z}^{P}(u)$. We can therefore conclude that the following statement holds for $\Delta^{P}(u,u)$ and $\Delta^{P'}(u,u)$.

$$\Delta^{P}(u,u) = \cup_{v \in \mathcal{D}_u} \Delta^{P}(u,v) \subseteq \cup_{v \in \mathcal{D}_u} \Delta^{P'}(u,v) = \Delta^{P'}(u,u).$$

Now we consider a $W^* \in W^{P}_{new}(u_i)$ so that $W^* \in \Delta^{P}(u_i,u_i)$ which by above condition means that $W^* \in \Delta^{P'}(u_i,u_i)$. Thus either $W^* \in W^{P'}_{new}(u_i)$ or $W^* \in \mathbb{W}^{P'}(u_i)$. In the latter case there exists a node $u_{i'}$ where $0 \leq i' < i$ such that $W^* \in W^{P'}_{new}(u_{i'})$ since $W^* \notin \mathbb{W}(u_0)$ and we have shown that $\mathbb{W}^{P'}(u_0) = \mathbb{W}^{P}(u_0)$. Thus, we observe that

$$\begin{aligned} L' &= |\cup_{u \in U} W^{P'}_{new}(u)| + \sum_{u \in \mathcal{T}', u \notin U} |W^{P'}_{new}(u)|, \\ &\geq |\cup_{u \in U} W^{P}_{new}(u)| + \sum_{u \in \mathcal{T}, u \notin U} |W^{P}_{new}(u)|, \\ &= L, \end{aligned}$$

where the second inequality holds since $\sum_{u \in \mathcal{T}', u \notin U} |W^{P'}_{new}(u)| = \sum_{u \in \mathcal{T}, u \notin U} |W^{P}_{new}(u)|$ and $|\cup_{u \in U} W^{P'}_{new}(u)| \geq |\cup_{u \in U} W^{P}_{new}(u)|$.

As discussed before, the modification procedure is such that at the end of the operation $u^* \notin \mathcal{U}$. Moreover nodes $u \succ u^*$ are not in $\mathcal{U}$ either. For each node $u \in \mathcal{U}$ let $d(u)$ denote the number of edges in the path connecting u to the root node. Our modification procedure is such that $d^* = \max_{u \in \mathcal{U}} d(u)$ is guaranteed to decrease over the course of the iterations. Indeed, if $|\mathcal{U}^*| = 1$, then at the end of the iteration d^* will definitely decrease. If $|\mathcal{U}^*| > 1$, then d^* will definitely decrease after the modification procedure is applied to all the nodes in $\mathcal{U}^*$. Thus, the sequence of iterations is guaranteed to terminate. This observation concludes the proof.

A.0.3 Proof of Lemma 1

Proof:

Given the conditions of the theorem, from Corollary 1 we can conclude that there exists an index $i^* \in \{1, \ldots, \alpha\}$ such that $\sum_{v' \in \mathcal{C}} \psi(v_{i^*}, v') < \min(\beta, K)$. We set i^* to be the smallest such index. Let

$\Pi^1(v_{i^*}) = \{v' \in \mathcal{C} : \psi(v_{i^*}, v') = 1\}$ and $\Pi^0(v_{i^*}) = \{v' \in \mathcal{C} : \psi(v_{i^*}, v') = 0, \mathbb{Z}(v') \nsubseteq \cup_{v \in \Pi^1(v_{i^*})} \mathbb{Z}(v)\}$. Note that $\Pi^0(v_{i^*})$ is non-empty since $|\cup_{v' \in \mathcal{C}} \mathbb{Z}(v')| = \min(\beta, K)$ and $\sum_{v' \in \mathcal{C}} \psi(v_{i^*}, v') < \min(\beta, K)$.

Next, we determine the set of nodes where v_{i^*} and the nodes in $\Pi^0(v_{i^*})$ meet, i.e., we define $\Lambda^0(v_{i^*}) = \{u \in \mathcal{T} : \exists v' \in \Pi^0(v_{i^*})$ such that v_{i^*} and v' meet at $u.\}$. Note that there is a topological ordering on the nodes in $\Lambda^0(v_{i^*})$. Pick the node $u^* \in \Lambda^0(v_{i^*})$ such that no element of $\Lambda^0(v_{i^*})$ is topologically higher than u^* (u^* is in the path from v_{i^*} to the root node). Let the corresponding node in $\Pi^0(v_{i^*})$ be denoted by v_{j^*} where $j^* \in \{\alpha + 1, \ldots, \alpha + \beta\}$. Note that v_{j^*} might not be unique.

Suppose that $\mathbb{Z}(v_{j^*}) = \{Z_k\}$ and that $\mathbb{D}(v_{i^*}) = X_{d_1, \ldots, d_K}$. We modify the instance P as follows. Set $d_k = N + 1$ (i.e., the index of the $N + 1$ file). Thus, the only change is in $\mathbb{D}(v_{i^*})$. Let us denote the new instance by $P' = P(\mathcal{T}', \alpha, \beta, L', N + 1, K)$.

We now analyze the value of L'. W.l.o.g. we assume that $v_{i^*} \in \mathcal{T}'_{u^*(l)}$ and $v_{j^*} \in \mathcal{T}'_{u^*(r)}$. Note that $W^{P'}_{new}(u) = W^{P}_{new}(u)$ for $u \in \mathcal{T}'_{u^*(r)}$ as the subtree $\mathcal{T}'_{u^*(r)}$ is identical to $\mathcal{T}_{u^*(r)}$. We also have

$$W^{P'}_{new}(u) = W^{P}_{new}(u) \text{ for } u \in \mathcal{T}'_{u^*(l)}.$$

To see this suppose that this is not true. This implies that the file W_{N+1} is recovered at some node in $\mathcal{T}'_{u^*(l)}$, i.e., there exists $v' \in \mathcal{C}$ such that $v' \in \mathcal{T}'_{u^*(l)}$, $\mathbb{Z}(v') = \{Z_k\}$, and that v' and v_{i^*} meet at some $u \succ u^*$. From $v_{j^*} \in \Pi^0(v_{i^*})$ we can conclude that $\{Z_k\} \nsubseteq \cup_{v \in \Pi^1(v_{i^*})}$ and $v' \in \Pi^0(v_{i^*})$ (as $\mathbb{Z}(v') = \{Z_k\}$). However this is a contradiction, since this implies the existence of node u that is topologically higher than u^* in the set $\Lambda^0(v_{i^*})$. It follows from eq. (2.6) that $\mathbb{W}^{P'}(u^*) = \mathbb{W}^{P}(u^*)$.

Next, we claim that $W^{P'}_{new}(u^*) = W^{P}_{new}(u^*) \cup \{W_{N+1}\}$. To see this consider the following series of arguments. Let the singleton subset $\Delta^P(v_{i^*}, v_{j^*}) = \{W^*\}$. Note that $\psi^P(v_{i^*}, v_{j^*}) = 0$. This implies that there exist $v \in \mathcal{D}_{u^*}$ and $v' \in \mathcal{C}_{u^*}$ such that v and v' meet above u^* and recover the file W^* where $(v, v') \neq (v_{i^*}, v_{j^*})$. Thus, as $\mathbb{Z}^{P'}(u^*) = \mathbb{Z}^{P}(u^*)$, we can conclude that

$$\begin{aligned}
\Delta^{P'}(u^*, u^*) &= Rec(\mathbb{Z}^{P'}(u^*), \mathbb{D}^{P'}(u^*)) \\
&= Rec(\mathbb{Z}^{P}(u^*), \mathbb{D}^{P'}(u^*)) \\
&= \Delta^{P}(u^*, u^*) \cup \{W_{N+1}\}.
\end{aligned}$$

Furthermore, in the following argument we show that $W^{P'}_{new}(u^*) = W^{P}_{new}(u^*) \cup \{W_{N+1}\}$ holds for node u^*. We have,

$$\begin{aligned} W^{P'}_{new}(u^*) &= \Delta^{P'}(u^*, u^*) \setminus \mathbb{W}^{P'}(u^*) \\ &= \Delta^{P}(u^*, u^*) \cup \{W_{N+1}\} \setminus \mathbb{W}^{P}(u^*) \\ &= W^{P}_{new}(u^*) \cup \{W_{N+1}\}, \text{ (since } W_{N+1} \notin \mathbb{W}^{P}(u^*)). \end{aligned}$$

For u such that $u^* \succ u$ we inductively argue that $W^{P'}_{new}(u) = W^{P}_{new}(u)$. To see this suppose that $u^* = u_r$. It is evident that $\Delta^{P'}_{rl}(u) = \Delta^{P}_{rl}(u)$. Next, $\Delta^{P'}_{lr}(u) = \Delta^{P}_{lr}(u)$ since $Z_k \notin \mathbb{Z}(u_l) \setminus \mathbb{Z}(u_r)$. Thus,

$$\begin{aligned} W^{P'}_{new}(u) &= \Delta^{P'}_{rl}(u) \cup \Delta^{P'}_{lr}(u) \setminus \mathbb{W}^{P'}(u) \\ &= \Delta^{P}_{rl}(u) \cup \Delta^{P}_{lr}(u) \setminus \mathbb{W}^{P'}(u) \\ &= \Delta^{P}_{rl}(u) \cup \Delta^{P}_{lr}(u) \setminus \mathbb{W}^{P}(u) \cup \{W_{N+1}\} \\ &= \Delta^{P}_{rl}(u) \cup \Delta^{P}_{lr}(u) \setminus \mathbb{W}^{P}(u) \text{ (since } W_{N+1} \notin \Delta^{P}_{rl}(u) \cup \Delta^{P}_{lr}(u)) \\ &= W^{P}_{new}(u). \end{aligned}$$

Next, we note that $\mathbb{W}(u) = \mathbb{W}(u_r) \cup W_{new}(u_r) \cup \mathbb{W}(u_l) \cup W_{new}(u_l)$. It is evident that $\mathbb{W}^{P'}(u_l) = \mathbb{W}^{P}(u_l)$ and $W^{P'}_{new}(u_l) = W^{P}_{new}(u_l)$. Next, $\mathbb{W}^{P'}(u_r) = \mathbb{W}^{P'}(u^*) = \mathbb{W}^{P}(u^*)$ (from above) and $W^{P'}_{new}(u^*) = W^{P}_{new}(u^*) \cup \{W_{N+1}\}$, so that $\mathbb{W}^{P'}(u) = \mathbb{W}^{P}(u) \cup \{W_{N+1}\}$.

As the induction hypothesis we assume that for any node u downstream of u^*, we have $W^{P'}_{new}(u) = W^{P}_{new}(u)$ and $\mathbb{W}^{P'}(u) = \mathbb{W}^{P}(u) \cup \{W_{N+1}\}$. Consider a node u' such that $u'_r = u$. As before we have $\mathbb{W}^{P'}(u'_l) = \mathbb{W}^{P}(u'_l)$, $W^{P'}_{new}(u'_l) = W^{P}_{new}(u'_l)$. Moreover, we have $\mathbb{W}^{P'}(u'_r) = \mathbb{W}^{P}(u'_r) \cup \{W_{N+1}\}$ and $W^{P'}_{new}(u'_r) = W^{P}_{new}(u'_r)$, by the induction hypothesis, so that $\mathbb{W}^{P'}(u') = \mathbb{W}^{P}(u') \cup \{W_{N+1}\}$.

Next, we argue similarly as above that $\Delta^{P'}_{rl}(u') = \Delta^{P}_{rl}(u')$ and $\Delta^{P'}_{lr}(u') = \Delta^{P}_{lr}(u')$ and the sequence of equations above can be used to conclude to that $W^{P'}_{new}(u') = W^{P}_{new}(u')$. We conclude that $L' = L + 1$.

A.0.4 Proof of Claim 5

Proof:

W.l.o.g we assume that $|\Gamma_l| \geq |\Gamma_r|$ for all $u \in \mathcal{T}$. We identify the set $\mathcal{U}$ as the set of nodes in $\mathcal{T}$ such that $\Gamma_r \nsubseteq \Gamma_l$. Let $\mathcal{U}^* \subset \mathcal{U}$ denote the set of nodes in $\mathcal{U}$ that are highest in the topological ordering.

Consider a node $u^* \in \mathcal{U}^*$. Note that since $|\Gamma_l| \geq |\Gamma_r|$, there exists an injective mapping $\phi : \Gamma_r \setminus \Gamma_l \to \Gamma_l \setminus \Gamma_r$. Let $\mathbb{Z}(u_r^*) = \{Z_{i_1}, \ldots, Z_{i_m}\}$. We construct the instance P' as follows. For each $v \in \mathcal{D}_{u_r^*}$ suppose $\mathbb{D}(v) = \{X_{d_1,\ldots,d_K}\}$. For $j = 1, \ldots, m$, if $d_{i_j} \in \Gamma_r \setminus \Gamma_l$, we replace it by $\phi(d_{i_j})$; otherwise, we leave it unchanged. In other words, we modify the delivery phase signals so that the files that are recovered in $\mathcal{T}_{u^*(r)}$ are a subset of those recovered in $\mathcal{T}_{u^*(l)}$.

As our change amounts to a simple relabeling of the sources, for $u \in \mathcal{T}_{u^*(r)}$ we have $|W_{new}^{P'}(u)| = |W_{new}^{P}(u)|$. For any $u \succ u^*$ we have $\Gamma_r^P(u) \subseteq \Gamma_l^P(u)$. Similarly, we can show that $\Gamma_r^{P'}(u) \subseteq \Gamma_l^{P'}(u)$. We note that $\Gamma^{P'}$ and Γ^P only differ in files such as W_d where d is in the domain of $\phi(\cdot)$, i.e., if $W_d \in \Gamma^P$ then $W_{\phi(d)} \in \Gamma^{P'}$. If there exist a file $W_d \in \Gamma_r^P(u)$ with d in domain of $\phi(\cdot)$ then $W_{\phi(d)} \in \Gamma_r^{P'}(u)$ and from $\Gamma_r^P(u) \subseteq \Gamma_l^P(u)$ we have $W_{\phi(d)} \in \Gamma_l^{P'}(u)$. Thus, we have $\Gamma_r^{P'}(u) \subseteq \Gamma_l^{P'}(u)$. This indicates that after applying this change, the property $\Gamma_r \subseteq \Gamma_l$ still holds in P' for all nodes u that are upstream of u^*. Furthermore, the relabeling of the sources only affects $u \in \mathcal{T}'$ such that $u^* \succ u$. Note that $\mathbb{W}^{P'}(u^*) \subset \mathbb{W}^P(u^*)$ (the inclusion is strict since at least one source in $\Gamma_r \setminus \Gamma_l$ is mapped to $\Gamma_l \setminus \Gamma_r$) since we have $\Gamma_r^{P'} \subseteq \Gamma_l^{P'}$ and $\Gamma_l^{P'} = \Gamma_l^P$.

Now, we note that

$$\Delta_{rl}^{P'}(u^*) = \Delta_{rl}^{P}(u^*), \text{ and}$$
$$\Delta_{lr}^{P'}(u^*) = \Delta_{lr}^{P}(u^*),$$

where the first equality holds since $\mathbb{Z}^P(u_r^*) = \mathbb{Z}^{P'}(u_r^*)$, $\mathbb{Z}^P(u_l^*) = \mathbb{Z}^{P'}(u_l^*)$ and $\mathbb{D}^P(u_l^*) = \mathbb{D}^{P'}(u_l^*)$. The second equality holds since our modification to the delivery phase signals in $\mathcal{T}_{u^*(r)}$ does not affect files that are recovered from $\mathbb{Z}^P(u_l^*) \setminus \mathbb{Z}^P(u_r^*)$. It follows therefore that $|W_{new}^{P'}(u^*)| \geq |W_{new}^{P}(u^*)|$.

We make an inductive argument for nodes u that are downstream of u^*; w.l.o.g. we assume that $u^* \in \mathcal{T}_{u(r)}$. Specifically, our induction hypothesis is that for a node u that is downstream of u^*, we have $\mathbb{W}^{P'}(u) \subseteq \mathbb{W}^{P}(u)$, $\Delta^{P'}_{rl}(u) = \Delta^{P}_{rl}(u)$ and $\Delta^{P'}_{lr}(u) = \Delta^{P}_{lr}(u)$.

Now consider a node u' downstream of u such that $u'_r = u$. We have, $\mathbb{W}(u') = \mathbb{W}(u'_l) \cup W_{new}(u'_l) \cup \mathbb{W}(u) \cup W_{new}(u)$. Note that we can express $\mathbb{W}(u) \cup W_{new}(u) = \mathbb{W}(u) \cup \Delta_{rl}(u) \cup \Delta_{lr}(u)$. It is evident that $\mathbb{W}^{P'}(u'_l) = \mathbb{W}^{P}(u'_l)$ and $W^{P'}_{new}(u'_l) = W^{P}_{new}(u'_l)$. Moreover, by the induction hypothesis, $\mathbb{W}^{P'}(u) \subseteq \mathbb{W}^{P}(u)$ and $\Delta^{P'}_{rl}(u) \cup \Delta^{P'}_{lr}(u) = \Delta^{P}_{rl}(u) \cup \Delta^{P}_{lr}(u)$. Thus, the induction step is proved.

We have shown that after applying the changes for u^*, the condition $\Gamma_r \nsubseteq \Gamma_l$ will not hold for $u \succeq u^*$. For each node $u \in \mathcal{U}$ let $d(u)$ denote the number of edges in path connecting u to the root node. Our modification procedure is such that $d^* = \max_{u \in \mathcal{U}} d(u)$ is guaranteed to decrease over the course of the iterations. Indeed, if $|\mathcal{U}^*| = 1$, then at the end of the iteration d^* will definitely decrease. If $|\mathcal{U}^*| > 1$, then d^* will definitely decrease after the modification procedure is applied to all the nodes in $\mathcal{U}^*$. Thus, the sequence of iterations is guaranteed to terminate. This observation concludes the proof.

As we have shown, the modification procedure is such that at the end of the operation u^* is removed from $\mathcal{U}$. Therefore, each node in $\mathcal{T}$ will be involved in the modification procedure at most once. In Appendix A.0.5, we show that there are $2(\alpha + \beta)$ nodes in $\mathcal{T}$. Thus, the modification procedure requires at most $2(\alpha + \beta)$ iterations to terminate. At each iteration we only need to apply the mapping $\phi(\cdot)$ to the indices of the delivery nodes connected to u^*. The complexity of this step is at most $\alpha\beta$. Therefore, the complexity of the modification is at most $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$.

Claim 10 *When* $\hat{\beta}_l = \min(\beta_l, K)$ *and* $\hat{\beta}_r = \min(\beta_r, K)$ *we have* $\min(\hat{\beta}_l, K - \hat{\beta}_r) = [\min(\beta_l, K - \beta_r)]^+$ *and* $\min(\hat{\beta}_r, K - \hat{\beta}_l) = [\min(\beta_r, K - \beta_l)]^+$.

Proof: First, we consider the case where $\beta_l+\beta_r \leq K$ so $\beta_l \leq K-\beta_r$ and $[\min(\beta_l, K-\beta_r)]^+ = \beta_l$. By assumption, $\beta_l + \beta_r \leq K$ implies $\hat{\beta}_l + \hat{\beta}_r \leq K$ thus $\min(\hat{\beta}_l, K - \hat{\beta}_r) = \hat{\beta}_l = \beta_l$. We now consider the $\beta_l + \beta_r \geq K$ case which in turns leads to $\hat{\beta}_l + \hat{\beta}_r \geq K$. Therefore,

$$\begin{aligned}\min(\hat{\beta}_l, K - \hat{\beta}_r) &= K - \hat{\beta}_r = K - \min(K, \beta_r) \\ &= \max(0, K - \beta_r) = [K - \beta_r]^+ = [\min(\beta_l, K - \beta_r)]^+.\end{aligned}$$

The same argument will show that $\min(\hat{\beta}_r, K - \hat{\beta}_l) = [\min(\beta_r, K - \beta_l)]^+$.

Claim 11 *Consider the integers $\alpha, \alpha_l, \alpha_r, \beta, \beta_l, \beta_r, K$ so that $\alpha = \alpha_l + \alpha_r$ and $\beta = \beta_l + \beta_r$. Then*

$$\begin{aligned}\alpha \min(\beta, K) &= \alpha_l \min(\beta_l, K) + \alpha_r \min(\beta_r, K) \\ &+ \alpha_l[\min(\beta_r, K - \beta_l)]^+ + \alpha_r[\min(\beta_l, K - \beta_r)]^+.\end{aligned}$$

Proof: First, we consider the case where $\beta \leq K$ thus $\beta_l \leq K - \beta_r$ and $\beta_r \leq K - \beta_l$. Then, the above relation reduces to $\alpha\beta = \alpha_l\beta_l + \alpha_r\beta_r + \alpha_l\beta_r + \alpha_r\beta_l$ which is true. For the case $\beta \geq K$, the relation reduces to $\alpha K = \alpha_l \left(\min(\beta_l, K) + [K - \beta_l]^+\right) + \alpha_r \left(\min(\beta_r, K) + [K - \beta_r]^+\right)$. However $\min(\beta_l, K) = K - [K - \beta_l]^+$ and $\min(\beta_r, K) = K - [K - \beta_r]^+$ and the result follows.

A.0.5 Complexity of the Algorithms 1, 2, 3, and 4

In this part we discuss the time-complexity of the algorithms used in this paper. Before proceeding, we note that the directed in-tree corresponding to the problem instance $P(\mathcal{T}, \alpha, \beta, L, N, K)$ contains $\alpha + \beta$ leaves and a single root. The degree (total number of incoming and outgoing edges) of the leaves and the root is 1. Based on Claim 1 the intermediate nodes have a total degree of 3. Thus,

$$\begin{aligned}2|A| &= \alpha + \beta + 1 + 3(|V| - \alpha - \beta - 1) \\ &= 3|V| - 2\alpha - 2\beta - 2.\end{aligned}$$

On the other hand, since the undirected version of $\mathcal{T}$ is a tree we have $|A| = |V| - 1$. Solving these two equations yields

$$|V| = 2(\alpha + \beta),$$
$$|A| = 2(\alpha + \beta) - 1.$$

A.0.5.1 Complexity of the Algorithm 1

The complexity of computing $\Delta(v_i, v_i)$ in second line of the algorithm is less than $\alpha\beta$. As there are $\alpha + \beta$ leaves thus the complexity of lines 1-5 of the algorithm is $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$. The while loop in the algorithm goes over all nodes except the leaves exactly once. Thus, the while loop is executed $\alpha + \beta$ times. At each phase of the while loop, computing $\Delta(u, u)$ has the largest running time among the other operation and its complexity is less than $\alpha\beta$. Therefore, the complexity of the while loop is also $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$. Thus, this algorithm has a time-complexity of $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$.

A.0.5.2 Complexity of the Algorithm 2

As there are $2(\alpha + \beta)$ nodes in $\mathcal{T}$ and $|W_{new}(u)| \leq \alpha\beta$, the complexity of the initialization part is $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$. In the remaining steps of the algorithm, the main complexity of the inner for loop is in finding the meeting point of v_i and v'. It is not hard to see that the complexity of finding this meeting point is at most $(\alpha + \beta)$, i.e., number of edges in $\mathcal{T}$. Therefore, the complexity of this part is $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$. Putting these together, complexity of the algorithm is $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$.

A.0.5.3 Complexity of the Algorithm 3

The initialization part of the Algorithm 3 takes $\mathcal{O}(1)$ running time. The while loop at "Tree Construction and Cache Nodes Labeling" goes over all nodes in $\mathcal{T}$ exactly once. As the operation inside the loop takes $\mathcal{O}(1)$ time, the complexity of this part of the algorithm is $\mathcal{O}(\alpha + \beta)$. The third part of the algorithm is "Delivery Nodes Labeling". It is not difficult to see that the first for loop in this part requires at most β running times. Also, the second for loop takes $\mathcal{O}(\alpha\beta)$ running time. Thus, complexity of this part is at most $\mathcal{O}(\alpha\beta)$. Finally, as we have shown in proof of the Claim 5,

the complexity of "Modifying Delivery Phase Signals" is $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$. Putting all these together, complexity of Algorithm 3 is $\mathcal{O}(\alpha^2\beta + \alpha\beta^2)$.

A.0.5.4 Complexity of the Algorithm 4

The algorithm needs $\mathcal{O}(\alpha\beta)$ memory units to save $N_{sat}(a, b, K)$ for $0 \leq a \leq \alpha$ and $0 \leq b \leq \beta$. Once $N_{sat}(\tilde{a}, \tilde{b}, K)$ is known for $(\tilde{a}, \tilde{b}) \in \mathcal{I}(a, b)$ then we are able to compute $N_{sat}(a+1, b+1, K)$ by using the recursive relationship.

The time complexity of populating the N_{sat} values can be determined as follows. At the initialization step we fill the first two rows and columns of the matrix N_{sat} corresponding to $a = 0, 1$ and $b = 0, 1$ respectively. Following this initialization step, the remaining rows and columns are populated. It is clear the that the initialization takes $\mathcal{O}(\alpha + \beta)$ time. In the main loop we compute each entry of matrix N_{sat} once. This computation takes at most $\mathcal{O}(\alpha\beta)$ operation as we look for minimum over set $\mathcal{I}(a, b)$ whose size is at most $\mathcal{O}(\alpha\beta)$. As we compute all entries of the matrix N_{sat} and each entry takes $\mathcal{O}(\alpha\beta)$ running times thus time complexity of the algorithm is $\mathcal{O}(\alpha^2\beta^2)$. The required memory is $\mathcal{O}(\alpha\beta)$ as determined above.

APPENDIX B. SUPPLEMENT FOR ASYNCHRONOUS CODED CACHING

B.0.1 Equivalence of LPs

We show that both linear programming problems in (3.1) and (3.2) result the same solution. Actually, we show that the feasible sets of both problems are equivalent and since both problem have the same objective function this implies that they return the same optimal value.

We first show that each feasible point in problem (3.1) is also a feasible point in (3.2). Let suppose that $\{x_U(\ell), y_{\{i,f\}}(U)\}$ is a feasible point of (3.1). We set $x_U^{(i)}(\ell) = x_U(\ell) - \alpha_U^{(i)}(\ell)$ with $\alpha_U^{(i)}(\ell)$'s are chosen so that $0 \leq \alpha_U^{(i)}(\ell) \leq x_U^{(i)}(\ell)$ and,

$$\sum_{\ell \in \mathcal{I}_U} \alpha_U^{(i)}(\ell) = \sum_{\ell \in \mathcal{I}_U} x_U(\ell) \; - \sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U). \tag{B.1}$$

Then, we claim that $\{x_U(\ell), y_{\{i,f\}}(U)\}$ together with the new defined $x_U^{(i)}(\ell)$'s form a feasible point in (3.2). To show this, we only need to show that

$$\sum_{f \in \mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) = \sum_{\ell \in \mathcal{I}_U} x_U^{(i)}(\ell),$$

and that $0 \leq x_U^{(i)}(\ell) \leq x_U(\ell)$ and $x_U^{(i)}(\ell) \leq |\Pi_\ell|$. The first condition holds by the way we choose $\alpha_U^{(i)}(\ell)$ and (B.1). Moreover, by such a setting of $x_U^{(i)}(\ell)$ and that $0 \leq x_U(\ell) \leq |\Pi_\ell|$ and $0 \leq \alpha_U^{(i)}(\ell) \leq x_U^{(i)}(\ell)$ the other conditions hold as well. This implies that each feasible point in (3.1) is a feasible point in (3.1) as well.

Now, we show that each feasible point in (3.2) is a equivalent to a feasible point in (3.1). To see this, we just need to show that

$$\sum_{f\in\mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) \leq \sum_{\ell\in\mathcal{I}_U} x_U(\ell),$$

holds. This can be verified by the fact that $x_U^{(i)}(\ell) \leq x_U(\ell)$ and

$$\sum_{f\in\mathcal{F}_{\{i,U\}}} y_{\{i,f\}}(U) = \sum_{\ell\in\mathcal{I}_U} x_U^{(i)}(\ell).$$

Therefore, we showed that problems (3.1) and (3.2) have the same optimal value.

B.0.2 Quadratic Projection and Primal Recovery in Dual Decomposition

For the projection of $\tilde{\gamma}_U^{(i)}(\ell,n)$ and $\tilde{\zeta}_\ell(n)$ to the constraint space we simply set $\zeta_\ell(n) = \max\left(\tilde{\zeta}_\ell(n),\ 0\right)$ and $\{\gamma_U^{(i)}(\ell,n),\ \forall\ i\in U\}$ is obtained via the following quadratic optimization.

$$\min_{\{v_U^{(i)}:\ \forall\ i\in U\}} \sum_{i\in U}\left(v_U^{(i)} - \tilde{\gamma}_U^{(i)}(\ell,n)\right)^2 \quad \text{s.t.} \quad \sum_{i\in U} v_U^{(i)} = 1. \tag{B.2}$$

In (Lun et al., 2006, Appendix I) an algorithm has been proposed to solve (B.2). This solution can be explained as follows. For fixed $\ell\in[\beta]$ and for each $U\in\mathcal{U}_\ell$, we sort $\tilde{\gamma}_U^{(i)}(\ell,n)$ so that $\tilde{\gamma}_U^{(i_1)}(\ell,n) \geq \ldots \geq \tilde{\gamma}_U^{(i_{|U|})}(\ell,n)$. We take $\hat{k}$ to be the minimum k such that

$$\frac{1}{k}\left(1 - \sum_{j=1}^{k}\tilde{\gamma}_U^{(i_j)}(\ell,n)\right) \leq -\tilde{\gamma}_U^{(i_{k+1})}(\ell,n)$$

or let $\hat{k} = |U|$ if such a k doesn't exist. Then $\gamma_U^{(i_j)}(\ell,n) = \tilde{\gamma}_U^{(i_j)}(\ell,n) + \frac{1-\sum_{l=1}^{\hat{k}}\tilde{\gamma}_U^{(i_l)}(\ell,n)}{\hat{k}}$ if $j\in[\hat{k}]$ and zero otherwise.

The initial setting for the dual variables is chosen as $\gamma_U^{(i)}(\ell,0) = 1/|U|$, for $i\in U,\ U\in\mathcal{U}_\ell,\ \ell\in[\beta]$, and $\zeta_\ell = 0$ for $\ell\in[\beta]$.

Primal Recovery: After solving the dual problem, the primal variables, i.e., $x_U(\ell,n)$'s, are recovered by the method of Sherali and Choi (1996) whereby

$$x_U(\ell,n) = \sum_{l=1}^{n}\mu_l(n)\left(\max_{i\in U}\ x_U^{(i)}(\ell,l)\right) \tag{B.3}$$

where $\mu_l(n)$'s are sequence of convex combination weights for each non-negative integer n, i.e. $\sum_{l=1}^{n} \mu_l(n) = 1$ and $\mu_l(n) \geq 0$ for all $l = 1, \ldots, n$. In Lun et al. (2006), it has been shown that if the step size θ_n and convex combination weights $\mu_l(n)$ are chosen so that

- $\eta_{l,n} \geq \eta_{l-1,n}$ for all $l = 2, \ldots, n$ and $n = 0, 1, \ldots$,
- $\Delta_{\eta_n}^{\max} \to 0$ as $n \leftarrow \infty$, and
- $\eta_{1,n} \to 0$ as $n \leftarrow \infty$ and $\eta_{n,n} \leq \delta$ for all $n = 0, 1, \ldots$ for some $\delta > 0$,

then $\{x_U(\ell, n)\}_{\ell \in [\beta],\ U \in \mathcal{U}_l}$ is an optimal primal solution. Here $\eta_{l,n} = \frac{\mu_l(n)}{\theta_n}$ and $\Delta_{\eta_n}^{\max} = \max_{l=2,\ldots,n}\{\eta_{l,n} - \eta_{l-1,n}\}$. Some sequences for θ_n and $\mu_l(n)$ that satisfy the above conditions has been proposed by Lun et al. (2006) . Among them we choose $\mu_l(n) = \frac{1}{n}$ and $\theta_n = n^{-\alpha}$ where $0 < \alpha < 1$. Then, the primal solution will be updated as,

$$x_U(\ell, n+1) = \frac{n}{n+1} x_U(\ell, n) + \frac{\max_{i \in U}\ x_U^{(i)}(\ell, n)}{n+1}. \tag{B.4}$$

B.0.3 Linear Programming with Front Loading

We note that in the above example if we transmit $B_3 \oplus C_2$ instead of transmitting A_4 then the online solution would be feasible. This property of the above counterexample guides us to another algorithm so that at each arrival time and among the feasible solutions consistent with the previously transmitted equations we pick the solution in which it has minimum transmission. Then we front load the solution. We show that even using this algorithm there is no guarantee on feasibility. To this aim, we have the following counterexample.

Example 16 *We consider an asynchronous coded caching system with $K = 6$ users, $N = 6$ files, and $M = 1$. The arrival times and deadlines of each user along with the user groups of the offline solution are brought in Fig. B.1 (a).*

As we can see, the offline solution is so that the server has to transmit an equation at each time slot. The bottleneck of this example is transmission in the third time slot. In Fig. B.1 (b) the first two possible steps of the online solution with solving lp with front loading is shown.

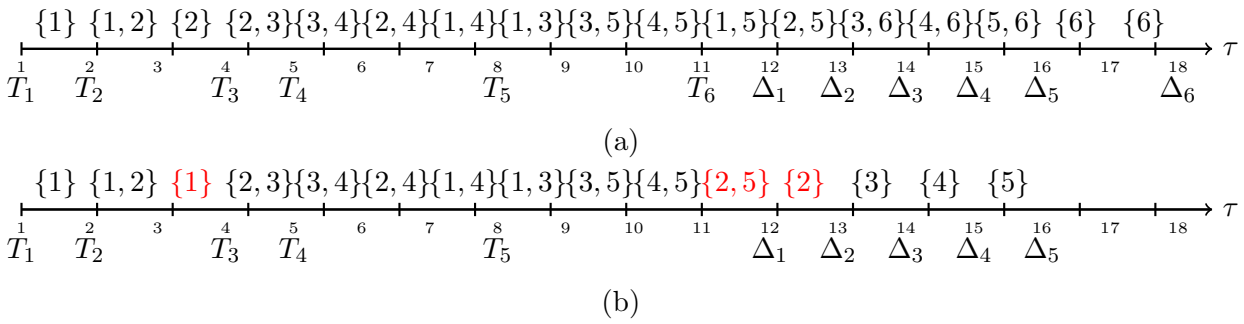


Figure B.1: A counterexample of LP solution with front loading.

In the third time slot there is no privilege in choosing $\{1\}$ over $\{2\}$ after solving LP with front loading. Indeed, both choices are optimal solution of the LP problem with front loading. If the server choose to transmit user group $\{1\}$ instead of $\{2\}$ then the online solution will be infeasible. Otherwise, the online solution will follow the exact procedure of the offline solution in Fig. B.1. In Fig. B.1 (b) we show how the online algorithm get stuck when the server chooses to transmit $\{1\}$ at the third time slot.

B.0.4 Counter-examples to Intuitive Heuristics

At the top level the online problem bears resemblance to scheduling on jobs on a server. Each file request needs to be processed within a certain time and the transmission of a subfile can be viewed as equivalent to the processing time of a task within a job. Nevertheless, as coded transmission simultaneously serves multiple users, the problem at hand also exhibits important differences.

Algorithms such as earliest-deadline first (EDF) have been well investigated in the scheduling literature Lu et al. (1999); Spuri and Buttazzo (1994); Ramamritham et al. (1990) and are known to be optimal under certain situations. We now show that an EDF-like algorithm can fail for the online scenario. Specifically, we demonstrate that the online algorithm is infeasible even though a feasible offline algorithm exists.

Example 17 *We consider a system with $K = N = 4$, $M = 1$, and $r = 1$ with the cache placement policy in Maddah-Ali and Niesen (2014). The files are denoted A, B, C and D. The arrival times and deadlines are depicted in Fig. B.2. W.l.o.g. we assume that users $1, \ldots, 4$ are interested in*

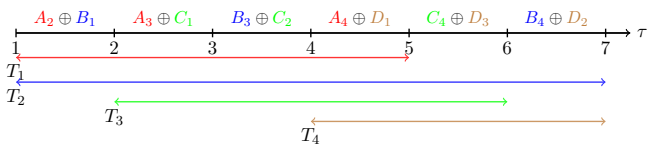


Figure B.2: A counterexample of immediate transmission with priority of closest deadline. Available time slot for each user is determined by a two direction arrow.

files A, B, C and D respectively. Note that the total number of unknowns are $4 \times 3 = 12$. Fig. B.2 shows an offline solution for this system where each of the six time-slots benefits two users (the maximum possible). Therefore, any online algorithm that transmits equations that benefit only a single user in any of the slots will definitely be infeasible.

In an EDF-like algorithm, at each step, the server transmits an equation benefiting the user(s) with the closest deadline. In time slot $[1, 2)$ the only choice is sending $A_2 \oplus B_1$ as only two users are active and this equation benefits both of them. In $[2, 3)$, the server chooses to benefit users 1 and 3 as their deadlines are before user 2. The problem emerges in time slot $[3, 4)$; the EDF policy will require transmission of A_4 as user 1 has the closest deadline. However, there is no equation involving A_4 that benefits two users. Thus, transmission of A_4 will definitely result in an infeasible solution.

The previous counter-example indicates that it may be preferable to transmit equations that benefit the maximum number of users while ensuring feasibility in a best-effort sense at each time-instant. This is in fact the key idea of our proposed heuristic for the online case. Specially, we repeatedly solve a new LP whenever a new user request comes while fixing the previous variables to their earlier values. Coded transmission corresponding to these variables continues (albeit with new features such as coding across missing subfiles for the same user). Furthermore, within the class of feasible solutions, we pick those where the equations that benefit a large number of users are transmitted first. Nevertheless, we emphasize that even this algorithm can fail as compared to the offline solution under certain adversarial choices of request arrival times/deadlines and file requests. We discuss a specific counter-example in the Appendix B.0.3.